

机器学习基础II： 支撑向量机

▼ 机器学习基础II： 支撑向量机

▼ 背景知识

- Norm范数
- 函数的凹凸性
- 优化问题
- 支撑向量机
- ▼ 模型的迭代
 - 训练集和测试集
 - 模型的评价
 - 泛化能力

背景知识

Norm范数

Norm范数：Norm是将非零向量映射为正数的函数

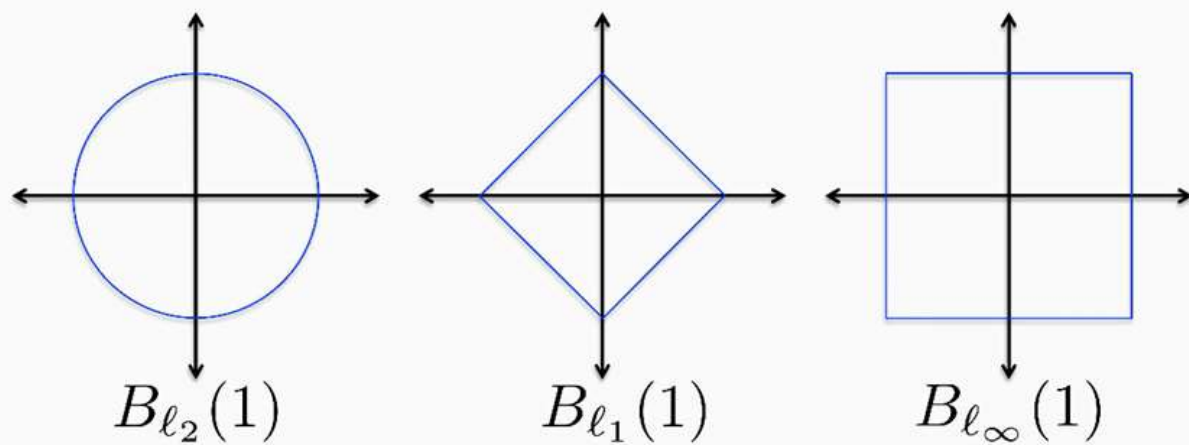
Norm的性质：

1. 齐次性： $||\alpha x|| = |\alpha| ||x||, for\ x \in \mathbb{R}^N\ and\ \alpha \in \mathbb{R}$
2. 三角不等式： $||x + y|| \leq ||x|| + ||y||, for\ x, y \in \mathbb{R}^N$
3. 分离性： $If\ and\ only\ if\ ||x|| = 0, then\ x = 0$

常见的Norm计算公式： $p \geq 1, a \in \mathbb{R}^N, ||a||_p = \left(\sum_{i=1}^N |a_i|^p \right)^{\frac{1}{p}}$

[Lemma](Minkowski's inequality) $1 \leq p \leq \infty, ||a + b||_p \leq ||a||_p + ||b||_p$

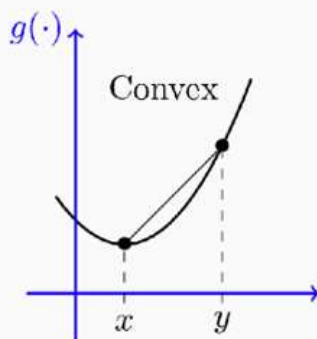
[Definition](l_p ball) $\epsilon \geq 0, B_{l_p}(\epsilon) = B_p(\epsilon) = \{a \mid ||a||_p \leq \epsilon\}$



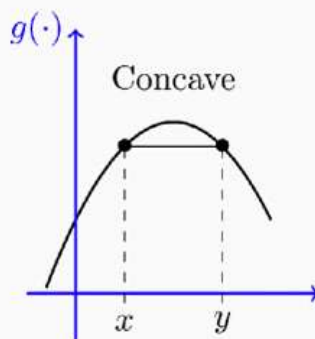
$B_p(1)$ is referred to as the *unit ball* (i.e, $\epsilon = 1$) .

函数的凹凸性

凸函数



$$g(\alpha x + (1 - \alpha)y) \leq \alpha g(x) + (1 - \alpha)g(y).$$



$$g(\alpha x + (1 - \alpha)y) \geq \alpha g(x) + (1 - \alpha)g(y).$$

优化问题

无约束的优化问题

$$\min f_0(x)$$

梯度下降、牛顿法等

有约束的优化问题

$$\min f_0(x)$$

$$s.t. \quad f_i(x) \leq 0, \quad i = 1, \dots, m$$

$$h_i(x) = 0, \quad i = 1, \dots, p$$

拉格朗日法

- 无约束的优化问题使用牛顿法解决：

牛顿法的基本思想是通过构造函数的泰勒级数展开来近似原函数，并使用近似函数的根或最小值来逐步逼近原函数的根或最小值。在每一次迭代中，牛顿法使用当前点的切线来估计函数的根或最小值，并将切线与x轴的交点作为下一次迭代的点。

$$f(x) = \frac{1}{0!}f(x_0) + \frac{1}{1!}(x-x_0)f'(x_0) + \frac{1}{2!}(x-x_0)^2f''(x_0) + \dots + \frac{1}{n!}(x-x_0)^nf^{(n)}(x_0) + R_n$$

截断: $f(x) = \frac{1}{0!}f(x_0) + \frac{1}{1!}(x-x_0)f'(x_0) + \frac{1}{2!}(x-x_0)^2f''(x_0)$

重写公式: $\varphi(x) = \frac{1}{0!}f(x^{(k)}) + \frac{1}{1!}(x-x^{(k)})f'(x^{(k)}) + \frac{1}{2!}(x-x^{(k)})^2f''(x^{(k)})$


求导后令其为0: $\varphi'(x) = f'(x^{(k)}) + f''(x^{(k)})(x-x^{(k)}) = 0$

得到切线与x轴的交点: $x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$

- 有约束的优化问题使用拉格朗日法：

- 将有约束的问题转化为无约束的问题：通过构建拉格朗日函数 $L(x, \lambda, \nu)$ ，令其偏导数都等于0，求得的解都满足原问题的等式约束；随后从这些解里寻找局部最优解

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x)$$


$$\max_{\lambda \geq 0, \nu} \min_x L(x, \lambda, \nu)$$

- 既含等式约束又含不等式约束的优化问题，要使得KKT条件成立：

拉格朗日乘数法

$$f_i(x^*) \leq 0, \quad i = 1, \dots, m$$

$$h_i(x^*) = 0, \quad i = 1, \dots, p$$

$$\lambda_i^* \geq 0, \quad i = 1, \dots, m$$

$$\lambda_i^* f_i(x^*) = 0, \quad i = 1, \dots, m$$

$$\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) = 0$$

KKT条件

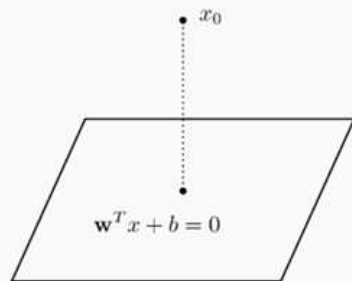
假如先不考虑 $f_i(x) \leq 0$ 这个条件，KKT条件会变为 $h_i(x^*) = \nabla_{v_i} L = 0, \nabla_x f_0(x^*) + \sum_{i=1}^p \nu_i^* \nabla_x h_i(x^*) = 0$ 这两个条件，也就是五个条件里面的第二行和最后一行。

支撑向量机

1. 基本推导

先用拉格朗日法计算点到平面的距离，约束条件是点必须在平面 $w^T x + b = 0$ 上。

点到平面/超平面的距离



$$\min \frac{1}{2} \|x - x_0\|^2$$

$$\text{s.t. } w^T x + b = 0$$

$$\min_x \max_{\beta} \frac{1}{2} \|x - x_0\|^2 + \beta(w^T x + b)$$

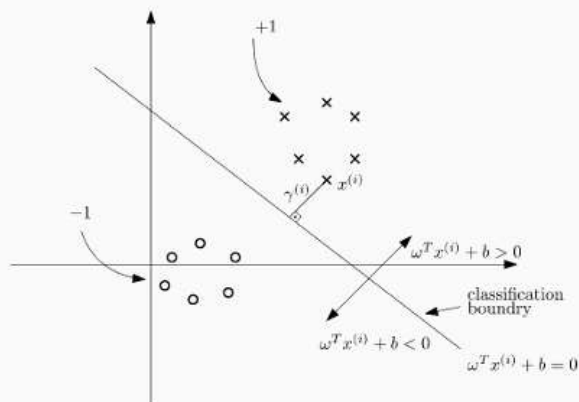
$$\nabla_x L(x, \beta) = x^* - x_0 + \beta^* w = 0$$

$$\nabla_{\beta} L(x, \beta) = w^T x^* + b = 0$$

$$\gamma_0 = \|x_0 - x^*\| = \left| \frac{w^T x_0 + b}{\|w\|^2} w \right| = \frac{|w^T x_0 + b|}{\|w\|}$$

2. 进入SVM

SVM的目标在于：找到一个平面能够正确地分开两个类别，且让两个类别的中任意一个点到该平面的最短距离尽可能大



$$\begin{aligned} \max_{\{w, b\}} \quad & \gamma \\ \text{s.t.} \quad & \gamma^{(i)} \geq \gamma, \quad \forall i \\ & y^{(i)} (w^T x^{(i)} + b) \geq 0, \quad \forall i \end{aligned}$$

1、让所有点到平面的距

离都比 γ 要大

2、第二个条件分类正确

$$\gamma_0 = \|x_0 - x^*\| = \left| \frac{w^T x_0 + b}{\|w\|^2} w \right| = \frac{|w^T x_0 + b|}{\|w\|}$$

点到平面的距离

对于第一个条件，代入点到平面的距离公式；第二条件代入分类对的条件进行化简

代入点到平面的距离

$$\begin{aligned} \max_{\{w, b\}} \quad & \gamma \\ \text{s.t.} \quad & \frac{|w^T x^{(i)} + b|}{\|w\|} \geq \gamma, \quad \forall i \\ & y^{(i)} (w^T x^{(i)} + b) \geq 0, \quad \forall i \end{aligned}$$

$$y^{(i)} (w^*{}^T x^{(i)} + b^*) = |w^*{}^T x^{(i)} + b^*|$$

对于分对的样本

$$\begin{aligned} \max_{\{w, b, \gamma\}} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)} (w^*{}^T x^{(i)} + b^*) \geq \gamma \|w\| \end{aligned}$$

因为我们的目标只需求一个平面表达式，下面去掉 λ 和 w

令 $\gamma' = \gamma ||\mathbf{w}'||$

$$\begin{aligned} \max_{\{\mathbf{w}, b, \gamma'\}} \quad & \frac{\gamma'}{||\mathbf{w}'||} \\ \text{s.t.} \quad & y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq \gamma' \end{aligned}$$

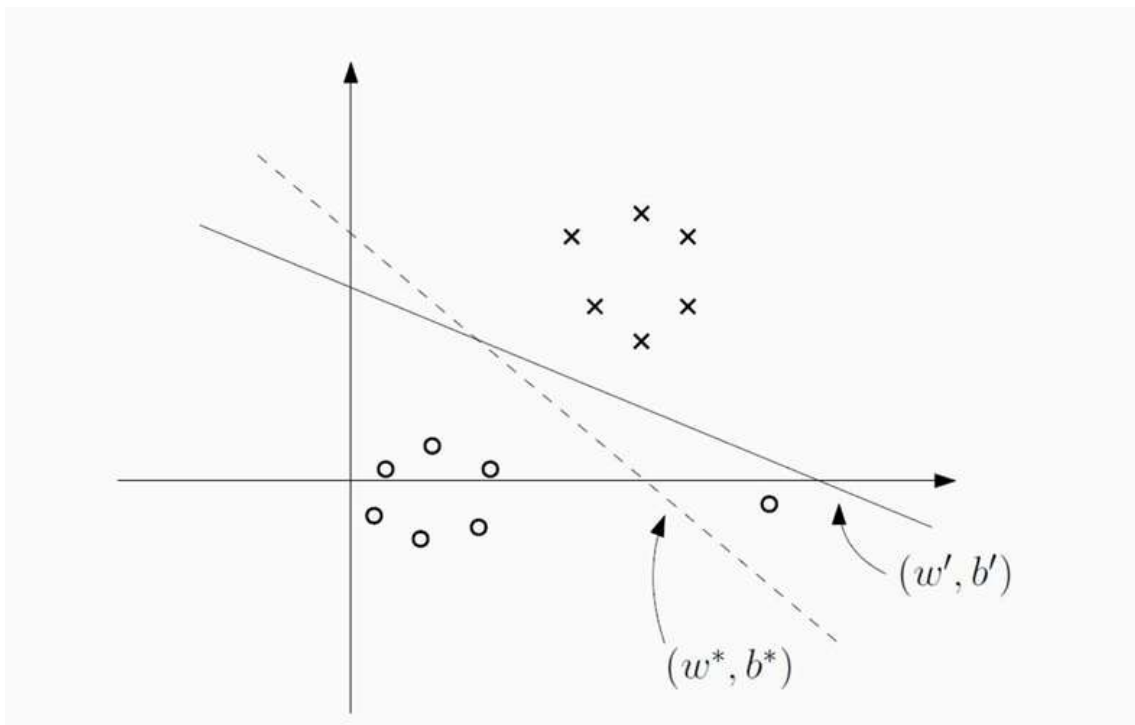
对间隔进行归一

$$\begin{aligned} \max_{\{\mathbf{w}, b\}} \quad & \frac{1}{||\mathbf{w}'||} \\ \text{s.t.} \quad & y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1, \quad \forall i \end{aligned}$$

从max到min

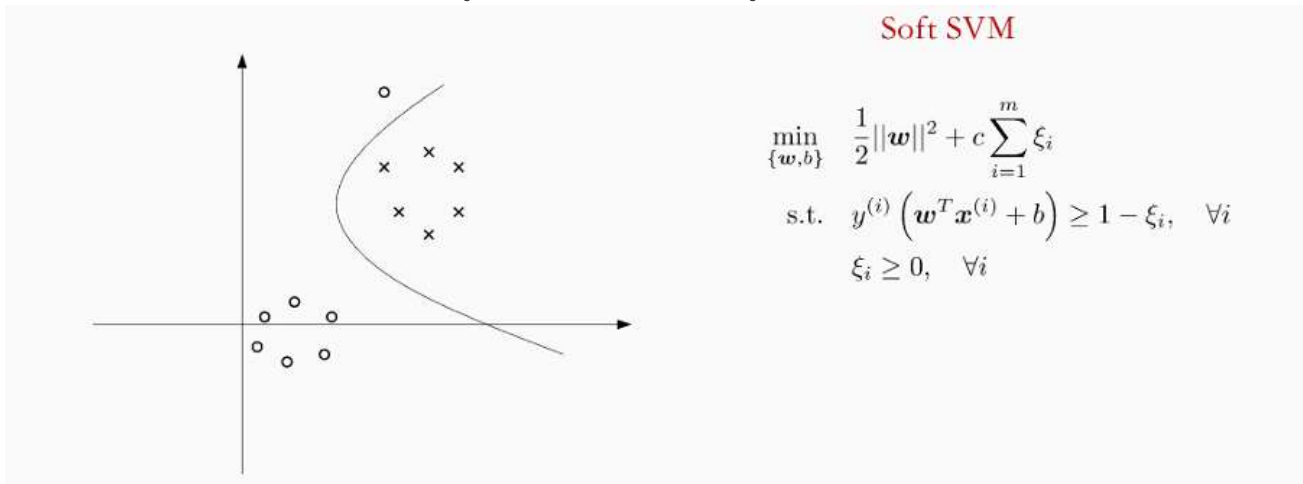
$$\begin{aligned} \min_{\{\mathbf{w}, b\}} \quad & \frac{1}{2} ||\mathbf{w}'||^2 \\ \text{s.t.} \quad & y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1, \quad \forall i \end{aligned}$$

以上SVM的形式叫Hard SVM，它不允许分类出错；这可能导致模型为一些与真实分布差的很远的样本而划出一个较差的平面



因此，我们引入Soft SVM：对于第 i 个点，若 $\xi_i = 0$,即该点分类正确；若 $\xi_i > 0$ ，分类错误。我们在优化时需要出错的累积最小 $\min \sum_{i=1}^m \xi_i$

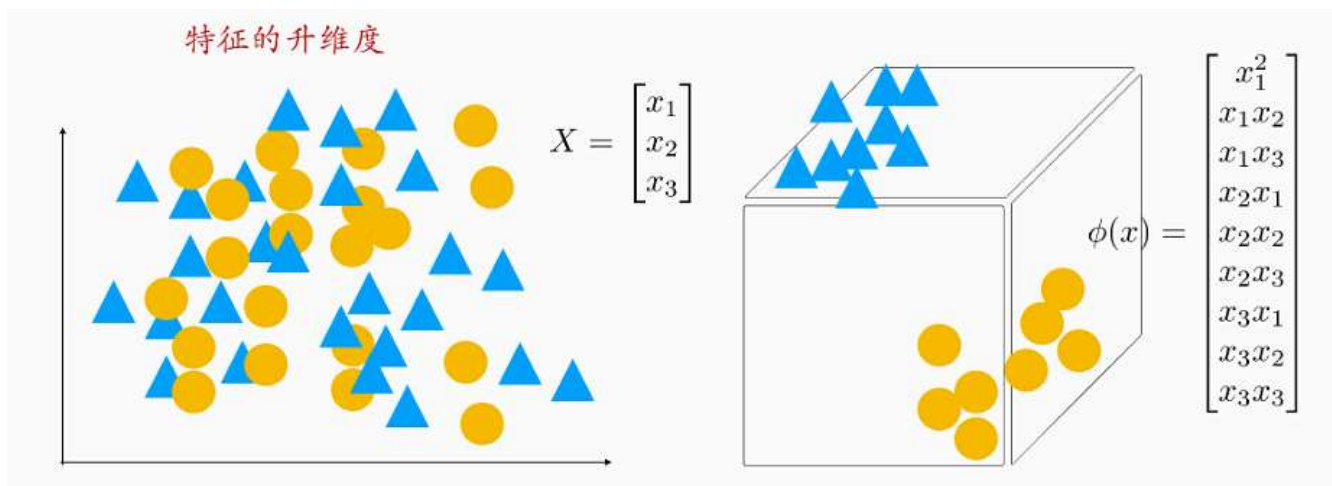
Soft SVM



$$\begin{aligned} \min_{\{w, b\}} \quad & \frac{1}{2} \|w\|^2 + c \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)} (w^T x^{(i)} + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

3. 核函数

核函数的意义在于，当一堆点在低维空间不能做到线性可分的情况下，我们将它映射到更高维让它们变得可分



对于Hard SVM，先将原问题转化为对偶问题，将优化条件转化为 $f_i = 1 - y^{(i)}(w^T x^{(i)} + b) \leq 0$
 同时代入KKT条件的最后一个条件，计算 L 对平面参数的梯度： $\nabla_w L = 0$ 和 $\nabla_b L = 0$

Hard SVM

$$\begin{aligned} \min_{\{w, b\}} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y^{(i)} (w^T x^{(i)} + b) \geq 1, \quad \forall i \end{aligned} \quad \Rightarrow \quad \begin{aligned} \min_{\{w, b\}} \max_{\alpha_i \geq 0} \quad & \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y^{(i)} (w^T x^{(i)} + b)) \\ \max_{\alpha_i \geq 0} \min_{\{w, b\}} \quad & \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y^{(i)} (w^T x^{(i)} + b)) \end{aligned}$$

$$\begin{aligned} \nabla_w \left(\frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y^{(i)} (w^T x^{(i)} + b)) \right) &= w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0 \\ \rightarrow w^* &= \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \\ \nabla_b \left(\frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y^{(i)} (w^T x^{(i)} + b)) \right) &= - \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}$$

随后将第一个式子关于 w^* 的式子代入 L 并化简

Hard SVM

$$\begin{aligned} & \max_{\alpha_i \geq 0} \min_{\{w, b\}} \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (1 - y^{(i)} (w^T x^{(i)} + b)) \\ & \quad \downarrow \\ & \begin{aligned} w^* &= \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \\ - \sum_{i=1}^m \alpha_i y^{(i)} &= 0 \end{aligned} \end{aligned}$$

$$\begin{aligned} & \max_{\alpha_i \geq 0} \frac{1}{2} \left\| \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right\|^2 + \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \alpha_i y^{(i)} (x^{(i)})^T \left(\sum_{j=1}^m \alpha_j y^{(j)} x^{(j)} \right) - \sum_{i=1}^m \alpha_i y^{(i)} b^* \\ & \quad \downarrow \\ & \max_{\alpha_i \geq 0} \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)} + \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)} \\ & \quad \downarrow \\ & \max_{\alpha_i \geq 0} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ & \quad \text{s.t.} \quad \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}$$

得到新的 L 函数: $L = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$

在新的 L 函数下, 我们需要最小化 $y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$

Hard SVM

对新样本的分类

$$\begin{aligned} w^* &= \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \\ & \quad \downarrow \\ & (w^*)^T x^{(new)} + b^* = \left(\sum_{i=1}^m \alpha_i^* y^{(i)} x^{(i)} \right)^T x^{(new)} + b^* \\ & \quad \downarrow \\ & = \sum_{i=1}^m \alpha_i^* y^{(i)} \langle x^{(i)}, x^{(new)} \rangle + b^* \end{aligned}$$

我们可以将点的特征通过一个映射函数映射到更高维度。例如原先样本点的特征个数是三维, 我们可以设计下图的函数将特征的个数从三维映射到九维:

支撑向量机—核函数

对新样本的分类


$$\sum_{i=1}^m \alpha^* y^{(i)} \langle x^{(i)}, x^{(new)} \rangle + b^*$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\phi(x) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

$$\max_{\alpha_i \geq 0} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$$

$$\text{s.t.} \quad \sum_{i=1}^m \alpha_i y^{(i)} = 0$$

$$\max \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$$


模型的迭代

下面介绍模型训练、评价和泛化能力的介绍

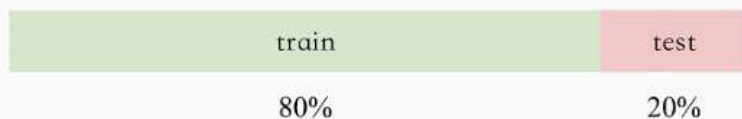
训练集和测试集

对于一批已有的样本，我们一般将它们划分为训练集和测试集两个部分。训练集对模型可见，测试集对模型不可见

下面是一些常见的训练集、测试集划分方法：

- 留出法

留出法 (Hold-out)



将数据集若干次随机划分为无交集的训练集和测试集，重复实验评估后取平均值作为结果。

通常训练集占总样本的80%，测试集占总样本的20%。

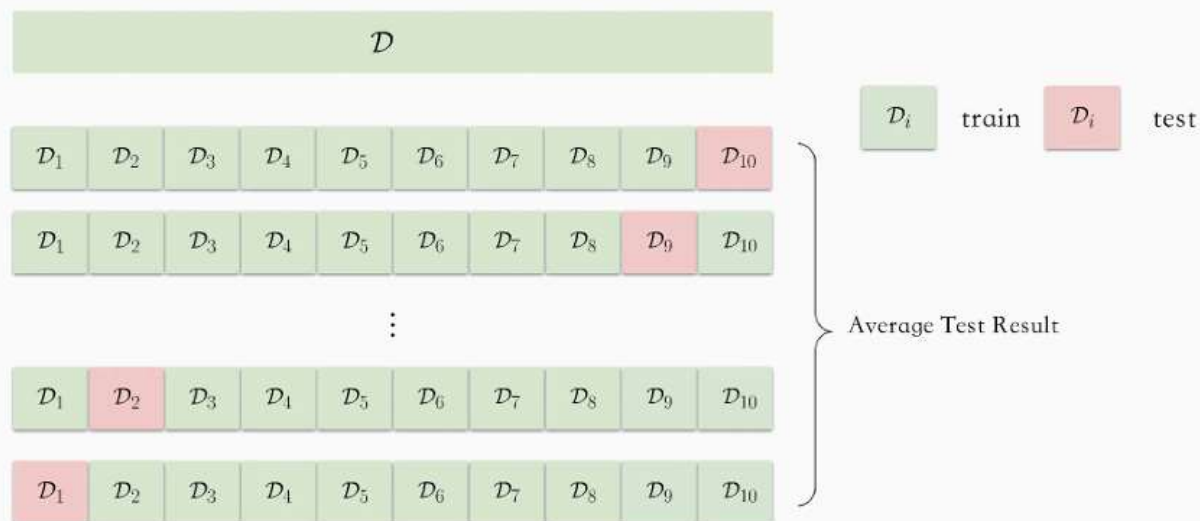


Hold-out 受到数据划分方式的影响，模型容易偏向于训练集中出现次数多的样本。

- 交叉验证法

交叉验证法 (k-fold cross-validation)

将数据集随机等分为k-fold，每次取一个fold作为测试集，输出k个结果的平均值。



- 留一法

留一法 (Leave-One-Out)

k-fold cross validation的特殊形式 (k 等于样本总数)，即每次取出一个样本作为测试样本，剩余样本作为训练集。



LOO不受数据划分方式的影响，与原始数据集分布近似，其评估结果通常认为比较准确，但当数据集较大时，计算开销也随之增大。

- 自助法

自助法 (Bootstrapping)

有放回地重复抽样 m 次



m samples



原数据集 D 中部分样本重复在 D' 中出现，部分样本从未在 D' 中出现。

原数据集 D 中不被 D' 包含的数据作为测试集
样本不被取到的概率为 $\left(1 - \frac{1}{m}\right)^m$

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m \rightarrow \frac{1}{e} \approx 0.368$$



自助法在数据集较小难以划分时比较有效，但改变了原有数据集的分布，会引入估计误差。

- 验证集

验证集是可见样本中的一个子集。在训练集上训练模型后，我们在验证集验证性能。根据性能好坏，调整训练策略、模型结构等人为设定的超参数。如此往复迭代，找到最好的模型。最后一般将训练集和验证集重新拼在一起训练模型，并在测试集评估最终性能。

验证集(validation set)

模型评估与选择中，常常从训练集中拆分一部分数据作为验证集(validation set)。



验证集上进行调参和模型的选择，而测试集是实际应用中的数据，用于最终的性能度量。

模型的评价

为了衡量一个机器学习模型的好坏，需要模型对测试集中的每一个样本进行预测，并根据预测结果计算评价分数，不同学习任务模型性能的评价标准不同

1. 回归模型常用均方误差

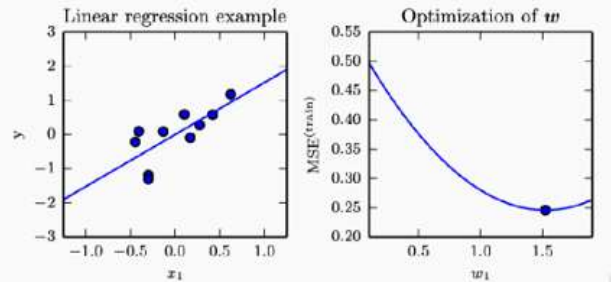
回归模型

比较学习器预测结果 $f(x)$ 与真实标记 y

均方误差(mean squared error, MSE)

$$E(f; D) = \frac{1}{N} \sum_{n=1}^N (f(x_n) - y_n)^2$$

$$E(f; D) = \int_{x \sim D} (f(x) - y)^2 p(x) dx$$



其中 D 是数据分布， $p(x)$ 是概率密度函数。

2. 分类模型常用准确率

分类模型

错误率(error rate)

$$E(f; D) = \frac{1}{m} \sum_{i=1}^M \mathbb{I}(f(x_i) \neq y_i)$$

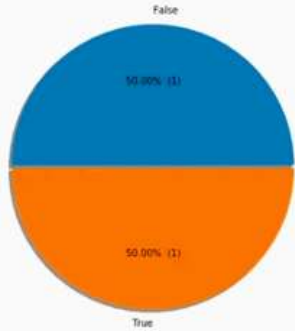
准确率(accuracy)

$$acc(f; D) = \frac{1}{m} \sum_{i=1}^M \mathbb{I}(f(x_i) = y_i) = 1 - E(f; D)$$

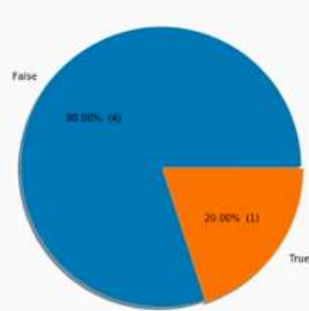
3. 若数据集非常不平衡，使用混淆矩阵

Is your Target Imbalanced? Do you still use Accuracy? Don't!

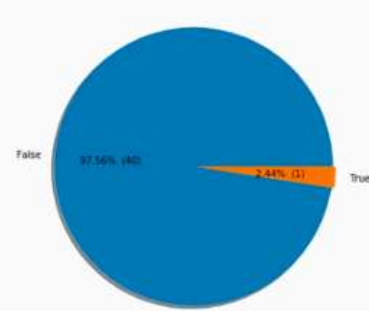
Titanic(Generic Problem)--Balanced Dataset



Actual Business Data--Imbalanced



Fraud Detection/Disease Detection--Highly Imbalanced



其中：Positive表示正类，Negative表示负类

- TP-预测与真实都是正类，预测成功
- TN-预测与真实都是负类，预测成功
- FP-预测正类，真实负类，预测失败
- FN-预测负类，真实正类，预测失败

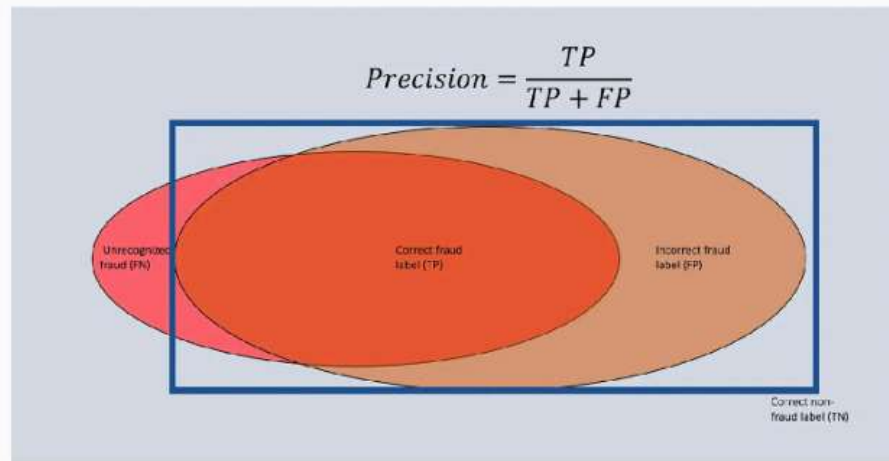
混淆矩阵(confusion matrix)

		Actual class	
		Positive	Negative
Predicted class	Positive	True positive (TP) N=800 0.08%	False positive (FP) N=700 0.07%
	Negative	False negative (FN) N=200 0.02%	True negative (TN) N=1,000,000 99.9%

4. 精确率、召回率、准确率

- 精确率

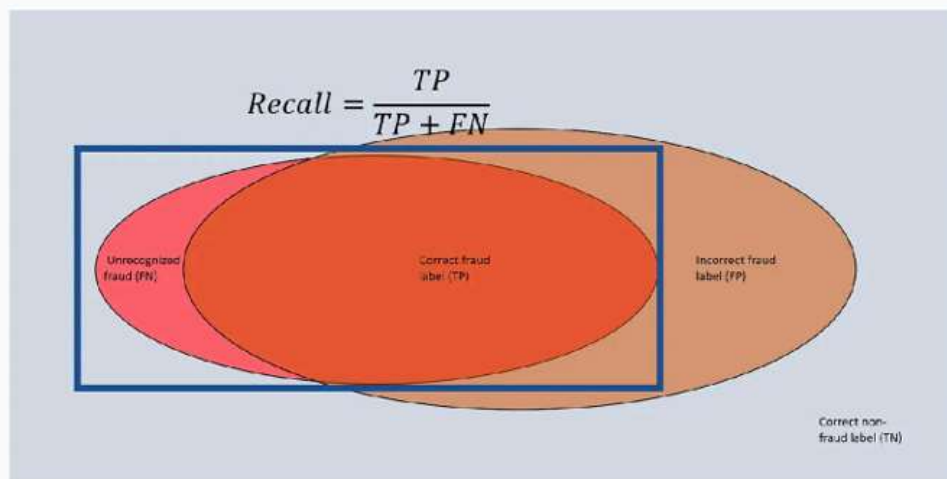
精确率(precision)



精确率(precision)说明了模型所有判别为诈骗的交易记录中，有多少比例的记录确实为诈骗行为。

- 召回率

召回率(recall)



召回率(recall)说明了所有诈骗记录中，被模型侦测出的诈骗记录的占比。

5. PR曲线、平均精度AP

PR曲线的横坐标为召回率R，纵坐标为查准率P，绘制步骤如下：

- 将预测结果按照预测为正类概率值降序排列；
- 从第一个样本开始（模型预测为正类概率最高）。对该样本，我们认为模型把它分为正类，如果它标签确实是正类，那么分类正确；反之错误。
- 对于第k个样本，我们将它以及它之前的样本（编号1-k）均视为模型分成正类。判断这些样本是否分对。对该第k个样本，精度P=前k个模型分对的个数/k，召回R=前k个模型分对的个数/10个样本总共的正样本数量。如此对后续所有10个样本计算精度和召回。
- 以P为纵坐标，R为横坐标绘制点，将所有点连成曲线后构成PR曲线。

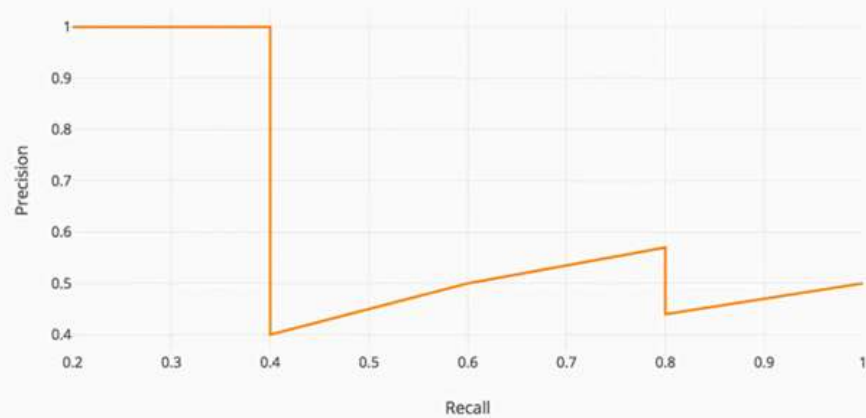
【例子】

编号	模型预测概率 Score	是否为正样本
1	0.91	1 (正)
2	0.81	1
3	0.7	0 (负)
4	0.6	0
5	0.55	0
6	0.43	1
7	0.32	1
8	0.2	0
9	0.1	0
10	0.05	1

Precision-Recall

排序	是否分对	精度	召回
1	是	1	0.2
2	是	1	0.4
3	否	0.67	0.4
4	否	0.5	0.4
5	否	0.4	0.4
6	是	0.5	0.6
7	是	0.57	0.8
8	否	0.5	0.8
9	否	0.55	0.8
10	是	0.5	1.0

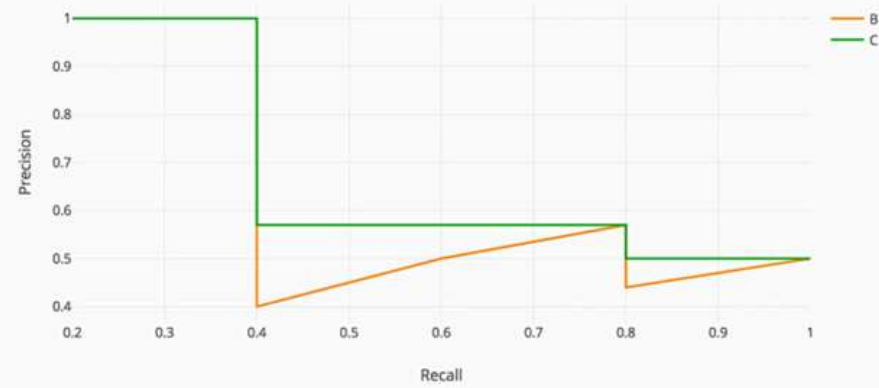
Precision-Recall



下面计算平均精度AP:

- 绘制绿线

Precision-Recall

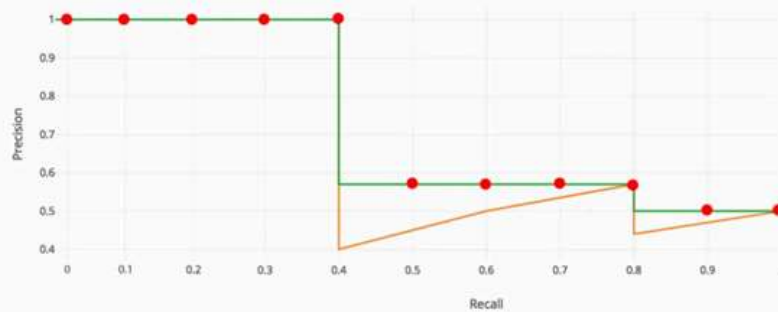


召回率 \hat{r} 的精度用 $\geq \hat{r}$ 召回下最大的精度

$$p_{interp}(r) = \max p(\hat{r}) \text{ for } \hat{r} \geq r$$

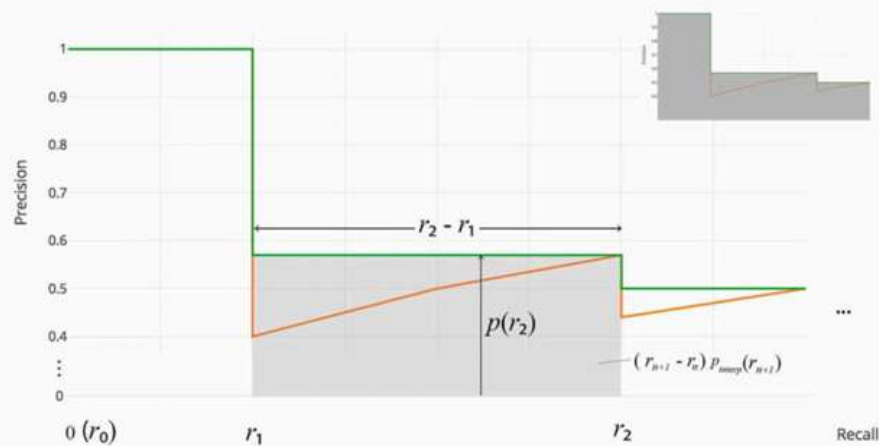
- 计算

Precision-Recall



$$AP_{11} = 1/11 (Ap(0) + Ap(0.1) + \dots + Ap(1.0)) \quad AP = (5 \times 1.0 + 4 \times 0.57 + 2 \times 0.5)/11$$

Precision-Recall

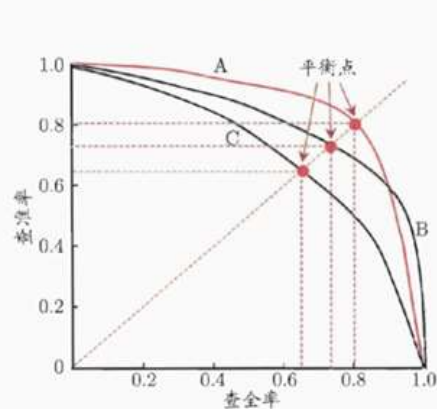


$$AP = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max p(\hat{r}) \text{ for } \hat{r} \geq r_{n+1}$$

平衡点：一般平衡点越靠右上，模型性能越优秀

平衡点(Break-Even Point, BEP)



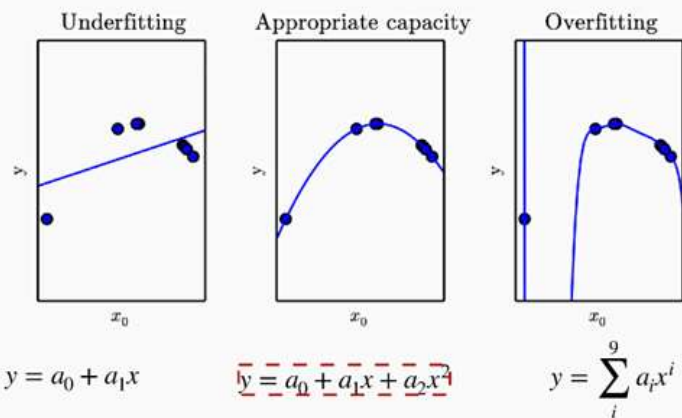
precision \longleftrightarrow recall

平衡点(BEP)是 $P=R$ 时, PR图的取值。

泛化能力

奥卡姆剃刀

在同样能够解释已知观测现象的假设中, 我们应该挑选“最简单”的那一个



1. 泛化能力(Generalizability): 指由该方法学习到的模型对未知数据的预测能力

2. 误差:

- 经验误差: 训练集上的误差 $\frac{1}{m^{(train)}} \|X^{(train)}w - y^{(train)}\|$

- 泛化误差: 测试集上的误差 $\frac{1}{m^{(test)}} \|X^{(test)}w - y^{(train)}\|$

泛化误差是泛化能力的一种度量, 是所学习到的模型的期望风险

- 泛化误差上界

泛化误差是指由训练集泛化至训练集外的过程中产生的误差。上界指它的最大值。d表示假设空间:

泛化误差上界

以二分类任务为例：

训练集 $\{(x_1, y_1), \dots, (x_n, y_n)\}, \quad x \in R^D, \quad y \in \{+1, -1\}$

模型假设空间 $\mathbf{f} = \{f_1, \dots, f_d\}$

训练误差 $\hat{R}(f) = \frac{1}{N} \sum_{n=1}^N I(y_n \neq f(x_n))$

测试误差的期望 $R(f) = E(x, y)[I(y \neq f(x))]$

泛化误差上界

Vapnik-Chervonenkis 维度

对任意一个函数，至少以概率 $1 - \delta$ ，以下不等式成立

$$R(f) = \hat{R}(f) + \sqrt{\frac{1}{2N} \left(\log d + \log \frac{1}{\delta} \right)}$$

误差上界

- 训练误差小的模型，泛化误差上界也会越小
- 样本数量增大，泛化误差上界减小
- 假设空间越大，泛化误差上界减小

3. 欠拟合、过拟合、针对的训练策略

- 欠拟合：经验误差和泛化误差都非常高
 - 改用表达能力更强的模型
 - 增加训练迭代次数
- 过拟合：经验误差降低，泛化误差反升
 - 增大训练数据
 - 参数正则化

正则化

正则化 (Regularization) 指对降低泛化误差而非训练误差的修改

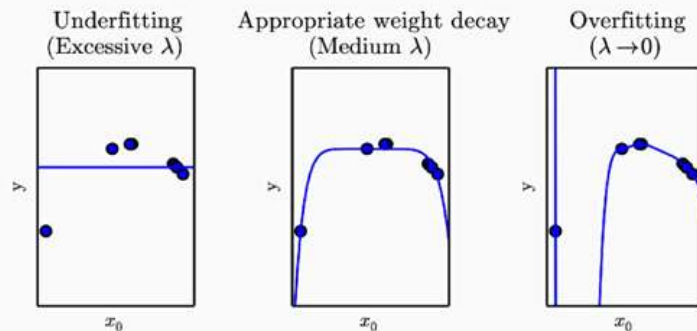
权重衰减 (Weight Decay)

以训练九次多项式为例

$$y = \sum_i^9 a_i x^i$$

$$w = [a_0, a_1, \dots, a_9]$$

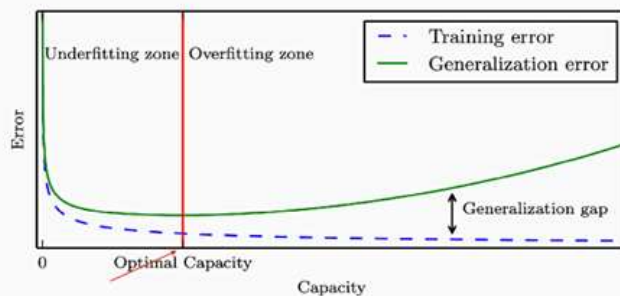
$$J(w) = MSE_{train} + \lambda w^T w$$



- 早停止策略

早停止策略

- 将数据划分为训练集和验证集，验证集用来估计误差
- 若训练集误差降低，验证集误差升高，则停止训练



停止

4. 偏差与方差

- 偏差(bias):期望输出 $\bar{f}(x)$ 与真实标记 y 的偏离程度

$$bias^2(x) = (\bar{f}(x) - y)^2$$

- 方差(Variance): x 在训练集 D 上学得模型 $f(x; D)$ 上的输出与期望输出 $\bar{f}(x)$ 之间的变动导致的学习性能的变化

$$var(x) = E_D[(f(x; D) - \bar{f}(x))^2]$$

噪声: $\epsilon^2 = E_D[(y_D - y)^2]$

计算测试误差的期望:

假定噪声期望 $E_D[y - y_D]$ 为 0

$$\begin{aligned} E(f; D) &= E_D[(f(x; D) - y_D)^2] \\ &= E_D[(f(x; D) - \bar{f}(x) + \bar{f}(x) - y_D)^2] \\ &= E_D[(f(x; D) - \bar{f}(x))^2] + E_D[(\bar{f}(x) - y_D)^2] + E_D[2(f(x; D) - \bar{f}(x))(\bar{f}(x) - y_D)] \\ &= E_D[(f(x; D) - \bar{f}(x))^2] + E_D[(\bar{f}(x) - y_D)^2] \\ &= var(x) + E_D[(\bar{f}(x) - y + y - y_D)^2] \\ &= var(x) + E_D[(\bar{f}(x) - y)^2] + E_D[(y - y_D)^2] + E_D[2(\bar{f}(x) - y)(y - y_D)] \\ &= var(x) + E_D[(\bar{f}(x) - y)^2] + E_D[(y - y_D)^2] \\ &= var(x) + bias^2(x) + \epsilon^2 \end{aligned}$$

$$E(f; D) = var(x) + bias^2(x) + \epsilon^2$$

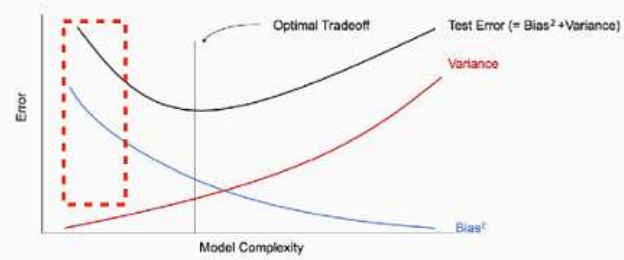
$var(x)$ 训练集的变动 (数据扰动) 导致的影响

$bias^2(x)$ 算法本身的拟合能力

ϵ^2 问题本身的难度

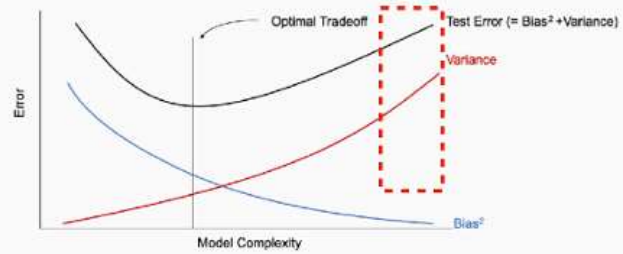
- 偏差、方差与拟合

high bias
underfit



complexity \longleftrightarrow fitting

high variance
overfit



complexity \longleftrightarrow fitting