机器学习作业 1-PUBG 排名预测

516021910700 郭远帆

516021910439 罗乙然

516021910561 唐荣俊

目录

任务介绍

任务说明

本次任务来源于 Kaggle 数据科学竞赛网站提供的比赛项目:**PUBG Finish Placement Prediction （绝地求生:大逃杀玩家排名预测）**

比赛说明网站: https://www.kaggle.com/c/pubg-finish-placement-prediction 网站提供了该任务的详细说明，输出要求以及数据（包括训练集和测试集）

这是一个非常有趣的项目，竞赛网站与 PUBG 制作商蓝洞公司合作，获取了超过 65，000 场游戏的玩家数据，要求竞赛参加者根据这些数据预测玩家在比赛中的排名。

数据集的使用

本项目原数据集相当庞大，训练集与测试集数据量一共有超过 650 万条数据，每一条数据拥有 25 个特征。在实际使用中,对于某些模型(如随机森林，决策树等)我们需要对数据进行采样来测试模型的性能，以缩短训练时间，从而在短时间内了解多个模型的特点以及调参方法。

由于数据集庞大，提交作业时不一并提交，可于百度网盘上下载

任务目标与规划

我们小组以此项目作为机器学习第一次作业，该问题属于回归问题。而由于该项目本身是数据科学竞赛项目，其中必然存在数据分析与特征提取的过程（但并非主要的），我们将任务划分为几个阶段:

第一阶段: 数据缺失处理，进行简单的数据可视化分析

第二阶段: 参考竞赛网站其它竞赛参加者提供的特征工程方案，提取简单特征

第三阶段: 构建不同的机器学习模型训练，并且进行模型的评估

其中第三为主要的工作。

通过本次任务，我们将学习到:

1. 利用 pandas 对数据集进行预处理
2. 利用 seaborn 和 matplotlib 对数据集进行简单分析
3. 如何调用 sklearn 中的机器学习模型，并理解其实现原理
4. 自己实现简单的模型并与 sklearn 库进行比较

小组分工

郭远帆:

完成基本特征工程以及针对线性模型优化的特征工程

实现梯度下降法线性回归模型
利用 sklearn 的线性回归模型进行多次实验并进行比较分析

唐荣俊:
完成对决策树模型的特征工程
利用 sklearn 的决策树模型进行多次实验和调参，并进行比较分析

罗乙然:
完成对随机森林模型的特征工程
利用 sklearn 的随机森林模型进行多次实验和调参，并进行比较分析

实现方案

编程环境说明

编程语言: Python3.6.4
使用的工具包:
Numpy: 线性代数库，支持强大的矩阵运算
pandas: 数据处理库
Matplotlib: Matlab 风格的画图工具
Seaborn: 统计画图工具
Sklearn: 最大的机器学习库，包含了一些常用的机器学习模型。
xgboost: 梯度提升库（集成学习方法）

基本特征工程（线性模型）

线性回归模型以模型简单、速度快而在工业界广泛应用。但是由于模型过于简单，最小二乘法的线性简单模型需要更多的特征提取工作，否则非常容易欠拟合而得到不满意的结果。在对训练集进行数据预处理以及相关性分析后，我们得到下图:

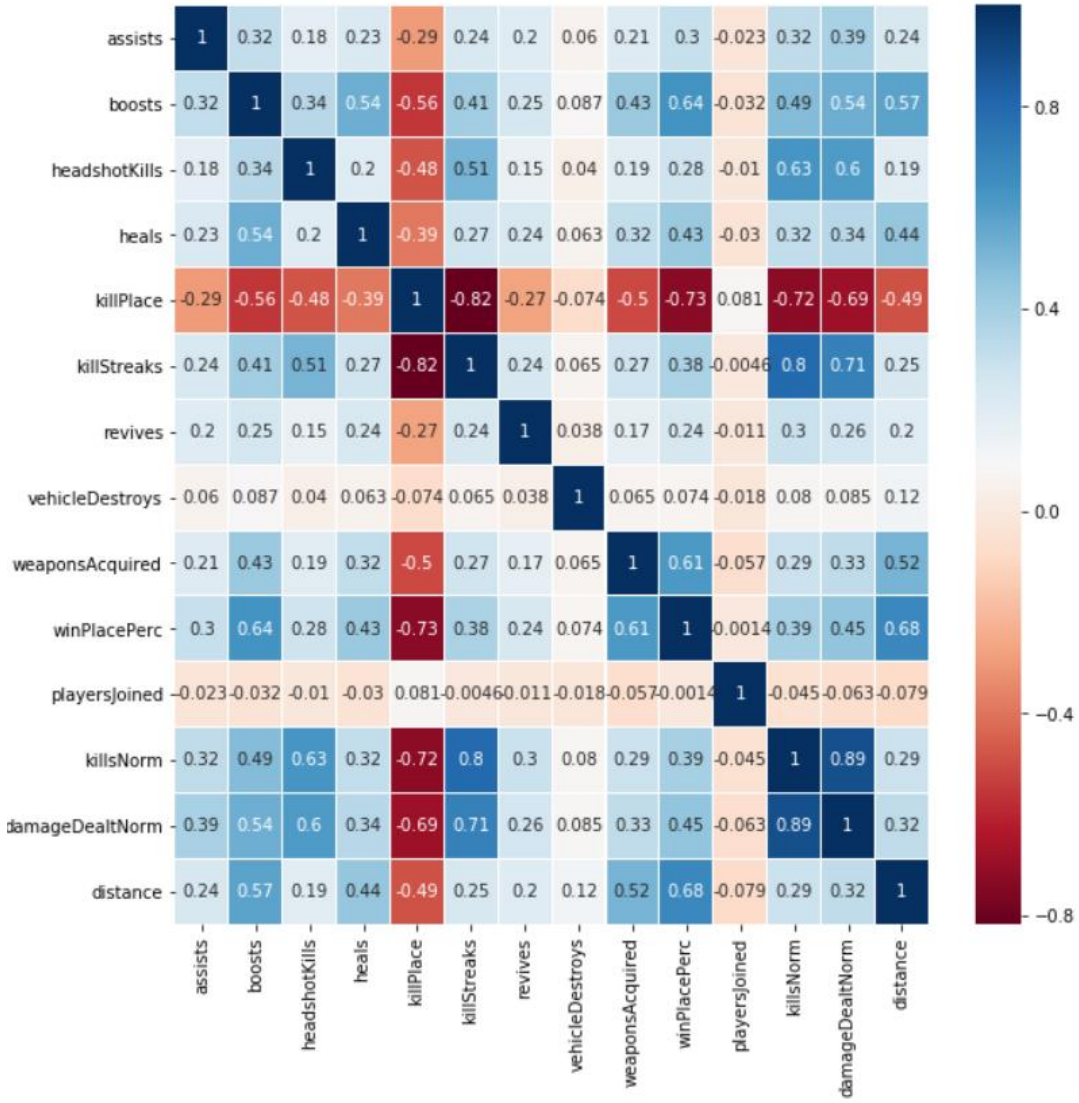


Figure1-特征相关性图

其中 winPlacePerc 为目标值(y)，从图中可以看到经过预处理后的特征的相关性，这些特征即线性模型中最终保留的特征。

模型评估方法

根据竞赛要求，该任务最终的模型评估方法为平均绝对值误差(mean absolute error)，根据模型不同还有不同的评价方式，详见实验结果中的相关描述。

实验结果

线性模型

模型说明

sklearn 库提供了多种线性模型，在本次实验中使用到其中三种: LinearRegression, Lasso, Ridge。

LinearRegression 的原理十分简单，设 **X** 为输入数据集，**y** 为预测结果，**w** 为各特征权重，其优化目标为:

$$\min_{w}\left|\left|\boldsymbol{Xw}-\boldsymbol{y}\right|\right|_2^2$$

在 sklearn 库中，其训练复杂度为$O(np^2)$，其中 n 为样本数量，p 为特征量。根据[1]中对 sklearn 的 LinearRegression()源码解读，解线性回归方程参数时，首先判断训练集 **X** 是不是稀疏矩阵，如是，就用 Golub & Kahan 双对角线化过程方法来求解；否则就调用 C 库 LAPACK 中的用基于分治法的奇异值分解来求解。

sklearn 中的 Lasso 和 Ridge 是引入了 L1 和 L2 正则化项的线性回归模型，即优化目标分别为:

$$\text{Lasso: }\min_{w}\frac{1}{2n}\left|\left|\boldsymbol{Xw}-\boldsymbol{y}\right|\right|_2^2+\alpha\left|\left|\boldsymbol{w}\right|\right|_1$$

$$\textit{Ridge: }\min_{w}\left|\left|\boldsymbol{XW}-\boldsymbol{y}\right|\right|_2^2+\alpha\left|\left|\boldsymbol{w}\right|\right|_2^2$$

和 LinearRegression 一样，在 sklearn 中采用双对角线化 或者 奇异值分解来进行优化，其计算复杂度也为$O(np^2)$

由于 sklearn 中的线性模型没有使用梯度下降法实现，为了实践课内学习到的理论，自己实现梯度下降法(GDLinear.py)并在下一部分与 sklearn 实现的模型进行比较。

实验结果

在完成一轮特征工程后，数据集中留下来的特征拥有较好的独立性，并且对于简单模型来说降低了数据的复杂度，因此也降低了拟合的难度。我们采用不同的线性模型进行拟合并观察实验结果，并尝试着对实验结果进行解释。

调用 sklearn 中的几个线性模型进行训练以及预测（采用全局数据）并对结果进行比较:

实验结果:

模型:	数据集	测试集比例	M（平均绝对误差）
LinearRegression	train.csv(49M)	0.25	0.095
Lasso	train.csv(49M)	0.25	0.095
Ridge	Train.csv(49M)	0.25	0.095

从表中可得，线性回归模型平均将引入 9.5%的误差。正则化项对模型的影响不大

鉴于该模型的低成本性，9.5%误差的结果可以接受。

接下来考虑自己实现的梯度下降法的线性回归模型（未引入正则化）：

学习速率(α)	数据集	测试集比例	最大迭代次数	M(平均绝对误差)
0.000001	Train.csv.sample(1000)	0.25	5000	0.0835
0.00001	Train.csv.sample(1000)	0.25	5000	0.1209
0.001	Train.csv.sample(1000)	0.25	5000	NaN

自己实现的梯度下降法线性模型还需要非常大的改进，目前的梯度下降法只能在小型数据集上进行测试，无法像 sklearn 中实现的线性回归模型一样高效快速。而且，即使对数据集进行了归一化处理，学习速率的调整依然非常困难，很容易出现无法收敛的情况。

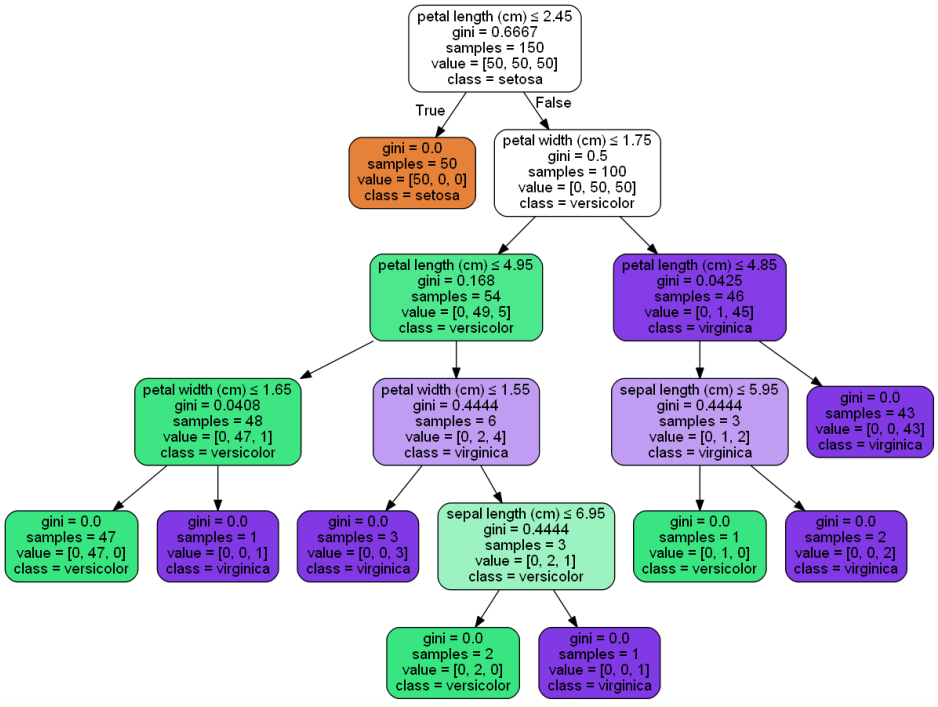
从与 sklearn 中的线性模型对比，可以得出:

1. 理论计算和实际中数值解的求取还是有差距的，sklearn 中的方法十分具有实际意义和工程性,其基于数值解的优化方式值得学习
2. 梯度下降法虽然实现简单，但拥有许多缺点，例如对学习速率超参数的强烈依赖。

决策树模型

决策树定义

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。机器学习中，决策树是一个预测模型，代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。下图为一个决策树模型的图示。



模型使用及结果

决策树模型在机器学习中既可以用作分类算法，也可以用作回归算法。这里我们采用它的回归形式来对结果做出预测，并使用 scikit-learn 中的 DecisionTreeRegressor 库进行回归预测。该库使用的是 CART 算法，是最优的决策树算法。

其主要优点有：

1）生成的决策树简单直观。

2）基本不需要预处理，不需要提前归一化，处理缺失值。

3）使用决策树预测的代价是 $O(log2m)$O(log2m)。 m 为样本数。

4）既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。

5）可以处理多维度输出的分类问题。

6）相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以得到很好的解释

7）可以交叉验证的剪枝来选择模型，从而提高泛化能力。

8）对于异常点的容错能力好，健壮性高。

我们再看看决策树算法的缺点:

1）决策树算法非常容易过拟合，导致泛化能力不强。可以通过设置节点最少样本数量和限制决策树深度来改进。

2）决策树会因为样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习之类的方法解决。

3）寻找最优的决策树是一个 NP 难的问题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习之类的方法来改善。

4）有些比较复杂的关系，决策树很难学习，比如异或。这个就没有办法了，一般这种关系可以换神经网络分类方法来解决。

5）如果某些特征的样本比例过大，生成决策树容易偏向于这些特征。这个可以通过调节样本权重来改善。

通过查阅资料，我们发现决策树回归模型有很多个参数需要调节。一些参数，如 criterion、splitter 等，采用默认的参数即可。而一些参数则需要根据我们数据的实际情况进行调节。由于我们的数据量大（共几百万条），学习的时间成本是必须考虑的问题。而且我们经过了数据预处理，忽略掉了一些显然无用的特征，例如 groupId、winPoints，也按一定规则合并和归一化了一些特征，例如 kills，所以实际上的特征量已经减少了很多，无需再限制最大特征数 max_features。经过考虑，我们最终确定了决策树最大深度 max_depth、内部节点再划分所需最小样本数 min_samples_split、叶子节点最少样本数 min_samples_leaf 等三个参数进行调节，以限制树的大小，防止过拟合。

Max_depth 即决策树的深度，默认不限制决策树的深度，在数据量巨大时应该加以限制，通常的值应该限制在 10-100。Min_samples_split 限制了子树继续划分的条件，如果某节点的样本数少于 min_samples_split，则不会继续再尝试选择最优特征来进行划分。默认值是 2，数据量大时应该增大它的值。min_samples_leaf 限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。默认是 1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。通过查阅资料和多次调试，最终我们确定了它们的值分别是 15、10、10。

模型在测试集上的表现为 M=0.06420，M 为平均绝对值误差。此外我们本来想使用 ipython 的库生成像前面决策树模型一样的图，但是由于模型过于巨大，生成速率慢且内存会出现崩溃的情况，只能放弃。

随机森林（Random Forest）

随机森林定义

随机森林是集成学习的一种，是 Bagging 的一个扩展变体。随机森林在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。具体来说，传统决策树在选择划分属性是是从当前结点的属性集合中选择最优属性；而在随机森林中，对基决策树的每个结点，先从该结点色属性集合中随机选出一些结点，再从选出的这些结点中选择最优的属性用于划分。因此，个体学习器的“独立性”得到了提升，所以泛化能力得以提高。

随机森林的使用与结果

Python 的机器学习库 scikit-learn 中提供了用于回归和分类的随机森林的模型（RandomForestRegressor 与 RandomForestClassifier），由于本次任务是回归问题，所以我们采用 RandomForestClassifier。

在第一个线性模型中我们进行了一定量的特征工程，并且丢掉了相当一部分特征，这是因为线性模型对于特征的独立性有一定要求，而在随机森林当中，因为要保证每棵决策树之间有差异性，所以保留更多在线性模型中被丢掉的重要程度次之的特征反而会提高模型的性能。我们的试验也验证了这一点：在模型的参数完全一样的前提下，使用丢掉很多“不重要”特征的训练集训练出的模型性能上是不如保留更多特征的训练集的。这也使我们更加明白针对不同模型的特点要采用不同的特征工程。

RandomForestClassifier 模型中有若干参数，其中 n_estimators（随机森林中的决策树数量），max_features（对每个结点寻找最优属性时要考虑的特征数量），min_samples_split（拆分内部节点所需的最小样本数）与 min_samples_leaf（叶子结点所需最小样本数）几个参数对模型性能影响较大，因此需要进行调参。

n_eatimators 理论上说越多越好，因为随着个体学习器数目的增加，随机森林通常会收敛到更低的泛化误差。但是由于处理器计算能力有限，更多的个体学习器会耗费更多的时间来训练模型。在进行了多次试验后，我们选择 n_estimators=50，在保证模型性能的前提下，有相对较短的训练时间。

max_features 也会影响到模型的性能以及训练时间，如果取值太小每一棵决策树的效果不佳，而如果取值太大，则会导致随机森林中树缺乏随机性，使泛化能力下降。西瓜书中提到一般情况下推荐使用 log2，但我们实际使用下来发现效果并不好，原因是我们的数据集中特征数量本来不多（小于 30），取 log2 之后只有 4 个特征，这对一棵决策树来说肯定是不够的。因此我们排除了 log2，同样也排除了 sqrt（算术平方根），经过多次试验，最终取值 max_feature=20。

min_samples_split 和 min_samples_leaf 这两个参数限制了决策树的生长。如果取值太大则会让决策树生长受限，性能不佳，而如果取值太小则会使模型更容易捕捉到训练数据中的噪声，同样影响模型性能。根据训练集的数量级并经过多次试验，最终选值 min_samples_split=100，min_samples_leaf=50。

最后，关于随机森林中的模型性能评估，我们选择用两个值来评估模型的性能：一个是 oob_score（袋外得分），他利用了随机森林在生成不同训练集时约有 36.8%的训练集不会被选中的特点，将未被选中作为训练集的数据作为测试集，用来评估模型的性能；另一个 mean_absolute_error（平均绝对误差），是手动使用 scikit-learn 库中的 train_test_split 函数

将训练集的 30%划分为测试集，在模型训练成功以后，使用测试集预测最终排名，并与真实排名比较，求平均绝对误差。
最终使用以上参数的模型 oob_score 得分 0.925，mean_absolute_error 为 0.05997，相比于线性模型和决策树有一定提升。

总结与感受

总结与后续工作

本文介绍了我们小组使用的不同非深度机器学习模型在同一数据集上的表现以及对其原理的简单分析，总结可得线性模型速度最快，但相应地需要很好的特征工程，与此对比，更加复杂的模型则可以对无关特征的容忍度高一些，但需要以计算时间为代价。此外，复杂的模型需要更多时间来调整超参数，参数的调整方法也需要大量的实验以及工程经验。在实际应用中，机器学习能否成功运用到数据集上，不仅与模型的好坏相关，也与数据处理是否到位有很大的关系。

小组成员第一次大作业感受

郭远帆:
本次作业可以说是收获良多啊！原本只是偶然在 Kaggle 数据科学竞赛网站上看到这么一个有趣的竞赛，就想着机器学习第一次大作业就做这个吧，没想到竟然找到了两个队友并且一起完成了这个作业。
在本次作业中我的分工是基本特征工程以及线性模型。原本只是想着利用 sklearn 中的线性模型进行求解并且评估其性能，当发现 sklearn 中的线性模型并非基于梯度下降法之后，就打算自己实现一个使用梯度下降法进行训练的线性回归模型并与 sklearn 的模型进行比较。
实际证明自己被 sklearn 完爆！自己设计的模型在 1000 个数据上都需要跑接近 5 分钟，而 sklearn 基于数值解的优化方法能够在完整数据集(490 万条数据)上有着 10 秒内跑完的优秀结果，这中间差距还相当的大！在实现的过程中也是漏洞百出，调了不少 bug 最终才实现了一个普普通通的模型。
但不管怎么样，通过本次作业，我们小组每个成员都对 sklearn 机器学习库有了一定的了解，对各个模型的调参方法懂了不少。从工程意义上来说，这是一次十分成功的学习(即使在自己实现模型的过程中遇到了不少坎坷)，我们看到了实际和理论的差距，并且尝试着去深入理解，收获了不少知识。

唐荣骏：
本次作业虽说不太复杂，但是对于机器学习刚入门的同学来说收获还是很大的。以前没有尝试过 python 的编程，而如今第一次使用 python 就是与同学合作完成一个项目，不得不说还是有一定挑战性。同时也让我对 python 这一脚本语言的便利性有了更深刻的理解，

python 中种类繁多的库也对我们今后的学习尤其是机器学习提供了巨大的帮助。

再就是对于决策树算法的理解更加深刻了，以前只知道它就相当于是一个多叉树来对叶节点进行分类，但是做完这个才发觉其中参数的调节对决策树的生成十分重要，其中最主要参数就是最大特征数 max_features、决策树最大深度 max_depth、内部节点再划分所需最小样本数 min_samples_split、叶子节点最少样本数 min_samples_leaf，而各种权重的参数由于此次实验数据过于庞大，还没来得及仔细调试，相信特征数更少时这些参数发挥的作用会更加大。

罗乙然:

这是机器学习课程的第一次作业，也是我首次使用 python 语言编程，收获不少。

首先是 python 的便利性给我印象很深，不仅语法上更贴近自然语言，还有大量的非常好用的库比如 scikit-learn 和 pandas 可以调用。另外，这一次作业也消除了我对机器学习忐忑的心理。因为机器学习可以说是近两年最炙手可热的计算机名词了，在此之前给我感觉是一个充满神秘的领域。然而经过这一次自己动手编程，他的神秘面纱终于被揭下，我意识到机器学习也并不是遥不可及的概念，只要沉下心来了解他的原理与实现过程，问题也能被一一解决。当看到搭建的模型通过自己一步一步调参后越来越好时，心中的满足感与成就感是不言而喻的。

这次作业之后，我熟悉了 python 语言的基本语法，掌握了库的调用与文件的读取等等，了解了不同模型的差异与各自的优劣，特别是对我负责的随机森林部分有了较深的理解，受益匪浅。

附录

参考文献

[1]https://blog.csdn.net/ybdesire/article/details/67701289 sklearn 中的 LinearRegression 关键源码解读

[2]http://scikit-learn.org/stable/user_guide.html sklearn 库官方文档

[3]https://en.wikipedia.org/wiki/Singular_value_decomposition#Solving_homogeneous_linear_equations 维基百科-奇异值分解

[4] https://blog.csdn.net/y0367/article/details/51501780?utm_source=blogxgwz2 随机森林算法

[5] https://blog.csdn.net/qq_16633405/article/details/61200502 随机森林算法以及参数调优

[6] http://sklearn.apachecn.org/cn/0.19.0/modules/tree.html

[7] http://www.cnblogs.com/pinard/p/6050306.html

[8] https://www.cnblogs.com/pinard/p/6056319.html

[9] http://www.cnblogs.com/pinard/p/6053344.html

[10] 《机器学习》周志华