

## 倾斜数据集的误差指标

某个类别的样本数只有另一个类别的1/10，这个数据集就是倾斜的。这种情况也常常出现在诸如诈骗检测、医学诊断、罕见事件预测等领域中，因为这些问题中正例的数量通常比负例少得多。

## 精确率（找的对）和召回率（找的全）的权衡

### Trading off precision and recall

Logistic regression:  $0 < f_{\vec{w},b}(\vec{x}) < 1$

→ Predict 1 if  $f_{\vec{w},b}(\vec{x}) \geq 0.5$  (crossed out)  $\rightarrow 0.3$  (blue)

→ Predict 0 if  $f_{\vec{w},b}(\vec{x}) < 0.5$  (crossed out)  $\rightarrow 0.3$  (blue)

Suppose we want to predict  $y = 1$  (rare disease) only if very confident.

→ higher precision, lower recall.

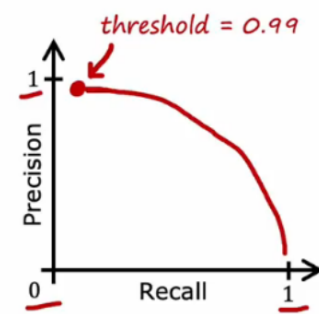
Suppose we want to avoid missing too many cases of rare disease (when in doubt predict  $y = 1$ )

→ lower precision, higher recall.

More generally predict 1 if:  $f_{\vec{w},b}(\vec{x}) \geq \text{threshold}$ .

precision =  $\frac{\text{true positives}}{\text{total predicted positive}}$

recall =  $\frac{\text{true positives}}{\text{total actual positive}}$



The graph shows Precision on the y-axis and Recall on the x-axis, both ranging from 0 to 1. A red curve starts at (0,1) and ends at (1,0). A red dot on the curve is labeled 'threshold = 0.99' with an arrow pointing to it.

高精确率：判定为true的置信度高

高召回率：判定为false的置信度高

高准确率：分类正确的置信度高

## 置信度

置信度是指一个统计推断中，对于一个参数或者假设所得到的推断结果的可信程度或者确定性程度。置信度通常使用置信区间来度量，表示参数或假设在一定置信水平下的可能取值范围。

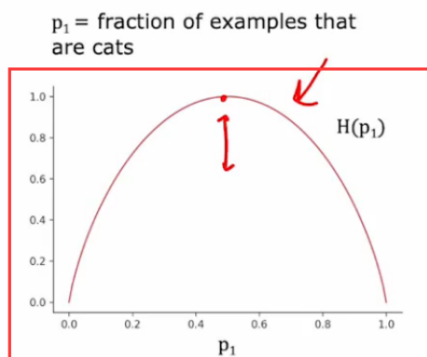
置信度高表示推断所得的结果越可信，即在一定置信水平下，得到的置信区间越小，说明参数或假设的可能取值范围越小，结果越精确可靠。例如，如果某个样本的置信区间为 $[2, 6]$ ，则在95%的置信水平下，该样本的真实值有95%的可能在区间 $[2, 6]$ 内。如果置信度更高，比如99%的置信水平，则样本的真实值有99%的可能在更小的区间内，如 $[2.5, 5.5]$ 。

## 决策树

需要最大限度地提高纯度（或最小限度地减少不纯）（纯度：近似于正确率）

熵:

# Entropy as a measure of impurity



$$p_0 = 1 - p_1$$

$$\begin{aligned} H(p_1) &= -p_1 \log_2(p_1) - p_0 \log_2(p_0) \\ &= -p_1 \log_2(p_1) - (1 - p_1) \log_2(1 - p_1) \end{aligned}$$

可以发现，当正例和反例比例相等时，即 $p_+ = p_- = 0.5$ 时，此时的熵取最大值1，表明此时样本集合最不确定，样本集合越不纯，熵值越大；而当样本集合中只包含一类样本（即纯样本集合）时，熵值为0，表明此时样本集合已经完全确定，很纯。

在决策树算法中，熵常常被用来选择最优划分点，即在样本的某个特征上根据熵的变化选取最优的划分点，使得划分后的样本子集中的熵最小。

## 信息增益

在决策树算法中，我们希望通过选择最优特征来划分数据集，从而使划分后的数据集更加纯净，即熵减少。而这个熵减少的量就被称为信息增益

信息增益是一种用于衡量一个特征对分类任务的贡献程度的指标，通常用于决策树算法中。在决策树算法中，我们需要在每个节点上选择一个特征，使得选定特征后能够最大程度地提高数据的纯度（即分类的准确性）。信息增益的计算方法是：首先计算数据集的熵，然后计算选定特征后的条件熵，两者相减即为该特征的信息增益。信息增益越大，代表该特征对分类任务的贡献越大。

## one-hot编码

它将每个类别值转换为一个长度为N的二进制向量，其中N是类别的总数。（01代替特征）

## 回归树

回归树是一种用于预测连续目标变量的决策树模型。它通过将数据集划分成若干个区域，来对每个区域内的目标值进行估计

从根节点开始，选择最佳特征及其切分点，将数据分割成不同的子集

在**划分时**，不去计算信息增益，不去衡量纯度，不如**减少数据的方差**，类似信息增益，**测量的是方差的减小，选择方差减少最大的那个特征分类标准**

## 有放回抽样

构造多个随机的训练集

## 随机森林

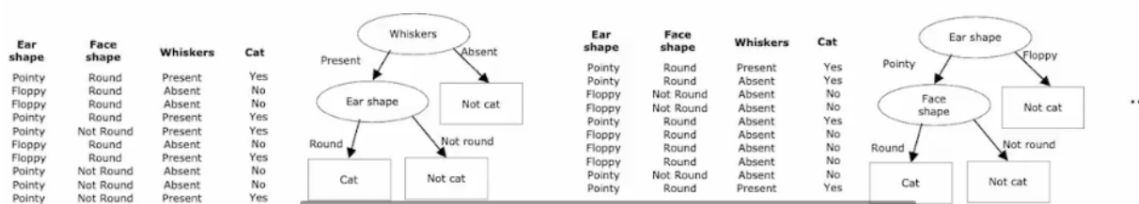
### Generating a tree sample

Given training set of size  $m$

For  $b = 1$  to  $B$ :  $B$  bag 袋装决策树

Use sampling with replacement to create a new training set of size  $m$

Train a decision tree on the new dataset



### Randomizing the feature choice

At each node, when choosing a feature to use to split, if  $n$  features are available, pick a random subset of  $k < n$  features and allow the algorithm to only choose from that subset of features.

$$k = \sqrt{n}$$

随机森林是一种集成学习算法，它由多个决策树组成，每个决策树之间相互独立

1 随机选取  $n$  个样本作为训练集，对于每个样本，随机选取  $k$  个特征进行决策树的构建。通常情况下， $n$  取样本总数的 70% 到 80%， $k$  取特征总数的平方根。

2 对于每个节点，随机选取  $k$  个特征进行划分，并在选取的特征中找到最优划分特征。这里的最优划分是通过计算信息增益或 Gini 系数等指标得出的。

3 重复步骤 1 和步骤 2，构建多棵决策树。

4 预测时，将新数据输入到每个决策树中，得到每棵树的预测结果，然后根据投票法或平均值等方法，得到随机森林的最终预测结果。

随机森林算法具有良好的泛化性能和抗噪性能，通常被用于分类和回归任务。

## XGBoost

在构建下一个决策树时，把更多的注意力放在做得不好的例子上，不是以相等的  $(1/m)$  概率从所有的例子中选取，而是更有可能选取以前训练的树所错误分类的例子

XGBoost的训练过程是一个逐步迭代的过程。它通过不断训练多棵决策树，并使用梯度下降来对每棵树的权重进行优化，以最小化损失函数。

具体来说，XGBoost在构建每棵树时，会先构建一个根节点，然后逐步将数据集分配到子节点中。每次分配数据集时，XGBoost都会根据一个指定的评价函数来选择最佳的特征进行分裂，并计算每个子节点的权重。通过这种方式，XGBoost可以在不断迭代中构建多棵树，并综合它们的预测结果来提高模型的准确率

XGBoost库是一个开源的机器学习库，专注于梯度提升决策树（GBDT）算法，广泛应用于分类和回归任务。

## 何时使用决策树、神经网络

- 决策树
  1. 决策树可以很好处理**表格数据（结构化数据）**
  2. 训练很快
  3. 小型的决策树具有可解释性
- 神经网络
  1. 适用于所有类型的数据，表格和非结构化数据
  2. 可能比较慢
  3. 可以迁移学习、预训练

# Decision Trees vs Neural Networks

## Decision Trees and Tree ensembles

- Works well on tabular (structured) data
- Not recommended for unstructured data (images, audio, text)
- Fast
- Small decision trees may be human interpretable

## Neural Networks

- Works well on all types of data, including tabular (structured) and unstructured data
- May be slower than a decision tree
- Works with transfer learning
- When building a system of multiple models working together, it might be easier to ~~string together multiple~~ **combine multiple** neural networks

© 2021 @何梦吉他

## 聚类算法

第一步，把每个点分配给簇质心，分配给离它最近的簇质心；第二步，把每个簇质心分配给它所有点的平均位置，直到簇质心没有变化

## K-means algorithm

Randomly initialize  $K$  cluster centroids  $\mu_1, \mu_2, \dots, \mu_K$

Repeat {

*# Assign points to cluster centroids*

for  $i = 1$  to  $m$

$c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid closest to  $x^{(i)}$

*# Move cluster centroids*

for  $k = 1$  to  $K$

$\mu_k :=$  average (mean) of points assigned to cluster  $k$

}

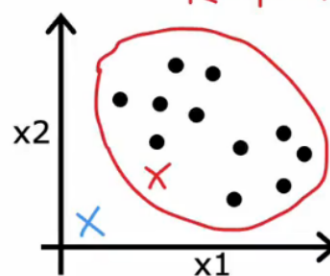
$\mu_1, \mu_2$

$x^{(1)}, x^{(2)}, \dots, x^{(n)}$

$n = 2$

$K = 2$

$K = K - 1$  ✓



如果没有点分配给它，则消除一个簇。

$C_i$  表示当前训练样本  $x_i$  所在的簇的索引  $1 \sim K$ ，是每个样本的分类标识

$\mu_k$  是簇质心的位置

当  $k = C_i$  时， $\mu_{C_i}$  是样本  $x_i$  被分配到的簇，这个簇的簇质心的位置

$m$  是所有样本个数

## K-means optimization objective

$c^{(i)}$  = index of cluster  $(1, 2, \dots, K)$  to which example  $x^{(i)}$  is currently assigned

$\mu_k$  = cluster centroid  $k$

$\mu_{c^{(i)}}$  = cluster centroid of cluster to which example  $x^{(i)}$  has been assigned

**Cost function**

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$\min_{c^{(1)}, \dots, c^{(m)}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$



## 聚类算法的损失函数

### Cost function for K-means

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Repeat {

*# Assign points to cluster centroids*

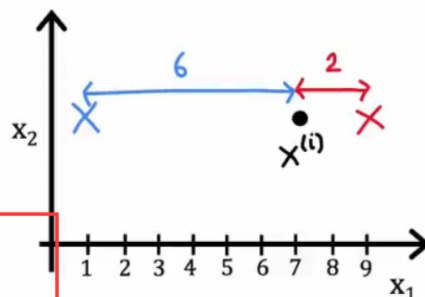
for  $i = 1$  to  $m$

$c^{(i)} :=$  index of cluster  
centroid closest to  $x^{(i)}$

*# Move cluster centroids*

for  $k = 1$  to  $K$

$\mu_k :=$  average of points in cluster  $k$



算法每一步会更新样本所属簇的类别即 $c_i$ 的值、  
或者移动簇质心的位置 $\mu_k$ ，以降低代价函数

## 初始化K-means

K-means 聚类算法第一步是为簇质心 $\mu_j$   $\mu_k$ 选择随机位置作为初始值

如果初始值选的不好，会陷入局部最小值，多次运行K-means，计算他们的J，选最小的那个

## 如何选择聚类的簇的数量K

肘部法则（一般不用）

运行具有各种K值的k-means，绘制代价函数-K的函数，找到肘关节位置，但比较主观

## 异常检测

假设已知数据集由多个高斯分布组成，每个高斯分布对应数据集中的一个簇，那么可以通过最大化似然函数来估计这些高斯分布的参数，从而得到整个数据集的概率分布模型。具体来说，对于一个新的样本数据，可以计算它在该模型下的概率密度值，如果该值较小，则认为该样本是异常值。

## 强化学习

总的来说，强化学习是一种机器学习方法，使智能体通过与环境的互动来学习如何做出决策。其基本思想是通过试错的方式，智能体在执行动作后根据得到的奖励反馈来优化其策略，从而最大化长期收益

在数学中，矩阵特征值和特征向量是很重要的概念，它们可以帮助我们理解和分析矩阵的性质和行为。其中，特征值是一个标量，表示矩阵沿着某个方向的缩放比例，而特征向量则是一个非零向量，表示在这个方向上的向量不变，只是被缩放了。

在机器学习中，我们通常会对数据进行特征提取和降维，其中一种常见的方法是使用主成分分析（Principal Component Analysis, PCA）。PCA通过对数据进行矩阵分解，将原始数据转化为一组线性无关的主成分，以实现数据降维的目的。

## 一些Scikit-Learn库

```
LogisticRegression().fit(X, y) #在训练数据上拟合这个模型
y_pred = lr_model.predict(X) #预测
lr_model.score(X, y) #计算在训练集上的准确率
```

到这里为止已经差不多了解了机器学习的很多基础算法和概念，也有了一点代码基础，再回头看李沐的视频障碍也少了很多，现在开始继续看李沐的教程，旨在了解一些经典的神经网络算法和大型神经网络案例，深化自己对深度学习的了解。

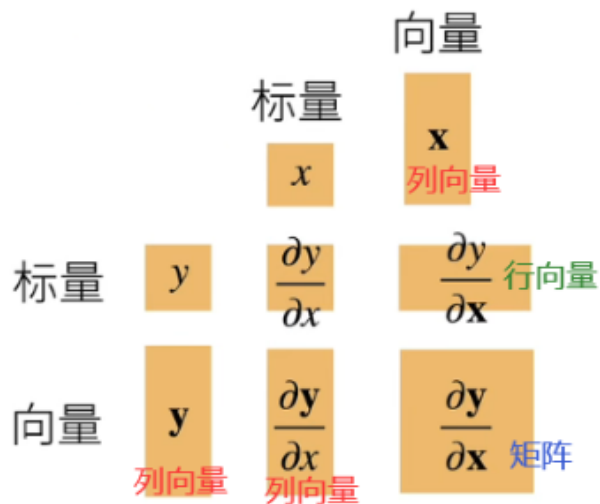
## 又从线性回归开始

### 广播原则

在深度学习中，广播原则（Broadcasting）是指在执行数组或张量运算时，较小的数组或张量可以“扩展”以匹配较大数组或张量的形状，从而允许不同形状的数组进行操作。

### 矩阵计算

将导数拓展到向量



CSDN @chenxiaowai\_

$y$	$a$	$x$	$Ax$	$x^T A$
$\frac{\partial y}{\partial x}$	$0$	$I$	$A$	$A^T$

$y$	$au$	$Au$	$u + v$
$\frac{\partial y}{\partial x}$	$a \frac{\partial u}{\partial x}$	$A \frac{\partial u}{\partial x}$	$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial x}$

求导改转置这里尤其注意

## 自动求导

样例：

假设  $X \in \mathbb{R}^{m \times n}$ ,  $w \in \mathbb{R}^n$ ,  $y \in \mathbb{R}^m$

$$z = \|Xw - y\|^2$$

计算  $\frac{\partial z}{\partial w}$

$$\begin{aligned} \frac{\partial z}{\partial w} &= \frac{\partial z}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial w} \\ &= \frac{\partial \|b\|^2}{\partial b} \frac{\partial a - y}{\partial a} \frac{\partial Xw}{\partial w} \end{aligned}$$

分解

$$\begin{aligned} a &= Xw \\ b &= a - y \\ z &= \|b\|^2 \end{aligned}$$

$$\begin{aligned} &= 2b^T \times I \times X \\ &= 2(Xw - y)^T X \end{aligned}$$



- 链式法则:  $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n} \frac{\partial u_n}{\partial u_{n-1}} \dots \frac{\partial u_2}{\partial u_1} \frac{\partial u_1}{\partial x}$
- 正向累积  $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n} \left( \frac{\partial u_n}{\partial u_{n-1}} \left( \dots \left( \frac{\partial u_2}{\partial u_1} \frac{\partial u_1}{\partial x} \right) \right) \right)$
- 反向累积、 又称反向传递

$$\frac{\partial y}{\partial x} = \left( \left( \left( \frac{\partial y}{\partial u_n} \frac{\partial u_n}{\partial u_{n-1}} \right) \dots \right) \frac{\partial u_2}{\partial u_1} \right) \frac{\partial u_1}{\partial x}$$

CSDN @chenxiaowai\_

### 补充一下dot乘法

- 对于向量, `dot` 计算内积。
- 对于矩阵, `dot` 计算矩阵乘法。

知识点学完了还剩下SVM、决策树、聚类算法的lab代码没有看, 争取每周做一个

沾室友的光被拉去打一个类似大创的比赛, 他有一个基于微调llama-3的知识点预测大模型的项目, 希望能从中学到一点大模型实际运用的知识