Lecture 3

汇报人：扣扣

大纲

- 机器学习项目流程
- 逻辑回归
- 偏差与方差

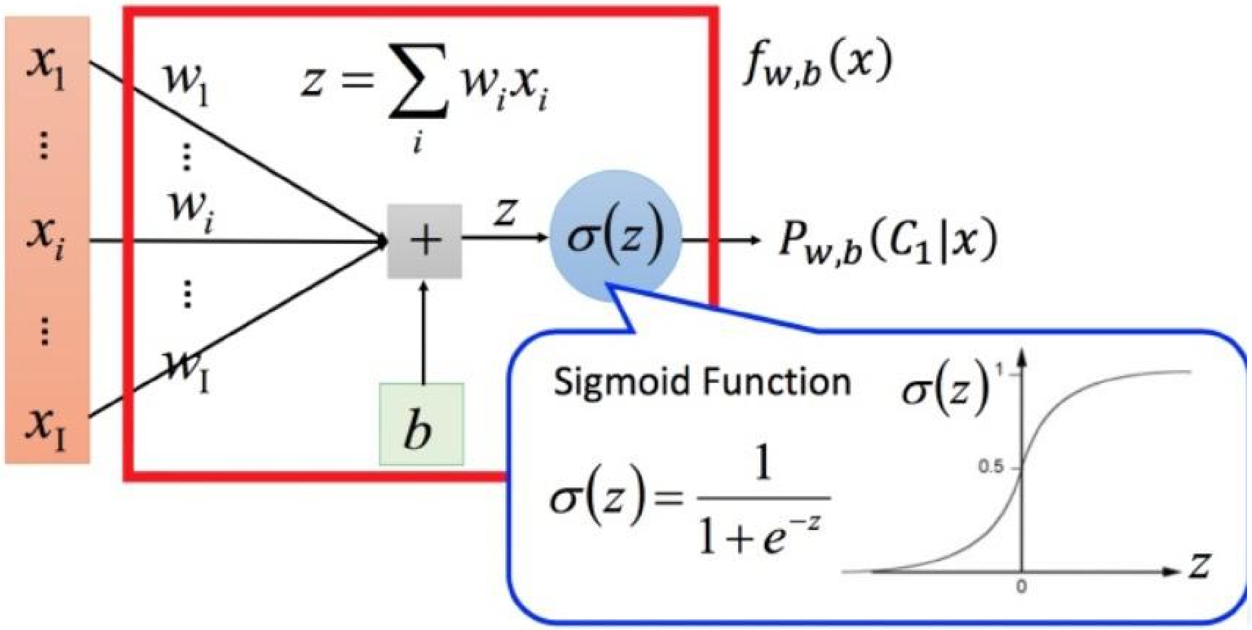
机器学习项目流程

项目启动	特征工程	模型搭建	训练与优化	服务上线
目标 用户习惯 性能偏重 系统评估 个性化配置 局限性	数据资源 数据成本 数据偏向性 数据平衡性 数据表征 （词向量）	模型搭建原则：由易至难，循序渐进 （逻辑回归）	系统性、科学性地调整优化 （方差与偏差）	训练过程与服务过程并非独立 模型改进的取舍 可解释性

逻辑回归

大纲

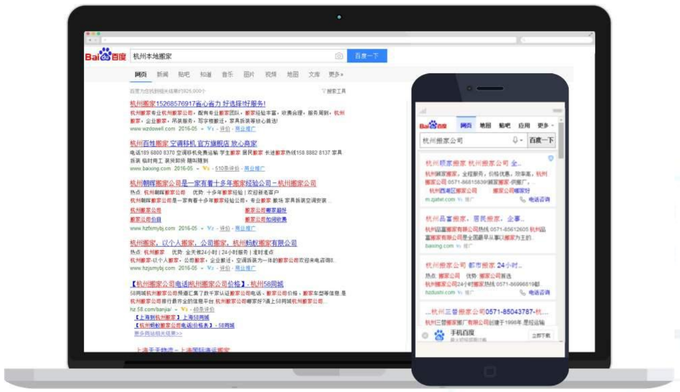
- 应用
- 逻辑回归 VS 线性回归
- 逻辑回归详解
- 优缺点
- 思考题
- python实现逻辑回归



应用

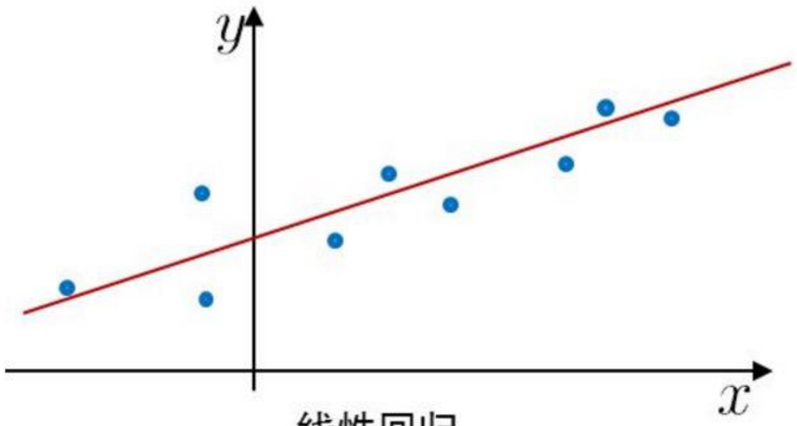
逻辑回归（Logistic Regression）主要解决二分类问题，用来表示某件事情发生的可能性。

- 一封邮件是垃圾邮件的可能性（是、不是）
- 用户购买一件商品的可能性（买、不买）
- 广告被点击的可能性（点、不点）



逻辑回归 VS 线性回归

分类和回归是机器学习可以解决两大主要问题，从预测值的类型上来区分，连续变量的预测称为回归；离散变量的预测称为分类。
例如：预测明天多少度，是一个回归任务；预测明天阴、晴、雨，就是一个分类任务。

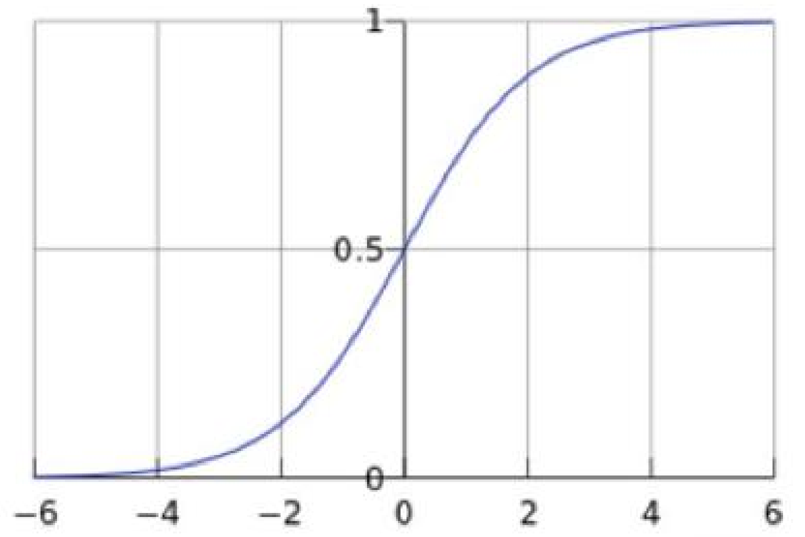


线性回归

- 在一维特征空间，线性回归是通过学习一条直线 $h_\theta(x)=\theta_0+\theta_1*x$，使得这条直线尽可能拟合所有已有的看到的点的y值（观测数据），并希望未看到的数据（测试数据）也尽可能落在这条线上（泛化性能），h_θ是预测值，y是实际值。
- 在多维空间，线性回归表示为：$h_\theta(x)=\theta_0+\theta_1*x_1+\theta_2*x_2+\cdots+\theta_n*x_n$

逻辑回归 VS 线性回归

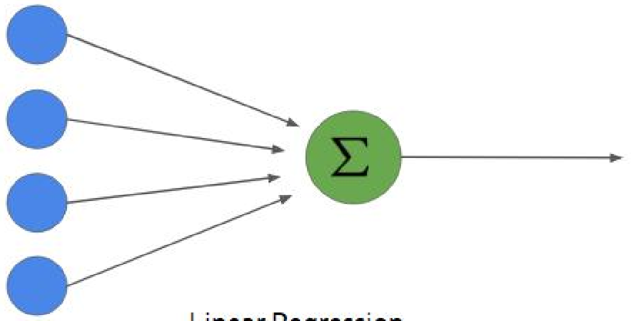
如何在回归的基础上进行二分类？



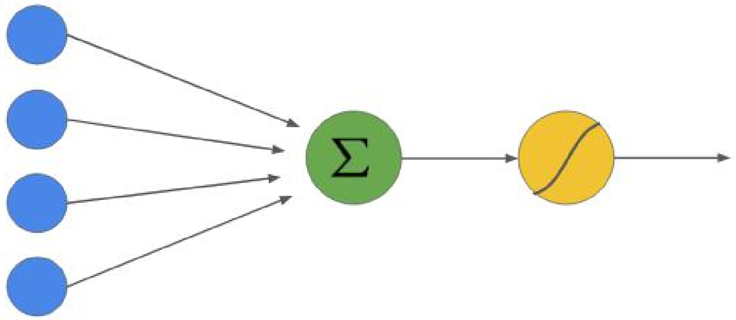
sigmoid函数

- 在多维空间，线性回归表示为：
  hθ(x)=θ0+θ1*x1+θ2*x2+…+θn*xn
- 逻辑回归表示为：
  g(x) = sigmoid(hθ(x))
- 将实际值映射到到0，1之间，如果>=0.5，则预测属于正例；如果<0.5，则预测属于负例。

逻辑回归 VS 线性回归



Linear Regression



Logistic Regression

线性回归 解决回归问题	线性回归 连续的变量	线性回归 符合线性关系	线性回归 直观表达变量关系
逻辑回归 解决分类问题	逻辑回归 离散的变量	逻辑回归 可以不符合线性关系	逻辑回归 无法直观表达变量关系

逻辑回归

数学表达式：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

现在有m个样本，如何求解参数值？逻辑回归的目标函数是什么？

极大似然估计，即利用已知的样本结果信息，反推最具有可能（最大概率）导致这些样本结果出现的模型参数值！

极大似然估计

$$P(y|x,\theta) = (h_\theta(x))^y(1-h_\theta(x))^{1-y}$$

$$P(y=1|x,\theta) = h_\theta(x)$$

$$L(\theta) = \prod_{i=1}^{m} P(yi|xi,\theta) = \prod_{i=1}^{m} (h_\theta(xi))^{yi}(1-h_\theta(xi))^{1-yi}$$

$$P(y=0|x,\theta) = 1-h_\theta(x)$$

$J(\theta)$
$= -1/m * logL(\theta)$
$=-1/m*\sum_{i=1}^{m} (yi * log(h_\theta(xi)) + (1-yi)log(1-h_\theta(xi)))$

y表示分类标签，x表示输入，θ表示参数

i指的是第i个样本，一共有m个样本
逻辑回归的目标函数为最大对数似然函数的相反数

交叉熵

交叉熵（Cross Entropy）是Shannon信息论中一个重要概念，主要用于度量两个概率分布间的差异性信息（越小，分布越相近）。假设现在有一个样本集中两个概率分布p,q，其中p为真实分布，q为非真实（预测）分布。对于离散变量采用以下的方式计算：

$$H(p,q)=\sum_{x} p(x)\cdot log\left(\frac{1}{q(x)}\right)$$

梯度下降法更新参数

$$\theta_j := \theta_j - \alpha * \frac{\partial J(\theta)}{\partial \theta_j}$$

梯度下降法更新公式

$$h(z) = \frac{1}{1+e^{-z}}$$

$$h'(z) = h(z) * (1 - h(z))$$

sigmoid导数

$$\frac{\partial J(\theta)}{\partial \theta_j} = -1/m \sum_{i=1}^{m} (yi * \frac{\partial log h_\theta(xi)}{\partial \theta_j} + (1-yi) * \frac{\partial log(1-h_\theta(xi))}{\partial \theta_j})$$

$$= -1/m \sum_{i=1}^{m} (yi * \frac{1}{h_\theta(xi)} * \frac{\partial h_\theta(xi)}{\partial \theta_j} + (1-yi) * \frac{1}{1-h_\theta(xi)} * \frac{\partial(1-h_\theta(xi))}{\partial \theta_j})$$

$$= -1/m \sum_{i=1}^{m} (\frac{yi}{h_\theta(xi)} + \frac{yi-1}{1-h_\theta(xi)}) * \frac{\partial h_\theta(xi)}{\partial \theta_j})$$

$$= -1/m \sum_{i=1}^{m} (\frac{yi-h_\theta(xi)}{h_\theta(xi)(1-h_\theta(xi))} * h_\theta(xi) * (1-h_\theta(xi)) \frac{\partial \theta^T X}{\partial \theta_j})$$

$$= -1/m \sum_{i=1}^{m} (yi - h_\theta(xi)) xi_j)$$

梯度

逻辑回归的优缺点

优点：

- 实现简单，广泛应用于工业问题；
- 分类时计算量非常小，速度很快，存储资源低；
- 便利的观测样本概率分数；
- 计算代价不高，易于理解和实现。

缺点：

- 当特征空间很大时，逻辑回归的性能不是很好；
- 容易欠拟合，一般准确度不太高
- 不能很好地处理大量多类特征或变量；

思考题

- 能否用最平方损失作为损失函数？
- 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？

思考题

能否用最平方损失作为损失函数？

$$j(\theta) = \frac{1}{2}\sum_{i=1}^{n}(z(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial \theta_j} j(\theta) = (z(x) - y) \cdot z'(x) \cdot x_j$$

由此可以看出，参数除了跟真实值与预测值之间的差距有关外，还和sigmoid函数在该点的导数有关，当这个点越靠近两端的时候梯度会变得非常小，这样会导致即使当真实值与预测值差距很大时，参数变化的非常缓慢，与我们的期望不符合。

思考题

- 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？
  - ✓ 易于求导？
  - ✓ 最大熵
  - ✓ 广义线性回归

思考题

■ 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？

指数族分布 (The exponential family distribution),在概率统计中，若某概率分布满足下式，我们就称之属于指数族分布。

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

确定了T,a,b,我们就可以确定某个参数为η的指数族分布，统计中很多熟悉的概率分布都是指数族分布的特定形式，如伯努利分布，高斯分布，多项分布 (multionmal), 泊松分布等。

思考题

- 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

$$\begin{aligned} p(y;\phi) &= \phi^y(1-\phi)^{1-y} \\ &= exp[y\log\phi + (1-y)\log(1-\phi)] \\ &= exp[y\log\frac{\phi}{1-\phi} + log(1-\phi)] \end{aligned}$$

$$\begin{aligned} T(y) &= y \\ \eta &= \log\frac{\phi}{1-\phi} \\ a(\eta) &= -\log(1-\phi) = \log(1+e^\eta) \\ b(y) &= 1 \end{aligned}$$

伯努利分布的指数族分布形式

思考题

- 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？

线性回归，逻辑回归均属于广义线性模型（Generalized linear model, GLM）

考虑一个分类或回归问题，目标为预测某个随机变量y，y是某些特征(feature)x的函数，有如下三个假设：

- $p(y|x;\theta)$服从指数族分布
- 给定x，我们的目的是为了预测T(y)在条件x下的期望。一般情况T(y)=y，这就意味着我们希望预测y = $h(x)=E[y|x]$
- 指数族分布的参数η和输入x线性相关： $\eta=\theta^T x$

思考题

- 逻辑回归为什么要选择sigmoid函数的形式，而不是其他将数值映射到0到1之间的形式？

考虑LR二分类问题，y∈0,1，因为是二分类问题，选择p(y|x;θ)~Bernoulli(ϕ)，即服从伯努利分布。

$$
\begin{aligned}
h_\theta(x) &= E[y|x;\theta] \\
&= \phi \\
&= \frac{1}{1+e^{-\eta}} \\
&= \frac{1}{1+e^{-\theta^T x}}
\end{aligned}
$$

- y在条件x下的期望
- 伯努利分布的期望
- 伯努利分布为指数族分布时的推导
- 假设参数η和输入x线性相关

python实现逻辑回归

课后阅读

应用sklearn进行逻辑回归

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

参考资料

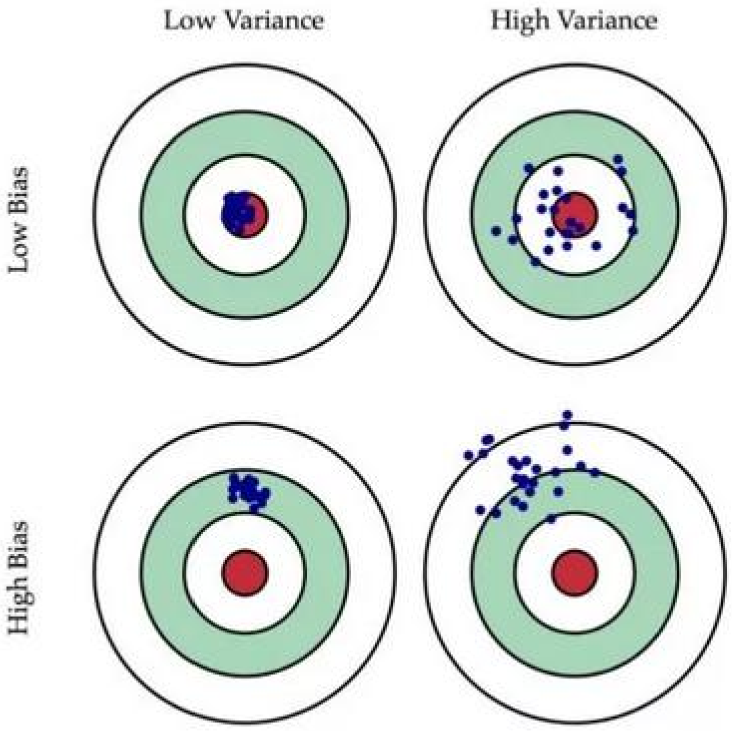
逻辑回归相关

- https://blog.csdn.net/jk123vip/article/details/80591619
- https://zhuanlan.zhihu.com/p/53387812
- https://www.dazhuanlan.com/2019/10/16/5da61ff59b8fb/
- http://easyai.tech/ai-definition/logistic-regression/
- https://towardsdatascience.com/building-a-logistic-regression-in-python-301d27367c24
- https://www.jianshu.com/p/a8d6b40da0cf

偏差与方差

大纲

- 前置知识
- 问题引入
- 概念定义
- 偏差与方差举例
- 减少偏差、方差的技术
- 偏差与方差间的权衡
- 参考资料



前置知识

- 训练集（training set）：
  训练算法。

- 开发集（development set）：
  调整参数、选择特征，以及对学习算法作出其它调整的决定。

- 测试集（test set）：
  开发集中选出的最优模型在测试集上进行评估，但不会据此改变学习算法或参数。（不能根据测试集指标对算法做出任何决策）

问题引入

- Q：目标为构建一个误差在 5%以内的猫狗二分类识别器，目前的训练集错误率为 15%，开发集错误率为 16%，这时候应该怎么办？

- A1：通过添加层/神经元数量来增加神经网络的大小。
- A2：增加训练集的数据量。

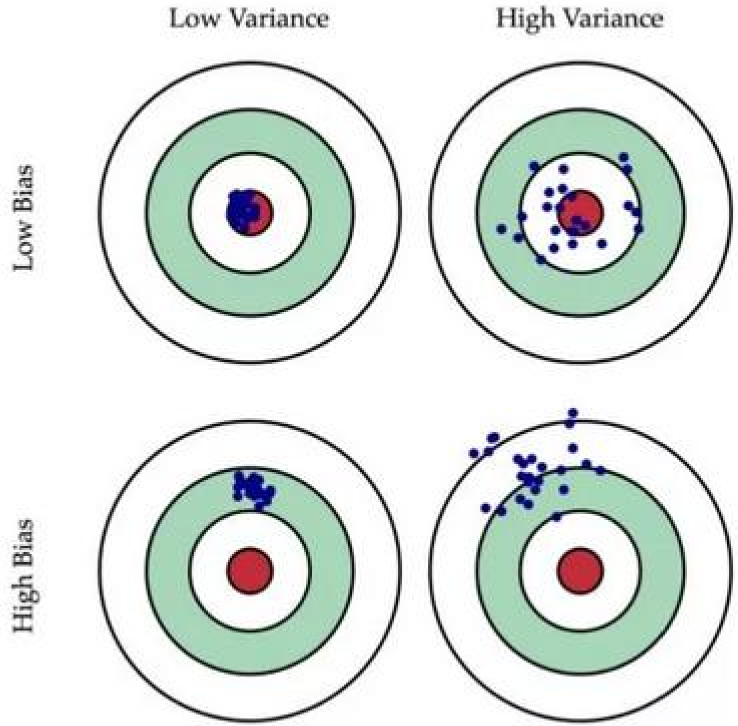
概念定义

- 偏差（bias）：

度量了学习算法的期望预测与真实结果的偏离程度，即刻画了算法本身的拟合能力，偏差越大，表明越偏离真实值。

- 方差（variance）：

度量了同样大小训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响，也就可以理解为衡量模型的稳定性（鲁棒性，泛化能力）。



概念定义

目标为构建一个误差 5%以内 的猫狗二分类识别器，目前的训练集错误率为 15%，开发集错误率为 16%。

- 偏差 (bias) ：算法在训练集上的错误率，在本例中是 15%。
- 方差 (variance) ：算法在开发集上的表现比训练集上差多少。在本例中，开发集表现比训练集差 1%。

- 误差 = 偏差 + 方差
- 粗略地说，偏差指的是算法在大型训练集上的错误率；方差指的是算法在测试集上的表现低于训练集的程度。

偏差和方差举例

- 训练错误率 = 1%
- 开发错误率 = 11%

方差为 10%（=11%-1%），很高，泛化能力弱，这也被叫做过拟合（overfitting）。

- 训练错误率 = 15%
- 开发错误率 = 16%

可知偏差为 15%，方差为 1%。该分类器的错误率高，没有很好地拟合训练集，但它在开发集上的误差不比在训练集上的误差高多少。因此，该分类器具有较高的偏差（high bias），该算法欠拟合（underfitting）。

偏差和方差举例

- 训练错误率 = 15%
- 开发错误率 = 30%

- 该分类器同时有高偏差和高方差（high bias and high variance），在训练集上表现得很差，因此有较高的偏差，在开发集上表现更差，方差同样较高（由于该分类器同时过拟合和欠拟合，过拟合/欠拟合术语很难准确应用于此）。

- 训练错误率 = 0.5%
- 开发错误率 = 1%

- 算法棒棒的。

偏差

- 假设现有一个珍稀动物识别系统，任务难度大，即便由人类来区分，也存在14%的错误率（最优错误率）。

现在算法达到：

- 训练错误率 = 15%
- 开发错误率 = 30%

可以将训练错误率（偏差）分解如下：

- 最优错误率（"不可避免偏差"）：14%，可以将其认为是学习算法的偏差"不可避免"的部分。
- 可避免偏差：1%。即训练错误率和最优误差率之间的差值。

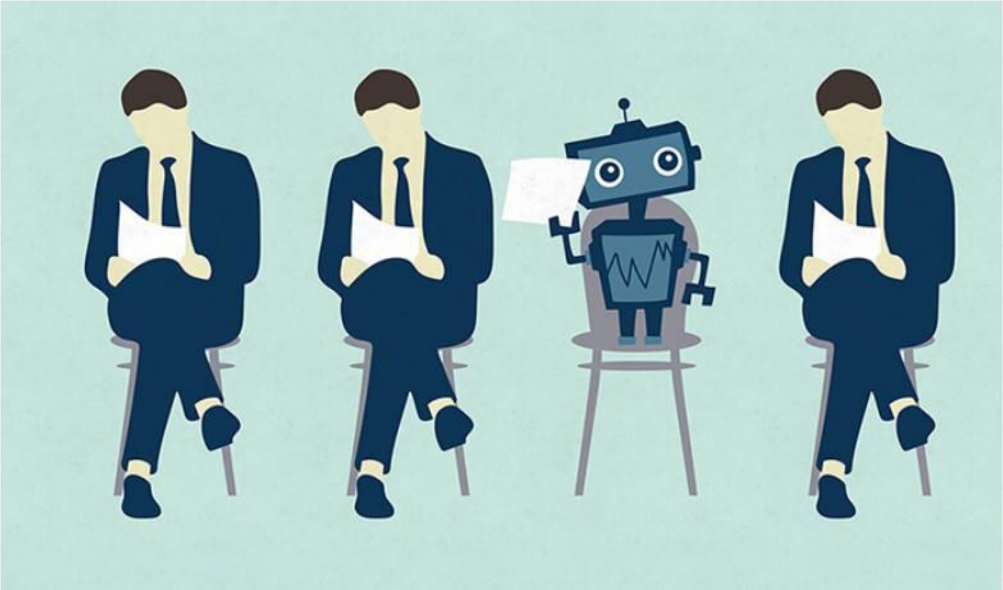
偏差

- 偏差 = 最佳误差率（“不可避免偏差”）+ 可避免的偏差

- 如何才能知道最优错误率是多少呢？

对于人类擅长的任务，例如图片识别或音频剪辑转录，测评人为标签相对于训练集标签的精度，这将给出最优错误率的估计。

如果是一项人类也很难解决的问题（例如股价预测，世界局势预测等），很难估计最优错误率。



减少可避免偏差的技术

- 加大模型规模（例如神经元/层的数量）
- 根据误差分析结果修改输入特征
- 减少或者去除正则化（L2 正则化，L1 正则化，dropout）
- 修改模型架构（比如神经网络架构）

减少方差的技术

- 添加更多的训练数据
- 加入正则化（L2 正则化，L1 正则化，dropout）
- 加入提前终止（Early stopping，例如根据开发集误差提前终止梯度下降）
- 通过特征选择减少输入特征的数量和种类
- 减小模型规模（比如神经元/层的数量）

减少方差的技术

下面是两种额外的策略，和解决偏差问题章节所提到的方法重复：

- 根据误差分析结果修改输入特征
- 修改模型架构（比如神经网络架构）

偏差和方差间的权衡

目前，在大部分针对学习算法的改进中，有一些能够减少偏差，但代价是增大方差，反之亦然。于是在偏差和方差之间就产生了“权衡”。

- 能够获取充足的数据
- 并且可以使用非常大的神经网络（深度学习）

有更多的选择可以在不损害方差的情况下减少偏差，反之亦然。

如果你选择了一个非常契合任务的模型架构，那么你也可以同时减少偏差和方差。只是选择这样的架构可能有点难度。

问题解决

- Q：目标为构建一个误差在 5%以内的猫狗二分类识别器，目前的训练集错误率为 15%，开发集错误率为 16%，这时候应该怎么办？

- A1：通过添加层/神经元数量来增加神经网络的大小。
- A2：增加训练集的数据量。

参考资料

Machine Learning Yearing