

# machine learning 笔记

徐世桐

## 1 基础定义

**二元分类**: 输出分类个数为 2

**多元分类**: 输出分类个数不限

*one - versus - the - rest* OvR: 计算属于每一分类的可能性, 取可能性最大的分类为输出分类

*one - versus - one* OvO: 对所有分类两两使用二元分类, 每一分类器训练只需一部分数据

**multilabel 多标签分类**: 目标检测, 对一图像中的物体加 label

**multioutput 多类分类**: 多标签分类, 每一标签可包含多种信息

**learning schedule**: 根据迭代次数更新学习率

**early stopping**: 提早结束训练

对于每一 epoch, 当验证集 MSE 值增高时, 证明开始 overfit, 停止训练

即在 epoch-error 图中泛化误差最低时停止训练

在训练中使用正则化代价函数, 训练结束后测试中代价函数不使用正则化项

## 2 数学计算

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \bar{x})^2$$

**rigid regression**: 回归方法,  $J(\theta) = \text{MSE}(\theta) + \frac{\alpha}{2} \sum_i \theta_i^2$

降低所有权重值

**lasso regression**: 回归方法,  $J(\theta) = \text{MSE}(\theta) + \alpha \sum_i |\theta_i|$

降低不重要的权重值

**elastic net**: 回归方法,  $J(\theta) = \text{MSE}(\theta) + \gamma \alpha \sum_i |\theta_i| + (1 - \gamma) \frac{\alpha}{2} \sum_i \theta_i^2$

**Normal Equation**:  $\hat{\theta} = (X^T X)^{-1} X^T y$

直接得到权重  $\hat{\theta}$ , 适用于仅有一个输出值的模型

$X$  为 (批量大小, 参数个数) 输入矩阵,  $y$  为 (批量大小, ) 向量

当  $X^T X$  无逆矩阵时, 用 pseudo inverse  $\hat{\theta} = X^+ y$

**pseudo inverse**:

对矩阵  $X = USV^T$ , pseudo inverse  $X^+ = VS^+U^T$ 。  $S^+$  求法:

1. 对所有  $S$  元素, 接近 0 的值赋为 0
2. 对所有非零元素取倒数
3. 取矩阵转置, 得到  $S^+$

**log loss**: 代价函数

$$J(\theta) = -\frac{1}{|B|} \sum_{i=1}^{|B|} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$$

标签值  $y^{(i)}$  为离散 1/0 值, 计算值  $\hat{p}^{(i)} \in [0, 1]$

微分: \*\* 推导 \*\*

$$\frac{dJ(\theta)}{d\theta_j} = \frac{1}{|B|} \sum_{i=1}^{|B|} (\hat{p}^{(i)} - y^{(i)}) x_j^{(i)}$$

**Hinge loss:** 代价函数

$$HingeLoss(y, \hat{y}) = \max(0, 1 - y * \hat{y})$$

应用于 SVM,  $y \in \{0, 1\}$ ,  $\hat{y} \in R$

代表当预测值  $\hat{y}$  和  $y$  同号,  $\hat{y} \geq 1$ , 则预测值和标签匹配, 代价 = 0。否则  $y * \hat{y} < 1$ 。代价值上升

**Gaussian Radial Basis Function RBF:** 一种 similarity function

$$\phi_\gamma(x, l) = \exp(-\gamma \|x - l\|^2)$$

$l$  为 landmark, 即  $\phi_\gamma$  由一样本  $x_i$  和一 landmark 的距离得来

**Lagrange multipliers method 拉格朗日乘数法**

将 有前提的多项式求最值 问题转化为 无前提多项式最值问题

定义:

对输入向量  $W$ ,  $g(W) \geq 0$  为 constrain。目标为在满足  $g(W) \geq 0$  的前提下取  $f(W)$  最值

Lagrange function  $\mathcal{L}(W, \alpha) = f(W) - \alpha(g(W))$

$\alpha$  为需要求解的变量之一, 参与最终计算  $W$  的值。

当有多个 constrain  $g^{(i)}(W)$  时,  $\vec{\alpha}$  为向量, 求偏导对每一  $\vec{\alpha}^{(i)}$  求导

只有当  $\alpha \geq 0$  或每一  $\vec{\alpha}^{(i)} \geq 0$ , 结果才有效

$\vec{\alpha}^{(i)} = 0$  代表对应的 constrain  $g^{(i)}(W)$  为一个 support vector

计算:

$$\text{对每一 } W \text{ 的元素 和 } \alpha \text{ 取偏导, 即向量 } \begin{bmatrix} \frac{d\mathcal{L}(W, \alpha)}{dw_1} \\ \frac{d\mathcal{L}(W, \alpha)}{dw_n} \\ \dots \\ \frac{d\mathcal{L}(W, \alpha)}{dw_n} \\ \frac{d\mathcal{L}(W, \alpha)}{d\alpha} \end{bmatrix}, \text{ 计算向量} = \vec{0} \text{ 时的 } W, \alpha \text{ 取值}$$

### 3 分类模型

**logistic regression:**

判断输入符合每一输出类别的可能性,

前向计算:

$$1. \hat{p} = \sigma(\theta^T x + b)$$

$$2. \hat{y} = 1(\text{if } \hat{p} \geq 0.5)$$

$$= 0(\text{if } \hat{p} < 0.5)$$

代价函数为 log loss

**SVM**

找到分界, 分离多种数据

support vector: 最靠近分界线的样本

hard margin classification 硬性分类: 限制数据必须被分界隔开, 同一类数据不可同时出现在分界 2

端

soft margin classification: 与硬性分类相反, 避免被 outlier 离群值影响

前向计算:  $\hat{p} = f(x_1, x_2, \dots)$ , 其余同 logistic regression

区别:  $f$  可为 polynomial, 非线性函数。可使用 kernel trick

线性分类训练:  $\hat{p} = W^T x + b$ ,  $W$  为参数向量

**硬性分类:**

$\|W\|_2$  代表线性函数斜率

最小化  $\frac{1}{2} W^T W$ , 使得分界平面的斜率最小, 最大化分界线和两种数据的距离

前提: 对每一样本  $i$ ,  $1 \cdot y^{(i)} \hat{p}^{(i)} \geq 1$ , 即标签和计算结果相同

求解: 1. 直接解以上带前提的不等式

当样本数高于参数数量时使用, 由于 dual form 的复杂度为  $O(|S|^2) - O(|S|^3)$ , 直接解复杂度为  $O(|S|)$

2. 使用拉格朗日乘数法得到 dual form, 其中  $\vec{\alpha}$  为向量。  $\mathbf{x}^{(i)}$  为第  $i$  样本的特征值向量  $\mathcal{L} = \frac{1}{2} W^T W - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)} (y^{(i)} \hat{p}^{(i)} - 1)$

使偏导向量为  $\vec{0}$ , 得到 2.  $W = \sum_{i=1}^m \vec{\alpha}_i y^{(i)} \mathbf{x}^{(i)}$ , 3.  $\sum_{i=1}^m \vec{\alpha}_i y^{(i)} = 0$

带入得  $\mathcal{L}(W, \vec{\alpha}) = \frac{1}{2} \sum_{i=1}^{|B|} \sum_{j=1}^{|B|} \vec{\alpha}_i \vec{\alpha}_j y^{(i)} y^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)}$   
 $= \frac{1}{2} \vec{\alpha}^T (\mathbf{x} * y)(\mathbf{x} * y)^T \vec{\alpha} - \sum_{i=1}^{|B|} \vec{\alpha}^{(i)}$

其中  $(\mathbf{x} * y)$  为广播乘法 将训练集矩阵每一样本乘以对应标签值,  $y$  为标签列向量

使用 QP solver 得到使  $\mathcal{L}(W, \vec{\alpha})$  最小,  $\vec{\alpha}^{(i)} \geq 0$  的向量  $\vec{\alpha}$

解  $W$ : 由  $\vec{\alpha}$  带入 2. 式计算,  $\vec{\alpha}$  已被 clamp, 见经验 2.

解  $b$ : 由于所有 support vector  $\mathbf{x}^{(i)}$  满足 1. 式, 则对所有 support vector 计算  $b$  取平均值

$$b = E_{a^{(i)} \geq 0} (y^{(i)} - W^T \mathbf{x}^{(i)})$$

3. 直接进行梯度下降, 代价函数  $J(W, b) = \frac{1}{2} W^T W + \text{const} \sum_i \text{HingeLoss}(y^{(i)}, \hat{p}^{(i)})$

**软性分类:**

最小化  $\frac{1}{2} W^T W + C \sum_{i=1}^{|B|} \zeta_i$

$\zeta_i$  定义第  $i$  样本被忽视为误差样本的可能性,  $C$  定义忽视率相对斜率的权重

前提: 对每一样本  $i$ ,  $y^{(i)} \hat{p}^{(i)} \geq 1 - \zeta^{(i)}$

**非线性分类方法:**

- 使用 polynomial 做  $f$

必须使用拉格朗日乘数法求解, 目的为对  $\phi(x)$  得到线性权重和偏差, 求解使用 dual form, 其中包含  $\phi(a)^T \cdot \phi(b)$  项即可使用 kernel method

权重  $W$  公式不再适用, 由于结果不为线性

偏差  $b = \sum_{\vec{\alpha}^{(i)} \geq 0} y^{(i)} - \sum_{\vec{\alpha}^{(j)} \geq 0} \vec{\alpha}^{(j)} * y^{(j)} * K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

- 使用 similarity function:

选择多个 landmark  $\mathcal{L} = l_1, l_2, \dots, l_n$ , 对每一样本  $x_i$  计算其和每一  $l_j$  的  $\phi_\gamma$  值  $\phi_\gamma(x_i, l_j)$

每个样本用新的向量  $x'_i = \begin{bmatrix} \phi_\gamma(x_i, l_1) \\ \phi_\gamma(x_i, l_2) \\ \dots \\ \phi_\gamma(x_i, l_n) \end{bmatrix}$  表示。新的向量组成训练集, 进行 SVM 训练

**kernel:**

定义: 能够从输入向量  $a, b$ , 不通过计算  $\phi(a), \phi(b)$  直接得到点乘结果  $\langle \phi(a), \phi(b) \rangle$  的函数

例: \*\* 是否通过取 linear 为 phi 得到 kernel 函数 \*\*

linear:  $f(a, b) = a^T b$

polynomial:  $f(a, b) = (\gamma a^T b + r)^d$

poly 的  $\phi(x)$  为对向量  $x$  每一元素进行 poly 运算, 结果向量元素数不变

Gaussian RBF:  $f(a, b) = \exp(-\gamma \|a - b\|^2)$

Sigmoid:  $f(a, b) = \tanh(\gamma a^T b + r)$

经验总结:

1. QP solver 中需限定  $\sum_{i=1}^m \tilde{\alpha}_i y^{(i)} = 0$ , 否则得出  $\hat{\alpha}$  不遵循此等式

2. 当样本有重叠, 仍可使用拉格朗日乘法, 异常样本被分入错误类别。

此时  $\tilde{\alpha}$  包含负值, 对应的样本在计算权重 偏差时被忽略, 即需 clamp 使  $\tilde{\alpha} \geq 0$

若不进行 clamp, 得到的分界仅有略微差别, 不会造成大幅误差。(在线性 非线性分类都有验证)

3. 梯度下降直接得到最优  $W, b$ , 无法通过梯度下降得到  $\tilde{\alpha}$  由于梯度下降忽略限制条件

4. QP solver 需要  $(\mathbf{x} * y)(\mathbf{x} * y)^T$  为 positive definite, 计算时加上对角矩阵  $\text{diag}(\epsilon)$  即可,  $\epsilon$  多取  $10^{-4}$

否则迭代解 QP 时出现 KKT condition not met 或 positive definite 条件不满足

条件不满足时中断得到的  $\tilde{\alpha}$  无法作为有效结果参与后续计算权重和偏差

优先将此矩阵转为 float64 类型, 否则需要  $\epsilon$  较大才能保证 positive definite

## 4 决策树

定义:

节点  $N_i$ :

节点条件: 判断样本进入哪一子节点, 叶节点没有节点条件

sample 属性  $S_i$ : 有多少样本进入  $N_i$  节点, 非满足  $N_i$  节点条件的样本个数

value 属性  $V_i = v_{i1}, \dots, v_{in}$ :  $S_i$  进入节点的样本中  $v_i$  个属于第  $i$  分类

gini 属性  $G_i$ : 数据混杂度,  $G_i = 1 - \sum_{j=1}^n (\frac{v_{ij}}{S_i})^2$

子节点仅有 2 个, 对应节点条件为 true/false 的情况

分类方式: 数据从根节点开始, 根据节点条件传向对应子节点。直到到达叶节点。叶节点中  $V$  属性中最大项即数据分类

**CART algorithm 创建决策树:**

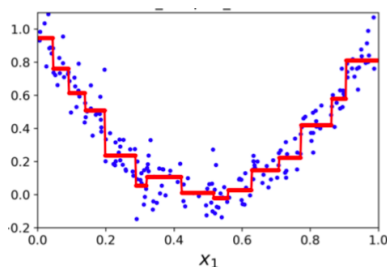
根节点初始化为叶节点, 没有节点条件

对每一叶节点  $S_i$  选取一特征  $k$ , 一特征阈值  $t_k$ , 将样本集分为 2 组  $S_{true}, S_{false}$ 。

选取  $(k, t_k)$  方式: 使代价函数  $J(k, t_k) = \frac{S_{true}}{S_i} G_{true} + \frac{S_{false}}{S_i} G_{false}$  最小

直到决策树层数达到固定上限, 或对所有分组条件  $(k, t_k)$ ,  $J(k, t_k) \geq G_i$

使用决策树进行 regression



输入样本，分类进不同值域

更改：

每一节点 value 值为一常数，为  $S_i$  样本的平均值。

输出值为叶节点的 value，非最大 value 对应的类别

$G_i$  为  $S_i$  样本的方差  $\frac{1}{S_i} \sum_{j=1}^{S_i} (x_i^{(j)} - \bar{x}_i)^2$

## 5 ensemble learning & 随机森林

**ensemble learning**: 使用一组预测机制进行学习，预测机制可为不同算法

**random forest 随机森林**:

训练方法：随机选择  $n$  个训练子集  $s_1, s_2, \dots, s_n \in S$ ，训练  $n$  个决策树  $t_1, \dots, t_n$ 。

前向计算：对  $n$  个树产生的  $n$  个分类结果，选取投票最多的一分类作为结果

训练子集选取：bagging: 子集可重复选取一样本，pasting: 样本不重复

out-off-bag oob 样本：当使用 bagging 选取时，平均只有  $1 - e^{-1}$  样本被选择，余下样本被称为 oob 样本

优化：

random patches 随机贴片：对特征和训练集同时取子集进行训练

random subspace 随机子空间：对特征取子集，对整个总训练集进行训练

extra-trees 极度随机森林：' 使用随机  $t_k$  而不使用最小化数据混杂度的  $t_k$ '

kfeature importance 特征重要性: 对所有取  $k$  为判断条件的节点  $N_i$ , 计算加权平均值  $\sum_i (S_i \text{imprity})$  降低百分比 )

(hypothesis) boosting: 合并多个预测机制据结果的方法

AdaBoost: 串联预测机制，对上一预测机制遗漏的样本加更高权重，进行训练

gradient boosting

## 6 维度下降

根据 manifold assumption, 高维空间中训练集参数点稀疏。则将数据压缩到低维

**principle component analysis PCA**:

对训练集参数矩阵取  $SVDUSV^T$

取  $V$  中前  $d$  个向量  $V' = [v_1, \dots, v_d]$ , 新训练集  $A_{compressed} = A_{origin} V'$

从 新训练集 延展回 原训练集纬度:  $A_{expand} = A_{compressed} V'^T$

**Incremental PCA**: 无需整个训练集存在内存中即可进行 SVD

**kernel PCA**: \*\*

**local linear Embedding LLE**:

对每一样本  $x^{(i)}$  寻找  $k$  个相邻样本 相邻样本 index 的集合称  $C_{x^{(i)}}$

构建  $(|S|, |S|)$  矩阵  $W$ :

每一行向量  $[W_{i1}, \dots, W_{i|S|}]$  满足  $x^{(i)} - \sum_{j \in C_{x^{(i)}}} W_{ij} x^{(j)}$

每一行向量  $W_i$  求和为 1:  $\sum_{i=1}^{|S|} W_i = 1$

由  $W$  创建新训练集:

令  $z^{(i)}$  为  $x^{(i)}$  在低维的投影

使所有  $z^{(i)}$  满足最小化  $(z^{(i)} - \sum_{j=1}^{|B|} w_{ij} z^{(j)})^2$

## 7 无监督学习

### clustering

#### K-mean:

将数据分为 k 个 cluster, 每个 cluster 有中心点称 centroid

算法:

1. 初始化随机选择 k 个样本位置做 centroid
  2. 分配样本: 每个样本分入距离最近的 centroid 的 cluster
  3. 更新 centroid: 新 centroid 为 cluster 中样本坐标平均值。
- 重复第 2.3. 步, 直至 centroid 不再移动

迭代:

多次随机初始化 centroid, 选择其中 inertia 最小的 centroid 取法进行训练

$$\text{inertia} = \frac{1}{|S|} \sum_x (C_x - x)^2.$$

$C_x$  为样本 x 距离最近的 centroid

k-mean++ 初始化 centroid:

1. 随机选择 1 个样本做 centroid
2. 剩余每一样本  $x^{(i)}$  有  $\frac{D(x^{(i)})}{\sum_{j=1}^{|S|} D(x^{(j)})}$  几率被选做新 centroid  
 $D(x^{(i)})$  为样本  $x^{(i)}$  距离最近的 centroid 的距离
3. 重复 2. 步直至得到 k 个 centroid

(在尝试多种 cluster 训练后) 选择 cluster 数量 k:

silhouette score: 所有样本的 silhouette coefficient 的均值

一样本  $x^{(i)}$  的 silhouette coefficient:  $\frac{b-a}{\max(a,b)}$

$a$  为  $x^{(i)}$  到同一 cluster 内所有样本的平均距离

$b = \min(E_{x^{(j)} \in \text{other cluster}} (D(x^{(i)} - x^{(j)})))$

silhouette score  $\in [-1, 1]$ , 偏向取 score 高的 cluster 数

使用 k-mean 进行数据预处理:

将数据首先进行 k-mean 分类, 将每一样本替换为 样本到最近的 centroid 距离, 传入另一模型进行学习

用于半无监督学习: 将数据进行 k-mean 分类, 从每一 cluster 选取离 centroid 最近的样本, 产生大小为 k 的训练集。则只需得到 k 个样本的标签即可进行训练

#### DBSCAN

适用于一 cluster 内样本密度较高的训练集

算法:

1. 对每一样本  $x_i$  计算集合  $S_{i\epsilon}$ , 称  $\epsilon$ -neighbourhood, 包含所有距离在  $\epsilon$  内的其他样本  
 $|S_{i\epsilon}| > \text{超参数 } s_{min}$  的样本称 core instance
2. 所有属于同一  $S_{i\epsilon}$  的样本判为属于同一 cluster, 当一样本  $x_i$  同时存在样本  $x_i, x_j$  的  $\epsilon$ -neighbourhood 中时, 合并  $S_{i\epsilon}, S_{j\epsilon}$ 。
3. 没有被分配进任何  $S_{i\epsilon}$  的样本判为异常值

#### Gaussian Mixtures

**GM Model:** 假设所有样本都由多个正态分部产生

## 8 GAN 对抗网络

**基本结构:**

generator: 得到正则噪声, 生成伪数据。

discriminator, 从实际数据集或 generator 得到数据判断真伪 (是否来自实际数据集)。

一次训练:

1. discriminator 从数据集得到一批量真实数据 和 一批量伪数据。使用二元交叉熵损失函数训练

目标为: 对伪数据输出 0 真数据 1

2. generator 生成图像

目标为生成 discriminator 输出为 1 的数据

**Deep Convolutional GAN**

## 9 RL 强化学习

**基本定义**

程序在 environment 环境中根据观测得到的 state 状态, 选择 action 行为, 得到 reward 反馈

模型整体符号定义  $\langle A, S, R, P \rangle$

Action space  $A$

State space  $S$

Reward  $R: \sum \times A \times S \rightarrow R$

Transition  $P: \sum \times A \rightarrow S$

第  $i$  决策的符号定义:

$a_i \in A$  采取的行为

$r_i \in R$  得到的 reward, 每一行为可以立即得到反馈值

exploring 探索: 模型尝试新行为

exploiting 利用: 模型使用已知高反馈行为

**policy**

根据观测选择  $a_i$  的算法

stochastic policy: policy 中有随机性

随机性提高模型 explore 新行为

generic algorithm: 遗传算法

policy gradients: 对参数求导, 更新参数

**credit assignment:**

对每一决策分配 discounted reward, 代表此决策对随后几次决策的反馈值影响

定义:

$l$  此次试验一共包含的决策数

计算:

第  $l-1$  决策有 discounted reward:  $d_i = r_i$

第  $i$  决策有 discounted reward:  $d_i = r_i + \gamma * d_{i+1}$

正则化：对所有实验中每一次决策  $r_i$  取整体平均值，方差，求标准化

### neural network policy

**前向传播**：使用神经网络得到行为可能性，根据可能性选择行为。属于 generic gradient policy

**单次迭代**：

定义：一次决策

1. 从 policy 得到行为可能性
2. 用交叉熵代价函数求代价值， $y_{\text{hat}}$  为 1. 中可能性， $y$  为实际采取的行为
3. 根据代价函数求斜率，斜率使神经网络输出可能性更偏向采取的行为。但不立即使用斜率

1. 随机初始化 1 次模型，对  $n$  个随机初始化环境进行试验，每一试验中包含多个决策

每一环境得到决策数不一定相同，取决于试验中进行的决策次数

2. 对每一决策求 discounted reward，结果包含  $n$  组数组，第  $l_i$  组数组对应第  $i$  次实验的 discount

reward 数组

3. 对所有 discounted reward 标准化，平均值 方差为所有 dis reward 值

4. 对每一参数每一决策的斜率乘对应决策的 dis reward。对结果中所有属于同一参数的乘积取平均值，为此参数的斜率

5. 使用斜率更新参数值

### Markov Decision Process MPD

每一状态  $s_i$  有可执行的行为集合  $A_{s_i} \subseteq A$ 。不同状态行为集合可有交集

行为  $a_{ij} \in A_{s_i}$  代表状态  $s_i$  执行行为  $a_j$ 。执行  $a_{ij}$  后有  $p_{a_{ij}, s_k}$  几率到达状态  $s_k$

optimal state value  $V^*(s_i)$ ：

模型到达状态  $s_i$  后 1. 选择最理想的  $a_{ij}$  能得到的 discounted reward 总和

$$V^*(s_i) = \max_j \sum_s [p_{a_{ij}, s_k} (R(s_i, a_{ij}, s_j) + \gamma \cdot V^*(s_j))]$$

Q-value iteration algorithm：

得到从  $s_i$  选择  $a_{ij}$  后期望的 discounted reward 值

$$\text{迭代: } Q_{n+1}(s_i, a_{ij}) = \sum_j [p(a_{ij}, s_j) (R(s_i, a_{ij}, s_j) + \gamma \cdot \max_{a_{jk}} (Q_n(s_k, a_{jk})))]$$

policy：状态为  $s_i$  时选取  $a_{ij} = \max_{a_{ij}} Q^*(s_i, a_{ij})$

### Temporal Difference Learning TD learning

在  $p_{a_{ij}, s_j}$ ,  $R$  未知的情况下迭代得到  $V(s_i)$

### Q-Learning

在  $p_{a_{ij}, s_j}$ ,  $R$  未知的情况下迭代得  $Q(s_i, a_{ij})$

$$Q_{n+1}(s_i, a_{ij}) = r'_i + \gamma \cdot \max_k (Q_n(s_i, a_{jk}))$$

### deep Q-networks DQNs

#### Generic algorithm

通过交换参数的一部分得到更优的参数组合

模型：

对参数向量  $\vec{x} = [x_1, x_2, \dots]$ ，评估方程  $f(\vec{x}) \in R$

目标：得到  $\vec{x}$  使  $f(\vec{x})$  值最大化

算法：

随机初始化  $n$  参数向量

每一迭代中取  $m$  个参数向量进入集合  $S$

取法 1：每一参数向量  $\vec{x}_i$  有  $\frac{f(\vec{x}_i)}{\sum_i f(\vec{x}_i)}$  几率被选中进入  $S$



取法 2: 将每一参数向量  $\vec{x}_i$  根据  $f(\vec{x}_i)$  排序, 排名第  $j$  向量有  $P(\vec{x}_i) = p * (1 - p)^{(j-1)}$  几率被选中进入  $S$

取法 3: 定义参数向量间间距  $D(\vec{x}_i, \vec{x}_j)$ , 用于保证选取的参数分部范围广泛

取  $f(\vec{x}_i)$  最大的  $\vec{x}_i$  放入  $m$  集合

随后选取  $m-1$  个向量, 选取  $\vec{x}_j$  使得  $f(\vec{x}_j) * E_{v \in S}(D(\vec{x}_j, v))$  最大的

即选取参数向量使得 其与已经选择的向量平均距离 \* 自身  $f(\vec{x}_j)$  值 最大化

根据最优  $m$  个参数向量, 交换参数得到  $m$  个下一代参数向量, 变异出剩余  $n-m$  个参数向量

交换参数: 对  $\vec{x} = [x_1, x_2, \dots]$ ,  $\vec{y} = [y_1, y_2, \dots]$  交换结果为  $\vec{x}' = [x_1, y_2, \dots]$ ,  $\vec{y}' = [y_1, x_2, \dots]$

交换元素的位置根据算法可变

变异向量由已有的  $m$  个后代参数向量变异得到

对  $\vec{x} = [x_1, x_2, \dots]$ , 变异参数向量每一元素  $x'_i$  有  $x'_i U(-step\_size + x_i, step\_size + x_i)$

使用的分部不一定为 Uniform 分部

## 10 分析结果

**confusion matrix 困惑矩阵:** 分析二元/多元分类

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

一行对应同一期望输出, 一列对应同一计算输出

$T/F$ : 此位置的计算输出是否和预计输出一致

$P/N$ : 此位置的预计输出是否为真

$$\text{precision} = \frac{TP}{TP+FP}$$

即  $P(\text{计算结果匹配} | \text{计算结果为正})$

$$\text{recall} = \frac{TP}{TP+FN}$$

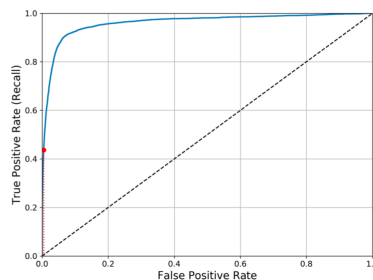
即  $P(\text{计算结果匹配} | \text{预计结果为正})$

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

precision 和 recall 的调和平均值

$$\text{specificity} = \frac{TN}{TN+FN}$$

**ROC curve:** 分析二元/多元分类

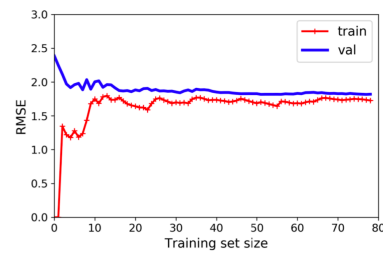


y 轴 recall 值, x 轴 false positive rate  $FPR = \frac{FN}{FN+TN} = \frac{FN}{1-\text{specificity}}$

期望的 ROC curve 为 recall 从 0 快速增长到 1。并保持直到  $FPR$  为 1。

即期望曲线下方面积接近 1

**learning curves:** 观察模型是否有 over underfit



x 轴为一整次训练 (包含多次 epoch) 使用的训练集大小, y 轴为 root MSE。

画出训练集 测试集在使用不同训练集大小后的 root MSE。

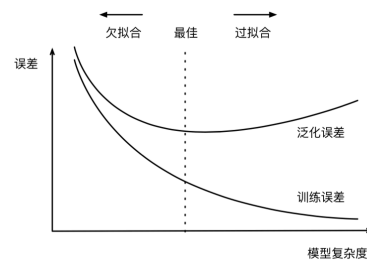
分析:

期望 2 曲线平缓值低且相近,

当 2 曲线平缓值差值较大, 测试集平缓值较低, 则过拟合

当 2 曲线平缓值较高, 则欠拟合

模型复杂度-error epoch-error:



2 种图, 形状类似, x 轴内容不同