INT104W05_分类与模型训练v0.2

分类

二元分类器（Binary Classifier）

- **分类器**：分类算法就是找到一个函数，这个函数可以判断输入的数据属于哪个类别。
- **二元分类器**：属于一种监督学习算法，可以将新的观测值分成两类

简单来说，二元分类器就是将事情分成是与非。如判断一个数字5，在二元分类器中就是用来分成两个类别：5（正）或非5（负）。此时随机梯度是一个很好的选择(SGD)。因为SGD可以独立处理实例，一次一个。

如果我们创造了一个分类器，评判分类器分类质量是很重要的一件事。

如何评价分类质量?

准确率（Accuracy）

在机器学习中，我们通常把整个数据集拆分为训练集（Train set）和测试集（Test set）（比如70%训练集，30%测试集）

[训练集-70%][测试集-30%]

评价机器学习的泛化（generalize）性能取决于他在测试集（独立于训练集）上的预测能力。

$$准确率 = \frac{预测正确的个数}{总预测数}$$

但我们发现，如果仅仅以一个独立的测试集评判机器学习的性能，我们获得的可能只是在该测试集上表现最好的参数，而不是在整个数据集上表现最好，也就是泛化能力最好的参数。为防止这一问题的产生，我们可以使用**交叉验证（Cross Validation）**把训练集再分为训练集（Training **fold**）和验证集（Validation **fold**）。

交叉验证（Cross Validation）

[训练集-(70-n)%][验证集-n%][测试集-30%]

训练集：用于拟合模型

验证集：用于估计模型选择的预测误差

测试集：用于评估最终所选模型的预测误差

K-flod 交叉验证

将训练集分为K份，每次选一份作为验证集，其余为训练集，从而可以交叉验证

假设这里分为3份：

	训练集			测试集
split1	**Flod1**	Flod2	Flod3	
split2	Flod1	**Flod2**	Flod3	
split3	Flod1	Flod2	**Flod3**	
最终评估				**Test data**

在split1~3中，每次以加粗的部分作为验证集，其余部分作为训练集，从而找到泛化能力最强的参数，最后在最终评估中用测试集测试能力。

问题：类别失衡（class imbalance），分类任务中不同类别的训练样例数目差别很大的情况。

假设一个二分问题，有1000个训练样本，其中正类995个，负类5个。如果我们的**优化目标**是错误率最小化（Accuracy最大化），那么模型完全可以把所有样本都分为正类，准确率高达99.5%。我们发现，在类别不平衡的情况下，模型对少数类（样本量过小的类）的错误分类对准确率的影响很小。很明显，此时尽管模型准确率高达99.5%，但对负类的分类正确率是0%（查全率=0），我们应当选择更科学的优化目标。我们可以引入混淆矩阵。

混淆矩阵（Confusion Matrix）（便于计算）

- TP（True Positive，真正）：将**正类**预测为**正类**
- TN（True Negative，真负）：将**负类**预测为**负类**
- FP（False Positive，假正）：将**负类**预测为**正类（误报）**
- FN（False Negative，假负）：将**正类**预测为**负类（漏报）**

		真标签		
		正	负	
预测标签	正	真正TP	假正FP	**精确率(Precision)**
	负	假负FN	真负TN	
		召回率(Recall)		**准确率(Accuracy)**

假设模型一共将数据集分为n类：

- 精确率（查准率）：模型**正确分类的正例**占**总预测正例**的比例。

$$Precision = \frac{\sum_1^n TP_n}{\sum_1^n (TP_n + FP_n)}$$

- 召回率（查全率）：模型**正确分类的正例**占**总实际正例**的比例。

$$Recall = \frac{\sum_1^n TP_n}{\sum_1^n (TP_n + FN_n)}$$

- 准确率：模型**正确分类的样本**占**总例**的比例。

$$Accuracy = \frac{\sum_1^n (TP_n + TN_n)}{\sum_1^n (TP_n + FP_n + FN_n + TN_n)}$$

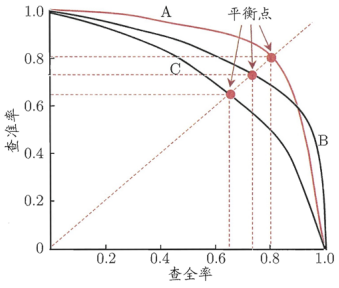


图 2.3 P-R曲线与平衡点示意图

查准率=查全率的平衡点（BEP Break-Even Point）是一种简易评估分类性能的方法，越大越好。p-r曲线面积也可以是一种性能评估方法。

常言道，错抓一万个也不能漏过一个。同时做到高查准率和查全率是几乎不可能的，减少误报往往会导致漏报，减少漏报往往会导致误报。在实际应用中，我们需要根据任务需求平衡精确率与召回率。如在面部识别逃犯时，我们宁愿多一些误报也不要漏过逃犯；而在面部识别门禁中，我们宁愿漏报（让住户的脸有时识别失败）也不希望误报（让陌生人进门）。

那么，如何更科学地评判精确率和召回率呢？

F1/F_β度量(F1/F_β Score)

精确率和召回率的调和平均值（相较于算术平均和几何平均，调和平均更重视最小值）

$$\frac{1}{F_1} = \frac{1}{2} * (\frac{1}{精确率} + \frac{1}{召回率}) \Rightarrow F_1 = \frac{2 * 精确率 * 召回率}{精确率 + 召回率}$$

如果对精确率和召回率的重视程度不同，我们还可以使用F1度量的更一般形式：F_β

$$\frac{1}{F_\beta} = \frac{1}{1+\beta^2} * (\frac{1}{精确率} + \frac{\beta^2}{召回率}) \Rightarrow F_\beta = \frac{(1+\beta^2) * 精确率 * 召回率}{\beta^2 精确率 + 召回率}$$

β>1时召回率影响更大，0<β<1时查准率影响更大。

ROC(Receiver Operating Characteristic curve 受试者工作特征曲线)

很多机器学习的学习器在做预测时并不是直接给出1-正类，0-负类的，而是给出一个处于[0, 1]之间的数字，将其与0.5（阈值Threshold）比较，大于阈值被判为正类1，小于阈值被判为负类0。想象把所有样本根据其预测值从低到高排列，我们可以通过改变阈值的大小把样本切成不同的两部分，低于阈值判负，高于阈值判正。将阈值提高，我们更重视查准率；将阈值降低，我们更重视查全率。我们可以通过从高到低调节这个阈值，改变模型预测的输出，进而画出 ROC 曲线。

ROC 曲线越接近左上角，模型预测准确率越高；在最理想的情况下，所有正类的预测值是1.0，负类0.0，是过(0,1)的直线。

真正例率TP Rate（召回率/命中率）：在所有事实上的正类中模型判对为正的占比

$$TPR = \frac{TP}{TP + FN}$$

假正例率FP Rate（误判率/假阳率）：在所有事实上的负类中模型判错为正的占比

$$FPR = \frac{FP}{FP + TN}$$

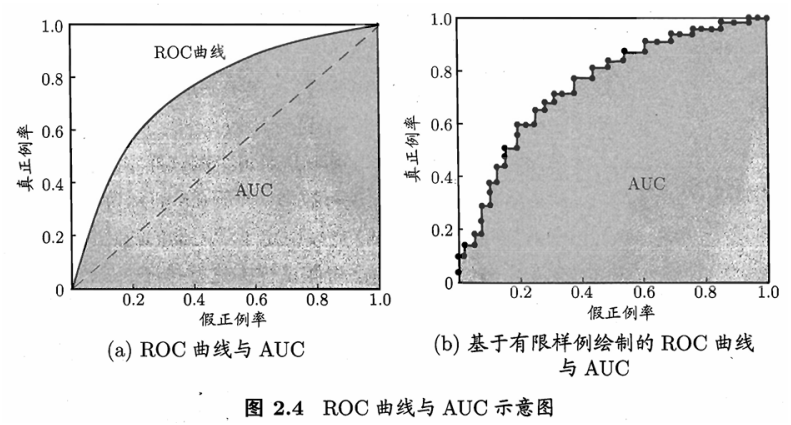


图 2.4 ROC 曲线与 AUC 示意图

基准线（ (0, 0) 到 (1, 1) 的对角线）代表了随机预测时的情况，低于他意味着分类性能不如蒙眼瞎猜。

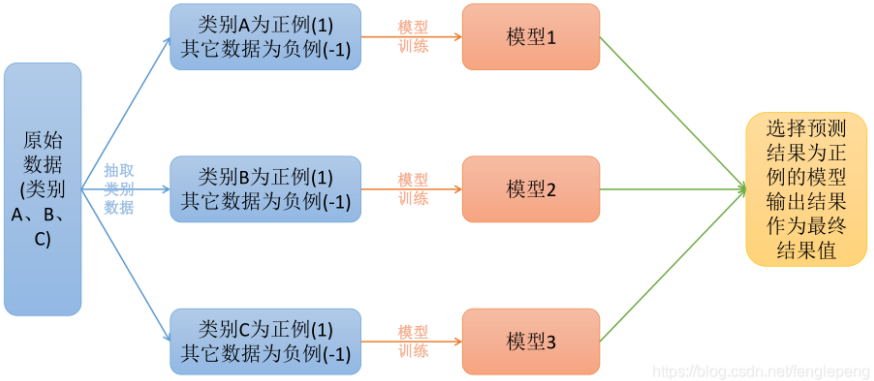
AUC(Area Under Curve)：曲线下面积，通过比较曲线下面积，我们可以比较预测性能的好坏，越大越好。

多类分类（multiclass classification）

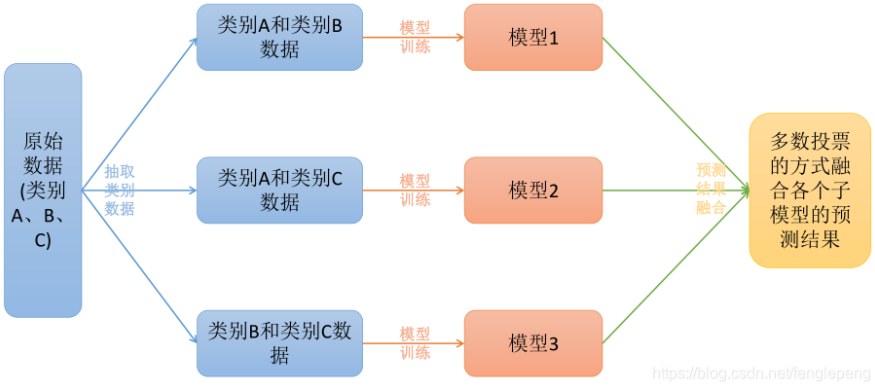
指可以区分两个以上类的分类任务。

策略：

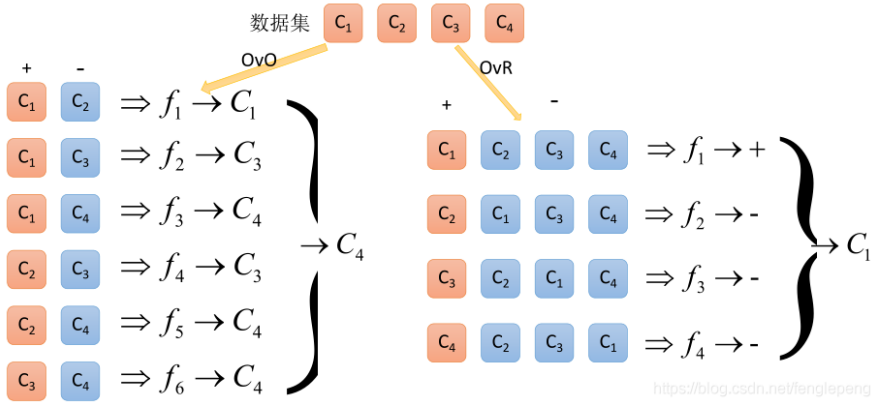
- One-Versus-The-Rest (OvR)：为每个类训练多个二元分类器，选择分类器输出最高分的类。训练n次



- One-versus-one (OvO)：为每对类训练一个二元分类器。训练n(n-1)/2次



区别：



- 多标签分类（**Multi-Label Machine Learning MLL**）：输出多个二进制标签

模型训练

线性回归

回归：试图确定一个因变量（pendent variable 通常用 Y 表示）和一系列其他变量（independent variables称为自变量）之间关系的强度和特征。

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

- x：输入的特征（feature value）
- y^hat：输出的预测值（predicted value）
- k,b：模型的参数（权重与偏置）

推广到n个参数的情况下：

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + +\theta_n x_n$$

- y^hat：输出的预测值（predicted value）
- x_i：第i个特征（feature value）
- θ_j：第j个模型的参数（$\theta_{1\sim n}$权重与θ_0偏置）

$$残差：\hat{e}_i = y_i - \hat{y}_i$$

残差：实际观察值与估计值（拟合值）之间的差。

$$残差平方和：RSS = \sum_{1}^{n} \hat{e}^2$$

损失函数Cost（loss） function：线性回归模型的均方误差（MSE mean-square error）

$$MSE(X, h_\theta) = \frac{1}{m} \sum_{1}^{m} (\theta^T x_i - y_i)^2$$

MSE越小说明loss越小，拟合越好。

正态方程Normal Equation：

TODO!

梯度下降（Gradient Descent）

*在机器学习领域，向量一般是列向量

思想：先任取点（x0,f(x0)），求f(x)在该点x0的导数f"(x0),在用x0减去导数值f"(x0),计算所得就是新的点x1。然后再用x1减去f"(x1)得x2...以此类推，循环多次，慢慢x值就无限接近极小值点。可理解为：一个人下山，以他当前的所处的位置为基准，寻找这个位置最陡峭的方向，然后朝着下降方向走一步，然后又继续以当前位置为基准，再找最陡峭的方向，再走，直到最后到达最低处。

目的是尽可能降低损失函数，从而确保模型精度足够高

从起始位置$\theta_0=[\theta_0, \theta_1, ..., \theta_n]^T$开始：计算梯度

$$\text{起始位置：}\Theta_0 = [\theta_0, \theta_1, ..., \theta_n]^T \tag{1}$$
$$\text{计算梯度：}\nabla_\theta MSE(\Theta) \tag{2}$$
$$\text{迭代：}\Theta_{i+1} = \Theta_i - \eta \nabla_\theta MES(\Theta_i) \tag{3}$$

其中，η是学习率（步长），是重要的

特征规模不同时可能要花很久到达最小，此时归一化特征规模很重要，确保各个数据的规模是一样的。

此外，除非损失函数是凸函数，普通的梯度下降法不能保证得到全局最优解。

全（批量）梯度下降（FGD）：TODO！计算mse

随机梯度下降（SGD）：每次选择一个随机的instance，尽管可能不能达到最低点，但快速且有效。每次迭代的噪声会导致损失函数振荡。

小批量梯度下降（Mini-batch gradient decent）：收敛性优于随机梯度下降。对于大型数据集具有计算效率。但不能保证收敛到全局最小值。

多项式回归（Polynomial Regression）

用多项式拟合数据，但注意不要过拟合

学习曲线（Learning Curves）

欠拟合：训练数据性能较差，验证数据性能较差。

过拟合：在训练数据上表现良好，但在验证数据上表现不佳。

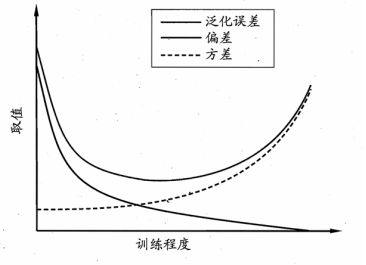


图 2.9 泛化误差与偏差、方差的关系示意图

岭回归（Ridge Regression）（L2正则化）

在普通的线性回归中，当特征数大于样本数时，模型会出现过拟合的问题。而岭回归通过限制参数的大小，可以有效地避免过拟合问题。
岭回归的实现方式是给损失函数添加一个正则项，这个正则项包含所有参数的平方和，并乘以一个系数alpha。
alpha的值越大，惩罚力度越大，参数越趋向于0，就越容易解决过拟合的问题。

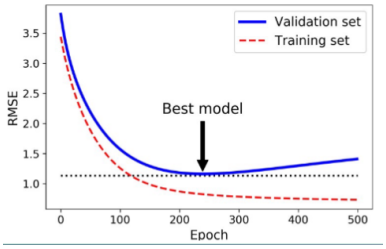
套索回归（Lasso Regression）（L1正则化）

一般线性回归模型的目标是最小化残差平方和，即通过拟合一个线性方程来预测目标变量。然而，在实际问题中，可能存在大量的自变量，其中一些可能对目标变量的预测能力较弱或冗余。此时，Lasso回归通过引入L1正则化（即Lasso惩罚项），可以将系数向量中小的权重变为0，从而实现特征选择和模型稀疏性。

正则化线性模型：

弹性网络（elastic net）

使用L1和L2先验作为正则化矩阵的线性回归模型



注意在模型训练时注意欠拟合和过拟合

逻辑回归（logistic regression）-softmax回归

适合离散二值数据

softmax函数

一种处理结果的方式

逻辑函数的一种推广。它能将一个含任意实数的K维向量z“压缩”到另一个K维实向量σ(z)中，使得每一个元素的范围都在(0,1)之间，并且所有元素的和为1。多用于多分类问题。

$$P(y=j)=\frac{e^{x^T W_j}}{\sum_{k=1}^{K} e^{x^T W_k}}$$