统计学习方法

本书已经出第二版，2019年5月之后所有内容更新参考第二版第一次印刷。

第一版内容见Release first_edition

工具包

为方便学习，整理一些工具说明。

- GitHub的markdown公式支持一般，推荐使用Chrome插件TeX All the Things来渲染TeX公式,，本地Markdown编辑器推荐Typora，注意Ctrl+, 打开Preferences，Syntax Support部分勾选inline Math。Ubuntu和Windows都正常。
- math_markdown.pdf为math_markdown.md的导出版本， 方便查看使用， markdown版本为最新版本，基本覆盖了书中用到的数学公式的$\LaTeX$表达方式。
- ref_downloader 是一个参考文献下载脚本，这本书一定要配合参考文献看，每章的大参考文献一定要看，对书的内容理解会很有帮助。
- glossary_index 是一个非正式的术语索引，这个书后面是有一个的，但是不方便展开，在这个部分添加了部分扩展的内容。
- symbol_index 是一个非正式的符号索引，第一版中有符号说明，第二版没有了，可能是无监督这部分涉及到的符号真的是太多了，总之，保留这部分，在感觉混淆的时候可以查下，看看是否有帮助。
- errata_se 非官方的errata，供参考。如果有内容感觉不清楚，可以参考看看，希望有帮助。

前前言

- 2019年5月，期待许久的第二版发布了，第一时间下了订单，预计母亲节这天可以发货。
- 5月13日新书到手，第二版配了一张新照片，短发，比之前显得年轻...
- 第二版修改了标点符号，第一版中逗号中文，句号英文。第二版将之前的英文句号更改成了中文句号。
- 第二版取消了符号表，可能是因为同一本书前后有些地方用了不同的符号？所以在这个repo里面，我们尝试加上符号表做说明，方便查询。
- 第二版增加了八个无监督的学习方法，至此，数据挖掘十大算法除了Apriori，全了。

如果需要引用这个Repo:

格式：`SmirkCao, Lihang, (2018), GitHub repository, https://github.com/SmirkCao/Lihang`

或者

```
@misc{SmirkCao,
  author = {SmirkCao},
  title = {Lihang},
  year = {2018},
  publisher = {GitHub},
  journal = {GitHub repository},
  howpublished = {\url{https://github.com/SmirkCao/Lihang}},
  commit = {c5624a9bd757a5cc88e78b85b89e9221deb08270}
}
```

前言

这部分内容并不对应《统计学习方法》中的前言，书中的**前言**写的也很好，引用如下:

> 1. 在内容选取上，侧重介绍那些最重要，最常用的方法，特别是关于**分类与标注**问题的方法.
> 2. 力图用统一框架来论述所有方法，使全书整体不失系统性。
> 3. 适用于信息检索及自然语言处理等专业大学生，研究生

另外还有一点要注意作者的工作背景

> 作者一直从事利用统计学习方法对文本数据进行各种智能性处理的研究， 包括自然语言处理、信息检索、文本数据挖掘。

- 每个人都有适合自己的理解方式，对同样的内容，会有不同的理解
- 书如数据，学如训练，人即模型。

如果用我这个模型来实现相似度查找，和李老师这本书神似的就是《半导体光电器件》了，只可惜昔时年少，未曾反复研读。

希望在反复研读的过程中，将整个这本书看厚，变薄。这个系列的所有的文档，以及代码，没有特殊说明的情况下"书中"这个描述指代的都是李航老师的《统计学习方法》。其他参考文献中的内容如果引用会给出链接。

在Refs中列出了部分参考文献，有些参考文献对于理解书中的内容是非常有帮助的。关于这些文件的描述和解释会在参考部分对应的Refs/README.md中补充。这个文档中也添加了其他参考文献的一些说明。

方便参考文献下载， 在review02的时候，添加了ref_downloader.sh，可以用来下载书中列举的参考文献，更新过程随着review02的进行逐渐完成。

另外，李航老师的这本书，~~真的很薄（第二版不薄了）~~，但是几乎每句话都会带出很多点，值得反复研读。

书中在目录之后有个符号表，解释了符号定义，所以如果有不理解的符号可以过来查表；在本书后面有个索引，可以通过索引查找对应的符号表示的含义在书中出现的位置。在本Repo中，维护了一个glossary_index.md，目的是给对应的符号补充一些说明，以及直接标注符号对应的页码，进度随review更新。

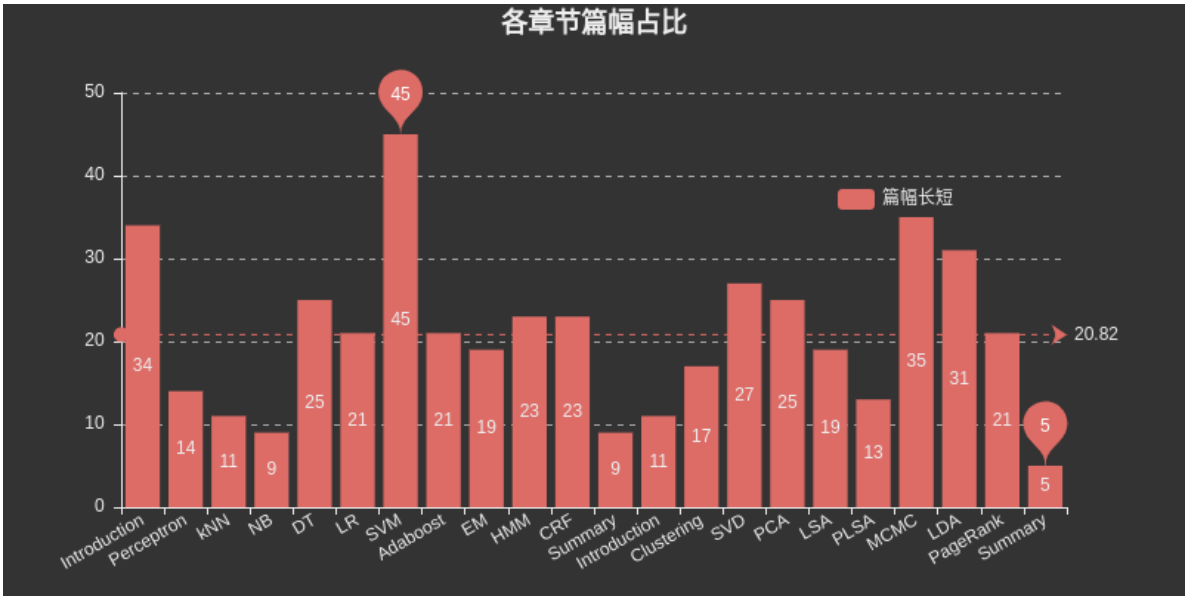
每个算法，示例结束之后会有一个■，表示这个算法或者例子到此结束。这个叫证明结束符，看文献多了就知道了。

关于对数底数

读书的时候经常会有关于对数底数是多少的问题，有些比较重要的，书中都有强调。 有些没有强调的，通过上下文可以理解。另外，因为有换底公式，所以，底具体是什么关系不是太大，差异在于一个常系数。但是选用不同的底会有物理意义和处理问题方面的考虑，关于这个问题的分析，可以看PRML 1.6中关于熵的讨论去体会。

另外关于公式中常系数的问题，如果用迭代求解的方式，有时对公式做一定的简化，可能会改善收敛速度。个中细节可以实践中慢慢体会。

关于篇幅



这里插入个图表，列举了各个章节所占篇幅，其中SVM是监督学习里面占用篇幅最大的，MCMC是无监督里面篇幅占用最大的，另外DT，HMM，CRF，SVD，PCA，LDA，PageRank也占了相对较大的篇幅。

章节之间彼此又有联系，比如NB和LR，DT和AdaBoost，Perceptron和SVM，HMM和CRF等等，如果有大章节遇到困难，可以回顾前面章节的内容，或查看具体章节的参考文献，一般都给出了对这个问题描述更详细的参考文献，可能会解释你卡住的地方。

CH01 统计学习及监督学习概论

Introduction

统计学习方法三要素:

- 模型
- 策略
- 算法

> 第二版对这一章的目录结构重新梳理了，更清晰。

CH02 感知机

Perceptron

- 感知机是二类分类的线性分类模型
- 感知机对应于特征空间中将实例划分为正负两类的分离超平面.

CH03 k近邻法

kNN

- kNN是一种基本的分类与回归方法
- k值的选择, 距离度量及分类决策规则是kNN的三个基本要素.

CH04 朴素贝叶斯法

NB

- 朴素贝叶斯法是基于贝叶斯定理与特征条件独立假设的分类方法.

1. IID →输入输出的联合概率分布

2. $Bayes \rightarrow$后验概率最大的输出

- x的某种组合在先验中没有出现的情况, 会出现概率为0的情况, 对应平滑处理方案

  $P_\lambda(X^{(j)}=a_{jl}|Y=c_k)=\frac{\sum_{i=1}^{N}I(x_i^{(j)}=a_{jl},y_i=c_k)+\lambda}{\sum_{i=1}^{N}I(y_i=c_k)+S_j\lambda}$

  - $\lambda=0$ 对应极大似然估计
  - $\lambda=1$ 对应拉普拉斯平滑
- 朴素贝叶斯法实际上学习到生成数据的机制, 所以属于生成模型.

CH05 决策树

DT

- 决策树是一种基本的分类与回归方法

CH06 逻辑斯谛回归与最大熵模型

LR

- 逻辑斯谛回归是统计学中的经典分类方法
- 最大熵是概率模型学习的一个准则, 将其推广到分类问题得到最大熵模型

关于最大熵的学习，推荐阅读该章节的参考文献[1]，Berger, 1996, 有益于书中例子的理解以及最大熵原理的把握。

那么, **为什么LR和Maxent要放在一章?**

- 都属于对数线性模型
- 都可用于二分类和多分类
- 两种模型的学习方法一般采用极大似然估计, 或正则化的极大似然估计. 可以形式化为无约束最优化问题, 求解方法有IIS, GD, BFGS等
- 在Logistic regression中有如下描述,

  > Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

- 还有这样的描述

  > Logistic regression is a special case of maximum entropy with two labels +1 and −1.

  这个章节的推导中用到了$y\in\mathcal{Y}=\{0,1\}$的性质
- 有时候我们会说，逻辑回归在NLP方面叫做Maxent

CH07 支持向量机

SVM

- 支持向量机是一种二分类模型。
- 基本模型是定义在特征空间上的间隔最大化的线性分类器, 间隔最大使他有别于感知机
- 这一章占了很大篇幅，因为margin这个思想几乎可以串起来整个分类问题。

CH08 提升方法

Boosting

- 提升方法是一种常用的统计学习方法, 应用广泛且有效.

----分割线----

姑且在这里分一下，因为后面HMM和CRF通常会引出概率图模型的介绍，在《机器学习，周志华》里面更是用了一个单独的**概率图模型**章节来包含HMM，MRF，CRF等内容。另外从HMM到CRF本身也有很多相关的点。

在书中第一章有说明监督学习的三种应用：分类，标注和回归。在第十二章中有补充，本书主要考虑前两者的学习方法。据此， 在这里分割也是合适的，前面介绍分类模型， 少部分提到了回归，后面主要介绍标注问题。

CH09 EM算法及其推广

EM

- EM算法是一种迭代算法，用于含有隐变量的概率模型参数**极大似然估计**，或者极大后验概率估计。(这里的极大似然估计和极大后验概率估计是**学习策略**)
- > 如果概率模型的变量都是观测变量，那么给定数据，可以直接用极大似然估计法，或贝叶斯估计法估计模型参数。

  注意书上这个描述如果不理解，参考CH04中朴素贝叶斯法的参数估计部分。
- 这部分代码实现了BMM和GMM，值得看下
- 关于EM，这个章节写的不多，EM是十大算法之一，EM和Hinton关系紧密，Hinton在2018年ICLR上发表了Capsule Network的第二篇文章《Matrix Capsules with EM Routing》
- 在CH22中将EM算法归类于基础机器学习方法，不涉及具体的机器学习模型，可用于无监督学习也可用于监督学习，半监督学习。

CH10 隐马尔可夫模型

HMM

- 隐马尔可夫模型是可用于标注问题的统计学习模型，描述由隐藏的马尔可夫链随机生成观测序列的过程，属于生成模型。
- 隐马尔可夫模型是关于时序的概率模型，描述由一个隐藏的马尔可夫链随机生成不可观测的状态的序列，再由各个状态速记生成一个观测而产生观测的序列的过程。
- 可用于**标注**(Tagging)问题，状态对应标记。
- 三个基本问题：概率计算问题，学习问题，预测问题。

CH11 条件随机场

CRF

- 条件随机场是给定一组输入随机变量条件下另一组输出随机变量的条件概率分布模型，其特点是假设输出随机变量构成**马尔可夫随机场。**
- 概率无向图模型，又称为马尔可夫随机场，是一个可以由无向图表示的**联合概率分布。**
- 三个基本问题：概率计算问题，学习问题，预测问题

CH12 监督学习方法总结

Summary

这章就简单的几页，可以考虑如下阅读套路：

- 和第一章一起看
- 在前面的学习中遇到不清楚的问题的时候，过一遍这个章节。
- 将这一章看厚，从这一章展开到其他十个章节。
- 注意这一章有个图12.2，这里面提到了逻辑斯谛损失函数，这里的y应该是定义在$\mathcal{Y}=\{+1,-1\}$中的，在前面介绍LR的时候y定义在$\mathcal{Y}=\{0,1\}$，这里注意下。

李老师这本书真的是每次刷都会有新的收获。

----分割线----

第二版增加了八个无监督学习方法：聚类，奇异值分解，主成分分析，潜在语义分析，概率潜在语义分析，马尔可夫链蒙特卡罗法，潜在狄利克雷分配，PageRank。

CH13 无监督学习概论

Introduction

- 无监督学习的基本问题：聚类，降维，话题分析和图分析。
- 横向**结构**和纵向**结构**这个问题，从存储的角度来考虑。
- 注意不同任务的策略：类别中心距离最小化，维度转换过程中信息损失的最小化，生成数据概率的最大化。
- 在无监督学习部分经常会提到**数据中的结构**， 是指数据中变量之间的关系。

CH14 聚类方法

Clustering

- 例子14.2很好，建议画出来先自己展开思考下，再往后看
- 聚类可以用于图像压缩

CH15 奇异值分解

- 基本机器学习方法
- 奇异值分解定理保证分解存在
- 奇异值矩阵唯一，U, V不唯一
- 有明确的几何解释

CH16 主成分分析

- 利用正交变换将线性相关变量表示的观测数据转换为少数几个由线性无关变量表示的数据，线性无关的变量称为**主成分**
- 主成分分析之前，需要对给定数据规范化，使得每一个变量均值为0，方差为1。
- 主成分并不对应原始数据的某一个特征， 可以通过因子负荷量来观察主成分与原始特征之间的关系。
- 这部分内容，还没有提到**话题**这个概念，后面章节开始介绍了很多话题分析相关的内容，LSA，PLSA，LDA都是和话题有关，MCMC是在LDA中使用的一个工具。
- 提到了总体主成分和样本主成分，前者是后者的基础。主要体现在总体考虑期望，样本考虑均值。样本主成分具有和总体主成分一样的性质。

CH17 潜在语义分析

- 在sklearn的定义中，LSA就是截断奇异值分解。
- 注意体会LSA和PCA的区别，主要在于是不是去均值。
- 在LSA中，话题向量空间是U，DOC在话题向量空间的表示是SV^{T}。但是在sklaern中，xtransformed是$U\Sigma$

CH18 概率潜在语义分析

CH19 马尔可夫链蒙特卡罗法

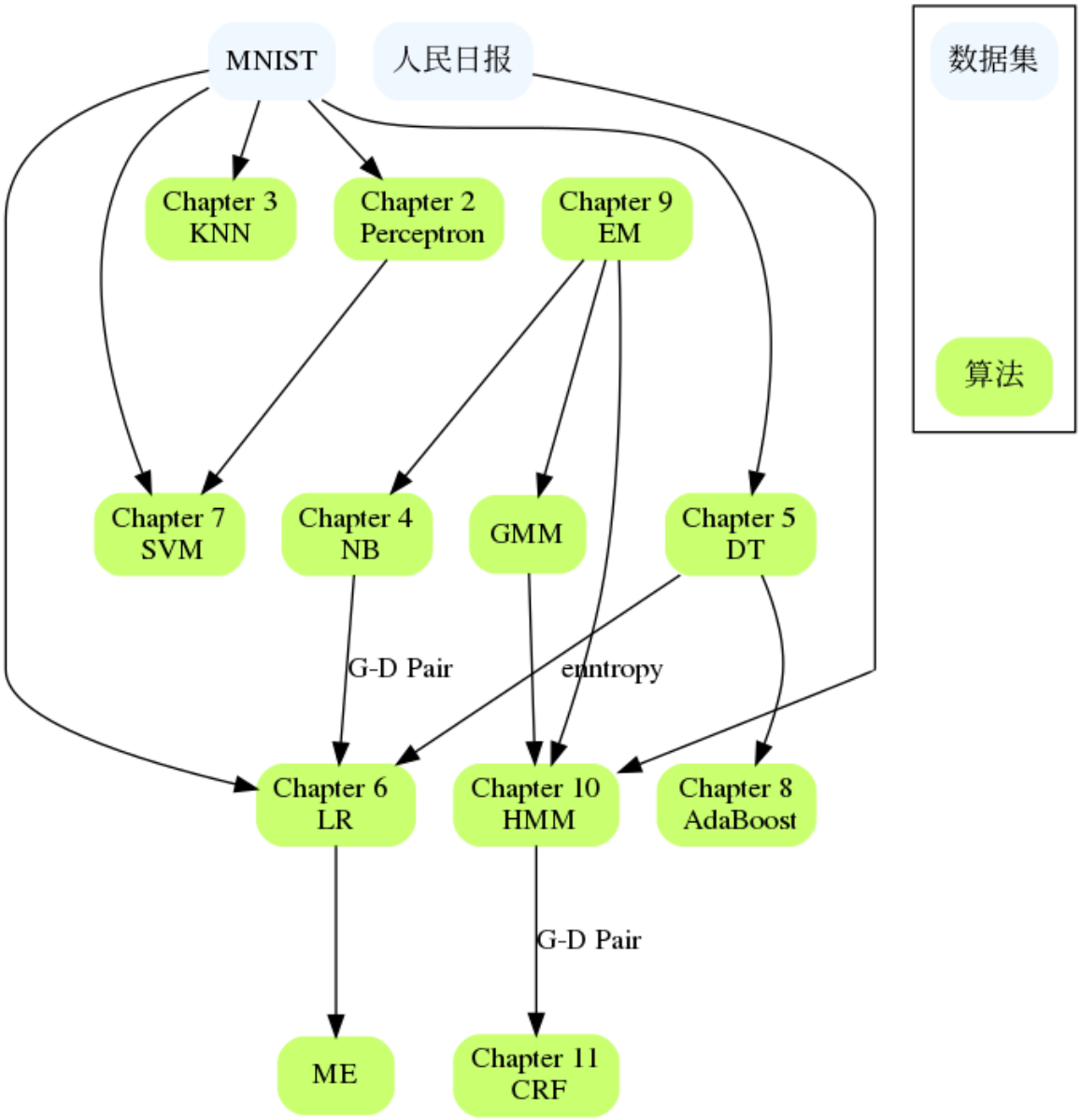
CH20 潜在狄利克雷分配

CH21 PageRank算法

CH22 无监督学习方法总结

后记

整个这本书里面各章节也不是完全独立的，这部分希望整理章节之间的联系以及适用的数据集。算法到底实现到什么程度，能跑什么数据集也是一方面。



参考

CH01 统计学习及监督学习概论

CH01 统计学习及监督学习概论

参考

前言

章节目录

导读

- ~~直接看目录结构，会感觉有点乱，就层级结构来讲感觉并不整齐。~~第二版重新梳理了这部分目录结构，舒服多了，尤其之前的分类，回归与标注因为出现了1.10造成目录中这部分不对齐，非常不爽。结果，第二版改了。赞
- 本章最后的三个部分，这三个问题可以对比着看，如果暂时没有概念，略过也可以，回头对各个算法有了感觉回头再看这里。
  这三部分怎么对比，三部分都有个图来说明，仔细看下差异，本文后面会对此展开。
- 关于损失函数，风险函数与目标函数注意体会差异
- 后面插点从深度学习角度拿到的点
    - 关于机器学习三要素, 复旦大学邱锡鹏教授也有解读[1]: 模型, 学习准则, 优化算法. 这个定义比较接近代码. 以Tensorflow为例. 通常会定义一个网络(模型), 定义Loss(学习准则), 定义优化算法(Optimizer), 然后开Session, 不停的把数据带入用Opitmizer去最小化Loss.
    - Losses, Metrics, 在Keras里面划分了两个模块, 解释是Losses是BP过程用到的, 而Metrics实际和损失函数类似, 用来评价模型的性能, 但是不参与反向传播. 从源码也能看到, Metrics里面import了很多Loss算法
- 书中例子1.1可以参考PRML中对应的表述， 更详细些。
- 在监督学习中输入和输出对称为**样本**，在无监督学习中输入是**样本**。

- 注意在介绍输入空间，输出空间等概念的时候，以及这一章的很多部分都会有个帽子，`监督学习中`， 书中也明确了`本书主要讨论监督学习的问题`，最后的概要总结部分对监督学习有这样的描述：`监督学习可以概括如下：从给定有限的训练数据出发，假设数据是独立同分布的，而且假设模型属于某个假设空间，应用某一评价准则，从假设空间中选取一个最优的模型，使它对已给的训练数据以及未知测试数据在给定评价标准意义下有最准确的预测。`，理解下这里的假设。
- 在贝叶斯学习部分，提到`将模型、为观测要素及其参数用变量表示，使用模型的先验分布是贝叶斯学习的特点。`注意这里面先验是**模型**的先验分布。
- 在泛化误差部分，用了$\hat{f}$表示最优估计，这个有时候也会用f^*表示意思差不多。有时候表示向量又要表示估计值，用*可能好看一点，比如$\vec{x}^*$，但是通常没本书都有自己的符号体系，向量可以通过字体表示，具体可以从书中的符号表部分了解。关于这一点，在第二版第一章就有所体现，监督和无监督学习中，模型用hat表示，在强化学习中，最优解用*表示。
- 提一下参考文献，几个大部头都在，ESL，PRML，DL，PGM，西瓜书，还有Sutton的强化学习，不过这本书2018年出了第二版，感兴趣的话可以看新版。

实现统计学习方法的步骤

统计学习方法三要素：模型，策略，算法

> 1. 得到一个有限的训练数据集合
> 2. 确定包含所有可能的模型的**假设空间**，即学习模型的集合
> 3. 确定模型选择的准则，即学习的**策略**
> 4. 实现求解最优模型的算法，即学习的**算法**
> 5. 通过学习方法选择最优的模型
> 6. 利用学习的最优模型对新数据进行预测或分析

统计学习分类

基本分类

这部分内容新增了无监督学习和强化学习。值得注意的一个点，之前因为只写了监督学习，样本表示(x, y)对，在无监督学习里面，样本就是x。

监督学习

无监督学习

强化学习

按模型分类

概率模型与非概率模型

在监督学习中，概率模型是生成模型，非概率模型是判别模型。

按算法分类

在线学习和批量学习，在线学习通常比批量学习更难。

按技巧分类

贝叶斯学习

核方法

统计学习方法三要素

模型

模型是什么?

在监督学习过程中，模型就是所要学习的**条件概率分布**或者**决策函数**。

注意书中的这部分描述，整理了一下到表格里：

	假设空间$\mathcal{F}$	**输入空间$\mathcal{X}$**	**输出空间$\mathcal{Y}$**	**参数空间**
决策函数	$\mathcal{F}=\{f_\theta \mid Y=f_\theta(x), \theta \in \mathbf{R}^n\}$	变量	变量	$\mathbf{R}^n$
条件概率分布	$\mathcal{F}=\{P \mid P_\theta(Y\|X), \theta \in \mathbf{R}^n\}$	随机变量	随机变量	$\mathbf{R}^n$

书中描述的时候，有提到**条件概率分布族**，这个留一下，后面CH06有提到确认逻辑斯谛分布属于指数分布族。

策略

损失函数与风险函数

> **损失函数**度量模型**一次预测**的好坏，**风险函数**度量**平均意义**下模型预测的好坏。

1. 损失函数(loss function)或代价函数(cost function)
   损失函数定义为给定输入X的**预测值**$f(X)$和**真实值**Y之间的**非负实值**函数，记作$L(Y, f(X))$
2. 风险函数(risk function)或期望损失(expected loss)
   这个和模型的泛化误差的形式是一样的
   $R_{exp}(f)=E_p[L(Y, f(X))]=\int_{\mathcal{X}\times\mathcal{Y}} L(y, f(x))P(x, y)\,\mathrm{d}x\mathrm{d}y$
   模型$f(X)$关于联合分布$P(X, Y)$的**平均意义下的**损失(**期望**损失)，但是因为$P(X, Y)$是未知的，所以前面的用词是**期望**，以及**平均意义下的。**

   这个表示其实就是损失的均值，反映了对整个数据的预测效果的好坏，P(x,y)转换成\frac {\nu(X=x, Y=y)} {N}$更容易直观理解, 可以参考CH09，6.2.2节的部分描述来理解，但是真实的数据N是无穷的。
3. **经验风险**(empirical risk)或**经验损失**(empirical loss)
   $R_{emp}(f)=\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$
   模型f关于**训练样本集**的平均损失
   根据大数定律，当样本容量N趋于无穷大时，经验风险趋于期望风险
4. **结构风险**(structural risk)
   $R_{srm}(f)=\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))+\lambda J(f)$
   $J(f)$为模型复杂度, $\lambda \geqslant 0$是系数，用以权衡经验风险和模型复杂度。

常用损失函数

损失函数数值越小，模型就越好

1. 0-1损失
   $$L(Y, f(X))=\begin{cases}1, Y \neq f(X)\\ 0, Y=f(X)\end{cases}$$
2. 平方损失
   $L(Y, f(X))=(Y-f(X))^2$
3. 绝对损失
   $L(Y, f(X))=|Y-f(X)|$
4. 对数损失
   这里$P(Y|X) \leqslant 1$，对应的对数是负值，所以对数损失中包含一个负号，为什么不是绝对值？因为肯定是负的。
   $L(Y, P(Y|X))=-\log P(Y|X)$

ERM与SRM

经验风险最小化(ERM)与结构风险最小化(SRM)

1. **极大似然估计**是经验风险最小化的一个例子
   当模型是条件概率分布，损失函数是对数损失函数时，经验风险最小化等价于极大似然估计
2. **贝叶斯估计**中的**最大后验概率估计**是结构风险最小化的一个例子
   当模型是条件概率分布，损失函数是对数损失函数，**模型复杂度由模型的先验概率表示**时，结构风险最小化等价于最大后验概率估计

算法

这章里面简单提了一下，具体可以参考CH12表格中关于学习算法的描述。

模型评估与模型选择

训练误差和测试误差是模型关于数据集的平均损失。

提到一句，`统计学习方法具体采用的损失函数未必是评估时使用的损失函数`，这句理解下。参考下在数据科学比赛中给出的评分标准，与实际学习采用的损失函数之间的关系。

过拟合与模型选择

这部分讲到了最小二乘法，给了PRML中的一个例子。

这个问题中**训练数据**为$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$

模型为

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

经验风险最小化策略下

$$L(w)=\frac{1}{2}\sum_{i=1}^{N}(f(x_i,w)-y_i)^2$$

将模型和训练数据带入到上式得到

$$L(w)=\frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=0}^{M}w_jx_i^j-y_i\right)^2=\frac{1}{2}\sum_{i=1}^{N}(w\cdot x_i-y_i)^2$$

这个问题要求$w=(w_0^*,w_1^*,\cdots,w_M^*)$

对w求偏导令其为零，得到一系列方程，求解可以用梯度下降或者矩阵分解。

求解线性方程组$Ax=b$，可以表示为$x=A/b$，问题展开之后可以涉及到**矩阵分解**。

TODO: 这个例子展开一下

正则化与交叉验证

正则化

模型选择的典型方法是正则化

交叉验证

另一种常用的模型选择方法是交叉验证

- 简单
- S折(K折, K-Fold)[2]
- 留一法

关于交叉验证，这里补充一点。

数据集的划分这个问题，书中有提到数据充足的情况下，将数据划分为三个部分，训练集，验证集和测试集。看到这里，不知道大家会不会有一样的问题：**验证集和测试集有什么区别?**

注意这里，在算法学习的过程中，测试集可能是固定的，但是验证集和训练集可能是变化的。比如K折交叉验证的情况下，分成K折之后，其中的K-1折作为训练集，1折作为验证集，这样针对每一个模型操作K次，计算平均测试误差，最后选择平均测试误差最小的模型。这个过程中用来验证模型效果的那一折数据就是**验证集**。交叉**验证**，就是这样一个使用**验证**集测试模型好坏的过程。他允许我们在模型选择的过程中，使用一部分数据（验证集）“偷窥”一下模型的效果。

泛化能力

- 现实中采用最多的方法是通过测试误差来评价学习方法的泛化能力
- 统计学习理论试图从理论上对学习方法的泛化能力进行分析
- 学习方法的泛化能力往往是通过研究泛化误差的**概率上界**进行的, 简称为泛化误差上界(generalization error bound)

  这本书里面讨论的不多，在CH08里面有讨论提升方法的误差分析, 提到$AdaBoost$不需要知道下界γ。在CH02中讨论算法的收敛性的时候有提到误分类次数的上界.

注意泛化误差的定义，书中有说**事实上，泛化误差就是所学习到的模型的期望风险**

生成模型与判别模型

监督学习方法可分为**生成方法**(generative approach)与**判别方法**(discriminative approach)

生成方法

generative approach

- 可以还原出**联合概率分布**$P(X, Y)$
- 收敛速度快, 当样本容量增加时, 学到的模型可以更快收敛到真实模型
- 当存在隐变量时仍可以用

判别方法

discriminative approach

- 直接学习**条件概率**$P(Y|X)$或者**决策函数**$f(X)$
- 直接面对预测, 往往学习准确率更高
- 可以对数据进行各种程度的抽象，定义特征并使用特征, 可以简化学习问题

分类问题、标注问题、回归问题

Classification, Tagging, Regression

- 图1.4和图1.5除了分类系统和标注系统的差异外，没看到其他差异，但实际上这两幅图中对应的输入数据有差异，序列数据的$x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$对应了
- 图1.5和图1.6，回归问题的产出为$Y = \hat{f}(X)$

参考

参考文献都是大部头，ESL，PRML在列

1.
2.

t top

CH02 感知机

前言

章节目录

1. 感知机模型
2. 感知机学习策略
    1. 数据集的线性可分性
    2. 感知机学习策略
    3. 感知机学习算法
3. 感知机学习算法
    1. 感知机学习算法的原始形式
    2. 算法的收敛性
    3. 感知机学习算法的对偶形式

导读

感知机是二类分类的**线性分类模型**。

- 损失函数$L(w,b)$的经验风险最小化
- 本章中涉及到向量内积，有超平面的概念，也有线性可分数据集的说明，在策略部分有说明损关于失函数的选择的考虑，可以和CH07一起看。另外， 感知机和SVM的更多联系源自margin的思想， 实际上在本章的介绍中并没有体现margin的思想，参考文献中有给出对应的文献。
- 本章涉及的两个例子，思考一下为什么$\eta = 1$，进而思考一下参数空间，这两个例子设计了相应的测试案例实现， 在后面的内容中也有展示。
- 在**收敛性证明**那部分提到了偏置合并到权重向量的技巧，合并后的权重向量叫做扩充权重向量，这点在LR和SVM中都有应用，但是这种技巧在书中的表示方式是不一样的，采用的不是统一的符号体系，或者说不是统一的。本书三个章节讨论过算法的收敛性，感知机， AdaBoost，EM算法。
- 第一次涉及Gram Matrix $G = [x_i \cdot x_j]_{N \times N}$
- 感知机的激活函数是符号函数。
- 感知机是神经网络和支持向量机的基础。
- 当我们讨论**决策边界**的时候, 实际上是在考虑算法的**几何解释**。
- 关于感知机为什么不能处理异或问题, 可以借助下图理解。

上面紫色和橙色为两类点, 线性的分割超平面应该要垂直于那些红粉和紫色的线.

- 提出感知机算法的大参考文献是本文第一篇文献， 这个文章发表在Psycological Review上。不过这个文章，真的不咋好看。
- 书中有提到函数间隔，几何间隔，这里间隔就是margin
- 在CH07中有说明， `分离超平面将特征空间划分为两个部分，一部分是正类， 一部分是负类。法向量指向的一侧为正类，另一侧为负类。`
- 感知机损失函数$L = \max(0, -y_i(w \cdot x_i + b))$，这个在CH07中将hinge loss的时候有说明。

三要素

模型

输入空间：$\mathcal{X} \subseteq \mathbf{R^n}$

输出空间：$\mathcal{Y} = +1, -1$

> 决策函数：$f(x) = sign(w \cdot x + b)$

策略

确定学习策略就是定义**(经验)**损失函数并将损失函数最小化。

注意这里提到了**经验**，所以学习是base在**训练数据集**上的操作

损失函数选择

> 损失函数的一个自然选择是误分类点的总数，但是，这样的损失函数**不是参数**w, b**的连续可导函数，不易优化**
>
> 损失函数的另一个选择是误分类点到超平面S的总距离，这是感知机所采用的

感知机学习的经验风险函数(损失函数)

$$L(w, b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

其中M是误分类点的集合

给定训练数据集T，损失函数$L(w, b)$是w和b的连续可导函数

算法

原始形式

> 输入：$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
> $x_i \in \mathcal{X} = \mathbf{R^n}, y_i \in \mathcal{Y} = \{-1, +1\}, i = 1, 2, \dots, N;\ \ 0 < \eta \leqslant 1$
>
> 输出：$w, b; f(x) = sign(w \cdot x + b)$
>
> 1. 选取初值w_0, b_0
> 2. 训练集中选取数据(x_i, y_i)
> 3. 如果$y_i(w \cdot x_i + b) \leqslant 0$
>
> $$w \leftarrow w + \eta y_i x_i$$
> $$b \leftarrow b + \eta y_i$$
>
> 4. 转至(2)，直至训练集中没有误分类点

注意这个原始形式中的迭代公式，可以对x补1，将w和b合并在一起，合在一起的这个叫做扩充权重向量，书上有提到。

对偶形式

对偶形式的基本思想是将w和b表示为实例x_i和标记y_i的线性组合的形式，通过求解其系数而求得w和b。

> 输入：$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
> $x_i \in \mathcal{X} = \mathbf{R^n}, \mathbf{y_i} \in \mathcal{Y} = \{-1, +1\}, i = 1, 2, \dots, \mathcal{N}; 0 < \eta \leqslant 1$
>
> 输出：

> $$\alpha, b; f(x) = sign\left(\sum_{j=1}^{N} \alpha_j y_j x_j \cdot x + b\right)$$
> $$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_N)^T$$
>
> 1. $\alpha \leftarrow 0, b \leftarrow 0$
> 2. 训练集中选取数据(x_i, y_i)
> 3. 如果$y_i\left(\sum_{j=1}^{N} \alpha_j y_j x_j \cdot x + b\right) \leqslant 0$
>
> $$\alpha_i \leftarrow \alpha_i + \eta$$
> $$b \leftarrow b + \eta y_i$$
>
> 4. 转至(2)，直至训练集中没有误分类点

Gram matrix

对偶形式中，训练实例仅以内积的形式出现。

为了方便可预先将训练集中的实例间的内积计算出来并以矩阵的形式存储，这个矩阵就是所谓的Gram矩阵

$$G = [x_i \cdot x_j]_{N \times N}$$

例子

例2.1

这个例子里面$\eta = 1$

> 感知机学习算法由于采用不同的初值或选取不同的误分类点，解可以不同。

另外，在这个例子之后，证明算法收敛性的部分，有一段**为了便于叙述与推导**的描述，提到了将偏置并入权重向量的方法，这个在涉及到内积计算的时候可能都可以用到，可以扩展阅读CH06，CH07部分的内容描述。

例2.2

这个例子也简单，注意两点

1. $\eta = 1$
2. $\alpha_i \leftarrow \alpha_i + 1, b \leftarrow b + y_i$

以上：

1. 为什么η选了1，这样得到的值数量级是1
2. 这个表达式中用到了上面的$\eta = 1$这个结果，已经做了简化

所以，这里可以体会下，调整学习率η的作用。学习率决定了参数空间。

Logic_01

经常被举例子的**异或**问题[2]，用感知机不能实现，因为对应的数据非线性可分。但是可以用感知机实现其他逻辑运算，也就是提供对应的逻辑运算的数据，然后学习模型。

这个例子的数据是二元的，其中NOT运算只针对输入向量的第一个维度

Logic_02

这个例子的数据是三元的

MNIST_01

这个选择两类数据进行区分，不同的选择应该得到的结果会有一定差异，数据不上传了，在sklearn里面有相应的数据，直接引用了，注意测试案例里面用的是01，相对来讲好区分一些。

问题

损失函数

知乎上有个问题

> 感知机中的损失函数中的分母为什么可以不考虑？
> 有些人解释是正数，不影响，但是分母中含有 w，而其也是未知数，在考虑损失函数的最值时候会不影响么？想不通

这个对应了书中P_{27}中`不考虑1/||w||，就得到感知机学习的损失函数`

题中问考虑损失函数最值的时候，不会有影响么？

1. 感知机处理线性可分数据集，二分类，$\mathcal{Y}=\{+1,-1\}$，所以涉及到的乘以 y_i 的操作实际贡献的是符号；
2. 损失函数 $L(w,b)=-\sum_{x_i\in M}y_i(w\cdot x_i+b)$,其中 M 是错分的点集合，线性可分的数据集肯定能找到超平面 S， 所以这个损失函数最值是0。
3. 如果正确分类， $y_i(w\cdot x_i+b)=|w\cdot x_i+b|$ ，错误分类的话，为了保证正数就加个负号，这就是损失函数里面那个负号，这个就是函数间隔；
4. $\frac{1}{||w||}$ 用来归一化超平面法向量，得到几何间隔，也就是点到超平面的距离， 函数间隔和几何间隔的差异在于同一个超平面 (w,b) 参数等比例放大成 (kw,kb) 之后，虽然表示的同一个超平面，但是点到超平面的函数间隔也放大了，但是几何间隔是不变的；
5. 具体算法实现的时候， w要初始化，然后每次迭代针对错分点进行调整，既然要初始化，那如果初始化个 $||w||=1$ 的情况也就不用纠结了，和不考虑 $\frac{1}{||w||}$ 是一样的了；
6. 针对错分点是这么调整的

   $$w\leftarrow w+\eta y_i x_i$$
   $$b\leftarrow b+\eta y_i$$

   前面说了 y_i 就是个符号，那么感知机就可以解释为针对误分类点，通过调整 w,b 使得超平面向该误分类点一侧移动，迭代这个过程最后全分类正确；
7. 感知机的解不唯一，和初值有关系，和误分类点调整顺序也有关系；
8. 这么调整就能找到感知机的解？能，Novikoff还证明了，通过有限次搜索能找到将训练数据完全正确分开的分离超平面。

所以，

如果只考虑损失函数的最值，那没啥影响，线性可分数据集，最后这个损失就是0； 那个分母用来归一化法向量，不归一化也一样用，感知机的解不唯一；说正数不影响的应该考虑的是不影响超平面调整方向吧

参考

1.

[↑ top](#)

CH03 k近邻法

前言

章节目录

1. k近邻算法
2. k近邻模型
    1. 模型
    2. 距离度量
    3. k值选择
    4. 分类决策规则
3. k近邻法的实现: KDTree
    1. 构造KDTree
    2. 搜索KDTree

导读

kNN是一种基本分类与回归方法。

- kNN是机器学习中被分析的最透彻的算法之一。
- 多数表决规则等价于0-1损失函数下的经验风险最小化，支持多分类， 有别于前面的感知机算法
- kNN的k和KDTree的k含义不同，书上这部分有注释说明
- KDTree是一种存储k维空间数据的树结构，KDTree是平衡二叉树
- KNN应用的一个实践问题是如何建立高效的索引。建立空间索引的方法在点云数据处理中也有广泛的应用，KDTree和八叉树在3D点云数据组织中应用比较广
- 书中的KDTree搜索实现的时候针对了一种$k=1$的特殊的情况，实际是**最**近邻搜索
- KDTree的搜索问题分为**k近邻查找**和**范围查找**，一个是已知k，求点集范围，一个是已知范围，求里面有k个点。范围查找问题在维度高的时候复杂度非常高，不太推荐用KDTree做范围查找。
- K近邻问题在杭电ACM里面有收录，HUD4347
- 图像的特征点匹配，数据库查询，图像检索本质上都是同一个问题--相似性检索问题。Facebook开源了一个高效的相似性检索工具Faiss，用于有效的相似性搜索和稠密矢量聚类。
- 这一章有个经典的图，在很多文章和教材上都能看到，就是第一个图3.1。这个图画出了1NN算法在实例空间上的决策面形状。这种类型的图经常被称为在训练集上的的Voronoi图，也叫Thiessen Polygons。可以通过检索Delaunay三角剖分和Voronoi划分进一步了解。
- 在scipy.spatial.KDTree中有KDTree的实现，KDTree在创建马赛克照片的时候可以用到。

最近邻算法

$k=1$的情形，称为最近邻算法。书中后面的分析都是按照最近邻做例子，这样不用判断类别，可以略去一些细节。

k近邻模型

算法

输入: $T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$, $x_i \in \mathcal{X} \subseteq \mathbf{R^n}, y_i \in \mathcal{Y}=\{c_1,c_2,\dots,c_k\}$; 实例特征向量$x$

输出: 实例所属的y

步骤:

1. 根据指定的**距离度量**，在T中查找x的**最近邻的k个点**，覆盖这k个点的x的邻域定义为$N_k(x)$
2. 在$N_k(x)$中应用**分类决策规则**决定x的类别y

$$y = \arg\max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, \ldots, N, j = 1, 2, \ldots, K$$

这里提到了k近邻模型的三要素，如算法描述中黑体标注的部分， 注意这里的三要素和前面说的统计学习方法的三要素不是一个东西。后面讲到隐马尔可夫模型的时候也有三要素。

距离度量

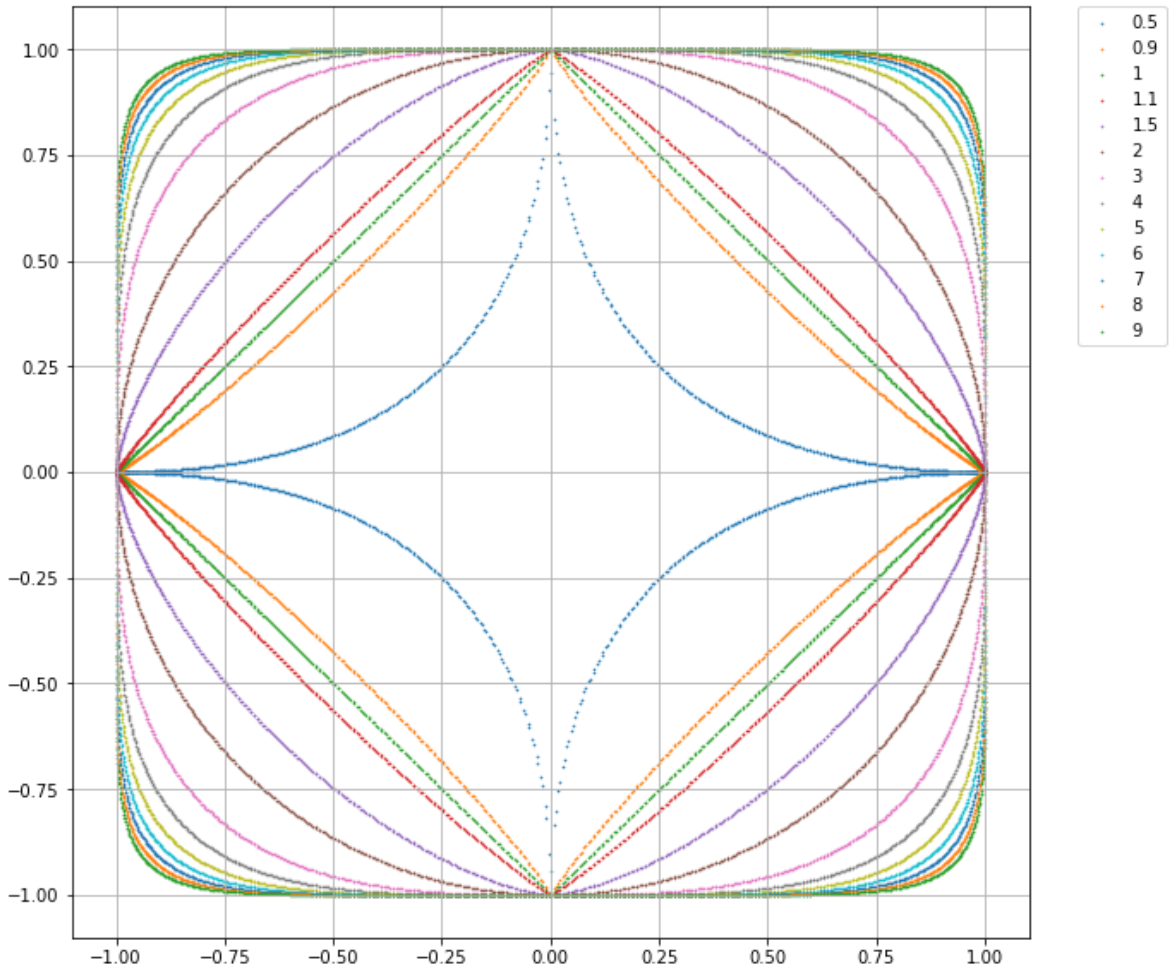
> **特征空间**中的两个实例点的距离是两个实例点相似程度的反映。

书中是如上描述的，这里要注意**距离越近(数值越小)， 相似度越大。**

这里用到了L_p距离, 可以参考Wikipedia上L_p Space词条[2]

1. $p = 1$ 对应 曼哈顿距离
2. $p = 2$ 对应 欧氏距离
3. 任意p 对应 闵可夫斯基距离

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} \left| x_i^{(l)} - x_j^{(l)} \right|^p \right)^{\frac{1}{p}}$$



考虑二维的情况，上图给出了不同的p值情况下与原点距离为1的点的图形。

这个图有几点理解下:

1. 与原点的距离
2. 与原点距离为1的点
3. 前一点换个表达方式，图中的点向量(x_1, x_2)的p范数都为1
4. 图中包含多条曲线，关于p=1并没有对称关系

5. 定义中$p \geqslant 1$，这一组曲线中刚好是凸的

这里要补充一点：

范数是对向量或者矩阵的度量，是一个标量，这个里面两个点之间的L_p距离可以认为是两个点坐标差值的p范数。

参考下例题3.1的测试案例，这个实际上没有用到模型的相关内容。

k值选择

1. 关于k大小对预测结果的影响，书中给的参考文献是ESL，这本书还有个先导书叫ISL。
2. 通过**交叉验证**选取最优k，算是超参数
3. 二分类问题，k选择奇数有助于避免平票

分类决策规则

Majority Voting Rule

误分类率

$\frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i \neq c_i) = 1 - \frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i = c_i)$

如果分类损失函数是0-1损失，误分类率最低即经验风险最小。

关于经验风险，参考书上CH01第一章 (1.11)和(1.16)

实现

kNN在实现的时候，要考虑多维数据的存储，这里会用到树结构。

在Scipy Cookbook里面有个kd树具体的实现[1]可参考

构造KDTree

KDTree的构建是一个递归的过程

注意: KDTree左边的点比父节点小，右边的点比父节点大。

这里面有提到，KDTree搜索时效率未必是最优的，这个和样本分布有关系。随机分布样本**KDTree搜索**(这里应该是**最**近邻搜索)的平均计算复杂度是$O(\log N)$，空间维数K接近训练样本数N时，搜索效率急速下降，几乎$O(N)$

看维度，如果维度比较高，搜索效率很低。当然，在考虑维度的同时也要考虑样本的规模。

考虑个例子

```
[[1, 1],
 [2, 1],
 [3, 1],
 [4, 1],
 [5, 1],
 [6, 1],
 [100, 1],
 [1000, 1]]
```

这个数据，如果找[100, 1]

搜索KDTree

这部分书中的例子是最近邻的搜索例子。

k近邻查找

KNN查找已知查询点p，树当前节点o，近邻数目k

可以用一个优先队列存储最优的k个点，每次比对回溯节点是否比当前最优点更优的时候，就只需用当前最优中距离p最远的节点来对比，而这个工作对于优先队列来说是$O(1)$的[3]

范围查询

给定一个范围，问其中有多少点。比较常见的应用是GIS类应用，使用者附近多大半径内包含多少单车，多少酒店等。

实际上为了实现快速搜索， 在空间数据的存储结构上要有考虑。

例子

例3.1

分析p值对最近邻点的影响，这个有一点要注意关于闵可夫斯基距离的理解：

- 两点坐标差的p范数

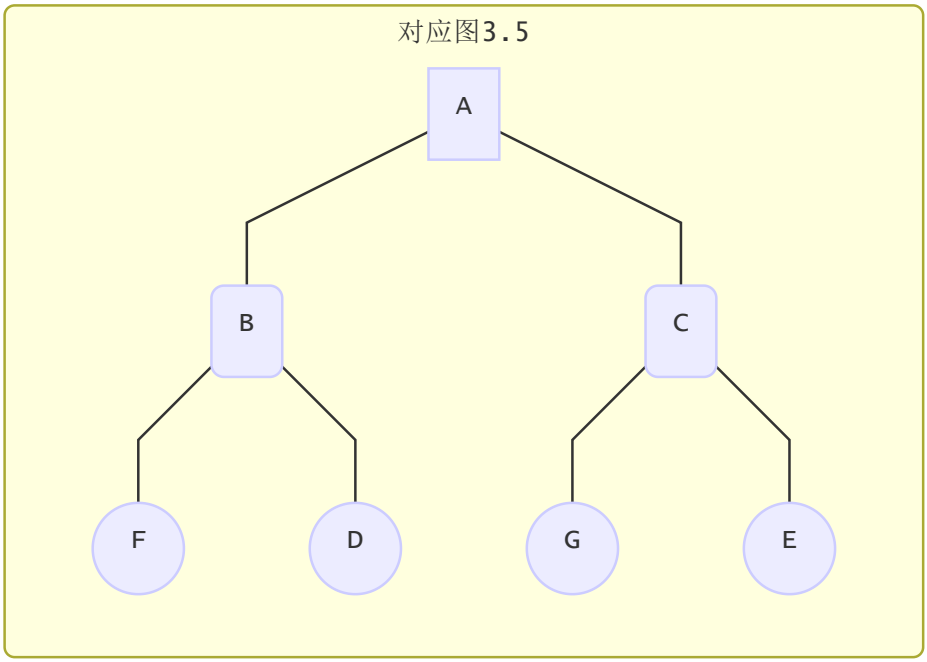
具体看相关测试案例的实现

例3.2

KDTree创建

例3.3

KDTree搜索



这个例子说明了搜索的方法，理解一下书中的图3.5，对应的KDTree如上。

参考

1.
2. ESL
3.
4.
5.

[↑ top](#)

CH04 朴素贝叶斯法

前言

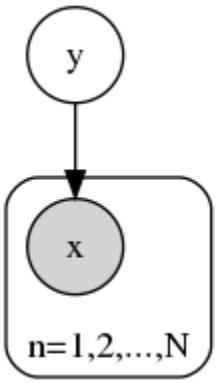
章节目录

1. 朴素贝叶斯法的学习与分类
    1. 基本方法
    2. 后验概率最大化的含义
2. 朴素贝叶斯法的参数估计
    1. 极大似然估计
    2. 学习与分类算法
    3. 贝叶斯估计

导读

- 书中讲解的生成模型不多，生成模型学习联合概率分布，NB是典型的生成模型，与之对应的判别模型，是逻辑回归可以参考CH06
- **0-1损失函数**时的期望风险最小化
- 给定数据集的时候，我们能知道不同类别的分布，这个很好统计的量叫**先验**$P(Y=c_k)$
    - 垃圾邮件识别可以用朴素贝叶斯的方法

- NB是常见的概率图模型之一[4]。最简单的概率图模型，有向图。概率图模型是执行贝叶斯推理的主流工具。



这个图中， y是输出变量，x是输入变量，是观测变量。生成模型直接描述了输出以多大的概率生成输入。在图模型里，**已知观测是灰色的，未知变量是白色的。**

- 第一个参考文献[2]，这里推荐阅读，链接中的手稿在2017年还有更新，开头部分介绍了很多符号的定义，可以配合CH01的内容来理解。讲述了生成模型和判别模型，朴素贝叶斯和逻辑回归
- 另外在讲到CRF的时候，也有对比贝叶斯和逻辑回归，可以在CH11的时候再回顾这里。

朴素贝叶斯法

参数数量

书中有这样一段内容

> 条件概率分布$P(X=x|Y=c_k)$有指数级数量的参数，其实际估计是不可行的。

上面这段有两点：

1. 指数级数量的参数
   这点书后面解释了，$K\prod_{j=1}^{n}S_j$，这个为什么是指数级数量，假设$S_j=S$，表达式变为KS^n，实际上在参考文献[1]中有这个问题的讨论，里面用的是二值函数，可以参考理解
2. 实际估计是不可行的

   估计这么多参数需要更多的样本来刷参数，实际上获取这么多样本是不可行的。

算法推导

朴素贝叶斯法是基于**贝叶斯定理**与**特征条件独立假设**的分类方法。

- 贝叶斯定理
- 特征条件独立假设

条件独立假设

independent and identically distributed (*i.i.d.* or *iid* or **IID**)

求$P(Y|X)$，其中$X\in\{X_1,X_2,\dots,X_n\}$，条件独立假设这里给定$Y$的情况下：

1. 每一个X_i和其他的每个X_k是条件独立的
2. 每一个X_i和其他的每个X_k的子集是条件独立的

条件独立性假设是:

$$
\begin{aligned}
P(X=x|Y=c_k) &= P(X^{(1)},\dots,X^{(n)}|Y=c_k) \\
&= \prod_{j=1}^{n} P(X^{(j)}=x^{(j)}|Y=c_k)
\end{aligned}
$$

上面这个公式可能看起来不是太容易理解独立在哪里，这里引用一下文献[1]中关于贝叶斯算法推导中的一部分

$$
\begin{aligned}
P(X|Y) &= P(X_1,X_2|Y) \\
&= P(X_1|X_2,Y)P(X_2|Y) \\
&= P(X_1|Y)P(X_2|Y)
\end{aligned}
$$

红色部分从上到下基于IID

条件独立假设等于是说用于分类的**特征**在**类确定**的条件下都是**条件独立**的。

参数估计

极大似然估计

> 为了估计状态变量的条件分布，利用贝叶斯法则，有
>
> $$
> \underbrace{P(X|Y)}_{posterior} = \frac{\overbrace{P(Y|X)}^{likelihood}\overbrace{P(X)}^{prior}}{\underbrace{P(Y)}_{evidence}} = \frac{\overbrace{P(Y|X)}^{likelihood}\overbrace{P(X)}^{prior}}{\underbrace{\sum_{x} P(Y|X)P(X)}_{evidence}}
> $$
>
> 其中$P(X|Y)$为给定Y下X的后验概率(Posterior)， $P(Y|X)$称为似然(Likelyhood)，$P(X)$称为先验(Prior)[2]。

- 后验概率最大化的含义

  朴素贝叶斯法将实例分到**后验概率最大的类**中， 这等价于**期望风险最小化。**

- 后验，观察到Y之后，对X的信念

贝叶斯估计

对于x的某个特征的取值没有在先验中出现的情况 ，如果用极大似然估计，这种情况的可能性就是0。
但是出现这种情况的原因通常是因为数据集不能全覆盖样本空间，出现未知的情况处理的策略就是做平滑。
公式(4.10)对应了出现未知样本的情况下，该给出一个什么样的值才合理的方案。

$$
P_\lambda(X^{(j)}=a_{jl}|Y=c_k) = \frac{\sum_{i=1}^{N} I(x_i^j=a_{jl},y_j=c_k)+\lambda}{\sum_{i=1}^{N} I(y_j=c_k)+S_j\lambda}
$$

其中$\lambda \geqslant 0$

当$\lambda = 0$的时候，就是极大似然估计。

当$\lambda = 1$的时候，这个平滑方案叫做Laplace Smoothing。拉普拉斯平滑相当于给未知变量给定了先验概率。

例子

例4.1

这个例子，有两个特征，一个标签，我们看看通过已知的数据表4.1能拿到哪些计算结果

1. 先验Prior，通过统计Y的数据分布可以知道

2. 不同X和Y的组合会产生多少参数，$X^{(1)}$可能的取值集合$A_1 = \{1, 2, 3\}$大小为$S_1 = 3$，$X^{(2)}$可能的取值集合$A_2 = \{S, M, L\}$大小为$S_2 = 3$，$Y \in C = \{1, -1\}$大小为$K = 2$
   参数的数量为$KS_1S_2 = 18$，具体的空间的分布是一个$3 \times 3 \times 2$的三维矩阵
3. 每个特征的增加，本来应该在原来的Y，X的基础上增加S_i倍的维度，但因为做了特征条件独立假设，增加的可能性，是base在给定的标签Y上的，也就是说实际上增加了S_i个取值
4. 这个题的解题过程可以考虑为：Groupby Y；Groupby Y，X1；Groupby Y，X2；对于新样本查表连乘。

例子4.2

扩展

树增强朴素贝叶斯

IID强限制放宽，就是TAN(Tree Augmented Naive Bayes)。可以看做是NB到贝叶斯网的过渡。

贝叶斯网

朴素贝叶斯法中假设输入变量都是条件独立的，如果假设他们之间存在概率依存关系，模型就变成了**贝叶斯网络**。这个书中简单提了一句作为扩展。

在SLAM的后端优化部分，因子图优化可以从概率的角度分析这个问题。从贝叶斯网络的角度，SLAM可以自然的表达成一个动态贝叶斯网络。在SLAM中有观测方程和运动方程，对应了贝叶斯网络中的条件概率关系。

LR与NB

逻辑斯谛回归与朴素贝叶斯的关系

这个在参考文献[1]中有描述，另外在CH06中也有说明如下：

> 逻辑斯谛回归与朴素贝叶斯模型的关系参见文献[4]

这里文献[4]和本章的参考文献[1]同源，都是Mitchell TM的Machine Learning，现在依然是第一版，第二版的更新章节Tom有挂在网上[3]

这里插一句Mitchell教授的介绍，CMU计算机学院院长，1997年创建自动化学习和探索中心，该中心是全球的高校中首个机器学习系，也是首个开设机器学习博士课程的机构。1997-2016，Mitchell是该中心负责人。

这些都还OK，主要是Mitchell教授加入了松鼠AI。

参考

1.
2.
3.
4.

↑ top

CH05 决策树

前言

章节目录

导读

- 决策树是一种基本的分类与回归方法. 在书中CART算法之前的章节说的都是分类树，ID3和C4.5都只能处理分类问题，从CART(Classification and Regression Tree)开始有回归树，统称为决策树
- 以上在章节目录部分添加了一部分标记，把这个章节按照模型，策略与算法进行划分，进一步重新整理了结构，希望可以帮助理清章节内容之间的关系
- 在CH12中有提到，决策树学习的损失函数是**对数似然损失**，关于决策树的剪枝，最重要的在于这个损失函数的理解。
- 这个章节的主题是决策树，内容内涵和外延都很广，这个章节推荐阅读图灵社区的一个访谈[2]，了解一下李航老师的故事，也可以对本章的最后三个参考文献[1][3][4]有进一步的了解.
- 引文中关于CART的介绍，是一本368页的书，2017年10月有了Kindle版本，书的共同作者Friedman JH也是另一本神书ESL[参考文献7]的共同作者。
- CART虽然在本书中排在ID3和C4.5后面，但是发表的时间顺序为CART->ID3->C4.5，了解决策树历史可以参考Loh的报告[5]
- 熵, 基尼指数衡量的都是集合的不确定性, 应用在推荐的场景, 不确定性越大, 覆盖率可能就越大.
- 书中有提到，`分类问题中， 决策树表示基于特征对实例进行分类的过程。它可以认为是if-then规则的集合， 也可以认为是定义在特征空间上与类空间的条件概率分布。`书中第一小节对这个问题做了解释。
- **剪枝**是模型压缩领域中的经典技术。剪枝可以降低模型复杂度，防止过拟合，提升模型泛化性能。LeCun等在1990年提出OBD[6]方法对神经网络进行剪枝。

概念

熵

$$H(p) = H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

熵只与X的分布有关，与X取值无关，这句注意理解

定义$0\log 0 = 0$，熵是非负的。

条件熵

随机变量(X, Y)的联合概率分布为

$P(X = x_i, Y = y_j) = p_{ij}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$

条件熵$H(Y|X)$表示在已知随机变量X的条件下随机变量Y的不确定性

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

其中$p_i = P(X = x_i), i = 1, 2, \ldots, n$

经验熵， 经验条件熵

> 当熵和条件熵中的概率由数据估计(特别是极大似然估计)得到时，所对应的熵与条件熵分别称为经验熵和经验条件熵

就是从已知的数据计算得到的结果。

信息增益

特征A对训练数据集D的信息增益$g(D|A)$，定义为集合D的经验熵$H(D)$与特征A给定的条件下D的经验条件熵$H(D|A)$之差

$$g(D, A) = H(D) - H(D|A)$$

熵与条件熵的差称为互信息.

决策树中的信息增益等价于训练数据集中的类与特征的互信息。

考虑ID这种特征， 本身是唯一的。按照ID做划分， 得到的经验条件熵为0, 会得到最大的信息增益。所以， 按照信息增益的准则来选择特征， 可能会倾向于取值比较多的特征。

信息增益比

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$
$$H_A(D) = -\sum_{i=1}^{n} \frac{D_i}{D} log_2 \frac{D_i}{D}$$

算法

这部分内容，原始的5.1数据中最后的标签也是是和否，表示树模型的时候，叶结点不是很明显，所以简单改了下数据标签。对应同样的树结构，输出的结果如下

```
# data_5-1.txt
{'有自己的房子': {'否': {'有工作': {'否': {'否': None}, '是': {'是': None}}}, '是': {'是': None}}}
# mdata_5-1.txt
{'有自己的房子': {'否': {'有工作': {'否': {'拒绝': None}, '是': {'批准': None}}}, '是': {'批准': None}}}
```

先导

算法5.1 信息增益

> 输入：训练数据集D和特征A
>
> 输出：特征A对训练数据集D的信息增益$g(D,A)$
>
> 1. 数据集D的经验熵$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$
> 2. 特征A对数据集D的经验条件熵$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$
> 3. 信息增益$g(D,A)=H(D)-H(D|A)$

算法5.2 ID3算法

> 输入：训练数据集D, 特征集A，阈值ϵ
> 输出：决策树T
>
> 1. 如果D属于同一类C_k，T为单节点树，类C_k作为该节点的类标记，返回T
> 2. 如果A是空集，置T为单节点树，实例数最多的类作为该节点类标记，返回T
> 3. 计算g, 选择信息增益最大的特征A_g
> 4. 如果A_g的信息增益小于ϵ，T为单节点树，D中实例数最大的类C_k作为类标记，返回T
> 5. A_g划分若干非空子集D_i，
> 6. D_i训练集，$A-A_g$为特征集，递归调用前面步骤，得到T_i，返回T_i

算法5.3 C4.5生成

> 输入：训练数据集D, 特征集A，阈值ϵ
> 输出：决策树T
>
> 1. 如果D属于同一类C_k，T为单节点树，类C_k作为该节点的类标记，返回T
> 2. 如果A是空集, 置T为单节点树，实例数最多的作为该节点类标记，返回T
> 3. 计算g, 选择**信息增益比**最大的特征A_g
> 4. 如果A_g的**信息增益比**小于ϵ，T为单节点树，D中实例数最大的类C_k作为类标记，返回T
> 5. A_g划分若干非空子集D_i，
> 6. D_i训练集，$A-A_g$为特征集，递归调用前面步骤，得到T_i，返回T_i
>
> ID3和C4.5在生成上，差异只在准则的差异。

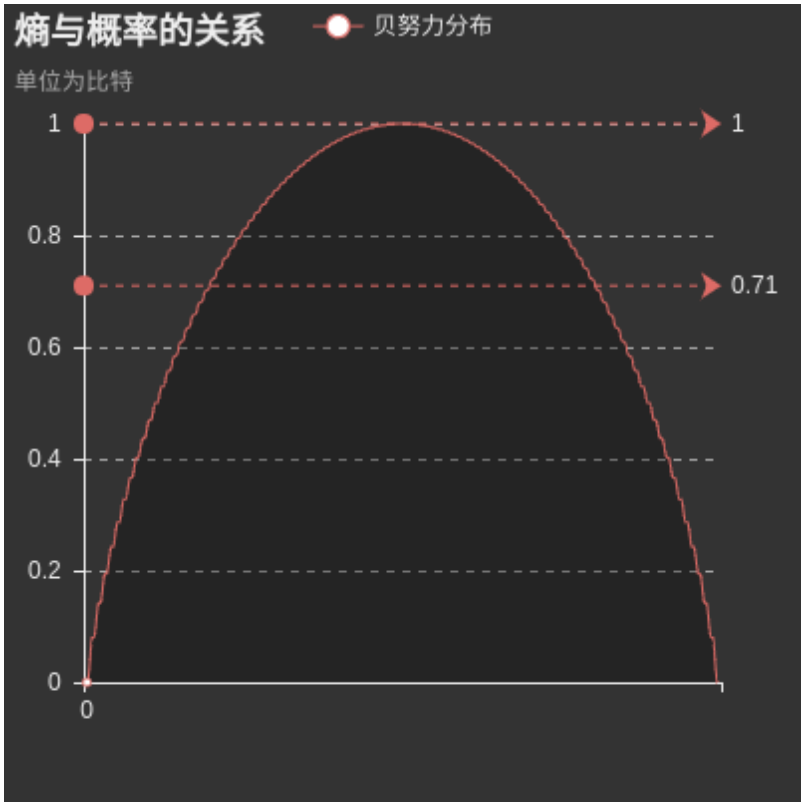
算法5.4 树的剪枝

决策树损失函数摘录如下：

> 树T的叶结点个数为$|T|$，t是树T的叶结点，该结点有N_t个样本点，其中k类的样本点有N_{tk}个，$H_t(T)$为叶结点t上的经验熵，$\alpha\geqslant 0$为参数，决策树学习的损失函数可以定义为
>
> $$C_\alpha(T)=\sum_{i=1}^{|T|}N_tH_t(T)+\alpha|T|$$
>
> 其中
>
> $$H_t(T)=-\sum_k\frac{N_{tk}}{N_t}\log\frac{N_{tk}}{N_t}$$
>
> $$C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}\log\frac{N_{tk}}{N_t}$$
>
> 这时有
>
> $$C_\alpha(T)=C(T)+\alpha|T|$$

> 其中$C(T)$表示模型对训练数据的误差，$|T|$表示模型复杂度，参数$\alpha \geqslant 0$控制两者之间的影响。

上面这组公式中，注意红色部分，下面插入一个图



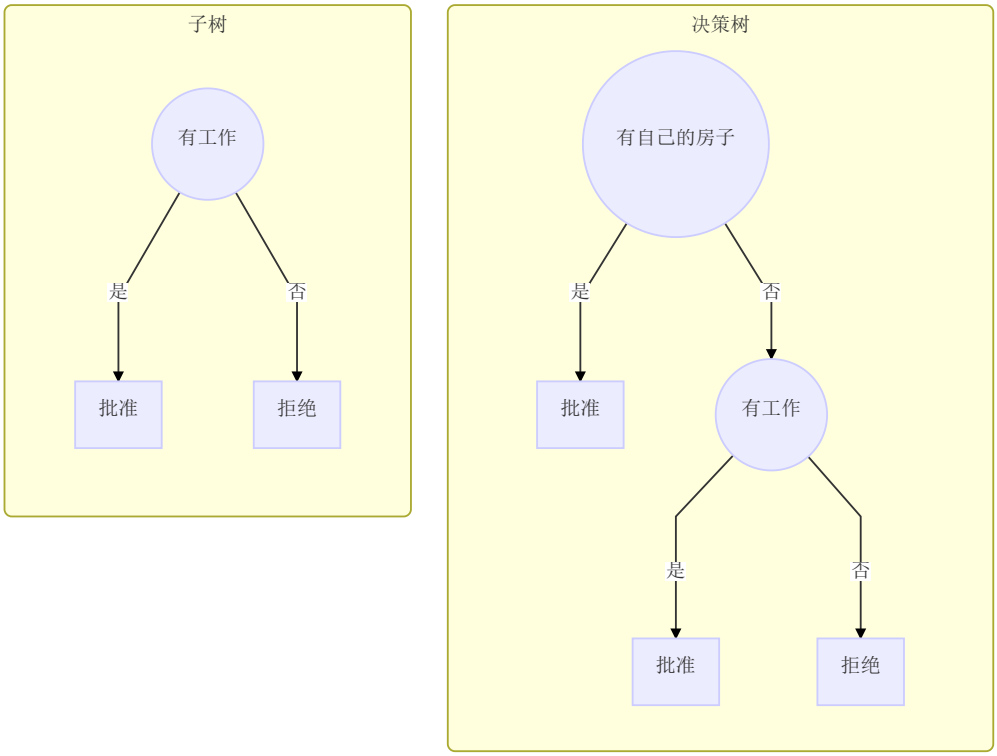
这里面没有直接对$H_t(T)$求和，系数N_t使得$C(T)$和$|T|$的大小可比拟。这个地方再理解下。

> 输入：生成算法生成的整个树T，参数α
>
> 输出：修剪后的子树T_α
>
> 1. 计算每个结点的经验熵
> 2. 递归的从树的叶结点向上回缩
>
> 假设一组叶结点回缩到其父结点之前与之后的整体树分别是T_B和T_A，其对应的损失函数分别是$C_\alpha(T_A)$和$C_\alpha(T_B)$，如果$C_\alpha(T_A) \leqslant C_\alpha(T_B)$则进行剪枝，即将父结点变为新的叶结点
>
> 3. 返回2，直至不能继续为止，得到损失函数最小的子树T_α

```
{'有自己的房子': {'否': {'有工作': {'否': {'拒绝': None}, '是': {'批准': None}}}, '是': {'批准': None}}}
```

重新看一下这个建好的树，同样是字典的key，但是是有区别的。

- 有自己的房子和有工作，是特征的索引
- 是，否，是特征的取值



这个图里面，每一个划分都能计算经验熵

```
ID  年龄  有工作  有自己的房子  信贷情况    类别
1   青年  否    否    一般   拒绝
2   青年  否    否    好    拒绝
15  老年  否    否    一般   拒绝
5   青年  否    否    一般   拒绝
6   中年  否    否    一般   拒绝
7   中年  否    否    好    拒绝

13  老年  是    否    好    批准
14  老年  是    否    非常好  批准
3   青年  是    否    好    批准

4   青年  是    是    一般   批准
8   中年  是    是    好    批准
9   中年  否    是    非常好  批准
10  中年  否    是    非常好  批准
11  老年  否    是    非常好  批准
12  老年  否    是    好    批准
```

这里先不考虑书中提到的剪枝方案，从上面划分的过程，思考如何做前剪枝，显然可以通过控制最后的叶子节点的样本数量来控制，最后的样本数量越少，就越可能出现模型过分的去拟合训练样本中的数据。

该章节代码中有这部分实现，在创建树的时候做预剪枝。

```
def _min_samples_leaf_check(self,
                            X):
    items, cnts = np.unique(X, return_counts=True)
    return np.min(cnts) < self.min_samples_leaf
```

剪枝，书中讲的算法是后剪枝，需要计算每个结点的经验熵，这个计算应该是在树构建的时候已经算过。

这里面没有具体的实现例子，给出的参考文献是李航老师在CL上的文章，文章介绍的MDL是模型选择的一种具体框架，里面有介绍KL散度，这部分可以参考下。

CART

关于CART的算法可以看下十大算法里面的第十章[7]，一转眼就十大数据挖掘算法都发表十年了，这个本书第十章作者放在了ResearchGate上，链接见参考部分。

算法5.5 最小二乘回归树生成

输入：训练数据集D

输出：回归树$f(x)$

步骤：

1. 遍历变量j，对固定的切分变量j扫描切分点s，得到满足上面关系的(j, s)

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

2. 用选定的(j, s), 划分区域并决定相应的输出值

$$R_1(j,s)=\{x|x^{(j)}\le s\}, R_2(j,s)=\{x|x^{(j)}>s\}$$
$$\hat{c}_m=\frac{1}{N}\sum_{x_i\in R_m(j,s)}y_j, x\in R_m, m=1,2$$

3. 对两个子区域调用(1)(2)步骤， 直至满足停止条件
4. 将输入空间划分为M个区域$R_1, R_2, \ldots, R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

算法5.6 CART分类树生成

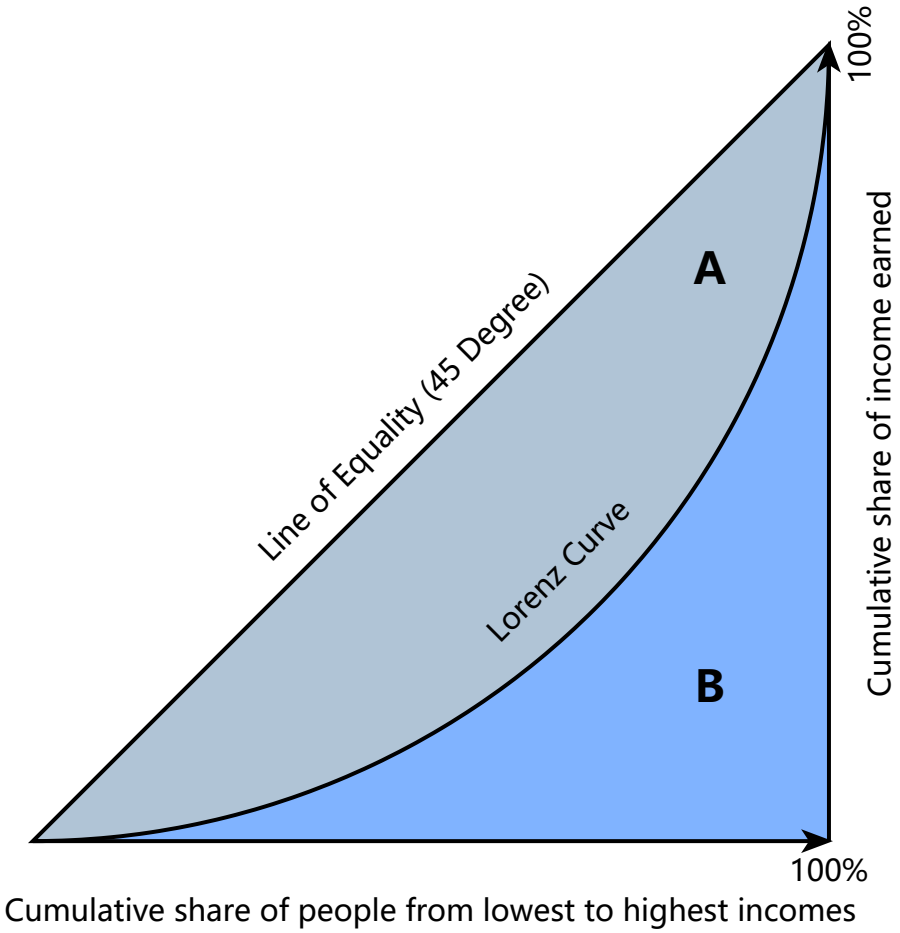
这个算法用到的策略是基尼系数，所以是分类树的生成算法。

> 概率分布的基尼指数定义

$$Gini(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2$$

基尼系数是一个来源于经济学的指标. 范围(0, 1), 有很多中表示形式, 比如衡量收入分布的基尼系数.

$$G=\frac{1}{n-1}\sum_{j=1}^{n}(2j-n-1)p(i_j)$$



经济学基尼系数的解释[8],基尼系数为$\frac{A}{A+B}$

算法5.7 CART剪枝

例子

例5.1

这个例子引出在特征选择的问题，后面跟着引出了熵，条件熵，信息增益与信息增益比的概念。这些是介绍决策树学习的基础。

例5.2

根据信息增益准则选择最优特征

习题的5.1是让用信息增益比生成树，和这个基本一样，换一个准则就可以了。在单元测试里面实现了这部分代码。

参考

1.
2.
3.
4.
5.
6.

7.
8.

↑ top

CH06 逻辑斯谛回归与最大熵模型

前言

章节目录

1. 逻辑斯谛回归模型
    1. 逻辑斯谛分布
    2. 二项逻辑斯谛回归模型
    3. 模型参数估计
    4. 多项逻辑斯蒂回归**模型**
2. 最大熵模型
    1. 最大熵原理
    2. 最大熵模型定义
    3. 最大熵模型学习
    4. 极大似然估计
3. 模型学习的最优化算法
    1. 改进的迭代尺度法
    2. 拟牛顿法

导读

> 在最大熵的通用迭代算法GIS中, E过程就是跟着现有的模型计算每一个特征的数学期望值, M过程就是根据这些特征的数学期望值和实际观测值的比值, 调整模型参数. 这里最大化的目标函数是**熵函数**.
>
> --吴军, 数学之美 P_{244}

这里的**熵函数**是条件熵.

- 这一章放在决策树后面，可能就因为熵的概念，对比前面CH05部分理解对应的损失函数，发现其中的区别和联系。
- LR做二分类的时候， $\mathcal{Y}=\{0,1\}$，在涉及到综合考虑各种模型损失函数作为0-1损失上界存在的情况中，$\mathcal{Y}=\{-1,+1\}$，这时方便使用函数间隔$yf(x)$
- 本章从逻辑斯谛分布开始，在CH04的时候，应该熟悉狄利克雷分布和高斯分布，对于离散和连续型的情况应该熟悉这两个分布，这样在这一章看到逻辑斯谛分布的时候会更适应。在书上有这样一句

    > 二项逻辑斯谛回归模型是一种分类模型，由条件概率分布$P(Y|X)$表示，**形式为参数化的逻辑斯谛分布**。

~~这一句是这两小节唯一的联系，可能不是很好理解。~~ 分类问题，可以表示成one-hot的形式，而one-hot可以认为是概率的一种表达，只是很确定的一种概率表达。而最大熵模型，是一种不确定的概率表达，其中这个概率，是一个条件概率，是构建的特征函数生成的概率。他们之间的关系有点像hard 和 soft，类似的思想还有kmeans和GMM之间的关系。
因为书中第四章并没有讲到高斯朴素贝叶斯(GNB)，有GNB做类比，这里可能更容易理解一点，这里重新推荐一下第四章的参考文献1[2]，配合理解NB和LR的关系。

- 在模型参数估计的部分用到了π，这个应该联想到狄利克雷分布
- 关于NB和LR的对比，Ng也有一篇文章[1]
- 平方误差经过Sigmoid之后得到的是非凸函数
- 书中LR篇幅不大， 注意这样一句，`在逻辑斯谛回归模型中，输出Y=1的对数几率是输入x的线性函数。或者说，输出Y=1的对数几率是由输入x的线性函数表示的模型，及逻辑斯谛回归模型。`
- LR 和 Maxent什么关系？有人说明了这两个是等价的。另外也有说在NLP里LR叫做Maxent。Maxent更多的是从信息的角度来分析这个算法。
- 书中没有直接给出LR的损失函数，在CH12中有提到LR的损失函数是**逻辑斯谛损失函数**。如果采用损失函数上界的那类分析方法，定义$\mathcal{Y}=\{+1,-1\}$， 有$\log_2(1+\exp(yf(x)))=\log_2(1+\exp(y(w\cdot x)))$
- 概率的表示方法之一是采用[0,1]之间的数字，也可以使用odds，概率较低的时候用赔率。书中逻辑斯谛分布的定义给的是CDF。
- LR和CRF也可以对比着看，在CRF中， 势函数是严格正的，通常定义为指数函数，概率无向图模型的联合概率分布$P(Y)$可以表示为如下形式：

$$P(Y)=\frac{1}{Z}\prod_C\Psi_C(Y_C)$$
$$Z=\sum_Y\prod_C\Psi_C(Y_C)$$

其实LR 也可以看成这样的形式，对应的Z有如下的表示形式

$$Z=1+\sum_{k=1}^{K-1}\exp(w_k\cdot x)$$

注意， 定义中的$x\in\mathbf{R}^{n+1}, w_k\in\mathbf{R}^{n+1}$，这样，上面常数1是考虑到偏置项归一化了
实际上可以写成下面的形式

$$P(Y=k|x)=\frac{\exp(w_k\cdot x)}{\sum_{k=1}^{K}\exp(w_k\cdot x)}$$

其中第K项对应了偏置，对应的x=1，所以是一个常数w_0，将分子归一化就得到了书中的表达方式，这就是出现个1的原因。

- 对于问题，什么情况下需要归一化，可以考虑下模型是不是要求这个特征要构成一个概率分布。
- 单纯形法是求解线性规划问题的有效方法，最初是为了解决空军军事规划问题，之后成为了解决线性规划问题的有效方法。这个在运筹学中有介绍，比较经典的参考是胡运权的《运筹学》。

模型

> Logistic regression is a special case of maximum entropy with two labels +1 and −1.

逻辑斯谛回归模型

这一章的这个部分，可以认为是对第四章的一个补充与延续，只是第四章最后没有说高斯朴素贝叶斯。在《机器学习，周志华》上把这个叫做对数几率回归。

逻辑斯谛分布

注意分布函数中关于位置参数，形状参数的说明，可以大致的和高斯对应理解。

$$F(x) = P(X \leqslant x) = \frac{1}{1 + \exp(-(x - \mu)/\gamma)}$$

关于逻辑斯谛， 更常见的一种表达是Logistic function

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

这个函数把实数域映射到(0, 1)区间，这个范围正好是概率的范围， 而且可导，对于0输入， 得到的是0.5，可以用来表示等可能性。

二项逻辑斯谛回归模型

书中给出的定义：

二项逻辑斯谛回归模型是如下的条件概率分布：

$$\begin{aligned} P(Y = 1|x) &= \frac{\exp(w \cdot x)}{1 + \exp(w \cdot x)} \\ &= \frac{\exp(w \cdot x)/\exp(w \cdot x)}{(1 + \exp(w \cdot x))/(\exp(w \cdot x))} \\ &= \frac{1}{e^{-(w \cdot x)} + 1} \\ P(Y = 0|x) &= \frac{1}{1 + \exp(w \cdot x)} \\ &= 1 - \frac{1}{1 + e^{-(w \cdot x)}} \\ &= \frac{e^{-(w \cdot x)}}{1 + e^{-(w \cdot x)}} \end{aligned}$$

这部分提到了几率，但是怎么就想到几率了。

之前一直不清楚为什么就联想到几率了, 从哪里建立了这种联系. 直到看了Think Bayes[3].

> One way to represent a probability is with a number between 0 and 1, **but that's not the only way**. If you have ever bet on a **football game or a horse race**, you have probably encountered another representation of probability, called odds

这本书有中文版，希望这部分内容的补充能增加一些博彩业的直觉...

写到这里，突然想到一个人: 吴军博士。不记得数学之美中关于LR是如何描述的, 但是觉得能外延阐述几率和概率的这种联系的内容也许会出现在他的某部作品里。于是翻了数学之美。但，并没有。

数学之美中有这样一个公式

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

然后几率和概率之间的关系有这样一种表达

$$o = \frac{p}{1 - p}$$
$$p = \frac{o}{1 + o}$$

看上面红色部分, **逻辑斯谛分布**对应了一种**概率**, **几率**为指数形式 e^z，z 为**对数几率**$logit$.

$$logit(p) = \log(o) = \log \frac{p}{1 - p}$$

上面是对数几率的定义， 这里对应了事件， 要么发生， 要么不发生。所以逻辑斯谛回归模型就表示成

$$\log \frac{P(Y=1|x)}{1-P(Y=1|x)} = \log \frac{P(Y=1|x)}{P(Y=0|x)} = w \cdot x$$

上面红色部分留一下，后面推广到多类时候用到。

多项逻辑斯谛回归

假设离散型随机变量Y的取值集合是$1, 2, \ldots, K$, 多项逻辑斯谛回归模型是

$$\begin{aligned} P(Y=k|x) &= \frac{\exp(w_k \cdot x)}{1+\sum_{k=1}^{K-1}\exp(w_k \cdot x)}, k=1,2,\ldots,K-1 \\ P(Y=K|x) &= \frac{1}{1+\sum_{k=1}^{K-1}\exp(w_k \cdot x)} \end{aligned}$$

下面看这个多分类模型怎么来的[4]。

二元推广

计算$K-1$种可能的取值发生的概率相对取值K发生的概率的比值， 假设其取对数的结果是x的线性模型， 有

$$\begin{aligned} \ln \frac{P(Y=1|x)}{P(Y=K|x)} &= w_1 \cdot x \\ \ln \frac{P(Y=2|x)}{P(Y=K|x)} &= w_2 \cdot x \\ &\cdots \\ \ln \frac{P(Y=K-1|x)}{P(Y=K|x)} &= w_{K-1} \cdot x \end{aligned}$$

得到取值$1, 2, \ldots, K-1$的概率表示

$$\begin{aligned} P(Y=1|x) &= P(Y=K|x)\exp(w_1 \cdot x) \\ P(Y=2|x) &= P(Y=K|x)\exp(w_2 \cdot x) \\ &\cdots \\ P(Y=K-1|x) &= P(Y=K|x)\exp(w_{K-1} \cdot x) \\ P(Y=k|x) &= P(Y=K|x)\exp(w_k \cdot x), k=1,2,\ldots,K-1 \end{aligned}$$

上面红色部分有点像书上的(6.7)， 又有K种可能取值概率和为1，可以得到下面推导

$$\begin{aligned} P(Y=K|x) &= 1-\sum_{j=1}^{K-1}P(Y=j|x) \\ &= 1-P(Y=K|x)\sum_{j=1}^{K-1}\exp(w_j \cdot x) \\ &= \frac{1}{1+\sum_{j=1}^{K-1}\exp(w_j \cdot x)} \end{aligned}$$

所以之前红色部分的表达可以表示为

$$\begin{aligned} P(Y=k|x) &= P(Y=K|x)\exp(w_k \cdot x), k=1,2,\ldots,K-1 \\ &= \frac{1}{1+\sum_{j=1}^{K-1}\exp(w_j \cdot x)}\exp(w_k \cdot x), k=1,2,\ldots,K-1 \\ &= \frac{\exp(w_k \cdot x)}{1+\sum_{j=1}^{K-1}\exp(w_j \cdot x)}, k=1,2,\ldots,K-1 \end{aligned}$$

这里公式和书上有点区别， 求和的用了j表示， 感觉不太容易造成误解。

对数线性模型

假设归一化因子Z， 有如下关系

$$\ln(ZP(Y=k|x)) = w_k \cdot x, k = 1, 2, \ldots, K$$
$$P(Y=k|x) = \frac{1}{Z}\exp(w_k \cdot x), k = 1, 2, \ldots, K$$

又对所有的$P(Y=k|x)$可以形成概率分布， 有

$$\begin{aligned}\sum_{k=1}^{K} P(Y=k|x) &= 1 \\ &= \sum_{k=1}^{K} \frac{1}{Z}\exp(w_k \cdot x) \\ &= \frac{1}{Z}\sum_{k=1}^{K}\exp(w_k \cdot x)\end{aligned}$$

得到

$$Z = \sum_{k=1}^{K}\exp(w_k \cdot x)$$

所以

$$P(Y=k|x) = \frac{1}{Z}\exp(w_k \cdot x) = \frac{\exp(w_k \cdot x)}{\sum_{k=1}^{K}\exp(w_k \cdot x)}, k = 1, 2, \ldots, K$$

上面这个叫Softmax，针对多项的情况也叫Softmax Regression。

模型参数估计

通过监督学习的方法来估计模型参数[这部分不完整]。

Logistic Regression

参数估计这里， 似然函数书中的表达

$$\prod_{i=1}^{N}[\pi(x_i)]^{y_i}[1-\pi(x_i)]^{1-y_i}$$

这里利用了$y_i \in \{0, 1\}$这个特点

更一般的表达

$$\prod_{i=1}^{N} P(y_i|x_i, W)$$

使用对数似然会更简单， 会将上面表达式的连乘形式会转换成求和形式。对数函数为单调递增函数， 最大化对数似然等价于最大化似然函数。

$$\begin{aligned}\log\prod_{i=1}^{N}[\pi(x_i)]^{y_i}[1-\pi(x_i)]^{1-y_i} &= \sum_{i=1}^{N}[y_i\log(\pi(x_i)) + (1-y_i)\log(1-\pi(x_i))] \\ &= \sum_{i=1}^{N}[y_i\log(\frac{\pi(x_i)}{1-\pi(x_i)}) + \log(1-\pi(x_i))] \\ &= \sum_{i=1}^{N}[y_i(w \cdot x_i) - \log(1+\exp(w \cdot x_i))]\end{aligned}$$

好像不用这样麻烦，似然函数表示为

$$\prod_{i=1}^{N} P(y_i|x_i, W) = \prod_{i=1}^{N}\frac{(\exp(w \cdot x_i))^{y_i}}{1+\exp(w \cdot x_i)}$$

使用对数技巧

$$\sum_{i=1}^{N} \log \frac{\exp(w \cdot x_i)}{1+\exp(w \cdot x_i)} = \sum_{i=1}^{N} [y_i (w \cdot x_i) - \log(1+\exp(w \cdot x_i))]$$

Softmax Regression

多类分类的情况这个表达式是什么样的？感觉不能用0，1这样的技巧了。

$$\prod_{i=1}^{N} P(y_i|x_i, W) = \prod_{i=1}^{N} \prod_{l=1}^{K} \left(\frac{\exp(w_k \cdot x_i)}{\sum_{k=1}^{K} \exp(w_k \cdot x_i)} \right)^{I(y_i=l)}$$

但是可以用指示函数。

最大熵模型

概念

逻辑斯谛回归模型和最大熵模型，既可以看作是概率模型，又可以看作是非概率模型。

信息量

信息量是对信息的度量, PRML中有关于信息量的讨论，信息是概率的单调函数.

$h(x) = -\log_2 p(x)$, 符号保证了非负性. 低概率事件对应了高的信息量. 对数底选择是任意的, 信息论里面常用2, 单位是比特.

- 信息和概率的关系参考PRML中1.6节信息论部分的描述.

> 如果我们知道某件事件一定会发生, 那么我们就不会接收到信息.
> 于是, 我们对于信息内容的度量将依赖于概率分布$p(x)$
>
> 如果我们有两个不相关的事件x, y, 那么我们观察到两个事件同时发生时获得的信息应该等于观察到事件各自发生时获得的信息之和, 即$h(x,y) = h(x) + h(y)$，这两个不相关的事件是独立的, 因此$p(x,y) = p(x)p(y)$
>
> **根据这两个关系**, 很容易看出$h(x)$一定与$p(x)$的对数有关. 所以有
>
> $h(x) = -\log_2 p(x) = \log_2 \frac{1}{p(x)}$
>
> - 负号确保了信息非负
> - 低概率事件x对应了高的信息.

熵和概率

熵可以从随机变量状态需要的平均信息量角度理解, 也可以从描述统计力学中无序程度的度量角度理解.

关于熵, 条件熵, 互信息, 这些内容在第五章5.2节有对应的描述.

> 下面看下信息熵在PRML中的表达
>
> 假设一个发送者想传输一个随机变量x的值给接受者. 在这个过程中, 他们传输的平均信息量可以通过求**信息$h(x)$关于概率分布$p(x)$的期望**得到.
>
> 这个重要的量叫做随机变量x的熵

Venn图辅助理解和记忆, 这个暂时不画, 下面考虑下为什么Venn图能帮助理解和记忆?

因为熵的定义把连乘变成了求和, 对数的贡献. 这样可以通过集合的交并来实现熵之间关系的理解.

1. **概率** $\sum_{i=1}^{n} p_i = 1$ $p \in [0,1]$
2. **熵**$Ent(D) \in [0, \log_2 |\mathcal{Y}|]$, 熵可以大于1。熵是传输一个随机变量状态值所需的比特位**下界**(信息论角度的理解)，也叫香农下界。
3. **信息熵**是度量样本集合纯度最常用的一种指标。

$$Ent(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

- if $p = 0$，then $p \log_2 p = 0$
- $Ent(D)$越小，D的纯度越高。非均匀分布比均匀分布熵要小。
- 熵衡量的是不确定性，概率描述的是确定性，其实确定性和不确定性差不多。

4. **联合熵(相当于并集)**

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) = H(X|Y) + H(Y|X) + I(X;Y)$$

这个通过Venn应该是相对容易记忆，是不是容易理解这个。

如果X和Y独立同分布，联合概率分布$P(X,Y) = P(X)P(Y)$

5. **条件熵**

条件熵是最大熵原理提出的基础，最大的是条件熵，这个在书中有写(定义6.3)

条件熵衡量了条件概率分布的均匀性

最大熵，就是最大这个**条件熵**

find

$$\begin{aligned} p^* &= \arg\max_{p \in \mathcal{C}} H(p) \\ &= \arg\max_{p \in \mathcal{C}} (-\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x)) \end{aligned}$$

接下来的概念，把熵的思想应用在模式识别问题中。

6. **互信息**

互信息(mutual information)，对应熵里面的交集，常用来描述差异性

一般的，熵$H(Y)$与条件熵$H(Y|X)$之差称为互信息。注意一下，这里第五章中用到了$H(D,A)$ 可以对应理解下。

1. Feature Selection
2. Feature Correlation，刻画的是相互之间的关系。 **相关性主要刻画线性，互信息刻画非线性**

7. **信息增益**

这个对应的是第五章的内容，决策树学习应用信息增益准则选择特征。

$$g(D,A) = H(D) - H(D|A)$$

信息增益表示得知X的信息而使类Y的信息的不确定性减少的程度。

在决策树学习中，信息增益等价于训练数据集中类与特征的互信息。

8. **相对熵 (KL 散度)**

相对熵(Relative Entropy)描述差异性，从分布的角度描述差异性，可用于度量两个概率分布之间的差异。

KL散度不是一个度量，度量要满足交换性。

KL散度满足非负性。

> 考虑由$p(x,y)$给出的两个变量x和y组成的数据集。如果变量的集合是独立的，那么他们的联合分布可以分解为边缘分布的乘积$p(x,y) = p(x)p(y)$
>
> 如果变量不是独立的，那么我们可以通过考察**联合分布**与**边缘分布乘积**之间的KL散度来判断他们是否"接近"于相互独立。
>
> $$I(x,y) = KL(p(x,y)|p(x)p(y)) = -\iint p(x,y) \ln \left(\frac{p(x)p(y)}{p(x,y)} \right)$$
>
> 这被称为变量x和变量y之间的互信息.
>
> --PRML 1.6.1

注意这里，参考下第五章中关于互信息的描述

> 决策树学习中的信息增益等价于训练数据集中**类**与**特征**的互信息

注意这里面类Y， 特征X。

互信息和条件熵之间的关系

$$I(x,y) = H(X) - H(x|y) = H(y) - H(y|x)$$

可以把互信息看成由于知道y值而造成的x的不确定性的减小(反之亦然)。*这个就是信息增益那部分的解释*。

9. **交叉熵**

刻画两个分布之间的差异

$$\begin{aligned} CH(p,q) &= -\sum_{i=1}^{n} p(x_i)\log q(x_i) \\ &= -\sum_{i=1}^{n} p(x_i)\log p(x_i) + \sum_{i=1}^{n} p(x_i)\log p(x_i) - \sum_{i=1}^{n} p(x_i)\log q(x_i) \\ &= H(p) + \sum_{i=1}^{n} p(x_i)\log\frac{p(x_i)}{q(x_i)} \\ &= H(p) + KL(p||q) \end{aligned}$$

CNN时候常用

对于各种熵的理解，是构建后面的目标函数的基础。

最大熵原理

最大熵原理(Maxent principle)是**概率模型**学习的一个准则。

书中通过一个例子来介绍最大熵原理，下面引用一下文献中关于这个例子的总结。

> Model all that is known and assume nothing about that which is unknown. In other words, given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible.
>
> -- Berger, 1996

书中关于这部分的总结如下：**满足约束条件下求等概率的方法估计概率分布**

关于最大熵原理有很多直观容易理解的解释，比如Berger的例子，比如吴军老师数学之美中的例子。

最大熵原理很常**见**，很多原理我们都一直在用，只是没有上升到理论的高度。

等概率表示了对事实的无知，因为没有更多的信息，这种判断是合理的。

最大熵原理认为要选择的概率模型首先必须满足**已有的事实**，即**约束条件**

最大熵原理根据已有的信息（**约束条件**），选择适当的概率模型。

最大熵原理认为不确定的部分都是等可能的，通过熵的最大化来表示**等可能性。**

最大熵的原则，承认已有的，且对未知无偏

最大熵原理并不直接关心特征选择，但是特征选择是非常重要的，因为约束可能是成千上万的。

最大熵原理几何解释

这部分书中只描述了模型空间$\mathcal{P}$，两个约束C_1和C_2是**一致性**约束的情况。

在Berger 1996里面有展开这部分，分了四个图，分别讨论了

1. 概率模型空间$\mathcal{P}$
2. 单一约束C_1
3. 一致性(consistent)约束C_1和C_2，这种情况下模型唯一确定$p = C_1 \bigcap C_2$
4. 非一致性(inconsistent)约束C_1和C_3，这种情况下没有满足约束条件的模型。

特征与约束条件

关于特征和约束，Berger有他的阐述

> 指示函数

> $$f(x,y)=\begin{cases}1 & if\ y=en\ and\ April\ follows\ in\\ 0 & otherwise\end{cases}$$

上面这个f直接引用自Berger的说明，原来的例子是英语in到法语的翻译。

这里面f就是**特征函数**，或者**特征**。

定义一个期望，如果是二值函数的话，就相当于计数。通过样本得到的这个统计。但是样本是有限的，并不是一个真实的分布， 所以叫经验分布，如果我们拿到的这个模型能够表示实际的分布，那么就可以假设经验分布和真实分布是相等的。这个，就是**约束方程**，或者**约束**。

一般模型的**特征**是关于x的函数，最大熵模型中的特征函数，是关于x和y的函数。注意理解$f(x)$与$f(x,y)$的区别。 关于特征函数可以参考条件随机场中例11.1关于特征部分的内容增强理解。

模型

假设分类模型是一个条件概率分布，$P(Y|X),\ X\in\mathcal{X}\subseteq\mathbf{R}^n$

给定一个训练集 $T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$

N是训练样本容量，$x\in\mathbf{R}^n$

联合分布$P(X,Y)$与边缘分布P(X)的经验分布分别为$\widetilde{P}(X,Y)$和$\widetilde{P}(X)$

$$\widetilde{P}(X=x,Y=y)=\frac{\nu(X=x,Y=y)}{N}$$
$$\widetilde{P}(X=x)=\frac{\nu(X=x)}{N}$$

上面两个就是不同的数据样本，在训练数据集中的比例。

如果增加n个**特征函数**, 就可以增加n个**约束条件**，特征也对应增加了一列。

假设满足所有约束条件的模型集合为

$\mathcal{C}\equiv\ \{P\in\mathcal{P}|E_P(f_i)=E_{\widetilde{P}}(f_i),i=1,2,\dots,n\}$

定义在条件概率分布$P(Y|X)$上的条件熵为

$H(P)=-\sum\limits_{x,y}\widetilde{P}(x)P(y|x)\log P(y|x)$

则模型集合$\mathcal{C}$中条件熵$H(P)$最大的模型称为最大熵模型，上式中对数为自然对数。

特征函数$f(x,y)$关于经验分布$\widetilde{P}(X,Y)$的期望值用$E_{\widetilde{P}}(f)$表示

$E_{\widetilde{P}}(f)=\sum\limits_{x,y}\widetilde{P}(x,y)f(x,y)$

特征函数$f(x,y)$关于模型$P(Y|X)$与经验分布$\widetilde{P}(X)$的期望值, 用$E_P(f)$表示

$E_P(f)=\sum\limits_{x,y}\widetilde{P}(x)P(y|x)f(x,y)$

如果模型能够获取训练数据中的信息，那么就有

$$\widetilde{P}(x,y)=P(y|x)\widetilde{P}(x)$$

就可以假设这两个期望值相等，即

$E_P(f)=E_{\widetilde{P}}(f)$

上面这个也是约束方程

算法实现

特征提取原理

通过对已知训练集数据的分析，能够拿到联合分布的经验分布和边缘分布的经验分布。

特征函数用来描述$f(x,y)$描述输入x和输出y之间的某一事实。

$$f(x,y)=\begin{cases}1 & x与y满足某一事实\\ 0 & 否则\end{cases}$$

这里，满足的事实，可以是in，显然，特征函数可以自己定义，可以定义多个，~~这些就是约束~~

之前理解的不对，看前面有描述特征和约束的关系。

预测分类原理

这里面重复一下书中的过程，在$L(P,w)$对P求导并令其为零的情况下解方程能拿到下面公式

$$P(y|x)=\exp\left(\sum_{i=1}^{n}w_i f_i(x,y)+w_0-1\right)=\frac{\exp\left(\sum\limits_{i=1}^{n}w_i f_i(x,y)\right)}{\exp\left(1-w_0\right)}$$

书中有提到因为$\sum_y P(y|x)=1$，然后得到模型

$$P_w(y|x)=\frac{1}{Z_w(x)}\exp\sum_{i=1}^{n}w_i f_i(x,y)$$

$$Z_w(x)=\sum_y\exp\sum_{i=1}^{n}w_i f_i(x,y)$$

注意这里面Z_w是归一化因子。

这里面并不是因为概率为1推导出了Z_w的表达式，而是因为Z_w的位置在分母，然后对应位置$\exp(1-w_0)$也在分母，凑出来这样一个表达式，意思就是遍历y的所有取值，求分子表达式的占比。

综上，如果$f_i(x,y)$只检测是不是存在这种组合，那么概率就是*归一化的*出现过的特征，系数求和再取e指数。

最大熵模型的学习

最大熵模型的学习过程就是求解最大熵模型的过程。

最大熵模型的学习可以形式化为约束最优化问题。

$$\min_{P\in\mathcal{C}}-H(P)=\sum_{x,y}\tilde{P}(x)P(y|x)\log P(y|x) \tag{6.14}$$

$$s.t.\ E_P(f_i)-E_{\tilde{P}}(f_i)=0,i=1,2,\ldots,n \tag{6.15}$$

$$\sum_y P(y|x)=1 \tag{6.16}$$

可以通过例6.2 来理解最大熵模型学习的过程，例6.2 考虑了两种约束条件，这部分内容可以通过python符号推导实现，下面代码整理整个求解过程。

例6.2

一个约束条件

```
from sympy import *

# 1 constrains
P1, P2, P3, P4, P5, w0, w1, w2 = symbols("P1, P2, P3, P4, P5, w0, w1, w2", real=True)
L = P1 * log(P1) + P2 * log(P2) + P3 * log(P3) + P4 * log(P4) + P5 * log(P5) \
    + w0 * (P1 + P2 + P3 + P4 + P5 - 1)
P1_e = (solve(diff(L, P1), P1))[0]
P2_e = (solve(diff(L, P2), P2))[0]
P3_e = (solve(diff(L, P3), P3))[0]
P4_e = (solve(diff(L, P4), P4))[0]
P5_e = (solve(diff(L, P5), P5))[0]
L = L.subs({P1: P1_e, P2: P2_e, P3: P3_e, P4: P4_e, P5: P5_e})
w = (solve([diff(L, w0)], [w0]))[0]
P = [P1_e.subs({w0: w[0]}),
     P2_e.subs({w0: w[0]}),
```

```
     P3_e.subs({w0: w[0]}),
     P4_e.subs({w0: w[0]}),
     P5_e.subs({w0: w[0]})]
P
```

两个约束条件

```
# 2 constrains
P1, P2, P3, P4, P5, w0, w1, w2 = symbols("P1, P2, P3, P4, P5, w0, w1, w2",real=True)
L = P1*log(P1) + P2*log(P2)+P3*log(P3)+P4*log(P4)+P5*log(P5)\
    +w1*(P1+P2-3/10)\
    +w0*(P1+P2+P3+P4+P5-1)
P1_e = (solve(diff(L,P1),P1))[0]
P2_e = (solve(diff(L,P2),P2))[0]
P3_e = (solve(diff(L,P3),P3))[0]
P4_e = (solve(diff(L,P4),P4))[0]
P5_e = (solve(diff(L,P5),P5))[0]
L = L.subs({P1:P1_e, P2:P2_e, P3:P3_e, P4:P4_e, P5:P5_e})
w = (solve([diff(L,w1),diff(L,w0)],[w0,w1]))[0]
P = [P1_e.subs({w0:w[0], w1:w[1]}),
     P2_e.subs({w0:w[0], w1:w[1]}),
     P3_e.subs({w0:w[0], w1:w[1]}),
     P4_e.subs({w0:w[0], w1:w[1]}),
     P5_e.subs({w0:w[0], w1:w[1]})]
P
```

三个约束条件

```
# 3 constrains
P1, P2, P3, P4, P5, w0, w1, w2 = symbols("P1, P2, P3, P4, P5, w0, w1, w2",real=True)
L = P1*log(P1) + P2*log(P2)+P3*log(P3)+P4*log(P4)+P5*log(P5)\
    +w2*(P1+P3-1/2)\
    +w1*(P1+P2-3/10)\
    +w0*(P1+P2+P3+P4+P5-1)
P1_e = (solve(diff(L,P1),P1))[0]
P2_e = (solve(diff(L,P2),P2))[0]
P3_e = (solve(diff(L,P3),P3))[0]
P4_e = (solve(diff(L,P4),P4))[0]
P5_e = (solve(diff(L,P5),P5))[0]
L = L.subs({P1:P1_e, P2:P2_e, P3:P3_e, P4:P4_e, P5:P5_e})
w = (solve([diff(L,w2),diff(L,w1),diff(L,w0)],[w0,w1,w2]))[0]
P = [P1_e.subs({w0:w[0], w1:w[1],w2:w[2]}),
     P2_e.subs({w0:w[0], w1:w[1],w2:w[2]}),
     P3_e.subs({w0:w[0], w1:w[1],w2:w[2]}),
     P4_e.subs({w0:w[0], w1:w[1],w2:w[2]}),
     P5_e.subs({w0:w[0], w1:w[1],w2:w[2]})]
P
```

模型学习

逻辑斯谛回归模型和最大熵模型学习归结为以**似然函数**为**目标函数**的最优化问题，通常通过迭代算法求解。

目标函数

逻辑斯谛回归模型

$$
\begin{aligned}
L(w) &= \sum_{i=1}^{N}[y_i \log \pi(x_i) + (1-y_i)\log(1-\pi(x_i))] \\
&= \sum_{i=1}^{N}[y_i \log \frac{\pi(x_i)}{1-\pi(x_i)} + \log(1-\pi(x_i))] \\
&= \sum_{i=1}^{N}[y_i (w \cdot x_i) - \log(1+\exp(w \cdot x_i)]
\end{aligned}
$$

最大熵模型

$$
\begin{aligned}
L_{\widetilde{P}}(P_w) &= \sum_{x,y} \widetilde{P}(x,y) \log P(y|x) \\
&= \sum_{x,y} \widetilde{P}(x,y) \sum_{i=1}^{n} w_i f_i(x,y) - \sum_{x,y} \widetilde{P}(x,y) \log \left(Z_w(x)\right) \\
&= \sum_{x,y} \widetilde{P}(x,y) \sum_{i=1}^{n} w_i f_i(x,y) - \sum_{x,y} \widetilde{P}(x) P(y|x) \log \left(Z_w(x)\right) \\
&= \sum_{x,y} \widetilde{P}(x,y) \sum_{i=1}^{n} w_i f_i(x,y) - \sum_{x} \widetilde{P}(x) \log \left(Z_w(x)\right) \sum_{y} P(y|x) \\
&= \sum_{x,y} \widetilde{P}(x,y) \sum_{i=1}^{n} w_i f_i(x,y) - \sum_{x} \widetilde{P}(x) \log \left(Z_w(x)\right)
\end{aligned}
$$

以上推导用到了$\sum_{y} P(y|x) = 1$

1. 逻辑斯谛回归模型与朴素贝叶斯的关系

   这部分内容书中引用了参考文献[4]，这是Mitchell的那本《机器学习》，应该说的是第六章中相关的部分推导。

   注意对应的部分还写了在神经网络中梯度搜索实现似然最大化，给出了结果与BP结果的差异，这部分可以看看。

2. 逻辑斯谛回归模型与AdaBoost的关系

3. 逻辑斯谛回归模型与核函数的关系

其他

课后习题的第一个题目提到了**指数族**(Exponential family)分布，这个概念在PRML中有单独的章节进行阐述。

扩展一下指数族分布：

正态分布					
两点分布					
二项分布					
泊松分布					
伽马分布					

大部分的算法实现，其实都是在刷权重，记得有个同学的昵称就是“一切源于wx+b”。另外，**其实算法能跑起来，能预测，并不能说明算法实现的正确的，或者说并不一定是最优的。** 但是其实这种情况也比较常见， 有的时候没有条件去创造一个专门的工具来解决问题的时候，或者没有更好的工具解决问题的时候， 我们会选择能解决部分问题，或者能解决问题的工具来**对付**

代码实现

关于代码实现，网上看似众多的版本，应该基本上都源自最早15年的一份GIS的程序。

无论怎样，这些代码的实现，都会有助于对Maxent的理解。推荐后面参考文献[1]

李航老师在本章给出的参考文献中[1, 2]是Berger的文章。

Demo

这部分代码没有LR的说明。

> 代码来源: https://vimsky.com/article/776.html
> 相关公式: https://vimsky.com/article/714.html

提几点:

1. 代码参考文献可以看berger的文章，公式编号基本对应。
2. 这份代码用defaultdict实现了稀疏存储。
3. 如果$f(x,y)$只是判断(x,y)在特征中出现的指示函数，那么特征可以简单的表示为(x,y)，这样给定一份含有标签的数据集，特征的数量就是 $m \times n$ 其中m是标签的数量，n是词表大小。注意书中注释过，$f(x,y)$可以是任意实值函数。
4. 这份代码思路很清晰，$(E_{\tilde{p}}, Z_x \Rightarrow P(y|x) \Rightarrow E_p) \Rightarrow \delta$，具体参考书中公式6.22，6.23，6.34
5. 体会一下在做直方图的时候，对于同一个样本，同样的特征出现多次的贡献是一样的。
6. 在未知的情况下，输出结果等概率。

Maxent

> 参考链接: https://github.com/WenDesi/lihang_book_algorithm/tree/master/maxENT

本来是想在这个代码的基础上更改，但是代码分解的不是非常容易理解。改来改去基本上面目全非了。保留链接以示感谢。博主写了一系列代码，至少有个成体系的参考。

提几点:

1. 没有用稀疏存储，所以，矩阵中会有很多零。需要除零错误处理

   ```
   with np.errstate(divide='ignore', invalid='ignore'):
       tmp = np.true_divide(self.EPxy, self.EPx)
       tmp[tmp == np.inf] = 0
       tmp = np.nan_to_num(tmp)
   ```

   分子分母都是0对应nan，分母为0对应inf

2. 尝试了三种数据，可以通过命令行参数实现数据选择。
   - Demo中用到的data
   - train_binary.csv 这份数据的来源是参考链接中的，只考虑了0，1两种数据，标签少。
   - sklearn中的digits，标签全，但是8x8大小，比mnist少。其实8x8也不是一个非常小的图了，因为数字相对简单一点，用OpenCV做级联分类器的训练的时候，建议的图片大小是20x20，或者40x40，或者60x60不要太大
3. 书中有一个地方还是不理解的，提到了$f^{\#}$是不是常数的问题。
4. 没有采用字典的方式实现稀疏存储，但是numpy的数组操作还是很便捷的，后面有空评估一下存储和计算资源的消耗情况。

5. 大多数算法都是在刷权重，考虑哪些量(特征)可以用，哪些方法(算法)可以让权重刷的更合理，哪些方法(优化方法)能刷的更快。

Mnist

有同学问LR实现中的GD，才发现那段代码不是很好读。而且，用到的train.csv已不在。

加了一个mnist_sample.py从Lecun那里下载数据，并按照类别采样300条。用来完成LR的Demo。

有些程序的问题，配合数据来理解。通常用到label乘法都是利用了label的符号，或者one-hot之后为了取到对应的类别的值。

代码更新了下，建议运行logistic_regression.py的时候在注释的位置断点，看下各个数据的shape，希望对理解代码有帮助。

参考

1. Berger,1995, A Brief Maxent Tutorial
2. [数学之美:信息的度量和作用]
3. [数学之美:不要把鸡蛋放在一个篮子里 谈谈最大熵模型]
4. 李航·统计学习方法笔记·第6章 logistic regression与最大熵模型（2）·最大熵模型
5. 最大熵模型与GIS ,IIS算法
6. 关于最大熵模型的严重困惑：为什么没有解析解？
7. 最大熵模型介绍 这个是Berger的文章的翻译.
8. 理论简介 代码实现
9. 另外一份代码
10. 如何理解最大熵模型里面的特征？
11. Iterative Scaling and Coordinate Descent Methods for Maximum Entropy Models
12.
13.
14.
15.
16. https://blog.csdn.net/u012328159/article/details/72155874

↑ top

CH07 支持向量机

前言

章节目录

1. [理论]线性可分支持向量机与硬间隔最大化
    1. 线性可分支持向量机
    2. 函数间隔和几何间隔
    3. 间隔最大化
    4. 学习的对偶算法
2. [理论]线性支持向量机与软间隔最大化
    1. 线性支持向量机
    2. 学习的对偶算法
    3. 支持向量
    4. 合页损失函数
3. [理论]非线性支持向量机与核函数
    1. 核技巧
    2. 正定核
    3. 常用核函数
    4. 非线性支持向量分类机
4. [实现]序列最小最优化算法
    1. 两个变量二次规划的求解方法
    2. 变量的选择方法
    3. SMO算法

导读

本章概要部分比较精简, 多刷几遍。

支持向量机是一种二类分类模型。

- **基本模型**是定义在特征空间上的间隔最大的**线性分类器**
- 支持向量机还包括**核技巧**， 这使它称为实质上的**非线性分类器。**
- 支持向量机学习策略是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。
- 支持向量机：线性可分支持向量机，线性支持向量机假设输入空间和特征空间的**元素一一对应**，并将输入空间中的输入映射为特征空间的特征向量；非线性支持向量机利用一个从输入空间到特征空间的**非线性映射**将输入映射为特征向量。
- 判别模型
- 符号函数, Sign, 0 对应了超平面(hyper plane), >0与<0对应了半空间(half space)
- 仿射变换是保凸变换
- `分离超平面将特征空间划分为两部分，一部分是正类，一部分是负类。法向量指向的一侧是正类， 另一侧为负类`
- 关于SVM的历史可以参考附录2[1]，Vapnik的1995年的那个文章名字叫Support-vector networks，主要是提出了soft margin，在这篇文章的附录中给出了线性可分支持向量机与线性支持向量机的推导。同年Vapnik将支持向量机推广到支持向量回归，发表了文章统计学习理论的本质， 这个有中译版本，见书中本章参考文献[4]
- Vapnik全名弗拉基米尔·万普尼克， 出生于苏联， VC理论的主要创建人之一，统计学习理论之父。1990年底移居美国，1991-2001年间，他在AT&T工作。2006年成为美国国家工程院院士。2014年加入Facebook人工智能实验室。
- 1995年SVN的文章结尾有这样的描述：`The support-vector network combines 3 ideas: the solution technique from optimal hy- perplanes (that allows for an expansion of the solution vector on support vectors), the idea of convolution of the dot-product (that extends the solution surfaces from linear to non-linear), and the notion of soft margins (to allow for errors on the training set).`
- KKT条件是该最优化问题的充分必要条件。
- 样本比较多的时候，Boosting和RF往往能得到最好的效果，但是数据集较少的时候，SVM的效果可能会比较好。

这一章介绍了几个算法

1. 算法7.1 最大间隔法，针对线性可分支持向量机，直接求w^*, b^*
2. 算法7.2 学习的对偶算法，求α进一步求w^*, b^*
3. 算法7.3 线性支持向量机的学习算法， 增加超参C，求α进一步求w^*, b^*
4. 算法7.4 非线性支持向量机的学习算法，进一步引入核函数$K(x, z)$，求α进一步求b^*
5. 算法7.5 序列最小最优化

注意前四个算法里面，都没有写该怎么去求解α，最后一节**序列最小最优化**讨论了具体实现。也就是算法7.5给出了求解$\hat{\alpha}$的方法。

另外，注意对比算法7.3和算法7.4，关注输出，关注b^*.

[1] 线性可分支持向量机

问题描述

$$\begin{aligned}&\min_{w,b} \frac{\hat{\gamma}}{\|w\|}\\&s.t. \quad y_i(w \cdot x_i + b) - 1 \geqslant 0, i = 1, 2, \ldots, N\end{aligned}$$

这是个凸二次规划问题.

如果求出了上述方程的解w^*, b^*，就可得到

分离超平面

$$w^* \cdot x + b^* = 0$$

相应的分类决策函数

$$f(x) = sign(w^* \cdot x + b^*)$$

算法

函数间隔

对于给定数据集T和超平面(w, b)，定义超平面(w, b)关于样本点(x_i, y_i)的函数间隔为

$$\hat{\gamma}_i = y_i(w \cdot x_i + b)$$

定义超平面(w, b)关于训练数据集T的函数间隔为超平面(w, b)关于T中所有样本点(x_i, y_i)的函数间隔之最小值，即

$$\hat{\gamma} = \min_{i=1,\cdots,N} \hat{\gamma}_i$$

函数间隔可以表示分类预测的**正确性**及**确信度**。

几何间隔

间隔最大化

支持向量和间隔边界

由于支持向量在确定分离超平面中起着决定作用，所以将这种分类模型称为支持向量机。

支持向量对应

$$y_i(w \cdot x_i + b) - 1 = 0$$

上式对应两个超平面

分离超平面对应

$$w \cdot x + b = 0$$

注意，在算法7.1中，并没有说明这个问题如何求得最优解，在7.1.4节中有描述该如何求解。

针对例7.1，可以先用$scipy$中的包求解，这个例子我们要清楚：

1. 该问题怎么定义的
2. 有哪些约束
3. 支持向量在哪里
4. 三个重要的超平面是怎么得到的

可以用如下程序实现

```
# data 2.1
# x_1 = (3, 3), x_2 = (4, 3), x_3 = (1, 1)
# ref: [example 16.3]
(http://www.bioinfo.org.cn/~wangchao/maa/Numerical_Optimization.pdf)
from scipy import optimize
import numpy as np

fun = lambda x: ((x[0]) ** 2 + (x[1]) ** 2)/2
cons = ({'type': 'ineq', 'fun': lambda x: 3 * x[0] + 3 * x[1] + x[2] - 1},
        {'type': 'ineq', 'fun': lambda x: 4 * x[0] + 3 * x[1] + x[2] - 1},
        {'type': 'ineq', 'fun': lambda x: -x[0] - x[1] - x[2] - 1})
res = optimize.minimize(fun, np.ones(3), method='SLSQP', constraints=cons)
res
```

对偶算法

1. 对偶问题往往更容易求解
2. 自然引入核函数，进而推广到非线性分类问题

针对每个不等式约束，定义拉格朗日乘子$\alpha_i \geq 0$，定义拉格朗日函数

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2} w \cdot w - \left[\sum_{i=1}^{N} \alpha_i [y_i (w \cdot x_i + b) - 1] \right] \\ &= \frac{1}{2} \|w\|^2 - \left[\sum_{i=1}^{N} \alpha_i [y_i (w \cdot x_i + b) - 1] \right] \\ &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^{N} \alpha_i \\ &\alpha_i \geqslant 0, i = 1, 2, \ldots, N \end{aligned}$$

其中$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)^T$为拉格朗日乘子向量

原始问题是极小极大问题

根据**拉格朗日对偶性**，原始问题的**对偶问题是极大极小问题**:

$$\max_{\alpha} \min_{w,b} L(w,b,\alpha)$$

关于对偶问题，引用Wikipedia中的描述

> Duality (optimization)

> In mathematical optimization theory, **duality** or the **duality principle** is the principle that optimization problems may be viewed from either of two perspectives, the **primal problem** or the **dual problem**. The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem.[1] However in general the optimal values of the primal and dual problems need not be equal. Their difference is called the duality gap. For convex optimization problems, the duality gap is zero under a constraint qualification condition.

转换后的对偶问题

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N} \alpha_i$$
$$s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$
$$\alpha_i \geqslant 0, i = 1, 2, \ldots, N$$

对于任意线性可分的两组点，他们在分类超平面上的投影都是线性不可分的。

α不为零的点对应的实例为支持向量，通过支持向量可以求得b值

核心公式两个

$$w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$$
$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i (x_i \cdot x_j)$$

这里面比较重要的是b^*的公式的理解，通过$\arg\max \alpha^*$实现，因为支持向量共线，所以通过任意支持向量求解都可以。

介绍学习的对偶算法之后，引入了例7.2解决之前同样的问题，这部分代码整理下

```
# data 2.1
data = np.array([[3, 3],
                 [4, 3],
                 [1, 1]])
label = np.array([1, 1, -1])

def fun(a):
    return 4 * (a[0]) ** 2 + 13 / 2 * (a[1]) ** 2 + 10 * a[0] * a[1] - 2 * a[0] - 2 *
a[1]
# cons = ({'type': 'ineq', 'fun': lambda a: a[0]},
#         {'type': 'ineq', 'fun': lambda a: a[1]})
bnds = ((0, None), (0, None))
res = optimize.minimize(fun, np.ones(2), method='SLSQP', bounds=bnds)

alpha = res["x"]
alpha = np.append(alpha, alpha[0] + alpha[1])
w = np.sum((alpha * label).reshape(-1, 1) * data, axis=0)
j = np.argmax(alpha)
b = label[j] - np.sum(alpha * label * np.dot(data, data[j, :]), axis=0)
#  0和2都OK，因为都为0.5 > 0
# b = label[0] - np.sum(alpha*label*np.dot(data,data[0,:]),axis=0)
# b = label[2] - np.sum(alpha*label*np.dot(data,data[2,:]),axis=0)
```

至此，两个例子求解的都是线性可分支持向量机。

[2] 线性支持向量机

问题描述

$$\min_{w,b,\xi}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i$$
$$s.t.\quad y_i(w\cdot x_i+b)\geqslant 1-\xi_i, i=1,2,\dots,N$$
$$\xi_i\geqslant 0, i=1,2,\dots,N$$

对偶问题描述

原始问题里面有两部分约束，涉及到两个拉格朗日乘子向量

$$\min_{\alpha}\ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i$$
$$s.t.\quad \sum_{i=1}^{N}\alpha_i y_i=0$$
$$0\leqslant\alpha_i\leqslant C, i=1,2,\dots,N$$

通过求解对偶问题， 得到α，然后求解w,b的过程和之前一样

注意， 书后总结部分，有这样一句描述：**线性支持向量机的解w^*唯一但b^*不一定唯一**

线性支持向量机是线性可分支持向量机的超集。

算法

软间隔最大化

> 对于线性支持向量机学习来说
> 模型为分离超平面$w^*\cdot x+b^*=0$
>
> 决策函数为$f(x)=sign(w^*\cdot x+b^*)$
>
> 学习策略为软间隔最大化
>
> 学习算法为凸二次规划

合页损失

另一种解释，最小化目标函数

$$\min_{w,b}\sum_{i=1}^{N}[1-y_i(w\cdot x+b)]_+ +\lambda\|w\|^2$$

其中

- 第一项是经验损失或经验风险，函数$L(y(w\cdot x+b))=[1-y(w\cdot x+b)]_+$称为合页损失，可以表示成$L=\max(1-y(w\cdot x+b),0)$
- 第二项是**系数为λ的w的L_2范数的平方**，是正则化项

书中这里通过定理7.4说明了用合页损失表达的最优化问题和线性支持向量机原始最优化问题的关系。

$$\min_{w,b,\xi}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i$$
$$s.t.\quad y_i(w\cdot x_i+b)\geqslant 1-\xi_i, i=1,2,\dots,N$$
$$\xi_i\geqslant 0, i=1,2,\dots,N$$

等价于

$$\min_{w,b}\sum_{i=1}^{N}[1-y_i(w\cdot x+b)]_+ +\lambda\|w\|^2$$

证明：

令合页损失$[1-y_i(w\cdot x+b)]_+=\xi_i$，合页损失非负，所以有$\xi_i\geq 0$，这个对应了原始最优化问题中的**一个约束【1】**。

还是根据合页损失非负，当$1-y_i(w\cdot x+b)\leq 0$的时候，有$[1-y_i(w\cdot x+b)]_+=\xi_i=0$，所以有

$1-y_i(w\cdot x+b)\leq 0=\xi_i$，这对应了原始最优化问题中的**另一个约束【2】**。

所以，在满足这**两个约束【1】【2】**的情况下，有

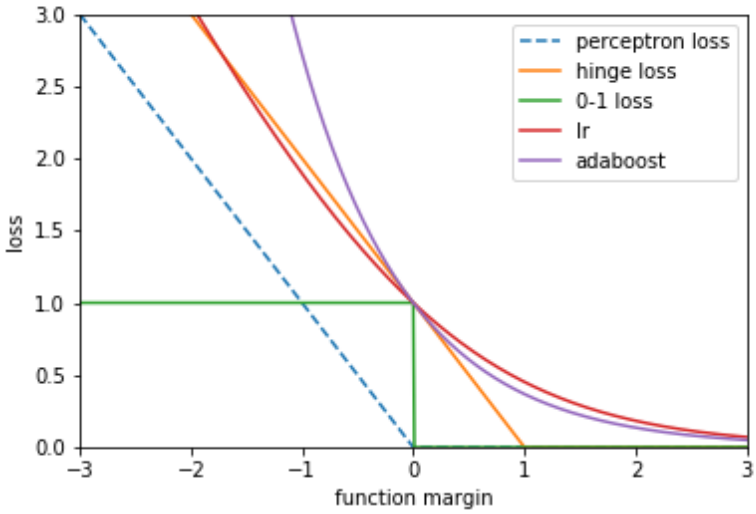
$$\min_{w,b}\sum_{i=1}^{N}[1-y_i(w\cdot x+b)]_+ +\lambda\|w\|^2$$

$$\min_{w,b}\sum_{i=1}^{N}\xi_i+\lambda\|w\|^2$$

$$\min_{w,b}\frac{1}{C}\left(\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}\xi_i\right),with\ \lambda=\frac{1}{2C}$$

看下下面这个图，其中合页损失和感知机损失函数之间的关系，合页损失要求函数间隔大于1的时候才没有损失（loss=0），而感知机只要函数间隔大于0就认为没有损失，所以说合页损失对学习有更高的要求。

再解释下，对于线性支持向量机大于一的意思是符号为正，且值要大于1；对于感知机大于零的意思是符号为正，就可以了。




```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-3,3,601)
# perceptron loss
y1 = list(map(lambda x: max(0, -x), x))
# hinge loss
y2 = list(map(lambda x: max(0,1-x), x))
# 0-1 loss
y3 = list(map(lambda x:1 if x <= 0 else 0, x))
# lr loss
y4 = list(map(lambda x:np.log2(1+np.exp(-x)), x))
# adaboost
y5 = list(map(lambda x:np.exp(-x), x))
plt.plot(x,y1,'--', label='perceptron loss')
plt.plot(x,y2, '-', label='hinge loss' )
plt.plot(x,y3, '-', label='0-1 loss')
plt.plot(x,y4, '-', label='lr')
```

```
plt.plot(x,y5, '-', label='adaboost')

plt.legend()
plt.xlim(-3,3)
plt.ylim(0,3)
plt.xlabel("functional margin")
plt.ylabel("loss")
plt.savefig("test.png")
plt.show()
```

以上：

- 0-1损失函数不是连续可导
- 合页损失认为是0-1损失函数的上界，在AdaBoost中也有说明，指数损失也是0-1损失函数的上界，在感知机中有提到`损失函数的自然选择是误分类点的个数`，这句在最开始见到的时候，可能不一定有上面图片的直觉。注意在本书CH12中也有这个图，可以对比理解下。
- 感知机误分类驱动， 选择函数间隔作为损失考虑分类的正确性，合页损失不仅要考虑分类正确， 还要考虑确信度足够高时损失才是0.

[3]非线性支持向量机

核技巧的想法是在学习和预测中只定义核函数$K(x,z)$，而不是显式的定义映射函数ϕ

通常，直接计算$K(x,z)$比较容易， 而通过$\phi(x)$和$\phi(z)$计算$K(x,z)$并不容易。

$$W(\alpha)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i$$

$$f(x)=sign\left(\sum_{i=1}^{N_s}\alpha_i^* y_i\phi(x_i)\cdot\phi(x)+b^*\right)=sign\left(\sum_{i=1}^{N_s}\alpha_i^* y_i K(x_i,x)+b^*\right)$$

学习是隐式地在特征空间进行的，不需要显式的定义特征空间和映射函数。这样的技巧称为核技巧，核技巧不仅引用于支持向量机，而且应用于其他统计学习问题。

TODO：字符串核函数

核函数

例7.3 主要是为了说明

> 对于给定的核$K(x,z)$，特征空间$\mathcal{H}$和映射函数$\phi(x)$的取法并不唯一，可以取不同的特征空间，即便是同一特征空间里也可以取不同的映射

注意这个例子里面$\phi(x)$实现了从低维空间到高维空间的映射。

$$K(x,z)=(x\cdot z)^2$$
$$\mathcal{X}=\mathbb{R}^2, x=(x^{(1)},x^{(2)})^{\mathcal{T}}$$
$$\mathcal{H}=\mathbb{R}^3, \phi(x)=((x^{(1)})^2,\sqrt{2}x^{(1)}x^{(2)},(x^{(2)})^2)^{\mathcal{T}}$$
$$\mathcal{H}=\mathbb{R}^4, \phi(x)=((x^{(1)})^2,x^{(1)}x^{(2)},x^{(1)}x^{(2)},(x^{(2)})^2)^{\mathcal{T}}$$

这里看下：

- 理解下$\mathbb{R}^n$
- 理解下计算K要比通过ϕ计算K要容易
- 核函数的定义相当于给出了$\phi(x)\cdot\phi(z)$的结果，而没有显式的给出ϕ的定义，ϕ实现了从输入空间到特征空间的变换，所以说，学习是隐式的从特征空间中进行的，不需要显式的定义特征空间和映射函数，这样的技巧称为核技巧，通过线性分类学习方法和核函数解决非线性问题。

核具有再生性即满足

$$K(\cdot,x)\cdot f=f(x)$$
$$K(\cdot,x)\cdot K(\cdot,z)=K(x,z)$$

称为再生核

问题描述

~~和线性支持向量机的问题描述一样~~，注意，这里是有差异的，将向量内积替换成了核函数，而后续SMO算法求解的问题是该问题。

构建最优化问题：

$$\begin{aligned}\min_{\alpha}\ & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i\\ s.t.\ & \sum_{i=1}^{N}\alpha_i y_i=0\\ & 0\leqslant\alpha_i\leqslant C,i=1,2,\dots,N\end{aligned}$$

求解得到$\alpha^*=(\alpha_1^*,\alpha_2^*,\cdots,\alpha_N^*)^T$

选择α^*的一个正分量计算

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i K(x_i,x_j)$$

构造决策函数

$$f(x)=sign\left(\sum_{i=1}^{N}\alpha_i^* y_i K(x,x_i)+b^*\right)$$

学习算法：序列最小最优化

支持向量机的学习问题可以形式化为求解凸二次规划问题，这部分主要参考文献是文献5[2]

问题描述

$$\begin{aligned}\min_{\alpha}\ & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i\\ s.t.\ & \sum_{i=1}^{N}\alpha_i y_i=0\\ & 0\leqslant\alpha_i\leqslant C,i=1,2,\dots,N\end{aligned}$$

这个问题中，变量是α，一个变量α_i对应一个样本点(x_i,y_i)，变量总数等于N

KKT 条件

KKT条件是该最优化问题的充分必要条件。这个参考附录C

SMO算法

整个SMO算法包括两~~步骤~~**部分**：

1. 求解两个变量二次规划的解析方法
2. 选择变量的启发式方法

$$\begin{aligned}\min_{\alpha} \ & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i=1}^{N}\alpha_i y_i=0 \\ & 0\leqslant\alpha_i\leqslant C, i=1,2,\ldots,N\end{aligned}$$

注意，这里是**两个部分**，而不是先后的两个步骤。

Part I

两变量二次规划求解

选择两个变量α_1,α_2

由等式约束可以得到

$$\alpha_1=-y_1\sum_{i=2}^{N}\alpha_i y_i$$

所以这个问题实质上是单变量优化问题。

$$\begin{aligned}\min_{\alpha_1,\alpha_2} W(\alpha_1,\alpha_2)=&\frac{1}{2}K_{11}\alpha_1^2+\frac{1}{2}K_{22}\alpha_2^2+y_1y_2K_{12}\alpha_1\alpha_2 \\ &-(\alpha_1+\alpha_2)+y_1\alpha_1\sum_{i=3}^{N}y_i\alpha_iK_{il}+y_2\alpha_2\sum_{i=3}^{N}y_i\alpha_iK_{i2} \\ s.t. \quad & \alpha_1y_1+\alpha_2y_2=-\sum_{i=3}^{N}y_i\alpha_i=\varsigma \\ & 0\leqslant\alpha_i\leqslant C, i=1,2\end{aligned}$$

上面存在两个约束：

1. **线性**等式约束
2. 边界约束

这里有个配图，其中这样一段描述**等式约束使得**(α_1,α_2)**在平行于盒子**$[0,C]\times[0,C]$**的对角线的直线上**

这句怎么理解？

$y_i\in\mathcal{Y}=\{+1,-1\}$所以又等式约束导出的关系式中两个变量的系数相等，所以平行于对角线。

要时刻记得，支持向量机是个二类分类模型。后面有研究者对这个问题做多分类扩展，应该不是简单的OvO或者OvR吧，不然能专门发文章写一下？

书中**剪辑**的概念对应了文献[2]中的**clipped**，剪辑的操作对应的是clipping，也叫truncation或者shortening。

Part II

变量的选择方法

1. 第一个变量α_1
   外层循环
   违反KKT条件**最严重**的样本点
2. 第二个变量α_2
   内层循环
   希望能使α_2有足够大的变化
3. 计算阈值b和差值E_i

算法7.5

> 输入：训练数据集$T=(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)$，其中
> $x_i \in \mathcal{X}=\mathbf{R^n}, \mathbf{y_i} \in \mathcal{Y}=\{-\mathbf{1},+\mathbf{1}\}, \mathbf{i}=\mathbf{1},\mathbf{2},\ldots,\mathbf{N}$,精度$\epsilon$
>
> 输出：近似解$\hat{\alpha}$
>
> 1. 取初值$\alpha_0=0$，令$k=0$
> 2. **选取**优化变量$\alpha_1^{(k)},\alpha_2^{(k)}$，解析求解两个变量的最优化问题，求得最优解$\alpha_1^{(k+1)},\alpha_2^{(k+1)}$，更新$\alpha$为$\alpha^{k+1}$
> 3. 若在精度ϵ范围内满足停机条件
>
> $$\sum_{i=1}^{N}\alpha_i y_i=0$$
> $$0\leqslant\alpha_i\leqslant C, i=1,2,\ldots,N$$
> $$y_i\cdot g(x_i)=\begin{cases}\geqslant 1,\{x_i|\alpha_i=0\}\\=1,\{x_i|0<\alpha_i<C\}\\\leqslant 1,\{x_i|\alpha_i=C\}\end{cases}$$
> $$g(x_i)=\sum_{j=1}^{N}\alpha_j y_j K(x_j,x_i)+b$$
>
> 则转4,否则，$k=k+1$转2
>
> 1. 取$\hat{\alpha}=\alpha^{(k+1)}$

TODO:

其实这里支持向量机的几个核心的思想是不难理解的，这里涉及到的主要是求解二次规划问题么?

扩展

对比支持向量机和提升方法

参考下CH08

对比二次规划求解工具和SMO

习题

7.3

线性支持向量机还可以定义成以下形式:

$$\begin{aligned}&\min_{w,b,\xi}\ \frac{1}{2}||w||^2+C\sum_{i=1}^{N}\xi_i^2\\&s.t.\ y_i(w\cdot x_i+b)\geq 1-\xi_i, i=1,2,\cdots,N\\&\quad\xi_i\geq 0, i=1,2,\cdots,N\end{aligned}$$

求其对偶形式。

参考

1.
2.

[↑ top]

CH08 提升方法

前言

章节目录

1. 提升方法AdaBoost算法
    1. 提升方法的基本思路
    2. AdaBoost算法
    3. AdaBoost的例子
2. AdaBoost算法的训练误差分析
3. AdaBoost算法的解释
    1. 前向分步算法
    2. 前向分布算法与AdaBoost
4. 提升树
    1. 提升树模型
    2. 提升树算法
    3. 梯度提升

导读

- 可能会有疑问，为什么支持向量机前置在这一章之前，可以参考本章的第一个参考文献[2]，Yoav Freund和Robert Schapire因此获得了2003年的哥德尔奖， 本书中关于误差率的表述与该文献是一致的， PRML中则对分类误差率进行了归一化。
- 在训练误差分析部分有这样一句`AdaBoost算法不需要知道下界$\gamma$，这正是Freund与Schapire设计AdaBoost时所考虑的`。如果对这句没有sense，了解下历史。书中也讲到1988年Kearns和Valiant提出"强可学习"和"弱可学习"的概念，并提出了一个问题这两种复杂性类别是否等价。Schapire给出了答案，是等价的，并给出了算法，这是最早的Boost，但是这个算法需要知道学习器的误差界。后来AdaBoost提出，不再需要知道误差界等信息。
- 关于AdaBoost和SVM的联系，主要在**函数间隔**的这个概念上文献[2]和本书CH02CH07中内容合并理解，本文后面在算法的解释部分会添加一个和SVM关系的对比摘要一下文献中的理解。这篇文章在介绍AdaBoost和SVM的关系的时候， 引用了三篇文献，在CH07中都有引用，是经典的SVM文献。
- 间隔包含了分类正确性与确信度的含义。
- 如果看过林轩田老师的课程(只需看第一课)，可能对hypothesis这个概念迷糊，没有sense。那可以翻下前面提到的这个文章，可能会对这些概念的理解有帮助。另外文章中有提到VC维，用来度量hypotheses的复杂度。
- 另外一篇文献，推荐下RE Schapire的文章[1]关于间隔的理解
- AdaBoost的两个性质：
    1. 能在学习过程中不断减少训练误差
    2. 训练误差是以指数速率下降的
- 基函数和基本分类器不是一个概念， 在加法模型部分有提到。

- 这章里面提到了，这样一句，`大多数的提升方法都是改变训练数据的概率分布（训练数据的权值分布），针对不同的训练数据分布调用弱学习算法学习一系列弱分类器`，这里改变训练数据的权值分布，可能不是很容易理解，字面理解好像是给数据乘了系数，实际上这个权值分布在计算分类误差率的时候才用到，通过$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$生成了对应的弱分类器的系数。另外，这个权值分布在更新的时候，做了归一化，使他满足了概率分布的条件。
- 提升树针对不同的问题，选择不同的损失函数：指数损失（分类问题），平方损失（回归问题），一般损失（一般问题），针对一般问题，优化方法采用梯度提升就有了GBDT。
- 书中讲的AdaBoost是用在二分类上，和SVM的二分类差不多，算法8.1中有用到符号函数。公式8.4也用到了$Y = \{1, -1\}$的性质, 和感知机部分的应用类似。
- 在sklearn中有引用书中参考文献5[8]，sklearn中的实现支持多分类，引用了参考文献[3]，本书中引用了更早的一个实现多分类的文献[4]
- 提升方法最初用来解决分类问题，这里面算法8.1描述的就是二分类的算法，
- 本章最后介绍了提升树，关于各种树相关的算法关系，林轩田有页slides，可以参考。[5]
- 算法8.3用来求回归问题的提升树，注意**拟合残差**这个内容的理解，可以理解例子8.2
- 关于参考文献9[7]，这个文章，简要说两句，提到了Bregman distance，这篇文章的编辑是Bengio
- Boosting不容易过拟合

加法模型+前向分步算法

回顾这一章其实可以划分成三个阶段：

1. 1加法模型 + 指数损失[特例,Adaboost又进一步是特例]
2. 2加法模型 + 平方损失[特例]
3. 3加法模型 + 一般损失

再复习下，之前李航老师讲到的：

> 统计学习方法之间的不同，主要来自器模型、策略、算法的不同。确定了模型、策略、算法，统计学习的方法也就确定了。这也就是将其称为统计学习三要素的原因。

再结构化一下这三个部分，好好理解：

- 模型：需要学习的条件概率分布或者决策函数
- 策略：按照什么样的准则学习或选择最优的模型。统计学习的目标在于从假设空间中选取最优模型。
  - 经验风险最小化(R_{emp})
  - 结构风险最小化(R_{srm})
- 算法：考虑用什么样的方法求解最优模型，这时统计学习问题归结为最优化问题，统计学习方法的算法称为求解最优化问题的算法。

Boosting方法是一种常用的统计学习方法，应用广泛且有效

- 【在分类问题中】改变训练样本权重，学习多个分类器
- 线性组合

提升方法AdaBoost算法

提升方法的基本思路

概率近似正确(PAC, Probably approximately correct)

在PAC学习框架下，一个概念是强可学习的**充分必要条件**是这个概念是弱可学习的。
两个问题

1. 在每一轮如何改变训练数据的权值或者概率分布
2. 如何将弱分类器组合成一个强分类器

Adaboost解决方案：

1. 提高前一轮被分错的分类样本的权值，降低被正确分类的样本的权值
2. 加权多数表决的方法

Adaboost算法

算法8.1

- 输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\},x\in\mathcal{X}\subseteq\mathbb{R}^n$, 弱学习方法
- 输出：最终分类器$G(x)$

步骤

1. 初始化训练数据的权值分布 $D_1=(w_{11},\cdots,w_{1i},\cdots,w_{1N},w_{1i}=\frac{1}{N})$
2. m = 1,2, M
    1. $G_m(x):X->-1,+1$
    2. 求G_m在训练集上的分类误差率 $e_m=\sum_{i=1}^{N}P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$
    3. 计算$G_m(x)$的系数，$\alpha_m=\frac{1}{2}log\frac{1-e_m}{e_m}$，自然对数
    4. $w_{m+1,i}=\frac{w_{mi}}{Z_m}exp(-\alpha_m y_i G_m(x_i))$
    5. $Z_m=\sum_{i=1}^{N}w_{mi}exp(-\alpha_m y_i G_m(x_i))$
3. $f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$
4. 最终分类器$G(x)=sign(f(x))=sign(\sum_{m=1}^{M}\alpha_m G_m(x))$■

从算法8.1的输入可以看出来，AdaBoost是个集成学习算法， 因为在他的输入中包含了**弱学习算法。**

注意这里面有个描述

> 使用具有权值分布D_m的训练数据集

这个怎么理解，是改变了数据么？

- 这里不是的
- 弱分类器的分类准则是错误率$e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$
- 弱分类器的分类准则是错误率$e_m=\sum_{i=1}^{N}P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$
- 以上这两条再考虑下
- 每次学习用到的数据集没有变，划分方式也没有变（比如阈值分类器中的分类点的选取方式），变是评价每个划分错误的结果。
- 不同的权值分布上，不同样本错分对评价结果的贡献不同，**分类器**中分类错误的会被放大，分类正确的系数会减小，错误和正确的系数比值为$e^{2\alpha_m}=\frac{1-e_m}{e_m}$，这个比值是分类器分类正确的**几率**($odds$)，关于几率在CH06中有讲到，这也是为什么在1999年的时候Schapire的文章中开篇用Horse racing gambler来引入Boosting，几率就是个源自赌博的概念。
- 书中对这点也有解释：误分类样本在下一轮学习中起更大的作用。不改变所给的训练数据，而不断改变训练数据权值的分布，似的训练数据在基本分类器的学习中起不同的作用， 这是AdaBoost的一个特点。

AdaBoost例子

例子8.1

数据见data_8-1.txt

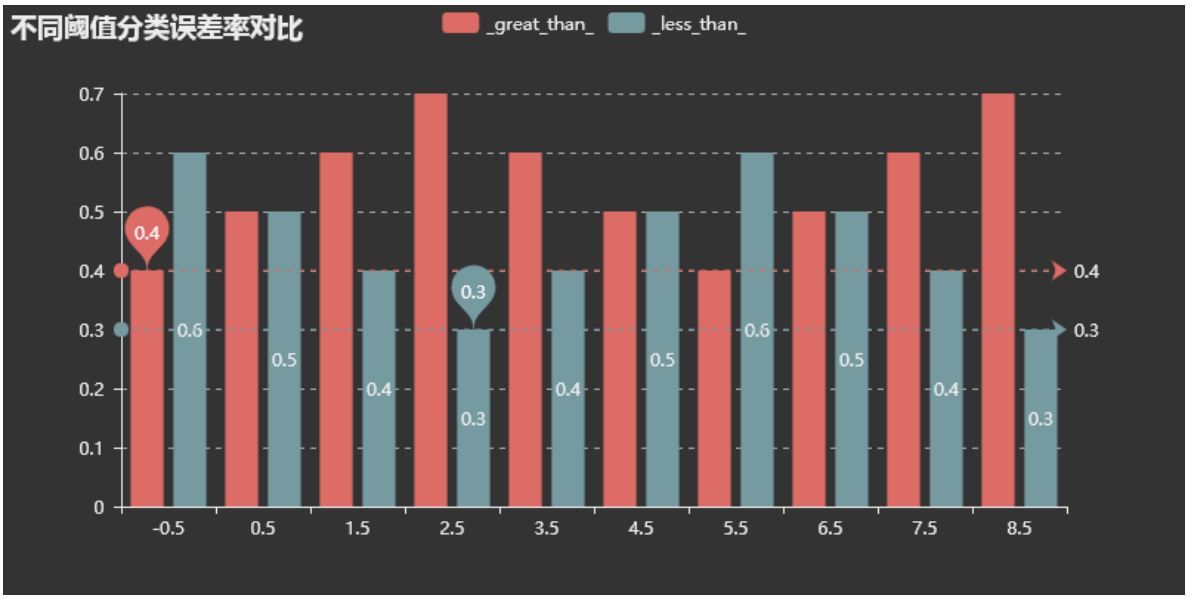
弱分类器选为阈值分类器，通过阈值将数据划分成两部分，标准是分类误差率最低。其实这个弱分类器就是决策树桩。

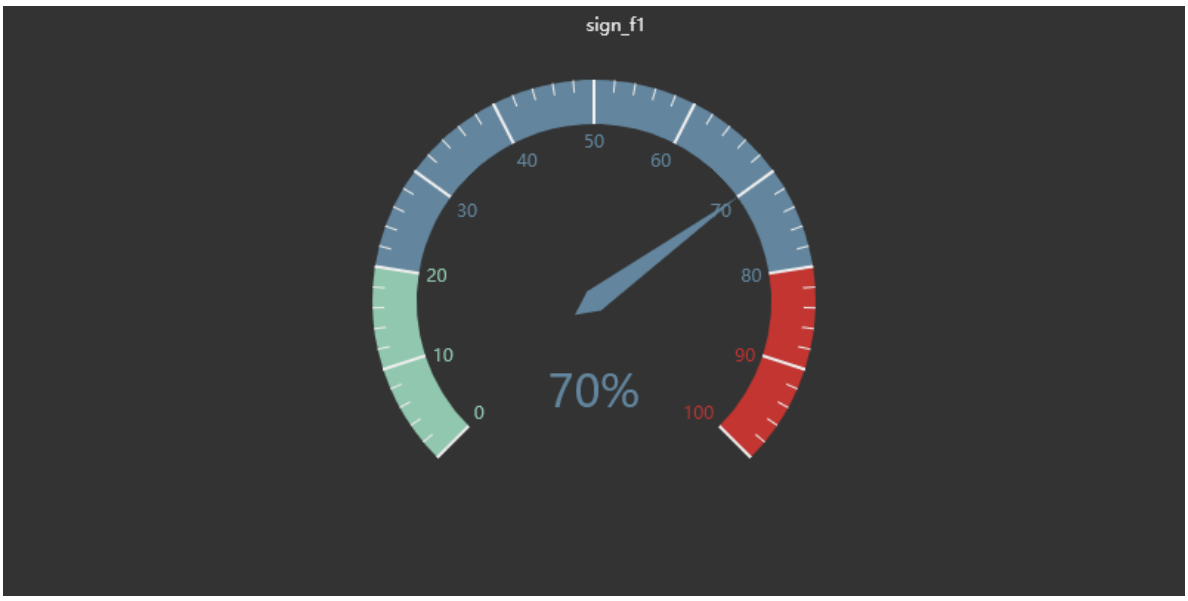
需要确定两个参数：

1. 阈值选在哪里
2. 划分的两部分类别指定方式

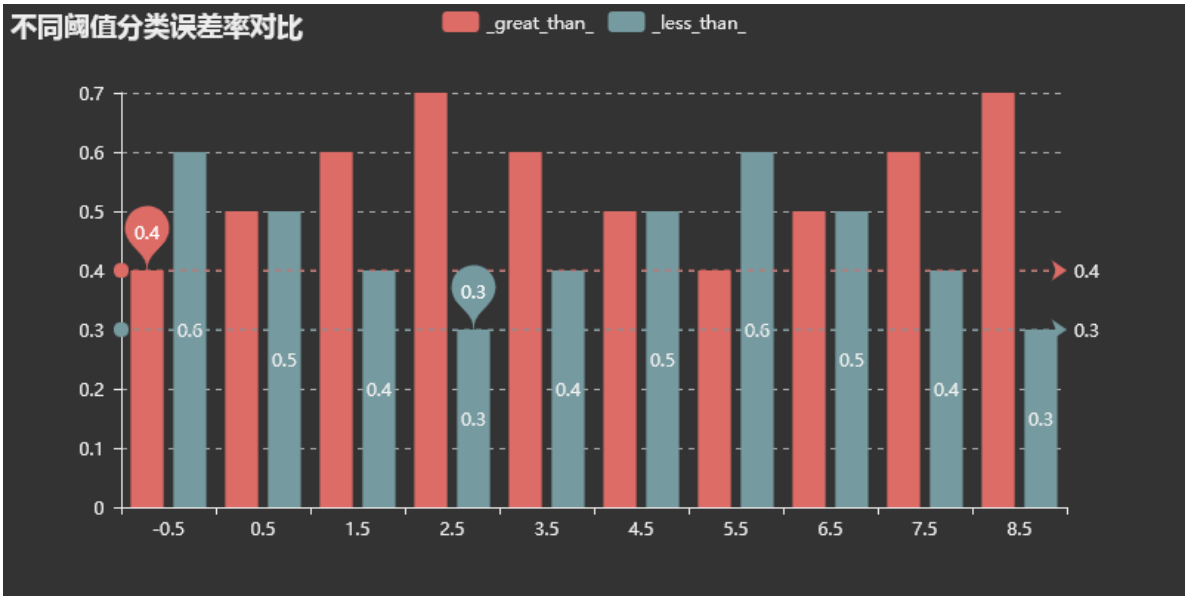
下面显示m=1，2，3时弱分类器的选择过程

m=1

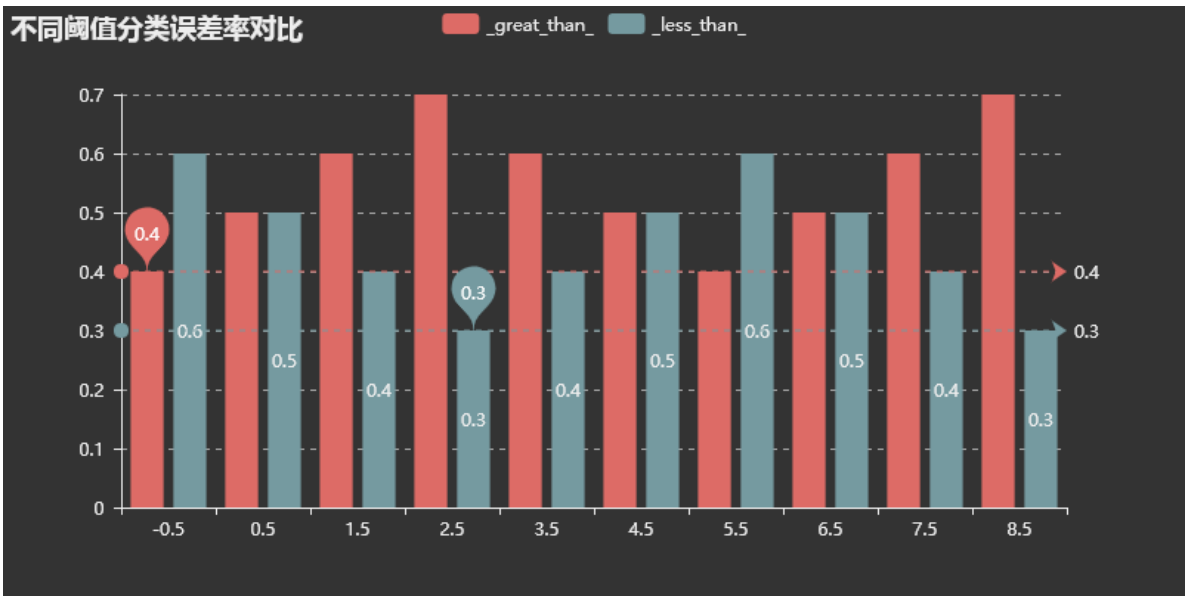

不同阈值分类误差率对比
_great_than_
_less_than_
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.5
0.5
1.5
2.5
3.5
4.5
5.5
6.5
7.5
8.5
0.4
0.3
0.4
0.6
0.5
0.4
0.3
0.4
0.5
0.6
0.5
0.4
0.3


sign_f1
0
10
20
30
40
50
60
70
80
90
100
70%

m=2

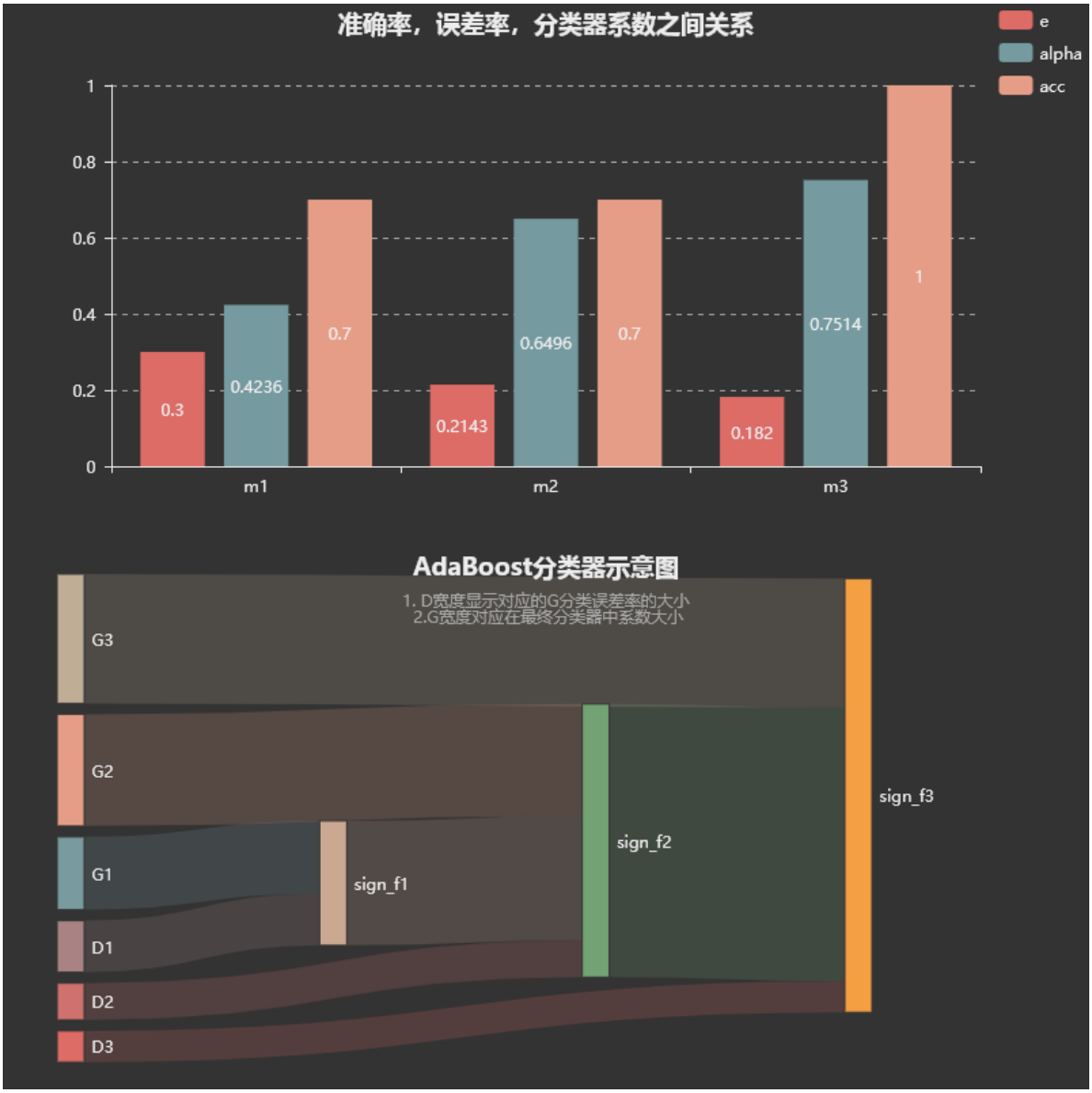

不同阈值分类误差率对比
_great_than_
_less_than_
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.5
0.5
1.5
2.5
3.5
4.5
5.5
6.5
7.5
8.5
0.4
0.3
0.4
0.6
0.5
0.4
0.3
0.4
0.5
0.6
0.5
0.4
0.3

m=3



数据显示了每一轮计算的结果

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
d1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
G1	1	1	1	-1	-1	-1	-1	-1	-1	-1
d2	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
f1	0.4236	0.4236	0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236
sign_f1	1.0	1.0	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
G2	1	1	1	1	1	1	1	1	1	-1
d3	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.0455
f2	1.0732	1.0732	1.0732	0.226	0.226	0.226	0.226	0.226	0.226	-1.0732
sign_f2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-1.0
G3	-1	-1	-1	-1	-1	-1	1	1	1	1
d4	0.125	0.125	0.125	0.1019	0.1019	0.1019	0.0648	0.0648	0.0648	0.125
f3	0.3218	0.3218	0.3218	-0.5254	-0.5254	-0.5254	0.9774	0.9774	0.9774	-0.3218
sign_f3	1.0	1.0	1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0



通过这个图来解释什么是加法模型

- 同样的数据集T，配合不同的权值分布，拿到不同的基分类器G
- 误差率的定义将权值系数分布与基分类器的结果联系在了一起
- 权值分布D的宽度代表分类器的*误差率*相对大小，D1→D3递减
- G的宽度代表最终模型中该分类器对应的系数大小，G1→G3递增
- 在模型的最终表示中有个∑

TODO：

增加不同轮次的样本权重的可视化。

AdaBoost 误差分析

这部分可以看下张潼老师的论文。其中提到这样一句， `The basic idea is to minimize a convex upper bound of the classification error function I(p,y).`

这样就自然的过度到了后面的AdaBoost的另外一种解释， 指数损失。

注意，张潼老师的论文里面提到了指示函数是error function。作为指示函数， 这里条件判断的是**是否相等**

书中的定理8.1如下描述

AdaBoost算法最终分类器的训练误差界为

$$\frac{1}{N}\sum_{i=1}^{N} I(G(x_i) \neq y_i) \leq \frac{1}{N}\sum_{i} \exp(-y_i f(x_i)) = \prod_{m} Z_m$$

这个的意思就是说指数损失是0-1损失的上界，然后通过递推得到了归一化系数的连乘。

定理8.2后面再看。

AdaBoost 算法的解释

加法模型， 指数损失， 前向分步， 二分类。

前向分步算法

算法8.2

输入：训练数据集$T=(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N),x_i\in\mathcal{X}\subseteq\mathcal{R}^n,y_i\in\{-1,1\}$， 损失函数$L(y,f(x))$; 基函数集合$\{b(x;\gamma)\}$

输出：加法模型$f(x)$

步骤：

1. 初始化$f_0(x)=0$
2. 对$m=1,2,\dots,M$
3. 极小化损失函数

$$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma))$$

4. 更新

$$f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$$

5. 得到加法模型

$$f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

提升树

提升方法实际采用加法模型（即基函数的线性组合）与前向分步算法。

提升树模型

以决策树为基函数的提升方法称为提升树。

决策树$T(x;\Theta_m)$

提升树模型可以表示成决策树的加法模型

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

上面这个公式用回归问题理解挺好理解(不等式求和)， 后面给了例子。

提升树算法

不同的问题， 主要区别在于损失函数不同：

1. 平方误差用于回归问题

2. 指数损失用于分类问题

算法8.3

回归问题的提升树算法

输入：训练数据集

输出：提升树$f_M(x)$

步骤：

1. 初始化$f_0(x)=0$
2. 对$m=1,2,\dots,M$
    1. 计算残差
    $$r_{mi}=y_i-f_{m-1}(x_i),i=1,2,\dots,N$$
    1. **拟合残差**r_{mi}学习一个回归树，得到$T(x;\Theta_m)$
    2. 更新$f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$
3. 得到回归问题提升树
$$f(x)=f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

5.5节中有回归树相关说明。

例子8.2

可以看代码中测试案例test_e82

梯度提升(GBDT)

算法8.4

输入： 训练数据集$T=(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N),x_i\in x\subseteq\mathbb{R}^n,y_i\in y\subseteq\mathbb{R}$；损失函数$L(y,f(x))$
输出：回归树$\hat{f}(x)$
步骤：

1. 初始化
$$f_0(x)=\arg\min_c\sum_{i=1}^{N}L(y_i,c)$$
2. $m=1,2,\dots,M$
3. $i=1,2,\dots,N$
$$r_{mi}=-\left[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$
4. 对r_{mi}拟合一个回归树，得到第m棵树的叶节点区域$R_{mj},j=1,2,\dots,J$
5. $j=1,2,\dots,J$
$$c_{mj}=\arg\min_c\sum_{xi\in R_{mj}}L(y_i,f_{m-1}(x_i)+c)$$
6. 更新

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$$

7. 得到回归树

$$\hat{f}(x) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$$

这个算法里面注意，关键是`用损失函数的负梯度，在当前模型的值作为回归问题提升树算法中的残差近似值，拟合回归树`

AdaBoost与SVM的关系

TODO: 训练数据集与测试数据集的误差率关系。

摘要文献中对AdaBoost和SVM的理解。

$|x|_1 = \sum_{i=1}^{N} |x|$，向量元素绝对值的和

$|x|_\infty = \max_i |x_i|$，向量所有元素绝对值中的最大值

$|x|_{-\infty} = \min_i |x_i|$，向量所有元素绝对值中的最小值

向量和矩阵的范数不同，具体可以参考numpy的帮助文档[6]

AdaBoost这个方法， 比较迷人的地方就在于训练数据集误差率降为0之后， 依然能继续降低测试误差，看起来，似乎不会过拟合。Schapire给出的解释主要是基于间隔理论， 但是， AdaBoost的间隔和SVM的间隔是不一样的。

关于AdaBoost的间隔理论， Schapire在1998年提出之后，受到过质疑，周志华老师在这个问题上给出了解释，并说明了当间隔分布无法继续提升的时候， 过拟合终将发生。

AdaBoost与LR的关系

第一次提到AdaBoost和LR 的关系是本书参考文献[6]， 给出了Boosting和LR损失函数之间的关系， 但是里面用到的损失小二乘。

本书的参考文献[9]，从Bregman散度的角度解释AdaBoost和LR的关系。

文献[9]中有说明，LR的特征对应了AdaBoost中的弱分类器，或者是基分类器，分类器对应了hypotheses。

习题

8.2

比较SVM，AdaBoost，LR的学习策略与算法

参考

1.
2.
3.
4.
5.
6.
7.
8.

t top

1. Generative and discriminative classifiers: Naive Bayes and logistic regression↩↩↩↩↩↩↩↩↩
2. Matrix Capsules with EM Routing↩↩↩↩↩↩↩↩↩↩
3. Machine Learning New Chapter↩↩↩↩↩
4. HUD4347↩↩↩↩
5. A Brief History of Classification and Regression Trees↩↩
6. numpy.linalg.norm↩↩
7. The Top Ten Algorithms in Data Mining↩↩
8. A decision-theoretic generalization of on-line learning and an application to boosting↩↩