QUANT第二周

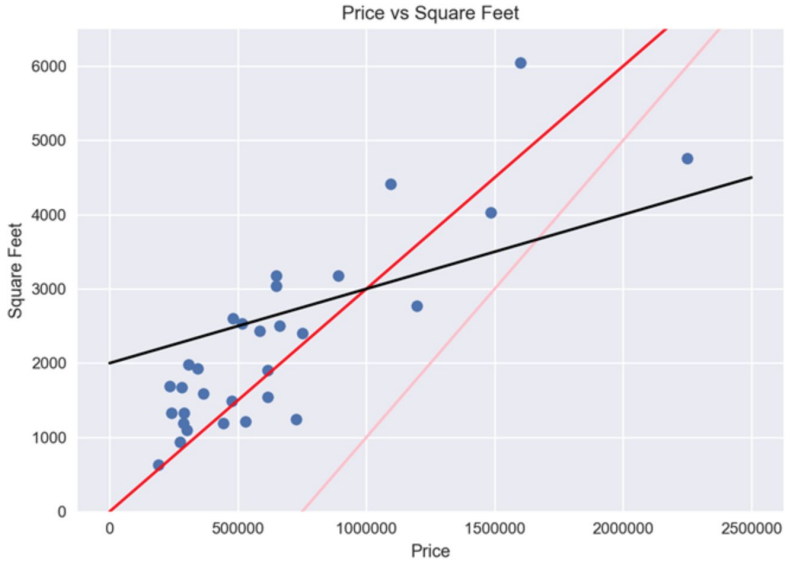
10.26.2019

杨雅迪
王亦婷

目录

线性回归

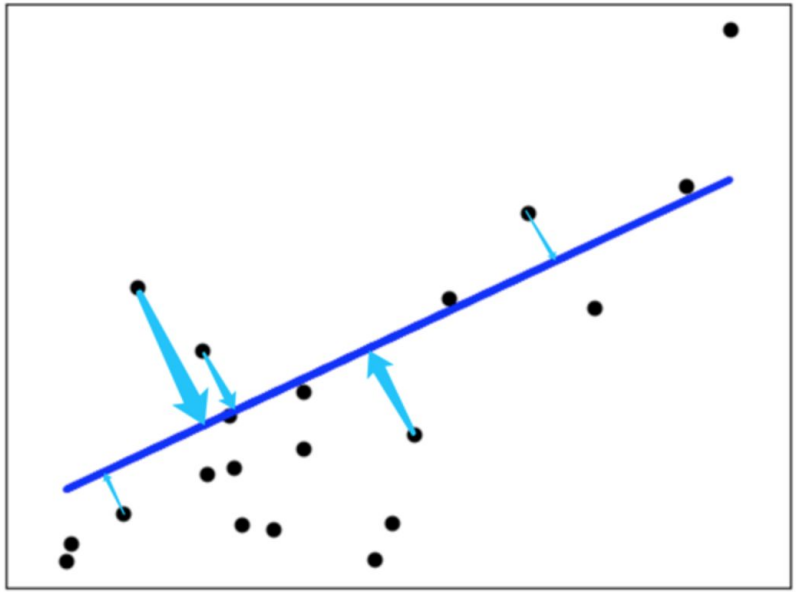
根据房子的居住尺寸来预测房价并得到房价与尺寸之间的线性关系



对于单个对象：$\text{minimize}(h(x_i) - y_i)^2$

对于所有样本：$\text{minimize}\ \frac{1}{2m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$.

均方误差其实计算的是预测点到实际点之间的距离：



欧式距离

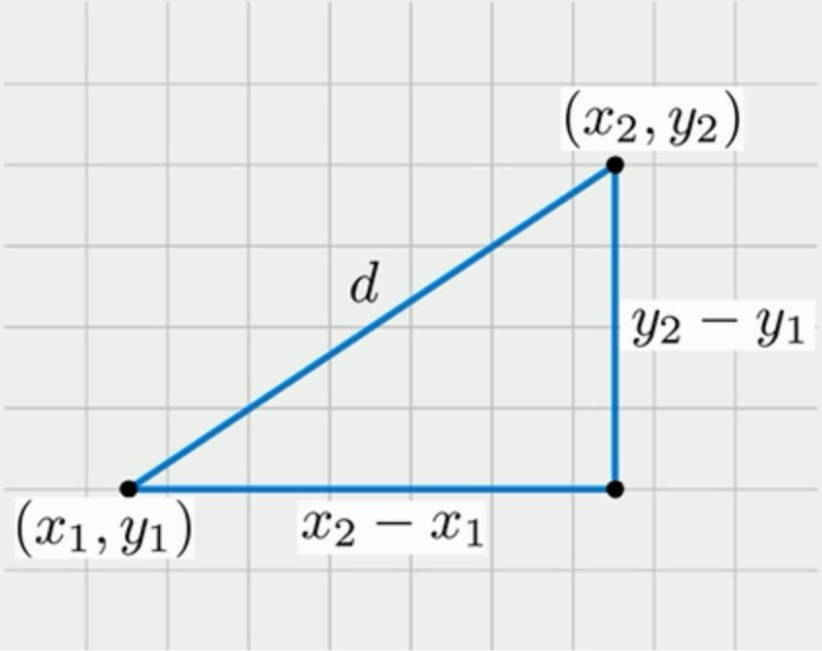
指在m维空间中两个点之间的真实距离。

- 平面上任意两点计算公式(x_1, y_1)、(x_2, y_2)：

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

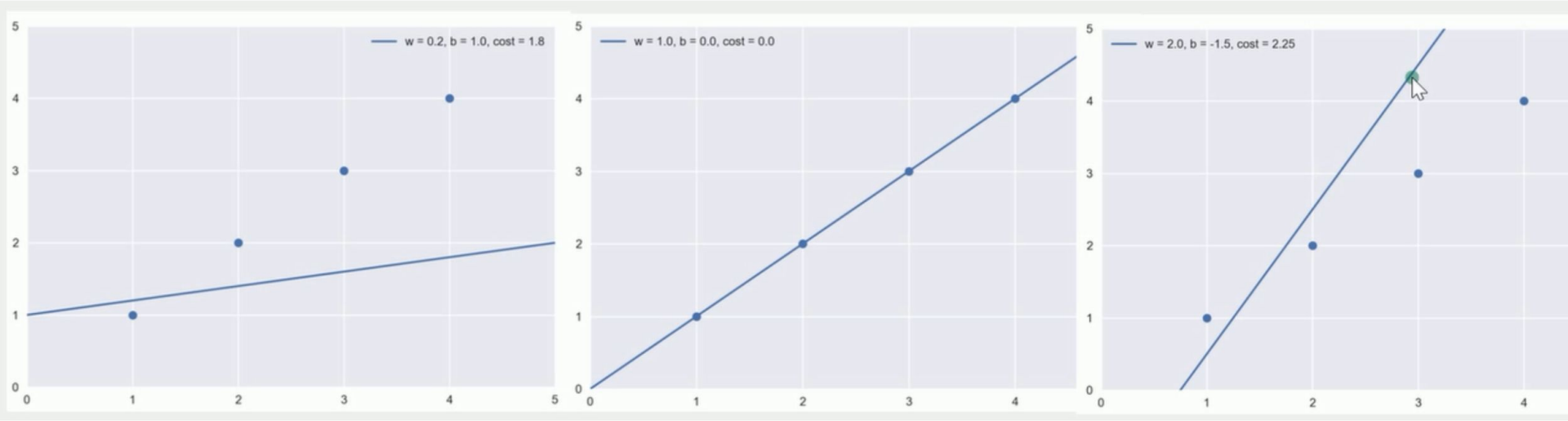
- n维向量间的距离计算公式$(x_1, x_2, \dots, x_n)$、$(y_1, y_2, \dots, y_n)$：

$$dist = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$



拟合曲线

在计算了每个点到各个拟合曲线之间的均方误差后，可以得到使得误差最小的拟合曲线，这样的曲线我们称之为最佳拟合曲线（best fitting line）。对于线性关系显著的数据集，最佳拟合曲线就是我们想要寻找的在该数据下的拟合曲线。我们可以看出最中间的曲线拟合效果最好。



线性回归要点总结

模型：$h(x) = wx + b$

优化参数：w, b

Cost function：$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$

目标：$minimize\ J(w,b)$

多元回归

1.向量化

$$W=\begin{bmatrix}w_1\\w_2\\w_3\\w_4\end{bmatrix},X=\begin{bmatrix}x^1\\x^2\\x^3\\x^4\end{bmatrix}\longrightarrow y:h(x)=W^TX+b$$

2.梯度下降

$$w_i=w_i-\alpha\frac{\partial}{\partial w_i}J(W,b)$$

$$b=b-\alpha\frac{\partial}{\partial b}J(W,b)$$

3.特征缩放（同一量级别）

3.1.比例调节（将数据的特征缩放到[0,1]或[-1,1]之间。缩放到什么范围取决于数据的性质）

$$x'=\frac{x-min(x)}{max(x)-min(x)}$$

3.2标准化

$$x'=\frac{(x-\bar{x})}{\sigma}$$

4.**多项式回归** (拟合曲线)

多项式回归的模型为:

$$h(x)=w_1x^1+w_2x^2+b=w_1x^1+w_2(x^2)^2+b$$

我们也可以选择构造其他的特征:

$$h(x)=w_1x^1+w_2\sqrt{(x^1)}+w_3(x^1)^2+w_4sin(x^1)+b$$

通过选择不同方法构造不同特征我们可以更好的拟合数据

```
# 兼容 python 2, 3
from __future__ import print_function

# 导入相关 python 库
import os
import numpy as np
import pandas as pd

# 设定随机数种子，保证模型输出稳定
np.random.seed(36)

# 使用 matplotlib 绘图
%matplotlib inline
import matplotlib
import seaborn
import matplotlib.pyplot as plt
```

梯度下降求解线性回归模型

本章编程内容主要是对梯度下降求解回归模型的一个简单的应用

```
import numpy as np

X = 2 * np.random.rand(100, 1)  #随机产生100个0-2之间的数，在我们没有实际数据时可以此种方式产生数据
y = 4 + 3 * X + .2*np.random.randn(100, 1)
```

对目标数据绘图表示

```
plt.figure(figsize=(10, 7)) #创建一个新的界面，并且设定图形尺寸大小
plt.scatter(X, y)  #绘制散点图
plt.xlabel("$x_1$", fontsize=18) #x轴标签
plt.ylabel("$y$", rotation=0, fontsize=18) #y轴标签
plt.axis([0, 2, 0, 15])  #分别是x和y坐标轴显示的大小
plt.show() #设置好上面的参数了，show你的图形
```

matplotlib.pyplot的官方文档，可以查看各个函数的用法

```
X_b = np.c_[np.ones((100, 1)), X]  #构造特征值
from sklearn.metrics import mean_squared_error
lr = 0.1 #学习率，步长
iterations = 1000 #设置循环次数
m = 100
theta = np.random.randn(2,1)
mse = []
for iteration in range(iterations):
    gradients = 2./m * X_b.T.dot(X_b.dot(theta) - y) #构造梯度下降
    theta = theta - lr * gradients
    preds = X_b.dot(theta)
    mse.append(mean_squared_error(preds, y))
```

sklearn.metrics官方文档

最后得出的拟合曲线

```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]
plt.figure(figsize=(10, 7))
plt.scatter(X, y)
plt.plot(X_new, X_new_b.dot(theta), 'r-', label='prediction')
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc='upper right')
plt.show()
```

KNN（k-nearest neighbours)

KNN回归原理：

1. 计算训练样本和测试样本中每个样本点的距离（常见的距离度量有欧式距离，马氏距离等）；
2. 对上面所有的距离值进行排序；
3. 选前k个最小距离的样本；
4. 根据这k个样本的对应值进行平均，得到最后的预测值；

我们一般最常用的距离函数是欧氏距离，也称作距离。如果和是维欧式空间上的两个点，那它们之间的距离是

$$d_P(x,y) = \left(\sum_{i=1} |x_i - y_i|^P\right)^{1/P}$$

KNN修正：

1.经典k邻域的样本点对预测结果的贡献度是相等的。

2.而一个简单的思想是距离更近的样本点应有更大的相似度，其贡献度应比距离更远的样本点大

修正公式：

$$w_i = 1/||x_i - x||$$

$$max \sum_{x_i \in N_k(x)} w_i * I(y_i = c_j)$$

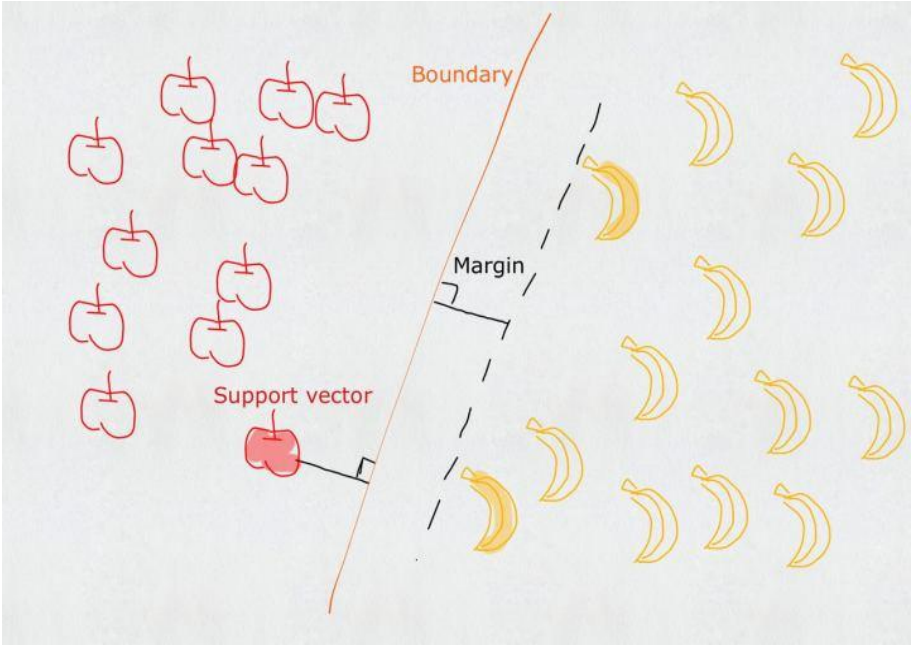
Knn code example

```
from sklearn.neighbors import KNeighborsClassifier

x = [[0],[1],[2],[3]]

y = [0,0,1,1]

neigh = KNeighborsClassifier(n_neighbors=3)  #when k=3, 三个最近的neighbor

neigh.fit(x,y)  #将x,y 拟合建模型

print(neigh.predict([[1.1]]))

print(neigh.predict_proba([[0.9]]))
```

SVM 支持向量机

找到一个线或者平面，用来完美划分linearl separable 的两类。
且这个线或者平面距两类的点一样远。

Note：找到support vector，最近的点



SVM code example

https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/

拉格朗日对偶性

通过给每一个约束条件加上一个拉格朗日乘子（**Lagrange multiplier**）α_i，定义拉格朗日函数（通过拉格朗日函数将约束条件融合到目标函数里去，从而只用一个函数表达式便能清楚的表达出我们的问题）：

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\left(y_i(w^Tx_i+b)-1\right)$$

然后令 $\theta(w)=\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)$

容易验证，当某个约束条件不满足时，例如 $y_i(w^Tx_i+b)<1$，那么显然有 $\theta(w)=\infty$

而当所有约束条件都满足时，则最优值为 $\theta(w)=\frac{1}{2}\|w\|^2$，亦即最初要最小化的量

因此，在要求约束条件得到满足的情况下最小化 $\theta(w)=\frac{1}{2}\|w\|^2$，实际上等价于直接最小化 $\theta(w)$

具体写出来，目标函数变成了：

$$\min_{w,b}\theta(w)=\min_{w,b}\max_{\alpha_i\geq 0}\mathcal{L}(w,b,\alpha)=p^*$$

这里用 p^* 表示这个问题的最优值，且和最初的问题是等价的。如果直接求解，那么一上来便得面对w和b两个参数，而又是不等式约束，这个求解过程不好做。不妨把最小和最大的位置交换一下（需满足**KTT条件**），变成：

$$\max_{\alpha_i\geq 0}\min_{w,b}\mathcal{L}(w,b,\alpha)=d^*$$

交换以后的新问题是原始问题的对偶问题，这个新问题的最优值用 d^* 来表示。而且有 $d^*\leq p^*$

在满足某些条件的情况下，这两者相等，这个时候就可以通过求解对偶问题来间接地求解原始问题。

换言之，之所以从minmax的原始问题 p^*，转化为maxmin的对偶问题，一者因为 d^* 是 p^* 的近似解，二者，转化为对偶问题后，更容易求解。

下面可以先求L 对w、b的极小，再求L 对 α 的极大。

一般地，一个最优化数学模型能够表示成下列标准形式：

$$\min f(\mathbf{x})$$

$$\text{s.t.}\quad h_j(\mathbf{x})=0, j=1,\ldots,p,$$

$$g_k(\mathbf{x})\leq 0, k=1,\ldots,q,$$

$$\mathbf{x}\in\mathbf{X}\subset\Re^n$$

其中，f(x)是需要最小化的函数，h(x)是等式约束，g(x)是不等式约束，p和q分别为等式约束和不等式约束的数量。

同时，得明白以下两点：

1. 凸优化的概念：$\mathcal{X}\subset\mathbb{R}^n$ 为一凸集，$f:\mathcal{X}\to\mathbb{R}$ 为一凸函数。凸优化就是要找出一点 $x^*\in\mathcal{X}$，使得每一 $x\in\mathcal{X}$ 满足 $f(x^*)\leq f(x)$。
2. KKT条件的意义：它是一个非线性规划（Nonlinear Programming）问题能有最优化解法的必要和充分条件。

而KKT条件就是指上面最优化数学模型的标准形式中的最小点 x* 必须满足下面的条件：

1. $h_j(\mathbf{x}_*)=0, j=1,\ldots,p, g_k(\mathbf{x}_*)\leq 0, k=1,\ldots,q,$
2. $\nabla f(\mathbf{x}_*)+\sum_{j=1}^{p}\lambda_j\nabla h_j(\mathbf{x}_*)+\sum_{k=1}^{q}\mu_k\nabla g_k(\mathbf{x}_*)=\mathbf{0},$

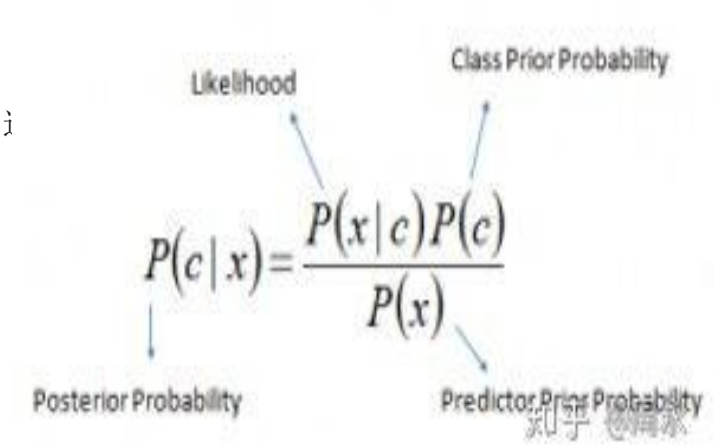
$\lambda_j\neq 0,\ \mu_k\geq 0,\ \mu_k g_k(\mathbf{x}_*)=0.$

3. 核函数：在前面的讨论中，我们假设训练样本是线性可分的，即存在一个划分超平面能将 训练样本正确分类。然而在现实任务中，原始样本空间内也许并不存在一个能正确划分两 类样本的超平面。对于这样的问题，svm的处理方法是选择一个核函数k，将样本从原始空间映射到一个更高维的特征空间，使得样本在这个特征空间内线性可分

Naive bayes 朴素贝叶斯

统计学概念：

x2…xn均发生条件下事件c的概率，



Note：朴素贝叶斯算法针对多元分类问题，假设在事件x1、x2…xn均发生条件下事件c的概率，

假设x1,x2, xn互相独立

那么P(x|c)的概率就可以计算为：P(x|c)= P(x1|c)* P(x2|c)*…P(x3|c)。

Naive bayes 公式推导

公式化推导:

$$P(Y=c_k|X=x)=\frac{P(X=x|Y=c_k)P(Y=c_k)}{P(X=x)}$$

其中$P(X=x)=\sum_k P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\middle|Y=c_k\right)$

其中$P(X=x|Y=c_k)=\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)$

带入得$P(Y=c_k|X=x)=\frac{P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)}{\sum_k P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\middle|Y=c_k\right)}$

目标函数可定义为:

$$y=arg\max_{c_k} P(Y=c_k|X=x)$$

$$=arg\max_{c_k}\frac{P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)}{\sum_k P(Y=c_k)\prod_j P\left(X^{(j)}=x^{(j)}\middle|Y=c_k\right)}$$

对于每一个c_k相同

等价为:

$$y=arg\max_{c_k} P(Y=c_k)\prod_j P(X^{(j)}=x^{(j)}|Y=c_k)$$

后验概率最大化等价推导

假设0-1损失函数:

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

则期望损失函数为:

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

$$=E_X\sum_{k=1}^{K}\left[L(Y,f(X))\right]P(c_k|X)$$

$$f(x)=arg\min_{y\in y}\sum_{k=1}^{K}L(c_k,y)P(c_k|X=x)$$

$$=\arg\min_{y\in y}\sum_{k=1}^{K}P(y\neq c_k|X=x)$$

$$=\arg\min_{y\in y}(1-P(y=c_k|X=x))$$

$$=\arg\max_{y\in y}P(y=c_k|X=x)$$

$$=\arg\max_{c_k}P(c_k|X=x)$$

Naive Bayes code example

https://blog.csdn.net/csqazwsxedc/article/details/69488938

例子：通过n1,n2,....等特征，通过可能性判断m特征（只有0/1两个可能性）

树模型

- Step1：特征选择
- Step2：决策树生成
- Step3：决策树减枝

Step1：特征选择

思考一个问题：我们以什么依据去选？如果是你，你认为什么特征是重要的

特征选择的作用：决定选取哪些特征来划分特征空间

信息增益：

在进一步讨论特征选择之前，首先我们需要根据之前数学基础信息论的知识，定义一个概念：**信息增益**

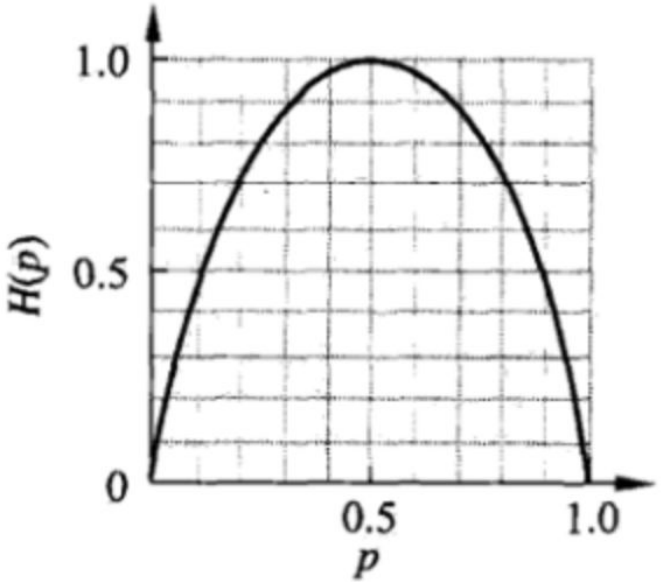
举例：

当随机变量只取两个值，例如1，0时，分布：

$P(X=1)=p$，$P(X=0)=1-p$

熵：$H(X) = -plog_2p - (1-p)log_2(1-p)$

熵变化的图形化示例：



帅？	性格好？	身高？	上进？	嫁与否
帅	不好	矮	不上进	不嫁
不帅	好	矮	上进	不嫁
帅	好	矮	上进	嫁
不帅	爆好	高	上进	嫁
帅	不好	矮	上进	不嫁
帅	不好	矮	上进	不嫁
帅	好	高	不上进	嫁
不帅	好	中	上进	嫁
帅	爆好	中	上进	嫁
不帅	不好	高	上进	嫁
帅	好	矮	不上进	不嫁
帅	好	矮	不上进	不嫁

信息熵是代表随机变量的复杂度(不确定度)

条件熵代表在某一个条件下，随机变量的复杂度(不确定度)

而我们的信息增益恰好是:信息熵-条件熵。

信息增益代表了在一个条件下，信息复杂度(不确定性)减少的程度。

可以求得随机变量X（嫁与不嫁）的信息熵为：

嫁的个数为6个，占1/2，那么信息熵为-1/2log1/2-1/2log1/2 = -log1/2=0.301

现在假如我知道了一个男生的身高信息。

身高有三个可能的取值{矮，中，高}

矮包括{1,2,3,5,6,11,12}，嫁的个数为1个，不嫁的个数为6个

中包括{8,9}，嫁的个数为2个，不嫁的个数为0个

高包括{4,7,10}，嫁的个数为3个，不嫁的个数为0个

先回忆一下条件熵的公式如下：

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X=x)$$

我们先求出公式对应的:

H(Y|X = 矮) = -1/7log1/7-6/7log6/7=0.178

H(Y|X=中) = -1log1-0 = 0

H(Y|X=高） = -1log1-0=0

p(X = 矮) = 7/12,p(X =中) = 2/12,p(X=高) = 3/12

则可以得出条件熵为：

7/12*0.178+2/12*0+3/12*0 = 0.103

那么我们知道信息熵与条件熵相减就是我们的信息增益，为

0.301-0.103=0.198

所以我们可以得出我们在知道了身高这个信息之后，信息增益是0.198

决策树中的过拟合问题

回忆在回归的时候，讲到的过拟合问题，这种在决策树中也是会遇到的，往往树的规模越大，在模型训练中的拟合效果虽然会更好，但模型的泛化能力会下降。

在这样的情况下我们就需要做Step3：决策树减枝

Step3:决策树分类剪枝（Pruning）：

实现方式：极小化决策树整体的损失函数（Loss Function）或者代价函数（Cost Function）

函数定义：设树T的叶子节点数为|T|，t是树T的叶子节点，记该叶子节点有N_t个样本点，其中k类的样本点有N_{tk}个，则定义损失函数为：

$$C_\alpha(T)=\sum_{i=1}^{|T|}N_tH_t(T)+\alpha|T|\text{，其中}H_t(T)=-\sum_k\frac{N_{tk}}{N_t}log\frac{N_{tk}}{N_t}$$

决策树分类算法的混淆矩阵

	预测的类		
实际的类	类 = 1	类 = 0	
类 = 1	TP	FN	P
类 = 0	FP	TN	N

符号定义：

P (Positive Sample)：正例的样本数量。

N(Negative Sample)：负例的样本数量。

TP(True Positive)：正确预测到的正例的数量。

FP(False Positive)：把负例预测成正例的数量。

FN(False Negative)：把正例预测成负例的数量。

TN(True Negative)：正确预测到的负例的数量。

2.准确率

准确率（accuracy）计算公式如下所示：

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\text{all data}} \tag{1}$$

准确率表示预测正确的样本（TP和TN）在所有样本（all data）中占的比例。

在数据集不平衡时，准确率将不能很好地表示模型的性能。可能会存在准确率很高，而少数类样本全分错的情况，此时应选择其它模型评价指标。

3.精确率（查准率）和召回率（查全率）

positive class的精确率（precision）计算公式如下：

$$\text{precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{预测为positive的样本}} \tag{2}$$

positive class的召回率（recall）计算公式如下：

$$\text{recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{真实为positive的样本}} \tag{3}$$

positive class的精确率表示在预测为positive的样本中真实类别为positive的样本所占比例；positive class的召回率表示在真实为positive的样本中模型成功预测出的样本所占比例。

positive class的召回率只和真实为positive的样本相关，与真实为negative的样本无关；而精确率则受到两类样本的影响。

信息增益

首先需要介绍一下熵的概念，这是一个物理学概念，表示“一个系统的混乱程度”。系统的不确定性越高，熵就越大。假设集合中的变量X={x1,x2…xn}，它对应在集合的概率分别是P={p1,p2…pn}。那么这个集合的熵表示为：

$$E(X) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

https://zhuanlan.zhihu.com/p/26596036
https://blog.csdn.net/guomutian911/article/details/78599450

树模型

https://static.dcxueyuan.com/content/disk/train/other/f627ee65-5b03-4ada-9596-09febe05a95f.html

https://blog.csdn.net/csqazwsxedc/article/details/65697652

优缺点

SVM的优缺点

优点：

(1)非线性映射是SVM方法的理论基础,SVM利用内积核函数代替向高维空间的非线性映射；

(2)对特征空间划分的最优超平面是SVM的目标,最大化分类边际的思想是SVM方法的核心；

(3)支持向量是SVM的训练结果,在SVM分类决策中起决定作用的是支持向量。

(4)SVM 是一种有坚实理论基础的新颖的小样本学习方法。它基本上不涉及概率测度及大数定律等,因此不同于现有的统计方法。从本质上看,它避开了从归纳到演绎的传统过程,实现了高效的从训练样本到预报样本的“转导推理”,大大简化了通常的分类和回归等问题。

缺点：

(1) SVM算法对大规模训练样本难以实施

由于SVM是借助二次规划来求解支持向量，而求解二次规划将涉及m阶矩阵的计算（m为样本的个数），当m数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间。

(2) 用SVM解决多分类问题存在困难

经典的支持向量机算法只给出了二类分类的算法，而在数据挖掘的实际应用中，一般要解决多类的分类问题。

KNN优缺点：

- **优点：精度高、对异常值不敏感、无数据输入假定。**
- **缺点：计算复杂度高、空间复杂度高。**
- **适用数据范围：数值型和标称型**

决策树

- 优点

1. 不需要任何领域知识或参数假设。
2. 适合高维数据。
3. 简单易于理解。
4. 短时间内处理大量数据，得到可行且效果较好的结果。

- 缺点

1. 对于各类别样本数量不一致数据，信息增益偏向于那些具有更多数值的特征。
2. 易于过拟合,特别是在特征多的情况下，容易引入噪声特征。
3. 忽略属性之间的相关性。
4. 不支持在线学习

KNN V.S. SVM

svm，就像是在河北和北京之 间有一条边界线，如果一个人居住在北京一侧就预测为北京人，在河北一 侧，就预测为河北人。但是住在河北的北京人和住在北京的河北人就会被 误判。

knn，就是物以类聚，人以群分。如果你的朋友里大部分是北京人，就 预测你也是北京人。如果你的朋友里大部分是河北人，那就预测你是河北人。不管你住哪里。

1.KNN对每个样本考虑。

SVM是要去找一个函数把达到样本可分。

2.朴素的KNN是不会去自主学习特征权重的。

SVM的本质就是在找权重。

3.KNN如果样本维度太高直接卡死。

SVM处理高维数据比较优秀。

KNN 与 SVM 的区别是什么？ - 知乎用户的回答 - 知乎

https://www.zhihu.com/question/19887252/answer/32932848

决策树 v.s. 朴素贝叶斯

决策树算法:决策树的主函数:本质上是个递归函数,该函数主要功能是根据某种规则生长出决策树的各个分支节点,并根据终止条件结束算法。

a) 输入需要分类的数据集和类别标签

b) 根据某种分类规则得到最优的划分特征,并创建特征的划分节点——计算最优特征子函数

c) 按照该特征的每个取值划分数据集为若干部分——划分数据集子函数

d) 根据划分子函数的计算结果构建出新的节点,作为树生长出的新分支

e) 检验是否符合递归终止条件

f) 将划分的新节点包含的数据集和类别标签作为输入,递归执行上述步骤

朴素贝叶斯:

根据贝叶斯定理,对一个分类问题,给定样本特征x,样本属于类别y的概率是

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

只要分别估计出,特征xi在每一类的条件概率就可以了。类别y的先验概率可以通过训练集算出,同样通过训练集上的统计,可以得出对应每一类上的,条件独立的特征对应的条件概率向量。

没有最好的分类器，只有最合适的分类器。

各种机器学习算法的应用场景分别是什么（比如朴素贝叶斯、决策树、K 近邻、SVM、逻辑回归最大熵模型）？ - xyzh的回答 - 知乎

https://www.zhihu.com/question/26726794/answer/151282052