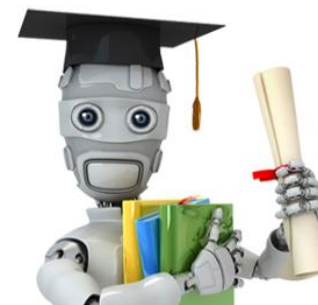


神经网络



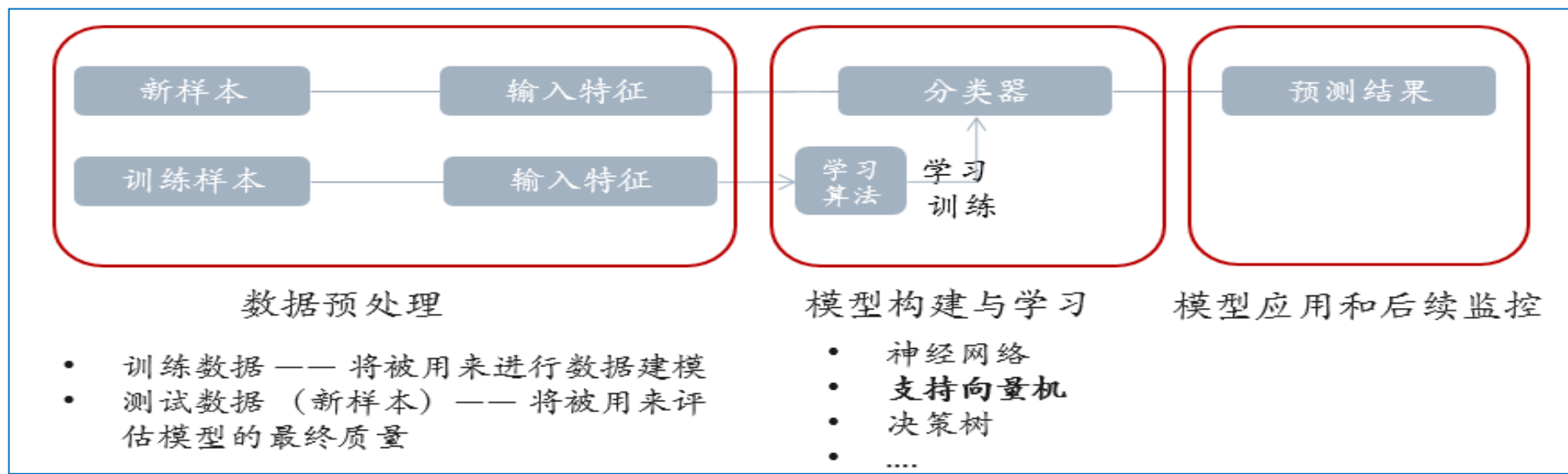
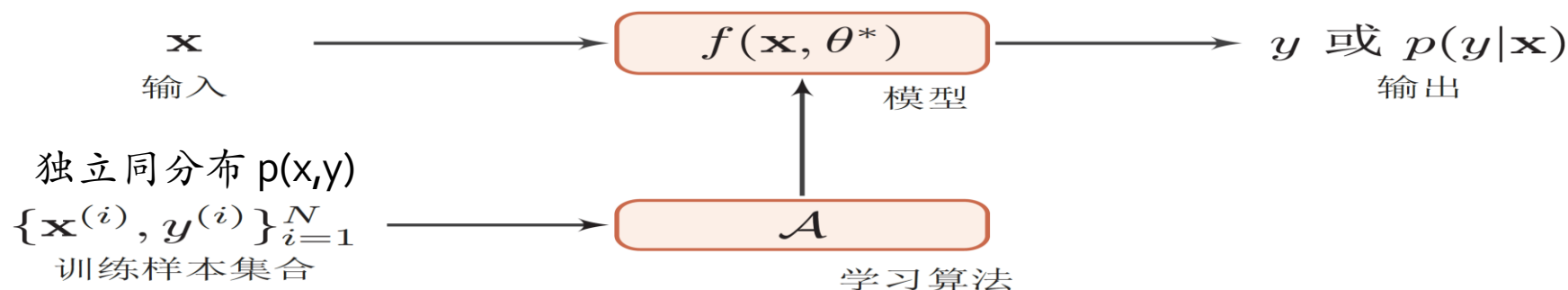
曾碧

zb9215@gdut.edu.cn

第2章 机器学习概述

什么是机器学习?

机器学习：从数据中获得决策（预测）函数使得机器可以根据数据进行自动学习，通过算法使得机器能**从大量历史数据中学习规律从而对新的样本做决策**。



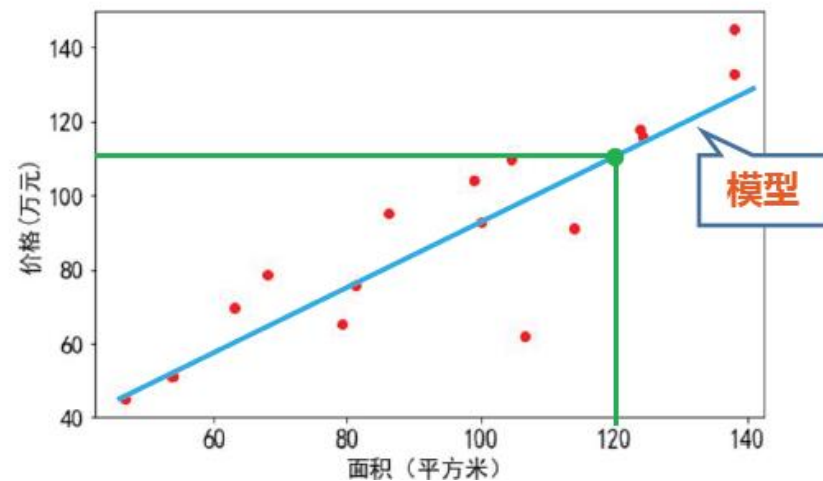
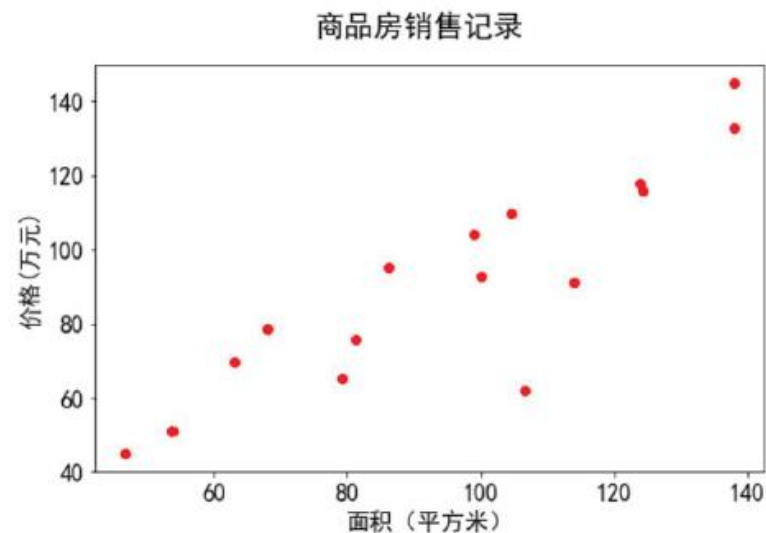
什么是机器学习?

□ 例：从房屋销售数据中学习

- 建立模型
- 学习模型
- 预测房价

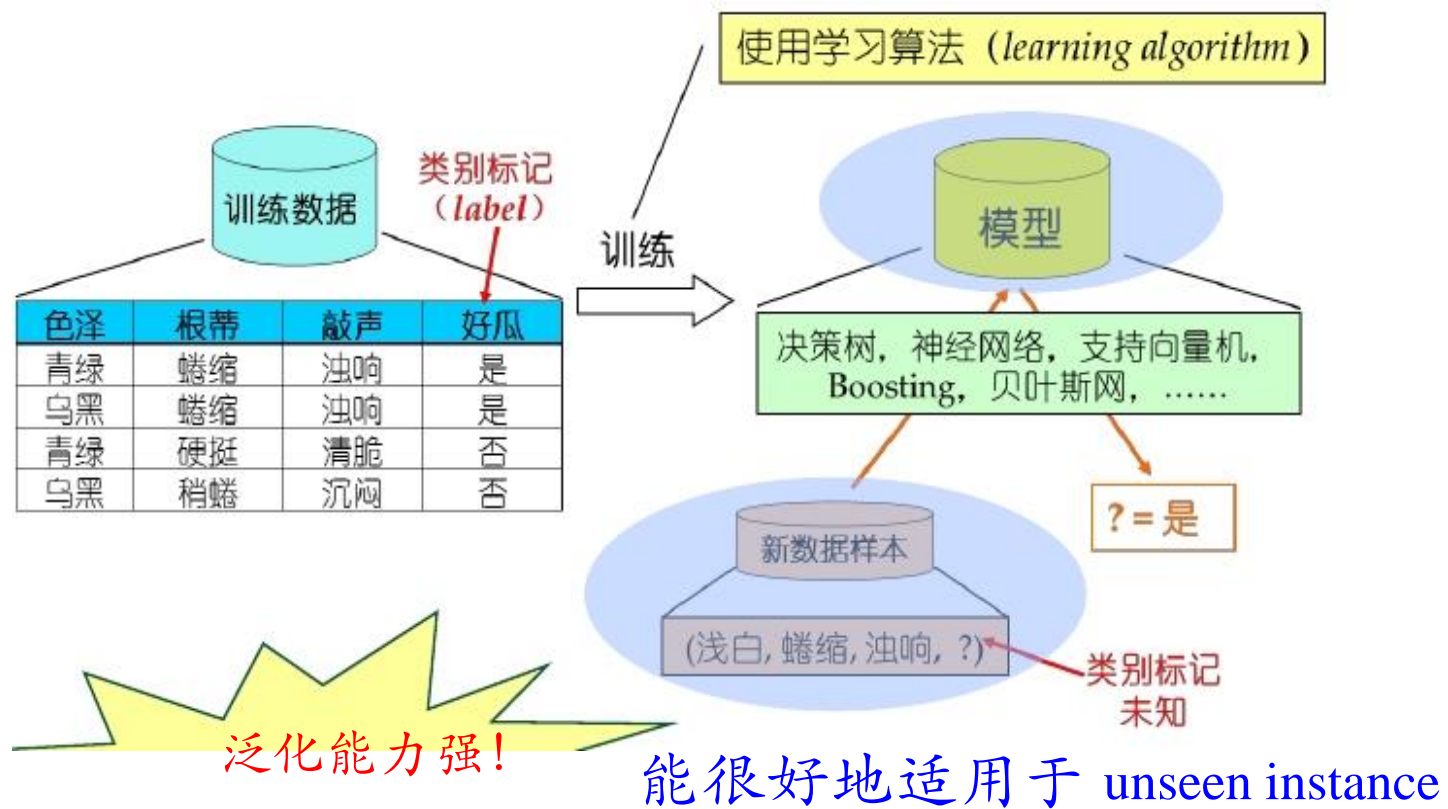
例：房屋销售记录

序号	面积 (平方米)	销售价格 (万元)	序号	面积 (平方米)	销售价格 (万元)
1	137.97	145.00	9	106.69	62.00
2	104.50	110.00	10	138.05	133.00
3	100.00	93.00	11	53.75	51.00
4	124.32	116.00	12	46.91	45.00
5	79.20	65.32	13	68.00	78.50
6	99.00	104.00	14	63.02	69.65
7	124.00	118.00	15	81.26	75.69
8	114.00	91.00	16	86.21	95.30



学习算法：从数据中产生模型的算法

典型的机器学习过程



模型的评估

□ 泛化误差v.s.经验误差

■ **泛化误差**：在“未来”样本上的误差，用测试集上的“测试误差”作为近似

■ **经验误差**：在训练集上的误差，亦称“训练误差”

➡ 泛化误差越小越好，表示模型预测越准确；

经验误差是否越小越好？ **NO!会产生过拟合问题**

□ **过拟合问题**：在训练集上表现良好，在测试集上表现糟糕

■ **过拟合(overfitting)**：同时拟合训练样本的**共性特征**和**个性特征**

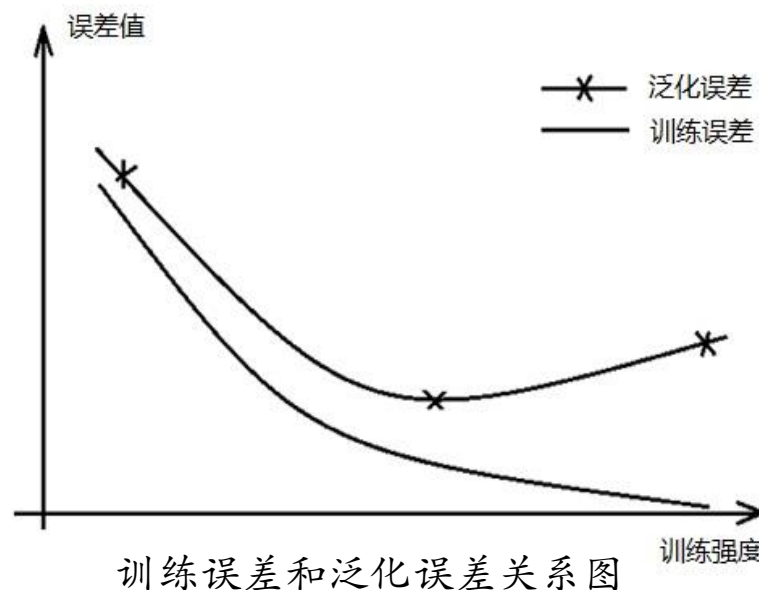
■ **欠拟合 (underfitting)**：未能充分拟合训练样本共性特征造成模型泛化误差较大而导致模型泛化能力较弱

过拟合成因

- 训练集和测试集特征分布不一
- 数据噪声太大
- 数据量太小
- 特征量太多
- 模型太过复杂

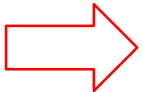
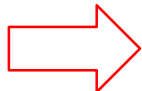
解决方法

- 控制模型的复杂度
- 有效样本使用：交叉验证训练
- 有效特征约简：特征变换、选择、Dropout



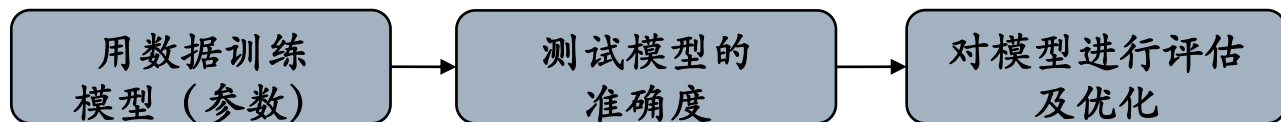
模型选择 (model selection)

两个关键问题:

- 如何获得测试结果?  评估方法
- 如何评估性能优劣?  性能度量

评估方法

□ 训练与测试



- 训练集：用来训练模型，估计参数
- 测试集：用来测试和评估训练出来的模型的预测表现能力

□ 关键：应如何获得测试集？

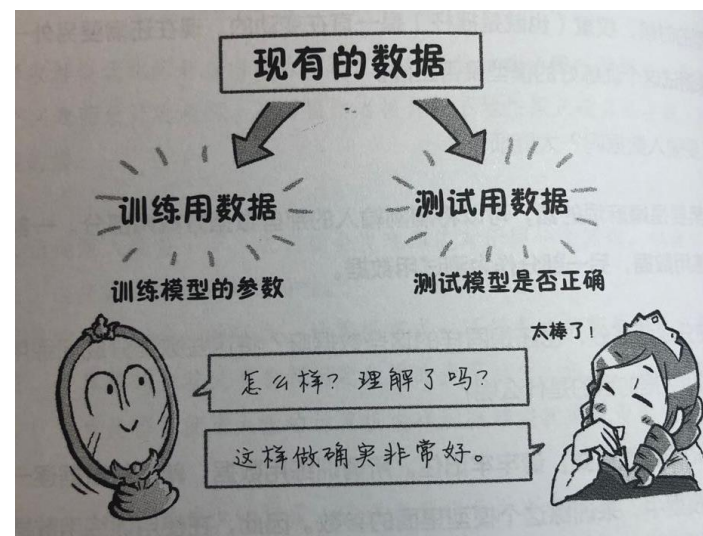
- 假设测试样本也是从样本真实分布中独立同分布采样所得；
- 测试集应该与训练集“互斥”，即测试样本尽量不在训练集中出现、未在训练过程中使用。

对包含 n 个样例的数据集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ ，要如何处理产生满足条件的训练集 S 与测试集 T ？

留出法

交叉验证法

自助法



评估方法-测试集形成方法

➤ 留出法

基本思路：直接从样本数据集 D 中随机划分出部分数据组成训练样本集 S ，剩下部分作为测试样本集 T 用于估计模型的泛化误差。

- 保持数据分布一致性（例如：分层采样）
- 多次重复划分（例如：100次随机划分）
- 测试集不能太大、不能太小（例如：1/5~1/3），一般将大约2/3~4/5的样本用于训练，剩下测试

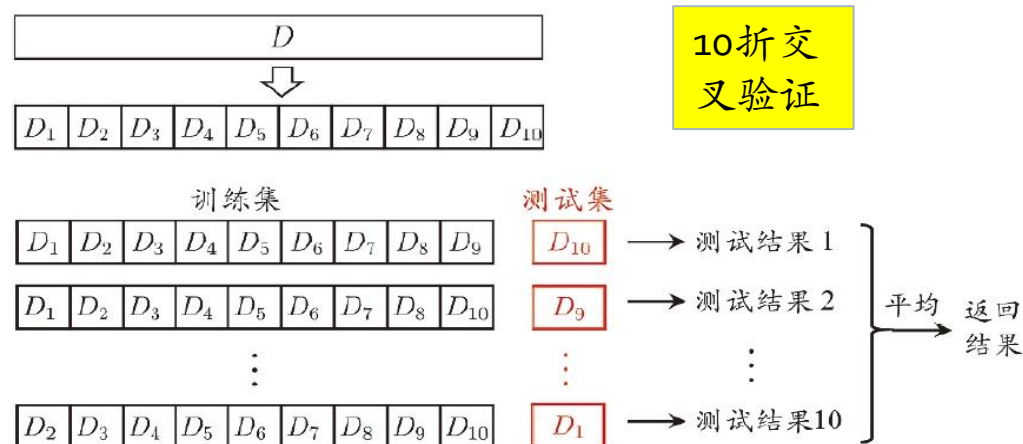
拥有的数据集



➤ k-折交叉验证法

方法：将数据集 D 等分为 K 子集 $D_i (i = 1, 2, \dots, K)$ ，然后依次保留其中一个子集作为测试集 T ，而将其余 $K - 1$ 个子集合进行合并后作为训练集 S 。

特例：若 $k = m$ (D 中包含的样本数)，则得到“留一法”
(leave-one-out, LOO)



评估方法-测试集形成方法

➤ 自助法

适用条件：当 D 中样本数量较少、难以有效划分训练/测试集时；需要产生多个不同训练集的集成学习。

基本思路：通过对 D 中样本进行可重复随机采样的方式构造训练集和测试集。

方法：假设数据集 D 中包含 n 个样本，自助法对数据集 D 中样本进行 n 次有放回的采样，并将采样得到的样本作为训练样本生成一个含有 n 个样本的训练样本集 S ，所有未被抽到的样本则作为测试样本构成测试集 T 。

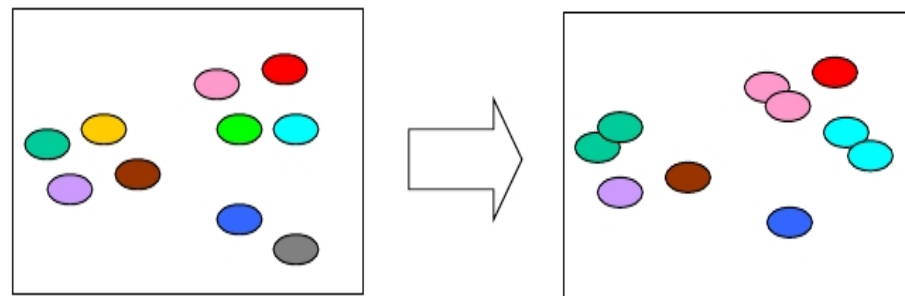
特点：

- 训练集与原样本集同规模
- 数据分布有所改变

对于 D 中的任一样本，该样本在自助采样中未被采样的概率为：

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

约有 36.8% 的样本没被采样到



使用未在训练集中出现的样本用于测试（约为数据总量1/3），这样的测试结果称为“包外估计”（out-of-bag estimation）

评估方法- “调参” 与最终模型

算法的参数：一般由人工设定，亦称“超参数”

模型的参数：一般由学习确定

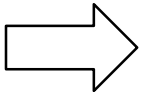
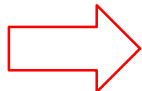
调参过程相似：先产生若干模型，然后基于某种评估方法进行选择

参数调得好不好往往对最终性能有关键影响

区别：训练集 vs. 测试集 vs. 验证集 (validation set)

算法参数选定后，要用“训练集+验证集”重新训练最终模型

两个关键问题:

- 如何获得测试结果?  评估方法
- 如何评估性能优劣?  性能度量

性能度量(performance measure)是衡量模型泛化能力的评价标准，使用不同的性能度量往往会导致不同的评判结果，主要取决于以下两方面：

- 算法与数据
- 任务需求（回归（预测）任务、分类任务）

回归任务：均方误差 $E(f; D) = \frac{1}{m} \sum_{i=1}^m (f(x_i) - y_i)^2$

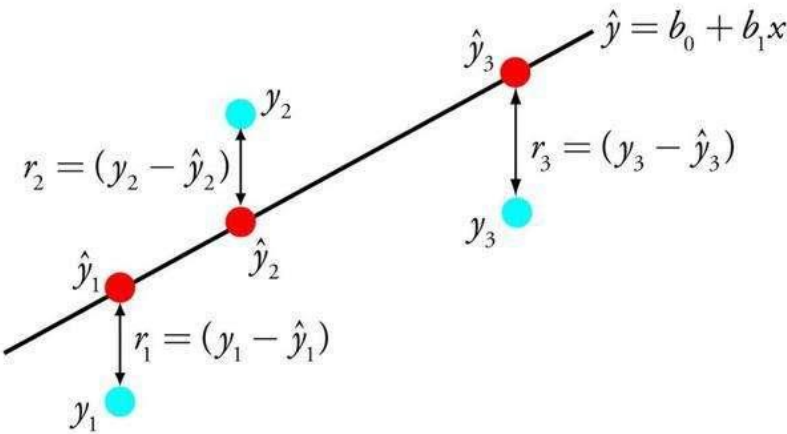
分类任务：错误率与精度、查准率、查全率与F1

a. 错误率：分类错误的样本数占样本总数的比例

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

b. 准确度：分类正确的样本数占样本总数的比例

$$\begin{aligned} \text{acc}(f; D) &= \frac{1}{m} \sum_{i=1}^m \mathbb{I}(f(x_i) = y_i) \\ &= 1 - E(f; D) = \frac{TP + TN}{ALL} \end{aligned}$$



真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

如果我们关心的是被预测为正样本中有多少比例是真的正样本，或者所有的正样本中有多少比例被预测为正样本，这个时候错误率就不够用了，还需要使用其他的性能度量。查准率与查全率就是更适用于此类需求的性能度量。

c. 查准率：所有预测为正的样例中，真的正例所占比例
(精确率)

$$P = \frac{TP}{TP + FP}$$

d. 查全率：所有真实为正的样例中，预测也真的正例所占比例
(召回率)

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

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)








分类结果的“混淆矩阵”

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

■ 使用PR图及BEP对学习器（模型）进行比较

算法对样本进行分类时，都会有置信度，即表示该样本是正样本的概率。通过置信度就可以对所有样本进行排序，再逐个样本的选择阈值，在该样本之前的都属于正例，该样本之后的都属于负例。每一个样本作为划分阈值时，都可以计算对应的查准率(P)和查全率(R)，那么就可以绘制P-R曲线。

精确度和召回率

数据	标签	预测
	0	0
	0	1
	1	0
	1	1
	0	1
	1	1
	0	0








$$\text{精确度} = \frac{\text{\#正确地预测为“正面”}}{\text{\#预测为“正面”}}$$

$$\text{召回率} = \frac{\text{\#正确地预测为“正面”}}{\text{\#测试集中标签为“正面”}}$$

$$\text{精确度} = \frac{2}{4}$$

$$\text{召回率} = \frac{2}{3}$$

最大化精确度








数据	标签	预测
	0	0
	0	0
	1	0
	1	0
	0	0
	1	0
	0	0

$$\text{精确度} = \frac{\text{\#正确地预测为“正面”}}{\text{\#预测为“正面”}}$$

$$\text{召回率} = \frac{\text{\#正确地预测为“正面”}}{\text{\#测试集中标签为“正面”}}$$

将每个都预测为负！

最大化召回率

数据	标签	预测
	0	1
	0	1
	1	1
	1	1
	0	1
	1	1
	0	1








$$\text{精确度} = \frac{\text{\#正确地预测为“正面”}}{\text{\#预测为“正面”}}$$

$$\text{召回率} = \frac{\text{\#正确地预测为“正面”}}{\text{\#测试集中标签为“正面”}}$$

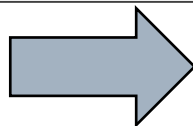
将每个都预测为正！

(所有都预测为正，必能正确预测原标签为正的例子，分子与分母相等，召回率=1)








置信度

数据	标签	预测	置信度
	0	0	0.75
	0	1	0.60
	1	0	0.20
	1	1	0.55
	0	1	0.50
	1	1	0.80
	0	0	0.90

按预测正负
类，对置信
度进行排序



正类：从高到低
负类：从低到高

数据	标签	预测	置信度
	1	1	0.80
	0	1	0.60
	1	1	0.55
	0	1	0.5
	1	0	0.20
	0	0	0.75
	0	0	0.90

按置信度

数据	标签	预测	置信度	精确度	召回率	
<div></div>	1	1	0.80	1/2=0.5	1/3=0.33	判为正例
<div></div>	0	1	0.60			
<div></div>	1	1	0.55			判为负例
<div></div>	0	1	0.5			
<div></div>	1	0	0.20			
<div></div>	0	0	0.75			
<div></div>	0	0	0.90			
正确预测为正						<div></div>

阈值
(低于该置信度的全部判为负例)

按置信度

数据	标签	预测	置信度	精确度	召回率
<div></div>	1	1	0.80	3/5=0.6	3/3=1
<div></div>	0	1	0.60		
<div></div>	1	1	0.55		
<div></div>	0	1	0.5		
<div></div>	1	0	0.20		
<div></div>	0	0	0.75		
<div></div>	0	0	0.90		








判为正例

判为负例

阈值
(高于该置信度的全部判为正例)

正确预测为正

按置信度

数据	标签	预测	置信度	精确度	召回率
	1	1	0.80	1.0	0.33
	0	1	0.60	0.5	0.33
	1	1	0.55	0.66	0.66
	0	1	0.5	0.5	0.66
	1	0	0.20	0.6	1.0
	0	0	0.75	0.5	1.0
	0	0	0.90	0.43	1.0

性能度量

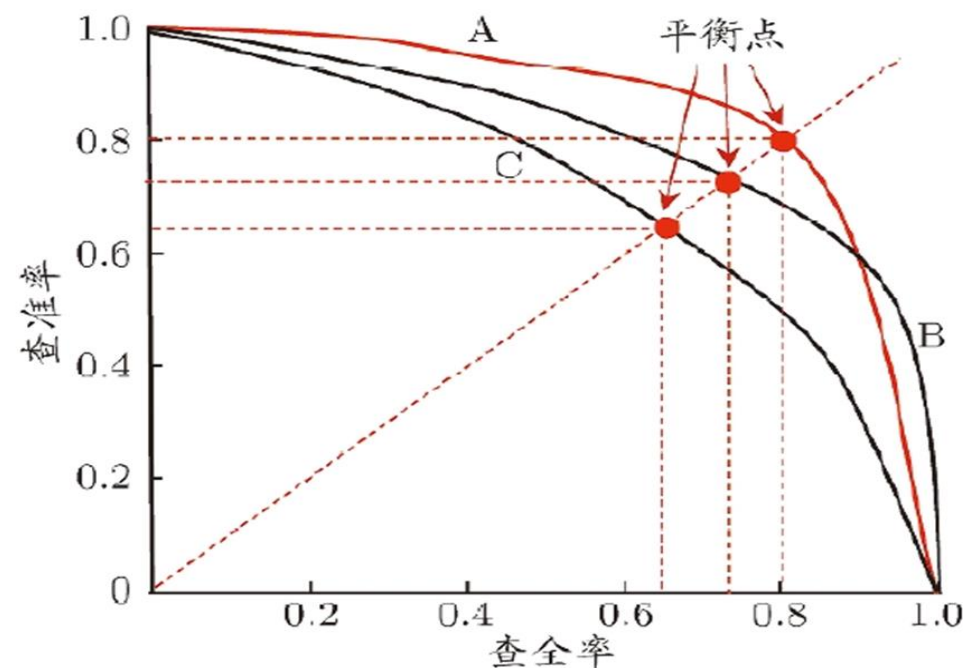
■ P-R图解读

- 若一个学习器的P-R曲线被另一个学习器的曲线完全“包住”，则可以断言后者的性能优于前者。

➡ 学习器 A 优于 学习器 C，学习器 B 优于 学习器 C

- 若两个学习器的P-R曲线有交叉，就难以一般性地断言两者哪个更优(A v.s. B)。一个比较合理的判据是比较P-R曲线下面积的大小(AUC, Area Under Curve)，但是这个值不容易估算。设计了平衡点以帮助判断性能，平衡点值较大的学习器性能较优。
- 平衡点(BEP)**：在P-R曲线中“查准率=查全率”时的取值。

➡ 学习器B的BEP是0.74，学习器A的BEP是0.8，则可认为学习器A的性能优于学习器B。



P-R曲线

性能度量

ROC (Receiver Operating Characteristic) 曲线与AUC

ROC曲线：从阈值选取角度出发来研究学习器泛化性能。当阈值从0变到最大，逐个把样本作为正例进行预测，计算两个重要的量，分别以它们为横、纵坐标作图。

阈值的选取与分类的关系：将学习器为测试样本产生的预测值与一个分类阈值进行比较，若大于阈值则分为正类，否则为反类。例如，若我们更重视“查准率”，则可以把阈值设置的大一些，让分类器的预测结果更有把握；若我们更重视“查全率”，则可以把阈值设置的小一些，让分类器预测出更多的正例。

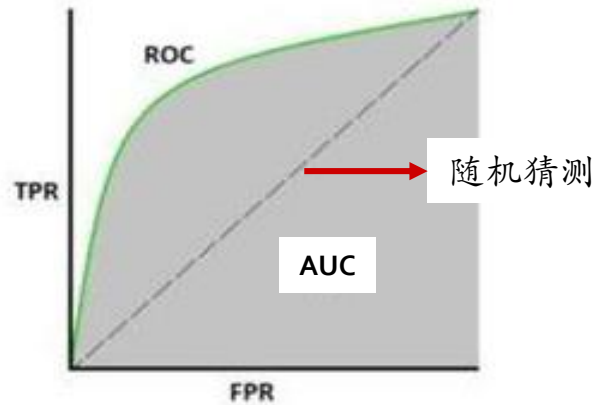
ROC曲线与P-R曲线的区别：P-R曲线的纵、横轴分别为查准率与查全率，ROC曲线的纵、横轴为真正例率与假正例率

(纵)真正例率: $TPR = \frac{TP}{TP + FN}$
(Ture positive rate)

(横)假正例率: $FPR = \frac{FP}{TN + FP}$
(False positive rate)

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

- **有助于选择最佳的阈值。**ROC曲线越靠近左上角，模型的查全率就越高。最靠近左上角的ROC曲线上的点是分类错误最少的最好阈值，其假正例和假反例总数最少。
- **可以对不同的学习器比较性能。**与P-R图相似，若一个学习器的ROC曲线被另一个学习器曲线完全“包住”，则后者优于前者；若有交叉，则计算ROC下区域面积(AUC)，AUC值越大越优。其中：AUC(Area Under ROC Curve)



性能度量

□ F测度（加权调和平均值）：评估精确度/召回率权衡的综合指标。

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

■ 当 $\alpha = 0.5$ ，F测度称为F1测度

$$F1 = \frac{1}{0.5 \frac{1}{P} + 0.5 \frac{1}{R}} = \frac{2PR}{P + R}$$

F1测度：比BEP更常用的度量

■ 若对查准率/查全率有不同偏好：

$$F_{\beta} = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

$\beta > 1$ 时查全率有更大影响； $\beta < 1$ 时查准率有更大影响

“误差”包含了哪些因素？

换言之，从机器学习的角度看，
“误差”从何而来？

- 对回归任务，模型的泛化误差可通过“**偏差-方差分解**”拆解为：

$$E(f; D) = E_D[(f(x; D) - y_D)^2]$$

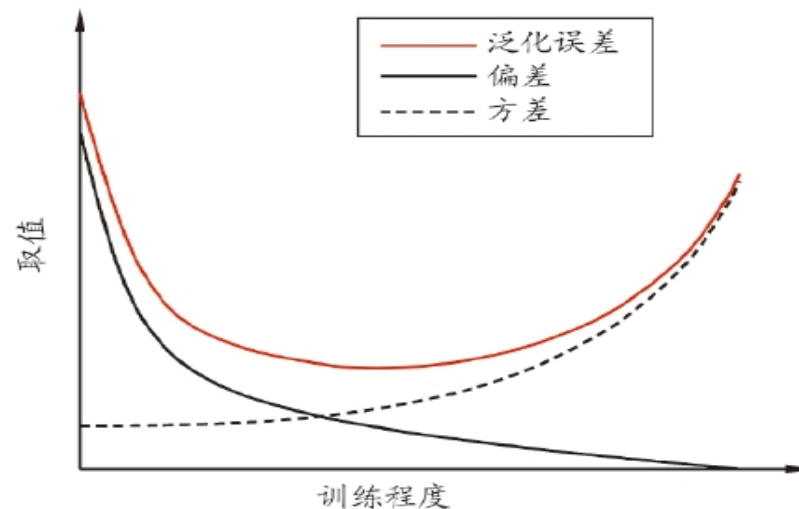
$$= \text{bias}^2(x) + \text{var}(x) + \varepsilon^2$$

偏差

方差

噪声

- 泛化误差(Error)：反映的是整个模型的准确度
- 偏差(Bias)：反映的是模型在样本上的输出与真实值之间的误差，即模型本身的精准度
- 方差(Variance)：反映的是模型每一次输出结果与模型输出期望之间的误差，即模型的稳定性。



泛化误差与偏差、方差间关系

一般而言，偏差与方差存在冲突：

- 训练不足时，学习器拟合能力不强，偏差主导
- 随着训练程度加深，学习器拟合能力逐渐增强，方差逐渐主导
- 训练充足后，学习器的拟合能力很强，方差主导

偏差-方差分解 (bias-variance decomposition)

对回归任务，泛化误差可通过“偏差-方差分解”拆解为：

$$E(f; D) = \underbrace{bias^2(\mathbf{x})}_{\text{red}} + \underbrace{var(\mathbf{x})}_{\text{blue}} + \underbrace{\varepsilon^2}_{\text{green}}$$

期望输出与真实
输出的差别

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

同样大小的训练集
的变动，所导致的
性能变化

$$var(\mathbf{x}) = \mathbb{E}_D \left[(f(\mathbf{x}; D) - \bar{f}(\mathbf{x}))^2 \right]$$

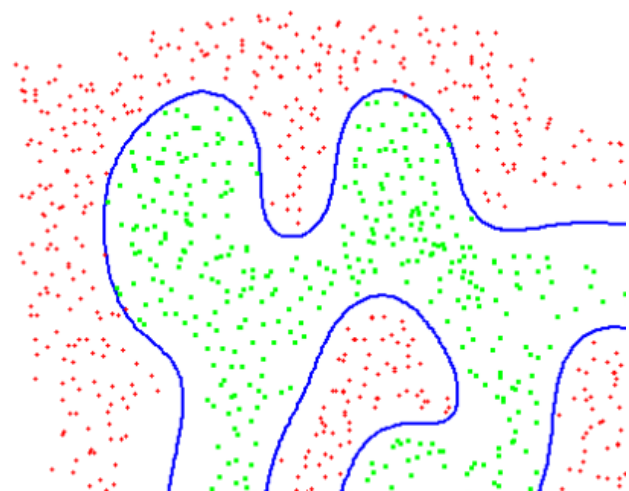
表达了当前任务上任何学习算法
所能达到的期望泛化误差下界

$$\varepsilon^2 = \mathbb{E}_D \left[(y_D - y)^2 \right]$$

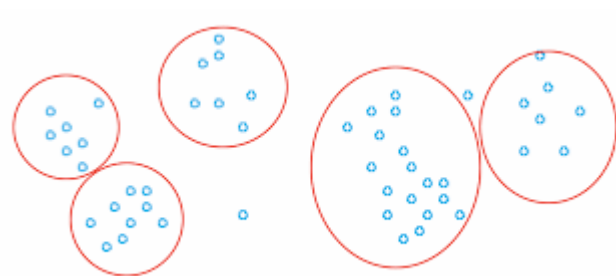
真实标记有区别

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

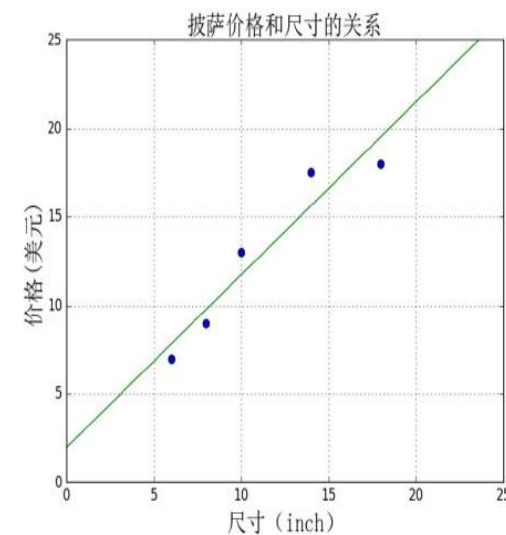
常见的机器学习问题



分类



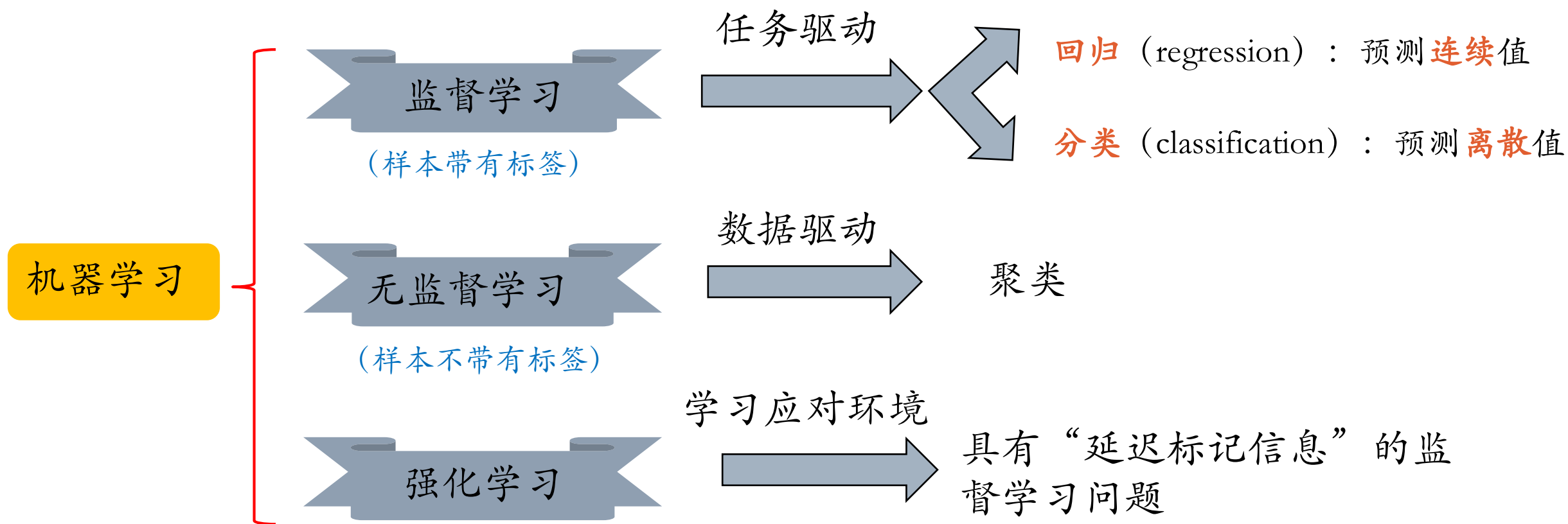
聚类



回归

机器学习分类

□ 机器学习分类（依据先验知识的不同形式）



神经元线性模型

□ 神经元模型

神经元输入向量 $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]^T$,
经过函数映射: $f_{\theta}: \mathbf{x} \rightarrow y$ 后得到输出 y , 其中
 θ 为函数 f 自身的参数。

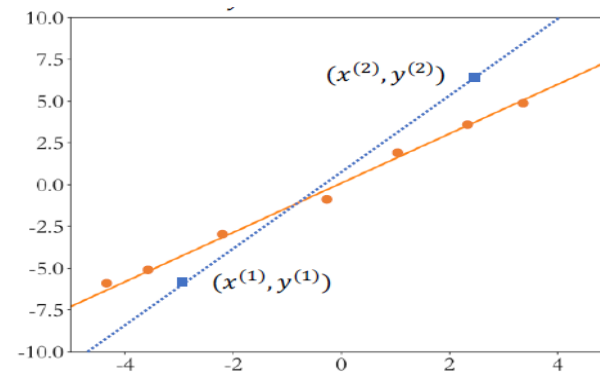
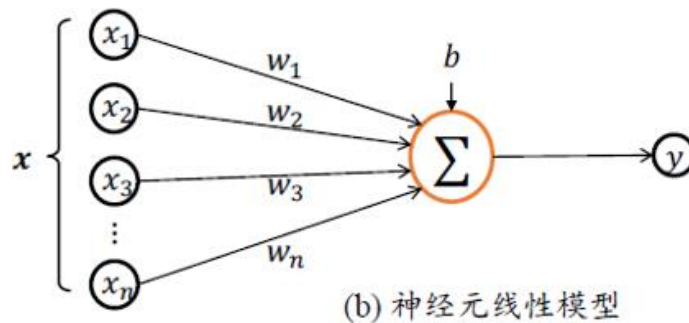
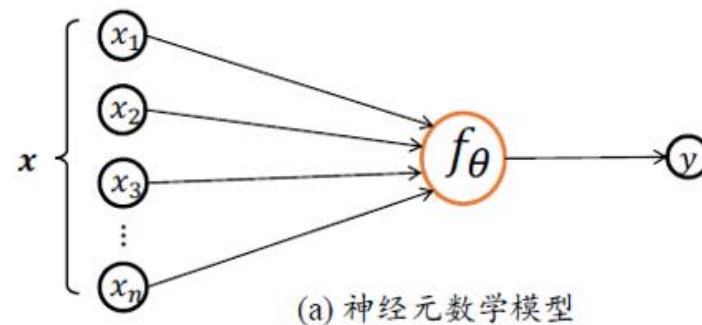
□ 当 $f(\mathbf{x})$ 取线性模型时, 有: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$,
展开为标量形式:

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots + w_n x_n + b$$

参数 $\theta = \{w_1, w_2, w_3, \dots, w_n, b\}$ 确定了神经元的状态,
其中 $b = -\text{threshold}$ (神经元的阈值), **通常把 b 称为偏置**。
当神经元输入节点数 $n = 1$ 时, 有直线方程:

$$y = wx + b$$

只需要观测两个不同数据点, 就可
求解单输入线性神经元模型的参数。



神经元线性模型的评价

最佳拟合直线应该使得所有点的残差累计值最小

□ 残差和最小

$$Loss = \sum_{i=1}^n (y_i - \hat{y}_i) = \frac{1}{2} \sum_{i=1}^n (y_i - (wx_i + b))$$

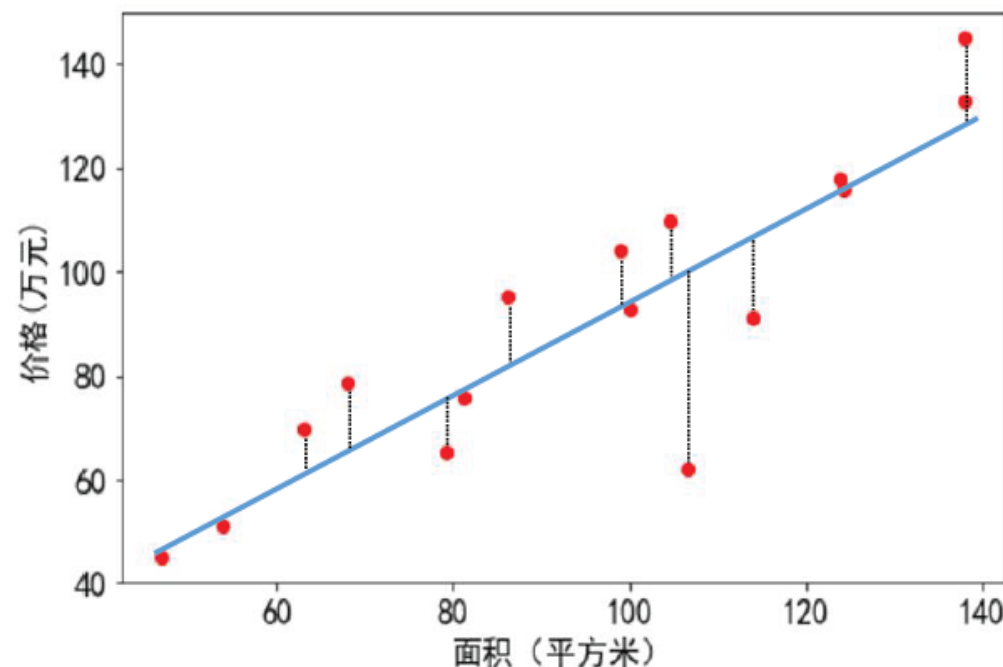
□ 残差绝对值和最小

$$Loss = \sum_{i=1}^n |y_i - \hat{y}_i| = \frac{1}{2} \sum_{i=1}^n |y_i - (wx_i + b)|$$

□ 残差平方和最小

$$Loss = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2} \sum_{i=1}^n (y_i - (wx_i + b))^2$$

商品房销售记录



损失函数 \mathcal{L}

□ 损失函数/代价函数 (cost function):

估量模型的**预测值**与**真实值**的不一致程度

□ 常见损失函数类型

● 损失函数取决于任务的类型:

○ **回归**: 网络预测连续的数值变量

■ 平均绝对误差, 均方误差

○ **分类**: 网络预测分类

■ Softmax、交叉熵损失

□ **loss** function

=**cost** function

=**objective** function

=**error** function

例1: 0-1损失函数

$$\mathcal{L}(y, f(x, \theta)) = \begin{cases} 0 & \text{if } y = f(x, \theta) \\ 1 & \text{if } y \neq f(x, \theta) \end{cases}$$

例2: 平方误差损失函数

$$\mathcal{L}(y, \hat{y}) = (y - f(x, \theta))^2$$

回归问题:

□ 什么是回归?

回归分析 (regression analysis) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。即研究X和Y之间关系的统计分析方法称之为回归。

□ 回归模型:

- 线性回归
- 逻辑回归

■ 线性回归模型:

$$y = wx + b$$

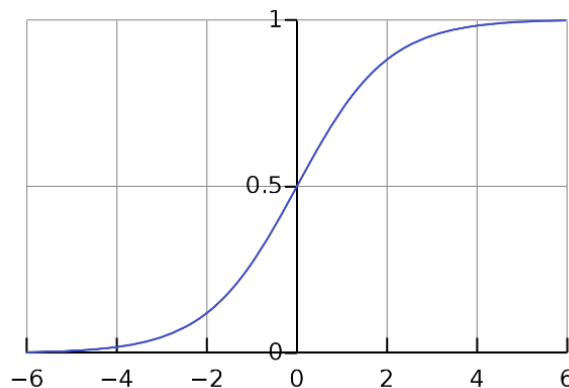
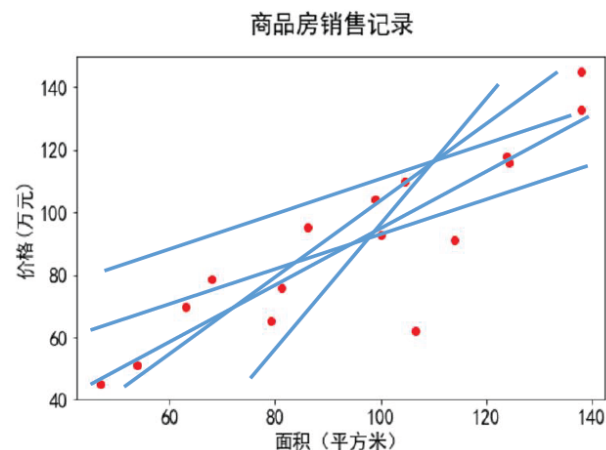
■ Logistic回归模型:

- 逻辑回归模型使用 *logistic* 函数 (或称为 *sigmoid* 函数)

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

$$\text{其中 } z = h_{\theta}(x) = \theta^T x$$

z 变量的范围为 $-\infty$ 到 $+\infty$, 值域限制在 0-1 之间。



回归问题的损失函数:

□ 回归问题常用的损失函数

- 平均绝对误差MAE(mean_absolute_error)
- 均方误差MSE (mean_squared_error)及均方根差RMSE
- Log loss, 或称交叉熵loss(cross - entropy loss)

线性回归的损失函数

回归性能度量方法(Regression metrics)

平均绝对误差MAE函数

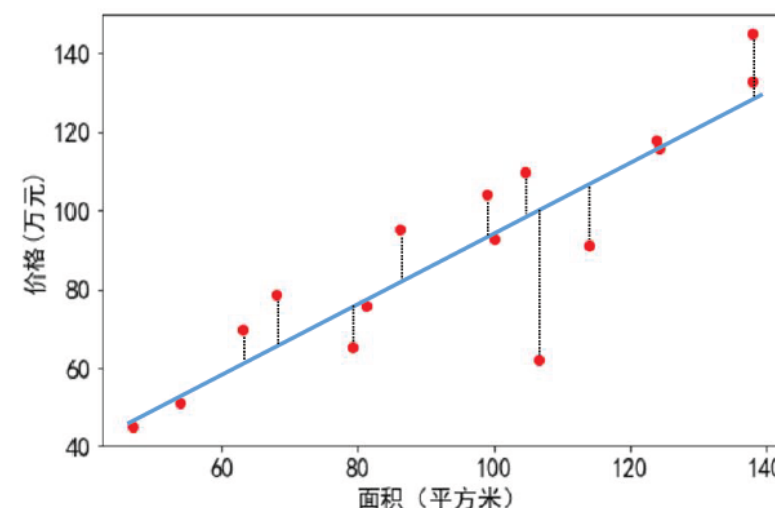
□ MAE (Mean absolute error) 是绝对误差损失(absolute error loss) 的期望值。

□ 如果 \hat{y}_i 是第 i 个样本的预测值, y_i 是相应的真实值, 那么在 $n_{samples}$ 个测试样本上的平均绝对误差 (MAE) 函数的定义如下:

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$$

$$Loss = \sum_{i=1}^n |y_i - \hat{y}_i| = \frac{1}{2} \sum_{i=1}^n |y_i - (wx_i + b)|$$

房屋销售记录



线性回归问题的损失函数

均方差MSE函数

□ **MSE(Mean squared error)**, 该指标对应于平方误差损失 (squared error loss) 的期望值。

□ 如 \hat{y}_i 果 y_i 是第 i 个样本的预测值, y_i 是相应的真实值, 那么在 $n_{samples}$ 个上的均方差 (MSE) 函数的定义如下:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|^2$$

■ **平方损失函数** (Square Loss)

$$Loss = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2} \sum_{i=1}^n (y_i - (wx_i + b))^2$$

■ **均方误差** (Mean Squre Error)

$$Loss = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2n} \sum_{i=1}^n (y_i - (wx_i + b))^2$$

□ **均方根差RMSE**: Root Mean Squared Error, RMSE, 是MSE的平方根

一元线性回归方程

□ **误差函数**：期望输出与真实输出的差别，误差越小，匹配程度越高。

■ 单个样本误差：训练样本 x ，标签值 y 及预测输出 $f(x)$ ，单个训练样本 x 的误差：

$$e = (y - f(x))^2$$

■ 多个样本误差：训练样本 x_1, x_2, \dots, x_m ，标签值： y_1, y_2, \dots, y_m 及预测输出 $f(x_1), f(x_2), \dots, f(x_m)$ ，优化计算的目标函数定义为：

$$J(\mathbf{w}) = \sum_{i=1}^m (y_i - f(x_i))^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

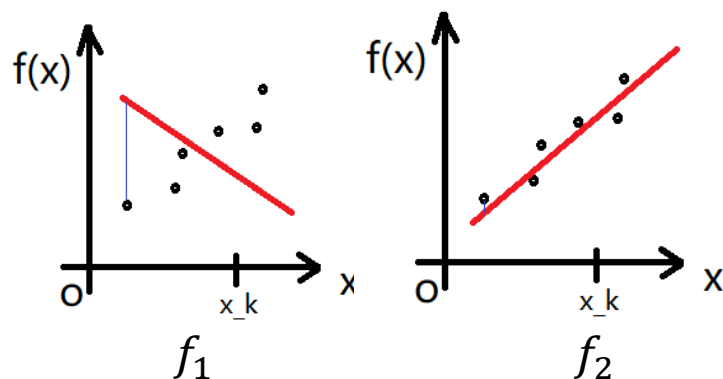
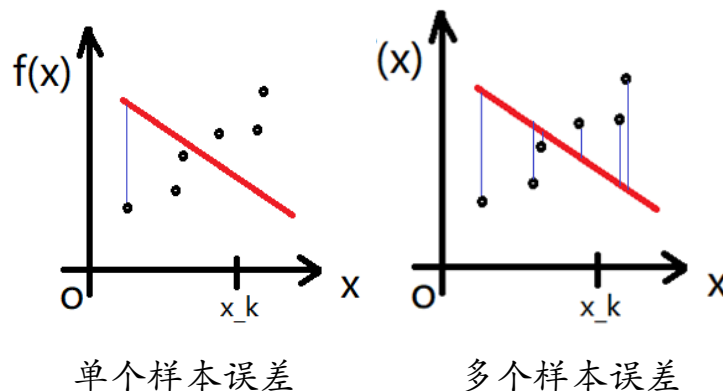
其中 $\mathbf{X} = (x_1, x_2, \dots, x_m)^T$ 为样本向量

即： $f(x) = wx_i + b$ 使得 $f(x_i) \simeq y_i$

□ **最小二乘估计**

基本思想：误差最小化 \Rightarrow 最优参数向量取值对应的整体误差最小

$$\Rightarrow \arg \min_{\mathbf{w}} J(\mathbf{w}) = \arg \min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$



一元线性回归方程

令均方误差最小化, 有

$$\begin{aligned}(w^*, b^*) &= \arg \min_{(w, b)} \sum_{i=1}^m (f(x_i) - y_i)^2 \\ &= \arg \min_{(w, b)} \sum_{i=1}^m (y_i - wx_i - b)^2\end{aligned}$$

对 $E_{(w, b)} = \sum_{i=1}^m (y_i - wx_i - b)^2$ 进行最小二乘参数估计

分别对 w 和 b 求导:

$$\begin{aligned}\frac{\partial E_{(w, b)}}{\partial w} &= 2 \left(w \sum_{i=1}^m x_i^2 - \sum_{i=1}^m (y_i - b) x_i \right) \\ \frac{\partial E_{(w, b)}}{\partial b} &= 2 \left(mb - \sum_{i=1}^m (y_i - wx_i) \right)\end{aligned}$$

令导数为 0, 得到闭式(closed-form)解:

$$\begin{aligned}w &= \frac{\sum_{i=1}^m y_i (x_i - \bar{x})}{\sum_{i=1}^m x_i^2 - \frac{1}{m} \left(\sum_{i=1}^m x_i \right)^2} \\ b &= \frac{1}{m} \sum_{i=1}^m (y_i - wx_i)\end{aligned}$$

多元(multi-variate)线性回归

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b \quad \text{使得} \quad f(\mathbf{x}_i) \simeq y_i$$

每一样本含多个变量 $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{id})$ $y_i \in \mathbb{R}$

把 \mathbf{w} 和 b 吸收入向量形式 $\hat{\mathbf{w}} = (\mathbf{w}; b)$ ，数据集表示为

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^T & 1 \end{pmatrix} \quad \mathbf{y} = (y_1; y_2; \dots; y_m)$$

多元线性回归

同样采用最小二乘法求解，有

$$\hat{\mathbf{w}}^* = \arg \min_{\hat{\mathbf{w}}} (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$$

令 $E_{\hat{\mathbf{w}}} = (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$ ，对 $\hat{\mathbf{w}}$ 求导：

$$\frac{\partial E_{\hat{\mathbf{w}}}}{\partial \hat{\mathbf{w}}} = 2\mathbf{X}^T (\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}) \quad \text{令其为零可得 } \hat{\mathbf{w}}$$

然而，麻烦来了：涉及矩阵求逆！

- 若 $\mathbf{X}^T \mathbf{X}$ 满秩或正定，则 $\hat{\mathbf{w}}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
- 若 $\mathbf{X}^T \mathbf{X}$ 不满秩（如变量数大于样例数的情况），则可解出多个 $\hat{\mathbf{w}}$

此时需求助于归纳偏好，或引入 **正则化** (regularization) → 第6、11章

似然函数与均方误差的关系

□ 似然函数定义

■ 定义1: (似然函数) 给定随机变量 Y , 定义

$$\text{Like}(\mathbf{w}|y^{(1)}, \dots, y^{(m)}) = \prod_{i=1}^m p_{\mathbf{w}}(Y = y^{(i)})$$

为 Y 的 m 个独立采样恰为 $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ 的概率, 称为概率分布 $p_{\mathbf{w}}$ 关于 $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ 的似然函数。

似然函数与均方误差的关系

□ 最大似然原则

■ 概率统计中的最大似然原则：

如果 $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ 为 Y 的 m 个独立采样，而 \mathbf{w}^* 是使得似然函数最大化的一组参数，即

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^n} \text{Like}(\mathbf{w} | y^{(1)}, \dots, y^{(m)})$$

则可以断定 Y 的概率分布是 $p_{\mathbf{w}^*}$ 。

似然函数与均方误差的关系

□ 线性模型均方误差的定义:

■ 定义2 (均方误差) 线性回归目标函数 (损失函数)

$$\frac{1}{m} \sum_{i=1}^m (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b - y^{(i)})^2$$

称为均方误差。式中, $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$ 均为 n 维向量, $b \in$

\mathbb{R} 为偏置项, $\langle \mathbf{w}, \mathbf{x} \rangle$ 表示 \mathbf{w} 与 \mathbf{x} 的内积。

似然函数与均方误差的关系

- 对于线性模型，设预测结果 $(\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b)$ 与真实结果 $y^{(i)}$ 之间误差为 ϵ^i ，即： $y^{(i)} = \langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b + \epsilon^i$
- 通常误差满足平均值为0的高斯分布，即正态分布。那么在一个样本 i 上 x 和 y 的概率密度公式为：

$$p = (y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2}{2\sigma^2}\right)$$

- 设基于参数为 \mathbf{w}, b 的以上正态分布模型对于 $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ 全部样本的似然函数为：

$$\text{最大化: } L(\mathbf{w}, b) = \text{Like}(\mathbf{w}, b | y^{(1)}, \dots, y^{(m)}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2}{2\sigma^2}\right)$$
$$\text{取对数: } \ln L(\mathbf{w}, b) = -m \ln(\sqrt{2\pi}\sigma) - \sum_{i=1}^m \frac{(y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2}{2\sigma^2}$$

$$\text{最小化: } -\ln L(\mathbf{w}, b) = m \ln(\sqrt{2\pi}\sigma) + \sum_{i=1}^m \frac{(y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2}{2\sigma^2}$$

似然函数与均方误差的关系

从而需要 $\sum_{i=1}^m (y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2$ 最小 \Rightarrow 线性模型损失函数: $\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^m (y^{(i)} - (\langle \mathbf{w}, \mathbf{x}^{(i)} \rangle + b))^2$

似然函数与均方误差的关系

□ 一般地，设参数为 \mathbf{w}, b 的正态分布 $N(h_{\mathbf{w},b}(\mathbf{x}), \sigma^2)$ 对于 $y^{(1)}, y^{(2)}, \dots, y^{(m)}$ 的似然函数为：

$$\text{Like}(\mathbf{w}, b | y^{(1)}, \dots, y^{(m)}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(h_{\mathbf{w},b}(\mathbf{x}^{(i)}) - y^{(i)})^2}{2\sigma^2}}$$

该似然函数与均方误差有以下关系：

$$\operatorname{argmax}_{\mathbf{w}, b} \text{Like}(\mathbf{w}, b | y^{(1)}, \dots, y^{(m)}) = \operatorname{argmin}_{\mathbf{w}, b} \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w},b}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

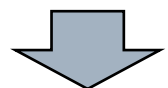
■ 即使：最大化似然函数等价于最小化均方误差。

线性模型的变化

- 假设样例 (x, y) 所对应的输出标记是在指数尺度上变化（如右图坐标点），能否使用线性模型进行逼近？

对于样例 (x, y) ，若希望线性模型的预测值逼近真实标记，

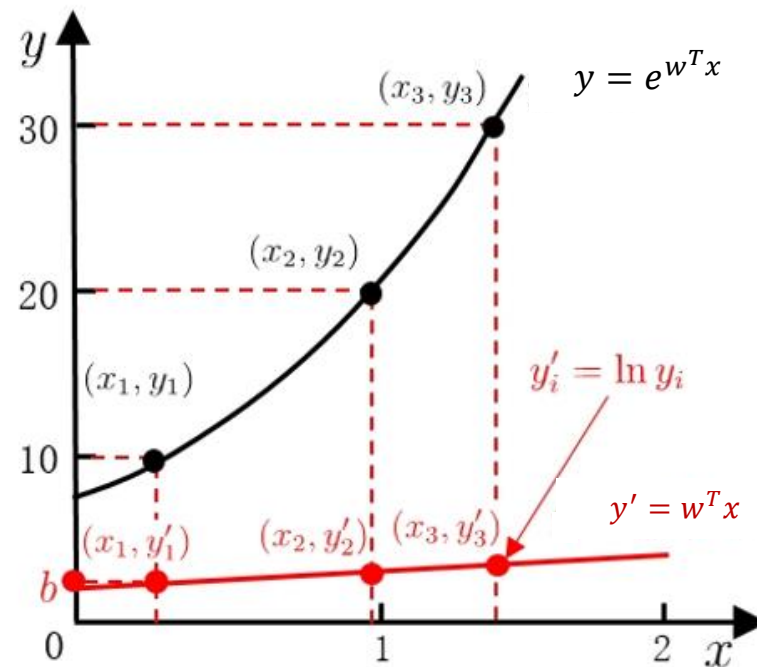
则从真实标记模型 $y = e^{w^T x}$



用线性模型逼近 y 的衍生物

对数线性回归： $\ln y = w^T x$

该模型实际上是用 $e^{w^T x}$ 逼近 y 。虽然形式上仍是线性回归，但实质上求取了输入空间到输出空间的非线性函数映射。--log link



对数线性回归

广义线性模型： 用单调可微函数 $g(\cdot)$ 将线性回归模型与真实标记联系： $y = g^{-1}(w^T x)$

广义线性模型

- 广义线性(generalized)模型是线性模型的扩展，主要通过联结函数 g (link function)，使预测值落在响应变量的变幅内。例如逻辑回归

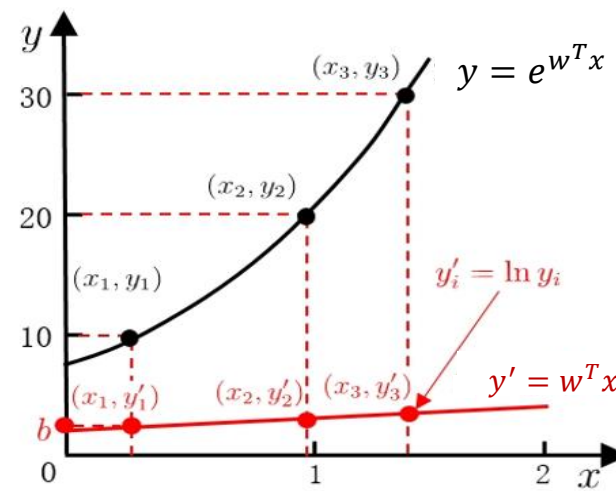
$$y = h_{\theta}(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad (\text{括号内为线性函数})$$

一般形式: $y = g^{-1}(w^T x + b)$

单调可微的 **联结函数** (link function)

令 $g(\cdot) = \ln(\cdot)$ 则得到对数线性回归

$$\ln y = w^T x + b$$



二分类任务

- 在分类问题中，每一个样本都有一个标签，用于表示这个样本所属的类别。
- 分类任务是对给定的特征组 x 预测对象属于每一个类别的概率。由于预测的对象是一个概率，所以线性回归模型就不适用了。

线性回归模型产生的实值输出 $z = \mathbf{w}^T \mathbf{x} + b$
期望输出 $y \in \{0, 1\}$

找 z 和 y 的联系函数，
将实值 z 转换为 0/1 值

理想的“单位阶跃函数” (unit-step function)

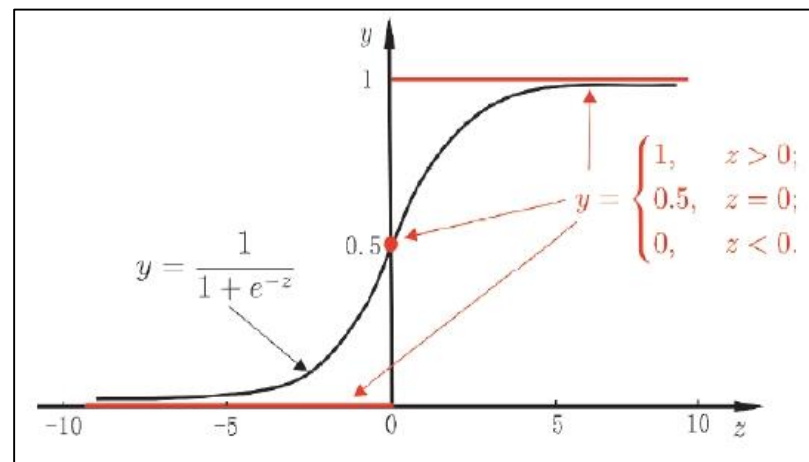
$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$

常用单调可微、
任意阶可导

性质不好，
需找“替代函数”
(surrogate function)

$$y = \frac{1}{1 + e^{-z}}$$

对数几率函数，简称“对率函数”
(logistic function)



二分类的对率回归

□ 以对率函数作为广义线性模型联系函数 g

$$y = \frac{1}{1 + e^{-z}} = g(w^T x + b) \quad \text{变为:} \quad y = \frac{1}{1 + e^{-(w^T x + b)}} \quad (\text{括号内为线性函数})$$

则: $\ln \frac{y}{1-y} = w^T x + b$

若将 y 看作类后验概率估计 $p(y = 1 | \mathbf{x})$

$\frac{y}{1-y}$ 称为几率(odds), 反映了 \mathbf{x} 作为正例的相对可能性

则 $\ln \frac{y}{1-y} = w^T x + b$ 可写为

$$\ln \frac{p(y = 1 | \mathbf{x})}{p(y = 0 | \mathbf{x})} = w^T x + b$$

称为: “对数几率” (log odds, 亦称 logit)

“对数几率回归” (logistic regression), 简称“对率回归”, 也称为“逻辑回归”

注意: 它是分类学习算法!

用线性回归模型的预测结果去逼近真实标记的对数几率

回归 (logistic) 问题的损失函数

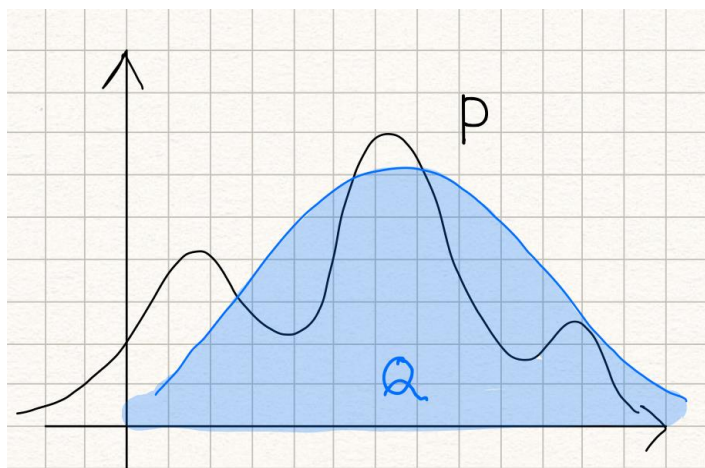
logistic回归损失(二类)

□ 简称Log loss，或交叉熵损失(cross-entropy loss)

- 设：真实条件概率 $p_r(y|x)$, 模型预测条件概率 $p_\theta(y|x)$
- 如何衡量两个条件分布的差异？

□ KL散度

$$D_{\text{KL}}(p_r(y|x) || p_\theta(y|x)) = \sum_{y=1}^C p_r(y|x) \log \frac{p_r(y|x)}{p_\theta(y|x)}$$
$$\propto - \sum_{y=1}^C p_r(y|x) \log p_\theta(y|x) \quad \text{交叉熵损失}$$



□ 交叉熵(cross entropy)

定义：交叉熵是信息论中一个重要的概念, 用于表征两个变量概率分布P, Q（假设P表示真实分布, Q为模型预测的分布）的**差异性**。交叉熵越大, 两个变量差异程度越大。

交叉熵公式：

$$H(P, Q) = - \sum_{x \in X} P(x) \log Q(x)$$

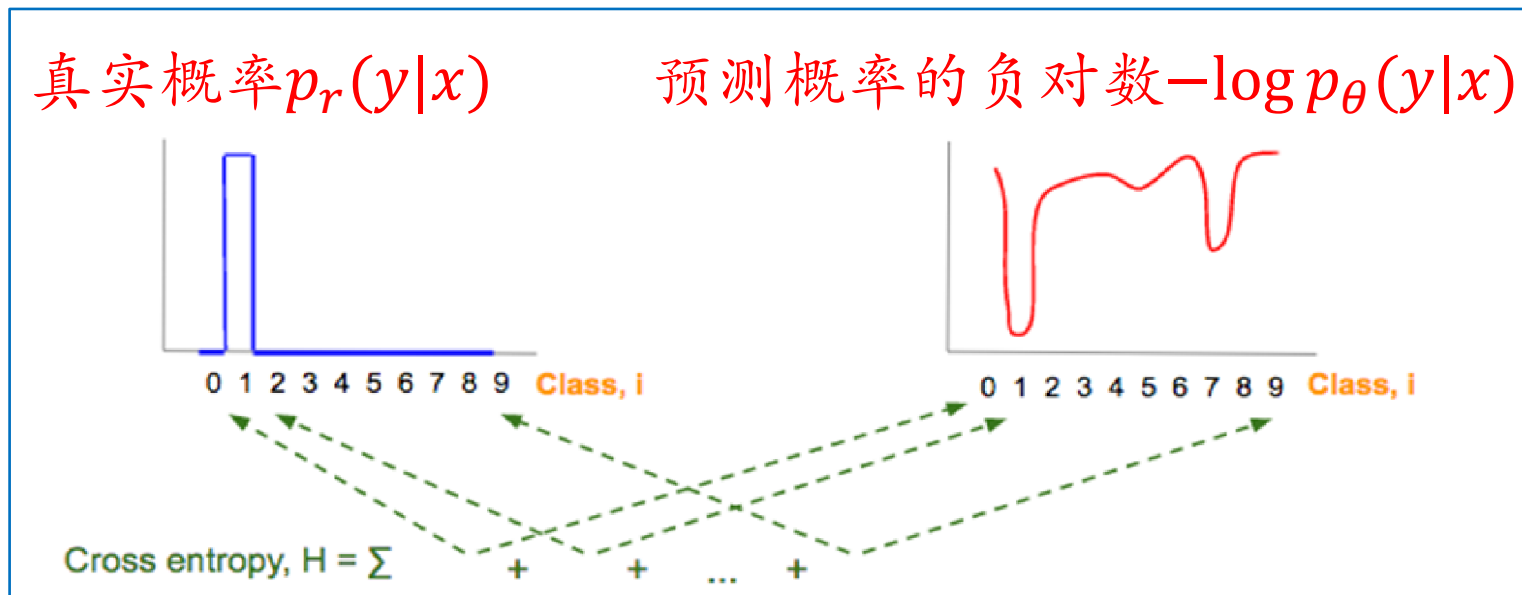
回归 (logistic) 问题的损失函数

□ 交叉熵损失:

(cross-entropy loss 损失)

$$-\sum_{y=1}^C p_r(y|x) \log p_\theta(y|x) \quad \text{负对数似然}$$

$$\mathcal{L}(\mathbf{y}, f(\mathbf{x}, \theta)) = -\sum_{c=1}^C y_c \log f_c(\mathbf{x}, \theta)$$



□ 对于二类分类问题:

- 假设某样本的真实标签为 y (取值为0或1), 概率估计为 $p = p_\theta(y = 1)$
- 每个样本的 log loss 是对分类器给定真实标签的负 log 似然估计 (negative log - likelihood)

$$L_{log}(y, p) = -\log(p_\theta(y|p)) = -(y \log(p) + (1 - y) \log(1 - p))$$

回归 (logistic) 问题的损失函数 (二分类)

交叉熵公式: $L_{\log}(y, p) = -\log(p_{\theta}(y|p)) = -(y \log(p) + (1 - y) \log(1 - p))$

- 假设 $y_true = [0, 0, 1, 1]$
 $y_pred = [[.9, .1], [.8, .2], [.3, .7], [.01, .99]]$

对于第一个样本, $y=0, p=0.1$

$$\text{则 } L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) = -(1 * \log 0.9)$$

对于第二个样本, $y=0, p=0.2,$

$$\text{则 } L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) = -(1 * \log 0.8)$$

对于第三个样本, $y=1, p=0.7,$

$$\text{则 } L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) = -(1 * \log 0.7)$$

对于第四个样本, $y=1, p=0.99,$

$$\text{则 } L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) = -(1 * \log 0.99)$$

以对数e为底数, 则上述四项的均值为0.1738 (Sklearn中默认为以e为底)

回归 (logistic) 问题的损失函数 (多分类)

- 对于多类问题(multiclass problem), 可将样本的真实标签 (true label) 编码成1-of-K (K为类别总数) 的二元指示矩阵Y:

- 转换举例: 假设K=3, 即三个类

$$Y_{\text{true}} = \begin{bmatrix} 1 & 2 & 3 \\ [1, 0, 0] & [0, 1, 0] & [0, 0, 1] \end{bmatrix}$$

分类标签向量表示为:
One-hot 编码

- 假设模型对测试样本的概率估计结果为P, 则在测试集(假设测试样本总数为N)上的交叉熵损失表示如下:

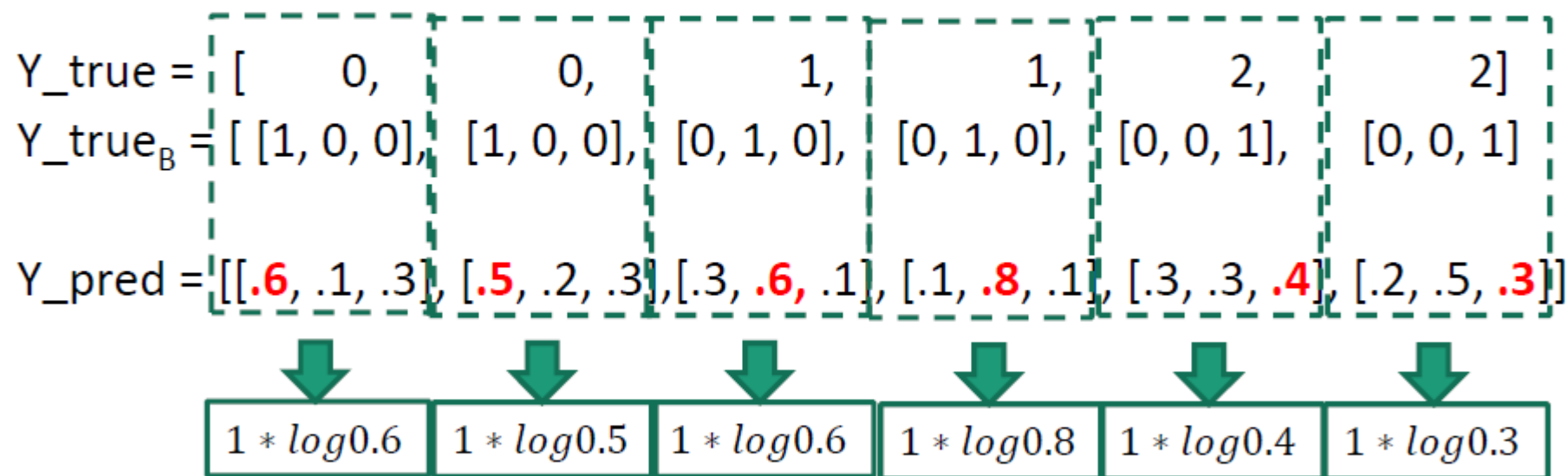
$$L_{\log}(Y, P) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}$$

其中 $y_{i,k}$ 表示第*i*个样本的第*k*个标签的真实值, 注意由于表示为“1-of-K”模式, 因此每个样本只有其中一个标签值为1, 其余均为0。 $p_{i,k}$ 表示模型对该样本的预测值。

回归 (logistic) 问题的损失函数 (多分类)

根据公式, $L_{log}(Y, P) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}$

● 举例: 6个样本, 三个类

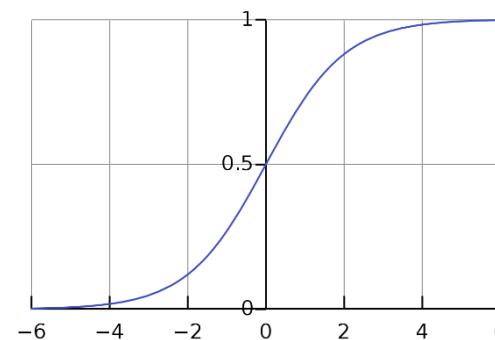


则 $L_{log}(Y, P) = -(\log 0.6 + \log 0.5 + \log 0.6 + \log 0.8 + \log 0.4 + \log 0.3)/6 = 0.6763$

两个重要的分类器

□ Logistic回归

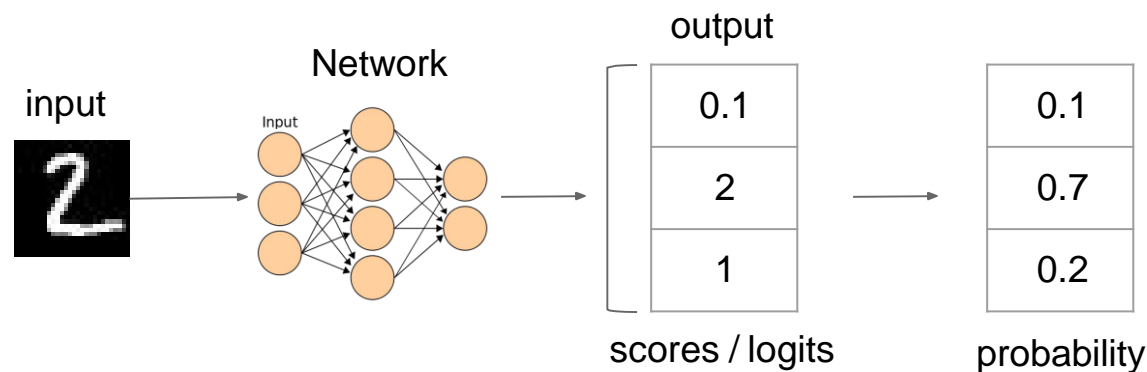
$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) \triangleq \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$



□ Softmax回归

$$P(y = c|\mathbf{x}) = \text{softmax}(\mathbf{w}_c^T \mathbf{x}) \\ = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^T \mathbf{x})}$$

- 把分数转换成概率
 - 从 0.0 到 1.0
 - 所有类的概率之和为1.0



回归和分类的区别和联系

□ 区别：

- 分类：使用训练集推断输入 x 所对应的离散类别（如：+1, -1）
- 回归：使用训练集推断输入 x 所对应的输出值，为连续实数。

□ 联系：

- 利用回归模型进行分类：可将回归模型的输出离散化以进行分类，
即： $y = \text{sign}(f(x))$
- 利用分类模型进行回归：也可利用分类模型的特点，输出其连续化的数值。

最优化问题（求loss函数最小值）

□ 机器学习问题转化为一个最优化问题

$$\min_x f(x)$$

例：均方误差 \mathcal{L}

$$Loss = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2n} \sum_{i=1}^n (y_i - (wx_i + b))^2$$

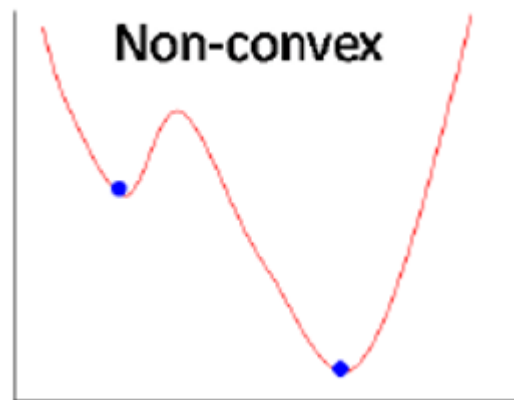
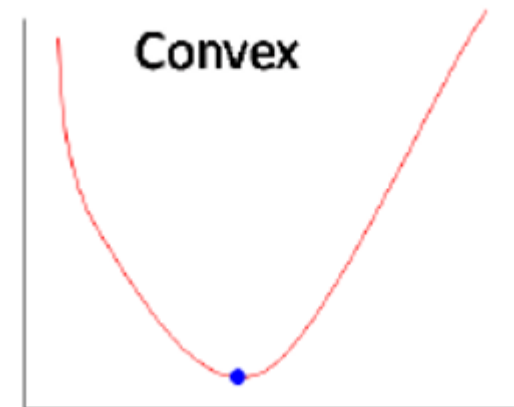
有：

$$\min_{w,b} Loss(w, b)$$

求极值问题：w.b取何值时，Loss取值最小？

经验风险最小化原则

□ 优化求取方法：1. 最小二乘法
2. 梯度下降法



梯度法

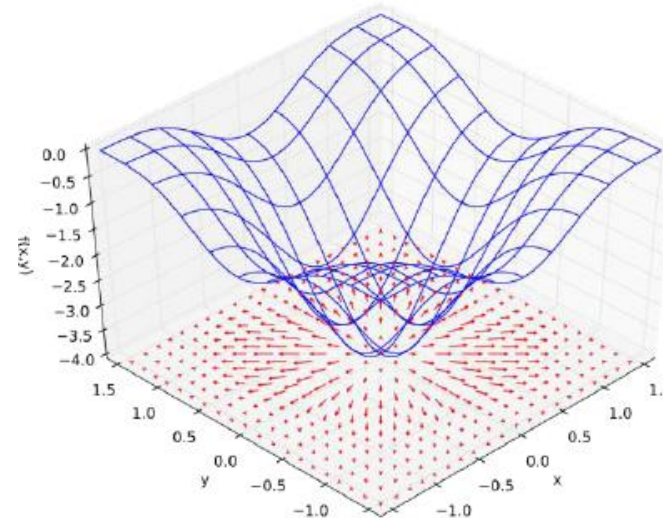
▣ 梯度的定义:

设二元函数 $z=f(x,y)$ 在平面区域D上具有一阶连续偏导数, 则对于每一个点 $P(x,y)$ 都可定出一个向量:

$$\left\{ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right\} = f_x(x,y)\bar{i} + f_y(x,y)\bar{j}$$

该函数就称为函数 $z=f(x,y)$ 在点 $P(x,y)$ 的梯度, 记作 $\text{grad}f(x,y)$ 或 $\nabla f(x,y)$

图中 xy 平面的红色箭头的长度表示梯度向量的模, 箭头的方向表示梯度向量的方向。可以看到, 箭头的方向总是指向当前位置函数值增速最大的方向, 函数曲面越陡峭, 箭头的长度也就越长, 梯度的模也越大。



函数及其梯度向

由图可见梯度的反方向 $-\nabla f$ 应指向函数值减少的方向

$$x' = x - \eta \cdot \nabla f$$


梯度法

对于神经网络: $f(\mathbf{w}) = f(w_1, w_2, \dots, w_m)$ 是 R^m 上的函数

$$df = \frac{\partial f}{\partial w_1} dw_1 + \frac{\partial f}{\partial w_2} dw_2 + \dots + \frac{\partial f}{\partial w_m} dw_m$$

定义:
$$\nabla f = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \dots, \frac{\partial f}{\partial w_m} \right)^T$$

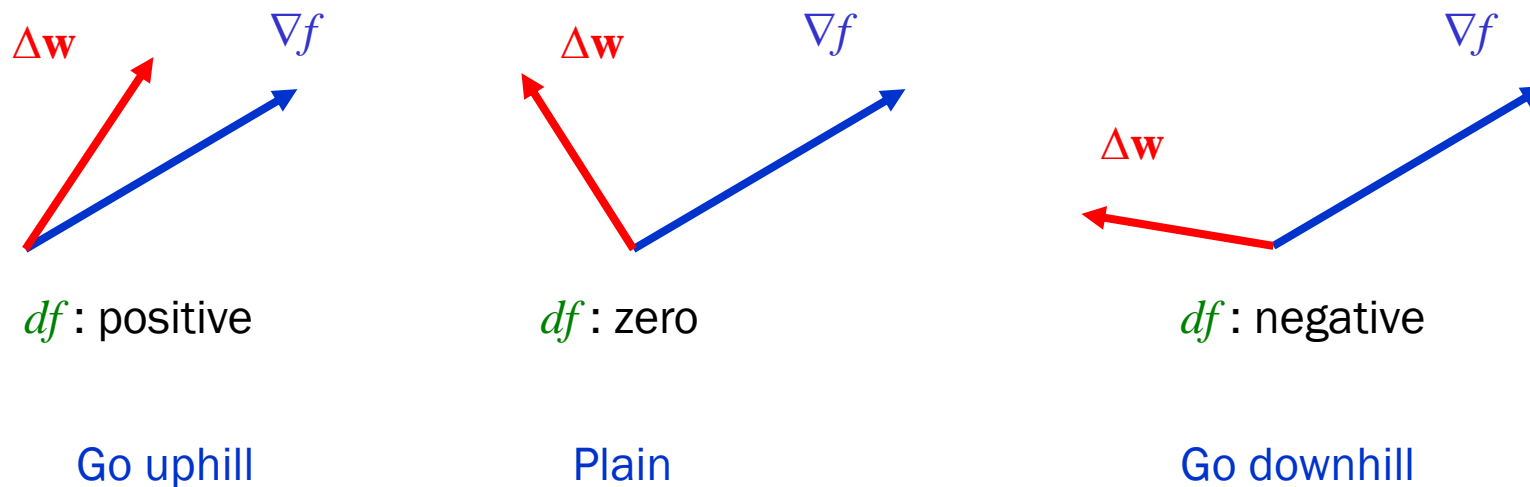
$$\Delta \mathbf{w} = (dw_1, dw_2, \dots, dw_m)^T$$


$$df = \langle \nabla f, \Delta \mathbf{w} \rangle = \nabla f \bullet \Delta \mathbf{w}$$

可见:

∇f 把 w 的变化关联为 f 的变化

梯度法



$$df = \langle \nabla f, \Delta \mathbf{w} \rangle = \nabla f \bullet \Delta \mathbf{w}$$

梯度最速下降方向

为了最小化 f , 选择

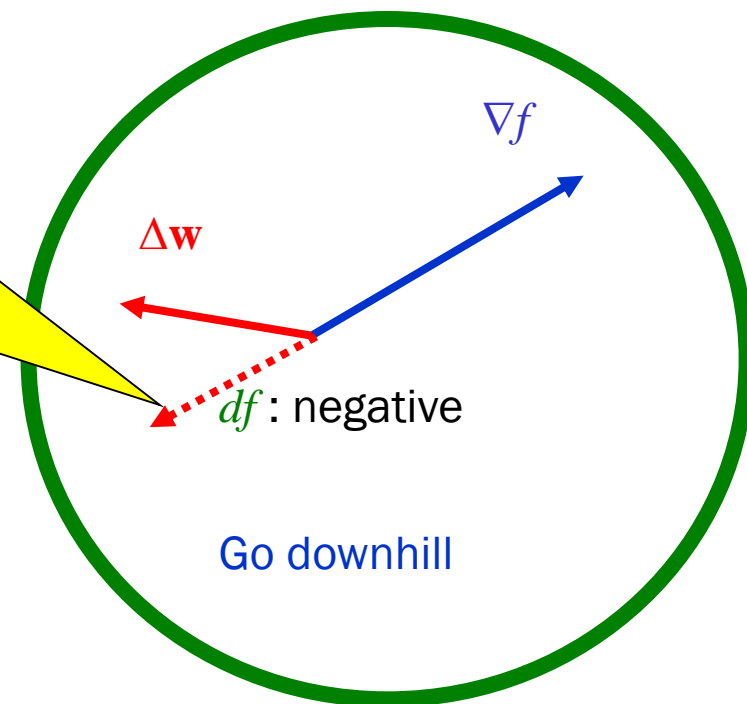
$$\Delta \mathbf{w} = -\eta \nabla f$$

df : positive

Go uphill

df : zero

Plain



$$df = \langle \nabla f, \Delta \mathbf{w} \rangle = \nabla f \bullet \Delta \mathbf{w}$$

优化：梯度下降法

梯度下降法，就是利用负梯度方向来决定每次迭代的新的搜索方向，使得每次迭代能使待优化的目标函数逐步减小。最速下降法的一种简单形式是：

$$f_x(k+1) = f_x(k) + \Delta f_x(k)$$

$$\text{由: } \Delta f_x(k) = \nabla f \cdot \Delta x, \quad \Delta x = -\eta \nabla f$$

$$\text{有: } \Delta f_x(k) = -\eta (\nabla f)^2$$

$$\Delta f_x(k) = -\eta \|\nabla f\|^2$$

由于 $\|\nabla f\|^2 > 0$ ，保证了 $\Delta f < 0$ ，

其中： η 称为学习速率，可以是较小的常数。 ∇f 是 $f(k)$ 的梯度。

梯度下降法公式：

$$\Delta x = -\eta \nabla f$$

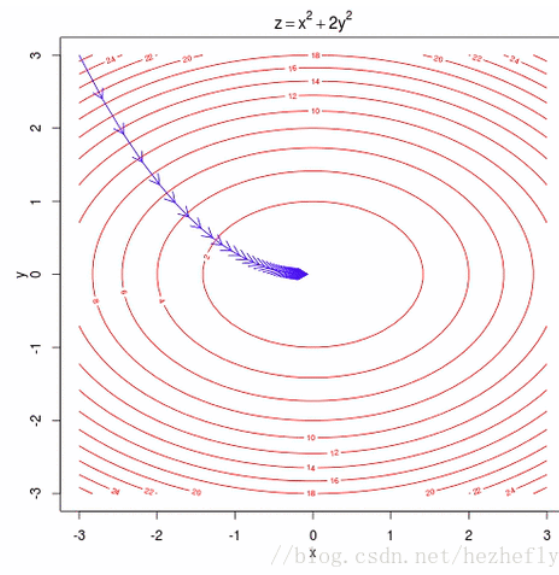
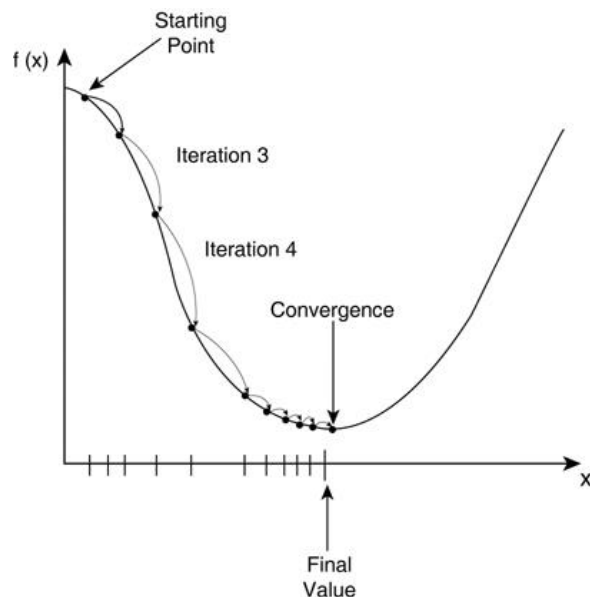
$$x(t+1) = x(t) - \eta \nabla f$$

神经网络问题

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (wx^{(i)} + b - y^{(i)})^2$$

$$w(t+1) = w(t) - \eta \frac{\partial \mathcal{L}}{\partial w}$$

$$b(t+1) = b(t) - \eta \frac{\partial \mathcal{L}}{\partial b}$$



$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w} &= \frac{1}{n} \sum_{i=1}^n 2(wx^{(i)} + b - y^{(i)}) \cdot \frac{\partial (wx^{(i)} + b - y^{(i)})}{\partial w} = \frac{1}{n} \sum_{i=1}^n 2(wx^{(i)} + b - y^{(i)}) \cdot x^{(i)} \\ &= \frac{2}{n} \sum_{i=1}^n (wx^{(i)} + b - y^{(i)}) \cdot x^{(i)}\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial b} &= \frac{\partial \frac{1}{n} \sum_{i=1}^n (wx^{(i)} + b - y^{(i)})^2}{\partial b} \\ &= \frac{1}{n} \sum_{i=1}^n \frac{\partial (wx^{(i)} + b - y^{(i)})}{\partial b} = \frac{1}{n} \sum_{i=1}^n 2(wx^{(i)} + b - y^{(i)}) \cdot \frac{\partial (wx^{(i)} + b - y^{(i)})}{\partial b} \\ &= \frac{1}{n} \sum_{i=1}^n (wx^{(i)} + b - y^{(i)}) \cdot 1\end{aligned}$$

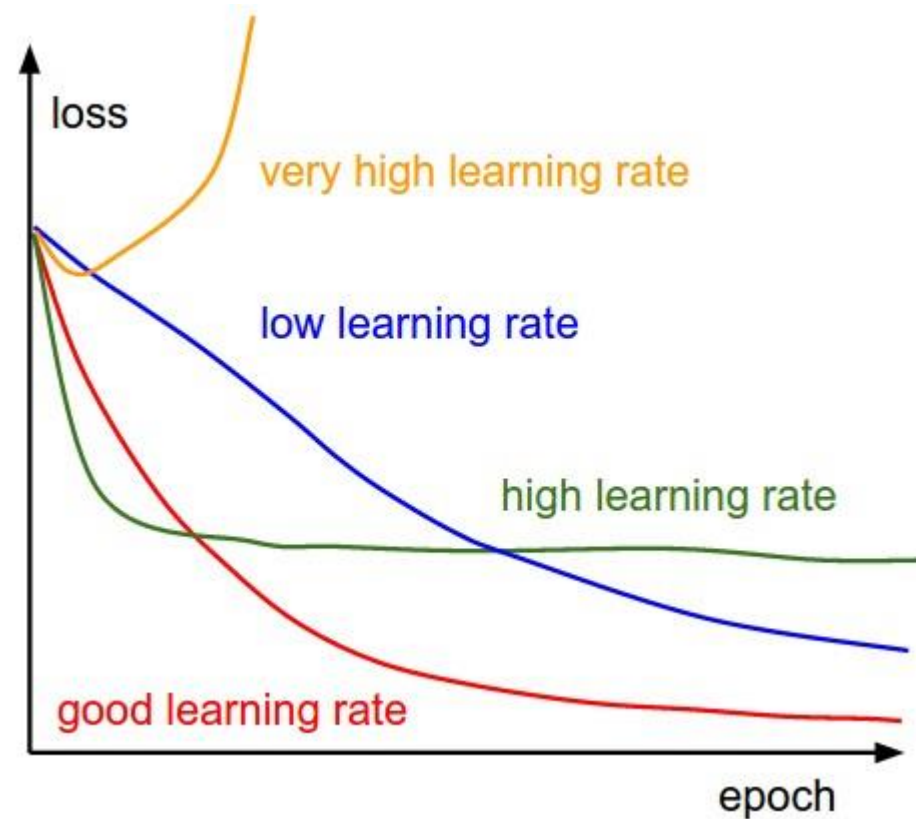
批量梯度下降法

$$Loss = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2n} \sum_{i=1}^n (y_i - (wx_i + b))^2$$

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

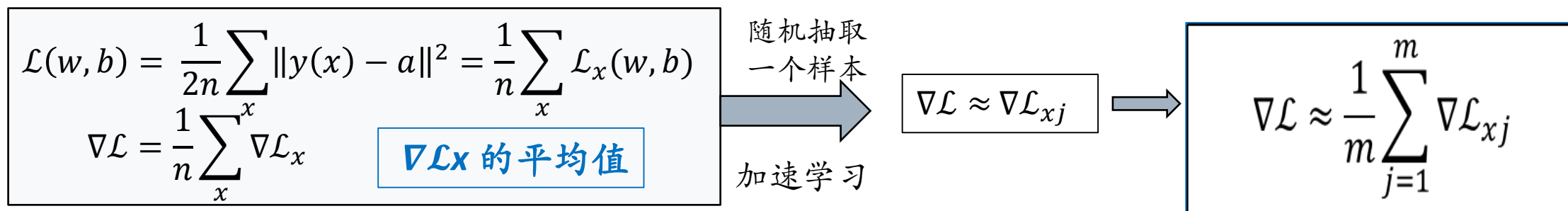
搜索步长 η 中也叫作学习率
(Learning Rate)



随机梯度下降法

□ 随机梯度下降法 (Stochastic Gradient Descent, SGD)

是一种加速学习的方法。也叫增量梯度下降，随机抽取一个样本计算。



□ 小批量 (Mini-Batch) 随机梯度下降法

将一些随机的训练输入标记为 $X_1; X_2; \dots; X_m$ ，并把它们称为一个小批量数据 (mini-batch)

假设样本数量 m 够大，我们期望 $\nabla \mathcal{L}_{xj}$ 的平均值大致等于整个 $\nabla \mathcal{L}_x$ 的平均值，即，

(Mini-Batch) 随机梯度下降法

The diagram shows the Mini-Batch gradient approximation and the corresponding parameter update rules. It starts with a box containing the approximation:

$$\nabla \mathcal{L} \approx \frac{1}{m} \sum_{j=1}^m \nabla \mathcal{L}_{xj}$$

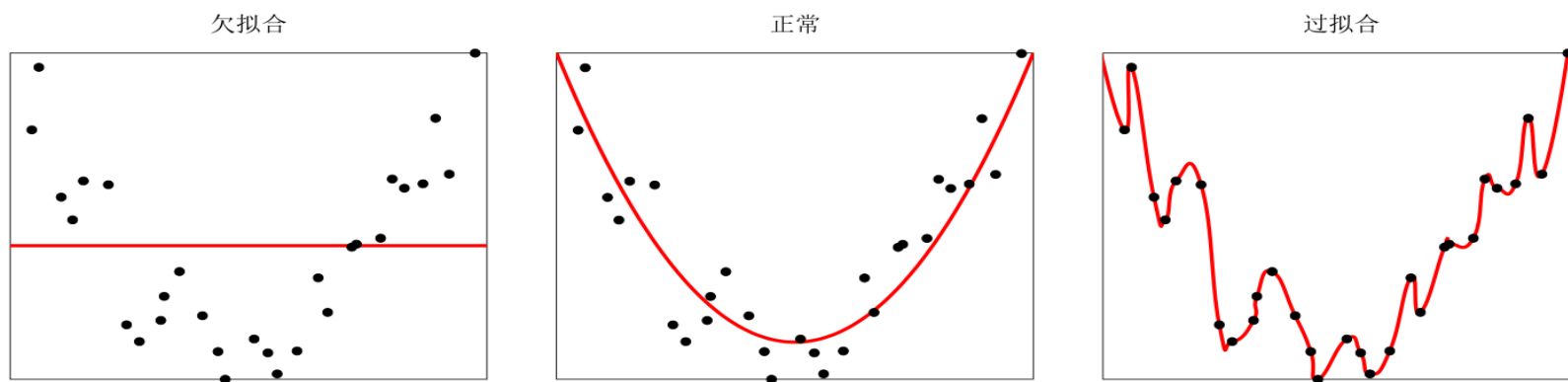
An arrow points to a box containing the update rules:

$$w_k \rightarrow w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$
$$b_l \rightarrow b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}$$

机器学习 = 优化?

机器学习 = 优化?

NO!



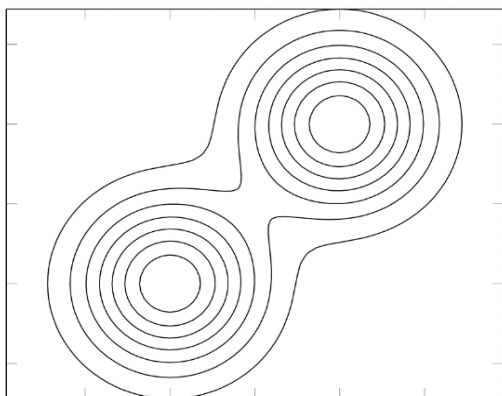
过拟合：经验风险最小化原则很容易导致模型在训练集上错误率很低，但是在未知数据上错误率很高。

泛化错误

期望风险

$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)],$$

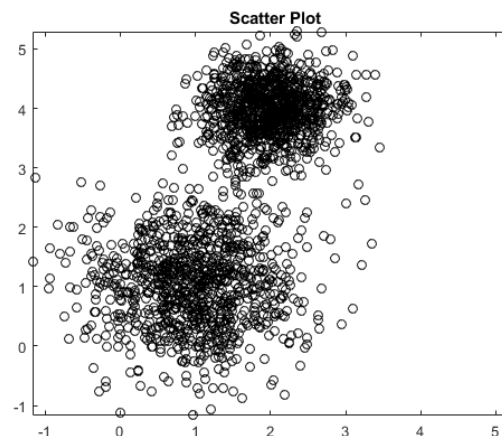
真实分布 p_r



\neq

经验风险

$$\mathcal{R}_D^{emp}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$



$$\mathcal{G}_D(f) = \mathcal{R}(f) - \mathcal{R}_D^{emp}(f)$$

泛化错误

如何减少泛化错误?

优化

经验风险最小

正则化

降低模型复杂度



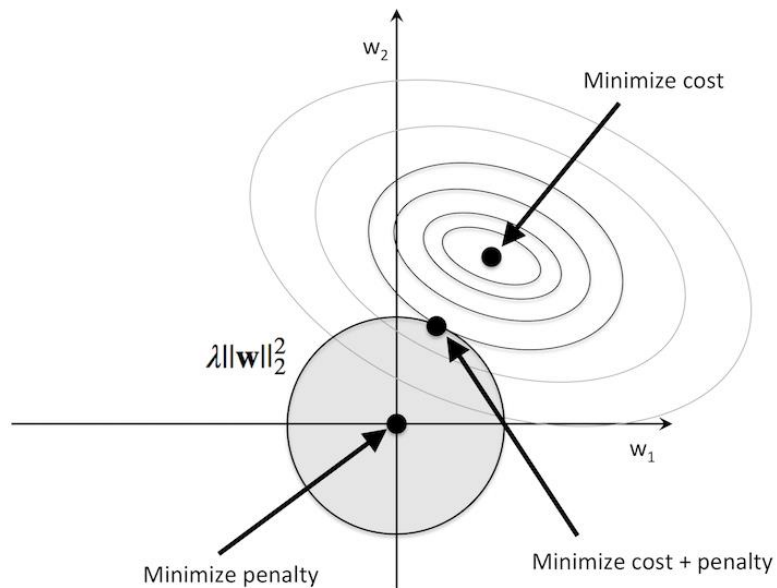
正则化 (regularization)

所有损害优化的方法都是正则化。

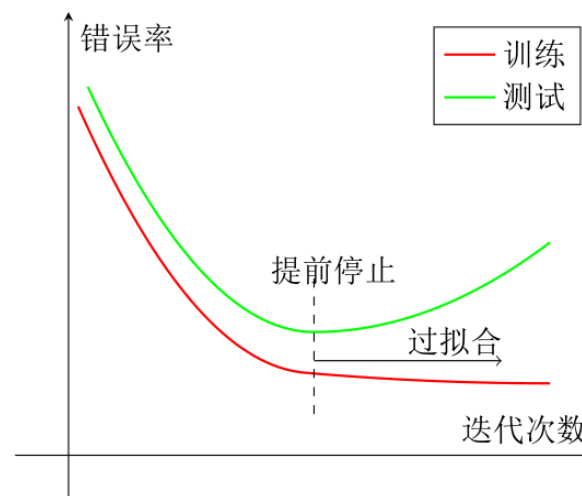
增加优化约束

干扰优化过程

L1/L2约束、数据增强



权重衰减、随机梯度下降、提前停止



机器学习的三要素

□ 模型

- 线性方法: $f(\mathbf{x}, \theta) = \mathbf{w}^T \mathbf{x} + b$

- 广义线性方法: $f(\mathbf{x}, \theta) = \mathbf{w}^T \phi(\mathbf{x}) + b$

- ◆ 如果 $\phi(\mathbf{x})$ 为可学习的非线性基函数, $f(\mathbf{x}, \theta)$ 就等价于神经网络。

□ 学习准则

- 期望风险

$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)], \quad \text{其中: } \mathcal{L} \text{ 为损失函数}$$

□ 优化

- 梯度下降

学习准则

□ 期望风险未知，通过经验风险近似

■ 训练数据： $\mathcal{D} = \{x^{(n)}, y^{(n)}\}, i \in [1, N]$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$

其中： \mathcal{L} 为损失函数
风险函数

□ 经验风险最小化

■ 在选择合适的风险函数后，我们寻找一个参数 θ^* ，使得经验风险函数最小化。

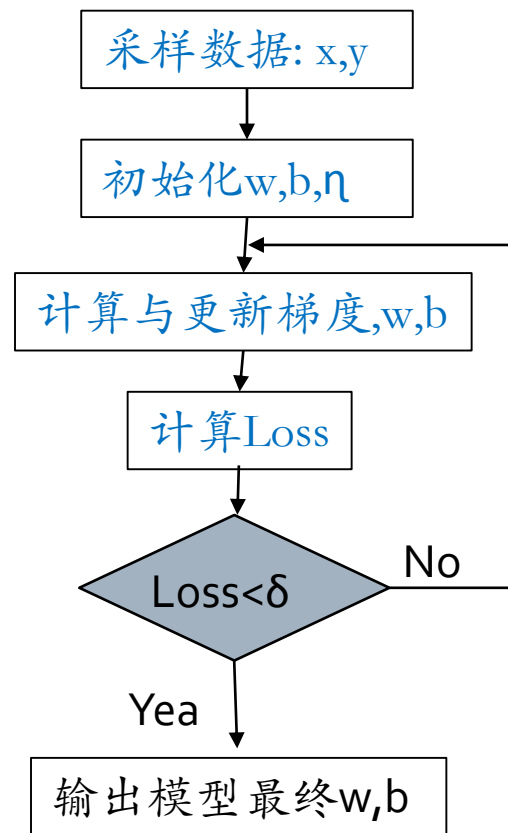
$$\theta^* = \arg \min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

□ 机器学习问题转化成为一个最优化问题

实例：

□ 例：从房屋销售数据中学习

- 采样数据: x, y
- 建立模型 ($f(x) = w^T x + b$)
- 学习模型参数
计算与更新梯度，迭代求取参数 w, b
- 预测房价

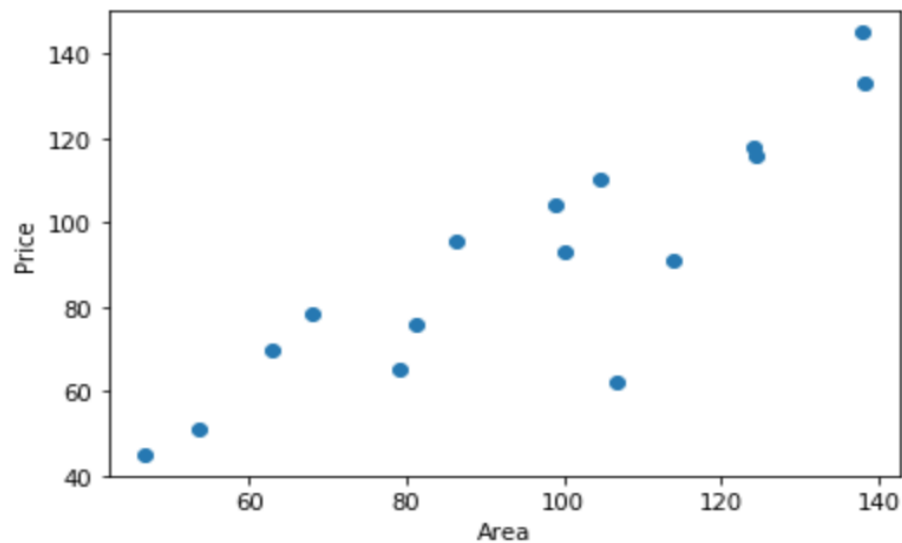


序号	面积 (平方米)	销售价格 (万元)	序号	面积 (平方米)	销售价格 (万元)
1	137.97	145.00	9	106.69	62.00
2	104.50	110.00	10	138.05	133.00
3	100.00	93.00	11	53.75	51.00
4	124.32	116.00	12	46.91	45.00
5	79.20	65.32	13	68.00	78.50
6	99.00	104.00	14	63.02	69.65
7	124.00	118.00	15	81.26	75.69
8	114.00	91.00	16	86.21	95.30

采集数据样本

```
In [7]: 1 import numpy as np
        2 import tensorflow as tf
        3 from matplotlib import pyplot as plt
        4
        5 x=np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00, 106.69, 138.05, 53.75, 46.91, 68.00, 63.02, 81.26, 86.21])
        6 y=np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00, 62.00, 133.00, 51.00, 45.00, 78.50, 69.65, 75.69, 95.30])
        7
        8 %matplotlib inline
        9
       10 plt.scatter(x, y)
       11 plt.xlabel('Area')
       12 plt.ylabel('Price')
       13
```

Out[7]: Text(0, 0.5, 'Price')



计算误差 (Loss函数)

```
1 # 计算误差
2 data=np.stack((x,y),axis=1) #转换为2D Numpy数组
3 print("data=",data)
4
5 def mse(b,w,data):
6     totalError=0
7     for i in range(0, len(data)) :
8         x=data[i, 0]
9         y=data[i, 1]
10        totalError+=(y-(w*x+b))**2
11    return totalError/float((data.size)/2)
12
```

```
data= [[137.97 145. ]
 [104.5  110. ]
 [100.   93. ]
 [124.32 116. ]
 [ 79.2  65.32]
 [ 99.   104. ]
 [124.   118. ]
 [114.   91. ]
 [106.69 62. ]
 [138.05 133. ]
 [ 53.75 51. ]
 [ 46.91 45. ]
 [ 68.   78.5 ]
 [ 63.02 69.65]
 [ 81.26 75.69]
 [ 86.21 95.3 ]]
```

计算梯度

#计算梯度

```
def step_gradient(b_current, w_current, data, lr):
```

```
    b_gradient=0
```

```
    w_gradient=0
```

```
    M=16
```

```
    for i in range(0, len(data)):
```

```
        x=data[i, 0]
```

```
        y=data[i, 1]
```

```
        b_gradient+=(2/M) * ((w_current * x + b_current) - y) # 根据公式  $b\_grade = 2(wx + b - y)$ 
```

```
        w_gradient+=(2/M) * x * ((w_current * x + b_current) - y) # 根据公式  $w\_grade = 2(wx + b - y) * x$ 
```

```
    new_b = b_current - (lr * b_gradient)
```

```
    new_w = w_current - (lr * w_gradient)
```

```
    return [new_b, new_w]
```

更新梯度

```
#更新梯度
def gradient_descent(data, star_b, star_w, lr, num_iterations):
    b=star_b
    w=star_w
    for step in range(num_iterations):
        b,w=step_gradient(b,w,data, lr)
        loss=mse(b,w, data)
        if step%5 ==0:
            print(f"iterations:{step}, loss: {loss}, w: {w}, b:{b}")
    return[b, w]

# 主函数
def main():
    lr=0.01
    init_b=0
    init_w=0
    num_iterations=1000

    [b, w]=gradient_descent(data, init_b, init_w, lr, num_iterations)
    loss= mse(b, w, data)
    print(f"Final Loss:{loss}, w: {w}, b:{b}")
```

本章结束

课后作业

- 使用以下商品房销售记录表数据，用梯度下降法，编程实现一个房价预测系统。

商品房销售记录

序号	面积 (平方米)	房间数	销售价格 (万元)	序号	面积 (平方米)	房间数	销售价格 (万元)
1	137.97	3	145.00	9	106.69	2	62.00
2	104.50	2	110.00	10	138.05	3	133.00
3	100.00	2	93.00	11	53.75	1	51.00
4	124.32	3	116.00	12	46.91	1	45.00
5	79.20	1	65.32	13	68.00	1	78.50
6	99.00	2	104.00	14	63.02	1	69.65
7	124.00	3	118.00	15	81.26	2	75.69
8	114.00	2	91.00	16	86.21	2	95.30
x_1			x_2	y			
x_1			x_2	y			