

104的笔记

目录

lec2数据预处理

2.1Data Collection

Data type数据类型

Data Storage and Presentation数据储存

2.2Discover and Visualize the Data数据可视化

Data Visualization in Python

Common Format

2.3Data Preprocessing数据预处理

什么是data

数据预处理 (无顺序要求)

lec3分类Classification

3.1分类Classification

3.2评估分类模型 Evaluating Classification

混淆矩阵 Confusion Matrix

Precision/Recall Trade-off

Area under Curve (AUC)曲线下的面积

3.3交叉验证 Cross Validation

3.4Multilabel Classification多元分类

lec4 TRAINING MODELS

4.1Linear Regression

4.2梯度下降法 (Gradient Descent)

4.3Polynomial Regression

4.4Learning Curves

4.5Regularized Linear Models

4.5Logistic Regression

lec5 DIMENSIONALITY REDUCTION

5.1 PCA

5.2Independent Component Analysis(ICA)

5.3Manifold Learning

lec6 Support Vector Machine (SVM)

6.1 Hard Margin Classification 硬间隔支持向量机 (线性可分SVM)

6.2 Soft Margin Classification

6.3Polynomial Feature

6.4 Polynomial Kernel 多项式核函数

6.5 RBF (Radial Basis Function Kernel) 高斯核

6.6 SVM Regression

lec7 Naïve Bayes

7.1 Bayes' Rule 贝叶斯法则

7.2 Naïve Bayes Classifier

lec8 Non-parametric Classification

8.1

8.2 decision tree

Gini 基尼指数

8.3 CART, 又名分类回归树

8.4 k-Nearest Neighbour

lec9Ensemble Learning and Random Forests

9.1 Voting Classifiers

9.2 Bagging and Pasting

9.3 Random Patches and Random Subspaces (features selection)

9.4 Random Forest

9.5 Feature Importance

9.6 AdaBoost

9.6 Gradient Boosting for regression

9.7 Stacking (Stacked Generalization, 层叠/堆栈泛化)

lec10 Unsupervised Learning

lec2 数据预处理

2.1 Data Collection

Data type 数据类型

structured or unstructured table or free text

Data Storage and Presentation 数据储存

1 CSV (Comma Separated Values) 逗号分隔

2 TSV (Tab Separated Values) tab分隔

3 XML (eXtensible Markup Language)

4 JSON (JavaScript Object Notation)

(3, 4 人可以看的懂点)

2.2 Discover and Visualize the Data 数据可视化

Data Visualization in Python

• Matplotlib • Seaborn • Pandas.plotting

例：pandas 读CSV文件做出table

```
1 | import pandas as pd
2 | #pd.read_csv("csv.txt")
3 | pd.read_csv("tsv.txt")
```

Common Format

• Line Charts • Bar Graphs • Histograms • Scatter Plots • Heat Maps 画出各种统计图

例：Matplotlib做出line charts 和 bar 和 his

```
1 | import matplotlib.pyplot as plt
2 | plt.plot(data_csv["before"])
3 | data_csv["before"].plot.bar()
4 | data_csv.hist()
```

2.3 Data Preprocessing 数据预处理

什么是data

Feature: an individual measurable property or characteristic of a phenomenon.

Instance: a sample or data point, refers to a single observation or example in the dataset

Target variable

Dataset: A dataset is a collection of instances, features, and target variables that are used to train and test machine learning models.

The diagram shows a table representing a dataset. The columns are labeled: longitude, latitude, housing_median_age, total_rooms, total_bedrooms, and population. There are six rows of data. An orange box highlights the second row from the bottom, with an arrow pointing down to it labeled 'Instance'. Another orange box highlights the third column from the left, with an arrow pointing down to it labeled 'Feature'.

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
-122.23	37.88	41	880	129	322
-122.22	37.86	21	7099	1106	2401
-122.24	37.85	52	1467	190	496
-122.25	37.85	52	1274	235	558
-122.25	37.85	52	1627	280	565
-122.25	37.85	52	919	213	413

CSDN @m0_74767877

数据预处理 (无顺序要求)

1 Data Cleaning

*data munging 整理例如把之前的没有结构的text一段话转化成整理成结构数据如table

*Handling Missing Data 处理一些丢失的数据 可以直接删除或者去设置为一些值 (如平均值, 0, 中位数等)

*Smooth Noisy Data (更难, 因为还要分辨定义什么是noisy) 处理一些离散值, 干扰值

```

1 alcohol= pd.read_excel("alcohol.xlsx")
2 alcohol.dropna(axis=1)#删除missingdata所在的列
3 median_heard=alcohol[ "Heart" ].median()
4 alcohol[ "Heart" ].fillna(median_heard, inplace=True)#填充na列为中值
5 alcohol.loc[alcohol[ "Deaths" ]<=0, "Deaths" ]=0#处理不合理的noisydata为0

```

2 Data Integration

处理一些单位问题, 重复什么的

Combine

- Combine data from multiple sources into a coherent storage place (e.g., a single file or a database).

Resolve conflicts

- Different representations or different scales; for example, metric vs. British units.

Remove redundant

- The same attribute may have different names in different databases.
- One attribute may be a “derived” attribute in another table; for example, annual revenue.
- Correlation analysis may detect instances of redundant data

CSDN @m0_74767877

3 Data Transformation

使数据更加的可读, 例如用Python把数据变成负一到一的数值以0为中心, 更加清晰地显示数据

使一个很分散的散点图变得集中更显示出其中的关系

Normalization

- Min-max normalization. (usually 0 - 1)

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Z-score normalization. (standardization)

Normalizing every value in a dataset such that the mean of all of the values is 0 and the standard deviation is 1

$$x_{scaled} = \frac{x - mean}{sd}$$

- Normalization by decimal scaling.

$$x_{scaled} = \frac{x}{10^j}$$

CSDN @m0_74767877

- Handling Text and Categorical Attributes

i.e. ["cat1"], ["cat2"], ["cat3"], ["cat4"]

- Ordinal encoder: `from sklearn.preprocessing import OrdinalEncoder`
[0], [1], [2], [3]

- One-hot encoder: `from sklearn.preprocessing import OneHotEncoder`
[1,0,0], [0,1,0], [0,0,1], [0,0,1]

CSDN @m0_74767877

4 Data Reduction数据规约

数据规约方法类似数据集的压缩，它通过维度的减少或者数据量的减少，来达到降低数据规模的目的，数据压缩（Data Compression）有无损！压缩。方法主要是下面两种：

1.维度规约（Dimensionality Reduction）：减少所需自变量的个数。代表方法为WT、PCA与FSS。

2.数量规约（Numerosity Reducton）：用较小的数据表示形式替换原始数据。代表方法为对数线性回归、聚类、抽样等。

关于Dimensionality Reduction降维的延伸

Feature Selection特征抽取和Feature Extraction特征选择是Dimensionality Reduction（降维）两种方法

这两者达到的效果是一样的，就是试图去减少特征数据集中的属性(或者称为特征)的数目；但是所采用的方式方法却不同：特征抽取的方法主要属性间的关系，如组合不同的属性得新的属性，这样就改变了原来的特征空间；而特征选择的方法是从原始特征数据集中选择出子集，是一种包关系，没有更改原始的特征空间。

1. Feature Selection特征抽取

主成分分析(PCA)和线性评判分析(LDA)是特征抽取的两种主要经典方法

2. Feature Extraction特征选择

过滤式(filter): 先进行特征选择，然后去训练学习器，所以特征选择的过程与学习器无关。相当于先对于特征进行过滤操作，然后用特征子集来分类器。

包裹式(wrapper)：直接把最后要使用的分类器作为特征选择的评价函数，对于特定的分类器选择最优的特征子集。

Filter和Wrapper组合式算法：先使用Filter进行特征选择，去掉不相关的特征，降低特征维度；然后利用Wrapper进行特征选择。

嵌入式(embedding)：把特征选择的过程与分类器学习的过程融合一起，在学习的过程中进行特征选择。最常见的使用L1正则化进行特征选择

lec3分类Classification

3.1分类Classification

这周主要讲二分类binary classification。两个label，圈或者叉去分类

利用训练集（特征值和label）训练模型，然后用这些模型去分类新的个体(只有特征值)



· 训练Training

依据训练集(training set)中的数据，训练集的数据被其标签明确分类，模型需要做的是学会按照这些给定数据确定分类标准

· 推论Inference

用于对未来的未知物体进行分类，在这之前需要估计模型的精度accuracy以评估模型：

评估模型好坏需要使用测试集test set，测试集需要独立于训练集，以测试模型对未探知的个体的分类精度。

Accuracy rate: correctly classified samples / test set (正确预测的除以所有的测试集)

3.2评估分类模型 Evaluating Classification

混淆矩阵 Confusion Matrix

		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive	False positive, Type I error
	Predicted condition negative	False negative, Type II error	True negative

True Positive (TP): 模型判断为 + (正P) , 实际情况也是 + (判断正确True)

True Negative (TN): 模型判断为 - (负N) , 实际情况也是 - (判断错误True)

False Positive (FP): 模型判断为 + (正P) , 实际情况却是 - (判断错误False)

False Negative (FN): 模型判断为 - (负N) , 实际情况却是 + (判断错误False)

利用TP,TN,FP,FN构建的算式，形容模型在某一方面的能力。如下

将实际情况为正的集合称为n+, 将实际情况为负的集合称为n-

1. **Recall** (Sensitivity): 所有实际为正的个体中分类为正的比例

$$TP / (TP+FN) = TP/n+$$

2. **Fall-out**:

$$FP / (TN+FP) = TP/n-$$

3. **Miss rate**

$$FN / (TP+FN) = FN/n+$$

4. **Specificity** (Selectivity)

$$TN / (TN+FP) = TN/n-$$

上面4个都是该格子在该列的占比，也叫 xxx Rate

Recall (Sensitivity)	True Positive Rate	(1)
Fall-out	False Positive Rate	(3)
Miss rate	False Negative Rate	(2)
Specificity (Selectivity)	True Negative Rate	(4)

5. **Precision**, 在所有被分类为正的个体中实际为正的比例

$$TP / (TP+FP)$$

6. **Accuracy**, 模型正确分类的比例

$$(TN+TP) / (TP+TN+FP+FN)$$

F-Measure

将Precision和Recall加权调和平均:

$$F_{\beta} = \frac{(\beta^2 + 1)Precision * Recall}{\beta^2 Precision + Recall}$$

β 置为1时，式子即为**F1-Measure**(应该会出关于这些的计算题？)：

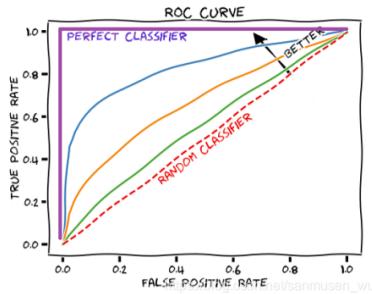
$$F_1 \text{ score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Precision/Recall Trade-off

一个不错杀一个不放过 精准度越高预测越准不错杀

Area under Curve (AUC)曲线下的面积

1是完美0是全错



3.3 交叉验证 Cross Validation

通过使用部分训练集作为交叉验证集，增强验证集表达整体数据的能力
可以用于模型评估，进而可以选择最好的学习模型（超参数选择）。



k折交叉测试集进行验证 k-fold Cross Validation

对模型的评估。

将数据划分为k个互斥且大小相等的子集，

进行k次训练和测试，每次选取第*i*个子集作为测试集 ($1 \leq i \leq k$)，其余子集作为训练集。用k次测试的结果的平均数去评估的模型的performance

该方法可以避免固定划分数据集的局限性、特殊性，这个优势在小规模数据集上更明显。

*小规模数据集的k应该偏大，以采纳更大的训练集（小训练集可能导致训练不充分）

*大规模数据集的k应该偏小，以采纳更大的测试集（训练集够充分了）

Experiment #1	Test set	
Experiment #2		Test set
Experiment # i		Test set
	:	
Experiment # k		Test set

留一交叉验证 Leave-one-out Cross Validation

k=数据样本的总数量n, n个数据划分成k个子集，每个子集只包含一个数据，每次只留下一个单个样本作为测试集，其余n-1个样本作为训练集。

训练集与验证集互斥：因为在训练过程中有选择表现较好的那一种模型的需要，所以使用验证集

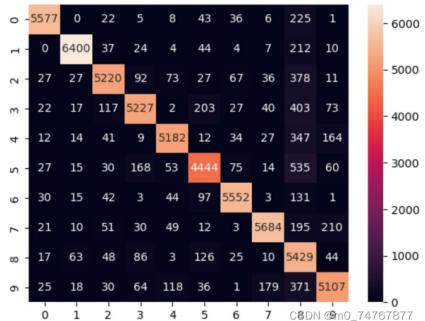
测试集与（训练集，验证集）互斥：因为有评估模型在应用到实际生活中的正确率的需要，而且又因为模型完全学习于训练集，导致需要与训练集互斥；模型使用了测试集选择最优模型，无形中依赖于测试集降低了一点错误率，为了真实性，需要与验证集互斥

3.4 Multilabel Classification 多元分类

一对剩余所有组成一个二元分类器

or 组多个一对二元分类器

- One-versus-the-rest(OvR) strategy: train multiple binary classifiers for each class, select the class whose classifier outputs the highest score.
 - train N times
- One-versus-one (OvO) strategy: train a binary classifier for every pair of classes
 - train $N(N-1)/2$ times



lec4 TRAINING MODELS

4.1 Linear Regression

简单线性回归单变量

$$\hat{y} = kx + b$$

\hat{y} : predicted value (output)

x : feature value (input)

k,b: model parameters (b: bias term, k: weight)

b也叫截距 预测值和实际值的差叫做残差 residuals \hat{e}

regressed value追求最小的残差平方和

convergence收敛

$$RSS = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

线性回归模型的向量形式

$$\hat{y} = h_{\theta}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$$

- $\boldsymbol{\theta}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n .
- \mathbf{x} is the instance's *feature vector*, containing x_0 to x_n , with x_0 always equal to 1.
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and \mathbf{x} , which is of course equal to $\theta_0x_0 + \theta_1x_1 + \theta_2x_2 + \dots + \theta_nx_n$.
- h_{θ} is the hypothesis function, using the model parameters $\boldsymbol{\theta}$.

Note

In Machine Learning, vectors are often represented as column vectors. If $\boldsymbol{\theta}$ and \mathbf{x} are column vectors, then the prediction is $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$, where $\boldsymbol{\theta}^T$ is the transpose of $\boldsymbol{\theta}$ (a row vector instead of a column vector)

对于这样一个预测模型采用Mean Squared error (MSE) (均方误差) 作为他的cost function (代价函数)

$$MSE(\mathbf{X}, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

找到最好的~~cost~~来最小化这个代价函数有两种办法

最小二乘法矩阵求解、正规方程法 (Normal Equation)

我看不懂啊看不懂！

对于简单的算法可以这样去优化

$$MSE(\mathbf{X}, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

$$h_{\theta}(\mathbf{x}^{(i)})$$

$$\alpha \nabla_j \left((\text{predicted} - \text{actual})^2 \right)$$

Linear Regression

$\tilde{x}^i \rightarrow (m+1) \times 1$

- **Cost function:** Mean Squared error (MSE) for a Linear Regression model

$$\text{MSE}(\mathbf{X}, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})^2 = \frac{1}{m} \|\mathbf{X}\theta - \mathbf{y}\|_2^2$$

Training the model is the process to find the value of θ that minimizes the cost function.

$$\frac{\partial \text{MSE}}{\partial \theta} = 0$$

- **Normal Equation:**

$$\hat{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- $\hat{\theta}$ is the value of θ that minimizes the cost function

- y is the vector of targeted values containing $y^{(1)}$ to $y^{(m)}$

$$\hat{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

CSDN @m0_7476

道爷，我悟了！

the cost function.

$$\frac{\partial \text{MSE}}{\partial \theta} = 0$$

Normal Equation:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\hat{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- $\hat{\theta}$ is the value of θ that minimizes the cost function
- y is the vector of targeted values containing $y^{(1)}$ to $y^{(m)}$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \end{array}$$

CSDN @m0_74767877

逆矩阵用待定系数法去求，乘数拆分去乘以被乘数再相加。x1默认等于1所以直接有bias term是ceta0

4.2 梯度下降法 (Gradient Descent)

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

描述： α 是学习率，它决定了我们下山过程中迈出的步子有多大。求导的目的是为了确定下山的方向，等式右面的 θ_j 是旧的参数，左边的 θ_j 是新的参数，每走一步（每一次迭代）我们都需要确定新的参数，因为只有更新了参数才能确定下一步的方向。

注意事项：1. 学习率不可过小也不可过大，会导致local minimal和oscillation（震荡）和slow convergence（慢收敛）

2. 如果函数很复杂有极小值和最小值，梯度下降容易卡住

可以多次选择初始ceta比对避免2情况的出现

对于多维特征

多维特征有可能比例尺或者说数量级不同导致高线图非常的扁导致迭代需要很多很多次，所以可以先normalization特征使图像变圆，使迭代次数j

梯度下降	正规方程
需要选择学习率 α	不需要
需要多次迭代	一次运算得出
当特征数量 n 大时也能较好适用	需要计算 $(X^T X)^{-1}$ 如果特征数量 n 较大则运算代价大, 因为矩阵逆的计算时间复杂度为 $O(n^3)$, 通常来说当 n 小于 10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型, 不适合逻辑回归模型等其他模型 CSDN @无咎.lsy

*批量梯度下降 (Batch Gradient Descent)

批量梯度下降法是最原始的形式, 它的具体思路是在更新每一参数时都使用所有的样本来进行梯度的更新。把那个抽出来的 $ceta$ 对所有样本算代价数 (mse)

优点:

- (1) 一次迭代是对所有样本进行计算, 此时利用矩阵进行运算, 实现了并行。
- (2) 由全数据集确定的方向能够更好地代表样本总体, 从而更准确地朝向极值所在的方向。当目标函数为凸函数时, 批量梯度下降一定能够得最优解。

缺点:

- (1) 有时我们会遇到样本数目 m 很大的训练集合, 这意味着我们每执行一次批梯度下降算法, 都要对 m 个样本进行求和。我们的程序也就需要这上百万的样本, 甚至我们完成值下降的第一步都十分困难。这样会导致, **训练过程很慢, 花费很长的时间**。

*随机梯度下降 (Stochastic Gradient Descent)

算法中对 Θ 的每次更新不需要再全部遍历一次整个样本, 只需要查看一个训练样本进行更新, 之后再用下一个样本进行下一次更新, 像批梯度一样不断迭代更新。只对一个样本算代价函数 (mse)。

优点:

- (1) 由于不是在全部训练数据上的损失函数, 而是在每轮迭代中, 随机优化某一条训练数据上的损失函数, 这样每一轮参数的**更新速度大大加强**

缺点:

- (1) **准确度下降**。由于即使在目标函数为强凸函数的情况下, SGD仍旧无法做到线性收敛。
- (2) 可能会收敛到局部最优, 由于单个样本并不能代表全体样本的趋势。
- (3) 不易于并行实现

*小批量梯度下降 (Mini-Batch Gradient Descent)

每次迭代使用 ** batch_size** 个样本来对参数进行更新。它克服上面两种方法的缺点, 又同时兼顾两种方法的优点。确定一个batch大小只对小算代价函数 (mse)。

优点:

- (1) 通过矩阵运算, 每次在一个batch上优化神经网络参数**并不会比单个数据慢太多**。
- (2) 每次使用一个batch可以**大大减小收敛所需要的迭代次数**, 同时可以使收敛到的结果更加**接近梯度下降**的效果。(比如上例中的30W, 设置batch_size=100时, 需要迭代3000次, 远小于随机梯度下降的30W次)
- (3) 可实现并行化。

缺点:

- (1) batch_size的不当选择可能会带来一些问题

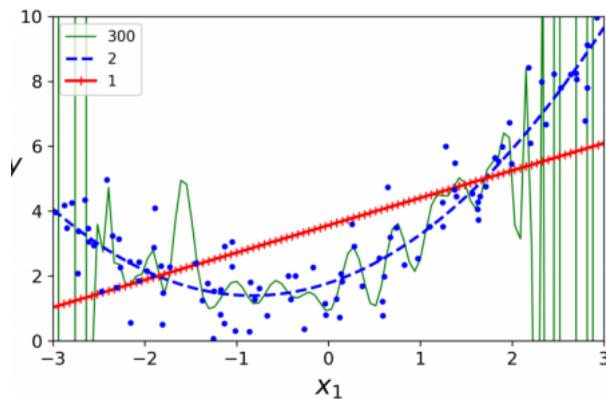
4.3 Polynomial Regression

将非线性的换元成多元线性回归问题, 例如里面有 x 平方就可以把它看成 x^2

4.4 Learning Curves

*过拟合: 一个假设在训练数据上能够获得比其他假设更好的拟合, 但是在测试数据集上却不能很好地拟合数据, 此时认为这个假设出现了过拟象。(模型过于复杂)

*欠拟合: 一个假设在训练数据上不能获得更好的拟合, 并且在测试数据集上也不能很好地拟合数据, 此时认为这个假设出现了欠拟合的现象。(过于简单)



红色欠拟合 蓝色很合适 绿色过拟合 (在测试集好, 验证集不好)

总结一下:

欠拟合: 泛化能力差, 训练样本集准确率低, 测试样本集准确率低。

过拟合: 泛化能力差, 训练样本集准确率高, 测试样本集准确率低。

合适的拟合程度: 泛化能力强, 训练样本集准确率高, 测试样本集准确率高

欠拟合原因:

训练样本数量少

模型复杂度过低

参数还未收敛就停止循环

欠拟合的解决办法:

增加样本数量

增加模型参数, 提高模型复杂度

增加循环次数

查看是否是学习率过高导致模型无法收敛

过拟合原因:

数据噪声太大

特征太多

模型太复杂

过拟合的解决办法:

清洗数据

减少模型参数, 降低模型复杂度

增加惩罚因子 (正则化), 保留所有的特征, 但是减少参数的大小 (magnitude)。

4.5 Regularized Linear Models

正则化是用来防止模型过拟合而采取的手段。我们对代价函数增加一个限制条件, **限制其较高次的参数大小不能过大**。所以如果我们能让这些高次的系数接近于0的话, 我们就能很好的拟合了, 因此, 我们对代价函数 $J(\theta)$ 进行修改

L2 正则化ridge

L2正则化对于绝对值较大的权重予以很重的惩罚, 对于绝对值很小的权重予以非常非常小的惩罚, 当权重绝对值趋近于0时, 基本不惩罚。这个L2的平方项有关系, 即越大的数, 其平方越大, 越小的数, 比如小于1的数, 其平方反而越小。

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

L1正则化lasso

随着海量数据处理的兴起, 工程上对于模型稀疏化的要求也随之出现了。这时候, L2正则化已经不能满足需求, 因为它只是使得模型的参数值趋近于0, 而不是等于0, 这样就无法丢掉模型里的任何一个特征, 因此无法做到稀疏化。这时, L1的作用随之显现。L1正则化的作用是使得大部分模型的值等于0, 这样一来, 当模型训练好后, 这些权值等于0的特征可以省去, 从而达到稀疏化的目的, 也节省了存储的空间, 因为在计算时, 值为0的特征都可以不用存储了。

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^n |\theta_i|$$

Elastic Net

ElasticNet 是一种使用L1和L2先验作为正则化矩阵的线性回归模型。这种组合用于只有很少的权重非零的稀疏模型, 比如: class:Lasso, 但是又能保持: class:Ridge 的正则化属性。我们可以使用 l1_ratio 参数来调节L1和L2的凸组合(一类特殊的线性组合)。

$$J(\boldsymbol{\theta}) = \text{MSE}(\boldsymbol{\theta}) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n \theta_i^2$$

附录：Early stopping防止过拟合

当我们训练深度学习神经网络的时候通常希望能获得最好的泛化性能（generalization performance，即可以很好地拟合数据）。但是所有的梯度学习神经网络结构如全连接多层感知机都很容易过拟合：当网络在训练集上表现越来越好，错误率越来越低的时候，实际上在某一刻，它在测试的表现已经开始变差。

4.5 Logistic Regression

逻辑回归是用来进行分类的。我们处理二分类问题。由于分成两类，我们便让其中一类标签为0，另一类为1。我们需要一个函数，对于输入的数据 $x^{(i)}$ ，都能映射成0~1之间的数。并且如果函数值大于0.5，就判定属于1，否则属于0。

代价函数cost function

就不好使用mse了会有一些问题

对于第一个 对了就是0，错了就是负无穷

第二个就是第一个的联合

第三个说是梯度下降，看着难但不是你要算

Cost function of a single training instance

$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1 - \hat{p}) & \text{if } y = 0 \end{cases}$$

Logistic Regression cost function (log loss)

$$J(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \right]$$

Logistic cost function partial derivatives

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m (\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

决策边界Decision Boundaries

就上面的逻辑回归，小于0.5的算0大于的算1，0.5就是决策边界

如果有多个class怎么办

1.如之前讲的可以建立一对剩余或者一对一的多个二元分类器

2.softmax regression主要用于解决多元分类问题。主要是估算对于输入样本 $x^{(i)}$ 属于每一类别的概率，所以softmax回归的假设函数如下：

Softmax Regression: for Multinomial Logistic Regression.

Softmax score for class k : $s_k(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}^{(k)}$

Softmax function: $\hat{p}_k = \sigma(s_k(\mathbf{x}))_k = \frac{\exp(s_k(\mathbf{x}))}{\sum_{j=1}^K \exp(s_j(\mathbf{x}))}$

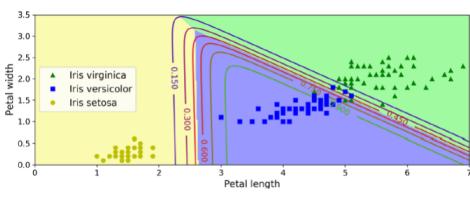


Figure 4-25. Softmax Regression decision boundaries

lec5 DIMENSIONALITY REDUCTION

5.1 PCA

Data with high dimensions:

- High computational complexity
- May contain many irrelevant or redundant features
- Difficulty in visualization
- With high risk of getting an overfitting model

主成分分析算法 (PCA) 是最常用的线性降维方法，它的目标是通过某种线性投影，将高维的数据映射到低维的空间中，并期望在所投影的维度的信息量最大（方差最大），以此使用较少的数据维度，同时保留住较多的原数据点的特性。

PCA降维的目的，就是为了在尽量保证“信息量不丢失”的情况下，对原始特征进行降维，也就是尽可能将原始特征往具有最大投影信息量的维度投影。将原特征投影到这些维度上，使降维后信息量损失最小。

1. 标准化连续初始变量的范围 (非结构化转成结构化) We must standardize the data first

- 线性函数归一化。将原始数据进行线性变换，使结果映射到[0,1]的范围，实现对原始数据的等比缩放。归一化公式如下：

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

- 零均值归一化。它会将原始数据映射到均值为0，标准差为1的分布上。具体来说，假设原始特征的均值 μ ，标准差为 σ ，那么归一化公式定义为：

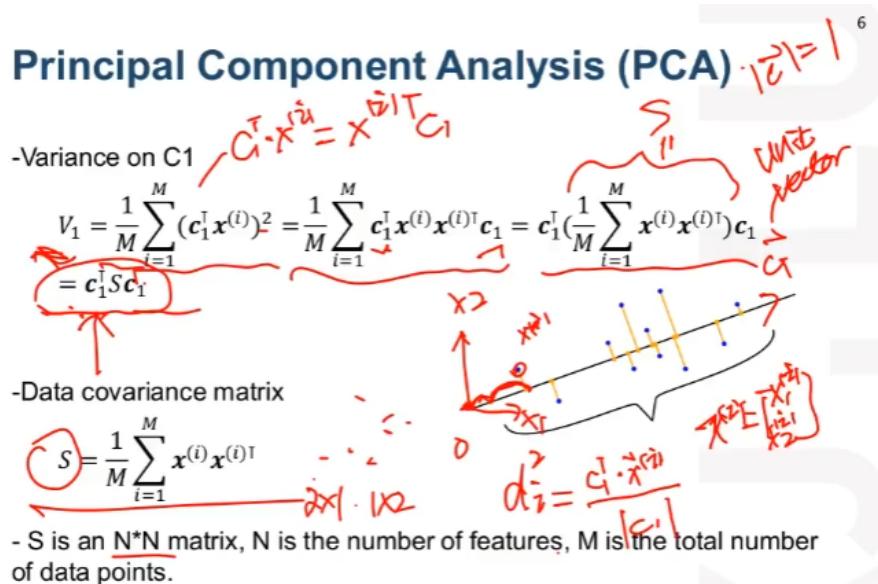
$$z = \frac{x - \mu}{\sigma}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n}}$$

2. 计算协方差矩阵以识别相关性 covariance matrix

σ 是variance (方差) 指离中心点的距离 用向量乘法求解

最大方差理论：方差越大，信息量就越大。协方差矩阵的每一个特征向量就是一个投影面，每一个特征向量所对应的特征值就是原始特征投影到影面之后的方差。由于投影过去之后，我们要尽可能保证信息不丢失，所以要选择具有较大方差的投影面对原始特征进行投影，也就是选择具有特征值的特征向量。然后将原始特征投影在这些特征向量上，投影后的值就是新的特征值。每一个投影面生成一个新的特征， k 个投影面就生成特征。



Principal Component Analysis (PCA)

-Constrained optimization problem

$$\max_{\mathbf{c}_1} \mathbf{c}_1^T S \mathbf{c}_1$$

subject to $\|\mathbf{c}_1\|^2 = 1$

-Lagrangian $L(\mathbf{c}_1, \lambda_1) = \mathbf{c}_1^T S \mathbf{c}_1 + \lambda_1(1 - \mathbf{c}_1^T \mathbf{c}_1)$

-Solve this constrained optimization problem

$$\frac{\partial L}{\partial \mathbf{c}_1} = 2S\mathbf{c}_1 - 2\lambda_1 \mathbf{c}_1 = 0$$

$$\frac{\partial L}{\partial \lambda_1} = 1 - \mathbf{c}_1^T \mathbf{c}_1 = 0$$

- Setting these partial derivatives to 0 gives us the relations:

$$\mathbf{S}\mathbf{c}_1 = \lambda_1 \mathbf{c}_1 \quad \text{and} \quad \mathbf{c}_1^T \mathbf{c}_1 = 1$$

Variance on \mathbf{C}_1

$$V_1 = \mathbf{c}_1^T S \mathbf{c}_1 = \lambda_1 \mathbf{c}_1^T \mathbf{c}_1 = \lambda_1$$



Practice: PCA

$$x = \frac{x - \text{mean}}{\text{std}}$$

feature 1 $u = 3.25 \quad \sigma = 2.5$

$$d u = 5.75 \quad \sigma = 2.5$$

Given a dataset that consists of the following points below:

$$A = (2, 3), B = (5, 5), C = (6, 6), D = (8, 9)$$

1. Calculate the covariance matrix for the dataset.

2. Calculate the eigenvalues and eigenvectors of the covariance matrix.

$$S = \frac{1}{n} \sum x^{(i)} x^{(i)T}$$

$$A' = (-1, -1)$$

$$B' = (0, 1)$$

$$C' = (0.3, 0.1)$$

$$D' = (1, 1)$$

$$S = \frac{1}{4} \left(\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right)$$



$$\begin{aligned}
 SC &= \lambda C \\
 (S - \lambda I)C &= 0 \\
 \xrightarrow{\text{Subtract } 0.75 - \lambda} \quad & \left| \begin{array}{cc} 0.75 - \lambda & 0.73 \\ 0.73 & 0.75 - \lambda \end{array} \right| = 0 \\
 & \nearrow \swarrow \quad \nearrow \searrow \\
 (0.75 - \lambda)^2 - 0.73^2 &= 0 \\
 \lambda_1 &= 1.48 \rightarrow c_1 \rightarrow \left[\begin{array}{cc} 0.75 - 1.48 & 0.73 \\ 0.73 & 0.75 - 1.48 \end{array} \right] \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = 0 \\
 \lambda_2 &= 0.02 \rightarrow c_2 \rightarrow \begin{bmatrix} c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \frac{1}{\sqrt{2}}
 \end{aligned}$$

$$\text{或 } (\lambda E - A)x = 0$$

$$\text{由 } Ax = \lambda x \quad |\lambda E - A| = 0$$

$$\Rightarrow (A - \lambda E)x = 0 \quad (*)_{\lambda}$$

而 $\alpha = \beta$, 即为区域方程组 (A) 的解.

$$\Leftrightarrow |A - \lambda L| = 0$$

方程组(\cdot_{λ})的解空间称为对应于 λ 的特征子空间.

Namita是特征值eigenvalues E是对角线为1其他是0的单位向量

Singular Value Decomposition (SVD)

在降维过程中，我们会减少特征的数量，将那些带有重复信息的特征合并，并删除那些带无效信息的特征等等——逐渐创造出能够代表原特征大部分信息的，特征更少的，新特征矩阵

SVD is another way to perform PCA

BOAT&CAMP是TECHNICAL MUSEUM TOKYO的附属机构

——它们并不了解特征分解，但他们知道从上面的矩阵中提取特征，只是向特征分解方法学习，但心里的衡量目标一去不复返了。使用方差作为信息量的衡量指标，并且特征值分解来找出空间V。降维时，它会通过一系列数学的神秘操作（比如说，产生协方差矩阵）将特征分解为以下三个矩阵，Namita？其对角线上的元素就是方差。

$X \rightarrow$ 数学的神秘宇宙 $\rightarrow Q\Sigma Q^{-1}$

SVD使用奇异值分解来找出空间V，其中 Σ 也是一个对角矩阵，不过它对角线上的元素是奇异值，这也是SVD中用来衡量特征上的信息量的指标。 V^T 分别是左奇异矩阵和右奇异矩阵，也都是辅助矩阵。

$X \rightarrow$ 另一个数学的神秘宇宙 $\rightarrow U\Sigma V^T$

- $U \in R^{m*m}$ is an orthogonal matrix with column vectors $u_i, i = 1, \dots, m$,

- $V \in R^{n*n}$ an orthogonal matrix with column vectors $v_j, j = 1, \dots, n$.

- Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0$ and $\Sigma_{ij} = 0, i \neq j$

- **The singular value matrix Σ is unique**



PCA

Singular Value Decomposition (SVD)

$$A = [x_1 \dots x_m]_{n*m} = U\Sigma V^\top = [u_1 \dots u_n]_{n*n} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix}_{n*m} [v_1 \dots v_m]_{m*m}^\top$$

V contains the unit vectors that define all the principal components that we are looking for.

12

Principal Component Analysis (PCA)

Principal components matrix

$$V = \begin{pmatrix} | & | & | \\ c_1 & c_2 & \dots & c_n \\ | & | & | \end{pmatrix}$$

c_1, c_2, \dots, c_n are orthogonal

$\cancel{N} \rightarrow S \rightarrow n \times n$

$$\begin{matrix} | & & | \\ c_1 & \cdots & c_n \\ | & & | \end{matrix}$$

Projecting Down to d Dimension:

$$X_{d\text{-proj}} = XW_d$$

$$W_d = \begin{bmatrix} | & | & | \\ c_1 & c_2 & \dots & c_d \\ | & | & | \end{bmatrix}$$

W_d is the first d eigen vectors of data covariance matrix

Explained Variance Ratio

$$C_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \dots + \lambda_n} \quad (\text{eigenvalue/ total eigenvalue})$$

$$= \frac{\sum \lambda_1 + \lambda_2}{\sum \lambda_i}$$

$$\begin{matrix} | & & | \\ \lambda_1 & \lambda_2 & \dots & \lambda_n \\ | & & | \end{matrix}$$



Choosing the Right Number of Dimensions:

- Choose the number of dimensions that add up to sufficiently large portion of the variance (e.g., 95%)

$$\frac{\lambda_1 + \dots + \lambda_d}{\lambda_1 + \lambda_2 + \dots + \lambda_n} > 95\%$$



Figure 8-8. Explained variance as a function of the number of dimensions

最好大于95

PCA for Compression

- Projecting Down to d Dimension

$$X_{d-proj} = XW_d$$

- PCA inverse transformation, back to the original number of dimensions

$$X_{recovered} = X_{d-proj}W_d^\top$$

5.2 Independent Component Analysis(ICA)

如果所有的特征都是独立的没有主次之分所以pca就不太够用

17

PCA vs ICA

PCA

Deals with the **Principal Components**.

Focuses on maximizing the variance

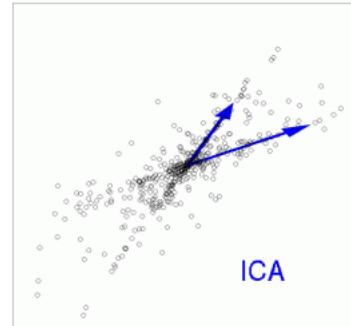
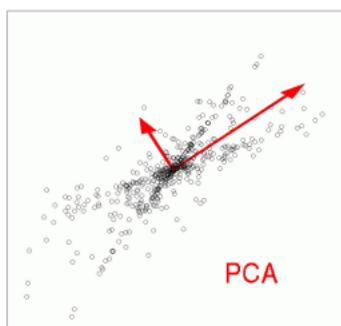
Vectors are orthogonal

ICA

Deals with the **Independent Components**

Focuses on the mutual independence of the components.

Vectors are **not** orthogonal

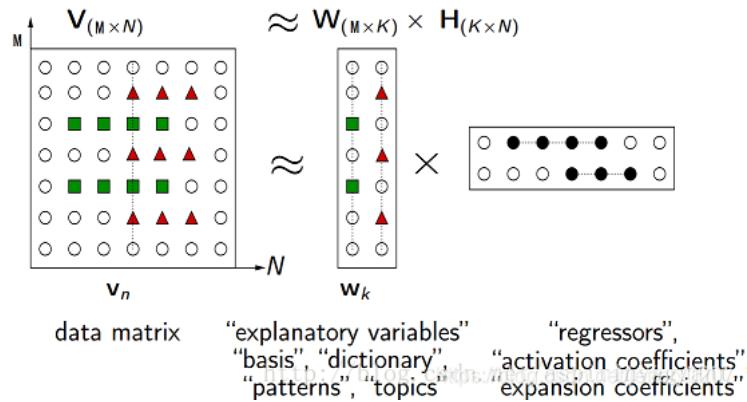


PCA主要用于数据的降维，将原本相关的数据映射到一个新的空间中，在这个空间中，数据的各个维度是不相关的；ICA是一种找到数据可能在各组成分量的方法，主要应用于信号解混，特征提取，使用ICA时往往对数据存在几点假设：源信号独立，且服从非高斯分布，甚至要假设源信率分布和个数。ICA使得WTx的非高斯性最大，找到独立分量。PCV往往是ICA预处理的一部分。

NMF(非负矩阵分解)

NMF在人脸识别的应用中和PCA还有VQ分解不同。VQ分解是用一张完整的图像直接代表源脸部图像；PCA是将几个完整人脸加减压成一张脸；NMF是取甲的眼睛，乙的鼻子，丙的嘴巴直接拼成一张脸，也就是说NMF分解后的基矩阵H是每张人脸的一个特征部分，例如眼睛，鼻子，嘴巴然后权重矩阵对这些特征进行加权处理从而得到一张完整的人脸。如下图所示3种矩阵分解方式的区别。

NMF(Non-negative matrix factorization)，即对于任意给定的一个非负矩阵V，其能够寻找到一个非负矩阵W和一个非负矩阵H，满足条件 $V=W \cdot H$ 将一个非负的矩阵分解为左右两个非负矩阵的乘积。^{**}其中，V矩阵中每一列代表一个观测(observation)，每一行代表一个特征(feature)；W矩阵称基矩阵，H矩阵称为系数矩阵或权重矩阵。这时用系数矩阵H代替原始矩阵，就可以实现对原始矩阵进行降维，得到数据特征的降维矩阵，从而储空间。^{**}过程如下图所示：



NMF is not unique

- A matrix factorization where everything is non-negative
- $-A \in R^{n*m}$ is original non-negative data
- $-W \in R^{n*k}$ is matrix of basis vectors, dictionary elements
- $-H \in R^{k*m}$ is matrix of activations, weights, or gains

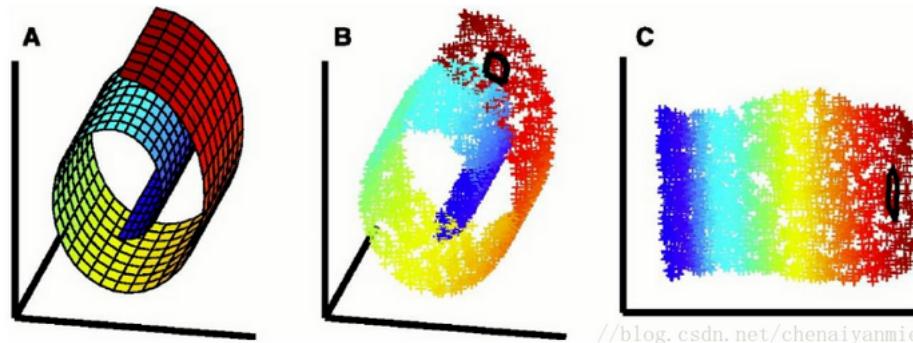
5.3 Manifold Learning

PCA是目前应用最广泛的线性降维方法，采用线性投影的方法进行数据降维，使数据在给定方向上的投影得到最大的投影方差，当流形是一个线形时，PCA得到的结果是最优的，PCA无法有效处理非线性流形上的数据。

主要的代表方法LLE(Locally Linear Embedding) 局部线性嵌入

一个流形在很小的局部邻域上可以近似看成是欧氏的，即局部线性的。那么，在小的局部邻域中，一个点就可以用它周围的点在最小二乘意义下的最优线性来表示。LLE方法即是把这种线性拟合的系数当成这个流形的局部几何性质的描述。同样的，一个后的低维空间数据表示，也就应该具有同样的局部几何性质，所以利用同样的线性表示的表达式，最终写成一个二次型的形式。

概括的来说就是：“流形在局部可以近似等价于欧氏空间”。



//blog.csdn.net/chenaiyanmie

LLE算法步骤：

1. 构建邻域图。对于原始空间中任一给定的样本点 x_i ，用K近邻法得到它的一组邻域点 x_j ；
2. 计算权重。由局部线性假设，样本点 x_i 可用 K 个邻域点 x_j 线性表示出来，即 $x_i \approx \sum W_{i,j}x_j$ ，用权重描述出每一样本点与其邻域点之间的关系，权重 $W_{i,j}$ 是使得样本点 x_i 用其 K 个邻域点 x_j 重构误差最小的解的系数：

$$\xi(W) = \sum_i |x_i - \sum_j W_{i,j}x_j|^2$$

使 $\xi(W)$ 的值为最小，则 $W_{i,j}$ 即为各样本点最优的线性表示参数，即权重。

3. 将这种局部线性结构嵌入低维空间。嵌入操作是通过最小化误差来尽可能多的保留原空间的性质：

$$\xi(y) = \sum_i |y_i - \sum_j W_{i,j}y_j|^2$$

这里的 $W_{i,j}$ 是第二步计算的权重值， y_i 与 y_j 是样本点在嵌入空间的投影。

21

Locally Linear Embedding (LLE)

LLE is a powerful *nonlinear dimensionality reduction* (NLDR) technique.
It is a Manifold Learning technique that does not rely on projections

Step one: Linearly modeling local relationships $\hat{x}^{(i)} \approx \sum w_{i,j}x^{(j)}$

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^m \left(\mathbf{x}^{(i)} - \sum_{j=1}^m w_{i,j} \mathbf{x}^{(j)} \right)^2$$

subject to $\begin{cases} w_{i,j} = 0 & \text{if } \mathbf{x}^{(j)} \text{ is not one of the } k \text{ c.n. of } \mathbf{x}^{(i)} \\ \sum_{j=1}^m w_{i,j} = 1 & \text{for } i = 1, 2, \dots, m \end{cases}$

Step two: Reducing dimensionality while preserving relationships

$$\widehat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^m \left(\mathbf{z}^{(i)} - \sum_{j=1}^m \widehat{w}_{i,j} \mathbf{z}^{(j)} \right)^2$$

$\mathbf{x}^{(1)} = [1, 0, 0]$, $\mathbf{x}^{(2)} = [0, 1, 0]$, $\mathbf{x}^{(3)} = [0, 0, 1]$
 $\mathbf{z}^{(1)} = [1, 0, 0]$, $\mathbf{z}^{(2)} = [0, 1, 0]$, $\mathbf{z}^{(3)} = [0, 0, 1]$
 $\widehat{\mathbf{z}} = [1, 1, 1]$, $\widehat{w}_{1,1} = 1, \widehat{w}_{1,2} = 2, \widehat{w}_{1,3} = 3$
 $\widehat{w}_{2,1} = 1, \widehat{w}_{2,2} = 2, \widehat{w}_{2,3} = 3$
 $\widehat{w}_{3,1} = 1, \widehat{w}_{3,2} = 2, \widehat{w}_{3,3} = 3$

其他方法

1) Isomap (等距映射)

Isomap = MDS(Multidimensional Scaling)多维尺度变换 + 测地线距离

MDS: MDS是理论上保持欧式距离的一种经典方法，MDS最早用来做数据的可视化，MDS得到的低维表示中心在原地，所以又可以说是保持P小，MDS降维后的任意两点的距离（内积）与高维空间的距离相等或近似相等。

因此MDS在流形数据处理上，保持欧式距离不变的理论不可行（失效），Isomap就是改进的MDS方法，在流形非线性降维领域的应用方法。

Isomap的理论框架为MDS，放在流形的理论框架中，原始的高维空间欧氏距离换成了流形上的测地线距离。Isomap是把任意两点的测地距离(即是最短距离)作为流形的几何描述，用MDS理论框架保持点与点之间的最短距离。

测地线的计算：在流形结构未知，数据采样有限的情况下，通过构造数据点间的邻接图(Graph)，用图上的最短距离来近似测地线距离，当数据无穷多时，这个估计近似距离趋向于真实的测地线距离。

此算法是计算机系中常用的图论经典算法。

somap 与LLE的比较：

Isomap与LLE从不同的角度出发来实现同一个目标，它们都能从某种程度上发现并在映射的过程中保持流形的几何特征。

Isomap希望保持任意两点间的测地线距离；LLE希望保持局部线性关系。

从保持几何角度来看，Isomap保留了更多信息，然而Isomap方法的一个问题是需要考虑任意两点之间的关系，这个数量随着数据点的增多而呈指数增长，从而增加计算负荷，在大数据时代，使用全局方法分析巨型数据结构正在变得越来越困难。

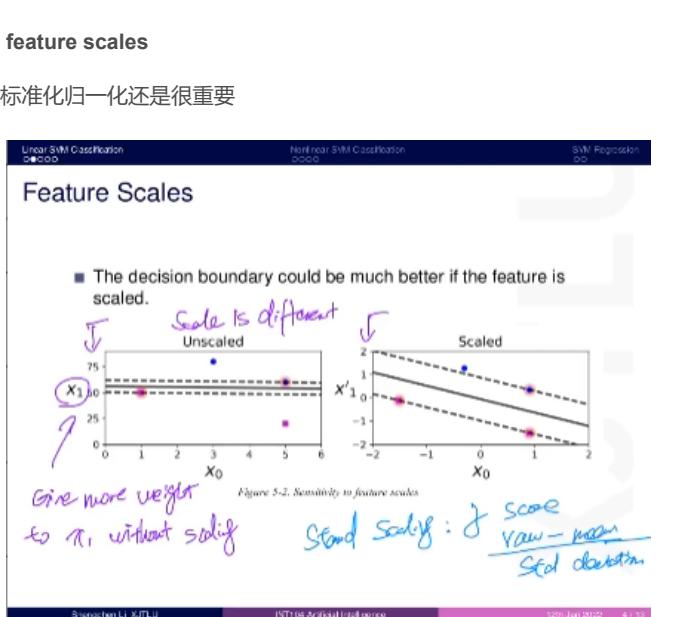
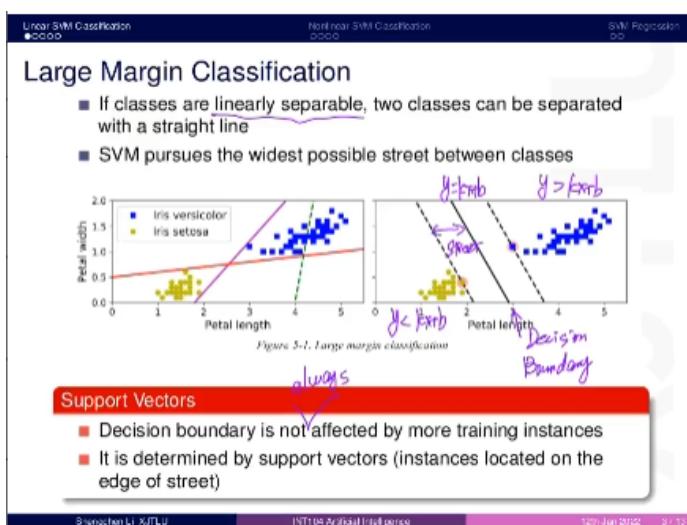
因此，以LLE为开端的局部分析方法的变种和相关理论正在受到越来越多的关注。

2) t-Distributed Stochastic Neighbor Embedding (t-SNE)

Trying to keep similar instances close and dissimilar instances apart.

lec6 Support Vector Machine (SVM)

通俗来讲，所谓支持向量机是一种分类器，对于做出标记的两组向量，给出一个最优分割超曲面把这两组向量分割到两边，使得两组向量中离最近的向量（即所谓支持向量，在街道的两边）到此超平面的距离都尽可能远。



6.1 Hard Margin Classification 硬间隔支持向量机 (线性可分SVM)

假设条件：

硬间隔SVM要求数据集是严格线性可分的。

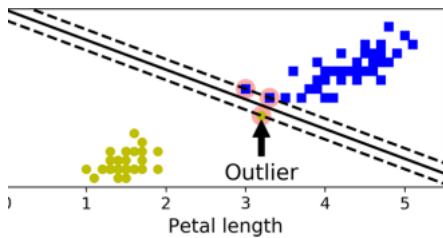
线性可分数据集的定义是：对于以上给定的数据集T，存在一个超平面能够将数据集中的正类点和负类点完全正确地划分到超平面的两侧

All instances being off the street and on the right side is named “hard margin classification”

The main limitation of hard margin classification is

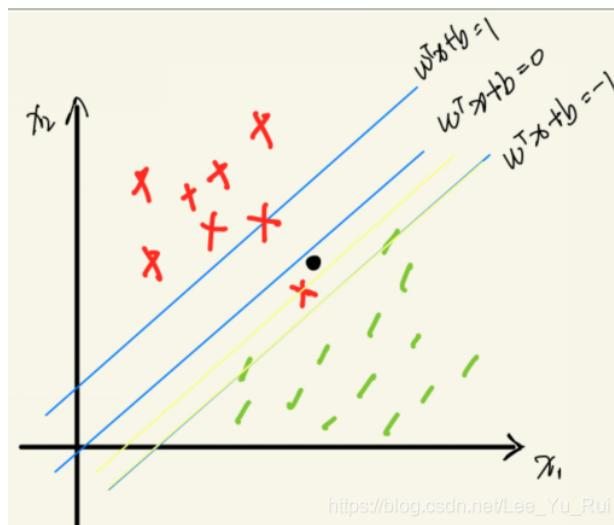
*The data must be linearly separable

*There are little outliers



6.2 Soft Margin Classification

数据是近似线性可分的，采用soft margin 就是说数据在一定的错误容忍度上，可以用一条直线将以区分；如下图，如果按照第一种情况，会产生黄色的边界，因为受到一个红色异常点的影响，这两个边界的间隔很小，如果目前有一个黑色的待分类的点，按照黄色线的分类器，就会划分为色，但实际上它可能是绿色，因为和绿色的聚集中心是比较近的。所以为了减少异常点的影响，我们允许分类器有一点错分的情况，这就是soft margin



Soft Margin Classification provides more flexibility

The algorithm balances *The width of street

*The amount of margin violations

- A hyper-parameter C is defined

- A low value of C leads to more margin violation
- A high value of C limits the flexibility

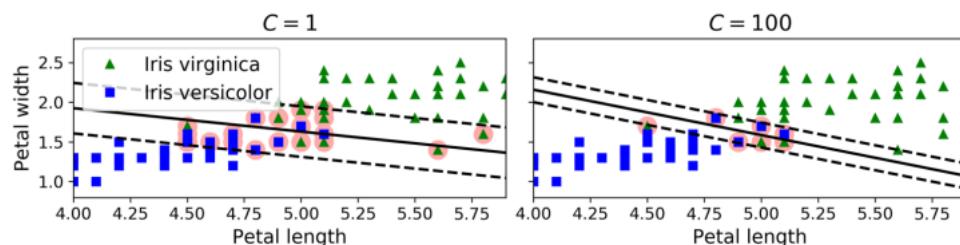


Figure 5-4. Large margin (left) versus fewer margin violations (right)

system?

- How can we find the best value of C ?
 - Empirical Study**
 - add noise**
 - add outliers**
 - Cross-Validation**

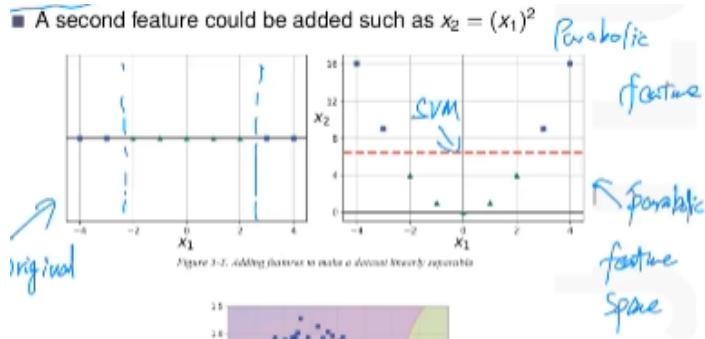
Discussion 2

What if the sample is not linearly separable?

做特征提取那些线性可分的

Extract features that is linearly separable

6.3 Polynomial Feature



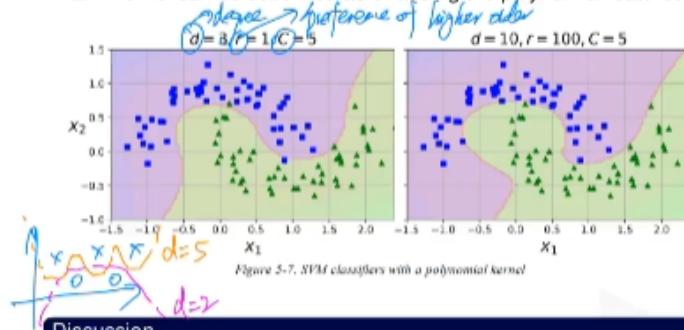
加了一个 x_2 使得它线性可分

6.4 Polynomial Kernel 多项式核函数

度数太大可能导致过拟合，使得在测试集上不好的表现

Polynomial Kernel

- A kernel can be used instead of adding the polynomial features



Will the order of polynomial function introduce overfitting problem?

Degrade system performance for testing dataset.

6.5 RBF (Radial Basis Function Kernel) 高斯核

多项式核函数而言，它的核心思想是将样本数据进行升维，从而使得原本线性不可分的数据线性可分。那么高斯核函数的核心思想是将每一个样本映射到一个无穷维的特征空间，从而使得原本线性不可分的数据线性可分。

$$\Phi_\gamma(\mathbf{x}, \mathbf{l}) = e^{-\gamma \|\mathbf{x} - \mathbf{l}\|^2}$$

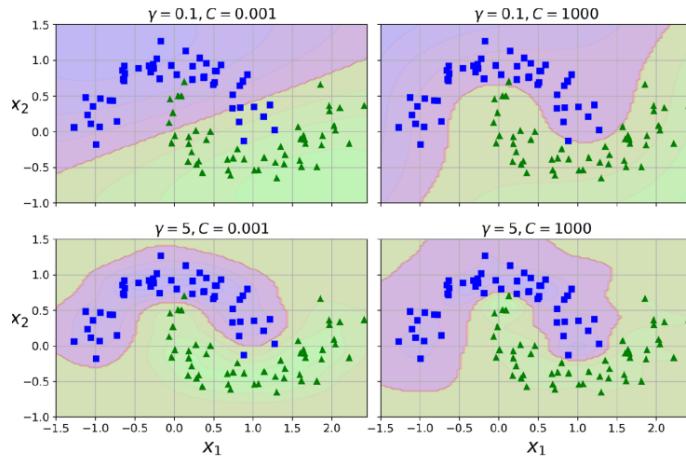


Figure 5-9. SVM classifiers using an RBF kernel

所以我们得出结论，当 γ 取值比较大时，数据训练结果趋于过拟合，当 γ 取值比较小时，数据训练结果趋于欠拟合。

2、高斯核。适用于没有先验经验的非线性分类。其中 σ 越小，使得映射的维度越高。

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

3、多项式核。适用于没有先验经验的分类。其中 d 越越小，使得映射的维度越高。

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{a} \mathbf{x}_i^\top \mathbf{x}_j + c)^d$$

6.6 SVM Regression

SVM解决回归问题的思路和解决分类问题的思路正好是相反的。我们回忆一下，在Hard Margin SVM中，我们希望在Margin区域中一个样本点都有，即便在Soft Margin SVM中也是希望Margin区域中的样本点越少越好。

而在SVM解决回归问题时，是希望Margin区域中的点越多越好。也就是找到一条拟合直线，使得这条直线的Margin区域中的样本点越多，说明越好，反之依然。

- Fit as many instances as possible on the street
- Limit margin violations
- Width of street is controlled by a hyper-parameter ϵ

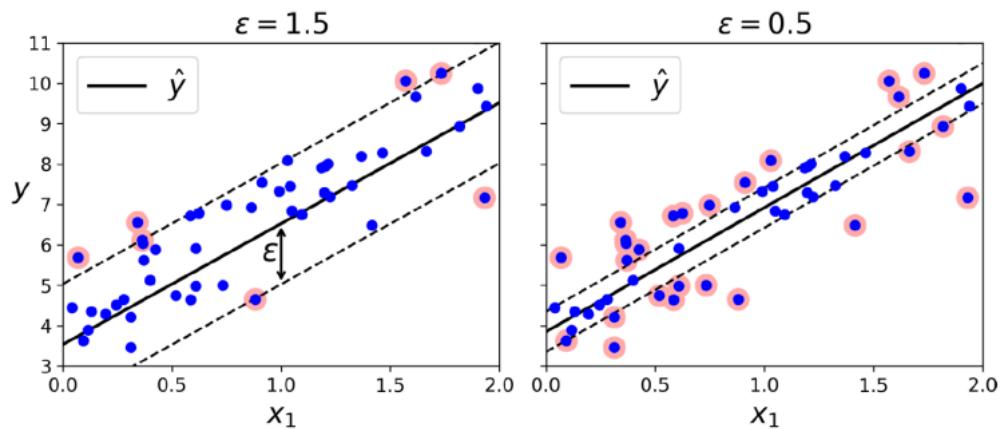


Figure 5-10. SVM Regression

- Kernel tricks could also be used

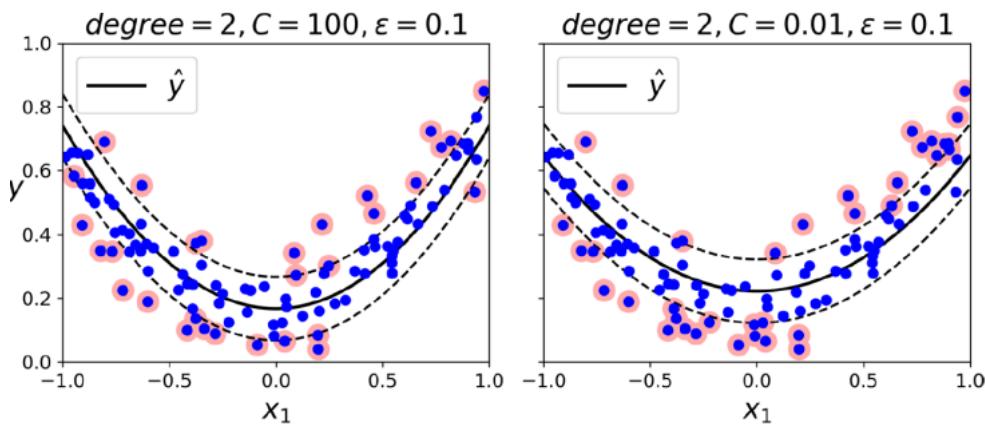


Figure 5-11. SVM Regression using a second-degree polynomial kernel

lec7 Naïve Bayes

朴素贝叶斯 (Naïve Bayes) 属于监督学习的生成模型，实现简单，没有迭代，学习效率高，在大样本量下会有较好的表现。但因为假设太强—特征条件独立，在输入向量的特征条件有关联的场景下并不适用。

7.1 Bayes' Rule 贝叶斯法则

$$\begin{aligned} P(a | b) P(b) &= P(a, b) = P(b | a) P(a) \\ P(a | b) &= P(b | a) P(a) / P(b) \end{aligned}$$

The famous Bayes' Rule states the relationship between prior probability distribution and posterior probability distribution

$$P(c|\mathbf{x}) = \frac{P(c)P(\mathbf{x}|c)}{P(\mathbf{x})} \quad (1)$$

where c is considered as a class, \mathbf{x} is considered as a set of samples

- $P(c)$ is named as prior probability
- $P(\mathbf{x}|c)$ is named as class-conditional probability (CCP, also known as "likelihood")
- $P(c|\mathbf{x})$ is named as posterior probability
- $P(\mathbf{x})$ is considered as evidence factor (observation)

Bayes' Rule can be used for various purposes such as parameter estimation, classification and model selection.

条件概率是指事件A在事件B发生的条件下发生的概率。条件概率表示为： $P(A|B)$ ，读作“ A 在 B 发生的条件下发生的概率”。若只有两个事件A, B, 那么, $P(A|B) = \frac{P(AB)}{P(B)}$ 。

Bayes定理可表述为：

后验概率 = (相似度*先验概率)/标准化常量

也就是说，后验概率与先验概率和相似度的乘积成正比。

另外，比例 $P(B|A)/P(B)$ 也有时被称作标准相似度 (standardised likelihood)，Bayes定理可表述为：

后验概率 = 标准相似度*先验概率

7.2 Naïve Bayes Classifier

分类方法

首先，我们定义训练集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，其类别 $y_i \in \{c_1, c_2, \dots, c_K\}$ ，则训练集中样本点数为 N ，类别数为 K 。输入待预测数据 x ，则预测类别

$$\arg \max_{c_k} p(y = c_k | x)$$

由贝叶斯定理可知：

$$p(y = c_k | x) = \frac{p(x|y = c_k)p(y = c_k)}{p(x)}$$

对于类别 c_k 而言， $p(x)$ 是恒等的，因此式子(???)等价于

$$\arg \max_{c_k} p(x|y = c_k)p(y = c_k)$$

从上面式子可以看出：朴素贝叶斯将分类问题转化成了求条件概率与先验概率的最大乘积问题。先验概率 $p(y = c_k)$ 可通过计算类别的频率得到，但如何计算条件概率 $p(x|y = c_k)$ 呢？

朴素贝叶斯对条件概率做了条件独立性的假设，即特征条件相互独立。设输入 x 为 n 维特征向量 $(x^{(1)}, x^{(2)}, \dots, x^{(j)}, \dots, x^{(n)})$ ，第 j 维特征 $x^{(j)}$ 的取值有 S_j 个。由概率论的知识可知：

$$p(x|y = c_k) = \prod_j p(x^{(j)}|y = c_k)$$

Example

Naïve Bayes

Outlook	Temprature	Humidity	Windy	Play
Overcast	Hot	High	False	Yes
Overcast	Cool	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Rainy	Mild	Normal	False	Yes
Rainy	Mild	High	True	No
Sunny	Hot	High	True	No
Sunny	Hot	High	False	No
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Sunny	Mild	Normal	True	Yes

Will you play on the day of Mild?

Solution

Naïve Bayes

$$\text{Posterior } P(C|X) = \frac{P(X|C) P(C)}{P(X)}$$

Temp.	Yes	No	p
Hot	2	2	0.28
Mild	4	2	0.43
Cool	3	1	0.28
p	0.64	0.36	

$$P(X|C) = \frac{P(X \cap C)}{P(C)}$$

By this table we have $p(\text{Mild|Yes}) = \frac{4}{9} = 0.44$ and $p(\text{Mild|No}) = \frac{2}{5} = 0.4$

$$\text{Posterior } p(\text{Yes|Mild}) = \frac{p(\text{Mild|Yes})p(\text{Yes})}{p(\text{Mild})} = \frac{0.44 \times 0.64}{0.43} = 0.65$$

$$\text{Posterior } p(\text{No|Mild}) = \frac{p(\text{Mild|No})p(\text{No})}{p(\text{Mild})} = \frac{0.4 \times 0.36}{0.43} = 0.33$$

As $p(\text{Yes|Mild}) > p(\text{No|Mild})$, it is likely to play.

Will the following condition be considered as a proper day for play?

■ Sunny, Windy

■ Overcast, Normal Humidity & Cool

Outlook	Yes	No	p
Overcast	4	0	0.28
Rainy	3	2	0.36
Sunny	2	3	0.36
p	0.64	0.36	

Wind	Yes	No	p
True	3	3	0.43
False	6	2	0.57
p	0.64	0.36	

Humid	Yes	No	p
Normal	6	1	0.5
High	3	4	0.5
p	0.64	0.36	

Sunny, Windy

$$p(\text{Yes}|\text{Sunny, Windy}) = \frac{p(\text{Sunny, Windy}|\text{Yes})p(\text{Yes})}{p(\text{Sunny, Windy})} \propto$$
$$p(\text{Sunny, Windy}|\text{Yes})p(\text{Yes}) = p(\text{Sunny}|\text{Yes})p(\text{Windy}|\text{Yes})p(\text{Yes}) =$$
$$0.22 \times 0.33 \times 0.64 = 0.05$$

$$p(\text{No}|\text{Sunny, Windy}) = \frac{p(\text{Sunny, Windy}|\text{No})p(\text{No})}{p(\text{Sunny, Windy})} \propto$$
$$p(\text{Sunny, Windy}|\text{No})p(\text{No}) = p(\text{Sunny}|\text{No})p(\text{Windy}|\text{No})p(\text{No}) =$$
$$0.6 \times 0.6 \times 0.36 = 0.13$$

As $p(\text{Yes}|\text{Sunny, Windy}) < p(\text{No}|\text{Sunny, Windy})$, so the combination of weather is unlikely to be suitable for playing

Rainy, Normal Humidity & Cool

$$p(\text{Yes}|\text{Rainy, Normal, Cool}) = \frac{p(\text{Rainy, Normal, Cool}|\text{Yes})p(\text{Yes})}{p(\text{Rainy, Normal, Cool})} \propto$$
$$p(\text{Rainy, Normal, Cool}|\text{Yes})p(\text{Yes}) =$$
$$p(\text{Rainy}|\text{Yes})p(\text{Normal}|\text{Yes})p(\text{Cool}|\text{Yes})p(\text{Yes}) = 0.33 \times 0.67 \times 0.33 \times 0.64 = 0.047$$

$$p(\text{No}|\text{Rainy, Normal, Cool}) = \frac{p(\text{Rainy, Normal, Cool}|\text{No})p(\text{No})}{p(\text{Rainy, Normal, Cool})} \propto$$
$$p(\text{Rainy, Normal, Cool}|\text{No})p(\text{No}) =$$
$$p(\text{Rainy}|\text{No})p(\text{Normal}|\text{No})p(\text{Cool}|\text{No})p(\text{No}) = 0.4 \times 0.2 \times 0.2 \times 0.36 = 0.0058$$

As $p(\text{Yes}|\text{Rainy, Normal, Cool}) > p(\text{No}|\text{Rainy, Normal, Cool})$, it is likely to play

lec8 Non-parametric Classification

8.1

1、参数模型 (*parametric models*)

在机器学习中，有一组训练数据 $X(x_1, x_2, \dots, x_n)$, $Y(y_1, y_2, \dots, y_n)$ ，通常都会先提出一个假设 H ，然后通过训练这个假设让 H 不断接近数据的真实的函数 f (也叫映射函数)。

注意这个真实的函数是未知的，我们要做的只是不断逼近真实的函数。

还有假设 H 其实就是一个方程，这个是人为定义的。比如根据数据的分布趋势，选取了线性回归 $y = ax + b$ ，则假设函数 H 便是 $y = ax + b$ 。这个假设中除了 x, y 是已知的， a, b 均是未知的且 a, b 是假设 H 方程的参数，则将训练数据去拟合该假设即可得到 a, b 。

正常来说，参数模型的模型的参数都是固定的，就如上面的线性回归，参数固定为 2 个；对于高斯函数，函数形式是固定的，参数 μ, σ 也是固定的。

总的来说，若要使用参数模型进行预测，则在用参数模型前就已经知道模型有哪些参数。

定义：

In statistics, a parametric model or parametric family or finite dimensional model is a particular class of statistical models. Specifically, a parametric model is a family of probability distributions that has a finite number of parameters.(Wikipedia)

在统计学中，参数模型通常假设总体（随机变量）服从某一个分布，该分布的一些参数确定（比如正太分布由均值和方差确定），在此基础上构建的模型称为参数模型。

优点：

- 简单，这些方法易于理解和解释结果。
- 学习快，参数模型能快速从数据中学习。
- 适合数据集少，参数模型不需要很多训练数据仍可以对数据进行拟合尽管效果不佳。

缺点：

- 函数约束，通过选择函数形式，这些方法受到指定形式的高度约束（参数模型可以用数学表达式表示出来）。
- 复杂度低，这些方法更适合于简单的问题。
- 拟合效果较差，实际上，这些方法不太可能匹配底层映射函数

2、非参数模型 (*nonparametric models*)

在机器学习中非参数模型不用提出假设函数 H ，而是直接对映射函数 f 学习。通过不做假设，他们可以自由地从训练数据中学习任何函数形式。

如K近邻算法，该算法通过计算训练样本之间的距离来对未知样本分类，在计算过程中只用到训练样本本身的数据，并没有需要拟合才能得到的其他参数。假设已知训练样本是 $x_n(x_n^1, \dots, x_n^d)$ 是 d 维数据，未知样本也是 d 维数据 $x_m(x_m^1, \dots, x_m^d)$ ，则两者距离为：

$$distance(x_n, x_m) = \sum_i^d (|x_n^i - x_m^i|^p)^{1/p}$$

上式中的 p 根据采取的距离度量方式自主选取，不需要通过模型训练获得。该方法对映射函数的形式不做任何假设，除了相近的模式可能具有类似的输出变量之外。

在构造映射函数时，非参数方法寻求对训练数据的最佳拟合，同时保持对未知数据的泛化能力。因此，它们能够适应大量的函数形式。

非参数模型对于总体的分布不做任何假设，只是知道总体是一个随机变量，其分布是存在的（分布中也可能存在参数），但是无法知道其分布的形式，更不知道分布的相关参数，只有在给定一些样本的条件下，能够依据非参数统计的方法进行推断。

注意非参数模型不是说没有参数，只是模型的参数是不固定的且事先没有定义或者这些参数对于我们来说是不可见（不可知）的。

优点：

- 灵活，能够适应大量的函数形式（非参数模型不可以用数学表达式表示出来）。
- 对于底层函数没有假设（或弱假设）
- 可以产生更高性能的预测模型。

缺点：

- 需要更多的训练数据来估计映射函数。
- 训练要慢得多，因为他们有更多的参数需要训练
- 训练数据过拟合风险更大，当数据维数较高时，很难对某些特定预测结果做出解释。

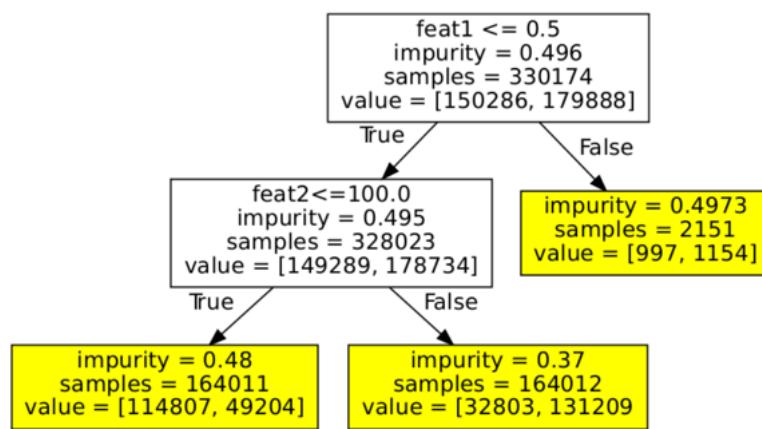
Parametric	Non-parametric	Application
polynomial regression	Gaussian processes	function approx.
logistic regression	gaussian precess classifier	classification
mixture models.k-means	dirichlet process mixtures	clustering
hidden markov models	infinite HMMs	time series
factor analysis/PCA/PMF	infiniter latent factor models	feature discovery
Perceptron	k-Nearest Neighbors	classification
Naive Bayes	Decision Trees like CART and C4.5	回归、分类
Simple Neural Networks	Support Vector Machines	回归、分类

Non-parametric models can be used with arbitrary distributions and without the **assumption that the forms of the underlying densities** are known.

Moreover, they can be used with **multimodal distributions** which are much more common in practice than unimodal distributions.

Here we introduce two major methods: **decision tree** and **k-nearest neighbour**

8.2 decision tree



CSDN @白小斗

每个方框代表一个节点。每个**非叶子节点** **节点 (node)** 有2个分支，一个是判定True，一个判定False。分别走两个不同的分支。叶子节点表示决策任何一个输入从root出发，总是会达到且唯一到达一个叶子节点。这就是决策树的工作原理。

决策树有两种节点：**中间节点**和**叶子节点**。

1. 每个中间节点有4个参数：

a) **判定函数**。是一个特征的取值。当特征小于等于这个值得时候决策路径走左边，当特征大于这个值得时候决策树走右边。

- b) 不纯度值(impurity value). 是当前节点的不纯度值. 关于不纯度值得意义后面会讲到. 他反应了当前节点的预测能力.
- c) 覆盖样本个数(n_samples). 是指参与此节点决策的样本个数. 父亲节点S和两个孩子节点(l,r)的样本个数的关系为: $n_samples^S = n_samples^l + n_samples^r$. 覆盖样本个数越多, 说明判定函数越稳定. 实际上很容易看出来, 所有的叶子节点所对应的样本是总样本的一个划分.
- d) 节点取值(node value). 节点取值是一个数组. 数组的长度为类目个数. $value = [997, 1154]$ 表示在2151个样本数中, 有997个属于class1, 1154个属于class2. (这是分类树的意义, 会归数的取值规则后面会讲.)
2. 每个叶子节点有3个参数. 除了没有决策函数之外, 其他参数意义一样.

决策树学习的关键在于如何选择**最优的划分属性**, 所谓的最优划分属性, 对于二元分类而言, 就是尽量使划分的样本属于同一类别, 即“**纯度**”最性.

*从构建决策树来理解决策树 (例子中使用的measure the impurity of node的方法是课上的信息熵(info entropy)基尼指数(Gini Index)中的entrc过熵算出信息增益选择纯度最高的特征)

编号	特征 (属性)						好瓜 Label
	色泽	根蒂	敲声	纹理	脐部	触感	
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

显然该数据集包含17个样本, 类别为二元的, 即 $K = 2$. 则, 正例 (类别为1的样本) 占的比例为: $p_1 = \frac{8}{17}$,
反例 (类别为0的样本) 占的比例为: $p_2 = \frac{9}{17}$. 根据信息熵的公式能够计算出数据集D的信息熵为:

$$Ent(D) = - \sum_{k=1}^2 p_k \log_2 p_k = -\left(\frac{8}{17} \log_2 \frac{8}{17} + \frac{9}{17} \log_2 \frac{9}{17}\right) = 0.998$$

从数据集中能够看出特征集为: {色泽、根蒂、敲声、纹理、脐部、触感}. 下面我们来计算每个特征的信息增益. 先看“色泽”, 它有三个可能的离值: {青绿、乌黑、浅白}, 若使用“色泽”对数据集D进行划分, 则可得到3个子集, 分别为: D^1 (色泽=青绿)、 D^2 (色泽=乌黑)、 D^3 (色泽=浅白). D^1 共包含6个样本{1, 4, 6, 10, 13, 17}, 其中正例占 $p_1 = \frac{3}{6}$, 反例占 $p_2 = \frac{3}{6}$. D^2 包含6个样本{2, 3, 7, 8, 9, 15}, 其中正例占 $p_1 = \frac{4}{6}$, 反例占 $p_2 = \frac{2}{6}$. D^3 包含了5个样本{5, 11, 12, 14, 16}, 其中正例占 $p_1 = \frac{1}{5}$, 反例占 $p_2 = \frac{4}{5}$. 因此, 可以计算出用“色泽”划分之后所获得的3个分支结点的信息熵为:

$$Ent(D^1) = -\left(\frac{3}{6}log_2\frac{3}{6} + \frac{3}{6}log_2\frac{3}{6}\right) = 1.000$$

$$Ent(D^2) = -\left(\frac{4}{6}log_2\frac{4}{6} + \frac{2}{6}log_2\frac{2}{6}\right) = 0.918$$

$$Ent(D^3) = -\left(\frac{1}{5}log_2\frac{1}{5} + \frac{4}{5}log_2\frac{4}{5}\right) = 0.722$$

因此，特征“色泽”的信息增益为：

$$\begin{aligned} Gain(D, \text{色泽}) &= Ent(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} Ent(D^v) \\ &= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722\right) \\ &= 0.109 \end{aligned}$$

同理可以计算出其他特征的信息增益：

$$\begin{aligned} Gain(D, \text{根蒂}) &= 0.143; \quad Gain(D, \text{敲声}) = 0.141; \\ Gain(D, \text{纹理}) &= 0.381; \quad Gain(D, \text{脐部}) = 0.289; \\ Gain(D, \text{触感}) &= 0.006. \end{aligned}$$

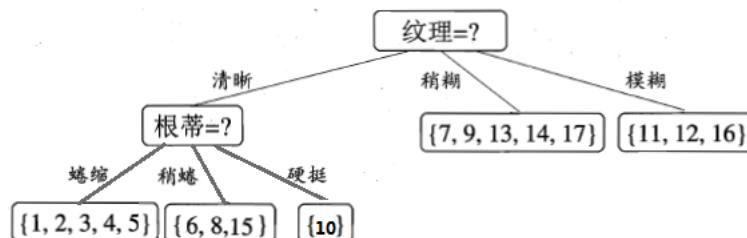
比较发现，特征“纹理”的信息增益最大，于是它被选为划分属性。因此可得：



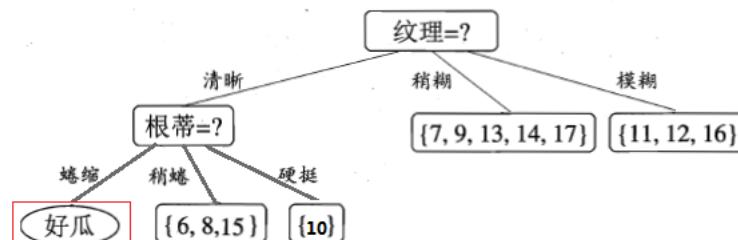
第二步、继续对上图中每个分支进行划分，以上图中第一个分支结点(“纹理=清晰”)为例，对这个结点进行划分，设该结点的样本集 D^1 {1, 2, 3, 4, 5, 6, 8, 10, 15}，共9个样本，可用特征集合为{色泽，根蒂，敲声，脐部，触感}，因此基于 D^1 能够计算出各个特征的信息增益：

$$\begin{aligned} Gain(D^1, \text{色泽}) &= 0.043; \quad Gain(D^1, \text{根蒂}) = 0.458; \\ Gain(D^1, \text{敲声}) &= 0.331; \quad Gain(D^1, \text{脐部}) = 0.458; \\ Gain(D^1, \text{触感}) &= 0.458. \end{aligned}$$

比较发现，“根蒂”、“脐部”、“触感”这3个属性均取得了最大的信息增益，可以随机选择其中之一作为划分属性（不妨选择“根蒂”）。因此可得：



第三步，继续对上图中的每个分支结点递归的进行划分，以上图中的结点(“根蒂=蜡缩”)为例，设该结点的样本集为{1, 2, 3, 4, 5}，共5个样本，但这5个样本的class label均为“好瓜”，因此当前结点包含的样本全部属于同一类别无需划分，将当前结点标记为C类（在这个例子中为“好瓜”）叶节点，递归返回。因此上图变为：



第四步，接下来对上图中结点(“根蒂=稍蜡”)进行划分，该点的样本集为 D^1 {6, 8, 15}，共有三个样本。可用特征集为{色泽，敲声，脐部，触感}，同样可以基于 D^1 计算出各个特征的信息增益，计算过程如下（写的比较详细，方便大家弄懂）：

首先计算 $\text{Ent}(D^1)$, D^1 中正样本比例为: $p_1 = \frac{2}{3}$, 负样本比例为: $p_2 = \frac{1}{3}$, 所以:

$$\text{Ent}(D^1) = -\sum_{k=1}^2 p_k \log_2 p_k = -\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) = 0.918$$

因为可用特征集为{色泽, 敲声, 脖部, 触感}, 接下来一个一个计算:

色泽:

若用色泽对 D^1 进行划分, 则可得到三个子集 (实际上是两个子集):

$$D^{11}(\text{色泽} = \text{青绿}) = \{6\}, \quad p_1 = \frac{1}{1}, \quad p_2 = \frac{0}{1};$$

$$D^{12}(\text{色泽} = \text{乌黑}) = \{8, 15\}, \quad p_1 = \frac{1}{2}, \quad p_2 = \frac{1}{2};$$

$$D^{13}(\text{色泽} = \text{浅白}) = \emptyset, \quad p_1 = 0, \quad p_2 = 0;$$

所以:

$$\text{Ent}(D^{11}) = -\left(\frac{1}{1} \log_2 1 + 0 * \log_2 0\right) = 0$$

$$\text{Ent}(D^{12}) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1$$

$$\text{Ent}(D^{13}) = 0$$

$$\text{则, } \text{Gain}(D^1, \text{ 色泽}) = 0.918 - \left(\frac{1}{3} * 0 + \frac{2}{3} * 1 + 0\right) = 0.251$$

敲声:

若用色泽对 D^1 进行划分, 则只有一个子集:

$$D^{11}(\text{敲声} = \text{清晰}) = \{6, 8, 15\}, \quad p_1 = \frac{2}{3}, \quad p_2 = \frac{1}{3};$$

$$\text{则 } \text{Ent}(D^{11}) = -\left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3}\right) = 0.918$$

则,

$$\text{Gain}(D^1, \text{ 敲声}) = 0.918 - \left(\frac{3}{3} * 0.918\right) = 0$$

注: 从信息增益的定义不难看出为什么 $\text{Gain}(D^1, \text{ 敲声}) = 0$, 信息增益的定义为: 得到属性 X 的信息使得类 Y 的信息的不确定性减少的程度 (摘自, 李航《统计学习方法》)。显而易见“敲声”无法对 D^1 进行任何划分, 即样本集 D^1 从属性“敲声”无法获得任何信息, 信息增益为 0。

脖部:

脖部和敲声一样都只能划分出一个子集, 因此 $\text{Gain}(D^1, \text{ 脖部}) = 0$

触感:

若用“触感”对 D^1 进行划分，则有两个子集：

$$D^{11}(\text{触感} = \text{硬滑}) = \{8\}, p_1 = \frac{1}{1}, p_2 = \frac{0}{1};$$

$$D^{12}(\text{触感} = \text{软粘}) = \{6, 15\}, p_1 = \frac{1}{2}, p_2 = \frac{1}{2};$$

所以：

$$\text{Ent}(D^{11}) = -\left(\frac{1}{1} \log_2 1 + 0 * \log_2 0\right) = 0$$

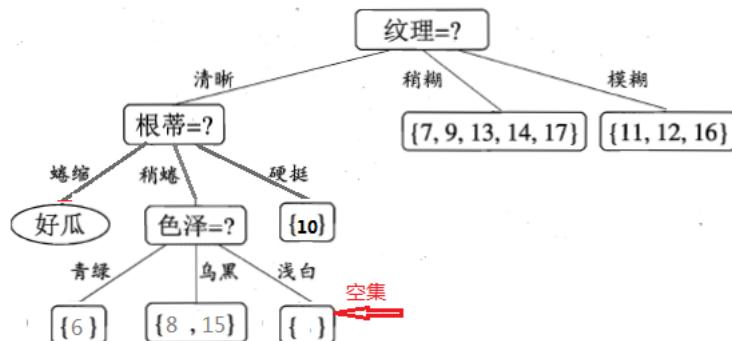
$$\text{Ent}(D^{12}) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1$$

$$\text{则, Gain}(D^1, \text{触感}) = 0.918 - \left(\frac{1}{3} * 0 + \frac{2}{3} * 1 + 0\right) = 0.251$$

比较 $\text{Gain}(D^1, \text{色泽}) = 0.251$, $\text{Gain}(D^1, \text{敲声}) = 0$, $\text{Gain}(D^1, \text{脐部}) = 0$, $\text{Gain}(D^1, \text{触感}) = 0.251$, 可知“色泽”, “触感”2个属性均取得了最大的信息增益，选一个属性作为划分属性。

(注：该图片版权所有，转载或使用请注明作者（天泽28）和出处)

不妨选择“色泽”属性作为划分属性，则可得到：



继续递归的进行，看“色泽=青绿”这个节点，只包含一个样本，无需再划分了，直接把当前结点标记为叶节点，类别为当前样本的类别，即好瓜。递归返回。然后对递归的对“色泽=乌黑”这个节点进行划分，就不再累述了。说下“色泽=浅白”这个节点，等到递归的深度处理完“色泽=乌黑”分支后，返回来处理“色泽=浅白”这个节点，因为当前结点

包含的样本集为空集，不能划分，对应的处理措施为：将其设置为叶节点，类别为设置为其父节点（根蒂=稍蜷）所含样本最多的类别，“根蒂=稍蜷”包含{6, 8, 15}三个样本，6, 8为正样本，15为负样本，因此“色泽=浅白”结点的类别为正（好瓜）。最终，得到的决策树如下图所示：



$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2 \quad (1)$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i th node

Question

A node applies to 0 Iris setosa, 49 Iris versicolor and 5 Iris virginica

$$\begin{aligned} Gini\ Index &= 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 \\ &= 0.168 \end{aligned}$$

GINI反应了从数据集D中随机抽取两个样本，其类别标记不一致的概率，因此， $Gini(D)$ 越小，则数据集D的纯度越高。所以在候选属性集合中选择那个使得划分后基尼指数最小的属性作为最优划分属性。

Entropy 信息熵

决策树学习的关键在于如何选择最优的划分属性，所谓的最优划分属性，对于二元分类而言，就是尽量使划分的样本属于同一类别，即“纯度”最高的属性。那么如何来度量特征（features）的纯度，这时候就要用到“信息熵（information entropy）”。先来看看信息熵的定义：假如当前样本集D中第k类样本所占的比例为 $p_k(k = 1, 2, 3, \dots |K|)$ ， K 为类别的总数（对于二元分类来说， $K = 2$ ）。则样本集的信息熵为：

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

$Ent(D)$ 的值越小，则D的纯度越高。（这个公式也决定了信息增益的一个缺点：即信息增益对可取值数目多的特征有偏好（即该属性能取得值越多，信息增益，越偏向这个），因为特征可取的值越多，会导致“纯度”越大，即 $ent(D)$ 会很小，如果一个特征的离散个数与样本数相等，那么 $ent(D)$ 值会为0）。再来看一个概念信息增益（information gain），假定离散属性 a 有 V 个可能的取值 $\{a^1, a^2, \dots, a^V\}$ ，如果使用特征 a 来对数据集D进行划分，则会产生 V 个分支结点，其中第 v （小v）个结点包含了数据集D中所有在特征 a 上取值为 a^v 的样本总数，记为 D^v 。因此可以根据上面信息熵的公式计算出信息熵，再考虑到不同的分支结点所包含的样本数量不同，给分支节点 $|D^v|$ 赋予权重 $\frac{|D^v|}{|D|}$ ，即样本数越多的分支节点的影响越大，因此，能够计算出特征 a 对样本集D进行划分所获得的“信息增益”：

$$Gain(D, a) = Ent(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} Ent(D^v)$$

一般而言，信息增益越大，则表示使用特征 a 对数据集划分所获得的“纯度提升”越大。所以信息增益可以用于决策树划分属性的选择，其实就是选择信息增益最大的属性，ID3算法就是采用的信息增益来划分属性。

Impurity

↑ Entropy is also one of the possible way to evaluate impurity

↑ Uncertainty

$$H_i = - \sum_{k=1}^n p_{i,k} \log_2(p_{i,k}) \quad (2)$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i th node ($p_{i,k}$ cannot be 0)

Question

A node applies to 0 Iris setosa, 49 Iris versicolor and 5 Iris virginica

$$-\frac{49}{54} \log_2 \left(\frac{49}{54} \right) - \frac{5}{54} \log_2 \left(\frac{5}{54} \right) = 0.445$$

Experiment

Design an experiment to compare the impurity measured by entropy and Gini index

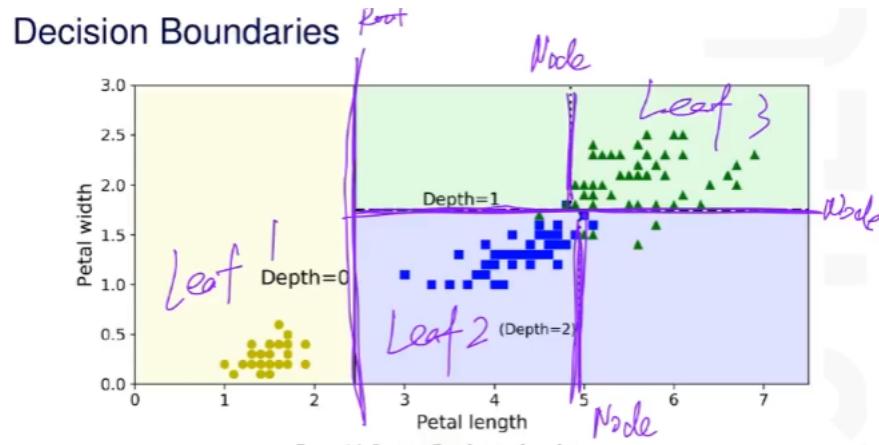


Figure 6-2. Decision Tree decision boundaries

White Box

Decision is a classical white box method. What knowledge can we get from the decision tree?

Estimating Class Probabilities

Question

A node applies to 0 Iris setosa, 49 Iris versicolor and 5 Iris virginica. How do you calculate $p(\text{Setosa})$, $p(\text{Versicolor})$, $p(\text{Virginica})$?

$$p(\text{Setosa}) = \frac{0}{0+49+5} = 0$$

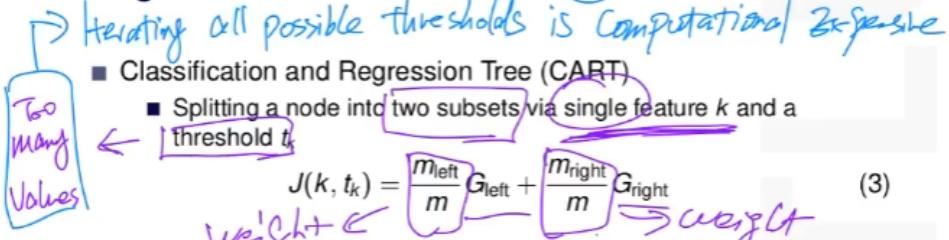
$$p(\text{Versicolor}) = \frac{49}{49+5}$$

8.3 CART, 又名分类回归树

CART算法的几个特点：

- (1) 由任务目标决定CART是分类树还是回归树。当CART是分类树时，采用基尼系数(gini)作为特征划分的度量；当CART是回归树时，采用误差(mse)作为特征划分的度量。
- (2) CART算法不仅可以处理二分属性，可以用来处理标称属性和连续属性(回归)，简单来说，就是如果特征值大于或等于给定值就将该记录到左子树，否则就划分到右子树。
- (3) 决策树生成只考虑了对训练数据进行更好的拟合，而决策树剪枝考虑了减小模型复杂度。决策树生成学习局部的模型，决策树剪枝学习整体模型。一般选择损失函数最小作为剪枝的标准。

Growing a tree



- $G_{\text{left/right}}$ measures the impurity of the left/right subset
- $m_{\text{left/right}}$ is the number of instances in the left/right subset

Discussion

- When the tree should stop grow?
- What problem is introduced if the tree grow until perfect impurity?
Oversplitting

一) 终止条件有：

- 特征已经用完了：没有可供使用的特征再进行分裂了，则树停止分裂；
 - 子结点中的样本已经都是同一类：此时，样本已经全部被划分出来了，不用再进行区分，该结点停止分裂（不过一般很难达到，达到的话，定过拟合）；
 - 子节点中没有样本了：此时该结点已经没有样本可供划分，该结点停止分裂；
- 二) 很多复杂的决策树算法（例如lightgbm）中还有额外的终止条件，为了防止过拟合：
- 树达到了最大深度： $\text{depth} \geq \text{max_depth}$ ，树停止分裂。
 - 结点的样本数量达到了阈值：如果一个集合（结点）的样本数量 $< \text{min_samples_leaf}$ ，则树停止分裂；
其中， max_depth 和 min_samples_leaf 都是人为制定的超参数

本文采用评价电信服务保障中的满意度预警专题来解释决策树算法，即假如我家办了电信的宽带，有一天宽带不能上网了，于是我打电话给电信报修，然后电信派相关人员进行维修，修好以后电信的回访专员询问我对这次修理障碍的过程是否满意，我会给我对这次修理障碍给出相应评价，满意或者不满意。根据历史数据可以建立满意度预警模型，建模的目的就是为了预测哪些用户会给出不满意的评价。目标变量为二分类变量：满意（记为0）和不满意（记为1）。自变量为根据修理障碍过程产生的数据，如障碍类型、障碍原因、修障总时长、最近一个月发生故障的次数、最近一个月不满意次数等等。简单的数据如下：

客户ID	故障原因	故障类型	修障时长	满意度
001	1	5	10.2	1
002	1	5	12	0
003	1	5	14	1
004	2	5	16	0
005	2	5	18	1
006	2	6	20	0
007	3	6	22	1
008	3	6	23	0
009	3	6	24	1
010	3	6	25	0

CART算法选择分裂属性的方式是比较有意思的，首先计算不纯度，然后利用不纯度计算Gini指标。以满意度预警模型为例，计算自变量故障原因的Gini指标时，先按照故障原因可能的子集进行划分，即可以将故障原因具体划分为如下的子集：{1,2,3}、{1, 2}、{1,3}、{2,3}、{1}、{2}、{3}、{}，共计 2^V 个子集。由于{1,2,3}和{}对于分类来说没有任何意义，因此实际分为 2^V-2 共计6个有效子集。然后计算这6个有效子集的不纯度和Gini指标，选取最小的Gini指标作为分裂属性。

不纯度的计算方式为：

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

p_i 表示按某个变量划分中，目标变量不同类别的概率。

某个自变量的Gini指标的计算方式如下：

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

对应到满意度模型中，A为自变量，即故障原因、故障类型、修障时长。D代表满意度，D1和D2分别为按变量A的子集所划分出的两个不同元组，如按子集{1,2}划分，D1即为故障原因属于{1,2}的满意度评价，共有6条数据，D2即故障原因不属于{1,2}的满意度评价，共有3条数据。计算子集{1,2}的不纯度时，即Gini(D1)，在故障原因属于{1,2}的样本数据中，分别有3条不满意和3条满意的数据，因此不纯度为 $1-(3/6)^2-(3/6)^2=0.5$ 。

以故障原因为例，计算过程如下：

$$\begin{aligned} Gini_{\text{故障原因}=\{1,2\}}(\text{满意度}) &= \frac{6}{10} Gini(D_1) + \frac{4}{10} Gini(D_2) \\ &= \frac{6}{10} * \left(1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right) + \frac{4}{10} * \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\ &= 0.5 \end{aligned}$$

计算子集故障原因={1,3}的子集的Gini指标时，D1和D2分别为故障原因={1,3}的元组共计7条数据，故障原因不属于{1,3}的元组即故障原因为2的数据，共计3条数据。详细计算过程如下：

$$\begin{aligned} Gini_{\text{故障原因}=\{1,3\}}(\text{满意度}) &= \frac{7}{10} Gini(D_1) + \frac{3}{10} Gini(D_2) \\ &= \frac{7}{10} * \left(1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2\right) + \frac{3}{10} * \left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) \\ &= 0.52 \end{aligned}$$

同理可以计算出故障原因的每个子集的Gini指标，按同样的方式还可以计算故障类型和修障时长每个子集的Gini指标，选取其中最小的Gini指标作为树的分支（Gini(D)越小，则数据集D的纯度越高）。连续型变量的离散方式与信息增益中的离散方式相同。

Regularisation Hyper-parameters

To prevent overfitting problem, there are several hyper-parameters can be adjusted

- maximum depth
- minimum samples for splitting
- minimum samples in a leaf
- maximum features evaluated

impurity in a leaf

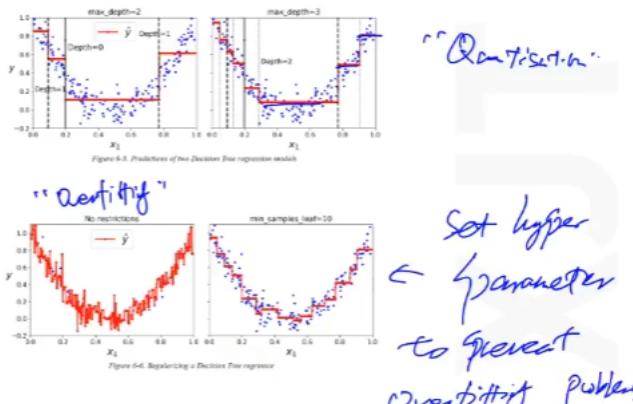
Classification of Machine Learning

- Nonparametric Model
- Parametric Model

Hyper-parameter

There are also hyper-parameters for regression

- Max Depth
- Min Samples



Model Parameters: These are the parameters that the model determines on its own while training on the dataset provided. These are the fitted parameters.

模型参数: 这些是模型在提供的数据集上进行训练时自行确定的参数。这些是拟合的参数。

Model Hyper-parameters: These are adjustable parameters that must be tuned, prior to model training, in order to obtain a model with optimal performance.

模型超参数: 这些是可调参数，必须在模型训练之前对其进行调整，以获取具有最佳性能的模型。

Decision Tree for Regression

分类任务: 预测目标是离散值，例如预测该用户是否会逾期，逾期是一类，用1表示，不逾期是另一类，用0表示。分类树采用GINI值作为结点分裂依据；

回归任务: 预测目标是连续值，例如预测用户的身高。回归树采用MSE(均方误差)作为结点分裂的依据。

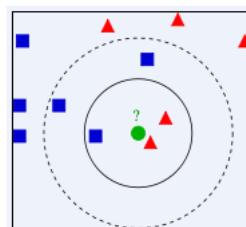
$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}}$$

- MSE means Mean Squared Error

8.4 k-Nearest Neighbour

1. k-NN算法简介

k近邻法是基本且简单的分类与回归方法，这里只讨论分类方法，利用数据集对特征向量空间进行划分，可以进行多分类。如下图：

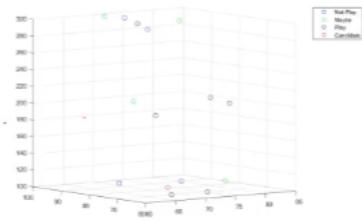
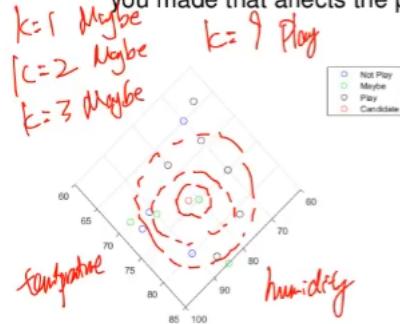


三角形与矩形分别代表两类数据，标签已知。现要对新输入的为分类点(绿色)进行分类，k-NN的做法是寻找与该绿点相邻最近的k个点(k-NN算法的k的含义，图中的距离为欧式距离)，然后通过多数表决的方式把绿点划分到这k个最近点出现频数最高的类。例如如果k取3，则绿点最近的3个点中频数最高为三角形类，所以归为三角形类；若k取5，则距离绿点最近的5个点中频数最高为矩形类，所以归绿点为矩形类。

outlook	temperature	humidity	windy	play
sunny	85	85	1.0	maybe
sunny	80	90	3.0	no
overcast	83	86	0.8	yes
rainy	70	96	0.2	maybe
rainy	68	80	0.1	yes
rainy	65	70	2.8	no
overcast	64	65	2.6	yes
sunny	72	95	0.3	no
sunny	69	70	0.5	yes
rainy	75	80	0.4	maybe
sunny	75	70	2.2	yes
overcast	72	90	2.4	maybe
overcast	81	75	0.0	yes
rainy	71	91	2.9	no

In-class Discussion

- Will we play golf in the case of 74 (temperature), 74 (humidity)?
- What value of k should be selected?
- Will we play golf in the case of sunny, 74, 82? What decision you made that affects the prediction?



1.1 模型

输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, 实例特征向量 x

总共 N 个样本, $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ 为实例的特征向量, $y_i \in \mathcal{Y} =$

$\{c_1, c_2, \dots, c_K\}$ 为实例的类别, $i = 1, 2, \dots, N$

输出：实例 x 所属的类 y

step1: 根据给定的距离度量, 在训练集 T 中找出与 x 最近的 k 个点, 涵盖这 k 个点的 x 的邻域记为 $N_k(x)$;

step2: 在 $N_k(x)$ 中根据决策规则 (如多数表决) 决定 x 的类别 y

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

其中 I 为指示函数

1.2 学习策略

$$\max_{c_j} \sum_{x_i \subseteq N_k(x)} I(y_i = c_j), \quad i = 1, 2, \dots, N, j = 1, 2, \dots, K;$$

1.3 学习算法

学习算法即如何求出以上的学习策略的最大值, 用的是 **多数表决法**, 即将 c_1 到 c_K 得到的指示函数和的值排序, 选择最大值对应的类别 c^* 作为输出。假设 c^* 为最大值对应的类别, 那么得到的最终模型为:

$$y = c^*$$

1.4 距离度量

从以上公式可以看出, 关键是确定输入实例 x 的邻域 $N_k(x)$, 而这个邻域又由两个方面决定, 一个就是如何计算各点与输入实例 x 的距离 (怎么判断离某些点近不近)?

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

- $p = 1$ 曼哈顿距离
 - $p = 2$ 欧氏距离
 - $p = \infty$ 切比雪夫距离
- 一般使用欧式距离。

1.5 k值选择

知道了怎么判断点之间离的近不近, 那么要确定邻域还得需要知道该邻域包含多少个“最近”点, 这就是 k 值决定的, 该邻域会包含 k 个离输入实例最近的点。 k 值越小, 模型越复杂, 过拟合的风险越大, 当 $k=1$ 时成为最近邻模型。而 k 值越大, 模型越简单, 最极端情况就是 $k=N$, 这时直接把样本中出现频数最高的类当成所有输入实例的类。

距离的度量以及 k 值作为超参数, 可以通过验证集来选择合适的超参数。

优点:

- 简单好用，容易理解，精度高，理论成熟，既可以用来做分类也可以用来做回归；
- 对异常值不敏感

缺点:

- 最大的缺点是无法给出数据的内在含义。
- 计算复杂性高；空间复杂性高；
- 样本不平衡问题（即有些类别的样本数量很多，而其它样本的数量很少）；
- 一般数值很大的时候不用这个，计算量太大。而且k近邻算法最终只用了最近k个点的信息来做决策，而舍弃了其余点。

lec9Ensemble Learning and Random Forests

9.1 Voting Classifiers

VotingClassifier背后的想法是组合概念上不同的机器学习分类器，并使用多数投票或平均预测概率（软投票）来预测类别标签。这样的分类用于一组性能同样良好的模型，以平衡它们各自的弱点。

- 分类

a) hard voting: **多数**投票法，也叫硬投票，根据少数服从多数的原则 (**Majority Class Labels**)

b) soft voting: **加权**投票法，增加了权重weight参数，使用加权平均概率 (Weighted Average Probabilities)，通常会比 hard voting 效果好-

9.2 Bagging and Pasting

虽然有很多的机器学习算法，但是从投票的角度看，仍然不够多，如果有成千上万的投票者，才能保证最终结果可信（**大数定律**）。

所以需要**创建更多的子模型**，并且**子模型之间要有差异性**，不能保持一致。

每个子模型不需要太高的准确率

对于数据集，每次放回取样叫做**Bagging**，不放回取样叫做**Pasting**，Bagging使用的多(统计学中放回取样叫做bootstrap，这也是sklearn放回不放回的参数)

2.1 Bagging (Bootstrap ~ aggregating)

- 算法流程

- Training set consists of m training examples drawn randomly **with replacement** from the original training set of n items. (The training set is called a bootstrap replicate)
- Train a classifier with the training set.
- Repeat the procedure L times and get L classifiers.
- Test: hard voting

- 应用场景

- 不太稳定的分类器：如决策树（SVM相对来说比较稳定）。

- evaluate

- 用每个 predictor 的 out-of-bag (oob, 即未被采样的样本) evaluate, 因为 Predictor 在 training 的过程中不会见到 oob 样本。
- ensemble 的性能可以通过 average 每个 Predictor 的 oob evaluation 得到。

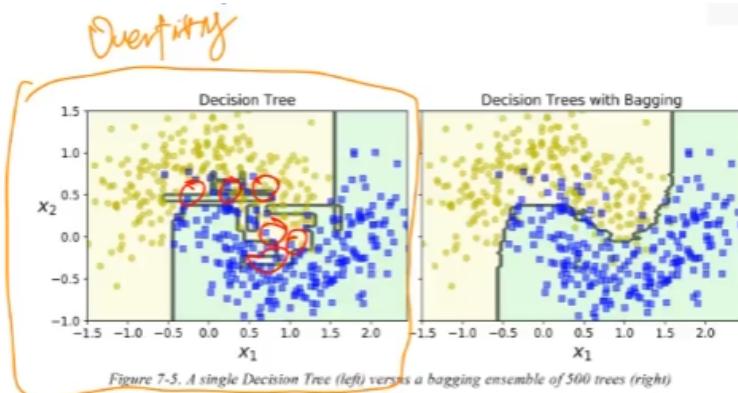
2.2 Pasting

与Bagging不同的是，采样过程是 **without replacement** 的。

【注意】

- bagging 和 pasting 都允许同一训练样本参与多个 predictors 的训练，但只有 bagging 才允许同一训练样本在同一 predictor 的训练中被多次采样。
- 每个独立的 predictor 的 bias 都会比在整个训练集上 train 得的 predictor 高，但是 aggregation 操作能够同时降低 bias 和 variance. 即，最终得到的 ensemble 具有相似的 bias，但是 variance 更小。
- 虽然训练时间更长，但是可以并行训练。

上面加了个evaluation注意！！！！！ oob评估



Discussion

Why the bagging ensemble leads to a more sensible decision boundary?

Reduces overfitting

Reduces the impact of outliers

减少过拟合通过减少离散值的影响

9.3 Random Patches and Random Subspaces (features selection)

random patches method

sampling both training instances and features (randomly select features **and** randomly select sample) 针对特征进行随机取样，不针对样本随机取样

random subspaces method

keeping all training instances but sampling features (randomly select features for training) 既针对样本，又针对特征进行随机取样

9.4 Random Forest

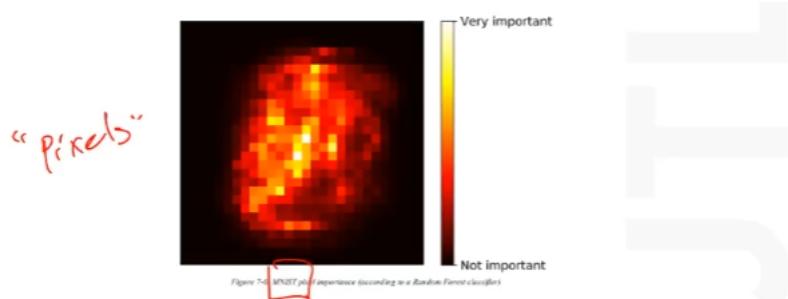
由很多棵决策树构成，最后通过voting等手段进行ensemble。由于决策树的构建具有很大的随机性，所以random forest能够有效降低决策树计算的variance，从而取得更好的效果。

随机森林本质上是许多以不同方式过拟合的决策树的集合，我们可以对这些互不相同的树的结果取平均值来降低过拟合，这样既能减少过拟合又能提高树的预测能力。

9.5 Feature Importance

一般来说，就单个决策树，越重要的特征距离根节点越近，因此，特征的重要度可能通过随机森林中各决策树中各特征的平均深度来衡量。在此基础上，可以进一步展开feature selection操作。

那些可以用来做预测的特征是重要的



Discussion

How can the importance of feature be measured?

Features that are used to make prediction.

9.6 AdaBoost

- 每一轮的训练数据样本赋予一个权重，并且每一轮样本的权值分布依赖上一轮的分类结果。
- 基分类器之间采用序列式的线性加权方式进行组合。

boost: Originally called hypothesis boosting.

General idea: train predictors sequentially, each trying to correct its predecessor.

代表性 booting 算法

AdaBoost (Adaptive Boosting)

Gradient Boosting

Adaboost

之前分类错的样本会被赋予更高的权重

- Motivation
 - finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule.
 - Combine these weak rules into a single prediction rule that will be much more accurate than any one of the weak rules.
- 核心思想
 - 序列学习的分类器，给错分的样本更大的采样权重
 - The final classifier is constructed by a weighted vote of the individual classifiers.
- 缺陷
 - 无法并行计算或只能部分并行计算

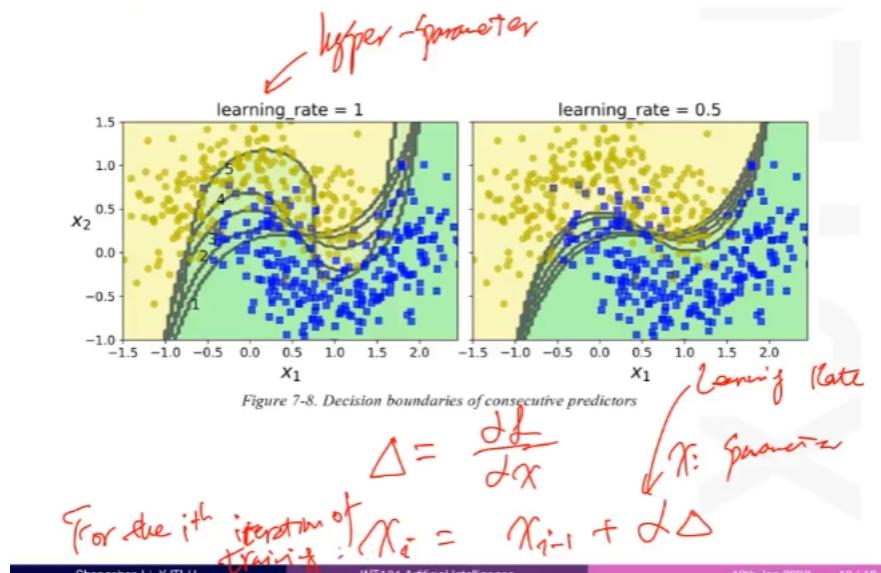
停止条件

- predictors 的数目达到上限
- A perfect predictor is found.

9.6 Gradient Boosting for regression

对比应该的值和预测的值

- 核心思想
 - working sequentially.
 - fit the new predictor to the **residual errors** made by the previous predictor.
 - 更多面向 regression task
- 典型应用: Gradient Tree Boosting / Gradient Boosted Regression Tree (GBRT)
 - base predictor: Regression Tree
 - 做预测: 把不同predictor的预测结果加起来即可
 - regularization: **shrinkage**
 - 通过调节learning_rate实现
 - learning_rate: scales the contribution of each tree
 - learning_rate越小, ensemble需要越多的回归树来拟合训练集, regularization效果越明显。
- 如何确定最优的 ensemble 包含的 tree 的数目?
 - **early stopping**: 监测ensemble中每加入n trees 后 validate ensemble 的预测准确度:
- Stochastic Gradient Boosting
 - 每次利用在原始训练集上进行采样得到的子训练集训练
 - trade a higher bias for a lower variance
 - speed up training considerably

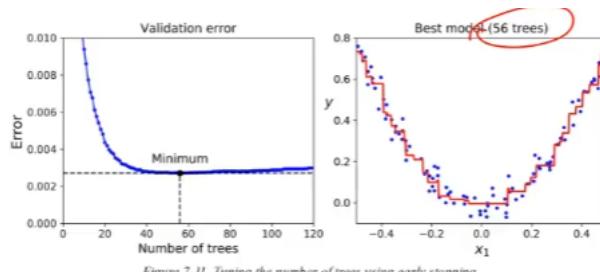


学习率过小容易导致计算时间花费很多以及拟合路程长?

Prevent Overfitting

要构造一个随机森林模型, 第一步是确定森林中树的数目, 通过模型的 `n_estimators` 进行调节。`n_estimators`越大越好, 但占用的内存与训练和预测的时间应增长, **过拟合的风险也在增加且边际效益是递减的**, 所以要在可承受的内存/时间内选取尽可能大的`n_estimators`。

`n_estimators` : 指定随机森林中的分类器的个数, 默认为10。一般来说`n_estimators`太小容易欠拟合, 太大计算量大, 故需要参数调优选择中的数值; 找一个最好的



9.7 Stacking (Stacked Generalization, 层叠/堆栈泛化)

- 核心思想
 - 训练一个模型(blender / meta learner)来生成最终对多个predictors的整合，而不是通过直接利用投票/加权和的方式来产生最终预测结果。
- to train a blender
 - 原始训练集 T 拆分为两个子集 T_1 和 T_2
 - T_1 : 用于训练构成 ensemble 的多个 predictors
 - T_2 : the hold-out set
 - 利用 predictors 进行预测，得到多个 predictions
 - 再利用 T_2 训练 blender
 - blender 输入特征的维度即为 predictors 的数目
- 可以同时训练多个 blenders
 - 例如训练服昂发分别为 Linear Regression, Radom Forest Regression and so forth.
 - 从而形成 a layer of blenders
 - 此时，需要把训练集分为 3 部分

lec10 Unsupervised Learning

10.1 Clustering

1. 将数据集在某些方面相似的数据成员进行分类组织的过程
2. 与回归与分类不同，聚类是无监督学习算法，无监督指的是只需要数据，不需要标记结果，自己给自己分类。它与分类问题最明显的区别：类问题有事先的标注，而聚类的分组是完全靠自己学习得来的。
3. 在现实生活中，很少直接用聚类算法，因为聚类效果的好坏不容易衡量（因为没有标记，就没有标准答案），有时候会用做监督学习中稀疏的预处理，把混乱的数据先分分看，看看大类如何。

聚类算法，顾名思义是将数据聚集起来的算法，是一个无监督学习方法。他是为了实现将数据按照某一标准（相似度）将整个数据集分为若干子（簇），最终的分类结果要尽量保证组内相似度尽可能大，组间相似度尽可能小。而所谓的标准，实则为各种的距离。在本博客中，主要讲述三类方法，分别是K-means聚类，层次聚类（AgglomerativeClustering），密度聚类（DBSCAN）。

聚类前需要进行数据标准化！

10.2 K-means聚类

K-means算法是很典型的基于距离的聚类算法。它是通过将样本划分为k个方差齐次的类来实现数据聚类。该算法需要指定划分的类的个数，即化误差函数的基础上将数据划分为预定的类数K，采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。

具体步骤

1. 随机选取k个数据点作为初始聚类中心centroids（某聚类中所有数据点的均值）
2. 计算每个数据点与各个种子聚类中心之间的距离，把每个数据点分配给距离它最近的聚类中心，形成一个聚类
3. 待所有数据点均被分配后，每个聚类根据当前聚类中的数据点，重新计算出新的聚类中心
4. 以上步骤不断重复，直到没有(或最小数目)数据点被重新分配给不同的聚类/没有(或最小数目)聚类中心再发生变化/误差平方和(SSE)局部最小

特点：

- 对于k值选取要求比较高，并且需要进行试探性去取值
- 当结果簇是密集的，而簇与簇之间区别明显时，处理类间方差差异大的样本时，有着很好的效果。
- 处理类规模较大的效果很好

缺点：

- 在簇的平均值被定义的情况下才能使用，这对于处理符号属性的数据不适用。
- 必须事先给出k（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
- 它对于“噪声”和孤立点数据是敏感的，少量的该类数据能够对平均值产生极大的影响。
- 采用迭代方法，可能只能得到局部的最优解，而无法得到全局的最优解。

总结：

K-means算法不适合处理比较离散的数据.同时对于k值得选取至关重要，换句话说，我们对k值得选取比较被动.在k值得选取方面可能需要尝试，最后指定相应的关于k值得散点图

要多次尝试去找到最好的那个

1. **Elbow Method:** The main calculation in the Elbow Method is the sum of squared errors (SSE), which is computed for different values of K. The SSE for a given K is calculated as:

$$SSE = \sum_{i=1}^n \sum_{j=1}^K w_{ij} \|x_i - c_j\|^2$$

where:

- n is the number of data points.
- K is the number of clusters.
- w_{ij} is 1 if data point i belongs to cluster j and 0 otherwise.
- x_i is the i -th data point.
- c_j is the centroid of cluster j .
- $\|x_i - c_j\|^2$ is the squared Euclidean distance between data point i and the centroid of cluster j .

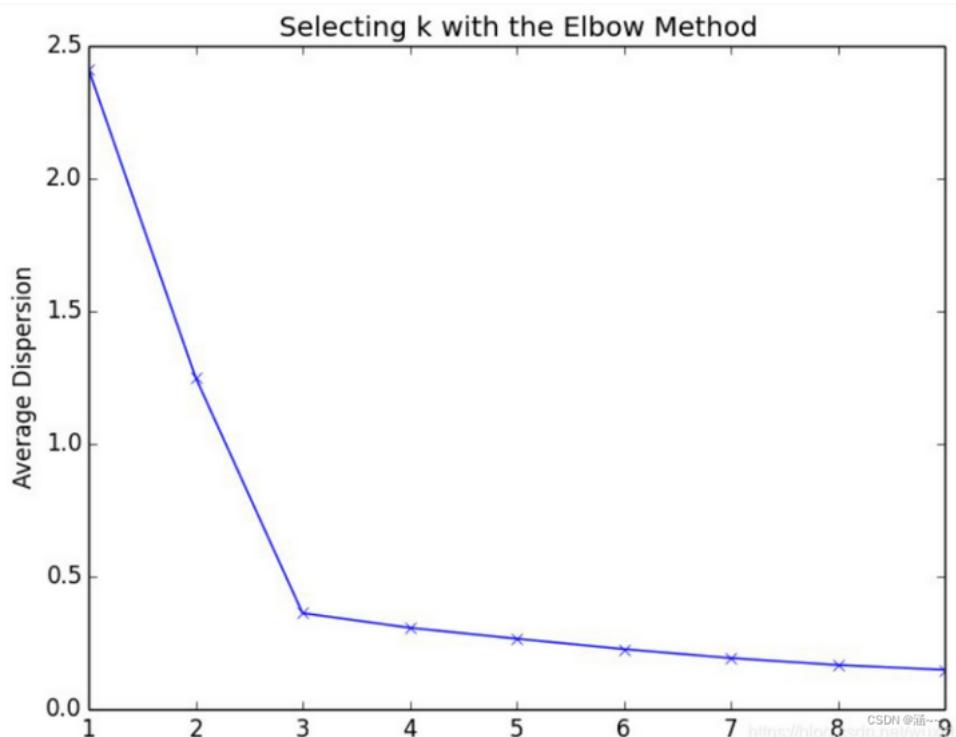
手肘法的核心思想是：随着聚类数k的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。并且，当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后随着k值的继续增大而趋于平缓，也就是说SSE和k的关系图是一个手肘的形状，而这个肘部对应的k值就是数据的真实聚类数。当然，这也是该方法被称为手肘法的原因。

步骤：

计算k从1到n的的SSE

SSE会逐渐减小，直到 $k=n$ 时 $SSE=0$,每个点都是质心

在SSE减小过程，会出现拐点，这个拐点就是肘，下降率突然变缓时，就是最佳K值。



2. **Silhouette Analysis:** The silhouette score for each data point is calculated using the formula:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where:

- $s(i)$ is the silhouette score for data point i .
- $a(i)$ is the average distance from data point i to the other points in the same cluster.
- $b(i)$ is the minimum average distance from data point i to points in a different cluster, minimized over clusters.

其中， a 是 X_i 与同簇的其他样本的平均距离，称为凝聚度， b 是 X_i 与最近簇中所有样本的平均距离。

$$C_j = \arg \min_{C_k} \frac{1}{n} \sum_{p \in C_k} |p - X_i|^2$$

http://blog.csdn.net/l5738501
CSDN @鱼~~

其中 p 是某个簇 C_k 中的样本。事实上，简单点讲，就是用 X_i 到某个簇所有样本平均距离作为衡量该点到该簇的距离后，选择离 X_i 最近的一个簇作为最近簇。

求出所有样本的轮廓系数后再求平均值就得到了平均轮廓系数。平均轮廓系数的取值范围为[-1,1]，且簇内样本的距离越近，簇间样本距离越远，平均轮廓系数越大，聚类效果越好。那么，很自然地，平均轮廓系数最大的 k 便是最佳聚类数。

10.2 Mini-batch K-Means

Mini Batch K-Means算法

- Mini Batch K-Means算法是K-Means算法的一种优化变种，采用**小规模的数据子集**(每次训练使用的数据集是在训练算法的时候随机抽取的数据子集)**减少计算时间**，同时试图优化目标函数；Mini Batch K-Means算法可以减少K-Means算法的收敛时间，而且产生的结果效果只是略差于标准K-Means算法
- 算法步骤如下：
 - 首先抽取部分数据集，使用K-Means算法构建出 K 个聚簇点的模型
 - 继续抽取训练数据集中的部分数据集样本数据，并将其添加到模型中，分配给距离最近的聚簇中心点
 - 更新聚簇的中心点值(每次更新都只用抽取出来的部分数据集)
 - 循环迭代第二步和第三步操作，直到中心点稳定或者达到迭代次数，停止计算操作

<https://blog.csdn.net/yiyue21>

10.3 Semi-Supervised Learning

收集了大量未标记的数据，您想在这些数据上训练模型。手动标记所有这些信息可能会花费你一大笔钱，除了需要几个月的时间来完成注释。这半监督机器学习方法就派上用场了。工作原理很简单。无需将标签添加到整个数据集，而是仅对**一小部分数据进行手工标记**并使用它来训练模型后将其应用于未标记数据的海洋。

Semi-Supervised Learning

- 1 Clustering with K-Means
- 2 Using Centroids to Approximate Samples
- 3 Train Classifier

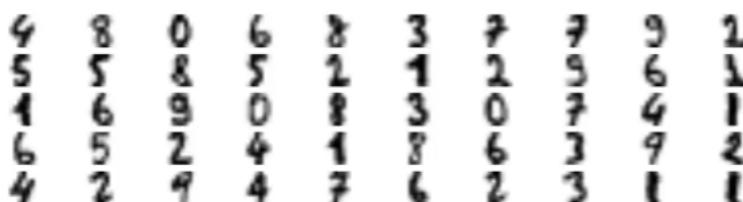


Figure 9-13. Fifty representative digit images (one per cluster)

Active Learning

Interactive labelling for low-confidence samples

- 1 Count instances located within ϵ , which is called the instance's ϵ -neighbourhood
- 2 A instance that has a certain number of instances in ϵ -neighbourhood is considered as a core instance
- 3 All instances in the neighbourhood of a core instance belong to the same cluster
- 4 Any instance that is not a core instance and does not have one in its neighbourhood is considered as an anomaly

10.4 Semi-Supervised Learning

层次聚类(Hierarchical Clustering)是聚类算法的一种，通过计算不同类别数据点间的相似度来创建一棵有层次的嵌套聚类树。在聚类树中，不同原始数据点是树的最低层，树的顶层是一个聚类的根节点。

算法维护一个簇集合，

第一次循环，簇集合内每个簇中只有一个个体

第二次循环，对所有簇集合，将两个最邻近的簇合并

第三次循环，对所有簇集合，将两个最邻近的簇合并

...

一直循环到只剩下一个簇

由于层次聚类将小簇从下到上合并成大簇，所以层次聚类自底向上bottom-up

► The general outline of agglomerative clustering is

1. For n objects v_1, \dots, v_n , assign each to a singleton cluster $C_i = \{v_i\}$
2. Use any computable cluster similarity measure $D(C_i, C_j)$ e.g. Euclidean distance, cosine distance etc.
3. Repeat{
 - identify the two most similar clusters C_j and C_k (could be ties - choose one pair)
 - delete C_j and C_k and add $(C_j \cup C_k)$ to the set of clusters
} until just one cluster remains
4. Use a dendrogram diagram to show the sequence of cluster mergers

· 如何计算点m与点n之间距离：

……一般题目会给出点间距离，PPT计算时用的是曼哈顿距离

· 如何计算簇 C_1 与簇 C_2 的距离：

1. single linkage:

D is taken as the minimum distance between samples in sub-clusters

计算 C_1 内各点到 C_2 内各点的距离，选取最短的那个。

2. complete linkage:

D is taken as the maximum distance between samples in sub-clusters

计算 C_1 内各点到 C_2 内各点的距离，选取最长的那个。

3. average linkage:

D is taken as the average distance between each pair of samples in sub-clusters

计算 C_1 内各点到 C_2 内各点的距离，选取它们的平均数。

特点：

- 层次聚类算法对于k值选取要求也比较高，并且也需要进行试探性去取值
- 对较为离散的数据，分类效果比Kmeans好

缺点：

- 计算复杂度高，不适合数据量大的数据集；

例题

给定距离矩阵（邻接矩阵），按照average linkage求层次聚类（题目要求用啥方法就用啥方法）：

	1	2	3	4	5
1	0				
2	8	0			
3	3	6	0		
4	5	5	8	0	
5	13	10	2	7	0

可以注意到除了被组合的两簇所在的行和列，其他行和列的值是不需要重新计算的，如：合并5, 3后的新矩阵的前三行 (0, 8 0, 5 5 0)，减少重复计算

	1	2	3	4	5
1	0				
2	8	0			
3	3	6	0		
4	5	5	8	0	
5	13	10	2	7	0

发现5, 3距离最小，合并，然后再次计算邻接矩阵

	1	2	4	(3,5)
1	0			
2	8	0		
4	5	5	0	
(3,5)	ave(3,13)=8	ave(6,10)=8	ave(8,7)=7.5	0

发现4,1和4,2距离最小都是5，这里选择合并4,2，然后计算邻接矩阵

	1	(2,4)	(3,5)
1	0		
(2,4)	(8+5)/2=6.5	0	
(3,5)	(3+13)/2=8	两两间4个距离平均=7.75	0

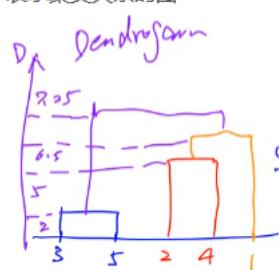
发现1,(2,4)距离最小，合并，算邻接矩阵

	(1,2,4)	(3,5)
(1,2,4)	0	
(3,5)	两三间6个距离平均=7.25	0

https://blog.csdn.net/sanmuse_wu

画层次聚类树状图dendrogram：

表示聚类关系的图



lec11 Gaussian Mixture Model and Bayesian Statistics

11.1 Gaussian Mixture Model

混合模型是一个可以用来表示在总体分布（distribution）中含有 K 个子分布的概率模型，换句话说，混合模型表示了观测数据在总体中的概率分布，它是一个由 K 个子分布组成的混合分布。混合模型不要求观测数据提供关于子分布的信息，来计算观测数据在总体分布中的概率。

这样的分布模型也可以做聚类，其中最经典的是高斯混合模型

与 k 均值类似，但更包容，为了解决数据缺失或存在未发现特征的情况。

不再用“距离中心最短”硬性分割类别，而是用“有多大高斯分布概率属于这个类”的思想去分类

每个高斯分量对应一个组，假设数据点符合高斯分布，通过最大期望算法 EM 找到每个组的高斯参数（即均值和标准差），进而求得每个组的高斯分布，从而得到组

最大期望算法 EM

EM算法包含两个步骤，E步和M步。E步也就是我们求期望的步骤，M步将E步所求的期望最大化，重复E步和M步直到收敛，也就是我们估计的数不再发生变化或者变化幅度很小。这就是EM算法的基本概括，

分为期望和最大化期望步骤：

期望Expectation step: find the posterior probability according to current model

期望最大化Maximisation step: calculate the new modal parameters

算法维护k个类的k个高斯函数：

第一次循环，随机初始化这些高斯函数。

每次循环一开始时，重新计算每个样本根据高斯函数划分到那个类的概率（拟分类）。

然后，求出每组中的样本的概率，重塑高斯函数使这个组的样本概率最大化（更新高斯函数）。

再次循环，直到每组内样本概率收敛。

The mixture model could be used for clustering as well

The most classical model is Gaussian Mixture Model (GMM)

The definition of Gaussian Mixture Model is

$$p(\mathbf{x}) = \sum_k \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k) \quad (1)$$

where

- \mathcal{N} represents Gaussian distribution (known as Gaussian component in GMM)
- π is the weight of Gaussian component
- μ is the mean of Gaussian component
- Σ is the co-variance matrix of Gaussian component

Each Gaussian component corresponds to a cluster

高斯混合模型可以看作是由K个单高斯模型组合而成的模型，这K个子模型是混合模型的隐变量(Hidden variable)。一般来说，一个混合模型可以使用任何概率分布，这里使用高斯混合模型是因为高斯分布具备很好的数学性质以及良好的计算性能。

说起高斯分布，大家都不陌生，通常身高、分数等都大致符合高斯分布。因此，当我们研究各类数据时，假设同一类的数据符合高斯分布，也是很自然的假设；当数据事实上有多个类，或者我们希望将数据划分为一些簇时，可以假设不同簇中的样本各自服从不同的高斯分布，由此得到的聚类算法称为高斯混合模型。

高斯混合模型的核心思想是，**假设数据可以看作从多个高斯分布中生成出来的。在该假设下，每个单独的分模型都是标准高斯模型，其均值 μ_i 和方差 Σ_i 是待估计的参数。此外，每个分模型都还有一个参数 π_i ，可以理解为权重或生成数据的概率。**高斯混合模型的公式为

$$p(x) = \sum_{i=1}^K \pi_i N(x | \mu_i, \Sigma_i)$$

CSDN @意念回复

高斯混合模型是一个生成式模型。可以这样理解数据的生成过程，假设一个最简单的情况，即只有两个一维标准高斯分布的分模型 $N(0,1)$ 和 $N(5,1)$ ，其权重分别为0.7和0.3。那么，在生成第一个数据点时，先按照权重的比例，随机选择一个分布，比如选择第一个高斯分布，接着从 $N(0,1)$ 中生成一个点，如-0.5，便是第一个数据点。在生成第二个数据点时，随机选择到第二个高斯分布 $N(5,1)$ ，生成了第二个点4.7。如此循环执行，便生成出了所有的数据点。

评价GMM

1. 模型拟然model likelihood

已知模型但未知参数，如：对于一个庞大的集，只知道符合高斯分布但不知道均值和方差，通过采样，可以获取部分子集的数据，然后通过最大似然估计来获得正态分布的均值与方差。

给定输出 x 时，关于参数 θ 的似然函数 $L(\theta|x)$ （在数值上）等于给定参数 θ 后变量 X 的概率：

$$L(\theta|x) = P(X=x|\theta)$$

2. 模型选择准则model selection criteria

· AIC Akaike Information Criterion

$$AIC = 2k - 2LL$$

· BIC Bayesian Information Criterion

$$BIC = k \log N - 2LL$$

Akaike Information Criterion

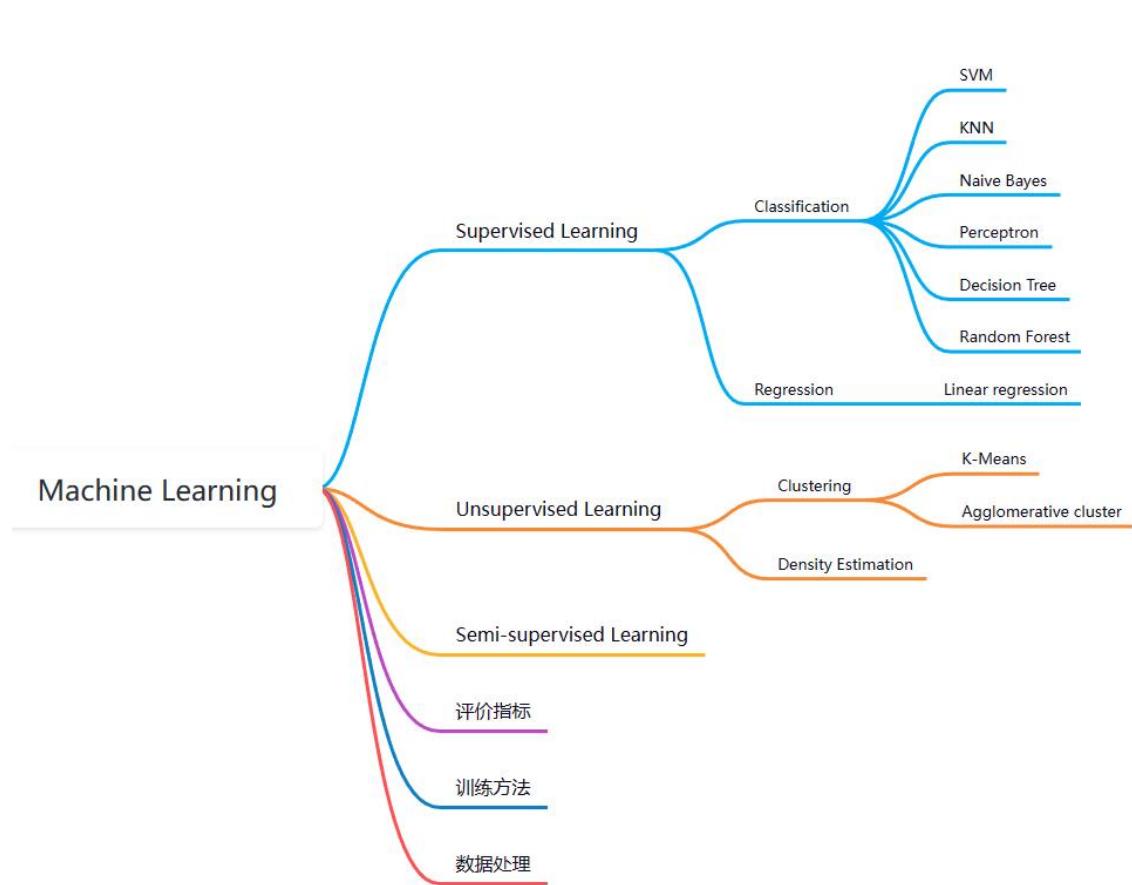
$$AIC = 2p - 2 \log \hat{L} \quad (2)$$

Bayesian Information Criterion

$$BIC = p \log(m) - 2 \log \hat{L} \quad (3)$$

- m is the number of instances
- p is the number of parameters learned by the model
- \hat{L} is the maximised value of the likelihood function of model

INT104 期末复习



根据往年试题，考试形式不外有监督/无监督模型算法(2021 年)。大题以模型的计算展开。

1.KNN

1. (10 points) The table below shows a training set with 10 examples that is used for training a **3-nearest-neighbors** classifier that uses Manhattan distance, i.e., the distance between two points at coordinates p and q is $|p-q|$. The only attribute, X , is real-valued, and the label Y has two possible classes, 0 and 1. The first fold contains the first 5 examples, and the second fold contains the last 5 examples. In case of ties in distance, use the example with smallest X value as the neighbor. Please compute the 2-fold cross validation accuracy (percentage correct classification).

X	0	1	2	3	4	5	6	7	8	9
Y	1	0	1	0	1	0	1	0	1	0

2.K-means

2. (10 points) You want to cluster 7 points into 3 clusters using the k-means clustering algorithm. Suppose after the first iteration, clusters C_1 , C_2 and C_3 contain the following two-dimensional points:

C_1 contains the 2 points: $\{(0, 6), (6, 0)\}$

C_2 contains the 3 points: $\{(2, 2), (4, 4), (6, 6)\}$

C_3 contains the 2 points: $\{(5, 5), (7, 7)\}$

Please compute the coordinates of cluster centers for these 3 clusters.

2. K-means

$$C_1 = \left(\frac{0+6}{2}, \frac{6+0}{2} \right) = (3, 3)$$

$$C_2 = \left(\frac{2+4+6}{3}, \frac{2+4+6}{3} \right) = (4, 4)$$

$$C_3 = \left(\frac{5+7}{2}, \frac{5+7}{2} \right) = (6, 6)$$

3. Naive Bayes(朴素贝叶斯)

3. (20 points) The following dataset as in the table is provided to build a naive Bayes classifier, where $\{x_1, x_2, x_3, x_4\}$ and l are the features and the label, respectively. Please give the process of building the classifier and predict the label of the unknown instance $x = [1, 0, 1, 1]^T$.

x_1	x_2	x_3	x_4	l
0	1	0	1	0
0	0	1	0	0
1	1	1	0	0
1	0	1	0	1

$$3. P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

$$P(y|x) \propto P(y)P(x|y)$$

↑ ↑ ↑
posterior prior CCP

① when $y=0$

$$P(y=0) = \frac{3}{4}$$

直接得

$$P(x_1=0|y=0) = \frac{2}{3}, P(x_1=1|y=0) = \frac{1}{3}$$

$$P(x_2=0|y=0) = \frac{1}{3}, P(x_2=1|y=0) = \frac{2}{3}$$

$$P(x_3=0|y=0) = \frac{1}{3}, P(x_3=1|y=0) = \frac{2}{3}$$

$$P(x_4=0|y=0) = \frac{2}{3}, P(x_4=1|y=0) = \frac{1}{3}$$

因此可以得到：

$$x = [1, 0, 1, 1]^T$$

$$P(y=0|x) = \frac{P(y=0) \cdot P(x|y=0)}{P(x)}$$

$$\propto P(y=0) \cdot P(x|y=0)$$

$$= P(x_1=1, x_2=0, x_3=1, x_4=1 | y=0)$$

$$= \prod_{i=1}^4 P(x_i|y=0) = \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{3^4}$$

$$= \frac{1}{81}$$

4. 感知机模型

4. (20 points) Perceptron is a function that maps input x to a label as follows

$$f(\mathbf{x}) = \begin{cases} 1, & w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

Now consider solving the logical **OR** and logical **XOR** problems (as shown in two tables) with the perceptron model.

$$y = f(\mathbf{x}) = \begin{cases} 1, & w_1x_1 + w_2x_2 + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

Table 1: Logical OR

x_1	x_2	y
0	1	1
1	1	1
1	0	1
0	0	0

Table 2: Logical XOR

x_1	x_2	y
0	1	1
1	1	0
1	0	1
0	0	0

1) (4 points) Please draw all datapoints of the tables in the two-dimensional space for logical OR and logical XOR problems, respectively, where different classes are marked with different shapes.

2) (16 points) Please explain separately whether the perceptron can mimic the output of logical OR and logical XOR or not. If so, please give an example of function $f(\mathbf{x})$; if not, please prove that there is no such function $f(\mathbf{x})$.

第一题

1. (10 points) The table below shows a training set with 10 examples that is used for training a **3-nearest-neighbors** classifier that uses Manhattan distance, i.e., the distance between two points at coordinates p and q is $|p-q|$. The only attribute, X , is real-valued, and the label Y has two possible classes, 0 and 1. The first fold contains the first 5 examples, and the second fold contains the last 5 examples. In case of ties in distance, use the example with smallest X value as the neighbor. Please compute the 2-fold cross validation accuracy (percentage correct classification).

X	0	1	2	3	4	5	6	7	8	9
Y	1	0	1	0	1	0	1	0	1	0

2020-2021 -final exam

1. KNN: X 训练集, X 测试集
 y 为预测值
① $X \in [0,4]$, $X \in [5,9]$

Input	3-nearest-neighbors	Predict
$X=5$	$X=2,3,4$	$y=1$
$X=6$	$X=2,3,4$	$y=1$
$X=7$	$X=2,3,4$	$y=1$
$X=8$	$X=2,3,4$	$y=1$
$X=9$	$X=2,3,4$	$y=1$

The accuracy of this fold is

$$A_1 = \frac{1}{5} \sum_{i=0}^4 I(y_i = Y_i)$$

$$= \frac{1}{5} (0 + 1 + 0 + 1 + 0) \\ = \frac{1}{5} \times 2 = \frac{2}{5} = 40\%$$

② $X \in [5,9]$, $X \in [0,4]$

Input	3-NV	Predict
$X=0$	$X=5,6,7$	$y=0$
$X=1$	$X=5,6,7$	$y=0$
$X=2$	$X=5,6,7$	$y=0$
$X=3$	$X=5,6,7$	$y=0$
$X=4$	$X=5,6,7$	$y=0$

$$A_2 = \frac{1}{5} \sum_{i=0}^4 I(y_i = Y_i) = \frac{1}{5} (0 + 1 + 0 + 1 + 0) \\ = \frac{2}{5} = 40\%$$

So the total accuracy A is

$$A = \frac{\sum A_i}{n} = \frac{A_1 + A_2}{2} = \frac{\frac{2}{5} + \frac{2}{5}}{2} = \frac{2}{5} \\ = 40\%$$

例题

Agglomerative Clustering(层次聚类)

例 14.1 给定 5 个样本的集合，样本之间的欧氏距离由如下矩阵 D 表示：

$$D = [d_{ij}]_{5 \times 5} = \begin{bmatrix} 0 & 7 & 2 & 9 & 3 \\ 7 & 0 & 5 & 4 & 6 \\ 2 & 5 & 0 & 8 & 1 \\ 9 & 4 & 8 & 0 & 5 \\ 3 & 6 & 1 & 5 & 0 \end{bmatrix}$$

其中 d_{ij} 表示第 i 个样本与第 j 个样本之间的欧氏距离。显然 D 为对称矩阵。应用聚合层次聚类法对这 5 个样本进行聚类。

解 (1) 首先用 5 个样本构建 5 个类, $G_i = \{x_i\}$, $i = 1, 2, \dots, 5$, 这样, 样本之间的距离也就变成类之间的距离, 所以 5 个类之间的距离矩阵亦为 D 。

(2) 由矩阵 D 可以看出, $D_{35} = D_{53} = 1$ 为最小, 所以把 G_3 和 G_5 合并为一个新类, 记作 $G_6 = \{x_3, x_5\}$ 。

(3) 计算 G_6 与 G_1, G_2, G_4 之间的最短距离, 有

$$D_{61} = 2, \quad D_{62} = 5, \quad D_{64} = 5$$

又注意到其余两类之间的距离是

$$D_{12} = 7, \quad D_{14} = 9, \quad D_{24} = 4$$

显然, $D_{61} = 2$ 最小, 所以将 G_1 与 G_6 合并成一个新类, 记作 $G_7 = \{x_1, x_3, x_5\}$ 。

(4) 计算 G_7 与 G_2, G_4 之间的最短距离:

$$D_{72} = 5, \quad D_{74} = 5$$

又注意到

$$D_{24} = 4$$

显然, 其中 $D_{24} = 4$ 最小, 所以将 G_2 与 G_4 合并成一个新类, 记作 $G_8 = \{x_2, x_4\}$ 。

(5) 将 G_7 与 G_8 合并成一个新类, 记作 $G_9 = \{x_1, x_2, x_3, x_4, x_5\}$, 即将全部样本聚成一类, 聚类终止。

上述层次聚类过程可以用图 14.2 所示的层次聚类图表示。

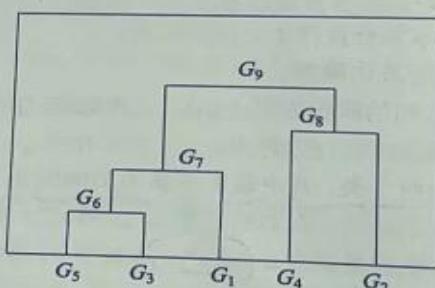


图 14.2 层次聚类图

K-Means

例 14.2 给定含有 5 个样本的集合

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

试用 k 均值聚类算法将样本聚到两个类中。

225

解 按照算法 14.2:

- (1) 选择两个样本点作为类的中心。假设选择 $m_1^{(0)} = x_1 = (0, 2)^T$, $m_2^{(0)} = x_2 = (0, 0)^T$ 。
(2) 以 $m_1^{(0)}$, $m_2^{(0)}$ 为类 $G_1^{(0)}$, $G_2^{(0)}$ 的中心, 计算 $x_3 = (1, 0)^T$, $x_4 = (5, 0)^T$, $x_5 = (5, 2)^T$ 与 $m_1^{(0)} = (0, 2)^T$, $m_2^{(0)} = (0, 0)^T$ 的欧氏距离平方。
(a) 对 $x_3 = (1, 0)^T$, $d(x_3, m_1^{(0)}) = 5$, $d(x_3, m_2^{(0)}) = 1$, 将 x_3 分到类 $G_2^{(0)}$ 。
(b) 对 $x_4 = (5, 0)^T$, $d(x_4, m_1^{(0)}) = 29$, $d(x_4, m_2^{(0)}) = 25$, 将 x_4 分到类 $G_2^{(0)}$ 。
(c) 对 $x_5 = (5, 2)^T$, $d(x_5, m_1^{(0)}) = 25$, $d(x_5, m_2^{(0)}) = 29$, 将 x_5 分到类 $G_1^{(0)}$ 。
(3) 得到新的类 $G_1^{(1)} = \{x_1, x_5\}$, $G_2^{(1)} = \{x_2, x_3, x_4\}$, 计算类的中心 $m_1^{(1)}$, $m_2^{(1)}$:

$$m_1^{(1)} = (2.5, 2.0)^T, \quad m_2^{(1)} = (2, 0)^T$$

- (4) 重复步骤 (2) 和步骤 (3)。将 x_1 分到类 $G_1^{(1)}$, 将 x_2 分到类 $G_2^{(1)}$, x_3 分到类 $G_2^{(1)}$, x_4 分到类 $G_2^{(1)}$, x_5 分到类 $G_1^{(1)}$, 得到新的类 $G_1^{(2)} = \{x_1, x_5\}$, $G_2^{(2)} = \{x_2, x_3, x_4\}$ 。

由于得到的新的类没有改变, 聚类停止。得到聚类结果:

$$G_1^* = \{x_1, x_5\}, \quad G_2^* = \{x_2, x_3, x_4\}$$

朴素贝叶斯

例 4.1 试由表 4.1 的训练数据学习一个朴素贝叶斯分类器并确定 $x = (2, S)^T$ 的类标记 y 。表中 $X^{(1)}$, $X^{(2)}$ 为特征, 取值的集合分别为 $A_1 = \{1, 2, 3\}$, $A_2 = \{S, M, L\}$, Y 为类标记, $Y \in C = \{1, -1\}$ 。

表 4.1 训练数据

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$X^{(1)}$	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$X^{(2)}$	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

解 根据算法 4.1, 由表 4.1 容易计算下列概率:

$$P(Y=1) = \frac{9}{15}, \quad P(Y=-1) = \frac{6}{15}$$

54

$$\begin{aligned} P(X^{(1)}=1|Y=1) &= \frac{2}{9}, & P(X^{(1)}=2|Y=1) &= \frac{3}{9}, & P(X^{(1)}=3|Y=1) &= \frac{4}{9} \\ P(X^{(2)}=S|Y=1) &= \frac{1}{9}, & P(X^{(2)}=M|Y=1) &= \frac{4}{9}, & P(X^{(2)}=L|Y=1) &= \frac{4}{9} \\ P(X^{(1)}=1|Y=-1) &= \frac{3}{6}, & P(X^{(1)}=2|Y=-1) &= \frac{2}{6}, & P(X^{(1)}=3|Y=-1) &= \frac{1}{6} \\ P(X^{(2)}=S|Y=-1) &= \frac{3}{6}, & P(X^{(2)}=M|Y=-1) &= \frac{2}{6}, & P(X^{(2)}=L|Y=-1) &= \frac{1}{6} \end{aligned}$$

对于给定的 $x = (2, S)^T$, 计算

$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1) = \frac{9}{15} \cdot \frac{3}{9} \cdot \frac{1}{9} = \frac{1}{45}$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1) = \frac{6}{15} \cdot \frac{2}{6} \cdot \frac{3}{6} = \frac{1}{15}$$

由于 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大, 所以 $y = -1$ 。