分类号 _____ 密级 _____

UDC _____ 编号 _____

中国科学院大学
博士学位论文

机器学习tricks小结

马宏文

指导教师 麦克雷 研究员

中国科学院自动化研究所

申请学位级别 博士 学科专业名称 智能控制与计算智能

论文提交日期 201X年X月 论文答辩日期 201X年X月

培养单位 中国科学院自动化研究所

学位授予单位 中国科学院大学

答辩委员会主席 守望先锋 教授

Summary of machine learning tricks

Howard

Supervisor:

Prof. Mac Ray

Institute of Automation
Chinese Academy of Sciences

November, 2016

Submitted in total fulfilment of the requirements for the degree of Ph.D. in Intelligent control and computational intelligence

摘　　要

随机森林由于鲁棒性强，使用PCA对数据进行降维处理的效果并没有显著变化。

记录机器学习过程中遇到的问题和思考，以及经验性的内容，学而时习之。**关键词：** 人工智能，机器学习

Abstract

Skip

Keywords: Artificial Intelligence, Machine Learning

目　　录

摘要 ..... **i**

Abstract ..... **iii**

目录 ..... **v**

第一章　绪论 **1**

1.1　研究背景和意义 1

1.2　为什么进行经验总结 1

1.3　组织结构 1

1.3.1　论文的研究内容 1

1.3.2　论文的组织结构 1

第二章　Probability **3**

2.1　基础知识 3

2.1.1　离散分布 3

2.1.2　连续分布 3

2.2　随机变量的变换 3

2.3　蒙特卡洛逼近 3

2.3.1　用蒙特卡洛方法估计π 3

2.4　信息论 3

2.4.1　KL散度 3

2.4.2　互信息 3

2.5　本章小结 3

第三章　离散数据的产生式模型 **5**

3.1　贝叶斯概念学习 5

3.1.1　似然 5

3.2　β-binomial 5

3.3　朴素贝叶斯 5

3.3.1　$\log-\sum-\exp$技巧 5

3.3.2　bag of words文档分类 5

3.4　本章小结 5

第四章 高斯模型 **7**
4.1 高斯判别分析 7
4.1.1 二次判别分析(QDA) 7
4.1.2 线性判别分析(LDA) 7
4.1.3 如何避免过拟合 7
4.2 联合高斯分布的推理机制 7
4.3 Wishart分布 7
4.3.1 可视化Wishart分布 7
4.4 多变量下的参数推理 7
4.4.1 μ的后验分布 7
4.4.2 Σ的后验分布 7
4.4.3 μ和Σ的后验分布 7
4.5 本章小结 7

第五章 贝叶斯统计 **9**
5.1 MAP点估计 9
5.2 贝叶斯决策 9
5.3 本章小结 9

第六章 线性回归 **11**
6.1 最大似然估计(least squares) 11
6.1.1 凸性质 11
6.1.2 鲁棒线性回归 11
6.2 岭回归 11
6.3 贝叶斯线性回归 11
6.4 本章小结 11

第七章 频率派统计 **13**
7.1 采样自举 13
7.2 本章小结 13

第八章 Logistic回归 **15**
8.1 拟合方法 15
8.1.1 最速下降法 15
8.1.2 牛顿法 15
8.2 贝叶斯logistic回归 15
8.3 在线学习和随机优化 15
8.3.1 LMS算法 15

8.4 产生式分类器VS. 判别式分类器 15
8.4.1 优缺点对比 15
8.4.2 如何处理丢失数据 15
8.5 本章小结 15

第九章 广义线性模型和指数函数族 17
9.1 基础知识 17
9.2 本章小结 17

第十章 有向图模型(贝叶斯网络) 19
10.1 基础知识 19
10.2 本章小结 19

第十一章 混合模型和EM算法 21
11.1 隐变量模型 21
11.2 EM算法 21
11.3 本章小结 21

第十二章 隐变量线性模型 23
12.1 因子分析(Factor Analysis) 23
12.2 PCA 23
12.3 独立成分分析(ICA) 23
12.4 本章小结 23

第十三章 稀疏线性模型 25
13.1 贝叶斯变量选择 25
13.2 L1正则 25
13.3 非凸正则算子 25
13.4 Automatic relevance determination(ARD) or sparse贝叶斯学习(SBL) 25
13.5 稀疏编码 25
13.6 本章小结 25

第十四章 核方法 27
14.1 各种核函数 27
14.2 Kernel machines 27
14.3 核技巧 27
14.4 支持向量机 27
14.5 本章小结 27

第十五章 高斯过程 **29**
15.1 回归中的高斯过程 29
15.2 GLMs中的高斯过程 29
15.3 其他方法与高斯过程的比较 29
15.3.1 线性模型VS. 高斯过程 29
15.3.2 线性smoothers VS. 高斯过程 29
15.3.3 SVMs VS. 高斯过程 29
15.4 本章小结 29

第十六章 自适应基函数模型 **31**
16.1 分类回归树(CART: Classification and regression trees) 31
16.2 广义加性模型 31
16.3 Boosting 31
16.4 前向神经网络（多层感知机） 31
16.5 黑箱模型 31
16.6 本章小结 31

第十七章 马尔科夫和隐马尔科夫模型 **33**
17.1 模型 33
17.1.1 马尔科夫模型 33
17.1.2 隐马尔科夫模型 33
17.2 HMMs 33
17.3 本章小结 33

第十八章 状态空间模型（SSMs） **35**
18.1 SLAM 35
18.2 LG-SSM 35
18.3 非线性非高斯SSMs中的渐进在线推理 35
18.3.1 Extended Kalman filter (EKF) 35
18.3.2 Unsented Kalman filter (UKF) 35
18.3.3 Assumed density filtering (ADF) 35
18.4 混合状态空间模型（离散+连续）的应用 35
18.4.1 data association and multi-target tracking 35
18.4.2 fault diagnosis 35
18.4.3 economic forecasting 35
18.5 本章小结 35

第十九章 无向图模型（UGMs）—（马尔可夫随机场，MRFs） **37**
19.1 UGMs的性质 37
19.2 MRFs参数化 37
19.2.1 Hammersley-Clifford定理 37
19.3 例子 37
19.4 结构化SVMs 37
19.5 本章小结 37

第二十章 图模型的精确推理 **39**
20.1 信念传播 39
20.2 Variable Elimination(VE)算法 39
20.3 本章小结 39

第二十一章 变分法（Variational inference） **41**
21.1 mean field method 41
21.2 变分贝叶斯 41
21.3 其他变分法 41
21.4 本章小结 41

第二十二章 蒙特卡洛方法 **43**
22.1 Box-Muller方法 43
22.2 拒绝采样 43
22.3 重要性采样 43
22.4 粒子滤波 43
22.5 Rao-Blackwellised粒子滤波 43
22.5.1 应用 43
22.6 本章小结 43

第二十三章 马尔科夫蒙特卡洛（MCMC） **45**
23.1 吉布斯采样 45
23.2 Metropolis Hastings算法 45
23.3 速度和精度 45
23.4 辅助变量 45
23.5 模拟退火方法 45
23.6 marginal likelihood 45
23.7 本章小结 45

第二十四章 聚类 **47**
24.1 Dirichlet过程混合模型 47
24.2 Affinity propagation 47
24.3 谱聚类 47
24.4 本章小结 47

第二十五章 图模型结构化学习 **49**
25.1 知识挖掘中的结构化学习 49
25.2 本章小结 49

第二十六章 离散数据的隐变量模型（LVMs） **51**
26.1 分布式状态LVMs 51
26.2 Latent Dirichlet allocation (LDA) 51
26.3 Restricted Boltzmann machines (RBMs) 51
26.4 本章小结 51

第二十七章 深度学习 **53**
27.1 深度产生式模型 53
27.1.1 深度有向网络 53
27.1.2 深度玻尔兹曼机 53
27.1.3 深度信念网络 53
27.1.4 Greedy layer-wise learning of DBNs 53
27.2 深度神经网络 53
27.2.1 深度多层感知机 53
27.2.2 深度auto-encoders 53
27.2.3 Stacked denoising auto-encoders 53
27.3 A statistical view of deep learning 53
27.3.1 Recursive GLMs 53
27.3.2 Auto-encoder and free energy 54
27.3.3 Memory and Kernels 54
27.4 应用 54
27.4.1 CUDA在Option Pricer中的应用（Python） 54
27.5 讨论 58
27.5.1 Summary of the community's knowledge 58
27.5.2 Neural networks' training tricks 60
27.6 本章小结 60

第二十八章 强化学习 **61**
28.1 策略迭代 61
28.2 值迭代 61
28.3 本章小结 61

第二十九章 集成学习 **63**
29.1 周志华的书 63
29.2 自助法——集成学习大杀器 63
29.2.1 定义 63
29.3 本章小结 63

第三十章 自然语言处理 **65**
30.1 数据预处理手法 65
30.2 常用算法和应用 65
30.3 本章小结 65

第三十一章 图像处理 **67**
31.1 不知道 67
31.2 本章小结 67

第三十二章 启发式算法 **69**
32.1 蚁群算法 69
32.2 粒子群算法 69
32.3 本章小结 69

第三十三章 天池微博大赛 **71**
33.1 注意事项 71
33.1.1 单纯的准确率与优化目标之间不存在等价关系 71
33.1.2 宏观和微观层面的NLP建模 71
33.2 数据处理 71
33.3 特征提取 71
33.4 算法 72
33.4.1 TF-IDF 72
33.4.2 词频方法(Word Frequency） 72
33.4.3 文档频次方法(Document Frequency) 72
33.4.4 互信息(Mutual Information) 73
33.4.5 期望交叉熵(Expected Cross Entropy) 74
33.4.6 二次信息熵(QEMI) 74

33.4.7 信息增益方法(Information Gain) ····· 74
33.4.8 x2统计量方法 ····· 74
33.4.9 文本证据权(The Weight of Evidence forText) ····· 75
33.4.10 优势率(Odds Ratio) ····· 75
33.4.11 遗传算法(Genetic Algorithm, GA) ····· 75
33.4.12 主成分分析法(Principal Component Analysis，PCA) ····· 75
33.4.13 模拟退火算法(Simulating Anneal，SA) ····· 76
33.4.14 N－Gram算法 ····· 76
33.4.15 各种方法的综合评价 ····· 76
33.5 本章小结 ····· 81

第三十四章 Kaggle Crowdflower Competition ····· 83
34.1 评判函数 ····· 83
34.2 代码 ····· 83
34.3 本章小结 ····· 85

第三十五章 神兵利器 ····· 87
35.1 大数据分析与机器学习领域Python兵器谱 ····· 87
35.1.1 Python网页爬虫工具集 ····· 87
35.1.2 Python文本处理工具集 ····· 87
35.1.3 Python科学计算工具包 ····· 89
35.1.4 Python 机器学习& 数据挖掘工具包 ····· 90
35.2 代码 ····· 93
35.3 本章小结 ····· 93

第三十六章 问题集 ····· 95
36.1 第一部分标题还未想好 ····· 95
36.2 本章小结 ····· 96

第三十七章 总结与展望 ····· 97
37.1 本文工作总结 ····· 97
37.2 工作展望 ····· 97

简历 ····· 99

表　　格

插　　图

27.1 CUDA VS. CPU 57

33.1 NLP特征下的转评赞对比 71

第一章　绪论

1.1　研究背景和意义

未完待续

1.2　为什么进行经验总结

学习轨迹

1.3　组织结构

1.3.1　论文的研究内容

社

1.3.2　论文的组织结构

本论文共分七章，内容安排如下：

第一章是绪论，

第二章　Probability

概率论知识[?]

2.1　基础知识

2.1.1　离散分布

2.1.2　连续分布

2.2　随机变量的变换

2.3　蒙特卡洛逼近

2.3.1　用蒙特卡洛方法估计π

2.4　信息论

2.4.1　KL散度

2.4.2　互信息

2.5　本章小结

本章首先介绍了。。。

第三章　离散数据的产生式模型

概率论知识

3.1　贝叶斯概念学习

3.1.1　似然

3.2　β-binomial

3.3　朴素贝叶斯

3.3.1　$\log - \sum - \exp$技巧

3.3.2　bag of words文档分类

3.4　本章小结

本章首先介绍了。。。

第四章　高斯模型

介绍

4.1　高斯判别分析

4.1.1　二次判别分析(QDA)

4.1.2　线性判别分析(LDA)

4.1.3　如何避免过拟合

4.2　联合高斯分布的推理机制

4.3　Wishart分布

4.3.1　可视化Wishart分布

4.4　多变量下的参数推理

4.4.1　μ的后验分布

4.4.2　Σ的后验分布

4.4.3　μ和Σ的后验分布

4.5　本章小结

本章首先介绍了。。。

第五章　贝叶斯统计

5.1　MAP点估计

5.2　贝叶斯决策

5.3　本章小结

本章首先介绍了。。。

第六章　线性回归

6.1　最大似然估计(least squares)

6.1.1　凸性质

6.1.2　鲁棒线性回归

6.2　岭回归

6.3　贝叶斯线性回归

6.4　本章小结

本章首先介绍了。。。

第七章　频率派统计

概率论知识

7.1　采样自举

7.2　本章小结

本章首先介绍了。。。

第八章　Logistic回归

8.1　拟合方法

8.1.1　最速下降法

8.1.2　牛顿法

8.2　贝叶斯logistic回归

8.3　在线学习和随机优化

8.3.1　LMS算法

8.4　产生式分类器VS. 判别式分类器

8.4.1　优缺点对比

8.4.2　如何处理丢失数据

8.5　本章小结

本章首先介绍了。。。

第九章　广义线性模型和指数函数族

9.1　基础知识

9.2　本章小结

本章首先介绍了。。。

第十章　有向图模型(贝叶斯网络)

10.1　基础知识

10.2　本章小结

本章首先介绍了。。。

第十一章　混合模型和EM算法

11.1　隐变量模型

11.2　EM算法

11.3　本章小结

本章首先介绍了。。。

第十二章　隐变量线性模型

12.1　因子分析(Factor Analysis)

12.2　PCA

12.3　独立成分分析(ICA)

12.4　本章小结

本章首先介绍了。。。

第十三章　稀疏线性模型

13.1　贝叶斯变量选择

13.2　L1正则

13.3　非凸正则算子

13.4　Automatic relevance determination(ARD) or sparse贝叶斯学习(SBL)

13.5　稀疏编码

13.6　本章小结

本章首先介绍了。。。

第十四章　核方法

14.1　各种核函数

RBF、Mercer核、线性核、Matern核、String核、金字塔匹配核

14.2　Kernel machines

14.3　核技巧

14.4　支持向量机

14.5　本章小结

本章首先介绍了。。。

第十五章　高斯过程

15.1　回归中的高斯过程

15.2　GLMs中的高斯过程

15.3　其他方法与高斯过程的比较

15.3.1　线性模型VS. 高斯过程

15.3.2　线性smoothers VS. 高斯过程

15.3.3　SVMs VS. 高斯过程

15.4　本章小结

本章首先介绍了。。。

第十六章　自适应基函数模型

16.1　分类回归树(CART: Classification and regression trees)

16.2　广义加性模型

16.3　Boosting

16.4　前向神经网络（多层感知机）

16.5　黑箱模型

16.6　本章小结

本章首先介绍了。。。

第十七章　马尔科夫和隐马尔科夫模型

17.1　模型

17.1.1　马尔科夫模型

17.1.2　隐马尔科夫模型

17.2　HMMs

17.3　本章小结

本章首先介绍了。。。

第十八章　状态空间模型（SSMs）

18.1　SLAM

18.2　LG-SSM

18.3　非线性非高斯SSMs中的渐进在线推理

18.3.1　Extended Kalman filter (EKF)

18.3.2　Unsented Kalman filter (UKF)

18.3.3　Assumed density filtering (ADF)

18.4　混合状态空间模型（离散+连续）的应用

18.4.1　data association and multi-target tracking

18.4.2　fault diagnosis

18.4.3　economic forecasting

18.5　本章小结

本章首先介绍了。。。

第十九章　无向图模型（UGMs）——（马尔可夫随机场，MRFs）

19.1　UGMs的性质

19.2　MRFs参数化

19.2.1　Hammersley-Clifford定理

19.3　例子

19.4　结构化SVMs

19.5　本章小结

本章首先介绍了。。。

第二十章　图模型的精确推理

20.1　信念传播

20.2　Variable Elimination(VE)算法

20.3　本章小结

本章首先介绍了。。。

第二十一章　变分法（Variational inference）

21.1　mean field method

21.2　变分贝叶斯

21.3　其他变分法

21.4　本章小结

本章首先介绍了。。。

第二十二章　蒙特卡洛方法

22.1　Box-Muller方法

22.2　拒绝采样

22.3　重要性采样

22.4　粒子滤波

22.5　Rao-Blackwellised粒子滤波

22.5.1　应用

22.6　本章小结

本章首先介绍了。。。

第二十三章　马尔科夫蒙特卡洛（MCMC）

23.1　吉布斯采样

23.2　Metropolis Hastings算法

23.3　速度和精度

23.4　辅助变量

23.5　模拟退火方法

23.6　marginal likelihood

23.7　本章小结

本章首先介绍了。。。

第二十四章　聚类

24.1　Dirichlet过程混合模型

24.2　Affinity propagation

24.3　谱聚类

24.4　本章小结

本章首先介绍了。。。

第二十五章　图模型结构化学习

25.1　知识挖掘中的结构化学习

25.2　本章小结

本章首先介绍了。。。

第二十六章　离散数据的隐变量模型（LVMs）

26.1　分布式状态LVMs

26.2　Latent Dirichlet allocation (LDA)

26.3　Restricted Boltzmann machines (RBMs)

26.4　本章小结

本章首先介绍了。。。

第二十七章　深度学习

27.1　深度产生式模型

27.1.1　深度有向网络

27.1.2　深度玻尔兹曼机

27.1.3　深度信念网络

27.1.4　Greedy layer-wise learning of DBNs

27.2　深度神经网络

27.2.1　深度多层感知机

27.2.2　深度auto-encoders

27.2.3　Stacked denoising auto-encoders

27.3　A statistical view of deep learning

27.3.1　Recursive GLMs

27.3.1.1　Generalized Linear Models

The basic linear regression model is a linear mapping from $P-$dimensional input features (or covariates) x, to a set of targets (or responses) y, using a set of weights (or regression coefficients) β and a bias (offset) β_0 . The outputs can also by multivariate, but I'll assume they are scalar here. The full probabilistic model assumes that the outputs are corrupted by Gaussian noise of unknown variance σ^2.

$$\eta = \beta^{\mathsf{T}} x + \beta_0, \quad y = \eta + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{27.1}$$

In this formulation, η is the systematic component of the model and ε is the random component. Generalised linear models (GLMs) allow us to extend this formulation to problems where the distribution on the targets is not Gaussian but some other distribution (typically a distribution in the exponential family). In this case, we can write the generalised regression problem, combining the coefficients and bias for more compact notation, as:

$$\eta = \beta^{\mathsf{T}} x, \quad \beta = [\hat{\beta}, \beta_0], \quad x = [\hat{x}, 1], \quad \mathbb{E}[y] = \mu = g^{-1}(\eta)$$

where $g(\cdot)$ is the link function that allows us to move from natural parameters η to mean parameters μ. If the inverse link function used in the definition of μ above were the logistic sigmoid, then the mean parameters correspond to the probabilities of y being a 1 or 0 under the Bernoulli distribution. There are many link functions that allow us to make other distributional assumptions for the target (response) y. In deep learning, the link function is referred to as the activation function and I list

in the table below the names for these functions used in the two fields. From this table we can see that many of the popular approaches for specifying neural networks that have counterparts in statistics and related literatures under (sometimes) very different names, such multinomial regression in statistics and softmax classification in deep learning, or rectifier in deep learning and tobit models is statistics.

Target Type	Regression	Link	Inv Link	Activation
Real	Linear	Identity	Identity	None
Binary	Logistic	Logit $\log \frac{\mu}{1-\mu}$	Sigmoid $\sigma = \frac{1}{1+\exp(-\eta)}$	Sigmoid
Binary	Probit	Inv Guass CDF $\Phi^{-1}(\mu)$	Guass CDF $\Phi(\eta)$	Probit
Binary	Gumbel	Compl. log-log $\log(-\log(\mu))$	Gumbel CDF $e^{-e^{-x}}$	None
Binary	Logistic	None	Hyperbolic Tangent $\tanh(\eta)$	tanh
Categorical	Multinomial	None	Multin. Logit $\frac{\eta_i}{\sum_j \eta_j}$	Softmax
Counts	Poisson	$\log(\mu)$	$\exp(\nu)$	None
Counts	Poisson	$\sqrt{\mu}$	ν^2	None
Non-Neg.	Gamma	Reciprocal $\frac{1}{\mu}$	$\frac{1}{\nu}$	None
Sparse	Tobit	None	max $\max(0, \nu)$	ReLU
Ordered	Ordinal	None	Cum.Logit $\sigma(\phi_k - \eta)$	None

参考链接：http://blog.shakirm.com/2015/01/a-statistical-view-of-deep-learning-i-recurs

27.3.2 Auto-encoder and free energy

Instead of considering perturbations in the observation space, we consider perturbations in the hidden space, obtained by using a prior $p(z)$. The hidden variables are now random, latent variables. Auto-encoders are now generative models that are straightforward to sample from. The encoder $q(z|y)$ is a mechanism for approximating the true posterior distribution of the latent/hidden variables $p(z|y)$. We are now able to explain the introduction of the penalty function in the GDAE objective in a principled manner. Rather than designing the penalty by hand, we are able to derive the form this penalty should take, appearing as the KL divergence between the the prior and the encoder distribution.

参考链接：http://blog.shakirm.com/2015/03/a-statistical-view-of-deep-learning-ii-auto-

27.3.3 Memory and Kernels

http://blog.shakirm.com/2015/04/a-statistical-view-of-deep-learning-iii-memory-and-ke

27.4 应用

27.4.1 CUDA在Option Pricer中的应用（Python）

Remark 1.

numbapro集成于numba中，其中的cudalib包改到了accelerate包中的cuda函数库里。另外，如果不在环境变量里手动添加CUDA库的路径，就需要在运行时调用os包添加环境变量，但只是在此程序运行时有效。具体见代码。

Listing 27.1: MC_OptionPricer.py

```
# Website: http://nbviewer.jupyter.org/gist/harrism/835a8ca39ced77fe751d#CUDA-Vectorize
import numpy as np                          # numpy namespace
from timeit import default_timer as timer   # for timing
from matplotlib import pyplot               # for plotting
import math

def step_numpy(dt, prices, c0, c1, noises):
    return prices * np.exp(c0 * dt + c1 * noises)

def mc_numpy(paths, dt, interest, volatility):
    c0 = interest - 0.5 * volatility ** 2
    c1 = volatility * np.sqrt(dt)

    for j in xrange(1, paths.shape[1]):    # for each time step
        prices = paths[:, j - 1]           # last prices
        # gaussian noises for simulation
        noises = np.random.normal(0., 1., prices.size)
        # simulate
        paths[:, j] = step_numpy(dt, prices, c0, c1, noises)

# Configurations
# stock parameter
StockPrice = 20.83
StrikePrice = 21.50
Volatility = 0.021
InterestRate = 0.20
Maturity = 5. / 12.
# monte-carlo parameter
NumPath = 3000000
NumStep = 100
# plotting
MAX_PATH_IN_PLOT = 50


# Driver: The driver measures the performance of the given pricer and plots the simulation paths.
def driver(pricer, do_plot=False):
    paths = np.zeros((NumPath, NumStep + 1), order='F')
    paths[:, 0] = StockPrice
    DT = Maturity / NumStep

    ts = timer()
    pricer(paths, DT, InterestRate, Volatility)
    te = timer()
    elapsed = te - ts

    ST = paths[:, -1]
    PaidOff = np.maximum(paths[:, -1] - StrikePrice, 0)
    print 'Result'
    fmt = '%20s: %s'
    print fmt % ('stock price', np.mean(ST))
    print fmt % ('standard error', np.std(ST) / np.sqrt(NumPath))
    print fmt % ('paid off', np.mean(PaidOff))
    optionprice = np.mean(PaidOff) * np.exp(-InterestRate * Maturity)
    print fmt % ('option price', optionprice)

    print 'Performance'
    NumCompute = NumPath * NumStep
    print fmt % ('Mstep/second', '%.2f' % (NumCompute / elapsed / 1e6))
    print fmt % ('time elapsed', '%.3fs' % (te - ts))

    if do_plot:
        pathct = min(NumPath, MAX_PATH_IN_PLOT)
        for i in xrange(pathct):
            pyplot.plot(paths[i])
        print 'Plotting %d/%d paths' % (pathct, NumPath)
        pyplot.show()
    return elapsed


from numba import vectorize
# Basic vectorize
@vectorize(['f8(f8, f8, f8, f8, f8)'])
def step_cpuvec(last, dt, c0, c1, noise):
```

```
    return last * math.exp(c0 * dt + c1 * noise)

def mc_cpuvec(paths, dt, interest, volatility):
    c0 = interest - 0.5 * volatility ** 2
    c1 = volatility * np.sqrt(dt)

    for j in xrange(1, paths.shape[1]):
        prices = paths[:, j - 1]
        noises = np.random.normal(0., 1., prices.size)
        paths[:, j] = step_cpuvec(prices, dt, c0, c1, noises)


# Paralell with multithread implementation
@vectorize(['f8(f8,_f8,_f8,_f8,_f8)'], target='parallel')
def step_parallel(last, dt, c0, c1, noise):
    return last * math.exp(c0 * dt + c1 * noise)

def mc_parallel(paths, dt, interest, volatility):
    c0 = interest - 0.5 * volatility ** 2
    c1 = volatility * np.sqrt(dt)

    for j in xrange(1, paths.shape[1]):
        prices = paths[:, j - 1]
        noises = np.random.normal(0., 1., prices.size)
        paths[:, j] = step_parallel(prices, dt, c0, c1, noises)


# CUDA Vectorize
import os
os.environ['NUMBAPRO_NVVM']=r'C:\Program_Files\NVIDIA_GPU_Computing_Toolkit\CUDA\v8.0\nvvm\bin\nvvm64_31_0.dll'
os.environ['NUMBAPRO_LIBDEVICE']=r'C:\Program_Files\NVIDIA_GPU_Computing_Toolkit\CUDA\v8.0\nvvm\libdevice'

@vectorize(['f8(f8,_f8,_f8,_f8,_f8)'], target='cuda')
def step_gpuvec(last, dt, c0, c1, noise):
    return last * math.exp(c0 * dt + c1 * noise)

def mc_gpuvec(paths, dt, interest, volatility):
    c0 = interest - 0.5 * volatility ** 2
    c1 = volatility * np.sqrt(dt)

    for j in xrange(1, paths.shape[1]):
        prices = paths[:, j - 1]
        noises = np.random.normal(0., 1., prices.size)
        paths[:, j] = step_gpuvec(prices, dt, c0, c1, noises)



# CUDA JIT
from numba import cuda, jit
from accelerate.cuda import rand

@jit('void(double[:], _double[:], _double,_double,_double,_double[:])', target='cuda')
def step_cuda(last, paths, dt, c0, c1, normdist):
    i = cuda.grid(1)
    if i >= paths.shape[0]:
        return
    noise = normdist[i]
    paths[i] = last[i] * math.exp(c0 * dt + c1 * noise)

def mc_cuda(paths, dt, interest, volatility):
    n = paths.shape[0]

    blksz = cuda.get_current_device().MAX_THREADS_PER_BLOCK
    gridsz = int(math.ceil(float(n) / blksz))

    # instantiate a CUDA stream for queueing async CUDA cmds
    stream = cuda.stream()
    # instantiate a cuRAND PRNG
    prng = rand.PRNG(rand.PRNG.MRG32K3A, stream=stream)

    # Allocate device side array
    d_normdist = cuda.device_array(n, dtype=np.double, stream=stream)

    c0 = interest - 0.5 * volatility ** 2
```

```
    c1 = volatility * math.sqrt(dt)

    # configure the kernel
    # similar to CUDA-C: step_cuda<<<gridsz, blksz, 0, stream>>>
    step_cfg = step_cuda[gridsz, blksz, stream]

    # transfer the initial prices
    d_last = cuda.to_device(paths[:, 0], stream=stream)
    for j in range(1, paths.shape[1]):
        # call cuRAND to populate d_normdist with gaussian noises
        prng.normal(d_normdist, mean=0, sigma=1)
        # setup memory for new prices
        # device_array_like is like empty_like for GPU
        d_paths = cuda.device_array_like(paths[:, j], stream=stream)
        # invoke step kernel asynchronously
        step_cfg(d_last, d_paths, dt, c0, c1, d_normdist)
        # transfer memory back to the host
        d_paths.copy_to_host(paths[:, j], stream=stream)
        d_last = d_paths
    # wait for all GPU work to complete
    stream.synchronize()


# Results comparisons
numpy_time = driver(mc_numpy, do_plot=True) # numpy
cpuvec_time = driver(mc_cpuvec, do_plot=True) # Basic vectorize
parallel_time = driver(mc_parallel, do_plot=True) # Parallel
gpuvec_time = driver(mc_gpuvec, do_plot=True) # CUDA Vectorize
cuda_time = driver(mc_cuda, do_plot=True) # CUDA JIT

def perf_plot(rawdata, xlabels):
    data = [numpy_time / x for x in rawdata]
    idx = np.arange(len(data))
    fig = pyplot.figure()
    width = 0.5
    ax = fig.add_subplot(111)
    ax.bar(idx, data, width)
    ax.set_ylabel('normalized_speedup')
    ax.set_xticks(idx + width / 2)
    ax.set_xticklabels(xlabels)
    ax.set_ylim(0.9)

perf_plot([numpy_time, cpuvec_time, parallel_time, gpuvec_time],
          ['numpy', 'cpu-vect', 'parallel-vect', 'gpu-vect'])
perf_plot([numpy_time, cpuvec_time, parallel_time, gpuvec_time, cuda_time],
          ['numpy', 'cpu-vect', 'parallel-vect', 'gpu-vect', 'cuda'])

# Save memory for next usage
# import gc
# gc.collect()
```

代码运行结果见如下对比图： 图27.1中可看出CUDA的性能高出普通CPU计算近10倍，但是此

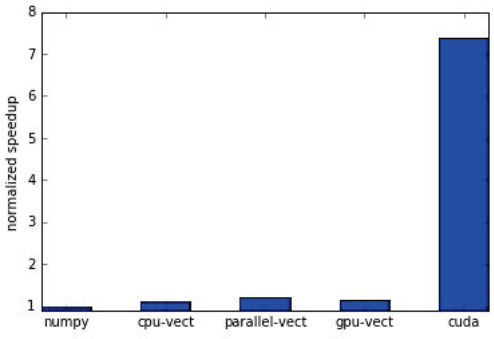

图 27.1: CUDA VS. CPU

本ThinkPad P50使用的是NVIDIA Quadro M1000M，性能相比GTX 960M差了不少。

27.5 讨论

27.5.1 Summary of the community's knowledge

Here is a summary of the community' s knowledge of what' s important and what to look after:

1. Preprocessing: it is essential to center the data so that its mean is zero and so that the variance of each of its dimensions is one. Sometimes, when the input dimension varies by orders of magnitude, it is better to take the $\log(1+x)$ of that dimension. Basically, it' s important to find a faithful encoding of the input with zero mean and sensibly bounded dimensions. Doing so makes learning work much better. This is the case because the weights are updated by the formula: change in w_{ij} proportional to $\frac{x_i \mathrm{d}L}{\mathrm{d}y_j}$ (w denotes the weights from layer x to layer y, and L is the loss function). If the average value of the x' s is large (say, 100), then the weight updates will be very large and correlated, which makes learning bad and slow. Keeping things zero-mean and with small variance simply makes everything work much better.

2. Get the data: Make sure that you have a high-quality dataset of input-output examples that is large, representative, and has relatively clean labels. Learning is completely impossible without such a dataset.

3. Minibatches: Use minibatches. Modern computers cannot be efficient if you process one training case at a time. It is vastly more efficient to train the network on minibatches of 128 examples, because doing so will result in massively greater throughput. It would actually be nice to use minibatches of size 1, and they would probably result in improved performance and lower over-fitting; but the benefit of doing so is outweighed the massive computational gains provided by minibatches. But do not use very large minibatches because they tend to work less well and overfit more. So the practical recommendation is: use the smaller minibatch that runs efficiently on your machine.

4. Gradient normalization: Divide the gradient by minibatch size. This is a good idea because of the following pleasant property: you won' t need to change the learning rate (not too much, anyway), if you double the minibatch size (or halve it).

5. Learning rate schedule: Start with a normal-sized learning rate (LR) and reduce it towards the end. A typical value of the LR is 0.1. Amazingly, 0.1 is a good value of the learning rate for a large number of neural networks problems. Learning rates frequently tend to be smaller but rarely much larger. Use a validation set – a subset of the training set on which we don't train – to decide when to lower the learning rate and when to stop training (e.g., when error on the validation set starts to increase). A practical suggestion for a learning rate schedule: if you see that you stopped making progress on the validation set, divide the LR by 2 (or by 5), and keep going. Eventually, the LR will become very small, at which point you will stop your training. Doing so helps ensure that you won' t be (over-)fitting the training data at the detriment of validation performance, which happens easily and often. Also, lowering the LR

is important, and the above recipe provides a useful approach to controlling via the validation set.

6. But most importantly, worry about the Learning Rate. One useful idea used by some researchers (e.g., Alex Krizhevsky) is to monitor the ratio between the update norm and the weight norm. This ratio should be at around 10^{-3}. If it is much smaller then learning will probably be too slow, and if it is much larger then learning will be unstable and will probably fail.

7. Weight initialization. Worry about the random initialization of the weights at the start of learning. You're now informed. Worry and care about your initialization. Try many different kinds of initialization. This effort will pay off. If the net doesn't work at all (i.e., never "gets off the ground"), keep applying pressure to the random initialization. It's the right thing to do. If you are lazy, it is usually enough to do something like 0.02 * randn(num_params). A value at this scale tends to work surprisingly well over many different problems. Of course, smaller (or larger) values are also worth trying. If it doesn't work well (say your neural network architecture is unusual and/or very deep), then you should initialize each weight matrix with the $\frac{init\_scale}{sqrt(layer\_width)} * randn$. In this case init_scale should be set to 0.1 or 1, or something like that. Random initialization is super important for deep and recurrent nets. If you don't get it right, then it'll look like the network doesn't learn anything at all. But we know that neural networks learn once the conditions are set. Fun story: researchers believed, for many years, that SGD cannot train deep neural networks from random initializations. Every time they would try it, it wouldn't work. Embarrassingly, they did not succeed because they used the"small random weight" for the initialization, which works great for shallow nets but simply doesn't work for deep nets at all. When the nets are deep, the many weight matrices all multiply each other, so the effect of a suboptimal scale is amplified. But if your net is shallow, you can afford to be less careful with the random initialization, since SGD will just find a way to fix it.

8. If you are training RNNs or LSTMs, use a hard constraint over the norm of the gradient (remember that the gradient has been divided by batch size). Something like 15 or 5 works well in practice in my own experiments. Take your gradient, divide it by the size of the minibatch, and check if its norm exceeds 15 (or 5). If it does, then shrink it until it is 15 (or 5). This one little trick plays a huge difference in the training of RNNs and LSTMs, where otherwise the exploding gradient can cause learning to fail and force you to use a puny learning rate like $1e^{-6}$ which is too small to be useful.

9. Numerical gradient checking: If you are not using Theano or Torch, you'll be probably implementing your own gradients. It is easy to make a mistake when we implement a gradient, so it is absolutely critical to use numerical gradient checking. Doing so will give you a complete peace of mind and confidence in your code. You will know that you can invest effort in tuning the hyper-parameters (such as the learning rate and the initialization) and be sure that your

efforts are channeled in the right direction.

10. If you are using LSTMs and you want to train them on problems with very long range dependencies, you should initialize the biases of the forget gates of the LSTMs to large values. By default, the forget gates are the *sigmoids* of their total input, and when the weights are small, the forget gate is set to 0.5, which is adequate for some but not all problems. This is the one non-obvious caveat about the initialization of the LSTM.

11. Data augmentation: be creative, and find ways to algorithmically increase the number of training cases that are in your disposal. If you have images, then you should translate and rotate them; if you have speech, you should combine clean speech with all types of random noise; etc. Data augmentation is an art (unless you're dealing with images). Use common sense.

12. Dropout. Dropout provides an easy way to improve performance. It's trivial to implement and there's little reason to not do it. Remember to tune the dropout probability, and to not forget to turn off Dropout and to multiply the weights by (namely by 1-dropout probability) at test time. Also, be sure to train the network for longer. Unlike normal training, where the validation error often starts increasing after prolonged training, dropout nets keep getting better and better the longer you train them. So be patient.

13. Ensembling. Train 10 neural networks and average their predictions. It's a fairly trivial technique that results in easy, sizeable performance improvements. One may be mystified as to why averaging helps so much, but there is a simple reason for the effectiveness of averaging. Suppose that two classifiers have an error rate of 70%. Then, when they agree they are right. But when they disagree, one of them is often right, so now the average prediction will place much more weight on the correct answer. The effect will be especially strong whenever the network is confident when it's right and unconfident when it's wrong.

27.5.2 Neural networks' training tricks

http://m.blog.csdn.net/article/details?id=45288129

参考书：Neural networks–Tricks of the trade

27.6 本章小结

本章首先介绍了。。。

第二十八章　强化学习

28.1　策略迭代

28.2　值迭代

28.3　本章小结

本章首先介绍了。。。

第二十九章　集成学习

29.1　周志华的书

29.2　自助法——集成学习大杀器

29.2.1　定义

给定包含m个样本的数据集D，每次随机从D中挑选一个样本，将其拷贝放入D'中，并放回样本。重复执行m次，得到包含m个样本的数据集D'，而剩下的从未被采集到的样本$D\backslash D'$作为测试集。未被采集到的样本所占比例：

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m \to \frac{1}{e}\approx 0.368" \tag{29.1}$$

这种方法称为“包外估计”（out-of-bag estimate）。

Remark 2.
自助法在数据集较小、难以有效划分训练集和测试集的情况下很有用，但会引入估计偏差，因而在数据足够的情况下宜使用留出法和交叉验证法。

29.3　本章小结

本章首先介绍了。。。

第三十章　自然语言处理

30.1　数据预处理手法

30.2　常用算法和应用

30.3　本章小结

本章首先介绍了。。。

第三十一章　图像处理

31.1　不知道

31.2　本章小结

本章首先介绍了。。。

第三十二章　启发式算法

32.1　蚁群算法

32.2　粒子群算法

32.3　本章小结

本章首先介绍了。。。

第三十三章　天池微博大赛

33.1　注意事项

33.1.1　单纯的准确率与优化目标之间不存在等价关系

举个例子：一个人发了100条微博，其中90条都是0转发，0评论，0点赞，10条都是33转发，33评论，33点赞，现在有两种策略：
（1）预测这100条微博的结果都是全0的
（2）预测这100条微博的结果都是全33的
如果以准确率为目标，方案（1）的准确率是方案（2）的约10倍；如果以本题的评分方案为目标，方案（2）的得分是方案（1）的约10倍；（当然这是个极端的例子，但可以说明：准确率最高与本题中的评分结果最高是有着很大差距的）如果使用准确率最高作为目标而不考虑权重影响的话，训练出来的目标很容易出问题，导致评分较低。如何解决权重这一关键问题呢？我推荐两种方案：
（1）将预测值分为几档。如0-5,5-10,10-100等等，将预测连续值变为一个分类问题，然后利用置信度和权值计算出一个能够使最终得分数学期望最高的预测值。
（2）直接使用评分目标作为最终目标。我们的目标不再是提高一条微博的预测准确率，而是从宏观上提高整体的得分。不过这需要自己动手来改造或实现一个模型，GBDT，神经网络，遗传算法，或是自己设计的简单模型。但这个模型要做到：每迭代一次，都需要使用新产生的模型计算测试集一个月微博的得分，然后向着最优的方向继续迭代。

33.1.2　宏观和微观层面的NLP建模

NLP特征的使用可分为宏观和微观两种，宏观上是指情感分析，主题抽取等句子维度的聚合特征，微观上是指词维度的统计特征。下图中可以看出不同词的统计结果差距还是很大的。如下图33.1所示：

原词	出现该词微博平均转发/所有微博平均转发	出现该词微博平均评论/所有微博平均评论	出现该词微博平均点赞/所有微博平均点赞
转发	25.87	12.29	2.38
红包	0.026	0.087	0.042

图 33.1: NLP特征下的转评赞对比

33.2　数据处理

发现有1214 个用户在训练集中不存在历史数据。

33.3　特征提取

构建特征：微博长度，是否包含链接，是否包含hashtag#，是否@他人，计算微博的相似度，构建兴趣集，计算活跃度。

33.4 算法

33.4.1 TF-IDF

单词权重最为有效的实现方法就是TF*IDF, 它是由Salton在1988 年提出的。其中TF 称为词频, 用于计算该词描述文档内容的能力; IDF 称为反文档频率, 用于计算该词区分文档的能力。TF*IDF 的指导思想建立在这样一条基本假设之上: 在一个文本中出现很多次的单词, 在另一个同类文本中出现次数也会很多, 反之亦然。所以如果特征空间坐标系取TF 词频作为测度, 就可以体现同类文本的特点。另外还要考虑单词区别不同类别的能力, TF*IDF 法认为一个单词出现的文本频率越小, 它区别不同类别的能力就越大, 所以引入了逆文本频度IDF 的概念, 以TF 和IDF 的乘积作为特征空间坐标系的取值测度。

TFIDF 法是以特征词在文档d中出现的次数与包含该特征词的文档数之比作为该词的权重，即其中，W_i表示第i个特征词的权重，$\mathrm{TF}_i(td)$表示词t在文档d中的出现频率，N表示总的文档数，DF(t)表示包含t的文档数。用TFIDF算法来计算特征词的权重值是表示当一个词在这篇文档中出现的频率越高，同时在其他文档中出现的次数越少，则表明该词对于表示这篇文档的区分能力越强，所以其权重值就应该越大。将所有词的权值排序, 根据需要可以有两种选择方式:
((1) 选择权值最大的某一固定数n 个关键词;
((2) 选择权值大于某一阈值的关键词。一些实验表示,人工选择关键词, 4∽7 个比较合适, 机选关键词10∽15 通常具有最好的覆盖度和专指度。

TFIDF算法是建立在这样一个假设之上的：对区别文档最有意义的词语应该是那些在文档中出现频率高，而在整个文档集合的其他文档中出现频率少的词语，所以如果特征空间坐标系取TF词频作为测度，就可以体现同类文本的特点。另外考虑到单词区别不同类别的能力，TFIDF法认为一个单词出现的文本频数越小，它区别不同类别文本的能力就越大。因此引入了逆文本频度IDF的概念，以TF和IDF的乘积作为特征空间坐标系的取值测度，并用它完成对权值TF的调整，调整权值的目的在于突出重要单词，抑制次要单词。但是在本质上IDF是一种试图抑制噪音的加权，并且单纯地认为文本频数小的单词就越重要，文本频数大的单词就越无用，显然这并不是完全正确的。IDF的简单结构并不能有效地反映单词的重要程度和特征词的分布情况，使其无法很好地完成对权值调整的功能，所以TFIDF法的精度并不是很高。

此外，在TFIDF算法中并没有体现出单词的位置信息，对于Web文档而言，权重的计算方法应该体现出HTML的结构特征。特征词在不同的标记符中对文章内容的反映程度不同，其权重的计算方法也应不同。因此应该对于处于网页不同位置的特征词分别赋予不同的系数，然后乘以特征词的词频，以提高文本表示的效果。

33.4.2 词频方法(Word Frequency）

词频是一个词在文档中出现的次数。通过词频进行特征选择就是将词频小于某一闭值的词删除，从而降低特征空间的维数。这个方法是基于这样一个假设，即出现频率小的词对过滤的影响也较小。但是在信息检索的研究中认为，有时频率小的词含有更多的信息。因此，在特征选择的过程中不宜简单地根据词频大幅度删词。

33.4.3 文档频次方法(Document Frequency)

文档频数(Document Frequency, DF)是最为简单的一种特征选择算法,它指的是在整个数据集中有多少个文本包含这个单词。在训练文本集中对每个特征计一算它的文档频次，并且根据预先

设定的阑值去除那些文档频次特别低和特别高的特征。文档频次通过在训练文档数量中计算线性近似复杂度来衡量巨大的文档集，计算复杂度较低，能够适用于任何语料，因此是特征降维的常用方法。

在训练文本集中对每个特征计算它的文档频数,若该项的DF 值小于某个阈值则将其删除,若其DF值大于某个阈值也将其去掉。因为他们分别代表了“没有代表性”和“没有区分度”2 种极端的情况。DF 特征选取使稀有词要么不含有用信息,要么太少而不足以对分类产生影响,要么是噪音,所以可以删去。DF 的优点在于计算量很小,而在实际运用中却有很好的效果。缺点是稀有词可能在某一类文本中并不稀有,也可能包含着重要的判断信息,简单舍弃,可能影响分类器的精度。

文档频数最大的优势就是速度快,它的时间复杂度和文本数量成线性关系,所以非常适合于超大规模文本数据集的特征选择。不仅如此,文档频数还非常地高效,在有监督的特征选择应用中当删除90%单词的时候其性能与信息增益和x2 统计的性能还不相上下。DF 是最简单的特征项选取方法,而且该方法的计算复杂度低, 能够胜任大规模的分类任务。

但如果某一稀有词条主要出现在某类训练集中,却能很好地反映类别的特征,而因低于某个设定的阈值而滤除掉,这样就会对分类精度有一定的影响。

33.4.4　互信息(Mutual Information)

互信息衡量的是某个词和类别之间的统计独立关系,某个词t和某个类别Ci传统的互信息定义如下:

互信息是计算语言学模型分析的常用方法，它度量两个对象之间的相互性。在过滤问题中用于度量特征对于主题的区分度。互信息的定义与交叉嫡近似。互信息本来是信息论中的一个概念,用于表示信息之间的关系, 是两个随机变量统计相关性的测度，使用互信息理论进行特征抽取是基于如下假设:在某个特定类别出现频率高,但在其他类别出现频率比较低的词条与该类的互信息比较大。通常用互信息作为特征词和类别之间的测度，如果特征词属于该类的话，它们的互信息量最大。由于该方法不需要对特征词和类别之间关系的性质作任何假设，因此非常适合于文本分类的特征和类别的配准工作。

特征项和类别的互信息体现了特征项与类别的相关程度, 是一种广泛用于建立词关联统计模型的标准。互信息与期望交叉熵的不同在于没有考虑特征出现的频率, 这样导致互信息评估函数不选择高频的有用词而有可能选择稀有词作为文本的最佳特征。因为对于每一主题来讲，特征t的互信息越大，说明它与该主题的共现概率越大，因此，以互信息作为提取特征的评价时应选互信息最大的若干个特征。互信息计算的时间复杂度类似于信息增益, 互信息的平均值就是信息增益。互信息的不足之处在于得分非常受词条边缘概率的影响。实验数据显示,互信息分类效果最差,其次是文档频率、CC 统计,CHI 统计分类效果最好。

对互信息而言,提高分类精度的方法有:
1) 可以增加特征空间的维数,以提取足够多的特征信息,这样就会带来了时间和空间上的额外开销;
2) 根据互信息函数的定义,认为这些低频词携带着较为强烈的类别信息,从而对它们有不同程度的倚重。

当训练语料库没有达到一定规模的时候,特征空间中必然会存在大量的出现文档频率很低(比如低于3次)的词条,他们较低的文档频率导致了他们必然只属于少数类别. 但是从抽取出来的特征词观察发现,大多数为生僻词,很少一部分确实带有较强的类别信息,多数词携带少量的类别信息,甚至是噪音词.

33.4.5 期望交叉熵(Expected Cross Entropy)

交叉嫡与信息量的定义近似，其公式为:

交叉嫡，也称KL距离。它反映了文本主题类的概率分布和在出现了某特定词汇的条件下文本主题类的概率分布之间的距离，词汇w的交叉嫡越大，对文本主题类分布的影响也越大。它与信息增益唯一的不同之处在于没有考虑单词未发生的情况，只计算出现在文本中的特征项。如果特征项和类别强相关, P (Ci — w)就大,若P(Ci) 又很小的话,则说明该特征对分类的影响大。

交叉熵反映了文本类别的概率分布和在出现了某个特定词的条件下文本类别的概率分布之间的距离, 特征词t 的交叉熵越大, 对文本类别分布的影响也越大。熵的特征选择效果都要优于信息增益。

33.4.6 二次信息熵(QEMI)

将二次熵函数应用于互信息评估方法中，取代互信息中的Shannon熵，就形成了基于二次熵的互信息评估函数。基于二次熵的互信息克服了互信息的随机性，是一个确定的量，因此可以作为信息的整体测度，另外它还比互信息最大化的计算复杂度要小，所以可以比较高效地用在基于分类的特征选取上。

二次熵的概念是在广义信息论中提出的。广义熵:

当，就得到了二次熵定义:

33.4.7 信息增益方法(Information Gain)

信息增益方法是机器学习的常用方法，在过滤问题中用于度量已知一个特征是否出现于某主题相关文本中对于该主题预测有多少信息。通过计算信息增益可以得到那些在正例样本中出现频率高而在反例样本中出现频率低的特征，以及那些在反例样本中出现频率高而在正例样本中出现频率低的特征。信息增益G(w)的训算公式如下:

其中P(w)是词w出现的概率，P(Ci)是取第i个目录时的概率，P(C,—w) 是假定w出现时取第i个目录的概率。

信息增益是一种基于熵的评估方法,涉及较多的数学理论和复杂的熵理论公式,定义为某特征项为整个分类所能提供的信息量,不考虑任何特征的熵与考虑该特征后的熵的差值。他根据训练数据,计算出各个特征项的信息增益,删除信息增益很小的项,其余的按照信息增益从大到小排序。

信息增益是信息论中的一个重要概念, 它表示了某一个特征项的存在与否对类别预测的影响, 定义为考虑某一特征项在文本中出现前后的信息熵之差。某个特征项的信息增益值越大, 贡献越大, 对分类也越重要。信息增益方法的不足之处在于它考虑了特征未发生的情况。特别是在类分布和特征值分布高度不平衡的情况下, 绝大多数类都是负类, 绝大多数特征都不出现。此时的函数值由不出现的特征决定, 因此, 信息增益的效果就会大大降低。信息增益表现出的分类性能偏低。因为信息增益考虑了文本特征未发生的情况，虽然特征不出现的情况肿可能对文本类别具有贡献，但这种贡献往往小于考虑这种情况时对特征分值带来的干扰。

33.4.8 x2统计量方法

x2统计量用于度量特征w和主题类C之间的独立性。而表示除w以外的其他特征，C表示除C以外的其他主题类，那么特征w和主题类C的关系有以下四种

情况: ，用A, B, C, D表示这四种情况的文档频次，总的文档数N=A+B+C+D,扩统计量的计算公式如下:

当特征w和主题类C之间完全独立的时候，x2统计量为0。x2统计量和互信息的差别在于它是归一化的统计量，但是它对低频特征的区分效果也不好。X2 统计得分的计算有二次复杂度, 相似于互信息和信息增益。在X2 统计和互信息之间主要的不同在于X2 是规格化评价, 因而X2 评估分值对在同类中的词是可比的, 但是X2 统计对于低频词来说是不可靠的。

利用x2 统计方法来进行特征抽取是基于如下假设:在指定类别文本中出现频率高的词条与在其他类别文本中出现频率比较高的词条,对判定文档是否属于该类别都是很有帮助的.

采用x2估计特征选择算法的准确率在实验中最高，其分类效果受训练集影响较小，比较稳定。而且在对文教类和政治类存在类别交叉现象的文本进行分类时，采用x2估计的分类系统表现出了优于其它方法的分类性能。X2估计的可靠性较好，便于对程序的控制，无需因训练集的改变而人为的调节特征阀值的大小。

33.4.9　**文本证据权**(The Weight of Evidence forText)

文本证据权衡量类的概率和给定特征时类的条件概率之间的差别。

33.4.10　**优势率**(Odds Ratio)

优势率只适用于二元分类的情况，其特点是只关心文本特征对于目标类的分值。Pos表示目标类，neg表示非目标类。

33.4.11　**遗传算法**(Genetic Algorithm, GA)

文本实际上可以看作是由众多的特征词条构成的多维空间,而特征向量的选择就是多维空间中的寻优过程,因此在文本特征提取研究中可以使用高效寻优算法。遗传算法(Genetic Algorithm, GA)是一种通用型的优化搜索方法,它利用结构化的随机信息交换技术组合群体中各个结构中最好的生存因素,复制出最佳代码串,并使之一代一代地进化,最终获得满意的优化结果。在将文本特征提取问题转化为文本空间的寻优过程中,首先对Web文本空间进行遗传编码,以文本向量构成染色体,通过选择、交叉、变异等遗传操作,不断搜索问题域空间,使其不断得到进化,逐步得到Web文本的最优特征向量。

基于协同演化的遗传算法不是使用固定的环境来评价个体,而是使用其他的个体来评价特定个体。个体优劣的标准不是其生存环境以外的事物,而是由在同一生存竞争环境中的其他个体来决定。协同演化的思想非常适合处理同类文本的特征提取问题。由于同一类别文本相互之间存在一定相关性,因而各自所代表的那组个体在进化过程中存在着同类之间的相互评价和竞争。因此,每个文本的特征向量,即该问题中的个体,在不断的进化过程中,不仅受到其母体(文本)的评价和制约,而且还受到种族中其他同类个体的指导。所以,基于协同演化的遗传算法不仅能反映其母体的特征,还能反映其他同类文本的共性,这样可以有效地解决同一主题众多文本的集体特征向量的提取问题,获得反映整个文本集合某些特征的最佳个体。

33.4.12　**主成分分析法**(Principal Component Analysis，PCA)

它不是通过特征选取的方式降维的，而是通过搜索最能代表原数据的正交向量，创立一个替换的、较小的变量集来组合属性的精华，原数据可以投影到这个较小的集合。PCA由于其处理方

式的不同又分为数据方法和矩阵方法。矩阵方法中，所有的数据通过计算方差一协方差结构在矩阵中表示出来，矩阵的实现目标是确定协方差矩阵的特征向量，它们和原始数据的主要成分相对应。在主成分方法中，由于矩阵方法的复杂度在n很大的情况以二次方增长，因此人们又开发使用了主要使用Hebbian学习规则的PCA神经网络方法。

主成分分析法是特征选取常用的方法之一，它能够揭示更多有关变量重要方向的信息。但它的问题在于矩阵方法中要使用奇异值分解对角化矩阵求解方差一协方差。

33.4.13 模拟退火算法(Simulating Anneal，SA)

特征选取可以看成是一个组合优化问题，因而可以使用解决优化问题的方法来解决特征选取的问题。模拟退火算法(Simulating Anneal，SA)就是其中一种方法。

模拟退火算法是一个很好的解决优化问题的方法，将这个方法运用到特征选取中，理论上能够找到全局最优解，但在初始温度的选取和邻域的选取t要恰当，必须要找到一个比较折中的办法，综合考虑解的性能和算法的速度。

33.4.14 N－Gram算法

它的基本思想是将文本内容按字节流进行大小为N的滑动窗口操作,形成长度为N的字节片段序列。每个字节片段称为gram,对全部gram的出现频度进行统计,并按照事先设定的阈值进行过滤,形成关键gram列表,即为该文本的特征向量空间,每一种gram则为特征向量维度。由于N－Gram算法可以避免汉语分词的障碍,所以在中文文本处理中具有较高的实用性。中文文本处理大多采用双字节进行分解,称之为bi-gram。但是bigram切分方法在处理20%左右的中文多字词时,往往产生语义和语序方面的偏差。而对于专业研究领域,多字词常常是文本的核心特征,处理错误会导致较大的负面影响。基于N－Gram改进的文本特征提取算法[2],在进行bigram切分时,不仅统计gram的出现频度,而且还统计某个gram与其前邻gram的情况,并将其记录在gram关联矩阵中。对于那些连续出现频率大于事先设定阈值的,就将其合并成为多字特征词。这样通过统计与合并双字特征词,自动产生多字特征词,可以较好地弥补N－Gram算法在处理多字词方面的缺陷。

33.4.15 各种方法的综合评价

上述几种评价函数都是试图通过概率找出特征与主题类之间的联系，信息增益的定义过于复杂，因此应用较多的是交叉熵和互信息。其中互信息的效果要好于交叉熵，这是因为互信息是对不同的主题类分别抽取特征词，而交叉熵跟特征在全部主题类内的分布有关，是对全部主题类来抽取特征词。这些方法,在英文特征提取方面都有各自的优势,但用于中文文本,并没有很高的效率。主要有2个方面的原因:

1) 特征提取的计算量太大,特征提取效率太低,而特征提取的效率直接影响到整个文本分类系统的效率。

2) 经过特征提取后生成的特征向量维数太高,而且不能直接计算出特征向量中各个特征词的权重。

目前使用评估函数进行特征选取越来越普遍，特征选取算法通过构造一个评估函数的方法，选取预定数目的最佳特征作为特征子集的结果。在几种评估方法中，每一种方法都有一个选词标准，遵从这个标准，从文本集的所有词汇中选取出有某个限定范围的特征词集。因为评估函数的构造不是特别复杂，适用范围又很广泛，所以越来越多的人们喜欢使用构造评估函数来进行特征的选取。

这些评估函数在Web文本挖掘中被广泛使用,特征选择精度普遍达到70% 80%,但也各自存在缺点和不足。例如,“信息增益”考虑了单词未发生的情况,对判断文本类别贡献不大,而且引入不必要的干扰,特别是在处理类分布和特征值分布高度不平衡的数据时选择精度下降。“期望交叉熵”与“信息增益”的唯一不同就是没有考虑单词未发生的情况,因此不论处理哪种数据集,它的特征选择精度都优于“信息增益”。与“期望交叉熵”相比,“互信息”没有考虑单词发生的频度,这是一个很大的缺点,造成“互信息”评估函数经常倾向于选择稀有单词。“文本证据权”是一种构造比较新颖的评估函数,它衡量一般类的概率和给定特征类的条件概率之间的差别,这样在文本处理中,就不需要计算W的所有可能值,而仅考虑W在文本中出现的情况。“优势率”不像前面所述的其他评估函数将所有类同等对待,它只关心目标类值,所以特别适用于二元分类器,可以尽可能多地识别正类,而不关心识别出负类。

从考虑文本类间相关性的角度,可以把常用的评估函数分为两类,即类间不相关的和类间相关的。“文档频数”(DF)是典型的类间不相关评估函数, DF的排序标准是依据特征词在文档中出现篇数的百分比,或称为篇章覆盖率。这种类型的评估函数,为了提高区分度,要尽量寻找篇章覆盖率较高的特征词,但又要避免选择在各类文本中都多次出现的无意义高频词,因此类间不相关评估函数对停用词表的要求很高。但是,很难建立适用于多个类的停用词表,停用词不能选择太多,也不能选择太少,否则都将会影响特征词的选择。同时,类间不相关评估函数还存在一个明显的缺点,就是对于特征词有交叉的类别或特征相近的类别,选择的特征词会出现很多相似或相同的词条,造成在特定类别间的区分度下降。类间相关的评估函数,例如期望交叉熵、互信息、文本证据权等,综合考虑了词条在已定义的所有类别中的出现情况,可以通过调整特征词的权重,选择出区分度更好的特征,在一定程度上提高了相近类别的区分度。但是，该区分度的提高仅体现在已定义的类别间,而对于尚未定义的域外类别,类间相关评估函数的选择效果也不理想。因此,在评估函数选择问题上,提高对域外类别文本的区分度是十分重要的研究课题。

传统的特征选择方法大多采用以上各评估函数进行特征权重的计算，由于这些评估函数是基于统计学的，其中一个主要缺陷就是需要用一个很庞大的训练集才能获得几乎所有的对分类起关键作用的特征．这需要消耗大量的时间和空间资源，况且，构建这样一个庞大的训练集也是一项十分艰巨的工作。然而，在现实应用中，考虑到工作效率，不会也没有足够的资源去构建一个庞大的训练集，这样的结果就是：被选中的甚至是权重比较高的特征，可能对分类没有什么用处，反而会干涉到正确的分类；而真正有用的特征却因为出现的频率低而获得较低的权重，甚至在降低特征空间维数的时候被删除掉了。

基于评估函数的特征提取方法是建立在特征独立的假设基础上,但在实际中这个假设是很难成立的,因此需要考虑特征相关条件下的文本特征提取方法。使用常规的统计学，用中值和均值来预测效果也不错。

代码33.1中对数据的处理如何代码化和函数结构的模块化值得参考。

Listing 33.1: predict_by_search.py

```
__author__ = "wepon,http://2hwp.com"
__date__ = "2015/08/14"

"""
It score 40.94% on training set when we predict with uid's (forward_median,comment_medain,like_median).
This is good,and we can go further based on this:

        for each uid, we first get its (f_min,f_median,f_max),(c_min,c_median,c_max),(l_min,l_medain,l_max),and then:
                1. fix c_median and l_medain, search a forward value between <f_min,f_max> , which cause a higher score than (f_medain,
                    c_medain,l_medain)
```

```
                    if there exist several result that get the same highest score, we choose the one near f_medain.
                    if not exist any result that get higher score than (f_medain,c_medain,l_medain), than we choose forward = f_medain
                2. search a comment value, by the same method
                3. search a like value, by the same method

"""

import cPickle
import numpy as np
from genUidStat import loadData,genUidStat
from evaluation import precision
from runTime import runTime
from multiprocessing import Pool

def score(uid_data,pred):
        """
        uid_data:
                pd.DataFrame
        pred:
                list , [fp,cp,lp]
        """
        uid_real_pred = uid_data[['forward','comment','like']]
        uid_real_pred['fp'] = pred[0]
        uid_real_pred['cp'] = pred[1]
        uid_real_pred['lp'] = pred[2]
        return precision(uid_real_pred.values)



#search and return the best target value for uid
def search(uid_data,target,args):
        """
        target:
                'forward','comment','like'

        args:
                (f_min,f_median,f_max,c_min,c_median,c_max,l_min,l_medain,l_max)
        """
        args = list(args)
        target_index = ['forward','comment','like'].index(target)
        target_min,target_median,target_max = args[3*target_index:3*target_index+3]
        del args[3*target_index:3*target_index+3]
        pred = (args[1],args[4])

        best_num = [target_median]
        best_pred = list(pred)
        best_pred.insert(target_index,target_median)
        best_score = score(uid_data,best_pred)
        for num in range(target_min,target_max+1):
                this_pred = list(pred)
                this_pred.insert(target_index,num)
                this_score = score(uid_data,this_pred)
                if this_score >= best_score:
                        if this_score > best_score:
                                best_num = [num]
                                best_score = this_score
                        else:
                                best_num.append(num)

        return best_num[np.array([abs(i - target_median) for i in best_num]).argmin()]

##search best target value for all uid,return a dictionary that store {uid:[f,c,l]}
def search_all_uid():
        """
        traindata,testdata = loadData()
        stat_dic = genUidStat()

        #for each uid,search its best fp,cp,lp
        uid_best_pred = {}
        for uid in stat_dic:
                print "search uid: {}".format(uid)
                uid_data = traindata[traindata.uid == uid]
                args = stat_dic[uid][[' forward_min','forward_median','forward_max','comment_min',\
```

```
                                        'comment_median','comment_max','like_min','like_median','like_max']]
                args = tuple([int(i) for i in args])
                fp = search(uid_data,'forward',args)
                cp = search(uid_data,'comment',args)
                lp = search(uid_data,'like',args)
                uid_best_pred[uid] = [fp,cp,lp]
        """
        #multiprocessing version for geting uid_best_pred
        traindata,testdata = loadData()
        stat_dic = genUidStat()
        uid_best_pred = {}
        pool = Pool()
        uids,f,c,l = [],[],[],[]
        for uid in stat_dic:
                print "search_uid:{}".format(uid)
                uid_data = traindata[traindata.uid == uid]
                arguments = stat_dic[uid][['forward_min','forward_median','forward_max','comment_min',\
                                        'comment_median','comment_max','like_min','like_median','like_max']]
                arguments = tuple([int(i) for i in arguments])
                f.append(pool.apply_async(search,args=(uid_data,'forward',arguments)))
                c.append(pool.apply_async(search,args=(uid_data,'comment',arguments)))
                l.append(pool.apply_async(search,args=(uid_data,'like',arguments)))
                uids.append(uid)
        pool.close()
        pool.join()
        f = [i.get() for i in f]
        c = [i.get() for i in c]
        l = [i.get() for i in l]

        for i in range(len(uids)):
                uid_best_pred[uids[i]] = [f[i],c[i],l[i]]

        try:
                cPickle.dump(uid_best_pred,open('uid_best_pred.pkl','w'))
        except Exception:
                pass

        return uid_best_pred


@runTime
def predict_by_search(submission=True):
        traindata,testdata = loadData()
        uid_best_pred = search_all_uid()
        print "search_done,now_predict_on_traindata_and_testdata..."

        #predict traindata with uid's best fp,cp,lp
        forward,comment,like = [],[],[]
        for uid in traindata['uid']:
                if uid_best_pred.has_key(uid):
                        forward.append(int(uid_best_pred[uid][0]))
                        comment.append(int(uid_best_pred[uid][1]))
                        like.append(int(uid_best_pred[uid][2]))
                else:
                        forward.append(0)
                        comment.append(0)
                        like.append(0)

        #score on the traindata
        train_real_pred = traindata[['forward','comment','like']]
        train_real_pred['fp'],train_real_pred['cp'],train_real_pred['lp'] = forward,comment,like
        print "Score_on_the_training_set:{0:.2f}%".format(precision(train_real_pred.values)*100)


        if submission:
                test_pred = testdata[['uid','mid']]
                forward,comment,like = [],[],[]
                for uid in testdata['uid']:
                        if uid_best_pred.has_key(uid):
                                forward.append(int(uid_best_pred[uid][0]))
                                comment.append(int(uid_best_pred[uid][1]))
                                like.append(int(uid_best_pred[uid][2]))
                        else:
                                forward.append(0)
```

```
                                comment.append(0)
                                like.append(0)
            test_pred['fp'], test_pred['cp'], test_pred['lp'] = forward,comment,like

            #generate submission file
            result = []
            filename = "weibo_predict_search.txt"
            for _,row in test_pred.iterrows():
                    result.append("{0}\t{1}\t{2},{3},{4}\n".format(row[0],row[1],row[2],row[3],row[4]))
            f = open(filename,'w')
            f.writelines(result)
            f.close()
            print 'generate_submission_file_"{}"'.format(filename)

if __name__ == "__main__":
            predict_by_search()
```

代码33.2在做特征提取时特别强调了特殊符号@，#，*http*:和“转发”的重要性，单独列出。

Listing 33.2: featurebuild.py

```
# -*- coding: utf-8 -*-
import codecs
import pandas as pd
import numpy as np

# #####calculate user mean
train_data = pd.read_table(r"weibo_train_data.txt", sep='\t', encoding='utf-8', header=None,
                           names=['uid', 'mid', 'time', 'forward_count', 'comment_count', 'like_count', 'content'])
userGroup = train_data.groupby(train_data.uid)
user = userGroup.grouper.result_index.tolist()
forwardMean = userGroup['forward_count'].mean().values.tolist()
commentMean = userGroup['comment_count'].mean().values.tolist()
likeMean = userGroup['like_count'].mean().values.tolist()
train_table = dict(zip(user, zip(forwardMean, commentMean, likeMean)))
totalforwardMean = train_data['forward_count'].mean()
totalcommentMean = train_data['comment_count'].mean()
totallikeMean = train_data['like_count'].mean()
train_table['default'] = (totalforwardMean, totalcommentMean, totallikeMean)

# ####build feature
fr1 = codecs.open('weibo_train_data.txt', mode='r', encoding='utf-8')
fr2 = codecs.open('weibo_predict_data.txt', mode='r', encoding='utf-8')
fw1 = codecs.open('train_feature.txt', mode='w', encoding='utf-8')
fw2 = codecs.open('predict_feature.txt', mode='w', encoding='utf-8')
# ###train feature
try:
    while True:
        text = fr1.readline()
        if not text:
            break
        items = text.split('\t')
        if items[0] == u'\r\n':
            continue
        feature = [None] * 8
        feature[0], feature[1], feature[2] = train_table[items[0]] if items[0] in train_table else train_table['default']
        feature[3] = 1
        feature[4] = 1 if u'@' in items[-1] else 0
        feature[5] = 1 if u'#' in items[-1] else 0
        feature[6] = 1 if u'http:' in items[-1] else 0
        feature[7] = 1 if u'杞骁' in items[-1] else 0 # Chinese word ZhuanFa if in disorder code
        try:
            toWrite = items[0] + '\t' + items[1] + '\t' + items[2] + '\t' + items[3] + '\t' + items[4] + '\t' + items[5] + '\t' + str(feature)
                + '\n'
            fw1.write(toWrite)
        except IndexError:
            continue
    while True:
        text = fr2.readline()
        if not text:
            break
        items = text.split('\t')
        if items[0] == u'\r\n':
```

```
52                continue
53            feature = [None] * 8
54            feature[0], feature[1], feature[2] = train_table[items[0]] if items[0] in train_table else train_table['default']
55            feature[3] = 1
56            feature[4] = 1 if u'@' in items[-1] else 0
57            feature[5] = 1 if u'#' in items[-1] else 0
58            feature[6] = 1 if u'http:' in items[-1] else 0
59            feature[7] = 1 if u'杞骁' in items[-1] else 0
60            try:
61                toWrite = items[0] + '\t' + items[1] + '\t' + items[2] + '\t' + str(feature) + '\n'
62                fw2.write(toWrite)
63            except IndexError:
64                continue
65    finally:
66        if fw1:
67            fw1.close()
68        if fw2:
69            fw2.close()
```

33.5　本章小结

本章首先介绍了。。。

第三十四章　Kaggle Crowdflower Competition

如何在Windows下安装xgboost已经写出：https://www.kaggle.com/c/otto-group-product-classi
forums/t/13043/run-xgboost-from-windows-and-python

34.1　评判函数

两个评判人，A（人类）和B（预测器），使用quadratic weighted kappa来测度两者之间的差异性，比使用简单的相似度函数更具鲁棒性。权重$w_{ij} = \frac{(i-j)^2}{(N-1)^2}$，kappa值计算公式如下：

$$\kappa = 1 - \frac{\sum_{ij} w_{ij} O_{ij}}{\sum_{ij} w_{ij} E_{ij}}. \tag{34.1}$$

其中矩阵O表示两个评测器A，B的打分统计记录，矩阵E表示期望的评测值(expected ratings)。具体定义参考wiki：https://en.wikipedia.org/wiki/Cohen%27s_kappa。

34.2　代码

代码34.1的benchmark script，使用pipeline做了一些预处理，得到一个0.59的分数，离最高分0.71有一段距离，但是很简单，基于简单的TfidfVectorizer()，运行速度快。

Listing 34.1: Kaggle Crowdflower.py

```
1
2  """
3  Beating the Benchmark
4  Search Results Relevance @ Kaggle
5  __author__ : Abhishek
6
7  """
8  import pandas as pd
9  import numpy as np
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.feature_extraction.text import TfidfVectorizer
12 from sklearn.svm import SVC
13 from sklearn.decomposition import TruncatedSVD
14 from sklearn.preprocessing import StandardScaler
15 from sklearn import decomposition, pipeline, metrics, grid_search
16
17 # The following 3 functions have been taken from Ben Hamner's github repository
18 # https://github.com/benhamner/Metrics
19 def confusion_matrix(rater_a, rater_b, min_rating=None, max_rating=None):
20     """
21     Returns the confusion matrix between rater's ratings
22     """
23     assert(len(rater_a) == len(rater_b))
24     if min_rating is None:
25         min_rating = min(rater_a + rater_b)
26     if max_rating is None:
27         max_rating = max(rater_a + rater_b)
28     num_ratings = int(max_rating - min_rating + 1)
29     conf_mat = [[0 for i in range(num_ratings)]
30                 for j in range(num_ratings)]
31     for a, b in zip(rater_a, rater_b):
32         conf_mat[a - min_rating][b - min_rating] += 1
33     return conf_mat
34
```

```
35
36  def histogram(ratings, min_rating=None, max_rating=None):
37      """
38      Returns the counts of each type of rating that a rater made
39      """
40      if min_rating is None:
41          min_rating = min(ratings)
42      if max_rating is None:
43          max_rating = max(ratings)
44      num_ratings = int(max_rating - min_rating + 1)
45      hist_ratings = [0 for x in range(num_ratings)]
46      for r in ratings:
47          hist_ratings[r - min_rating] += 1
48      return hist_ratings
49
50
51  def quadratic_weighted_kappa(y, y_pred):
52      """
53      Calculates the quadratic weighted kappa
54      axquadratic_weighted_kappa calculates the quadratic weighted kappa
55      value, which is a measure of inter-rater agreement between two raters
56      that provide discrete numeric ratings.  Potential values range from -1
57      (representing complete disagreement) to 1 (representing complete
58      agreement).  A kappa value of 0 is expected if all agreement is due to
59      chance.
60      quadratic_weighted_kappa(rater_a, rater_b), where rater_a and rater_b
61      each correspond to a list of integer ratings.  These lists must have the
62      same length.
63      The ratings should be integers, and it is assumed that they contain
64      the complete range of possible ratings.
65      quadratic_weighted_kappa(X, min_rating, max_rating), where min_rating
66      is the minimum possible rating, and max_rating is the maximum possible
67      rating
68      """
69      rater_a = y
70      rater_b = y_pred
71      min_rating=None
72      max_rating=None
73      rater_a = np.array(rater_a, dtype=int)
74      rater_b = np.array(rater_b, dtype=int)
75      assert(len(rater_a) == len(rater_b))
76      if min_rating is None:
77          min_rating = min(min(rater_a), min(rater_b))
78      if max_rating is None:
79          max_rating = max(max(rater_a), max(rater_b))
80      conf_mat = confusion_matrix(rater_a, rater_b,
81                                  min_rating, max_rating)
82      num_ratings = len(conf_mat)
83      num_scored_items = float(len(rater_a))
84
85      hist_rater_a = histogram(rater_a, min_rating, max_rating)
86      hist_rater_b = histogram(rater_b, min_rating, max_rating)
87
88      numerator = 0.0
89      denominator = 0.0
90
91      for i in range(num_ratings):
92          for j in range(num_ratings):
93              expected_count = (hist_rater_a[i] * hist_rater_b[j]
94                                / num_scored_items)
95              d = pow(i - j, 2.0) / pow(num_ratings - 1, 2.0)
96              numerator += d * conf_mat[i][j] / num_scored_items
97              denominator += d * expected_count / num_scored_items
98
99      return (1.0 - numerator / denominator)
100
101
102 if __name__ == '__main__':
103
104     # Load the training file
105     train = pd.read_csv('train.csv')
106     test = pd.read_csv('test.csv')
107
108     # we dont need ID columns
```

```
    idx = test.id.values.astype(int)
    train = train.drop('id', axis=1)
    test = test.drop('id', axis=1)

    # create labels. drop useless columns
    y = train.median_relevance.values
    train = train.drop(['median_relevance', 'relevance_variance'], axis=1)

    # do some lambda magic on text columns
    traindata = list(train.apply(lambda x:'%s %s' % (x['query'],x['product_title']),axis=1))
    testdata = list(test.apply(lambda x:'%s %s' % (x['query'],x['product_title']),axis=1))

    # the infamous tfidf vectorizer (Do you remember this one?)
    tfv = TfidfVectorizer(min_df=3, max_features=None,
            strip_accents='unicode', analyzer='word',token_pattern=r'\w{1,}',
            ngram_range=(1, 5), use_idf=1,smooth_idf=1,sublinear_tf=1,
            stop_words = 'english')

    # Fit TFIDF
    tfv.fit(traindata)
    X = tfv.transform(traindata)
    X_test = tfv.transform(testdata)

    # Initialize SVD
    svd = TruncatedSVD()

    # Initialize the standard scaler
    scl = StandardScaler()

    # We will use SVM here..
    svm_model = SVC()

    # Create the pipeline
    clf = pipeline.Pipeline([('svd', svd), ('scl', scl), ('svm', svm_model)])

    # Create a parameter grid to search for best parameters for everything in the pipeline
    param_grid = {'svd__n_components' : [200, 400],
                  'svm__C': [10, 12]}

    # Kappa Scorer
    kappa_scorer = metrics.make_scorer(quadratic_weighted_kappa, greater_is_better = True)

    # Initialize Grid Search Model
    model = grid_search.GridSearchCV(estimator = clf, param_grid=param_grid, scoring=kappa_scorer,
                                     verbose=10, n_jobs=-1, iid=True, refit=True, cv=2)

    # Fit Grid Search Model
    model.fit(X, y)
    print("Best score: %0.3f" % model.best_score_)
    print("Best parameters set:")
    best_parameters = model.best_estimator_.get_params()
    for param_name in sorted(param_grid.keys()):
        print("\t%s: %r" % (param_name, best_parameters[param_name]))

    # Get best model
    best_model = model.best_estimator_

    # Fit model with best parameters optimized for quadratic_weighted_kappa
    best_model.fit(X,y)
    preds = best_model.predict(X_test)

    # Create your first submission file
    submission = pd.DataFrame({"id": idx, "prediction": preds})
    submission.to_csv("beating_the_benchmark_yet_again.csv", index=False)
```

34.3 本章小结

本章首先介绍了。。。

第三十五章　神兵利器

35.1　大数据分析与机器学习领域Python兵器谱

35.1.1　Python网页爬虫工具集

一个真实的项目，一定是从获取数据开始的。无论文本处理，机器学习和数据挖掘，都需要数据，除了通过一些渠道购买或者下载的专业数据外，常常需要大家自己动手爬数据，这个时候，爬虫就显得格外重要了，幸好，Python提供了一批很不错的网页爬虫工具框架，既能爬取数据，也能获取和清洗数据，我们也就从这里开始了：1. Scrapy Scrapy, a fast high-level screen scraping and web crawling framework for Python.

鼎鼎大名的Scrapy，相信不少同学都有耳闻，课程图谱中的很多课程都是依靠Scrapy抓去的，这方面的介绍文章有很多，推荐大牛pluskid早年的一篇文章：《Scrapy 轻松定制网络爬虫》，历久弥新。官方主页：http://scrapy.org/ Github代码页: https://github.com/scrapy/scrapy

2. Beautiful Soup You didn’t write that awful page. You’re just trying to get some data out of it. Beautiful Soup is here to help. Since 2004, it’s been saving programmers hours or days of work on quick-turnaround screen scraping projects. 读书的时候通过《集体智慧编程》这本书知道Beautiful Soup的，后来也偶尔会用用，非常棒的一套工具。客观的说，Beautifu Soup不完全是一套爬虫工具，需要配合urllib使用，而是一套HTML/XML数据分析，清洗和获取工具。官方主页：http://www.crummy.com/software/BeautifulSoup/ 3. Python-Goose Html Content / Article Extractor, web scrapping lib in Python Goose最早是用Java写得，后来用Scala重写，是一个Scala项目。Python-Goose用Python重写，依赖了Beautiful Soup。前段时间用过，感觉很不错，给定一个文章的URL, 获取文章的标题和内容很方便。Github主页：https://github.com/grangier/python-goose

35.1.2　Python文本处理工具集

从网页上获取文本数据之后，依据任务的不同，就需要进行基本的文本处理了，譬如对于英文来说，需要基本的tokenize，对于中文，则需要常见的中文分词，进一步的话，无论英文中文，还可以词性标注，句法分析，关键词提取，文本分类，情感分析等等。这个方面，特别是面向英文领域，有很多优秀的工具包，我们一一道来。1. NLTK －Natural Language Toolkit NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and an active discussion forum. 搞自然语言处理的同学应该没有人不知道NLTK吧，这里也就不多说了。不过推荐两本书籍给刚刚接触NLTK或者需要详细了解NLTK的同学: 一个是官方的《Natural Language Processing with Python》，以介绍NLTK里的功能用法为主，同时附带一些Python知识，同时国内陈涛同学友情翻译了一个中文版，这里可以看到: 推荐《用Python进行自然语言处理》中文翻译-NLTK配套书；另外一本是《Python Text Processing with NLTK 2.0 Cookbook》，这本书要深入一些，会涉及到NLTK的代码结构，同时会介

绍如何定制自己的语料和模型等，相当不错。官方主页：http://www.nltk.org/ Github代码页：https://github.com/nltk/nltk 2. Pattern Pattern is a web mining module for the Python programming language. It has tools for data mining (Google, Twitter and Wikipedia API, a web crawler, a HTML DOM parser), natural language processing (part-of-speech taggers, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, clustering, SVM), network analysis and canvas visualization. Pattern由比利时安特卫普大学CLiPS实验室出品，客观的说，Pattern不仅仅是一套文本处理工具，它更是一套web数据挖掘工具，囊括了数据抓取模块（包括Google, Twitter, 维基百科的API，以及爬虫和HTML分析器），文本处理模块（词性标注，情感分析等），机器学习模块(VSM, 聚类，SVM）以及可视化模块等，可以说，Pattern的这一整套逻辑也是这篇文章的组织逻辑，不过这里我们暂且把Pattern放到文本处理部分。我个人主要使用的是它的英文处理模块Pattern.en, 有很多很不错的文本处理功能，包括基础的tokenize, 词性标注，句子切分，语法检查，拼写纠错，情感分析，句法分析等，相当不错。官方主页：http://www.clips.ua.ac.be/pattern 3. TextBlob: Simplified Text Processing TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. TextBlob是一个很有意思的Python文本处理工具包，它其实是基于上面两个Python工具包NLKT和Pattern做了封装（TextBlob stands on the giant shoulders of NLTK and pattern, and plays nicely with both），同时提供了很多文本处理功能的接口，包括词性标注，名词短语提取，情感分析，文本分类，拼写检查等，甚至包括翻译和语言检测，不过这个是基于Google的API的，有调用次数限制。TextBlob相对比较年轻，有兴趣的同学可以关注。官方主页：http://textblob.readthedocs.org/en/dev/ Github代码页：https://github.com/sloria/textblob 4. MBSP for Python MBSP is a text analysis system based on the TiMBL and MBT memory based learning applications developed at CLiPS and ILK. It provides tools for Tokenization and Sentence Splitting, Part of Speech Tagging, Chunking, Lemmatization, Relation Finding and Prepositional Phrase Attachment. MBSP与Pattern同源，同出自比利时安特卫普大学CLiPS实验室，提供了Word Tokenization, 句子切分，词性标注，Chunking, Lemmatization，句法分析等基本的文本处理功能，感兴趣的同学可以关注。官方主页：http://www.clips.ua.ac.be/pages/MBSP 5. Gensim: Topic modeling for humans Gensim是一个相当专业的主题模型Python工具包，无论是代码还是文档，我们曾经用《如何计算两个文档的相似度》介绍过Gensim的安装和使用过程，这里就不多说了。官方主页：http://radimrehurek.com/gensim/index.html github代码页：https://github.com/piskvorky/gensim 6. langid.py: Stand-alone language identification system 语言检测是一个很有意思的话题，不过相对比较成熟，这方面的解决方案很多，也有很多不错的开源工具包，不过对于Python来说，我使用过langid这个工具包，也非常愿意推荐它。langid目前支持97种语言的检测，提供了很多易用的功能，包括可以启动一个建议的server，通过json调用其API，可定制训练自己的语言检测模型等，可以说是“麻雀虽小，五脏俱全”。Github主页：https://github.com/saffsd/langid.py 7. Jieba: 结巴中文分词“结巴”中文分词：做最好的Python中文分词组件“Jieba”(Chinese for “to stutter”) Chinese text segmentation: built to be the best Python Chinese word segmentation module. 好了，终于可以说一个国内的Python文本处理工具包了：结巴分词，其功能包括支持三种分词模式（精确模式、全模式、搜索引擎模式），支持繁体分词，支持自定义词典等，是目前一个非常不错的Python中文分词解决方

案。Github主页：https://github.com/fxsjy/jieba 8. xTAS xtas, the eXtensible Text Analysis Suite, a distributed text analysis package based on Celery and Elasticsearch. 感谢微博朋友@大山坡的春提供的线索：我们组同事之前发布了xTAS，也是基于python的text mining工具包，欢迎使用，链接：http://t.cn/RPbEZOW。看起来很不错的样子，回头试用一下。Github代码页：https://github.com/NLeSC/xtas

35.1.3　Python科学计算工具包

说起科学计算，大家首先想起的是Matlab，集数值计算，可视化工具及交互于一身，不过可惜是一个商业产品。开源方面除了GNU Octave在尝试做一个类似Matlab的工具包外，Python的这几个工具包集合到一起也可以替代Matlab的相应功能：NumPy+SciPy+Matplotlib+iPython。同时，这几个工具包，特别是NumPy和SciPy，也是很多Python文本处理& 机器学习& 数据挖掘工具包的基础，非常重要。最后再推荐一个系列《用Python做科学计算》，将会涉及到NumPy, SciPy, Matplotlib，可以做参考。1. NumPy NumPy is the fundamental package for scientific computing with Python. It contains among other things: 1）a powerful N-dimensional array object 2）sophisticated (broadcasting) functions 3）tools for integrating C/C++ and Fortran code 4）useful linear algebra, Fourier transform, and random number capabilities Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. NumPy几乎是一个无法回避的科学计算工具包，最常用的也许是它的N维数组对象，其他还包括一些成熟的函数库，用于整合C/C++和Fortran代码的工具包，线性代数、傅里叶变换和随机数生成函数等。NumPy提供了两种基本的对象：ndarray（N-dimensional array object）和ufunc（universal function object）。ndarray是存储单一数据类型的多维数组，而ufunc则是能够对数组进行处理的函数。官方主页：http://www.numpy.org/ 2. SciPy：Scientific Computing Tools for Python SciPy refers to several related but distinct entities: 1）The SciPy Stack, a collection of open source software for scientific computing in Python, and particularly a specified set of core packages. 2）The community of people who use and develop this stack. 3）Several conferences dedicated to scientific computing in Python – SciPy, EuroSciPy and SciPy.in. 4）The SciPy library, one component of the SciPy stack, providing many numerical routines. “SciPy是一个开源的Python算法库和数学工具包，SciPy包含的模块有最优化、线性代数、积分、插值、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解和其他科学与工程中常用的计算。其功能与软件MATLAB、Scilab和GNU Octave类似。Numpy和Scipy常常结合着使用，Python大多数机器学习库都依赖于这两个模块。”—-引用自“Python机器学习库”官方主页：http://www.scipy.org/ 3. Matplotlib matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell (ala MATLAB?* or Mathematica??), web application servers, and six graphical user interface toolkits. matplotlib 是python最著名的绘图库，它提供了一整套和matlab相似的命令API，十分适合交互式地进行制图。而且也可以方便地将它作为绘图控件，嵌入GUI应用程序中。Matplotlib可以配合ipython shell使用，提供不亚于Matlab的绘图体验，总之用过了都说好。官方主页：http://matplotlib.org/ 4. iPython IPython provides a rich architecture for interactive computing with: 1）Powerful interactive shells (terminal and Qt-based). 2）A browser-based notebook with support for code, text, mathematical expressions, inline plots

and other rich media. 3）Support for interactive data visualization and use of GUI toolkits. 4）Flexible, embeddable interpreters to load into your own projects. 5）Easy to use, high performance tools for parallel computing. “iPython 是一个Python 的交互式Shell，比默认的Python Shell 好用得多，功能也更强大。她支持语法高亮、自动完成、代码调试、对象自省，支持Bash Shell 命令，内置了许多很有用的功能和函式等，非常容易使用。”启动iPython的时候用这个命令“ipython -pylab”，默认开启了matploblib的绘图交互，用起来很方便。官方主页：http://ipython.org/

35.1.4 Python 机器学习& 数据挖掘工具包

机器学习和数据挖掘这两个概念不太好区分，这里就放到一起了。这方面的开源Python工具包有很多，这里先从熟悉的讲起，再补充其他来源的资料，也欢迎大家补充。1. scikit-learn: Machine Learning in Python scikit-learn (formerly scikits.learn) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. 首先推荐大名鼎鼎的scikit-learn，scikit-learn是一个基于NumPy, SciPy, Matplotlib的开源机器学习工具包，主要涵盖分类，回归和聚类算法，例如SVM，逻辑回归，朴素贝叶斯，随机森林，k-means等算法，代码和文档都非常不错，在许多Python项目中都有应用。例如在我们熟悉的NLTK中，分类器方面就有专门针对scikit-learn的接口，可以调用scikit-learn的分类算法以及训练数据来训练分类器模型。这里推荐一个视频，也是我早期遇到scikit-learn的时候推荐过的：推荐一个Python机器学习工具包Scikit-learn以及相关视频 - Tutorial: scikit-learn - Machine Learning in Python 官方主页：http://scikit-learn.org/ 2. Pandas: Python Data Analysis Library Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. 第一次接触Pandas是由于Udacity上的一门数据分析课程“Introduction to Data Science”的Project需要用Pandas库，所以学习了一下Pandas。Pandas也是基于NumPy和Matplotlib开发的，主要用于数据分析和数据可视化，它的数据结构DataFrame和R语言里的data.frame很像，特别是对于时间序列数据有自己的一套分析机制，非常不错。这里推荐一本书《Python for Data Analysis》，作者是Pandas的主力开发，依次介绍了iPython, NumPy, Pandas里的相关功能，数据可视化，数据清洗和加工，时间数据处理等，案例包括金融股票数据挖掘等，相当不错。官方主页：http://pandas.pydata.org/ ========== 分割线，以上工具包基本上都是作者自己用过的，以下来源于其他同学的线索，特别是《Python机器学习库》，《23个python的机器学习包》，做了一点增删修改，欢迎大家补充==========
3. mlpy mlpy is a Python module for Machine Learning built on top of NumPy/SciPy and the GNU Scientific Libraries. mlpy provides a wide range of state-of-the-art machine learning methods for supervised and unsupervised problems and it is aimed at finding a reasonable compromise among modularity, maintainability, reproducibility, usability and efficiency. mlpy is multiplatform, it works with Python 2 and 3 and it is Open Source, distributed under the GNU General Public License version 3. 官方主页：http://mlpy.sourceforge.net/ 4. MDP：The Modular toolkit for Data Processing Modular toolkit for Data Processing (MDP) is a Python data processing framework. From the user’s perspective, MDP is a collection of supervised and unsupervised learning algorithms and

other data processing units that can be combined into data processing sequences and more complex feed-forward network architectures. From the scientific developer’s perspective, MDP is a modular framework, which can easily be expanded. The implementation of new algorithms is easy and intuitive. The new implemented units are then automatically integrated with the rest of the library. The base of available algorithms is steadily increasing and includes signal processing methods (Principal Component Analysis, Independent Component Analysis, Slow Feature Analysis), manifold learning methods ([Hessian] Locally Linear Embedding), several classifiers, probabilistic methods (Factor Analysis, RBM), data pre-processing methods, and many others. “MDP用于数据处理的模块化工具包，一个Python数据处理框架。从用户的观点，MDP是能够被整合到数据处理序列和更复杂的前馈网络结构的一批监督学习和非监督学习算法和其他数据处理单元。计算依照速度和内存需求而高效的执行。从科学开发者的观点，MDP是一个模块框架，它能够被容易地扩展。新算法的实现是容易且直观的。新实现的单元然后被自动地与程序库的其余部件进行整合。MDP在神经科学的理论研究背景下被编写，但是它已经被设计为在使用可训练数据处理算法的任何情况中都是有用的。其站在用户一边的简单性，各种不同的随时可用的算法，及应用单元的可重用性，使得它也是一个有用的教学工具。”官方主页：http://mdp-toolkit.sourceforge.net/ 5. PyBrain PyBrain is a modular Machine Learning Library for Python. Its goal is to offer flexible, easy-to-use yet still powerful algorithms for Machine Learning Tasks and a variety of predefined environments to test and compare your algorithms. PyBrain is short for Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library. In fact, we came up with the name first and later reverse-engineered this quite descriptive “Backronym”. “PyBrain(Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network)是Python的一个机器学习模块，它的目标是为机器学习任务提供灵活、易应、强大的机器学习算法。（这名字很霸气）PyBrain正如其名，包括神经网络、强化学习(及二者结合)、无监督学习、进化算法。因为目前的许多问题需要处理连续态和行为空间，必须使用函数逼近(如神经网络)以应对高维数据。PyBrain以神经网络为核心，所有的训练方法都以神经网络为一个实例。”官方主页：http://www.pybrain.org/ 6. PyML – machine learning in Python PyML is an interactive object oriented framework for machine learning written in Python. PyML focuses on SVMs and other kernel methods. It is supported on Linux and Mac OS X. “PyML是一个Python机器学习工具包，为各分类和回归方法提供灵活的架构。它主要提供特征选择、模型选择、组合分类器、分类评估等功能。”项目主页：http://pyml.sourceforge.net/ 7. Milk：Machine learning toolkit in Python. Its focus is on supervised classification with several classifiers available: SVMs (based on libsvm), k-NN, random forests, decision trees. It also performs feature selection. These classifiers can be combined in many ways to form different classification systems. “Milk是Python的一个机器学习工具箱，其重点是提供监督分类法与几种有效的分类分析：SVMs(基于libsvm)，K-NN，随机森林经济和决策树。它还可以进行特征选择。这些分类可以在许多方面相结合，形成不同的分类系统。对于无监督学习，它提供K-means和affinity propagation聚类算法。”官方主页：http://luispedro.org/software/milk http://luispedro.org/software/milk 8. PyMVPA: MultiVariate Pattern Analysis (MVPA) in Python PyMVPA is a Python package intended to ease statistical learning analyses of large datasets. It offers an extensible framework with a high-level interface to a broad range of algorithms for classification, regression, feature selection, data import and export. It is designed to integrate well with related software packages,

such as scikit-learn, and MDP. While it is not limited to the neuroimaging domain, it is eminently suited for such datasets. PyMVPA is free software and requires nothing but free-software to run. “PyMVPA(Multivariate Pattern Analysis in Python)是为大数据集提供统计学习分析的Python工具包，它提供了一个灵活可扩展的框架。它提供的功能有分类、回归、特征选择、数据导入导出、可视化等”官方主页：http://www.pymvpa.org/ 9. Pyrallel – Parallel Data Analytics in Python Experimental project to investigate distributed computation patterns for machine learning and other semi-interactive data analytics tasks. “Pyrallel(Parallel Data Analytics in Python)基于分布式计算模式的机器学习和半交互式的试验项目，可在小型集群上运行”Github代码页：http://github.com/pydata/pyrallel 10. Monte – gradient based learning in Python Monte (python) is a Python framework for building gradient based learning machines, like neural networks, conditional random fields, logistic regression, etc. Monte contains modules (that hold parameters, a cost-function and a gradient-function) and trainers (that can adapt a module’s parameters by minimizing its cost-function on training data). Modules are usually composed of other modules, which can in turn contain other modules, etc. Gradients of decomposable systems like these can be computed with back-propagation. “Monte (machine learning in pure Python)是一个纯Python机器学习库。它可以迅速构建神经网络、条件随机场、逻辑回归等模型，使用inline-C优化，极易使用和扩展。”官方主页：http://montepython.sourceforge.net 11. Theano Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features: 1）tight integration with NumPy – Use numpy.ndarray in Theano-compiled functions. 2）transparent use of a GPU – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only) 3）efficient symbolic differentiation – Theano does your derivatives for function with one or many inputs. 4）speed and stability optimizations – Get the right answer for log(1+x) even when x is really tiny. 5）dynamic C code generation – Evaluate expressions faster. 6）extensive unit-testing and self-verification – Detect and diagnose many types of mistake. Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (IFT6266 at the University of Montreal). “Theano 是一个Python 库，用来定义、优化和模拟数学表达式计算，用于高效的解决多维数组的计算问题。Theano的特点：紧密集成Numpy；高效的数据密集型GPU计算；高效的符号微分运算；高速和稳定的优化；动态生成c代码；广泛的单元测试和自我验证。自2007年以来，Theano已被广泛应用于科学运算。theano使得构建深度学习模型更加容易，可以快速实现多种模型。PS：Theano，一位希腊美女，Croton最有权势的Milo的女儿，后来成为了毕达哥拉斯的老婆。”12. Pylearn2 Pylearn2 is a machine learning library. Most of its functionality is built on top of Theano. This means you can write Pylearn2 plugins (new models, algorithms, etc) using mathematical expressions, and theano will optimize and stabilize those expressions for you, and compile them to a backend of your choice (CPU or GPU). “Pylearn2建立在theano上，部分依赖scikit-learn上，目前Pylearn2正处于开发中，将可以处理向量、图像、视频等数据，提供MLP、RBM、SDA等深度学习模型。”官方主页：http://deeplearning.net/software/pylearn2/ 其他的，欢迎大家补充，这里也会持续更新这篇文章。

35.2　代码

35.3　本章小结

本章首先介绍了。。。

第三十六章　问题集

36.1　第一部分标题还未想好

1. 如何理解泛化误差=偏差+方差+噪声?

   首先这只在均方误差的回归任务中能推倒得到，但可以一定程度上说明问题。偏差度量了学习算法的期望预测与真实结果的偏离程度，刻画了学习算法本身的拟合能力；方差度量了同样大小的训练集的变动所导致的学习性能的变化，刻画了数据扰动所造成的影响；噪声则表达了在当前任务上学习算法所能达到的期望泛化误差的下界，刻画了学习问题本身的难度。在训练不足时，学习器的拟合能力不够强，训练数据的扰动不足以使学习器产生显著的变化，此时偏差主导了泛化错误率；当训练程度增加，学习器的拟合能力逐渐增强，训练数据发生的扰动渐渐能被学习器学到，方差逐渐主导泛化错误率；训练充足后，若训练数据自身的、非全局的特性被学到了，则发生过拟合。

2. 算法参数（超参数）和模型参数的区别?

   算法参数，通常在10个以内，由人工设定的多个参数候选值来产生模型。模型参数数目很多，例如“深度学习”通过学习来产生多个候选模型（例如神经网络在不同轮数停止训练）。

3. 测试集（validation set）和验证集（test set）的区别?

   验证集用于对模型进行选择和调优，而测试集用于对最终的模型泛化能力进行测试（只用一次，不对模型训练进行指导，否则污染数据源）。

4. 学习算法和模型参数已经在训练集下确定后，是否需要把验证集和训练集合起来?

   Yes，最后要用所有的数据训练模型，最终呈现给用户。

5. 查准率（precision，准确率）、查全率（recall，召回率）和F_1 score的定义?

   precision: 搜索到的结果中有多少是用户感兴趣的内容，$P = \dfrac{TP}{TP+FP}$

   recall: 用户感兴趣的内容有多少体现在搜索结果中，$R = \dfrac{TP}{TP+FN}$

   $F_1 = \dfrac{P+R}{2PR}$，$\dfrac{1}{F_\beta} = \dfrac{1}{1+\beta^2}(\dfrac{1}{P} + \dfrac{\beta^2}{R})$，其中$\beta > 0$度量了recall对precision的相对重要性，大于1则recall有更大影响，反之亦然。调和平均值更重视较小值。更多定义请参见micro-F1和macro-F1。

6. ROC（Receiver Operating Curve）曲线的特性?

   对角线上的点表示预测正反例各50%，相当于随机猜测。而离左上角（0,1）越近的点，模型性能越好。若靠近右下角很近，也可预测与之相反的结果，得到一个正确率不错的预测模型。

7. 为什么使用T检验和F检验?

为了确定从样本(sample)统计结果推论至总体时所犯错的概率，我们会利用统计学家所开发的一些统计方法，进行统计检定。若不是巧合，则能够拒绝虚无假设null hypothesis。t检验过程，是对两样本均数(mean)差别的显著性进行检验。惟t检验须知道两个总体的方差(Variances)是否相等；t检验值的计算会因方差是否相等而有所不同。就是因为要评估两个总体的方差(Variances)是否相等，要做Levene's Test for Equality of Variances，要检验方差，故所以就有F值。t检验适用于两个变量均数间的差异检验，多于两个变量间的均数比较要用方差分析。

8. 统计学意义（P值）?

   结果的统计学意义是结果真实程度（能够代表总体）的一种估计方法。专业上，P值为结果可信程度的一个递减指标，P值越大，我们越不能认为样本中变量的关联是总体中各变量关联的可靠指标。P值是将观察结果认为有效即具有总体代表性的犯错概率。如P=0.05提示样本中变量关联有5%的可能是由于偶然性造成的。即假设总体中任意变量间均无关联，我们重复类似实验，会发现约20个实验中有一个实验，我们所研究的变量关联将等于或强于我们的实验结果。（这并不是说如果变量间存在关联，我们可得到5%或95%次数的相同结果，当总体中的变量存在关联，重复研究和发现关联的可能性与设计的统计学效力有关。）在许多研究领域，0.05的P值通常被认为是可接受错误的边界水平。

9. 卡方检验的结果，值是越大越好，还是越小越好?

   与其它检验一样，所计算出的统计量越大，在分布中越接近分布的尾端，所对应的概率值越小。如果试验设计合理、数据正确，显著或不显著都是客观反映。没有什么好与不好。

10. 回归分析和相关分析的联系和区别?

    回归分析(Regression)：Dependant variable is defined and can be forecasted by independent variable.相关分析(Correlation)：The relationship btw two variables. – A dose not define or determine B. 回归更有用自变量解释因变量的意思，有一点点因果关系在里面，并且可以是线性或者非线形关系；相关更倾向于解释两两之间的关系，但是一般都是指线形关系，特别是相关指数，有时候图像显示特别强二次方图像，但是相关指数仍然会很低，而这仅仅是因为两者间不是线形关系，并不意味着两者之间没有关系，因此在做相关指数的时候要特别注意怎么解释数值，特别建议做出图像观察先。任何事物的存在都不是孤立的，而是相互联系、相互制约的。身高与体重、体温与脉搏、年龄与血压等都存在一定的联系。说明客观事物相互间关系的密切程度并用适当的统计指标表示出来，这个过程就是相关分析。

36.2 本章小结

本章首先介绍了。。。

第三十七章　总结与展望

37.1　本文工作总结

未完待续

37.2　工作展望

未完待续

简　　历

基本情况

马宏文，男，博士研究生。

教育状况

2008 年 9 月至 2012 年 6 月，““获得工学学士学位。

2012 年 9 月至 2017 年 6 月，中国科学院 自动化研究所，直博生，专业：控制理论与控制工程。

攻读博士学位期间参与项目

1. 国家高技术研究发展计划(863 计划) 重点项目 ““““
2. 国家重点基础研究发展规划项目(973 计划) ““
3. 国家自然科学基金项目““

研究兴趣

分布式控制算法，神经网络，事件驱动。

联系方式

通讯地址：北京市海淀区中关村东路95号中国科学院自动化研究所智能化大厦10层

邮编：100190

E-mail: mahongwen2012@ia.ac.cn