

PRML (Pattern Recognition And Machine Learning)

读书会讲课记录合集

前言

读书会成立属于偶然，一次群里无聊到极点，有人说 Pattern Recognition And Machine Learning 这本书不错，加之有好友之前推荐过，便发了封群邮件组织这个读书会，采用轮流讲课的方式，如果任务能分配下去就把读书会当作群员的福利开始进行，分配不下去就算了。后来我的几位好友：网神兄、戴玮博士、张巍博士、planktonli 老师、常象宇博士纷纷出来支持这个读书会。待任务分配完，设置好主持人和机动队员，我认为就不需要再参与了，但进行不久，也充当机动队员讲了第二、六、九、十一章，并承担了所有的整理回顾工作。随着读书会的进行渐渐发现 PRML 这本书可以用惊艳二字来形容，每讲一章之前我们都花费大量时间精力做准备，然后用聊天的方式白话讲课，尽量做到通俗易懂，一章内容太多便分几次讲，讲的不满意便重新讲，一共讲课 23 次，加上整理回顾前后进行了两遍，历时一年半这份讲稿合集才与大家见面。以下是各章的简介：

第一章 Introduction 由西安交通大学常象宇博士主讲，深入浅出的介绍了机器学习的基本概念、学习理论、模型选择、维灾等。

第二章 Probability Distributions 的贝塔-二项式、狄利克雷-多项式共轭、高斯分布、指数族等很基础也很重要。出于各种原因我前后讲了三次，直到比较满意为止。

理解机器学习莫过于从最基础的线性模型开始，第三章 Linear Models for Regression 由西北大学 planktonli 老师主讲，介绍了线性基函数模型、正则化方法、贝叶斯线性回归及其与核函数的联系等内容，为后面几章打下了良好基础。

第四章 Linear Models for Classification 仍由西北大学 planktonli 老师主讲，介绍了贝叶斯的 marginalization 概念、Fisher 线性判别、感知机、分类器概率生成和判别模型的区别与联系、逻辑回归的最大似然参数估计、贝叶斯逻辑回归的 Laplace 近似推断等内容。

第五章 Neural Networks 由网神（新浪微博：@豆角茄子麻酱凉面）主讲，精彩内容有：神经网络做回归和分类的训练目标函数、BP 误差后向传播的链式求导法则、正则化、卷积网络等。

第六章 Kernel Methods，介绍了核函数的定义、构建方法，通过线性回归的 Dual Representations 推导说明由基于特征到基于样本学习的转换；最后是动感十足的高斯过程 Gaussian Processes，包括 GP 的协方差矩阵形式、超参、预测等内容。

第七章 Sparse Kernel Machines 由工业界高手‘网神’（新浪微博：@豆角茄子麻酱凉面）主讲。主要内容：推导了支持向量机（support vector machine）的 Dual Representations；由 KKT 条件说明了解的稀疏性；为提高泛化能力增加松弛变量后的 SVM；最后是加了先验有更稀疏解的 RVM。

第八章 Graphical Models 由‘网神’（新浪微博：@豆角茄子麻酱凉面）主讲，精彩内容有：贝叶斯网络和马尔科夫随机场的概念、联合概率分解、条件独立表示；图的概率推断 inference。

第九章 Mixture Models and EM，主要内容有：Kmeans 算法；混合高斯模型以及 EM（Expectation Maximization）算法在 GMM 中的应用；一般 EM 算法性质的推导和证明。

第十章的主要内容是变分推断（Variational Inference），由中科院自动化所戴玮博士（新浪微博：@戴玮_CASIA）前后分三次讲完。精彩内容有：为什么需要近似推断、变分推断用到的 KL 散度、根据平均场（Mean Field）思想的分解以及迭代求最优解的推导，最后用了三个例子来加深理解。

第十一章的主要内容是 MCMC（Markov Chain Monte Carlo），包括：马尔科夫链平稳分布的定义及其充分条件：细致平稳条件的证明；Metropolis-Hastings 及其接受率满足细致平稳条件的推导，接受率恒为 1 的 Gibbs Sampling；最后是 Slice Sampling、Hamiltonian MCMC。

第十二章连续隐变量，由中科院自动化所戴玮博士（新浪微博：@戴玮_CASIA）分三次讲完。精彩内容有：从最大方差和最小重构误差两个角度解释了 PCA；包含连续隐变量的概率生成模型 PPCA，其最大似然闭式解的推导以及 EM 求解方法；核 PCA 的变换；最后介绍了 Autoencoder、非线性流形思想

第十三章 Sequential Data，由中科院软件所张巍博士（新浪微博：@张巍_ISCAS）主讲，精彩内容有：Hidden Markov Models 的数据生成过程及其参数的 EM 求解方法、HMM 的预测和解码。

最后一章 Combining Models，由‘网神’（新浪微博：@豆角茄子麻酱凉面）主讲，精彩内容有：committees；Boosting、AdaBoost，并从最优化指数损失函数的角度对其步骤作了解释；最后是决策树和条件混合模型。

感谢以上 PRML 所有参讲人员，并对正在进行中的 MLAPP (Machine Learning: A Probabilistic Perspective) 读书会的参讲人员：黄浩军（新浪微博: @Copper_PKU）、余磊博士（新浪微博: @红烧鱼_机器学习）、SIAT（新浪微博: @princeton）、皮搦子狐狸（新浪微博: @unluckyAllen）、Zealot 等人一并感谢。坚持把 PRML 这本书跟下来的同学也辛苦了。最后对 QQ 群、微博、微信等各个平台上所有参与和支持我们读书会的人表示感谢，有几次都想放弃了，是大家对于机器学习的热情一直在推动着读书会前进。

读书会 QQ 群：一号群：177217565（已满）；

二号群：424269692 加群时请在申请里用简短的话描述一模型或算法的关键思想。

微信平台请扫下面二维码：



PRML 这本书结束了，但读书会仍会继续，最新动态请关注我的新浪微博：

@Nietzsche_复杂网络机器学习

(<http://weibo.com/dmalgorithms>)

第一章 Introduction

主讲人 常象宇

Introduction to Machine Learning

Likrain

May 4, 2013

Likrain ()

May 4, 2013 1 / 29

大家好，我是 likrain，本来我和网神说的是我可以作为机动，大家不想讲哪里我可以试试，结果大家不想讲第一章。估计都是大神觉得第一章比较简单，所以就由我来吧。我的背景是统计与数学，稍懂些计算机，大家以后有问题可以讨论。

今天我们来讲一下 PRML 第一章，这一章的内容是基于一些简单的例子对于机器学习中的基本概念给与介绍。这是为后续章节的介绍给一个铺垫。我今天讲的内容包括以下几个部分：

Outline

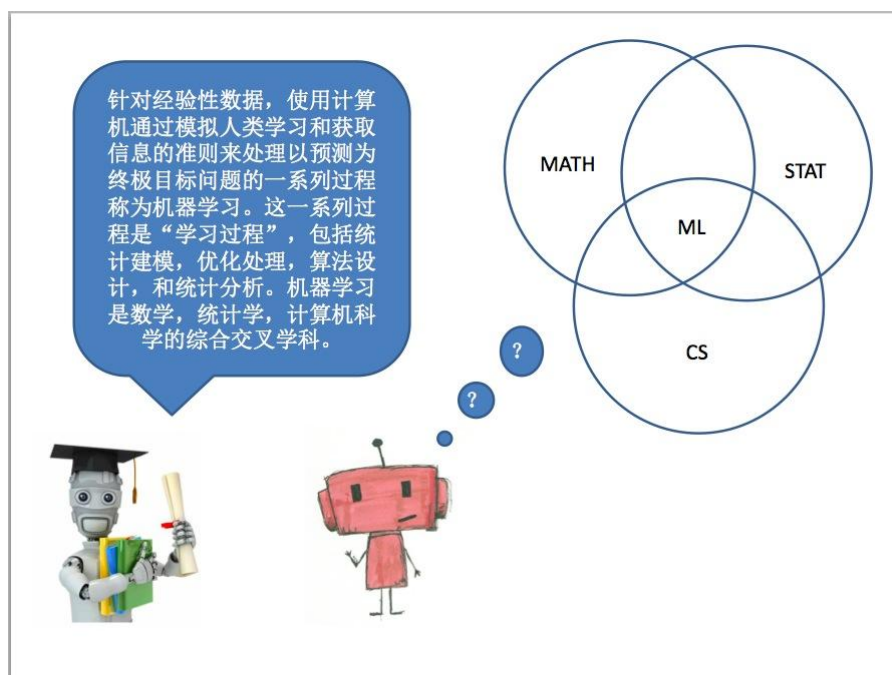
- ① Machine Learning
 - What is the Machine Learning?
 - Paradigms of Machine Learning
 - International Machine Learning Groups
- ② Basic Concepts
 - Probability Theory
- ③ The Challenges of Machine Learning
 - Model Selection
 - The Curse of Dimensionality
- ④ Tow Basic Theories
 - Information Theory
 - Decision Theory

Likrain ()

May 4, 2013 2 / 29

把书上的知识点做了个总结大概。

首先我们来看一下，我个人理解的机器学习的定义：



机器学习的分类有很多种，一般是基于两点：数据类型与学习过程。

是否有标签->监督（分类，回归），半监督，无监督（聚类）；

学习过程不同->主动学习，强化学习，转导学习。

Paradigms of Machine Learning Machine Learning Method:

We want to seek the understanding form data set $\{x_n, y_n\}_{n=1}^N$.

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Transductive Learning
- Active Learning
- Reinforcement Learning
-

=====讨论=====

这里我可以稍微停一下

是否有问题？

planktonli(1027753147) 19:01:30

这里讲的要更清楚些哦，learning 的区别啊

丛(373242125) 19:01:36

$\{x_n, y_n\}_{n=1}^N$.

HX(458728037) 19:01:57

学习过程不同->主动学习, 强化学习, 转导学习是怎么分的呢?

likrain(261146056) 19:02:22

提问结束否?

大家如果没问题了我就开始回答

planktonli(1027753147) 19:02:50

Paradigms of Machine Learning



- Supervised Learning

- Given $D = \{X_i, Y_i\}$, learn $f(\cdot): Y_i = f(X_i)$, s.t. $D^{\text{new}} = \{X_j\} \Rightarrow \{Y_j\}$

- Unsupervised Learning

- Given $D = \{X_i\}$, learn $f(\cdot): Y_i = f(X_i)$, s.t. $D^{\text{new}} = \{X_j\} \Rightarrow \{Y_j\}$

- Reinforcement Learning

- Given $D = \{\text{env, actions, rewards, simulator/trace/real game}\}$

learn policy: $e, r \rightarrow a$
utility: $a, e \rightarrow r$, s.t. $\{\text{env, new real game}\} \Rightarrow a_1, a_2, a_3 \dots$

- Active Learning

- Given $D \sim G(\cdot)$, learn $D^{\text{new}} \sim G'(\cdot)$ and $f(\cdot)$, s.t. $D^{\text{all}} \Rightarrow G'(\cdot), \text{policy}, \{Y_j\}$

likrain(261146056) 19:02:59

第一问题@planktonli 我不是很清楚哈哈

这样大家如果问题都说完了, 我就开始解释, 确定可以开始了?

丛(373242125) 19:04:38

Unsupervised Learning 无监督学习, 只能用于类聚?

网神(66707180) 19:04:43

如果回答的内容在接下来的讲课计划里, 可以先不回答。

likrain(261146056) 19:05:32

第一问题@planktonli 我没明白他想问什么,

第二个@丛: 特指我有 N 个样本, 用 x_n, y_n 代表其中一个

第三个@HX: 学习过程可以想象成一种学习机制, 这种机制是模拟人类的学习或者想法的一种过程

然后, 我对于第三个问题举个例子

planktonli(1027753147) 19:06:42

第二个@丛: 无监督学习包括: clustering, dimension reduction, density estimation

likrain(261146056) 19:07:02

例如转导学习: 大家可以这样想象, 我们对于一个学习过程总是希望得到一个规则, 数学上与形式上, 而我们人经常判断不是这样的

你是看到了一些苹果之后, 给你了个苹果你去判断, 你不是个机器需要有个函数规则, 而是通过你看到的以前的苹果直接判断苹果, 基于这种想法的学习过程叫做转导学习, 例如你可以

google, transductive SVM

planktonli(1027753147) 19:08:33

inductive learning 是和 transductive learning 区别的

likrain(261146056) 19:08:36

我先发言到这里，这个。。。区别很明显

planktonli(1027753147) 19:09:09

我补充下，inductive learning 主要用于 semi-supervised learning

likrain(261146056) 19:09:26

不好意思我看错了哈哈，我理解这是一个东西，然后换了个词，不知道其他人意见

=====讨论结束=====

继续

学习理论：一套标准的框架，用统计学，概率论，数学的严格化语言去解释（收敛速率与泛化性能）或者比较不同学习方法与模型的性能。其中最经典的例子：统计学习理论。

统计学习理论的目标：去研究所谓的泛化误差界（Generalization Bounds）

Paradigms of Machine Learning Learning Theory:

Generalization error:

$$\sup_{f \in H} \{ \mathcal{E}(f) - \mathcal{E}_N(f) \} \leq B(N, H, \beta, \sigma, \dots)$$

- Consistency
- Bias vs Variance
- Enough Sample Size
- Learning Rate
- Convergence
- Stability
- Confidence
-

Likrain ()

May 4, 2013 5 / 29

这是一些理论涉及到的 topic，这里面涉及到的数学技巧与数理统计工具比较多，我没法全部解释

International Machine Learning Groups Machine Group:

- Berkeley
- MIT
- CMU
- CSML
- Groups in China
-

Likrain ()

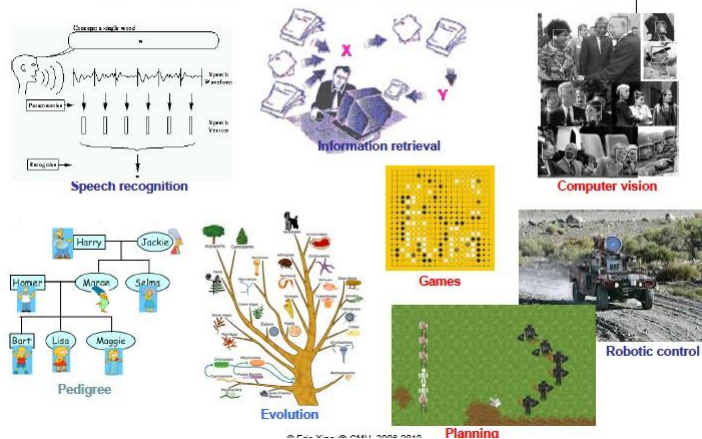
May 4, 2013 6 / 29

机器学习的研究小组：美国，欧洲，中国例子。伯克利的研究小组比较厉害，当然大家主要知道出名的，所谓的乔丹哈哈，MIT 的人工智能实验室、CMU 的 machine learning department、欧洲的 CSML 放在 UCL，还有我们国内的是吧。。。哈哈

机器学习的例子已经深入到我们每天的生活，这里我们可以讨论一下生活中机器学习的例子。大家发言呵呵

planktonli(1027753147) 19:15:43

Where Machine Learning is being used or can be useful?



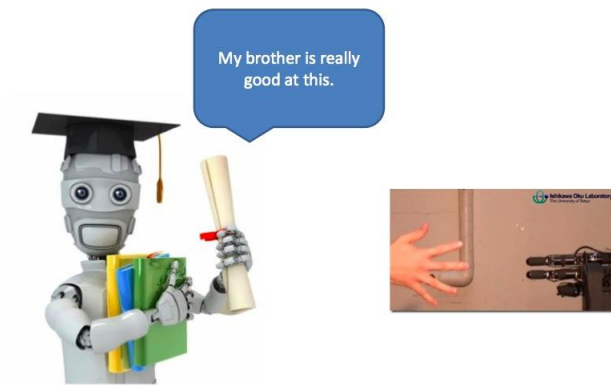
likrain(261146056) 19:17:17

那我讲个例子吧：

Advanced Examples

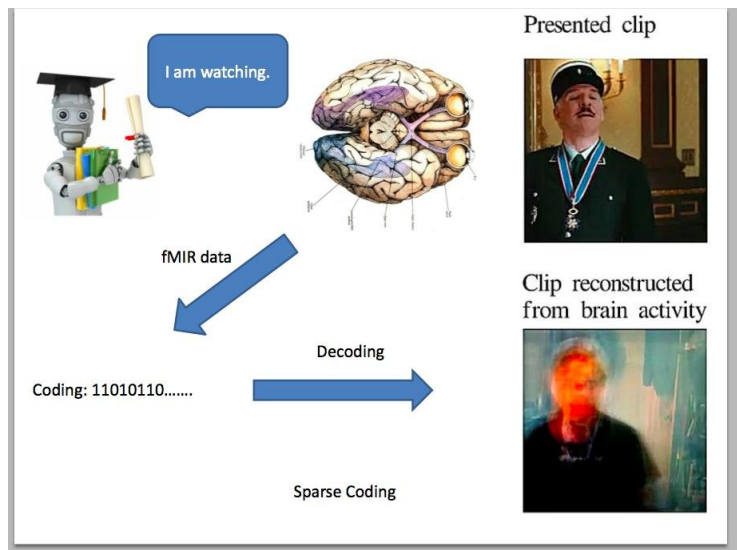
- 野球拳
- 读心术
- Siri

我比较推崇的第一个野球拳：

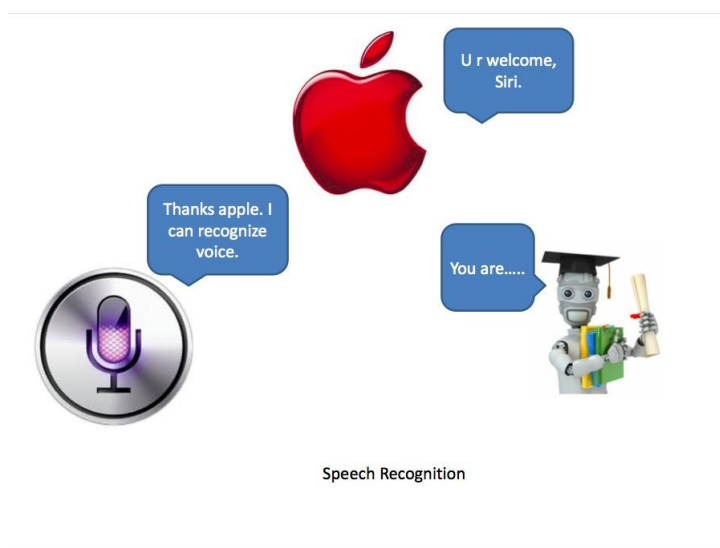


Computer Vision

读心术：



siri :



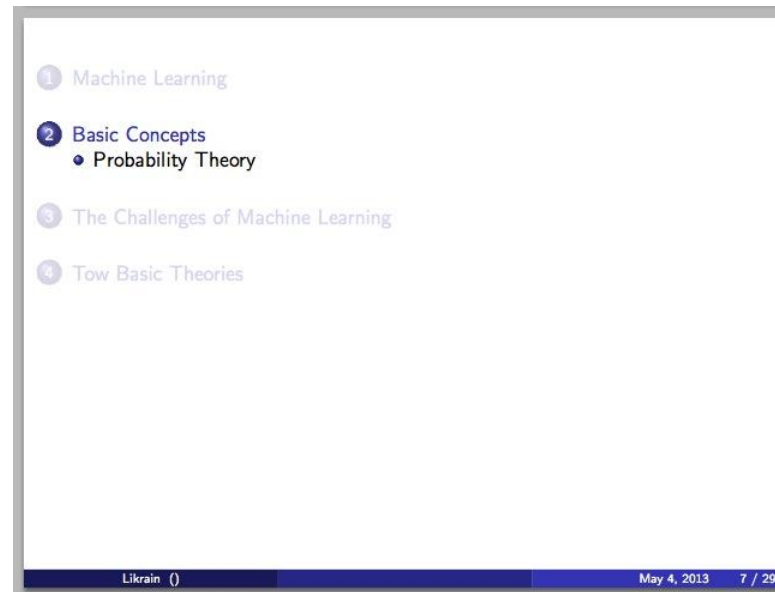
Speech Recognition

这是我以前讲东西的课件，我觉得这三个例子比较有意思。第一个野球拳，男生懂的哈哈，游戏，通过计算机视觉来完成的高速相机拍摄；第二个是最近伯克利刚刚完成的，使用 fMRI 的数据，预测你看到的电影图像。也就说建立一种预测机制，从你的脑部的 fMRI 数据，把你看到的视频解码，所以说读心术，具体方法以后可以介绍。一个潜在的应用就是说，以后不用测谎仪了，我直接可以根据你的脑部的 fMRI 数据去看你看过什么东西。

进入正题

第一部分：概率基本概念

我认为比较简单，我把一些基本知识点贴上来，然后大家讨论。



Probability Theory

- Probability Densities
- Expectations, Covariances and Moments
- The Gaussian Distribution

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$$

The Gaussian distribution defined on a D-dimensional vector x of continuous variables, which is given by

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{2\pi^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right\}$$

我就把书上的抄了下哈，有问题可以提哈

满阶落叶(1316061080) 19:25:06

怎么把高斯分布转到 D 维的？

likrain(261146056) 19:25:33

这个需要推导，我举个例子

planktonli(1027753147) 19:25:46

Expectation: the average value of a function $f(x)$ under a pdf or pmf $p(x)$.

$$\mathbb{E}[f] = \sum_x p(x)f(x), \text{ or } \mathbb{E}[f] = \int p(x)f(x)dx.$$

Given $\{x_n\} i.i.d. \sim p(x)$, the expectation can be approximated by

$$\mathbb{E}[f] \approx \frac{1}{N} \sum_{n=1}^N f(x_n).$$

Conditional expectation w.r.t. a conditional distribution

$$\mathbb{E}_x[f|y] = \int_x p(x|y)f(x)dx,$$

which is a function of y .

A nice property: $\mathbb{E}[\mathbb{E}[x|y]] = \mathbb{E}[x]$.

likrain(261146056) 19:25:50

首先你有 $x_1, x_2, x_3 \dots x_n$, 独立同分布是高斯

planktonli(1027753147) 19:26:17

Variance of $f(x)$ is defined by

$$\text{var}[f] = \mathbb{E}\left[\left(f(x) - \mathbb{E}[f(x)]\right)^2\right]$$

which can be rewritten as

$$\text{var}[f] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$$

In particular, variance of a random variable

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

likrain(261146056) 19:26:28

你把他们的密度函数都乘起来就是他们的密度函数，然后如果他们不是标准正态分布，那么你做个变换 就可以，下面就是会出现相关性怎么处理，也可以做个变换转化为独立的。。。我大概说了下过程。。。ok，那我继续哈

① Machine Learning

② Basic Concepts

③ The Challenges of Machine Learning

- Model Selection
- The Curse of Dimensionality

④ Tow Basic Theories

第二部分：模型选择与高维灾难

PRML 中使用多项式拟合的例子引出了模型选择的概念。这里大家可以从两个方面来理解。对于这个例子存在两个模型选择问题：第一函数空间的选择：即去拟合三角函数为何使用了多项式函数。第二确定使用多项式拟合后，多项式的阶数问题。

A Example: Polynomial Curve Fitting

Now suppose that we generated a training set from function $\sin(2\pi x)$ comprising N observations of

$$\{(x_n, t_n) | x_n \in [0, 1], t_n = \sin(2x_n\pi) + \text{random noise}\}_{n=1}^N.$$

Our goal is to exploit this training set in order to make predictions of the value t_{new} of the target variable for some new x_{new} of the input variable. For the moment, we can suppose the target function has the polynomial form, namely, $y(x, w) = \sum_{i=1}^M w_i x^i$, where the parameter M is called order of the polynomial function.

Normally, we can use error function that measures the misfit between the function $y(x, w)$, for any given value of w , and training set data points. For example: the sum of error squares function

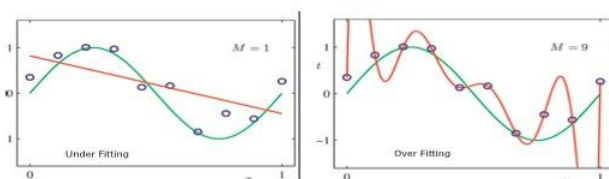
$$w^* = \arg \min_w \frac{1}{2} \sum_{n=1}^N \{y_n(x, w) - t_n\}^2.$$

Likrain ()

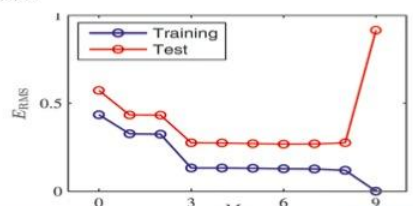
May 4, 2013 8 / 29

进一步的，例子中给出了使用不同阶数去拟合该三角函数的如何产生过拟合现象。避免过拟合现象的基本方法：正则化方法，regularization.

Model Selection



In this example, the model selection that is how to select the order of the polynomial function.



Likrain ()

May 4, 2013 9 / 29

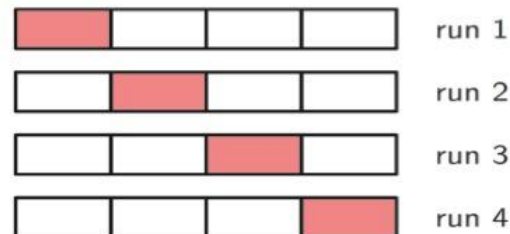
理解正则化方法有很多，这里我解释其中一种，正则化方法你可以理解为，我对于我选择函数空间的函数做了一种限制。使得这个函数空间比原来的函数空间小，所以不会把过分拟合的函数选择进入需要的函数空间。

例如：例子中使用了多项式空间，但是加入正则化之后等价于对于一些函数空间的限制，那些过分拟合的9次多项式不会被选择进来。

模型选择方法：必须要懂的至少要有交叉验证，AIC。

Model Selection Method

- Cross-Validation



- Leave-One-Out(Loocv)

- Information Criterion

$$IC_M = -2\{\log(L_M(\mathbf{y}(\mathbf{x}, \mathbf{w}), \mathbf{x})) - \phi(\mathbf{N})\mathbf{M}\}.$$

=====讨论=====

这里我先讲到这里哈哈，大家可以开始发问了哈，这里比较模糊。。。。很难文字说的特别清楚，如果不做预习，可能以前没看过的人就比较困难了。

范涛@推荐系统(289765648) 19:35:33

做回归分析也有必要做交叉验证吗？

满阶落叶(1316061080) 19:35:47

交叉验证能稍微解释一下嘛？举个例子。。

范涛@推荐系统(289765648) 19:35:57

回归分析时候，不是有假设检验码，看 p-value 不可以吗？

HX(458728037) 19:36:14

cross validation 是一种正则化的方法吗？

lost(549294286) 19:37:49

AIC ?

likrain(261146056) 19:38:23

@范涛：交叉验证只是模型选择的一种方法，如果你有模型选择问题，你就可以用交叉验证。例如你做线性回归，你有 10 个变量，你就有 1024 个模型需要选择，你就可以使用交叉验证或者 AIC，做任何一件事情，都会从不同的目的去做，使用交叉验证是从预测的角度去做，使用 AIC 是从模型的复杂度与模型的拟合角度去做。

范涛@推荐系统(289765648) 19:40:20

那我们回归分析时候，经常会看变量的 p-value

likrain(261146056) 19:40:23

使用 p-value 使用假设检验的角度去做，模型选择都是选择方法。

范涛@推荐系统(289765648) 19:40:36

哦，明白

likrain(261146056) 19:40:43

只是传统的统计学里面对于线性模型给出了这种方式

lost(549294286) 19:40:58

AIC, 最小信息准则。。

likrain(261146056) 19:41:07

可能是这个名字吧, 还有 BIC、EBIC 反正很多

DM-福(864914991) 19:41:30

$$IC_M = -2\{\log(L_M(\mathbf{y}(\mathbf{x}, \mathbf{w}), \mathbf{x})) - \phi(\mathbf{N})M\}.$$

这个是信息熵吗

likrain(261146056) 19:41:37

不是, 第一项是似然函数, 第二项是模型复杂度

范涛@推荐系统(289765648) 19:42:01

交叉验证, 现在有没有加速方法, 变量多的话, 模型成指数增长啊

DM-福(864914991) 19:42:10

信息增益好像也是这样写的

likrain(261146056) 19:42:39

指数增长? 不对吧哈哈, 加速方法我没感觉到有啥哈

DM-福(864914991) 19:43:00

2^n

范涛@推荐系统(289765648) 19:43:27

说错了, 恩 2^n

likrain(261146056) 19:43:35

信息熵是有的, 交叉验证没有

范涛@推荐系统(289765648) 19:44:09

2^n 个模型, 都验证, 貌似不现实

likrain(261146056) 19:44:13

所以信息熵可以有办法改进, 我可以告诉你们一些 paper

范涛@推荐系统(289765648) 19:44:37

现在在互联网上都是至少几百万维阿

likrain(261146056) 19:44:57

所以都不是这么做的, 互联网哈哈, 那个啥。。。。家里吃饭中哈哈, 咱们稍微暂停哈哈, 😊 抱歉

网络上的尼采(813394698) 19:45:51

好, 你先去吃饭, 现在自由讨论

likrain(261146056) 19:45:56

嗯我回来解释哈哈

苍井空指导老师(794717493) 19:46:14

@likrain 例如: 例子中使用了多项式空间, 但是加入正则化之后等价于对于一些函数空间的限制, 那些过分拟合的 9 次多项式不会被选择进来。9 次项也出现吧。。。只是系数比较小

HX(458728037) 19:46:26

这个公式估计就是交叉验证希望的目标函数最小的公式吧

苍井空指导老师(794717493) 19:48:51

大家对机器学习研究的定位怎么看。。。很多问题在数学上统计上都是很古老的问题了

范涛@推荐系统(289765648) 19:49:24

的确是这样的, 我只关注互联网上应用

网神(66707180) 19:49:44

机器学习热主要是现在互联网应用引起的，因为互联网有大量的数据，这是以前的机器学习或数据挖掘缺少的。

范涛@推荐系统(289765648) 19:50:19

还有互联网数据规模带来的一系列新问题

HX(458728037) 19:50:32

你意思是说常规的很多机器学习算法在 互联网的领域根本没法用？

范涛@推荐系统(289765648) 19:50:45

很多事这样的，尤其那种时间复杂度高的

HX(458728037) 19:51:33

那互联网一般都是怎么处理的呢？降维吗？

范涛@推荐系统(289765648) 19:52:26

这个得找群里那些资深人士解读下了

HX(458728037) 19:52:26

还有就是那你说的维数很大 会有多大呢

丛(373242125) 19:52:49

文本分类中,有几万维吧.

planktonli(1027753147) 19:52:59

恩,google 的有这么大，他们采用的是 naive bayes + distributed computing

范涛@推荐系统(289765648) 19:53:26

文本几万维不止吧

Xiaoming(50777501) 19:53:35

既然是第一章，我们说说大方向问题吧。现在很多 ML 方法都涉及统计。大家总体感觉，说说统计究竟是不是一个正确的方向？

范涛@推荐系统(289765648) 19:53:42

几万维现在貌似真不算什么

planktonli(1027753147) 19:53:53

google 一次要跑 1TB 的数据，要一次性载入

HX(458728037) 19:54:21

我没做过文本 都是做图像，图像里面的特征 最大也就是原图的像素的数量，为什么文本的会有那么大的维数呢？

丛(373242125) 19:55:01

词向量

阿邦(1549614810) 19:55:04

n gram

苍井空指导老师(794717493) 19:55:06

因为词汇量

HX(458728037) 19:55:30

文本 怎么转化为向量的？

苍井空指导老师(794717493) 19:55:42

词频

丛(373242125) 19:55:42

中文分词

planktonli(1027753147) 19:55:45

To Xiaoming: 统计的东西现在占主流, 还有一些 优化+几何的东西

网神(66707180) 19:56:22

请问 HX 图像提取像素特征, 性能上有没有测试过, 比如提取特征平均时间在什么量级, 几百微妙还是几十毫秒?

范涛@推荐系统(289765648) 19:56:44

最优化蛮有用的

HX(458728037) 19:57:06

图像特征有简单的也有复杂的, 然后还要看原图的分辨率大小, 但是 还真没测试到微妙级别过

范涛@推荐系统(289765648) 19:57:43

彩色图像维度也挺大的吧

网神(66707180) 19:58:05

都在几十毫秒-几百毫秒范围吗

planktonli(1027753147) 19:58:44

图像特征看是什么了, RGB histgoram, 纹理的快些, SIFT\shape context 这些速度都不快

HX(458728037) 19:58:54

假如是一张 800*600 像素的图像, 那么最大的特征也就是 800*600*3 这么大, 一般不可能再大了

Xiaoming(50777501) 20:00:26

To planktonli: 是啊。感觉统计和几何是能不是喔解决问题, 但是 "Learning" 的本质并不是一定基于统计, 这个很难探讨。现在是, 统计能解决到一些问题, 就说机器可以学习, 就盖上 "学习" 的帽子了。但本质呢, 不一定的。

HX(458728037) 20:01:24

我怎么感觉机器学习 全都是建立在统计上面做的

丛(373242125) 20:01:43

神经网络训练出来的模型, 人类容易理解么?

planktonli(1027753147) 20:01:57

To Xiaoming: 不过统计本身在表达很多现实世界的时候有一定优势, 因此 出现了 统计物理\统计金融, 其实 机器学习现在可以看成 统计计算机的东西

苍井空指导老师(794717493) 20:02:20

人类为什么要理解 @丛

Xiaoming(50777501) 20:02:27

To HX: 如果这样, 应该叫"机器统计", 而不是机器学习。

丛(373242125) 20:02:58

希望计算出来的模型, 是否符合人类的逻辑。

HX(458728037) 20:03:03

可以举出一个例子 某种分类器的算法不是基于统计的?

苍井空指导老师(794717493) 20:03:17

感觉 机器学习 就是 高级数据拟合 🤖

planktonli(1027753147) 20:03:27

很多啊, neural network/support vector machine

丛(373242125) 20:03:48

或者说, 人类希望从计算出来的模型提取有用的信息。

planktonli(1027753147) 20:03:49

可以看成优化的

Xiaoming(50777501) 20:03:52

@丛 神经网络 是不基于统计，但并不代表现有的 NN 就是机器学习的正解。

planktonli(1027753147) 20:04:02

其实数学的很多分枝也是交融的

HX(458728037) 20:04:06

SVM 不是典型的基于统计的吗？

逆风飞扬(374350284) 20:04:52

统计学习理论

planktonli(1027753147) 20:05:03

SVM 只是统计的解释多些，真正的求解和 model 是优化的东西的

likrain(261146056) 20:04:53

我回来了 哈哈，大家可以总结一下争论的问题吗？主持可以发个言，哈哈

网神(66707180) 20:06:14

水平有限，大家好像在讨论机器学习完全是基于统计的吗

范涛@推荐系统(289765648) 20:06:19

按这样说，各种回归，分类器的参数计算，哪个不是通过最优化训练求解出来的

likrain(261146056) 20:06:28

看了一下大家争论的问题，我觉得本身的争论可能不需要，因为我刚才写的定义，大家可能需要看看。

针对经验性数据,使用计算机通过模拟人类学习和获取信息的准则来处理以预测为终极目标问题的一系列过程称为机器学习。这一系列过程是“学习过程”，包括统计建模,优化处理,算法设计和统计分析，几乎所有的机器学习的模型都有统计解释，所以叫做统计建模。

有了模型，就要去学习一些目的，这些目的得到了优化模型，解优化模型，就要设计算法，完成算法就需要写 code，所以本来就是交叉学科。最后分析理论。

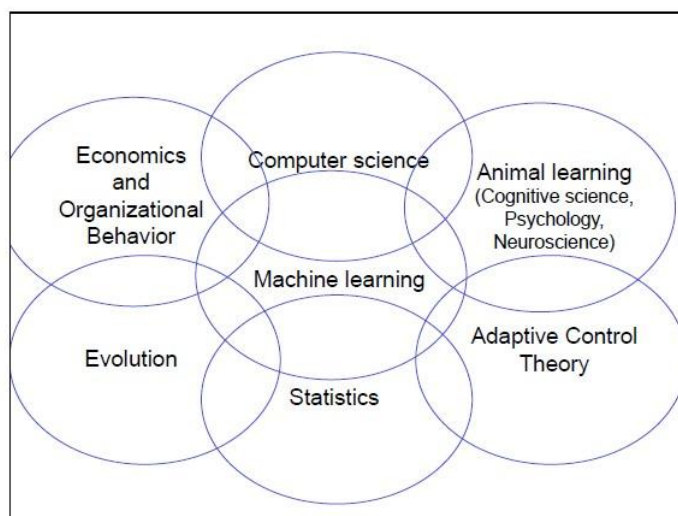
苍井空指导老师(794717493) 20:08:39

理解成高级数据拟合可以么？

likrain(261146056) 20:08:52

远远超出拟合，因为遇到的问题，已经远远超出了拟合的范畴。

planktonli(1027753147) 20:09:21



=====讨论结束=====

likrain(261146056) 20:10:28

The relationship of the Least Squares and Maximum Likelihood

First, we assume that, given the value of x , the corresponding value of t has a Gaussian distribution with a mean equal to the value $y(x, w)$ of the polynomial curve. We have

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}).$$

Likelihood Function:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n|y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}).$$

Maximum Likelihood:

$$w_{ML} = \arg \max \ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \arg \max \left\{ -\frac{\beta}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \right\}.$$

Maximum Likelihood method under the assumption that error term has the Gaussian distribution equal to minimizing the sum of squares error function.

Likrain ()

May 4, 2013

12 / 29

网神(66707180) 20:10:31

欢迎 likrain 继续 

likrain(261146056) 20:10:39

The relationship of the Bayesian Method and Regularization Method

By maximum likelihood method we can obtain $p(t|x, \mathbf{w}_{ML}, \beta_{ML})$, where

$$\beta_{ML}^{-1} = \frac{1}{N} \sum_{n=1}^N \{y(x_n, \mathbf{w}_{ML}) - t_n\}^2.$$

- Prior Distribution

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\alpha^{-1}\mathbf{I}) = \frac{\alpha^{(M+1)/2}}{2\pi} \exp\left\{-\frac{\alpha}{2} \mathbf{w}^T \mathbf{w}\right\}$$

- Posterior Distribution

$$P(w|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{w}, \mathbf{x}, \beta) p(\mathbf{w}|\alpha).$$

- Maximum posterior is equivalent to regularization method.

$$w_{MP} = \arg \min \frac{\beta}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w}$$

Likrain ()

May 4, 2013

13 / 29

likrain(261146056) 20:10:44

两个重要观点：

最小二乘数学建模等价于高斯噪声最大释然估计统计建模，正则化最小二成等价于基于高斯噪声的最大化后验概率统计建模。

这里就像我说的，几乎所有的机器学习方法也许建立之初没有什么统计解释，最后大家发现，都可以通过统计的原理解释。

所以这里只是两个简单的例子，所以因为这些观点，所以也就只需要从统计建模，建立机器学习模型就好了，所以就是统计机器学习。

planktonli(1027753147) 20:12:25

我感觉这样是不是有些牵强啊，先有了方法,然后用统计的东西给个模型解释。

likrain(261146056) 20:12:41

针对数据的建模过程，基于概率分布的建模过程，都是最后解释成了统计机器学习，发挥的淋漓尽致的就是 graphic model。

@planktonli：所以既然都有统计解释，为什么不直接从统计上直接建模呢，这本书基本就是这个观点，所以才要说 generative model，所谓生成模型。

planktonli(1027753147) 20:14:36

@likrain,明白

范涛@推荐系统(289765648) 20:14:40

回到最小二乘和最大似然的关系吧

likrain(261146056) 20:14:41

我刚才发的 ppt 有问题吗？

HX(458728037) 20:14:53

generative model 和 discriminative model

范涛@推荐系统(289765648) 20:14:56

都是 loss function 问题，是吧？

likrain(261146056) 20:15:23

loss function?

planktonli(1027753147) 20:15:35



likrain(261146056) 20:15:37

为什么突然提到这里？

planktonli(1027753147) 20:15:55

都是 objective function

likrain(261146056) 20:16:06

你可以理解为： l_2 loss function = gaussian noise

所以就是统计解释

范涛@推荐系统(289765648) 20:16:44

我理解的最小二乘，无非就是求解模型参数的方法

likrain(261146056) 20:16:53

是的，这个就是数学建模，你可以想想，牛顿和你的理解是一样的，所以发明了牛顿第二定律。

而统计学家说 ok：我给你个统计解释，只要是高斯噪声，对应的从最大熵估计就是最小二乘，所以这是统计建模。

lost(549294286) 20:18:09



likrain(261146056) 20:18:29

所以如果你的模型是个线性回归，你的 noise 是拉普拉斯

范涛@推荐系统(289765648) 20:18:38

我们做回归分析，一个好的模型出来的，残差分布最好是个正态分布？？

likrain(261146056) 20:18:45

你如果用最小二乘，你就完了，正确的应该用最小一乘，叫做 LAD，这是一个非常大的领域。

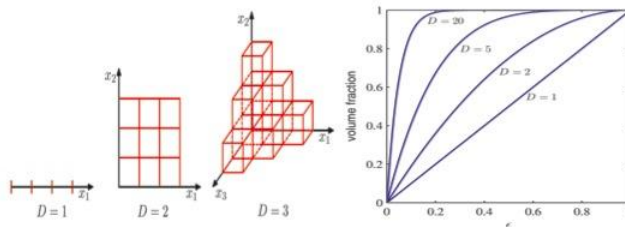
机器学习上面叫做=误差建模，统计上面=稳健估计

我回答这个问题结束。。。。。

维数灾难：

The Curse of Dimensionality

- How to select the current model? There are many models can be selected.
For example: $x_1, x_2, x_3, 4^3$.



- In high dimension vector space, we can see anything that is not clear.

Likrain ()

May 4, 2013 15 / 29

两个方面：

第一，模型的复杂性。

由于维数很大，简单的例子就是如果我们有 n 个变量那么我们如果回归也有 2^n 个模型
使用多项式函数空间，所以这几乎很难做到，如果我们函数空间选的更大，那么几乎是无法完成的。

第二，几何体的难以想象的各种突变

这个地方没有很多经验我个人觉得比较难理解，你如何想象高维空间中的球体的数据，其实都集中在球壳附近。。。。。

如何想象高维空间的各种几何体，其实和三维空间中的完全不一样。我没有什么好的建议，如果大家真的想看看，就去学学 Functional Geometrical Analysis. 至于书上的例子，我不知道大家有什么问题没有。

对于最后一个高斯分布的例子，需要自己推导一下。这里我没时间，还没有具体完成呵呵呵

GFA 我学过一个学期，直接崩溃，颠覆我对世界的认识，有时间大家可以尝试读读，它会告诉你高维空间的数据分布的一些惊人的例子

ok，这里暂停一下哈，讨论 😊

planktonli(1027753147) 20:25:15

恩,Functional Geometrical Analysis 是比较难，和直观的现实世界完全不同，尤其 high dimension 的时候。

likrain(261146056) 20:25:55

如果不去读是没理解的。。。。所以很多高维空间的数据处理的方式都是从数据本身高维空间的“样子”给出的，当然所谓高维空间说的有些含糊了，严格的要给出各种度量，各种测度等。

planktonli(1027753147) 20:27:39

恩

likrain(261146056) 20:28:04

ok 那我继续哈哈

lost(549294286) 20:28:11

2^n 个模型

模型的个数用变量数 n 衡量吗

likrain(261146056)

是的

- ① Machine Learning
- ② Basic Concepts
- ③ The Challenges of Machine Learning
- ④ Tow Basic Theories
 - Information Theory

lost(549294286) 20:28:38

那非线性拟合

likrain(261146056) 20:28:53

那就更复杂了，度量函数空间，简单的例子

planktonli(1027753147) 20:29:16

范函

likrain(261146056) 20:29:20

你可以去用所谓的“数数”

lost(549294286) 20:29:28

额。。

likrain(261146056) 20:29:30

去数多项式空间，你怎么去“数数”，三角函数空间

所以 vapnik 在统计学习那本书里面写了那么多看似乱七八糟的定理，各种熵，VC 维才能度量函数空间
然后再去数数。

lost(549294286) 20:30:39

还是继续把

likrain(261146056) 20:30:45

哦了

Entropy

How to express the information of a event?

The measure of the information is related to the probability of the event.

If two events x, y are unrelated, then they satisfies

$$h(x, y) = h(x) + h(y).$$

So, the measure of the information is defined by $h(x) = -\log_2 p(x)$ (binary digits = bit)

信息论，是机器学习的很多方面的另一种解释，可以用它去解释很多机器学习的模型，所以也可以想想很多人做信息论的就直接使用信息论的方法，重新建模机器学习方法。

这里可以举个国内的吧，jun zhu，可以看看他的东西。

Entropy

- Entropy for discrete random variable

$$H(x) = - \sum p(x) \ln p(x).$$

- Entropy for continued random variable

$$H(x) = - \int p(x) \ln p(x) dx.$$

Maximum Entropy

- Maximum Entropy for discrete random variable corresponded to x is form the homogeneous distribution.
- Maximum Entropy for continued random variable corresponded to x is form the Gaussian distribution.

对于信息论~~这个领域本身太大了，我就把我认为重要的概念贴上来了。。。。

How to use the KL divergence.

How to use the KL divergence.

- Minimizing the KL divergence is equivalent to maximizing the likelihood function.

$$KL(p||q) \approx \sum_{n=1}^N \{-\ln q(x_n|\theta) + \ln p(x_n)\}$$

- Mutual information

$$I(x, y) = KL(p(x, y) || p(x)p(y)) = - \int \int p(x, y) \ln \left\{ \frac{p(x)p(y)}{p(x, y)} \right\} dx dy.$$

KL Divergence

Consider some unknown distribution $p(x)$, and suppose that we have modelled this using an approximating distribution $q(x)$. As this viewpoint we must give a measure of the dissimilarity of the two distribution $p(x)$ and $q(x)$.

- relative entropy or KL divergence

$$KL(p||q) = - \int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} dx.$$

- The property of the KL divergence

$$KL(p||q) \neq KL(q||p)$$

$$KL(p||q) \geq 0$$

好了我还真不知道这些概念需要说什么哈哈

就说这么多这次哈哈，多谢大家耐心听我唠叨

shrake-DM(965229647) 21:43:44

统计与机器学习 ikrain 已经解释的十分全面了，只是补充一下，最小二乘用的是 square loss；svm 是 hinge loss；所以说前者是统计的，后者在这个意义下也应该是可以划入统计范畴的，而且 alex 及其追随者，把 loss 这里作了很多非常统一的 common sense，2000 年左右无数本书，可以看看，前面 ikrain 都提到了；GFA 有时间可以学下，cmu 有这个相关的课，很有启发，对于 random projection 启发大一些。我忘了很多了，但是高维空间的球的质量分布在球壳上或赤道上（记不清了），这个比较违反我们的直觉。一个统计的应用是高维高斯分布（维数真的要很高），随机产生点，球内是几乎找不到的，只有在球壳（或是赤道）这点忘了，出了球壳记得也是几乎没有点的。

第二章 Probability Distributions

主讲人 网络上的尼采

(新浪微博:@Nietzsche_复杂网络机器学习)

网络上的尼采(813394698) 9:11:56

开始吧，先不要发言了，先讲 PRML 第二章 Probability Distributions。今天的内容比较多，还是边思考边打字，会比较慢，大家不要着急，上午讲不完下午会接着讲。

顾名思义，PRML 第二章 Probability Distributions 的主要内容有：伯努利分布、二项式 -beta 共轭分布、多项式分布 -狄利克雷共轭分布、高斯分布、频率派和贝叶斯派的区别联系、指数族等。

先看最简单的伯努利分布：

$$\text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x} \quad (2.2)$$

which is known as the *Bernoulli* distribution. It is easily verified that this distribution is normalized and that it has mean and variance given by

$$\mathbb{E}[x] = \mu \quad (2.3)$$

$$\text{var}[x] = \mu(1-\mu). \quad (2.4)$$

最简单的例子就是抛硬币，正反面的概率。

再看二项式分布：

written

$$\text{Bin}(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}$$

where

$$\binom{N}{m} \equiv \frac{N!}{(N-m)!m!}$$

抛 N 次有 m 次是正面或反面的概率，所以伯努利分布是二项式分布的特例。

向大家推荐一本好书，陈希孺的《数理统计简史》，对数理统计的一些基本东西的来龙去脉介绍的很详细，这样有助于理解。先 818 二项式分布，正态分布被发现前，二项式分布是大家研究的主要内容。

由二项式分布可以推出其他很多分布形式，比如泊松定理：

$$\lim_{n \rightarrow \infty} \binom{n}{x} p_n^x (1-p_n)^{n-x} = \frac{\lambda^x}{x!} e^{-\lambda}$$

泊松分布是二项式分布的极限形式，这个估计大家都推导过。由二项式分布也能推出正态分布。

贝叶斯思想也是当时对二项式分布做估计产生的，后来沉寂了一百多年。

数据少时用最大似然方法估计参数会过拟合，而贝叶斯方法认为模型参数有一个先验分布，因此共轭分布在贝叶斯方法中很重要，现在看二项式分布的共轭分布 beta 分布：

$$\text{Beta}(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1} (1-\mu)^{b-1} \quad (2.13)$$

where $\Gamma(x)$ is the gamma function defined by (1.141), and the coefficient in (2.13) ensures that the beta distribution is normalized, so that

$$\int_0^1 \text{Beta}(\mu|a, b) d\mu = 1. \quad (2.14)$$

The mean and variance of the beta distribution are given by

$$\mathbb{E}[\mu] = \frac{a}{a+b} \quad (2.15)$$

$$\text{var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)}. \quad (2.16)$$

结合上面的二项式分布的形式,不难看出 beta 分布和二项式分布的似然函数有着相同的形式,这样用 beta 分布做二项式分布参数的先验分布,乘似然函数以后得到的后验分布依然是 beta 分布。
a b 是超参,大家可以看到 beta 分布的形式非常灵活:

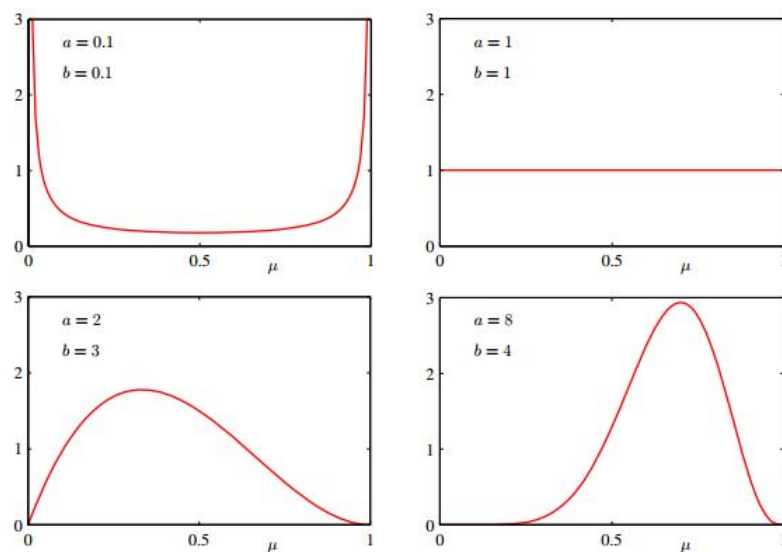


Figure 2.2 Plots of the beta distribution $\text{Beta}(\mu|a, b)$ given by (2.13) as a function of μ for various values of the hyperparameters a and b .

假设抛硬币 N 次, l 和 m 分别为正反面的记数,那么参数的后验分布便是:

$$p(\mu|m, l, a, b) = \frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)} \mu^{m+a-1} (1-\mu)^{l+b-1}. \quad (2)$$

不难看出,后验分布是先验和数据共同作用的结果。

这种数据矫正先验的形式可以通过序列的形式进行,非常适合在线学习。

单拿一步来说明问题:

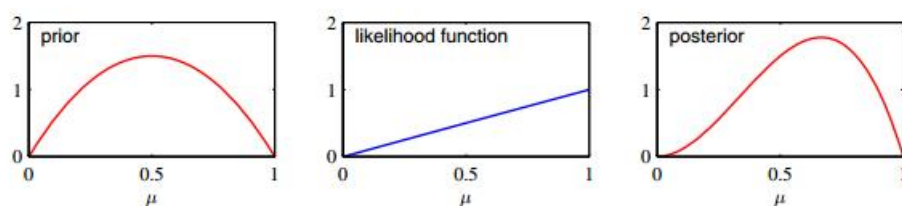


Figure 2.3 Illustration of one step of sequential Bayesian inference. The prior is given by a beta distribution with parameters $a = 2, b = 2$, and the likelihood function, given by (2.9) with $N = m = 1$, corresponds to a single observation of $x = 1$, so that the posterior is given by a beta distribution with parameters $a = 3, b = 2$.

可以看出, a 的记数增加了 1。

书上通过序列数据流的形式来矫正先验的描述,每次可以用一个观测数据也可以用 small batches,很适合实时的学习:

We see that this *sequential* approach to learning arises naturally when we adopt a Bayesian viewpoint. It is independent of the choice of prior and of the likelihood function and depends only on the assumption of i.i.d. data. Sequential methods make use of observations one at a time, or in small batches, and then discard them before the next observations are used. They can be used, for example, in real-time learning scenarios where a steady stream of data is arriving, and predictions must be made before all of the data is seen. Because they do not require the whole data set to be stored or loaded into memory, sequential methods are also useful for large data sets. Maximum likelihood methods can also be cast into a sequential framework.

回到上面的二项式-beta 共轭，随着数据的增加， m, l 趋于无穷大时，这时参数的后验分布就等于最大似然解。

有些先验分布可以证明，随着数据的增加方差越来越小，分布越来越陡，最后坍缩成狄拉克函数，这时贝叶斯方法和频率派方法是等价的。举个第三章的贝叶斯线性回归的例子，对于下图中间参数 W 的高斯先验分布，随着数据不断增加，参数后验分布的不确定性逐渐减少，朝一个点坍缩：

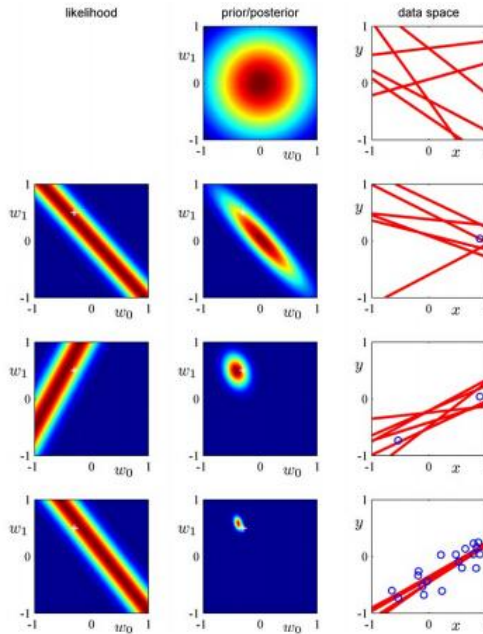


Figure 3.7 Illustration of sequential Bayesian learning for a simple linear model of the form $y(x, w) = w_0 + w_1 x$. A detailed description of this figure is given in the text.

接着看多项式分布：把抛硬币换成了掷骰子

We can consider the joint distribution of the quantities $m_1, \dots, m_K$, conditioned on the parameters μ and on the total number N of observations. From (2.29) this takes the form

$$\text{Mult}(m_1, m_2, \dots, m_K | \mu, N) = \binom{N}{m_1 m_2 \dots m_K} \prod_{k=1}^K \mu_k^{m_k} \quad (2.34)$$

which is known as the *multinomial* distribution. The normalization coefficient is the number of ways of partitioning N objects into K groups of size $m_1, \dots, m_K$ and is given by

$$\binom{N}{m_1 m_2 \dots m_K} = \frac{N!}{m_1! m_2! \dots m_K!}. \quad (2.35)$$

Note that the variables m_k are subject to the constraint

$$\sum_{k=1}^K m_k = N. \quad (2.36)$$

同样它的共轭分布狄利克雷分布也得和似然函数保持相同的形式。

狄利克雷分布：

The normalized form for this distribution is by

$$\text{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \prod_{k=1}^K \mu_k^{\alpha_k-1} \quad (2.38)$$

which is called the *Dirichlet* distribution. Here $\Gamma(x)$ is the gamma function defined by (1.141) while

$$\alpha_0 = \sum_{k=1}^K \alpha_k. \quad (2.39)$$

后验形式：

Multiplying the prior (2.38) by the likelihood function (2.34), we obtain the posterior distribution for the parameters $\{\mu_k\}$ in the form

$$p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\alpha}) \propto p(\mathcal{D}|\boldsymbol{\mu})p(\boldsymbol{\mu}|\boldsymbol{\alpha}) \propto \prod_{k=1}^K \mu_k^{\alpha_k+m_k-1}. \quad (2.40)$$

We see that the posterior distribution again takes the form of a Dirichlet distribution, confirming that the Dirichlet is indeed a conjugate prior for the multinomial. This allows us to determine the normalization coefficient by comparison with (2.38) so that

$$\begin{aligned} p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\alpha}) &= \text{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha} + \mathbf{m}) \\ &= \frac{\Gamma(\alpha_0 + N)}{\Gamma(\alpha_1 + m_1) \cdots \Gamma(\alpha_K + m_K)} \prod_{k=1}^K \mu_k^{\alpha_k+m_k-1} \end{aligned} \quad (2.41)$$

大家依然能看到记数。

下面讲高斯分布，大家看高斯分布的形式：

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$$

多元高斯分布的形式：

multivariate Gaussian distribution takes the form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}$$

高斯分布有着优良的性质，便于推导，很多时候会得到解析解。一元高斯分布是个钟形的曲线，大部分都集中在均值附近，朝两边的概率呈指数衰减，这个可以用契比雪夫不等式来说明，偏离均值超过 3 个标准差的概率就非常低了：

The standard deviation is a simple but valuable measure of how far values of x are likely to depart from the mean. Its very name suggests that it is the standard or typical amount one should expect a randomly drawn value for x to deviate or differ from μ . *Chebyshev's inequality* (or Bienaymé-Chebyshev inequality) provides a mathematical relation between the standard deviation and $|x - \mu|$:

$$\Pr\{|x - \mu| > n\sigma\} \leq \frac{1}{n^2}. \quad (45)$$

This inequality is not a tight bound (and it is useless for $n < 1$); a more practical rule of thumb, which strictly speaking is true only for the normal distribution, is that 68% of the values will lie within one, 95% within two, and 99.7% within three standard deviations of the mean (Fig. A.1). Nevertheless, Chebyshev's inequality shows the

正态分布是如何发现的，在《数理统计简史》有详细的介绍，当时已经有很多人包括拉普拉斯在找随机误差的分布形式，都没有找到，高斯是出于一个假设找到的，也就是随机误差分布的最大似然解是算数平均值，只有正态分布这个函数满足这个要求。

然后高斯进一步将随机误差的正态分布假设和最小二乘联系到了一块，两者是等价的：

使用这个误差分布，就容易对最小二乘法给出一种解释。回到四章的方程(3)，其中 $(x_0, \dots, x_n)$, $i=1, \dots, n$ ，是观测数据。记

$$e_i = x_{0i} + x_1\theta_1 + \dots + x_n\theta_n, 1 \leq i \leq n.$$

理论它们应为 0，但因有测量误差存在，实际不必为 0，故 $e_1, \dots,$

e_n 可视误差。按高斯的第一个原则(极大似然)，结合误差密度(11)， $(e_1, \dots, e_n)$ 的概率为

$$(\sqrt{2\pi}h)^{-n} \exp \left\{ -\frac{1}{2h^2} \sum_{i=1}^n (x_{0i} + x_1\theta_1 + \dots + x_n\theta_n)^2 \right\}.$$

要此式达到最大，必须取 $\theta_1, \dots, \theta_n$ 之值，使表达式 $\sum_{i=1}^n (x_{0i} + x_1\theta_1 + \dots + x_n\theta_n)^2$ 达到最小，于是得到 $\theta_1, \dots, \theta_n$ 的最小二乘估计。要注

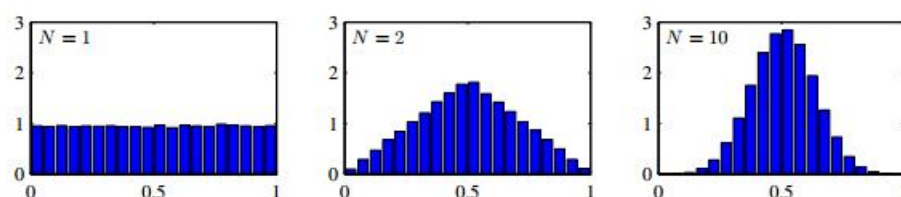
后来就是拉普拉斯迅速跟进，提出了中心极限定理，大量随机变量的和呈正态分布，这样解释了随机误差是正态分布的原因。中心极限定理的公式：

Now consider N random variables with pdf's (not necessarily Gaussian) $p(x_i)$, each with mean μ and variance σ^2 . We assume each variable is **independent and identically distributed** or **iid** for short. Let $S_N = \sum_{i=1}^N X_i$ be the sum of the rv's. This is a simple but widely used transformation of rv's. One can show that, as N increases, the distribution of this sum approaches

$$p(S_N = s) = \frac{1}{\sqrt{2\pi N\sigma^2}} \exp \left(-\frac{(s - N\mu)^2}{2N\sigma^2} \right) \quad (2.96)$$

大家看 PRML 上的图，很形象的说明高斯分布是怎么生长出来的：

Figure 2.6 Histogram plots of the mean of N uniformly distributed numbers for various values of N . We observe that as N increases, the distribution tends towards a Gaussian.



从[0,1]随机取 N 个变量，然后算它们的算术平均，随着 N 的增大，均值的分布逐渐呈现出高斯分布，可以比较直观的了解中心极限定理。

接着看高斯分布的几何形式：

先给出样本到均值的马氏距离 $\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T.$$

把协方差矩阵的逆 带入上式

$$\Delta^2 = \sum_{i=1}^D \frac{y_i^2}{\lambda_i}$$

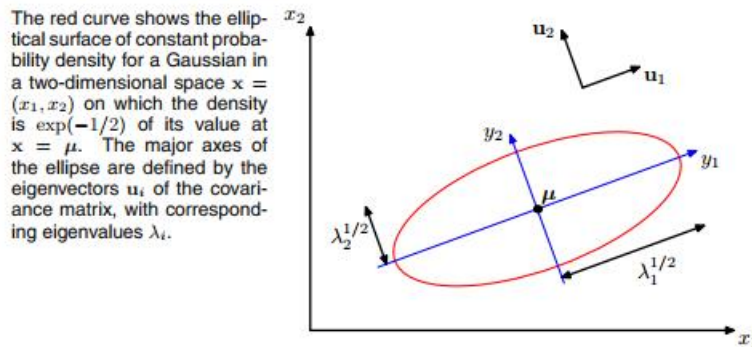
会得到以协方差矩阵的特征值平方根为轴长的标准椭圆方程

其中 $y_i = \mathbf{u}_i^T (\mathbf{x} - \boldsymbol{\mu})$.

$\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$ ，也就是原来的坐标系经过平移和旋转，由协方差矩阵特征向量组成的矩阵 $\mathbf{U}$ 负责旋转

坐标轴。

看下面张图就很明白了：



接着是条件高斯分布和边缘高斯分布，这两个分布由高斯分布组成，自身也是高斯分布。

条件高斯分布的推导过程略过，大家记住这个结论：

From these we obtain the following expressions for the mean and covariance of the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \quad (2.81)$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba}. \quad (2.82)$$

上面是条件高斯分布的均值和方差，以后的 Gaussian Processes 在最后预测时会用到均值。

另一个是线性高斯模型 $p(y|x)$ 均值是 x 的线性函数，协方差与 x 独立，也会经常用到。

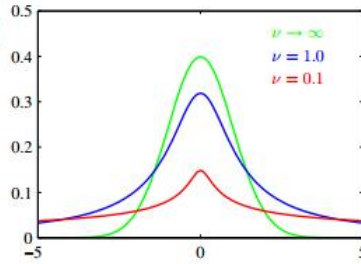
接下来是关于高斯分布的贝叶斯方法，方差已知均值未知，先验用高斯分布；均值已知方差未知用 Gamma 分布；都不知道用 Gaussian-Gamma distribution。这方面的推导略过，大家用到时翻书查看就行了：

2. Conjugate prior: lead to posterior distribution having the same functional form as the prior

Distribution	Conjugate Prior
Bernoulli	Beta distribution
Multinomial	Dirichlet distribution
Gaussian, Given variance, mean unknown	Gaussian distribution
Gaussian, Given mean, variance unknown	Gamma distribution
Gaussian, both mean and variance unknown	Gaussian-Gamma distribution

接下来看 Student t-distribution , Student 是笔名 , 此人在数理统计史上是非常 nb 的人物。

Figure 2.15 Plot of Student's t-distribution (2.159) for $\mu = 0$ and $\lambda = 1$ for various values of ν . The limit $\nu \rightarrow \infty$ corresponds to a Gaussian distribution with mean μ and precision λ .



$$\begin{aligned} p(x|\mu, a, b) &= \int_0^\infty \mathcal{N}(x|\mu, \tau^{-1}) \text{Gam}(\tau|a, b) d\tau \\ &= \int_0^\infty \frac{b^a e^{(-b\tau)} \tau^{a-1}}{\Gamma(a)} \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\left\{-\frac{\tau}{2}(x-\mu)^2\right\} d\tau \\ &= \frac{b^a}{\Gamma(a)} \left(\frac{1}{2\pi}\right)^{1/2} \left[b + \frac{(x-\mu)^2}{2}\right]^{-a-1/2} \Gamma(a+1/2) \end{aligned} \quad (2.158)$$

where we have made the change of variable $z = \tau[b + (x-\mu)^2/2]$. By convention we define new parameters given by $\nu = 2a$ and $\lambda = a/b$, in terms of which the distribution $p(x|\mu, a, b)$ takes the form

$$\text{St}(x|\mu, \lambda, \nu) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)} \left(\frac{\lambda}{\pi\nu}\right)^{1/2} \left[1 + \frac{\lambda(x-\mu)^2}{\nu}\right]^{-\nu/2-1/2} \quad (2.159)$$

上面是 t 分布的形式 , 具体如何发现的可以参看《数理统计简史》 , 大家看上面的积分形式 , t 分布其实是无限个均值一样 , 方差不同的高斯分布混合而成 , 高斯分布是它的特例 , 相比高斯分布 , t 分布对 outliers 干扰的鲁棒性要强很多。

从这个图就可以看出 , 高斯分布对右边孤立点的干扰很敏感 , t 分布基本上没有变化 :

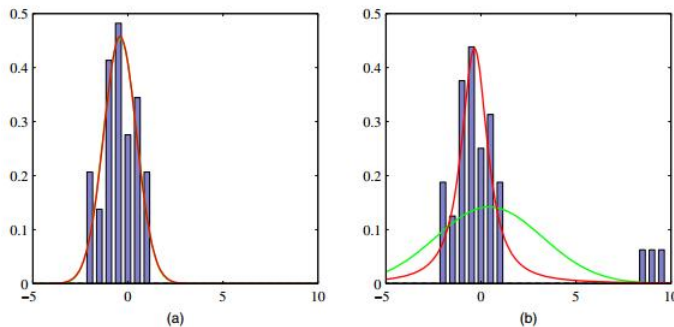
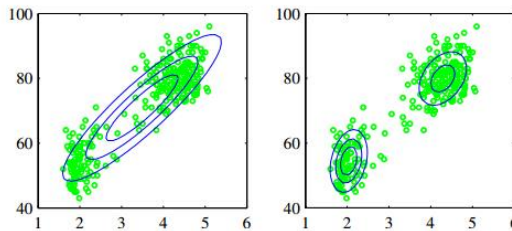


Figure 2.16 Illustration of the robustness of Student's t-distribution compared to a Gaussian. (a) Histogram distribution of 30 data points drawn from a Gaussian distribution, together with the maximum likelihood fit obtained from a t-distribution (red curve) and a Gaussian (green curve, largely hidden by the red curve). Because the t-distribution contains the Gaussian as a special case it gives almost the same solution as the Gaussian. (b) The same data set but with three additional outlying data points showing how the Gaussian (green curve) is strongly distorted by the outliers, whereas the t-distribution (red curve) is relatively unaffected.

接着讲混合高斯分布 : 看下图里的例子 , 单个高斯分布表达能力有限 , 无法捕捉到两个簇结构 :

Figure 2.21 Plots of the 'old faithful' data in which the blue curves show contours of constant probability density. On the left is a single Gaussian distribution which has been fitted to the data using maximum likelihood. Note that this distribution fails to capture the two clumps in the data and indeed places much of its probability mass in the central region between the clumps where the data are relatively sparse. On the right the distribution is given by a linear combination of two Gaussians which has been fitted to the data by maximum likelihood using techniques discussed Chapter 9, and which gives a better representation of the data.



我们可以多个高斯分布的线性组合来逼近复杂的分布，并且对非指数族的分布也一样有效。

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

混合高斯分布的形式：

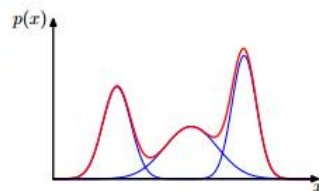
RIVERS(773600590) 11:01:09

可不可以使用非线性的组合呢？

网络上的尼采(813394698) 11:01:48

那就太复杂了

e 2.22 Example of a Gaussian mixture distribution in one dimension showing three Gaussians (each scaled by a coefficient) in blue and their sum in red.



这个图是三个高斯分布混合逼近一个复杂分布的例子。

混合高斯模型里面有一个隐变量，也就是数据点属于哪个高斯分布。

这个就是隐变量的期望：

$$\begin{aligned} \gamma_k(\mathbf{x}) &\equiv p(k|\mathbf{x}) \\ &= \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)}{\sum_l \pi_l \mathcal{N}(\mathbf{x} | \mu_l, \Sigma_l)} \end{aligned}$$

这个是我们的最大似然目标函数：

$$\ln p(\mathbf{X} | \pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k) \right\}$$

可以用 EM 算法，一边是隐变量，一边是模型的参数，迭代着来回倒腾，收敛到局部最优。混合高斯我在第九章详细讲了，感兴趣的可以看下原来的记录。

xunyu(2118773) 11:09:18

隐变量和最大似然函数的联系在哪里

落英缤纷(348609341) 11:10:16

不设置隐变量直接用 ML 不好解

网络上的尼采(813394698) 11:10:31

下面讲指数族，很多分布包括我们上面提到的二项式分布、beta 分布、多项式分布、狄利克雷分布、高斯分布都可以转换成这种指数族的形式：

The exponential family of distributions over $\mathbf{x}$, given parameters η , is defined to be the set of distributions of the form

$$p(\mathbf{x} | \eta) = h(\mathbf{x}) g(\eta) \exp \{ \eta^T \mathbf{u}(\mathbf{x}) \} \quad (2.194)$$

其中 η 是参数， $g(\eta)$ 是归一化因子， $u(x)$ 是 x 的函数。

... η_n), the natural sufficient statistic is given by

$$p(\mathbf{X}|\boldsymbol{\eta}) = \left(\prod_{n=1}^N h(\mathbf{x}_n) \right) g(\boldsymbol{\eta})^N \exp \left\{ \boldsymbol{\eta}^T \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n) \right\}.$$

指数族的似然函数：

$$-\nabla \ln g(\boldsymbol{\eta}_{\text{ML}}) = \frac{1}{N} \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n)$$

对 $\ln p(\mathbf{X}|\boldsymbol{\eta})$ 关于 $\boldsymbol{\eta}$ 求导，令其等于 0，会得到最大似然解的形式：

$$\sum_{n=1}^N \mathbf{u}(\mathbf{x}_n)$$

很显然，是充分统计量。充分统计量其实很好理解，拿最简单的二项式分布来说，抛硬币我们只

需要记住正反面出现的次数就行，原来的数据就可以丢弃了。

DUDA 是指数族专家，这是从他书上截的图，大家可以看下表中的指数族：

Table 3.1: Common Exponential Distributions and their Sufficient Statistics.

Name	Distribution	Domain		s	$[g(s, \boldsymbol{\theta})]^{1/n}$
Normal	$p(x \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \boldsymbol{\theta}_2 (x - \boldsymbol{\theta}_1)^2}$	$\boldsymbol{\theta}_2 > 0$		$\frac{1}{n} \sum_{k=1}^n x_k$ $\frac{1}{n} \sum_{k=1}^n x_k^2$	$\sqrt{\boldsymbol{\theta}_2} e^{-\frac{1}{2} \boldsymbol{\theta}_2 (s_2 - 2\boldsymbol{\theta}_1 s_1 + \boldsymbol{\theta}_1^2)}$
Multi-variate Normal	$p(\mathbf{x} \boldsymbol{\theta}) = \frac{ \boldsymbol{\Theta}_2 ^{1/2}}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\theta}_1)^T \boldsymbol{\Theta}_2 (\mathbf{x} - \boldsymbol{\theta}_1)}$	$\boldsymbol{\Theta}_2$ positive definite		$\frac{1}{n} \sum_{k=1}^n \mathbf{x}_k$ $\frac{1}{n} \sum_{k=1}^n \mathbf{x}_k \mathbf{x}_k^T$	$ \boldsymbol{\Theta}_2 ^{1/2} e^{-\frac{1}{2} [\text{tr} \boldsymbol{\Theta}_2 s_2 - 2\boldsymbol{\theta}_1^T \boldsymbol{\Theta}_2 s_1 + \boldsymbol{\theta}_1^T \boldsymbol{\Theta}_2 \boldsymbol{\theta}_1]}$
Exponential	$p(x \theta) = \begin{cases} \theta e^{-\theta x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$	$\theta > 0$		$\frac{1}{n} \sum_{k=1}^n x_k$	$\theta e^{-\theta s}$
Rayleigh	$p(x \theta) = \begin{cases} 2\theta x e^{-\theta x^2} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$	$\theta > 0$		$\frac{1}{n} \sum_{k=1}^n x_k^2$	$\theta e^{-\theta s}$
Maxwell	$p(x \theta) = \begin{cases} \frac{4}{\sqrt{\pi}} \theta^{3/2} x^2 e^{-\theta x^2} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$	$\theta > 0$		$\frac{1}{n} \sum_{k=1}^n x_k^2$	$\theta^{3/2} e^{-\theta s}$
Gamma	$p(x \boldsymbol{\theta}) = \begin{cases} \frac{\theta^{\theta_1+1}}{\Gamma(\theta_1+1)} x^{\theta_1} e^{-\theta_2 x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$	$\theta_1 > -1$ $\theta_2 > 0$		$\left[\left(\prod_{k=1}^n x_k \right)^{1/n} \right]$ $\frac{1}{n} \sum_{k=1}^n x_k$	$\frac{\theta^{\theta_1+1}}{\Gamma(\theta_1+1)} s_1^{\theta_1} e^{-\theta_2 s_2}$
Beta	$p(x \boldsymbol{\theta}) = \begin{cases} \frac{\Gamma(\theta_1+\theta_2+2)}{\Gamma(\theta_1+1)\Gamma(\theta_2+1)} x^{\theta_1} (1-x)^{\theta_2} & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$	$\theta_1 > -1$ $\theta_2 > -1$		$\left(\prod_{k=1}^n x_k \right)^{1/n}$ $\left(\prod_{k=1}^n (1-x_k) \right)^{1/n}$	$\frac{\Gamma(\theta_1+\theta_2+2)}{\Gamma(\theta_1+1)\Gamma(\theta_2+1)} s_1^{\theta_1} s_2^{\theta_2}$
Poisson	$P(x \theta) = \frac{\theta^x}{x!} e^{-\theta} \quad x = 0, 1, 2, \dots$	$\theta > 0$		$\frac{1}{n} \sum_{k=1}^n x_k$	$\theta^s e^{-\theta}$
Bernoulli	$P(x \theta) = \theta^x (1-\theta)^{1-x} \quad x = 0, 1$	$0 < \theta < 1$		$\frac{1}{n} \sum_{k=1}^n x_k$	$\theta^s (1-\theta)^{1-s}$
Binomial	$P(x \theta) = \frac{m!}{x!(m-x)!} \theta^x (1-\theta)^{m-x} \quad x = 0, 1, \dots, m$	$0 < \theta < 1$		$\frac{1}{n} \sum_{k=1}^n x_k$	$\theta^s (1-\theta)^{m-s}$
Multinomial	$P(\mathbf{x} \boldsymbol{\theta}) = \frac{m!}{\prod_{i=1}^d x_i!} \prod_{i=1}^d \theta_i^{x_i} \quad \begin{matrix} x_i = 0, 1, \dots, m \\ \sum_{i=1}^d x_i = m \end{matrix}$	$0 < \theta_i < 1$ $\sum_{i=1}^d \theta_i = 1$		$\frac{1}{n} \sum_{k=1}^n \mathbf{x}_k$	$\prod_{i=1}^d \theta_i^{s_i}$

$$p(\boldsymbol{\eta}|\boldsymbol{\chi}, \nu) = f(\boldsymbol{\chi}, \nu) g(\boldsymbol{\eta})^\nu \exp \{ \nu \boldsymbol{\eta}^T \boldsymbol{\chi} \}$$

指数族的共轭先验形式：

$$p(\boldsymbol{\eta}|\mathbf{X}, \boldsymbol{\chi}, \nu) \propto g(\boldsymbol{\eta})^{\nu+N} \exp \left\{ \boldsymbol{\eta}^T \left(\sum_{n=1}^N \mathbf{u}(\mathbf{x}_n) + \nu \boldsymbol{\chi} \right) \right\}.$$

后验形式：

第三章 Linear Models for Regression

主讲人 planktonli

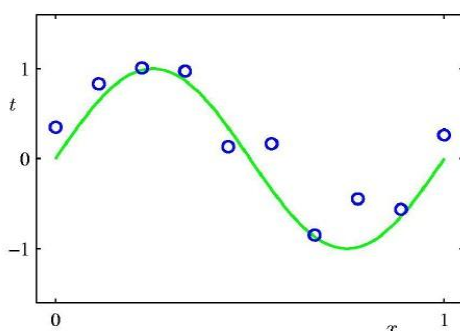
planktonli(1027753147) 18:58:12

大家好，我负责给大家讲讲 PRML 的第 3 讲 linear regression 的内容，请大家多多指教，群主让我们每个主讲人介绍下自己，赫赫，我也说两句，我是 applied mathematics + computer science 的，有问题大家可以直接指出，互相学习。大家有兴趣的话可以看看我的博客: <http://t.qq.com/keepuphero/mine>，当然我给大家推荐一个好朋友的，他对计算机发展还是很有心得的，他的网页 <http://www.zhizhihu.com/> 对 machine learning 的东西有深刻的了解。

好，下面言归正传，开讲第 3 章，第 3 章的名字是 linear regression，首先需要考虑的是：为什么在讲完 introduction、probability distributions 之后就直讲 linear regression? machine learning 的 essence 是什么？

机器学习的本质问题：我个人理解，就是通过数据集学习未知的最佳逼近函数，学习的收敛性、界等等都是描述这个学习到的 function 到底它的性能如何。但是，从数学角度出发，函数是多样的，线性、非线性、跳跃、连续、非光滑，你可以组合出无数的函数，那么这些函数就组成了函数空间，在这些函数中寻找到一个满足你要求的最佳逼近函数，无疑大海捞针。我们再来回顾下第一章的 曲线拟和问题：

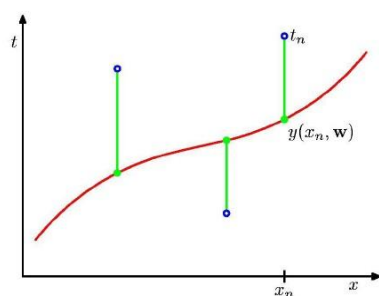
Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

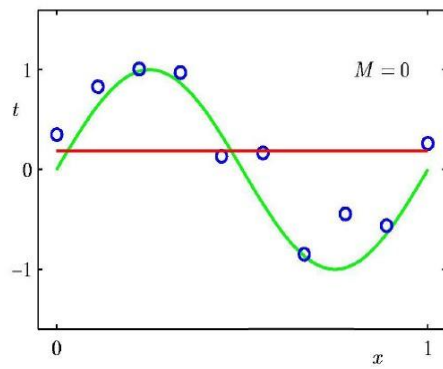
需要逼近的函数是: $\sin(2\pi x)$ ，M 阶的曲线函数可以逼近么？这是我们值得思考的问题。

Sum-of-Squares Error Function

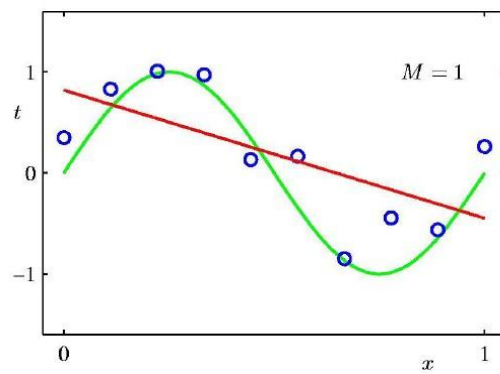


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

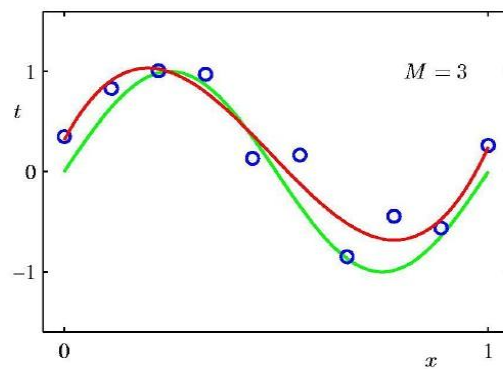
0th Order Polynomial



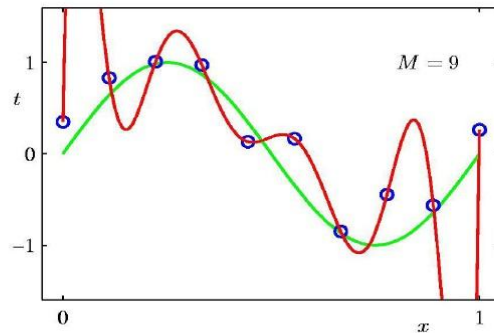
1st Order Polynomial



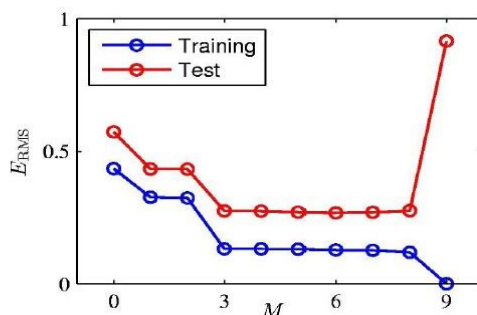
3rd Order Polynomial



9th Order Polynomial



Over-fitting



Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

要曲线拟和, 那么拟和的标准是什么?这里用了 2 范数定义,也就是误差的欧式距离, 当然,你可以用 L1,L 无穷, 等等了, 只是 objective 不同罢了。现在的疑问是: 为什么要用 Polynomial Fitting?有数学依据么, 这里牵扯到 范函的问题, 就是函数所张成的空间, 举一个简单的例子, 大家还都记得 taylor 展式吧:

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x - x_0) + \frac{f''(x)}{2!}(x - x_0)^2 + \dots$$

这表明 任意一个函数可以表示成 x 的次方之和, 也就是 任意一个函数 可以放到 $(1, x, x^2, x^3, \dots)$ 所张成的函数空间, 如果是有限个基的话就称为欧式空间, 无穷的话 就是 Hilbert 空间, 其实 傅里叶变换 也是这样的一个例子, 既然已经明白了 任意函数可以用 Polynomial Fitting, 那么下面就是什么样的 Polynomial 是最好的。

Wilbur_中博(1954123) 19:28:26

泰勒展开是局部的、 x_0 周围的, 而函数拟合是全局的, 似乎不太一样吧?

planktonli(1027753147) 19:29:21

恩,泰勒展开是局部的, 他是在 x_0 点周围的一个 表达, 函数拟合是全局的,我这里只是用一个简单的例子说明 函数表达的问题。

Wilbur_中博(1954123) 19:30:41



planktonli(1027753147) 19:31:03

其实,要真正解释这个问题是需要范函的东西的。

Wilbur_中博(1954123) 19:31:45

抱歉，打断了一下，因为我觉得这个问题留到讨论就不太好了，呵呵。了解了，请继续吧。

planktonli(1027753147) 19:31:51

由于大多数群友未学过这个课程,我只是想说下这个思想,呵呵,没事,讨论才能深刻理解问题,其实,wavelet 这些,包括 kernel construcion 这些东西都牵扯到 范函。

Bishop 用上面这个例子说明：

1) 可以用 Polynomial Fitting 拟和 sin 类的函数 2) 存在过拟和问题

而且这里的 Polynomial Fitting 是一个线性 model，这里 Model 是 w 的函数, w 是线性的：

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

$\sin(2\pi x)$ 是线性的么，肯定不是，那么 让我们再分析下 研究的问题

$\sin(2\pi x)$ 中的 x 是 1 维的

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

上面的 x 变成了 $[1, x, x^2, x^3, \dots]^T$

$y(x, w) = W^T X$ ，非常有意思的是：维数升高了，同时这个 model 具有了表达非线性东西的能力。这

里的思想,可以说贯穿在 NN,SVM 这些东西里，也就是说,线性的 model 如果应用得当的话,可以表达非线性的东西。与其在所有函数空间盲目的寻找,还不如从一个可行的简单 model 开始，这就是为什么 Bishop 在讲完基础后直接切入 Linear regression 的原因,当然这个线性 model 怎么构造,是单层的 linear model,还是多层的 linear model 一直争论不休，BP 否定了 perceptron 的 model，SVM 否定了 BP model 现在 deep learning 又质疑 SVM 的 shallow model，或许这就是 machine learning 还能前进的动力。

让咱们再回来看看 linear regression 的模型，这里从标准形式到扩展形式，也就是引入基函数后,Linear regression 的模型可以表达非线性的东西了，因为基函数可能是非线性的：

Linear Regression

The basic form of linear regression

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1x_1 + \dots + w_Dx_D$$

where $\mathbf{x} = (x_1, \dots, x_D)^T$ is the input variable, and $\mathbf{w} = (w_1, \dots, w_D)^T$ is the parameters.

More general form of linear regression

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x})$$

where $\phi_j(\mathbf{x})$'s are known as *basis functions*.

Parameter w_0 is called a *bias* parameter. By adding $\phi_0(\mathbf{x}) = 1$, we have

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

基函数的形式，这些基函数都是非线性的：

Basis functions

Polynomial functions

$$\phi_j(x) = x^j$$

Gaussian functions

$$\phi_j(\mathbf{x}) = \exp \left\{ -\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j)}{2s^2} \right\}$$

Sigmoid basis functions

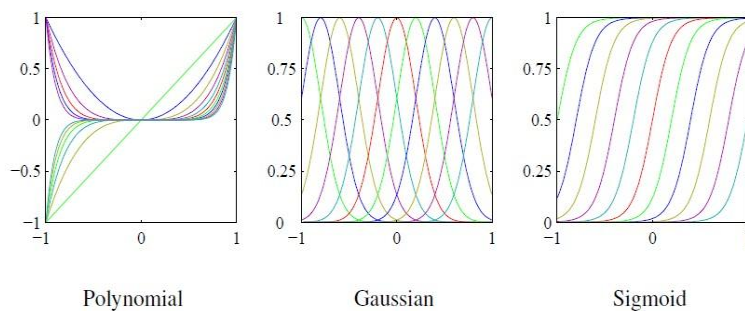
$$\phi_j(\mathbf{x}) = \sigma \left(\frac{\mathbf{b}^T \mathbf{x} - \mu_j}{s} \right)$$

where $\sigma(a)$ is the logistic sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp \{-a\}}$$

Other basis functions: wavelets

Basis functions



Probabilistic Formulation

Assume the target variable t is given by a deterministic function $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

where ϵ is a zero mean Gaussian random variable with precision (inverse variance) β .
Therefore

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

Conditional mean

$$\mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt = y(\mathbf{x}, \mathbf{w}) + \mathbb{E}[\epsilon] = y(\mathbf{x}, \mathbf{w})$$

在 Gaussian 零均值情况下, Linear model 从频率主义出发的 MLE 就是 Least square :

Maximum Likelihood Estimator

Consider a data set of inputs $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = (t_1, \dots, t_N)^T$. The likelihood function (a function of $\mathbf{w}$ and β) is

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

Log likelihood

$$\begin{aligned} \log p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^N \log \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \beta E_D(\mathbf{w}) \end{aligned}$$

where the sum-of-square error function is

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2$$

最小 2 乘的解就是广义逆矩阵乘输出值 :

Maximum Likelihood Estimator: $\mathbf{w}$

With β fixed, we obtain the gradient of the log likelihood

$$\nabla \log p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n)) \phi(\mathbf{x}_n)^T$$

Setting the gradient to be zero gives

$$\begin{aligned} \mathbf{w}^T \left(\sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right) &= \sum_{n=1}^N t_n \phi(\mathbf{x}_n)^T \\ \mathbf{w}^T \Phi^T \Phi &= \mathbf{t}^T \Phi \\ \Phi^T \Phi \mathbf{w} &= \Phi^T \mathbf{t} \\ \mathbf{w}_{\text{ML}} &= (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t} \end{aligned}$$

where the $N \times M$ design matrix Φ is $\Phi = [\phi(\mathbf{x}_1) \dots \phi(\mathbf{x}_N)]^T$. The quantity $\Phi^\dagger \equiv (\Phi^T \Phi)^{-1} \Phi^T$ is called *pseudo-inverse* of the matrix Φ .

Gaussian 的 precision 也可以计算出来 :

Maximum Likelihood Estimator: β

Recall log likelihood

$$\log p(\mathbf{t}|\mathbf{w}, \beta) = \frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \beta E_D(\mathbf{w})$$

Setting the partial derivative w.r.t. β to zero

$$\frac{\partial}{\partial \beta} \log p(\mathbf{t}|\mathbf{w}, \beta) = \frac{N}{2} \frac{1}{\beta} - E_D(\mathbf{w})$$

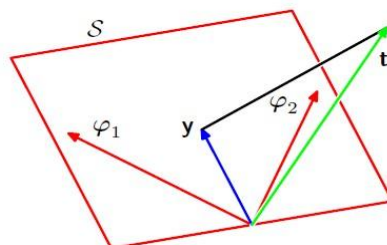
We obtain

$$\beta_{\text{ML}} = \frac{N}{(\mathbf{t} - \Phi \mathbf{w}_{\text{ML}})^T (\mathbf{t} - \Phi \mathbf{w}_{\text{ML}})}$$

最小 2 乘的解可以看成到基张成空间的投影：

Geometrical Interpretation of Least Squares

The least-squares regression function is obtained by finding the orthogonal projection of the data vector $\mathbf{t}$ onto the subspace spanned by the basis functions $\phi_j(\mathbf{x})$ in which each basis function is viewed as a vector φ_j of length $|N|$ with elements $\phi_j(\mathbf{x}_n)$.



频率主义会导致 过拟和，加入正则,得到的最小 2 乘解：

Regularized Least Squares

The over-fitting issue can be addressed through adding a *regularization* term

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

where λ is the regularization coefficient. For example

$$E_W(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

and the error function becomes

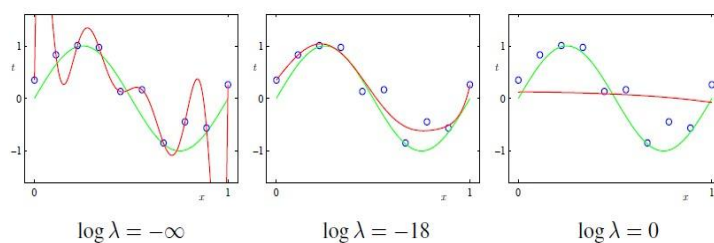
$$\frac{1}{2} (\mathbf{t} - \Phi \mathbf{w})^T (\mathbf{t} - \Phi \mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

which leads to solution

$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

正则参数对 model 结果的影响：

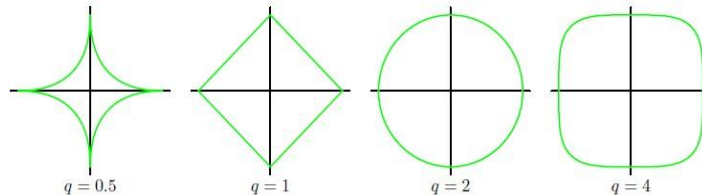
Effects of λ



消除过拟和，正则的几何解释：

Other Regularizers

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^M |w_j|^q$$



When $q = 1$, the regularizer is called *lasso* in statistics as sparse models are preferred.

正则方法不同,就会出现很多 model,例如 lasso, ridge regression。LASSO 的解是稀疏的,例如:sparse coding,Compressed sensing 是从 L0--> L1sparse 的问题,现在也很热的。

Regularized Least Squares and MAP



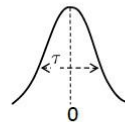
What if $(\mathbf{A}^T \mathbf{A})$ is not invertible ?

$$\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^n | \beta, \sigma^2)}_{\text{log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

I) Gaussian Prior

$$\beta \sim \mathcal{N}(0, \tau^2 \mathbf{I})$$

$$p(\beta) \propto e^{-\beta^T \beta / 2\tau^2}$$



$$\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \|\beta\|_2^2 \quad \text{Ridge Regression}$$

Closed form: HW constant(σ^2, τ^2)

Regularized Least Squares and MAP



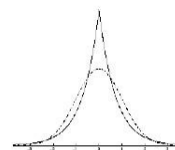
What if $(\mathbf{A}^T \mathbf{A})$ is not invertible ?

$$\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^n | \beta, \sigma^2)}_{\text{log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

II) Laplace Prior

$$\beta_i \stackrel{iid}{\sim} \text{Laplace}(0, t)$$

$$p(\beta_i) \propto e^{-|\beta_i|/t}$$



$$\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \|\beta\|_1 \quad \text{Lasso}$$

Closed form: HW constant(σ^2, t)

Ridge Regression vs Lasso

$$\min_{\beta} (\mathbf{A}\beta - \mathbf{Y})^T (\mathbf{A}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

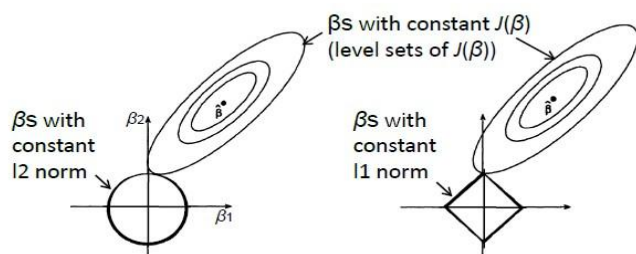
Ridge Regression:

$$\text{pen}(\beta) = \|\beta\|_2^2$$

Lasso:

$$\text{pen}(\beta) = \|\beta\|_1$$

HOT



Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

下面看 Bias-Variance Decoposition , 正则就是在 训练数据的模型上加一个惩罚项, shrink 模型的参数,

让它不要学习的太过, 这里 $E_D(\mathbf{w})$ 是对训练数据学习到的模型, $E_W(\mathbf{w})$ 是学习到的参数的惩罚模型

Expected Squared Loss

Recall that t is the value for each input $\mathbf{x}$ and $y(\mathbf{x})$ is our estimate. We incur a loss $L(t, y(\mathbf{x}))$, and the expected loss is

$$\mathbb{E}[L] = \int \int L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt.$$

A common choice of loss function is the squared loss $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$, and the expected squared loss is

$$\mathbb{E}[L] = \int \int \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

Our goal is to choose $y(\mathbf{x})$ to minimize $\mathbb{E}[L]$. From Euler-Lagrange

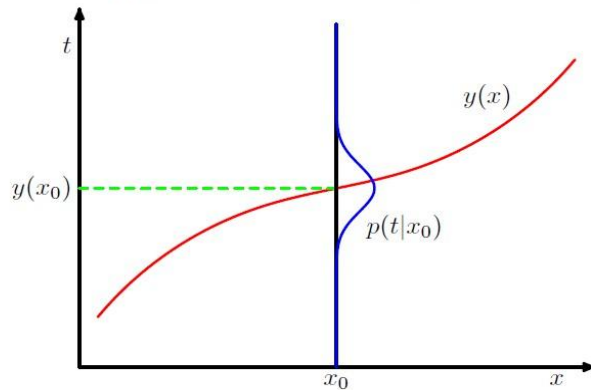
$$\frac{\delta \mathbb{E}[L]}{\delta y(\mathbf{x})} = 2 \int \{y(\mathbf{x}) - t\} p(\mathbf{x}, t) dt = 0$$

So

$$y(\mathbf{x}) = \frac{\int t p(\mathbf{x}, t) dt}{p(\mathbf{x})} = \int t p(t|\mathbf{x}) dt = \mathbb{E}_t[t|\mathbf{x}]$$

Regression Function

The function $y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$ is known as the *regression function*.



Decomposition of the Loss Function

Define

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt$$

Since the optimal solution is the conditional expectation, we can expand the square term

$$\begin{aligned} \{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - h(\mathbf{x})\}^2 + 2\{y(\mathbf{x}) - h(\mathbf{x})\}\{h(\mathbf{x}) - t\} + \{h(\mathbf{x}) - t\}^2 \end{aligned}$$

Therefore

$$\begin{aligned} \mathbb{E}[L] &= \int \int \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \\ &= \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \end{aligned}$$

When $y(\mathbf{x}) = h(\mathbf{x})$, the first term will vanish. The second term is the variance of the distribution of t averaged over $\mathbf{x}$, representing the intrinsic variability of the target data and can be regarded as noise and the irreducible minimum value of the loss function.

Decomposition of the Loss Function

In practice, $h(\mathbf{x})$ is unknown. We have a data set $\mathcal{D}$ containing only a finite number N of data points. The loss function becomes

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2$$

We can rewrite it as

$$\begin{aligned} \{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\} \end{aligned}$$

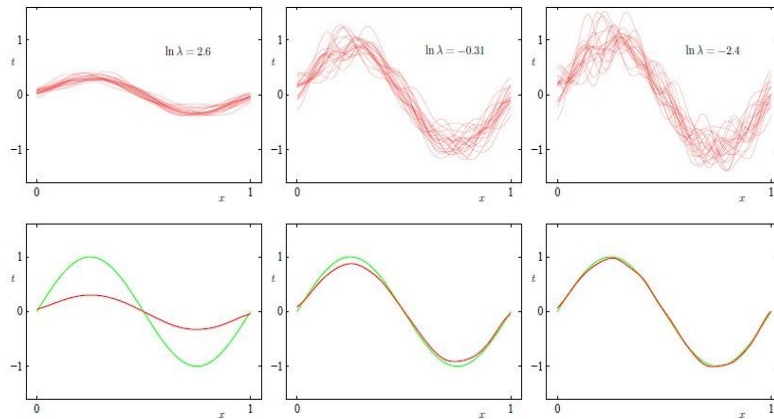
Take the expectation w.r.t. $\mathcal{D}$

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}} \end{aligned}$$

上面这么多 PPT 无非就是说，学习到的模型和真实的模型的期望由 2 部分组成：

1--> Bias 2--> Variance。Bias 表示的是学习到的模型和真实模型的偏离程度,Variance 表示的是学习到的模型和它自己的期望的偏离程度。从这里可以看到正则项在控制 Bias 和 Variance :

Bias-Variance as a Function of Complexity



Wilbur_中博(1954123) 20:33:07

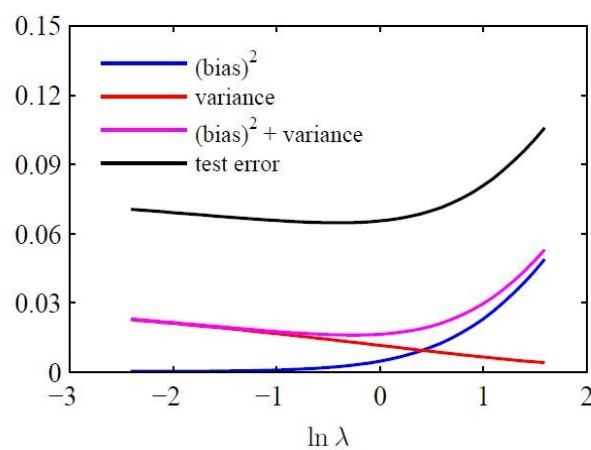
这个是关键，呵呵

planktonli(1027753147) 20:33:25

Variance 小的情况下,Bias 就大 , Variance 大的情况下,Bias 就小 , 我们就要 tradeoff 它们。

从这张图可以看到 Bias 和 Variance 的关系 :

Bias-Variance as a Function of Complexity



这个 Bias-Variance Decoposition 其实没有太大的实用价值，它只能起一个指导作用。

下面看看 Bayesian Linear Regression :

Conjugate Prior

Recall the likelihood function

$$\begin{aligned} p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \mathcal{N}(\mathbf{t} | \Phi \mathbf{w}, \beta^{-1} \mathbf{I}) \end{aligned}$$

is the exponential of a quadratic function of $\mathbf{w}$. So the conjugate prior is given by a Gaussian distribution of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_0, \mathbf{S}_0)$$

Bayesian Theorem for Gaussian

Theorem (Bayesian Theorem for Linear Gaussian)

Given prior and likelihood

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}(\mathbf{y} | \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \end{aligned}$$

the marginal $p(\mathbf{y})$ and posterior $p(\mathbf{x}|\mathbf{y})$ are

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y} | \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T + \mathbf{L}^{-1}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\Sigma}[\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}], \boldsymbol{\Sigma}) \end{aligned}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1}.$$

Posterior

Apply the theorem and we obtain the posterior

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_N, \mathbf{S}_N)$$

where

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t}) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\Phi^T\Phi \end{aligned}$$

The prior can be a zero-mean isotropic Gaussian governed by a single precision parameter α

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1}\mathbf{I})$$

and the corresponding posterior distribution is

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\Phi^T\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\Phi^T\Phi \end{aligned}$$

Bayesian MAP vs. Regularization

The log posterior takes the following form

$$\log p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}(\mathbf{t} - \Phi\mathbf{w})^T(\mathbf{t} - \Phi\mathbf{w}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const}$$

The regularized least square takes the form

$$\frac{1}{2}(\mathbf{t} - \Phi\mathbf{w})^T(\mathbf{t} - \Phi\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Therefore, $\lambda = \frac{\alpha}{\beta}$.

从 Bayesian 出发,关注的不是参数的获取,而更多的是 新预测的值,通过后验均值可以得到 linear model 和核函数的联系,当然也可以建立 gaussian process 这些东西。

Wilbur_中博(1954123) 20:51:25

这里可以讲细一点么,如何建立联系?

planktonli(1027753147) 20:54:44

Bayesian Linear Regression (2)

A common choice for the prior is

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

for which

$$\begin{aligned}\mathbf{m}_N &= \beta\mathbf{S}_N\Phi^T\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\Phi^T\Phi.\end{aligned}$$

Equivalent Kernel (1)

The predictive mean can be written

$$\begin{aligned}y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^T\phi(\mathbf{x}) = \beta\phi(\mathbf{x})^T\mathbf{S}_N\Phi^T\mathbf{t} \\ &= \sum_{n=1}^N \underbrace{\beta\phi(\mathbf{x})^T\mathbf{S}_N\phi(\mathbf{x}_n)}_{k(\mathbf{x}, \mathbf{x}_n)} t_n \\ &= \sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) t_n.\end{aligned}$$

Equivalent kernel or smoother matrix.

This is a weighted sum of the training data target values, t_n .

这里就可以看到了啊,看到了么, Wilbur?

Wilbur_中博(1954123) 20:57:24

👉 在看

planktonli(1027753147) 20:58:08

如果共扼先验是 0 均值情况下,linear model 就可以变成 kernel 了:

Equivalent Kernel (4)

The kernel as a covariance function: consider

$$\begin{aligned}\text{cov}[y(\mathbf{x}), y(\mathbf{x}')] &= \text{cov}[\phi(\mathbf{x})^T \mathbf{w}, \mathbf{w}^T \phi(\mathbf{x}')] \\ &= \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}') = \beta^{-1} k(\mathbf{x}, \mathbf{x}').\end{aligned}$$

We can avoid the use of basis functions and define the kernel function directly, leading to *Gaussian Processes* (Chapter 6).

Equivalent Kernel (5)

$$\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) = 1$$

for all values of $\mathbf{x}$; however, the equivalent kernel may be negative for some values of $\mathbf{x}$.

Like all kernel functions, the equivalent kernel can be expressed as an inner product:

$$k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x})^T \psi(\mathbf{z})$$

where $\psi(\mathbf{x}) = \beta^{1/2} \mathbf{S}_N^{1/2} \phi(\mathbf{x})$.

最后讲了 bayesian model 比较 :

Bayesian Model Comparison (1)

How do we choose the 'right' model?

Assume we want to compare models $\mathcal{M}_i$, $i=1, \dots, L$, using data $\mathcal{D}$; this requires computing

$$p(\mathcal{M}_i | \mathcal{D}) \propto p(\mathcal{M}_i) p(\mathcal{D} | \mathcal{M}_i).$$

Posterior	Prior	Model evidence or marginal likelihood
-----------	-------	--

Bayes Factor: ratio of evidence for two models

$$\frac{p(\mathcal{D} | \mathcal{M}_i)}{p(\mathcal{D} | \mathcal{M}_j)}$$

Bayesian Model Comparison (2)

Having computed $p(\mathcal{M}_i|\mathcal{D})$, we can compute the predictive (mixture) distribution

$$p(t|\mathbf{x}, \mathcal{D}) = \sum_{i=1}^L p(t|\mathbf{x}, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D}).$$

A simpler approximation, known as *model selection*, is to use the model with the highest evidence.

Bayesian Model Comparison (3)

For a model with parameters $\mathbf{w}$, we get the model evidence by marginalizing over $\mathbf{w}$

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i) p(\mathbf{w}|\mathcal{M}_i) d\mathbf{w}.$$

Note that

$$p(\mathbf{w}|\mathcal{D}, \mathcal{M}_i) = \frac{p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i) p(\mathbf{w}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_i)}$$

选择最大信任的 model 来作为模型选择，而非用交叉验证，信任近似：

The Evidence Approximation (1)

The fully Bayesian predictive distribution is given by

$$p(t|\mathbf{t}) = \iiint p(t|\mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta) p(\alpha, \beta|\mathbf{t}) d\mathbf{w} d\alpha d\beta$$

but this integral is intractable. Approximate with

$$p(t|\mathbf{t}) \simeq p(t|\mathbf{t}, \hat{\alpha}, \hat{\beta}) = \int p(t|\mathbf{w}, \hat{\beta}) p(\mathbf{w}|\mathbf{t}, \hat{\alpha}, \hat{\beta}) d\mathbf{w}$$

where $(\hat{\alpha}, \hat{\beta})$ is the mode of $p(\alpha, \beta|\mathbf{t})$, which is assumed to be sharply peaked; a.k.a. *empirical Bayes*, *type II* or *generalized maximum likelihood*, or *evidence approximation*.

The Evidence Approximation (2)

From Bayes' theorem we have

$$p(\alpha, \beta | \mathbf{t}) \propto p(\mathbf{t} | \alpha, \beta) p(\alpha, \beta)$$

and if we assume $p(\alpha, \beta)$ to be flat we see that

$$\begin{aligned} p(\alpha, \beta | \mathbf{t}) &\propto p(\mathbf{t} | \alpha, \beta) \\ &= \int p(\mathbf{t} | \mathbf{w}, \beta) p(\mathbf{w} | \alpha) d\mathbf{w}. \end{aligned}$$

General results for Gaussian integrals give

$$\ln p(\mathbf{t} | \alpha, \beta) = \frac{M}{2} \ln \alpha + \frac{N}{2} \ln \beta - E(\mathbf{m}_N) + \frac{1}{2} \ln |\mathbf{S}_N| - \frac{N}{2} \ln(2\pi).$$

Limitations of Fixed Basis Functions

- M basis function along each dimension of a D -dimensional input space requires M^D basis functions: the curse of dimensionality.
- In later chapters, we shall see how we can get away with fewer basis functions, by choosing these using the training data.

固定基存在缺陷为 NN,SVM 做铺垫, NN,SVM 都是变化基, BP 是梯度下降 error, 固定基, RBF 是聚类寻找基, SVM 是 2 次凸优化寻找基。好了,就讲到这里吧,肯定还有讲的不对,或者不足的地方,请大家一起讨论和补充,谢谢。

=====讨论=====

Wilbur_中博(1954123) 21:08:29

RBF 不是固定径向基找系数的么, SVM 也是固定基的吧, 这里寻找基是什么意思?

planktonli(1027753147) 21:09:01

SVM 是寻找那些 系数不为 0 的作为基, RBF,我说的是 RBF 神经网络, 不是 RBF 基函数, 呵呵

Wilbur_中博(1954123) 21:11:07

嗯, 但咱们现在这一章, 比如多项式基, 也可以说是寻找系数不为 0 的 x^k 吧, SVM 也仍然是固定了某一种核, 比如多项式核或者高斯核。嗯, 我知道是说 RBF 网络。

planktonli(1027753147) 21:11:40

恩,可以这么说

Wilbur_中博(1954123) 21:12:35

还有就是, 固定一组基的话, 也有很多选择, 有多项式、也有高斯、logistic 等等, 那我们应该怎么选择用什么基去做回归呢? 这一章讲得大多都是有了基以后怎么选择 w , 但怎么选择基这一点有没有什么说法。

planktonli(1027753147) 21:13:37

我说的固定指的是, SVM 不知道基是谁, 而是通过优化获取的。

Wilbur_中博(1954123) 21:13:41

或者小波傅里叶什么的。。好多基

planktonli(1027753147) 21:14:03

3.6. Limitations of Fixed Basis Functions

这里提出了固定基的问题，基的选择要看样本的几何形状，一般都是选择 gaussian，当然也可以一个个测试着弄。

Wilbur_中博(1954123) 21:15:55

SVM 里有个叫 multiple kernel learning 的，感觉像是更广泛的变化基的解决方案。嗯，就是说大多是经验性的是吧，选基这个还是蛮有趣的，我觉得。

planktonli(1027753147) 21:16:45

恩,MK 是多个 kernel 的组合，尝试用多个几何形状的 kernel 去寻找一个更 power 的。

Wilbur_中博(1954123) 21:17:05

嗯，呵呵

planktonli(1027753147) 21:17:16

恩,kernel construction 是 ML 的主要研究内容之一

Wilbur_中博(1954123) 21:18:14

好的，我没什么问题了，谢谢，以后多交流。看其他朋友还有什么问题。

planktonli(1027753147) 21:50:29

本次的讲义有些内容是群共享里的 Linear1.pdf

下次的 linear classification 主要讲的内容在群共享中为 Linear2.pdf

第四章 Linear Models for Classification

主讲人 planktonli

planktonli(1027753147) 19:52:28

现在我们就开始讲第四章，第四章的内容是关于 线性分类模型，主要内容有四点：

- 1) Fisher 准则的分类,以及它和最小二乘分类的关系 (Fisher 分类是最小二乘分类的特例)
- 2) 概率生成模型的分类模型
- 3) 概率判别模型的分类模型
- 4) 全贝叶斯概率的 Laplace 近似

需要注意的是，有三种形式的贝叶斯：

- 1) 全贝叶斯
- 2) 经验贝叶斯
- 3) MAP 贝叶斯

我们大家熟知的是 MAP 贝叶斯

MAP(poor man's Bayesian) :不涉及 marginalization ,仅是一种按后验概率最大化的 point estimate。这里的 MAP(poor man's Bayesian)是属于 点概率估计的。而全贝叶斯可以看作对 test 样本的所有参数集合的加权平均，PRML 说的 Bayesian 主要还是指 Empirical Bayesian：

Fully Bayesian: 需要 marginalize with respect to hyper-parameters as well as parameters。而往往是 analytical intractable 的。

对于 curve fitting 的例子 ($p(w|\alpha) = N(w|0, \alpha^{-1}I)$, $p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1})$)

来说，就是：

$$p(t|t) = \iiint p(t|w, \beta) p(w|\alpha, \beta) p(\alpha, \beta|t) d\alpha d\beta dw$$

Empirical Bayes/type 2 maximum likelihood/evidence approximation: 对 hyper-parameter 采

取这样的策略，即先求出使得 marginal likelihood 最大化的参数 α^* 和 β^* ，然后使

hyper-parameter 取固定的值 α^* 和 β^* ，再对 w 进行 marginalize:

$$p(t|t) \approx p(t|t, \alpha^*, \beta^*) = \int p(t|w, \beta^*) p(w|t, \alpha^*, \beta^*) dw$$

这里的 α^* 和 β^* 为超参。

Curve fitting 为例子：

- 1) MLE，直接对 likelihood function 求最大值，得到参数 w。该方法属于 point estimate。
- 2) MAP (poor man's bayes)，引入 prior probability，对 posterior probability 求最大值，得到 w。MAP 此时相当于在 MLE 的目标函数 (likelihood function) 中加入一个 L2 penalty。该方法仍属于 point estimation。
- 3) fully Bayesian approach 需使用 sum rule 和 product rule(因为 “degree of belief” 的 machinery 和概率相同，因此这两个 rule 对 “degree of belief” 成立)，而要获得 predictive distribution 又需要 marginalize (sum or integrate) over the whole of parameter space w：

$$p(t|x, X, t) = \int p(t|x, w) p(w|X, t) dw$$

其中，x 是待预测的点，X 是观察到的数据集，t 是数据集中每个数据点相应的 label。其实是用参数 w 的

后验概率为权,对 probability 进行一次加权平均 ;因此这个过程需要对 w 进行积分 ,即 marginalization。
 由于 marginalization 通常是非常难求取的 ,所以一般在针对 Graphical Model 的时候就需做
 Laplace approximation、Variation inference、MCMC 采样这些。

所以我们要建立的概念是:Graphical Model 的东西是一个需要 marginalization 的。

下面我们看看本讲的内容 :

首先将上节 LS(Least Square)方法直接用于求分类问题 ,就可以得到 Least squares for classification。

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \mathbf{T} = \widetilde{\mathbf{X}}^\dagger \mathbf{T}$$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} = \mathbf{T}^T (\widetilde{\mathbf{X}}^\dagger)^T \widetilde{\mathbf{x}}.$$

Linear Discriminant Function

Linear discriminant function for a vector $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

where $\mathbf{w}$ is called weight vector, and w_0 is a bias.

The classification function is

$$C(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + w_0)$$

where step function $\text{sign}(\cdot)$ is defined as

$$\text{sign}(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

一般线性模型 Generalized Linear Model: an activation function acting on a linear function of the feature variables :

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}) + w_0)$$

称为 Generalized Linear Model (GLM), 其中 $\mathbf{x}$ 为输入数据向量, $\phi(\mathbf{x})$ 为 $\mathbf{x}$ 的 feature vector,

而 f 是一个函数, 称为 activation function, f 的反函数称为 link function。

(1) 当 f 是 nonlinear function 的时候, GLM 是一个 classification model;

(2) 当 f 是 identity function 的时候, GLM 是一个 regression model。

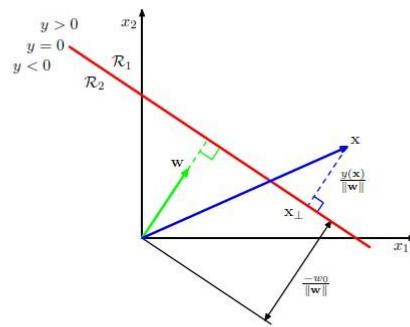
Linear Model 对于回归和分类的区别在于 : 激活函数的不同

$$\text{sign}(\mathbf{w}^T \mathbf{x} + w_0)$$

这里 sign 就是一个非线性的函数 , 其实是一个间断函数,非连续的。

下图证明了点到平面的距离公式。超平面 : 在一个 D 维 Euclidean space 中的超平面是一它的一个 D-1 维流形, 而且该空间是一个线性空间。Linearly separable : 分布于 D 维空间中的全部数据点可以用超平面无错地分隔成类。Coding scheme : 1-of-K binary coding scheme , 即如果有 K 个类 , 某数据点属于第 i 个类 , 则表示为一个 K 维向量 , 该向量除了第 i 个分量是 1 , 其余都是 0。

Properties of Linear Discriminant Function



$y(\mathbf{x}) = 0$ for $\mathbf{x}$ on the decision surface. The normal distance from the origin to the decision surface is

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

So w_0 determines the location of the decision surface.

Properties of Linear Discriminant Function

Let

$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

where $\mathbf{x}_\perp$ is the projection $\mathbf{x}$ on the decision surface. Then

$$\begin{aligned} \mathbf{w}^T \mathbf{x} &= \mathbf{w}^T \mathbf{x}_\perp + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \\ \mathbf{w}^T \mathbf{x} + w_0 &= \mathbf{w}^T \mathbf{x}_\perp + w_0 + r \|\mathbf{w}\| \\ y(\mathbf{x}) &= r \|\mathbf{w}\| \\ r &= \frac{y(\mathbf{x})}{\|\mathbf{w}\|} \end{aligned}$$

Simpler notion: define $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\tilde{\mathbf{x}} = (1, \mathbf{x})$ so that

$$y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$$

关于超平面,线性可分的一些概念,在多类情况,可以使用 1 对 1,1 对多分类器的方式,例如:

你要分类 3 类物体: 苹果,西瓜,香蕉

那么 1 对 1 就是建立 6 个分类器

那么 1 对 1 就是建立 3 个分类器

苹果,西瓜

苹果,香蕉

西瓜,香蕉

1 对多分类器就是:

苹果和非苹果

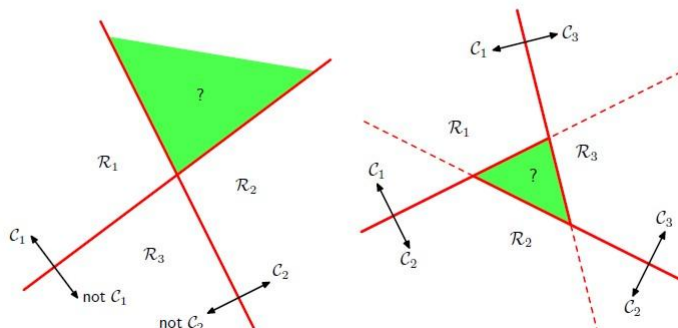
西瓜和非西瓜

香蕉和非香蕉

Multiple Classes: Simple Extension

One-versus-the-rest classifier: classify C_k and samples not in C_k .

One-versus-one classifier: classify every pair of classes.



左边是 1 对多,右边是 1 对 1, 都存在一些无法分类的情况,也就是绿色区域部分。
多分类的决策域是单连通,而且是凸的, 下面给出了证明：

Multiple Classes: K-Class Discriminant

A single K -class discriminant comprising K linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

Decision function

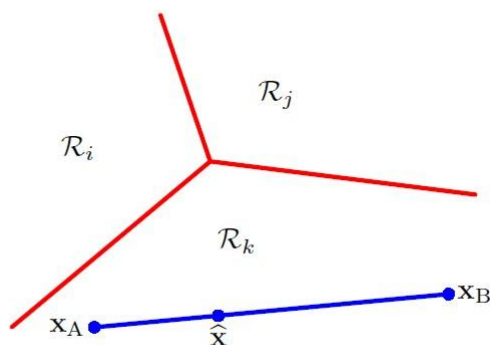
$$C(\mathbf{x}) = k, \text{ if } y_k(\mathbf{x}) > y_j(\mathbf{x}) \forall j \neq k$$

The decision boundary between class C_k and C_j is given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

证明的图形示意：

Property of the Decision Regions



上面有没有问题?没有就继续讲 Fisher's Linear Discriminant 了。

echo<echotooo@gmail.com> 20:26:46

无法分类的情况一般怎么办？就是绿色区域了

planktonli(1027753147) 20:27:35

恩,那就是可能出现判断错误了, 这个没有办法。

echo<echotooo@gmail.com> 20:28:12

哦, 好的, pass。

planktonli(1027753147) 20:28:54

好了,现在看看 Fisher's Linear Discriminant , Linear Discriminant Analysis, LDA),也叫做 Fisher 线性判别, 这个要和 Graphical Model 的 Latent Dirichlet allocation 区分开。我个人认为 LDA 可以看成有一个监督降维的东西, 这些 PCA(主成分分析),ICA(独立成分分析)也是降低维的, 不过是无监督的东西, 包括 manifold dimension reduction 的, 都是无监督的。LDA 是降低到一个投影方向上,使得它的可分性最好而 PCA 是要找它的主要成分也就是使得 Loss 最小的方向, LDA 要求 类间散度最大,类内聚合度最强。

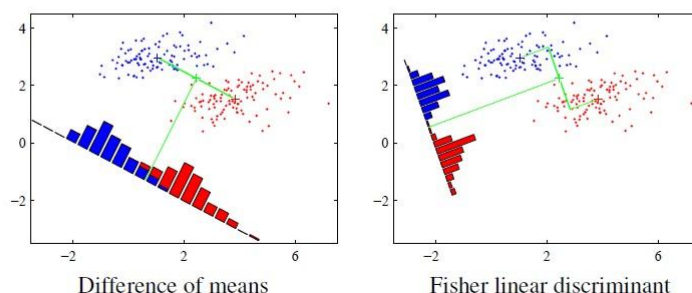
Fisher's Linear Discriminant

Pursue the optimal linear projection on which the two classes can be maximally separated

$$y = \mathbf{w}^T \mathbf{x}$$

The mean vectors of the two classes

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$



类间散度最大是通过它们的均值距离体现的, 而 类内聚合度最强 是通过 类内的点到均值的散的程度表达的, 也就是说 Fisher 分类是 LS 的特例, 好了,看大家对 Fisher 还有什么疑问?

echo<echotooo@gmail.com> 20:41:53

....LDA 是基于高斯分布假设上的吗?

planktonli(1027753147) 20:44:12

LDA 的样本是需要在 Gaussian 假设下,会有 power performance 的, 如果 data 的 distribution 是非常不规则的, 那么 LDA 肯定是失效的。那就需要用些 Kernel 等的 trick 了。

echo<echotooo@gmail.com> 20:45:29

这是为什么, 是因为它的公式推导的时候有用到高斯分布的假设吗?

planktonli(1027753147) 20:45:32

推导不需要 Gaussian 假设

网络上的尼采(813394698) 20:46:01

用到了均值

echo<echotooo@gmail.com> 20:46:40

协方差部分呢?

planktonli(1027753147) 20:47:00

需要两类的 Between class Variance, 这个是通过 均值差表达的。

echo<echotooo@gmail.com> 20:47:28

嗯嗯，好像懂了，谢谢。

planktonli(1027753147) 20:47:45

如果两个类的 mean 完全相等,那么 LDA 肯定是失效的。

if the distribution of data is not so good, then we may use Kernel Fisher discriminant analysis

I mean that the distribution doesn't meet the gaussian.

echo<echotooo@gmail.com> 20:50:35

难道 kfda 不对高斯分布有偏好吗？

planktonli(1027753147) 20:50:42

the detail info you can c the web site http://en.wikipedia.org/wiki/Kernel_Fisher_discriminant_analysis KDA 算法步骤，大部分跟 LDA 相同，不同的地方是用到了 Kernel 方法构造了矩阵 S_b , S_w ，这里的 KDA 就是 kernel fisher 了。

电闪雷鸣(37633749) 20:52:11

OK，明白

planktonli(1027753147) 20:52:24

好了,下面看看 NN 神经网络的 perceptron，这个是一个单层的東西，注意它的 training error 函数，优化过程用的是梯度下降法：

The Perceptron Algorithm

More general form of a linear classifier

$$y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}))$$

where $\phi(\mathbf{x})$ is the feature vector of $\mathbf{x}$ (recall the basis functions in linear regression), and typically includes a bias component $\phi_0(\mathbf{x}) = 1$.

Given a set of samples $\{\mathbf{x}_n, t_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^D$, $t_n \in \{-1, 1\}$, our goal is to find the optimal parameter $\mathbf{w}$ to minimize the training error

$$E_p(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi_n t_n$$

where $\phi_n = \phi(\mathbf{x}_n)$ and $\mathcal{M}$ denotes the set of all misclassified patterns.

The Perceptron Algorithm

Let $\mathcal{M} = \{n\}$ where $\mathbf{x}_n$ is misclassified. Then

$$E_p(\mathbf{w}) = -\mathbf{w}^T \phi_n t_n$$

Stochastic gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_p(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

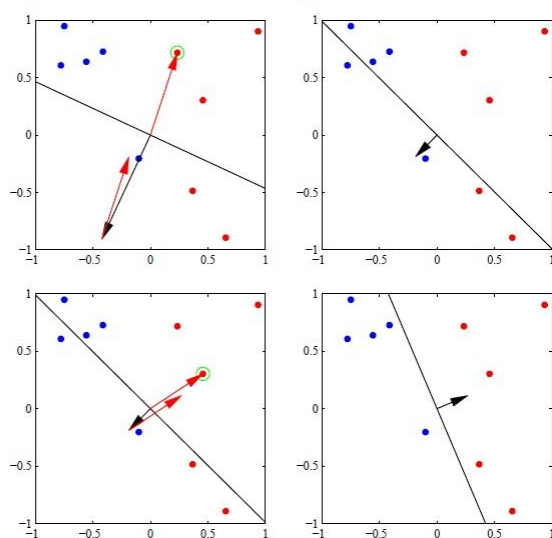
where η is the learning rate parameter and τ is an integer that indexes the steps of the algorithm. Without loss of generality, we can set $\eta = 1$.

For sample n

$$-\mathbf{w}^{(\tau+1)T} \phi_n t_n = -\mathbf{w}^{(\tau)T} \phi_n t_n - (\phi_n t_n)^T \phi_n t_n < -\mathbf{w}^{(\tau)T} \phi_n t_n$$

where we have set $\eta = 1$. This does not imply that the contribution to the error function from the other misclassified patterns will have been reduced.

Illustration of the Perceptron Algorithm



Properties of the Perceptron Algorithm

Perceptron convergence theorem: if there exists an exact solution (which means the training data is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

In practice, the number of steps required to achieve convergence could be substantial. Until convergence is achieved, we are not able to distinguish between a nonseparable problem and one that is simply slow to converge.

The perceptron algorithm does not converge when the data are not linearly separable.

好了,perceptron 是比较简单的。

下面看, Probabilistic Generative Models, 通过 MAP 方式建立概率模型, 需要 先验概率, 类条件概率和边缘概率。2 类的 Probabilistic Generative Models 就是 logistic sigmoid function :

Generative Models

Bayesian decision theory

$$\mathcal{C}(\mathbf{x}) = \mathcal{C}_k, \text{ if } p(\mathcal{C}_k|\mathbf{x}) \geq p(\mathcal{C}_j|\mathbf{x}) \forall j \neq k$$

The *generative* approach: model the class-conditional density $p(\mathbf{x}|\mathcal{C}_k)$ and the priors $p(\mathcal{C}_k)$, and use the posterior $p(\mathcal{C}_k|\mathbf{x})$ through Bayes' theorem.

Two-Class Generative Model

The posterior for class $\mathcal{C}_1$:

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\ &= \frac{1}{1 + \exp(-a)} \\ &= \sigma(a) \end{aligned}$$

where we have defined

$$a = \log \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

and $\sigma(a)$ is the *logistic sigmoid* function defined by

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

which is also called 'squashing function' because it maps the real axis to a finite

Logistic Sigmoid and Logit Functions

Logistic sigmoid function has the property

$$\sigma(-a) = 1 - \sigma(a)$$

The inverse of the logistic sigmoid is given by

$$a = \log \frac{\sigma}{1 - \sigma}$$

which is known as the *logit function*, or *log odds*.

这种方法需要假设 input 的分布，即得到 class-conditional distribution，用贝叶斯定理转化成后验概率后，就是和 Discriminant model 一样进行 make decision 了。

Multi-class Generative Model

For the case of $K > 2$ classes, we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

which is known as the *normalized exponential*, and a_k 's are defined by

$$a_k = \log p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

The normalized exponential is also known as the *softmax* function. If $a_k \gg a_j \forall j \neq k$, then $p(\mathcal{C}_k|\mathbf{x}) \approx 1$, and $p(\mathcal{C}_j|\mathbf{x}) \approx 0$.

在 gaussian 分布的情况下,我们分析：

Gaussian Likelihood Function

Assume that the class-conditional densities are Gaussians that share the same covariance matrix. The pdf for class C_k is

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu) \right\}$$

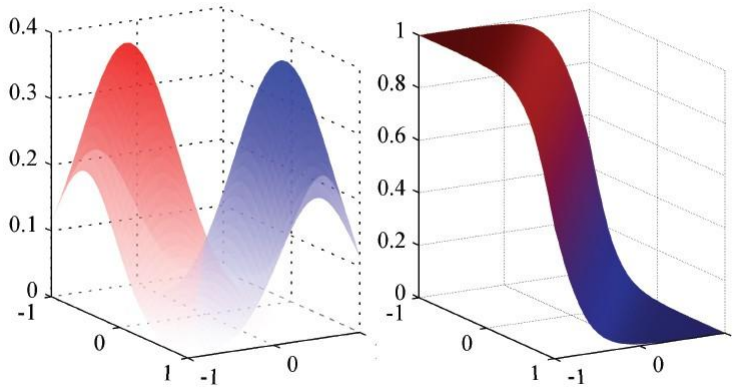
For the two-class case, we have

$$\begin{aligned} a(\mathbf{x}) &= \log \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \\ &= -\frac{1}{2}(\mathbf{x} - \mu_1)^T \Sigma^{-1}(\mathbf{x} - \mu_1) + \frac{1}{2}(\mathbf{x} - \mu_2)^T \Sigma^{-1}(\mathbf{x} - \mu_2) + \log \frac{p(C_1)}{p(C_2)} \\ &= (\mu_1 - \mu_2)^T \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 + \log \frac{p(C_1)}{p(C_2)} \\ &= \mathbf{w}^T \mathbf{x} + w_0 \end{aligned}$$

where

$$\begin{aligned} \mathbf{w} &= \Sigma^{-1}(\mu_1 - \mu_2) \\ w_0 &= -\frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 + \log \frac{p(C_1)}{p(C_2)} \end{aligned}$$

Two-Class Gaussian Likelihood Function



Multi-Class Gaussian Likelihood Function

For general K classes, we have

$$\begin{aligned} a_k &= \log p(\mathbf{x}|C_k)p(C_k) \\ &= \mu_k^T \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(C_k) + \mathbf{x}^T \Sigma^{-1} \mathbf{x} + \text{const} \end{aligned}$$

Notice that $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$ is cancelled in the softmax function. Therefore

$$a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

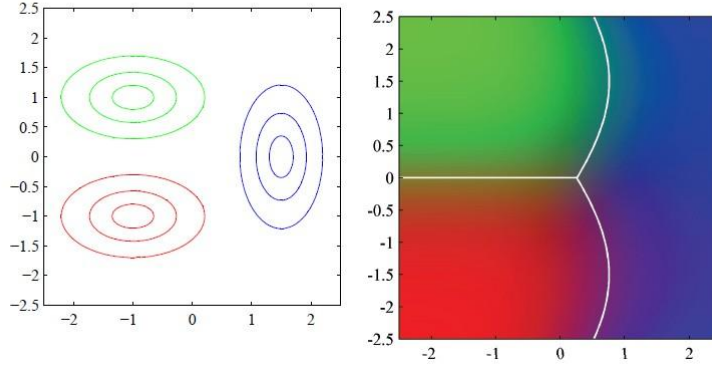
where

$$\begin{aligned} \mathbf{w}_k &= \Sigma^{-1} \mu_k \\ w_{k0} &= -\frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(C_k) \end{aligned}$$

Hence $a_k(\mathbf{x})$ is again linear function of $\mathbf{x}$.

Multi-Class Gaussian Likelihood Function

The covariance matrix of each class may not be identical, and the cancellation no longer occurs. This leads to *quadratic discriminant*.



协方差矩阵不同,则变成了 2 次分类了, 在 2 类情况下,我们用 MLE 方法, 估计参数:

Maximum Likelihood Estimator for Two Classes

Given a data set $\{\mathbf{x}_n, t_n\}_{n=1}^N$, $t_n \in \{0, 1\}$. Denote the prior class probability $p(\mathcal{C}_1) = \pi$ and $p(\mathcal{C}_2) = 1 - \pi$.

For a data point $\mathbf{x}_n$ generated from class $\mathcal{C}_1$ we have $t_n = 1$, and hence

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

For a data point $\mathbf{x}_n$ generated from class $\mathcal{C}_2$ we have $t_n = 0$, and hence

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Therefore, the likelihood function is

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$.

MLE for π

Maximize the log likelihood

$$\log p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \sum_{n=1}^N \left\{ t_n \log \pi + t_n \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + (1 - t_n) \log(1 - \pi) + (1 - t_n) \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right\}$$

The terms that depend on π are

$$\sum_{n=1}^N \{ t_n \log \pi + (1 - t_n) \log(1 - \pi) \}$$

Setting the derivative w.r.t. π equal to zero, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

where N_1 denotes the total number of data points in class $\mathcal{C}_1$ and N_2 denotes the total number of data points in class $\mathcal{C}_2$.

MLE for μ_1 and μ_2

The terms in the log likelihood that depend on μ_1

$$\frac{1}{N} \sum_{n=1}^N t_n \log \mathcal{N}(\mathbf{x}_n | \mu_1, \Sigma) = -\frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \mu_1)^T \Sigma^{-1} (\mathbf{x}_n - \mu_1) + \text{const}$$

Setting the derivative w.r.t. μ_1 to zero and obtain

$$\mu_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n$$

or simply the mean of all the input vectors $\mathbf{x}_n$ labeled as class $\mathcal{C}_1$. Similarly

$$\mu_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n$$

MLE for Σ

The terms in the log likelihood that depend on Σ

$$\begin{aligned} & -\frac{1}{2} \sum_{n=1}^N t_n \log |\Sigma| - \frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \mu_1)^T \Sigma^{-1} (\mathbf{x}_n - \mu_1) \\ & -\frac{1}{2} \sum_{n=1}^N (1 - t_n) \log |\Sigma| - \frac{1}{2} \sum_{n=1}^N (1 - t_n) (\mathbf{x}_n - \mu_2)^T \Sigma^{-1} (\mathbf{x}_n - \mu_2) \\ & = -\frac{N}{2} \log |\Sigma| - \frac{N}{2} \text{Tr} \Sigma^{-1} \mathbf{S} \end{aligned}$$

where

$$\begin{aligned} \mathbf{S} &= \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2 \\ \mathbf{S}_1 &= \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mu_1)(\mathbf{x}_n - \mu_1)^T \\ \mathbf{S}_2 &= \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mu_2)(\mathbf{x}_n - \mu_2)^T \end{aligned}$$

MLE for Σ

Using what we derived for the MLE for the covariance matrix for a Gaussian distribution, we obtain

$$\Sigma = \mathbf{S}$$

Therefore, the MLE for the parameters

$$\begin{aligned} \pi &= \frac{N_1}{N_1 + N_2} \\ \mu_1 &= \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n \\ \mu_2 &= \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n \\ \Sigma &= \mathbf{S} \end{aligned}$$

步骤小结：

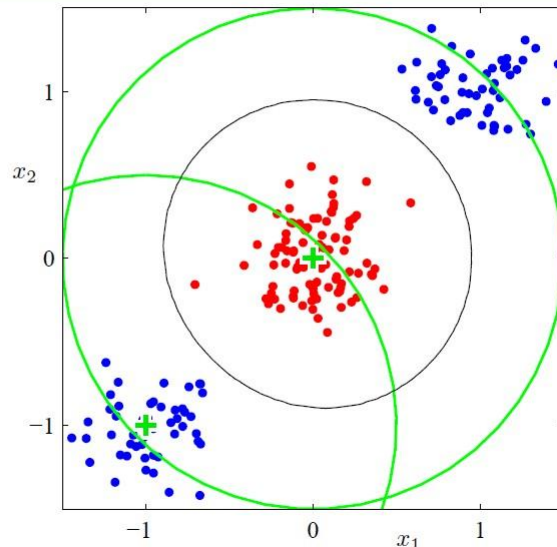
- 1) 假定 class-conditional distribution 的分布形式，MLE 估计该分布中的参数（从而得到了 class-conditional distribution）
 - 2) 计算每个类别的后验概率。在上面的例子中，得到的后验概率刚好是一个 GLM 模型（Logistic）
- 好了,这部分结束了。大家讨论下，没问题就继续了。

Probabilistic Discriminative Models：

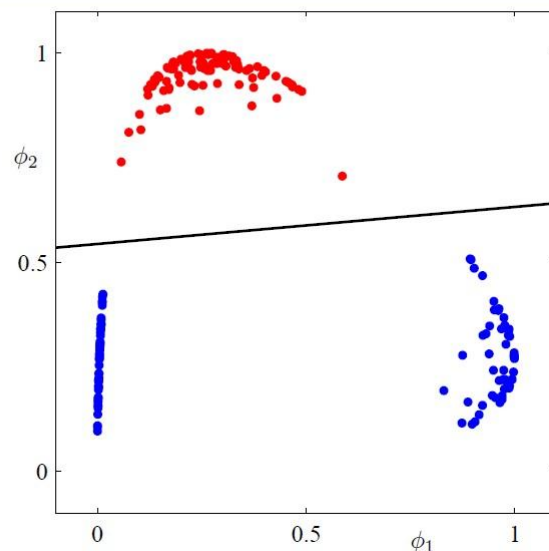
直接建立分类函数模型,而不是建立生成过程模型，生成模型和判别模型的区别：

mapping the data from the original sapce to a new space may make it linearly separable

Linearly Not Separable in the Original Space



Linearly Separable in the Space of Basis Functions



Work in the New Space

We will work in the feature space $\phi(\mathbf{x})$ which includes $\phi_0(\mathbf{x}) = 1$ so that the corresponding parameter w_0 plays the role of a bias.

Nonlinear basis functions are double-edged sword. It can ease the problem, but can also introduce more overlap of the posterior. Domain expertise is often required to design good nonlinear basis functions.

Logistic Regression

We can directly express the posterior in the space of basis functions

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$

where $p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$ and $\sigma(\cdot)$ is the *logistic sigmoid* function.

For an M -dimensional feature space ϕ , this model has M parameters. In contrast, for a Gaussian generative model using MLE, we have $2M$ parameters for the means, $M(M+1)/2$ for the shared covariance matrix. Together with the class prior $p(\mathcal{C}_1)$, there are in total $M(M+5)/2 + 1$ parameters.

逻辑回归的最大似然参数估计方法：

MLE for Logistic Regression

Given a data set $\{\phi_n, t_n\}_{n=1}^N$, $t_n \in \{0, 1\}$ and $\phi_n = \phi(\mathbf{x}_n)$. The likelihood function is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

where $\mathbf{t} = (t_1, \dots, t_N)^T$ and $y_n = \sigma(\mathbf{w}^T \phi_n)$.

The error function (negative log likelihood) is

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\}$$

Recall the logistic sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

Therefore

$$\frac{d\sigma}{da} = \frac{\exp(-a)}{(1 + \exp(-a))^2} = \sigma(1 - \sigma)$$

MLE for Logistic Regression

Because $y_n = \sigma(\mathbf{w}^T \phi_n)$, we have

$$\begin{aligned}\nabla E(\mathbf{w}) &= - \sum_{n=1}^N \left\{ t_n \frac{y_n(1-y_n)}{y_n} - (1-t_n) \frac{y_n(1-y_n)}{(1-y_n)} \right\} \phi_n \\ &= - \sum_{n=1}^N \{ t_n(1-y_n) - (1-t_n)y_n \} \phi_n \\ &= \sum_{n=1}^N (y_n - t_n) \phi_n\end{aligned}$$

The gradient is contributed by the error $y_n - t_n$ times the basis function vector ϕ_n . This update is similar to that of perceptron algorithm.

However, gradient descent algorithms tend to converge to bad minimums.

Newton-Raphson Update

Gradient and Hessian

$$\begin{aligned}\nabla E(\mathbf{w}) &= \sum_{n=1}^N (y_n - t_n) \phi_n = \Phi^T (\mathbf{y} - \mathbf{t}) \\ \mathbf{H} = \nabla \nabla E(\mathbf{w}) &= \sum_{n=1}^N y_n(1-y_n) \phi_n \phi_n^T = \Phi^T \mathbf{R} \Phi\end{aligned}$$

where $\Phi = (\phi_1, \dots, \phi_N)^T \in \mathbb{R}^{N \times M}$ is the design matrix, $\mathbf{y} = (y_1, \dots, y_N)^T$ is the vector of the posterior based on the current $\mathbf{w}$, and $\mathbf{R}$ is a diagonal matrix

$$R_{nn} = y_n(1-y_n)$$

This Hessian matrix H is positive definite because $0 < y_n < 1$.

Newton-Raphson Update

Newton-Raphson update

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \left\{ \Phi^T \mathbf{R} \Phi \mathbf{w}^{(\text{old})} - \Phi^T (\mathbf{y} - \mathbf{t}) \right\} \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z}\end{aligned}$$

where

$$\mathbf{z} = \Phi \mathbf{w}^{(\text{old})} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})$$

In other words

$$\mathbf{w}^{(\text{new})} - \mathbf{w}^{(\text{old})} = -(\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t})$$

Linearization

We can derive from another perspective.

Let $\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} + \Delta \mathbf{w}$. Taylor expansion leads to

$$\begin{aligned} y_n^{(\text{new})} &= \sigma(\mathbf{w}^{(\text{new})T} \phi_n) \\ &= \sigma(\mathbf{w}^{(\text{old})T} \phi_n + \Delta \mathbf{w}^T \phi_n) \\ &\approx y_n^{(\text{old})} + y_n^{(\text{old})}(1 - y_n^{(\text{old})}) \phi_n^T \Delta \mathbf{w} \end{aligned}$$

Now we can drop the superscript (old) and put it into the derivative

$$\begin{aligned} \nabla E(\mathbf{w}) &= \sum_{n=1}^N (y_n - t_n) \phi_n \\ &= \sum_{n=1}^N \phi_n [y_n - t_n + y_n(1 - y_n) \phi_n^T \Delta \mathbf{w}] \\ &= \Phi^T (\mathbf{y} - \mathbf{t}) + \Phi^T \mathbf{R} \Phi \Delta \mathbf{w} \end{aligned}$$

Setting the gradient to zero we obtain

$$\Delta \mathbf{w} = -(\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t})$$

Iterative Reweighted Least Square

1. Initialize $\mathbf{w}$
2. Compute current prediction $y_n = \sigma(\mathbf{w}^T \phi_n)$, and weight $R_{nn} = y_n(1 - y_n)$
3. Solve linear system: $\Delta \mathbf{w} = -(\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t})$
4. $\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w}$
5. If $\Delta \mathbf{w} < \varepsilon$, go to 2.

Frequentist 版本	Bayesian 版本	解模型所用的方法
Linear basis function regression	Bayesian linear basis function regression	前者和后者皆有 closed-form solution
Logistic regression	Bayesian logistic regression	前者牛顿迭代(IRLS), 后者 Laplace approximation

注意这里, Logistic regression 是用于分类的,而不是回归。

好了,这就是 Probabilistic Discriminative Models 的内容,其实质还是 point estimation。

最后看看 Bayesian Logistic Regression :

这里是 we want to approximate the posterior using Gaussian, 就是用高斯分布近似后验概率来看 Laplace Approximation :

The Laplace Approximation

Suppose the distribution $p(z)$ is defined by

$$p(z) = \frac{1}{Z}f(z)$$

where $Z = \int f(z)dz$ is the normalization constant and is unknown.

Our goal is to find a Gaussian approximation to a continuous pdf at a mode. In general, it is a very useful technique to use one simple distribution (such as Gaussian) to approximate more complicated distributions.

The Laplace Approximation

The first step is to find a mode of $p(z)$, z_0 , such that $p'(z_0) = 0$, or

$$\left. \frac{df(z)}{dz} \right|_{z=z_0} = 0$$

The logarithm of a Gaussian distribution is a quadratic function. Therefore, consider a Taylor expansion of $\log f(z)$ center at mode z_0

$$\log f(z) \approx \log f(z_0) - \frac{1}{2}A(z - z_0)^2$$

where

$$A = - \left. \frac{d^2}{dz^2} \log f(z) \right|_{z=z_0}.$$

The first-order term in the Taylor expansion does not appear because

$$\left. \frac{d}{dz} \log f(z) \right|_{z=z_0} = \frac{f'(z_0)}{f(z_0)} = 0$$

The Laplace Approximation

Taking the exponential we obtain

$$f(z) \approx f(z_0) \exp \left\{ -\frac{A}{2}(z - z_0)^2 \right\}$$

We then obtain a normalized distribution $q(z)$

$$q(z) = \left(\frac{A}{2\pi} \right)^{\frac{1}{2}} \exp \left\{ -\frac{A}{2}(z - z_0)^2 \right\}$$

The distribution is valid iff $A > 0$, or the second order derivative is negative at z_0 , indicating that z_0 is a local maximum.

Laplace 近似将任意一个分布近似成了高斯分布

High-Dimensional Distribution

Extend the Laplace method to approximate a distribution $p(\mathbf{z}) = f(\mathbf{z})/Z$ where $\mathbf{z} \in \mathbb{R}^M$. At a local maximum $\mathbf{z}_0$ the gradient $\nabla f(\mathbf{z})$ vanishes. So the Taylor expansion leads to

$$\log f(\mathbf{z}) \approx \log f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0)$$

where the $M \times M$ Hessian matrix $\mathbf{A}$ is defined by

$$\mathbf{A} = -\nabla \nabla \log f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

Taking the exponential of both sides we obtain

$$f(\mathbf{z}) \approx f(\mathbf{z}_0) \exp \left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \right\}$$

Therefore, the approximate Gaussian distribution is

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{\frac{1}{2}}}{(2\pi)^{\frac{M}{2}}} \exp \left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \right\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1})$$

Bayesian Inference

Since we want to approximate the posterior using Gaussian, it is natural to set a Gaussian prior for the parameter

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

Recall the likelihood

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

Therefore the logarithm of the posterior is

$$\begin{aligned} \log p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \Sigma_0^{-1}(\mathbf{w} - \mathbf{m}_0) \\ &\quad + \sum_{n=1}^N \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\} \end{aligned}$$

where $y_n = \sigma(\mathbf{w}^T \Phi_n)$.

Laplace Approximation

Compute gradient and Hessian

$$\begin{aligned} -\nabla \log p(\mathbf{w}|\mathbf{t}) &= \Phi^T(\mathbf{y} - \mathbf{t}) + \mathbf{S}_0^{-1}\mathbf{w} - \mathbf{S}_0^{-1}\mathbf{m}_0 \\ \mathbf{H} &= \mathbf{S}_0^{-1} + \Phi^T \mathbf{R} \Phi = \mathbf{S}_N^{-1} \end{aligned}$$

So

$$\Delta \mathbf{w} = -(\mathbf{S}_0^{-1} + \Phi^T \mathbf{R} \Phi)^{-1}(\Phi^T(\mathbf{y} - \mathbf{t}) + \mathbf{S}_0^{-1}\mathbf{w} - \mathbf{S}_0^{-1}\mathbf{m}_0)$$

We can use IRLS to obtain the *maximum a posteriori* (MAP) $\mathbf{w}_{\text{MAP}}$.

The Gaussian approximation to the posterior therefore takes the form

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{S}_N(\mathbf{w}_{\text{MAP}}))$$

Predictive Distribution

The predictive distribution for class $\mathcal{C}_1$, given a new feature vector $\phi(\mathbf{x})$, is obtained by marginalizing w.r.t. the posterior distribution $p(\mathbf{w}|\mathbf{t})$, which is approximated by a Gaussian distribution $q(\mathbf{w})$:

$$p(\mathcal{C}_1|\phi, \mathbf{t}) = \int p(\mathcal{C}_1|\phi, \mathbf{w})p(\mathbf{w}|\mathbf{t})d\mathbf{w} \approx \int \sigma(\mathbf{w}^T \phi)q(\mathbf{w})d\mathbf{w}$$

The corresponding probability for class $\mathcal{C}_2$ is given by

$$p(\mathcal{C}_2|\phi, \mathbf{t}) = 1 - p(\mathcal{C}_1|\phi, \mathbf{t})$$

Predictive Distribution

Note that

$$\sigma(\mathbf{w}^T \phi) = \int \delta(a - \mathbf{w}^T \phi) \sigma(a) da$$

where $\delta(\cdot)$ is the Dirac delta function. Therefore

$$\begin{aligned} \int \sigma(\mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w} &= \int \int \delta(a - \mathbf{w}^T \phi) \sigma(a) q(\mathbf{w}) d\mathbf{w} da \\ &= \int \sigma(a) \int \delta(a - \mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w} da \\ &= \int \sigma(a) p(a) da \end{aligned}$$

where

$$p(a) = \int \delta(a - \mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w}$$

The function $\delta(a - \mathbf{w}^T \phi)$ can be approximated by a Gaussian

$$\delta(a - \mathbf{w}^T \phi) \approx \mathcal{N}(\phi^T \mathbf{w}, \alpha \mathbf{I}), \alpha \rightarrow \infty$$

Predictive Distribution

Therefore

$$\begin{aligned} \mu_a &= \phi^T \mathbf{w}_{\text{MAP}} \\ \sigma_a^2 &= \phi^T \mathbf{S}_N \phi + \frac{1}{\alpha} \mathbf{I} = \phi^T \mathbf{S}_N \phi \end{aligned}$$

So the predictive distribution becomes

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a) p(a) da = \int \sigma(a) \mathcal{N}(a|\mu_a, \sigma_a^2) da$$

The integration is difficult due to the logistic function $\sigma(\cdot)$. Instead, we can use the inverse *probit* function $\Phi(a)$ to approximate the *logistic* function

$$\Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta|0, 1) d\theta$$

好了,最后的 Bayesian Logistic Regression 也完了。

=====讨论=====

zeno(117127143) 21:25:06

没太明白，生成模型和判别模型的区别，优缺点呢？

planktonli(1027753147) 21:26:02

一个 model $p(x,y)$ 生成式的，一个 model $p(y|x)$ 判别式的，判别式的只在乎 boundary 的这些点，生成式的需要知道这些点是怎么生成的：

Generative and Discriminative classifiers



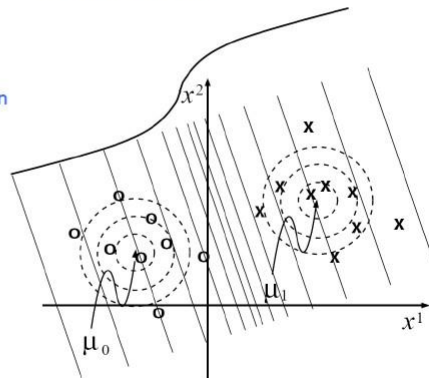
- Goal: Wish to learn $f: X \rightarrow Y$, e.g., $P(Y|X)$

- Generative:

- Modeling the joint distribution of all data

- Discriminative:

- Modeling only points at the boundary



zeno(117127143) 21:26:38

是判别式要好一点吗？

planktonli(1027753147) 21:27:34

生成模型: Naive Bayes, Graphical Model，判别模型: NN, SVM, LDA, Decision Tree 等。不能说谁好谁不好，还有生成模型 + 判别模型的，例如: SVM 的 kernel construction 你可以用 generative 的方式去做，那就是 Generative + Discriminant 的了。

zeno(117127143) 21:30:38

生成式要求 X 互相独立吧？

planktonli(1027753147) 21:31:39

恩，生成式是需要 IID 的，这个在 statistics 上通常都有 IID 假设的，否则不好整，还有什么问题？

HX(458728037) 21:33:12

我想问一个简单的问题，其实生成模型最后做分类的时候不还是根据一个 boundary 来判断的吗？

planktonli(1027753147) 21:34:32

生成模型在 two class 而且是 gaussian distribution 的时候，就可以转化为判别式的，这个验证起来很简单，看看下面的 PPT 就明白了：

Gaussian Discriminative Analysis



- learning $f: X \rightarrow Y$, where
 - X is a vector of real-valued features, $\mathbf{x}_n = \langle X_{n,1} \dots X_{n,m} \rangle$
 - Y is an indicator vector
- What does that imply about the form of $P(Y|X)$?
 - The joint probability of a datum and its label is:



$$p(\mathbf{x}_n, y_n^k = 1 | \mu, \sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n | y_n^k = 1, \mu, \sigma)$$

$$= \pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2} (\mathbf{x}_n - \mu_k)^2\right\}$$

- Given a datum $\mathbf{x}_n$, we predict its label using the conditional probability of the label given the datum:

$$p(y_n^k = 1 | \mathbf{x}_n, \mu, \sigma) = \frac{\pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2} (\mathbf{x}_n - \mu_k)^2\right\}}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2} (\mathbf{x}_n - \mu_{k'})^2\right\}}$$

Naïve Bayes Classifier



- When X is multivariate-Gaussian vector:
 - The joint probability of a datum and its label is:



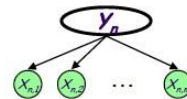
$$p(\mathbf{x}_n, y_n^k = 1 | \tilde{\mu}, \Sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n | y_n^k = 1, \tilde{\mu}, \Sigma)$$

$$= \pi_k \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2} (\mathbf{x}_n - \tilde{\mu}_k)^T \Sigma^{-1} (\mathbf{x}_n - \tilde{\mu}_k)\right\}$$

- The naïve Bayes simplification

$$p(\mathbf{x}_n, y_n^k = 1 | \mu, \sigma) = p(y_n^k = 1) \times \prod_j p(x_{n,j} | y_n^k = 1, \mu_{k,j}, \sigma_{k,j})$$

$$= \pi_k \prod_j \frac{1}{(2\pi\sigma_{k,j}^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma_{k,j}^2} (x_{n,j} - \mu_{k,j})^2\right\}$$



- More generally: $p(\mathbf{x}_n, y_n | \eta, \pi) = p(y_n | \pi) \times \prod_{j=1}^m p(x_{n,j} | y_n, \eta)$
 - Where $p(\cdot | \cdot)$ is an arbitrary conditional (discrete or continuous) 1-D density

The predictive distribution

- Understanding the predictive distribution

$$p(y_n^k = 1 | x_n, \bar{\mu}, \Sigma, \pi) = \frac{p(y_n^k = 1, x_n | \bar{\mu}, \Sigma, \pi)}{p(x_n | \bar{\mu}, \Sigma)} = \frac{\pi_k N(x_n, | \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} N(x_n, | \mu_{k'}, \Sigma_{k'})} *$$

- Under naïve Bayes assumption:

$$p(y_n^k = 1 | x_n, \bar{\mu}, \Sigma, \pi) = \frac{\pi_k \exp \left\{ -\sum_j \left(\frac{1}{2\sigma_{k,j}^2} (x_n^j - \mu_{k,j}^j)^2 - \log \sigma_{k,j} - C \right) \right\}}{\sum_{k'} \pi_{k'} \exp \left\{ -\sum_j \left(\frac{1}{2\sigma_{k',j}^2} (x_n^j - \mu_{k',j}^j)^2 - \log \sigma_{k',j} - C \right) \right\}} **$$

- For two class (i.e., $K=2$), and when the two classes have the same variance, ** turns out to be a **logistic function**

$$p(y_n^1 = 1 | x_n) = \frac{1}{1 + \frac{\pi_2 \exp \left\{ -\sum_j \left(\frac{1}{2\sigma_j^2} (x_n^j - \mu_2^j)^2 - \log \sigma_j - C \right) \right\}}{\pi_1 \exp \left\{ -\sum_j \left(\frac{1}{2\sigma_j^2} (x_n^j - \mu_1^j)^2 - \log \sigma_j - C \right) \right\}}} = \frac{1}{1 + \exp \left\{ -\sum_j \left(x_n^j \frac{1}{\sigma_j^2} (\mu_1^j - \mu_2^j) + \frac{1}{\sigma_j^2} ([\mu_1^j]^2 - [\mu_2^j]^2) + \log \frac{\pi_1}{\pi_2} \right) \right\}} = \frac{1}{1 + e^{-\theta^T x_n}}$$

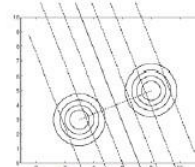
The decision boundary

- The predictive distribution

$$p(y_n^1 = 1 | x_n) = \frac{1}{1 + \exp \left\{ -\sum_{j=1}^M \theta_j x_n^j - \theta_0 \right\}} = \frac{1}{1 + e^{-\theta^T x_n}}$$

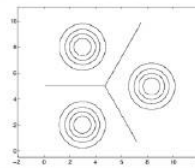
- The Bayes decision rule:

$$\ln \frac{p(y_n^1 = 1 | x_n)}{p(y_n^2 = 1 | x_n)} = \ln \left(\frac{\frac{1}{1 + e^{-\theta^T x_n}}}{\frac{e^{-\theta^T x_n}}{1 + e^{-\theta^T x_n}}} \right) = \theta^T x_n$$



- For multiple class (i.e., $K>2$), * correspond to a **softmax function**

$$p(y_n^k = 1 | x_n) = \frac{e^{-\theta_k^T x_n}}{\sum_j e^{-\theta_j^T x_n}}$$



19.048 厘米

© Eric Xing @ CMU, 2006-2010

10

zeno(117127143) 21:36:52

好的，谢谢，总是看见生成判别，弄不太清楚，关键是强调分清楚生成判别，对解决问题选那种工具有何影响，一直搞不清，我感觉 feature 不独立时用判别，feature 多时用判别，那么生成式有什么优势呢？

阳阳(236995728) 21:34:22

判别模型用的是频率学派观点，也就是最大似然估计法，生成模型用的是贝叶斯学派观点，也就是最大后验概率。

planktonli(1027753147) 21:37:10

阳阳，这个是不对的。MLE 也是 statistics 的东西，判别模型则根本不考虑 statistics，你可以看看 NN 的东西，包括今天的 LDA，它们直接求 boundry 的。

晴(498290717) 21:39:41

查了下别人的博客，我觉得这个是两者最最本质的区别

判别模型 (Discriminative Model)，又可以称为条件模型，或条件概率模型。估计的是条件概率分布

生成模型 (Generative Model)，又叫产生式模型。估计的是联合概率分布

ant/wxony(824976094) 21:42:18

产生式模型和判别式模型我觉得就是把人的知识用在模型的构建还是 feature 的设计上，特征多的时候一般还是判别式模型猛些。

HX(458728037) 21:44:51

生成式的模型在无监督的一些学习上应该才好用吧？

ant/wxony(824976094) 21:45:43

嗯，无监督学习上确实

planktonli(1027753147) 21:48:38

我把 Generative verses discriminative classifier 的一个 PPT 发到群里，大家有兴趣的看看：

"Lecture2.pdf" 下载

第五章 Neural Networks

主讲人 网神

(新浪微博:@豆角茄子麻酱凉面)

网神(66707180) 18:55:06

那我们开始了啊，前面第 3,4 章讲了回归和分类问题，他们应用的主要限制是维度灾难问题。今天的第 5 章神经网络的内容：

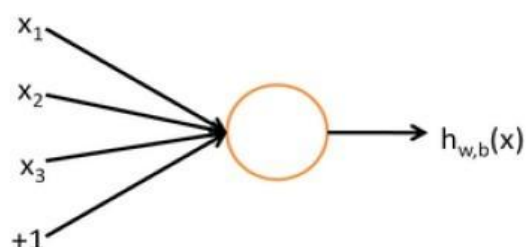
1. 神经网络的定义

2. 训练方法：error 函数，梯度下降，后向传导

3. 正则化：几种主要方法，重点讲卷积网络

书上提到的这些内容今天先不讲，以后有时间再讲：BP 在 Jacobian 和 Hessian 矩阵中求导的应用；混合密度网络；贝叶斯解释神经网络。

首先是神经网络的定义，先看一个最简单的神经网络，只有一个神经元：



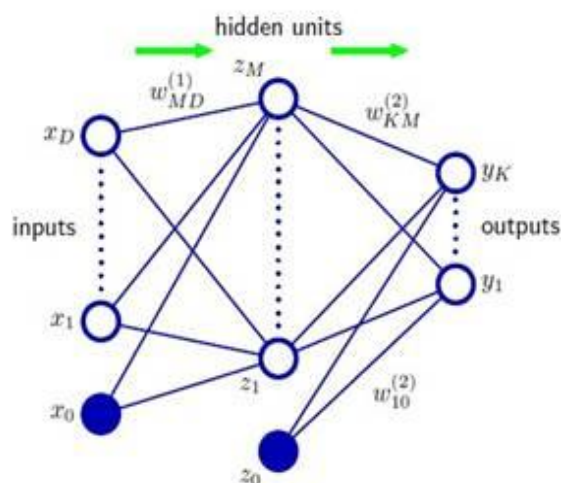
这个神经元是一个以 x_1, x_2, x_3 和截距 1 为输入的运算单元，其输出是：

$$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^3 W_i x_i + b)$$

其中函数 f 成为“激活函数”，activation function. 激活函数根据实际应用确定，经常选择 sigmoid 函数.

如果是 sigmoid 函数，这个神经元的输入-输出的映射就是一个 logistic 回归问题。

神经网络就是将许多个神经元连接在一起组成的网络，如图：



神经网络的图跟后面第八章图模型不同，图模型中，每个节点都是一个随机变量，符合某个分布。神经网络中的节点是确定的值，由与其相连的节点唯一确定。

上图这个神经网络包含三层，左边是输入层， $x_1 \dots x_d$ 节点是输入节点， x_0 是截距项。最右边是输出层， $y_1, \dots, y_K$ 是输出节点。中间是隐藏层 hidden level, $z_1, \dots, z_M$ 是隐藏节点，因为不能从训练样本中观察到他们的值，所以叫隐藏层。

可以把神经网络描述为一系列的函数转换。首先，构造 M 个输入节点的线性组合：

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

上式中, $j=1,\dots,M$, 对应 M 个隐藏节点. 上标(1)表示这是第一层的参数. w_{ji} 是权重 weight, w_{j0} 是偏置. a_j 叫做 activation. 中文叫激活吧, 感觉有点别扭。

把 activation a_j 输入到一个非线性激活函数 $h(\cdot)$ $z_j = h(a_j)$. 就得到了隐藏节点的值 z_j 。

在这个从输入层到隐藏层的转换中, 这个激活函数不能是线性的, 接下来, 将隐藏节点的值线性组合得到输出节点的 activation:

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

每个 a_k 输入一个输出层激活函数 $y_k = \sigma(a_k)$, 就得到了输出值。

这个从隐藏层到输出层的激活函数 σ , 根据不同应用, 有不同的选择, 例如回归问题的激活函数是 identity 函数, 既 $y = a$. 分类问题的激活函数选择 sigmoid 函数, multiclass 分类选择是 softmax 函数。

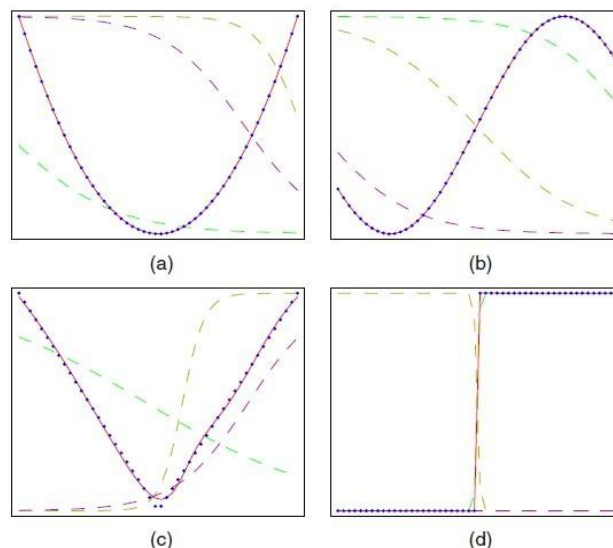
把上面各阶段的计算连起来, 神经网络整个的计算过程就是:

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

上面计算过程是以 2 层神经网络为例。实际使用的 NN 可能有多层, 记得听一个报告现在很火的 deep learning 的隐藏层有 5-9 层. 这个计算过程 forward propagation, 从前向后计算。与此相反, 训练时, 会用 back propagation 从后向前来计算偏导数。

神经网络有很强的拟合能力, 可以拟合很多种的曲线, 这个图是书上的一个例子, 对四种曲线的拟合:

Figure 5.3 Illustration of the capability of a multilayer perceptron to approximate four different functions comprising (a) $f(x) = x^2$, (b) $f(x) = \sin(x)$, (c), $f(x) = |x|$, and (d) $f(x) = H(x)$ where $H(x)$ is the Heaviside step function. In each case, $N = 50$ data points, shown as blue dots, have been sampled uniformly in x over the interval $(-1, 1)$ and the corresponding values of $f(x)$ evaluated. These data points are then used to train a two-layer network having 3 hidden units with 'tanh' activation functions and linear output units. The resulting network functions are shown by the red curves, and the outputs of the three hidden units are shown by the three dashed curves.



第一部分神经网络的定义就这么多, 大家有没有什么问题啊?

=====讨论=====

阳阳(236995728) 19:20:43

输入层到隐藏层使用的激活函数与隐藏层到输出层的激活函数是否要一致?

网神(66707180) 19:21:11

两层激活函数不一定一致，输入层到隐藏层 经常是 sigmoid 函数。而隐藏层到输出层，根据应用不同，选择不同。

牧云(1106207961) 19:22:30

一般的算法，比如神经网络，都有分类和拟合的功能，分类和拟合有共同点吗？为什么能拟合，这个问题我没有找到地方清晰的解释。

独孤圣者(303957511) 19:23:47

拟合和分类，在我看来实际上是一样的，分类无非是只有 2 个或多个值的拟合而已。2 个值的拟合，被称为二值分类

ant/wxony(824976094) 19:24:30

不是吧，svm 做二值分类，就不是以拟合为目标。二值拟合可以用做分类。

独孤圣者(303957511) 19:29:42

可以作为概率来理解吧，我个人认为啊，sigmoid 函数，是将分类结果做一个概率的计算。

阳阳(236995728) 19:24:07

输入层到隐藏层 经常是 sigmoid 函数 那么也就是特征值变成了【0-1】之间的数了？

网神(66707180) 19:25:19

是的，我认为是机器学习经常需要做特征归一化，也是把特征归一化到[0, 1]范围。当然这里不只是归一化。

阳阳(236995728) 19:26:56

这里应该不是归一化的作用@网神

网神(66707180) 19:27:33

嗯，不是归一化，我觉得神经网络的这些做法，没有一个科学的解释。不像 SVM 每一步都有严谨的理论。

罗杰兔(38900407) 19:29:00

参数 W,b 是算出来的，还是有些技巧得到的。因为常听人说，神经网络难就难在调试参数上。

阳阳(236995728) 19:29:11

我觉得神经网络也是在学习新的特征 而这些特征比较抽象难于理解@牧云 @网神

网神(66707180) 19:29:48

对，我理解隐藏层的目的就是学习特征

阳阳(236995728) 19:30:24

是的 但是这些特征难于理解较抽象@网神

网神(66707180) 19:30:47

最后一个隐藏层到输出层之间，就是做分类或回归。前面的不同隐藏层，则是提取特征。

独孤圣者(303957511) 19:30:52

隐藏层的目的就是学习特征，这个表示赞同

网神(66707180) 19:31:33

后面讲到卷积网络时，可以很明显感受到，隐藏层的目的就是学习特征。

阳阳(236995728) 19:31:59

但是我不理解为什么隐藏层学习到的特征值介于【0-1】之间是合理的？是归一化的作用？

牧云(1106207961) 19:32:47

归一化映射

Wilbur_中博(1954123) 19:33:59

要归一化也是对每一个输入变量做吧。。各个输入变量都组合在一起了，归一化干嘛。我理解就是一个非线性变换，让神经网络具有非线性能力。sigmoid 之外，其他非线性变换用的也蛮多的。和归一化没什么关系。不一定要在[0, 1]之间。

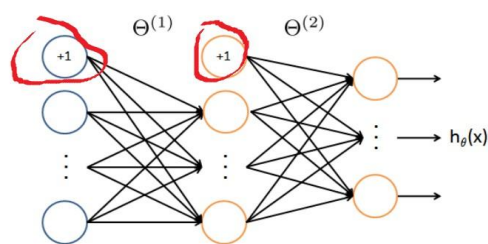
独孤圣者(303957511) 19:39:50

这个解释我很同意, tanh 函数也经常替换 sigmoid 函数, 在神经网络中普遍使用。

网神(66707180) 19:39:59

嗯, 让 NN 具有非线性能力是主要原因之一。

牧云(1106207961) 19:40:03



这个 1 要的干嘛

网神(66707180) 19:40:18

这个是偏置项, $y = wx + b$, 那个 +1 就是为了把 b 合入 w , 变成 $y = w x$

罗杰兔(38900407) 19:41:36

Ng 好象讲 b 是截距

网神(66707180) 19:43:16

截距和偏执项是一个意思。放在坐标系上, b 就是截距

阳阳(236995728) 19:40:33

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

它可以逼近任意的非线性函数吗@网神

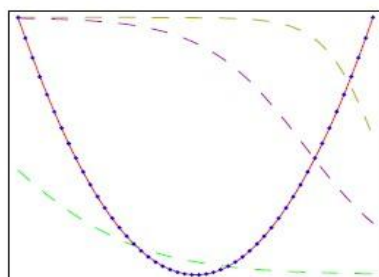
网神(66707180) 19:42:13

书上没说可以逼近任意曲线, 说的是逼近很多曲线

阳阳(236995728) 19:42:58

它到底是怎么逼近的啊@网神 我想看看多个函数叠加的?

网神(66707180) 19:44:30



就是多个函数叠加。这个图上那个抛物线就是最终的曲线, 由另外三条曲线叠加而成, 另外那三条, 每条是一个隐藏节点生成的曲线。

阳阳(236995728) 19:45:05

这三条曲线用的函数形式是不是一样的

zeno(117127143) 19:45:31

理解 nn 怎么逼近 xor 函数, 就知道怎么非线性的了

HX(458728037) 19:45:56

问一句, 你现在讲的是 BP 网络吧?

网神(66707180) 19:46:07

是 BP 网络

=====讨论结束=====

网神(66707180) 19:46:59

接下来是神经网络的训练。神经网络训练的套路与第三，四章相同：确定目标变量的分布函数和似然函数，取负对数，得到 error 函数，用梯度下降法求使 error 函数最小的参数 w 和 b。

NN 的应用有：回归、binary 分类、K 个独立 binary 分类、multiclass 分类。不同的应用决定了不同的激活函数、不同的目标变量的条件分布，不同的 error 函数。

首先，对于回归问题，输出变量 t 的条件概率符合高斯分布：

$$p(t|x, w) = \mathcal{N}(t|y(x, w), \beta^{-1})$$

因此，给定一个训练样本集合，其似然函数是：

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N p(t_n|x_n, \mathbf{w}, \beta).$$

error 函数是对似然函数取 负对数，得到：

$$\frac{\beta}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln(2\pi)$$

最小化这个 error 函数，只需要最小化第一项，所以回归问题的 error 函数定义是：

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

回归问题的隐藏层到输出层的激活函数是 identity 函数，也就是 $y_k = a_k$

a_k 就是对隐藏层节点的线性组合：

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

这样有了 error 函数，就可以同梯度下降法求极值点。

再说说 binary 分类和 multiclass 分类的 error 函数：

binary 分类的目标变量条件分布式伯努利分布： $p(t|x, w) = y(x, w)^t \{1 - y(x, w)\}^{1-t}$

这样，通过对似然函数取负对数，得到 error 函数如下：

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

另外 binary 分类，隐藏层到输出层的激活函数是 sigmoid 函数。对 multiclass 问题，也就是结果是 K 个互斥的类别中的一个。类似思路可以得到其 error 函数。这里就不说了。书上都有。

知道了 error 函数，和激活函数，这里先做一个计算，为后面的 BP 做准备，将 $E(\mathbf{w})$ 对激活 a_k 求导，可得：

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

这里有意思的是，不管是回归、binary 分类还是 multiclass 分类，尽管其 error 函数不同，这个求导得出的结果都是 $y_k - t_k$

回归问题这个求导很明显，下式中， $y = a$ ，所以对 a 求导就是 $y - t$ 。

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2$$

其他错误函数，需要推导一下，这里就不推了。有了 error 函数，用梯度下降法求其极小值点。

因为输入层到隐藏层的激活函数是非线性函数，所以 error 函数是非凸函数，所以可能存在很多个局部极值点。

两栖动物(9219642) 20:07:40

你这里说的回归问题是用神经网络拟合随机变量？

网神(66707180) 20:07:56

是的，就是做线性回归要做的事。只是用神经网络的方法，可以拟合更复杂的曲线。这是我的理解。

梯度下降法的思路就是对 error 函数求导，然后按下式对参数作调整：

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

一直到导数为 0，就找到了极值点。

这是基本思路，实际上很多算法，如共轭梯度法，拟牛顿法等，可以更有效的寻找极值点。因为梯度下降是机器学习中求极值点的常用方法，这些算法不是本章的内容，就不讲了，有专门的章节讲这些算法。

这里要讲的是 error 函数对参数 w 的求导方法，也就是 back propagation 后向传导法。

可以高效的计算这个导数。

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}.$$

这个导数的计算利用的求导数的链式法则，

把 E 对 w 的求导，转换成 E 对 activation a 的求导和 a 对 w 的求导。这个地方因为涉及的式子比较多，我在纸上写了一下，我整个贴上来吧，大家看看，可以讨论一下：

Handwritten notes on a piece of paper showing the derivation of the backpropagation formula for the error derivative with respect to the weight w_{ji} .

Forward propagation:

$$a_j = \sum_i w_{ji} z_i \quad (1)$$
$$z_j = h(a_j) \quad (2)$$

error 函数对 w_{ji} 的偏导数:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \quad \text{因为 } (1) \quad \frac{\partial E_n}{\partial a_j} \cdot z_i = \delta_j \cdot z_i$$

其中 δ_j 是引入的一个符号, $\delta_j = \frac{\partial E_n}{\partial a_j} \quad (3)$

为了求 $\frac{\partial E_n}{\partial w_{ji}}$, 只要求 δ_j .

对 output unit, $\delta_j = y_j - t_j$.

对于 hidden units 的 δ_j , 利用链式法则 chain rule:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

将 $\frac{\partial E_n}{\partial a_k}$ 代入 δ_k :

$$\delta_j = \sum_k \delta_k \frac{\partial a_k}{\partial a_j}$$

将 $\frac{\partial a_k}{\partial a_j}$ 代入 $\frac{\partial}{\partial a_j} (\sum_i w_{ki} z_i)$:

$$\delta_j = \sum_k \delta_k \frac{\partial}{\partial a_j} (w_{k1} z_1 + w_{k2} z_2 + \dots + w_{kj} z_j + \dots + w_{km} z_m)$$
$$= \sum_k \delta_k \cdot w_{kj} \frac{\partial z_j}{\partial a_j} \quad \text{因为 } \frac{\partial}{\partial a_j} z_j = h'(a_j)$$
$$= \sum_k \delta_k \cdot w_{kj} h'(a_j)$$

阳阳(236995728) 20:21:30

error function 是负对数似然函数吧

网神(66707180) 20:21:38

是的

天上的月亮(785011830) 20:22:35

更新 hidden 层的 w，为什么利用偏导的 chain rule 呢？

网神(66707180) 20:22:49

最主要的结论就是：

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}.$$

对于输出层的节点，

$$\frac{\partial E_n}{\partial a_j} = y_k - t_k$$

对于隐藏层的节点，

$$\frac{\partial E_n}{\partial a_j} = h'(a_j) \sum_k w_{kj} \delta_k$$

E 对隐藏层节点 w 的导数，通过上式，就由输出层节点的导数 求得了。

有了上面求偏导数的方法，整个训练过程就是：

1.输入一个样本 X_n ，根据 forward propagation 得到每个 hidden 单元和 output 单元的激活值 a.

2.计算每个 output 单元的偏导数 δ_k ：

3.根据反向传导公式，计算每个 hidden 单元的 δ_j

4.计算偏导数。

5.用一定的步长更新参数 w。

循环往复，一直找到满意的 w。

独孤圣者(303957511) 20:32:09

BP 时，是不是每个样本都要反馈一次？

网神(66707180) 20:33:32

应该是每个样本都反馈一次，batch 也是 batch 形式的每个样本反馈一次，将每个样本的偏导数求和，如下式，得到所有样本的偏导数，然后做一次参数 w 的调整。

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}.$$

接下来就讲正则化，神经网络参数多，训练复杂，所以正则化方法也被研究的很多，书上讲了很多种的正则化方法。第一个，就是在 error 函数上，加上正则项。

第三章回归的正则项如下：

$$\tilde{E}(w) = E(w) + \frac{\lambda}{2} w^T w. \quad (5.112)$$

后面那一项对过大的 w 做惩罚。

但是对神经网络，正则项需要满足 一个缩放不变性，所以正则项的形式如下：

$$\tilde{E}(w) = E(w) + \frac{\lambda_1}{2} \sum_{w \in \mathcal{W}_1} w^2 + \frac{\lambda_2}{2} \sum_{w \in \mathcal{W}_2} w^2$$

这个形式怎么推导出来的，大家看书吧，这里不说了。

书上讲到的第二种正则化方法是 early stop，我理解的思路就是在训练集上每调整一次 w ，得到新的参数，就在验证集上验证一下。开始在验证集上的 error 是逐渐下降的，后来就不下降了，会上升。那么在开始上升之前，就停止训练，那个参数就是最佳情况。

忘了说一个正则化注意对象了，就是隐藏节点的数量 M 。这个 M 是预先人为确定的，他的大小决定了整个模型的参数多少。所以是影响 欠拟合还是过拟合的 关键因素。

monica(909117539) 20:42:38

开始上升是出现了过拟合么？节点数量和层数都是怎样选择的呀？

网神(66707180) 20:42:56

开始上升就是过拟合了，是的，在训练集上，error 还是下降的，但验证集上已经上升了。节点数量 M 是人为确定，我想这个数怎么定，是 NN 调参的重点。

monica(909117539) 20:44:36

:)

网神(66707180) 20:44:38

ML 牛的人，叫做 调得一手好参，我觉得这里最能体现。

这个图是不同的 M 值对拟合情况的影响：

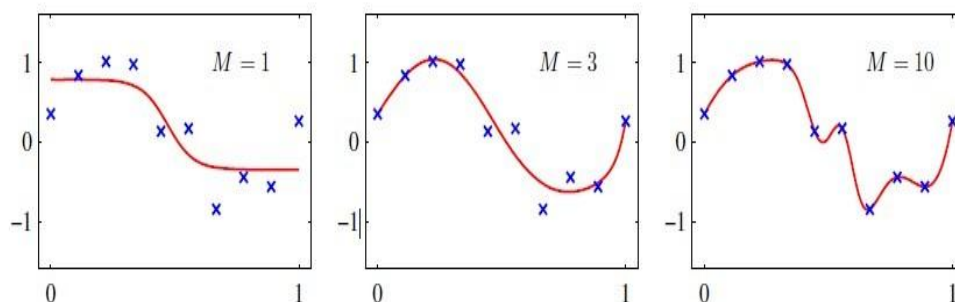


Figure 5.9 Examples of two-layer networks trained on 10 data points drawn from the sinusoidal data set. The graphs show the result of fitting networks having $M = 1, 3$ and 10 hidden units, respectively, by minimizing a sum-of-squares error function using a scaled conjugate-gradient algorithm.

另外，因为 NN 又不是凸函数，导致 train 更麻烦

monica(909117539) 20:47:01

嗯，这个能理解，前面模型太简单，欠拟合了，后面模型太复杂，就过拟合了。

网神(66707180) 20:47:08

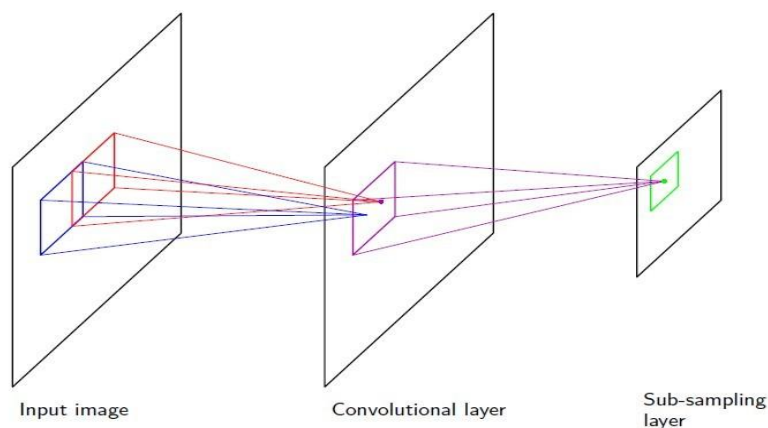
需要尝试不同的 M 值，对于每一个 M 值，又需要 train 多次，每次随即初始化 w 的值，然后尝试找到不同的局部极值点。

上面说了三种正则化方法。接下来，因为神经网络在图像识别中，应用广泛，所以图像识别对正则化有些特殊的需求，就是 平移、缩放、旋转不变性。不知道这个需求在其他应用中，是否存在，例如 NLP 中，反正书上都是以图像处理为例讲的。

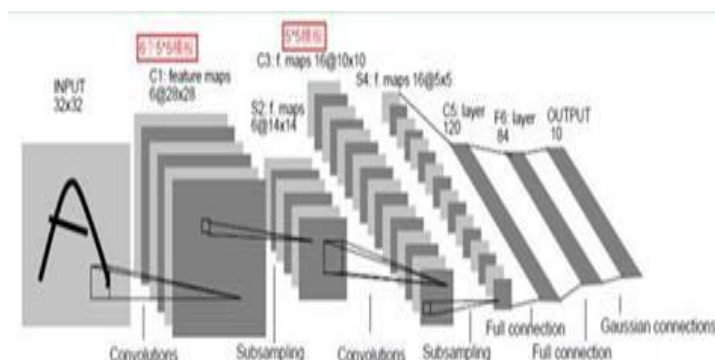
这些不变形，就是图像中的物体位置移动、大小变化、或者一定程度的旋转，对 output 层的结果应该没有影响。

这里主要讲卷积神经网络。这个东东好像很有用，我看在 Ng 的 deep learning 教程中也重点讲了，当图像是大图时，如 100x100 像素以上，卷积网络可以大大减少参数 w 的个数。

卷积网络的 input 层的 unit 是图像的像素值。卷积网络的 hidden 层由卷积层和子采样层组成，如图：



一个卷积网络可能包括多个卷积层和子采样层，如下图有两对卷积/子采样层：



卷积层的 units 被划分成多个 planes，每个 plane 是一个 feature map，一个 feature map 里的一个 unit 只连接 input 层的一个小区域，一个 feature map 里的所有 unit 使用相同的 weight 值。

卷积层的作用可以认为是提取特征。每个 plane 就是一个特征滤波器，通过卷积的方式提取图像的一种特征，过滤掉不是本特征的信息。比如提取 100 种特征，就会有 100 个 plane，上图中，提取 6 个特征，第二层有 6 个平面。

一个 plane 里面的一个 unit 与图像的一个子区域相连，unit 的值表示从该区域提取的一个特征值。共有多少个子区域，plane 里就有多少个 unit。

以上图为例，输入图像是 32x32 像素，所以 input 层有 32x32 个 unit。取子区域的大小为 4x4，那么共有 28x28 个子区域。每个子区域提取得到 6 个特征，对应第二层有 6 个 plane，每个 plane 有 28x28 个 unit。

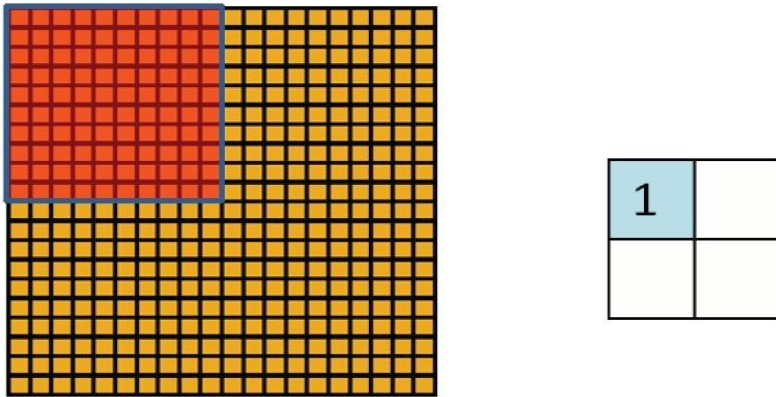
这样一个 plane 里的一个 unit 与输入层的 4x4 个 unit 相连，有 16 个权重 weight 值，这个 plane 里的其他 unit 都公用 16 个 weight 值。

这样，6 个平面，共有 $16 \times 6 = 96$ 个 weight 参数和 6 个 bias 参数。

下面说子采样层，上面说道，6 个平面，每个平面有 28x28 个 unit，所以第二层共有 $6 \times 28 \times 28$ 个 unit。这是从图像中提取出来的特征，作为后一层的输入，用于分类或回归。

但实际中，6 个特征远远不够，图像也不一定只有 32x32 像素。假如从 96x96 像素中提取 400 个特征，子区域还是 4x4，则第二层的 unit 会有 $400 \times (96-4) \times (96-4) = 300$ 多万。超过 300 多万输入的分类器很难训练，也容易过拟合。

这就通过子采样来减少特征。子采样就是把卷积层的一个平面划分成几部分，每个部分取最大值或平均值，作为子采样层的一个 unit 值。如下图：



左边表示卷积层的一个 plane，右边是这个 plane 的子采样结果。一个格子表示一个 unit。
把左边整个 plane 分成 4 部分，互相不重合，每个部分取最大值或平均值，作为右边子采样层的一个 unit。这样子采样层一个 plane 只有 4 个 unit。

通过卷积和子采样：得到的特征作为下一层分类或回归的输入。

主要好处：

- 这样输入层的各种变形在后面就会变得不敏感。如果有多对 卷积/子采样，每一对就将不变形更深入一层。
- 减少参数的个数。就这么多了，大家慢慢看。

阳阳(236995728) 21:01:30

关于卷积神经网络的材料有吗？@网神

阿邦(1549614810) 21:01:44

问个问题，cnn 怎么做的 normalization？

网神(66707180) 21:03:26

卷积网络的材料，我看了书上的，还有 Andrew Ng 的 deep learning 里讲的，另外网上搜了一篇文章
这里我贴下链接。

罗杰兔(38900407) 21:04:19

不大明白为什么可以采样，怎么保证采样得到的信息正好是我们需要的

阿邦(1549614810) 21:05:01

采样是 max pooling，用于不变性

网神(66707180) 21:05:02

<http://ibillxia.github.io/blog/2013/04/06/Convolutional-Neural-Networks/>

牧云(1106207961) 21:05:02

采样是随机的吗？

网神(66707180) 21:06:40

传了个 deep learning 教程。这是 Andrew Ng 上课的 notes, 网上一些人众包翻译的, 我觉得翻译的不错。
里面讲了神经网络、卷积网络，和其他深度网络的东西。

采样不是随机的。

牧云(1106207961) 21:08:04

卷积提取的特征具有稳定性吗

阿邦(1549614810) 21:08:56

据说要数据量很大才可以

网神(66707180) 21:09:03

卷积层和子采样层主要是 解决 不变性问题 和减少参数 w 数量问题，所以可以起到泛化作用。

第六章 Kernel Methods

主讲人 网络上的尼采

(新浪微博:@Nietzsche_复杂网络机器学习)

网络上的尼采(813394698) 9:16:05

今天的主要内容：Kernel 的基本知识，高斯过程。边思考边打字，有点慢，各位稍安勿躁。

机器学习里面对待训练数据有的是训练完得到参数后就可以抛弃了，比如神经网络；有的是还需要原来的训练数据比如 KNN，SVM 也需要保留一部分数据--支持向量。

很多线性参数模型都可以通过 dual representation 的形式表达为核函数的形式。所谓线性参数模型是通过非线性的基函数的线性组合来表达非线性的东西 模型还是线性的。比如线性回归模型是 $y = \mathbf{w}^T \phi(\mathbf{x}_n)$ ， $\phi(\mathbf{x}_n)$ 是一组非线性基函数，我们可以通过线性的模型来表达非线性的结构。

核函数的形式： $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ ，也就是映射后高维特征空间的内积可以通过原来低维的特征得到。因此 kernel methods 用途广泛。

核函数有很多种，有平移不变的 stationary kernels $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ ，还有仅依赖欧氏距离的径向

基核： $k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$

非线性转化为线性的形式的好处不言而喻，各种变换推导、闭式解就出来了。下面推导下线性回归模型的 dual representation，有助于我们理解核函数的作用：

根据最小二乘，我们得到下面的目标函数 $J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$ ，加了 L2 正则。

我们对 $\mathbf{w}$ 求导，令 $J(\mathbf{w})$ 的梯度等于 0，得到以下解：

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\} \phi(\mathbf{x}_n) = \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}$$

Φ 是个由基函数构成的样本矩阵，向量 $\mathbf{a}$ 里面的元素由 $a_n = -\frac{1}{\lambda} \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}$ 组成：

where Φ is the design matrix, whose n^{th} row is given by $\phi(\mathbf{x}_n)^T$. Here the vector $\mathbf{a} = (a_1, \dots, a_N)^T$, and we have defined

$$a_n = -\frac{1}{\lambda} \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}. \quad (6.4)$$

我们把 $\mathbf{w} = \Phi^T \mathbf{a}$ 代入最初的 $J(\mathbf{w})$ 得到：

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi^T \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \Phi \Phi^T \mathbf{a}$$

咱们用核矩阵 K 来替换 $\Phi \Phi^T$ ，其中矩阵 K 里面的元素是 $K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$

于是得到 $J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K} \mathbf{a}$.

然后 $J(\mathbf{a})$ 对 $\mathbf{a}$ 求导，令其梯度等于 0，得到解 $\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$.

所以原来的线性回归方程就变成了 $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \Phi \phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$

$K(\mathbf{x})$ 的含义：we have defined the vector $\mathbf{k}(\mathbf{x})$ with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$ ，上面的 DUAL 形式的含义非常明显，就是根据已知的训练数据来做预测。至此原来线性回归方程的参数 $\mathbf{w}$ 消失，由核函数来表示回归方程，以上方式把基于特征的学习转换成了基于样本的学习。这是线性回归的 DUAL 表示，svm 等很多模型都有 DUAL 表示。

80(850639048) 10:09:50

professor 核函数其实是为了求基函数的内积对吗？

网络上的尼采(813394698) 10:12:57

如果有很多基的话维度势必会很高，计算内积的花销会很大，有些是无限维的，核函数能绕过高维的内积计算，直接用核函数得到内积。

接下来看下核函数的性质及构造方法。核函数的一般形式：

$$k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^M \phi_i(x) \phi_i(x')$$

where $\phi_i(x)$ are the basis functions.

下面是个简单的例子说明为什么 $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$ 是个核函数：

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T \mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\ &= \phi(\mathbf{x})^T \phi(\mathbf{z}). \end{aligned}$$

很明显 $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$ 是个核函数，它能写成核

函数的一般形式。

核函数的一个充分必要定理也就是 mercer 定理：核矩阵是半正定的：

More generally, however, we need a simple way to test whether a function constitutes a valid kernel without having to construct the function $\phi(\mathbf{x})$ explicitly. A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel (Shawe-Taylor and Cristianini, 2004) is that the Gram matrix $\mathbf{K}$, whose elements are given by $k(\mathbf{x}_n, \mathbf{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathbf{x}_n\}$. Note that a positive semidefinite matrix is not the same thing as a matrix whose elements are nonnegative.

我们可以通过以下规则用简单的核函数来构造复杂的核函数：

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

过会我们讲高斯过程时再举个核函数线性组合的例子。

介绍一个经常用到的径向基核函数，高斯核： $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2)$ ，这个核函数能把数据映射到无限维的空间：

omitted. We can see that this is a valid kernel by expanding the square

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^T \mathbf{x} + (\mathbf{x}')^T \mathbf{x}' - 2\mathbf{x}^T \mathbf{x}'$$

to give

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^T \mathbf{x}/2\sigma^2) \exp(\mathbf{x}^T \mathbf{x}'/\sigma^2) \exp(-(\mathbf{x}')^T \mathbf{x}'/2\sigma^2)$$

中间 $\exp(\mathbf{x}^T \mathbf{x}'/\sigma^2)$ 可以展开成无限维的，然后核函数可以表示成内积的形式。

内积的含义就是表示相似性，所以核函数还有其他的用法。比如我们可以通过生成模型来构造核。

Given a generative model $p(\mathbf{x})$ we can define a kernel by

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}').$$

两个变量的概率都很高相似性就越大，其实这样做就是映射到一维的内积。

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i).$$

我们可以引入离散的隐变量：

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{x}'|\mathbf{z})p(\mathbf{z}) d\mathbf{z}$$

连续的隐变量：

举个这样做有啥用的例子，我们可以用来比较 HMM 由同一条隐马尔科夫链生成的两条序列的相似性：

Now suppose that our data consists of ordered sequences of length L so that an observation is given by $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$. A popular generative model for sequences is the hidden Markov model, which expresses the distribution $p(\mathbf{X})$ as a marginalization over a corresponding sequence of hidden states $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_L\}$. We can use this approach to define a kernel function measuring the similarity of two sequences $\mathbf{X}$ and $\mathbf{X}'$ by extending the mixture representation (6.29) to give

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z}) \quad (6.31)$$

so that both observed sequences are generated by the same hidden sequence $\mathbf{Z}$. This model can easily be extended to allow sequences of differing length to be compared.

网络上的尼采(813394698) 10:40:34

接下来讲我们今天的重点 Gaussian Processes

牧云(1106207961) 10:41:02



数据海洋(1009129701) 10:41:14

我先再理解，理解这些。

网络上的尼采(813394698) 10:42:41

Gaussian Processes 是贝叶斯学派的一个大杀器，用处很广。不同于参数模型，Gaussian Processes 认为函数在函数空间里存在一个先验分布。

高斯过程和很多模型是等价的：ARMA (autoregressive moving average) models, Kalman filters, radial basis function networks，还有特定情况下的神经网络。

现在我们从贝叶斯线性回归自然的引出高斯过程：

前面我们提到的线性回归的形式 $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$

贝叶斯方法为参数加了一个高斯分布 $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$

大家发现了没有，这样做直接导致了函数有个预测分布，并且也是高斯的，因为方程是线性的并且参数是高斯分布。线性的东西和高斯分布总是不分家的。

我们定义向量 $\mathbf{y}$ ：

$\mathbf{x}_1, \dots, \mathbf{x}_N$. We are therefore interested in the joint distribution of the function values $y(\mathbf{x}_1), \dots, y(\mathbf{x}_N)$, which we denote by the vector $\mathbf{y}$ with elements $y_n = y(\mathbf{x}_n)$ ， $\mathbf{y}$ 就是个多元的高斯分布。

它的每一维 $y(\mathbf{x}_n)$ 都是个高斯分布，这也是高斯过程的由来。

$\mathbf{y}$ 可以表示为 $\mathbf{y} = \Phi \mathbf{w}$

where Φ is the design matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$

Φ 是基函数组成的样本矩阵。

高斯过程可以由均值和协方差矩阵完全决定。由于 $\mathbf{w}$ 的均值是 0，所以我们也认为高斯过程的均值是 0，剩下的就是根据定义求它的协方差矩阵，很自然地就得出来了：

$$\text{cov}[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = \Phi \mathbb{E}[\mathbf{w}\mathbf{w}^T] \Phi^T = \frac{1}{\alpha} \Phi \Phi^T = \mathbf{K}$$

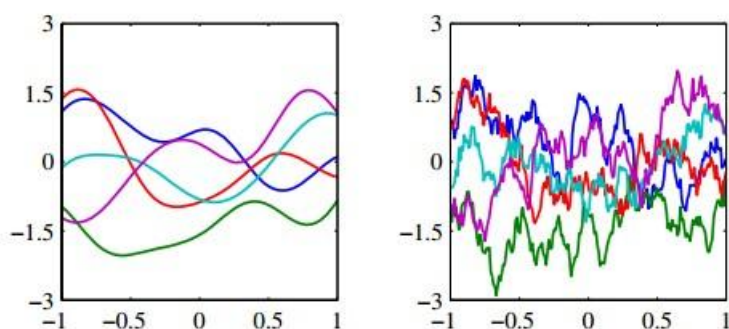
where $\mathbf{K}$ is the Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

矩阵 $\mathbf{K}$ 里的元素 $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$ 都是核函数的形式。

选用什么样的核函数也是问题，下面的图是对采用高斯核和指数核的高斯过程的取样，一共取了 5 条，可以看到两者的区别：

Figure 6.4 Samples from Gaussian processes for a 'Gaussian' kernel (left) and an exponential kernel (right).



接下来我们就用 GP 来做回归：

我们观测的目标值是包含噪音的，噪音是高斯分布。

$$t_n = y_n + \epsilon_n$$

那么根据线性高斯模型的性质， $p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$ ，其中 β 是噪音的参数

对于向量 $\mathbf{t} = (t_1, \dots, t_N)^T$ 和向量 $\mathbf{y} = (y_1, \dots, y_N)^T$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N)$$

咱们前面说过了， $p(\mathbf{y})$ 可以表示为 $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$ 。

所以 marginal distribution：
$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

其中矩阵 $\mathbf{C}$ 的元素 $C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}$ ， δ_{nm} 是单位矩阵的元素，其实就是把 β^{-1}

加在了矩阵 $\mathbf{K}$ 的对角线上。这个不难理解，一开始 $t_n = y_n + \epsilon_n$ ，都是高斯的，协方差是两者的相加，噪音每次取都是独立的，所以只在协方差矩阵对角线上有。

现在确定下用什么核的问题，举个例子，下面这个核函数用了高斯核，线性核，以及常数的线性组合，这样做是为了更灵活，过会再讲如何确定里面的这些超参：

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp \left\{ -\frac{\theta_1}{2} \|\mathbf{x}_n - \mathbf{x}_m\|^2 \right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m$$

下图是不同的超参对高斯过程的影响：

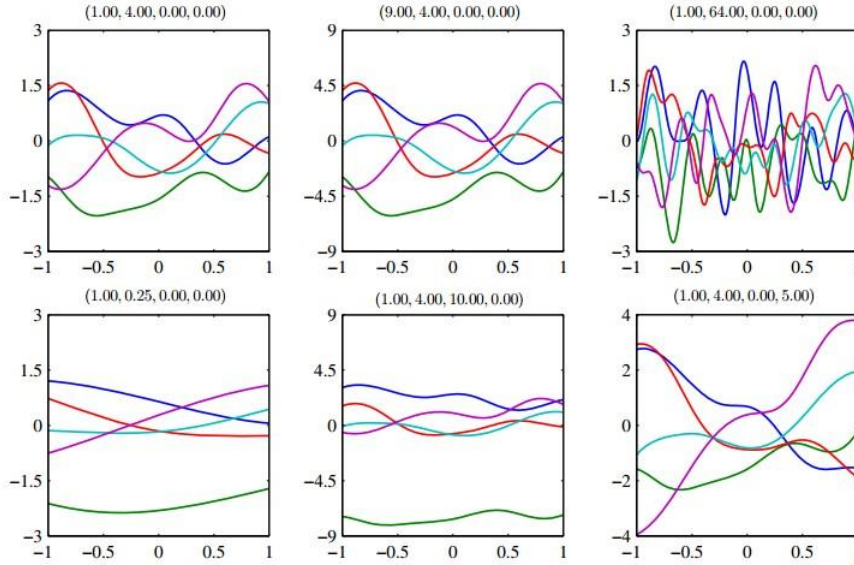


Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

解决了核函数的问题，我们再回来，通过前面的结论，不难得出 $p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$

如何确定矩阵 $\mathbf{C}_{N+1}$ 呢，其实我们在原来矩阵 $\mathbf{C}_N$ 的基础上补上就行。

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix}$$

$\mathbf{k}$ 和 c 比较容易理解：

the vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \dots, N$,

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

咱们的最终目标就是得 $p(t_{N+1} | \mathbf{t})$ ，由于这两个都是高斯分布，用第二章条件高斯分布的公式套一下，其中均值是 0：

From these we obtain the following expressions for the mean and covariance of the conditional distribution $p(\mathbf{x}_a | \mathbf{x}_b)$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} (\mathbf{x}_b - \boldsymbol{\mu}_b) \quad (2.81)$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} \boldsymbol{\Sigma}_{ba}. \quad (2.82)$$

就会得到 $p(t_{N+1} | \mathbf{t})$ 的均值和方差：

ditional distribution $p(t_{N+1} | \mathbf{t})$ is a Gaussian distribution with mean and covariance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \quad (6.67)$$

可以看出均值和方差都是 $\mathbf{x}_{N+1}$ 的函数，我们做预测时用均值就行了。

最后一个问题就是高斯过程是由它的协方差矩阵完全决定的，我们如何学习矩阵里面的超参呢？包括我们刚才提到的核函数里面的参数以及噪音的参数。

其实由于高斯分布的原因，我们可以方便的利用 log 最大似然：

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \ln |\mathbf{C}_N| - \frac{1}{2} \mathbf{t}^T \mathbf{C}_N^{-1} \mathbf{t} - \frac{N}{2} \ln(2\pi)$$

求最优解时可以用共轭梯度等方法，梯度：

$$\frac{\partial}{\partial \theta_i} \ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \text{Tr} \left(\mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta_i} \right) + \frac{1}{2} \mathbf{t}^T \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta_i} \mathbf{C}_N^{-1} \mathbf{t}.$$

到这里，用高斯过程做回归就结束了。

有了做回归的基础，咱们再看下如何做分类。

类似逻辑回归，加个 sigmoid 函数就能做分类了：

$$p(\mathbf{x}) = \frac{1}{1 + e^{-a(\mathbf{x})}}$$

where $a(\mathbf{x})$ comes from a Gaussian process

分类与回归不同的是 $p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}$ ，是个伯努利分布。

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$$

这里还和前面一样。

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu \delta_{nm}$$

对于二分类问题，最后我们要得到是：

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1}$$

但是这个积分是不容易求的， $\mathbf{t}_N$ 是伯努利分布， a_{N+1} 是高斯分布，不是共轭的。求积分的方法有很多，

可以用 MCMC，也可以用变分的方法。书上用的是 Laplace approximation。

今天就到这里吧，我去吃饭，各位先讨论下吧。

上面 Gaussian Processes 的公式推导虽然有点多，但都是高斯分布所以并不复杂，并且 GP 在算法实现上也不难。

另外给大家推荐一个机器学习视频的网站，

<http://blog.videolectures.net/100-most-popular-machine-learning-talks-at-videolectures-net/>

里面有很多牛人比如 Jordan 的 talks，第一个视频就是剑桥的 David MacKay 讲高斯过程，他的一本书 Information Theory, Inference and Learning Algorithms 也很出名。

两栖动物(9219642) 14:35:09

$$\Phi$$
 是个由基函数构成的矩阵，向量 $\mathbf{a}$ 里面的元素由 $a_n = -\frac{1}{\lambda} \{ \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) - t_n \}$ 组成。

Φ 的维度是基函数的个数， $\mathbf{a}$ 的维度是样本的个数把？

网络上的尼采(813394698) 14:36:44

对

两栖动物(9219642) 14:37:48

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \} \phi(\mathbf{x}_n) = \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}$$

哪这 2 个怎么后来乘在一起了？维度不是不一样吗？

网络上的尼采(813394698) 14:49:01

@两栖动物 Φ 不是方阵，可以相乘

Φ is the design matrix, whose n^{th} row is given by $\phi(\mathbf{x}_n)^T$. 明白了吧，另外这个由基函数表示的样本

矩阵只在推导里存在。

两栖动物(9219642) 14:56:08

明白了，谢谢

第七章 Sparse Kernel Machines

主讲人 网神

(新浪微博: @豆角茄子麻酱凉面)

网神(66707180) 18:59:22

大家好，今天一起交流下 PRML 第 7 章。第六章核函数里提到，有一类机器学习算法，不是对参数做点估计或求其分布，而是保留训练样本，在预测阶段，计算待预测样本跟训练样本的相似性来做预测，例如 KNN 方法。

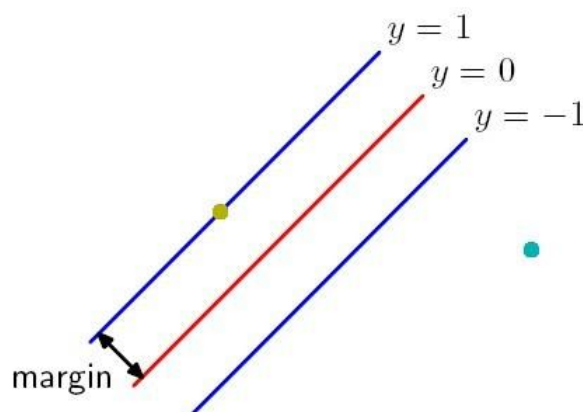
将线性模型转换成对偶形式，就可以利用核函数来计算相似性，同时避免了直接做高维度的向量内积运算。本章是稀疏向量机，同样基于核函数，用训练样本直接对新样本做预测，而且只使用了少量训练样本，所以具有稀疏性，叫 sparse kernel machine。

本章包括 SVM 和 RVM(relevance vector machine)两部分，首先讲 SVM，支持向量机。首先看 SVM 用于二元分类，并先假设两类数据是线性可分的。

二元分类线性模型可以用这个式子表示： $y(x) = \mathbf{w}^T \phi(x) + b$ 。其中 $\phi(x)$ 是基函数，这些都跟第三章和第四章是一样的。

两类数据线性可分，当 $y(x_i) > 0$ 时，分类结果是 $t_i = +1$ ； $y(x_i) < 0$ 时，分类结果 $t_i = -1$ ；也就是对所有训练样本总是有 $t_i y(x_i) > 0$ 。要做的就是确定决策边界 $y(x) = 0$

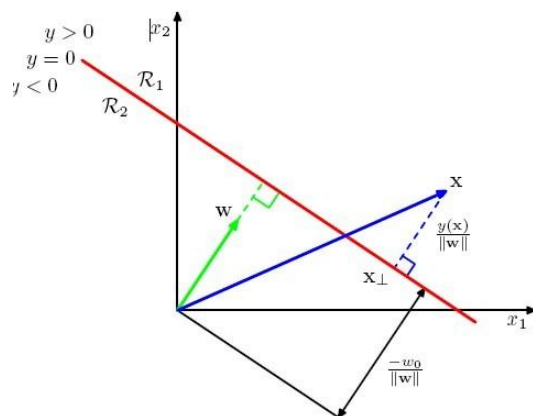
为了确定决策边界 $y(x) = \mathbf{w}^T \phi(x) + b$ ，SVM 引入 margin 的概念。margin 定义为决策边界 $y(x)$ 到最近的样本的垂直距离。如下图所示：



SVM 的目标是寻找一个 margin 最大的决策边界。我们来看如何确定目标函数：

首先给出一个样本点 x 到决策边界 $\mathbf{w}^T \phi(x) + b = 0$ 的垂直距离公式是什么，先给出答案： $|y(x)|/||\mathbf{w}||$

这个距离怎么来的，在第四章有具体介绍。看下图：



图例，我们看点 x 到 $y=0$ 的距离 r 是多少：

$x_{\perp}$ 是 x 在 $y=0$ 上的投影，因为 w 跟 $y=0$ 是垂直的，所以 $\overrightarrow{x_{\perp}x}$ 跟 w 平行， $\overrightarrow{x_{\perp}x} = r \frac{w}{\|w\|}$ 。

r 是距离。根据向量相加的公式，有 $x = x_{\perp} + r \frac{w}{\|w\|}$ 。两边都乘上 w^T 并加上 b ，得到

$$y(x) = y(x_{\perp}) + r \|w\|, \text{ 因为 } y(x_{\perp}) = 0, \text{ 所以 } r = \frac{y(x)}{\|w\|}.$$

上面我们得到了任意样本点 x 到 $y(x)=0$ 的距离，要做的是最大化这个距离。

同时，要满足条件 $t_n y(x_n) > 0$

所以目标函数是：
$$\arg \max_{w, b} \left\{ \min_n \left[\frac{t_n (w^T \phi(x_n) + b)}{\|w\|} \right] \right\}$$

求 w 和 b ，使所有样本中，与 $y=0$ 距离最小的距离 最大化，整个式子就是最小距离最大化
这个函数优化很复杂，需要做一个转换

可以看到，对 w 和 b 进行缩放， $w \rightarrow \kappa w$ and $b \rightarrow \kappa b$ ，距离 $\frac{t_n y(x_n)}{\|w\|}$ 并不会变化

根据这个属性，调整 w 和 b ，使到决策面最近的点满足： $t_n (w^T \phi(x_n) + b) = 1$

从而左右样本点都满足 $t_n (w^T \phi(x_n) + b) \geq 1$

这样，前面的目标函数可以变为：
$$\xi \arg \max_{w, b} \frac{1}{\|w\|} \quad (\text{即 } \arg \min_{w, b} \frac{1}{2} \|w\|^2)$$

同时满足约束条件： $t_n (w^T \phi(x_n) + b) \geq 1$

这是一个不等式约束的二次规划问题，用拉格朗日乘子法来求解

构造如下的拉格朗日函数：
$$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^N a_n \{t_n (w^T \phi(x_n) + b) - 1\}$$

a_n 是拉格朗日乘子，这个函数分别对 w 和 b 求导，令导数等于 0，可以得到 w 和 b 的表达式：

$$w = \sum_{n=1}^N a_n t_n \phi(x_n)$$

$$0 = \sum_{n=1}^N a_n t_n.$$

将 w 带入前面的拉格朗日函数 $L(w,b,a)$ ，就可以消去 w 和 b ，变成 a 的函数 $\tilde{L}(a)$ ，这个函数是拉格朗日函数的对偶函数：

$$\tilde{L}(a) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(x_n, x_m)$$

为什么要转换成对偶函数，主要是变形后可以借助核函数，来解决线性不可分的问题，尤其是基函数的维度特别高的情况。求解这个对偶函数，得到参数 a_n ，就确定了分类模型

把 $w = \sum_{n=1}^N a_n t_n \phi(x_n)$ 带入 $y(x) = w^T \phi(x) + b$ ，就是用核函数表示的分类模型：

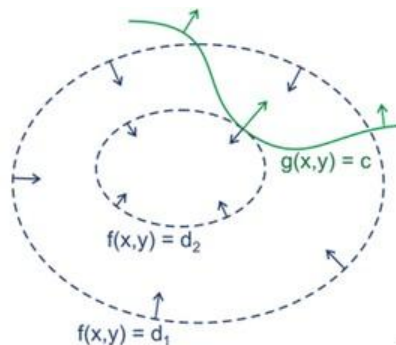
$$y(x) = \sum_{n=1}^N a_n t_n k(x, x_n) + b$$

这就是最终的分类模型，完全由训练样本 x_n ， $n=1 \dots N$ 决定。

SVM 具有稀疏性，这里面对大部分训练样本， a_n 都等于 0，从而大部分样本在新样本预测时都不起作用。

我们来看看为什么大部分训练样本， a_n 都等于 0。这主要是由 KKT 条件决定的。我们从直观上看下 KKT 条件是怎么回事：

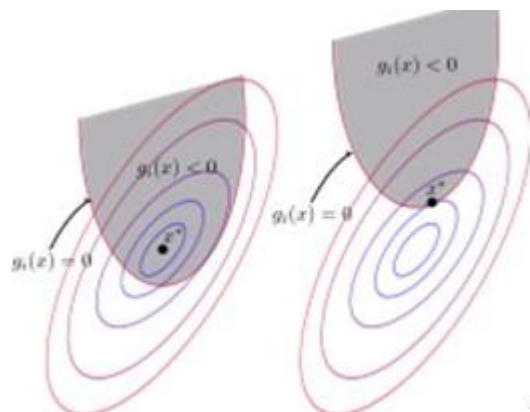
KKT 是对拉格朗日乘子法的扩展，将其从约束为等式的情况扩展为约束为不等式的情况。所以先看约束为等式的情况：例如求函数 $f(x_1, x_2)$ 的极大值，同时满足约束 $g(x_1, x_2)=0$ ，拉格朗日乘子法前面已经介绍，引入拉式乘子，构造拉式函数，然后求导，解出的值就是极值。这里从直观上看一下，为什么这个值就是满足条件的极值。设想取不同的 z 值，使 $f(x_1, x_2)=z$ ，就可以得到 $f(x_1, x_2)$ 的不同等高线，如图：



$g(x_1, x_2)=0$ 构成图中的曲线，图中标记的 $g=c$ ，对于这种情况，改成 $g-c=0$ 就可以了。假设 g 与 f 的某

些等高线相交，交点就是同时满足约束条件和目标函数的值，但不一定是极大值。。有两种相交形式，一种是穿过，一种是相切。因为穿过意味着在该条等高线内部还存在着其他等高线与 g 相交，新等高线与目标函数的交点的值更大。只有相切时，才可能取得最大值。因此，在极大值处， f 的梯度与 g 的梯度是平行的，因为梯度都垂直于 g 或 f 曲线，也就是存在 λ ，使得 $\nabla f + \lambda \nabla g = 0$ ，这个式子正是拉格朗日函数 $L(x, \lambda) = f(x) + \lambda g(x)$ 对 x 求导的结果。

接下来看看约束条件为不等式的情况，例如约束为 $g(x) \geq 0$ ，先看个图：



图里的约束是 $g < 0$ ，不影响解释 KKT 条件。不等式约束分两种情况，假设极值点是 x^* ，当 $g(x^*) > 0$ 时，也就是图中左边那部分，此时该约束条件是 inactive 的，对于极值点的确定不起作用。因此拉格朗日函数 $L(x, \lambda) = f(x) + \lambda g(x)$ 中， λ 等于 0，极值完全由 f 一个人确定，相当于 λ 等于 0。当 $g(x^*) = 0$ 时，也就是图中右边部分，极值出现在 g 的边界处，这跟约束条件为等式时是一样的。

总之，对于约束条件为不等式的拉格朗日乘子法，总有 $\lambda g(x) = 0$ ，不是 λ 等于 0，就是 $g = 0$ 。这个结论叫 KKT 条件，总结起来就是：

$$g(x) \geq 0$$

$$\lambda \geq 0$$

$$\lambda g(x) = 0$$

再返回来看 SVM 的目标函数构造的拉格朗日函数：

$$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^N a_n \{t_n (w^T \phi(x_n) + b) - 1\}.$$

根据 KKT 条件，有 $a_n \geq 0$ ， $t_n y(x_n) - 1 \geq 0$ ， $a_n \{t_n y(x_n) - 1\} = 0$ ，所以对于 $t_n y(x_n)$ 大于 1 的那些样本点，其对应的 a_n 都等于 0。只有 $t_n y(x_n)$ 等于 1 的那些样本点对保留下来，这些点就是支持向量。

这部分大家有什么意见和问题吗？

=====讨论=====

Fire(564122106) 20:01:56

他为什么要符合 KKT 条件啊

网神(66707180) 20:02:33

因为只有符合 KKT 条件，才能有解，否则拉格朗日函数没解，我的理解是这样的

Fire(564122106) 20:03:54

我上次看到一个版本说只有符合 KKT 条件 对偶解才和原始解才相同，不知道怎么解释。

kxkr<lxfkxkr@126.com> 20:04:18

貌似统计学习方法 附录里面 讲了这个

Wolf <wuwjia@foxmail.com> 20:04:19

an 为 0 为什么和 kkt 条件相关

kxkr<lxfkxkr@126.com> 20:04:20

不过忘记了，我上次看到一个版本说只有符合 KKT 条件，对偶解才和原始解相同。

YYKuaiXian(335015891) 20:04:40

Ng 的讲义就是用这种说法

苦瓜炒鸡蛋(852383636) 20:04:47

因为大部分的样本都不是 sv

Wolf <wuwjia@foxmail.com> 20:05:05

如果两类正好分布在 margin 上，那么所有的点都是 sv

YYKuaiXian(335015891) 20:05:40

Under our above assumptions, there must exist w^*, α^*, β^* so that w^* is the solution to the primal problem, α^*, β^* are the solution to the dual problem, and moreover $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$. Moreover, w^*, α^* and β^* satisfy the **Karush-Kuhn-Tucker (KKT) conditions**, which are as follows:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, n \quad (3)$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, l \quad (4)$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \dots, k \quad (5)$$

$$g_i(w^*) \leq 0, \quad i = 1, \dots, k \quad (6)$$

$$\alpha^* \geq 0, \quad i = 1, \dots, k \quad (7)$$

Moreover, if some w^*, α^*, β^* satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

Wolf <wuwjia@foxmail.com> 20:05:54

只有符合 kkt 条件，primary 问题和 due 问题的解才是一样的，否则胖子里面的瘦子总比瘦子里面的胖子大。

kxkr<lxfkxkr@126.com> 20:07:03

这个比喻 好！

高老头(1316103319) 20:07:08

对偶问题和原问题是什么关系，一个问题怎么找到它的对偶问题？

Wolf <wuwjia@foxmail.com> 20:07:19

所以 kkt 条件和 sv 为什么大部分为 0 没有直接关系，sv 为 0 个人觉得是分界面的性质决定的，分界面是一个低维流形。

Fire(564122106) 20:08:38

我也感觉 sv 是和样本数据性质有关的

Wolf <wuwjia@foxmail.com> 20:09:26

比如在二维的时候，分界面是一个线性函数，导致 sv 比较少，当投影到高维空间，分界面变成了一个超平

面，导致 sv 变多了，另外，很多样本变成 sv 也是 svm 慢的一个原因。

网神(66707180) 20:09:37

sv 本质上是 svm 选择的错误函数决定的，在正确一边分类边界以外的样本点，错误为 0，在边界以内或在错误一边，错误大于 0。

苦瓜炒鸡蛋(852383636) 20:11:04

sv 确定的超平面 而非是超平面确定的 sv

Wolf <wuwjia@foxmail.com> 20:11:30

sv 确定的超平面 而非是超平面确定的 sv，一样的，hinge 为什么会稀疏？什么样的优化问题才有对偶问题，我也在疑问。对于一些规划问题（线性规划，二次规划）可以将求最大值（最小值）的问题转化为求最小值最大值的问题。

网神(66707180) 20:12:24

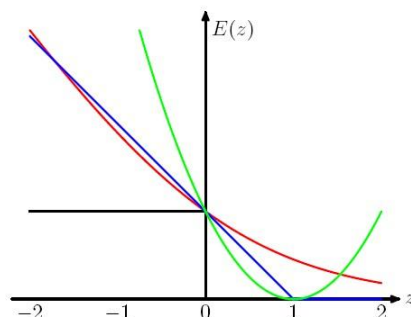
kkt 是从一个侧面解释稀疏，从另一个侧面，也就是错误函数是 hinge 函数，也可以得出稀疏的性质。svm 跟逻辑回归做对比，hinge 损失导致稀疏，我们先讲下这吧，svm 的错误函数可以这么写：

$$\sum_{n=1}^N E_{SV}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

其中 $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$

where $[\cdot]_+$ denotes the positive part

这就是 hinge 错误函数，图形如图中的蓝色线



而逻辑回归的错误函数是：

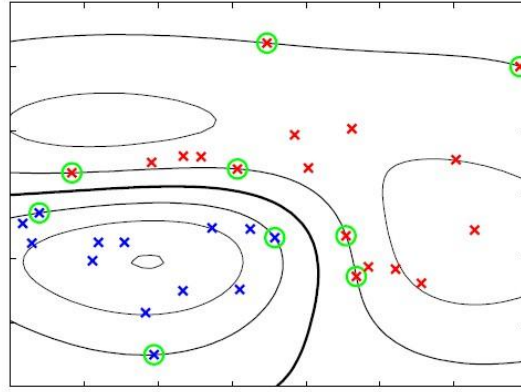
$$\sum_{n=1}^N E_{LR}(y_n t_n) + \lambda \|\mathbf{w}\|^2.$$

$$E_{LR}(yt) = \ln(1 + \exp(-yt)).$$

如图中的红色线，红色线跟蓝色线走势相近，区别是 hinge 函数在 $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$ 图中 $z > 1$ 时，错误等于 0，也就是 $yt > 1$ 的那些点都不产生损失。这个性质可以带来稀疏的解。

=====讨论结束=====

我接着讲了，后面还有挺多内容，刚才说的都是两类训练样本可以完全分开的情况，比如下面这个图，采用了高斯核函数的支持向量机，可以很清楚的看到决策边界，支持向量：



但实际中两类数据的分布会有重叠的情况，另外也有噪音的存在，导致两类训练数据如果一定要完全分开，泛化性能会很差。因此 svm 引入一些机制，允许训练时一些样本被误分类。我们要修改目标函数，允许样本点位于错误的一边，但会增加一个惩罚项，其大小随着数据点到边界的距离而增大这个惩罚项叫松弛变量，slack variables，记为 ξ_n ，并且大于等于 0。其中下标 $n=1, \dots, N$ ，也就是每个训练样本对应一个 ξ_n ，

对于位于正确的 margin 边界上或以内的数据点，其松弛变量 $\xi_n=0$ ，其他样本点 $\xi_n = |t_n - y(x_n)|$

这样，如果样本点位于决策边界 $y(x)=0$ 上， $\xi_n=1$

如果被错分，位于错误的一边， $\xi > 1$ ，因此目标函数的限制条件由 $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1$ 修改为

$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$ ，因此目标函数的限制条件由 $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1$ 修改为

$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$ ，目标函数修从最小化 $\frac{1}{2} \|\mathbf{w}\|^2$ 改为最小化 $C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$ ，其中参数 C 用

于控制松弛变量和 margin 之间的 trade-off，因为对于错分的点，有 $\xi > 1$ ，所以 $\sum \xi_n$ 是错分样本数的一个上限 upper bound，所以 C 相当于一个正则稀疏，控制着最小错分数和模型复杂度的 trade-off.

SVM 在实际使用中，需要调整的参数很少，C 是其中之一。

看这个目标函数，可以看到，C 越大，松弛变量就越倍惩罚，就会训练出越复杂的模型，来保证尽量少的样本被错分。当 C 趋于无穷时，每个样本点就会被模型正确分类。

我们现在求解这个新的目标函数，加上约束条件，拉格朗日函数如下：

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n$$

其中 a_n 和 μ_n 是拉式乘子，分别对 $\mathbf{w}$, b 和 $\{\xi_n\}$ 求导，令导数等于 0，得到 $\mathbf{w}$, b , ξ_n 的表示，带入 $L(\mathbf{w}, b, \mathbf{a})$ ，

消去这些变量，得到以拉格朗日乘子为变量的对偶函数：

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

新的对偶函数跟前面对偶函数形式相同，只有约束条件有不同。这就是正则化的 SVM。

接下来提一下对偶函数的解法，对偶函数都是二次函数，而且是凸函数，这是 svm 的优势，具有全局最优

解，该二次规划问题的求解难度是参数 a_n 的数量很大，等于训练样本的数量。书上回顾了一些方法，介

绍不详细，主要思想是 chunking，我总结一下，总结的不一定准确：

1. 去掉 $\mathbf{a} = 0$ 对应的核函数矩阵的行和列，将二次优化问题划分成多个小的优化问题；
2. 按固定大小划分成小的优化问题。
3. SVM 中最流行的是 SMO, sequential minimal optimization。每次只考虑两个拉格朗日乘子。

SVM 中维度灾难问题：核函数相当于高维(甚至无限维)的特征空间的内积，避免了显示的高维空间运算，貌似是避免了维度过高引起的维度灾难问题。但实际上并没有避免。书上举了个例子，看这个二维多项式核函数：

$$\begin{aligned}k(\mathbf{x}, \mathbf{z}) &= (1 + \mathbf{x}^T \mathbf{z})^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\&= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\&= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\&= \phi(\mathbf{x})^T \phi(\mathbf{z}).\end{aligned}\quad (7.42)$$

这个核函数表示一个六维空间的内积。 $\Phi(\mathbf{x})$ 是从输入空间到六维空间的映射。映射后，六个维度每个维度的值是由固定参数的，也就是映射后，六维特征是有固定的形式。因此，原二维数据 $\mathbf{x}$ 都被限制到了六维空间的一个 nonlinear manifold 中。这个 manifold 之外就没有数据。

网神(66707180) 20:48:59

大家有什么问题吗？

高老头(1316103319) 20:49:58

manifold 是什么意思？

网神(66707180) 20:50:26

我的理解是空间里一个特定的区域，原空间的数据，如果采样不够均匀，映射后的空间，仍然不会均匀，不会被打散到空间的各个角落，而只会聚集在某个区域。

接下来讲下 SVM 用于回归问题。

在线性回归中，一个正则化错误函数如下：
$$\frac{1}{2} \sum_{n=1}^N \{y_n - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

为了获得稀疏解，将前面的二次错误函数用 ϵ -insensitive 错误函数代替

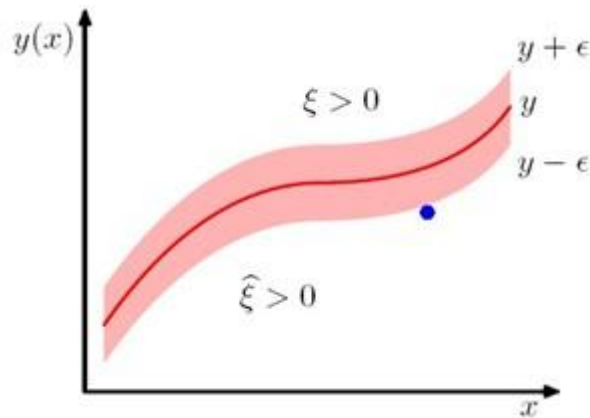
$$E_{\epsilon}(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

这个错误函数在 $y(\mathbf{x})$ 和 t 的差小于 ϵ 时等于 0。错误函数变为：

$$C \sum_{n=1}^N E_{\epsilon}(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

我们再引入松弛变量，对每个样本，有两个松弛变量，分别对应 $t_n > y(\mathbf{x}_n) + \epsilon$ 和 $t_n < y(\mathbf{x}_n) - \epsilon$

如图：



没引入松弛变量前，样本值 t 预测正确的条件是 $y_n - \epsilon \leq t_n \leq y_n + \epsilon$

$$t_n \leq y(x_n) + \epsilon + \xi_n$$

引入松弛变量后，变为： $t_n \geq y(x_n) - \epsilon - \hat{\xi}_n$

错误函数变为：

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

加上约束条件 $\xi_n \geq 0$ 和 $\hat{\xi}_n \geq 0$ ，就可以写出拉格朗日函数

下面就跟前面的分类一样了。

关于统计学习理论，书上简单提了一下 PAC(probably approximately correct)和 VC 维，简单总结一下书上的内容：PAC 的目的是理解多大的数据集可以给出好的泛化性，以及研究损失的上限。PAC 里的一个关键概念是 VC 维，用于提供一个函数空间复杂度的度量，将 PAC 理论推广到了无限大的函数空间上。

=====讨论=====

Fire(564122106) 21:05:56

有哪位大神想过对 svm 提速的啊，svm 在非线性大数据的情况下，速度还是比较慢的啊

网神(66707180) 21:07:36

svm 分布式训练的方案研究过吗？

Fire(564122106) 21:09:29

没有，不过将来肯定要研究的！现在只是单机，现在在有在单机的情况下，分布式进入内存的方案，有兴趣的可以看下：

Selective Block Minimization for Faster Convergence of Limited Memory Large_Scale Linear Models 这个有介绍，我共享下啊。

苦瓜炒鸡蛋(852383636) 21:11:05

韩家炜的一个学生 提出了一个 仿照层次聚类的思想 改进的 svm 速度好像挺快的

Making SVMs Scalable to Large Data Sets using Hierarchical Cluster Indexing 这个就是那篇论文的题目 发在 Data Mining and Knowledge Discovery

Fire(564122106) 21:16:29

哦 我看下，我现在看的都是台湾林的

 Fire 分享文件 21:14:33

"Selective Block Minimization for Faster Convergence of Limited Memory Large_Scale Linear Models.pdf" 下载

苦瓜炒鸡蛋(852383636) 21:17:41

有那个大神 在用 svm 做聚类, Support Vector Clustering 这篇能做 就是时间复杂度太高了 $O(n^3)$

=====讨论结束=====

网神(66707180) 8:54:01

咱们开始讲 RVM, 前面讲了 SVM, SVM 有一些缺点, 比如输出是 decision 而不是概率分布, SVM 是为二元分类设计的, 多类别分类不太实用, 虽然有不少策略可以用于多元分类, 但也各有问题参数 C 需要人工选择, 通过多次训练来调整, 感觉实际应用中这些缺点不算什么大缺点, 但是 RVM 可以避免这些缺点。RVM 是一种贝叶斯方式的稀疏核方法, 可以用于回归和分类, 除了避免 SVM 的主要缺点, 还可以更稀疏, 而泛化能力不会降低。先看 RVM 回归, RVM 回归的模型跟前面第三章形式相同, 属于线性模型, 但是参数 w 的先验分布有所不同。

这个不同导致了稀疏性, 等下再看这个不同

线性回归模型如下:

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}), \beta^{-1}) \text{ 其中 } \beta = \sigma^{-2}, \text{ 是噪音的精度 precision}$$

$$y(\mathbf{x}) = \sum_{i=1}^M w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

均值 $y(\mathbf{x})$ 定义为:

RVM 作为一种稀疏核方法, 它是如何跟核函数搭上边的, 就是基函数 $\Phi_i(\mathbf{x})$ 采用了核函数的形式

每个核与一个训练样本对应, 也就是:

$$y(\mathbf{x}) = \sum_{n=1}^N w_n k(\mathbf{x}, \mathbf{x}_n) + b$$

这个形式跟 SVM 用于回归的模型形式是相同的, 看前面的式子(7.64)最后求得的 SVM 回归模型是:

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$

可以看到, RVM 回归和 SVM 回归模型相同, 只是前面的系数从 a_n 换成了 w_n , 接下来分析如何确定

RVM 模型中的参数 w , 下面的分析过程跟任何基函数都适用, 不限于核函数。

确定 w 的过程可以总结为: 先假设 w 的先验分布, 一般是高斯分布; 然后给出似然函数, 先验跟似然函数相乘的到 w 的后验分布, 最大化后验分布, 得到参数 w 。

先看 w 的先验, w 的先验是以 0 为均值, 以 α 为精度的高斯分布, 但是跟第三章线性回归的区别是, RVM

为每个 w_i 分别引入一个精度 α_i , 而不是所有 w_i 用一个的共享的精度

所以 w 的先验是:

$$p(\mathbf{w} | \alpha) = \prod_{i=1}^M \mathcal{N}(w_i | 0, \alpha_i^{-1})$$

对于线性回归模型, 根据这个先验和似然函数可以得到其后验分布:

$$p(\mathbf{w} | \mathbf{t}, \mathbf{X}, \alpha, \beta) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \Sigma)$$

均值和方差分别是：

$$\mathbf{m} = \beta \Sigma \Phi^T \mathbf{t}$$

$$\Sigma = (\mathbf{A} + \beta \Phi^T \Phi)^{-1}$$

这是第三章的结论，推导过程就不说了

其中 Φ 是 $N \times M$ 的矩阵， $\Phi_{ni} = \phi_i(\mathbf{x}_n)$ ， $\mathbf{A}$ 是对角矩阵 $\mathbf{A} = \text{diag}(\alpha_i)$

对于 RVM，因为基函数是核函数，所以 $\Phi = \mathbf{K}$ ， $\mathbf{K}$ 是 $N \times N$ 维的核矩阵，其元素是 $k(\mathbf{x}_n, \mathbf{x}_m)$

接下来需要确定超参数 α 和 β 。一个是 w 先验的精度，一个是线性模型 $p(\mathbf{t}|\mathbf{x}, w)$ 的精度

确定的方法叫做 evidence approximation 方法，又叫 type-2 maximum likelihood，这在第三章有详细介绍，这里简单说一下思路：

该方法基于一个假设，即两个参数是的后验分布 $p(\alpha, \beta | \mathbf{t})$ 是 sharply peaked 的，其中心值是 $\hat{\alpha}$ 和 $\hat{\beta}$ ，

根据贝叶斯定理， $p(\alpha, \beta | \mathbf{t}) \propto p(\mathbf{t} | \alpha, \beta) p(\alpha, \beta)$ ，先验 $p(\alpha, \beta)$ 是 relatively flat 的，所以只要看

$p(\mathbf{t} | \alpha, \beta)$ ， $\hat{\alpha}$ 和 $\hat{\beta}$ 就是使的 $p(\mathbf{t} | \alpha, \beta)$ 最大的值。

$p(\mathbf{t} | \alpha, \beta)$ 是对 w 进行积分的边界分布：

$$p(\mathbf{t} | \alpha, \beta) = \int p(\mathbf{t} | w, \beta) p(w | \alpha) dw$$

这个分布是两个高斯分布的卷积，其 log 最大似然函数是：

$$\begin{aligned} \ln p(\mathbf{t} | \mathbf{X}, \alpha, \beta) &= \ln \mathcal{N}(\mathbf{t} | \mathbf{0}, \mathbf{C}) \\ &= -\frac{1}{2} \{ N \ln(2\pi) + \ln |\mathbf{C}| + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} \} \end{aligned}$$

其中 $\mathbf{C}$ 是 $N \times N$ 矩阵， $\mathbf{C} = \beta^{-1} \mathbf{I} + \Phi \mathbf{A}^{-1} \Phi^T$

$\ln p(\mathbf{t} | \mathbf{X}, \alpha, \beta) = \ln \mathcal{N}(\mathbf{t} | \mathbf{0}, \mathbf{C})$ 这一步，以及 $\mathbf{C}$ 的值，是第三章的内容，大家看前面吧。

我们可以通过最大化似然函数，求得 $\hat{\alpha}$ 和 $\hat{\beta}$ ，书上提到了两种方法，一种是 EM，一种是直接求导迭代。

前者第九章尼采已经讲了，这里看下后者。

首先我们分别求这个 log 似然函数对所有参数 α_i 和 β 求偏导，并令偏导等于 0，求得参数的表达式：

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{m_i^2}$$

$$(\beta^{\text{new}})^{-1} = \frac{\|\mathbf{t} - \Phi \mathbf{m}\|^2}{N - \sum_i \gamma_i}$$

其中 m_i 是 w 的后验均值 $\mathbf{m}$ 的第 i 个元素， γ_i 是度量 w_i 被样本集合影响的程度 $\gamma_i = 1 - \alpha_i \Sigma_{ii}$

Σ_{ii} 是 w 的后验方差 Σ 的对角线上的元素。

求 $\hat{\alpha}$ 和 $\hat{\beta}$ 是一个迭代的过程：

先选一个 $\hat{\alpha}$ 和 $\hat{\beta}$ 的初值，然后用下面这个公式得到后验的均值和方差：

$$\begin{aligned}\mathbf{m} &= \beta \Sigma \Phi^T \mathbf{t} \\ \Sigma &= (\mathbf{A} + \beta \Phi^T \Phi)^{-1}\end{aligned}$$

然后再用这个公式重新计算 $\hat{\alpha}$ 和 $\hat{\beta}$

$$\begin{aligned}\alpha_i^{\text{new}} &= \frac{\gamma_i}{m_i^2} \\ (\beta^{\text{new}})^{-1} &= \frac{\|\mathbf{t} - \Phi \mathbf{m}\|^2}{N - \sum_i \gamma_i}\end{aligned}$$

这样迭代计算，一直到到达一个人为确定的收敛条件，这就是确定 $\hat{\alpha}$ 和 $\hat{\beta}$ 的过程。

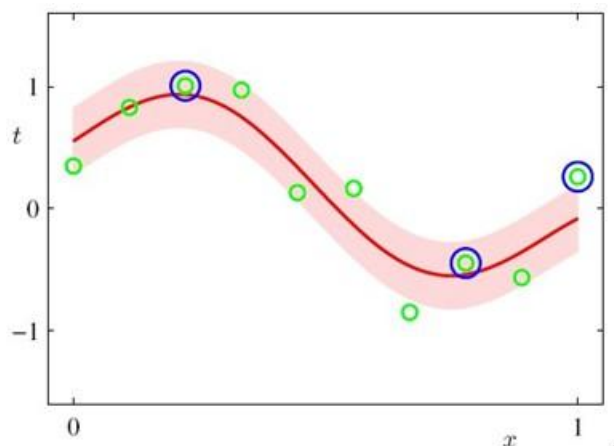
通过计算，最后的结果中，大部分参数 $\{\alpha_i\}$ 都是非常大甚至无穷大的值，从而根据 w 后验均值和方差的公式，其均值和方差都等于 0，这样 $\mathbf{w}_i$ 的值就是 0，其对应的基函数就不起作用了，从而达到了稀疏的目的。这就是 RVM 稀疏的原因。

需要实际推导一下整个过程，才能明白为什么大部分 $\{\alpha_i\}$ 都趋于无穷大。那些 $\mathbf{w}_i$ 不为 0 的 $\mathbf{x}_i$ 叫做 relevance vectors，相当于 SVM 中的支持向量。需要强调，这种获得稀疏性的机制可以用于任何基函数的线性组合中。这种获得稀疏性的机制似乎非常普遍的。

求得超参数，就可以通过下面式子得到新样本的分布：

$$\begin{aligned}p(t|\mathbf{x}, \mathbf{X}, \mathbf{t}, \alpha^*, \beta^*) &= \int p(t|\mathbf{x}, \mathbf{w}, \beta^*) p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha^*, \beta^*) d\mathbf{w} \\ &= \mathcal{N}(t|\mathbf{m}^T \phi(\mathbf{x}), \sigma^2(\mathbf{x})).\end{aligned}$$

下面看一个图示：



可以看到，其相关向量的数量比 SVM 少了很多，跟 SVM 相比的缺点是，RVM 的优化函数不是凸函数，训练时间比 SVM 长，书上接下来专门对 RVM 的稀疏性进行分析，并且介绍了一种更快的求 $\hat{\alpha}$ 和 $\hat{\beta}$ 的

方法：

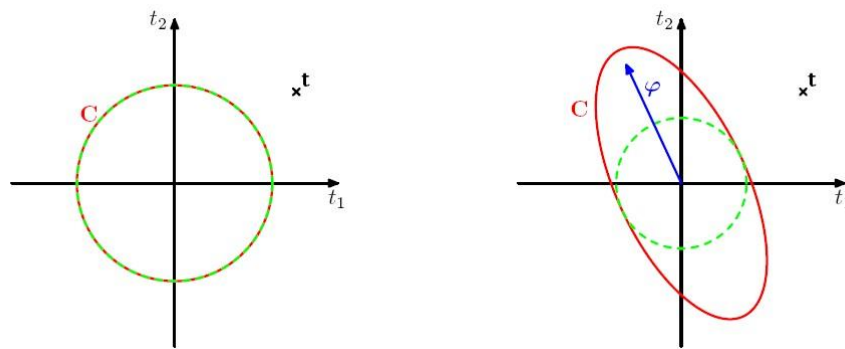


Figure 7.10 Illustration of the mechanism for sparsity in a Bayesian linear regression model, showing a training set vector of target values given by $\mathbf{t} = (t_1, t_2)^T$, indicated by the cross, for a model with one basis vector $\phi = (\phi(x_1), \phi(x_2))^T$, which is poorly aligned with the target data vector $\mathbf{t}$. On the left we see a model having only isotropic noise, so that $\mathbf{C} = \beta^{-1}\mathbf{I}$, corresponding to $\alpha = \infty$, with β set to its most probable value. On the right we see the same model but with a finite value of α . In each case the red ellipse corresponds to unit Mahalanobis distance, with $|\mathbf{C}|$ taking the same value for both plots, while the dashed green circle shows the contribution arising from the noise term β^{-1} . We see that any finite value of α reduces the probability of the observed data, and so for the most probable solution the basis vector is removed.

我接着讲 RVM 分类，我们看逻辑回归分类的模型：

$$y(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$$

σ 是 sigmoid 函数，我们引入 $\mathbf{w}$ 的先验分布，跟 RVM 回归相同，每个 $\mathbf{w}_i$ 对应一个不同的精度

这种先验叫做 ARD 先验，跟 RVM 回归相比，在求 $p(\mathbf{t} | \alpha, \beta)$ 的分布时，不再对 $\mathbf{w}$ 进行积分。

我们看 RVM 回归时，是这么求 $p(\mathbf{t} | \alpha, \beta)$ 的：

$$p(\mathbf{t} | \alpha, \beta) = \int p(\mathbf{t} | \mathbf{w}, \beta) p(\mathbf{w} | \alpha) d\mathbf{w} \quad \text{从而得到：}$$

$$\begin{aligned} \ln p(\mathbf{t} | \mathbf{X}, \alpha, \beta) &= \ln \mathcal{N}(\mathbf{t} | \mathbf{0}, \mathbf{C}) \\ &= -\frac{1}{2} \{ N \ln(2\pi) + \ln |\mathbf{C}| + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} \} \end{aligned}$$

在 RVM 分类时，因为涉用到 sigmoid 函数，计算积分很难，具体的为什么难，在第四章 4.5 节有更多的介绍，我们这里用 Laplace approximation 来求 $p(\mathbf{t} | \alpha, \beta)$ 的近似高斯分布，Laplace approximation 我

叫拉普拉斯近似，后面都写中文了。

先看下拉普拉斯近似的原理，拉普拉斯近似的目的是找到连续变量的分布函数的高斯近似分布，也就是用高斯分布 近似模拟一个不是高斯分布的分布。

假设一个单变量 z ，其分布是 $p(z) = \frac{1}{Z} f(z)$ ，分母上的 Z 是归一化系数 $Z = \int f(z) dz$ ，目标是找到一个可以近似 $p(z)$ 的高斯分布 $q(z)$ 。

第一步是先找到 $p(z)$ 的 mode(众数)，众数 mode 是一个统计学的概念，可以代表一组数据，不受极端数据的影响，比如可以选择中位数做一组实数的众数，对于高斯分布，众数就是其峰值。一组数据可能没有众数也可能有几个众数。

拉普拉斯分布第一步要找到 $p(z)$ 的众数 z_0 ，这是 $p(z)$ 的一个极大值点，可能是局部的，因为 $p(z)$ 可能有多

个局部极大值。在该点，一阶导数等于 0， $p'(z_0) = 0$ ，后面再说怎么找 z_0 。找到 z_0 后，用 $\ln f(x)$ 的泰勒展开来构造一个二次函数：

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2} A (z - z_0)^2$$

其中 A 是 $f(z)$ 在 z_0 的二阶导数再取负数。上式中，没有一阶导数部分，因为 z_0 是局部极大值，一阶导数为 0，把上式两边取指数，得到：

$$f(z) \approx f(z_0) \exp\left\{-\frac{A}{2} (z - z_0)^2\right\}$$

把 $f(z_0)$ 换成归一化系数，得到近似的高斯分布：

$$q(z) = \left(\frac{A}{2\pi}\right)^{1/2} \exp\left\{-\frac{A}{2} (z - z_0)^2\right\}$$

拉普拉斯分布得到的近似高斯分布的一个图示：

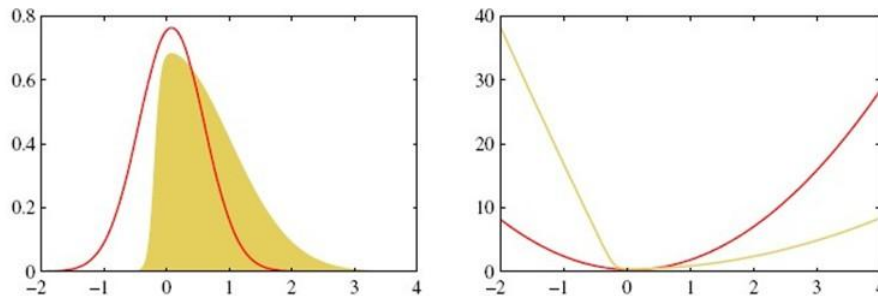


Figure 4.14 Illustration of the Laplace approximation applied to the distribution $p(z) \propto \exp(-z^2/2)\sigma(20z + 4)$ where $\sigma(z)$ is the logistic sigmoid function defined by $\sigma(z) = (1 + e^{-z})^{-1}$. The left plot shows the normalized distribution $p(z)$ in yellow, together with the Laplace approximation centred on the mode z_0 of $p(z)$ in red. The right plot shows the negative logarithms of the corresponding curves.

注意，高斯近似存在的条件是，原分布的二阶导数取负数、也就是高斯近似的精确度 $A > 0$ ，也就是驻点 z_0 必须是局部极大值， $f(x)$ 在改点出的导数为负数。当 z 是一个 M 维向量时，近似方法跟单变量的不同只是二阶导数的负数 A 变成了 $M \times M$ 维的海森矩阵的负数。

多维变量近似后的高斯分布如下：

$$q(z) = \frac{|A|^{1/2}}{(2\pi)^{M/2}} \exp\left\{-\frac{1}{2} (z - z_0)^T A (z - z_0)\right\} = N(z|z_0, A^{-1})$$

A 是海森矩阵的负数，mode 众数 z_0 一般是通过数值优化算法来寻找的，不讲了。

再回来看用拉普拉斯分布来近似 RVM 分类中的 $p(t|\alpha, \beta)$ ：

刚才拉普拉斯分布忘了说一个公式，就是求得 $q(z)$ 后，确定 $p(z) = \frac{1}{Z} f(z)$ 中的分母，也就是归一化系数

的公式：

$$\begin{aligned} Z &= \int f(z) dz \\ &\approx f(z_0) \int \exp\left\{-\frac{1}{2} (z - z_0)^T A (z - z_0)\right\} dz \\ &= f(z_0) \frac{(2\pi)^{M/2}}{|A|^{1/2}} \end{aligned}$$

这个一会有用。先看 RVM 中对 $\mathbf{w}$ 的后验分布的近似，先求后验分布的 mode 众数，通过最大化 log 后验分布 $\ln p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha})$ 来求 mode. 先写出这个 log 后验分布：

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}) &= \ln \{p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} - \ln p(\mathbf{t}|\boldsymbol{\alpha}) \\ &= \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} + \text{const} \quad (7.109)\end{aligned}$$

其中 $\mathbf{A} = \text{diag}(\alpha_i)$

最后求得的高斯近似的均值(也就是原分布的 mode)和精度如下：

$$\begin{aligned}\mathbf{w}^* &= \mathbf{A}^{-1} \boldsymbol{\Phi}^T (\mathbf{t} - \mathbf{y}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Phi}^T \mathbf{B} \boldsymbol{\Phi} + \mathbf{A})^{-1}\end{aligned}$$

现在用这个 $\mathbf{w}$ 后验高斯近似来求边界似然 $p(\mathbf{t}|\boldsymbol{\alpha}) = \int p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w}$

根据前面那个求归一化系数 Z 的公式 $Z = \frac{f(\mathbf{z}_0) (2\pi)^{M/2}}{|\mathbf{A}|^{1/2}}$

有：

$$\begin{aligned}p(\mathbf{t}|\boldsymbol{\alpha}) &= \int p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w} \\ &\simeq p(\mathbf{t}|\mathbf{w}^*)p(\mathbf{w}^*|\boldsymbol{\alpha})(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}.\end{aligned}$$

RVM 这部分大量基于第二章高斯分布和第三、四两章，公式推导很多，需要前后关联才能看明白。

第八章 Graphical Models

主讲人 网神

(新浪微博: @豆角茄子麻酱凉面)

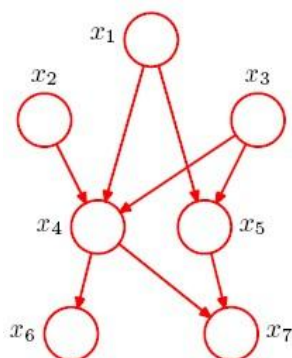
网神(66707180) 18:52:10

今天的内容主要是：

1.贝叶斯网络和马尔科夫随机场的概念，联合概率分解，条件独立表示；2.图的概率推断 inference。

图模型是用图的方式表示概率推理，将概率模型可视化，方便展示变量之间的关系，概率图分为有向图和无向图。有向图主要是贝叶斯网络，无向图主要是马尔科夫随机场。对两类图，prml 都讲了如何将联合概率分解为条件概率，以及如何表示和判断条件依赖。

先说贝叶斯网络，贝叶斯网络是有向图，用节点表示随机变量，用箭头表示变量之间的依赖关系。一个例子：



这是一个有向无环图，这个图表示的概率模型如下：

$$p(x_1, x_2, \dots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5). \quad \text{形式化}$$

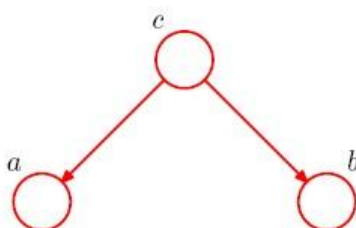
一下，贝叶斯网络表示的联合分布是：

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | \text{pa}_k)$$

其中 pa_k 是 x_k 的所有父节点。

以上是贝叶斯网络将联合概率分解为条件概率的方法，比较直观易懂，就不多说了。下面说一下条件独立的表示和判断方法。条件独立是，给定 a, b, c 三个节点，如果 $p(a, b | c) = p(a | c)p(b | c)$ ，则说给定 c ， a 和 b 条件独立。当然 a, b, c 也可以是三组节点，这里只以单个节点为例。用图表示，有三种情况。

第一种情况如图：



c 位于两个箭头的尾部，称作 tail-to-tail，这种情况， c 未知的时候， a, b 是不独立的。 c 已知的时候，

a,b 条件独立。来看为什么，首先，这个图联合概率如下：

在 c 未知的时候,p(a,b)如下求解:

$$p(a, b) = \sum_c p(a|c)p(b|c)p(c).$$

可以看出，无法得出: $p(a,b)=p(a)p(b)$ ，所以 a,b 不独立。

如果 c 已知，则：

$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

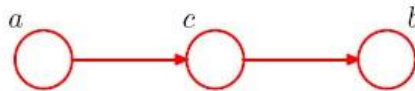
所以 a,b 条件独立于 c。条件独立用以下符号表示：

$$a \perp\!\!\!\perp b \mid c.$$

a,b 不独立的符号表示：

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

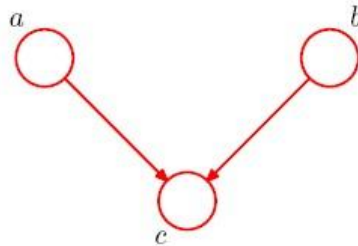
这是图表示的条件独立的第一种形式，叫做 tail-to-tail。第二种是 tail-to-head，如图：



这种情况也是 c 未知时，a 和 b 不独立。c 已知时，a 和 b 条件独立于 c，推导如下：

$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

第三种情况是 head-to-head，如图：



这种情况反过来了，c 未知时，a 和 b 是独立的；但当 c 已知时，a 和 b 不满足条件独立，

因为： $p(a, b, c) = p(a)p(b)p(c|a, b)$.

计算该概率的边界概率，得

$$p(a, b) = p(a)p(b)$$

所以 a 和 b 相互独立.

但 c 已知时：

$$\begin{aligned}
 p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
 &= \frac{p(a)p(b)p(c|a,b)}{p(c)}
 \end{aligned}$$

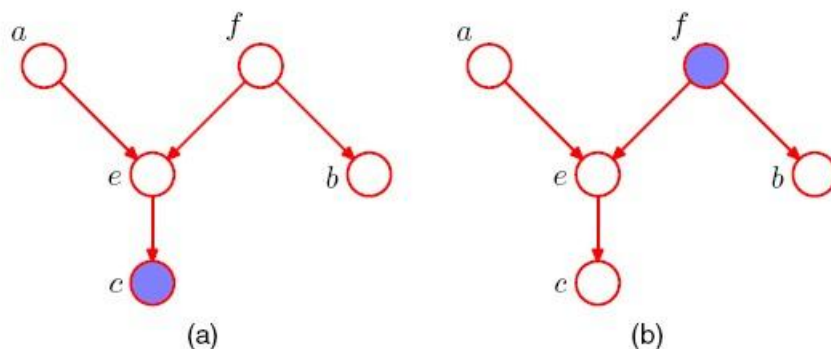
无法得到 $p(a,b|c)=p(a|c)p(b|c)$

将这三种情况总结，就是贝叶斯网络的一个重要概念，D-separation，这个概念的内容就是：

A,B,C 三组节点，如果 A 中的任意节点与 B 的任意节点的所有路径上，存在以下节点，就说 A 和 B 被 C 阻断：

1. A 到 B 的路径上存在 tail-to-tail 或 head-to-tail 形式的节点，并且该节点属于 C
2. 路径上存在 head-to-head 的节点，并且该节点不属于 C

举个例子：



左边图上，节点 f 和节点 e 都不是 d-separation. 因为 f 是 tail-to-tail，但 f 不是已知的，因此 f 不属于 C。e 是 head-to-head，但 e 的子节点 c 是已知的，所以 e 也不属于 C。

speedmancs<speedmancs@qq.com> 19:23:05

2 漏了一点，该节点包括所有后继

网神(66707180) 19:23:21

右边图，f 和 e 都是 d-separation. 理由与上面相反. 对，是漏了这一点。看到这个例子才想起来，这部分大家有什么问题没？

speedmancs<speedmancs@qq.com> 19:24:54

这个还是抽象了一些，我之前看的 prml 这一章，没看懂，后来看了 PGM 前三章，主要看了那个学生成绩的那个例子，就明白了。姑妄记之，其实蛮不好记的。讲得挺好的，继续。

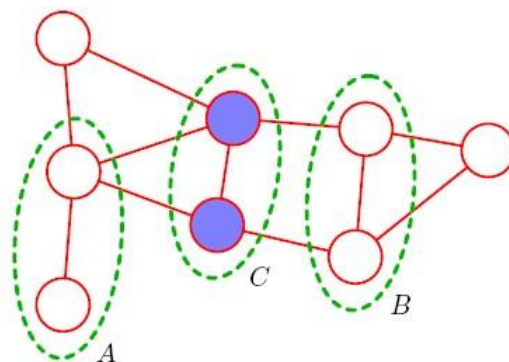
网神(66707180) 19:27:14

因为有了这些条件独立的规则，可以将图理解成一个 filter。

既给定一系列随机变量，其联合分布 $p(x_1, x_2, \dots, x_n)$ 理论上可以分解成各种条件分布的乘积，但过一遍图，不满足图表示依赖关系和条件独立的分布就被过滤掉。所以图模型，用不同随机变量的连接表示各种关系，可以表示复杂的分布模型。

接下来是马尔科夫随机场，是无向图，也叫马尔科夫网络，马尔科夫网络也有条件独立属性。

用 MRF(Markov random field)表示马尔科夫网络，MRF 因为是无向的，所以不存在 tail-to-tail 这些概念。MRF 的条件独立如图：



如果 A 的任意节点和 B 的任意节点的任意路径上，都存在至少一个节点属于 C

speedmancs<speedmancs@qq.com> 19:33:16

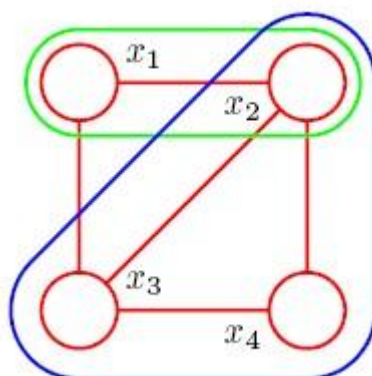
无向图的条件独立 比有向图简单多了。

网神(66707180) 19:33:18

那么 A 和 B 条件独立于 C，可以理解为，如果 C 的节点都是已知的，就阻断了 A 和 B 的所有路径。

网神(66707180) 19:33:49

嗯，MRF 的概率分解就概念比较多了，不像有向图那么直观，MRF 联合概率分解成条件概率。用到了 clique 的概念，我翻译成“团”，就是图的一个子图，子图上两两节点都有连接。例如这个图，最大团有两个，分别是(x1,x2,x3)和(x2,x3,x4)：



MRF 的联合概率分解另一个概念是 potential function，联合概率分解成一系列 potential 函数的乘积：

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C).$$

$\mathbf{x}_C$ 是一个最大团的所有节点，一个 potential 函数 $\psi_C(\mathbf{x}_C)$ ，是最大团的一个函数。

这个函数具体的定义是依赖具体应用的，一会举个例子。

上面式子里那个 Z 是 normalization 常量：

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

speedmancs<speedmancs@qq.com> 19:42:47

这个 Z 很麻烦

网神(66707180) 19:43:15

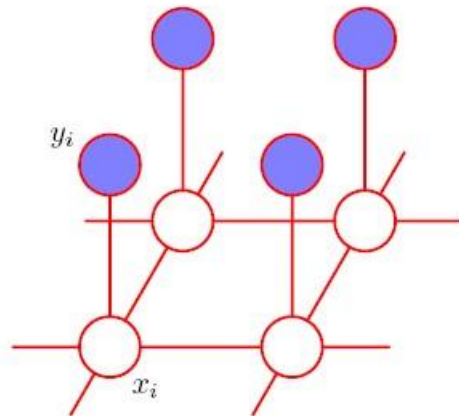
$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C).$$

在这个式子里， $p(\mathbf{x})$ 是一系列 potential 函数的乘积。换一种理解方式，定义将 potential 函数表示成指数函数：

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

这样 $p(\mathbf{x})$ 就可以表示成一系列 $E(\mathbf{x}_C)$ 的和的指数函数， $E(\mathbf{x}_C)$ 叫做能量函数，这么转换之后，可以将图理解成一个能量的集合，他的值等于各个最大团的能量的和。先举个例子看看 potential 函数和能量函数在具体应用中是什么样的，大家再讨论。

要把噪声图片尽量还原成 原图，用图的方式表示噪声图和还原后的图，每个像素点是一个节点：



上面那层 y_i ，是噪声图，紫色表示这些是已知的，是观察值。下面那层 x_i 是未知的，要求出 x_i ，使 x_i 作为像素值得到的图，尽量接近无噪声图片。每个 x_i 的值，与 y_i 相关，也与相邻的 x_j 相关。这里边，最大团是 (x_i, y_i) 和 (x_i, x_j) ，两类最大团。

对于 (x_i, y_i) ，选择能量函数 $E(x_i, y_i) = -\eta x_i y_i$ ；对于 (x_i, x_j) ，选择能量函数 $E(x_i, x_j) = -\beta x_i x_j$

两个能量函数的意思都是，如果 x_i 和 y_i (或 x_i 和 x_j) 的值相同，则能量小；如果不同，则能量大。

整个图的能量如下：

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i, j\}} x_i x_j - \eta \sum_i x_i y_i$$

$h \sum_i x_i$ 是一个偏置项

speedmancs <speedmancs@qq.com> 19:56:55

偏置项是一种先验吧

网神(66707180) 19:57:28

有了这个能量函数，接下来就是求出 x_i ，使得能量 $E(\mathbf{x}, \mathbf{y})$ 最小。求最小，书上简单说了一下，我的理解也是用梯度下降类似的方法。

speedmancs <speedmancs@qq.com> 19:57:34

这里表示 -1 的点更多吧。

网神(66707180) 19:58:20

对偏执项的作用，书上这么解释：

Such a term has the effect of biasing the model towards pixel values that have one particular sign in preference to the other.

speedmancs<speedmancs@qq.com> 19:59:29

恩，对，让能量最小

网神(66707180) 19:59:39

为了使 $E(x,y)$ 尽量小，是尽量让 x_i 选择 -1，而不是 1。 $E(x,y)$ 越小，得到的图就越接近无噪声的图，因为：

$$p(x, y) = \frac{1}{Z} \exp\{-E(x, y)\}$$

所以 $E(x,y)$ 越小， $p(x,y)$ 就越大。

η<liyitan2144@163.com> 20:01:13

是不是可以这样看， $\beta \sum_{(i,j)} x_i x_j$ 表示平滑； $\eta \sum_i x_i y_i$ 表示似然。

网神(66707180) 20:02:18

liyitan2144 说得好，👍

speedmancs<speedmancs@qq.com> 20:02:33

恩，所以这是一个产生式模型。

网神(66707180) 20:02:59

有向图和无向图的概率分解和条件独立都说完了。有向图和无向图是可以互相转换的，有向图转换成无向图，如果每个节点都只有少于等于 1 个父节点，比较简单。如果有超过 1 个父节点，就需要在转换之后的无向图上增加一些边，来避免都是有向图上的一些关系，这部分就不细说了。

下面要说图的 inference 了。前面大家有啥要讨论的？先讨论一下吧。

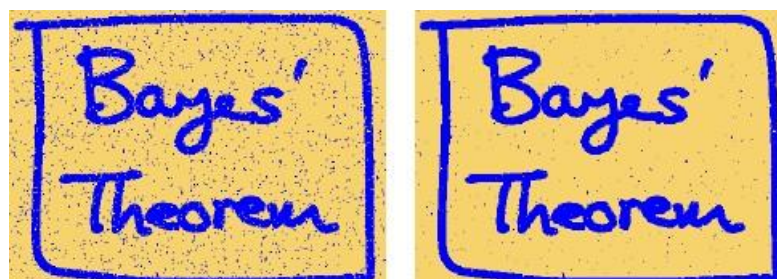
=====讨论=====

speedmancs<speedmancs@qq.com> 20:06:45

刚才那个例子中，那几个 β , h 等参数如何得到？

网神(66707180) 20:07:30

那个是用迭代求解的方法，求得这几个参数，书上提到了两种方法，一种 ICM, iterated conditional modes 一种 max-product 方法，其中 max-product 方法效果比较好，在后面的 inference 一节里详细讲了这个方法，而 ICM 方法只是提了一下，原理没有细说。两种方法的效果看下图，左边是 ICM 的结果，右边是 max-product 的方法：



speedmancs<speedmancs@qq.com> 20:15:07

稍微插一句，刚才那个 denoise 的例子，最好的那个结果是 graph cut，而且那几个 β 参数是事先固定了。

η<liyitan2144@163.com> 20:16:12

graph cut 是指？

kxkr<lxfkxkr@126.com> 20:16:30

the graph-cut algorithm on the right. ICM produces an image where 96% of the pixels agree with the original image, whereas the corresponding number for graph-cut is 99%.

kxkr<lxfkxkr@126.com> 20:17:18

tribution. However, for certain classes of model, including the one given by (8.42), there exist efficient algorithms based on *graph cuts* that are guaranteed to find the global maximum (Greig *et al.*, 1989; Boykov *et al.*, 2001; Kolmogorov and Zabih, 2004). The lower right panel of Figure 8.30 shows the result of applying a graph-cut algorithm to the de-noising problem.

网神(66707180) 20:18:11

这个例子只是为了说明能量函数和潜函数.所以不一定是最佳方法, 这两段截屏是 prml 上的, 咋没看到捏

kxkr<lxfkxkr@126.com> 20:19:05

那几个参数如何得到 文中好像并没有说, 只是说了 ICM、graph-cut 能够得到最后的去噪图像, 第一段在 figure8.30, 第二个截图在 section8.4, figure8.32 下面。

网神(66707180) 20:20:15

看到了, 先不管这个吧, 主要知道能量函数是啥样的就行了

kxkr<lxfkxkr@126.com> 20:21:25

嗯

=====讨论结束=====

网神(66707180) 20:22:24

接着说 inference 了, inference 就是已知一些变量的值, 求另一些变量的概率

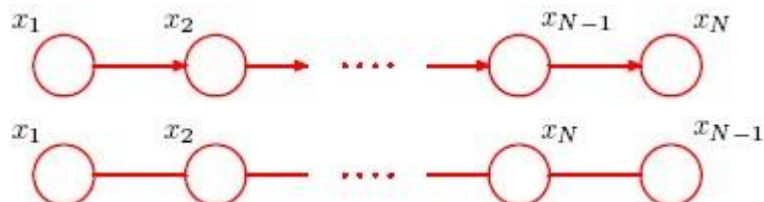


比如上图, 已知 y , 求 x 的概率

这个简单的图可以用典型的贝叶斯法则来求

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

对于复杂点的情况, 比如链式图:



为了求 $p(x_N)$, 就是求 x_N 的边缘概率。这里都假设 x 的值是离散的。如果是连续的, 就是积分, 为了求这个边缘概率, 做这个累加动作, 如果 x 的取值是 k 个, 则要做 k 的 $(N-1)$ 次方次计算, 利用图结构, 可以简化计算, 这个简化方法就不讲了。

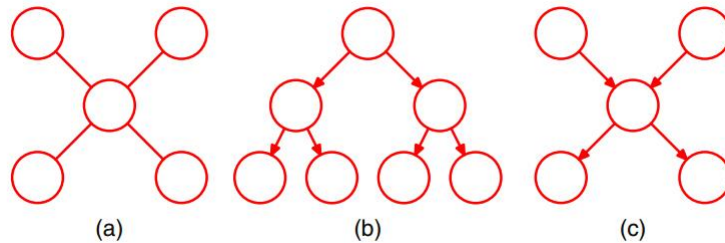
下面讲通用的图 inference 的方法, 就是 factor graph 方法, 链式图或树形图, 都比较好求边缘概率。

所以 factor graph 就是把复杂的图转换成树形图, 针对树形图来求.先说一下什么是树形图, 树形图有两

种情况：

1. 每个节点只有一个父节点。
2. 如果有的节点有多个父节点，必须图上每个节点之间只有一条路径。

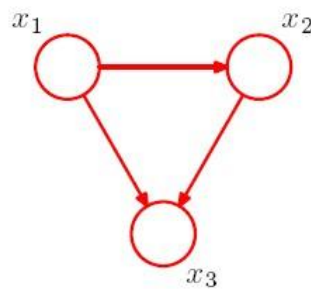
这是树形图的三种情况：



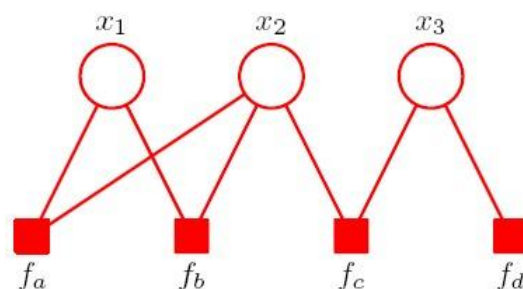
对于无向图，只要无环，都可以看做树形图；对于有向图，必须每两个节点之间只有一条路径；中间那种是典型的树.右边那种多个有多个节点的叫 polytree.

这种书结构的图，都比较好求边界概率. 具体怎么求，就不说了。

这里主要说怎么把复杂的图转换成树形图. 这种转换引入 factor 节点，从而将普通的图转换成 factor 图. 先看个例子：



对于这个有向图， $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$ ，等号右边的三个概率作为三个 factor 每个 factor 作为一个节点，加入新的 factor 图中：



上面的 x_1, x_2, x_3 是原图的随机变量，下面的 f_a, f_b, f_c, f_d 是 factor 节点，只有随即变量节点和 factor 节点之间有连接，每类节点自身互不连接，每个 factor 连接的变量节点，是相互依赖的节点。

kxkr<lxfkxkr@126.com> 20:42:10

就是一个 clique 中的节点吧

网神(66707180) 20:42:19

对，严格的说，也不是。

kxkr<lxfkxkr@126.com> 20:44:15

对于有向图？

网神(66707180) 20:45:10

x_1, x_2, x_3 是一个最大团. 可以只用一个 factor 节点, 如中间那个图

kxkr<lxfkxkr@126.com> 20:45:17

嗯

网神(66707180) 20:45:22

也可以用多个 factor 节点, 如右边图, 但什么情况下用一个 factor 什么情况用多个 factor, 我没想明白

kxkr<lxfkxkr@126.com> 20:47:10

嗯, 继续

网神(66707180) 20:47:21

转换成 factor 图后, 就是树形图了, 符合前面树形图的两种情况, 这时候, 要求一个节点或一组节点的边界概率, 用一种叫做 sum-product 的方法, 已求一个节点的边界概率为例.

η<liyitan2144@163.com> 20:49:50

网神(66707180) 20:46:31

但什么情况下用一个 factor 什么情况用多个 factor 虽然也对具体的应用情景不清楚, 但是一个 factor 能表达的信息比使用多个 factor 能表达的信息多.

网神(66707180) 20:51:06

单个的 factor 表达的信息多, 而且节点越少, 计算越简单, 所以是不是尽量用少的 factor? max-product 求单个节点的边界概率, 其思想是以该节点为 root.

η<liyitan2144@163.com> 20:52:38

但是, 从设计模型的角度看, 节点少了, 一个节点设计的复杂程度就大了, 不一定容易设计, 拆解成多个 factor, 每个 factor 都很简单, 设计方便.

kxkr<lxfkxkr@126.com> 20:54:37

文中貌似倾向于一个单个的 factor

If we are given a distribution that is expressed in terms of an undirected graph, then we can readily convert it to a factor graph. To do this, we create variable nodes corresponding to the nodes in the original undirected graph, and then create additional factor nodes corresponding to the maximal cliques $\mathbf{x}_s$. The factors $f_s(\mathbf{x}_s)$ are then set equal to the clique potentials. Note that there may be several different factor graphs that correspond to the same undirected graph. These concepts are illustrated in Figure 8.41.

η<liyitan2144@163.com> 20:56:57

这貌似是在表达一个通用的方法, 和是把大的 clique 的 factor 拆解成小的多个 factor 没有关系.

kxkr<lxfkxkr@126.com> 20:57:53

拆成多个 factor 是不是会造成参数变多, 不容易求呢, 继续吧, 这个有待实践.

η<liyitan2144@163.com> 21:00:00

参数应该会变多吧, 不过模型其实简单了, 确实有待实践, 我觉得这和具体应用有关, 比如在一副图像里面, 如果仅仅表达“像素”的相似度, 我们完全可以使用小 factor.

网神(66707180) 21:00:39

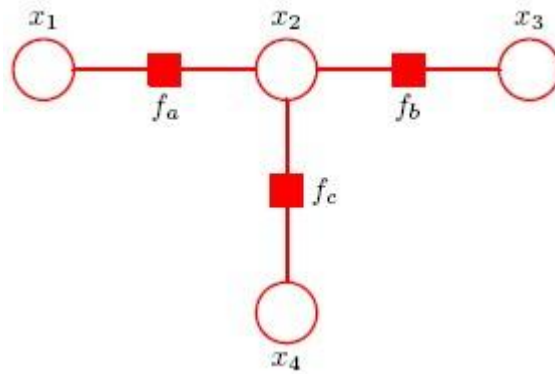
我继续, prml 这章图模型只讲了基础的概念, 没讲常用的图模型实例, HMM 和 CRF 这两大主流图模型方法还没讲, 感觉不够直观.

我继续说 inference. 转换成 factor 图后, 求节点 x_i 的边缘概率. 把 x_i 作为 root 节点, 把求 root 边界概率理解成一个信息(message)传递的过程, 从叶子节点传递概率信息到 root 节点. 传递的规则是:

从叶子节点开始, 如果叶子是变量节点, 发送 1 给父节点, 如果叶子是 factor 节点, 发送 $f(x)$ 给父节点.

对于非叶子节点, 如果是变量节点, 将其收到的 message 相乘, 发给父节点, 如果是 factor 节点, 将其收到的 message 和自身 $f(x)$ 相乘, 然后做一个 sum, 发给父节点.

举个例子：



这个图中求 x_3 的边缘概率，message 传递的过程是：

$$\begin{aligned}\mu_{x_1 \rightarrow f_a}(x_1) &= 1 \\ \mu_{f_a \rightarrow x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\ \mu_{x_4 \rightarrow f_c}(x_4) &= 1 \\ \mu_{f_c \rightarrow x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\ \mu_{x_2 \rightarrow f_b}(x_2) &= \mu_{f_a \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2) \\ \mu_{f_b \rightarrow x_3}(x_3) &= \sum_{x_2} f_b(x_2, x_3) \mu_{x_2 \rightarrow f_b}.\end{aligned}$$

$\mu_{x_1 \rightarrow f_a}(x_1)$ 中的 $x_1 \rightarrow f_a$ 表示从节点 x_1 传递到 f_a ，最后 $p(x_3)$ 是等于 $\mu_{f_b \rightarrow x_3}(x_3)$ ，因为只有一个 f_b 节点向其传入信息，如果要求 x_2 的边缘概率，因为 x_2 有三个节点出入信息，分别是：

$$\mu_{f_a \rightarrow x_2}(x_2) \quad \mu_{f_c \rightarrow x_2}(x_2) \quad \mu_{f_b \rightarrow x_2}(x_2)$$

所以 $p(x_2)$ 就等于这三个信息的乘积

$$\tilde{p}(x_2) = \mu_{f_a \rightarrow x_2}(x_2) \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2)$$

这个结果与边界分布的定义是相符的， x_2 的边界定义如下：

$$\tilde{p}(x_2) = \sum_{x_1} \sum_{x_3} \sum_{x_4} \tilde{p}(x)$$

$$\tilde{p}(x_2) = \mu_{f_a \rightarrow x_2}(x_2) \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2) \quad \text{通过这个，可以推导出上面的边界分布：}$$

$$\begin{aligned}\tilde{p}(x_2) &= \mu_{f_a \rightarrow x_2}(x_2) \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2) \\ &= \left[\sum_{x_1} f_a(x_1, x_2) \right] \left[\sum_{x_3} f_b(x_2, x_3) \right] \left[\sum_{x_4} f_c(x_2, x_4) \right] \\ &= \sum_{x_1} \sum_{x_2} \sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\ &= \sum_{x_1} \sum_{x_3} \sum_{x_4} \tilde{p}(x)\end{aligned}$$

上面就是用 sum-product 来求边缘分布的方法。

不知道讲的是否明白，不明白就看书吧，一起研究🤔今天就讲到这了，大家有啥问题讨论下。

huajh7(284696304) 21:34:52

补充几点，一是 temporal model ,如 Dynamical Bayesian newtwork(DBN), plate models ，这是图模型的表达能力; 二是 belief Bropagation ，包括 exact 和 approximation ，loopy 时的收敛性; Inference 包括 MCMC,变分法。这是图模型在 tree 和 graph 的推理能力。三是 Structure Learning ，BIC score 等。这是图模型的学习能力。

第九章 Mixture Models and EM

主讲人 网络上的尼采

(新浪微博: @Nietzsche_复杂网络机器学习)

网络上的尼采(813394698) 9:10:56

今天的主要内容有 k-means、混合高斯模型、EM 算法。

对于 k-means 大家都不会陌生，非常经典的一个聚类算法，已经 50 多年了，关于 clustering 推荐一篇不错的 survey: Data clustering: 50 years beyond K-means。k-means 表达的思想非常经典，就是对于复杂问题分解成两步不停的迭代进行逼近，并且每一步相对于前一步都是递减的。

k-means 有个目标函数：

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

假设有 k 个簇， $\boldsymbol{\mu}_k$ 是第 k 个簇的均值；每个数据点都有一个向量表示属于哪个簇， r_{nk} 是向量的元素，如果点 $\mathbf{x}_n$ 属于第 k 个簇，则 r_{nk} 是 1，向量的其他元素是 0。

上面这个目标函数就是各个簇的点与簇均值的距离的总和，k-means 要做的就是使这个目标函数最小。这是个 NP-hard 问题，k-means 只能收敛到局部最优。

算法的步骤非常简单：

先随机选 k 个中心点

第一步也就是 E 步把离中心点近的数据点划分到这个簇里；

第二步 M 步根据各个簇里的数据点重新确定均值，也就是中心点。

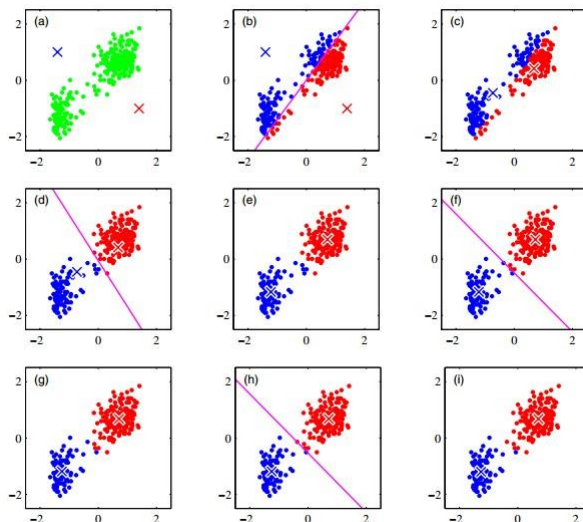
然后就是迭代第一步和第二步，直到满足收敛条件为止。

自强<ccab4209211@qq.com> 9:29:00

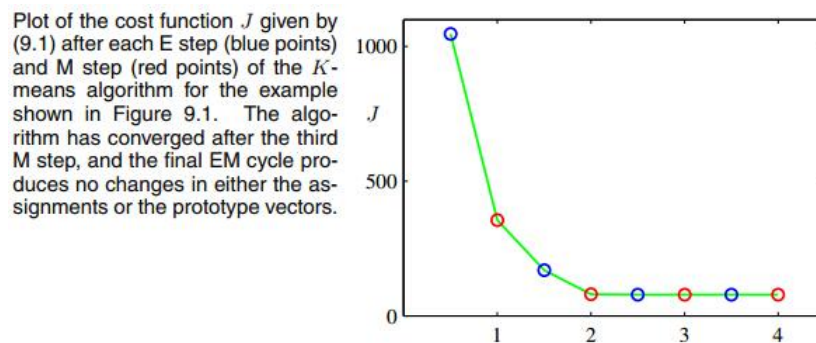
收敛是怎么判断的呀？

网络上的尼采(813394698) 9:30:16

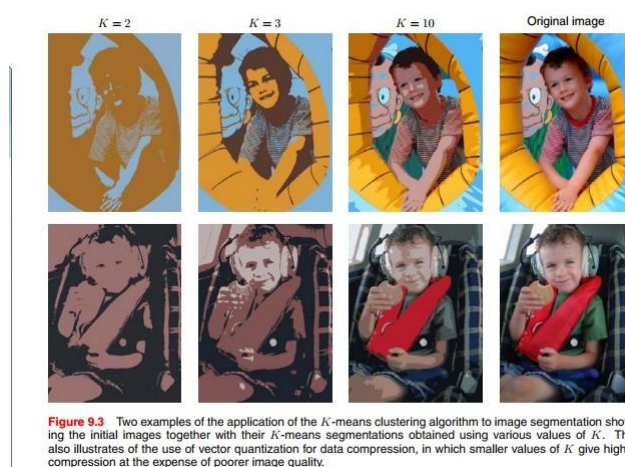
不再发生大的变化。大家思考下不难得出：无论 E 步还是 M 步，目标函数都比上一步是减少的。下面是划分两个簇的过程：



下面这个图说明聚类过程中目标函数单调递减，经过三轮迭代就收敛了，由于目标函数只减不增，并且有界，所以 k-means 是可以保证收敛的：



书里还举例一个 k-means 对图像分割和压缩的例子：



图像分割后，每个簇由均值来表示，每个像素只存储它属于哪个簇就行了。压缩后图像的大小是 k 的函数：

pixel intensity vectors we transmit the identity of the nearest vector μ_k . Because there are K such vectors, this requires $\log_2 K$ bits per pixel. We must also transmit the K code book vectors μ_k , which requires $24K$ bits, and so the total number bits required to transmit the image is $24K + N \log_2 K$ (rounding up to the nearest integer).

现在讨论下 k-means 的性质和不足：

首先对初值敏感，由于只能收敛到局部最优，初值很大程度上决定它收敛到哪里；

从算法的过程可以看出，k-means 对椭球形状的簇效果最好，不能发现任意形状的簇；

对孤立点干扰的鲁棒性差，孤立点是异质的，可以说是均值杀手，k-means 又是围绕着均值展开的，试想下，原离簇的孤立点对簇的均值的拉动作用是非常大的。

针对这些问题后来又有了基于密度的 DBSCAN 算法，最近 Science 上发了另一篇基于密度的方法：

Clustering by fast search and find of density peaks. 基于密度的方法无论对 clustering 还是 outliers detection 效果都不错，和 k-means 不同，这些算法已经没有了目标函数，但鲁棒性强，可以发现任意形状的簇。

另外如何自动确定 k-means 的 k 是目前研究较多的一个问题。k-means 就到这里，现在一块讨论下。

口水猫(465191936) 9:49:20

如果对于这批数据想做 k-mean 聚类，那么如果去换算距离？

网络上的尼采(813394698) 9:49:53

k-means 一般基于欧式距离，关于距离度量是个专门的方向，点集有了度量才能有拓扑，有专门的度量学习这个方向。

口水猫(465191936) 9:50:08

嗯嗯 有没有一些参考意见了, k 的选择可以参考 coursera 上的视频 选择 sse 下降最慢的那个拐点的 k
网络上的尼采(813394698) 9:51:41

关于选哪个 k 是最优的比较主观, 有从结果稳定性来考虑的, 毕竟一个算法首先要保证的是多次运行后结果相差不大, 关于这方面有一篇 survey: Clustering stability an overview。另外还有其他自动选 k 的方法, DP、MDL 什么的。MDL 是最短描述长度, 从压缩的角度来看 k-means; DP 是狄利克雷过程, 一种贝叶斯无参方法, 感兴趣可以看 JORDAN 小组的文章。

我们下面讲混合高斯模型 GMM, 第二章我们说过, 高斯分布有很多优点并且普遍存在, 但是单峰函数, 所以对于复杂的分布表达能力差, 我们可以用多个高斯分布的线性组合来逼近这些复杂的分布。我们看下 GMM 的线性组合形式:

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

对于每个数据点是哪个分布生成的, 我们假设有个 Z 隐变量, 和 k-means 类似, 对于每个数据点都有一个向量 z, 如果是由第 k 个分布生成, 元素 $z_k=1$, 其他为 0。

$z_k=1$ 的概率的先验就是高斯分布前的那个系数:

$$p(z_k = 1) = \pi_k$$

下面是 $z_k=1$ 概率的后验, 由贝叶斯公式推导的:

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} | z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x} | z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \end{aligned}$$

其实很好理解, 如果没有 π_k 限制, 数据点由哪个分布得出的概率大 $z_k=1$ 的期望就大, 但前面还有一个系数限制, 所以期望形式是:

$$\frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

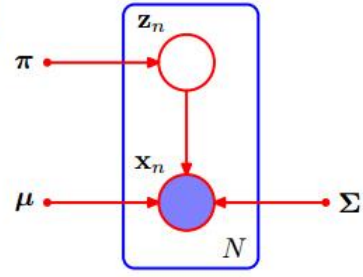
刚才有人问如何确定模型的参数, 我们首先想到的就是 log 最大似然:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

但是我们可以观察下这个目标函数, log 里面有加和, 求最优解是非常困难的, 混合高斯和单个高斯的参数求法差别很大, 如果里面有一个高斯分布坍塌成一个点, log 似然函数会趋于无穷大。由于直接求解困难, 这也是引入 EM 的原因。

下面是 GMM 的图表示:

Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{z_n\}$, where $n = 1, \dots, N$.



我们可以试想下，如果隐变量 z_n 是可以观测的，也就是知道哪个数据点是由哪个分布生成的，那么我们求解就会很方便，可以利用高斯分布直接得到解析解。但关键的是 z_n 是隐变量，我们没法观测到。但是我们可以用它的期望来表示。现在来看一下 EM 算法在 GMM 中的应用：

EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means μ_k , covariances Σ_k and mixing coefficients π_k , and evaluate the initial value of the log likelihood.
2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}. \quad (9.23)$$

上面是 EM 对 GMM 的 E 步

我们首先对模型的参数初始化

E 步就是我们利用这些参数得 z_{nk} 的期望，这种形式我们前面已经提到了：

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}$$

现在我们有隐藏变量的期望了，由期望得新的模型参数也就是 M 步，高斯分布的好处就在这儿，可以推导出新参数的闭式解：

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

然后不断迭代 E 步和 M 步直到满足收敛条件。

为什么要这么做，其实 EM 算法对我们前面提到的 log 最大似然目标函数：

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

是单调递增的。

karnon(447457116) 10:39:42

用 EM 来解 GMM 其实是有问题的，解出来的解并不是最优的。。

网络上的尼采(813394698) 10:40:14

嗯，这个问题最后讲。

我们再来看 EM 更一般的形式：

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) \text{ 是我们的目标函数，加入隐藏变量可以写成这种形式： } \ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \right\}$$

先初始化模型的参数，由于隐变量无法观测到，我们用参数来得到它的后验 $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$

然后呢，我们通过隐藏变量的期望得到新的完整数据的最大似然函数：

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

以上是 E 步，M 步是求这个似然函数的 Q 函数的最优解，也就是新的参数：
$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$$

注意这个 Q 函数是包含隐变量的完整数据的似然函数，不是我们一开始的目标函数 $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

其实求完整数据的最大似然是在逼近我们目标函数的局部最优解，这个在后面讲。

下面这个是一般化 EM 算法的步骤：

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

9.3. An Alternative View of EM

441

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \quad (9.32)$$

where

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \quad (9.33)$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \quad (9.34)$$

and return to step 2.

EM 算法之所以用途广泛在于有潜在变量的场合都能用，并不局限于用在 GMM 上。现在我们回过头来看混合高斯模型的 M 步，这些得到的新参数就是 Q 函数的最优解：

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad (9.24)$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T \quad (9.25)$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (9.27)$$

再思考下 k-means，其实它是 EM 的特例，只不过是 k-means 对数据点的分配是硬性的，在 E 步每个数据点必须分配到一个簇，z 里面只有一个 1 其他是 0，而 EM 用的是 z 的期望：

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] \rightarrow -\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 + \text{const.}$$

下面这个图说明 k-means 是 EM 算法特例，这与前面 k-means 聚类的过程的图对应：

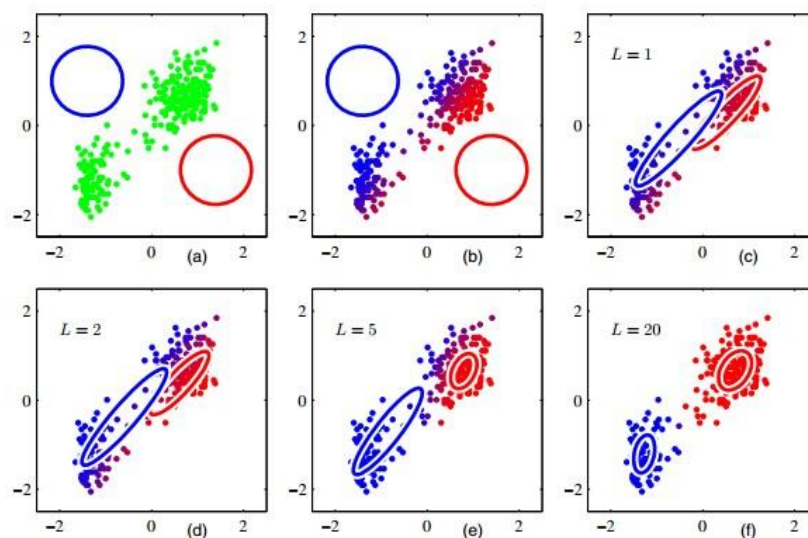


Figure 9.8 Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the *K*-means algorithm in Figure 9.1. See the text for details.

对于 EM 算法性质的证明最后讲，下面讲混合伯努利模型，高斯分布是针对连续的属性，伯努利是针对离散属性。混合形式：

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^K \pi_k p(\mathbf{x} | \boldsymbol{\mu}_k)$$

目标函数，log 里面同样有加和：

$$\ln p(\mathbf{X} | \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k p(\mathbf{x}_n | \boldsymbol{\mu}_k) \right\},$$

$z_{nk}=1$ 的期望：

$$\begin{aligned}\gamma(z_{nk}) = \mathbb{E}[z_{nk}] &= \frac{\sum_{z_{nj}} z_{nk} [\pi_k p(\mathbf{x}_n | \boldsymbol{\mu}_k)]^{z_{nk}}}{\sum_{z_{nj}} [\pi_j p(\mathbf{x}_n | \boldsymbol{\mu}_j)]^{z_{nj}}} \\ &= \frac{\pi_k p(\mathbf{x}_n | \boldsymbol{\mu}_k)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n | \boldsymbol{\mu}_j)}.\end{aligned}$$

Q 函数：

$$\begin{aligned}\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\pi})] &= \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \left\{ \ln \pi_k \right. \\ &\quad \left. + \sum_{i=1}^D [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})] \right\}\end{aligned}$$

以上是 E 步，下面是 M 步的解，这两个：

$$\boldsymbol{\mu}_k = \bar{\mathbf{x}}_k, \quad \pi_k = \frac{N_k}{N}$$

其中：

$$\begin{aligned}N_k &= \sum_{n=1}^N \gamma(z_{nk}) \\ \bar{\mathbf{x}}_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n\end{aligned}$$

书里举了一个手写字聚类的例子，先对像素二值化，然后聚成 3 个簇：

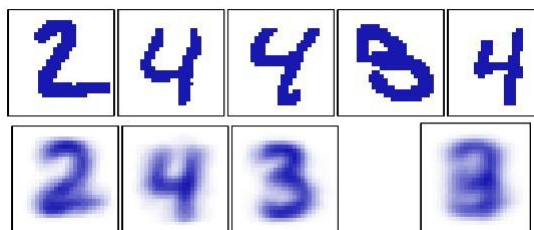


Figure 9.10 Illustration of the Bernoulli mixture model in which the top row shows examples from the digits c after associating the pixel values from gray scale to binary using a threshold of 0.5. On the bottom row the

这是 3 个簇的代表：



k=1 时簇的代表：



最后说下 EM 算法为什么能收敛到似然函数的局部最优解：

我们的目标函数是分布 $p(\mathbf{X} | \boldsymbol{\theta})$ 的最大 log 似然

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

引入潜在变量，分布表示为：

定义隐藏变量的分布为 $q(\mathbf{z})$ ，目标函数可以表达为下面形式，这个地方是神来一笔：

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q\|p)$$

where we have defined

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \\ \text{KL}(q\|p) &= - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}\end{aligned}$$

其中 $\text{KL}(q\|p)$ 是 $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ 与 $q(\mathbf{z})$ 的 KL 散度； $\mathcal{L}(q, \boldsymbol{\theta})$ 是 $q(\mathbf{z})$ 的泛函形式。

我们原来讲过根据 Jensen 不等式来证明 KL 散度是非负的，这个性质在这里发挥了作用。由于 $\text{KL}(q\|p)$ 是

大于等于零的，所以 $\mathcal{L}(q, \boldsymbol{\theta})$ 是目标函数 $\ln p(\mathbf{X}|\boldsymbol{\theta})$ 的下界。 $\mathcal{L}(q, \boldsymbol{\theta})$ 与目标函数什么时候相等呢？其实就

是 $\text{KL}(q\|p)$ 等于 0 的时候，也就是 $q(\mathbf{z})$ 与 $\mathbf{z}$ 的后验分布 $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ 相同时，这个时候就是 E 步 $\mathbf{z}$ 取它的期

望时候。 $\mathcal{L}(q, \boldsymbol{\theta})$ 与目标函数相等是为了取一个比较紧的 bound。

M 步就是最大化 $\mathcal{L}(q, \boldsymbol{\theta})$ ，随着 $\mathcal{L}(q, \boldsymbol{\theta})$ 的增大， $\text{KL}(q\|p)$ 开始大于 0，也就是目标函数比 $\mathcal{L}(q, \boldsymbol{\theta})$ 增大的

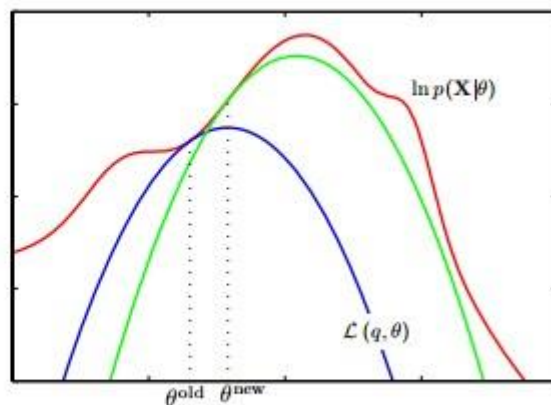
幅度更大。这个过程是使目标函数一直单调递增的。下面是 $\mathcal{L}(q, \boldsymbol{\theta})$ 与 Q 函数的关系，这也是为什么 M 步

新参数取完整数据的最大似然解的原因：

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \\ &= Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) + \text{const}\end{aligned}\tag{9.74}$$

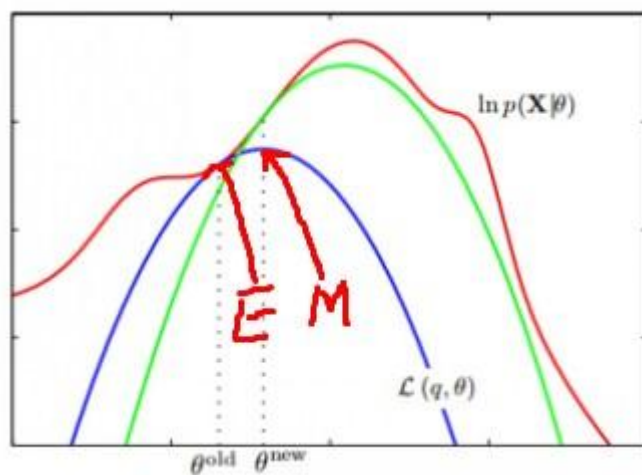
最后上一张非常形象的图，解释为什么 EM 能收敛到目标函数的局部最优：

The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



红的曲线是目标函数；蓝的绿的曲线是两步迭代。

咱们先看蓝的 E 步和 M 步：



E 步时就是取 z 的期望的时候，这时目标函数与 $\mathcal{L}(q, \theta)$ 相同；

M 步就是最大化 $\mathcal{L}(q, \theta)$

绿线是下一轮的迭代，EM 过程中，目标函数一直是单调上升的，并且有界，所以 EM 能够保证收敛。但不一定能收敛到最优解，这与初始值有很大关系，试想一下，目标函数的曲线变动下，EM 就有可能收敛到局部最优了。

第十章 Approximate Inference

主讲人 戴玮

(新浪微博: @戴玮_CASIA)

Wilbur_中博(1954123) 20:02:04

我们在前面看到，概率推断的核心任务就是计算某分布下的某个函数的期望、或者计算边缘概率分布、条件概率分布等等。比如前面在第九章尼采兄讲 EM 时，我们就计算了对数似然函数在隐变量后验分布下的期望。这些任务往往需要积分或求和操作。但在很多情况下，计算这些东西往往不那么容易。因为首先，我们积分中涉及的分布可能有很复杂的形式，这样就无法直接得到解析解，而我们当然希望分布是类似指数族分布这样具有共轭分布、容易得到解析解的分布形式；其次，我们要积分的变量空间可能有很高的维度，这样就把我们做数值积分的路都给堵死了。因为这两个原因，我们进行精确计算往往是不可行的。为了解决这一问题，我们需要引入一些近似计算方法。

近似计算有随机和确定两条路子。随机方法也就是 MCMC 之类的采样法，我们会在讲第十一章的时候专门讲到，而确定近似法就是我们这一章讲的变分。变分法的优点主要是：有解析解、计算开销较小、易于在大规模问题中应用。但它的缺点是推导出想要的形式比较困难。也就是说，人琢磨的部分比较复杂，而机器算的部分比较简单。这和第十一章的采样法的优缺点恰好有互补性。所以我们可以不同的场合应用变分法或采样法。这里我的一个问题是：是否可以结合二者的优点，使得人也不用考虑太多、机器算起来也比较简单？

变分法相当于把微积分从变量推广到函数上。我们都知道，微积分是用来分析变量变化、也就是函数性质的，这里函数定义为 $f: x \rightarrow f(x)$ ，而导数则是 df/dx ；与之相对，变分用到了泛函的概念： $F: f \rightarrow F(f)$ ，也就是把函数映射到某个值，而相应地，也有导数 dF/df ，衡量函数是如何变化的。比如我们熟悉的信息论中的熵，就是把概率分布这个函数映射到熵这个值上。和微积分一样，我们也可以通过导数为 0 的条件求解无约束极值问题，以及引入拉格朗日乘子来求解有约束极值问题。比如说，我们可以通过概率分布积分为 1 的约束，求解最大熵的变分问题。PRML 的附录 D 和 E 有比较详细的解释，我们后面也还会看到，这里就不多说了。

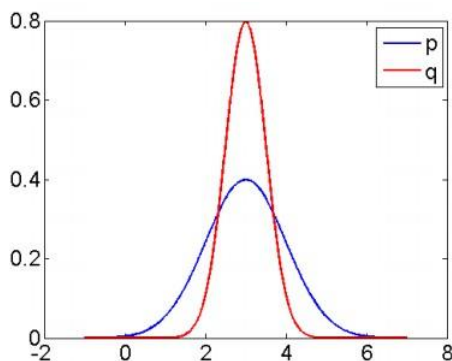
变分法这名字听起来比较可怕，但它的核心思想，就是从某个函数空间中找到满足某些条件或约束的函数。我们在统计推断当中用到的变分法，实际上就是用形式简单的分布，去近似形式复杂、不易计算的分布，这样再做积分运算就会容易很多。比如，我们可以在所有高斯分布当中，选一个和目标分布最相似的分布，这样后面做进一步计算时就容易获得解析解。此外，我们还可以假设多元分布的各变量之间独立，这样积分的时候就可以把它们变成多个一元积分，从而解决高维问题。这也是最简单的两种近似。

概率推断中的变分近似方法，最根本的思想，就是想用形式简单的分布去近似形式复杂、不易计算的分布。比如，我们可以在指数族函数空间当中，选一个和目标分布最相像的分布，这样计算起来就方便多了。显然，我们这里需要一个衡量分布之间相似性或差异性的度量，然后我们才能针对这个度量进行最优化，求相似性最大或差异性最小的分布。一般情况下，我们会选用 KL 散度：

$$KL(q||p) = E_{q(x)}[\log(q(x)) - \log(p(x))]$$

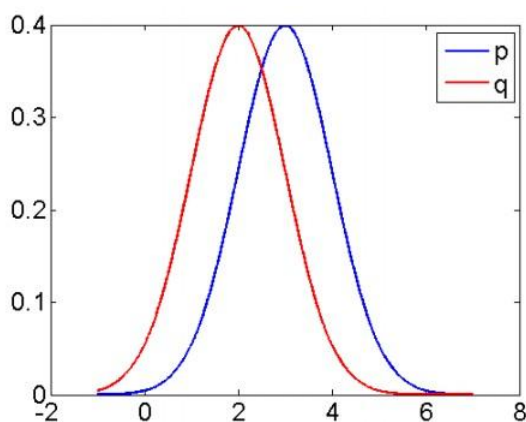
或者 $KL(q||p) = \int q(x) \log \frac{q(x)}{p(x)} dx$ ，当然离散分布就不是积分而是在离散状态上求和。这个值是非负的，而且只在两分布完全相同的情况下取 0，所以可以看成两分布之间的距离。但这种度量是不对称的，也就是 $KL(q||p) \neq KL(p||q)$ ，而我们在优化的时候，这两种度量实际上都可以使用。这样一来，

我们后面也会看到，会造成一些有趣且奇怪的现象。有了这个度量，我们就可以对某个给定的概率分布，求一个在某些条件下和它最相似或距离最小的分布。这里我们看几个例子，直观地认识一下 KL 散度的不对称性、以及产生这种不对称性的原因。这是两个方差不同的一元高斯分布，其中方差较小的是 q （红色曲线），方差较大的是 p （蓝色曲线）：

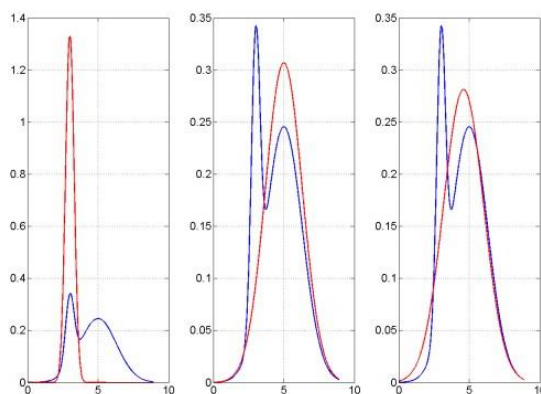


$$KL(q||p) = \int q(x) \log \frac{q(x)}{p(x)} dx$$

根据 KL 散度的公式，我们能否估计一下，是 $KL(q||p)$ 较大，还是 $KL(p||q)$ 较大？我们可以看到，在曲线的中间部分， $q(x) > p(x)$ ，因此，如果光考虑这部分，显然 $KL(q||p)$ 会比较大。但是，考虑两边 $q(x) < p(x)$ 的部分，我们可以看到， $q(x)$ 很快趋近于 0，此时 $p(x)/q(x)$ 会变得很大，比中间部分要大得多（打个比方， $0.8/0.4$ 和 $0.01/0.001$ ）。尽管还要考虑 $\log$ 前面的 $q(x)$ ，但当 $q(x)$ 不等于 0 时，分母趋近于 0 造成的影响还是压倒性的。所以综合考虑， $KL(q||p)$ 要小于 $KL(p||q)$ 。它们的精确值分别为 0.32 和 0.81。另一个例子是，如果两个高斯分布方差相等，则 KL 散度也会相等：

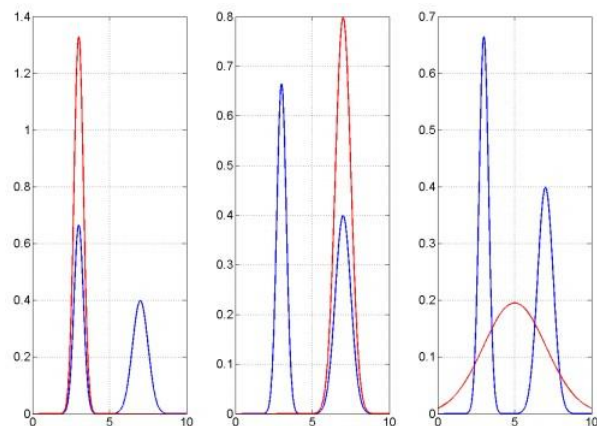


这一点很容易理解。再来看一个复杂一点的例子。在这个例子中， q 是单峰高斯分布， p 是双峰高斯分布：



这三种情况中， p 的两个峰没有分开，有一定粘连，而 q 则分别拟合了 p 的左峰、右峰（见 PRML 4.4 节）

的拉普拉斯近似，上次读书会也简单介绍过，可参看上次读书会的总结），以及拟合 p 的均值和方差（即单峰高斯分布的两个参数）。三种拟合情况对应左、中、右三图。对于这三种情况， $KL(q||p)$ 分别为 1.17、0.09、0.07， $KL(p||q)$ 分别为 23.2、0.12、0.07。可以看到，无论是哪一种 KL 散度，在 p 的双峰没有完全分开的情况下，用单峰高斯 q 去近似双峰高斯 p 得到的最优解，都相当于拟合 p 的均值和方差。如果 p 的两个峰分开的话，情况会如何呢？

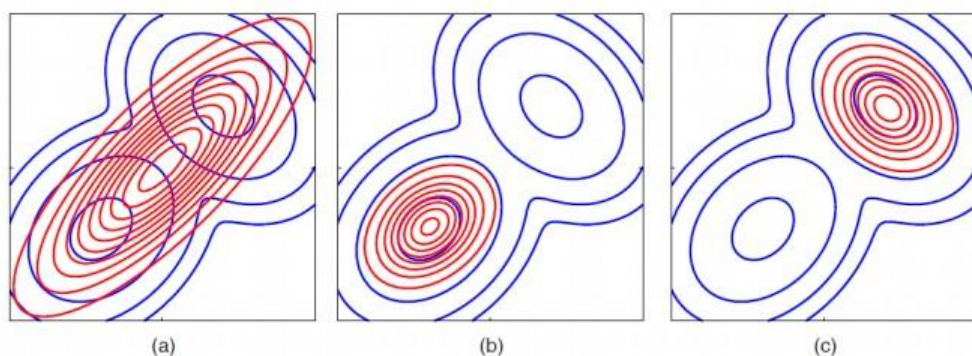


和前一个例子一样，我们分别拟合 p 的左峰、右峰，以及均值和方差。显然，这里由于 p 中间有一段概率密度为 0 的区域，所以可以想见， $KL(q||p)$ 可能会比较大。实际情况也是如此： $KL(q||p)$ 分别为 0.69、0.69、3.45， $KL(p||q)$ 分别为 43.9、15.4、0.97。可以看到，如果用 $KL(p||q)$ 做最优化，结果和双峰粘连时一样，仍然是拟合 p 的均值和方差，也就是所谓的 moment-matching；而用 $KL(q||p)$ 做最优化，结果则会有所变化：会拟合双峰的其中一峰，也就是所谓的 mode-seeking。

我们从前面这几个例子中，可以总结一个规律：用 $KL(q||p)$ 做最优化，是希望 $p(x)$ 为 0 的地方 $q(x)$ 也要为 0，否则 $q(x)/p(x)$ 就会很大，刚才例子的右图在中间部分（5 附近）就违背了这一点；反之，如果用 $KL(p||q)$ 做最优化，就要尽量避免 $p(x)$ 不为 0 而 $q(x)$ 用 0 去拟合的情况，或者说 $p(x)$ 不为 0 的地方 $q(x)$ 也不要为 0，刚才例子的左、中两图也违反了这一点。

所以， $KL(q||p)$ 得到的近似分布 $q(x)$ 会比较窄，因为它希望 $q(x)$ 为 0 的地方可能比较多；而 $KL(p||q)$ 得到的近似分布 $q(x)$ 会比较宽，因为它希望 $q(x)$ 不为 0 的地方比较多。

最后看一个多元高斯分布的例子，书上的图 10.3：



即有了前面的讲解，我们可以猜一下，哪些图是 $KL(q||p)$ 得到的最优解，哪些图是 $KL(p||q)$ 得到的最优解。由于 $KL(q||p)$ 至少可以拟合到其中的一个峰上，而 $KL(p||q)$ 拟合的结果，其概率密度最大的地方可能没什么意义，所以很多情况下， $KL(q||p)$ 得到的结果更符合我们的需要。到这里有什么问题吗。。理解理解。。KL 散度这东西。

=====讨论=====

飞羽(346723494) 20:24:23

$KL(q||p)$ 就是相当于用 q 去拟合 p ？

Yuli(764794071) 20:25:31

KL 就是 KL Divergence (相对熵) 吧 用信息论来解释的话 是用来衡量两个正函数是否相似

飞羽(346723494) 20:25:57

对, 就是相对熵

Wilbur_中博(1954123) 20:27:06

嗯, 我们现在有一个分布 p , 很多时候是后验分布, 但它形式复杂, 所以想用形式比较简单的 q 去近似 p 。其实也可以直接用后验分布的统计量, 比如 mode 或 mean 去代替整个分布, 进行进一步计算, 比如最大后验什么的。但现在如果用近似分布去做预测的话, 性能会好得多。

linbo-phd-bayesian(99878724) 20:27:15

请问为何 $KL(q||p) = 0$, 为何没有 $KL(p||q)$ 啊, 有知道的吗?

飞羽(346723494) 20:28:06

相对熵的值为非负数:

$$D_{KL}(P||Q) \geq 0,$$

由吉布斯不等式 (en:Gibbs' inequality) 可知, 当且仅当 $P = Q$ 时 $D_{KL}(P||Q)$ 为零。

尽管从直觉上 KL 散度是个度量或距离函数, 但是它实际上并不是一个真正的度量或距离。因为 KL 散度不具有对称性: 从分布 P 到 Q 的距离 (或度量) 通常并不等于从 Q 到 P 的距离 (或度量)。

$$D_{KL}(P||Q) \neq D_{KL}(Q||P)$$

Wilbur_中博(1954123) 20:29:21

那个不太难证, 利用 $\ln$ 凹函数性质可以证出来。。不过细节我忘记了, 呵呵。查一查吧。。应该很多地方都有的。

逸风(421723497) 20:30:44

PRML P56

Wilbur_中博(1954123) 20:31:50

总之就是利用 KL 作为目标函数, 去做最优化。。找到和已知复杂分布最相近的一个近似分布。这一章的基本思路就是这样。具体动机最开始的时候已经提到过了。

逸风(421723497) 20:35:31

为什么要用 KL 散度这样一个不具备对称性的“距离”, 而不采用对称性的测度呢? 有什么好处?

Wilbur_中博(1954123) 20:37:15

似乎没有特别好的对称的度量? PRML 的公式(10.20)提过一种叫 Hellinger distance 的度量, 是对称的, 但后来也没有用这个。不知道为什么。不容易优化? 有没有了解原因的朋友。。比如说, 为啥不用 $(p(x) - q(x))^2$ 做积分作为度量? 或者其他什么的。

WayneZhang(824976094) 20:41:52

我感觉是优化求解过程中近似时自然而然导出了 KL 这个度量。

karnon(447457116) 20:42:24

KL 算是熵的增益, 所以一定是那种形式, 这取决于你怎么定义“近似”, 认为信息增益最少就是“近似”也是一种合理的定义

Wilbur_中博(1954123) 20:43:07

这里目的是为了找近似分布

=====讨论结束=====

我们在 PRML 这本书的 4.4 节, 其实看到过一种简单的近似方法, 或者可以说是最简单的近似方法之一: 拉普拉斯近似。它是用高斯分布去近似目标分布的极值点也就是 mode。这里并没有涉及到变分的概念。它只是要求高斯分布的 mode 和目标分布的 mode 位置相同, 方法就是把目标分布在 mode 处做泰勒级

数展开到第二阶，然后用对应的高斯分布去代替，就是把未知系数给凑出来：

$$\mathcal{L}(\theta^*) + \frac{1}{2} \mathcal{L}''(\theta^*) (\theta - \theta^*)^2$$

这是目标分布在 θ^* (mode) 的二阶泰勒展开：

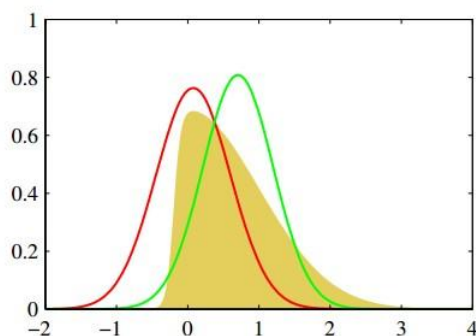
$$\begin{aligned} \ln \mathcal{N}(\theta | \mu, \eta^{-1}) &= \frac{1}{2} \ln \eta - \frac{1}{2} \ln 2\pi - \frac{\eta}{2} (\theta - \mu)^2 \\ &= \frac{1}{2} \ln \frac{\eta}{2\pi} + \frac{1}{2} (-\eta) (\theta - \mu)^2 \end{aligned}$$

一比较就知道高斯分布的两个参数应该取：

$$\mu = \theta^*$$

$$\eta = -\mathcal{L}''(\theta^*)$$

也就是 PRML 图 10.1 的红线：



棕色部分是目标分布，绿线是我们用变分近似，在高斯分布中选一个和目标函数的 KL 散度最小的分布。反正就均值和方差两个未知参数，优化起来应该不难。

下面开始讲 10.1.1 可分解分布，这一节非常重要，可以说是本章的基础和最重点的部分。基本思想就是，我们把近似分布限制在可分解分布的范围内，也就是满足(10.5)式：

$$q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i)$$

可以说，这个分布的各组变量 $\mathbf{Z}_i$ 互相之间是独立的。这样一来，我们计算这个分布的积分时，就可以变成多个较低维度的积分，就算用数值积分什么的也会简单很多。在统计物理当中，这种可分解形式的变分近似方法称作平均场 (mean field) 方法，这个名字实际上是很直观的，和它最后得到的解的形式有关，我们马上会看到。不过现在不仅在统计物理领域，机器学习很多时候也就管它叫 mean field 了。现在很火的 RBM 什么的，求参数时经常能看到这个术语。

上一章曾经讲过，最小化 KL 距离，和最大化下界 $L(q)$ 是一回事，也就是(10.2)到(10.4)这三个式子：

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \text{KL}(q||p)$$

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$$

$$\text{KL}(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}.$$

这和 9.4 节当中(9.70)到(9.72)实际上是一样的，区别在于 Z 不仅是隐变量还把参数吸收了进来。等式左边那项和我们想求的 Z 无关，所以可以看成常数，而右边的 $p(Z|X)$ 是我们想去近似的，不知道具体形式，所以可以间接通过最大化右边第一项来达到最小化右边第二项也就是 KL 散度的目的。

根据上面的(10.5)式会得到公式(10.6)：

$$\begin{aligned}\mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} d\mathbf{Z} \\ &= \int q_j \left\{ \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right\} d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \quad (10.6)\end{aligned}$$

我们这里也可以看 MLAPP 的(21.28)到(21.31)：

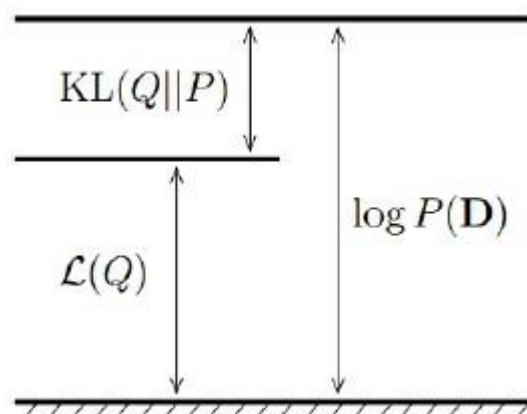
$$\begin{aligned}L(q_j) &= \sum_{\mathbf{x}} \prod_i q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] \\ &= \sum_{\mathbf{x}_j} \sum_{\mathbf{x}_{-j}} q_j(\mathbf{x}_j) \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\log \tilde{p}(\mathbf{x}) - \sum_k \log q_k(\mathbf{x}_k) \right] \\ &= \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) \\ &\quad - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\sum_{k \neq j} \log q_k(\mathbf{x}_k) + q_j(\mathbf{x}_j) \right] \\ &= \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log f_j(\mathbf{x}_j) - \sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \log q_j(\mathbf{x}_j) + \text{const}\end{aligned}$$

推导得要详细得多。。所以多备几本参考书是必要的。

MLAPP 是 Machine Learning - A Probabilistic Perspective 的缩写。。群共享里应该有吧。很不错的机器学习书。

huajh7(284696304) 21:02:10

插一句。这里优化的目标其实是最大化 low bound $L(q)$ ($\log P(D)$ 是对数证据，常数， $KL(Q||P)=0$ 时， $L(Q)$ 最大)。也就是找到一个最合适的 q 分布，而不是优化参数。优化过程中，求导，拉格朗日什么，是针对 q 分布的，也就是泛函。这是为什么叫变分法：



Wilbur_中博(1954123) 21:03:04

好，谢谢。我看了你的博客 <http://www.blog.huajh7.com/variational-bayes/>，文章写得很好。你好像毕业论文就是专门做这个的吧？也许你下次可以再专门讲一讲你对变分近似的心得体会，呵呵。
简单说，这里的推导就是每一步只看 q_j 相关的那些项，和 q_j 无关的项全都归到常数里去。比如(21.30)的这部分：

$$\sum_{\mathbf{x}_j} q_j(\mathbf{x}_j) \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \left[\sum_{k \neq j} \log q_k(\mathbf{x}_k) \right]$$

实际上就全扔到常数里去了。哦。。还少了个(21.32)：

where

$$\log f_j(\mathbf{x}_j) \triangleq \sum_{\mathbf{x}_{-j}} \prod_{i \neq j} q_i(\mathbf{x}_i) \log \tilde{p}(\mathbf{x}) = \mathbb{E}_{-q_j} [\log \tilde{p}(\mathbf{x})]$$

这里我们是在除了 $\mathbf{x}_j$ 之外的其他 $\mathbf{x}_i$ 上求期望，也就是这个东西：

$\mathbb{E}_{-q_j} [\log \tilde{p}(\mathbf{x})]$ ，它是关于 $\mathbf{x}_j$ 的函数。

下面讲一下 10.1.1 的可分解分布，也就是刚才说过的，假设多元分布可分解为多个一元分布的乘积，即

用 $q(\mathbf{x}) = \prod_i q_i(\mathbf{x}_i)$ 去近似 $p(\mathbf{x})$ 。由于各个变量之间是解耦的，所以我们可以每次只关注单个变量的最优化，也就是用所谓坐标下降 (coordinate descent) 的方式来做最优化。具体做法，就是把最小化 KL 散度转化为最大化 $L(q)$ (参见公式(10.2)到(10.4))，然后把公式(10.5)代入(10.3)，每次把 $L(q)$ 其中一个 q_j 当做变量，而把其他 q_i 当做常数，对 $L(q)$ 进行最优化：

$$\begin{aligned} \mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} d\mathbf{Z} \\ &= \int q_j \left\{ \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right\} d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \end{aligned} \quad (10.6)$$

$$\text{这里：} \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const.} \quad (10.7)$$

前面讲过，KL 散度也可以写成： $KL(q||p) = \mathbb{E}_{q(x)} [\log(q(x)) - \log(p(x))]$ ，可以看到，(10.6)最

后得到的这个 $\int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j$ ，恰好是负 KL 散度的形式。我们知道，KL 散度为 0 也就是最小的时候，两分布恰好相同，因此每一步的最优化结果可得到：

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const.}$$

也就是每一步更新的结果，可得到分解出来的变量的分布为：

$$q_j^*(\mathbf{Z}_j) = \frac{\exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})]) d\mathbf{Z}_j}.$$

就是两边都取 exp 然后归一化。由于是以其他变量的均值作为当前变量分布的形式，所以这种方法也称作 mean-field。这部分内容也可以参见 MLAPP 的 21.3.1，那一节讲得感觉比 PRML 清楚一些。那个公式是比较头疼。。不过只要记住只有一个 q_j 是变量，其他都当成常数，推一推应该也 ok。

重新回顾下前面的内容：

变分推断的核心思想是：用形式比较简单、做积分运算比较容易的分布，去替代原先形式复杂、不易求积分的分布。因此，这里的主要问题就是：如何找到和原分布近似程度较高的简单分布。前面我们讲了一些变分推断的背景知识和 KL divergence (KLD) 的相关知识，还稍微讲了讲假设分布可分解时是如何推导出 mean field 形式的。KLD 是衡量两个分布差异大小的方式之一，KLD 越大则差异越大，反之则两分布越相似。因此，我们可以将 $KL(q||p)$ 作为目标函数，并限定 q 为较简单的分布形式，找到这类分布中最接近原分布 p 的那个分布。我们这里主要关注的近似对象是后验分布。因为我们前面一直在讲如何求后验分布，但后验分布求出来的形式往往不那么好用，所以需要简单分布去近似。然而，计算 $p(Z|X)$ 需要计算归一化因子 $p(X)$ 。 $p(X)$ 是边缘分布，需要对 $p(Z,X)$ 做积分，而 $p(Z,X)$ 又不那么容易积分。因此，我们可以直接用未归一化的 $p(Z,X)$ 作为近似计算的目标，也就是下面这个关系：

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + KL(q||p) \quad (10.2)$$

其中：

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z} \quad (10.3)$$

$$KL(q||p) = - \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}. \quad (10.4)$$

这里 $\ln(p(X))$ 只是个常数，所以极小化 KLD 和极大化 L 得到的结果是一样的，但对 L 做最优化可直接用联合概率分布去做、而不用归一化。：我们想要得到的简单分布具有什么样的形式？我们喜欢的一种简单分布是可分解分布，就是说，我们可以假设各个隐变量 Z_i 之间是独立的，因此可拆成各隐变量分布的乘积：

$$q(\mathbf{Z}) = \prod_{i=1}^M q_i(\mathbf{Z}_i). \quad (10.5)$$

那么，各个隐变量的 L 可写为：

$$\begin{aligned} \mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(\mathbf{X}, \mathbf{Z}) - \sum_i \ln q_i \right\} d\mathbf{Z} \\ &= \int q_j \left\{ \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i \right\} d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \\ &= \int q_j \ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) d\mathbf{Z}_j - \int q_j \ln q_j d\mathbf{Z}_j + \text{const} \end{aligned} \quad (10.6)$$

其中

$$\ln \tilde{p}(\mathbf{X}, \mathbf{Z}_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const}. \quad (10.7)$$

这里 $\mathbb{E}_{i \neq j}$ 表示：

$$\mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] = \int \ln p(\mathbf{X}, \mathbf{Z}) \prod_{i \neq j} q_i d\mathbf{Z}_i. \quad (10.8)$$

这是对 Z_j 之外的其他所有随机变量求期望，也叫做 mean field。极大化 L 相当于极小化 $KL(q_j || \tilde{p})$ ，显然 q_j 取和 $\tilde{p}$ 完全相同的形式时，KLD 极小，同时 L 极大。所以我们有最优解：

$$\ln q_j^*(Z_j) = \mathbb{E}_{i \neq j} [\ln p(\mathbf{X}, \mathbf{Z})] + \text{const.} \quad (10.9)$$

这里 p 是已知的，所以可对它做积分。对除 Z_j 以外的随机变量求期望得到的分布，就是分解出来的 q_j 的分布。我们每一步迭代都对每一个分解分布 q_j 进行求解。这种方法也称做 coordinate descent。

=====讨论=====

一夏 吕(992463596) 21:15:06

10.6 后面有，10.7 的 const 没有必要吧？我当时好像看懂了 做的笔记 现在一下看不懂了。。后面那个很简单 是因为 其他的 Z_i 积分为 1。我记起来了，把后面的 $\ln q_i$ 的和拆开，只把 j 的那一项留着，其他的都可以积分积掉，划到 const 里，这里主要是把 j 的那一项拿出来表示，其他的 不相干的都不管。

huajh7(284696304) 21:25:23

有必要吧。否则不相等了，这里 const 表示归一化常量。实际上需要特别注意 const，尤其自己推导的时候，const 更多是表示与 z_j 无关的量，而不是指一个常量。在概率图中就是不在 z_j 的马尔科夫毯上的量。

阿邦(1549614810) 21:26:08

mean filed 看 koller 的最清楚

一夏 吕(992463596) 21:26:57

注意 Z 是大写，所以 j 的那个积分里 其他的 i 都积分为 1 了。

huajh7(284696304) 21:27:48

const 有必要。exp($E_{i \sim j}[\dots]$) 是没有归一化的。

一夏 吕(992463596) 21:28:23

哪里有 exp

huajh7(284696304) 21:28:32

ln .

软件所-张巍<zh3f@qq.com> 21:26:42

问个问题：用分解的分布去近似原始分布，精度怎么保证，有没有直观点的解释。

Wilbur_中博(1954123) 21:28:32

@软件所-张巍 的问题是好问题啊。。一般来说，似乎是把变分近似看作在 MAP 和贝叶斯推断（用整个分布）之间的一种 trade-off？

huajh7(284696304) 21:29:54

给个图：

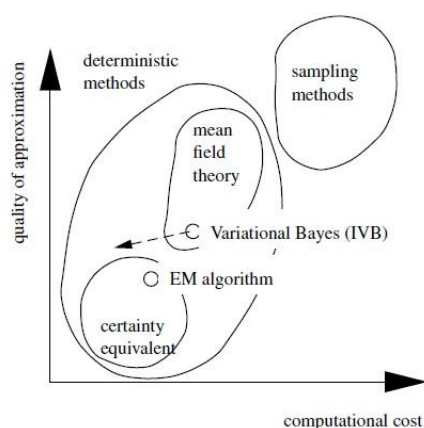


Fig. 1.3. The accuracy-vs-complexity trade-off in the VB method.

Wilbur_中博(1954123) 21:30:01

因为一个是用后验分布的点估计，一个是用整个分布，不错，这是哪里的图？

huajh7(284696304) 21:32:23

variational bayesian 可以说是分布式 distributional approximation，也就是 wilbur 说的，用的是整个分布。The Variational Bayes Method in Signal Processing 这本书的 第 9 页。

李笑一(94755934) 21:32:46

@张巍，我记得变分法能保证收敛到 local minimum。一般情况下，最大似然是 non convex 的，但是变分下界却是 convex，下界的 minimum 就是下一步要前进的方向。

一夏 吕(992463596) 21:33:23

但是变分法的前提是把 dependence 去掉了，这样才能把总概率拆成各自概率的积。即使是 convex 的，也只是逼近原先 intractable 的形式。10.7 的那个我还是觉得 const 没必要。

Wilbur_中博(1954123) 21:36:04

其实就是没归一化的，所以要加个 const，(10.9)那个也是这样

一夏 吕(992463596) 21:36:17

后面那个是求期望，就是上面那个花括号里的

李笑一(94755934) 21:36:30

@huajh7，图上看来，EM 更好使？？？

一夏 吕(992463596) 21:37:16

EM 是可解的时候用的，只是有隐变量

秦淮/sun 人家(76961223) 21:37:31

EM 是可以求得精确地后验

Happy(467594602) 21:38:30

直观解释 请参照 jordan 写的 introduction

huajh7(284696304) 21:38:33

10.6-10.9 就是利用 $KL(q||p) = 0$

$$q_j^*(Z_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})])}{\int \exp(\mathbb{E}_{i \neq j}[\ln p(\mathbf{X}, \mathbf{Z})]) dZ_j}.$$

分母就是 const，VB 也可以看成是 EM。

一夏 吕(992463596) 21:41:00

@Happy jordan 的 introduction 是他的那本书吗？

Happy(467594602) 21:41:20

introduction to variational methods in graphical model

用简单分布的族 把复杂分布包裹起来，然后复杂分布的每一点都有一个简单分布的参数来近似

一夏 吕(992463596) 21:42:47

thanks 他还有一本书 是 Graphical Models, Exponential Families, and Variational Inference

huajh7(284696304) 21:43:25

Neal,Hinton A view of the EM algorithm that justifies incremental, sparse, and other variants.pdf

这篇文章 说 EM，其实就是变分贝叶斯。

Happy(467594602) 21:43:25

后一本太难了。。

李笑一(94755934) 21:43:26

@huajh, 弱弱的问一下, 分母将 Z marginalize 掉这步只是在推导中出现是吧, 编程的时候不会出现实际的过程?

huajh7(284696304) 21:44:06

写程序的时候, 还是还归一化的。比如, GMM 中的隐变量, 全部算出来之后, 然后再归一化。

一夏 吕(992463596) 21:45:37

如果隐变量很多不是 exponential 个组合了

huajh7(284696304) 21:46:01

就转化为 exponential

Wilbur_中博(1954123) 21:46:01

mean field 的过程中呢? 每个 Z_j 的分布也都要归一化么? @huajh7

Happy(467594602) 21:46:16

我咋记得不用归一化。。mean-field

一夏 吕(992463596) 21:46:18

那就很费时间

huajh7(284696304) 21:46:45

后来会知道。算的是充分统计量。

一夏 吕(992463596) 21:46:46

如果有 64 个, 每个 01 分布就是 2^{64} 次方

李笑一(94755934) 21:47:33

恩, partition function 永远是问题

Happy(467594602) 21:47:35

程序里面没有归一化步骤吧, 推导中体现了

李笑一(94755934) 21:48:12

啥叫充分统计量?

huajh7(284696304) 21:48:16

概率才归一化啊。

Happy(467594602) 21:48:30

指数族里面有

huajh7(284696304) 21:48:35

充分统计量能完全表示一个分布。对

一夏 吕(992463596) 21:49:29

不用归一化吧, 看 10.9 10.10 中间那个公式下面那段话

huajh7(284696304) 21:49:42

为什么是指数族。。。一个最主要的原因就是其充分统计量是可计算的

Happy(467594602) 21:50:03

这个 jordan 后面那个书有深入介绍。。

一夏 吕(992463596) 21:51:09

通常不要求出分布, 而是得到分布的类型和参数

huajh7(284696304) 21:51:33

@一夏 吕 可能理解不一样。归一化不是指计算那个积分(partition function)。。

一夏 吕(992463596) 21:51:41

通常就是指数族, 自然服从积分为 1

Happy(467594602) 21:52:25

没有自然哈 也有归一化系数

一夏 吕(992463596) 21:53:00

恩 但是那个是和分布本身有关的，知道了参数就可以推，比如高斯的方差

李笑一(94755934) 21:56:38

这部分有没有类似的书写的不错的。直接讲替代教材得了

Happy(467594602) 21:57:29

jordan 那个不错，不过主要是针对 graphical model 的

Wilbur_中博(1954123) 21:58:14

我除了 PRML 和 MLAPP，还看了一下 Bayesian Reasoning and Machine Learning 的最后一章

一夏 吕(992463596) 21:58:16

有看多 lda 的吗 那个里面的 variational inference 和这个方法完全不同

Happy(467594602) 21:58:28

肿么不同。。

秦淮/sun 人家(76961223) 21:58:33

@一夏 吕 其实是一样的

huajh7(284696304) 21:58:33

bishop 不喜欢详细推导的。讲清楚就行。这里有篇：

A Tutorial on Variational Bayesian Inference (http://staffwww.dcs.shef.ac.uk/people/c.fox/fox_vbtut.pdf) 还是很清楚的。LDA。其实是一样的。。建立的图模型上，比较直观。

秦淮/sun 人家(76961223) 21:59:37

LDA 那篇文章就是使用的 mean field

一夏 吕(992463596) 21:59:38

blei 用拉格朗日乘子法做的。。

Happy(467594602) 21:59:46

一样的啊。。

秦淮/sun 人家(76961223) 22:00:02

不同的优化方法而已.....

huajh7(284696304) 22:00:03

嗯。其实是一样的。

秦淮/sun 人家(76961223) 22:00:12

本质是一样的

Happy(467594602) 22:00:21

直觉一致

秦淮/sun 人家(76961223) 22:00:26

不一样的是 Expectation propagation 那篇

huajh7(284696304) 22:00:37

对。那个感觉有些难。

一夏 吕(992463596) 22:00:40

恩 也是搞 kl 距离 方法各不相同

Happy(467594602) 22:01:52

对这些有兴趣就看 jordan 的大作吧，这些全部都归到架构里去了

一夏 吕(992463596) 22:05:45

<http://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf>

推荐这个 blei 的讲义

一夏 吕(992463596) 22:09:18

variational inference 是不是只是对指数族的才有用?

Happy(467594602) 22:09:26

一样的 统计模型下

一夏 吕(992463596) 22:10:04

我一般只在贝叶斯学派的文章里见到,一般都用 Gibbs sampling

Happy(467594602) 22:10:16

也不一定。。

一夏 吕(992463596) 22:11:00

比如 rbm 就不能用 variational inference

Happy(467594602) 22:11:21

可以啊, mean-field 必须可以用

天际一线(1002621044) 22:19:40

lda 那个话题模型 谁有完整的算法啊

Happy(467594602) 22:23:48

lda 老模型了吧。。程序应该多如牛毛

秦淮/sun 人家(76961223) 22:24:22

对啊, mean field , expectation propagation , gibbs sampling , distributed , online 的都有一堆

Matrix(690119781) 22:28:18

https://github.com/sudar/Yahoo_LDA 这个可能满足你的要求

陪你听风(407365525) 22:31:18

在效果上, variational inference , gibbs sampling 两个谁更好呢

秦淮/sun 人家(76961223) 22:38:35

sampling 近似效果好, 慢, 不好分布式计算

陪你听风(407365525) 22:39:08

vb 比较容易分布式吗

huajh7(284696304) 22:40:24

噗。VB 是可以很自然地分布式的。。

李笑一(94755934) 22:42:16

弱问。。VB 为啥自然可以用分布式

huajh7(284696304) 22:45:22

利用 variational message passing 框架下即可。。节点之间传递充分统计量。充分统计量(一阶矩, 二阶矩)的 consensus 或 diffusion 是有较多 paper 的。图模型中的 BP 或 loopy BP 算一种分布式嘛?

李笑一(94755934) 22:48:28

有个问题, 不同问题的 vb 是否需要自己推导出来? 不能随意套用别人的推导呢?

huajh7(284696304) 22:49:05

推导框架。如出一辙。。但自己推并不容易的。

李笑一(94755934) 23:04:01

karnon, 一篇 jmlr 的文章, 在一个问题上证了 vb 的全局解

Global Analytic Solution of Fully-observed Variational Bayesian Matrix Factorization

看明白了给讲讲。。

huajh7(284696304) 23:10:47

2,3 年前就出来了。。这篇估计是 combined and extended。

light(513617306) 23:15:13

这个是证明了在矩阵分解这个问题上的全局最有，不证明在其他模型上也是这样。》？

karnon(447457116) 23:15:34

这就已经很牛了

李笑一(94755934) 23:17:43

vb 对于不同问题有不同的解，我觉得除非熟到一定程度了，否则不可能拿来一个问题就能用 vb 的

karnon(447457116) 23:21:29

我看看，我知道最近有些文章研究全局收敛的矩阵分解问题，粗翻了一下，好像说的是把 vb 转成一个等价的 svd 问题？

=====讨论结束=====

接着主要讲几个变分推断的例子，试图阐述清楚变分推断到底是如何应用的。首先是二元高斯分布的近似。我们假设二元高斯分布是可分解的，也就是两变量之间独立。

二元高斯分布 $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$

其中

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix} \quad (10.10)$$

可分解形式为： $q(\mathbf{z}) = q_1(z_1)q_2(z_2)$

我们想用 $q(\mathbf{z})$ 去近似 $p(\mathbf{z})$ ，用前面推导出来的(10.9)：

$$\begin{aligned} \ln q_1^*(z_1) &= \mathbb{E}_{z_2}[\ln p(\mathbf{z})] + \text{const} \\ &= \mathbb{E}_{z_2} \left[-\frac{1}{2}(z_1 - \mu_1)^2 \Lambda_{11} - (z_1 - \mu_1) \Lambda_{12} (z_2 - \mu_2) \right] + \text{const} \\ &= -\frac{1}{2} z_1^2 \Lambda_{11} + z_1 \mu_1 \Lambda_{11} - z_1 \Lambda_{12} (\mathbb{E}[z_2] - \mu_2) + \text{const}. \quad (10.11) \end{aligned}$$

因为是求 z_1 的分布，所以按(10.9)，我们在 z_2 上求期望，得到(10.11)。然后，我们就可以祭出第二章修炼的法宝——配方法，从(10.11)得到高斯分布：

$$q^*(z_1) = \mathcal{N}(z_1|m_1, \Lambda_{11}^{-1}) \quad (10.12)$$

其中

$$m_1 = \mu_1 - \Lambda_{11}^{-1} \Lambda_{12} (\mathbb{E}[z_2] - \mu_2). \quad (10.13)$$

同样， z_2 的分布也可如法炮制：

$$q_2^*(z_2) = \mathcal{N}(z_2|m_2, \Lambda_{22}^{-1}) \quad (10.14)$$

其中

$$m_2 = \mu_2 - \Lambda_{22}^{-1} \Lambda_{21} (\mathbb{E}[z_1] - \mu_1). \quad (10.15)$$

它们是完全对称的。因为 m_1 里有 z_2 的期望，而 m_2 里又有 z_1 的期望，所以我们可以设一个初始值，然后迭代求解。但实际上这两个式子恰好有解析解： $\mathbb{E}[z_1] = \mu_1$ 和 $\mathbb{E}[z_2] = \mu_2$ ，我们可把它们代入(10.13)和(10.15)验证一下。

下面我们重点看一下参数推断问题，但其核心思想实际上和前面讲的例子区别不大。同样还是先看一下高斯分布：

我们想推断后验高斯分布的均值 μ 和精度 τ

假如我们观察到 N 个数据 $\mathcal{D} = \{x_1, \dots, x_N\}$ ，那么似然函数就是：

$$p(\mathcal{D}|\mu, \tau) = \left(\frac{\tau}{2\pi}\right)^{N/2} \exp\left\{-\frac{\tau}{2} \sum_{n=1}^N (x_n - \mu)^2\right\}. \quad (10.21)$$

另外引入先验分布，均值服从高斯分布、精度服从 Gamma 分布：

$$p(\mu|\tau) = \mathcal{N}(\mu|\mu_0, (\lambda_0\tau)^{-1}) \quad (10.22)$$

$$p(\tau) = \text{Gam}(\tau|a_0, b_0) \quad (10.23)$$

其实这个问题我们前面第二章就讲过，不用变分推断也能直接求出来，但这里用变分推断实际上增加了更多的灵活性，因为如果先验和似然的形式不是高斯-Gamma 的形式，而是更加复杂，那么我们也可以利用变分推断来算参数，这是非常方便的。我们这里只是用我们熟悉的高斯分布来举例子，把这个弄明白，以后再推广到其他例子上就容易多了。

利用 mean field 形式(10.9)，我们可计算出 μ 的分布：

$$\begin{aligned} \ln q_\mu^*(\mu) &= \mathbb{E}_\tau [\ln p(\mathcal{D}|\mu, \tau) + \ln p(\mu|\tau)] + \text{const} \\ &= -\frac{\mathbb{E}[\tau]}{2} \left\{ \lambda_0(\mu - \mu_0)^2 + \sum_{n=1}^N (x_n - \mu)^2 \right\} + \text{const}. \end{aligned} \quad (10.25)$$

可以看到， μ 服从高斯分布形式 $\mathcal{N}(\mu|\mu_N, \lambda_N^{-1})$ ，且通过配方，可得到该分布参数为：

$$\mu_N = \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \quad (10.26)$$

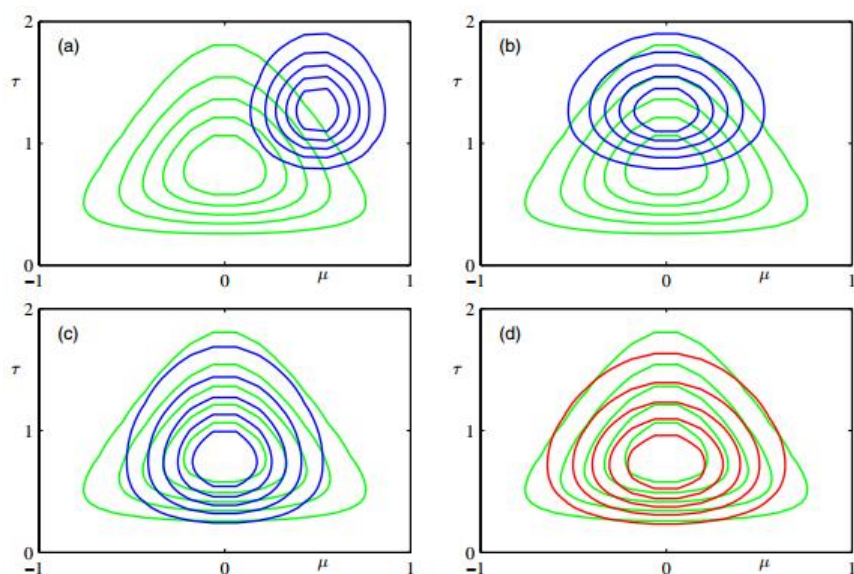
$$\lambda_N = (\lambda_0 + N)\mathbb{E}[\tau]. \quad (10.27)$$

注意到，样本越多也就是 N 越大时，均值会趋向于样本均值 $\mu_N = \bar{x}$ ，同时精度趋向于无穷大。同样

可用(10.9)计算 τ 的分布，得到：

$$\begin{aligned} \ln q_\tau^*(\tau) &= \mathbb{E}_\mu [\ln p(\mathcal{D}|\mu, \tau) + \ln p(\mu|\tau)] + \ln p(\tau) + \text{const} \\ &= (a_0 - 1) \ln \tau - b_0\tau + \frac{N}{2} \ln \tau \\ &\quad - \frac{\tau}{2} \mathbb{E}_\mu \left[\sum_{n=1}^N (x_n - \mu)^2 + \lambda_0(\mu - \mu_0)^2 \right] + \text{const} \end{aligned} \quad (10.28)$$

它服从 Gamma 分布形式 $\text{Gam}(\tau|a_N, b_N)$ ，可以看到，(10.27)和(10.30)里，仍然有和另一分布相关的期望需要计算，所以我们可以设定初始值，然后迭代计算。迭代过程和收敛后的结果图书上 10.4 所示：



再看一个例子，是用变分推断计算线性回归的参数。线性回归的参数 $\mathbf{w}$ ，有似然和先验如下：

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N \mathcal{N}(t_n|\mathbf{w}^T \phi_n, \beta^{-1}) \quad (10.87)$$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \quad (10.88)$$

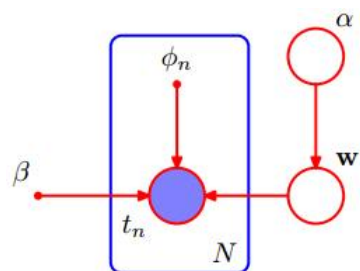
2.3.6 讲过， α 的共轭先验是 Gamma 分布：

$$p(\alpha) = \text{Gam}(\alpha|a_0, b_0) \quad (10.89)$$

这样联合分布就是：

$$p(\mathbf{t}, \mathbf{w}, \alpha) = p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)p(\alpha). \quad (10.90)$$

其概率图模型为图 10.8：



利用变分推断来计算 $\mathbf{w}$ 和 α ，同样是假设它们有可分解形式：

$$q(\mathbf{w}, \alpha) = q(\mathbf{w})q(\alpha). \quad (10.91)$$

再用(10.9) (这个绝对是看家宝) 来搞, 得到:

$$\begin{aligned}\ln q^*(\alpha) &= \ln p(\alpha) + \mathbb{E}_{\mathbf{w}} [\ln p(\mathbf{w}|\alpha)] + \text{const} \\ &= (a_0 - 1) \ln \alpha - b_0 \alpha + \frac{M}{2} \ln \alpha - \frac{\alpha}{2} \mathbb{E}[\mathbf{w}^T \mathbf{w}] + \text{const.}\end{aligned}\quad (10.92)$$

可看到它服从 Gamma 分布:

$$q^*(\alpha) = \text{Gam}(\alpha|a_N, b_N) \quad (10.93)$$

其中

$$a_N = a_0 + \frac{M}{2} \quad (10.94)$$

$$b_N = b_0 + \frac{1}{2} \mathbb{E}[\mathbf{w}^T \mathbf{w}]. \quad (10.95)$$

以及:

$$\ln q^*(\mathbf{w}) = \ln p(\mathbf{t}|\mathbf{w}) + \mathbb{E}_{\alpha} [\ln p(\mathbf{w}|\alpha)] + \text{const} \quad (10.96)$$

$$= -\frac{\beta}{2} \sum_{n=1}^N \{\mathbf{w}^T \phi_n - t_n\}^2 - \frac{1}{2} \mathbb{E}[\alpha] \mathbf{w}^T \mathbf{w} + \text{const} \quad (10.97)$$

$$= -\frac{1}{2} \mathbf{w}^T (\mathbb{E}[\alpha] \mathbf{I} + \beta \Phi^T \Phi) \mathbf{w} + \beta \mathbf{w}^T \Phi^T \mathbf{t} + \text{const.} \quad (10.98)$$

可看到它服从高斯分布:

$$q^*(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (10.99)$$

其中

$$\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t} \quad (10.100)$$

$$\mathbf{S}_N = (\mathbb{E}[\alpha] \mathbf{I} + \beta \Phi^T \Phi)^{-1}. \quad (10.101)$$

(10.95)和(10.97)里还有奇怪的东西 $\mathbb{E}[\alpha]$ 和 $\mathbb{E}[\mathbf{w}\mathbf{w}^T]$, 从附录 B 可知, 它们分别是:

$$\mathbb{E}[\alpha] = a_N / b_N \quad (10.102)$$

$$\mathbb{E}[\mathbf{w}\mathbf{w}^T] = \mathbf{m}_N \mathbf{m}_N^T + \mathbf{S}_N. \quad (10.103)$$

所以我们仍然可以迭代计算: 给初始值, 每一步都算出 a_N 、 b_N 和 $\mathbf{m}_N$ 、 $\mathbf{S}_N$, 代入求解。

掌握了上面的三个例子, 我想推广到其他情况也都没有太大难度了。其实书中还有一个例子也非常重要, 就是 10.2 所讲到的用变分推断计算高斯混合模型的参数。不过我想尼采兄讲第九章时已经打下了很好的基础, 再加上刚才讲的这一章的例子, 看懂这部分应该不难。

后面还有一些有趣的内容, 比如 Expectation Propagation, 是说对 $\text{KL}(p\|q)$ 做极小化, 而不是

$\text{KL}(q\|p)$ 。因为积分里前面那项变成了 $p(Z)$ 而不是 $q(Z)$, 而 $p(Z)$ 又是复杂分布, 所以这里处理方式有所不同。感兴趣的朋友可以看看 10.7 节是如何做的。

我讲的内容就到这里。我个人的一点心得体会就是: 高斯分布以及其他常用分布的形式、还有第二章讲到的配方法一定要掌握好, 这是识别分布和直接计算分布参数的最大利器。然后就是这一章的(10.9), 也就是用可分解分布去做近似得到的 mean field, 这也是比较常用的。其实群里有不少对变分推断很了解的高手, 比如@huajh7, 大家对这一块有什么问题也可以找他们交流讨论。

=====讨论=====

数据挖掘(983272906) 21:44:16

$p(\mathbf{t}, \mathbf{w}, \alpha) = p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)p(\alpha).$ (10.90) 这种分解有没有什么限制条件

Wilbur_中博(1954123) 21:45:20

这不是分解，是从先验和似然算联合分布。可分解的简单分布形式是(10.91)。

Y(414474527) 21:47:49

变分推断怎么应用到实际问题中呢

tzk<tangzk2011@163.com> 21:48:29

LDA 的原始论文用的也是变分呢。。

<(523723864) 21:48:43

10.9 式一定是 tractable 的吗？

zeno(117127143) 21:52:58

平均场假设可以有效减少参数。

Wilbur_中博(1954123) 21:53:54

@Y 实际问题吗？我觉得就是作为一种工具，求解模型参数的时候会比较简单吧。之前在稀疏编码里看到过一些，我觉得这篇文章不错：

http://ipg.epfl.ch/~seeger/lapmalmainweb/papers/seeger_wipf_spm10.pdf。另外 RBM 似乎也有用这个的。

zeno(117127143) 21:54:19

变分把推断变为求极值问题，怎么求是另外一门课

Wilbur_中博(1954123) 21:54:43

@< 我觉得不一定。。还得看 $p(X,Z)$ 是什么样的。

@zeno 嗯👍

<(523723864) 21:55:01

按照 10.9 主要是推式子咯，事先不知道 q_j 的分布

Wilbur_中博(1954123) 21:55:57

嗯，应该是。。但是一般来说都可以想办法搞出来吧，(10.9)的积分。

karnon(447457116) 21:59:27

为什么一开始又要用复杂分布呢，建模时用那些复杂的模型，最后到求解时都退化成 naive 模型，所以事实上，和 naive 模型一样

Wilbur_中博(1954123) 22:02:31

可能一开始就用简单分布的话，推出后验分布有连锁效应，就会越来越差吧。现在搞出后验分布再用简单分布去近似，我觉得道理上还是能说得通。

zeno(117127143) 22:03:27

那为啥有泰勒展开，展开把高次舍弃，不都不是原来函数了吗

<(523723864) 22:04:21

关键是每次迭代的时候 lower bound 会不会上去

karnon(447457116) 22:04:22

如果你要用 Taylor 展开来近似，那就得证明近似后你的解的性质不变，所以不是任何问题都能随便近似

Wilbur_中博(1954123) 22:04:43

@< 是，我觉得这个蛮关键的

karnon(447457116) 22:06:49

就是你的解为什么好，它好在哪，近似之后，这些好处是不是还保留着，这在变分法中，完全没有讨论

zeno(117127143) 22:10:58

要是 KL 跟概率差异定量关系就没问题了，平均场本来就是假设，变分推断是合理的，KL 嘛，不好说，反正不像熟悉的欧式度量，pgm 不只变分一种推断方法，所以也不能建成简单模型。说实在如果能解决一类小问题效果不错就已经很好了，mrf, hmm, crf, 都能算到 pgm 中。pgm 解决不少问题。

阿邦(1549614810) 23:41:20

推断方法不坑，主要还是模型的问题

karnon(447457116) 0:02:10

我总感觉，一定有基于非概率模型的方法

弹指一瞬间(337595903) 6:31:34

昨晚大家讨论的好热闹啊。@karnon：我觉得近似推理对原模型的好处还是保留着的。虽然求解的时候是在简单模型上做，但是简单模型的求解目标是去近似原模型的最优而不是简单模型的最优。这个和一上来就做简单模型假设是不大一样的。近似推理可以理解为在最优解附近找一个次优解，但总体目标还是原模型最优解的方向。而简单模型求解可能目标就不一样了。相比之下，还是用近似推理来解原问题比较好。

(个人理解不一定对，欢迎跟帖👉)

zeno(117127143) 6:52:44

我喜欢概率模型，概率既能对不确定性建模更能对未知建模。做单选题 25% 表达的是学生对答案的未知，同样的题对老师就是已知的。同样问题用非概率解你需要知道的更多。同样四道单选题三道不会，其他三道分别选 a, b, c。第四道用概率方法根据一定先验会尽量选 d。不用概率方法根本做不了这种问题。

同样如果知道了答案，肯定不会用概率方法，概率比通常非概率方法麻烦。

karnon(447457116) 7:33:16

这只是理想的情况，概率模型的缺点，在于它需要精确地刻画细节。

第十一章 Sampling Methods

主讲人 网络上的尼采

(新浪微博: @Nietzsche_复杂网络机器学习)

网络上的尼采(813394698) 9:05:00

今天的主要内容：Markov Chain Monte Carlo，Metropolis-Hastings，Gibbs Sampling，Slice Sampling，Hybrid Monte Carlo。

上一章讲到的平均场是统计物理学中常用的一种思想，将无法处理的复杂多体问题分解成可以处理的单体问题来近似，变分推断便是在平均场的假设约束下求泛函 $L(Q)$ 极值的最优化问题，好处在于求解过程中可以推出精致的解析解。变分是从最优化的角度通过坐标上升法收敛到局部最优，这一章我们将通过计算从动力学角度见证 Markov Chain Monte Carlo 收敛到平稳分布。

先说 sampling 的原因，因为统计学中经常会遇到对复杂的分布做加和与积分，这往往是 intractable 的。MCMC 方法出现后贝叶斯方法才得以发展，因为在那之前对不可观测变量（包括隐变量和参数）后验分布积分非常困难，对于这个问题上一章变分用的解决办法是通过最优化方法寻找一个和不可观测变量后验分布 $p(Z|X)$ 近似的分布，这一章我们看下 sampling 的解决方法，举个简单的例子：比如我们遇到这种形式

$$\mathbb{E}[f] = \int f(z)p(z) dz$$
， z 是个连续随机变量， $p(z)$ 是它的分布，我们求 $f(z)$ 的期望。如果我们从 $p(z)$

中 sampling 一个数据集 $z^{(l)}$ ，然后再求个平均
$$\hat{f} = \frac{1}{L} \sum_{l=1}^L f(z^{(l)})$$
来近似 $f(z)$ 的期望，so, 问题就解决了，关键是如何从 $p(z)$ 中做无偏的 sampling。

为了说明 sampling 的作用，我们先举个 EM 的例子，最大似然计算中求分布的积分问题，我们在第九章提到了，完整数据的 log 似然函数是对隐变量 Z 的积分：

$$Q(\theta, \theta^{\text{old}}) = \int p(Z|X, \theta^{\text{old}}) \ln p(Z, X|\theta) dZ$$

如果 Z 是比较复杂的分布，我们就需要对 Z 进行采样，从而得到：

$$Q(\theta, \theta^{\text{old}}) \simeq \frac{1}{L} \sum_{l=1}^L \ln p(Z^{(l)}, X|\theta).$$

具体就是从 Z 的后验分布 posterior distribution $p(Z|X, \theta^{\text{old}})$ 中进行采样。

如果我们从贝叶斯的观点，把 EM 参数 θ 也当成一个分布的话，有下面一个 IP 算法：

IP Algorithm

I-step. We wish to sample from $p(\mathbf{Z}|\mathbf{X})$ but we cannot do this directly. We therefore note the relation

$$p(\mathbf{Z}|\mathbf{X}) = \int p(\mathbf{Z}|\boldsymbol{\theta}, \mathbf{X})p(\boldsymbol{\theta}|\mathbf{X}) d\boldsymbol{\theta} \quad (11.30)$$

and hence for $l = 1, \dots, L$ we first draw a sample $\boldsymbol{\theta}^{(l)}$ from the current estimate for $p(\boldsymbol{\theta}|\mathbf{X})$, and then use this to draw a sample $\mathbf{Z}^{(l)}$ from $p(\mathbf{Z}|\boldsymbol{\theta}^{(l)}, \mathbf{X})$.

P-step. Given the relation

$$p(\boldsymbol{\theta}|\mathbf{X}) = \int p(\boldsymbol{\theta}|\mathbf{Z}, \mathbf{X})p(\mathbf{Z}|\mathbf{X}) d\mathbf{Z} \quad (11.31)$$

we use the samples $\{\mathbf{Z}^{(l)}\}$ obtained from the I-step to compute a revised estimate of the posterior distribution over $\boldsymbol{\theta}$ given by

$$p(\boldsymbol{\theta}|\mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^L p(\boldsymbol{\theta}|\mathbf{Z}^{(l)}, \mathbf{X}). \quad (11.32)$$

By assumption, it will be feasible to sample from this approximation in the I-step.

I 步，我们无法直接对 $P(\mathbf{Z}|\mathbf{X})$ 取样，我们可以先对 $P(\boldsymbol{\theta}|\mathbf{X})$ 取样 $\boldsymbol{\theta}^{(l)}$ ，然后再对 $\mathbf{Z}$ 的后验分布进行取样：

draw a sample $\mathbf{Z}^{(l)}$ from $p(\mathbf{Z}|\boldsymbol{\theta}^{(l)}, \mathbf{X})$.

P 步，利用上一步对 $P(\mathbf{Z}|\mathbf{X})$ 的取样，来确定新的参数分布：

$$p(\boldsymbol{\theta}|\mathbf{X}) \simeq \frac{1}{L} \sum_{l=1}^L p(\boldsymbol{\theta}|\mathbf{Z}^{(l)}, \mathbf{X}).$$

然后按这个 I 步和 P 步的方式迭代。

接下来我们讲 sampling methods，为了节省时间，两个基本的 rejection sampling 和 Importance sampling 不讲，这两种方法在高维下都会失效，我们直奔主题 MCMC (Markov Chain Monte Carlo)。蒙特卡洛，是个地中海海滨城市，气候宜人，欧洲富人们的聚集地，更重要的是它是世界三大赌城之一，用这个命名就知道这种方法是基于随机的，过我们会讲到。马尔科夫大神就不用多说了，他当时用自己的名字命名马尔科夫链是预见到这个模型巨大作用。他还有一个师弟叫李雅普洛夫，控制论里面的李雅普洛夫函数说的就是这位，他们的老师叫契比雪夫，都是圣彼得堡学派的。俄国数学家对人类的贡献是无价的 orz

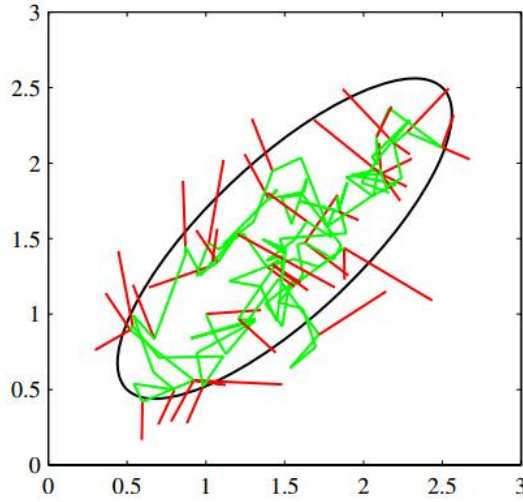
最早的 MCMC 方法是美国科学家在研制原子弹时算积分发明的。我们先介绍一个最基本的 Metropolis 方法，这种方法的接受率是：

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min \left(1, \frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})} \right)$$

但有个要求，就是 proposal distribution 满足 $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$ 。过程很简单，我们先找

个比较容易采样的分布即 proposal 分布，然后从这个分布中取一个样本 $\mathbf{Z}^*$ ，如果 $\frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})}$ 大于 1 直接接受，如果小于 1 就接着算出接受率，并且从 $(0, 1)$ 之间取一个随机数和这个接受率做比较来决定是否接受这个样本。过会会在 Metropolis-Hastings algorithm 方法中具体说。下图是一个简单的例子，对高斯分布做采样，绿线是表示接受的步骤，红线表示拒绝的：

A simple illustration using Metropolis algorithm to sample from a Gaussian distribution whose one standard-deviation contour is shown by the ellipse. The proposal distribution is an isotropic Gaussian distribution whose standard deviation is 0.2. Steps that are accepted are shown as green lines, and rejected steps are shown in red. A total of 150 candidate samples are generated, of which 43 are rejected.



讲 Metropolis-Hastings 方法前，我们先来回顾下马尔科夫链的性质，这个很重要。markov chains 最基本的性质就是无后效性，就是这条链的下一个节点的状态由当前节点状态完全决定：

$$p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}).$$

特定的齐次马尔科夫链可以收敛到平稳分布，也就是经过相当长的一段时间转移，收敛到的分布和初始值无关，转移核起着决定的作用。关于马尔科夫链的收敛，**我们将介绍一个充分条件：detailed balance 细致平稳条件。**

先介绍两个公式，马尔科夫链节点状态的 marginal distribution 计算公式，由于无后效性我们可以得到

$$p(\mathbf{z}^{(m+1)}) = \sum_{\mathbf{z}^{(m)}} p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}) p(\mathbf{z}^{(m)}). \quad (11.38)$$

上面公式的加和结合马尔科夫链的状态转移矩阵是比较容易理解。

平稳分布的定义就是下面的形式：

$$p^*(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}', \mathbf{z}) p^*(\mathbf{z}'). \quad (11.39)$$

其中 $T(\mathbf{z}', \mathbf{z})$ 是 $\mathbf{z}'$ 到 $\mathbf{z}$ 的转移概率，上面的公式不难理解，就是转移后分布不再发生变化。

下面我们给出细致平稳条件的公式：

$$p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) \quad (11.40)$$

满足细致平稳条件就能收敛到平稳分布，下面是推导：

$$\sum_{\mathbf{z}'} p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) = \sum_{\mathbf{z}'} p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}) \sum_{\mathbf{z}'} p(\mathbf{z}' | \mathbf{z}) = p^*(\mathbf{z}). \quad (11.41)$$

我们把上面细致平稳条件的公式 11.40 代入公式 11.41 最左边，从左朝右推导就是平稳分布的公式 11.39。

细致平稳条件的好处，就是我们能控制马尔科夫链收敛到我们指定的分布 $p^*(\mathbf{z})$ 。以后的 Metropolis-Hastings 方法及改进都是基于这个基础的。

前面我们提到，Metropolis 方法需要先选一个比较容易取样的 proposal distribution，从这个分布里取样，然后通过接受率决定是否采用这个样本。一个简单的例子就是对于 proposal distribution 我们可以

采用 Gaussian centred on the current state，其实很好理解，就是上一步节点的值可以做下一步节点需要采样的 proposal distribution 即高斯分布的均值，这样下一步节点的状态由上一步完全决定，这就是一个马尔科夫链。马尔科夫链有了，我们怎么保证能收敛到目标分布呢？就是前面说的细致平稳条件，我们可以通过设置接受率的形式来满足这个条件。Metropolis-Hastings 接受率的形式：

$$A_k(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min \left(1, \frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(\tau)} | \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^* | \mathbf{z}^{(\tau)})} \right). \quad (11.44)$$

$\tilde{p}(\mathbf{z})$ 来自于 $p(\mathbf{z}) = \tilde{p}(\mathbf{z}) / Z_p$ ，分布 q 便是 proposal distribution。

范涛@推荐系统(289765648) 10:49:15

Zp 是什么？

网络上的尼采(813394698) 10:52:04

Zp 是分布中和 z 无关的部分。

为了使各位有个形象的理解，我描述一下过程，我们把 $\mathbf{z}^{(\tau)}$ 当做高斯分布的均值，方差是固定的。然后从

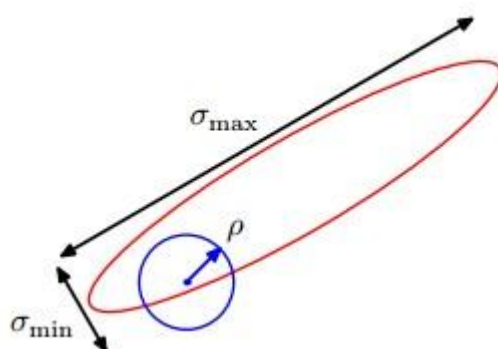
这个分布取一个样本就是 $\mathbf{z}^*$ ，如果 $\frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(\tau)} | \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^* | \mathbf{z}^{(\tau)})}$ 大于 1 肯定接受，如果小于 1，我们便从 $(0, 1)$ 之间取一个随机数，和这个接受率做比较，如果接受率大于这个随机数便接受，反之便拒绝。接受率这么设就能满足细致平稳条件的原因，看这个 (11.45) 公式：

$$\begin{aligned} p(\mathbf{z}) q_k(\mathbf{z}' | \mathbf{z}) A_k(\mathbf{z}', \mathbf{z}) &= \min (p(\mathbf{z}) q_k(\mathbf{z}' | \mathbf{z}), p(\mathbf{z}') q_k(\mathbf{z} | \mathbf{z}')) \\ &= \min (p(\mathbf{z}') q_k(\mathbf{z} | \mathbf{z}'), p(\mathbf{z}) q_k(\mathbf{z}' | \mathbf{z})) \\ &= p(\mathbf{z}') q_k(\mathbf{z} | \mathbf{z}') A_k(\mathbf{z}, \mathbf{z}') \end{aligned}$$

$$\begin{aligned} p(\mathbf{z}) \boxed{q_k(\mathbf{z}' | \mathbf{z}) A_k(\mathbf{z}', \mathbf{z})} &= \min (p(\mathbf{z}) q_k(\mathbf{z}' | \mathbf{z}), p(\mathbf{z}') q_k(\mathbf{z} | \mathbf{z}')) \\ &= \min (p(\mathbf{z}') q_k(\mathbf{z} | \mathbf{z}'), p(\mathbf{z}) q_k(\mathbf{z}' | \mathbf{z})) \\ &= p(\mathbf{z}') \boxed{q_k(\mathbf{z} | \mathbf{z}') A_k(\mathbf{z}, \mathbf{z}')} \end{aligned}$$

我们把接受率公式 11.44 代入上面的公式的左边，会推出左右两边就是细致平稳条件的形式，红框部分便是细致平稳条件公式 11.40 的转移核，书上的公式明显错了，上面的这个是勘误过的。

刚才说了 proposal distribution 一般采用 Gaussian centred on the current state，高斯分布的方差是固定的，其实方差就是步长，如何选择步长这是一个 state of the art 问题，步子太小扩散太慢，步子太大，拒绝率会很高，原地踏步。书中的一个例子，当用 Gaussian centred on the current state 作 proposal distribution 时，步长设为目标高斯分布的最小标准差最合适：



下面讲 Gibbs Sampling, Gibbs Sampling 其实是每次只对一个维度的变量进行采样, 固定住其他维度的变量, 然后迭代, 可以看做是 Metropolis-Hastings 的特例, 它的接受率一直是 1.

Gibbs Sampling

1. Initialize $\{z_i : i = 1, \dots, M\}$
2. For $\tau = 1, \dots, T$:
 - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \dots, z_M^{(\tau)})$.
 - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \dots, z_M^{(\tau)})$.
 - $\vdots$
 - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \dots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \dots, z_M^{(\tau)})$.
 - $\vdots$
 - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \dots, z_{M-1}^{(\tau+1)})$.

步骤是比较容易理解的, 跟上一章的变分法的有相似之处。假设有三个变量的分布 $p(z_1, z_2, z_3)$,

先固定住 z_2, z_3 对 z_1 进行采样, $p(z_1 | z_2^{(\tau)}, z_3^{(\tau)})$;

然后固定住 z_1, z_3 对 z_2 进行采样, $p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)})$;

然后是 z_3 , $p(z_3 | z_1^{(\tau+1)}, z_2^{(\tau+1)})$

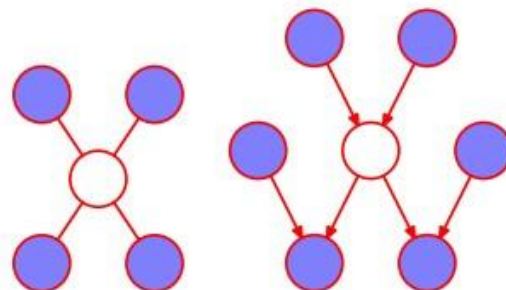
如此迭代。

根据 Metropolis-Hastings, 它的接受率恒为 1。看下面的推导:

$$A(\mathbf{z}^*, \mathbf{z}) = \frac{p(\mathbf{z}^*)q_k(\mathbf{z}|\mathbf{z}^*)}{p(\mathbf{z})q_k(\mathbf{z}^*|\mathbf{z})} = \frac{p(z_k^*|\mathbf{z}_{\setminus k}^*)p(\mathbf{z}_{\setminus k}^*)p(z_k|\mathbf{z}_{\setminus k}^*)}{p(z_k|\mathbf{z}_{\setminus k})p(\mathbf{z}_{\setminus k})p(z_k^*|\mathbf{z}_{\setminus k})} = 1$$

因为其他维度是固定不变的, 所以 $\mathbf{z}_{\setminus k}^* = \mathbf{z}_{\setminus k}$, 代入上式就都约去了, 等于 1.

最后对于图模型采用 gibbs sampling, 条件概率 $p(z_k | \mathbf{z}_{\setminus k})$ 可以根据马尔科夫毯获得, 下面一个是无向图, 一个是有向图, 蓝色的节点是和要采样的变量有关的其他变量:



关于更多的 gibbs sampling 的内容可以看 MLAPP, 里面有 blocked gibbs 和 collapsed gibbs.

刚才提到 Metropolis-Hastings 对步长敏感, 针对这个问题, 下面介绍两个增加辅助变量的方法, 这些方法也是满足细致平稳条件的。先介绍 slice sampling, 这种方法增加了一个变量 U , 可以根据分布的特

征自动调整步长：

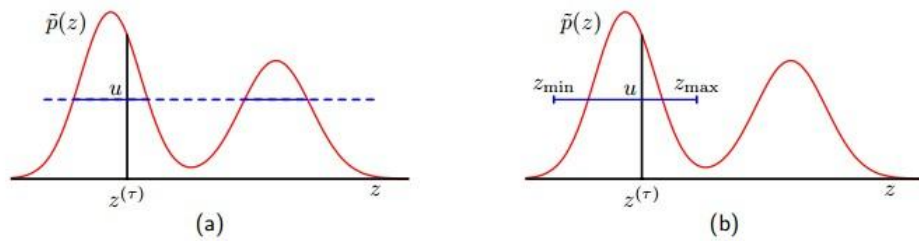


Figure 11.13 Illustration of slice sampling. (a) For a given value $z^{(\tau)}$, a value of u is chosen uniformly in the region $0 \leq u \leq \tilde{p}(z^{(\tau)})$, which then defines a 'slice' through the distribution, shown by the solid horizontal lines. (b) Because it is infeasible to sample directly from a slice, a new sample of z is drawn from a region $z_{\min} \leq z \leq z_{\max}$, which contains the previous value $z^{(\tau)}$.

步骤很简单：在 $z^{(\tau)}$ 与 $\tilde{p}(z)$ 之间的这段距离随机取个值 u ，然后通过 u 画个横线，然后在包含 $z^{(\tau)}$ 并且 $\{z : \tilde{p}(z) > u\}$ 这段横线对 z 进行随机采样，然后按这种方式迭代。图(b)为了实际中便于操作，有时

还需要多出那么一段，因为我们事先不知道目标分布的具体形式，所以包含 $z^{(\tau)}$ 并且 $\{z : \tilde{p}(z) > u\}$ 这段横线没法确定，只能朝外延伸加单位长度进行试，最后会多出来一段，这一点书上并没有介绍详细。

下面介绍 The Hybrid Monte Carlo Algorithm (Hamiltonian MCMC)：哈密顿，神童，经典力学三巨头之一，这个算法引入了哈密顿动力系统的概念，计算接受率时考虑的是系统的总能量。

Hybrid Monte Carlo 定义了势能和动能两种能量，它们的和便是系统总能量哈密顿量。先看势能，分布可以写成这种形式：

$$p(\mathbf{z}) = \frac{1}{Z_p} \exp(-E(\mathbf{z}))$$

$E(\mathbf{z})$ 便是系统的势能。

另外增加一个变量，状态变量变化的速率： $r_i = \frac{dz_i}{d\tau}$

系统的动能便是：

$$K(\mathbf{r}) = \frac{1}{2} \|\mathbf{r}\|^2 = \frac{1}{2} \sum_i r_i^2$$

总的能量便是： $H(\mathbf{z}, \mathbf{r}) = E(\mathbf{z}) + K(\mathbf{r})$

比如高斯分布的哈密顿量就可表示为：

$$H(\mathbf{z}, \mathbf{r}) = \frac{1}{2} \sum_i \frac{1}{\sigma_i^2} z_i^2 + \frac{1}{2} \sum_i r_i^2.$$

下面这个公式便是 Hybrid Monte Carlo 的接受率：

$$\min(1, \exp\{H(\mathbf{z}, \mathbf{r}) - H(\mathbf{z}^*, \mathbf{r}^*)\})$$

可以证明 这种接受率是满足 detailed balance 条件的。

ORC(267270520) 12:14:03

推荐一本相关的书 *Introducing Monte Carlo Methods with R (use R)* , PS : R 做 MCMC 很方便。

赞尼采讲的很精彩, 学习了, 嘿嘿

网络上的尼采(813394698) 12:36:37

Markov Chain Monte Carlo In Practice(Gilks)这本书也挺不错。

红烧鱼(403774317) 12:38:06

这本读研的时候生读过, 非常实用, 随书附带 code

网络上的尼采(813394698)

最后需要补充的是: 判断 MCMC 的 burn-in 何时收敛是个问题, koller 介绍了两种方法, 即同一条链上设置不同的时间窗做比较, 另一种同时跑多条链然后作比较。当然也有一条链跑到黑的。

第十二章 Continuous Latent Variables

主讲人 戴玮

(新浪微博: @戴玮_CASIA)

Wilbur_中博(1954123) 20:00:49

我今天讲 PRML 的第十二章，连续隐变量。既然有连续隐变量，一定也有离散隐变量，那么离散隐变量是什么？我们可能还记得之前尼采兄讲过的 9.2 节的高斯混合模型。它有一个 K 维二值隐变量 z ，不仅只能取 0-1 两个值，而且 K 维中只能有 1 维为 1、其他维必须为 0，表示我们观察到的 x 属于 K 类中的哪一类。显然，这里的隐变量 z 就是个离散隐变量。不过我们容易想到，隐变量未必像 kmeans 或 GMM 这种聚类算法那样，非此即彼、非白即黑，我们当然也可能在各个聚类或组成成分之间连续变化。而且很多情况下，连续变化都是更合理、更容易推广的。所以，我们这一章引入了连续隐变量。

书中举了一个例子：从某张特定的手写数字图像，通过平移和旋转变换生成多张图像。虽然我们观察到的是整个图像像素的一个高维数据空间中的样本，但实际上只是由平移和旋转这三个隐变量产生的，这里的平移和旋转就是连续隐变量。还举了个石油流量的例子，是从两个隐变量经过测量得到 12 个观察变量，那里的两个隐变量也是连续的。一般来说，样本不会精确处在由隐变量表示的低维流形上，而是可能稍有偏差，这种偏差可视作噪声。噪声的来源各种各样，不是我们能把握的，一般只能统一把它们看成单一的噪声项来处理。

最简单的情况下，我们可以把隐变量和观察变量都假设为高斯分布，并且利用 2.3.1 讲过的条件分布与边缘分布之间的线性高斯关系，来建立观察变量与隐变量之间的线性模型。这样，我们就可以建立主成分分析 (PCA) 以及与之相关的因子分析 (FA) 的概率模型。不过在此之前，我们还是看看传统视角是如何处理主成分分析的：

PCA 也叫 Karhunen-Loève transform (KL 变换) 或 Hotelling transform (霍特林变换)。它有两种可产生相同算法的等价视角：最大方差和最小重构误差。两种视角都希望找到一组正交投影，把原数据投影到低维的线性子空间上。但最大方差视角是说，我们希望数据投影之后在投影方向上有最大方差；而最小重构误差视角是说，我们希望投影后的数据和原数据之间的均方差最小。前者由 Hotelling 于 1933 年提出，后者由 Pearson 于 1901 年提出。

先来看最大方差视角。首先定义样本均值和样本协方差：

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \quad (12.1)$$

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T. \quad (12.3)$$

然后，我们可以得到某个投影方向 $\mathbf{u}_1$ 上的方差：

$$\frac{1}{N} \sum_{n=1}^N \{\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 \quad (12.2)$$

不失一般性，我们令 $\mathbf{u}_1^T \mathbf{u}_1 = 1$ ，这样我们就可以将 $\mathbf{u}_1^T \mathbf{u}_1 = 1$ 作为约束，将方差最大作为目标函数，把这个问题看作有约束最优化问题，因此可用拉格朗日乘法求解：

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1). \quad (12.4)$$

令其导数为 0，可得到：

$$\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1 \quad (12.5)$$

这是我们熟悉的线性代数中的特征值分解问题， λ_1 和 $\mathbf{u}_1$ 分别是 $\mathbf{S}$ 的特征值和特征向量。而且可以看到，这里求出的 $\mathbf{u}_1$ 方向的最大方差就是：

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1 \quad (12.6)$$

在余下的方向中依次选择最大方差方向，就是 $\mathbf{S}$ 由大到小给出的各个特征值以及对应的特征向量，这也容易从 $\mathbf{S}$ 是实对称矩阵、因此得到的特征向量之间是正交的这一点看出来。

再来看最小重构误差视角，由投影方向之间的标准正交关系，我们可以得到样本在 D 个投影方向下的表示：

$$\mathbf{x}_n = \sum_{i=1}^D (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i. \quad (12.9)$$

但我们不想用 D 个投影方向，而是想用 $M < D$ 个方向来表示样本，并且希望这样表示尽可能接近原样本。那么原样本与 M 个方向重构得到的样本之间的误差，用均方差来衡量就是：

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2. \quad (12.11)$$

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=M+1}^D \{(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i\} \mathbf{u}_i \quad (12.14)$$

上面的公式 12.14 展开之后就是：

$$J = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)^2 = \sum_{i=M+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i. \quad (12.15)$$

我们想最小化这个重构误差项。因为投影方向之间正交，所以也可以逐一求解，也就是目标函数：

$$J = \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2$$

约束条件是：

$$\mathbf{u}_2^T \mathbf{u}_2 = 1$$

同样可以由拉格朗日乘子法得到：

$$\tilde{J} = \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2 + \lambda_2 (1 - \mathbf{u}_2^T \mathbf{u}_2). \quad (12.16)$$

这和最大方差视角一样，也是特征值问题。只不过这里是去掉较小特征值对应的方向，因为那些方向对应着较小的重构误差，而先前是保留较大特征值对应的方向。但得到的结果是完全一样的。

在 D 个特征向量中选择前 M 个作为投影方向，得到的重构误差就是：

$$J = \sum_{i=M+1}^D \lambda_i \quad (12.18)$$

下面简单谈谈 PCA 的复杂度问题。我们知道， $\mathbf{S}$ 是 $D \times D$ 维矩阵，对 $\mathbf{S}$ 做特征值分解需要 $O(D^3)$ 的复

杂度。如果仅需要前 M 个最大的特征值以及特征向量，那么有一些算法可达到 $O(M \cdot D^2)$ 复杂度，比如 power method (Golub and Van Loan, 1996)。然而，即使是 $O(M \cdot D^2)$ ，在 D 较大的时候也是难以接受的。比如我们可能会用 PCA 做人脸识别的一些处理，人脸图像一般是几万维的，那么 $O(M \cdot D^2)$ 就是至少好几亿的复杂度，这显然是无法接受的。一会我们要讲的概率角度下用 EM 算法求解 PPCA 可以进一步降低复杂度。但仅从矩阵分解角度，实际上有一个非常巧妙的方法，也是 PCA 实现中常用的一种技巧：我们知道，对 S 做特征值分解的公式是：

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{u}_i = \lambda_i \mathbf{u}_i. \quad (12.26)$$

我们可以把左右都左乘 X ，得到：

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{u}_i) = \lambda_i (\mathbf{X} \mathbf{u}_i). \quad (12.27)$$

令 $\mathbf{v}_i = \mathbf{X} \mathbf{u}_i$ 得到：

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i \quad (12.28)$$

这里可以看到，对 $\frac{1}{N} \mathbf{X} \mathbf{X}^T$ 做特征值分解，与对 $\frac{1}{N} \mathbf{X}^T \mathbf{X}$ 做特征值分解，它们有着相同的非零特征值，但特征向量是不一样的。

不过，对(12.28)左右同时左乘 $\mathbf{X}^T$ 之后，我们可以得到：

$$\left(\frac{1}{N} \mathbf{X}^T \mathbf{X} \right) (\mathbf{X}^T \mathbf{v}_i) = \lambda_i (\mathbf{X}^T \mathbf{v}_i) \quad (12.29)$$

则 $\mathbf{u}_i \propto \mathbf{X}^T \mathbf{v}_i$

因为我们要求投影方向是标准正交的，所以归一化之后就是

$$\mathbf{u}_i = \frac{1}{(N \lambda_i)^{1/2}} \mathbf{X}^T \mathbf{v}_i. \quad (12.30)$$

因为 $\mathbf{X} \mathbf{X}^T$ 是 $N \times N$ 矩阵，和样本数量相关而和特征维度无关，所以如果是特征空间维度很高，但样本数量相对来说不那么大，那么我们就可以对 $\mathbf{X} \mathbf{X}^T$ 做特征值分解，而不是 $\mathbf{X}^T \mathbf{X}$ 。这样算法求解起来会快很多。但是，如果样本数量也很大、特征维度也很高，这样做也不合适了。这时就需要借助概率模型来求解。

=====讨论=====

Wilbur_中博(1954123) 20:30:56

好了，有问题现在可以提问。

dxhtml(601765336) 20:32:36

好像还没看出和连续隐变量的关系，是不是数据在主成分方向的投影是隐变量，因为是连续的，所以。。。

Wilbur_中博(1954123) 20:33:35

嗯，还没讲到连续隐变量。。前面说了，这是传统视角的 PCA，而不是一会要讲的概率视角。概率视角才有隐变量观察变量那套东西。

我觉得这部分应该都蛮熟悉的，不会有什么问题吧。

Phinx(411584794) 20:34:46

为什么维度太高了，就不合适啊？

coli<fwforever@126.com> 20:35:19

现在应该是理论阶段吧

布曲(15673189) 20:35:22

矩阵运算太复杂了

Wilbur_中博(1954123) 20:36:11

因为 $O(D^3)$ 里的 D 就是维度，太高了矩阵分解很慢，就算是 $O(D^2)$ 也够慢的。前面说了，上万维的图像，一平方就好几亿了。

也不是理论阶段了，其实那个把 $D \times D$ 变为 $N \times N$ 的技巧就很实用的。没什么问题就继续了哈

布曲(15673189) 20:37:54

是的 大家可以去网上下 pca 人脸识别的 matlab 小代码

=====讨论结束=====

Wilbur_中博(1954123) 20:40:45

前面讲的传统视角，是把原数据空间往较低维子空间上做线性投影，找这个投影方向。下面，我们从概率角度、用隐变量来建模。PCA 也可以从这种角度导出。PPCA 相较于传统 PCA，有以下优点：

1. 既可以捕捉数据中的线性相关关系，又可以对高斯分布中的自由参数加以限制；
2. 在少数几个特征向量的特征值较大、其他都较小时，用 EM 算法求解 PCA 是很高效的，而且它不用计算协方差阵作为中间步骤；
3. 概率模型和 EM 的结合，使我们可以解决数据集中的 missing values；
4. PPCA 的混合模型也可用 EM 求解；
5. 贝叶斯方法可自动确定数据的主子空间维度；
6. 它的似然函数可以有概率解释，因此可与其他概率密度模型做比较；
7. PPCA 可作为类条件概率用到分类器中；
8. 可从中采样得到新样本。

PPCA 是前面 2.3.1 讲过的线性高斯模型的一个实例。线性高斯模型就是说，边缘分布是高斯的，条件分布也是高斯的，而且条件分布的均值与边缘分布之间是线性关系。这里我们引入隐变量 z 及其分布 $p(z)$ ，并且观察变量 x 是由 z 产生的，因此有条件分布 $p(x|z)$ 。

不失一般性，设 z 服从 0 均值单位方差的高斯分布，则有：

$$p(z) = \mathcal{N}(z|0, I). \quad (12.31)$$

以及

$$p(x|z) = \mathcal{N}(x|Wz + \mu, \sigma^2 I) \quad (12.32)$$

这里的参数有三个：从隐变量空间到观察变量空间的线性变换 W 、均值 μ 和方差 σ^2 。从产生式的观点来看，我们先从 $p(z)$ 采样得到隐变量 z 的值，然后用这个值经过线性变换，并加上均值项（可看做一个常数偏移）和高斯噪声项，就可以得到 x ：

$$x = Wz + \mu + \epsilon \quad (12.33)$$

这里 z 是 M 维高斯隐变量， ϵ 是 D 维零均值高斯分布噪声，且有方差 $\sigma^2 I$ ，也就是说，我们假定各维噪声之间是独立的、且方差相等。从这个概率框架可以看出，我们先是用 W 建立了隐变量与观察变量之间的关系，再用 μ 和 ϵ 描述了观察变量的概率分布是什么样的。

我觉得 PPCA 比起传统 PCA，有一点优势就是利用了概率分布。因为我们的数据即使在某个低维子空间上，也不可能分布在整个子空间，而是只处在其中一个小区域。概率模型就很好地利用了这一点。当然，除了生成数据之外，概率模型更大的优势还是通过观察变量，也就是手里的数据，去推断参数也就是 W 、

μ 、 σ^2 是什么。这就要利用一些统计推断方法，比如最大似然法。但想用最大似然，必须先知道似然函数是什么。由 2.3.1, 2.3.2, 2.3.3 这几节的锤炼，我们应该对高斯分布的边缘分布、条件分布等形式的推导已经很熟练了。所以我们从前面给出的 $p(z)$ 和 $p(x|z)$ ，容易得到边缘分布 $p(x)$ ：

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}) \quad (12.35)$$

其中

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \quad (12.36)$$

实际上把(12.31)和(12.32)的协方差，代入到 2.3.3 最后给出的边缘分布公式中，可以得到这里给出的边缘分布 $p(x)$ 。当然也可以像书上讲的那样，从(12.33)来推导。

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}] = \boldsymbol{\mu} \quad (12.37)$$

$$\begin{aligned} \text{cov}[\mathbf{x}] &= \mathbb{E}[(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})^T] \\ &= \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \end{aligned} \quad (12.38)$$

因此我们得到了似然函数的具体形式：它也是一个高斯分布，以及它的期望和协方差是什么。高斯分布的指数里有协方差的逆，这里协方差矩阵 $\mathbf{C}$ 是 $D \times D$ 维的：

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \quad (12.36)$$

所以直接求逆的话复杂度也很高。我们也可以利用一些技巧，把直接求逆的 $O(D^3)$ 复杂度降到 $O(M^3)$

$$\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^T \quad (12.40)$$

$$\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I} \quad (12.41)$$

有了似然函数，我们就可以用最大似然法来从观察变量求解未知参数了。首先，为了方便求解，我们对似然函数 $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$ 取对数，这是我们都很熟悉的做法了，得到：

$$\begin{aligned} \ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^N \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\mathbf{C}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}). \end{aligned} \quad (12.43)$$

前面说过，我们的未知参数有三个： $\mathbf{W}$, $\boldsymbol{\mu}$, σ^2 ，似然函数逐一对它们求导，令导数为 0，就可以得到各自的参数估计。

$\boldsymbol{\mu}$ 的形式最简单 $\boldsymbol{\mu} = \bar{\mathbf{X}}$ 代回(12.43)得到：

$$\ln p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = -\frac{N}{2} \{ D \ln(2\pi) + \ln |\mathbf{C}| + \text{Tr}(\mathbf{C}^{-1}\mathbf{S}) \} \quad (12.44)$$

这里利用了 $\text{sum}(\mathbf{x}_i^T \mathbf{S}^* \mathbf{x}_i) = \text{Tr}(\mathbf{X}^T \mathbf{S}^* \mathbf{X})$ ，其中 $\mathbf{X}$ 的第 i 列是 $\mathbf{x}_i$ ，这是因为 $\mathbf{X}^T \mathbf{S}^* \mathbf{X}$ 的第 (i,i) 个元素等于 $\mathbf{x}_i^T \mathbf{S}^* \mathbf{x}_i$ 。因此，(12.43)的最后一项可变为(12.44)的最后一项。还要利用 trace 的性质： $\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{BCA}) = \text{Tr}(\mathbf{CAB})$ 。

但对剩下的两个参数求导以及求解的过程，这里就比较含糊其辞了，只是说求解比较复杂，但仍然可以得

到解析解。直接给出书上写的解析解形式：

$$\mathbf{W}_{ML} = \mathbf{U}_M (\mathbf{L}_M - \sigma^2 \mathbf{I})^{1/2} \mathbf{R} \quad (12.45)$$

这里的 $\mathbf{U}_M$ 由 $\mathbf{S}$ (样本协方差阵) 的最大的 M 个特征值对应的特征向量组成时, 可以取到最大值, 如果不是最大的而是任意特征向量, 那么只是鞍点, 这是由 Tipping and Bishop (1999b) 给出的证明。 $\mathbf{L}_M$ 是相应特征值的对角阵, 而 $\mathbf{R}$ 是任意的 $M \times M$ 维旋转矩阵。因为有 $\mathbf{R}$ 的存在, 所以可以认为, 从概率角度看, 这些隐变量张成的子空间以及从该子空间如何生成样本比较重要, 但取这个子空间的哪些方向则不是那么重要。只要子空间一致, 我们可以任意旋转张成子空间的这些方向。我觉得这也是 PPCA 和传统 PCA 的区别之一。

σ^2 的最大似然解是：

$$\sigma_{ML}^2 = \frac{1}{D - M} \sum_{i=M+1}^D \lambda_i \quad (12.46)$$

直观来看, 它的物理意义是, 被我们丢弃的那些维度的方差的平均值。从(12.32)和(12.33)看到, σ^2 实际上代表了噪声的方差, 那么这个方差当然是越小越好, 说明 $\mathbf{W}$ 已经尽可能保留了分布信息。它也可以看成某种平均重构误差。

你们可以先讨论讨论。。看看有什么问题

dxhml(601765336) 21:36:52

这样的模型的表达能力有限, $\mathbf{X}$ 如果不是高斯分布呢?

Wilbur_中博(1954123) 21:40:43

@dxhml 是啊, 所以这是最简单的模型, 肯定还是要由浅入深。后面还会讲到高斯分布之外的假设。

继续, 既然给出了解析形式, 那么我们仍然可以用特征值分解的方法来求解, 因为 $\mathbf{W}$ 和 σ^2 的最大似然估计都是和特征值以及特征向量相关的。但因为有似然函数, 所以借助各种最优化工具, 比如共轭梯度法, 或者一会要讲到的 EM, 求解起来会更快。这一小节最后还讲到了 PPCA 因为建立了隐变量与观察变量之间的关系, 所以自由度上远比直接估计高斯分布要小, 这样估计起来所需要的数据也就小不少, 速度也会快很多。

下面讲 PPCA 的 EM 求解, EM 法有这样几个好处: 高维空间下比解析解速度快, 可扩展到其他没有解析解的模型比如因子分析 (FA) 上, 可处理 missing data 的情况。EM 法尼采兄前面已经讲得很清楚了, 就是在 E、M 两步间来回迭代。这里因为我们在 E 步求出 $\mathbf{z}_n$ 和 $\mathbf{z}_n \mathbf{z}_n^T$ 的后验期望, 然后在 M 步把这两个期望代回去, 就可以做对数似然的最大化了, 然后可以得到新的 $\mathbf{W}$ 和 σ^2 。

前面我们看到, $\boldsymbol{\mu}$ 的形式非常简单, 所以 $\boldsymbol{\mu}$ 就直接用样本均值代替了, 迭代求的只有 $\mathbf{W}$ 和 σ^2 。也就是：

$$\begin{aligned} \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \mathbf{W}, \sigma^2)] = & - \sum_{n=1}^N \left\{ \frac{D}{2} \ln(2\pi\sigma^2) + \frac{1}{2} \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T]) \right. \\ & + \frac{1}{2\sigma^2} \|\mathbf{x}_n - \boldsymbol{\mu}\|^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^T \mathbf{W}^T (\mathbf{x}_n - \boldsymbol{\mu}) \\ & \left. + \frac{1}{2\sigma^2} \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}^T \mathbf{W}) \right\}. \end{aligned} \quad (12.53)$$

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1} \mathbf{W}^T (\mathbf{x}_n - \bar{\mathbf{x}}) \quad (12.54)$$

$$\text{其中 E 步: } \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] = \sigma^2 \mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^T \quad (12.55)$$

M 步：

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^T \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \right]^{-1} \quad (12.56)$$

$$\sigma_{\text{new}}^2 = \frac{1}{ND} \sum_{n=1}^N \left\{ \|\mathbf{x}_n - \bar{\mathbf{x}}\|^2 - 2 \mathbb{E}[\mathbf{z}_n]^T \mathbf{W}_{\text{new}}^T (\mathbf{x}_n - \bar{\mathbf{x}}) + \text{Tr} \left(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}_{\text{new}}^T \mathbf{W}_{\text{new}} \right) \right\}. \quad (12.57)$$

EM 法并不需要显式构造协方差阵，它的主要开销在于 $\mathbf{W}$ 和 σ^2 的更新中对整个数据集做求和操作，所以复杂度是 $O(NDM)$ ，可以看到，因为高维情况下 D 比较大，而我们降维后的维度 M 较小，所以比特征值分解最好的 $O(ND^2)$ 要好不少。而且 EM 的 online 版本可以进一步提高效率。missing data 就不讲了。。自己看看吧。

=====讨论=====

布曲(15673189) 22:07:20

ppac 优于传统的 pca 主要是因为运算速度吗，因为引入隐变量 可以用 em 更快的求解？

Wilbur_中博(1954123) 22:10:20

嗯，我觉得运算速度上是有优势的，因为特征值分解比较耗时。不过传统角度应该也可以用一些最优化方法和 online 方法求解那个代价函数吧。没有具体总结过。感觉更大的优势还是在概率角度解释数据比较好。前面我说过一个：我觉得 PPCA 比起传统 PCA，有一点优势就是利用了概率分布。因为我们的数据即使在某个低维子空间上，也不可能分布在整个子空间，而是只处在其中一个小区间。概率模型就很好地利用了这一点。

布曲(15673189) 22:12:12

嗯 同意

Wilbur_中博(1954123) 22:12:14

我觉得这可能是比较有优势的。传统 PCA 投影角度感觉解释得有点粗糙了。是吧。

=====讨论结束=====

Wilbur_中博(1954123) 20:57:53

从 12.2.3 继续讲，上次我们看到了怎么从贝叶斯角度解释 PCA，也就是 PPCA。简单说，就是假设有个隐变量 $\mathbf{z}$ 服从标准高斯分布，它通过某个线性变换 $\mathbf{W}$ 生成 $\mathbf{x}$ ，这个变换过程还可能有某些未知的随机因素比如噪声等，可将其假设为 0 均值同方差随机变量。所以，我们要做的就是通过手里拿到的数据 $\mathbf{X}$ 去推断未知参数：线性变换 $\mathbf{W}$ 、偏移量同时也是 $\mathbf{x}$ 的均值 μ 、未知随机量的方差 $\sigma^2 \mathbf{I}$ 。这里 $\mathbf{W}$ 的各列张成了我们感兴趣的主成分空间。我们讲到，虽然通过最大似然解得到的 $\mathbf{W}$ 与原先传统 PCA 相比，求出来的主方向差不多，但用 EM 法求解复杂度降低了不少，而且从贝叶斯角度也可以对 PCA 的模型假设和生成过程有更深刻的认识。但有一个问题我们没有谈到，就是如何选择 $\mathbf{W}$ 中的哪些主方向是最有用的，或者说如何更有效地降维。

我们上次讲过，从(12.45)：

$$\mathbf{W}_{\text{ML}} = \mathbf{U}_M (\mathbf{L}_M - \sigma^2 \mathbf{I})^{1/2} \mathbf{R} \quad (12.45)$$

可以看到，它的最大似然解仍然与最大的那些特征值以及对应的特征向量相关，所以我们可以像传统 PCA 那样，把特征值从大至小排列出来，看看是不是在哪个值上突然下降，把特征值分成较大和较小的两部分，那么我们就可以选择较大的那部分特征值以及与其对应的特征向量。

但是，一方面，特征值未必存在这种突然下降的趋势，可能是比较平滑地逐渐下降；另一方面，我们既然用了贝叶斯方法，那就应该将贝叶斯进行到底。

我们在第 7 章也讲过，用某些先验可以将特征的线性表示稀疏化，进而起到特征选择的效果，那么这里，

我们也可以利用这一思想做降维。既然用贝叶斯，那就肯定要对 W 设一个先验分布。这里介绍了一种比较简单的方法：设 W 的先验分布为各维独立的高斯分布，且各维方差由 precision 超参数 α_i 控制，前面讲过，precision 就是方差的倒数。那么，先验分布可以写作：

$$p(\mathbf{W}|\boldsymbol{\alpha}) = \prod_{i=1}^M \left(\frac{\alpha_i}{2\pi} \right)^{D/2} \exp \left\{ -\frac{1}{2} \alpha_i \mathbf{w}_i^T \mathbf{w}_i \right\} \quad (12.60)$$

稍后可以看到， α_i 在迭代计算的过程中，有些会趋向于无穷大，那么相应地，那个维度的 w_i 就会趋向于 0，说明那个维度的特征没什么用。这样我们有用的特征就是 α 有限的那些维度，而且 α 越小的维度越有用。这是比较常见的稀疏化方法，更具体的内容可参见 PRML 的 7.2 节。求解方法就是一般贝叶斯求解的 EM 法，首先把 W 积掉，得到似然函数：

$$p(\mathbf{X}|\boldsymbol{\alpha}, \boldsymbol{\mu}, \sigma^2) = \int p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) p(\mathbf{W}|\boldsymbol{\alpha}) d\mathbf{W} \quad (12.61)$$

这个边缘分布的积分是不易求的，我们可以借助前面讲过的 4.4 节的 Laplace approximation，对 α_i 求边缘分布的极值，得到：

$$\alpha_i^{\text{new}} = \frac{D}{\mathbf{w}_i^T \mathbf{w}_i} \quad (12.62)$$

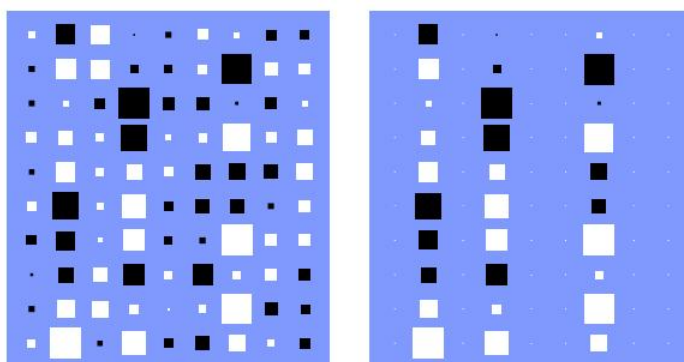
这是 EM 当中 M 步的估计 α_i 的计算公式。

M 步估计 σ^2 的公式和先前 PPCA 的 EM 法讲过的(12.57)一样，而 W 的公式涉及到先验，所以稍有改变：

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^T \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] + \sigma^2 \mathbf{A} \right]^{-1} \quad (12.63)$$

E 步的公式没有变化。

这样经过多次迭代之后，就可能有某些 α_i 会趋向于无穷大，我们就可以把那些方向去掉。当然，我们也可以进一步去掉那些较大的 α_i ，只留下较小的。不过一般来说去掉无穷大的那些已经足够了。下面是分别使用前面讲过的 EM 法和这里讲的带有稀疏化效果的 EM 法的图示，可以看到，降维效果是十分明显的：



下面看一下 12.2.4 的因子分析 FA，FA 和之前 PPCA 的不同在于，PPCA 的条件分布是：

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}) \quad (12.32)$$

而 FA 是：

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}) \quad (12.64)$$

求解方法什么的也都差不多。这里就不多讲了。说实话我对 FA 不理解得并不深，而且这里讲的 FA 和我之前理解的 FA 似乎也不太一样。我原先以为 FA 就是类似于 PCA，各方向可任意旋转，不一定是特征值从大到小排列的那种，但各方向之间仍保持正交。不知道大家是怎么看 FA 的？

下面重点讲一下 12.3 核 PCA (KPCA)：

第六章讲过，我们可以方便地把内积扩展到非线性核，从而可以利用它来隐式求解非线性情况。这里，我们也利用核方法来求解 PCA。因为要利用内积的非线性核，所以必须把传统 PCA 以内积方式表示出来。前面讲过，传统 PCA 是求样本协方差阵的特征值与特征向量：

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \quad (12.71)$$

样本协方差阵是 $D \times D$ 维的：

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T, \quad (12.72)$$

并且特征向量有约束 $\mathbf{u}_i^T \mathbf{u}_i = 1$

现在我们利用非线性变换 ϕ 把样本变换为 $\phi(\mathbf{x}_n)$ ，暂时为了方便假定变换后的向量有 0 均值

$\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$ ，之后再考虑不是 0 均值的情况。这样，变换后样本的协方差阵就变为：

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \quad (12.73)$$

协方差阵的特征值分解变为：

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad (12.74)$$

现在的形式是 $\phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T$ ，我们想把它变为包含 $\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_n)$ 的形式，以便利用核而不是直接用变换后的样本 $\phi(\mathbf{x}_n)$ 来求解。

把刚才给出的(12.73)代入(12.74)，我们可以得到：

$$\frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \{ \phi(\mathbf{x}_n)^T \mathbf{v}_i \} = \lambda_i \mathbf{v}_i \quad (12.75)$$

这说明 $\mathbf{v}_i$ 可以表示为 $\phi(\mathbf{x}_n)$ 的线性组合：

$$\mathbf{v}_i = \sum_{n=1}^N a_{in} \phi(\mathbf{x}_n). \quad (12.76)$$

但我们不知道这个系数 a 是什么，因为我们并不显式地知道非线性映射 $\phi(\mathbf{x})$ 。所以，这个 a 是我们之后要去求解的对象。

将 12.76 代回到 12.75，可以得到：

$$\frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \sum_{m=1}^N a_{im} \phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^N a_{in} \phi(\mathbf{x}_n). \quad (12.77)$$

两边都左乘 $\phi(\mathbf{x}_l)^T$ ，核这个神秘角色就出现在我们面前了：

$$\frac{1}{N} \sum_{n=1}^N k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^N a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^N a_{in} k(\mathbf{x}_l, \mathbf{x}_n). \quad (12.78)$$

写成矩阵形式，就是：

$$\mathbf{K}^2 \mathbf{a}_i = \lambda_i N \mathbf{K} \mathbf{a}_i \quad (12.79)$$

两边都消去一个 $\mathbf{K}$ （这样做不影响非 0 特征值）：

$$\mathbf{K} \mathbf{a}_i = \lambda_i N \mathbf{a}_i \quad (12.80)$$

和传统 PCA 一样，有归一化约束：

$$1 = \mathbf{v}_i^T \mathbf{v}_i = \sum_{n=1}^N \sum_{m=1}^N a_{in} a_{im} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = \mathbf{a}_i^T \mathbf{K} \mathbf{a}_i = \lambda_i N \mathbf{a}_i^T \mathbf{a}_i. \quad (12.81)$$

这样，这个问题同样变成了特征值分解问题，我们可以(12.80)和(12.81)求出 $\mathbf{a}_i$ 和 λ_i 。我们经过变换的样本 $\phi(\mathbf{x})$ 再经过特征向量投影后的值，也就可以表示为：

$$y_i(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{v}_i = \sum_{n=1}^N a_{in} \phi(\mathbf{x})^T \phi(\mathbf{x}_n) = \sum_{n=1}^N a_{in} k(\mathbf{x}, \mathbf{x}_n) \quad (12.82)$$

注意这里是用 $\mathbf{x}$ 和 $\mathbf{x}_n$ 的核的线性组合来表示的。由于主方向并不是线性的，所以我们无法求出它的形式是什么，而是只能求出如何通过核函数来得到投影到主方向后的降维形式如何用核函数表示出来。

我们先前为了简化计算而假定非线性映射 $\phi(\mathbf{x}_n)$ 有 0 均值，现在去掉这个假设，将其扩展到一般形式。

我们要减去均值以及计算减去均值之后的核矩阵：

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_l) \quad (12.83)$$

$$\begin{aligned} \tilde{K}_{nm} &= \tilde{\phi}(\mathbf{x}_n)^T \tilde{\phi}(\mathbf{x}_m) \\ &= \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_l) \\ &\quad - \frac{1}{N} \sum_{l=1}^N \phi(\mathbf{x}_l)^T \phi(\mathbf{x}_m) + \frac{1}{N^2} \sum_{j=1}^N \sum_{l=1}^N \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_l) \\ &= k(\mathbf{x}_n, \mathbf{x}_m) - \frac{1}{N} \sum_{l=1}^N k(\mathbf{x}_l, \mathbf{x}_m) \\ &\quad - \frac{1}{N} \sum_{l=1}^N k(\mathbf{x}_n, \mathbf{x}_l) + \frac{1}{N^2} \sum_{j=1}^N \sum_{l=1}^N k(\mathbf{x}_j, \mathbf{x}_l). \end{aligned} \quad (12.84)$$

用 K 弯的矩阵形式：

$$\tilde{K} = K - \mathbf{1}_N K - K \mathbf{1}_N + \mathbf{1}_N K \mathbf{1}_N \quad (12.85)$$

就可以像先前一样表示样本经过非线性变换在主方向下的投影。

KPCA 有一个缺点，就是 K 是 $N \times N$ 维的，而传统 PCA 的样本协方差阵是 $D \times D$ 维的，那么在数据规模相对于特征维度来说很大时，计算效率就比较低。

这么顺着讲公式是有点枯燥。。可能还没上次讲点基本的东西比较有趣。特别是一些细节。大家自己也要多看看，PRML 这种书真是每看一遍都有新发现。

我现在讲第 12 章最后的一小部分：12.4 的非线性隐变量模型。

先来看看独立成分分析 ICA。前面讲到，PCA 假设隐变量服从高斯分布，这样想得到各维之间独立的隐变量，只需旋转隐变量使其协方差阵为对角阵即可，即隐变量各维之间线性无关。对于高斯分布来说，线性无关就是独立，因此其联合分布可分解为各维分布的乘积。

如果不要求隐变量一定是高斯分布，但仍保持观察变量和隐变量之间具有线性关系，那就是这一小节讲到 ICA。这样的话，协方差阵为对角阵就不够了，因为对于任意分布来说，线性无关只是独立的必要条件而非充分条件。

得到这个独立解可以有很多方法，比如用高阶统计量而不仅仅是二阶的协方差来度量独立性，或者用互信息等方式来度量。书中介绍了一种描述隐变量的重尾分布：

$$p(z_j) = \frac{1}{\pi \cosh(z_j)} = \frac{1}{\pi(e^{z_j} + e^{-z_j})} \quad (12.90)$$

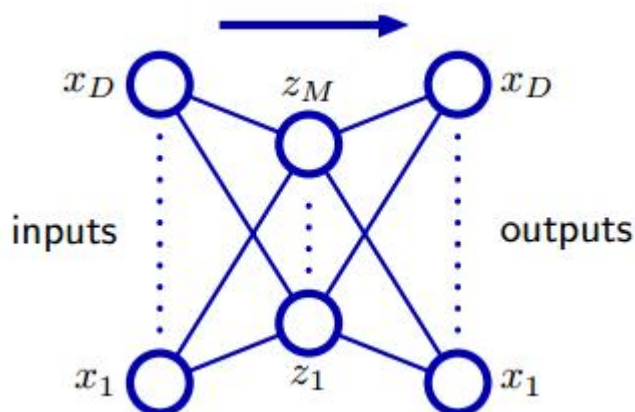
但也可以用其他分布。实际上这类分布可以分为超高斯和次高斯两种。具体细节可以看那本 ICA 的专著，可以在这里下载：<http://ishare.iask.sina.com.cn/f/16979796.html>

<http://ufldl.stanford.edu/tutorial/index.php/ICA> 和 <http://ufldl.stanford.edu/tutorial/index.php/RICA> 也有一些 ICA 的介绍，而且提供了一个不错的代码框架。我们如要实现那里介绍的方法，只需填入一小部分代码即可。

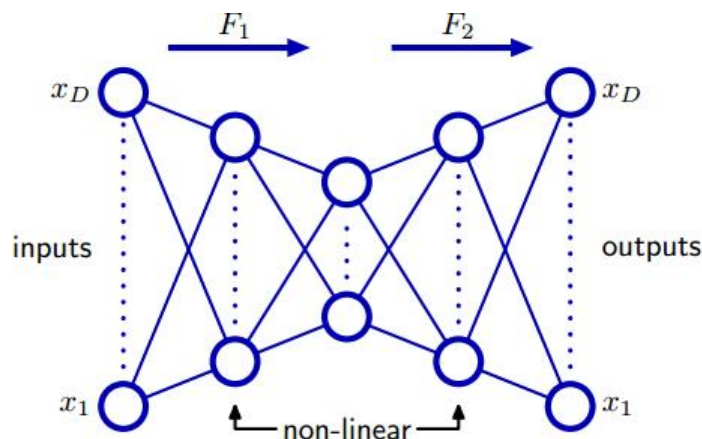
下面看一下 autoassociative neural networks：

这一小节讲到的 autoassociative neural networks，实际上就是最近 deep learning 中研究比较多的 autoencoder。它的基本思想，就是让输入经过多层线性或非线性变换后，得到的输出尽量接近输入，然后取中间一层神经元作为隐变量。

最简单的两层情况是：



多层的话就是：



这里提到一点：中间隐变量的数量 M 需要小于输入变量的维度 D 。不过，现在 deep learning 中一般会对隐变量添加稀疏约束，这样就可以让 M 大于 D 且具有稀疏性，可以更好地描述数据。

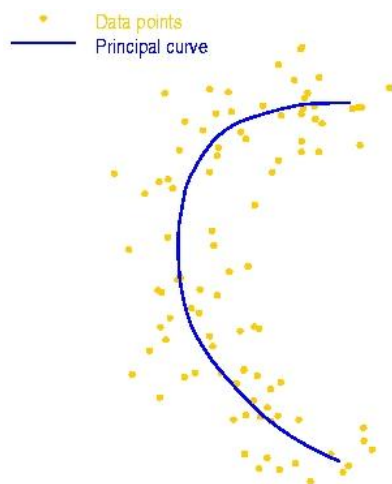
本小节最后也提到，由于包含了非线性变换，因此目标函数不再是简单的二次函数，会变成非凸问题，优化起来容易陷入局部极值。deep learning 解决的问题之一，就是利用逐层优化的 pretraining 方式，使得在最后整体优化之前，找到一个比较好的初始解，这样即使陷入局部极值，找到的也会是比较好的局部极值。优化方式和传统神经网络相似，也是反向传播（BP）算法。具体步骤可参见：

<http://ufldl.stanford.edu/tutorial/index.php/Autoencoders>，和 ICA 一样，也有一个不错的代码框架，实现起来非常简单。

最后看一下 12.4.3 的非线性流形思想。我们可用多种方法来构造非线性流形：

首先，我们可以用分段线性的方法，去捕捉非线性流形的信息。比如，可以先用 kmeans 等聚类算法，把数据分成多个聚类，然后在每个聚类上应用 PCA 等线性降维方法。

其次，我们可以参数化曲线，并像 PCA 那样找到合适的参数，以使用曲线形式表示数据的主成分：



这种方法称作主曲线分析，后来也有主曲面分析等推广。可参见：

<http://www.iro.umontreal.ca/~kegl/research/pcurves/>

最后看一下 MDS、ISOMAP 和 LLE。MDS 是从点对距离出发、希望降维后样本之间的点对距离能够尽量保持之前的点对距离。目标函数和求解方法都比较简单，这里就不赘述了。值得注意的是，这里我们并不需要每个样本的具体坐标，而是只需要知道样本之间的相对关系即可。而且，在欧氏距离下，MDS 得到的坐标和 PCA 投影到最大主方向上的坐标是完全等价的。当然，MDS 中使用的点对距离可以是任意的距离度量，因此得到的结果也可能和 PCA 投影坐标相去甚远。

ISOMAP 和 LLE 是点燃流形学习这把大火的两篇论文，都发表在 science 上。ISOMAP 利用了 MDS，只不过使用的距离度量是测地线距离，也就是通过样本相连的最短距离。比如从 A 经过 B、C 到 D，而且这条路径是最短的 A 与 D 之间的路径，那么这条最短路径的长度就是 A 与 D 之间的测地线距离。LLE 的思想同样很简单：它用样本点 x 的局部邻域内的样本去线性表示 x ，然后希望映射到低维空间后，其局部线性关系尽量保持不变，也就是说，局部的线性表示系数和邻域样本都尽量保持不变。LLE 具体可参见：<http://www.cs.nyu.edu/~roweis/lle/>，有很详细的算法介绍和写得非常漂亮的 matlab 代码。可惜作者 sam roweis 前几年不幸去世，是机器学习界的一大损失。

可以看到，前面介绍的以 ISOMAP 和 LLE 为代表的流形学习，其思想都是在低维空间中保持原先在高维空间中的某种样本之间的关系。这正是流形学习的核心思想：通过流形可能具有的先验信息，以保持样本中蕴含的关系为约束，来找到高维样本嵌入到低维流形的坐标。

后面还介绍了 GTM 和 SOM 两种方法，提到前者可看作后者的概率方法，但我并不太了解它们，所以这里就不多说了。谢谢大家。这章就是这样：)

=====讨论=====

快乐的孩纸(328509423) 22:05:41

autoencoder 实际上就是 input x ，output x ，来学习隐层，然后用隐层去做预测 y ？

Wilbur_中博(1954123) 22:07:31

是的，我做了一些实验，还挺好用的。有一些变种，比如对输入加一些噪声，但输出仍然是不带噪声的 x ，这样就是 denoising autoencoder 了。

快乐的孩纸(328509423) 22:08:47

那如果是两层的 autoencoder 只有一个隐层，要是多个隐层呢？比如中间有 3 个隐层，得到这 3 个隐层，怎么用呢？这意味着 input x 这个 vector 被表达成了，三个有层次的 vectors 了。

Wilbur_中博(1954123) 22:11:11

不是，就用最中间的那个隐层。你可以看成输入经过多次映射到达中间那个隐层，然后又经过多次映射恢复回先前的输入到达输出。

快乐的孩纸(328509423) 22:12:09

那就是 autoencoder 每次都是奇数层，然后得到中间的隐层，抽出来后用作 regression 来预测 y ？

你是谁？(271182928) 22:12:42

对，就保留隐含层。没想到这样对 x 重新表达后，再用 regression 来预测 y 会效果很好。那隐含层的 x 向量，一般就假设是相互独立了吧。也就是开始的 x 里的 correlation 被抹去了。

Wilbur_中博(1954123) 22:15:30

对。比如说从 100 维输入到 80 维，再到 60 维，然后恢复的时候也是先到 80 维，再到 100 维。但其实我想有没有可能不必对称？比如从 100 到 80 到 60，恢复的时候直接重构回 100？我没有试过。我觉得可以试试。现在 autoencoder 的发展也蛮快的。但理论方面其实还差得很多，讲不出什么道理。效果确实还可以。

淘沙(271937125) 22:11:47

能推荐 流形学习 的资料吗？

Wilbur_中博(1954123) 22:15:30

流形学习的资料蛮多的。里面思想和分支也很多。dodo 的这篇文章非常好：

http://blog.sina.com.cn/s/blog_4d92192101008end.html

我觉得还是要抓流形学习的主要思想吧。而且流形学习比较好的一点就是，一般都会提供代码，我们可以从代码中学到很多。浙大的何晓飞和蔡登在这方面也很厉害，你可以去他们的主页看看。

初学者(75553395) 22:16:42

弱问 这样的意思是之后的输入 只用隐含层就行了？

Wilbur_中博(1954123) 22:17:50

对, 新来样本也用那个神经网络的权值映射到隐含层, 然后用隐含层的输出作为分类器等传统方法的输入。不过其实还有各种技巧吧。。需要有监督的 fine-tuning 之类的。

快乐的孩子(328509423) 22:19:49

autoencoder 我不熟悉, 我还有一个问题。现在感觉和 pca 挺像的, 就是 learning 的过程中只依赖原始 input : x , 利用 x 内部的 correlation 和 interaction 来重新表达隐含层。但是从直觉上看, 在 learning 过程中, 不仅考虑 x 的 correlation, 以及 x 与 y 的 correlation, 来最后得到 x 的新表达, 理论上更利于 y 的预测精确度。比如 PLS.

Wilbur_中博(1954123) 22:24:35

你说的对。其实要考虑的东西有很多, 比如 x 本身的结构, x 和 y 的关系, 如果有多任务的话那么 y 也是矩阵形式的, 也可以研究 y 之间的结构和关系。还有我们从 x 提取出来的特征, 特征之间其实也还有关系, 因为提取特征也只是某一方面的假定而已。现在很火的 CNN, 如果在特征之间做一些工作, 效果肯定还能更好。所以机器学习要研究的问题可以说无穷无尽。

快乐的孩子(328509423) 22:26:03

如果是多任务的话, 那是 multitask learning 了。多个 y 之间也有 correlation。Jieping Ye 那个组经常搞这个

XXX<liyitan2144@163.com> 22:26:04

@Wilbur 流型学习, 近两年哪些方面比较火?

Wilbur_中博(1954123) 22:26:21

@快乐的孩子 没错

XXX<liyitan2144@163.com> 22:26:27

Jieping Ye 都快把 multitask learning 玩坏了

快乐的孩子(328509423) 22:26:42

那公式推的一把一把的。@Wilbur 有没有 autoencoder 的好的源码 framework 啊? 给我一个链接, 我能 try 一下么?

Wilbur_中博(1954123) 22:27:55

@XXX 流形学习这两年基本沉寂了, 火的时候已经过去了。不过还是非常实用的。

XXX<liyitan2144@163.com> 22:30:04

问一个肤浅的问题 🤔, 现在火啥? (除了 Deep learning) @Wilbur

Wilbur_中博(1954123) 22:30:51

@快乐的孩子 你可以照着

http://deeplearning.stanford.edu/wiki/index.php/Exercise:Sparse_Autoencoder 实现一下

@XXX 我觉得 hashing trick 也挺火的, 值得关注。另外组合优化相关的近似算法和快速算法也可以关注。

XXX<liyitan2144@163.com> 22:33:04

hashing trick 是指 learning to hash? 组合优化相关的近似算法? 这个哪里有?

Wilbur_中博(1954123) 22:35:23

倒不一定是 learning to hash。简单的 hash code 也可以有不错的效果。比如 google 搞的 WTA hashing 和今年 CVPR 的 best paper 就是不错的 hashing trick 的文章。我个人不是太喜欢 learning to hash。。所有以前有的连续情况的算法都搞成二元约束算法, 有点恶心。

组合优化方面的比如 submodular optimization 了, 还有:

Towards Efficient and Exact MAP-Inference for Large Scale Discrete Computer Vision Problems via Combinatorial Optimization

我觉得做 CV 的人也值得关注。还有挺多的。。这是个大方向。

XXX<liyitan2144@163.com> 22:39:33

那 快速算法 又是指？难道是搞个收敛速率高的算法，再证个 bound？

Wilbur_中博(1954123) 22:45:26

嗯，组合优化的技巧很多。。松弛到较大的解集、限制到较小的解集，看起来背道而驰但还都能用，就是分析起来思路不太一样。还有其他各种各样的思路。不过我其实也不是特别了解，就不妄谈了。。但机器学习或计算机视觉里的组合优化，肯定还是会有很大发展空间的。

第十三章 Sequential Data

主讲人 张巍

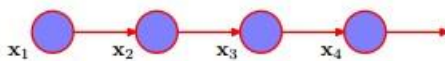
(新浪微博: @张巍_ISCAS)

软件所-张巍<zh3f@qq.com> 19:01:27



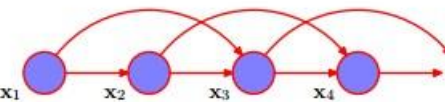
我们开始吧，十三章是关于序列数据，现实中很多数据是有前后关系的，例如语音或者 DNA 序列，例子就不多举了，对于这类数据我们很自然会想到用马尔科夫链来建模：

Figure 13.3 A first-order Markov chain of observations $\{x_n\}$ in which the distribution $p(x_n|x_{n-1})$ of a particular observation x_n is conditioned on the value of the previous observation x_{n-1} .



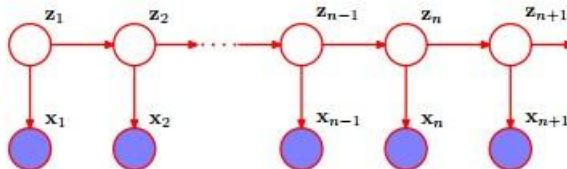
例如直接假设观测数据之间服从一阶马尔科夫链，这个假设显然太简单了，因为很多数据时明显有高阶相关性的，一个解决方法是用高阶马尔科夫链建模：

Figure 13.4 A second-order Markov chain, in which the conditional distribution of a particular observation x_n depends on the values of the two previous observations x_{n-1} and x_{n-2} .



但这样并不能完全解决问题：1、高阶马尔科夫模型参数太多；2、数据间的相关性仍然受阶数限制。一个好的解决方法，是引入一层隐变量，建立如下的模型：

Figure 13.5 We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variable. This important graphical structure forms the foundation both for the hidden Markov model and for linear dynamical systems.



这里我们假设隐变量之间服从一阶马尔科夫链，观测变量由其对应的隐变量生成。从上图可以看出，隐变量是一阶的，但是观测变量之间是全相关的，今天我们主要讨论的就是上图中的模型。如果隐变量是离散的，我们称之为 Hidden Markov Models；如果是连续的，我们称之为: Linear Dynamical Systems。现在我们先来看一下 HMM，从图中可以看出，要完成建模，我们需要指定一下几个分布：

1、转移概率：

$$p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) = \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1}, j, z_{nk}}. \quad (13.7)$$

2、马尔科夫链的初始概率：

$$p(\mathbf{z}_1 | \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}} \quad (13.8)$$

3、生成观测变量的概率(emission probabilities)：

$$p(\mathbf{x}_n | \mathbf{z}_n, \phi) = \prod_{k=1}^K p(x_{nk} | \phi_k)^{z_{nk}}. \quad (13.9)$$

对于 HMM，这里 1 和 2 我们已经假设成了离散分布，由隐变量 Z_n 生成观测数据可以用混合高斯模型或者神经网络，书上的 Z_n 是一个 k 维的布尔变量，由此再看隐变量转移概率公式、观测数据的生成公式就容易理解了。模型建好了，我们接下来主要讨论下面三个问题：

1、学习问题：就是学习模型中的参数；

2、预测问题：即 $p(\mathbf{x}_{N+1} | \mathbf{X})$ ，给定当前序列预测下一个观测变量；

3、解码问题：即 $p(\mathbf{Z} | \mathbf{X})$ ，给定观测变量求隐变量，例如语音识别；

游侠(419504839) 19:24:21

什么是解码问题？

软件所-张巍<zh3f@qq.com> 19:25:18

例如观测到了一段语音，要求识别其对应的句子。@游侠 我前面没怎么举例子，不知道这样说清楚没？

游侠(419504839) 19:27:20

这个和一般说的“推断”一样不

软件所-张巍<zh3f@qq.com> 19:28:27

这个也可以叫推断，只是推断是个比较一般的词汇。

我们来看一下 HMM 有多少参数要学，对应于刚才说到的三个分布，我们也有三组参数要学。

球猫(250992259) 19:30:46

其实就是假设东西是一个马尔科夫模型生成的。。然后把参数用某种方法弄出来，最后根据模型的输出来给答案.....是这样吧？

软件所-张巍<zh3f@qq.com> 19:32:45

@球猫 对，都是这个思路，先把参数学出来，然后就可以做任何想要的推断了，在这里所谓的解码问题只是大家比较关心。

软件所-张巍<zh3f@qq.com> 19:32:51

好，继续，我们先来看 1、学习问题。这里我们用 EM 算法来学习 HMM 的参数：

1、是转移概率对应的转移矩阵；2、初始概率对应的离散分布参数；3、观测变量对应的分布参数（这里暂不指定）。

用 EM 我们要做的就是：

E 步里根据当前参数估计隐变量的后验：

$$p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$$

M 步里最大化下面的期望：

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \theta). \quad (13.12)$$

先来看 M 步，这里相对简单一点，整个模型的全概率展开为：

$$p(\mathbf{X}, \mathbf{Z} | \theta) = p(\mathbf{z}_1 | \pi) \left[\prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) \right] \prod_{m=1}^N p(\mathbf{x}_m | \mathbf{z}_m, \phi) \quad (13.10)$$

把 13.10 代入 13.12,我们会发现计算时需要下面两个式子：

$$p(\mathbf{z}_n | \mathbf{X}, \theta^{\text{old}}) \quad \text{和} \quad p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \theta^{\text{old}}).$$

为了方便，我们就定义：

$$\gamma(z_n) = p(z_n | \mathbf{X}, \theta^{\text{old}}) \quad (13.13)$$

$$\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n | \mathbf{X}, \theta^{\text{old}}). \quad (13.14)$$

这样我们在 E 步就主要要求出这两个式子就行了，当然这也就意味着求出了整个后验 $p(\mathbf{Z} | \mathbf{X}, \theta^{\text{old}})$ ，利用这两个式子，13.12 可以化为：

$$\begin{aligned} Q(\theta, \theta^{\text{old}}) = & \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ & + \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n | \phi_k). \end{aligned} \quad (13.17)$$

这个时候就可以用一些通用方法，例如 Lagrange 来求解了，结果也很简单：

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})} \quad (13.18)$$

$$A_{jk} = \frac{\sum_{n=2}^N \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^K \sum_{n=2}^N \xi(z_{n-1,j}, z_{nl})}. \quad (13.19)$$

对于观测变量的分布参数，与其具体分布形式相关，如果是高斯： $p(\mathbf{x} | \phi_k) = \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$ ，对应的最优解为：

$$\mu_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^N \gamma(z_{nk})} \quad (13.20)$$

$$\Sigma_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^N \gamma(z_{nk})}. \quad (13.21)$$

$$p(\mathbf{x} | \mathbf{z}) = \prod_{i=1}^D \prod_{k=1}^K \mu_{ik}^{x_i z_k}$$

如果是离散：

对应的最优解为：

$$\mu_{ik} = \frac{\sum_{n=1}^N \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^N \gamma(z_{nk})}. \quad (13.23)$$

好，M 步就这样，现在来看 E 步，也是 HMM 比较核心的地方。刚才我们看到，E 步要求的是：

$$\gamma(z_n) = p(z_n | \mathbf{X}, \theta^{\text{old}}) \quad (13.13)$$

$$\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n | \mathbf{X}, \theta^{\text{old}}). \quad (13.14)$$

由马尔科夫的性质，我们可以推出：

$$\gamma(z_n) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n, z_n)p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | z_n)}{p(\mathbf{X})} = \frac{\alpha(z_n)\beta(z_n)}{p(\mathbf{X})} \quad (13.33)$$

其中：

$$\alpha(z_n) \equiv p(\mathbf{x}_1, \dots, \mathbf{x}_n, z_n) \quad (13.34)$$

$$\beta(z_n) \equiv p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | z_n). \quad (13.35)$$

接下来我们就建立 $\alpha(z_n)$ 和 $\beta(z_n)$ 的递推公式

$$\begin{aligned} \alpha(z_n) &= p(\mathbf{x}_1, \dots, \mathbf{x}_n, z_n) \\ &= p(\mathbf{x}_1, \dots, \mathbf{x}_n | z_n)p(z_n) \\ &= p(\mathbf{x}_n | z_n)p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | z_n)p(z_n) \\ &= p(\mathbf{x}_n | z_n)p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1}, z_n) \\ &= p(\mathbf{x}_n | z_n) \sum_{z_{n-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1}, z_{n-1}, z_n) \\ &= p(\mathbf{x}_n | z_n) \sum_{z_{n-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1}, z_n | z_{n-1})p(z_{n-1}) \\ &= p(\mathbf{x}_n | z_n) \sum_{z_{n-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | z_{n-1})p(z_n | z_{n-1})p(z_{n-1}) \\ &= p(\mathbf{x}_n | z_n) \sum_{z_{n-1}} p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1}, z_{n-1})p(z_n | z_{n-1}) \end{aligned}$$

Making use of the definition (13.34) for $\alpha(z_n)$, we then obtain

$$\alpha(z_n) = p(\mathbf{x}_n | z_n) \sum_{z_{n-1}} \alpha(z_{n-1})p(z_n | z_{n-1}). \quad (13.36)$$

其中：

$$\alpha(z_1) = p(\mathbf{x}_1, z_1) = p(z_1)p(\mathbf{x}_1 | z_1) = \prod_{k=1}^K \{\pi_k p(\mathbf{x}_1 | \phi_k)\}^{z_{1k}} \quad (13.37)$$

这样我们从 $\alpha(z_1)$ 开始，可以递推出所有的 $\alpha(z_n)$ ，对于 $\beta(z_n)$ ，也进行类似的推导：

$$\begin{aligned} \beta(z_n) &= p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | z_n) \\ &= \sum_{z_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N, z_{n+1} | z_n) \\ &= \sum_{z_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | z_n, z_{n+1})p(z_{n+1} | z_n) \\ &= \sum_{z_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | z_{n+1})p(z_{n+1} | z_n) \\ &= \sum_{z_{n+1}} p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N | z_{n+1})p(\mathbf{x}_{n+1} | z_{n+1})p(z_{n+1} | z_n). \end{aligned}$$

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n). \quad (13.38)$$

从上式可以看到 $\beta(\mathbf{z}_n)$ 是一个逆推过程, 所以我们需要初始值 $\beta(\mathbf{z}_N)$, 定义 13.35 并没有明确 $\beta(\mathbf{z}_N)$ 的定义:

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n) \quad (13.34)$$

$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n). \quad (13.35)$$

因为 $\mathbf{z}_N$ 后没有观测数据, 不过我们可以从

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n) \beta(\mathbf{z}_n)}{p(\mathbf{X})} \quad (13.33)$$

, 得出:

$$p(\mathbf{z}_N | \mathbf{X}) = \frac{p(\mathbf{X}, \mathbf{z}_N) \beta(\mathbf{z}_N)}{p(\mathbf{X})} \quad (13.39)$$

这样 $\beta(\mathbf{z}_N)$ 就等于 1, 现在我们可以方便的求出所有的 $\alpha(\mathbf{z}_n)$ 和 $\beta(\mathbf{z}_n)$ 了, 利用 13.13 也就可以求出所

有的 $\gamma(\mathbf{z}_n)$ 。类似的, 我们可以求出 $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$:

$$\begin{aligned} \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}) \\ &= \frac{p(\mathbf{X} | \mathbf{z}_{n-1}, \mathbf{z}_n) p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}) \beta(\mathbf{z}_n)}{p(\mathbf{X})} \end{aligned} \quad (13.43)$$

这样在 M 步里求解所需要的分布就都求出来了, 也就可以用 EM 来学习 HMM 的参数了, 这里式子比较多, 大家自己推一下会比较好理解。

第一个学习问题就这样了, 接下来是预测问题, 预测问题可以直接推导:

$$\begin{aligned} p(\mathbf{x}_{N+1} | \mathbf{X}) &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}, \mathbf{z}_{N+1} | \mathbf{X}) \\ &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1}) p(\mathbf{z}_{N+1} | \mathbf{X}) \\ &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}, \mathbf{z}_N | \mathbf{X}) \\ &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} | \mathbf{z}_N) p(\mathbf{z}_N | \mathbf{X}) \\ &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} | \mathbf{z}_N) \frac{p(\mathbf{z}_N, \mathbf{X})}{p(\mathbf{X})} \\ &= \frac{1}{p(\mathbf{X})} \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} | \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} | \mathbf{z}_N) \alpha(\mathbf{z}_N) \end{aligned} \quad (13.44)$$

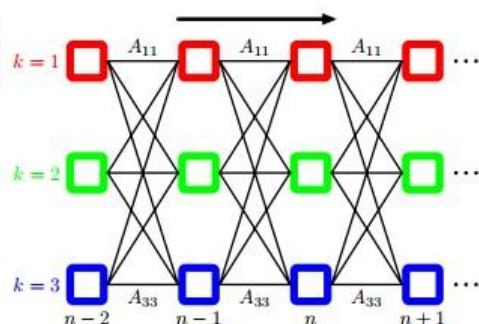
现在就剩最后一个解码问题，也就是 $\text{argmax}_Z \{p(Z|X)\}$ ，刚才我们在 E 步已经求出了：

$$\gamma(z_n) = p(z_n|X, \theta^{\text{old}}) \quad (13.13)$$

$$\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n|X, \theta^{\text{old}}). \quad (13.14)$$

但是现在的问题要复杂一点，因为我们要求概率最大的隐变量序列，用 13.13 可以求出单个隐变量，但是他们连在一起形成的整个序列可能概率很小，这个问题可以归结为一个动态规划：

Figure 13.7 If we unfold the state transition diagram of Figure 13.6 over time, we obtain a lattice, or trellis, representation of the latent states. Each column of this diagram corresponds to one of the latent variables z_n .



我们把 HMM 化成如上图的样子，最大化后验等价于最大化全概率，对于上图中的边，我们赋值为：

$$\log \left(p(z_n|z_{n-1}, A) = \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1}, j, z_{nk}} \right)$$

初始节点赋值为：

$$\log(p(z_1|\pi) = \prod_{k=1}^K \pi_k^{z_{1k}} * p(x_1|z_1))$$

其余节点赋值为：

$$\log(p(x_n|z_n, \phi) = \prod_{k=1}^K p(x_n|\phi_k)^{z_{nk}})$$

这样任何一个序列 Z ，其全概率等于 $\exp(Z \text{ 对应路径上节点和边的值求和})$ ，这样，解码问题就转化为

一个最长路径问题，用动态规划可以直接求解。大家看这里有没有问题，HMM 的主要内容就是这些 😊

接下来的 Linear Dynamical Systems 其实和 HMM 大同小异，只是把离散分布换成了高斯，然后就是第二章公式的反复应用，都是细节问题，就不在这里讲了，大家看看有问题我们可以讨论。这一章还是式子主导的，略过了不少式子，大家推的时候有问题我们可以随时讨论。

天涯游(872352128) 21:09:21

我对 hmm 的理解，觉得这麻烦的是概率的理解的了，概率分解才是 hmm 的核心，当然了还有动态规划了。概率分解其实是实验事件的分解，如前向 和后向了，还有就是 EM 算法了。

第十四章 Combining Models

主讲人 网神

(新浪微博: @豆角茄子麻酱凉面)

网神(66707180) 18:57:18

大家好，今天我们讲一下第 14 章 combining models，这一章是联合模型，通过将多个模型以某种形式结合起来，可以获得比单个模型更好的预测效果。包括这几部分：

committees, 训练多个不同的模型，取其平均值作为最终预测值。

boosting: 是 committees 的特殊形式，顺序训练 L 个模型，每个模型的训练依赖前一个模型的训练结果。

决策树：不同模型负责输入变量的不同区间的预测，每个样本选择一个模型来预测，选择过程就像在树结构中从顶到叶子的遍历。

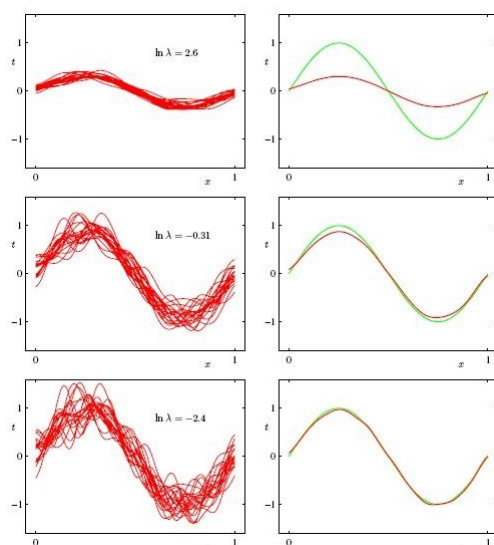
conditional mixture model 条件混合模型：引入概率机制来选择不同模型对某个样本做预测，相比决策树的硬性选择，要有很多优势。

本章主要介绍了这几种混合模型。讲之前，先明确一下混合模型与 Bayesian model averaging 的区别，贝叶斯模型平均是这样的：假设有 H 个不同模型 h，每个模型的先验概率是 $p(h)$ ，一个数据集的分布是：

$$p(\mathbf{X}) = \sum_{h=1}^H p(\mathbf{X}|h)p(h).$$

整个数据集 X 是由一个模型生成的，关于 h 的概率仅仅表示是由哪个模型来生成的 这件事的不确定性。而本章要讲的混合模型是数据集中，不同的数据点可能由不同模型生成。看后面讲到的内容就明白了。

首先看 committees，committees 是一大类，包括 boosting，首先将最简单的形式，就是讲多个模型的预测的平均值作为最后的预测。主要讲这么做的合理性，为什么这么做会提高预测性能。从频率角度的概念，bias-variance trade-off 可以解释，这个理论在 3.5 节讲过，我们把这个经典的图 copy 过来：



这个图大家都记得吧，左边一列是对多组数据分别训练得到一个模型，对应一条 sin 曲线，看左下角这个图，正则参数 lamda 取得比较小，得到一个 bias 很小，variance 很大的一个模型。每条线的 variance 都很大，这样模型预测的错误就比较大，但是把这么多条曲线取一个平均值，得到右下角图上的红色线，红色线跟真实 sin 曲线也就是蓝色线 基本拟合。所以用平均之后模型来预测，variance 准确率就提高了很多，这是直观上来看，接下来从数学公式推导看下：

有一个数据集，用 bootstrap 方法构造 M 个不同的训练集 bootstrap 方法就是从数据集中随机选 N 个放到训练集中，做 M 次，就得到 M 个训练集，M 个训练集训练的到 M 个模型，用 $y_m(\mathbf{x})$ 表示，那么用 committees 方法，对于某个 x，最终预测值是：

$$y_{\text{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}).$$

我们来看这个预测值是如何比单个 $y_m(\mathbf{x})$ 预测值准确的，假设准确的预测模型是 $h(\mathbf{x})$ ，那么训练得到的 $y(\mathbf{x})$ 跟 $h(\mathbf{x})$ 的关系是：

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}).$$

平均平方和错误如下：

$$\mathbb{E}_{\mathbf{x}} [\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

也就是单个模型的期望 error 是：

$$\mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

如果用 M 个模型分别做预测，其平均错误是：

$$E_{\text{AV}} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2]$$

而如果用 committees 的结果来做预测，其期望错误是：

$$\begin{aligned} E_{\text{COM}} &= \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right] \end{aligned}$$

这个 $\frac{1}{M}$ 跑到了平方的里面，如果假设不同模型的 error 都是 0 均值，并且互不相关，也就是：

$$\begin{aligned} \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})] &= 0 \\ \mathbb{E}_{\mathbf{x}} [\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})] &= 0, \quad m \neq l \end{aligned}$$

就可以得到：

$$E_{\text{COM}} = \frac{1}{M} E_{\text{AV}}.$$

在不同模型 error 互不相关的假设的下，committees 错误是单个模型 error 的 1/M，但实际上，不同模型的 error 通常是相关的，因此 error 不会减少这么多，但肯定是小于单个模型的 error，接下来讲 boosting，可以不考虑那个假设，取得实质的提高。boosting 应该是有不同的变种，其中最出名的就是 AdaBoost, adaptive boosting. 它可以将多个弱分类器结合，取得很好的预测效果，所谓弱分类器就是，只要比随即预测强一点，大概就是只要准确率超 50% 就行吧，这是我的理解。

boosting 的思想是依次预测每个分类器，每个分类器训练时，对每个数据点加一个权重。训练完一个分类器模型，该模型分错的，再下一个模型训练时，增大权重；分对的，减少权重，具体的算法如下，我把整个算法帖出来，再逐步解释：

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \dots, N$.
2. For $m = 1, \dots, M$:
 - (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \quad (14.15)$$

where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

- (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}} \quad (14.16)$$

and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \quad (14.17)$$

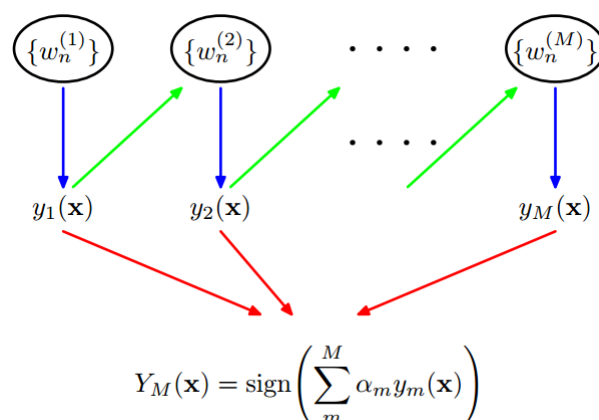
- (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(y_m(\mathbf{x}_n) \neq t_n) \} \quad (14.18)$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right). \quad (14.19)$$

大家看下面这个图比较形象：



第一步，初始化每个数据点的权重为 $1/N$ 。接下来依次训练 M 个分类器，每个分类器训练时，最小化加权的错误函数(14.15)，错误函数看上面贴的算法，从这个错误函数可以看出，权重相同时，尽量让更多的 x 分类正确，权重不同时，优先让权重大的 x 分类正确，训练完一个模型后，式(14.16)计算 ϵ_m ，既分类

错误的样本的加权比例. 然后式(14.17)计算:

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$$

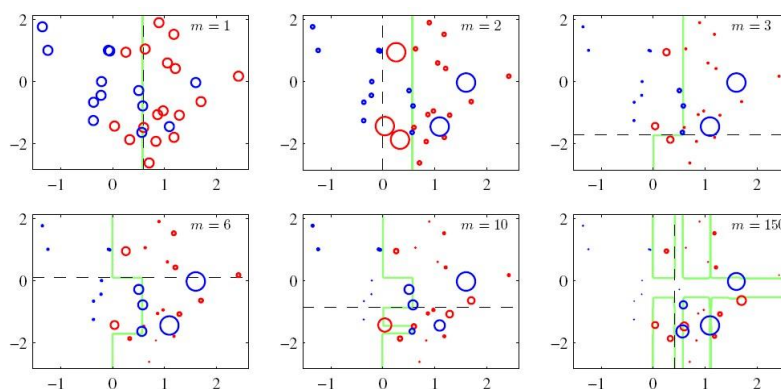
只要分类器准确率大于 50%, ϵ_m 就小于 0.5, α_m 就大于 0。而且 ϵ_m 越小(既对应的分类器准确率越高), α_m 就越大, 然后用 α_m 更新每个数据点的权重, 即式(14.18):

$$w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(y_m(\mathbf{x}_n) \neq t_n) \} \quad (14.18)$$

可以看出, 对于分类错误的数据点, α_m 大于 0, 所以 $\exp(a)$ 就大于 1, 所以权重变大。但是从这个式子看不出, 对于分类正确的样本, 权重变小。这个式子表明, 分类正确的样本, 其权重不变, 因为 $\exp(0)=1$. 这是个疑问。如此循环, 训练完所有模型, 最后用式(14.19)做预测:

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right)$$

从上面过程可以看出, 如果训练集中某个样本在逐步训练每个模型时, 一直被分错, 他的权重就会一直变大, 最后对应的 α_m 也越来越大, 下面看一个图例:



图中有蓝红两类样本, 分类器是单个的平行于轴线的阈值, 第一个分类器(m=1)把大部分点分对了, 但有 2 个蓝点, 3 个红点不对, m=2 时, 这几个错的就变大了, 圈越大, 对应其权重越大, 后面的分类器似乎是专门为了这个几个错分点而在努力工作, 经过 150 个分类器, 右下角那个图的分割线已经很乱了, 看不出到底对不对, 应该是都已经分对了。

网神(66707180) 19:59:59

@ZealotMaster 不知道是否明白点了, 大家有啥问题?

ZealotMaster(850458544) 20:00:14

嗯, 清晰一些了, 这个也涉及 over fitting 嘛? 感觉 m=150 好乱.....

苦瓜炒鸡蛋(852383636) 20:02:23

是过拟合

网神(66707180) 20:02:25

不同的分割线, 也就是不同的模型, 是主要针对不同的点的, 针对哪些点, 由模型组合时的系数 α_m 来影响。

苦瓜炒鸡蛋(852383636) 20:04:50

这是韩家炜 那个数据挖掘书的那一段：

"How does boosting compare with bagging?" Because of the way boosting focuses on the misclassified tuples, it risks overfitting the resulting composite model to such data. Therefore, sometimes the resulting "boosted" model may be less accurate than a single model derived from the same data. Bagging is less susceptible to model overfitting. While both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

网神(66707180) 20:04:56

嗯，这章后面也讲到了 boosting 对某些错分的样本反应太大，一些异常点会对模型造成很大的影响。

=====

接下来讲 boosting 的错误函数，我们仔细看下对 boosting 错误函数的分析，boosting 最初用统计学习理论来分析器泛化错误的边界 bound，但后来发现这个 bound 太松，没有意义。实际性能比这个理论边界好得多，后来用指数错误函数来表示。从优化指数损失函数来解释 adaboost 比较直观，每次固定其他分类器和系数将常量分出来，能推出单分类器的损失函数及系数，再根据常量的形式能得出下一步数据权重的更新方式。指数错误函数如下：

$$E = \sum_{n=1}^N \exp \{-t_n f_m(\mathbf{x}_n)\}$$

其中 N 是 N 个样本点， $f_m(\mathbf{x})$ 是多个线性分类器的线性组合：

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^m \alpha_l y_l(\mathbf{x})$$

$t_n \in \{-1, 1\}$ 是分类的目标值。我们的目标是训练系数 α_l 和分类器 $y_l(\mathbf{x})$ 的参数，使 E 最小。

最小化 E 的方法，是先只针对一个分类器进行最小化，而固定其他分类器的参数，例如我们固定

$y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$ 和其系数 $\alpha_1, \dots, \alpha_{m-1}$ ，只针对 α_m and $y_m(\mathbf{x})$ 做优化，这样错误函

数 E 可以改写为：

$$\begin{aligned} E &= \sum_{n=1}^N \exp \left\{ -t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \\ &= \sum_{n=1}^N w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \end{aligned}$$

也就是把固定的分类器的部分都当做一个常量： $w_n^{(m)} = \exp \{-t_n f_{m-1}(\mathbf{x}_n)\}$ ，只保留

$y_m(\mathbf{x})$ 相关的部分。我们用 $\mathcal{T}_m$ 表示 $y_m(\mathbf{x})$ 分对的数据集， $\mathcal{M}_m$ 表示分错的数据集，E 又可以写成如下形式：

$$\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^N w_n^{(m)}.
\end{aligned}$$

上式中，因为将数据分成两部分，也就确定了 t_n 个 $y_m(\mathbf{x}_n)$ 是否相等，所以消去了这两个变量
看起来清爽点了：

$$= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^N w_n^{(m)}. \quad (14.22)$$

这里面后一项是常量，前一项就跟前面 boosting 的算法里所用的错误函数：

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \quad \text{形式一样了。}$$

上面是将：

$$= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^N w_n^{(m)}. \quad (14.23)$$

对 $y_m(\mathbf{x}_n)$ 做最小化得出的结论,即指数错误函数就是 boosting 里求单个模型时所用的错误函数类似，

也可以得到指数错误函数里的 α_m 就是 boosting 里的 ϵ_m ，确定了 α_m and $y_m(\mathbf{x})$ ，根据

$$\begin{aligned}
E &= \sum_{n=1}^N \exp \left\{ -t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \\
&= \sum_{n=1}^N w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\} \quad (14.22) \quad \text{以及 } w_n^{(m)} = \exp \{ -t_n f_{m-1}(\mathbf{x}_n) \}
\end{aligned}$$

可以得到，更新 w 的方法：

$$w_n^{(m+1)} = w_n^{(m)} \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\}.$$

又因为 $t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$ 有：

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp \{ \alpha_m I(y_m(\mathbf{x}_n) \neq t_n) \} \quad \text{这又跟 boosting 里更}$$

新数据点权重的方法以一致。

总之，就是想说明，用指数错误函数可以描述 boosting 的 error 分析,接下来看看指数错误函数的性质，再贴一下指数错误函数的形式：

$$E = \sum_{n=1}^N \exp \{-t_n f_m(\mathbf{x}_n)\}$$

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^m \alpha_l y_l(\mathbf{x})$$

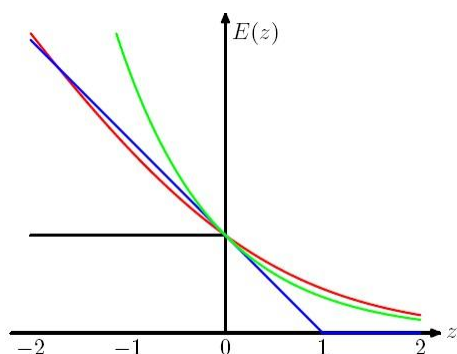
其期望 error 是：

$$\mathbb{E}_{\mathbf{x},t} [\exp\{-ty(\mathbf{x})\}] = \sum_t \int \exp\{-ty(\mathbf{x})\} p(t|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}.$$

然后最所有的 $y(\mathbf{x})$ 做 variational minimization, 得到：

$$y(\mathbf{x}) = \frac{1}{2} \ln \left\{ \frac{p(t=1|\mathbf{x})}{p(t=-1|\mathbf{x})} \right\}$$

这是 half the log-odds，看下指数错误函数的图形：



绿色的线是指数错误函数 $z = ty(\mathbf{x})$, 可以看到，对于分错的情况，既 $z < 0$ 时，绿色的线呈指数趋势

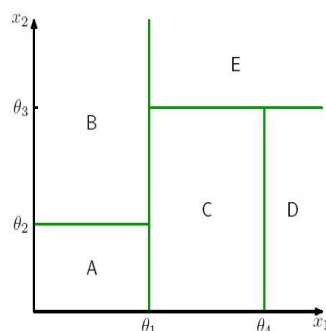
上升，所以异常点会对训练结果的影响很大。图中红色的线是 corss-entropy 错误，是逻辑分类中用的错误函数 对分错的情况 随错误变大 其函数值基本是线性增加的，蓝色线是 svm 用的错误函数，叫 hinge 函数。

=====

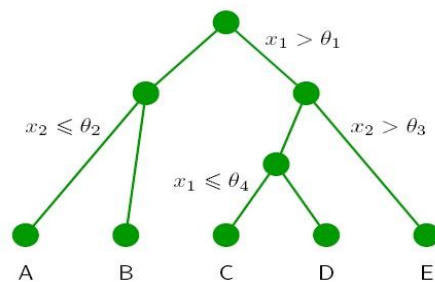
大家有没有要讨论的？公式比较多，需要推导一下才能理解。接下来讲决策树和条件混合模型。决策树概念比较简单，容易想象是什么样子的，可以认为决策树是多个模型，每个模型负责 x 的一个区间的预测，通过树形结构来寻找某个 x 属于哪个区间，从而得到预测值。

决策树有多个算法比较出名，ID3, C4.5, CART，书上以 CART 为例讲的。

CART 叫 classification and regression trees 先看一个图示：



这个二维输入空间，被划分成 5 个区间，每个区间的类别分别是 A-E，它的决策树如下，很简单明了：



决策树在一些领域应用比较多，最主要的原因是，他有很好的可解释性。那如何训练得到一个合适的决策树呢？也就是如何决定需要哪些节点，节点上选择什么变量作为判断依据，以及判断的阈值是多大。

首先错误函数是残差的平方和，而树的结构则用贪心策略来演化。

开始只有一个节点，也就是根节点，对应整个输入空间，然后将空间划分为 2，在多个划分选择之中，选择使残差最小的那个划分，书上提到这个区间划分以及划分点阈值的选择，是用穷举的方法。然后对不同的叶子节点再分别划分子区间。这样树不停长大，何时停止增加节点呢？简单的方法是当残差小于某个值的时候。但经验发现经常再往多做一些划分后，残差又可以大幅降低

所以通常的做法是，先构造一个足够大的树，使叶子节点多到某个数量，然后再剪枝，合并的原则是使叶子尽量减少，同时残差又不要增大。

$$C(T) = \sum_{\tau=1}^{|T|} Q_{\tau}(T) + \lambda|T| \quad \text{其中:} \quad Q_{\tau}(T) = \sum_{\mathbf{x}_n \in \mathcal{R}_{\tau}} \{t_n - y_{\tau}\}^2.$$

也就是最小化这个值：

$$y_{\tau} = \frac{1}{N_{\tau}} \sum_{\mathbf{x}_n \in \mathcal{R}_{\tau}} t_n$$

y_{τ} 是某个叶子负责的那个区间里的所有数据点的预测值的平均值：

t_n 是某个数据点的预测值， $|T|$ 是叶子的总数， λ 控制残差和叶子数的 trade-off，这是剪枝的依据。

以上是针对回归说的，对于分类，剪枝依据仍然是：

$$C(T) = \sum_{\tau=1}^{|T|} Q_{\tau}(T) + \lambda|T|$$

只是 $Q(T)$ 不再是残差的平方和，而是 cross-entropy 或 Gini index

交叉熵的计算是：

$$Q_{\tau}(T) = \sum_{k=1}^K p_{\tau k} \ln p_{\tau k}$$

$p_{\tau k}$ 是第 τ 个叶子，也就是第 τ 个区间里的数据点，被赋予类别 k 的比例。

Gini index 计算是：

$$Q_{\tau}(T) = \sum_{k=1}^K p_{\tau k} (1 - p_{\tau k}).$$

决策树的优点是直观，可解释性好。缺点是每个节点对输入空间的划分都是根据某个维度来划分的，在坐标空间里看，就是平行于某个轴来划分的，其次，决策树的划分是硬性的，在回归问题里，硬性划分更明显。

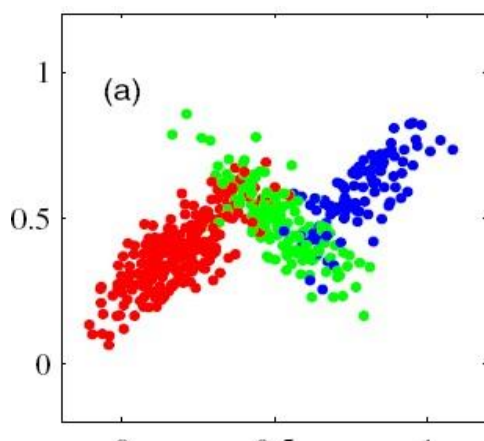
=====

决策树就讲这么多,接下来是 conditional mixture models 条件混合模型。条件混合模型,我的理解是,将不同的模型依概率来结合,这部分讲了线性回归的条件混合,逻辑回归的条件混合,和更一般性的混合方法 mixture of experts,首先看线性回归的混合。

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

第 9 章讲过高斯混合模型,其模型是这样的:

这是用多个高斯密度分布来拟合输入空间,就像这种 $\mathbf{x}$ 的分布:



线性回归混合的实现,可以把这个高斯混合分布扩展成条件高斯分布,模型如下:

$$p(t|\boldsymbol{\theta}) = \sum_{k=1}^K \pi_k \mathcal{N}(t | \mathbf{w}_k^T \boldsymbol{\phi}, \beta^{-1})$$

这里面,有 K 个线性回归 $\mathbf{w}_k^T \boldsymbol{\phi}$, 其权重参数是 $\mathbf{W}_k$, 将多个线性回归的预测值确定的概率联合起来,得

到最终预测值的概率分布。模型中的参数 $\boldsymbol{\theta}$ 包括 $\mathbf{W} = \{\mathbf{w}_k\}$ $\boldsymbol{\pi} = \{\pi_k\}$, and β 三部分,接下来看

如何训练得到这些参数,总体思路是用第 9 章介绍的 EM 算法,引入一个隐藏变量 $\mathbf{Z} = \{\mathbf{z}_n\}$, 每个训

练样本点对应一个 $\mathbf{z}_n$, $\mathbf{z}_n$ 是一个 K 维的二元向量, $z_{nk} \in \{0, 1\}$, 如果第 k 个模型负责生成第 n 个

数据点,则 z_{nk} 等于 1, 否则等于 0, 这样,我们可以写出 log 似然函数:

$$\ln p(\mathbf{t}, \mathbf{Z} | \boldsymbol{\theta}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \ln \{ \pi_k \mathcal{N}(t_n | \mathbf{w}_k^T \boldsymbol{\phi}_n, \beta^{-1}) \}$$

然后用 EM 算法结合最大似然估计来求各个参数, EM 算法首先选择所有参数的初始值,假设是 $\boldsymbol{\theta}^{\text{old}}$

在 E 步根据这些参数值,可以得到模型 k 相对于数据点 n 的后验概率:

$$\gamma_{nk} = \mathbb{E}[z_{nk}] = p(k | \boldsymbol{\phi}_n, \boldsymbol{\theta}^{\text{old}}) = \frac{\pi_k \mathcal{N}(t_n | \mathbf{w}_k^T \boldsymbol{\phi}_n, \beta^{-1})}{\sum_j \pi_j \mathcal{N}(t_n | \mathbf{w}_j^T \boldsymbol{\phi}_n, \beta^{-1})}.$$

书上提高一个词 这个后验概率又叫做 responsibilities 大概是这个数据点 n 由模型 k 负责生成的概率吧，有了这个 responsibilities，就可以计算似然函数的期望值了，如下：

$$Q(\theta, \theta^{\text{old}}) = \mathbb{E}_{\mathbf{Z}} [\ln p(\mathbf{t}, \mathbf{Z} | \theta)] = \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} \{ \ln \pi_k + \ln \mathcal{N}(t_n | \mathbf{w}_k^T \phi_n, \beta^{-1}) \}$$

在 EM 的 M 步，我们令 γ_{nk} 为固定值，最大化这个期望值 $Q(\theta, \theta^{\text{old}})$ ，从而求得新的参数 θ

首先看 π_k ， π_k 是各个模型的混合权重系数，满足 $\sum_k \pi_k = 1$ ，用拉格朗日乘子法，可以求得：

$$\pi_k = \frac{1}{N} \sum_{n=1}^N \gamma_{nk}.$$

接下来求线性回归的参数 $\mathbf{w}_k$ ，将似然函数期望值的式子里的高斯分布展开，可以得到如下式子：

$$Q(\theta, \theta^{\text{old}}) = \sum_{n=1}^N \gamma_{nk} \left\{ -\frac{\beta}{2} (t_n - \mathbf{w}_k^T \phi_n)^2 \right\} + \text{const}$$

要求第 k 个模型的 $\mathbf{w}_k$ ，其他模型的 $\mathbf{w}$ 都对其没有影响，可以统统归做后面的 const，这是因为 log 似然函数，每个模型之间是相加的关系，一求导数，其他模型的项就都消去了。上面的式子是一个加权的最小平方损失函数，每个数据点对应一个权重 $\beta \gamma_{nk}$ ，这个权重可以看做是数据点的有效精度，将这个式子对 $\mathbf{w}_k$ 求导，可以得到：

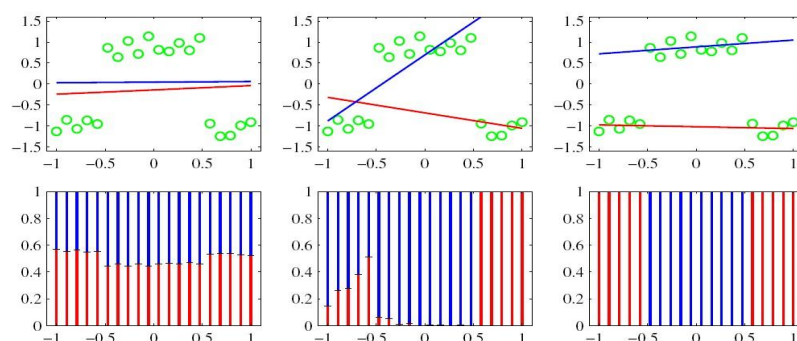
$$0 = \sum_{n=1}^N \gamma_{nk} (t_n - \mathbf{w}_k^T \phi_n) \phi_n$$

最后求得 $\mathbf{w}_k = (\Phi^T \mathbf{R}_k \Phi)^{-1} \Phi^T \mathbf{R}_k \mathbf{t}$.

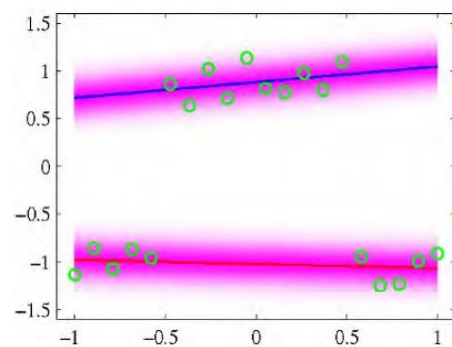
其中 $\mathbf{R}_k = \text{diag}(\gamma_{nk})$ ，同样，对 β 求导，得到：
$$\frac{1}{\beta} = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} (t_n - \mathbf{w}_k^T \phi_n)^2$$

这样的到了所有的参数的新值，再重复 E 步和 M 步，迭代求得满意的最终的参数值。

接下来看一个线性回归混合模型 EM 求参数的图示，两条直线对应两个线性回归模型用 EM 方法迭代求参数，两条直线最终拟合两部分数据点，中间和右边分别是经过了 30 轮和 50 轮迭代，下面那一排，表示每一轮里，每个数据点对应的 responsibilities，也就是类 k 对于数据点 n 的后验概率：



最终求得的混合模型图示：



接下来讲逻辑回归混合模型，逻辑回归模型本身就定义了一个目标值的概率，所以可以直接用逻辑回归本身作为概率混合模型的组件，其模型如下：

$$p(t|\phi, \theta) = \sum_{k=1}^K \pi_k y_k^t [1 - y_k]^{1-t}$$

其中 $y_k = \sigma(\mathbf{w}_k^T \phi)$ ，这里的参数 θ 包括 $\{\pi_k\}$ and $\{\mathbf{w}_k\}$ 两部分。求参数的方法也是用 EM，这里就不细讲了，要提一下的是在 M 步，似然函数不是一个 closed-form 的形式，不能直接求导来出参数，需要用 IRLS(iterative reweighted least squares)等迭代方法来求，下图是一个逻辑回归混合模型的训练的图示，左图是两个类别的数据，蓝色和红色表示实际的分布，中间图是用一个逻辑回归来拟合的模型，右图是用两个逻辑回归混合模型拟合的图形：

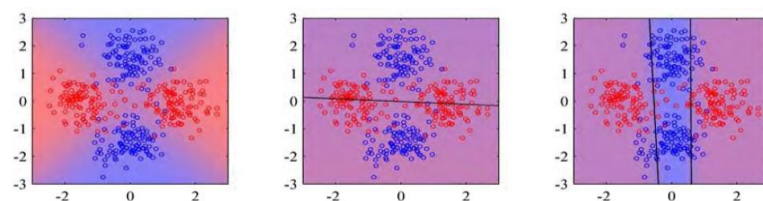


Figure 14.10 Illustration of a mixture of logistic regression models. The left plot shows data points drawn from two classes denoted red and blue, in which the background colour (which varies from pure red to pure blue) denotes the true probability of the class label. The centre plot shows the result of fitting a single logistic regression model using maximum likelihood, in which the background colour denotes the corresponding probability of the class label. Because the colour is a near-uniform purple, we see that the model assigns a probability of around 0.5 to each of the classes over most of input space. The right plot shows the result of fitting a mixture of two logistic regression models, which now gives much higher probability to the correct labels for many of the points in the blue class.

接下来讲 mixtures of experts，mixture of experts 是更一般化的混合模型，前面讲的两个混合模型，其混合系数是固定值，我们可以让混合系数是输入 x 的函数即：

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^K \pi_k(\mathbf{x}) p_k(\mathbf{t}|\mathbf{x}).$$

系数 $\pi_k(\mathbf{x})$ 叫做 gating 函数，组成模型 $p_k(\mathbf{t}|\mathbf{x})$ 叫做 experts，每个 experts 可以对输入空间的不同区域建模，对不同区域的输入进行预测，而 gating 函数则决定一个 experts 该负责哪个区域。

gating 函数满足 $0 \leq \pi_k(\mathbf{x}) \leq 1$ and $\sum_k \pi_k(\mathbf{x}) = 1$ ，因此可以用 softmax 函数来作为 gating 函数，如果 expert 函数也是线性函数，则整个模型就可以用 EM 算法来确定。在 M 步，可能需要用 IRLS，来迭代求解，这个模型仍然有局限性，更一般化的模型是 hierarchical mixture of experts 也就是混合模型的每个组件又可以是一个混合模型。好了就讲这么多吧，书上第 14 章的内容都讲完了。