模型选择

一、泛化能力

1. 为了评估机器学习算法的能力，必须给定其性能的衡量指标。
2. 有些情况下，很难决定衡量指标是什么：
   - 如：翻译任务中，应该衡量整个翻译结果的准确率，还是衡量每个单词翻译的准确率？
   - 如：密度估计任务中，很多模型都是隐式地表示概率分布。此时计算样本空间某个点的真实概率是不可行的，因此也就无法判断该点的概率估计的准确率。
3. 通常利用最小化训练误差来训练模型，但是真正关心的是测试误差。因此通过测试误差来评估模型的泛化能力。
   - 训练误差是模型在训练集的平均损失，其大小虽然有意义，但是本质上不重要。
   - 测试误差是模型在测试集上的平均损失，反应了模型对未知测试数据集的预测能力。
4. 模型对未知数据的预测能力称作模型的泛化能力，它是模型最重要的性质。

   泛化误差可以反映模型的泛化能力：泛化误差越小，该模型越有效。
5. 假设训练集和测试集共同的、潜在的样本分布称作数据生成分布，记作 $p_{data}(\vec{\mathbf{x}},y)$ 。则泛化误差定义为模型的期望风险，即：

$$R_{exp}(f)=\mathbb{E}[L(y,f(\vec{\mathbf{x}}))]=\int_{\mathcal{X}\times\mathcal{Y}}L(y,f(\vec{\mathbf{x}}))p_{data}(\vec{\mathbf{x}},y)d\vec{\mathbf{x}}\ dy$$

   - 通常泛化误差是不可知的，因为无法获取联合概率分布 $p_{data}(\vec{\mathbf{x}},y)$ 以及无限的采样点。
   - 现实中通常利用测试误差评估模型的泛化能力。由于测试数据集是有限的，因此这种评估结果不完全准确。
6. 统计理论表明：如果训练集和测试集中的样本都是独立同分布产生的，则有 **模型的训练误差的期望等于模型的测试误差的期望** 。
7. 机器学习的“没有免费的午餐定理”表明：在所有可能的数据生成分布上，没有一个机器学习算法总是比其他的要好。
   - 该结论仅在考虑所有可能的数据分布时才成立。
   - 现实中特定任务的数据分布往往满足某类假设，从而可以设计在这类分布上效果更好的学习算法。
   - 这意味着机器学习并不需要寻找一个通用的学习算法，而是寻找一个在关心的数据分布上效果最好的算法。
8. 正则化是对学习算法做的一个修改，这种修改趋向于降低泛化误差（而不是降低训练误差）。
   - 正则化是机器学习领域的中心问题之一。
   - 没有免费的午餐定理说明了没有最优的学习算法，因此也没有最优的正则化形式。

二、过拟合、欠拟合

1. 当使用机器学习算法时，决定机器学习算法效果的两个因素：降低训练误差、缩小训练误差和测试误差的差距。

   这两个因素对应着机器学习中的两个主要挑战：欠拟合和过拟合。
2. 过拟合 `overfitting` ：选择的模型包含的参数过多，以至于该模型对于已知数据预测得很好，但是对于未知数据预测的很差，使得训练误差和测试误差之间的差距太大。

- 过拟合的原因是：将训练样本本身的一些特点当作了所有潜在样本都具有的一般性质，这会造成泛化能力下降。
- 过拟合无法避免，只能缓解。因为机器学习的问题通常是`NP`难甚至更难的，而有效的学习算法必然是在多项式时间内运行完成。如果可以避免过拟合，这就意味着构造性的证明了`P=NP`。

3. 欠拟合`underfitting`：选择的模型包含的参数太少，以至于该模型对已知数据都预测的很差，使得训练误差较大。

   欠拟合的原因一般是学习能力低下造成的。

4. 通过调整模型的容量`capacity`可以缓解欠拟合和过拟合。

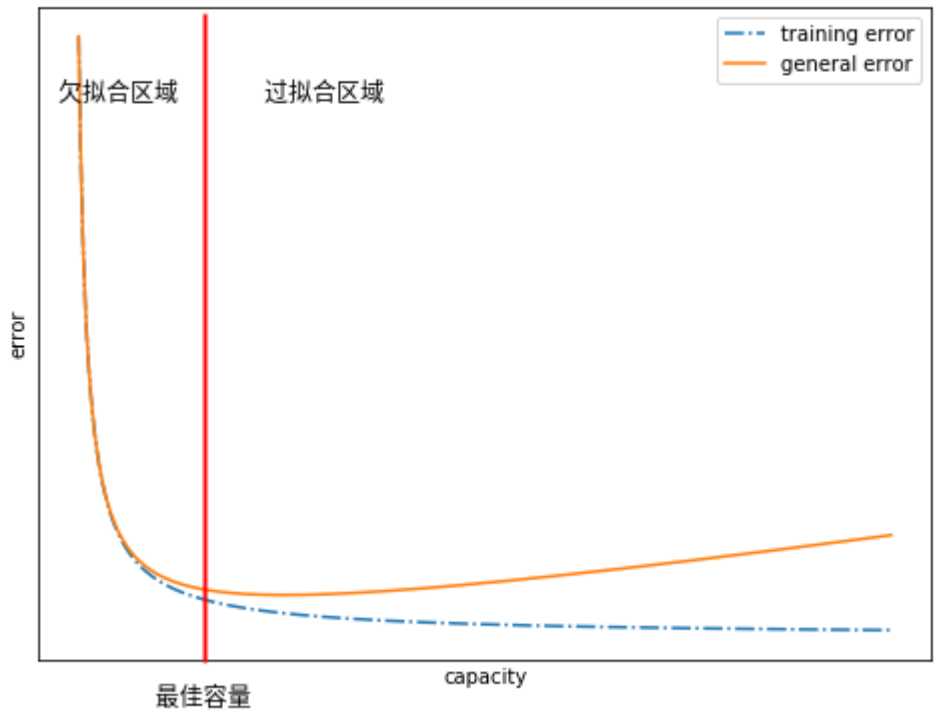
2.1 模型容量

1. 模型的容量是指其拟合各种函数的能力。
   - 容量低的模型容易发生欠拟合，模型拟合能力太弱。
   - 容量高的模型容易发生过拟合，模型拟合能力太强。
2. 通过选择不同的假设空间可以改变模型的容量。

   模型的假设空间指的是：代表模型的函数集合。这也称作模型的表示容量`representational capacity`。

   由于额外的限制因素（比如优化算法的不完善），模型的有效容量`effective capacity`一般会小于模型的表示容量。

3. 通常在模型的假设空间中出最佳的函数是非常困难的优化问题，实际应用中只是挑选一个使得训练误差足够低的函数即可。
4. 统计学习理论提供了量化模型容量的方法，其中最出名的是`VC`维理论：**训练误差与泛化误差之间差异的上界随着模型容量增长而增长，随着训练样本增多而下降**。
5. 虽然`VC`维理论对于机器学习算法有很好的指导作用，但是它在深度学习很难应用。原因有二：
   - 边界太宽泛。
   - 难以确定深度学习的容量。由于深度学习模型的有效容量受限于优化算法，因此确定深度学习模型的容量特别困难。
6. 通常泛化误差是关于模型容量的`U`形函数。随着模型容量增大：
   - 训练误差会下降直到逼近其最小值。
   - 泛化误差先减小后增大。
   - 泛化误差与训练误差的差值会增大。



2.2 缓解过拟合

1. 缓解过拟合的策略：
   - 正则化。
   - 数据集增强：通过人工规则产生虚假数据来创造更多的训练数据。
   - 噪声注入：包括输入噪声注入、输出噪声注入、权重噪声注入。将噪声分别注入到输入/输出/权重参数中。
   - 早停：当验证集上的误差没有进一步改善时，算法提前终止。

   > 具体内容参考深度学习《正则化》章节。

2. 正则化 ：基于结构化风险最小化（ `SRM` ）策略的实现，其中 $J(f)$ 为正则化项。

   在不同的问题中，正则化项可以有不同的形式：
   - 回归问题中，损失函数是平方损失，正则化项是参数向量的 L_2 范数。
   - 贝叶斯估计中，正则化项对应于模型的先验概率 $\log \frac{1}{g(\theta)}$ 。

2.3 缓解欠拟合

1. 缓解欠拟合的策略：选择一个模型容量更高的模型。

三、偏差方差分解

3.1 点估计

1. 点估计：对参数 θ 的一个预测，记作 $\hat{\theta}$。

   假设 $\{x_1, x_2, \cdots, x_m\}$ 为独立同分布的数据点，该分布由参数 θ 决定。则参数 θ 的点估计为某个函数：

   $$\hat{\theta}_m = g(x_1, x_2, \cdots, x_m)$$

   注意：点估计的定义并不要求 g 返回一个接近真实值 θ 。

2. 根据频率学派的观点：

   - 真实参值 θ 是固定的，但是未知的。
   - $\hat{\theta}_m$ 是数据点的函数。
   - 由于数据是随机采样的，因此 $\hat{\theta}_m$ 是个随机变量。

3.2 偏差

1. 偏差定义为： $bias(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$，期望作用在所有数据上。
   - 如果 $bias(\hat{\theta}_m) = 0$ ，则称估计量 $\hat{\theta}_m$ 是无偏的。
   - 如果 $\lim_{m\to\infty} bias(\hat{\theta}_m) = 0$，则称估计量 $\hat{\theta}_m$ 是渐近无偏的。
2. 无偏估计并不一定是最好的估计。
3. 偏差的例子：
   - 一组服从均值为 θ 的伯努利分布的独立同分布样本 $\{x_1, x_2, \cdots, x_m\}$ ： $\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^m x_i$ 为 θ 的无偏估计。
   - 一组服从均值为 μ，方差为 σ^2 的高斯分布的独立同分布样本 $\{x_1, x_2, \cdots, x_m\}$：
     - $\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^m x_i$ 为 μ 的无偏估计。
     - $\hat{\sigma}_m^2 = \frac{1}{m}\sum_{i=1}^m (x_i - \hat{\mu}_m)^2$ 为 σ^2 的有偏估计。因为 $\mathbb{E}[\hat{\sigma}_m^2] = \frac{m-1}{m}\sigma^2$
     - $\tilde{\sigma}_m^2 = \frac{1}{m-1}\sum_{i=1}^m (x_i - \hat{\mu}_m)^2$ 为 σ^2 的无偏估计。

3.3 一致性

1. 通常希望当数据集的大小 m 增加时，点估计会收敛到对应参数的真实值。即：

   $$\text{plim}_{m\to\infty}\hat{\theta}_m = \theta$$

   plim 表示依概率收敛。即对于任意的 $\epsilon > 0$，当 $m \to \infty$ 时，有： $P(|\hat{\theta}_m - \theta|) > \epsilon \to 0$

2. 上述条件也称做一致性。它保证了估计偏差会随着样本数量的增加而减少。
3. 渐近无偏不一定意味着一致性。

   如：在正态分布产生的数据集中，可以用 $\hat{\mu}_m = x_1$ 作为 μ 的一个估计。

   - 它是无偏的，因为 $\mathbb{E}[x_1] = \mu$，所以不论观测到多少个数据点，该估计都是无偏的
   - 但它不是一致的，因为他不满足 $\text{plim}_{m\to\infty}\hat{\mu}_m = \mu$

3.4 方差

1. 估计量的方差记作 $Var(\hat{\theta})$，标准差记作 $SE(\hat{\theta})$ 。

   它们刻画的是：从潜在的数据分布中独立的获取样本集时，估计量的变化程度。

2. 例：一组服从均值为 θ 的伯努利分布的独立同分布样本 $\{x_1, x_2, \cdots, x_m\}$
   - $\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^m x_i$ 为 θ 的无偏估计。
   - $Var(\hat{\theta}_m) = \frac{1}{m}\theta(1-\theta)$ 。表明估计量的方差随 m 增加而下降。

3. 估计量的方差随着样本数量的增加而下降，这是所有估计量的共性。
4. 例：均值估计 $\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^m x_i$ ，其标准差为：

$$SE(\hat{\mu}_m) = \sqrt{Var\left[\frac{1}{m}\sum_{i=1}^m x_i\right]} = \frac{\sigma}{\sqrt{m}}$$

其中 σ 是样本 x_i 的真实标准差，但是这个量难以估计。实际上 $\sqrt{\frac{1}{m}\sum_{i=1}^m (x_i - \hat{\mu}_m)^2}$ 和 $\sqrt{\frac{1}{m-1}\sum_{i=1}^m (x_i - \hat{\mu}_m)^2}$ 都不是真实标准差 σ 的无偏估计，这两种方法都倾向于低估真实的标准差。

实际应用中，$\sqrt{\frac{1}{m-1}\sum_{i=1}^m (x_i - \hat{\mu}_m)^2}$ 是一种比较合理的近似估计，尤其是当 m 较大的时候。

3.5 偏差方差分解

1. 偏差和方差衡量的是估计量的两个不同误差来源：
   - 偏差衡量的是偏离真实值的误差的期望。
   - 方差衡量的是由于数据采样的随机性可能导致的估计值的波动。
2. 通常希望的是：
   - 估计量的偏差比较小，即：估计量的期望值接近真实值。
   - 估计量的方差比较小，即：估计量的波动比较小。
3. 假设：
   - 在训练集为 $\mathbb{D}$ 上学习到的模型为 $f_{\mathbb{D}}(\vec{\mathbf{x}};\mathbb{D})$ 。

     > 不同的训练集训练得到不同的模型，因此模型与训练集 $\mathbb{D}$ 相关。

   - 样本 $\vec{\mathbf{x}}$ 的观测值为 $\tilde{y}$ ，其真实值为 y。其中 $\tilde{y} = y + \epsilon$， ϵ 为观测误差。

     > 观测误差是由人工标注失误引起的。

   - 观察误差的期望为0： $\mathbb{E}_{\mathbb{D}}(\epsilon) = 0$ 。
   - 观测误差 ϵ 与真实值 y 是相互独立的。即有：$\mathbb{E}_{\mathbb{D}}(y\epsilon) = \mathbb{E}_{\mathbb{D}}(y) \times \mathbb{E}_{\mathbb{D}}(\epsilon) = 0$ 。
   - 样本 $\vec{\mathbf{x}}$ 的估计量为 $\hat{y}_{\mathbb{D}} = f_{\mathbb{D}}(\vec{\mathbf{x}};\mathbb{D})$ 。

   定义：
   - 损失函数为平方损失函数：$L(\tilde{y}, \hat{y}_{\mathbb{D}}) = (\tilde{y} - \hat{y}_{\mathbb{D}})^2$ 。
   - 对未知样本 $\vec{\mathbf{x}}$ ：
     - 预测偏差为：$bias = (\mathbb{E}_{\mathbb{D}}(\hat{y}_{\mathbb{D}}) - y)^2$ 。它刻画了期望输出与真实值之间的差别。
     - 预测方差为： $var = Var(\hat{y}_{\mathbb{D}}) = \mathbb{E}_{\mathbb{D}}\left[(\mathbb{E}(\hat{y}_{\mathbb{D}}) - \hat{y}_{\mathbb{D}})^2\right]$ 。它刻画了模型输出随着训练集 $\mathbb{D}$ 的不同从而导致的波动。
     - 噪声方差为：$noise = Var(\epsilon) = \mathbb{E}_{\mathbb{D}}\left[(\tilde{y} - y)^2\right]$ 。它刻画了不同训练集 $\mathbb{D}$ 中的噪音波动。

   则未知样本 $\vec{\mathbf{x}}$ 的泛化误差定义为损失函数的期望：$Loss = \mathbb{E}_{\mathbb{D}}\left[(\hat{y} - \tilde{y})^2\right] = \mathbb{E}_{\mathbb{D}}\left[(f_{\mathbb{D}}(\vec{\mathbf{x}};\mathbb{D}) - \tilde{y})^2\right]$ 。其中使用观测值 $\tilde{y}$ 而不是真实值 y ，是因为观测值已知而真实值未知。

   则有：

$$\begin{aligned} Loss &= \mathbb{E}_{\mathbb{D}}\left[(\hat{y}_{\mathbb{D}} - \tilde{y})^2\right] = \mathbb{E}_{\mathbb{D}}\left[(\hat{y}_{\mathbb{D}} - \mathbb{E}_{\mathbb{D}}(\hat{y}_{\mathbb{D}}))^2\right] + (\mathbb{E}_{\mathbb{D}}(\hat{y}_{\mathbb{D}}) - y)^2 + \mathbb{E}_{\mathbb{D}}\left[(\tilde{y} - y)^2\right] \\ &= var + bias + var \end{aligned}$$

   于是泛化误差可以分解为偏差、方差和噪声之和：

- 偏差 $bias$ ：度量了学习算法的期望预测与真实结果之间的偏离程度，刻画了学习算法本身的拟合能力。
- 方差 var：度量了训练集的变动所导致的学习性能的变化，刻画了数据扰动造成的影响。
- 噪声 var ：度量了在当前任务上任何学习算法所能达到的期望泛化误差的下界，刻画了学习问题本身的难度。

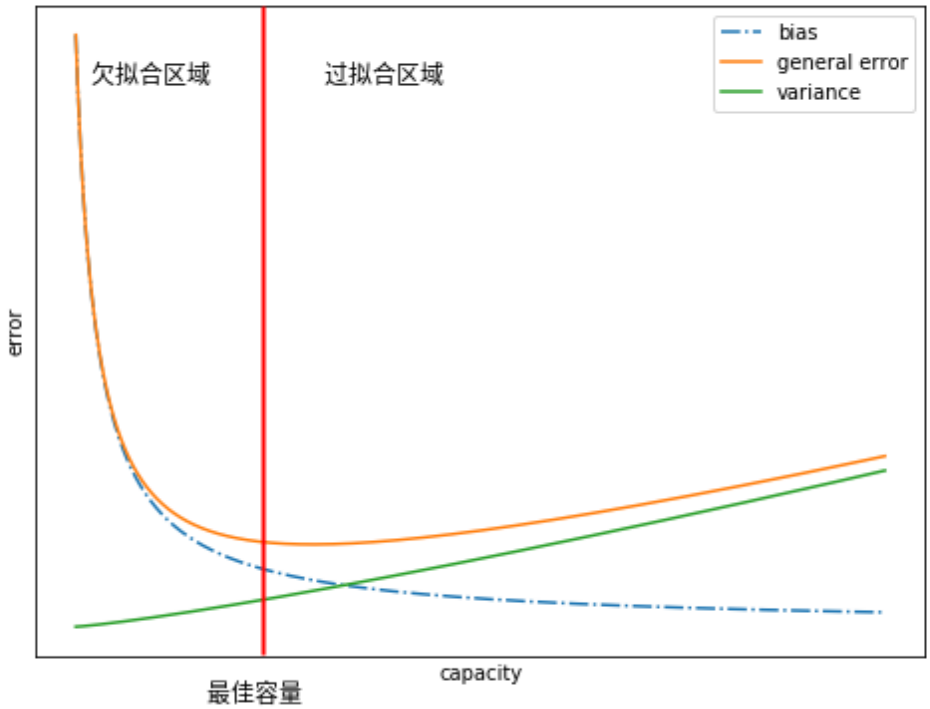
偏差-方差分解表明：泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度共同决定的。

4. 偏差-方差分解中，噪声也可以称作最优误差或者贝叶斯误差。如：在图像识别的问题中，人眼识别的错误率可以视作最优误差。

   在工程实际中，通常会考察`特征完全相同，但是标签不同`的那些样本的数量。这种样本越多，则代表`必须犯的错误`越大。因此实际应用中可以将这个比例作为贝叶斯误差。

5. 偏差、方差与模型容量有关。用`MSE`衡量泛化误差时，增加容量会增加方差、降低偏差。

   - 偏差降低，是因为随着容量的增大，模型的拟合能力越强：对给定的训练数据，它拟合的越准确。
   - 方差增加，是因为随着容量的增大，模型的随机性越强：对不同的训练集，它学得的模型可能差距较大。



6. 一般来说，偏差和方差是由冲突的，这称作偏差-方差窘境`bias-variance dilemma`。

   给定学习任务：

   - 在训练不足时模型的拟合能力不够强，训练数据的扰动不足以使模型产生显著变化，此时偏差主导了泛化误差。
   - 随着训练程度的加深模型的拟合能力逐渐增强，训练数据发生的扰动逐渐被模型学习到，方差逐渐主导了泛化误差。
   - 在训练充分后模型的拟合能力非常强，训练数据发生的轻微扰动都会导致模型发生显著变化。

若训练数据自身的、非全局的特性被模型学到了，则将发生过拟合。

3.6 误差诊断

1. 通常偏差方差反映了模型的过拟合与欠拟合。
   - 高偏差对应于模型的欠拟合：模型过于简单，以至于未能很好的学习训练集，从而使得训练误差过高。

     此时模型预测的方差较小，表示预测较稳定。但是模型预测的偏差会较大，表示预测不准确。
   - 高方差对应于模型的过拟合：模型过于复杂，以至于将训练集的细节都学到，将训练集的一些细节当做普遍的规律，从而使得测试集误差与训练集误差相距甚远。

     此时模型预测的偏差较小，表示预测较准确。但是模型预测的方差较大，表示预测较不稳定。
2. 误差诊断：通过训练误差和测试误差来分析模型是否存在高方差、高偏差。
   - 如果训练误差较高：说明模型的方差较大，模型出现了欠拟合。
   - 如果训练误差较低，而训练误差较高：说明模型的偏差较大，出现了过拟合。
   - 如果训练误差较低，测试误差也较低：说明模型的方差和偏差都适中，是一个比较理想的模型。
   - 如果训练误差较高，且测试误差更高：说明模型的方差和偏差都较大。

   上述分析的前提是：训练集、测试集的数据来自于同一个分布，且最优误差较小。否则讨论更复杂。

3.7 误差缓解

1. 高方差和高偏差是两种不同的情况。如果算法存在高偏差的问题，则准备更多训练数据其实没什么卵用。

   所以首先要清楚：问题是高偏差还是高方差还是二者兼有。
2. 如果模型存在高偏差，则通过以下策略可以缓解：
   - 选择一个容量更大、更复杂的模型。
   - 使用更先进的最优化算法。该策略通常在神经网络中使用。
3. 如果模型存在高方差，则通过以下策略可以缓解：
   - 增加更多的训练数据。它通过更多的训练样本来对模型参数增加约束，会降低模型容量。

     如果有更多的训练数据，则一定会降低方差。
   - 使用正则化。它通过正则化项来对模型参数增加约束，也会降低模型容量。

     有时候更多的训练数据难以获取，只能使用正则化策略。
4. 通常优先解决高偏差的问题。这是最低标准，要反复尝试，直到训练误差降低到足够小。

   然后试图降低方差。

   总之就是不断重复尝试，直到找到一个低偏差、低方差的模型。

四、参数估计准则

4.1 最大似然估计

1. 假设数据集 $\mathbf{X}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_m\}$ 中的样本独立同分布地由 $p_{data}(\vec{\mathbf{x}})$ 产生，但是该分布是未知的。

   $p_{model}(\vec{\mathbf{x}};\theta)$ 是一族由 θ 参数控制的概率分布函数族，希望通过 $p_{model}(\vec{\mathbf{x}};\theta)$ 来估计真实的概率分布函数 $p_{data}(\vec{\mathbf{x}})$ ，也就是要估计 θ 参数。
2. 最大似然估计最大化数据集 $\mathbf{X}$ 出现的概率。即：

$$\theta_{ML}=\arg\max_{\theta}p_{model}(\mathbf{X};\theta)=\arg\max_{\theta}\prod_{i=1}^{m}p_{model}(\vec{\mathbf{x}}_i;\theta)$$

- 由于概率的乘积会因为很多原因不便使用（如容易出现数值下溢出），因此转换为对数的形式：$\theta_{ML}=\arg\max_{\theta}\sum_{i=1}^{m}\log p_{model}(\vec{\mathbf{x}}_i;\theta)$ 。
- 因为 m 与 θ 无关，因此它也等价于： $\theta_{ML}=\arg\max_{\theta}\sum_{i=1}^{m}\frac{1}{m}\log p_{model}(\vec{\mathbf{x}}_i;\theta)$ 。
- 由于数据集的经验分布为：$\hat{p}_{data}(\vec{\mathbf{x}})=\frac{1}{m}\sum_{i=1}^{m}\delta(\vec{\mathbf{x}}-\vec{\mathbf{x}}_i)$ ，其中 $\delta(\cdot)$ 为狄拉克函数。因此：$\theta_{ML}=\arg\max_{\theta}\mathbb{E}_{\vec{\mathbf{x}}\sim\hat{p}_{data}}\log p_{model}(\vec{\mathbf{x}};\theta)$ 。

3. 考虑数据集的经验分布 $\hat{p}_{data}$ 和真实分布函数的估计量 p_{model} 之间的差异，`KL` 散度为：

$$D|_{KL}(\hat{p}_{data}||p_{model};\theta)=\mathbb{E}_{\vec{\mathbf{x}}\sim\hat{p}_{data}}[\log\hat{p}_{data}(\vec{\mathbf{x}})-\log p_{model}(\vec{\mathbf{x}};\theta)]$$

由于 $\log\hat{p}_{data}(\vec{\mathbf{x}})$ 与 θ 无关，因此要使得 $D|_{KL}(\hat{p}_{data}||p_{model};\theta)$ 最小，则只需要最小化 $\mathbb{E}_{\vec{\mathbf{x}}\sim\hat{p}_{data}}[-\log p_{model}(\vec{\mathbf{x}};\theta)]$ 。也就是最大化 $\mathbb{E}_{\vec{\mathbf{x}}\sim\hat{p}_{data}}\log p_{model}(\vec{\mathbf{x}};\theta)$ 。

因此：**最大似然估计就是最小化数据集的经验分布 $\hat{p}_{data}$ 和真实分布函数的估计量 p_{model} 之间的差异**。

4. 最大似然估计可以扩展到估计条件概率。

假设数据集 $\mathbf{X}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_m\}$，对应的观测值为 $\mathbf{Y}=\{y_1,y_2,\cdots,y_m\}$。则条件概率的最大似然估计为：$\theta_{ML}=\arg\max_{\theta}p(\mathbf{Y}\mid\mathbf{X};\theta)$ 。

如果样本是独立同分布的，则可以分解成：$\theta_{ML}=\arg\max_{\theta}\sum_{i=1}^{m}\log p(y_i\mid\vec{\mathbf{x}}_i;\theta)$ 。

5. 最大似然估计有两个很好的性质：

- 在某些条件下，最大似然估计具有一致性。这意味着当训练样本数量趋向于无穷时，参数的最大似然估计依概率收敛到参数的真实值。

  这些条件为：

  - 真实分布 p_{data} 必须位于分布函数族 $p_{model}(\cdot;\theta)$ 中；否则没有估计量可以表示 p_{data} 。
  - 真实分布 p_{data} 必须对应一个 θ 值；否则从最大似然估计恢复出真实分布 p_{data} 之后，也不能解出参数 θ 。
- 最大似然估计具有很好的统计效率`statistic efficiency`。即只需要较少的样本就能达到一个良好的泛化误差。

6. 最大似然估计通常是机器学习中的首选估计准则。
7. 当样本数量太少导致过拟合时，正则化技巧是最大似然的有偏估计版本。

4.2 贝叶斯估计

4.2.1 贝叶斯估计 vs 最大似然估计

1. 在最大似然估计中，频率学派的观点是：真实参数 θ 是未知的固定的值，而点估计 $\hat{\theta}$ 是随机变量。因为数据是随机生成的，所以数据集是随机的。

   在贝叶斯估计中，贝叶斯学派认为：数据集是能够直接观测到的，因此不是随机的。而真实参数 θ 是未知的、不确定的，因此 θ 是随机变量。

   - 对 θ 的已知的知识表示成先验概率分布 $p(\theta)$ ：表示在观测到任何数据之前，对于参数 θ 的可能取值的一个分布。

     在机器学习中，一般会选取一个相当宽泛的（熵比较高）的先验分布，如均匀分布。

   - 假设观测到一组数据 $\mathbf{X}=\{\vec{\mathbf{x}}_1,\vec{\mathbf{x}}_2,\cdots,\vec{\mathbf{x}}_m\}$ ，根据贝叶斯法则，有：

$$p(\theta \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \theta)p(\theta)}{p(\mathbf{X})}$$

2. 贝叶斯估计与最大似然估计有两个重要区别：
   - 贝叶斯估计预测下，一个样本的分布为：

     $$p(\vec{\mathbf{x}}_{m+1} \mid \vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \cdots, \vec{\mathbf{x}}_m) = \int p(\vec{\mathbf{x}}_{m+1} \mid \theta)p(\theta \mid \vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \cdots, \vec{\mathbf{x}}_m)d\theta$$

     而最大似然估计预测下，一个样本的分布为： $p_{model}(\vec{\mathbf{x}};\theta)$
   - 贝叶斯估计会使得概率密度函数向着先验概率分布的区域偏移。
3. 当训练数据有限时，贝叶斯估计通常比最大似然估计泛化性能更好。

   当训练样本数量很大时，贝叶斯估计往往比最大似然估计计算代价较高。

4.2.2 最大后验估计

1. 有时候希望获取参数 θ 的一个可能的值，而不仅仅是它的一个分布。此时可以通过最大后验估计`MAP`选择后验概率最大的点：

   $$\theta_{MAP} = \arg\max_{\theta} p(\theta \mid \mathbf{X}) = \arg\max_{\theta} \left[\log p(\mathbf{X} \mid \theta) + \log p(\theta)\right]$$

2. 最大后验估计具有最大似然估计没有的优势：拥有先验知识带来的信息。该信息有助于减少估计量的方差，但是增加了偏差。
3. 一些正则化方法可以被解释为最大后验估计，正则化项就是对应于 $\log p(\theta)$ 。
   - 并非所有的正则化方法都对应为某个最大后验估计。

     如：有些正则化项依赖于数据，则显然不是一个先验概率分布
4. 最大后验估计估计`MAP`提供了一个直观的方法去设计复杂的、可解释的正则化项。

   更复杂的正则化项可以通过先验分布为混合高斯分布得到（而不仅仅是一个单独的高斯分布）。

五、泛化能力评估

1. 模型泛化能力的评估：用测试集对模型进行评估。通常有下列方法：
   - 留出法`hold-out`。
   - `K`折交叉验证法`cross validation`。
   - 留一法`Leave-One-Out:LOO`。
   - 自助法`bootstrapping`。

5.1 留出法

1. 留出法：直接将数据切分为三个互斥的部分（也可以切分成两部分，此时训练集也是验证集），然后在训练集上训练模型，在验证集上选择模型，最后用测试集上的误差作为泛化误差的估计。
2. 数据集的划分要尽可能保持数据分布的一致性，避免因数据划分过程引入额外的偏差而对最终结果产生影响。若训练集、验证集、测试集中类别比例差别很大，则误差估计将由于训练/验证/测试数据分布的差异而产生偏差。

   如：在分类任务中至少要保持样本的类别比例相似。如果从采样的角度来看到数据集的划分过程，则保留类别比例的采样方式称作“分层采样“(`strafified sampling`)。

3. 即使进行了分层采样，仍然存在多种划分方式对数据集进行划分（比如排序后再划分、随机划分...）。这些不同的划分将导致不同的训练集/验证集/测试集。因此单次留出法得出的估计结果往往不够稳定可靠。

   在使用留出法时，往往采用若干次随机划分、重复进行实验评估后，取平均值作为留出法的评估结果。

5.2 K 折交叉验证

1. `K` 折交叉验证法：数据随机划分为`K` 个互不相交且大小相同的子集，利用 `K-1` 个子集数据训练模型，利用余下的一个子集测试模型（一共有 $C_K^{K-1}=K$ 种组合）。

   对 `K` 种组合依次重复进行，获取测试误差的均值，将这个均值作为泛化误差的估计。

2. 与留出法相似，将数据集划分为 `K` 个子集同样存在多种划分方式。为了减少因为样本划分不同而引入的差别， `K` 折交叉验证通常需要随机使用不同划分重复`p` 次，这 `p` 次 `K` 折交叉验证的测试误差均值作为最终的泛化误差的估计。

5.3 留一法

1. 留一法：假设数据集中存在 N 个样本，令 $K=N$ 则得到了 `K` 折交叉验证的一个特例。
2. 优点：由于训练集与初始数据集相比仅仅少一个样本，因此留一法的训练数据最多。

   缺点：在数据集比较大时，训练 N 个模型的计算量太大。

5.4 自助法

1. 在留出法和 `K` 折交叉验证法中，由于保留了一部分样本用于测试，因此实际训练模型使用的训练集比初始数据集小，这必然会引入一些因为训练样本规模不同而导致的估计偏差。

   留一法受训练样本规模变化的影响较小，但是计算复杂度太高。

   自助法是一个以自助采样法(`bootstrap sampling`)为基础的比较好的解决方案。

2. 自助采样法：给定包含 N 个样本的数据集 $\mathbb{D}$ ，对它进行采样产生数据集 $\mathbb{D}'$ ：
   - 每次随机从 $\mathbb{D}$ 中挑选一个样本，将其拷贝放入 $\mathbb{D}'$ 中，然后再将该样本放回初始数据集 $\mathbb{D}$ 中（该样本下次采样时仍然可以被采到）。
   - 重复这个过程 N 次，就得到了包含 N 个样本的数据集 $\mathbb{D}'$ 。
3. 显然， $\mathbb{D}$ 中有些样本会在 $\mathbb{D}'$ 中多次出现了； $\mathbb{D}$ 中有些样本在 $\mathbb{D}'$ 中从不出现。$\mathbb{D}$ 中某个样本始终不被采到的概率为 $(1-\frac{1}{N})^N$ 。

   根据极限 $\lim_{N\to\infty}(1-\frac{1}{N})^N=\frac{1}{e}\simeq 0.368$ ，即通过自助采样，初始数据集 $\mathbb{D}$ 中约有 36.8% 的样本未出现在采样数据集 $\mathbb{D}'$中。

   将 $\mathbb{D}'$ 用作训练集， $\mathbb{D}-\mathbb{D}'$ 用作测试集，这样的测试结果称作包外估计`out-of-bag estimate`。

4. 自助法在数据集较小时很有用。
   - 优点：能从初始数据集中产生多个不同的训练集，这对集成学习等方法而言有很大好处。
   - 缺点：产生的数据集改变了初始数据集的分布，这会引入估计偏差。因此在初始数据量足够时，留出法和折交叉验证法更常用。

六、训练集、验证集、测试集

6.1 训练集

1. 训练集用于训练模型。理论上训练集越大越好。

6.2 验证集

1. 大多数机器学习算法具有超参数，超参数的值无法通过学习算法拟合出来（比如正则化项的系数、控制模型容量的参数 ）。
2. 为了解决这个问题，可以引入验证集。将训练数据分成两个不相交的子集：训练集用于学习模型，验证集用于更新超参数。
3. 通常要求验证集足够大。如果验证集很小，那么模型的超参数可能就记住了一个小验证集里的样本，模型将对验证集严重过拟合。
4. 验证集通常会低估泛化误差。因此当超参数优化完成后，需要通过测试集来估计泛化误差。

6.3 测试集

1. 测试集用于评估模型的泛化误差。理论上测试集越大，则模型的泛化误差评估的越准确。
2. 测试集中的样本一定不能是训练样本。如果将训练样本放入测试集中，则会低估泛化误差。
3. 测试集 vs 验证集：
    - 测试集通常用于对模型的预测能力进行评估，它提供了模型预测能力的无偏估计。

      如果你不需要对模型预测能力的无偏估计，则不需要测试集。
    - 验证集用于超参数的选择，它无法提供模型预测能力的有偏估计。

      因为模型依赖于超参数，而超参数依赖于验证集。因此验证集参与了模型的构建，这意味着模型已经考虑了验证集的信息。

6.4 拆分

1. 对于小批量数据，数据的拆分的常见比例为：
    - 如果未设置验证集，则将数据三七分：70% 的数据用作训练集、30% 的数据用作测试集。
    - 如果设置验证集，则将数据划分为：60% 的数据用作训练集、20%的数据用过验证集、20% 的数据用作测试集。
2. 对于大批量数据，验证集和测试集占总数据的比例会更小。
    - 对于百万级别的数据，其中1万条作为验证集、1万条作为测试集即可。
    - 验证集的目的就是验证不同的超参数；测试集的目的就是比较不同的模型。
        - 一方面它们要足够大，才足够评估超参数、模型。
        - 另一方面，如果它们太大，则会浪费数据（验证集和训练集的数据无法用于训练）。
3. 在`k` 折交叉验证中：先将所有数据拆分成 `k` 份，然后其中`1` 份作为测试集，其他`k-1` 份作为训练集。
    - 这里并没有验证集来做超参数的选择。所有测试集的测试误差的均值作为模型的预测能力的一个估计。
    - 使用`k` 折交叉的原因是：样本集太小。如果选择一部分数据来训练，则有两个问题：
        - 训练数据的分布可能与真实的分布有偏离。`k` 折交叉让所有的数据参与训练，会使得这种偏离得到一定程度的修正。
        - 训练数据太少，容易陷入过拟合。`k` 折交叉让所有数据参与训练，会一定程度上缓解过拟合。

6.5 分布不匹配

1. 深度学习时代，经常会发生：训练集和验证集、测试集的数据分布不同。

   如：训练集的数据可能是从网上下载的高清图片，测试集的数据可能是用户上传的、低像素的手机照片。
    - 必须保证验证集、测试集的分布一致，它们都要很好的代表你的真实应用场景中的数据分布。

- 训练数据可以与真实应用场景中的数据分布不一致，因为最终关心的是在模型真实应用场景中的表现。

2. 如果发生了数据不匹配问题，则可以想办法让训练集的分布更接近验证集。
   - 一种做法是：收集更多的、分布接近验证集的数据作为训练集合。
   - 另一种做法是：人工合成训练数据，使得它更接近验证集。

     该策略有一个潜在问题：你可能只是模拟了全部数据空间中的一小部分。导致你的模型对这一小部分过拟合。
3. 当训练集和验证集、测试集的数据分布不同时，有以下经验原则：
   - 确保验证集和测试集的数据来自同一分布。

     因为需要使用验证集来优化超参数，而优化的最终目标是希望模型在测试集上表现更好。
   - 确保验证集和测试集能够反映未来得到的数据，或者最关注的数据。
   - 确保数据被随机分配到验证集和测试集上。
4. 当训练集和验证集、测试集的数据分布不同时，分析偏差和方差的方式有所不同。
   - 如果训练集和验证集的分布一致，那么当训练误差和验证误差相差较大时，我们认为存在很大的方差问题。
   - 如果训练集和验证集的分布不一致，那么当训练误差和验证误差相差较大时，有两种原因：
     - 第一个原因：模型只见过训练集数据，没有见过验证集的数据导致的，是数据不匹配的问题。
     - 第二个原因：模型本来就存在较大的方差。

     为了弄清楚原因，需要将训练集再随机划分为：`训练-训练集`、`训练-验证集`。这时候，`训练-训练集`、`训练-验证集` 是同一分布的。
     - 模型在`训练-训练集` 和 `训练-验证集` 上的误差的差距代表了模型的方差。
     - 模型在`训练-验证集` 和 验证集上的误差的差距代表了数据不匹配问题的程度。

七、性能度量

1. 给定训练集 $\mathbb{D}=\{(\vec{\mathbf{x}}_1,\tilde{y}_1),(\vec{\mathbf{x}}_2,\tilde{y}_2),\cdots,(\vec{\mathbf{x}}_N,\tilde{y}_N)\}$ ，测试集合 $\mathbb{T}=\{(\vec{\mathbf{x}}'_1,\tilde{y}'_1),(\vec{\mathbf{x}}'_2,\tilde{y}'_2),\cdots,(\vec{\mathbf{x}}'_{N'},\tilde{y}'_{N'})\}$ 。

   对于样本 $\vec{\mathbf{x}}$ ，假设其真实标记为 $\tilde{y}$ ，模型预测输出为 $\hat{y}$ 。
2. 理论上性能度量都是在测试集上进行。
   - 如果是在训练集上度量，则相应的指标为：训练准确率、训练错误率、训练 auc ...
   - 如果是在验证集上度量，则相应的指标为：验证准确率、验证错误率、验证 auc ...

7.1 分类问题性能度量

7.1.1 准确率、错误率

1. 测试准确率：测试数据集上的准确率（其中 I为示性函数）：

$$r_{test}=\frac{1}{N'}\sum_{i=1}^{N'}I(\tilde{y}'_i=\hat{y}'_i)$$

   准确率衡量的是有多少比例的样本被正确判别。
2. 测试错误率：测试数据集上的错误率：

$$e_{test}=\frac{1}{N'}\sum_{i=1}^{N'}I(\tilde{y}_i'\neq\hat{y}_i')$$

错误率衡量的是有多少比例的样本被判别错误，它也是损失函数为 0-1 损失时的测试误差。

7.1.2 查准率、查全率

1. 对于二分类问题，通常将关注的类作为正类，其他类作为负类。令：
   - `TP`：分类器将正类预测为正类的数量(`True Positive`)。即：真`正类`的数量。
   - `FN`：分类器将正类预测为负类的数量(`False Negative`)。即：假`负类`的数量。
   - `FP`：分类器将负类预测为正类的数量(`False Positive`)。即：假`正类`的数量。
   - `TN`：分类器将负类预测为负类的数量(`True Negative`)。即：真`负类`的数量。

   分类结果的混淆矩阵(`confusion matrix`)定义为：

	预测：正类	**预测：反类**
真实：正类	TP	FN
真实：反类	FP	TN

2. 查准率(`precision`)：$P=\frac{TP}{TP+FP}$。

   它刻画了所有预测为正类的结果中，真正的正类的比例。

3. 查全率(`recall`)：$R=\frac{TP}{TP+FN}$。

   它刻画了真正的正类中，被分类器找出来的比例。

4. 不同的问题中，有的侧重差准率，有的侧重差全率。
   - 对于推荐系统，更侧重于查准率。即推荐的结果中，用户真正感兴趣的比例。因为给用户展示的窗口有限，必须尽可能的给用户展示他真实感兴趣的结果。
   - 对于医学诊断系统，更侧重与查全率。即疾病被发现的比例。因为疾病如果被漏诊，则很可能导致病情恶化。
5. 查准率和查全率是一对矛盾的度量。一般来说查准率高时查全率往往偏低，而查全率高时查准率往往偏低。
   - 如果希望将所有的正例都找出来（查全率高），最简单的就是将所有的样本都视为正类，此时有`FN=0`。此时查准率就偏低（准确性降低）。
   - 如果希望查准率高，则可以只挑选有把握的正例。最简单的就是挑选最有把握的那一个样本。此时有`FP=0`。此时查全率就偏低（只挑出了一个正例）。

7.1.3 P-R 曲线

1. 对二类分类问题，可以根据分类器的预测结果对样本进行排序：排在最前面的是分类器认为“最可能”是正类的样本，排在最后面的是分类器认为“最不可能”是正类的样本。

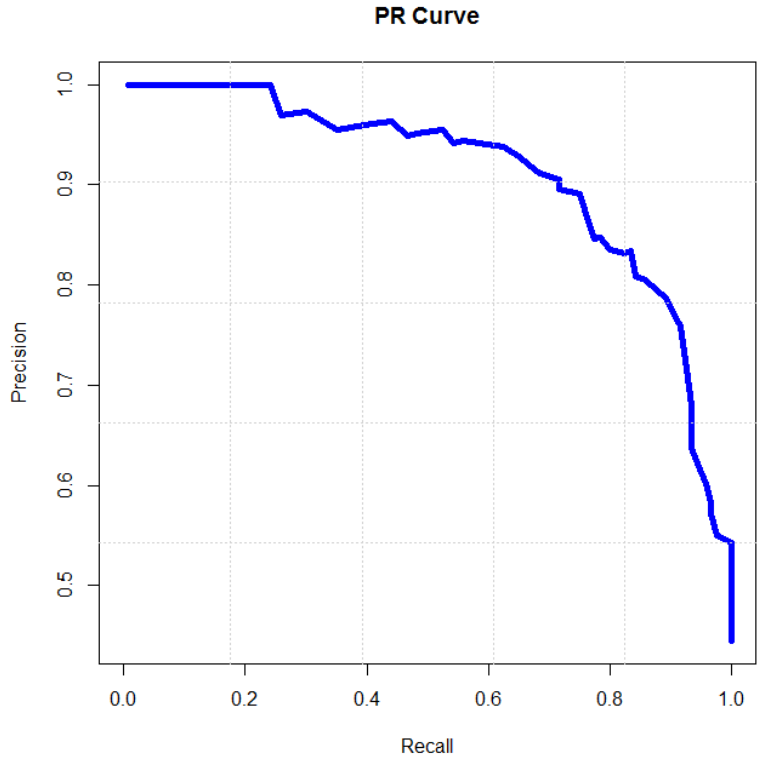
   假设排序后的样本集合为 $(\vec{\mathbf{x}}_1,\tilde{y}_1),(\vec{\mathbf{x}}_2,\tilde{y}_2),\cdots,(\vec{\mathbf{x}}_N,\tilde{y}_N)$，预测为正类的概率依次为 $(p_1,p_2,\cdots,p_N)$。

   在第 i 轮，将 p_i 作为分类阈值来。即：

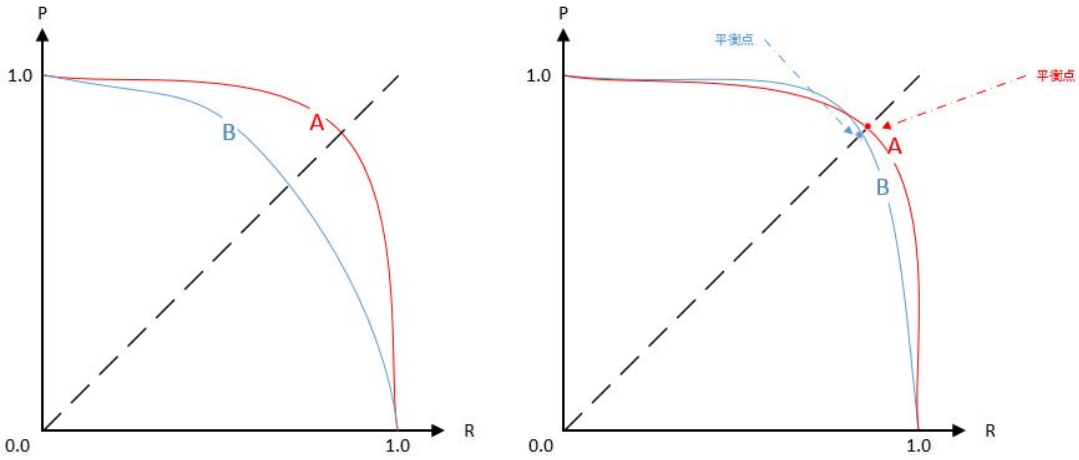
$$\hat{y}_j=\begin{cases}1, & \text{if } p_j\geq p_i\\ 0, & \text{else}\end{cases},\quad j=1,2,\cdots,N$$

   此时计算得到的查准率记做 P_i，查全率记做 R_i。

以查准率为纵轴、查全率为横轴作图，就得到查准率-查全率曲线，简称 `P-R` 曲线。该曲线由点 $\{(R_1,P_1),(R_2,P_2),\cdots,(R_N,P_N)\}$ 组成。



2. `P-R` 曲线从左上角 `(0,1)` 到右下角 `(1,0)` 。
   - 开始时第一个样本（最可能为正例的）预测为正例，其它样本都预测为负类。此时：
     - 查准率很高，几乎为1。
     - 查全率很低，几乎为0，大量的正例没有找到。
   - 结束时所有的样本都预测为正类。此时：
     - 查全率很高，正例全部找到了，查全率为1。
     - 查准率很低，大量的负类被预测为正类。
3. `P-R` 曲线直观显示出分类器在样本总体上的查全率、查准率。因此可以通过两个分类器在同一个测试集上的 `P-R` 曲线来比较它们的预测能力：
   - 如果分类器 `B` 的 `P-R` 曲线被分类器 `A` 的曲线完全包住，则可断言：`A` 的性能好于 `B` 。
   - 如果分类器 `A` 的 `P-R` 曲线与分类器 `B` 的曲线发生了交叉，则难以一般性的断言两者的优劣，只能在具体的查准率和查全率下进行比较。
     - 此时一个合理的判定依据是比较 `P-R` 曲线下面积大小，但这个值通常不容易计算。
     - 可以考察平衡点。平衡点 `Break-Even Point:BEP` 是 `P-R` 曲线上查准率等于查全率的点，可以判定：平衡点较远的 `P-R` 曲线较好。



B被A完全包住　　A与B有交叉

7.1.4 ROC曲线

1. 定义真正例率(`True Positive Rate`) 为： $TPR=\frac{TP}{TP+FN}$ 。

   它刻画了真正的正类中，模型预测为正类的概率。它也就等于正类的查全率。

2. 定义假正例率(`False Positive Rate`) 为：$FPR=\frac{FP}{TN+FP}$ 。

   它刻画了真正的负类中，模型预测为正类的概率。它就等于 1 减去负类的查全率。

3. 对二类分类问题，可以根据分类器的预测结果对样本进行排序：排在最前面的是分类器认为“最可能”是正类的样本，排在最后面的是分类器认为“最不可能”是正类的样本。

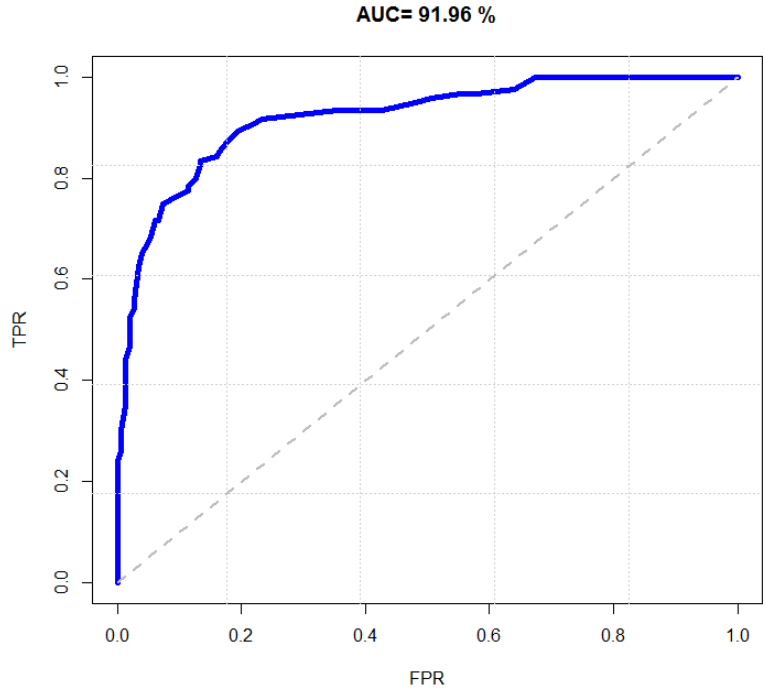
   假设排序后的样本集合为 $(\vec{\mathbf{x}}_1,\tilde{y}_1),(\vec{\mathbf{x}}_2,\tilde{y}_2),\cdots,(\vec{\mathbf{x}}_N,\tilde{y}_N)$ ，预测为正类的概率依次为 $(p_1,p_2,\cdots,p_N)$ 。

   在第 i 轮，将 p_i 作为分类阈值来。即：

   $$\hat{y}_j=\begin{cases}1, & \text{if } p_j\geq p_i\\ 0, & \text{else}\end{cases},\quad j=1,2,\cdots,N$$

   此时计算得到的真正例率记做 TPR_i ，假正例率记做 FPR_i。

   以真正例率为纵轴、假正例率为横轴作图，就得到 `ROC` 曲线。该曲线由点 $\{(TPR_1,FPR_1),(TPR_2,FPR_2),\cdots,(RPR_N,FPR_N)\}$ 组成。



4. `ROC`曲线从左下角`(0,0)`到右上角`(1,1)`。
   - 开始时第一个样本（最可能为正例的）预测为正例，其它样本都预测为负类。此时：
     - 真正例率很低，几乎为0，因为大量的正例未预测到。
     - 假正例率很低，几乎为0，因为此时预测为正类的样本很少，所以几乎没有错认的正例。
   - 结束时所有的样本都预测为正类。此时：
     - 真正例率很高，几乎为1，因为所有样本都预测为正类。
     - 假正例率很高，几乎为1，因为所有的负样本都被错认为正类。
5. 在`ROC`曲线中:
   - 对角线对应于随机猜想模型。
   - 点`(0,1)`对应于理想模型：没有预测错误，`FPR`恒等于0，`TPR`恒等于1。
   - 通常`ROC`曲线越靠近点`(0,1)`越好。
6. 可以通过两个分类器在同一个测试集上的`ROC`曲线来比较它们的预测能力：
   - 如果分类器`A`的`ROC`曲线被分类器`B`的曲线完全包住，则可断言：`B`的性能好于`A`。
   - 如果分类器`A`的`ROC`曲线与分类器`B`的曲线发生了交叉，则难以一般性的断言两者的优劣。

     此时一个合理的判定依据是比较`ROC`曲线下面积大小，这个面积称作`AUC:Area Under ROC Curve`。
7. `P-R`曲线和`ROC`曲线刻画的都是阈值的选择对于分类度量指标的影响。

   通常一个分类器对样本预测的结果是一个概率结果，比如正类概率 0.7。但是样本是不是正类还需要与阈值比较。

   这个阈值会影响了分类器的分类结果，比如：是阈值 0.5 还是阈值 0.9。
   - 如果更重视查准率，则将阈值提升，比如为 0.9 。
   - 如果更看重查全率，则将阈值下降，比如为 0.5 。
8. `P-R`曲线和`ROC`曲线上的每一个点都对应了一个阈值的选择，该点就是在该阈值下的`(查准率，查全率)`/`(真正例率，假正例率)`。

   沿着横轴的方向对应着阈值的下降。

9. `AUC` 是 `ROC` 曲线的面积，其物理意义为：从所有正样本中随机挑选一个样本，模型将其预测为正样本的概率为 p_1 ；从所有负样本中随机挑选一个样本，模型将其预测为正样本的概率为 p_0 。$p_1 > p_0$ 的概率就等于 `AUC` 。
   - 如果对完全随机的对样本进行分类，则 $p_1 > p_0$ 的概率为0.5，因此 `AUC=0.5` 。
   - `AUC` 在样本不平衡的条件下依然适用。如：在反欺诈场景下，假设正常用户为正类（设占比 99.9%），欺诈用户为负类（设占比 0.1%）。

     如果使用准确率评估，则将所有用户预测为正类即可获得 99.9%的准确率。很明显这并不是一个很好的预测结果，因为欺诈用户全部未能找出。

     如果使用 `AUC` 评估，则此时 `FPR=1,TPR=1` ，对应的 `AUC=0.5` 。因此 `AUC` 成功的指出了这并不是一个很好的预测结果。
   - `AUC` 反应的是模型对于样本的排序能力（根据样本预测为正类的概率来排序）。如：`AUC=0.8` 表示：给定一个正样本和一个负样本，在 `80%` 的情况下，模型对正样本预测为正类的概率大于对负样本预测为正类的概率。
   - `AUC` 对于均匀采样不敏感。如：上述反欺诈场景中，假设对正常用户进行均匀的降采样。任意给定一个负样本 neg_i，设模型对其预测为正类的概率为 p_{neg_i} 。降采样前后，由于是均匀采样，因此预测为正类的概率大于 p_{neg_i} 和小于 p_{neg_i} 的真正样本的比例没有发生变化。因此 `AUC` 保持不变。

     但是如果是非均匀的降采样，则预测为正类的概率大于 p_{neg_i} 和小于 p_{neg_i} 的真正样本的比例会发生变化，这也会导致 `AUC` 发生变化。
   - 正负样本之间的预测为正类概率之间的差距越大，则 `AUC` 越高。因为这表明正负样本之间排序的把握越大，区分度越高。

     如：在电商场景中，点击率模型的 `AUC` 要低于购买转化模型的 `AUC` 。因为点击行为的成本低于购买行为的成本，所以点击率模型中正负样本的差别要小于购买转化模型中正负样本的差别。

7.1.5 F1 值

1. F_1 为查准率与查全率的调和均值：$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R}$ 。
2. F_1 更一般的形式：$\frac{1}{F_\beta} = \frac{1}{(1+\beta^2)\times P} + \frac{\beta^2}{(1+\beta^2)\times R}$ ，其中 $\beta^2 > 0$ 度量了查全率对查准率的相对重要性。

7.1.6 代价矩阵

1. 实际应用过程中，不同类型的错误所造成的后果可能有所不同。如：将健康人诊断为患者，与将患者诊断为健康人，其代价就不同。

   为权衡不同类型错误所造成的不同损失，可以为错误赋予非均等代价（ `unequal cost` ）。

   对于二类分类问题，可以设定一个“代价矩阵”(`cost matrix`)，其中 $cost_{ij}$ 表示将第 `i` 类样本预测为第 `j` 类样本的代价。通常 $cost_{ii} = 0$ 表示预测正确时的代价为0 。

	预测：第0类	预测：第1类
真实：第0类	0	$cost_{01}$
真实：第1类	$cost_{10}$	0

> 前面讨论的性能度量都隐式的假设均等代价，即 $cost_{01} = cost_{10}$

2. 在非均等代价下，希望找到的不再是简单地最小化错误率的模型，而是希望找到最小化总体代价`total cost`的模型。
3. 在非均等代价下，`ROC`曲线不能直接反映出分类器的期望总体代价，此时需要使用代价曲线`cost curve`。
   - 代价曲线的横轴就是正例概率代价。

     $$P_{+cost}=\frac{p\times cost_{01}}{p\times cost_{01}+(1-p)\times cost_{10}}$$

     其中 p 为正例（第0类）的概率。
   - 代价曲线的纵轴为：

     $$cost_{norm}=\frac{FNR\times p\times cost_{01}+FPR\times(1-p)\times cost_{10}}{p\times cost_{01}+(1-p)\times cost_{10}}$$

     其中：
     - `FPR`为假正例率 $FPR=\frac{FP}{TN+FP}$ 。

       它刻画了模型将真实的负样本预测为正类的概率。
     - `FNR`为假负例率 $FNR=1-TPR=\frac{FN}{TP+FN}$ 。

       它刻画了模型将真实的正样本预测为负类的概率。

7.1.7 宏查准率/查全率、微查准率/查全率

1. 有时候可能得到了多个二分类混淆矩阵。如：在多个数据集上进行训练/测试。

   此时希望在多个二分类混淆矩阵上综合考察查准率和查全率。
2. 假设有 m 个二分类混淆矩阵，有两种方法来综合考察：
   - 宏查准率、宏查全率：先在各个混淆矩阵上分别计算查准率和查全率，记作 $(P_1,R_1),(P_2,R_2),\cdots,(P_m,R_m)$ ；然后计算平均值。

     这样得到的是宏查准率(`macro-P`)，宏查全率(`macro-F`)，宏`F1`(`macro-F1`)：

     $\text{macro-P}=\frac{1}{m}\sum_{i=1}^{m}P_i,\ \text{macro-R}=\frac{1}{m}\sum_{i=1}^{m}R_i,\ \frac{2}{\text{macro-F}_1}=\frac{1}{\text{macro-P}}+\frac{1}{\text{macro-R}}$ 。
   - 微查准率、微查全率：先将个混淆矩阵对应元素进行平均，得到 TP,FP,TN,FN 的平均值，记作 $\overline{TP},\overline{FP},\overline{TN},\overline{FN}$；再基于这些平均值计算微查准率(`micro-P`)，微查全率(`micro-F`)，微`F1`(`micro-F1`)：

     $\text{micro-P}=\frac{\overline{TP}}{\overline{TP}+\overline{FP}},\ \text{micro-R}=\frac{\overline{TP}}{\overline{TP}+\overline{FN}},\ \frac{2}{\text{micro-F}_1}=\frac{1}{\text{micro-P}}+\frac{1}{\text{micro-R}}$ 。

7.2 回归问题性能度量

1. 均方误差`mean square error:MSE` ：$MSE=\frac{1}{N'}\sum_{i=1}^{N'}(\tilde{y}_i'-\hat{y}_i')^2$ 。
2. 均方根误差`root mean squared error:RMSE` ：$RMSE=\sqrt{\frac{1}{N'}\sum_{i=1}^{N'}(\tilde{y}_i'-\hat{y}_i')^2}$ 。
3. 均方根对数误差`root mean squared logarithmic error:RMSLE` ：

   $RMLSE=\sqrt{\frac{1}{N'}\sum_{i=1}^{N'}[\log(\tilde{y}_i')-\log(\hat{y}_i')]^2}$ 。

   为使得`log`有意义，也可以使用：$RMSLE=\sqrt{\frac{1}{N'}\sum_{i=1}^{N'}[\log(\tilde{y}_i'+1)-\log(\hat{y}_i'+1)]^2}$ 。

优势：

- 当真实值的分布范围比较广时（如：年收入可以从 0 到非常大的数），如果使用`MAE、MSE、RMSE`等误差，这将使得模型更关注于那些真实标签值较大的样本。

  而`RMSLE`关注的是预测误差的比例，使得真实标签值较小的样本也同等重要。

- 当数据中存在标签较大的异常值时，`RMSLE`能够降低这些异常值的影响。

4. 平均绝对误差`mean absolute error:MAE`：$MAE=\frac{1}{N'}\sum_{i=1}^{N'}|\tilde{y}_i'-\hat{y}_i'|$。

七、超参数调节

1. 大多数学习算法都有些超参数需要设定。超参数配置不同，学得的模型性能往往有显著差别，这就是参数调节(`parameter tuning`)：对每种超参数配置都训练出模型，然后把对应最好模型的超参数作为结果。
2. 由于很多超参数是在实数范围内取值，因此现实中常用做法是对每个超参数选定一个范围和变化步长。如在`[0,1)`范围内以`0.2`为步长。

   这样选出的超参数可能不是最佳的，但是这是在计算开销和性能之间取折中的结果。

3. 当模型选择完成后，学习算法和超参数配置已经选定，此时应该用数据集 $\mathbb{D}$ 重新训练模型。

   这个模型在训练过程中使用了 N 个样本，这才是最终提交的模型。

7.1 搜索策略

1. 超参数搜索有三种常见的策略：
   - 手动搜索：手动选择超参数。
   - 网格搜索：当超参数的数据相对较少时，这个方法很实用。
   - 随机搜索：通常推荐这种方式。

7.1.1 手动搜索

1. 手动选择超参数需要了解超参数做了些什么，以及机器学习模型如何才能取得良好的泛化。
2. 手动搜索超参数的任务是：在给定运行时间和内存预算范围的条件下，最小化泛化误差。
3. 手动调整超参数时不要忘记最终目标：提升测试集性能。
   - 加入正则化只是实现这个目标的一种方法。
   - 如果训练误差很低，也可以通过收集更多的训练数据来减少泛化误差。

     如果训练误差太大，则收集更多的训练数据就没有意义。

   - 实践中的一种暴力方法是：不断提高模型容量和训练集的大小。

     这种方法增加了计算代价，只有在拥有充足的计算资源时才可行

7.1.2 网格搜索

1. 网格搜索的做法是：
   - 对于每个超参数，选择一个较小的有限值集合去搜索。
   - 然后这些超参数笛卡尔乘积得到多组超参数。
   - 网格搜索使用每一组超参数训练模型，挑选验证集误差最小的超参数作为最好的超参数。
2. 如何确定搜索集合的范围？
   - 如果超参数是数值，则搜索集合的最小、最大元素可以基于先前相似实验的经验保守地挑选出来。

- 如果超参数是离散的，则直接使用离散值。

3. 通常重复进行网格搜索时，效果会更好。假设在集合 `{-1,0,1}` 上网格搜索超参数 α ：
   - 如果找到的最佳值是 1，那么说明低估了 α 的取值范围。此时重新在 `{1,2,3}` 上搜索。
   - 如果找到的最佳值是 0，那么可以细化搜索范围以改进估计。此时重新在 `{-0.1,0,0.1}` 上搜索。
4. 网格搜索的一个明显问题时：计算代价随着超参数数量呈指数级增长。

   如果有 m 个超参数，每个最多取 n 个值，那么所需的试验数将是 $O(n^m)$ 。虽然可以并行试验，但是指数级增长的计算代价仍然不可行。

7.1.3 随机搜索

1. 随机搜索是一种可以替代网格搜索的方法，它编程简单、使用方便、能更快收敛到超参数的良好取值。
   - 首先为每个超参数定义一个边缘分布，如伯努利分布（对应着二元超参数）或者对数尺度上的均匀分布（对应着正实值超参数）。
   - 然后假设超参数之间相互独立，从各分布中抽样出一组超参数。
   - 使用这组超参数训练模型。
   - 经过多次抽样 -> 训练过程，挑选验证集误差最小的超参数作为最好的超参数。
2. 随机搜索的优点：
   - 不需要离散化超参数的值，也不需要限定超参数的取值范围。这允许我们在一个更大的集合上进行搜索。
   - 当某些超参数对于性能没有显著影响时，随机搜索相比于网格搜索指数级地高效，它能更快的减小验证集误差。
3. 与网格搜索一样，通常会基于前一次运行结果来重复运行下一个版本的随机搜索。
4. 随机搜索比网格搜索更快的找到良好超参数的原因是：没有浪费的实验。
   - 在网格搜索中，两次实验之间只会改变一个超参数（假设为 β ）的值，而其他超参数的值保持不变。

     如果这个超参数 β 的值对于验证集误差没有明显区别，那么网格搜索相当于进行了两个重复的实验。
   - 在随机搜索中，两次实验之间，所有的超参数值都不会相等，因为每个超参数的值都是从它们的分布函数中随机采样而来。因此不大可能会出现两个重复的实验。
   - 如果 β 超参数与泛化误差无关，那么不同的 β 值：
     - 在网格搜索中，不同 β 值、相同的其他超参数值，会导致大量的重复实验。
     - 在随机搜索中，其他超参数值每次也都不同，因此不大可能出现两个重复的实验（除非所有的超参数都与泛化误差无关）。

7.2 调整原则

1. 通常先对超参数进行粗调，然后在粗调中表现良好的超参数区域进行精调。
2. 超参数随机搜索，并不意味着是在有效范围内随机均匀取值。需要选择合适的缩放来进行随机选取。
   - 对于学习率，假设其取值范围为 `0.000001~1`。

     如果进行均匀取值，取10个，那么有 90% 的随机值都位于区间 `[0.1,1]`。则 `[0.000001,0.1]` 之间没有足够的探索。这种做法明显不合理。

     此时需要使用对数缩放，在对数轴上均匀随机取点。
   - 对于指数加权移动平均的超参数 β。假设其取值范围为 `0.9~0.9999`。

     由于 $\frac{1}{1-\beta}$ 刻画了结果使用过去多少个周期的数据来加权平均。因此如果进行均匀取值，则：

- β 在 `0.9~0.9005` 之间取值时， $\frac{1}{1-\beta}$ 变化不大。
- β 在 `0.9990~0.9995` 之间取值时， $\frac{1}{1-\beta}$ 变化非常大。

β 越接近 1， $\frac{1}{1-\beta}$ 对于它的变化越敏感。此时，需要对 $1-\beta$ 使用对数缩放，在对数轴上均匀随机取点。

- 如果选择了错误的缩放，如果取值的总量足够大，也可以得到不错的结果。

  尤其当配合了 `粗调 -> 精调` 策略时，最终还是会聚焦到合适的超参数范围上。

3. 通常情况下，建议至少每隔几个月重新评估或者修改超参数。因为随着时间的变化，真实场景的数据会逐渐发生改变：
   - 可能是由于用户的行为、偏好发生了改变。
   - 可能是采样的方式发生了改变。
   - 也可能仅仅是由于数据中心更新了服务器。

   由于这些变化，原来设定的超参数可能不再适用。
4. 有两种超参数调整策略：
   - 如果数据足够大且没有足够的计算资源，此时只能一次完成一个试验。

     则可以每天观察模型的表现，实时的、动态的调整超参数。
   - 如果数据不大，有足够的计算资源可以同一时间完成大量的试验，则可以设置多组超参数设定，然后选择其中表现最好的那个。

八、传统机器学习的挑战

1. 传统机器学习算法的两个困难：
   - 维数灾难：当数据的维数很高时，很多机器学习问题变得相当困难。因为许多传统机器学习算法简单地假设：**一个新样本的输出应该大致与最接近的训练样本的输出相同** 。
   - 选择性偏好：某些算法偏好于选择某类函数。

     最广泛的隐式偏好是：要学习的函数是平滑的或者局部不变性的。

     这个先验知识表明：要学习的函数不会在一个小区域内发生较大的变化。很多简单算法完全依赖此先验知识来达到良好的泛化。