



北京邮电大学

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

《机器学习》课程总结

姓名: 许金良

学院: 网络技术研究院

学号: 2014111469

授课教师: 杜军平教授

2015年1月5日

目录

第一章 简介	1
1.1 章节简介	1
1.2 本书来源	1
第二章 感知机模型介绍	3
2.1 感知机要解决的问题	3
2.2 感知机模型	3
2.3 感知机学习策略	4
2.4 优化算法	5
2.4.1 随机梯度下降法与标准梯度下降	5
2.4.2 感知机优化算法及其实现	5
2.5 感知机优化算法的收敛性证明	6
2.6 感知机学习优化算法的对偶形式	7
第三章 LDA,SVD和PCA	15
3.1 矩阵工具	15
3.1.1 协方差矩阵	15
3.1.2 特征值分解	16
3.1.3 实对称矩阵性质	17
3.2 LDA	18
3.2.1 二类问题	18
3.2.2 多类问题	20
3.2.3 LDA的局限	21
3.3 PCA	21
3.3.1 PCA算法	21
3.3.2 最大化方差解释PCA	22
3.3.3 最小化损失解释PCA	24
3.4 SVD	26
3.4.1 几何方法导出SVD	26
3.4.2 SVD剖析	28
第四章 Naive Bayes	35
4.1 Naive Bayes基本方法	35

4.2	极大似然估计参数	36
4.3	后验概率与期望风险的关系	36
4.4	单调递增变换与线性求和	37
4.5	独立假设的利弊及扩展	39
4.5.1	为什么独立性假设可行	39
4.5.2	收缩系数的引入	39
4.6	拉普拉斯平滑	41
4.6.1	先验概率修正	42
4.6.2	条件概率修正	42
第五章	PageRank算法	45
5.1	PageRank算法	45
5.2	PageRank的扩展: Timed-PageRank	50
5.3	Perron - Frobenius 定理	50

第一章 简介

1.1 章节简介

第一章介绍了感知机模型的数学原理、模型推导和算法，并附有一个Python实现的例子。感知机模型是神经网络模型和支持向量机模型的基础。第二章介绍了线性判别模型(LDA)、奇异值分解(SVD)和主成分分析模型(PCA)，这三种模型往往用于数据预处理的降维过程。第三章介绍了朴素贝叶斯模型，这是一个概率模型，具有模型简单、效果显著，常用语文本分类等等。第四章介绍了网页排名(PageRank)模型的数学原理。

1.2 本书来源

这四章内容（感知机模型，LDA、SVD、PCA模型，朴素贝叶斯模型，PageRank模型）分别来自于本学期我在CSDN上写的四篇博客，它们的博文名字、网址和发表时间如下表所示

表 1.1: 四篇博客的博文名字、网址和发表时间表

名字	网址	发表时间
LDA_PCA_SVD导论	http://blog.csdn.net/u012176591/article/details/41923105	2014-12-14
感知机导论	http://blog.csdn.net/u012176591/article/details/41792857	2014-12-07
Naive Bayes导论	http://blog.csdn.net/u012176591/article/details/41681907	2014-12-02
PageRank背后的数学	http://blog.csdn.net/u012176591/article/details/41597605	2014-11-29

我的CSDN博客的名字是“金良山庄”，取自于我本人的名字，意为“与天下志同道合的人分享学习的过程”，地址<http://blog.csdn.net/u012176591/article>。我自去年3月到4月之间做本科毕设时开的博客，最初是以做笔记的形式写博客的，至今已累计188篇博文，虽然都比较粗陋，毕竟还只是一个好的开始，今天我的博客访问量已达80453人次。以下是我的博客的截图



图 1.1: 我的CSDN博客示意图

第二章 感知机模型介绍

2.1 感知机要解决的问题

感知机(perception)是线性可分二分类模型，其输入为实例的特征向量，输出是实例的类别标签，取+1和-1值。感知机学习旨在求出能够将训练数据进行线性划分的分离超平面，由于训练数据本来就线性可分，所以感知机总是能够找到可行解。理论上，在不同的初始值或训练参数下，感知机模型可以得到无数个可行解。感知机模型是神经网络(ANN) 和支持向量机(SVM)的基础。

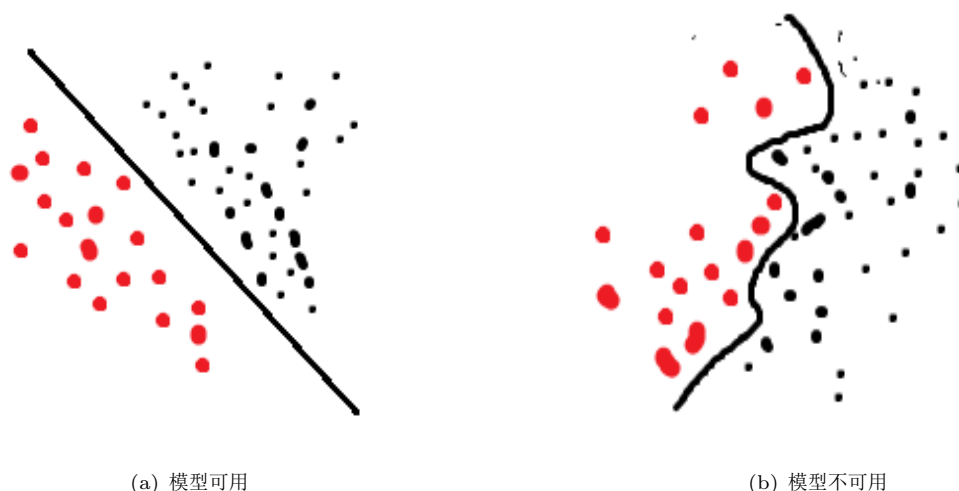


图 2.1: 感知机模型是线性二分类模型，故只能用于线性可分的数据(image 4.1(a))，而不能用于线性不可分的数据(image 4.1(b))。当训练数据不可分时，感知机模型学习过程不收敛，迭代结果会发生震荡。

2.2 感知机模型

定义 1 感知机模型 假设输入空间(特征空间)是 $\chi \subseteq \mathcal{R}^n$ ，输出空间是 $\mathcal{Y} = \{+1, -1\}$ 。输入 $x \in \chi$ 表示实例的特征向量，对应于输入空间(特征空间)的一个点；输入 $y \in \mathcal{Y}$ 表示实例的类别。由输入空间到输出空间的如下函数



$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \quad (2.2.1)$$

称为感知机。其中， \mathbf{w} 和 b 为感知机模型参数， $\mathbf{w} \in \mathcal{R}^n$ 叫作权值向量(weight vector)， $b \in \mathcal{R}$ 叫作偏置(bias)。

感知机模型的假设空间是定义在特征空间中的所有线性分类模型(linear classification model)或线性分类器(linear classifier)，即函数集合 $\{f | f(x) = \mathbf{w} \cdot \mathbf{x} + b\}$ 。

感知机的几何解释：线性方程

$$\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = 0 \quad (2.2.2)$$

对应于特征空间 \mathcal{R}^n 中的一个超平面 S ,其中 \mathbf{w} 是超平面的法向量, b 是超平面的截距。这个超平面将特征空间划分为两部分, 分别位于这两部分的点(特征向量)集被分为正类(有 $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} > 0 \Rightarrow \mathbf{f}(\mathbf{x}) = 1$)和负类(有 $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} < 0 \Rightarrow \mathbf{f}(\mathbf{x}) = -1$)。因此, 超平面 S 称为分离超平面(separating hyperplane)。如图(2.2)所示, 分别标记为  和  的两类实例被分离超平面区分开, 分别位于分离超平面上下两侧。

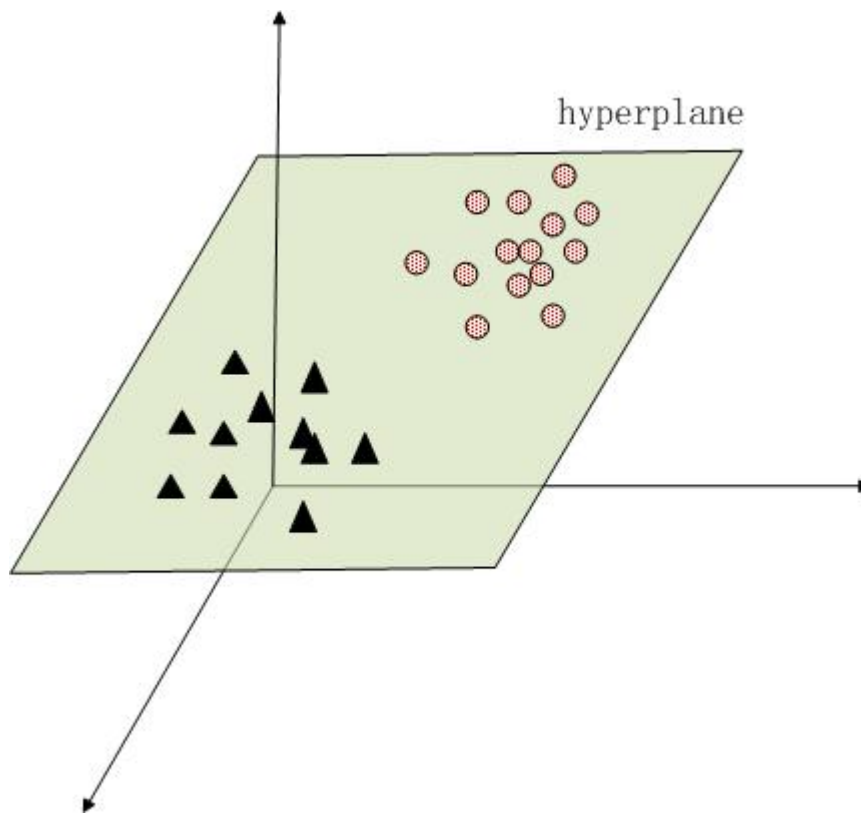


图 2.2: 三维空间的分离超平面示意图。不同标记的两类点集被分离超平面分开。

感知机的学习, 就是根据数据集 $T = \langle (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N) \rangle$ 逐步修正、获得模型参数 \mathbf{w} 和 b 的过程。而感知机的预CE, 就是通过学习得到的感知机模型, 对新的实例 \mathbf{x} 确定其类别标签 y 。

2.3 感知机学习策略

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。

定义 2 损失函数 假设超平面 S 的误分类点集合为 M , 那么感知机的损失函数定义为

$$L(\mathbf{w}, b) = - \sum_{\mathbf{x}_i \in M} y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \quad (2.3.1)$$

其实 $L(\mathbf{w}, b)$ 就是所有误分类点到分离超平面的函数距离之和。

显然, 损失函数 $L(\mathbf{w}, b)$ 是非负的。更重要的是, 损失函数 $L(\mathbf{w}, b)$ 是 \mathbf{w} , b 的连续可导函数。

2.4 优化算法

感知机学习算法是误分类驱动的，具体采用随机梯度下降法 (stochastic gradient descent)。

2.4.1 随机梯度下降法与标准梯度下降

如果采用标准梯度下降法，根据损失函数式(2.3.1)，那么每次迭代就要求如下的梯度

$$\begin{aligned}\nabla_{\mathbf{w}}(\mathbf{w}, b) &= - \sum_{\mathbf{x}_i \in M} y_i \mathbf{x}_i \\ \nabla_b(\mathbf{w}, b) &= - \sum_{\mathbf{x}_i \in M} y_i\end{aligned}\tag{2.4.1}$$

由式(2.4.1)可知，标准梯度下降法在每次迭代都要用到 M 的所有数据，当 $|M|$ 很大时，复杂度可想而知。

如果采用随机梯度下降法，每次只需要随机取一个误分类点 $(\mathbf{x}_i, y_i) \in M$ 并仅用它求梯度

$$\begin{aligned}\nabla_{\mathbf{w}}(\mathbf{w}, b) &= -y_i \mathbf{x}_i \\ \nabla_b(\mathbf{w}, b) &= -y_i\end{aligned}\tag{2.4.2}$$

进而对参数进行更新

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i \\ b &\leftarrow b + \eta y_i\end{aligned}\tag{2.4.3}$$

当学习率(learning rate) η 足够小时，随机梯度下降法可以任意逼近标准梯度下降法。虽然不是每次迭代得到的损失函数都向着全局最优方向，但是大的整体的方向是向全局最优解的，最终的结果往往是在全局最优解附近。

2.4.2 感知机优化算法及其实现

感知机优化算法伪代码如算法1所示。

Algorithm 1 The perceptron learning algorithm.

Require: training dataset $T = \langle (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N) \rangle$ where $\mathbf{x} \in \mathcal{R}^n, y \in \{+1, -1\}$; learning rate $\eta (0 < \eta \leq 1)$

Output: \mathbf{w}, b

```

1:  $\mathbf{w} \leftarrow \mathbf{w}_0$ 
2:  $b \leftarrow b_0$ 
3: repeat
4:   randomly select a  $(\mathbf{x}_i, y_i)$  such that  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \leq 0$ 
5:    $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$ ;  $b \leftarrow b + \eta y_i$ 
6: until  $\forall t_i \in T, y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0$ 
```

例 1 如表(3.1)的数据，feature vector 一栏为特征向量，label 一栏为类标签，一共10组数据。求其感知机模型 $f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b)$ (设初值 $\mathbf{w} = (0, 0)^T, b = 0$ ，学习率 $\eta = 1$)。

例 1 的数据画在二维坐标上的示意图如(??)所示。

表 2.1: 变量或函数定义

feature vector	label	feature vector	label
(0.5, 1.9)	-1	(2, 3)	1
(2, 2)	1	(1, 1)	-1
(1, 2)	1	(2, 0.5)	-1
(3, 2)	1	(1.5, 0.6)	-1
(3, 1.5)	1	(2.5, 1)	1

2.5 感知机优化算法的收敛性证明

感知机算法的收敛性，也就是算法经过有限次迭代可以得到一个将训练数据集完全正确划分的分离超平面，即感知机模型。

我们在定义(1)所涉及到的变量定义的基础上定义新的变量

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{w}^T, b)^T \\ \hat{\mathbf{x}} &= (\mathbf{x}^T, 1)^T\end{aligned}\tag{2.5.1}$$

这样， $\hat{\mathbf{x}} \in \mathcal{R}^{n+1}$, $\hat{\mathbf{w}} \in \mathcal{R}^{n+1}$ 。显然， $\hat{\mathbf{w}} \cdot \hat{\mathbf{x}} = \mathbf{w} \cdot \mathbf{x} + b$ 。我们将算法(1)用扩充后的向量 $\hat{\mathbf{w}}$ 和 $\hat{\mathbf{x}}$ 重写一遍，如算法(2)。

Algorithm 2 The perceptron learning algorithm using augment vector.

Require: training dataset $T = \langle (\hat{\mathbf{x}}_1, y_1), (\hat{\mathbf{x}}_2, y_2), \dots, (\hat{\mathbf{x}}_N, y_N) \rangle$ where $\hat{\mathbf{x}} \in \mathcal{R}^{n+1}, y \in \{+1, -1\}$; learning rate $\eta (0 < \eta \leq 1)$

Output: $\hat{\mathbf{w}} \in \mathcal{R}^{n+1}$

```

1:  $\hat{\mathbf{w}} \leftarrow \mathbf{0}$ 
2: repeat
3:   randomly select a  $(\hat{\mathbf{x}}_i, y_i)$  such that  $y_i(\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}_i) \leq 0$ 
4:    $\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} + \eta y_i \hat{\mathbf{x}}_i$ 
5: until  $\forall t_i \in T, y_i(\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}_i) > 0$ 
```

定理 1 (Novikoff) Assume that there exists some parameter vector $\hat{\mathbf{w}}^*$ such that $\|\hat{\mathbf{w}}^*\| = 1$, and some $\gamma > 0$ such that for all $i = 1, 2, \dots, n$,

$$y_i(\hat{\mathbf{w}}^* \cdot \hat{\mathbf{x}}_i) \geq \gamma$$

Assume in addition that for all $t = 1, 2, \dots, n, \|\hat{\mathbf{x}}_i\| \leq \mathcal{R}$, then the perceptron algorithm makes at most

$$\frac{\mathcal{R}^2}{\gamma^2}$$

errors. (The definition of an error is as follows: an error occurs whenever we have $y_i(\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}_i) \leq 0$ for the $(\hat{\mathbf{x}}_i, y_i)$ pair we select).

Note that for any vector $\hat{\mathbf{x}}$, we use $\|\hat{\mathbf{x}}\|$ to refer to the Euclidean norm of $\hat{\mathbf{x}}$, i.e., $\|\hat{\mathbf{x}}\| = \sqrt{\sum_j (\hat{x}^{(j)})^2}$.

Proof. First, define $\hat{\mathbf{w}}$ to be the parameter vector when the algorithm makes its k 'th error. Note that we have

$$\hat{\mathbf{w}}_1 = \mathbf{0}$$

Next, assuming the k 'th error is made on example i , we have

$$\hat{\mathbf{w}}_{k+1} \cdot \hat{\mathbf{w}}^* = (\hat{\mathbf{w}}_k + \eta y_i \hat{\mathbf{x}}_i) \cdot \hat{\mathbf{w}}^* \quad (2.5.2)$$

$$= \hat{\mathbf{w}}_k \cdot \hat{\mathbf{w}}^* + \eta y_i \hat{\mathbf{x}}_i \cdot \hat{\mathbf{w}}^* \quad (2.5.3)$$

$$\geq \hat{\mathbf{w}}_k \cdot \hat{\mathbf{w}}^* + \eta \gamma \quad (2.5.4)$$

Eq. 2.5.2 follows by the definition of the perceptron updates. **Eq. 2.5.4** follows because by the assumptions of the theorem, we have

$$y_i \hat{\mathbf{x}}_i \cdot \hat{\mathbf{w}}^* \geq \gamma$$

It follows by induction on k (recall that $\|\hat{\mathbf{w}}_1\| = 0$), that

$$\hat{\mathbf{w}}_{k+1} \cdot \hat{\mathbf{w}}^* \geq k\eta\gamma$$

In addition, because $\|\hat{\mathbf{w}}_{k+1}\| = \|\hat{\mathbf{w}}_{k+1}\| \cdot \|\hat{\mathbf{w}}^*\| \geq \hat{\mathbf{w}}_{k+1} \cdot \hat{\mathbf{w}}^*$, we have

$$\|\hat{\mathbf{w}}_{k+1}\| \geq k\eta\gamma \quad (2.5.5)$$

In the second part of the proof, we will derive an upper bound on $\|\hat{\mathbf{w}}_{k+1}\|$. We have

$$\|\hat{\mathbf{w}}_{k+1}\|^2 = \|\hat{\mathbf{w}}_k + y_i \hat{\mathbf{x}}_i\|^2 \quad (2.5.6)$$

$$= \|\hat{\mathbf{w}}_k\|^2 + y_i^2 \|\hat{\mathbf{x}}_i\|^2 + 2y_i \hat{\mathbf{x}}_i \cdot \hat{\mathbf{w}}_k \quad (2.5.7)$$

$$\leq \|\hat{\mathbf{w}}_k\|^2 + \mathcal{R}^2 \quad (2.5.8)$$

The equality in **Eq. 2.5.6** follows by the definition of the perceptron updates. **Eq. 2.5.8** follows because we have 1) $y_i^2 \|\hat{\mathbf{x}}_i\|^2 = \|\hat{\mathbf{x}}_i\|^2 \leq \mathcal{R}^2$ by the assumptions of theorem, and because $y_i^2 = 1$; 2) $y_i \hat{\mathbf{x}}_i \cdot \hat{\mathbf{w}}_k \leq 0$ because we know that the parameter vector $\hat{\mathbf{w}}_k$ gave an error on the i 'th example. It follows by induction on k (recall that $\|\hat{\mathbf{w}}_1\| = 0$), that

$$\|\hat{\mathbf{w}}_{k+1}\|^2 \leq k\mathcal{R}^2 \quad (2.5.9)$$

Combining the bounds in **Eq. 2.5.5** and **Eq. 2.5.9** gives

$$k^2\gamma^2 \leq \|\hat{\mathbf{w}}_{k+1}\|^2 \leq k\mathcal{R}^2$$

from which it follows that

$$k \leq \frac{\mathcal{R}^2}{\gamma^2} \quad (2.5.10)$$

□

2.6 感知机学习优化算法的对偶形式

在这一节我们仍沿用式(2.5.1)定义的扩充的向量 $\hat{\mathbf{x}} \in \mathcal{R}^{n+1}$, $\hat{\mathbf{w}} \in \mathcal{R}^{n+1}$. 算法2 设初值 $\hat{\mathbf{w}} = \mathbf{0}$, 对误分类点 $(\hat{\mathbf{x}}_i, y_i)$ 通过

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} + \eta y_i \hat{\mathbf{x}}_i$$

逐步修改 $\hat{\mathbf{w}}$ ，设算法2从开始执行到执行结束对第 i 个实例 $(\hat{\mathbf{x}}_i, y_i)$ 一共修改了 n_i 次，则最终的 $\hat{\mathbf{w}}$ 可以表示为

$$\hat{\mathbf{w}} = \sum_{i=1}^N \eta n_i y_i \hat{\mathbf{x}}_i = \sum_{i=1}^N \alpha_i y_i \hat{\mathbf{x}}_i \quad (2.6.1)$$

其中 $\alpha_i = \eta n_i$ 。

则有

$$\langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle = \left\langle \sum_{i=1}^N \alpha_i y_i \hat{\mathbf{x}}_i, \hat{\mathbf{x}} \right\rangle = \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}} \rangle \quad (2.6.2)$$

好了，下面我们直接给出感知机算法的对偶形式：

Algorithm 3 The perceptron learning algorithm using augment vector.

Require: training dataset $T = \langle (\hat{\mathbf{x}}_1, y_1), (\hat{\mathbf{x}}_2, y_2), \dots, (\hat{\mathbf{x}}_N, y_N) \rangle$ where $\hat{\mathbf{x}} \in \mathcal{R}^{n+1}, y \in \{+1, -1\}$; learning rate $\eta (0 < \eta \leq 1)$

Output: $\alpha_i, i = 1, 2, \dots, N$

1: $\alpha_i \leftarrow 0, i = 1, 2, \dots, N$

2: **repeat**

3: randomly select a $(\hat{\mathbf{x}}_j, y_j)$ such that $y_j \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \leq 0$ and update $\alpha_i = \alpha_i + \eta y_j y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle$ for all $i = 1, 2, \dots, N$

4: **until** $\forall t_j \in T, y_j \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle > 0$

从算法3可以看到，经过这样的变换（对偶变换）后，感知机的学习已经完全与权值向量 $\hat{\mathbf{w}}$ 没有关系了，变成了求解 $\alpha_i, i = 1, 2, \dots, N$ 。感知机的对偶形式算法训练中只用到了训练实例的特征向量两两之间的内积，而没有用到具体的特征向量。为了提高训练速度，可以预先将训练集实例间的内积计算出来并以 $N \times N$ 的矩阵形式存储，这个矩阵就是所谓的Gram 矩阵(Gram matrix)

$$\mathbf{G} = [\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle]_{N \times N} \quad (2.6.3)$$

例 2 用感知机学习算法的对偶性时求感知机模型，正样本点是 $\mathbf{x}_1 = (3, 3)^T, \mathbf{x}_2 = (4, 3)^T$ ，负样本点 $\mathbf{x}_3 = (1, 1)^T, \mathbf{x}_4 = (2, 1)^T$ 。

样本点的扩展特征向量(式(2.5.1))是 $\hat{\mathbf{x}}_1 = (3, 3, 1)^T, \hat{\mathbf{x}}_2 = (4, 3, 1)^T, \hat{\mathbf{x}}_3 = (1, 1, 1)^T, \hat{\mathbf{x}}_4 = (2, 1, 1)^T$ ，其Gram 矩阵

$$\mathbf{G} = \begin{bmatrix} \langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_1 \rangle & \langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle & \langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_3 \rangle & \langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_4 \rangle \\ \langle \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_1 \rangle & \langle \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_2 \rangle & \langle \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3 \rangle & \langle \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_4 \rangle \\ \langle \hat{\mathbf{x}}_3, \hat{\mathbf{x}}_1 \rangle & \langle \hat{\mathbf{x}}_3, \hat{\mathbf{x}}_2 \rangle & \langle \hat{\mathbf{x}}_3, \hat{\mathbf{x}}_3 \rangle & \langle \hat{\mathbf{x}}_3, \hat{\mathbf{x}}_4 \rangle \\ \langle \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_1 \rangle & \langle \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_2 \rangle & \langle \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_3 \rangle & \langle \hat{\mathbf{x}}_4, \hat{\mathbf{x}}_4 \rangle \end{bmatrix} = \begin{bmatrix} 19 & 22 & 7 & 10 \\ 22 & 26 & 8 & 12 \\ 7 & 8 & 3 & 4 \\ 10 & 12 & 4 & 6 \end{bmatrix}$$

令 $\mathbf{y} = (y_1, y_2, y_3, y_4) = (1, 1, -1, -1)$ ， $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ ，则表达式

$$\boldsymbol{\alpha} * \mathbf{y} \times \mathbf{G} * \mathbf{y} = \left(y_1 \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_1 \rangle, y_2 \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_2 \rangle, y_3 \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_3 \rangle, y_4 \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_4 \rangle \right) \quad (2.6.4)$$

的元素就是算法(3)中用来判断 $t_j \in T$ 是否为误分类的表达式 $y_j \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle$ 。式(2.6.4)中的运算符 $*$ 表示元素乘，运算符 \times 表示矩阵乘。

按照随机梯度下降的思路，假设某次迭代我们抽取到误分类的点是 $(\hat{\mathbf{x}}_j, y_j)$ ，则损失函数则我们对其代表的损失函数 $L(\boldsymbol{\alpha}, j) = -y_j \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle$ 求其关于 $\boldsymbol{\alpha}$ 的导数

$$\begin{aligned} \frac{\partial L(\boldsymbol{\alpha}, j)}{\partial \boldsymbol{\alpha}} &= \frac{-\partial y_j \sum_{i=1}^N \alpha_i y_i \langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle}{\partial \boldsymbol{\alpha}} \\ &= -y_j \mathbf{y} * G[j] \end{aligned} \quad (2.6.5)$$

然后更新 $\boldsymbol{\alpha}$ 向量

$$\boldsymbol{\alpha} = \boldsymbol{\alpha} - \eta \frac{\partial L(\boldsymbol{\alpha}, j)}{\partial \boldsymbol{\alpha}} \quad (2.6.6)$$

perception.py

```

1 import numpy
2 import random
3 import matplotlib.pyplot as pyplot
4 def myplot(x,y,omega,b,iter):
5     for i in range(x.shape[0]):
6         if y[i]==-1:
7             pyplot.plot(x[i][0],x[i][1], 'mo')
8         else:
9             pyplot.plot(x[i][0],x[i][1], 'k*')
10    if (omega[1]==0.0)&(omega[0]==0.0):
11        return
12    elif omega[1]==0:
13        xline=[-b/omega[0]]*100
14        yline =numpy.linspace(0,5,100)
15    elif omega[0]==0:
16        xline =numpy.linspace(0,5,100)
17        yline=[-b/omega[1]]*100
18    else:
19        xline=numpy.linspace(0,5,100)
20        yline=(-b-omega[0]*xline)/omega[1]
21    pyplot.plot(xline,yline, 'b-')
22    pyplot.xlim(0,4);    pyplot.ylim(0,4)
23    pyplot.xlabel('x(1)');    pyplot.ylabel('x(2)')
24    pyplot.title('The perception example')
25    pyplot.savefig('./pic/perception1/'+str(iter)+'.jpg');
26    pyplot.close('all')#close the picture
27 x=numpy.array([[0.5,1.9],[2,3],[2,2],[1,1],[1,2],[2,0.5],[3,2],[1.5,0.6],[3,1.5],[2.5,1]])
28 y=numpy.array([-1,1,1,-1,1,-1,1,-1,1,1])
29 omega=numpy.array([0,0]);b=0;eta=0.03145 errorset=[]
30 for i in range(x.shape[0]):
31     if ((x[i].dot(omega)+b)*y[i]<=0):
32         errorset.append(i)
33 iter=0
34 while len(errorset)>0:
35     errornum=len(errorset)
36     num=random.randrange(0,errornum)
37     loc=errorset[num]
38     omega =omega+eta*y[loc]*x[loc]
39     b =b+eta*y[loc]
40     print eta*y[loc]*x[loc],y[loc],x[loc],omega,b
41     myplot(x,y,omega,b,iter)
42     iter+=1
43     errorset=[]
44     for i in range(x.shape[0]):
45         if ((x[i].dot(omega)+b)*y[i]<=0):
46             errorset.append(i)

```

perception.py

```
1  #-*- coding: cp936 -*-
2  import numpy
3  import random
4  import matplotlib.pyplot as pyplot
5
6  def add(x):
7      temp=[]
8      for i in range(x.shape[0]):
9          a=x[i].tolist()
10         a.append(1)
11         temp.append(a)
12     x=numpy.array(temp)
13     return x
14
15 def Gram(x):
16     G=numpy.zeros([x.shape[0],x.shape[0]])
17     for i in range(x.shape[0]):
18         for j in range(i+1):
19             G[i][j]=G[j][i]=x[i].dot(x[j])
20     return G
21
22 def Main():
23     x=numpy.array([[3,3],[4,3],[1,1],[2,1]])
24     y=numpy.array([1,1,-1,-1])
25     x=add(x)
26     G=Gram(x)
27     alpha=numpy.zeros([1,x.shape[0]])
28     eta=0.31415
29     indic=alpha*y.dot(G)*y
30     errorset=[]
31     for i in range(indic.shape[1]):
32         if indic[0][i]<=0:
33             errorset.append(i)
34     while len(errorset)>0:
35         num=random.randrange(0,len(errorset))
36         i=errorset[num]
37         alpha=alpha+eta*y[i]*y*G[i]
38         print alpha
39         indic=alpha*y.dot(G)*y
40         errorset=[]
41         for i in range(indic.shape[1]):
42             if indic[0][i]<=0:
43                 errorset.append(i)
44     print (alpha*y).dot(x)
45 Main()
```


参考文献

- [1] 李航, 统计学习方法. 清华大学出版社, 北京, 2014.
- [2] OldPanda, 《统计学习方法》读书笔记——感知机, <http://www.cnblogs.com/OldPanda/archive/2013/04/12/3017100.html>, 4/12 2013.
- [3] Michael Collins, Convergence Proof for the Perceptron Algorithm, <http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf>, 2012.

第三章 LDA,SVD和PCA

3.1 矩阵工具

3.1.1 协方差矩阵

假如我们从某个实验中得到相同类型的 N 个记录 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ ，每条记录都是 M 维的，依次标记为 $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_M$ ，记录的数据直观地用表3.1表示

表 3.1: 数据表。 \mathbf{x} 表示一条记录， \mathbf{c} 表示一个维度

	\mathbf{c}_1	\mathbf{c}_2	\dots	\mathbf{c}_M
\mathbf{x}_1	x_{11}	x_{12}	\dots	x_{1M}
\mathbf{x}_2	x_{21}	x_{22}	\dots	x_{2M}
\mathbf{x}_3	x_{31}	x_{32}	\dots	x_{3M}
\vdots	\vdots	\vdots	\ddots	\vdots
\mathbf{x}_N	x_{N1}	x_{N2}	\dots	x_{NM}

根据表3.1，我们规定第 i 条记录表示为列向量 $\mathbf{x}_n = (x_{n1}, x_{n2}, \dots, x_{nM})^T$ ，所有记录的第 m 维的数据表示为有序列向量 $\mathbf{c}_m = (x_{1m}, x_{2m}, x_{3m}, \dots, x_{Nm})^T$ 。另外我们用列向量 $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$ 表示由 N 条记录所得到的各个维度的均值组成的向量，可以看出 $\boldsymbol{\mu}$ 的维度也是 M 。

现在我们开始讨论表3.1记录的数据的协方差矩阵 $\boldsymbol{\Sigma}$ 。首先可以明确的一点是 $\boldsymbol{\Sigma}$ 为 M 阶对称的方阵，因为它描述的是维度两两之间（当然也包括与本身，体现在协方差矩阵的对角线上）的协方差，或者相关程度。它有以下两种表示方法

$$1. \boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$$

$$2. \boldsymbol{\Sigma} = \frac{1}{N} \begin{bmatrix} (\mathbf{c}_1 - \boldsymbol{\mu}_1)^T (\mathbf{c}_1 - \boldsymbol{\mu}_1) & (\mathbf{c}_1 - \boldsymbol{\mu}_1)^T (\mathbf{c}_2 - \boldsymbol{\mu}_2) & \dots & (\mathbf{c}_1 - \boldsymbol{\mu}_1)^T (\mathbf{c}_M - \boldsymbol{\mu}_M) \\ (\mathbf{c}_2 - \boldsymbol{\mu}_2)^T (\mathbf{c}_1 - \boldsymbol{\mu}_1) & (\mathbf{c}_2 - \boldsymbol{\mu}_2)^T (\mathbf{c}_2 - \boldsymbol{\mu}_2) & \dots & (\mathbf{c}_2 - \boldsymbol{\mu}_2)^T (\mathbf{c}_M - \boldsymbol{\mu}_M) \\ (\mathbf{c}_3 - \boldsymbol{\mu}_3)^T (\mathbf{c}_1 - \boldsymbol{\mu}_1) & (\mathbf{c}_3 - \boldsymbol{\mu}_3)^T (\mathbf{c}_2 - \boldsymbol{\mu}_2) & \dots & (\mathbf{c}_3 - \boldsymbol{\mu}_3)^T (\mathbf{c}_M - \boldsymbol{\mu}_M) \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{c}_M - \boldsymbol{\mu}_M)^T (\mathbf{c}_1 - \boldsymbol{\mu}_1) & (\mathbf{c}_M - \boldsymbol{\mu}_M)^T (\mathbf{c}_2 - \boldsymbol{\mu}_2) & \dots & (\mathbf{c}_M - \boldsymbol{\mu}_M)^T (\mathbf{c}_M - \boldsymbol{\mu}_M) \end{bmatrix}$$

下面证明这两种表示方法是等效的：

Proof. :

Step 1 方法1表示的方差的第*i*行、第*j*列的元素

$$\begin{aligned}\Sigma_{ij} &= \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})_i (\mathbf{x}_n - \boldsymbol{\mu})_j \\ &= \frac{1}{N} \sum_{n=1}^N (x_{ni} - \mu_i) (x_{nj} - \mu_j)\end{aligned}\quad (3.1.1)$$

Step 2 方法2表示的方差的第*i*行、第*j*列的元素

$$\begin{aligned}\Sigma_{ij} &= \frac{1}{N} \sum_{n=1}^N \langle \mathbf{c}_i - \mu_i, \mathbf{c}_j - \mu_j \rangle \\ &= \frac{1}{N} \sum_{n=1}^N (x_{ni} - \mu_i) (x_{nj} - \mu_j)\end{aligned}\quad (3.1.2)$$

Step 3 由式4.4.1和式4.4.2知二者等效，都可以用来计算协方差矩阵。

□

3.1.2 特征值分解

给定*N*阶方阵 \mathbf{A} ，则*N*维列向量 $\boldsymbol{\nu}$ 是 \mathbf{A} 的特征向量，当且仅当下式成立

$$\mathbf{A}\boldsymbol{\nu} = \lambda\boldsymbol{\nu}\quad (3.1.3)$$

其中 λ 是标量，是特征向量 $\boldsymbol{\nu}$ 对应的特征值。

假设*N*阶方阵 \mathbf{A} 有*N*个线性无关的特征向量 $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \dots, \boldsymbol{\nu}_N$ ，分别对应特征值 $\lambda_1, \lambda_2, \dots, \lambda_N$ ，则每个特征向量—特征值对 $\{\boldsymbol{\nu}_i, \lambda_i\}, i = 1, 2, \dots, N$ 都满足式3.1.3。

构造*N*阶对角矩阵 $\boldsymbol{\Lambda}$ ，满足

$$\Lambda_{ij} = \begin{cases} \lambda_i & i = j \\ 0 & otherwise \end{cases}$$

构造*N*阶方阵 \mathbf{Q} ，满足

$$\mathbf{Q} = (\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \dots, \boldsymbol{\nu}_N)$$

则矩阵 $\mathbf{A}, \mathbf{Q}, \boldsymbol{\Lambda}$ 满足关系

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}\boldsymbol{\Lambda}\quad (3.1.4)$$

进而得到

$$\mathbf{A} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1}\quad (3.1.5)$$

式3.1.5就是特征值分解，又叫谱分解（Spectral decomposition）。

对特征值和特征向量的一种解释是：*N*阶方阵 \mathbf{A} 可以看成是一个拉伸变换，其*N*个特征向量可以看成是*N*个拉伸方向。需要说明的是这*N*个方向的拉伸幅度是不相同的，这个度量由各个特征向量对应的特征值表示。具体的，特征值越大，表示在该方向的拉伸幅度越大，在整个拉伸变换中它的权重越大。

如果我们选取最大的*k*个特征值组成*k*阶对角阵 $\boldsymbol{\Lambda}_k$ ，用对应的*k*个特征向量组成 $N \times k$ 的矩阵 \mathbf{Q}_k ，仿照式3.1.4,得到

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\boldsymbol{\Lambda}_k\quad (3.1.6)$$

观察式3.1.6，左边的运算的实际意义是将矩阵 \mathbf{A} 投射到权重最大的*k*个拉伸方向上。令 $\mathbf{A}' = \mathbf{A}\mathbf{Q}_k$ ，对比 \mathbf{A}' （每行用*k*维向量表示）和 \mathbf{A} （每行用*N*维向量表示），可以发现实现了降维的目的。

3.1.3 实对称矩阵性质

协方差矩阵是实对称矩阵，所以我们有必要给出实对称矩阵的几条性质。首先介绍一个关系式：考虑 $N \times N$ 阶对称矩阵 \mathbf{A} ，它也可以视为定义于 \mathcal{R}^N 域的线性算子，即 $A: \mathcal{R}^n \rightarrow \mathcal{R}^n$ 。因为 $\mathbf{A}^H = \mathbf{A}$ ，则对于 \mathcal{R}^N 中任意向量 \mathbf{x} 和 \mathbf{y} ，有

$$(\mathbf{A}\mathbf{x})^H \mathbf{y} = \mathbf{x}^H \mathbf{A}^H \mathbf{y} = \mathbf{x}^H (\mathbf{A}\mathbf{y}) \quad (3.1.7)$$

定理 2 (实对称矩阵的特征值是实数，其对应的特征向量为实向量)

令 \mathbf{A} 为实对称矩阵， λ 为其特征值， $\boldsymbol{\nu}$ 为该特征值对应的特征向量，我们要证明 $\lambda \in \mathcal{R}$ 。

根据特征值定义，有

$$\mathbf{A}\boldsymbol{\nu} = \lambda\boldsymbol{\nu} \quad (3.1.8)$$

两边取共轭转置，有

$$\boldsymbol{\nu}^H \mathbf{A}^H = \bar{\lambda} \boldsymbol{\nu}^H \quad (3.1.9)$$

因为 \mathbf{A} 为实对称矩阵，则其共轭转置等于其本身，有

$$\boldsymbol{\nu}^H \mathbf{A} = \bar{\lambda} \boldsymbol{\nu}^H \quad (3.1.10)$$

两边同时乘以 $\boldsymbol{\nu}$ ，有

$$\boldsymbol{\nu}^H \mathbf{A} \boldsymbol{\nu} = \bar{\lambda} \boldsymbol{\nu}^H \boldsymbol{\nu} \quad (3.1.11)$$

带入式 3.1.8，有

$$\boldsymbol{\nu}^H \lambda \boldsymbol{\nu} = \bar{\lambda} \boldsymbol{\nu}^H \boldsymbol{\nu} \quad (3.1.12)$$

进而得到

$$(\lambda - \bar{\lambda}) \boldsymbol{\nu}^H \boldsymbol{\nu} = 0 \quad (3.1.13)$$

因为 $\boldsymbol{\nu}^H \boldsymbol{\nu} \neq 0$ ，所以有 $(\lambda - \bar{\lambda}) = 0$ ，即 $\lambda = \bar{\lambda}$ ， λ 为实数。

在数学中，一个算子 \mathbf{A} 的零空间 $N(\mathbf{A})$ 是方程 $\mathbf{A}\boldsymbol{\nu} = 0$ 的所有解 $\boldsymbol{\nu}$ 的集合，即

$$N(\mathbf{A}) = \{\boldsymbol{\nu} \in \mathcal{V} | \mathbf{A}\boldsymbol{\nu} = 0\} \quad (3.1.14)$$

如果 \mathbf{A} 是矩阵，它的零空间是所有向量的空间的线性子空间。

回到证明过程，则 $N(\mathbf{A} - \lambda \mathbf{I})$ 是 \mathcal{R}^N 的子空间，而 $\boldsymbol{\nu} \in N(\mathbf{A} - \lambda \mathbf{I})$ ，故 $\boldsymbol{\nu}$ 是实向量。

定理 3 (不同特征值对应的特征向量正交)

\mathbf{A} 为方矩阵， λ_1, λ_2 为其特征值，且有 $\lambda_1 \neq \lambda_2$ ， λ_1, λ_2 分别对应的特征向量为 $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2$ ，要证明 $\boldsymbol{\nu}_1^H \boldsymbol{\nu}_2 = 0$ 。

首先推演以下等式

$$\lambda_1 \boldsymbol{\nu}_1^H \boldsymbol{\nu}_2 = (\lambda_1 \boldsymbol{\nu}_1)^H \boldsymbol{\nu}_2 = (\mathbf{A}\boldsymbol{\nu}_1)^H \boldsymbol{\nu}_2 = \boldsymbol{\nu}_1^H (\mathbf{A}\boldsymbol{\nu}_2) = \boldsymbol{\nu}_1^H (\lambda_2 \boldsymbol{\nu}_2) = \lambda_2 \boldsymbol{\nu}_1^H \boldsymbol{\nu}_2$$

从而得到等式

$$\lambda_1 \boldsymbol{\nu}_1^H \boldsymbol{\nu}_2 - \lambda_2 \boldsymbol{\nu}_1^H \boldsymbol{\nu}_2 = 0 \quad (3.1.15)$$

即

$$(\lambda_1 - \lambda_2) \boldsymbol{\nu}_1^H \boldsymbol{\nu}_2 = 0 \quad (3.1.16)$$

因为 $(\lambda_1 - \lambda_2) \neq 0$ 所以有 $\nu_1^H \nu_2 = 0$

定理 4 (实对称矩阵是满秩的, 有 N 个线性无关的特征向量, 且某特征值的重数等于该特征值所对应的线性无关的特征向量的个数)

定理 5 (必存在正交矩阵 P , 使得 $P^{-1}AP = \Lambda$, 且 Λ 是以 A 的 N 个特征值为对角元素的对角矩阵)

3.2 LDA

LDA全称是Linear Discriminant Analysis (线性判别分析), 又叫Fisher's Linear Discriminant, 属于supervised learning. LDA试图从带有标签的大量数据样本学习到一种投射方法,

$$y = \mathbf{w}^T \mathbf{x}, \quad \text{in which } y \text{ is a scalar} \quad (3.2.1)$$

能够利用该投射将高维数据降到低维, 并且尽可能多地保留类别判定信息, 如图3.1。

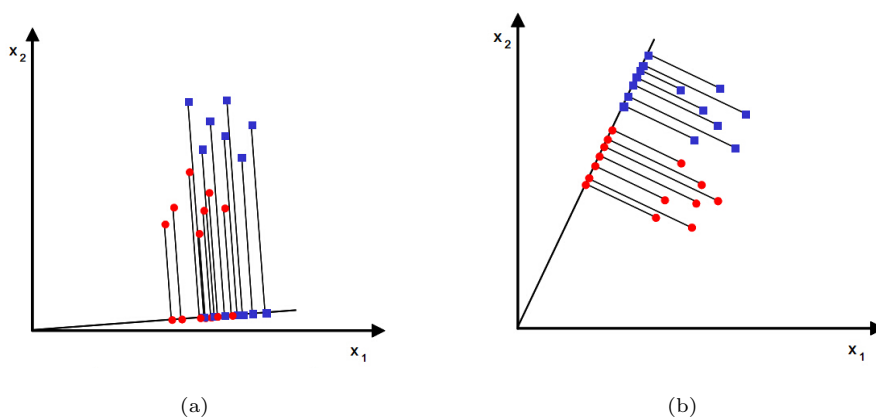


图 3.1: LDA将二维空间的两类数据(分别标记为红、蓝两色)集投射到一维示意图。子图4.1(a)将数据投射到一维空间后, 两种类别的数据点交错在一起, 给类别判定带来困难; 而子图4.1(b)投射到一维空间后保留了类别判定信息, 能够在一维空间很好地进行分类, LDA就是找到最佳的投射方向

3.2.1 二类问题

首先定义数据集, 特征向量 $\mathbf{x} \in \mathcal{R}^n$, 类别标签 $\{c_1, c_2\}$, 要求的参数就是投射函数式3.2.1中的投射向量 $\mathbf{w} \in \mathcal{R}^n$. 因为是二分类问题, 所以投射向量 \mathbf{w} 只有一个。设类别 c_1 的实例数目 n_1 , c_2 的实例数目 n_2 , $N = n_1 + n_2$, 再定义变量

$$\boldsymbol{\mu}_1 = \frac{1}{n_1} \sum_{\mathbf{x} \in c_1} \mathbf{x} \quad (3.2.2)$$

$$\boldsymbol{\mu}_2 = \frac{1}{n_2} \sum_{\mathbf{x} \in c_2} \mathbf{x} \quad (3.2.3)$$

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{\forall \mathbf{x}} \mathbf{x} = \frac{1}{N} (n_1 \boldsymbol{\mu}_1 + n_2 \boldsymbol{\mu}_2) \quad (3.2.4)$$

其中 $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ 分别是类别 c_1, c_2 的实例的特征向量的中心点, $\boldsymbol{\mu}$ 是所有实例的中心点。则投射到一维后的三个中心点

$$\widetilde{\mu}_1 = \mathbf{w}^T \boldsymbol{\mu}_1 = \frac{1}{n_1} \sum_{\mathbf{x} \in c_1} \mathbf{w}^T \mathbf{x} \quad (3.2.5)$$

$$\widetilde{\mu}_2 = \mathbf{w}^T \boldsymbol{\mu}_2 = \frac{1}{n_2} \sum_{\mathbf{x} \in c_2} \mathbf{w}^T \mathbf{x} \quad (3.2.6)$$

$$\widetilde{\mu} = \frac{1}{N} \sum_{\forall \mathbf{x}} \mathbf{w}^T \mathbf{x} = \frac{1}{N} (n_1 \widetilde{\mu}_1 + n_2 \widetilde{\mu}_2) \quad (3.2.7)$$

我们定义初始样本各个类别内的实例的特征向量的分散程度的度量 S_1, S_2

$$S_1 = \sum_{\mathbf{x} \in c_1} (\mathbf{x} - \boldsymbol{\mu}_1)(\mathbf{x} - \boldsymbol{\mu}_1)^T \quad (3.2.8)$$

$$S_2 = \sum_{\mathbf{x} \in c_2} (\mathbf{x} - \boldsymbol{\mu}_1)(\mathbf{x} - \boldsymbol{\mu}_1)^T \quad (3.2.9)$$

在此基础上定义初始样本总的类别内的实例的特征向量的分散程度的度量 S_W

$$S_W = S_1 + S_2 = \sum_{i=1}^2 \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T \quad (3.2.10)$$

然后我们定义初始样本类别间的分散程度的度量 S_B

$$S_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \quad (3.2.11)$$

投影后的类别内的分散程度的度量 \widetilde{S}_W

$$\begin{aligned} \widetilde{S}_W &= \sum_{i=1}^2 \sum_{\mathbf{c} \in c_i} (y - \mu_i)(y - \mu_i)^T \\ &= \sum_{i=1}^2 \sum_{\mathbf{c} \in c_i} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \boldsymbol{\mu}_i)(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \boldsymbol{\mu}_i)^T \\ &= \sum_{i=1}^2 \sum_{\mathbf{c} \in c_i} \mathbf{w}^T (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{w} \\ &= \mathbf{w}^T \left(\sum_{i=1}^2 \sum_{\mathbf{c} \in c_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T \right) \mathbf{w} \\ &= \mathbf{w}^T S_W \mathbf{w} \end{aligned} \quad (3.2.12)$$

投影后的类别间的分散程度的度量 \widetilde{S}_B

$$\begin{aligned} \widetilde{S}_B &= (\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2)(\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2)^T \\ &= (\mathbf{w}^T \boldsymbol{\mu}_1 - \mathbf{w}^T \boldsymbol{\mu}_2)(\mathbf{w}^T \boldsymbol{\mu}_1 - \mathbf{w}^T \boldsymbol{\mu}_2)^T \\ &= \mathbf{w}^T S_B \mathbf{w} \end{aligned} \quad (3.2.13)$$

由于投影后的实例的类间分散程度(用 \widetilde{S}_B)越大, 类内分散程度(用 \widetilde{S}_W)越小, 效果越好, 所以我们定义如下模型评价函数

$$J(\mathbf{w}) = \frac{\widetilde{S}_B}{\widetilde{S}_W} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \quad (3.2.14)$$

式3.2.14是关于投影向量 \mathbf{w} 的函数, 分子和分母都是标量, 而且矩阵 S_B, S_W 都可以用公式3.2.11, 3.2.10 根据初始数据直接计算得出。下面我们求解式3.2.14得到最佳的投影向量 \mathbf{w} 的目标函数:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} - \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \quad (3.2.15)$$

观察式3.2.14, 这是一个不随 \mathbf{w} 的尺度改变而改变的量, 所以我们可以加个条件来限定 \mathbf{w} 的尺度, 最简单的, 规定 $\mathbf{w}^T S_W \mathbf{w} = 1$, 则目标函数改写为

$$\begin{aligned} \mathbf{w}^* &= \arg \min_{\mathbf{w}} - \frac{1}{2} \mathbf{w}^T S_B \mathbf{w} \\ s.t. \quad &\mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned} \quad (3.2.16)$$

这里之所以加个系数 $\frac{1}{2}$ 只是为了方便后边的求导。

式3.2.16对应的Lagrangian形式

$$\mathcal{L}_J = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda (\mathbf{w}^T S_W \mathbf{w} - 1) \quad (3.2.17)$$

KKT条件告诉我们问题的解 \mathbf{w}^* 必满足

$$\frac{\partial \mathcal{L}_J}{\partial \mathbf{w}^*} = S_B \mathbf{w}^* - \lambda S_W \mathbf{w}^* = 0 \Rightarrow S_B \mathbf{w}^* = \lambda S_W \mathbf{w}^* \quad (3.2.18)$$

由于矩阵 S_B 是实对称矩阵, 根据第3.1.3节, 它必是可逆的, 所以有

$$S_W^{-1} S_B \mathbf{w}^* = \lambda \mathbf{w}^* \quad (3.2.19)$$

问题的求解变成了求解 n 阶矩阵 $S_W^{-1} S_B$ 的特征向量。

由于矩阵 $S_B S_W$ 都是实对称矩阵, 所以 $S_W^{-1} S_B$ 也是实对称矩阵, 根据第3.1.3节, 它有 n 个正交的特征向量, 我们取 $S_W^{-1} S_B$ 的最大的特征值 λ^* 所对应的特征向量作为最佳的投影向量 \mathbf{w}^* 。

下面给出证明: 假设有矩阵 $S_W^{-1} S_B$ 的特征值 λ 和其对应的特征向量 \mathbf{w} , 将其带入到式3.2.14, 得到

$$\begin{aligned} J(\mathbf{w}) &= \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \\ &= \mathbf{w}^T S_B \mathbf{w} \\ &= \lambda \mathbf{w}^T S_W \mathbf{w} \\ &= \lambda \end{aligned} \quad (3.2.20)$$

也就是说, $J(\mathbf{w})$ 的大小就是特征值 λ , 所以能够最大化 $J(\mathbf{w})$ 的 \mathbf{w}^* 为 $S_W^{-1} S_B$ 的最大的特征值 λ^* 所对应的特征向量。

3.2.2 多类问题

将LDA扩展到用于多分类(如 c 个类别)问题, 我们仍沿用上一小节的投影列向量 \mathbf{w} , 思路和上一小节的二分类大致相同。例如下面几个变量的定义方法都没有改变

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in c_i} \mathbf{x} \quad (3.2.21)$$

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{\forall \mathbf{x}} \mathbf{x} = \frac{1}{N} \sum_{i=1}^c n_i \boldsymbol{\mu}_i \quad (3.2.22)$$

$$\widetilde{\boldsymbol{\mu}}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in c_i} \mathbf{w}^T \mathbf{x} \quad (3.2.23)$$

$$\widetilde{\boldsymbol{\mu}} = \frac{1}{N} \sum_{\forall \mathbf{x}} \mathbf{w}^T \mathbf{x} = \frac{1}{N} \sum_{i=1}^c n_i \mathbf{w}^T \boldsymbol{\mu}_i \quad (3.2.24)$$

$$S_i = \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T \quad (3.2.25)$$

$$S_W = \sum_{i=1}^c S_i = \sum_{i=1}^c \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T \quad (3.2.26)$$

$$\widetilde{S}_W = \mathbf{w}^T S_W \mathbf{w} \quad (3.2.27)$$

$$\widetilde{S}_B = \mathbf{w}^T S_B \mathbf{w} \quad (3.2.28)$$

只有矩阵 S_B 的定义作了修改, 重新定义如下

$$S_B = \sum_{i=1}^c n_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \quad (3.2.29)$$

还有一个不同于二分类问题的是, c 分类问题可以选取最多 $c-1$ 个特征向量(当然是最大的 $c-1$ 个特征值所对应的特征向量)。假设我们最终选取 k ($k \leq c-1$)个特征向量, 这 k 个特征向量用矩阵表示为

$$\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k) \quad (3.2.30)$$

则实例 \mathbf{x} 就可以用下面的方式映射到一个 k 维列向量

$$\mathbf{y} = \mathbf{w}^T \mathbf{x} \quad (3.2.31)$$

然后我们就可以通过对低维空间（ k 个特征）的 \mathbf{y} 的分类来达到对高维空间（ n 个特征）的 \mathbf{x} 进行分类的目的。

3.2.3 LDA的局限

LDA方法假设各个类别内部的实例遵循高斯分布（unimodal Gaussian），如果不是，也就是说如果数据有别的与分类有关的复杂的结构信息，LDA方法将会在降维的过程中把这些结果信息丢掉，从而得不偿失。图3.2的几种情况不能用LDA方法

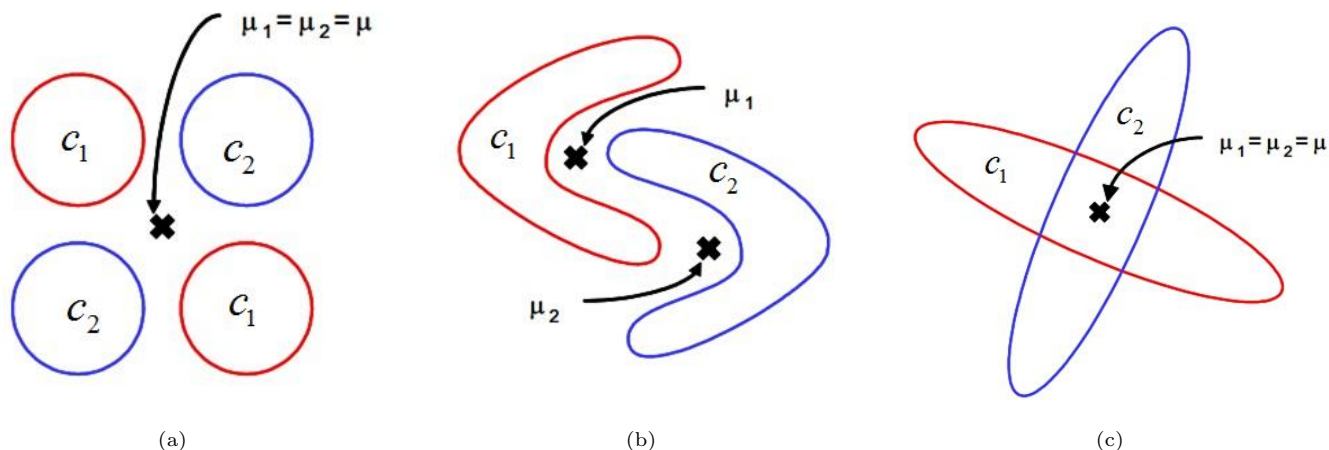


图 3.2: 子图3.2(a)的两个类别(分别用红、蓝两种颜色区分)分别由两个均值不同高斯分布组成，不是unimodal Gaussian 所以不能用LDA;子图3.2(b)的分布也不是高斯分布;子图子图3.2(c)的两个分布的均值相等，这将使式3.2.14恒为0，所以也不能用LDA

3.3 PCA

PCA，即主成分分析（Principal components analysis），是一种unsupervised learning，主要用于数据预处理中的特征提取环节，将数据集用为数较少的有效特征来表示，同时使得原来的数据信息尽可能的保留。

3.3.1 PCA算法

沿用第3.1.1小节（协方差矩阵）的例子，我们从某个实验中得到相同类型的 N 个记录数据 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ ，每条记录都是 M 维的列向量，对应于 M 个数据特征，下面我们介绍一下用PCA来提取少数有效特征来降维的过程。

1. 用 $M \times N$ 阶的矩阵 $\mathbf{A} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ 代表数据，计算 \mathbf{A} 的协方差矩阵（方法见第3.1.1），得到 $M \times M$ 的协方差矩阵 $\mathbf{\Sigma}$
2. 对 $\mathbf{\Sigma}$ 进行特征值分解（方法见第3.1.2），由于 $\mathbf{\Sigma}$ 是协方差矩阵，根据第3.1.3小节的结论，必得到 M 个特征根(计入重根)，按从小到大排列 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$ ，以及对应的特征向量 $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M$ 。
3. 从前向后选择 k 个，也就是最大的 k 个特征值 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$ 对应的特征向量，组成 $M \times k$ 阶的矩阵 $\mathbf{w} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)$ 。

4. 计算 $\mathbf{w}^T \times \mathbf{A}$, 得到 $k \times N$ 的矩阵, 将其标记为 $\tilde{\mathbf{A}}$, 则 $\tilde{\mathbf{A}}$ 就是PCA 降维后的数据矩阵。

选取的前 k 个主成分 (即特征向量) 的累计贡献率的计算方法

$$\eta = \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^M \lambda_i} \quad (3.3.1)$$

它用来表示保留的信息的比重, 或者用这 k 个主成分来解释原始数据的能力。如果原始的维度之间具有较高的相关性, 则前面少数几个主成分的累积贡献率通常就能达到一个较高的水平。

3.3.2 最大化方差解释PCA

在信号处理中认为信号具有较大的方差, 噪声有较小的方差, 信噪比就是信号与噪声的方差之比, 越大越好。可以参考图3.3 来帮助理解该概念。

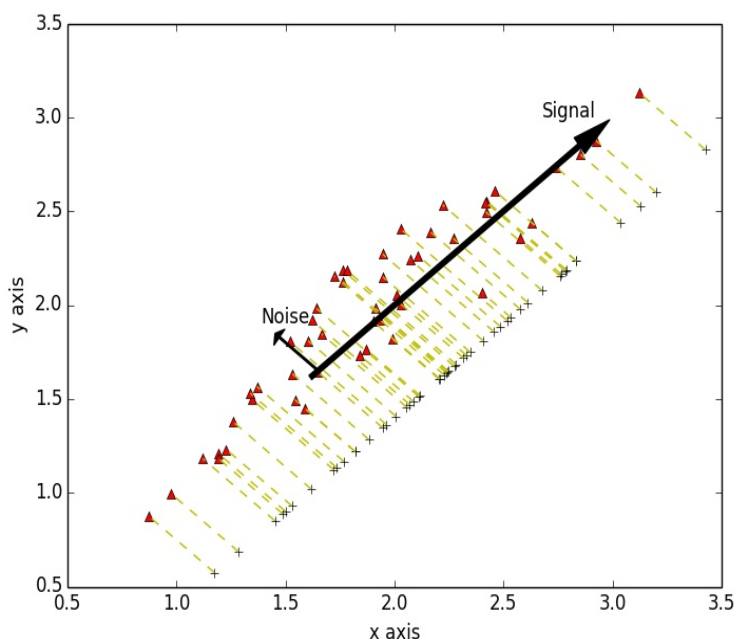


图 3.3: 最大化方差示意图。图中的红色三角是检测到的信号, 可以看到基本上沿对角线分布 (即图示的较粗的箭头指示的方向), 检测信号沿该方向的投影 (图示的叉所形成的线) 可以看成是真实的信号 (Signal), 很明显的该投影数据方差最大。而垂直于该方向的投影 (即图示的较细的箭头指示的方向, 投影数据未画出) 可以看成是噪声 (Noise)。我们可以摒弃噪声数据, 只提取在方差最大方向上的投影数据, 就实现了用一维数据表示原来的二维数据, 这就是用PCA来降维的基本原理。

主成分是这样的投射向量 \mathbf{w} , 样本投影到 \mathbf{w} 上之后被广泛散布, 使得样本之间的差别变得最明显, 即最大化方差。

设第一个投射向量 \mathbf{w}_1 , 根据前面的讲述, 原始数据投射到该方向得到的新数据为

$$\tilde{\mathbf{x}} = (\mathbf{w}_1^T \mathbf{x}_1, \mathbf{w}_1^T \mathbf{x}_2, \mathbf{w}_1^T \mathbf{x}_3, \dots, \mathbf{w}_1^T \mathbf{x}_N)$$

$\tilde{\mathbf{x}}$ 的中心

$$\tilde{\mu} = \frac{1}{N} \sum_{i=1}^N \mathbf{w}_1^T \mathbf{x}_i \quad (3.3.2)$$

则 $\tilde{\mathbf{x}}$ 的方差

$$\begin{aligned}\tilde{\Sigma} &= \frac{1}{N} \sum_{i=1}^N (\mathbf{w}_1^T \mathbf{x}_i - \tilde{\mu})^T (\mathbf{w}_1^T \mathbf{x}_i - \tilde{\mu}) \\ &= \mathbf{w}_1^T \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \tilde{\mu})^T (\mathbf{x}_i - \tilde{\mu}) \right) \mathbf{w}_1 \\ &= \mathbf{w}_1^T \Sigma \mathbf{w}_1\end{aligned}\quad (3.3.3)$$

其中 Σ 就是原始数据的方差矩阵。则我们的问题转化成

$$\begin{aligned}\mathbf{w}_1^* &= \arg \max_{\mathbf{w}_1} \mathbf{w}_1^T \Sigma \mathbf{w}_1 \\ \mathbf{w}_1^T \mathbf{w}_1 &= 1\end{aligned}\quad (3.3.4)$$

其对应的Lagrangian形式

$$\mathcal{L}_J(\mathbf{w}_1) = -\frac{1}{2} \mathbf{w}_1^T \Sigma \mathbf{w}_1 + \frac{1}{2} \lambda (\mathbf{w}_1^T \mathbf{w}_1 - 1) \quad (3.3.5)$$

对 \mathcal{L}_J 关于 \mathbf{w}_1 求导为0, 得

$$\frac{\partial \mathcal{L}_J(\mathbf{w}_1)}{\partial \mathbf{w}_1} = -\Sigma \mathbf{w}_1 + \lambda \mathbf{w}_1 = 0 \Rightarrow \Sigma \mathbf{w}_1 = \lambda \mathbf{w}_1 \quad (3.3.6)$$

将 $\Sigma \mathbf{w}_1 = \lambda \mathbf{w}_1$ 和 $\mathbf{w}_1^T \mathbf{w}_1 = 1$ 带入式3.3.5得

$$\mathcal{L}_J(\mathbf{w}_1) = -\frac{1}{2} \mathbf{w}_1^T \Sigma \mathbf{w}_1 = -\frac{1}{2} \mathbf{w}_1^T \lambda_1 \mathbf{w}_1 = -\frac{1}{2} \lambda_1 \quad (3.3.7)$$

可以看出, 第一个主成分 \mathbf{w}_1^* 应该取为方差矩阵 Σ 最大特征值所对应的特征向量, 而该特征值 λ_1^* 就是该主成分在原始数据的贡献率。

在求得第一个主成分 \mathbf{w}_1^* 的基础上, 我们求解下一个主成分 \mathbf{w}_2 , \mathbf{w}_2 是与 \mathbf{w}_1^* 正交的使得投射在其上的数据的方差最大的投射向量。为了简化符号, 我们把第一个主成分 \mathbf{w}_1^* 标记为 \mathbf{w}_1 , 其对应的特征值 λ_1^* 标记为 λ_1 。参考上面的推导得到优化目标

$$\begin{aligned}\mathbf{w}_2^* &= \arg \max_{\mathbf{w}_2} \mathbf{w}_2^T \Sigma \mathbf{w}_2 \\ \mathbf{w}_2^T \mathbf{w}_2 &= 1 \\ \mathbf{w}_2^T \mathbf{w}_1 &= 0\end{aligned}\quad (3.3.8)$$

对应的Lagrangian形式

$$\mathcal{L}_J(\mathbf{w}_2) = -\frac{1}{2} \mathbf{w}_2^T \Sigma \mathbf{w}_2 + \frac{1}{2} \lambda (\mathbf{w}_2^T \mathbf{w}_2 - 1) + \beta (\mathbf{w}_2^T \mathbf{w}_1 - 0) \quad (3.3.9)$$

对 $\mathcal{L}_J(\mathbf{w}_2)$ 关于 \mathbf{w}_2 求导为0, 得

$$\Sigma \mathbf{w}_2 - \lambda \mathbf{w}_2 - \beta \mathbf{w}_1 = 0 \quad (3.3.10)$$

对上式左乘 \mathbf{w}_1^T , 得

$$\mathbf{w}_1^T \Sigma \mathbf{w}_2 - \lambda \mathbf{w}_1^T \mathbf{w}_2 - \beta \mathbf{w}_1^T \mathbf{w}_1 = 0 \quad (3.3.11)$$

去掉为0项, 化简得

$$\mathbf{w}_1^T \Sigma \mathbf{w}_2 - \beta \mathbf{w}_1^T \mathbf{w}_1 = 0 \quad (3.3.12)$$

由于 $\mathbf{w}_1^T \Sigma \mathbf{w}_2$ 为标量且协方差矩阵 Σ 为对称矩阵, 所以有

$$\mathbf{w}_1^T \Sigma \mathbf{w}_2 = (\mathbf{w}_1^T \Sigma \mathbf{w}_2)^T = \mathbf{w}_2^T \Sigma \mathbf{w}_1 = \mathbf{w}_2^T \lambda_1 \mathbf{w}_1 = 0 \quad (3.3.13)$$

将 $\mathbf{w}_1^T \Sigma \mathbf{w}_2 = 0$ 代入式3.3.12知 $\beta = 0$ ，将 $\beta = 0$ 代入式3.3.10，得等式

$$\Sigma \mathbf{w}_2 = \lambda \mathbf{w}_2 \quad (3.3.14)$$

由式3.3.14知 \mathbf{w}_2 是 Σ 的一个特征向量， λ 是 \mathbf{w}_2 对应的特征值。将式3.3.14代入式，并结合前面的条件和结论，得到

$$\mathcal{L}_J(\mathbf{w}_2) = -\frac{1}{2}\lambda_2 \quad (3.3.15)$$

这表明第二个主成分 \mathbf{w}_2^* 应该是次大的特征值 λ_2^* 对应的特征向量。

按照前面用最大化方差方法求得第一、第二主成分的方法，我们可以依次求得后续的主成分。PCA的全部工作简单点说，就是对原始的空间中顺序地找一组相互正交的坐标轴，第一个轴是使得方差最大的，第二个轴是在与第一个轴正交的平面中使得方差最大的，第三个轴是在与第1,2个轴正交的平面中方差最大的，这样假设在 N 维空间中，我们可以找到 N 个这样的坐标轴，我们取前 k 个去近似这个空间，这样就从一个 N 维的空间压缩到 k 维的空间了，但是我们选择的 k 个坐标轴能够使得空间的压缩使得数据的损失最小。

3.3.3 最小化损失解释PCA

最小化损失从另一个角度解释PCA为什么要用最大的特征值所对应的特征向量，答案是，为了最小化损失。

假设已经找到了 N 个单位正交向量 $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N$ 来表示 N 维的数据 \mathbf{x} ，如下

$$\mathbf{x}_n = \sum_{i=1}^N (\mathbf{x}_n^T \mathbf{w}_i) \mathbf{w}_i \quad (3.3.16)$$

用它们的线性组合 $\tilde{\mathbf{x}}_n$ 来近似原来的点 \mathbf{x}_n

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i \quad (3.3.17)$$

观察式3.3.17，注意 z_{ni} 对每个 x_n 都是不同的，而 b_i 对所有的 x_n 都是相同的。如果明白这一点，你将不至于对后面的求导不知所措。另外，由于 b_i 对所有的 x_n 都是无差别的，或者说 b_i 是与具体的 x_n 无关的，用PCA降到低维后我们将不保留这些信息。而 z_{ni} 是 x_n 之间的差异化信息，降到低维后得到的就是 z_{ni} 。式3.3.17表示用PCA将数据从原来的 N 维空间降到 K 维空间。

PCA降维肯定损失了一些信息，这种损失可以形式化为

$$\begin{aligned} J(z, b) &= \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \\ &= \frac{1}{N} \sum_{n=1}^N \left\| \sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \right\|^2 \\ &= \frac{1}{N} \sum_{n=1}^N \left(\sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \right)^T \left(\sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \right) \end{aligned} \quad (3.3.18)$$

要让损失函数 $J(z, b)$ 最小化，则必有对 z, b 求导为0。下面是求导过程。

$$\begin{aligned}
 \frac{\partial J(z, b)}{\partial z_{nj}} &= \frac{1}{N} \left(\sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \right)^T \mathbf{w}_j \\
 &= \frac{1}{N} \left(\sum_{i=1}^K z_{ni} \mathbf{w}_i^T \mathbf{w}_j + \sum_{i=K+1}^N b_i \mathbf{w}_i^T \mathbf{w}_j - \mathbf{x}_n^T \mathbf{w}_j \right) \\
 &= \frac{1}{N} (z_{nj} \mathbf{w}_j^T \mathbf{w}_j - \mathbf{x}_n^T \mathbf{w}_j) \\
 &= \frac{1}{N} (z_{nj} - \mathbf{x}_n^T \mathbf{w}_j) = 0
 \end{aligned} \tag{3.3.19}$$

得到

$$z_{nj} = \mathbf{x}_n^T \mathbf{w}_j \tag{3.3.20}$$

$$\begin{aligned}
 \frac{\partial J(z, b)}{\partial b_j} &= \frac{1}{N} \sum_{n=1}^N \left(\sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \right)^T \mathbf{w}_j \\
 &= \frac{1}{N} \sum_{n=1}^N (b_j \mathbf{w}_j^T \mathbf{w}_j - \mathbf{x}_n^T \mathbf{w}_j) \\
 &= \frac{1}{N} \sum_{n=1}^N (b_j - \mathbf{x}_n^T \mathbf{w}_j) \\
 &= b_j - \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n^T \mathbf{w}_j \\
 &= b_j - \left(\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n^T \right) \mathbf{w}_j \\
 &= b_j - \left(\frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \right)^T \mathbf{w}_j \\
 &= b_j - \boldsymbol{\mu}^T \mathbf{w}_j = 0
 \end{aligned} \tag{3.3.21}$$

得到

$$b_j = \boldsymbol{\mu}^T \mathbf{w}_j \tag{3.3.22}$$

将式3.3.20和式3.3.22，化简 $\mathbf{x}_n - \widetilde{\mathbf{x}}_n$

$$\begin{aligned}
 \widetilde{\mathbf{x}}_n - \mathbf{x}_n &= \sum_{i=1}^K z_{ni} \mathbf{w}_i + \sum_{i=K+1}^N b_i \mathbf{w}_i - \mathbf{x}_n \\
 &= \sum_{i=1}^K (\mathbf{x}_n^T \mathbf{w}_i) \mathbf{w}_i + \sum_{i=K+1}^N (\boldsymbol{\mu}^T \mathbf{w}_i) \mathbf{w}_i - \mathbf{x}_n \\
 &= \sum_{i=1}^K (\mathbf{x}_n^T \mathbf{w}_i) \mathbf{w}_i + \sum_{i=K+1}^N (\boldsymbol{\mu}^T \mathbf{w}_i) \mathbf{w}_i - \sum_{i=1}^N (\mathbf{x}_n^T \mathbf{w}_i) \mathbf{w}_i \\
 &= \sum_{i=K+1}^N (\boldsymbol{\mu}^T \mathbf{w}_i) \mathbf{w}_i - \sum_{i=K+1}^N (\mathbf{x}_n^T \mathbf{w}_i) \mathbf{w}_i \\
 &= \sum_{i=K+1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i) \mathbf{w}_i
 \end{aligned} \tag{3.3.23}$$

代入损失函数化简

$$\begin{aligned}
 J(z, b) &= \frac{1}{N} \sum_{n=1}^N \left(\sum_{i=K+1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i) \mathbf{w}_i \right)^T \left(\sum_{i=K+1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i) \mathbf{w}_i \right) \\
 &= \frac{1}{N} \sum_{n=1}^N \sum_{i=K+1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i)^2 \mathbf{w}_i^T \mathbf{w}_i \\
 &= \frac{1}{N} \sum_{n=1}^N \sum_{i=K+1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i)^2 \\
 &= \sum_{i=K+1}^N \frac{1}{N} \sum_{n=1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i)^2 \\
 &= \sum_{i=K+1}^N \frac{1}{N} \sum_{n=1}^N ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i)^T ((\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i) \\
 &= \sum_{i=K+1}^N \frac{1}{N} \sum_{n=1}^N \mathbf{w}_i^T (\boldsymbol{\mu} - \mathbf{x}_n) (\boldsymbol{\mu} - \mathbf{x}_n)^T \mathbf{w}_i \\
 &= \sum_{i=K+1}^N \mathbf{w}_i^T \left(\frac{1}{N} \sum_{n=1}^N (\boldsymbol{\mu} - \mathbf{x}_n) (\boldsymbol{\mu} - \mathbf{x}_n)^T \right) \mathbf{w}_i \\
 &= \sum_{i=K+1}^N \mathbf{w}_i^T \boldsymbol{\Sigma} \mathbf{w}_i \\
 &= \sum_{i=K+1}^N \mathbf{w}_i^T \lambda_i \mathbf{w}_i \\
 &= \sum_{i=K+1}^N \lambda_i
 \end{aligned} \tag{3.3.24}$$

所以要使损失函数 J 最小，只要让未选取的 $N-K$ 个特征向量所对应的特征值的和最小，所以我们要选取最大的 K 个特征值对应的特征向量作为主成分才能使PCA损失最小。

3.4 SVD

SVD(Singular Value Decomposition)，翻译成中文是奇异值分解，它可以将一个非方阵的一般矩阵分解成三个矩阵的乘的形式，可以用于矩阵特征提取、降维和压缩等等。

3.4.1 几何方法导出SVD

给定一个方矩阵，我们先从二阶方矩阵说起，将该矩阵用 \mathbf{A} 表示，必存在正交的单位列向量 $\mathbf{v}_1, \mathbf{v}_2$ (如图3.4)，满足 $\mathbf{A}\mathbf{v}_1$ 与 $\mathbf{A}\mathbf{v}_2$ 也是正交的。设 $\mathbf{A}\mathbf{v}_1 = \sigma_1 \mathbf{u}_1, \mathbf{A}\mathbf{v}_2 = \sigma_2 \mathbf{u}_2$ ，其中 $\mathbf{u}_1, \mathbf{u}_2$ 为单位向量。形式化表达如下：

$$\begin{aligned}
 \forall \mathbf{A}, \exists \mathbf{A}\mathbf{v}_1 = \sigma_1 \mathbf{u}_1, \mathbf{A}\mathbf{v}_2 = \sigma_2 \mathbf{u}_2 \\
 s.t. \quad \mathbf{v}_1^T \mathbf{v}_2 = 0, \mathbf{u}_1^T \mathbf{u}_2 = 0 \\
 \|\mathbf{v}_1\| = \|\mathbf{v}_2\| = \|\mathbf{u}_1\| = \|\mathbf{u}_2\| = 1
 \end{aligned} \tag{3.4.1}$$

对一个向量 \mathbf{x} ，由于 $\mathbf{v}_1, \mathbf{v}_2$ 是正交的，则有

$$\mathbf{x} = (\mathbf{v}_1^T \mathbf{x}) \mathbf{v}_1 + (\mathbf{v}_2^T \mathbf{x}) \mathbf{v}_2 \tag{3.4.2}$$

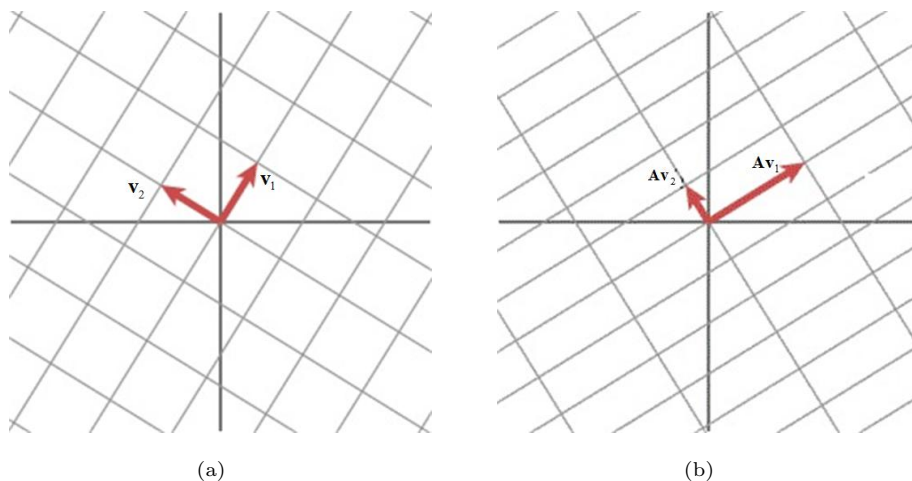


图 3.4: 对任一二阶方阵 \mathbf{A} , 必存在正交的单位列向量 $\mathbf{v}_1, \mathbf{v}_2$, 满足 $\mathbf{A}\mathbf{v}_1$ 与 $\mathbf{A}\mathbf{v}_2$ 也是正交的。

则有

$$\begin{aligned}
 \mathbf{A}\mathbf{x} &= \mathbf{A}((\mathbf{v}_1^T \mathbf{x}) \mathbf{v}_1 + (\mathbf{v}_2^T \mathbf{x}) \mathbf{v}_2) \\
 &= (\mathbf{v}_1^T \mathbf{x}) (\mathbf{A}\mathbf{v}_1) + (\mathbf{v}_2^T \mathbf{x}) (\mathbf{A}\mathbf{v}_2) \\
 &= (\mathbf{v}_1^T \mathbf{x}) (\sigma_1 \mathbf{u}_1) + (\mathbf{v}_2^T \mathbf{x}) (\sigma_2 \mathbf{u}_2) \\
 &= (\sigma_1 \mathbf{u}_1) (\mathbf{v}_1^T \mathbf{x}) + (\sigma_2 \mathbf{u}_2) (\mathbf{v}_2^T \mathbf{x}) \\
 &= (\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T) \mathbf{x} + (\sigma_2 \mathbf{u}_2 \mathbf{v}_2^T) \mathbf{x} \\
 &= (\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T) \mathbf{x}
 \end{aligned} \tag{3.4.3}$$

从而可以将矩阵 \mathbf{A} 写成如下形式

$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T \tag{3.4.4}$$

根据矩阵块运算的规则, 对上面的表达式进一步改写

$$\begin{aligned}
 \mathbf{A} &= \mathbf{u}_1 \sigma_1 \mathbf{v}_1^T + \mathbf{u}_2 \sigma_2 \mathbf{v}_2^T \\
 &= (\mathbf{u}_1 \sigma_1, \mathbf{u}_2 \sigma_2) \begin{pmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{pmatrix} \\
 &= (\mathbf{u}_1 \sigma_1, \mathbf{u}_2 \sigma_2) (\mathbf{v}_1, \mathbf{v}_2)^T \\
 &= (\mathbf{u}_1, \mathbf{u}_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} (\mathbf{v}_1, \mathbf{v}_2)^T
 \end{aligned} \tag{3.4.5}$$

令 $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2)$, $\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$, $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2)$, 则矩阵 \mathbf{A} 可以分解为三个矩阵

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \tag{3.4.6}$$

前面一直假定矩阵 \mathbf{A} 是方阵, 现在我们假定 \mathbf{A} 是 $m \times n$ 阶的一般矩阵, 则其分解成的三个矩阵以及各矩阵的阶(用下标表示)如下式所示

$$\mathbf{A}_{mn} = \mathbf{U}_{mm} \mathbf{\Sigma}_{mn} \mathbf{V}_{nn}^T \tag{3.4.7}$$

式 3.4.7 就是奇异值分解(SVD)的表达式。其中 \mathbf{U} 是正交的矩阵, 称为左奇异矩阵, 组成它的列向量称为左奇异向量; \mathbf{V} 也是正交矩阵, 称为右奇异矩阵, 组成它的列向量称为右奇异向量; $\mathbf{\Sigma}$ 是对角阵, 其对角线上的元素是奇异

值，是其对应的左奇异向量经过 \mathbf{A} 转换到对应的右奇异向量后的被拉伸(或压缩)的系数（见图3.4）。非0的奇异值的个数是矩阵 \mathbf{A} 的秩。

3.4.2 SVD剖析

SVD与特征值分解

假设 \mathbf{A} 是 5×3 阶的，则其SVD分解示意图如下所示

$$\mathbf{A}_{5,3} = \mathbf{U}_{5,5} \mathbf{\Sigma}_{5,3} \mathbf{V}_{3,3}^T$$

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$$

因为 \mathbf{U}, \mathbf{V} 是正交矩阵（正交矩阵转置等于其逆）， $\mathbf{\Sigma}$ 为对角阵，所以有如下推导

$$\begin{aligned} \mathbf{A}^T \mathbf{A} &= \mathbf{V} \mathbf{\Sigma} (\mathbf{U}^T \mathbf{U}) \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix} \mathbf{V}^T = \mathbf{V} \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix} \mathbf{V}^{-1} \\ \mathbf{A} \mathbf{A}^T &= \mathbf{U} \mathbf{\Sigma} (\mathbf{V}^T \mathbf{V}) \mathbf{\Sigma} \mathbf{U}^T = \mathbf{U} \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{U}^T = \mathbf{U} \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{U}^{-1} \end{aligned} \quad (3.4.8)$$

上述两个关系式式的右边描述了关系式左边的特征值分解。于是有如下结论：

- \mathbf{V} 的列向量（右奇异向量）是 $\mathbf{A}^T \mathbf{A}$ 的特征向量。
- \mathbf{U} 的列向量（左奇异向量）是 $\mathbf{A} \mathbf{A}^T$ 的特征向量。
- $\mathbf{\Sigma}$ 的非零奇异值是 $\mathbf{A}^T \mathbf{A}$ 或者 $\mathbf{A} \mathbf{A}^T$ 的非零特征值的平方根。

SVD与PCA

在第3.3节讲到PCA可以用来做特征提取（数据降维），下面讲怎么用SVD来降维。

请看图3.5，上面的部分是SVD矩阵分解的示意图，矩阵 \mathbf{A}_{mn} 被分解成方阵 $\mathbf{U}_{mm}, \mathbf{V}_{nn}^T$ 以及对角阵 $\mathbf{\Sigma}_{mn}$ 。如果我们仅取 $\mathbf{\Sigma}_{mn}$ 前 r 个对角元素组成的子方阵 $\widetilde{\mathbf{\Sigma}}_{rr}$ ， $\widetilde{\mathbf{\Sigma}}_{rr}$ 包含的 r 个奇异值所对应的左奇异向量组成的矩阵 $\widetilde{\mathbf{U}}_{mr}$ （即矩阵 \mathbf{U}_{mm} 的左边 r 列），以及这 r 个奇异值所对应的右奇异向量组成的矩阵 $\widetilde{\mathbf{V}}_{nr}^T$ （即矩阵 \mathbf{V}_{nn}^T 的上边 r 行），则它们的矩阵乘积 $\widetilde{\mathbf{A}}_{mn}$ 可以作为原始矩阵 \mathbf{A}_{mn} 的近似，而且近似程度用 $\eta = \frac{\sum_{i=1}^r \sigma_i^2}{\sum_{i=1}^{\max(m,n)} \sigma_i^2}$ 表达。实际上，由于奇异值 $\sigma_1, \sigma_2, \dots$ 下降的很快，一般 r 取到很小时就能使 η 达到85%~95%。由于 $\mathbf{U}_{mr}, \mathbf{\Sigma}_{rr}, \mathbf{V}_{nr}^T$ 这三个矩阵所包含的元素个数一般远小于初始矩阵 \mathbf{A}_{mn} 所包含的元素数，所以我们可以根据这个规律来达到压缩矩阵 \mathbf{A}_{mn} 的目的。

图3.5图下半部分所示的矩阵压缩过程可以形式化为

$$\widetilde{\mathbf{A}}_{mn} = \widetilde{\mathbf{U}}_{mr} \widetilde{\mathbf{\Sigma}}_{rr} \widetilde{\mathbf{V}}_{nr}^T \quad (3.4.9)$$

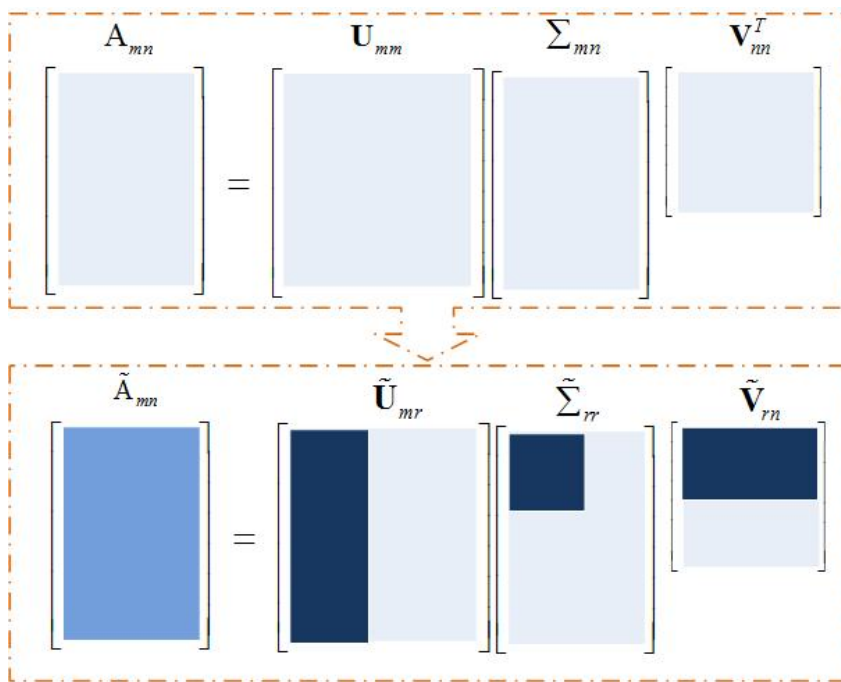


图 3.5: SVD矩阵压缩示意图。取前 r 个奇异值（矩阵 $\widetilde{\Sigma}_{rr}$ ）及其对应的左奇异矩阵（ \widetilde{U}_{mr} ）和右奇异矩阵（ \widetilde{V}_{rn}^T ）的矩阵乘得到矩阵 \widetilde{A}_{mn} ，可以近似原来的 A_{mn} 。通常仅用前面的很少的特征值就可能以很小的损失恢复出原矩阵。

对式3.4.9两边右乘 \widetilde{V}_{nr} ，得到一个 $m \times r$ 阶的矩阵，我们用符号 \widetilde{A}_{mr} 来标记它，注意不要将其与式3.4.9的 \widetilde{A}_{mn} 混淆。

$$\begin{aligned}\widetilde{A}_{mr} &= \widetilde{A}_{mn} \widetilde{V}_{nr} = \widetilde{U}_{mr} \widetilde{\Sigma}_{rr} \widetilde{V}_{rn}^T \widetilde{V}_{nr} \\ &= \widetilde{U}_{mr} \widetilde{\Sigma}_{rr}\end{aligned}\quad (3.4.10)$$

矩阵 \widetilde{A}_{mr} 是一个 $m \times r$ 阶的矩阵，它实现了对原始矩阵 A_{mn} 的列的压缩，它的 r 个列可以看成是从 A_{mn} 的 n 个列所提取出来的 r 个典型特征。这与我们在第3.3节所讲的用PCA提取特征的方法有异曲同工之妙。

讲完了对矩阵的列提取特征，再将对矩阵的行提取特征。对式3.4.9两边左乘 \widetilde{U}_{mr}^T ，得到一个 $r \times n$ 阶的矩阵，我们用符号 \widetilde{A}_{rn} 来标记它，

$$\begin{aligned}\widetilde{A}_{rn} &= \widetilde{U}_{mr}^T \widetilde{A}_{mn} = \widetilde{U}_{mr}^T \widetilde{U}_{mr} \widetilde{\Sigma}_{rr} \widetilde{V}_{rn}^T \\ &= \widetilde{\Sigma}_{rr} \widetilde{V}_{rn}^T\end{aligned}\quad (3.4.11)$$

这样我们得到了一个 $r \times n$ 阶的矩阵 \widetilde{A}_{rn} ，它的每一行都是原始矩阵 A_{mn} 的特征行。

可以看到对矩阵用SVD方法一次，既可以提取行特征又可以提取列特征。在第3.3节用PCA方法是对矩阵提取列特征，如果要对行特征，则在求矩阵协方差阶段需要求的是初始矩阵的转置的协方差。3.3

sample.py

```

1 import numpy
2 import scipy
3 import random
4 import matplotlib.pyplot as pyplot
5 import scipy.linalg as linalg
6 import scipy.stats as stats
7 def loaddata():
8     dataset=[]; locset =[]
9     center = numpy.array([2,2]); direct = numpy.array([-1,1])
10    gauss = stats.norm(0,0.6)
11    direct = numpy.divide(numpy.array([-1,1]),linalg.norm(numpy.array([-1,1])))
12    for i in range(50):
13        bias = random.gauss(0,2)
14        while numpy.abs(bias)>1.6:
15            bias = random.gauss(0,0.4)
16        loc=center +bias*center/linalg.norm(center)
17        locset.append(loc)
18        gausspdf = gauss.pdf(bias)
19        label =random.random()-0.5
20        dataset.append(loc+label*gausspdf*direct)
21    return numpy.array(dataset),numpy.array(locset)
22 def plot(dataset,locset):
23    pyplot.plot(dataset[:,0],dataset[:,1], 'r^')
24    locset[:,0] = locset[:,0]+0.3;locset[:,1]=locset[:,1]-0.3
25    pyplot.plot(locset[:,0],locset[:,1], 'k+')
26    for i in range(dataset.shape[0]):
27        pyplot.plot([locset[i,0],dataset[i,0]],[locset[i,1],dataset[i,1]], 'y—')
28    pyplot.annotate(' ',xytext=(1.6,1.6),xy=(3.0,3.0),arrowprops=dict(facecolor='black',width =3, shrink=0.01))
29    pyplot.annotate(' ',xytext=(1.65,1.65),xy=(1.45,1.85),arrowprops=dict(facecolor='black',width =1, shrink
    =0.01))
30    pyplot.text(1.5,1.9, 'Noise',color="black",ha="center")
31    pyplot.text(2.8,3, 'Signal',color="black",ha="center")
32    pyplot.xlim(0.5,3.5); pyplot.ylim(0.5,3.5)
33    pyplot.xlabel("x axis"); pyplot.ylabel("y axis")
34    pyplot.show()
35 dataset,locset =loaddata()
36 plot(dataset,locset)

```

svd.py

```
1 import numpy
2 import random
3 import matplotlib.pyplot as pyplot
4
5 data = numpy.array([[1,0.6,1,0,0.4],\
6                     [2,1.7,2,0.2,0],\
7                     [1,0,1,0.1,0],\
8                     [3.2,4.8,5,0,0.4],\
9                     [1,1,1.3,2,2],\
10                    [0,0,0,3.2,3],\
11                    [0,0,0,0.9,1]])
12 U,Sigma,VT=numpy.linalg.svd(data)
13 print U.shape,VT.shape,Sigma.shape
14 print Sigma
15 i=2
16 columnclass=data.dot(VT[:i,:].T)
17 print columnclass
18 rowclass = U[:, :i].T.dot(data)
19 print rowclass
20
21 pyplot.figure(1)
22 ax1=pyplot.subplot(211)
23 ax2=pyplot.subplot(212)
24 pyplot.sca(ax1)
25 pyplot.plot(columnclass[:,0],columnclass[:,1], 's')
26 pyplot.sca(ax2)
27 pyplot.plot(rowclass[0,:],rowclass[1,:], '^')
28 pyplot.show()
```


参考文献

- [1] LeftNotEasy,机器学习中的数学(5)-强大的矩阵奇异值分解(SVD)及其应用, <http://www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html>, 01/19 2011.
- [2] LeftNotEasy,机器学习中的数学(4)-线性判别分析(LDA), 主成分分析(PCA), <http://www.cnblogs.com/LeftNotEasy/archive/2011/01/08/lda-and-pca-machine-learning.html>, 01/08 2011.
- [3] jerrylead,主成分分析(Principal components analysis) -最大方差解释, <http://www.cnblogs.com/jerrylead/archive/2011/04/18/2020209.html>, 04/18 2011.
- [4] 260682605,PCA原理及应用, <http://220.181.112.102/view/eecebe0110b4e767f5acfce8b.html>, 11/03 2012.
- [5] Vincent乐,线性判别分析(LDA), <http://blog.csdn.net/chlele0105/article/details/13005527>, 10/24 2013.
- [6] Linear discriminants analysis, http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf.
- [7] Max Welling,Fisher Linear Discriminant Analysis, http://www.ics.uci.edu/~welling/classnotes/papers_class/Fisher-LDA.pdf
- [8] Vincent乐,主成分分析(Principal components analysis), <http://blog.csdn.net/chlele0105/article/details/13004499>, 10/24 2013.
- [9] Vincent乐,Singular Value Decomposition (SVD) tutorial, http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm, 10/24 2013.
- [10] We Recommend a Singular Value Decomposition, <http://www.ams.org/samplings/feature-column/fcarc-svd>.
- [11] ccjou,實對稱矩陣可正交對角化的證明, <https://ccjou.wordpress.com/2011/02/09/%E5%AF%A6%E5%B0%8D%E7%A8%B1%E7%9F%A9%E9%99%A3%E5%8F%AF%E6%AD%A3%E4%BA%A4%E5%B0%8D%E8%A7%92%E5%8C%96%E7%9A%84%E8%AD%89%E6%98%8E/>, 02/09 2011.
- [12] iMetaSearch,Latent Semantic Analysis (LSA) Tutorial, <http://www.puffinwarellc.com/index.php/news-and-articles/articles/33-latent-semantic-analysis-tutorial.html?start=6>.

第四章 Naive Bayes

朴素贝叶斯(Naive Bayes)法是基于贝叶斯定理和特征条件独立假设的分类。对于给定的训练数据集 \mathbf{T} (属性向量 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ 和类标签 y 组成的对 $\langle \mathbf{x}, c \rangle$ 的集合), 首先基于特征条件独立假设学习输入/输出的联合概率分布; 然后基于此模型, 对给定的输入 \mathbf{x} , 利用贝叶斯定理求出后验概率最大的输出 c 。Naive Bayes最大的特点就是原理和实现都很简单, 但是学习和预测的效率都很高。

4.1 Naive Bayes基本方法

输入空间 $\mathcal{X} \subseteq \mathbf{R}^n$ 为 n 维向量的集合, 输出空间为包含 K 个元素的类标记集合 $\mathcal{Y} = \{c_1, c_2, \dots, c_K\}$, 输入是特征向量 $\mathbf{x} \in \mathcal{X}$, 输出为类标签(class label) $y \in \mathcal{Y}$ 。 \mathbf{X} 是定义在输入空间 \mathcal{X} 上的随机向量, Y 是定义在输入空间 \mathcal{Y} 上的随机变量。 $P(\mathbf{X}, Y)$ 是 \mathbf{X} 和 Y 的联合概率分布。训练数据集

$$\mathbf{T} = \{\langle \mathbf{x}_1, y_1 \rangle, \langle \mathbf{x}_2, y_2 \rangle, \dots, \langle \mathbf{x}_N, y_N \rangle\}$$

可以看成由 $P(\mathbf{X}, Y)$ 独立同分布产生。

Naive Bayes就是通过训练数据集学习刚刚提到的联合概率分布 $P(\mathbf{X}, Y)$, 其实在学习过程中直接学到的是先验概率分布及条件概率分布, 其表达式罗列如下:

先验概率分布

$$P(Y = c_k) \quad , k = 1, 2, \dots, K \quad (4.1.1)$$

条件概率分布

$$P(\mathbf{X} = \mathbf{x} | Y = c_k) = P(\mathbf{X}^{(1)} = \mathbf{x}^{(1)}, \dots, \mathbf{X}^{(n)} = \mathbf{x}^{(n)} | Y = c_k), \quad k = 1, 2, \dots, K \quad (4.1.2)$$

Naive Bayes对条件概率分布作了条件独立性的假设, 这是一个较强的假设, Naive Bayes因而得名。具体地, 条件独立假设是:

$$\begin{aligned} P(\mathbf{X} = \mathbf{x} | Y = c_k) &= P(\mathbf{X}^{(1)} = \mathbf{x}^{(1)}, \dots, \mathbf{X}^{(n)} = \mathbf{x}^{(n)} | Y = c_k) \\ &= \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k) \end{aligned} \quad (4.1.3)$$

Naive Bayes法分类时, 对给定的输入 \mathbf{x} , 通过学习到的模型计算后验概率 $P(Y = c_k | \mathbf{X})$, 将后验概率最大的类作为 \mathbf{x} 的输出, 后验概率的计算根据贝叶斯定理进行:

$$P(Y = c_k | \mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x} | Y = c_k) P(Y = c_k)}{\sum_{k=1}^K P(\mathbf{X} = \mathbf{x} | Y = c_k) P(Y = c_k)} \quad (4.1.4)$$

将式(4.1.3)带入式(4.1.4)有

$$P(Y = c_k | \mathbf{X} = \mathbf{x}) = \frac{P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)}{\sum_{k=1}^K P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)}, \quad k = 1, 2, \dots, K \quad (4.1.5)$$

于是, Naive Bayes分类器可以表示为

$$y = f(\mathbf{x}) = \arg \max_{c_k} \frac{P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)}{\sum_{k=1}^K P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)} \quad (4.1.6)$$

注意到式(4.1.6)的分母等于 $P(\mathbf{x})$, 这是一个与 c_k 无关的量, 所以

$$y = \arg \max_{c_k} P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k) \quad (4.1.7)$$

以上就是Naive Bayes的基本原理, 到这一步, 你就可以实现一个简单的贝叶斯分类器了。

4.2 极大似然估计参数

由式(4.1.7)可知, Naive Bayes的学习就是要从训练样本估计 $P(Y = c_k)$ 和 $P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)$, 可以应用极大似然估计法估计相应的概率。变量定义如表4.1 所示

表 4.1: 变量或函数定义

Symbol	Meaning
y_i	第 <i>i</i> 个样本标签
c_k	第 <i>k</i> 类
\mathbf{x}_i	第 <i>i</i> 个样本
$\mathbf{x}_i^{(j)}$	第 <i>i</i> 个样本的第 <i>j</i> 个特征
a_{jl}	第 <i>j</i> 个特征可能取的第 <i>l</i> 个值
I	指示函数

先验概率 $P(Y = c_k)$ 的极大似然估计是

$$P(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k)}{N}, \quad k = 1, 2, \dots, K \quad (4.2.1)$$

设第*j*个特征 $x^{(j)}$ 可能的取值集合为 $\{a_{j1}, a_{j2}, \dots, a_{jS_j}\}$, 条件概率 $P(X^{(j)} = a_{jl} | Y = c_k)$ 的极大似然估计是

$$P(X^{(j)} = a_{jl} | Y = c_k) = \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{jl}, y_i = c_k)}{\sum_{i=1}^N I(y_i = c_k)} \quad (4.2.2)$$

$j = 1, 2, \dots, n; l = 1, 2, \dots, S_j; k = 1, 2, \dots, K$

4.3 后验概率与期望风险的关系

Naive Bayes法将输入实例 \mathbf{x} 划归到后验概率最大(后验概率就是式(4.1.4), 后验概率最大由式(4.1.7)体现), 这里我们将要证明上一节的“后验概率最大化”蕴含着“期望风险最小化”这一指导准则。

假设选择0-1损失函数:

$$L(Y, f(\mathbf{X})) = \begin{cases} 1, & Y \neq f(\mathbf{X}) \\ 0, & Y = f(\mathbf{X}) \end{cases}$$

式中 $f(\mathbf{X})$ 是分类决策函数。这时，期望风险函数为

$$R_{exp}(f) = E[L(Y, f(\mathbf{X}))]$$

期望是对联合分布 $P(\mathbf{X}, Y)$ 取的，由此对上式作进一步推导如下：

$$\begin{aligned} R_{exp}(f) &= E_{\mathbf{X}, Y}[L(Y, f(\mathbf{X}))] \\ &= \sum_{\mathbf{X}} \sum_Y L(Y, f(\mathbf{X})) P(\mathbf{X}, Y) \\ &= \sum_{\mathbf{X}} \sum_{k=1}^K L(c_k, f(\mathbf{X})) P(\mathbf{X}, c_k) \\ &= \sum_{\mathbf{X}} \sum_{k=1}^K L(c_k, f(\mathbf{X})) P(c_k | \mathbf{X}) P(\mathbf{X}) \\ &= \sum_{\mathbf{X}} P(\mathbf{X}) \sum_{k=1}^K L(c_k, f(\mathbf{X})) P(c_k | \mathbf{X}) \\ &= E_X \sum_{k=1}^K L(c_k, f(\mathbf{X})) P(c_k | \mathbf{X}) \end{aligned}$$

所以可以将期望风险函数写作

$$R_{exp}(f) = E_X \sum_{k=1}^K L(c_k, f(\mathbf{X})) P(c_k | \mathbf{X}) \quad (4.3.1)$$

观察式(4.3.1)，可以发现对于每个 \mathbf{x} ，其对应的风险函数 $L(c_k, f(\mathbf{X})) P(c_k | \mathbf{X})$ 是独立计算的。为了使期望风险最小化，可以对 $X = \mathbf{x}$ 逐个极小化，由此得到

$$\begin{aligned} f(\mathbf{x}) &= \arg \min_{y \in \mathcal{Y}} \sum_{k=1}^K L(c_k, y) P(c_k | \mathbf{X} = \mathbf{x}) \\ &= \arg \min_{y \in \mathcal{Y}} \sum_{k=1}^K P(y \neq c_k | \mathbf{X} = \mathbf{x}) \\ &= \arg \min_{y \in \mathcal{Y}} \sum_{k=1}^K (1 - P(y = c_k | \mathbf{X} = \mathbf{x})) \\ &= \arg \min_{y \in \mathcal{Y}} (1 - P(y = c_k | \mathbf{X} = \mathbf{x})) \\ &= \arg \max_{y \in \mathcal{Y}} P(y = c_k | \mathbf{X} = \mathbf{x}) \end{aligned} \quad (4.3.2)$$

由式(4.3.2)可以知道，根据期望风险最小化准则可以推导出后验概率最大化准则：

$$f(\mathbf{x}) = \arg \max_{c_k} P(c_k | \mathbf{X} = \mathbf{x}) \quad (4.3.3)$$

也就是Naive Bayes法采用的原理。可以看到式(4.3.3)与式(4.1.7)是等效的。

4.4 单调递增变换与线性求和

Naive Bayes的另一个迷人之处在于我们能找到另一种与基本模型等价的形式，那就是我们可以使用严格单调变换对 $P(c|\mathbf{x})$ 和分类阈值 t 同时变换，而分类结果不变。令 T 是严格单调递增函数，则下面三个例子有助于我们加深

对严格单调变换的理解。

$$P(c|\mathbf{x}) > t \iff T(P(c|\mathbf{x})) > T(t)$$

$$P(c|\mathbf{x}) > P(c|\mathbf{y}) \iff T(P(c|\mathbf{x})) > T(P(c|\mathbf{y}))$$

$$f(\mathbf{x}) = \arg \max_{y \in \mathcal{Y}} P(y = c_k | \mathbf{X} = \mathbf{x}) \iff \arg \max_{y \in \mathcal{Y}} T(P(y = c_k | \mathbf{X} = \mathbf{x}))$$

定义函数式(4.4.1),(4.4.2)分别代表的函数就是两种不同的单调递增变换:

$$T_1(x) = \frac{x}{1-x} \quad 0 \leq x < 1 \quad (4.4.1)$$

$$T_2(x) = \ln x \quad x > 0 \quad (4.4.2)$$

对式(4.1.5)(假设 $K = 2$, 即类标签只有 c_1, c_2 两类)作公式(4.4.1)的变换($k = 1$ 为例, $k = 2$ 同 $k = 1$):

$$\begin{aligned} T_1(P(Y = c_1 | \mathbf{X} = \mathbf{x})) &= \frac{P(Y = c_1 | \mathbf{X} = \mathbf{x})}{1 - P(Y = c_1 | \mathbf{X} = \mathbf{x})} \\ &= \frac{P(Y = c_1 | \mathbf{X} = \mathbf{x})}{P(Y = c_2 | \mathbf{X} = \mathbf{x})} \\ &= \frac{P(c_1)}{P(c_2)} \frac{P(\mathbf{x} | c_1)}{P(\mathbf{x} | c_2)} \\ &= \frac{P(c_1)}{P(c_2)} \prod_{i=1}^n \frac{P(x_i | c_1)}{P(x_i | c_2)} \end{aligned} \quad (4.4.3)$$

对式(4.4.3)作公式(4.4.2)的变换,可得到另一个打分器:

$$T_2(T_1(P(Y = c_1 | \mathbf{X} = \mathbf{x}))) = \ln \frac{P(c_1)}{P(c_2)} + \sum_{i=1}^n \ln \frac{P(x_i | c_1)}{P(x_i | c_2)} \quad (4.4.4)$$

我们定义 $w_0 = \ln \frac{P(c_1)}{P(c_2)}$ 和 $w_i(x_i) = \ln \frac{P(x_i | c_1)}{P(x_i | c_2)}$, 并定义函数

$$T(x) = T_2(T_1(x)) \quad (4.4.5)$$

其实式(4.4.5)也是一个单调递增变换函数。则式(4.4.4)可重写为:

$$T(P(Y = c_1 | \mathbf{X} = \mathbf{x})) = w_0 + \sum_{i=1}^n w_i(x_i) \quad (4.4.6)$$

以上面的推导为基础, 我们定义打分器函数

$$S(c_1, \mathbf{x}) = w_0 + \sum_{i=1}^n w_i(x_i) \quad (4.4.7)$$

表示输入 \mathbf{x} 对类别 c_1 的打分, 当然 \mathbf{x} 对类别 c_2 的打分器函数 $S(c_2, \mathbf{x})$ 可以类比定义, 如果没有特别说明, 这里一律对 $S(c_1, \mathbf{x})$ 进行讨论。有了 $S(c_1, \mathbf{x})$ 、 $S(c_2, \mathbf{x})$ 两个评分, 我们就可以把 \mathbf{x} 划归到评分高的类别, 这就实现了分类器的目的。

再看上面的几个变换函数, 可以知道分值 $S = w_0 + \sum_{i=1}^n w_i(x_i)$ 是直接由 $P(c|\mathbf{x})$ 的估计值计算出来的, 而且Naive Bayes模型其实就是对原始 x_i 的变换的简单求和。

如果每个变量都是离散的或者是将连续变量分割为小的单元来离散化, 式(4.4.7)的形式就特别简单。记变量 x_i 取其第 k_i 个单元的值是 $x_i^{(k_i)}$, 那么 $w_i(x_i^{(k_i)})$ 就是“两个份额”之比的对数, 即: 类别 c_1 的点在变量 x_i 上取值落入的第 k_i 个单元的份额除以类别 c_2 的点在变量 x_i 上取值落入的第 k_i 个单元的份额。有时将这个 $w_i(x_i^{(k_i)})$ 称为权重, 即第 i 个变量对总分值的贡献, 或者第 i 个变量为将对象 \mathbf{x} 划归到类别 c_1 提供的证据。根据这些证据权重, 可以识别出哪些变量对于判断任一特定对象的类别归属是重要的。

4.5 独立假设的利弊及扩展

第1节提到Naive Bayes对条件概率分布作了条件独立性的假设，在这一节我们重点讲述该独立性假设的利弊以及人们对其弊端提出的改善措施。

4.5.1 为什么独立性假设可行

降低复杂度

式(4.1.2)所表达的条件概率分布 $P(\mathbf{X}^{(1)} = \mathbf{x}^{(1)}, \dots, \mathbf{X}^{(n)} = \mathbf{x}^{(n)} | Y = c_k)$ 有指数级数量的参数。假定 $x^{(j)}$ 可取值有 S_j 个, $j = 1, 2, \dots, n$, 则 \mathbf{x} 可能的情况有 $\prod_{j=1}^n S_j$ 种; 又假设 Y 可能的取值有 K 种情况, 那么参数的个数为 $K \prod_{j=1}^n S_j$ 个。如果对所有参数进行估计, 将需要大量的运算和足够多的训练数据集, 而且它们会随着 \mathbf{X} 的维度的增加而爆炸性增长, 维数爆炸问题是如此令人沮丧以至于所有的研究者都选择避其锋芒。

当我们作条件独立性假设后, 即

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

, 复杂度发生了戏剧性地改善, 参数的个数变成了线性数量级, 具体的 $K \sum_{j=1}^n S_j$, 这种改善使贝叶斯方法具有实用性。

训练数据已做预处理

数据集通常在用于将机器学前已经进行了一个变量选择过程, 例如线性回归中的变量选择方法。该过程将高度相关变量剔除, 剩下的变量之间可能接近于独立关系。

距离不遥远

独立性假设可能会导致差的概率估计或差的 $\frac{P(c_1|\mathbf{x})}{c_2|\mathbf{x}}$ 比率估计, 但这并不意味着估计得到的决策面和真实的决策面就相差很远。

非线性分类器

Naive Bayes所产生的决策面可以具有复杂的非线性形状: 虽然决策面(式(4.4.7))关于 $w_i(x_i)$ 是线性的, 但关于原始变量 x_i 是高度的非线性, 所以她能拟合出非常复杂的曲面。

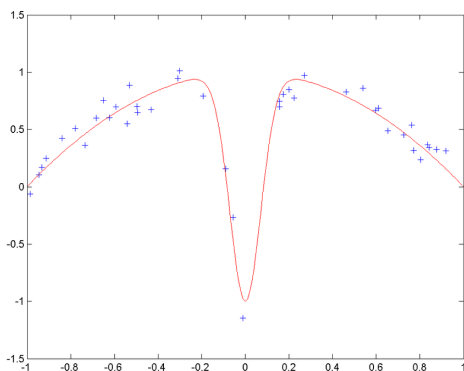
4.5.2 收缩系数的引入

属性间的独立性假设暗示了Naive Bayes可能是有局限性的, 毕竟在实际问题中极少有变量独立的情形。接下来我们从“偏差-方差”均衡原理的角度说明模型假设对偏差的影响并引入收缩系数来缓解这种影响。

“偏差-方差”均衡原理

在统计学和数据挖掘领域, “偏差-方差”均衡原理(bias-variance tradeoff)用于同时优化有导师学习算法产生的模型对训练数据集之外的数据集(如交叉验证中的CE试集)的泛化能力的两种误差:

- 偏差(bias)来源于错误的模型假设
- 方差(variance)来源于对训练数据的波动的敏感性



(a) Function and noisy data

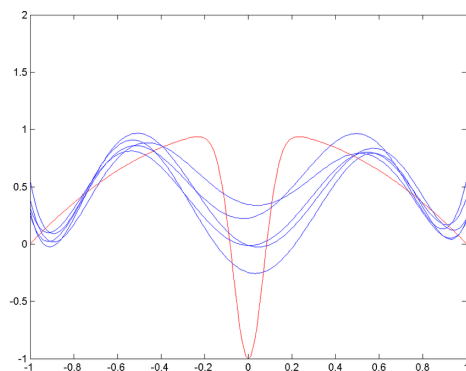
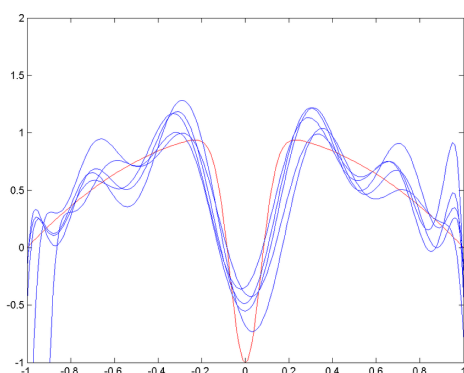
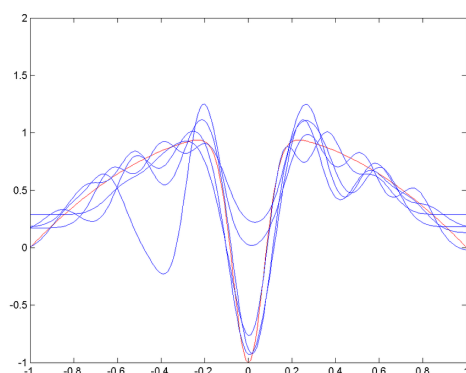
(b) $\sigma=5$ (c) $\sigma=1$ (d) $\sigma=0.1$

图 4.1: A function (red) is approximated using RBF (blue). Several trials are shown in each graph. For each trial, a few noisy data points are provided as training set. For a big σ (image 4.1(b)) the bias is high(i.e.the RBFs cannot fully approximate the function,especially the central dip), but the variance between different trials is low. As σ decreases (image 4.1(c) and 4.1(c)) the bias decreases(i.e.the blue curves more closely approximate the red). However, depending on the noise in different trials the variance between trials increases. In image4.1(d) the approximated values for $x=0$ varies wildly depending on where the data points were located.

图4.1用径向基函数逼近拟合数据的过程。可以看到随着模型假设的复杂度的提高(径向基函数的扩展速度 σ 的减小),模型对相应的训练数据集的偏差(bias)降低,但是不同训练数据集训练出来的模型之间的方差(variance)却增加(如图4.2),所以模型假设的复杂度要取折中,这就是“偏差-方差”均衡原理。

顺便提一下,“偏差-方差”分解方法将算法的泛化错误问题分解为方差、偏差和残差(噪声)这三部分进行的分析。“偏差-方差”均衡原理用于各种有导师学习算法: classification, regression, and structured output learning。她也用于解释启发式算法的有效性。

收缩系数

在对为什么要对属性之间作独立性假设的讨论中我们知道, n 个一元变量编辑分布的复杂性远远小于一个 n 元变量的联合分布。这意味着,为了达到同样的模型精度,独立模型比非独立模型需要更少的数据点。换句话说,如果将模型限制为变量具有独立性,那么基于给定可用样本的估计的方差就比较小。当然,如果该假设与实际不符,就

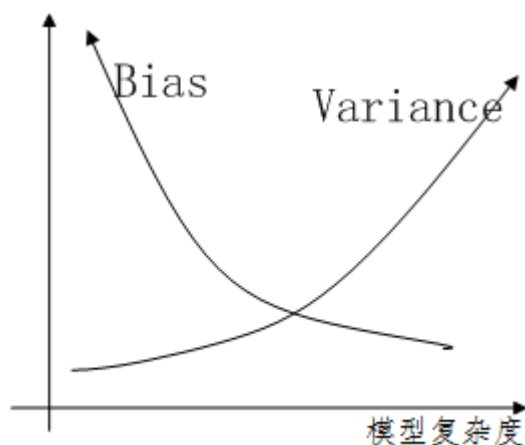


图 4.2: “方差-偏差”均衡示意图,为了实现方差和偏差二者的均衡,需要对模型假设的复杂度取折中,即模型太复杂了或太简单了都不可取

会有偏差增大的风险。这也是“偏差-方差”均衡原理的体现。

为了减少独立性假设带来的偏差风险, Naive Bayes的一个简单的修正方法被提出。为了理解这个修正的工作机制, 先来分析一种极端情形——所有的变量完全相关, 在这种情况下, 所有的变量具有相同的边际分布且该边际分布与联合分布相等, 假设该值为 r 。其Naive Bayes估计, 即作独立性假设下的条件概率:

$$\frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})} = \frac{P(c_1)}{P(c_2)} \prod_{i=1}^n \frac{P(x_i|c_1)}{P(x_i|c_2)} = r^n \quad (4.5.1)$$

其实真实的优势比(odd ratio)是:

$$\frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})} = \frac{P(c_1)}{P(c_2)} \frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)} = r \quad (4.5.2)$$

通过对式(4.5.1),(4.5.2)的对比, 相关性的存在会使Naive Bayes高估 ($r > 1$) 或低估 ($r < 1$) 优势比 $\frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})}$ 。

从这一现象可以得到一个修改Naive Bayes估计的策略, 就是对 $\frac{P(x_i|c_1)}{P(x_i|c_2)}$ 取一个小于1的幂, 将整体估计向真实的优势比收缩靠拢。这就形成一种改良的Naive Bayes估计:

$$\frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})} = \frac{P(c_1)}{P(c_2)} \prod_{i=1}^n \left(\frac{P(x_i|c_1)}{P(x_i|c_2)} \right)^\beta, \beta \leq 1 \quad (4.5.3)$$

通常要搜索 β 所有可能的取值, 用交叉验证等方法对每个取值对应的模型进行评估, 最后确定能产生最好的预CE结果的 β 值。如果用式(4.4.7)分析, 相当于给 $w_i(x_i)$ 增加一个收缩系数。

4.6 拉普拉斯平滑

用极大似然估计可能会出现要估计的概率值为0的情况, 这时会影响到后验概率的计算结果。例如式(4.1.5) (重写如下) 分子中只要有一项为0, 后验概率就为0, 而不管其他项取值多少, 这显然是不公平的。

$$P(Y = c_k | \mathbf{X} = \mathbf{x}) = \frac{P(Y = c_k) \prod_{j=1}^n P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)}{P(\mathbf{X})}$$

对上式, 我们不用考虑 $P(\mathbf{X} = 0)$ 的情况, 只需要设法使条件概率 $P(\mathbf{X}^{(j)} = \mathbf{x}^{(j)} | Y = c_k)$ 和先验概率 $P(c_k)$ 这两个式子不为0就可以了, 下面对其分别讨论。

下面是修正过程(这里仍沿用表4.1定义的变量)

4.6.1 先验概率修正

1. 要修正的表达式:

$$P(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k)}{N}, \quad k = 1, 2, \dots, K \quad (4.6.1)$$

2. 作修正的目的:

$$P(Y = c_k) > 0 \quad (4.6.2)$$

3. 不可破坏的条件:

$$\sum_{k=1}^K P(Y = c_k) = 1 \quad (4.6.3)$$

很自然地，我们的修正结果为

$$P(Y = c_k) = \frac{\sum_{i=1}^N I(y_i = c_k) + \lambda_1}{N + K\lambda_1} \quad \lambda_1 > 0 \quad (4.6.4)$$

4.6.2 条件概率修正

1. 要修正的表达式:

$$P(X^{(j)} = a_{jl} | Y = c_k) = \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{jl}, y_i = c_k)}{\sum_{i=1}^N I(y_i = c_k)} \quad j = 1, 2, \dots, n; l = 1, 2, \dots, S_j; k = 1, 2, \dots, K \quad (4.6.5)$$

2. 作修正的目的:

$$P(X^{(j)} = a_{jl} | Y = c_k) > 0 \quad (4.6.6)$$

3. 不可破坏的条件:

$$\sum_{l=1}^{S_j} P(X^{(j)} = a_{jl} | Y = c_k) = 1 \quad (4.6.7)$$

很自然地，我们的修正结果为

$$P(X^{(j)} = a_{jl} | Y = c_k) = \frac{\sum_{i=1}^N I(x_i^{(j)} = a_{jl}, y_i = c_k) + \lambda_2}{\sum_{i=1}^N I(y_i = c_k) + S_j \lambda_2} \quad \lambda_2 > 0 \quad (4.6.8)$$

λ_1 和 λ_2 的取值过程是完全分离的，所以二者可以相等也可以不相等。一般取 $\lambda_1 = \lambda_2 = 1$ ，这时称为拉普拉斯平滑(Laplace smoothing)。

参考文献

- [1] 李航, 统计学习方法. 清华大学出版社, 北京, 2014.
- [2] Xindong Wu,Vipin Kumar, *The Top Ten Algorithms in Data Mining*. CRC press, Taylor& Francis, 2009.
- [3] leftnoteasy,线性回归,偏差与方差权衡, http://www.cnblogs.com/LeftNotEasy/archive/2010/12/19/mathmatic_in_machine_learning_2_regression_and_bias_variance_trade_off.html, 12/19 2010.
- [4] Bias - variance tradeoff, http://en.wikipedia.org/wiki/Bias-variance_tradeoff, 12/19 2010.
- [5] 吴军, 数学之美-第三章统计学习语言. 人民邮电出版社, 北京, 2013.

第五章 PageRank算法

Google采用“网页级别（PageRank）”与“页面分析”两种技术来确保检索质量与精确率。所谓PageRank技术是基于整个网络的链接结构，按网页的链接广度（Link Popularity）来决定网页重要性。而“页面分析”则按页面标题是否出现关键词、网页内关键词出现的频率（keyword density）及关键词出现在什么位置等，来确定哪些网页与正在执行的搜索密切相关。这里重点讲述前者的数学机制。

5.1 PageRank算法

传统的搜索异于互联网搜索的特点：

1. 搜索对象的数量较小——比如一本字典收录的字通常只有一两万个，一家图书馆收录的不重复图书通常不超过几十万种，一家商店的商品通常不超过几万种。
2. 搜索对象具有良好的分类或排序——比如字典里的字按拼音排序，图书馆里的图书按主题分类，商店里的商品按品种或用途分类等等。
3. 搜索结果的重复度较低——比如字典里的同音字通常不超过几十个，图书馆里的同名图书和商店里的同种商品通常也不超过几十种。

对于互联网搜索的前两个特点，可以分别通过增加服务器计算能力和存储空间的手段来解决，而对于第三个特点，就只能事先就知道搜索结果的重要性的排序，并把排名靠前的结果优先展示给用户。

每个网页的排序是不能靠网页自己来控制的，例如一个垃圾网页无论把关键词重复多少次，它仍然是垃圾网页，用户在搜索该关键词时该网页的排序仍然是要靠后的。

学术界评判学术论文重要性的通用方法，那就是看论文的引用次数。在互联网上，与论文的引用相类似的是显然是网页的链接，那么通过研究网页间的相互链接来确定排序或许也是一种思路。实际上，PageRank也是这么做的。具体地说，一个网页被其它网页链接得越多，它的排序就应该越靠前。不仅如此，一个网页越是被排序靠前的网页所链接，它的排序就也应该越靠前。这一条的意义也是不言而喻的，就好比一篇论文被诺贝尔奖得主所引用，显然要比被普通研究者所引用更说明其价值。

首先我们阐述两个相关的概念：

1. 网页 i 的入链(in-links)：那些指向网页 i 的来自于其它网页的超链接，通常不包括来自于同一站点内网页的超链接。
2. 网页 i 的出链(out-links)：那些从网页 i 指向其它网页的超链接，通常不包括指向同一站点内网页的超链接。

前面从论文引用的角度阐述了PageRank的思想，现在我们从排序声望(rank prestige)的角度进行进一步阐述：

1. 从一个网页指向另一网页的超链接是PageRank值的隐含式传递，网页的PageRank值是由指向它的所有的网页所传递过来的PageRank值总和决定的。这样，网页*i*的入链越多，它的PageRank值可能就越高，它得到的声望也就越高。
2. 一个网页指向多个其他网页，那么它传递的声望值就会被它所指向的多个网页分享。也就是说，即使网页*i*被一个PageRank值很高的网页*j*所指向，如果网页*j*的出链非常多，网页*i*从网页*j*得到的声望值可能因被稀释也很小。

我们可以把Web抽象成一个有向图 $G = (V, E)$ ，其中 V 是图的节点集合(一个节点对应一个网页)， E 是图的有向边集合(有向边对应超链接)。设Web上的网页总数为 n (即， $n = |V|$)。上述思想可以形式化为：

$$P(i) = \sum_{(j,i) \in E} \frac{P(j)}{O_j}$$

其中 $P(i)$ 表示网页*i*的PageRank值， O_j 是网页*j*出链的数量， $(j, i) \in E$ 表示存在网页*j*指向网页*i*的超链接。从数学的观点看就存在一个包含 n 个未知量的线性方程组，可以用一个矩阵来表示。首先作一个符号的约定，用列向量 \mathbf{P} 表示 n 个网页的PageRank值，如下：

$$\mathbf{P} = (P(1), P(2), \dots, P(n))^T$$

再用矩阵 \mathbf{A} 表示有向图的邻接矩阵，并按如下规则为每条有向边赋值：

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{if } (i, j) \in E \\ 0 & \text{else} \end{cases}$$

比如下面的网络链接结构图 其对应的邻接矩阵 \mathbf{A} 如下：

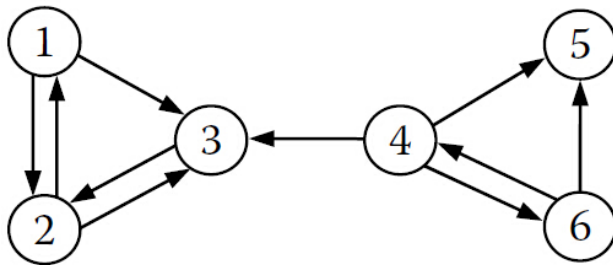


图 5.1: 网络的超链接图的例子

$$\mathbf{A} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

我们可以得到如下方程组：

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \quad (5.1.1)$$

我们的任务就是在已知矩阵 \mathbf{A} 的条件下,求解向量 \mathbf{P} 。再看方程组,发现 \mathbf{P} 是循环定义的,所以看以用著名的迭代方法求解 \mathbf{P} 。我们定义可以给定初值 \mathbf{P}_0 ,定义 \mathbf{P}_n 是经过第 n 次迭代得到的 \mathbf{P} 值,则可以形式化如下:

$$\mathbf{P}_{n+1} = \mathbf{A}^T \mathbf{P}_n, n = 0, 1, \dots, \infty, \text{ given } \mathbf{P}_0 \quad (5.1.2)$$

满足公式(1)方程组的解 \mathbf{P}^* 就是 $\lim_{n \rightarrow \infty} \mathbf{P}_n$ 。

当然,也可以用马尔科夫链(Markov chain)进行建模,这时 \mathbf{P}_n 就可以看成是Markov chain的一个状态(state), \mathbf{A} 可以表示状态转移矩阵(state transition matrix),这样就可以转换成马尔科夫链的遍历性和极限分布问题。

接下来我们要处理的问题就是:

1. $\lim_{n \rightarrow \infty} \mathbf{P}_n$ 是否存在?
2. 如果极限存在,它是否与 \mathbf{P}_0 的选取无关?
3. 如果极限存在,并且与 \mathbf{P}_0 的选取无关,它作为网页排序的依据是否真的合理?

如果这三个问题的答案都是肯定的,那么网页排序问题就算解决了。反之,哪怕只有一个问题的答案是否定的,网页排序问题也就不能算是得到了满意解决。那么实际答案如何呢?很遗憾,是后一种,而且三个问题的答案全都是否定的。这可以由一些简单的例子看出。比方说,在只包含两个相互链接网页的迷你型互联网上,如果 $\mathbf{P}_0 = (1, 0)^T$,极限就不存在,因为概率分布将在 $(1, 0)^T$ 和 $(0, 1)^T$ 之间无穷振荡。而存在几个互不连通(即互不链接)区域的互联网则会使极限——即便存在——与 \mathbf{P}_0 的选取有关,因为把 \mathbf{P}_0 选在不同区域内显然会导致不同极限。至于极限存在,并且与 \mathbf{P}_0 的选取无关时它作为网页排序的依据是否真的合理的问题,虽然不是数学问题,答案却也是否定的,因为任何一个“悬挂网页”都能象黑洞一样,把其它网页的概率“吸收”到自己身上而不再向外“释放”,这显然是不合理的。这种不合理效应是如此显著,以至于在一个连通性良好的互联网上,哪怕只有一个“悬挂网页”,也足以使整个互联网的网页排序失效。

indent 在进一步讨论之前,我们引入几个概念:

- 正矩阵(Positive matrix):每个矩阵元都大于0的矩阵。每个元素都大于等于0的矩阵是非负矩阵(Nonnegative matrix)。
- 素阵(Primitive matrix):素矩阵是指自身的某个次幂为正矩阵(Positive matrix)的矩阵。设 \mathbf{A} 为一个 $n \times n$ 的方阵,如果存在正整数 k 使得矩阵

$$\mathbf{A}^k > 0$$

那么,称矩阵 \mathbf{A} 为素矩阵。

- 随机矩阵(stochastic matrix):随机矩阵又叫概率矩阵(probability matrix)、转移矩阵(transition matrix)、马尔科夫矩阵(Markov matrix)等。随机矩阵通常表示左随机矩阵(left stochastic matrix),如果方阵 $\mathbf{A}_{n \times n}$ 为(左)随机矩阵,则其满足以下条件:

suppose $a_{i,j}$ is the element from \mathbf{A} at row i , column j , then we have

$$\forall i = 1..n, j = 1..n, a_{i,j} \geq 0$$

$$\forall i = 1..n, \sum_{j=1}^n a_{ij} = 1$$

左随机矩阵的转置称为右随机矩阵(right stochastic matrix)。

- 不可约矩阵 (irreducible matrix): 方阵 \mathbf{A} 是不可约的当且仅当与矩阵 \mathbf{A} 对应的有向图是强连通的。有向图 $G = (V, E)$ 是强连通的当且仅当对每一节点对 $u, v \in V$, 存在从 u 到 v 的路径。也就是说, 如果某状态转移矩阵不可约, 则其每个状态都可来自任意的其它状态。
- 周期图 (Periodicity): 说状态 i 是周期的并且具有周期 $k > 1$, 是指存在一个最小的正整数 k , 使得从状态 i 出发又回到状态 i 的所有路径的长度都是 k 的整数倍。如果一个状态不是周期的或者 $k = 1$, 那它就是非周期的。如果一个马尔科夫链的所有状态都是非周期的, 那么就说这个马尔科夫链是非周期的。下图所示, 从状态1出发回到状态1的路径只有一条1-2-3-1, 需要的转移次数是3, 所以这是一个周期为3的马尔科夫链。

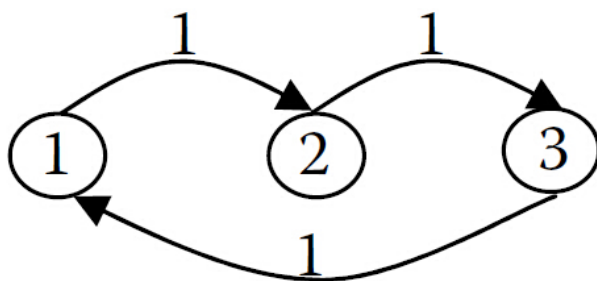


图 5.2: 周期为3的马尔科夫链

现在回到我们前边提出的3个问题。这里能够给前两个问题以肯定回答（也就是 $\lim_{n \rightarrow \infty} \mathbf{P}_n$ 存在且与初值 \mathbf{P}_0 的选取无关）的转移矩阵 \mathbf{A} 应该满足的3个条件：

1. \mathbf{A} 必须是随机矩阵
2. \mathbf{A} 必须是不可约的
3. \mathbf{A} 必须是非周期的

现在我们来考察一下图5.1能否满足上述条件。我们首先来分析第一个条件是否一定成立？答案是否，即 \mathbf{A} 不是随机（转移）矩阵，因为 \mathbf{A} 的第五行的和是0。究其原因，是因为节点5没有出链，节点5代表的网页叫做悬挂网页。互联网上的悬挂网页是很多的，所以我们需要对 \mathbf{A} 中代表悬挂网页的行进行修正。怎么修正呢？我们可以从悬挂网页 i 向每一个网页（包括它自己）引一条链接，就可以解决这个问题。再具体些，将网页 i 到每个网页的转换概率都设为 $\frac{1}{n}$, 相当于均匀分布。

我们引进一个含有 n 个元素的列向量作为悬挂网页的指标向量(indicator vector) \mathbf{a} , 其第 i 个元素取值视第 i 个网页是否为悬挂网页而定，具体地，为悬挂网页取1，否则取0，另外定义一个含有 n 个元素的单位列向量 \mathbf{e} , 则上述思想可以形式化为

$$\mathbf{S} \leftarrow \mathbf{A} + \frac{\mathbf{a}\mathbf{e}^T}{n} \quad (5.1.3)$$

对图5.1的转移矩阵做上述修正后的转移矩阵如下：

$$S = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

解决了悬挂网页的问题后，我们分析 S 是否一定满足第二、第三个条件呢？很容易就可以给予否定。好在我们仅仅用一个策略就可以同时解决这两个问题，将矩阵 S 改造成不可约和非周期的矩阵。

对矩阵 S 修正如下：

$$G \leftarrow \alpha S + (1 - \alpha) \frac{ee^T}{n} \quad (5.1.4)$$

其中 e 是含有 n 个元素的单位列向量，参数 $0 < \alpha < 1$ ，也称阻尼因子，一般取值0.85。

经过公式(3)、(4)的两步修正，我们就把转移矩阵 A 改造成了随机的、不可约的、非周期的转移矩阵 G ，这保证了 $\lim_{n \rightarrow \infty} P_n$ 存在且与初值 P_0 的选取无关。

继续上面的例子，得到 G 如下（ $\alpha = 0.85$ ）：

$$G = \begin{pmatrix} 0.0250 & 0.4500 & 0.4500 & 0.0250 & 0.0250 & 0.0250 \\ 0.4500 & 0.0250 & 0.4500 & 0.0250 & 0.0250 & 0.0250 \\ 0.0250 & 0.8750 & 0.0250 & 0.0250 & 0.0250 & 0.0250 \\ 0.0250 & 0.0250 & 0.3083 & 0.0250 & 0.3083 & 0.3083 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.0250 & 0.0250 & 0.0250 & 0.4500 & 0.4500 & 0.0250 \end{pmatrix}$$

上述对转移矩阵的两步修正的过程确保了PageRank算法在数学上的可行性，然而这是不够的，我们必须对其作出现实意义的解释，这样才能确保我们之前提出的第三个问题，即“网页排序的依据是否真的合理?”。

接下来我们依次对这两个过程进行分析，即对公式(3)、(4)进行解释：

● 对公式(3)的解释：

当互联网用户访问到“悬挂网页”时，不可能“在一棵树上吊死”，而是会自行访问其它网页。对于单个用户来说，自行访问的网页显然与个人兴趣有关，但对于无数的互联网用户的整体来说，自行访问哪个网页完全是随机的。用数学语言来说，这相当于是把转移矩阵 A 中代表悬挂网页的行向量的所有的零向量都换成 $\frac{e}{n}$ （其中 e 是所有分量都为1的行向量， n 为互联网上的网页总数）。用数学语言表达就是公式(3)。

● 对公式(4)的解释：

互联网用户是由一个个活生生的人组成的，他们多少都有自己的“性格”，不会完全受当前网页所限，死板地只访问其所提供的链接。具体地说，他们假定虚拟用户在每一步都有一个小于1的几率 α 访问当前网页所提供的链接，同时却也有一个几率 $1 - \alpha$ 不受那些链接所限，随机访问互联网上的任何一个网站。用数学语言表达就是公式(4)。

通过对公式(3)、(4)的现实意义的解释，我们得到了这样的结论：转移矩阵 G 不但对网页排名的计算提供了数学上的可行的保证，而且它的推导过程具有对应的真实应用场景的强力支撑。

把公式(4)中 G 的表达式替代公式(1)、(2)中的 A ，得到如下的两个公式：

$$P = \left[\alpha \left(A^T + \frac{ee^T}{n} \right) + (1 - \alpha) \frac{ee^T}{n} \right] P \quad (5.1.5)$$

$$\mathbf{P}_{n+1} = \left[\alpha \left(\mathbf{A}^T + \frac{\mathbf{e}\mathbf{a}^T}{n} \right) + (1 - \alpha) \frac{\mathbf{e}\mathbf{e}^T}{n} \right] \mathbf{P}_n, n = 0, 1, \dots, \infty, \text{ given } \mathbf{P}_0 \quad (5.1.6)$$

到这一步之后，摆在我们面前的唯一问题，就是求解了，具体的，确定公式(5)的 \mathbf{P} 或者与之等价的公式(6)的 $\lim_{n \rightarrow \infty} \mathbf{P}_n$ 。

用幂迭代法求解的算法很简单，可以赋予任意的PageRank初始值，每迭代依次PageRank值被修正依次，当PageRank值不再显著变化或者趋近收敛时，迭代算法结束。下面的算法的结束条件是插值向量的一阶范数小于一个预设定的阈值 ϵ 后就结束算法迭代。

Algorithm 4 PageRank-Iterate(\mathbf{G}).

```

1:  $\mathbf{P}_0 \leftarrow \mathbf{e}/n$ 
2:  $k \leftarrow 1$ 
3: repeat
4:    $\mathbf{P}_{k+1} = \left[ \alpha \left( \mathbf{A}^T + \frac{\mathbf{e}\mathbf{a}^T}{n} \right) + (1 - \alpha) \frac{\mathbf{e}\mathbf{e}^T}{n} \right] \mathbf{P}_k$ 
5:    $k \leftarrow k + 1$ 
6: until  $\|\mathbf{P}_k - \mathbf{P}_{k+1}\|_1 < \epsilon$ 
7: return  $\mathbf{P}_k$ 
  
```

公式(6),即 $\mathbf{P}_{k+1} = \mathbf{G}\mathbf{P}_k$ 的平稳分布(即PageRank值)是转移概率矩阵 \mathbf{G} 的最大特征值($= 1$)所对应的特征向量,它对矩阵扰动的敏感性,依赖于它与其它特征值的分离度。若阻尼因子 α 值靠近1,矩阵 \mathbf{G} 的次特征值会随之靠近1,从而导致PageRank对 α 值得选择敏感依赖,计算特征向量算法的收敛速度会降低。也就是说,阻尼因子 α 越小,收敛速度越快。但 α 也不能太小,因为太小的话,“PageRank”中最精华的部分,即以网页间的彼此链接为基础的排序思路就被弱化了(因为这部分的贡献正比于 α)，这显然是得不偿失的。因此，在 α 的选取上有很多折衷的考虑要做，佩奇和布林最终选择的数值是 $\alpha = 0.85$ 。

5.2 PageRank的扩展：Timed-PageRank

互联网上的新网页不断增加，同时不断地有旧的网页变得过时，然而这些旧的网页很多都不会删掉，这会给搜索带来麻烦，因为这些过时的网页往往在过去的长时间里积累了大量入链从而排序很高，而那些载有新信息的高质量新页面却因为没有足够的入链而排序偏低。

为了应对这种情况，Timed-PageRank算法在PageRank算法基础上增加了一个时间维度，它的思想其实很简单，主干上仍然是沿用PageRank的随机上网和马尔科夫链模型，不同之处在于Timed-PageRank不再适用常量阻尼因子 α ，而是引入一个时间递减函数 $f(t)$ ($0 \leq f(t) \leq 1$)来“惩罚”过时的网页（此处的 t 是指当前时间和网页上次更新时间的差值）。函数 $f(t)$ 被定义为点击网页上链接的概率， $1 - f(t)$ 就是不借助网页上的链接而直接跳到一个随机选择的外部网页的概率。那么对于一个特定的网页 i ，有两种选择：

- 以 $f(t)$ 的概率随机选择一个外链跳出。
- 以 $1 - f(t)$ 的概率不通过链接跳到某个随机网页。

5.3 Perron - Frobenius 定理

设 $\mathbf{A} = (a_{ij})$ 为一个 $n \times n$ 的正矩阵： $\forall 1 \leq i, j \leq n, a_{ij} > 0$ ，则该矩阵有以下性质：

1. \mathbf{A} 存在一个正实数的特征值 λ^* ，叫做Perron根或者Perron - Frobenius特征值，使得其它所有特征值(包括复数特征值)的模都比它小。

2. λ^* 只对应一个特征向量 \mathbf{v} 。
3. λ^* 所对应的特征向量 \mathbf{v} 的所有元素都为正实数。
4. λ^* 以外的其它特征值所对应的特征向量的元素至少有一个为负数或复数。
5. $\lim_{k \rightarrow \infty} \frac{\mathbf{A}^k \mathbf{e}}{\lambda^{*k}} = \mathbf{v}$
6. $\min_i \sum_j a_{ij} \leq \lambda^* \leq \max_i \sum_j a_{ij}$

参考文献

- [1] 卢昌海,谷歌背后的数学, http://www.changhai.org/articles/technology/misc/google_math.php, 12/05 2010.
- [2] 姜启源,谢金星,数学建模案例选集.
- [3] Cannel_2020,Google搜索引擎的奥秘, http://blog.csdn.net/cannel_2020/article/details/7672042, 06/18 2012.
- [4] 谱半径, <http://zh.wikipedia.org/wiki/%E8%B0%B1%E5%8D%8A%E5%BE%84>.
- [5] 特征向量的应用: PageRank算法Google搜索引擎的奥秘, <http://f.dataguru.cn/thread-330418-1-1.html>, 07/20 2014.
- [6] 检索的智能化与优化, 第五节Google PageRank算法, http://chaxin.gdinfo.net/web/www/dialog/index.php/Dialog/%E7%B3%BB%E7%BB%9F%E6%A6%82%E8%BF%B0/%E7%B3%BB%E7%BB%9F%E6%A6%82%E8%BF%B0ch5_5.htm.
- [7] 姜启源,数学模型 (第三版)
- [8] Perron - Frobenius theorem, <http://www.cnblogs.com/ZhangShuo/articles/1866748.html>, 10/24 2013.
- [9] Xindongwu, Vipin Kumar,Top 10 Algorithms in Data Mining. ICDM 2006 Panel 12/21/2006
- [10] We Recommend a Singular Value Decomposition, <http://www.ams.org/samplings/feature-column/fcarc-svd>.
- [11] ccjou,實對稱矩陣可正交對角化的證明, <https://ccjou.wordpress.com/2011/02/09/%E5%AF%A6%E5%B0%8D%E7%A8%B1%E7%9F%A9%E9%99%A3%E5%8F%AF%E6%AD%A3%E4%BA%A4%E5%B0%8D%E8%A7%92%E5%8C%96%E7%9A%84%E8%AD%89%E6%98%8E/>, 02/09 2011.
- [12] iMetaSearch,Latent Semantic Analysis (LSA) Tutorial , <http://www.puffinwarellc.com/index.php/news-and-articles/articles/33-latent-semantic-analysis-tutorial.html?start=6>.