CSCI 3320 机器学习

> Fundamentals of Machine Learning

①②③④⑤⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺㊻㊼㊽㊾㊿

https://www.cse.cuhk.edu.hk/~cslui/csci3320.html

threshold	临界点
marble	弹珠
union bound	并集上界
generalization bound	泛化误差界
dichotomy	二分

Lecture 0 课程大纲

Assessment: 50% Homework + 50% Final exam

> 5 个 Homework，每个 10%

Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)

> 这本书找不到资源

Machine Learning in Action

> 下载上限

Email: stlu@cse.cuhk.edu.hk

Lecture 1 感知器

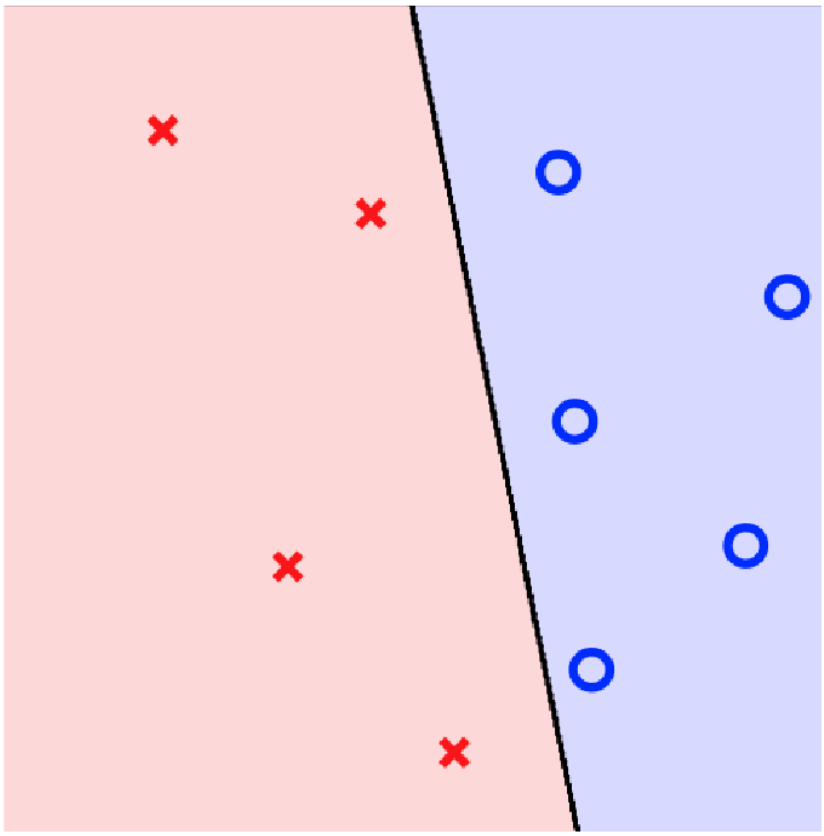
1.1 机器学习的类型

> Types of Machine Learning

① 二分类

> Binary Classification Problems

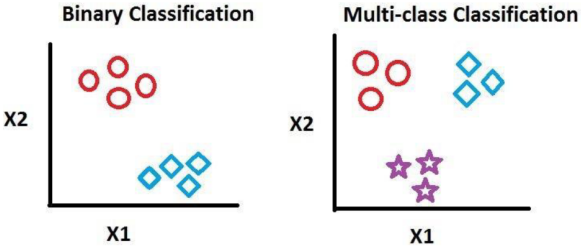
core and important as tools for other machine learning tasks.

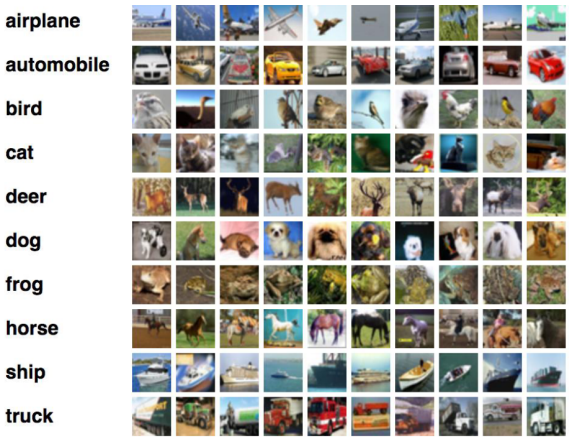


② 多分类

> Multiclass Classification

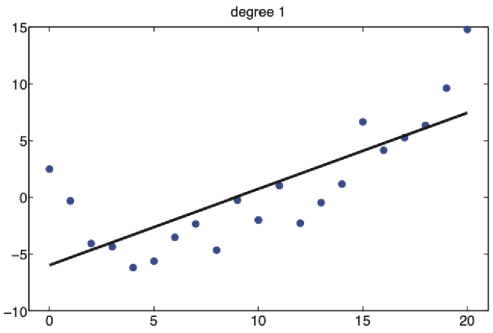
many applications in practice, especially for recognition.

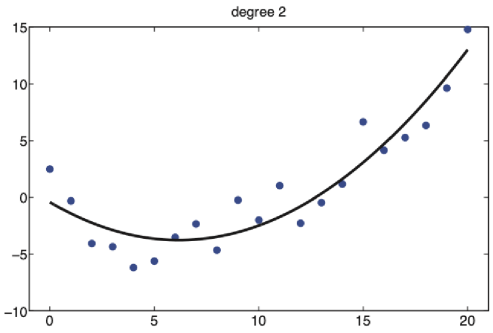




③ 回归

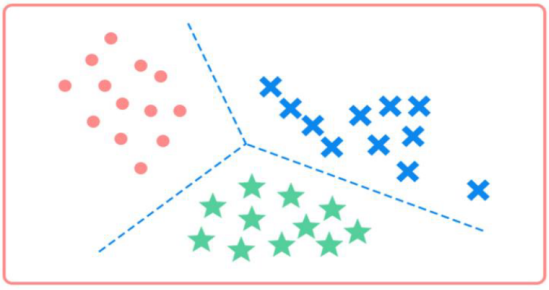
> Regressions: to predict a continuous outcome variable



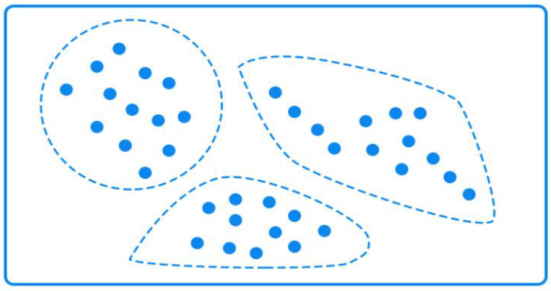


④ 无监督学习

> Unsupervised Learning: Categorizing data points
>
> 与监督学习不同，无监督学习不依赖于预先标注的输出标签.
>
> 无监督学习通过分析未标注数据的分布和结构，自动发现数据中的潜在模式、聚类或降维表示. 它通常用于探索性数据分析和特征提取.



Supervised Learning



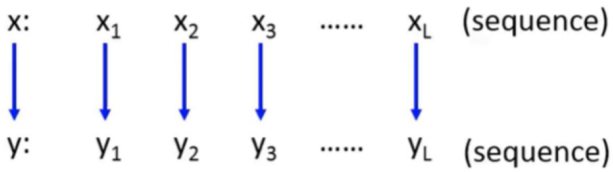
Unsupervised Learning

⑤ 半监督学习

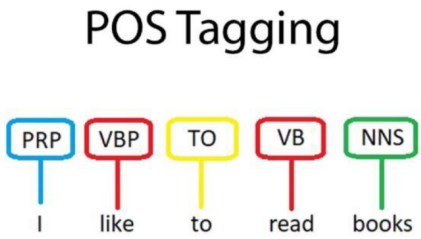
> Semi-supervised Learning: leverage unlabeled data to avoid expensive labelling
>
> 半监督学习通过结合少量标注数据和大量未标注数据，提升模型性能.

⑥ 结构化学习

> Structured Learning
>
> 与传统的分类或回归任务不同，结构化学习的输出空间通常是高维且具有内部依赖关系的.



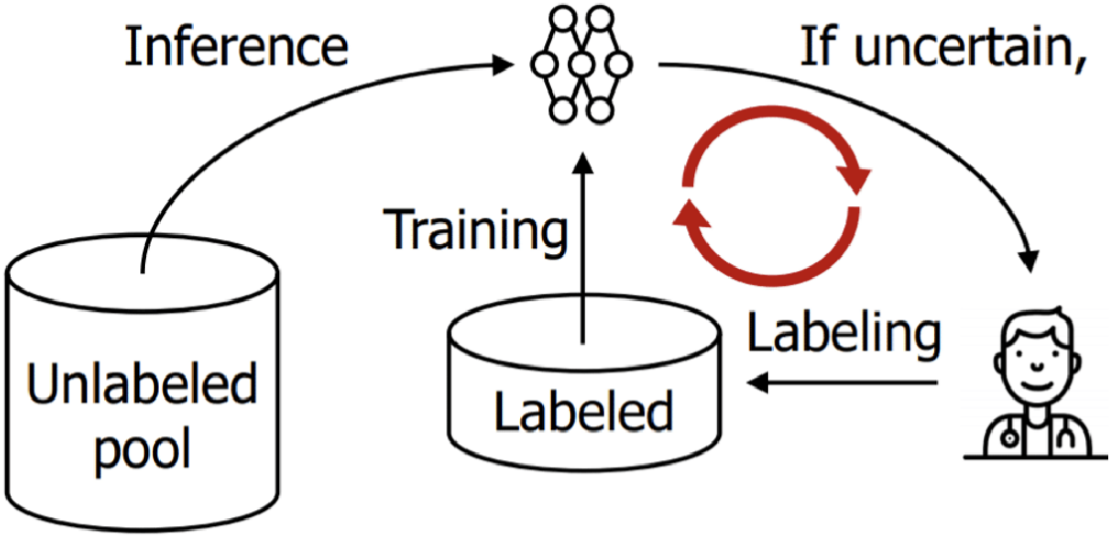
Protein folding



Part-of-speech

⑦ 主动学习

> Active Learning
>
> 主动学习通过迭代选择对模型最有帮助的样本进行标注，而不是随机选择样本. 它假设模型可以通过主动选择样本，更快地学习到数据的分布和边界.



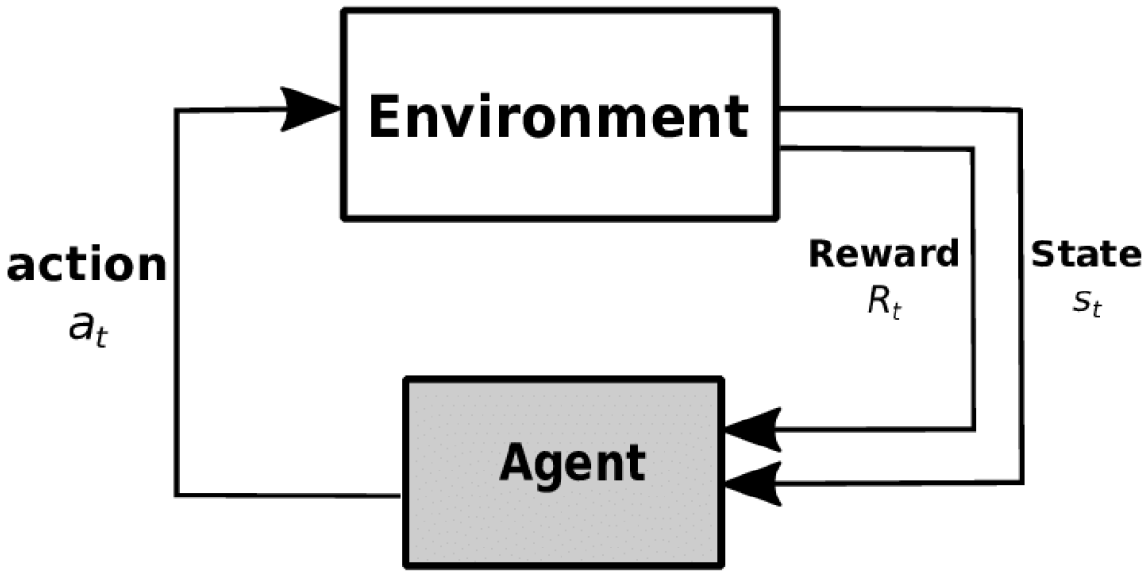
⑧ 强化学习

> Reinforcement Learning
>
> 强化学习的核心是智能体（Agent）在环境（Environment）中通过试错学习. 智能体根据当前状态（State）选择动作（Action），环境反馈奖励（Reward）并转移到新状态. 目标是学习一个策略，使长期累积奖励最大化.

a "very different" but natural way of learning

What if we have X, and we don't have the exact labels of Y?

We have Rewards!



1.2 信用卡问题

> Credit Approval Problem，经典的二分类问题.

1.2.1 问题描述

Input

假设用户有 d 个维度的因素供考虑，是否允许其使用信用卡：

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \in \mathbf{X}$$

> 这些权重可以是 salary, debt, years in residence, etc.
>
> We also called $\mathbf{x}$ as the **features**（特征）！

Output

用 $+1$ 表示 Approve，-1 表示 Reject：

$$y \in \{-1, +1\} = Y$$

Target Function

$$f : \mathbf{X} \to Y$$

> ideal credit approval formula
>
> unknown pattern to be learned
>
> 向量空间映射到标量空间.

没有固定公式判断是否允许用户使用信用卡，but Banks have data to learn it.

> 即 f 没有解析形式.

Training Data

$$D = \{(\mathbf{x_1}, y_1),\ \ldots,\ (\mathbf{x_N}, y_N)\}$$

> data in the bank

Hypothesis

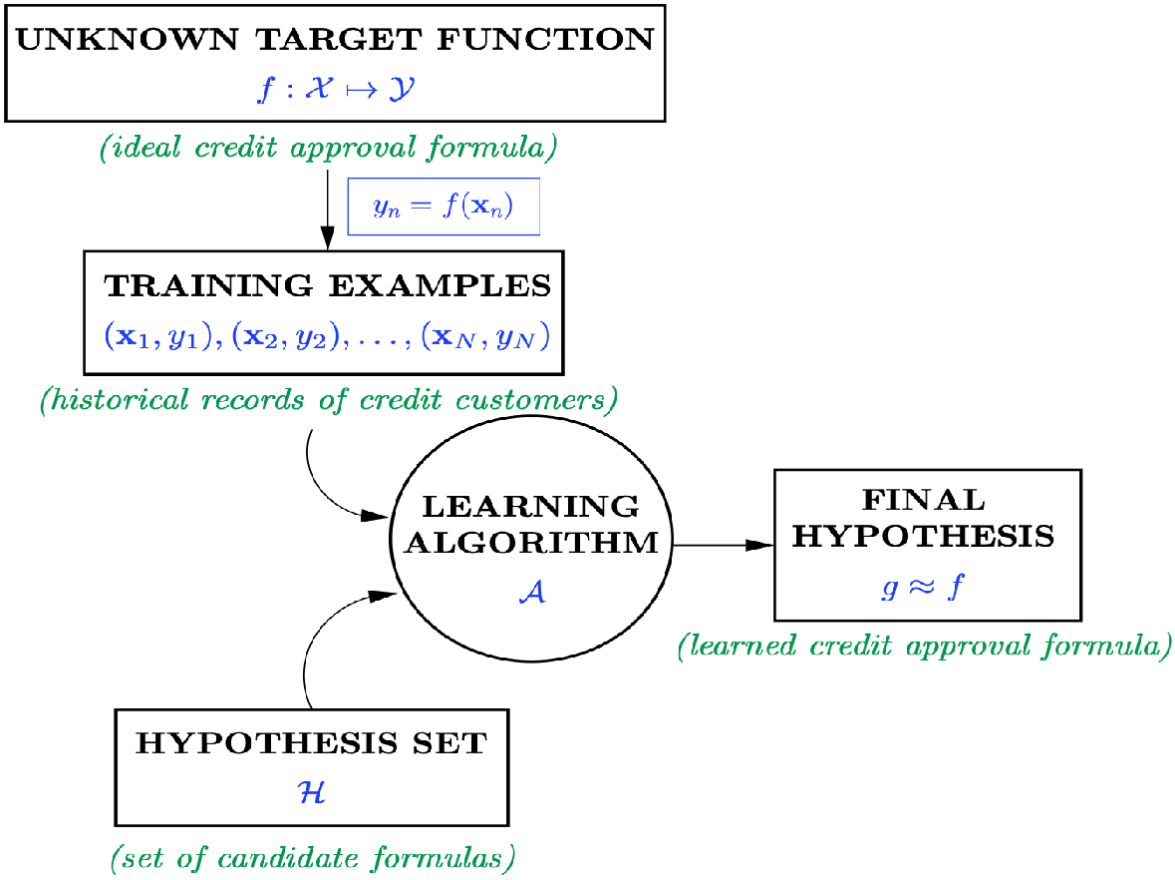
$$g : \mathbf{X} \to Y$$

> learned formula to be used
>
> 越接近 f 越好.

$$sign(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases}$$

> 忽略 $x = 0$ 的情况.

Machine Learning: use data to compute hypothesis g that approximates target f.



Artificial Intelligence: compute something that shows intelligent behavior (e.g., human reasoning).

> Machine Learning is one possible route to realize AI.

Statistics: use data to make inference about an unknown process.

> Statistics have many useful tools for Machine Learning.

下面介绍一个简单的学习模型.

1.2.2 感知器模型

> perceptron 感知器

$$h(x) = sign(\sum_{i=1}^{d} w_i x_i - threshold)$$

我们要学习权重矩阵

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

和临界值 $threshold$.

这里 $\mathbf{w}$ 和 $threshold$ 都是需要学习的参数. 方便起见，通过给 $\mathbf{x}$ 和 $\mathbf{w}$ 拓展一个维度，把 $threshold$ 项合并到求和中：

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

令 $x_0 = 1$, $w_0 = -threshold$，可以直接判断点乘的正负.

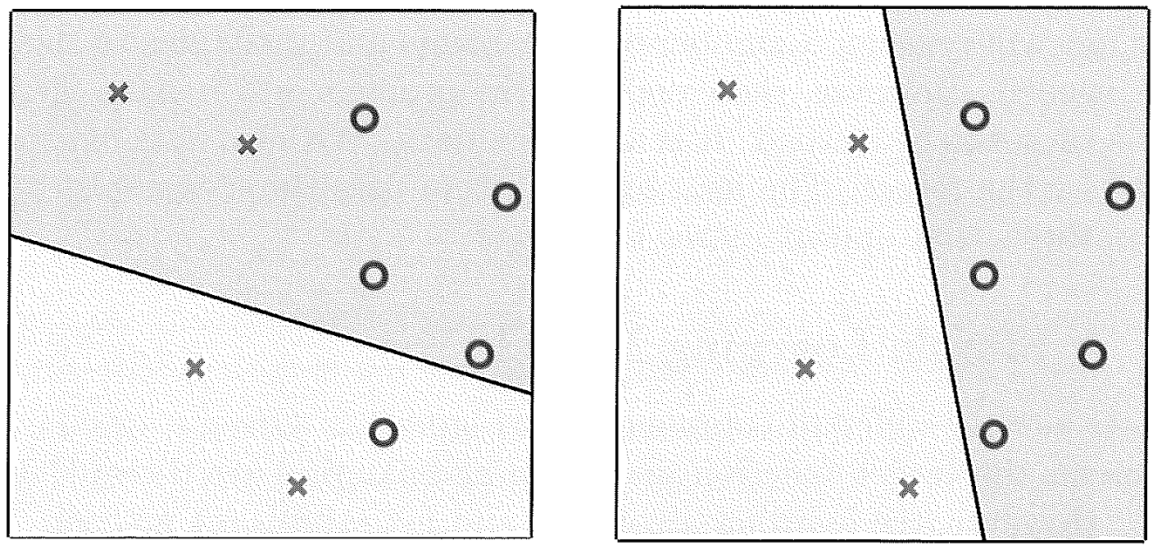
$$h(x) = sign(\mathbf{w}^T \mathbf{x})$$

> 关于向量内积，见 `大一 term 2 线性代数 6.1 ① 内积的定义` .

Each $\mathbf{w}$ represents a hypothesis h!

假设 $\mathbf{x}$ 除 x_0 外有两个维度，$h(x) = sign(w_0 + w_1 x_1 + w_2 x_2)$

$w_0 + w_1 x_1 + w_2 x_2 = 0$ 是分界线，且它恰好是 x_1, x_2 平面的一条直线. 因此可以实现可视化：



(a) Misclassified data　　(b) Perfectly classified data

This Hypothesis set is also called the Perceptron or Linear Separator.

H = all possible perceptrons, so how to determine/select g?

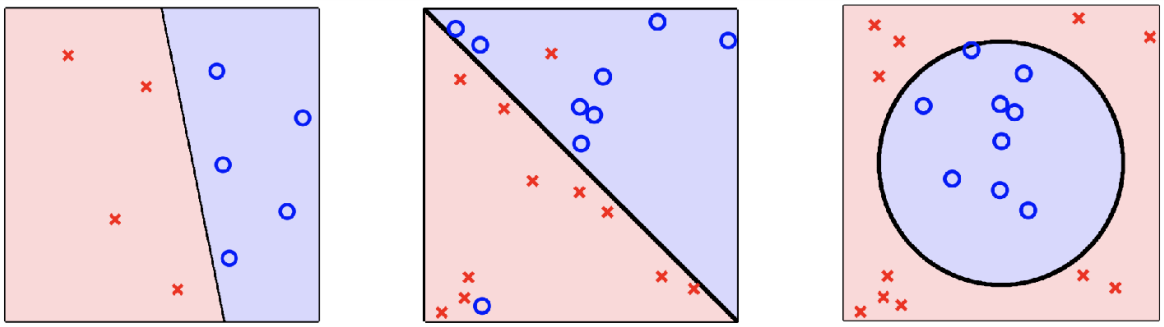
- Want: $g \approx f$ (hard when f unknown)
- Almost necessary: $g \approx f$ on D, ideally $g(\mathbf{x_n}) = f(\mathbf{x_n}) = y_n$

> 理想情况下，至少在训练集上实现 $g \approx f$

- Difficult: H is of infinite size
- Idea: start from some g_0, and 'correct' its mistakes on D

Linear Separable

> 线性可分的数据集



(linear separable) (not linear separable) (not linear separable)

$$Linear\ separable\ D \Leftrightarrow exist\ perfect\ \mathbf{w_f}\ such\ that$$

$$y_n = sign({\mathbf{w_f}}^T \mathbf{x_n})$$

for each $(\mathbf{x_n}, y_n) \in D$.

感知器学习算法

> The Perceptron Learning Algorithm (PLA)

初始化：

$$\mathbf{w}(1) = \mathbf{0}$$

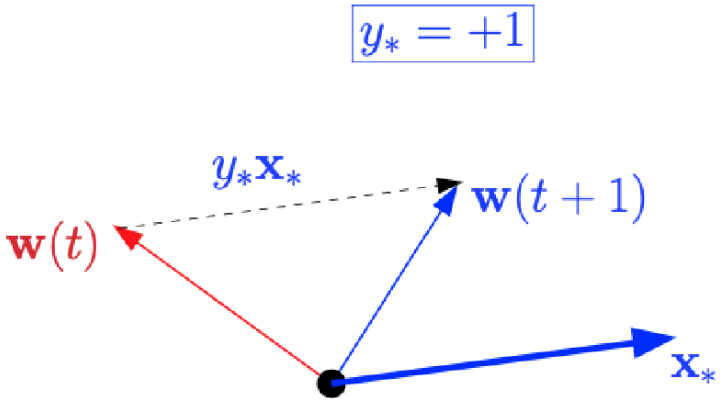
第 t 轮迭代，选择一对误分类的数据 $(\mathbf{x}_*, y_*) \in D$：

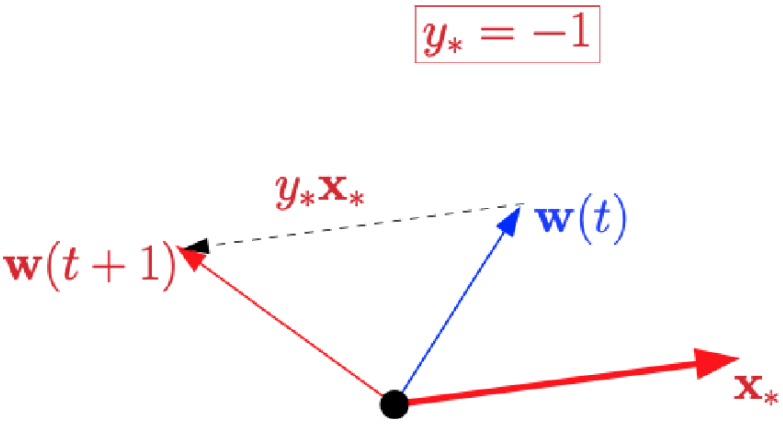
$$sign(\mathbf{w}(t) \cdot \mathbf{x}_*) \neq y_*$$

更新权重：

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y_* \mathbf{x}_*$$

> 这个公式对 $y_* = +1$ 和 $y_* = -1$ 都适用.
>
> 可视化：$\mathbf{w}$ 矢量旋转，使 $\mathbf{x}$ 在 $\mathbf{w}$ 上的投影趋向正确选择（正或负）.
>
> 一轮迭代无法保证误分类的数据直接变成正确分类，所以需要多轮迭代.





Return the final $\mathbf{w}$ as the g.

Theorem: if the data can be fit by a linear separator, then after some finite number of steps, the Perceptron Learning Algorithm will find one.

> 感知器收敛定理：如果训练数据是线性可分的，感知器算法可以在有限步数内收敛，即找到一组使得所有样本正确分类的权值. 如果数据线性不可分，算法不会收敛，此时可能需要其他模型（如多层感知器或支持向量机）.

证明:

前提条件与定义

① 训练样本线性可分

存在权重向量 $\mathbf{w_*}$，对于任意样本 $(\mathbf{x_i}, y_i) \in D$，有

$$y_i({\mathbf{w_*}}^T\mathbf{x_i}) > 0$$

> 同正同负，即所有样本分类正确.

② 归一化

为简化证明，归一化最终的权重向量，即 $||\mathbf{w_*}|| = 1$.

> 归一化对它的分辨能力没有影响.

③ 定义间隔 γ

定义所有样本到分隔超平面的最小距离为

$$\gamma = \min \frac{y_i({\mathbf{w_*}}^T\mathbf{x_i})}{||\mathbf{w_*}||} = \min y_i({\mathbf{w_*}}^T\mathbf{x_i})$$

> y_i 用于控制符号，距离恒为正，统一表达便于分析.
>
> 不影响距离的大小，因为 y_i 绝对值为 1.

④ 定义样本范数的上界 R

假设所有样本 $\mathbf{x_i}$ 满足

$$||\mathbf{x_i}|| \leq R$$

⑤ 感知器更新规则

初始权重 $\mathbf{w_1} = \mathbf{0}$.

假设在第 t 次迭代中，样本 $(\mathbf{x_i}, y_i)$ 被错误分类，即

$$y_i({\mathbf{w_t}}^T\mathbf{x_i}) \leq 0$$

则进行更新：

$$\mathbf{w_{t+1}} = \mathbf{w_t} + y_i\mathbf{x_i}$$

> 注意，我们只选择错误分类的样本进行迭代.

证明步骤

考虑第 t 次迭代后，当前感知器 $\mathbf{w_{t+1}}$ 在最终感知器 $\mathbf{w_*}$ 上的投影 ${\mathbf{w_{t+1}}}^T\mathbf{w_*}$.

下界：

$$\begin{aligned}{\mathbf{w_{t+1}}}^T\mathbf{w_*} = {\mathbf{w_t}}^T\mathbf{w_*} + y_i{\mathbf{x_i}}^T\mathbf{w_*} &\geq {\mathbf{w_t}}^T\mathbf{w_*} + \gamma \\ &\geq \cdots \\ &\geq t\gamma\end{aligned}$$

> 注意 ${\mathbf{x_i}}^T\mathbf{w_*} = {\mathbf{w_*}}^T\mathbf{x_i}$.

上界：

$$
\begin{aligned}
\mathbf{w_{t+1}}^T\mathbf{w}_* &\leq ||\mathbf{w_{t+1}}||||\mathbf{w}_*|| \\
&= ||\mathbf{w_{t+1}}|| \\
&= \sqrt{||\mathbf{w_t} + y_i\mathbf{x_i}||^2} \\
&= \sqrt{||\mathbf{w_t}||^2 + 2y_i\mathbf{w_t}^T\mathbf{x_i} + R^2} \\
&\leq \sqrt{||\mathbf{w_t}||^2 + R^2} \\
&\leq \cdots \\
&\leq \sqrt{t}R
\end{aligned}
$$

结合上下界，$t\gamma \leq \mathbf{w_{t+1}}^T\mathbf{w}_* \leq \sqrt{t}R \Rightarrow t \leq (\frac{R}{\gamma})^2$

注意，迭代轮数可以超过样本个数，因为对于某些样本，仅一次更新可能不足以使感知器正确分辨它，需要多迭代几次.

$t \leq (\frac{R}{\gamma})^2$ 说明感知器算法的迭代轮数不会超过一个有限大的值 $(\frac{R}{\gamma})^2$，即在 $(\frac{R}{\gamma})^2$ 步内必将收敛（找到一个正确分类所有训练样本的权重向量 $\mathbf{w}_*$）.

> 注意，我们只选择错误分类的样本进行迭代.

Lecture 2 概统与泛化

2.1 数据集之外

> Outside the Data Set

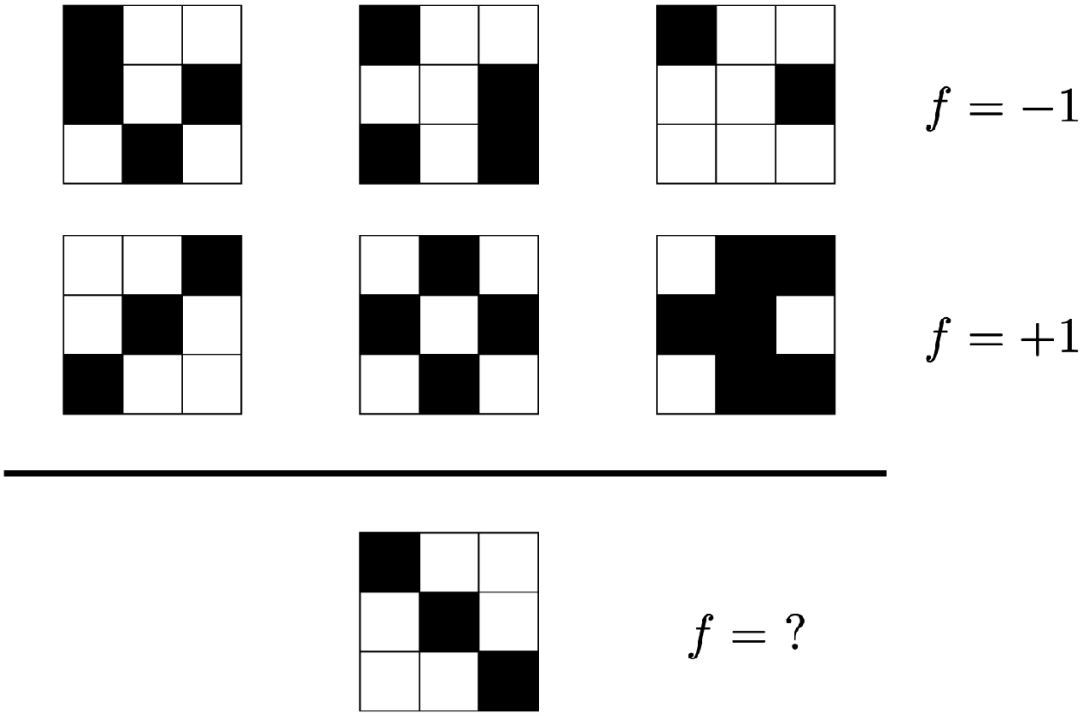
Does the data set D tell us anything outside?

- If YES, we have learned something!
- If NO, learning is not feasible!

The whole purpose of learning is to be able to predict the value of f on the points that we haven't seen before.

> 训练集是已知的，模型在训练集上表现再好都是次要的，重要的是能否正确预测未知样本.

The performance outside D is all that matters in learning!



Did you say $f = 1$?

> f is measuring symmetry

Did you say $f = -1$?

> f only cares about the top left pixel

这个例子说明，同一个训练集可以训练出不同模型，并且都在训练集中表现良好，但对于未知样本的表现可能截然不同.

$\mathcal{D}$

x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	o	o	o	o	o	o	o	o	o	o
0 0 1	×	×	×	×	×	×	×	×	×	×
0 1 0	×	×	×	×	×	×	×	×	×	×
0 1 1	o	o	o	o	o	o	o	o	o	o
1 0 0	×	×	×	×	×	×	×	×	×	×
1 0 1		?	o	o	o	o	×	×	×	×
1 1 0		?	o	o	×	×	o	o	×	×
1 1 1		?	o	×	o	×	o	×	o	×

$g \approx f$

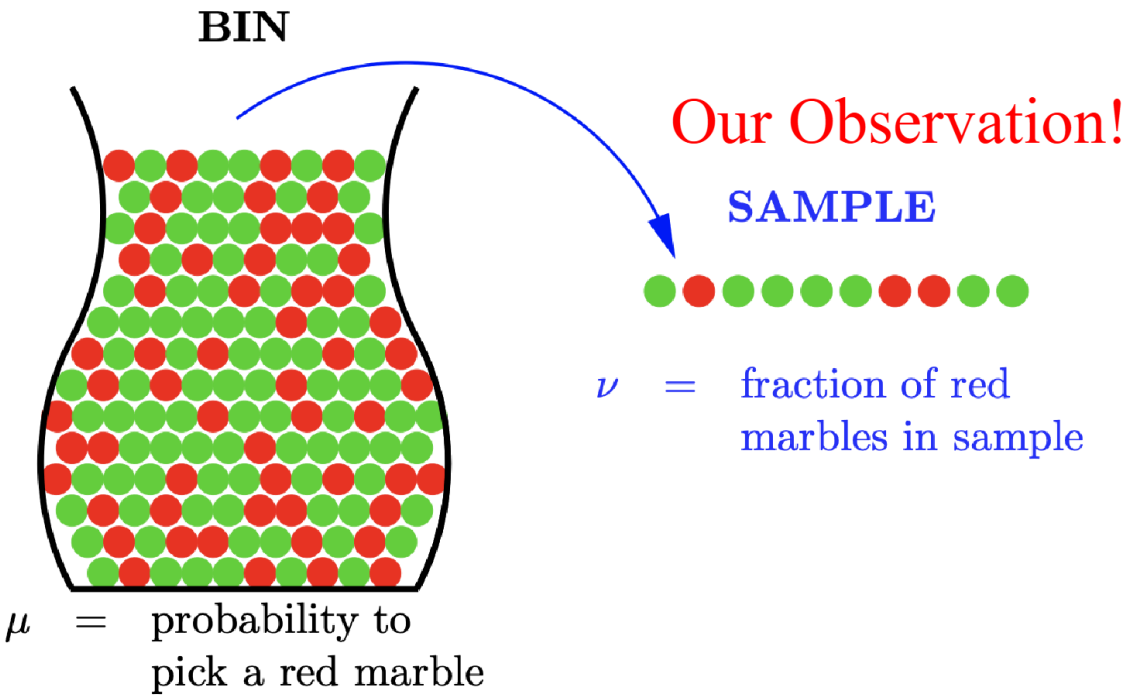
$g \approx f$?

看起来仅用训练数据无法获取任何未知样本的信息. 机器学习难道和直接标记的效果一样?

Fortunately, learning is alive and well. We can infer something outside D using only D, but in a **probabilistic** way. We will start with the simplest case of picking a sample.

2.2 BIN 模型

> The BIN Model
>
> 2025.1.15



> 样本占比估计总体概率.

The BIN Model

- Bin with red and green marbles.
- Pick a sample of N marbles independently.

μ: probability to pick a red marble.

ν: fraction of red marbles in the sample.

Sample $\rightarrow$ the data set $\rightarrow \nu$

BIN $\rightarrow$ outside the data $\rightarrow \mu$

Question: can we say anything about μ (outside the data) after observing ν?

Answer: we can get mostly green marbles in the sample, while the bin has mostly red marbles.

> possible, but by no means probable.
>
> 极端情况虽然可能发生，但概率很低.

Then, what does the value of ν tell us about the value of μ?

2.2.1 概率近似正确

> Probably Approximately Correct

PAC，机器学习重要概念. 由计算机科学家 Leslie Valiant 在 1984 年提出，描述机器学习算法的能力.

核心思想：算法能否在有限训练样本和计算资源下，学到一个接近正确的模型?

① 基本概念

1. Probably (高概率)

   算法以「高概率」学到一个接近正确的模型.

2. Approximately (近似正确)

   算法学习的模型误差是「可接受的」，即小于某个误差阈值 ϵ.

3. Concept Class (概念类)

   PAC 理论假设存在一个目标函数（或目标概念），这个目标函数从数据中生成正确标签. 学习算法的目标是从一个概念类（可能的假设集合）中找到一个接近目标函数的假设.

4. Sample Complexity (样本复杂度)

   学习算法需要多少样本，才能以高概率学到一个接近正确的模型.

5. Computational Complexity (计算复杂度)

   算法在计算上是否可行.

② 理论框架

在 PAC 中，假设数据是从未知的分布 D 中独立同分布 (i.i.d.) 采样的. 目标是学习一个假设函数 h，使得它的误分类率尽可能接近目标函数 f 的误分类率.

具体来说：

- 误分类率 (error) 指的是在分布 D 上，假设函数 h 和目标函数 f 预测不一致的概率：
  $error(h) = P_{x \sim D}[h(x) \neq f(x)]$
- PAC 学习要求：
  1. 学习算法在有限样本和计算时间内，能够输出一个假设函数 h.
  2. 这个假设函数的误分类率满足以下条件：
     $error(h) \leq \epsilon$
     其中，ϵ 是一个小的误差阈值.
  3. 以上条件以高概率（至少 $1 - \delta$）成立，其中 δ 是一个小概率.

PAC 目标：为达到上述要求，学习算法需要多少样本（样本复杂度）以及需要多少计算资源（计算复杂度）.

③ 例子

假设我们想学习一个简单的二分类模型：判断某个水果是「苹果」还是「橙子」. 目标是通过有限样本训练出一个分类器，能以高概率正确分类新水果.

1. **Probably**: 我们希望分类器的表现是「高概率」好的，比如正确分类率达到 95% 或更高.
2. **Approximately**: 我们允许分类器有一定误差，比如最多允许 5% 的分类错误.
3. **Sample Complexity**: PAC 理论可以帮助我们计算，训练分类器需要多少样本才能保证上述目标.
4. **Computational Complexity**: PAC 理论还可以分析分类器的训练算法是否计算上可行.

④ 意义

1. **解释学习可行性**：PAC 理论提供了一个数学框架，用于判断某个学习问题是否在有限样本和计算资源下是可解的.
2. **样本复杂度分析**：它可以帮助我们估计为保证学习效果需要的最小样本量.
3. **算法设计指导**：PAC 理论可以指导我们设计高效的学习算法.

⑤ 扩展

PAC 和现代深度学习有些距离，但它提供了重要基础. 现代机器学习有许多基于 PAC 的扩展概念，例如：

- **PAC-Bayes 理论**：结合 PAC 和贝叶斯推断，用于分析复杂模型的泛化能力.
- **弱 PAC 学习与强 PAC 学习**：描述学习算法在不同误差和概率条件下的表现.

2.2.2 霍夫丁不等式

> Hoeffding's Inequality

Hoeffding's 不等式是概率论中的一种重要的集中不等式，它为一组相互独立且有界的随机变量的和（或平均值）偏离其期望值的概率提供了一个上界.

① 数学表述

设随机变量 X_1，X_2，…，X_n 相互独立，且对于每个 k，满足

$$X_k \in [a_k, b_k]$$

则对于任意正数 t，有以下不等式成立：

$$P\left(\sum_{k=1}^{n}(X_k - E[X_k]) \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k - a_k)^2}\right)$$

同理，对负偏差也成立：

$$P\left(\sum_{k=1}^{n}(E[X_k] - X_k) \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k - a_k)^2}\right)$$

通常写成对称形式，描述随机变量和其期望的偏离：

$$P\left(\left|\sum_{k=1}^{n}(X_k - E[X_k])\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k - a_k)^2}\right)$$

② 证明

引入偏差变量和目标

设 X_1，X_2，…，X_n 是相互独立的随机变量，每个 X_k 满足 $X_k \in [a_k, b_k]$. 令 $Y_k = X_k - E[X_k]$，则有 $E[Y_k] = 0$，同时 Y_k 的取值区间为 $Y_k \in [a_k - E[X_k], b_k - E[X_k]]$，因此其宽度不大于 $(b_k - a_k)$. 我们的目标是证明对于 $t > 0$，有

$$P\left(\sum_{k=1}^{n} Y_k \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k - a_k)^2}\right)$$

利用 Chernoff 技巧（指数鞅方法）

对于任意 $\lambda > 0$，根据 Markov 不等式，有

$$\begin{aligned} P\left(\sum_{k=1}^{n} Y_k \geq t\right) &= P\left(\exp\left(\lambda\sum_{k=1}^{n} Y_k\right) \geq \exp(\lambda t)\right) \\ &\leq \frac{E[\exp\left(\lambda\sum_{k=1}^{n} Y_k\right)]}{\exp(\lambda t)} \\ &= \exp(-\lambda t)\prod_{k=1}^{n} E[e^{\lambda Y_k}] \end{aligned}$$

由于随机变量相互独立，指数函数的期望可以分解成各自的乘积.

使用 Hoeffding 引理对每个 Y_k 的矩生成函数进行界定

> Hoeffding's Lemma，见 2.2.2 霍夫丁不等式 © 霍夫丁引理 .

随机变量 Y_k 满足 $E[Y_k]=0$ 且 $Y_k \in [a_k - E[X_k], b_k - E[X_k]]$，有

$$E[e^{\lambda Y_k}] \leq \exp\left(\frac{\lambda^2 (b_k-a_k)^2}{8}\right)$$

$$\prod_{k=1}^{n} E[e^{\lambda Y_k}] \leq \exp\left(\frac{\lambda^2}{8}\sum_{k=1}^{n}(b_k-a_k)^2\right)$$

合并不等式并选择最佳参数 λ

将上一步结果代入第二步的不等式：

$$P\left(\sum_{k=1}^{n} Y_k \geq t\right) \leq \exp\left(-\lambda t + \frac{\lambda^2}{8}\sum_{k=1}^{n}(b_k-a_k)^2\right)$$

令

$$f(\lambda) = -\lambda t + \frac{\lambda^2}{8}\sum_{k=1}^{n}(b_k-a_k)^2$$

这是一个开口朝上的二次函数，当

$$\lambda = -\frac{b}{2a} = \frac{4t}{\sum\limits_{k=1}^{n}(b_k-a_k)^2}$$

时，$f(\lambda)$ 取最小值

$$\frac{4ac-b^2}{4a} = -\frac{2t^2}{\sum\limits_{k=1}^{n}(b_k-a_k)^2}$$

于是得到

$$P\left(\sum_{k=1}^{n} Y_k \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k-a_k)^2}\right)$$

对称形式

同样方法可以证明相反方向的偏差，即

$$P\left(\sum_{k=1}^{n} Y_k \leq -t\right) \leq \exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k-a_k)^2}\right)$$

因此，对绝对偏差有：

$$P\left(\left|\sum_{k=1}^{n}(X_k - E[X_k])\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k-a_k)^2}\right)$$

证毕.

③ 意义

- 偏差的指数衰减

Hoeffding's 不等式说明，n 个独立随机变量的和偏离期望的概率是指数级减少的. 这意味着即使是较小的偏差，发生概率也非常低.

- 使用条件宽泛

此不等式只要求随机变量有界，而不需要假设它们服从具体的概率分布. 这使得它在很多场合（例如在未知分布情况下）都能有效使用.

④ 应用

Hoeffding's 不等式在多个领域发挥重要作用，例如：

- **统计学和概率论：** 用于证明大数定律的定量版本，描述样本均值如何快速收敛到期望值附近.
- **机器学习：** 在学习理论中，它被用来证明经验风险最小化（Empirical Risk Minimization, ERM）的泛化能力，从而说明训练误差和真实误差之间的差距是可控的.
- **算法理论：** 在随机算法分析中，Hoeffding's 不等式能帮助估计算法输出与其期望之间的偏差概率，从而评估算法稳定性和准确性.

⑤ 霍夫丁 vs 切比雪夫

> 关于 Chebyshev's Inequality，见 `ESTR 2018 概率论 13.6 切比雪夫不等式`.

切比雪夫不等式依靠随机变量的方差给出偏离均值的概率上界.

Suppose $X_1, X_2, \ldots, X_n$ are **IID** with mean $\mathbb{E}[X_i] = \mu$ and variance $Var[X_i] = \sigma^2$. Then, **Chebyshev's inequality** shows

$$\mathbb{P}\left(|\frac{X_1 + \cdots + X_n}{n} - \mu| \geq \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2}$$

这种界是基于二阶矩的，适用于任意具有有限方差的独立同分布随机变量，不要求取值范围有界.

> 这里的二阶矩指中心二阶矩，即方差. 基于二阶矩取到的上界较宽松，因为它只能反应分布的「整体波动性」，而无法捕捉更细致的结构；它也无法充分反映随机变量取值的界限，得到的概率上界不够紧.

霍夫丁不等式是针对**有界**随机变量的一种集中不等式. 设每个 X_i 都满足 $X_i \in [a_i, b_i]$，并令

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

当随机变量独立时，对于任意 $\epsilon > 0$ 有

$$\mathbb{P}(|\overline{X}-\mu|\geq\epsilon)\leq 2\exp\left(-\frac{2n^2\epsilon^2}{\sum_{i=1}^n(b_i-a_i)^2}\right)$$

如果所有 X_i 都取值在 $[0,1]$ 内，则

$$\mathbb{P}(|\overline{X}-\mu|\geq\epsilon)\leq 2e^{-2n\epsilon^2}$$

这个结果给出了**指数衰减**的概率上界.

比较

- **应用条件：**
  - 切比雪夫不等式仅要求随机变量具有有限均值和方差，并不需要知道其取值范围.
  - 霍夫丁不等式则要求随机变量取值范围已知，即每个 X_i 有确定的上下界.
- **概率衰减的速度：**
  - 切比雪夫不等式给出的上界是 $\frac{\sigma^2}{n\epsilon^2}$，随着 n 增大，衰减速度是多项式级别，大致是 $O(\frac{1}{n})$.
  - 霍夫丁不等式在 $[0,1]$ 情况下，给出的上界是 $2e^{-2n\epsilon^2}$，衰减是指数级的，因此对于较大的 n 来说，概率上界远远小于切比雪夫不等式给出的值.
- **定量信息：**
  - 切比雪夫不等式提供的是一个较为宽松的界，因为它只利用了方差信息.
  - 霍夫丁不等式利用了随机变量的有界性，因而在同样的条件下能够得到更为紧致的界限，这一点在有限样本分析中尤为重要.

在有界随机变量场景下，霍夫丁不等式更强、更细致.

⑥ 霍夫丁引理

> Hoeffding's Lemma

设随机变量 X 满足 $X\in[a,b]$ 且 $E[X]=0$，则对于任意 $\lambda\in\mathbb{R}$，有

$$E[\exp(\lambda X)]\leq\exp\left(\lambda^2\frac{(b-a)^2}{8}\right)$$

证明：

定义对数母函数

$$\varphi(\lambda)=\ln E[\exp(\lambda X)]$$

注意到有

$$\varphi(0)=\log E[1]=0$$
$$\varphi'(0)=E[X]=0$$

计算二阶导数

$$
\begin{aligned}
\varphi''(\lambda) &= Var_{Q_\lambda}(X) \\
&= \frac{E[X^2 e^{\lambda X}]}{E[e^{\lambda X}]} - \left(\frac{E[X e^{\lambda X}]}{E[e^{\lambda X}]} \right)^2 \\
&\leq \frac{(b-a)^2}{4}
\end{aligned}
$$

其中 Q_λ 是指数倾斜（exponential tilt）的概率度量.

> 关于概率测度的知识，自行搜索. 在通过指数倾斜构造出的新概率测度 Q_λ 下，随机变量 X 的取值范围不变.
>
> 待证明.

利用积分不等式获得 $\varphi(\lambda)$ 的上界

> Taylor 展开积分余项形式

对于一个二阶可微函数 $\varphi(\lambda)$，在 $\lambda = 0$ 附近的 Taylor 展开可写作

$$
\varphi(\lambda) = \varphi(0) + \varphi'(0)\lambda + \int_0^\lambda (\lambda - t)\varphi''(t)dt
$$

> 待证明.

由于 $\varphi(0) = 0$ 且 $\varphi'(0) = 0$，化简后得到

$$
\begin{aligned}
\varphi(\lambda) &= \int_0^\lambda (\lambda - t)\varphi''(t)dt \\
&\leq \int_0^\lambda (\lambda - t)\frac{(b-a)^2}{4}dt \\
&= \frac{\lambda^2 (b-a)^2}{8}
\end{aligned}
$$

还原到原来的期望表达式

上式说明，

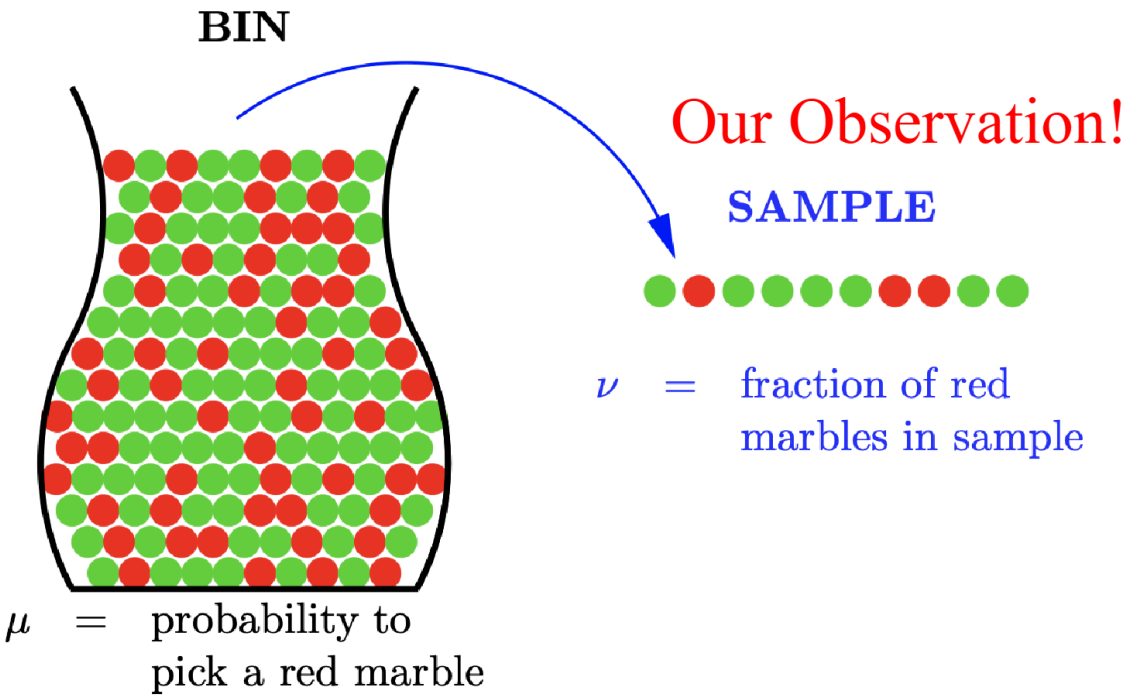
$$
\ln E[\exp(\lambda X)] \leq \frac{\lambda^2 (b-a)^2}{8}
$$

取指数后得到

$$
E[\exp(\lambda X)] \leq \exp\left(\lambda^2 \frac{(b-a)^2}{8} \right)
$$

证毕.

⑦ BIN 模型



回到 BIN 模型. 已知霍夫丁不等式

$$P\left(\left|\sum_{k=1}^{n}(X_k - E[X_k])\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{k=1}^{n}(b_k - a_k)^2}\right)$$

我们可以将每次抽球看作一次独立的伯努利试验，定义随机变量

$$X_i = \begin{cases} 1 & \text{如果第 } i \text{ 次抽到红球} \\ 0 & \text{如果第 } i \text{ 次抽到绿球} \end{cases}$$

那么每个 $X_i \in [0,1]$ 且 $E[X_i] = \mu$.

从盒中抽出 N 个球作为样本，其中红球总数为

$$S_N = X_1 + X_2 + \cdots + X_N$$

红球比例为

$$\nu = \frac{S_N}{N}$$

且 $E[S_N] = N\mu,\ E[\nu] = \mu$.

将不等式作用于比例 ν，得

$$\begin{aligned} \mathbb{P}\left(|\nu - \mu| > \epsilon\right) &= \mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)\right| > \epsilon\right) \\ &= \mathbb{P}\left(\left|\sum_{i=1}^{N}(X_i - \mu)\right| > N\epsilon\right) \\ &\leq 2\exp\left(-\frac{2(N\epsilon)^2}{N}\right) \\ &= 2\exp\left(-2N\epsilon^2\right) \end{aligned}$$

即

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$
$$\mathbb{P}[|\nu - \mu| \leq \epsilon] \geq 1 - 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$

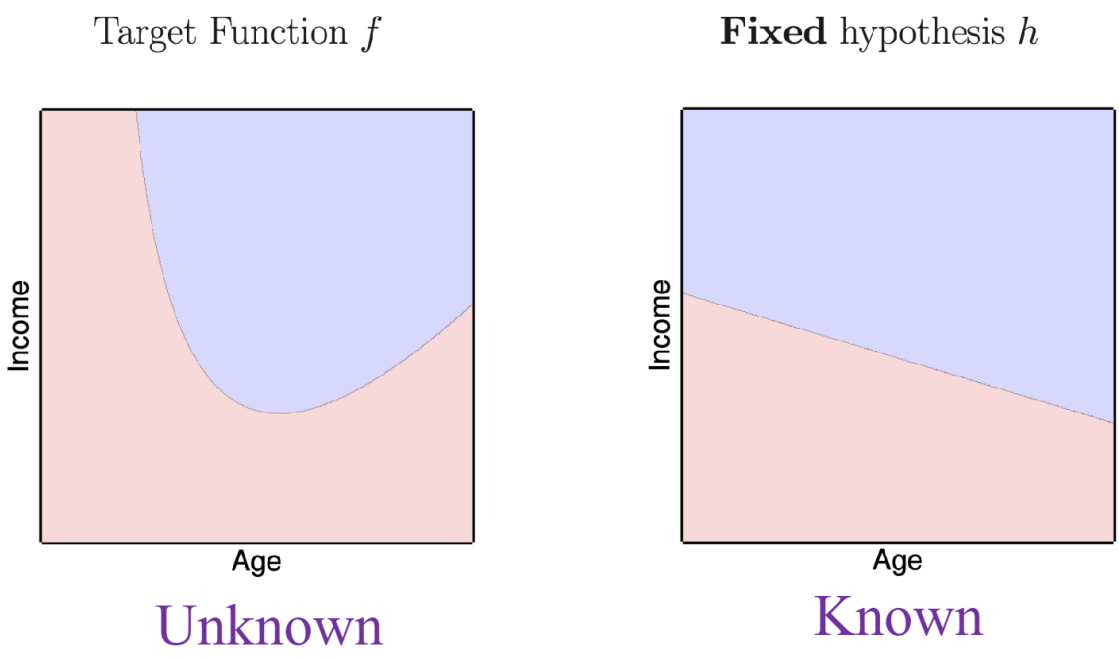
只要样本量够大，就能很好地估计整体.

- Valid for all N and ϵ
- Does not depend on μ, no need to know μ
- Larger sample size N or looser gap $\epsilon \Rightarrow$ higher probability for $\nu \approx \mu$
- If large $N (N \gg \frac{1}{\epsilon^2})$, we can probably infer unknown μ by known ν

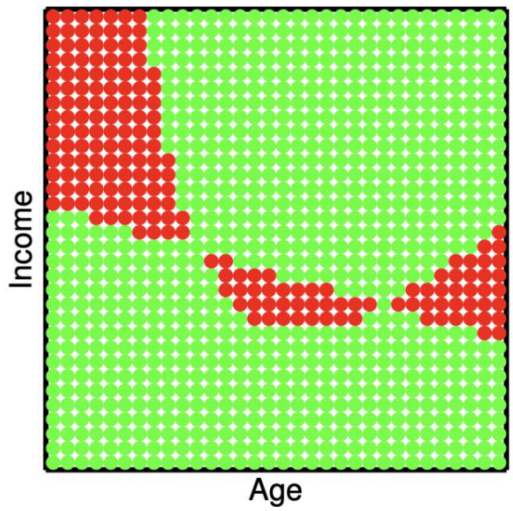
2.2.3 BIN 与机器学习

> Relating the Bin to Learning

① ML 类比 BIN



In learning, the unknown is an entire function f; in the bin, it was a single number μ.



$$\text{green “marble”: } h(\mathbf{x}) = f(\mathbf{x})$$
$$\text{red “marble”: } h(\mathbf{x}) \neq f(\mathbf{x})$$
$$\text{BIN: } \mathcal{X}$$

$$E_{\text{out}}(h) = \mathbb{P}_{\mathbf{x}}[h(\mathbf{x}) \neq f(\mathbf{x})]$$

out-of-sample

UNKNOWN

If we pick $\mathbf{x}$ at random according to some distribution P over the input space $\mathbf{X}$, $\mathbf{x}$ will be red with some probability, i.e., μ, and green with probability $1 - \mu$.

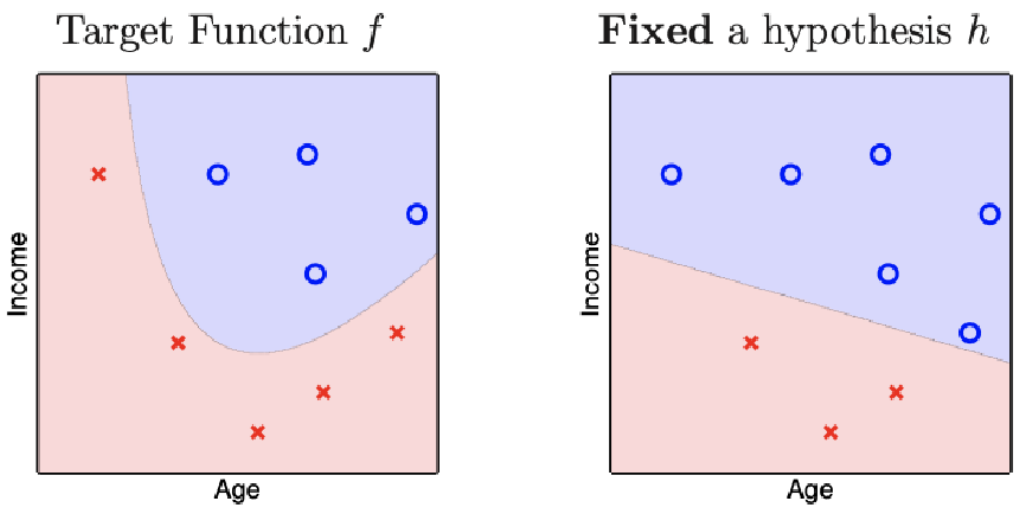
> 把泛化误差 $E_{out}(h)$ 与 BIN 模型中的 μ 联系起来.
>
> 这里 P 指采样的分布方式，例如在整个空间随机采样，还是在某些地方更集中. 当采用随机采样时，μ 才是红色面积占比.

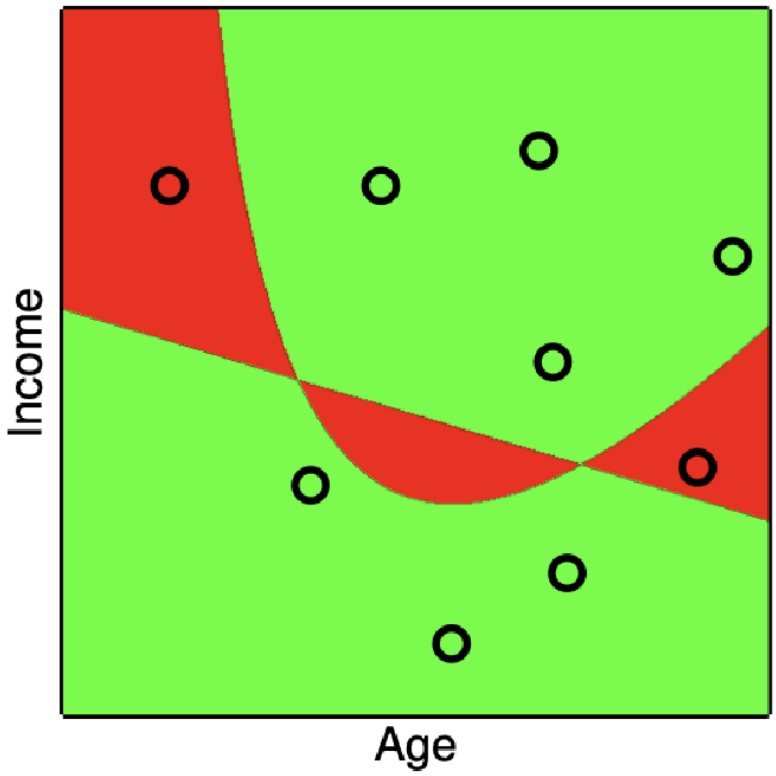
注意：泛化误差未知函数 $f(\mathbf{x})$ 不能积分，只能采样然后用 Bin 模型估计误差（误差即红色区域占比）

> 目的是往 $f(\mathbf{x})$ 靠近，$f(\mathbf{x})$ 本身是未知的.

例如，我们在 $\mathbf{X}$ 上随机采样 9 次，发现 h 和 f 的判断有 2 次不同.

> $\mathbf{X}$ 即整个矩形，有无数个点.





② 经验 & 泛化误差

经验误差 $E_{in}(h) = \text{fraction of red data in sample}$

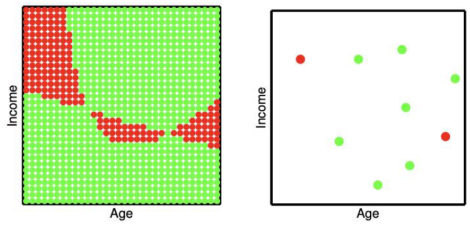
> 在训练数据上的误差，样本带来的信息. 红色区域内表示误分类.

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x_n}) \; != f(\mathbf{x_n}))$$

> 这里 $h(\mathbf{x_n}) \; != f(\mathbf{x_n})$ 是布尔表达式，true 返回 1，false 返回 0.
>
> 这个公式就是计算样本中误分类数据占比.

泛化误差 $E_{out}(h)$：未知.

> 在真实分布上的误差.



Unknown f and $P(\mathbf{x})$, **fixed** h

Learning

input space $\mathcal{X}$

$\mathbf{x}$ for which $h(\mathbf{x}) = f(\mathbf{x})$

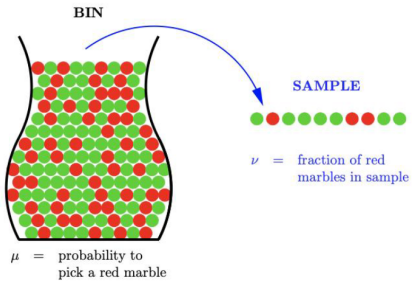
$\mathbf{x}$ for which $h(\mathbf{x}) \neq f(\mathbf{x})$

$P(\mathbf{x})$

data set $\mathcal{D}$

Out-of-sample Error: $E_{\text{out}}(h) = \mathbb{P}_{\mathbf{x}}[h(\mathbf{x}) \neq f(\mathbf{x})]$

In-sample Error: $E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$



Bin Model

Bin

green marble

red marble

randomly picking a marble

sample of N marbles

μ = probability of picking a red marble

ν = fraction of red marbles in the sample

这样就把 the learning problem 等价于 a bin problem:

the inputs in D are picked independently according to some distribution P on $\mathbf{X}$. Any P will translate to some μ in the equivalent bin.

> 就是把 learning problem 的训练集当作整体的抽样. 只要训练样本足够多，就能很好地泛化整体.

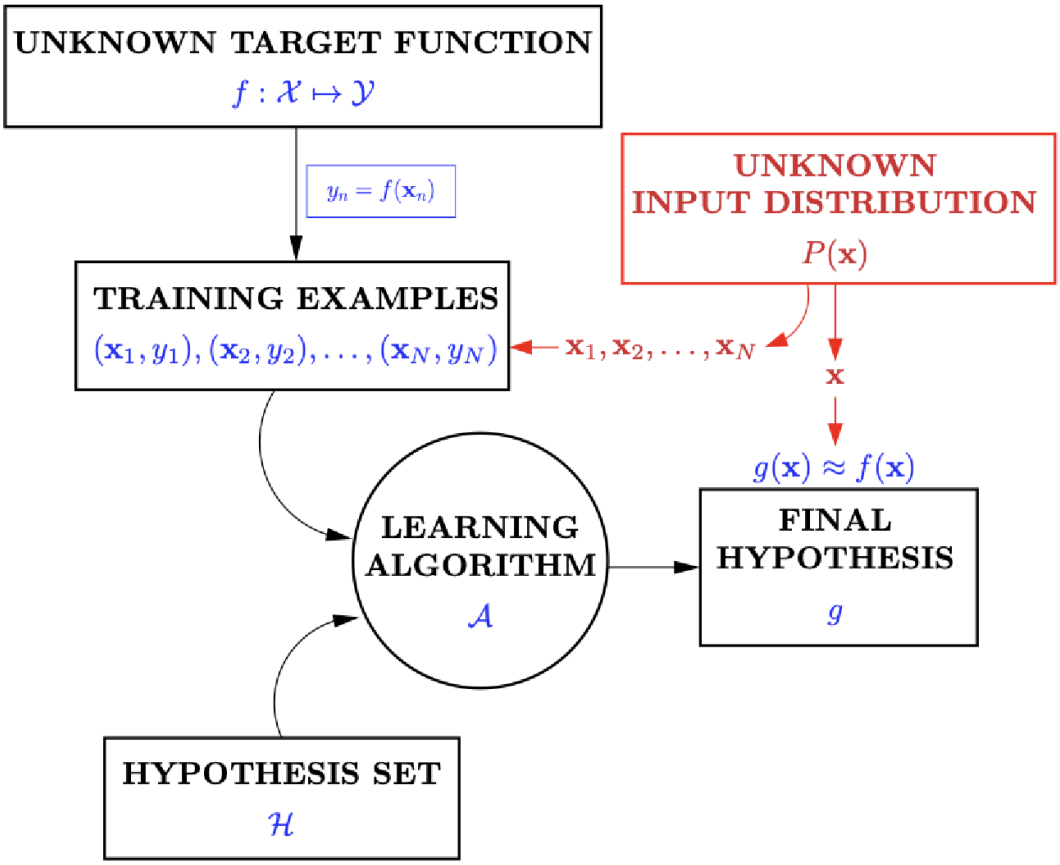
既然 BIN Model 可以用霍夫丁不等式来限制 ν 和 μ 的偏差，那么 learning problem 同样可以用霍夫丁不等式来限制经验误差和泛化误差的偏差，即用霍夫丁不等式求出需要多少样本才能使 E_{in} 和 E_{out} 的偏差在可接受范围内.

With this equivalence, the Hoeffding Inequality can be applied to the learning problem, allowing us to make a prediction outside of D.

> 注意，样本足够大只是保证了在这些样本上的效果足够估计在整体上的效果，本身并不能帮你找出一个优良的模型. 在样本上效果好的模型需要通过扩大 Hypothesis Set 碰运气碰出来（后面章节会有更多方法，目前只能这么做）.

③ 修正 ML 框架

Revising the Learning Problem: Adding in Probability



> 注意，这里除了分类标准 f 未知，连数据在空间中的分布 P 也未知，P 同样也要从有限的样本中近似估计. 但机器学习中是否知道 P 不太重要，只要有大量现实训练样本，把这些样本当作 P 在空间中采样得到的即可，因为这些样本就是现实数据，那自然符合 P 分布.
>
> 当然，这里监督学习的目标进一步精细化为 $g(\mathbf{x})$ 和 $f(\mathbf{x})$ 在 P 上的误差尽量小. 即考虑了数据的分布.

④ 霍夫丁应用

In-sample error E_{in}: fraction of D where f and h disagree.

> corresponds to ν in the bin model

Out-of-sample error E_{out}: corresponds to μ in the bin model

$$E_{out}(h) = \mathbb{P}_{\mathbf{X}}(h(\mathbf{x}) \neq f(\mathbf{x}))$$
$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x_n}) \; != f(\mathbf{x_n}))$$

For any fixed h, when the sample size N is large, the in-sample error E_{in} is probably close to out-of-sample error E_{out} (within ϵ)

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$
$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| \leq \epsilon] \geq 1 - 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$

- Valid for all N and ϵ
- Does not depend on E_{out}, no need to know E_{out}

⑤ 单假设

> Verification of One h
>
> 关于单假设和多假设时泛化误差的区别，见 `Assignment 1 Question 5`.

For any fixed h, when the sample size N is large, the in-sample error E_{in} is probably close to out-of-sample error E_{out} (within ϵ)

$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$
$$\mathbb{P}[|E_{in}(h) - E_{out}(h)| \leq \epsilon] \geq 1 - 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$

This tells you, $E_{in}(h) \approx E_{out}(h)$

But, in order to claim $g \approx f$, $E_{in}(h)$ has to be small!

> 这就不得不使用多假设，从中选择最优.

用法 1：给定 N，计算偏差在可接受范围内（外）的概率下（上）界.

> 直接代入计算.

用法 2：给定偏差在可接受范围内的概率下界（例如 at least $1-\delta$，即偏差超出可接受范围的概率 at most δ），计算至少需要多大的 N

> 这里可以直接令 $2e^{-2\epsilon^2 N} = \delta$，但严谨的做法是用不等式.

To ensure that the deviation is bounded by ϵ with probability at least $1-\delta$, the sample size N must satisfy

$$N \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$$

> 见 `附录 Assignment 1 Question 5 (b)`.

Verification v.s. Real Learning

Verification	**Real Learning**
Fixed single hypothesis h	Fixed *hypothesis set* $\mathcal{H}$
h to be certified	g to be certified
h does not depend on $\mathcal{D}$	g results after searching $\mathcal{H}$ to fit $\mathcal{D}$
No control over E_{in}	Pick best E_{in}

Verification: can we say something outside the data about h?

-- to tell how good a particular hypothesis is.

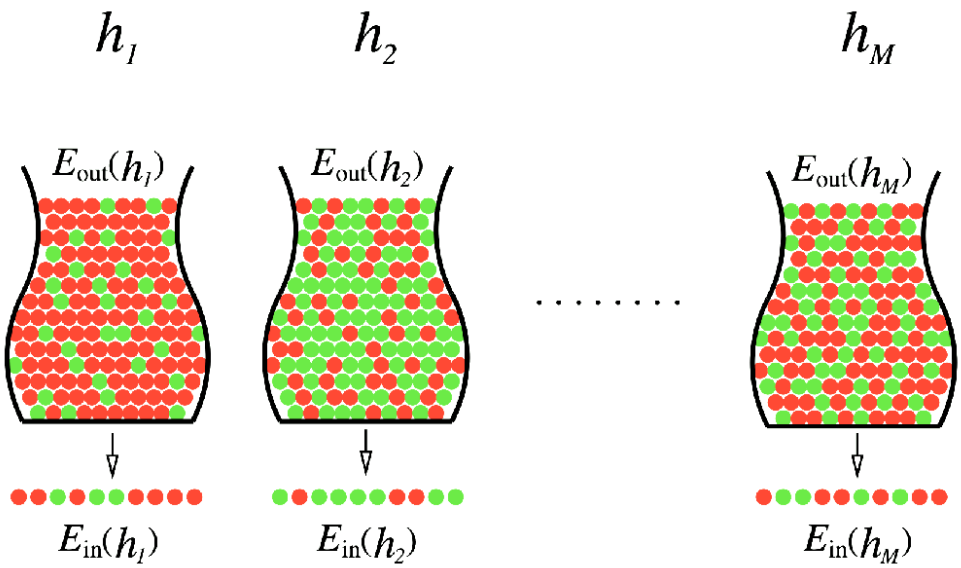
Learning: can we say something outside the data about g?

-- to extend the bin equivalence to the case of many hypotheses, in order to capture real learning.

注意：for a fixed h 或 for each hypothesis h_i 都是单假设 verification，后者的意思是把假设集 H 中的每个假设单独抽出来验证，它和多假设的表述（"the maximum deviation over all"）有明显不同.

⑥ 多假设

> Real Learning: Multiple Hypotheses



$$\mathcal{H} = \{h_1, h_2, \cdots, h_M\}$$

注意，多假设的最终选择 g 是使 $E_{in}(h)$ 最小的 h_i，这并不能保证 $|E_{in}(g) - E_{out}(g)|$ 在所有假设中也最小. 例如，有 5 个假设，我们将它们的 $|E_{in}(h) - E_{out}(h)|$ 从大到小排列，那么我们选择的 h_i 可能是其中任意一个，甚至可能是偏差最大的. 为了保证我们无论怎么选，$\mathbb{P}[|E_{in}(h_i) - E_{out}(h_i)| > \epsilon]$ 都要在我们的可接受范围内，那么我们必须保证 5 个假设中 $|E_{in}(h) - E_{out}(h)|$ 最大的那个，$\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon]$ 也要在可接受范围内. 换句话说，如果 $\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon]$ 超出了可接受范围，可以得知一定是 5 个中的某一个超出了可接受范围，但我们不知道是哪个，所以是 “第一个超出” 或 “第二个超出” 或 $\cdots$ 或 "第五个超出"，即使用并集上界（union bound）.

Rule 1: if $B_1 \Rightarrow B_2$, then $\mathbb{P}[B_1] \leq \mathbb{P}[B_2]$

Rule 2: $\mathbb{P}[\bigcup_{i=1}^{M} B_i] \leq \sum_{i=1}^{M} \mathbb{P}[B_i]$

> $\bigcup_{i=1}^{M} B_i$ 意思是 B_1 or B_2 or $\cdots$ or B_M.
>
> Rule 2 叫 Boole's inequality（布尔不等式）或者 union bound（并集上界）.

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \mathbb{P}\left[\bigcup_{i=1}^{M} |E_{in}(h_i) - E_{out}(h_i)| > \epsilon\right] \leq 2Me^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| \leq \epsilon] \geq \mathbb{P}\left[\max_{i=1,\ldots,M} |E_{in}(h_i) - E_{out}(h_i)| \leq \epsilon\right] \geq 1 - 2Me^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0$$

> 注意这里除了霍夫丁还用了并集上界，相当于两次放缩，界比较松.

用法 1：给定 M, N，计算偏差在可接受范围内（外）的概率下（上）界.

> 直接代入计算.

用法 2：给定 M，以及所有假设的最大偏差在可接受范围内的概率下界（例如 at least $1-\delta$，即最大偏差超出可接受范围的概率 at most δ），计算至少需要多大的 N

> 这里可以直接令 $2Me^{-2\epsilon^2 N}=\delta$，但严谨的做法是用不等式.

One way to use the inequality $\mathbb{P}[|E_{in}(g)-E_{out}(g)|>\epsilon] \leq \mathbb{P}\left[\bigcup_{i=1}^{M}|E_{in}(h_i)-E_{out}(h_i)|>\epsilon\right] \leq 2Me^{-2\epsilon^2 N}$ is to pick a tolerable difference ϵ as well as a tolerable BAD probability δ, and then gather data with size N large enough to achieve those tolerance criteria. Given ϵ, δ, M, what is the data size N needed?

To ensure that **the maximum deviation over all hypotheses** is bounded by ϵ with probability at least $1-\delta$, the sample size N must satisfy

$$N \geq \frac{1}{2\epsilon^2}\ln\frac{2M}{\delta}$$

> 见 `附录 Assignment 1 Question 5 (c)`.

注意题目问的是 "**for each** hypothesis h_i, the deviation is bounded by ϵ" 还是 "**the maximum deviation over all** hypotheses is bounded by ϵ". 前者是单假设，后者是多假设.

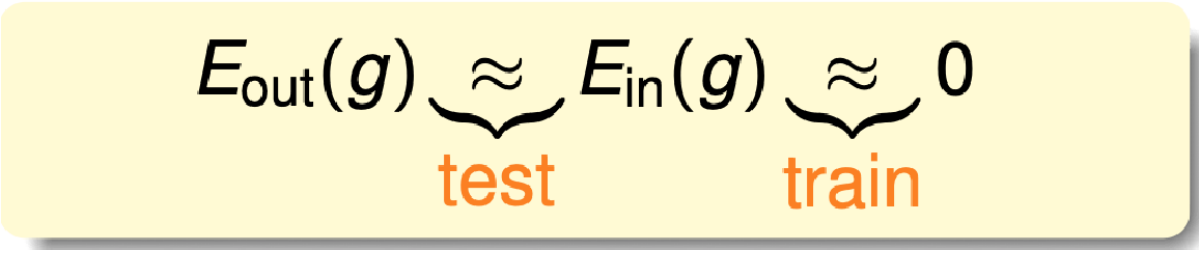
> 见 `附录 Assignment 1 Question 5`.

对于有限大的假设集 H，我们有：

$$\mathbb{P}[|E_{in}(g)-E_{out}(g)|>\epsilon] \leq \mathbb{P}\left[\bigcup_{i=1}^{|H|}|E_{in}(h_i)-E_{out}(h_i)|>\epsilon\right] \leq 2|H|e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon>0$$

$$\mathbb{P}[|E_{in}(g)-E_{out}(g)|\leq\epsilon] \geq \mathbb{P}\left[\max_{i=1,\ldots,|H|}|E_{in}(h_i)-E_{out}(h_i)|\leq\epsilon\right] \geq 1-2|H|e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon>0$$

If $|H|=M$ finite, N large enough, for whatever g picked by the learning Algorithm, $E_{in}(g)\approx E_{out}(g)$



Learning is still possible! However, in Perceptron, **infinite** $|H|$!!

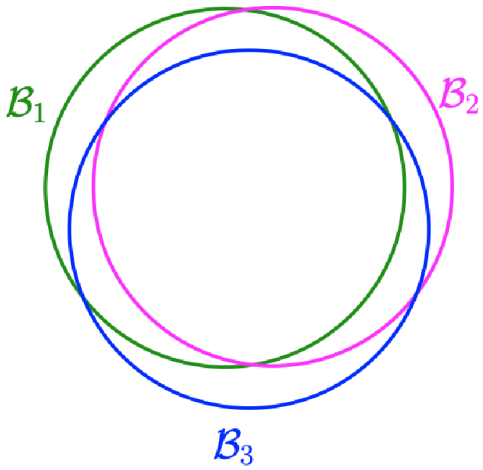
> 无限多假设时，霍夫丁无法解决，需要寻求新方法.
>
> 这里提到的感知器无限大 $|H|$，指的是感知器权重向量的各分量是实数，因此有无限个可能的权重向量.

Where did $|H|$ come from?

- Suppose we define the **bad** event as B_m : $|E_{in}(h_m)-E_{out}(h_m)|>\epsilon$

- Then, the worst case: all B_m non-overlapping

> 不等号取等条件是所有 “偏差超过可接受范围” 的情况不重叠，即不会同时发生，但实际上有很多相似的假设（例如感知器中，参数只差一点点的权重向量）它们的 bad event 很可能同时发生，所以 union bound 给出的界过于松了.



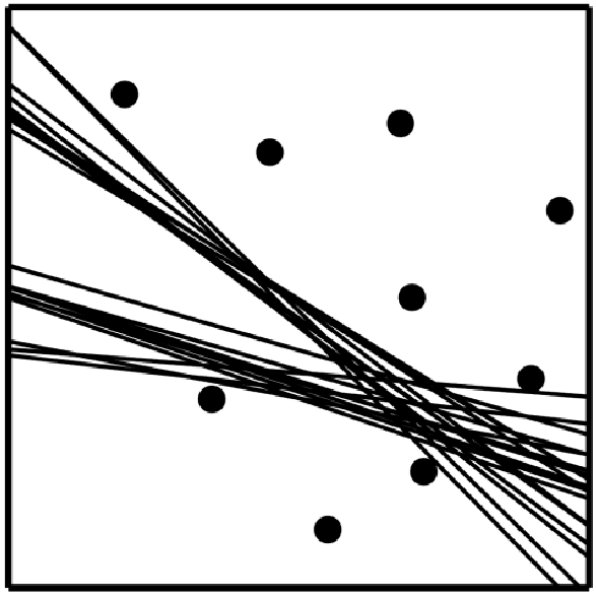
- B_m are events (sets of outcomes); they can overlap.
- If the B_m overlap, the union bound is loose.
- If many h_m are similar, the B_m overlap.
- There are "effectively" fewer than $|H|$ hypotheses.
- We can replace $|H|$ by something smaller.

⑦ 并集上界的局限

Why is $|H|$ an overkill?

- Overlapping for similar hypotheses $h_1 \approx h_2$
- The union bound over-estimating!

注：虽然我们无法判断 h_1 和 h_2 偏差是否同时大于 ϵ，但是从感知器模型中很容易发现，至少 E_{in} 相同的 hypotheses 有很多组. 例如：



Many lines but only one dichotomy!

Lecture 3 二分法

> Dichotomy

Roadmap

$$\mathbb{P}\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\right] \leq 2|\mathcal{H}|e^{-2\epsilon^2 N}, \quad \text{for any } \epsilon > 0.$$

$$\mathbb{P}\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| \leq \epsilon\right] \geq 1 - 2|\mathcal{H}|e^{-2\epsilon^2 N}, \quad \text{for any } \epsilon > 0.$$

Dichotomy → Break point → Sauer's lemma

$|\mathcal{H}|$ → $m_{\mathcal{H}}(N)$ → $B(N,k)$ → N^{k-1} → VC

Growth function — Bounding function

3.1 二分法

> Dichotomy

If $h \in H$ is applied to a finite sample $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$, we can get an N-tuple $h(\mathbf{x_1}), \cdots, h(\mathbf{x_N})$ of ± 1's. Such an N-tuple is called a dichotomy since it splits $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$ into two groups: those points for h is -1 and those for h is $+1$.

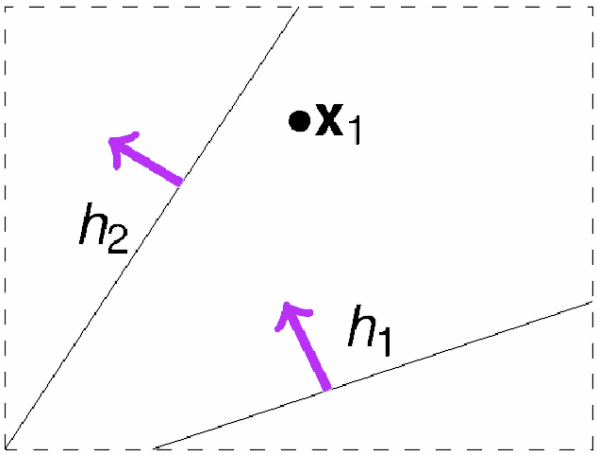
- A hypothesis h: $\mathbf{X} \to \{-1, +1\}$
- A dichotomy h: $\{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}\} \to \{-1, +1\}$

3.2 有效线数

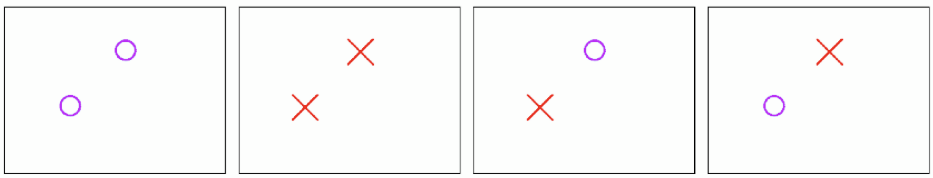
> Effective Number of Lines
>
> 把平面上 N 个点进行不同二分的直线种类数，相同二分的直线族记作一种.

Question: How many lines are there?

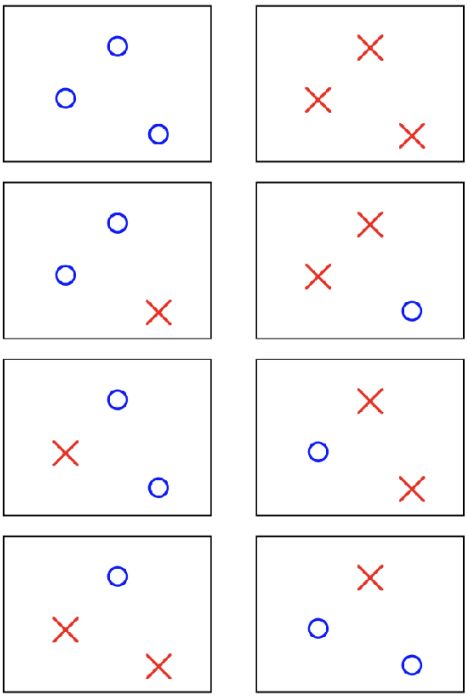
- For a single point, only two kinds of lines, $+$ or $-$



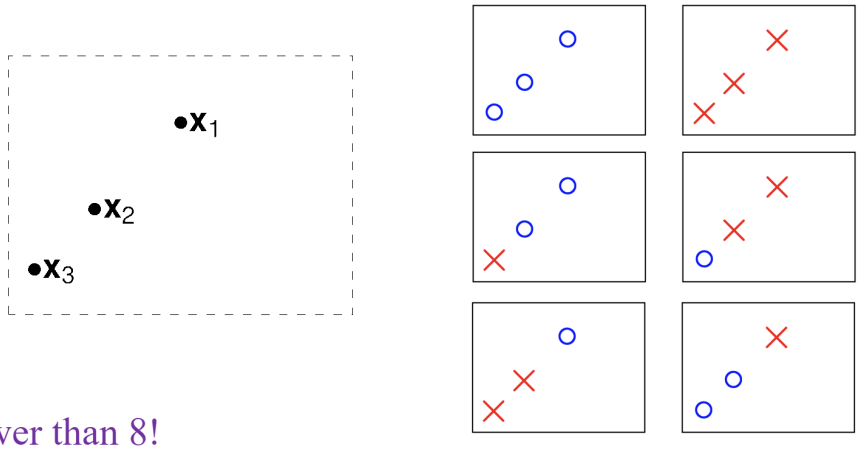
- For two points, there are four kinds of lines.



- For three points, 总有 2^3 种吗?



在某些特定排列方式下，二分直线可能不到 8 种：



> $\mathbf{x}_1, \mathbf{x}_3$ 分为 1，$\mathbf{x}_2$ 分为 -1 或 $\mathbf{x}_1, \mathbf{x}_3$ 分为 -1，$\mathbf{x}_2$ 分为 1 时是非线性二分，不能由直线划分，所以少了两种.

有效线数：Maximum kinds of lines with respect to N inputs data

> Effective Number of Lines

Lines in 2D space

- $N = 1$, effective $(N) = 2$
- $N = 2$, effective $(N) = 4$
- $N = 3$, effective $(N) = 8$
- $N = 4$, effective $(N) = 14$

> $0-4$ 分法一种，$1-3$ 分法四种，$2-2$ 分法两种. 共七种，± 1 对调翻倍.

3.3 增长函数

> The Growth Function

	hypotheses $\mathcal{H}$	dichotomies $\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$
e.g.	all lines in $\mathbb{R}^2$	$\{\circ\circ\circ\circ, \circ\circ\circ\times, \circ\circ\times\times, \ldots\}$
size	possibly infinite	upper bounded by 2^N

对于一个假设集（例如 $\mathbb{R}^2$ 中的所有直线），当我们取固定 N 个数据点时，每个假设会在这些点上给出一种二分类；那么这个假设集中的所有假设都能对这 N 个定点进行二分，有些二分相同，有些二分不同，最多有 2^N 种.

然而，这样 $|H(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})|$ 仍然 depend on inputs $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$. 我们评估的对象是假设集，希望不受输入样本的影响，于是定义 Growth Function: remove dependence by taking **max** of all possible $(\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N})$

$$m_H(N) = \max_{\mathbf{x_1}, \ldots, \mathbf{x_N}} |H(\mathbf{x_1}, \ldots, \mathbf{x_N})|.$$

> 在定义增长函数时，我们并不是仅考虑某一固定的一组 N 个点，而是考虑了输入空间中所有可能的 N 个点组合. 也就是说，我们对每一组可能的 N 个点计算假设集所能产生的二分类方式的数目，然后取其中的最大值作为增长函数的值. 因此，N 个点并非局限于一组，而是遍历了所有可能的组合.
>
> 求增长函数值，要动态地移动 N 个点，去达到最复杂的状态，使得该假设集对该状态下的这 N 个点有最多种二分方式.
>
> 增长函数是所有输入组合中的最大二分种数，那么对于一组特定的 N 个点，二分种数一定小于等于增长函数值.
>
> 增长函数可以看作假设集 H 的有效数量. 增长函数是假设集的属性，必须和特定的假设集同时出现，去衡量该假设集对平面上 N 个点的最大二分能力. 对于由所有 2D 直线构成的假设集，增长函数等于有效线数.
>
> 增长函数衡量的是假设集合在其输入空间上的划分能力. 所以，输入样本集必须与假设集定义的输入空间一致. 如果假设集合针对一条直线构造（例如正射线或正区间），那么计算时就应仅使用直线上的点，否则不能准确地反映模型的实际能力. 所以输入空间不一定是二维平面的点，主要看假设集（分类器）定义的输入空间.

增长函数同样有界：Finite, upper-bounded by 2^N

- $m_H(N) \leq 2^N$: finite grouping of infinite lines
- If the effective number of H can replace $|H|$, learning is feasible for infinite H.

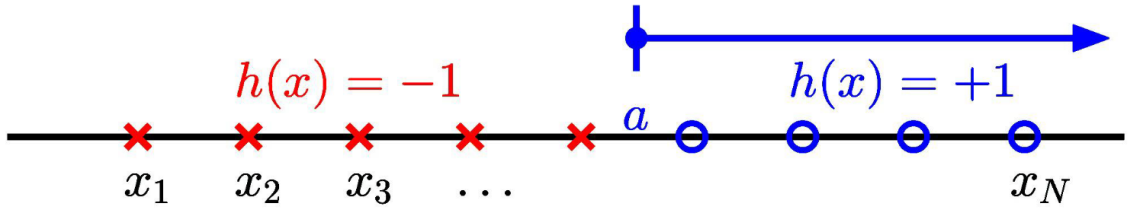
3.3.1 2D 感知器

2-D Perceptron Model: 假设集是平面上所有直线. 此时增长函数等于有效线数，即

$$m_H(3) = 8 = 2^3$$
$$m_H(4) = 14 < 2^4$$

3.3.2 正射线

> Positive Rays



> 注意正射线定义：如果一个输入数据点落在正射线区域内，则判为正类（例如标签为 $+1$）；否则判为负类（例如标签为 -1）. 因此不能对调 ± 1.

one dichotomy for $a \in$ each spot $(\mathbf{x_n}, \mathbf{x_{n+1}})$

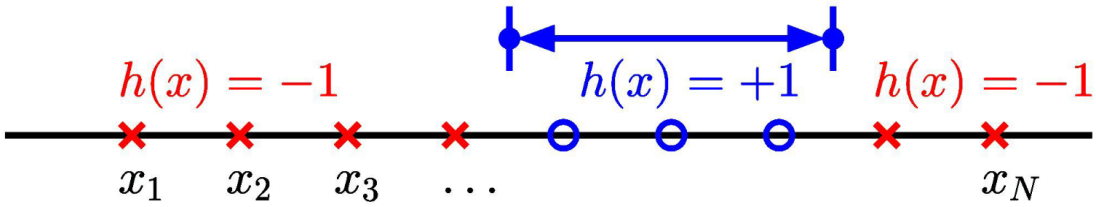
H is the set of $h : \mathbb{R} \to \{-1, +1\}$

$h(x) = sign(x - a)$

$m_H(N) = N + 1 = O(N) \ll 2^N$ when N is large.

3.3.3 正区间

> Positive Intervals



H is the set of $h : \mathbb{R} \to \{-1, +1\}$

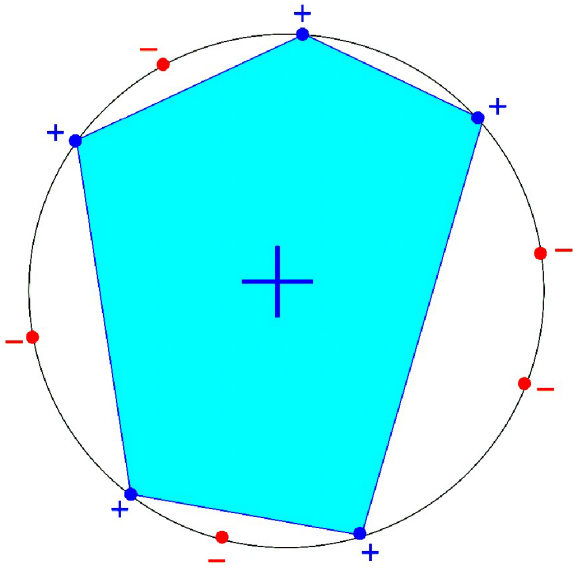
Place interval ends in two of $N + 1$ spots

- One dichotomy for each interval $\binom{N+1}{2}$
- All -1 case: 一个

$$
\begin{aligned}
m_H(N) &= \binom{N+1}{2} + 1 \\
&= \frac{1}{2}N^2 + \frac{1}{2}N + 1 = O(N^2) \ll 2^N \text{ when N is large}
\end{aligned}
$$

3.3.4 凸集合

> Convex Sets



> 假设空间中每个候选函数都是基于“凸集”来构造分类规则的：给定一个凸区域，函数在该区域内给出一种预测（比如 +1），在区域外给出另一种预测（比如 -1）.

H is the set of $h : \mathbb{R} \to \{-1, +1\}$

$h(\mathbf{x}) = +1$ is convex

$m_H(N) = 2^N$

The N points are **"shattered"** by convex sets

3.4 击碎

> shatter

If H is capable of generating all possible dichotomies on $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$, then we say H can shatter $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$ or say these N inputs can be shattered by H.

3.5 断点

> Break point of H

观察增长函数的几个例子：

- H: positive rays

$$m_H(N) = N + 1 = O(N)$$

- H: positive intervals

$$m_H(N) = \frac{1}{2}N^2 + \frac{1}{2}N + 1 = O(N^2)$$

- H: convex sets

$$m_H(N) = 2^N$$

Is there an upper bound of $m_H(N)$?

If so, can we bound $m_H(N)$ by a polynomial in N?

> 多项式增长速度慢于 2^N，可以进一步加紧泛化误差界.

Can we replace $|H|$ with $m_H(N)$ in the generalization bound?

> generalization bound，泛化误差界.

定义 H 的断点:

If no data point of size k can be shattered by H, then k is said to be a break point for H.

$$m_H(k) < 2^k$$

> 注意，只要找出一组 k 个数据能被 H 击碎，就能说明 k 不是 H 的断点.

3.5.1 2D 感知器

H: 平面中所有直线

k : $4, 5, \dots$

3.5.2 正射线

k : $2, 3, \dots$

> 两个点时，正射线无法将左边的点标记成 $+1$，右边的点标记成 -1.

3.5.3 正区间

k : $3, 4, \dots$

> 三个点时，正区间无法将中间的点标为 -1，两边的点标为 $+1$.

3.5.4 凸集合

$k = \infty$

因为 $m_H(N) = 2^N$ 恒成立.

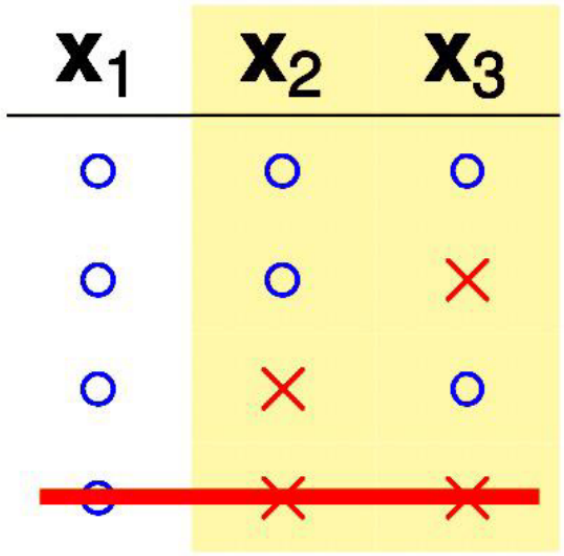
3.6 界函数

> Bounding Function $B(N, k)$

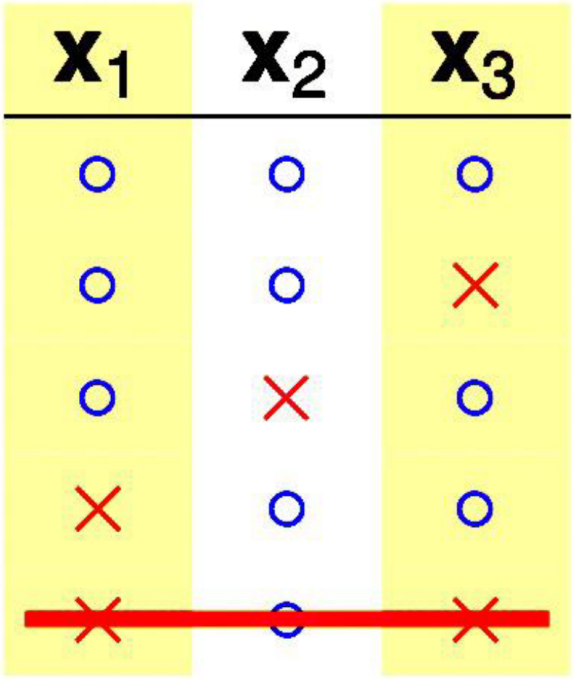
首先我们来解决一个具体问题：

What's the **maximum** possible $m_H(N)$ when $N = 3$ and the break point $k = 2$?

> 解释一下为什么是 maximum：首先，给定断点 $k = 2$，不代表这是最小断点；其次，即使 $k = 2$ 是最小断点，H 也有多种可能，例如 $N = 2$ 时，增长函数是比 2^2 少了一种还是少了两种？当我们求 maximum 时，意味着一切都是最优情况，只有在两个点的第四种组合出现时才要剔除掉. 具体如下方题解所示.



> 这里按二进制排序，到 011 时发现后两位已经出现了第四种二分，这和断点 $k = 2$ 的条件矛盾，所以剔除掉这种二分.



> 同理，101 时第一位和第三位也出现了第四种二分，剔除.

同理可得，110 和 111 时第一位和第三位出现了第四种二分，剔除. 最终 maximum possible: 4 dichotomies $\ll 2^3$

定义界函数:

$B(N,k)$ is the maximum number of dichotomies on N points such that no subset of size k of the N points can be shattered by these dichotomies.

> 这里 "be shattered by these dichotomies" 严格来说是被实现这些最大数量二分的假设空间 H 击碎.
>
> 注意界函数与 H 无关，可以任选满足 break point $= k$ 的假设空间.

$B(2,2)=3,\ B(3,2)=4$

- Break point $= k$ restricts $m_H(N)$
- Idea: Maximum possible $m_H(N)$ when break point $= k$
- Our goal: $B(N,k)$ can be bounded by a polynomial in N.
- Irrelavant of details of H

Another example to extend $B(4,3)$

> 待完成.

Optional Reading: Once Broken, Forever Polynomial

> 待完成.

Lecture 4 VC 维

> VC Dimension

4.1 VC 维

- The tightest bound is obtained with the smallest break point k^*.
- Definition: $d_{VC} = k^* - 1$
- Other definition: largest N for which $m_H(N) = 2^N$

The VC dimension is the largest N which can be shattered.

- $N \leq d_{VC}$: H could shatter your data (H can shatter some N points)

  > 注意，这里只能保证存在这么一组 N 能被击碎，不保证你用来训练的一组 N 恰好也能被击碎.

- $N > d_{VC}$: N is a break point for H; H cannot possibly shatter your data.

  > 这里可以保证任何一组 N 都不可能被 H 击碎.

- $m_H(N) \leq N^{d_{VC}} + 1$

VC-dimension is an Effective Number of Parameters:

	N							
	1	2	3	4	5	$\cdots$	#Param	d_{VC}
2-D perceptron	2	4	8	14	$\cdots$		3	3
1-D pos. ray	2	3	4	5	$\cdots$		1	1
2-D pos. rectangles	2	4	8	16	$< 2^5$	$\cdots$	4	4
pos. convex sets	2	4	8	16	32	$\cdots$	∞	∞

- 模型复杂度的度量：VC 维反映了一个模型能够“表达”或“拟合”的数据复杂程度. 它表示在理论上，该模型可以完美分类（区分）多少个点. 当一个模型能够对一组点的任意标记方式都得出正确的分类时，这组点就称为被该模型“打断”（shattered）. 模型的 VC 维就是能够被打断的最大点数.
- 不同模型与几何对象的关系：例如，在二维空间中，感知机模型的 VC 维是 3，这意味着这种模型在二维情况下能够完美分类最多 3 个点. 如果点数超过这个数量，模型不一定能够处理所有可能的分类方式. 类似地，其他几何结构（如一维射线、二维矩形或凸集）的 VC 维分别反映了它们在相应空间中的表达能力. 某些模型（例如定位凸集）的VC 维可能是无限大的，表示它们理论上能够拟合任意复杂的分类边界.

- 实际意义：VC 维高的模型具有较强的表达能力，能够捕捉数据中的复杂模式，但同时也容易出现过拟合的问题；而 VC 维较低的模型简单，泛化能力较强，但可能不足以捕捉复杂的数据结构. 在模型选择和学习算法设计中，理解和利用 VC 维有助于在拟合能力与泛化能力之间找到平衡.

4.1.1 1D 感知器

> 1-D Perceptron: What is the VC-dimension?

$m_H(N) = 2N, d_{VC} = 2$

> 最小断点是 3.
>
> 注意一维感知器相当于正射线加负射线，即决策树桩（Decision Stump）. 二分数量叠加时要去掉重复的全 $+1$ 和全 -1，即 $2 \times (N+1) - 2 = 2N$.

4.1.2 2D 感知器

> 2-D Perceptron: What is the VC-dimension?

$d_{VC} = 3$

> 最小断点是 4. 见 `3.2 有效线数` . 注意 $N = 3$ 时有可能无法击碎，但存在能被击碎的一组 3 个数据，而 $m_H(N)$ 是从所有组 N 个数据中选出二分数量最大的，只要存在一组能被击碎则值为 2^N，则 N 不为断点.

4.1.3 d 维感知器

> What is the VC-dimension of the Perceptron in $\mathbb{R}^d$?

Answer: $d + 1$

Why?

- There is a set of $d + 1$ points that can be shattered.
- Every set of $d + 2$ points cannot be shattered.

> 待证明：为什么为什么d维感知器的VC维是d+1（要用线性代数知识）.

4.2 VC 界

> The Vapnik-Chervonenkis Bound (VC Bound)

$$\mathbb{P}\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\right] \leq 4m_{\mathcal{H}}(2N)e^{-\epsilon^2 N/8}, \quad \text{for any } \epsilon > 0.$$

$\mathbb{P}[|E_{\text{in}}(g)-E_{\text{out}}(g)|>\epsilon] \leq 2|\mathcal{H}|e^{-2\epsilon^2 N}$ ← finite $\mathcal{H}$

$$\mathbb{P}\left[|E_{\text{in}}(g) - E_{\text{out}}(g)| \leq \epsilon\right] \geq 1 - 4m_{\mathcal{H}}(2N)e^{-\epsilon^2 N/8}, \quad \text{for any } \epsilon > 0.$$

$\mathbb{P}[|E_{\text{in}}(g)-E_{\text{out}}(g)|\leq\epsilon] \geq 1-2|\mathcal{H}|e^{-2\epsilon^2 N}$ ← finite $\mathcal{H}$

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N}\log\frac{4m_{\mathcal{H}}(2N)}{\delta}}, \quad \text{w.p. at least } 1-\delta.$$

$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{1}{2N}\log\frac{2|\mathcal{H}|}{\delta}}$ ← finite $\mathcal{H}$

$$m_{\mathcal{H}}(N) \leq \sum_{i=1}^{k-1}\binom{N}{i} \leq N^{k-1}+1 \quad k \text{ is a break point.}$$

> 注意第三个不等式，还有一个下界，但是我们更关注上界，因为我们想尽量减少泛化误差.

Sample Complexity: How Many Data Points Do You Need?

> 待完成.

Lecture 5 偏差-方差分析

> Bias-Variance Analysis

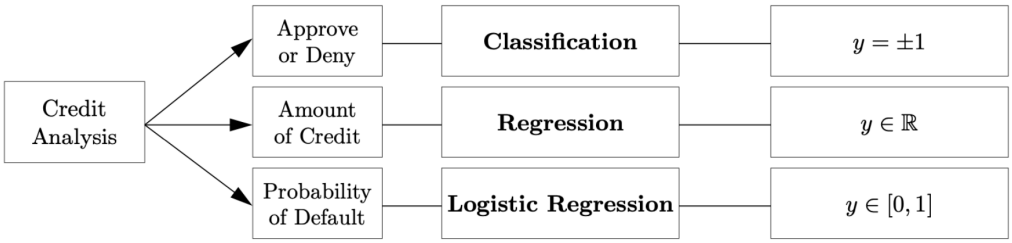
Lecture 6 线性模型

> Linear Models

6.1 三种学习任务

> Three Examples of Learning

机器学习有三种常见的学习任务. 以信用分析（Credit Analysis）为例：



对于不同类型的学习任务，可以使用不同的学习方法来解决.

6.1.1 分类

> Classification

任务：决定是否批准贷款 / 信用卡.

方法：分类模型.

目标变量：$y = \pm 1$（二分类）

6.1.2 回归

> Regression

任务：预测贷款 / 信用卡额度.

方法：回归模型.

目标变量：$y \in \mathbb{R}$.

> 贷款金额可以是任意连续数值.

6.1.3 逻辑回归

> Logistic Regression

任务：预测贷款违约的概率.

方法：逻辑回归.

目标变量：$y \in [0, 1]$

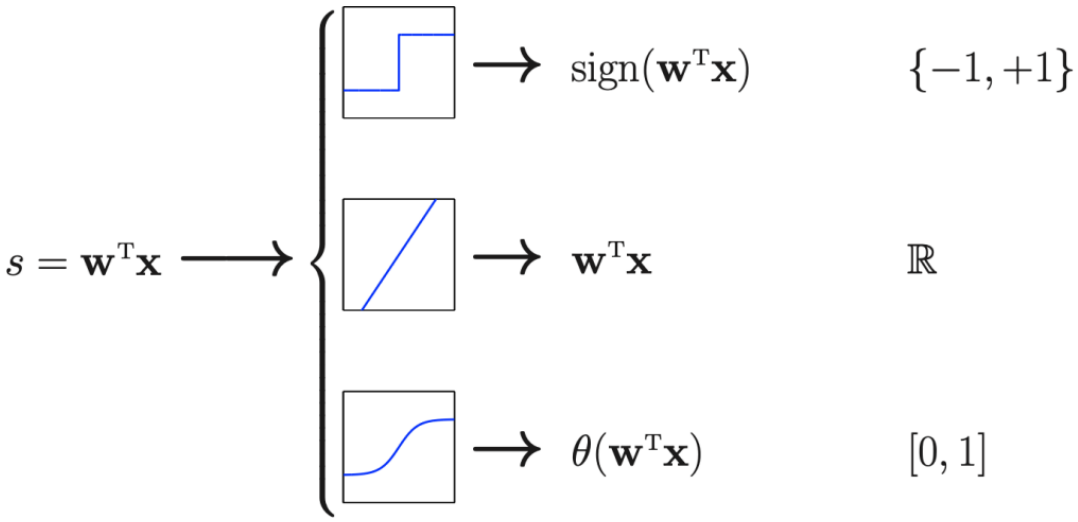
> 概率值，表示借款人违约的可能性.

这三种任务中，线性模型（Linear Models）是最基础的模型. 建模时，线性模型通常是最先尝试的方法. 前面几章节主要介绍了线性模型在二分类中的应用（感知器），本章则介绍线性模型在回归和逻辑回归中的应用.

6.2 线性信号

> The Linear Signal

输入数据 $\mathbf{x}$ 以线性变换 $s = \mathbf{w}^T\mathbf{x}$ 为桥梁，可生成不同类型的输出 y，用于不同任务.



- 线性模型在 $\mathbf{x}$ 上是线性的，因此它可以用于找到数据的线性分隔超平面.

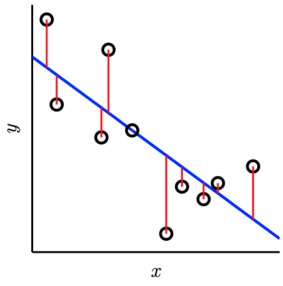
  > line / hyperplane separator

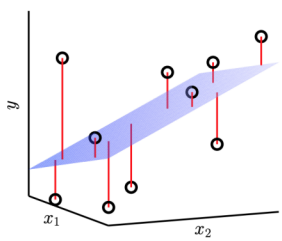
- $y = f(s)$ 表示最终输出是对 $s = \mathbf{w}^T\mathbf{x}$ 进行某种变换 $f(s)$ 得到的.

6.3 线性回归

> Linear Regression

age	32 years
gender	male
salary	40,000
debt	26,000
years in job	1 year
years at home	3 years
...	...





> 左：输入特征.
>
> 中：2D 线性回归.

> 右：3D 线性回归.

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \begin{cases} E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}(h(\mathbf{x}_i) - y_i)^2 & \text{in-sample error} \\ E_{out}(h) = \mathbb{E}_{\mathbf{x}}[(h(\mathbf{x}) - y)^2] & \text{out-of-sample error} \end{cases}$$

6.3.1 矩阵表示

> Using Matrices for Linear Regression

$$\underbrace{\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1\text{—} \\ \text{—}\mathbf{x}_2\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N\text{—} \end{bmatrix}}_{\text{data matrix, } N \times (d+1)} \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}} \qquad \underbrace{\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{w}^{\mathrm{T}}\mathbf{x}_1 \\ \mathbf{w}^{\mathrm{T}}\mathbf{x}_2 \\ \vdots \\ \mathbf{w}^{\mathrm{T}}\mathbf{x}_N \end{bmatrix} = \mathrm{X}\mathbf{w}}_{\text{in-sample predictions}}$$

线性回归的目标是找到一个权重向量 $\mathbf{w}$，使得输入变量 X（矩阵）经过线性变换 $\hat{y} = X\mathbf{w}$ 后，尽可能接近目标值 $\mathbf{y}$.

① 数据矩阵

> data matrix

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}$$

- X 是 $N \times (d+1)$ 维矩阵，其中 N 是样本数量，$d+1$ 是特征数.
- 每个样本第一项均为常数项偏置 $x_{i0} = 1$.

  > 见 `6.3.2 常数项偏置` .

② 目标向量

> target vector

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- $\mathbf{y}$ 是 $N \times 1$ 向量，第 i 项表示第 i 个样本 $\mathbf{x}_i$ 对应的真实输出值.

③ 预测值

> in-sample predictions

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}_1 \\ \mathbf{w}^T\mathbf{x}_2 \\ \vdots \\ \mathbf{w}^T\mathbf{x}_N \end{bmatrix} = X\mathbf{w}$$

6.3.2 常数项偏置

> Bias Term

在线性回归模型中，通常建模一个线性函数：

$$h(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_dx_d$$

这个模型假设回归线（或超平面）必须通过原点 $(0, \cdots, 0)$，但这在很多实际问题中是不合理的.

> 一个样本所有输入特征都为 0 时，预测值不一定要为 0，取决于实际问题.

因此，引入一个额外的偏置项（Bias Term）w_0，使模型变为：

$$h(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_dx_d$$

其中：

- w_0 为截距项（Intercept Term），$\mathbf{x} = [1, 0, \cdots, 0]$ 时的预测值.
- $w_1, \cdots, w_d$ 是有效特征对应的权重.
- 为匹配 $\mathbf{w}$ 的维度，所有输入向量额外拓展一个维度（变为 $d+1$ 维），即 $x_0 = 1$.
- 偏置项 w_0 可以看作截距项 w_0 和恒定的 $x_0 = 1$ 之乘积. 这么做相当于额外添加一个需要训练的参数，放在 $\mathbf{w}$ 中统一训练.
- $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$ 是包含偏置项的参数向量.

注意，引入偏置项是有利无害的，因为带偏置项后可以表示任何线性关系，包括过原点的超平面，但不带偏置项的模型无法表示不过原点的超平面. 即引入偏置项大大提升了模型的表达能力；同时，用最小二乘法或梯度下降优化线性回归时，增加偏置项可以减少偏差，提高收敛速度.

> 实际应用中，几乎所有线性模型都会包含偏置项，以提高拟合效果和模型稳定性.

6.3.3 训练误差

> In-Sample Error

线性回归的目标是最小化**均方误差**（MSE）：

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

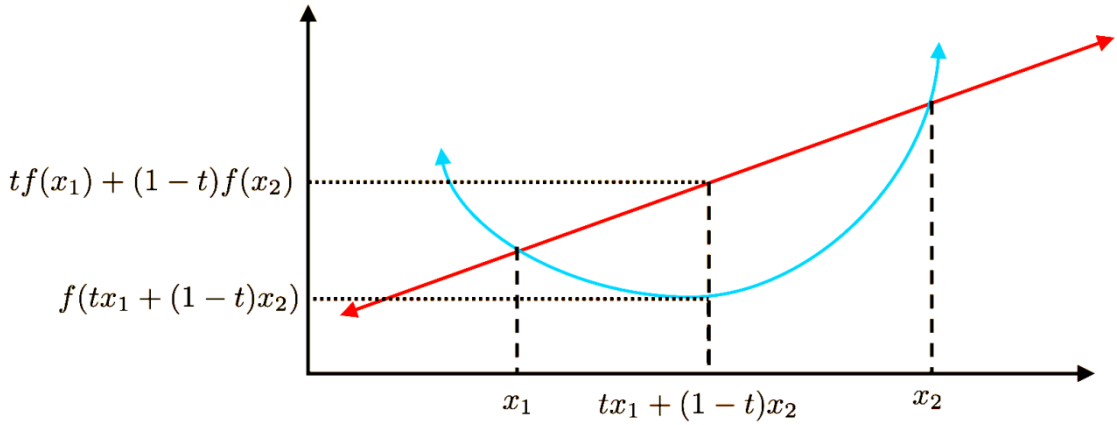
使用矩阵形式可以重写为：

$$\begin{aligned} E_{in}(\mathbf{w}) &= \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \\ &= \frac{1}{N}\|X\mathbf{w} - \mathbf{y}\|_2^2 \\ &= \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \end{aligned}$$

关于范数的定义、符号规定，以及该公式的推导，见 `附录 2. 范数`.

6.3.4 凸性

> Convexity
>
> 待补充.



6.3.5 优化方法

优化目标：找到最优的 $\mathbf{w}$ 使得**均方误差**（MSE）最小化. 误差的数学表达式为：

$$E_{in}(\mathbf{w}) = \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{y}^T\mathbf{y}).$$

① 正规方程

误差函数连续、可微、凸，因此可以通过令梯度为 $\mathbf{0}$ 找到最优解：

$$\nabla E_{in}(\mathbf{w}) = \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0} \\ \frac{\partial E_{in}}{\partial w_1} \\ \vdots \\ \frac{\partial E_{in}}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

计算可得：

$$\nabla E_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T\mathbf{y})$$

> 推导见 `3. 线性回归梯度计算` .

最优解满足梯度等于 $\mathbf{0}$，即

$$X^T X\mathbf{w} = X^T\mathbf{y}$$

这就是正规方程（Normal Equation）. 在 $X^T X$ 可逆的前提下，其闭式解为：

$$\mathbf{w} = (X^T X)^{-1} X^T\mathbf{y}.$$

此解即最小二乘估计（Least Squares Estimation）.

> 注意此解有效的前提：$X^T X$ 可逆.

② 梯度下降

对于小规模数据，通过正规方程直接求解：

$$\mathbf{w} = (X^T X)^{-1} X^T\mathbf{y}.$$

对于大规模数据，用梯度下降（Gradient Descent）更高效.

> 本节不讨论梯度下降，后续会讨论.

6.3.6 误差分析

> 待补充.

6.3.7 潜在问题

> 待补充.

- **非线性关系**：如果数据具有非线性关系，简单的线性回归可能无法很好地拟合数据.
- **误差项的相关性**：如果误差项是正相关的，可能会在残差图上观察到“跟踪”模式.
- **误差方差不恒定**：如果误差的方差随预测值变化，可通过对因变量进行变换（如对数变换）来改善.
- **多重共线性**：当两个或多个自变量高度相关时，可能导致估计不稳定.

6.4 逻辑回归

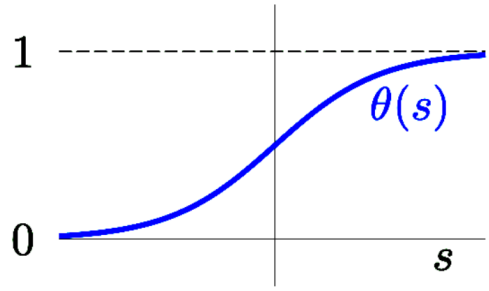
> Logistic Regression

6.4.1 逻辑函数

> The logistic function

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

图像:



This formula can be interpreted as a probability.

Example: the probability of someone that might need to apply for a credit next year.

Input $\mathbf{x}$: salary this year, debt at this moment, other features including job market information, etc...

$\theta(s)$: probability of applying for a credit next year

6.4.2 真实概率

> Genuine Probability

在逻辑回归中，我们假设模型输出的值可以被解释为目标变量为 1 的真实概率.

> 即假设数据是由某种概率机制产生的.

Data $(\mathbf{x}, y)$ with binary y, generated by a noisy target:

$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

> noisy：标签 y 不是由一个确定性的规则决定的，而是带有随机性的，由一个概率机制生成的. 这种不确定性 / 随机性也叫做噪声（noise）.

The target $f : \mathbb{R}^d \to [0, 1]$ is the probability

Learn $g(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) \approx f(\mathbf{x})$

注意，真实概率是一种建模假设，并不保证现实世界真的满足这个概率机制. 可能的偏离包括：

- 数据并非由概率产生.

  > 例如分类结果由严格规则（如物理定律）决定，没有噪声或随机性.

- 数据的概率形式不是 sigmoid.

  > 如果真实的 $f(x)$ 不是 sigmoid 形状，那么逻辑回归就无法很好地拟合它.

- 输入特征不充分.

  > 如果特征中缺少关键变量，那么即使有概率机制，模型学到的概率也会偏离真实值.

6.4.3 似然函数

> Likelihood

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

For each $(\mathbf{x}, y)$, y is generated by probability $f(\mathbf{x})$. 用 $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$ 来近似 $f(\mathbf{x})$. 我们的目标是学习到一个 $h(\mathbf{x})$，使得它的表现尽可能接近 $f(\mathbf{x})$.

> 也就是学习到一个权重向量 $\mathbf{w}$.

6.4.4 训练误差

> In-Sample Error

由于诸多原因，MSE 不适合逻辑回归. Plausible error measure based on likelihood: 最大化似然 - 最小化似然负对数 - 最小化对数损失 / 交叉熵损失.

解释：如果模型预测得好，则

- 对于 $y=+1$ 的样本，$h(\mathbf{x})$ 应该接近 1；
- 对于 $y=-1$ 的样本，$h(\mathbf{x})$ 应该接近 0.

我们要寻找一个最有可能观测到训练集 $\mathcal{D}=\{(\mathbf{x}_1,y_1),\dots,(\mathbf{x}_n,y_n)\}$ 的模型，即最大化似然 $\prod\limits_{n=1}^{N}P(y_n|\mathbf{x}_n)=\prod\limits_{n=1}^{N}\theta(y_n\mathbf{w}^T\mathbf{x}_n)$.

> 注意到 $\theta(-s)=\frac{e^{-s}}{1+e^{-s}}=\frac{1}{1+e^{s}}=\frac{1+e^{s}-e^{s}}{1+e^{s}}=1-\theta(s)$.
>
> 当 $y_n=1$ 时，$P(y_n|\mathbf{x}_n)=\theta(\mathbf{w}^T\mathbf{x}_n)$;
>
> 当 $y_n=-1$ 时，$P(y_n|\mathbf{x}_n)=1-\theta(\mathbf{w}^T\mathbf{x}_n)=\theta(-\mathbf{w}^T\mathbf{x}_n)$.
>
> 综上，$P(y_n|\mathbf{x}_n)=\theta(y_n\mathbf{w}^T\mathbf{x}_n)$.

$$
\begin{aligned}
& \max \quad \prod_{n=1}^{N}P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \max \quad \ln\left(\prod_{n=1}^{N}P(y_n|\mathbf{x}_n)\right) \\
\Leftrightarrow \quad & \max \quad \sum_{n=1}^{N}\ln P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \min \quad -\frac{1}{N}\sum_{n=1}^{N}\ln P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \min \quad \frac{1}{N}\sum_{n=1}^{N}\ln \frac{1}{P(y_n|\mathbf{x}_n)} \\
\Leftrightarrow \quad & \min \quad \frac{1}{N}\sum_{n=1}^{N}\ln \frac{1}{\theta(y_n\mathbf{w}^T\mathbf{x}_n)} \\
\Leftrightarrow \quad & \min \quad \frac{1}{N}\sum_{n=1}^{N}\ln \left(1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right)
\end{aligned}
$$

则 $E_{in}(\mathbf{w})=\frac{1}{N}\sum\limits_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right)$.

注意，逻辑回归的训练误差 / 损失函数 $E_{in}(\mathbf{w})=\frac{1}{N}\sum\limits_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right)$ 和交叉熵损失 $\mathcal{L}(\mathbf{w})=\frac{1}{N}\sum\limits_{n=1}^{N}[-y_n\log h(\mathbf{x}_n)-(1-y_n)\log(1-h(\mathbf{x}_n))]$ 等价，但形式略有不同.

> 见 附录 4. 交叉熵损失.

注意，线性回归中，$E_{in}(\mathbf{w}) = \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y})$ 是关于 $\mathbf{w}$ 的凸二次函数，令它导数（梯度）为零可得到一个线性方程组 $X^T X\mathbf{w} = X^T \mathbf{y}$，在 $X^T X$ 可逆前提下，可通过线性代数知识求出闭式解 $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$.

然而，逻辑回归中，$E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right)$ 是非线性且非凸二次形式，无法通过简单的代数运算求出解析解（闭式解，closed-form solution），必须使用数值优化方法（如梯度下降）求出数值解（迭代解，iterative solution）.

6.4.5 梯度下降

> Iterative Method: gradient descent

梯度下降法（Gradient Descent）

- 用于求解无法用闭式解表示的问题（如逻辑回归）.
- 基本思路：沿最陡下降方向迭代更新参数.
- 学习率（Step Size）权衡选择：太大不收敛，太小收敛慢.
- 可能遇到局部最小值或复杂地形.

Logistic regression algorithm

1: Initialize the weights at $t = 0$ to $\mathbf{w}(0)$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Compute the gradient

$$\nabla E_{\text{in}} = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^{\mathsf{T}}(t) \mathbf{x}_n}}$$

4: Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}}$
5: Iterate to the next step until it is time to stop
6: Return the final weights $\mathbf{w}$

逻辑回归中，损失函数 $E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right)$ 对 $\mathbf{w}$ 的梯度：

考察某一项 $f_n(\mathbf{w}) = \ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$.

$$\nabla f_n(\mathbf{w}) = \frac{1}{1 + e^{-y_n \mathbf{w}^{\mathbf{T}} \mathbf{x}_n}} \cdot (-e^{-y_n \mathbf{w}^{\mathbf{T}} \mathbf{x}_n}) \cdot (y_n \mathbf{x}_n) = -\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}.$$

对所有项求和，得

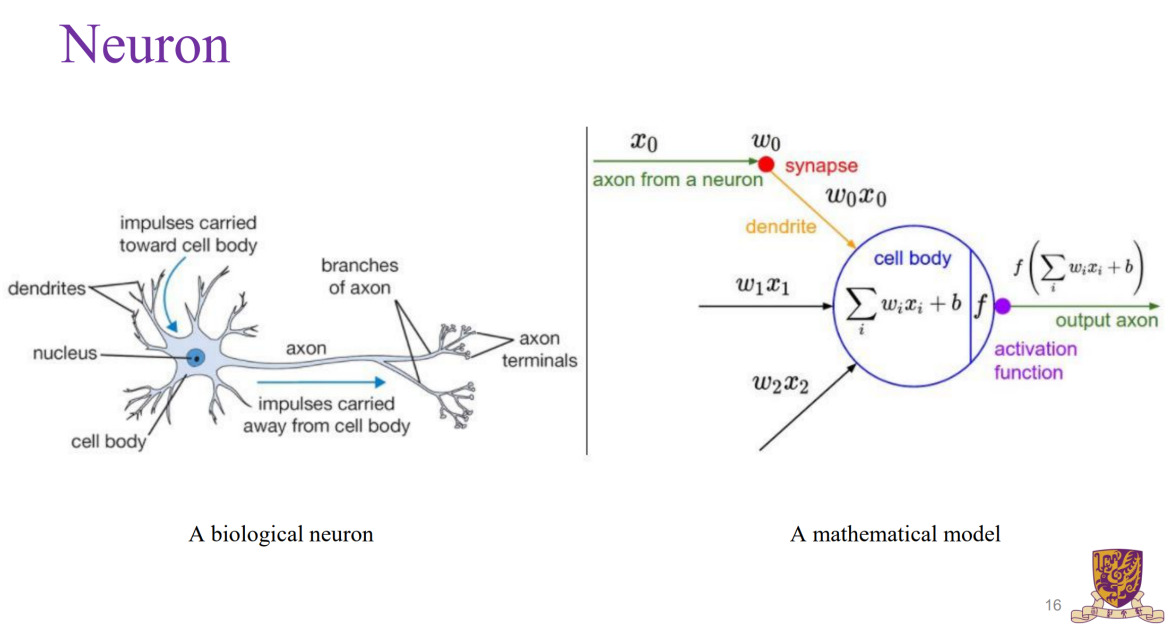
$$
\begin{aligned}
\nabla E_{in}(\mathbf{w}) &= \nabla \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right) \\
&= \frac{1}{N} \sum_{n=1}^{N} \nabla \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right) \\
&= \frac{1}{N} \sum_{n=1}^{N} -\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}} \\
&= -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}
\end{aligned}
$$

Lecture 7 神经网络

> Neural Networks

7.1 神经元

> Neuron



- 生物神经元的简化建模.
- 数学模型：输入 + 权重 + 激活函数 = 输出.

dendrites 树突

nucleus	细胞核
cell body	细胞体
impulses carried toward cell body	信号传向细胞体
axon	轴突（传出信号的通道）
branches of axon	轴突的分支
axon terminals	轴突末端
impulses carried away from cell body	信号从细胞体传出
axon from a neuron	来自其他神经元的轴突
synapse	突触（连接处）
activation function	激活函数
output axon	输出轴突

生物神经元：树突接收来自其他神经元的信号，信号汇聚到细胞体处理，再通过轴突传送到下一个神经元的树突. 这个过程就是神经信号的传播过程.

人工神经元（感知机 / 神经单元）：每个输入 x_i（树突接收信号）通过一个权重 w_i（突触连接信号）进行加权，所有加权输入相加，再加上偏置 b（细胞体处理信号），形成 $z = \sum_i w_i x_i + b$，然后通过激活函数 $f(\cdot)$（激活 / 放电），输出结果 $f\left(\sum_i w_i x_i + b\right)$（轴突发送信号），这个输出就是神经元的激活值，会被传递到下一层神经元.

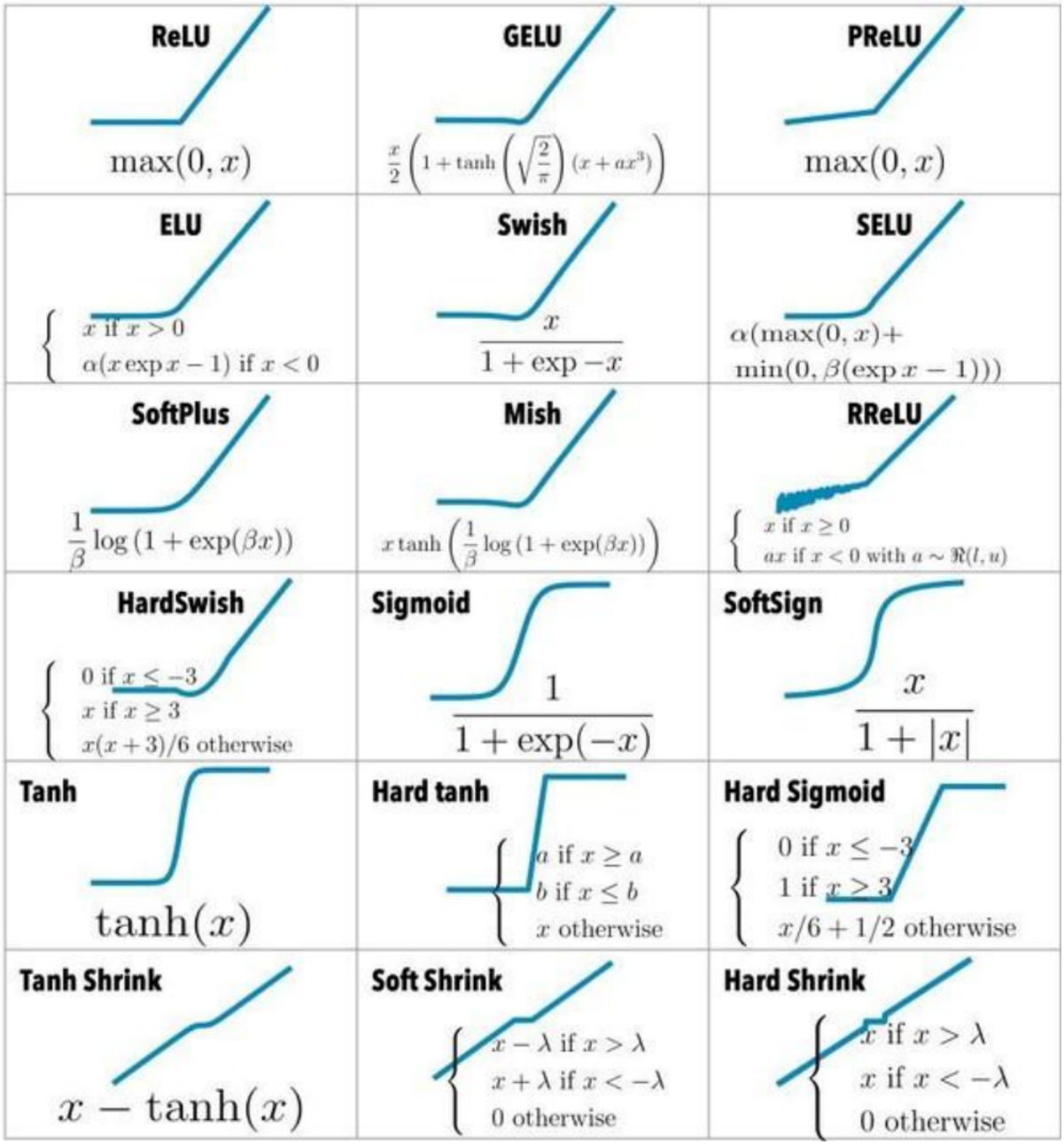
7.2 激活函数

> Activation Functions

- 引入非线性，使模型具备拟合非线性函数的能力.
- 常见激活函数：ReLU、sigmoid、tanh

Activation function is a function that runs on the neurons of the neural network and responsible for mapping the input of the neuron to the output.

The activation function introduces a non-linear factor to the neuron, so that the neural network can approach any non-linear function.

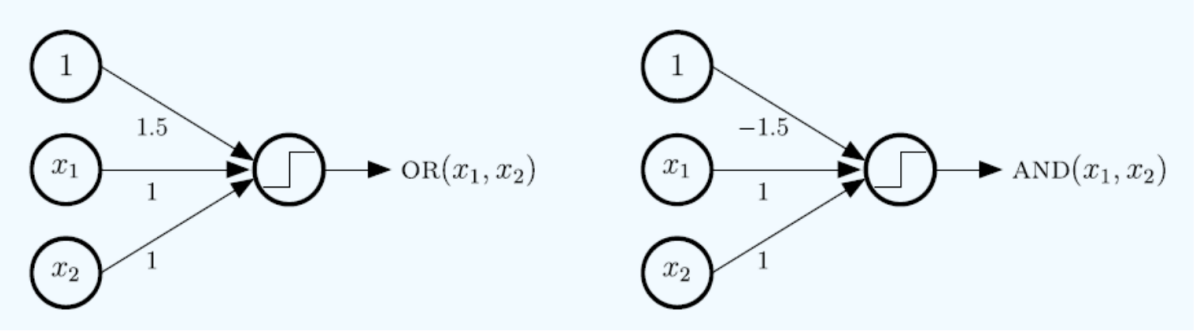


注意：当前大多数激活函数是**凭直觉、经验、实验验证**出来的，人为设计居多，而它们**在网络中具体怎么发挥作用，仍缺乏统一的定量衡量标准**. 这正是深度学习仍被称为「黑箱」的原因之一.

7.3 感知器

> Perceptron

- 可实现简单逻辑：AND / OR

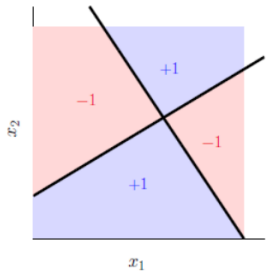


> $\mathrm{OR}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 + 1.5)$;
>
> $\mathrm{AND}(x_1, x_2) = \mathrm{sign}(x_1 + x_2 - 1.5)$.
>
> （注意，感知机中常用 $-1, +1$ 表示逻辑 False, True.

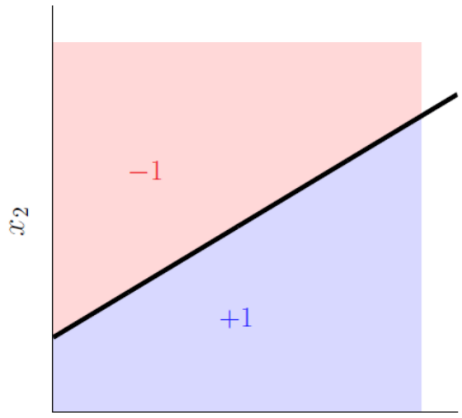
- XOR 无法线性分割，需要多层结构.

例：分解异或函数

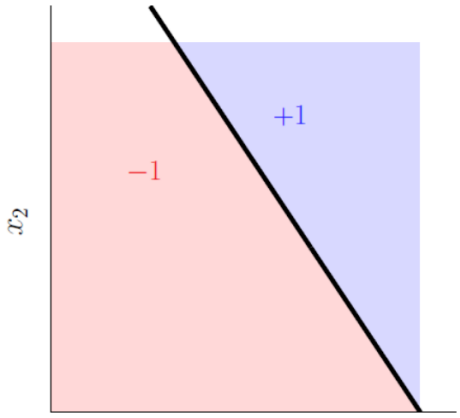
> Decomposing XOR



$$f = h_1\overline{h_2} + \overline{h_1}h_2$$



$$h_1(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})$$



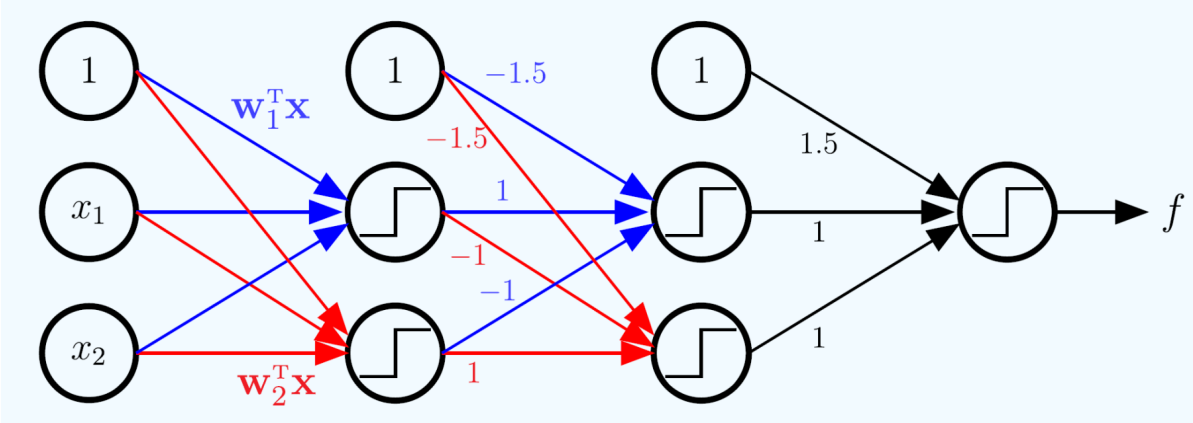
$$h_2(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$$

XOR 是一个经典的非线性分类问题，不能用一个简单的线性模型（如一个感知机）解决. 但我们可以将它拆分为两个线性分类器，再通过组合构造出 XOR. 这就涉及到信号在不同层神经元之间的传递，即多层感知器.

7.4 多层感知器

> The Multi-layer Perceptron (MLP) for a Complex Target

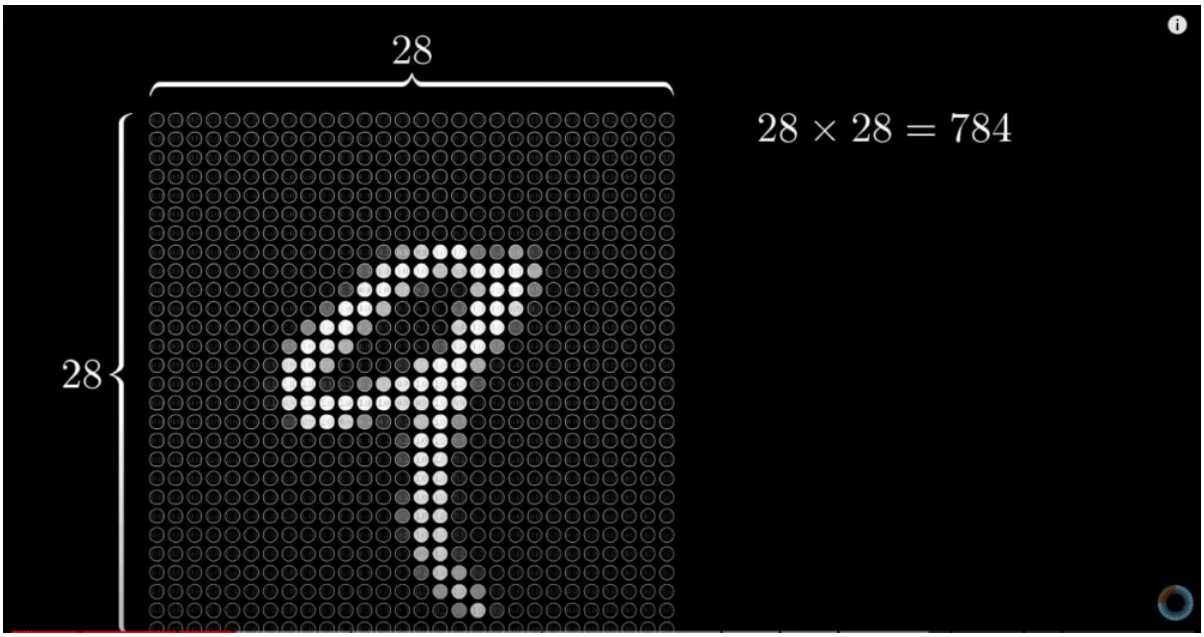
- 利用隐藏层实现复杂函数表示能力.
- 可分解实现 XOR 等复杂目标.



> 这里 $-1.5, -1, 1, 1.5$ 是权重，不是具体传递的值.
>
> 输入层（3 个输入节点）：x_1, x_2，加一个常数偏置 1. 可以理解为输入一个向量 $\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$.
>
> 第一个隐藏层： 两个神经元，接收输入层输入，各自加权并激活，
> $h_1 = \text{sign}(\mathbf{w}_1^T\mathbf{x}), h_2 = \text{sign}(\mathbf{w}_2^T\mathbf{x})$，相当于两个不同的感知器. 这两个感知器的激活值 h_1, h_2 加上一个偏置 1，作为下一层的输入.
>
> 第二个隐藏层：两个神经元，以第一个隐藏层的激活输出加一个偏置 1 作为输入，即输入 $\begin{bmatrix} 1 \\ h_1 \\ h_2 \end{bmatrix}$，各自加权并激活，$h_1\overline{h_2} = \text{sign}\left([-1.5, 1, -1] \begin{bmatrix} 1 \\ h_1 \\ h_2 \end{bmatrix} \right)$，$\overline{h_1}h_2 = \text{sign}\left([-1.5, -1, 1] \begin{bmatrix} 1 \\ h_1 \\ h_2 \end{bmatrix} \right)$，同样是两个不同的感知器. 这两个感知器的激活值 $h_1\overline{h_2}, \overline{h_1}h_2$ 加上一个偏置 1，作为下一层的输入.
>
> 输出层：一个神经元，以第二个隐藏层的激活输出加一个偏置 1 作为输入，即输入 $\begin{bmatrix} 1 \\ h_1\overline{h_2} \\ \overline{h_1}h_2 \end{bmatrix}$，加权并激活，$h_1\overline{h_2} + \overline{h_1}h_2 = \text{sign}\left([1.5, 1, 1] \begin{bmatrix} 1 \\ h_1\overline{h_2} \\ \overline{h_1}h_2 \end{bmatrix} \right)$，是一个感知器. 这个感知器的激活值即整个模型的最终输出.

注意，这个网络十分简单，仅作为示例，展示神经网络如何通过组合多个线性分类器实现非线性逻辑函数. 该网络不需要训练， 因为 XOR 是一个小且已知的布尔函数，可以基于真值表写出表达式，所有的权重和偏置已经人为设定好，是硬编码的，不需要反向传播、不需要梯度下降、不需要任何训练过程.

> Examples of MLP: Handwritten Digit Recognition



- Concatenate pixels in each row get a input column vector $\mathbf{x} \in \mathbb{R}^{784}$.
- Network use the input vector to compute 10 probability of 10 digits
- The digit which has max probability is the final classification result

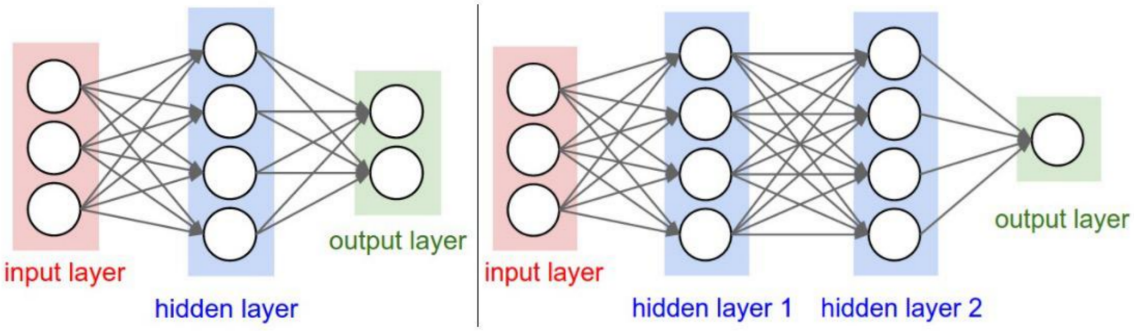
7.5 神经网络

> Neural Network

Neural Network is a technical reproduction of biological neural network in a simplified sense.

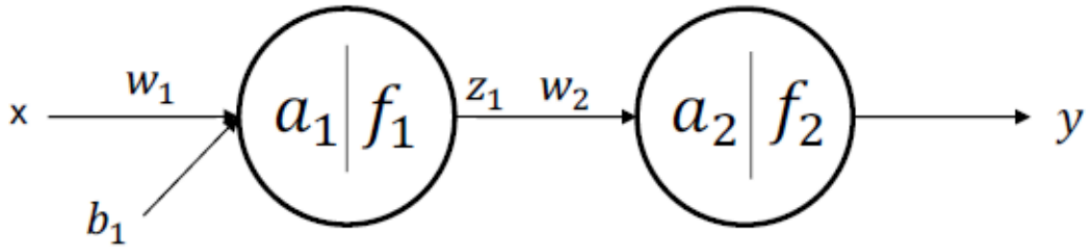
What problems it can solve?

- Classification task
- Clustering task
- Prediction task
- Structure learning task



下面以一个简单网络为例，讨论神经网络训练相关概念.

Suppose a simple network like this:



其中，

$a_1 = w_1 x + b_1$

$z_1 = f_1(a_1)$

$a_2 = w_2 z_1$

$y = f_2(a_2)$

7.5.1 损失函数

> Loss Functions

Many choices of loss functions. Choose one for the task:

- Mean-Squared-Error Loss
- Cross-Entropy Loss
- Ranking Loss

这里 Suppose the Loss function is MSE Loss, then

$$L = \frac{1}{2}(y - \hat{y})^2$$

> 这里加上常数 $\frac{1}{2}$ 是数学技巧，方便反向传播（求导）时消去常数因子. 它不会影响最终的训练结果，因为常数不会改变梯度方向.

Let's find $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial b_1}$, $\frac{\partial L}{\partial w_2}$.

7.5.2 前向传播

> Forward propagation

7.5.3 反向传播

> Backward propagation

7.5.4 Softmax

> Softmax: before computing losses

Softmax 是一种将向量中的实数压缩为概率分布的函数.

> 类似一种归一化处理，用 e 指数的形式放大微小波动.

给定向量 $\mathbf{z} = [z_1, \ldots, z_k]$，Softmax 定义如下：

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \qquad \text{for } i = 1, 2, \ldots, k$$

其中，

σ：softmax

$\mathbf{z}$：input vector

k：number of classes in the multi-class classifier

The softmax activation layer is usually used in the last layer of neural network to turn the numerical output into values in the range $[0, 1]$.

Softmax values can be considered as a probability distribution given the output sums up to 1.

7.6 优化方法

> Optimization Alternatives Rather than Simple Gradient Descent

7.6.1 随机梯度下降

> Stochastic Gradient Descent (SGD)

待完成.

7.6.2 优化器

> Other Optimizers: Adaptively Decide the Next Step to Take

相对于传统的 SGD，现代优化器会根据历史梯度、自适应调整学习率，用更聪明的方式决定「下一步该怎么走」.

7.6.3 学习率调度器

> Learning Rate Scheduler

You may notice that some optimizers just adaptively adjust the learning rate. In fact, you might design a specific learning rate adjust strategy by yourself.

优化器（Optimizer）和学习率调度器（Learning Rate Scheduler）是两个不同的机制，但它们协同工作来帮助神经网络更好地收敛. 优化器负责执行更新，调度器负责改变学习率，两者经常搭配使用.

Lecture 8 过拟合

> Overfitting

过拟合：训练误差（经验误差）低，测试误差（泛化误差）高.

原因可能包括：模型复杂度高，数据量小，噪声存在.

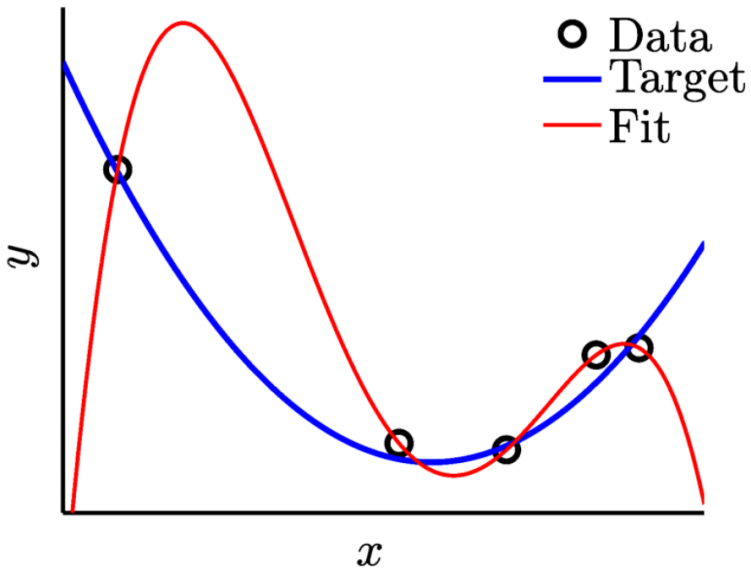
8.1 多项式函数

> Polynomial function

$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

8.2 泛化能力差

> Bad Generalization



Regression for $x \in \mathbb{R}$ with $N = 5$ samples.

> 对实数域中的变量 x 进行回归，但总共只有 5 个训练样本.

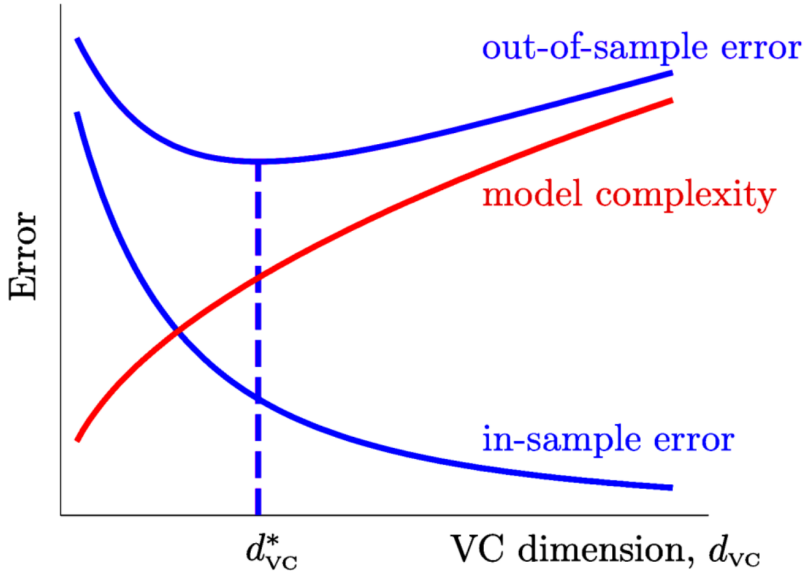
Target polynomial $f(x)$, $n = 2$ polynomial

label $y_n = f(x_n) +$ very small noise

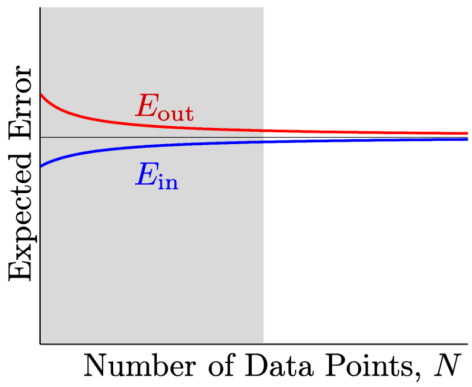
真正的目标函数 $f(x)$ 是一个二阶多项式，每个样本点的标签 y_n 是目标函数的输出再加上一点点噪声.

We fit the five points with red curve（四阶），可以看到，红线几乎完美穿过所有的黑点 —— 训练误差非常小，但它在数据点之外的区域波动很大，偏离了真实的蓝线. 这就是模型过于复杂（维度过高）以及样本量过少导致的过拟合.

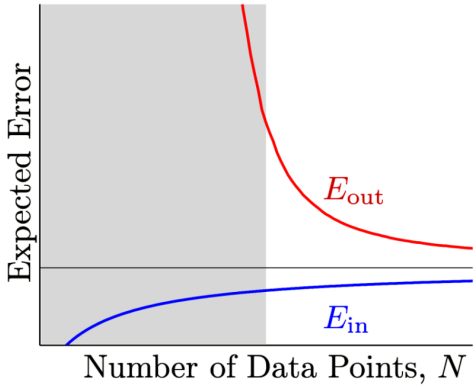
$E_{in} \approx 0, E_{out} \gg 0$，说明即使训练误差很低，也不代表模型好，重要的是在拟合能力和泛化能力之间作出权衡.



The complex model has much larger generalization error for small N; Simple model might win in E_{out} when N is small



A simple model



A complex model

8.3 确定性噪声

> Deterministic Noise

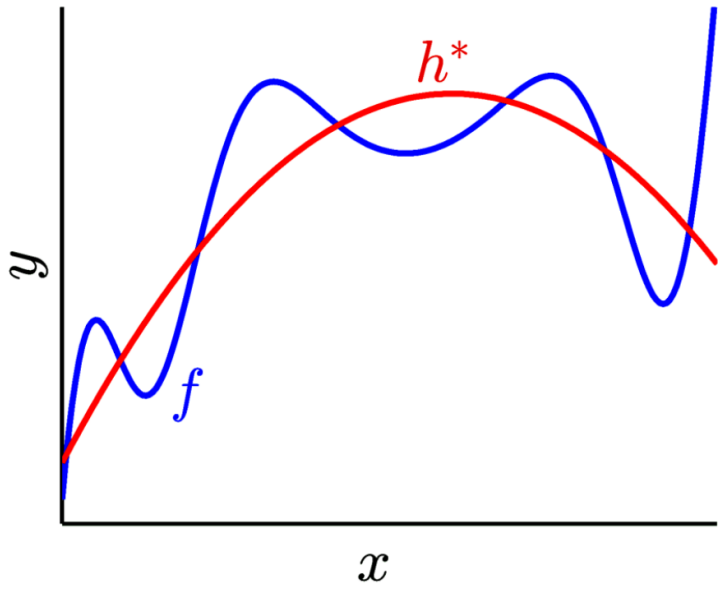
确定性噪声指目标函数 $f(x)$ 中，假设空间 H 无法表示或学习到的那一部分. 用公式表示为

$$\text{Deterministic Noise} = f(x) - h^*(x)$$

其中，

$f(x)$：真实的目标函数

$h^*(x)$：在假设空间 H 中表现最好的函数（即能学到的最接近 f 的函数）



与随机噪声的区别

特性	确定性噪声（Deterministic Noise）	随机噪声（Stochastic Noise）
来源	模型表达能力有限	数据本身不可预测性
是否依赖假设空间 H	✅ 是	❌ 否
对于固定输入 x 是否固定	✅ 是	❌ 否（每次可能不同）

确定性噪声不是由数据的随机性引起的，而是因为你的模型太简单，**无法表达目标函数的复杂性**.

它是学习误差的一部分，尤其当你的假设空间不足够强大时.

想减少确定性噪声的方法包括：扩大假设空间、使用更复杂的模型（比如从线性模型切换到非线性模型）.

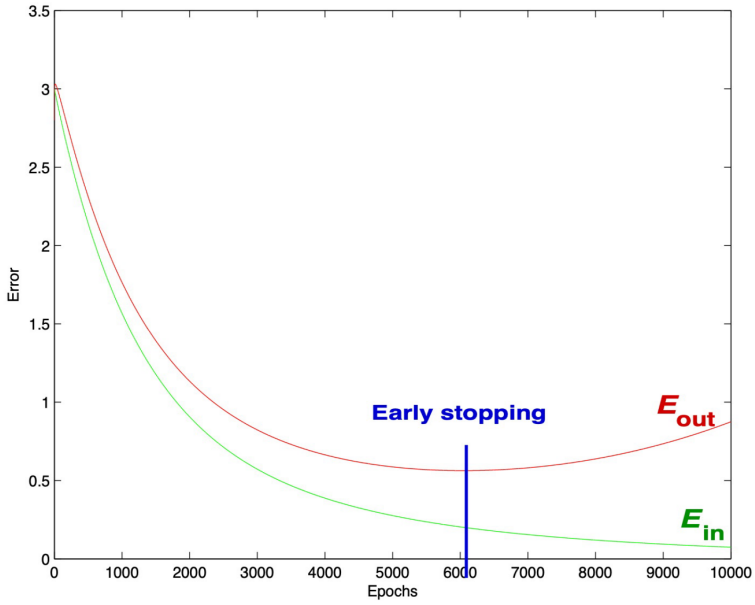
8.4 处理方案

> Dealing with the Overfitting

Regularization（正则化）

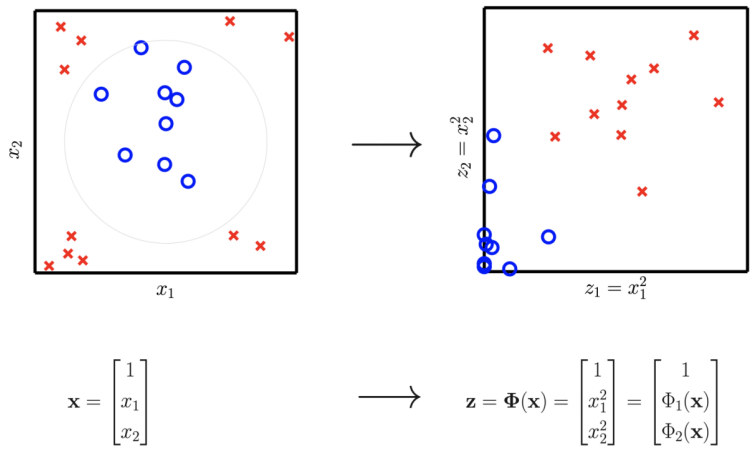
Model Selection and Validation（模型选择与验证）

early stopping



8.5 多项式变换

When Linear Separable is not OK, Some Transformation might work.



多项式变换：将原始输入特征 $\mathbf{x}$ 映射到一个更高维的特征空间，从而使线性模型能够拟合非线性关系.

我们用 $\Phi_d(x)$ 表示将输入 $\mathbf{x} = (x_1, x_2)$ 进行 d 次多项式变换后得到的新特征向量.

原始线性特征：$\Phi_1(x) = (1, x_1, x_2)$

加入二次项：$\Phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

加入三次项：$\Phi_3(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_3^2)$

加入四次项...（维度大幅增加）

- 更高阶的多项式能更好地拟合训练数据.
- 每提升一次多项式阶数，特征维度会**快速增长**（指数级增长）
- 这会带来计算复杂性和过拟合风险.

Approximation-generalization trade-off: Higher degree gives lower (even 0) E_{in} but worse generalization.

We here note the transformation from X space to Z space, $Z = \Phi(X)$.

在新的空间进行线性回归. 原数据矩阵如下：

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}$$

> 见 6.3.1 矩阵表示 ① 数据矩阵.

变换后，数据矩阵如下：

$$Z = \Phi(X) = \begin{bmatrix} \Phi(\mathbf{x}_1)^T \\ \Phi(\mathbf{x}_2)^T \\ \vdots \\ \Phi(\mathbf{x}_N)^T \end{bmatrix}$$

> 因为对于每一个输入 $\mathbf{x}_n$，有 $\mathbf{z}_n = \Phi(\mathbf{x}_n)$.

模型预测值变为

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \mathbf{w}^T\mathbf{z}_1 \\ \mathbf{w}^T\mathbf{z}_2 \\ \vdots \\ \mathbf{w}^T\mathbf{z}_N \end{bmatrix} = Z\mathbf{w}$$

写出训练误差，线性回归的目标是最小化**均方误差**（MSE）：

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

使用矩阵形式可以重写为：

$$\begin{aligned} E_{in}(\mathbf{w}) &= \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \\ &= \frac{1}{N}\|Z\mathbf{w} - \mathbf{y}\|_2^2 \\ &= \frac{1}{N}(\mathbf{w}^T Z^T Z\mathbf{w} - 2\mathbf{w}^T Z^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \end{aligned}$$

同样用正规方程求解，得到最优解：

$$\mathbf{w}_{\text{lin}} = (Z^T Z)^{-1} Z^T\mathbf{y}.$$

> 见 6.3 线性回归.

Z 空间中的线性回归预测向量：

$$\hat{\mathbf{y}} = Z\mathbf{w}_{\text{lin}}$$

> 本质上还是对输入 $\mathbf{x}$ 对应的标签 y 进行预测.

Lecture 9 正则化

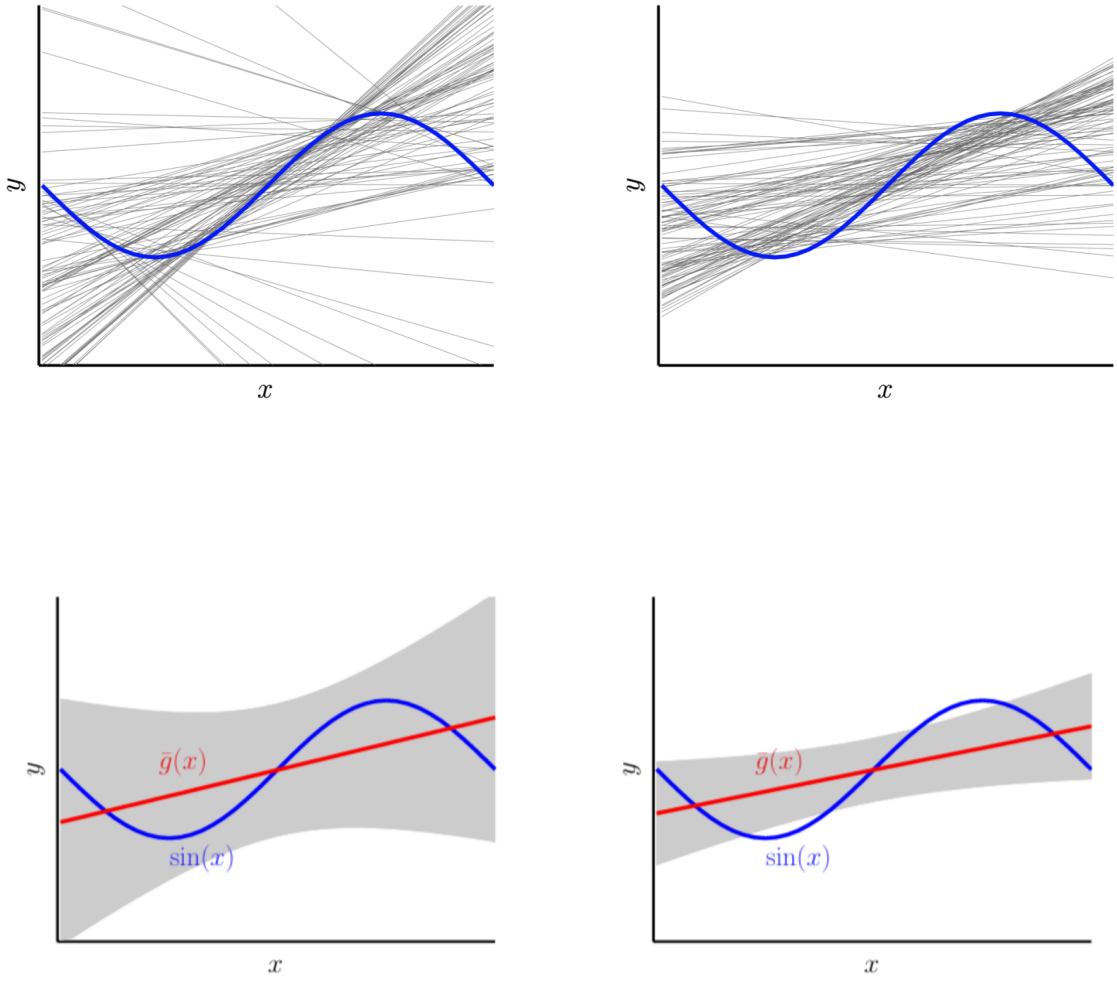
> Regularization

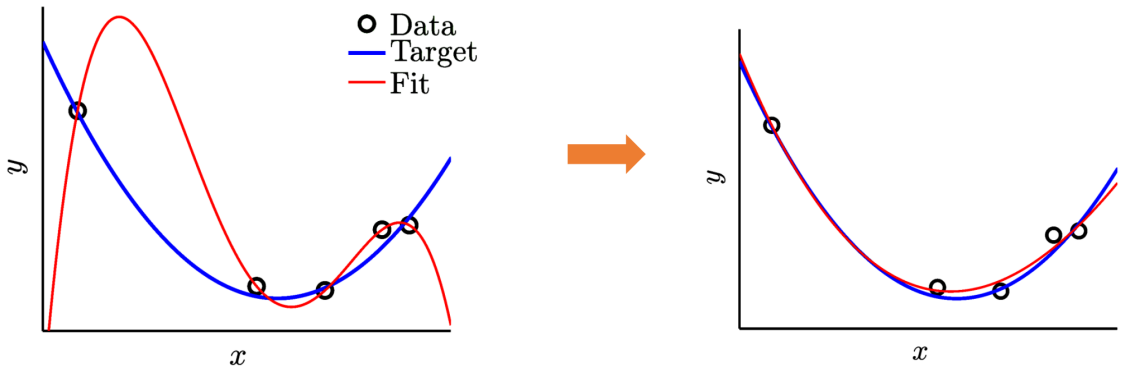
正则化：通过对模型进行限制，防止过拟合，提升泛化能力.

name history: function approximation for ill-posed problems（病态问题）

直接展示效果.

以线性回归模型为例，左图：无正则化；右图：有正则化.





9.1 正则化方法

核心思想：在训练过程中对模型的复杂度进行**惩罚**，让模型不要太“自由”. 而模型的复杂度是可以通过约束其权重来控制的.

① 硬约束

> Hard Constraints

直接把高阶项权重设置为 0.

Hypothesis w in $\mathcal{H}_{10}$

$$\mathcal{H}_{10} = \left\{ h(x) = w_0 + w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + \cdots + w_{10}\Phi_{10}(x) \right\}$$

Hypothesis w in $\mathcal{H}_2$

$$\mathcal{H}_2 = \left\{ \begin{array}{c} h(x) = w_0 + w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + \cdots + w_{10}\Phi_{10}(x) \\ \text{such that: } w_3 = w_4 = \cdots = w_{10} = 0 \end{array} \right\}$$

虽然可以防止过拟合，但缺点更多：

缺点	原因
不灵活	完全禁止某些特征或参数
难以优化	需要满足等式或不等式约束，优化复杂
可解释性差	很难平衡拟合能力与模型复杂度

② 软约束

> Soft Constraints

$$\mathcal{H}_C = \left\{ \begin{array}{c} h(x) = w_0 + w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + \cdots + w_{10}\Phi_{10}(x) \\ \text{such that: } \sum_{q=0}^{10} w_q^2 \leq C \end{array} \right\}$$

From the perspective of VC dimension, H_C is not larger than H_{10}.

> $H_C \subseteq H_{10}$.

虽然从约束的数学形式上看，所有项都被同等惩罚，但实际操作中，因为高阶项更容易造成不稳定，优化时更容易被压制.

不是说“不能用高阶多项式”，而是说“用了高阶多项式就要付出代价（惩罚项）”.

实际训练中，一般采用软约束（看起来就更靠谱）.

优点	说明
✅ 易于实现	只需在损失函数加一项
✅ 可调节	用超参数 λ 控制约束强度
✅ 更灵活	不强制限制模型，而是“引导”它简单化
✅ 更稳定	可以连续优化，易于梯度下降

9.2 正则化线性回归

> Regularized Regression Problem

原版优化问题：

$$\min : E_{in}(\mathbf{w}) = \frac{1}{N}(Z\mathbf{w} - \mathbf{y})^T(Z\mathbf{w} - \mathbf{y})$$

加入软约束 $\mathbf{w}^T\mathbf{w} \leq C$ 后，正则化版本的优化问题变为：

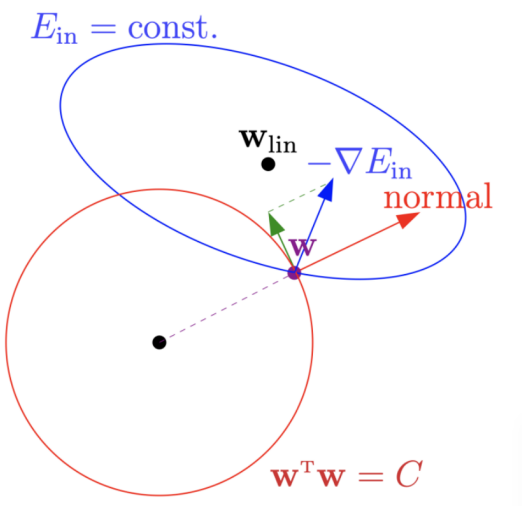
$$\begin{aligned} \min : E_{in}(\mathbf{w}) &= \frac{1}{N}(Z\mathbf{w} - \mathbf{y})^T(Z\mathbf{w} - \mathbf{y}) \\ \text{subject to} : \mathbf{w}^T\mathbf{w} &\leq C \end{aligned}$$

> subject to 是 “满足以下约束条件” 的意思.

9.2.2 L2 正则化

如何求解带有 L2 正则化的线性回归解，也叫 Ridge Regression 或 正则化最小二乘问题.

$$\mathbf{w}_{\text{lin}} = (\mathrm{Z}^{\mathrm{T}}\mathrm{Z})^{-1}\mathrm{Z}^{\mathrm{T}}\mathbf{y}$$



> 这里红色圈的中心是原点.

如果 $\mathbf{w}_{\text{lin}}$ 本身已满足约束条件（在红色圈里），优化结果即 $\mathbf{w}_{\text{lin}}$，和没有正则化的结果一样. 我们关心的是 $\mathbf{w}_{\text{lin}}$ 落在红圈外的情况.

当 $\mathbf{w}_{\text{lin}}$ 不满足约束，我们要重新求解满足约束的最优解 $\mathbf{w}_{\text{REG}}$. 分析可得：

$$-\nabla E_{in}(\mathbf{w}_{\text{REG}}) \propto \mathbf{w}_{\text{REG}}$$

因为

- 如果梯度不是指向边界法向量方向，我们还可以继续梯度下降，在边界上沿切向行走，进一步下降误差.
- 但既然是“最优点”，就说明已经没法在边界上继续下降了；
- 所以梯度方向只能是和约束边界法向量一致（或反向）.

9.2.3 拉格朗日乘子法

> Lagrange multipliers

考虑一个最优化问题：

$$\min_{\mathbf{w}} f(\mathbf{w}) \quad \text{subject to} \quad g(\mathbf{w}) = 0$$

> 这里 $g(\mathbf{w}) = 0$ 是一个等式约束，例如 $\mathbf{w}^T\mathbf{w} = C$.

在约束边界上的最优点，目标函数的梯度 ∇f 与约束函数的梯度 ∇g 是线性相关的. 原因：优化过程中，只能在约束边界上移动（切向移动），当找到最优点，意味着 f 在切向上没有下降可能，即 ∇f 与约束边界在该最优点的切向正交，即 ∇f 与约束边界在该最优点的法向共线. 约束边界在该最优点的法向即 $\pm\nabla g$ 方向. 用公式表达就是

$$\nabla f(\mathbf{w}^*) + \lambda\nabla g(\mathbf{w}^*) = 0$$

其中 λ 是拉格朗日乘子.

构造拉格朗日函数

$$L(\mathbf{w}, \lambda) = f(\mathbf{w}) + \lambda \cdot g(\mathbf{w})$$

最优解 $\mathbf{w}^*$ 满足

$$\begin{cases} \nabla_{\mathbf{w}} L(\mathbf{w}^*, \lambda) = 0 \\ \nabla_{\lambda} L(\mathbf{w}^*, \lambda) = 0 \end{cases}$$

> 第一个式子：最优点的 “最优” 推出的性质；
>
> 第二个式子：最优点仍满足约束条件.

求解该方程组即可得到 $\mathbf{w}^*$.

9.2.4 KKT 条件

> Karush-Kuhn-Tucker conditions
>
> 拉格朗日乘子法的推广，适用于**不等式约束**的问题

现在，考虑不等式约束 $g(\mathbf{w}) \leq 0$. 即考虑最优化问题

$$\min_{\mathbf{w}} f(\mathbf{w}) \quad \text{subject to} \quad g(\mathbf{w}) \leq 0$$

引入拉格朗日乘子法的推广：KKT 条件.

构造拉格朗日函数

$$L(\mathbf{w}, \lambda) = f(\mathbf{w}) + \lambda \cdot g(\mathbf{w})$$

不等式约束下，最优点 $\mathbf{w}^*$ 必须满足四个条件：

① 梯度条件

> stationarity

与等式约束的拉格朗日乘子法一样，需要满足

$$\nabla f(\mathbf{w}^*) + \lambda \nabla g(\mathbf{w}^*) = 0$$

② 可行性条件

$$g(\mathbf{w}^*) \leq 0$$

> 即最优点也要满足约束.

③ 乘子非负

$$\lambda \geq 0$$

④ 互补松弛性

$$\lambda \cdot g(\mathbf{w}^*) = 0$$

i) 如果 $g(\mathbf{w}^*) < 0$，即约束不紧（最优点不在边界），则乘子 $\lambda = 0$.

> 当约束“不紧”时，它对最优解没有影响，解等于无约束解
>
> 如果约束在最优点处 **不是“刚好等于”边界**（即不紧），
>
> 就相当于这个约束**没有“起作用”**，
>
> 那么最优解就是**原始无约束问题的解**.

ii) 如果 $g(\mathbf{w}^*) = 0$（最优点在边界上），则 λ 可以非零.

通过对互补松弛性进行讨论（约束是否紧），可以解出最优解 $\mathbf{w}^*$.

9.2.5 $\mathbf{w}_{\mathrm{REG}}$ 解法

回到 L2 正则的线性回归最优化问题：

$$\begin{aligned} \min &: E_{in}(\mathbf{w}) = \frac{1}{N}(Z\mathbf{w} - \mathbf{y})^T(Z\mathbf{w} - \mathbf{y}) \\ \text{subject to} &: \mathbf{w}^T\mathbf{w} \leq C \end{aligned}$$

构造拉格朗日函数：

$$L(\mathbf{w}, \lambda) = E_{in}(\mathbf{w}) + \frac{\lambda}{N}(\mathbf{w}^T\mathbf{w} - C)$$

> 注意，这里用 $\frac{\lambda}{N}$ 代替标准解法中的拉格朗日乘子，能让最后的解更好看.

根据 KKT 条件，最优解 $\mathbf{w}_{\mathrm{REG}}$ 满足：

$$\begin{cases} \nabla_{\mathbf{w}} L(\mathbf{w}_{\mathrm{REG}}, \lambda) = 0 \\ \mathbf{w}_{\mathrm{REG}}^T\mathbf{w}_{\mathrm{REG}} - C \leq 0 \\ \frac{\lambda}{N} \geq 0 \\ \frac{\lambda}{N} \cdot (\mathbf{w}_{\mathrm{REG}}^T\mathbf{w}_{\mathrm{REG}} - C) = 0 \end{cases}$$

讨论：

i) $\mathbf{w}_{\mathrm{REG}}^T\mathbf{w}_{\mathrm{REG}} - C < 0$（约束不紧）

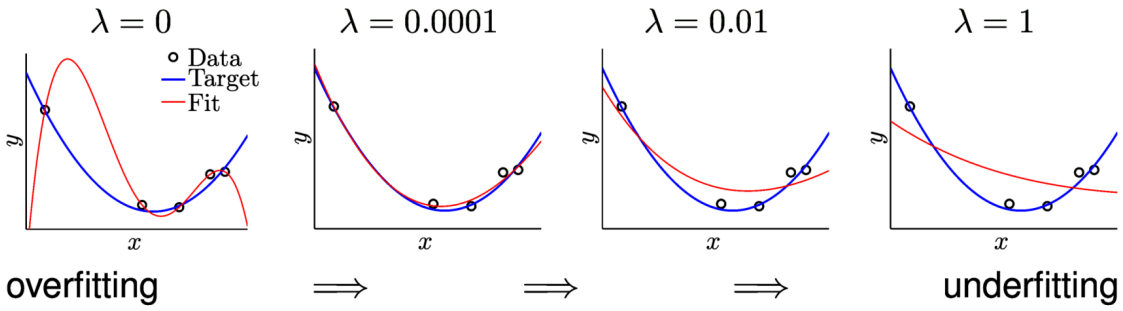
$$\frac{\lambda}{N} = 0, \mathbf{w}_{\mathrm{REG}} = \mathbf{w}_{\mathrm{lin}} = (Z^TZ)^{-1}Z^T\mathbf{y}.$$

> 相当于没有约束.
>
> 注意，这里在真实题目中要验证 $\mathbf{w}_{\mathrm{lin}}$ 是否满足约束. 如果不满足，要舍去.

ii) $\mathbf{w}_{\mathrm{REG}}^T\mathbf{w}_{\mathrm{REG}} - C = 0$

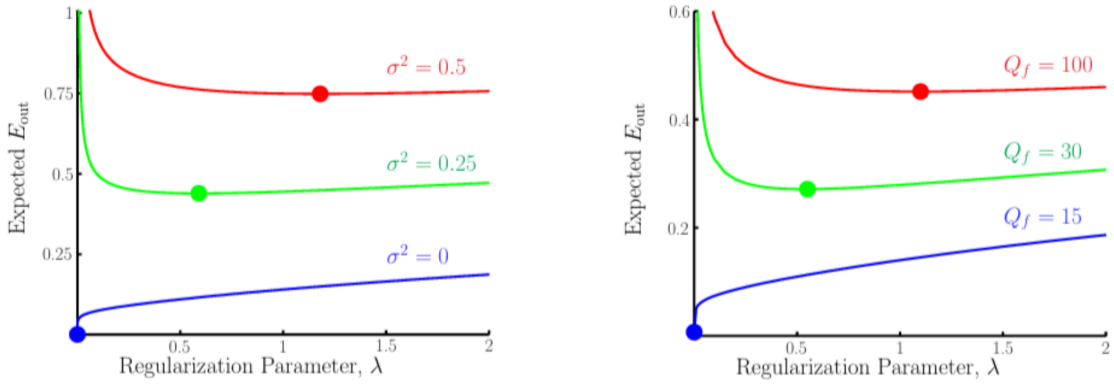
$$
\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}_{\text{REG}}, \lambda) &= 0 \\
\frac{2}{N}(Z^T Z \mathbf{w}_{\text{REG}} - Z^T \mathbf{y}) + \frac{2\lambda}{N}\mathbf{w}_{\text{REG}} &= 0 \\
(Z^T Z + \lambda I)\mathbf{w}_{\text{REG}} &= Z^T \mathbf{y} \\
\mathbf{w}_{\text{REG}} &= (Z^T Z + \lambda I)^{-1} Z^T \mathbf{y}
\end{aligned}
$$

注意，这里 λ 和 C 是一起变化的（$C = \mathbf{w}_{\text{REG}}^T \mathbf{w}_{\text{REG}}$），其实我们也不清楚 C 设置多少比较合适，所以直接去试不同的 λ 看看泛化效果（准备一个测试集），然后得到一个比较满意的 λ，它对应的 C 就是比较合适的 C（但这时候 C 已经无关紧要了，我们已经获得了正则化的模型）.

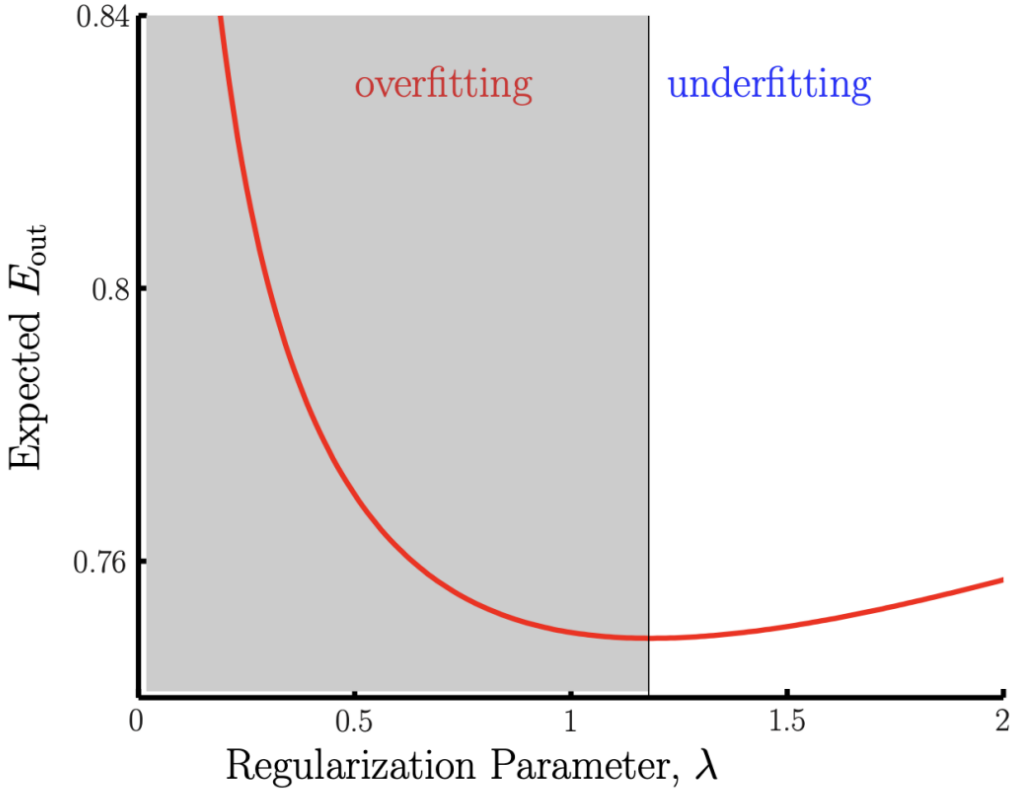


9.2.6 λ 与泛化性能

正则化参数 λ 如何影响模型泛化性能



- 噪声越大 / 目标函数越复杂，整体测试误差曲线越高（因为更难学）
- $\lambda = 0$ 时，无正则化，模型可能过拟合噪声，导致 E_{out} 高.
- 每条曲线有一个最优点，对应最佳 λ



9.3 验证集

> Model Selection by Validation

$$E_{\text{in}}(h) \quad \uparrow \quad \underbrace{\mathcal{D}}_{\text{size } N} \quad \downarrow \quad g_m = \mathcal{A}_m(\mathcal{D}) \qquad \rightarrow \qquad \underbrace{\mathcal{D}_{\text{train}}}_{\text{size } N-K} \quad \downarrow \quad g_m^- = \mathcal{A}_m(\mathcal{D}_{\text{train}}) \qquad \cup \qquad E_{\text{val}}(h) \quad \uparrow \quad \underbrace{\mathcal{D}_{\text{val}}}_{\text{size } K}$$

Validation Set

在训练多个模型（或同一模型的不同超参数）之后，我们希望选出最好的一个. 与原本的完全使用训练集来评估模型相比，把数据集分为两份，一份训练集，一份验证集，可以避免过拟合.

9.4 留一交叉验证

考虑极端情况：Leave-One-Out Cross Validation（LOOCV）

> K = 1 的交叉验证

是交叉验证的一种极端形式

把数据集分成 N 份（每次只留出一个样本当验证集，其余 $N-1$ 个样本用来训练）

重复 N 次，每个样本都被“留出一次”作为验证

$$E_{\text{loocv}}(\mathcal{H},\mathcal{A})=\frac{1}{N}\sum_{n=1}^{N}e_n=\frac{1}{N}\sum_{n=1}^{N}\text{err}(g_n^-(\mathbf{x}_n),y_n)$$

- g_n^-：用除了第 n 个样本以外的 $N-1$ 个训练数据训练出的模型.
- $\text{err}(g_n^-(\mathbf{x}_n),y_n)$：这个模型在第 n 个样本上的预测误差.
- 最终结果对所有 N 个样本的误差做平均.

通过 LOOCV 得到的 E_{loocv} 的期望很接近泛化误差的期望.

缺点：计算开销很大.

解决方案：V-fold Cross Validation

9.5 V-fold 交叉验证

> V-fold Cross Validation

用 V 个折叠划分数据，而不是 N 个. 每次留出 $\frac{1}{V}$ 的数据作为验证，其余作为训练. 重复 V 次（每个折叠做一次验证集），最后平均 V 次验证误差.

Lecture 10 生成式学习算法

> Generative Learning Algorithm

10.1 贝叶斯统计

10.1.1 贝叶斯法则

$$p(y|x)=\frac{p(x|y)p(y)}{p(x)}$$

10.1.2 最大似然估计

> MLE

最大似然估计：选择使训练数据概率最大的参数.

例 1：线性回归的最优解（最小二乘解）

$$\mathbf{w}_{\text{lin}} = \arg\min_{\mathbf{w}} (\mathbf{y} - X\mathbf{w})^T (\mathbf{y} - X\mathbf{w})$$

等价于高斯噪声下的最大似然估计解.

证明：

- 输入矩阵 $X \in \mathbb{R}^{n \times d}$（每行一个样本）
- 输出向量 $\mathbf{y} \in \mathbb{R}^n$

假设模型为

$$\mathbf{y} = X\mathbf{w} + \epsilon$$

其中噪声向量 $\epsilon \sim N(\mathbf{0}, \sigma^2 I)$.

给定 X 和参数 $\mathbf{w}$，输出 $\mathbf{y}$ 的条件概率为

$$p(\mathbf{y}|X, \mathbf{w}) = N(X\mathbf{w}, \sigma^2 I)$$

最大似然估计就是最大化这个概率：

$$\hat{\mathbf{w}}_{\text{MLE}} = \arg\max_{\mathbf{w}} p(y|X, \mathbf{w})$$

多元高斯分布的概率密度函数为

$$p(y|X, \mathbf{w}) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - X\mathbf{w}\|^2 \right)$$

取对数（log-likelihood）

$$\log p(y|X, \mathbf{w}) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\mathbf{y} - X\mathbf{w}\|^2$$

> 或者取负对数，然后最小化. NLL is the the negative log likelihood.

第一项和 $\mathbf{w}$ 无关，因此最大化 log-likelihood 等价于最小化

$$\|\mathbf{y} - X\mathbf{w}\|^2$$

即回到了线性回归的优化内容. 因此二者等价.

例 2：多项分布

假设掷一个 K 面骰子 N 次，每次投掷的结果 $Y_n = \{1, \ldots, K\} \sim Cat(\theta)$. 数据集即所有结果 $D = \{y_n : n = 1 : N\}$

其中，$\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ 是我们要估计的参数，表示每一面朝上的概率.

设 N_k 是第 k 面骰子出现的次数. 构建似然:

$$L(\theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

取 NLL（负对数似然），我们要最小化

$$NLL(\theta) = -\sum_{k=1}^{K} N_k \log \theta_k$$

约束条件:

$$\begin{cases} \sum_{k=1}^{K} \theta_k = 1 \\ \theta_k \geq 0 \text{ for all } k \end{cases}$$

解法：拉格朗日乘子法（Lagrange multipliers）

构造拉格朗日函数

$$L(\theta, \lambda) = -\sum_{k=1}^{K} N_k \log \theta_k + \lambda \left(\sum_{k=1}^{K} \theta_k - 1 \right)$$

对每个 θ_k 求偏导，并设为 0:

$$\frac{\partial L}{\partial \theta_k} = -\frac{N_k}{\theta_k} + \lambda = 0 \Rightarrow \theta_k = \frac{N_k}{\lambda}$$

代入约束，求解 λ

$$\sum_{k=1}^{K} \theta_k = \sum_{k=1}^{K} \frac{N_k}{\lambda} = \frac{N}{\lambda} = 1 \Rightarrow \lambda = N$$

最大似然估计:

$$\hat{\theta}_k = \frac{N_k}{N} \text{ for } k = 1, 2, \ldots, K$$

> 这个解显然满足第二个约束.

也就是说，第 k 类的概率估计等于它在样本中出现的频率（符合常识）.

10.1.3 最大后验估计

> Maximum A Posteriori Estimation, 简称 MAP

如果已知（或者自己假设）参数先验，也可以使用 MAP 估计:

$$\begin{aligned} \mathbf{w}_{\text{MAP}} &= \arg\max_{\mathbf{w}} \ln p(\mathbf{w}|\mathbf{y}, X) \\ &= \arg\max_{\mathbf{w}} \ln \frac{p(\mathbf{y}|\mathbf{w}, X) p(\mathbf{w})}{p(\mathbf{y}|X)} \quad (\text{参数 } \mathbf{w} \text{ 和输入 } X \text{ 独立}) \\ &= \arg\max_{\mathbf{w}} \ln p(\mathbf{y}|\mathbf{w}, X) + \ln p(\mathbf{w}) - \ln p(\mathbf{y}|X) \\ &= \arg\max_{\mathbf{w}} \ln p(\mathbf{y}|\mathbf{w}, X) + \ln p(\mathbf{w}) \end{aligned}$$

其中，$p(\mathbf{y}|X)$ 是边际似然（所有 $\mathbf{w}$ 的先验加权积分得到，在该优化问题中不随 $\mathbf{w}$ 变化，可忽略该项）. 所以，我们只要最大化前两项，即在 MLE 的基础上加一个先验项.

> 实际上大多数问题，边际似然根本算不出来. 所以"这项可忽略"这个事实是很重要的.

例：线性回归

对参数设置一个高斯先验

$$p(\mathbf{w}) = N(\mathbf{w}|0, \tau^2 I)$$

有

$$\begin{aligned}
\mathbf{w}_{\text{MAP}} &= \arg\max_{\mathbf{w}} \ln p(\mathbf{y}|\mathbf{w}, X) + \ln p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} -\frac{1}{2\sigma^2}\|\mathbf{y} - X\mathbf{w}\|^2 - \frac{1}{2\tau^2}\|\mathbf{w}\|^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{\sigma^2}\|\mathbf{y} - X\mathbf{w}\|^2 + \frac{1}{\tau^2}\|\mathbf{w}\|^2 \\
&= \arg\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|^2 + \frac{\sigma^2}{\tau^2}\|\mathbf{w}\|^2
\end{aligned}$$

记 $\frac{\sigma^2}{\tau^2} = \lambda$，则 MAP 等价于正则化线性回归的优化问题. 结果为

$$W_{\text{MAP}} = \mathbf{w}_{\text{REG}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

贝叶斯学派对正则化参数 λ 的解释：先验信念

- τ^2 越小，先验越强，λ 越大，容易欠拟合.

> 贝叶斯学派对正则化参数的可解释性更强. 频率学派完全是基于数学技巧（即拉格朗日乘子法解约束问题）引入的正则化参数，没有说明这个参数的含义.

10.2 多元正态分布

> Multivariate Normal Distribution

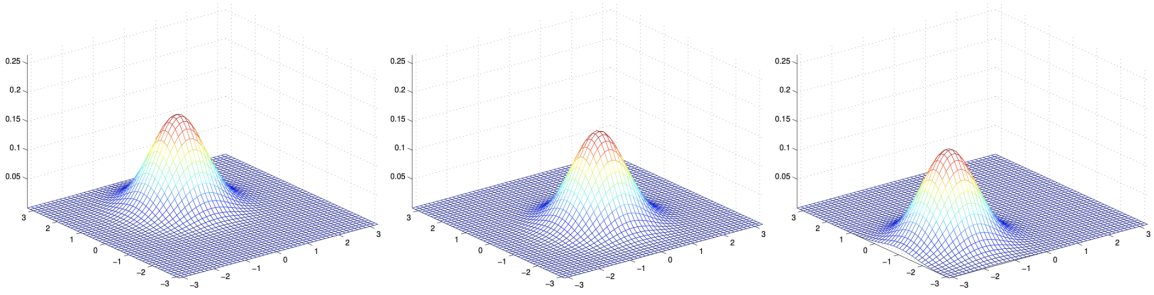
多元正态分布是一类在 d 维空间中的概率分布，也叫多元高斯分布，记作 $N(\mu, \Sigma)$. 其中，

- $\mu \in \mathbb{R}^d$：均值向量（mean vector）
- $\Sigma \in \mathbb{R}^{d\times d}$：协方差矩阵（covariance matrix）

概率密度函数：

$$p(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$

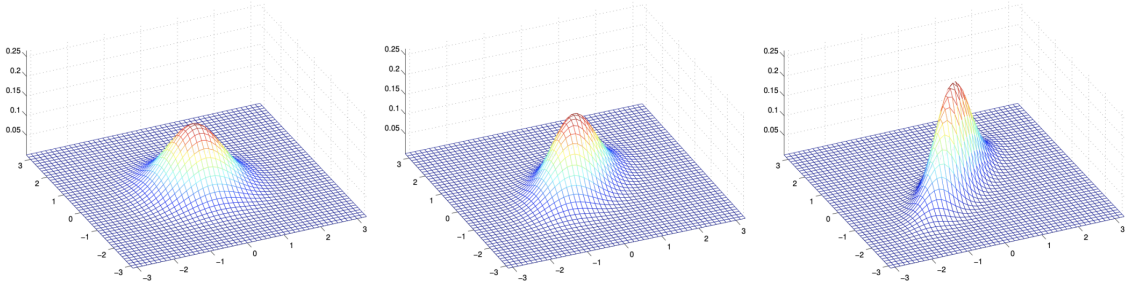
均值向量控制整体位置.



$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \qquad \mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$$

协方差矩阵控制各维度之间的相关性，和每个维度的方差.

> 下图保持各维度方差不变，调整相关性.



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

协方差矩阵定义

设有一个 d 维随机向量

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_d \end{bmatrix}$$

其协方差矩阵 $\Sigma \in \mathbb{R}^{d \times d}$ 定义为

$$\Sigma = \mathbb{E}[(\mathbf{X} - \mu)(\mathbf{X} - \mu)^T]$$

展开后，Σ 是一个对称矩阵：

$$\Sigma = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) & \cdots & Cov(X_1, X_d) \\ Cov(X_2, X_1) & Var(X_2) & \cdots & Cov(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_d, X_1) & Cov(X_d, X_2) & \cdots & Var(X_d) \end{bmatrix}$$

其中，$Cov(X_i, X_j)$ 表示两个随机变量 X_i, X_j 的协方差，定义为

$$Cov(X_i, X_j) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

> $Cov(X_i, X_j) > 0$：正相关.
>
> $Cov(X_i, X_j) = 0$：负相关.
>
> $Cov(X_i, X_j) > 0$：不存在线性关系（可能独立，也可能非线性相关）.

10.3 生成式 vs. 判别式

> Generative vs. Discriminative classifier

`10.1.3 最大后验估计` 是给参数一个先验，然后 MAP 估计，得到参数的最优解后，可以对新的输入进行判定.

生成式模型（以高斯判别分析为例）则是对标签 y 的先验以及 $p(\mathbf{x}|y)$ 进行建模，然后用 MAP 选出给定参数 $\theta = (\theta_1, \ldots, \theta_C)$（一共 C 类标签）和输入 $\mathbf{x}$ 的条件下，最有可能的 y（标签后验），并作为判定结果.

> 其中 $\theta_c = (\pi_c, \mu_c, \Sigma_c)$，包含了 y 的先验参数以及 $p(\mathbf{x}|y)$ 的参数. 这些参数是待优化的.

这里在 MAP 的时候要用贝叶斯，因此我们会对 $p(\mathbf{x}|y)$ 进行建模. 于是，“生成式” 的意思就是，标签 y 决定特征 $\mathbf{x}$ 的生成方式. 不同的 y 标签在不同区域生成 $\mathbf{x}$.

优化方法：最大似然（见 `10.3.3 GDA 优化`）.

在优化得到一个最优参数后，我们可以用 $p(\mathbf{x}|y)$ 和 $p(y)$ 不断生成新数据. 这是判别式模型无法做到的.

数学表示

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|y = c; \boldsymbol{\theta})p(y = c; \boldsymbol{\theta})}{\sum_{c'} p(\boldsymbol{x}|y = c'; \boldsymbol{\theta})p(y = c'; \boldsymbol{\theta})}$$

Prior over class c

Class conditional density

分类器类型	建模方式	举例	优点	缺点
生成式	建模 $p(\mathbf{x}\|y)$ 和 $p(y)$，再求 $p(y\|\mathbf{x})$	高斯判别分析（GDA）、朴素贝叶斯	可用于生成数据；能处理缺失数据	对模型假设敏感（如特征独立性）
判别式	直接建模 $p(y\|\mathbf{x})$ 或 决策边界	逻辑回归、SVM、神经网络	分类性能更强，灵活性更高	没有建模 $\mathbf{x}$ 的分布，不能生成数据

10.4 几种生成式模型

10.4.1 高斯判别分析

> Gaussian Discriminant Analysis model, GDA
>
> 对 $p(\mathbf{x}|y)$ 进行高斯建模的分类器. 这里标签/类别是离散的，因此标签的先验分布就是 PMF，离散的 π_c

We consider a generative classifier where the class conditional densities are multivariate Gaussians:

$$p(\mathbf{x}|y=c,\theta)=N(\mathbf{x}|\mu_c,\Sigma_c)$$

于是由贝叶斯，class posterior 满足

$$p(y=c|\mathbf{x},\theta)\propto\pi_c N(\mathbf{x}|\mu_c,\Sigma_c)$$

> 其中 $\pi_c=P(y=c)$ 是标签的先验概率.

取对数可以得到

$$\log p(y=c\mid\mathbf{x};\theta)=\log\pi_c-\tfrac{1}{2}\log|2\pi\Sigma_c|-\tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c)+\text{const}$$

This is called the **discriminant function**（判别函数）, e.g., the decision boundary between any two classes will be a quadratic function of x

> （相邻两类的）决策边界是二次方程.

具体推导（Assignment 5 Q3）

By Bayes rule,

$$p(y=c\mid\mathbf{x};\theta)=\frac{\pi_c\mathcal{N}(\mathbf{x}\mid\mu_c,\Sigma_c)}{\sum_{c'}\pi_{c'}\mathcal{N}(\mathbf{x}\mid\mu_{c'},\Sigma_{c'})}.$$

Take log, we have

$$\log p(y=c\mid\mathbf{x};\theta)=\log\pi_c+\log\mathcal{N}(\mathbf{x}\mid\mu_c,\Sigma_c)-\log Z,$$

where $Z=\sum_{c'}\pi_{c'}\mathcal{N}(\mathbf{x}\mid\mu_{c'},\Sigma_{c'})$.

The PDF of Multivariate Normal Distribution is

$$\begin{aligned}\mathcal{N}(\mathbf{x}\mid\mu_c,\Sigma_c)&=\frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{c})^T\Sigma_c^{-1}(\mathbf{x}-\mu_c)\right)\\\Rightarrow\log\mathcal{N}(\mathbf{x}\mid\mu_c,\Sigma_c)&=-\tfrac{d}{2}\log(2\pi)-\tfrac{1}{2}\log|\Sigma_c|-\tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c),\end{aligned}$$

Hence, the discriminant function is

$$\log p(y=c\mid\mathbf{x};\theta)=\log\pi_c-\tfrac{1}{2}\log|\Sigma_c|-\tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c)+\text{const}.$$

Here $\text{const}=-\frac{d}{2}\log(2\pi)-\log\sum_{c'}\pi_{c'}\mathcal{N}(\mathbf{x}\mid\mu_{c'},\Sigma_{c'})$.

Note: In lecture slides this is written as

$$\log p(y=c \mid \mathbf{x};\theta) = \log \pi_c - \tfrac{1}{2}\log|2\pi\Sigma_c| - \tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c) + \text{const},$$

which is the same as my answer, just different in const.

对于给定参数和每一个输入 $\mathbf{x}$，我们可以代入不同的标签，选判别函数值最大的那个. 边界方程由令两个类别的判别函数值相等得到：

> Assignment 5 Q2

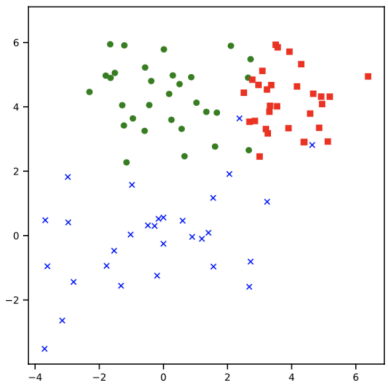
(1) For classes c and c', the decision boundary is found by setting the discriminant functions equal. That is, we require

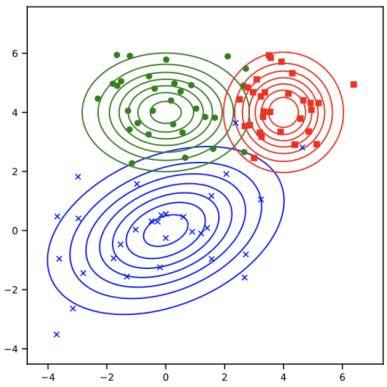
$$\log \pi_c - \tfrac{1}{2}\log|2\pi\Sigma_c| - \tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c) = \log \pi_{c'} - \tfrac{1}{2}\log|2\pi\Sigma_{c'}| - \tfrac{1}{2}(\mathbf{x}-\mu_{c'})^T\Sigma_{c'}^{-1}(\mathbf{x}-\mu_{c'}).$$

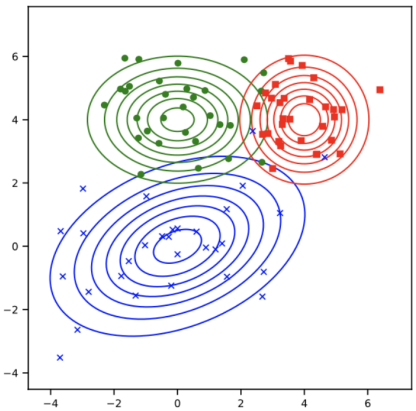
After reordering, the decision boundary is given by

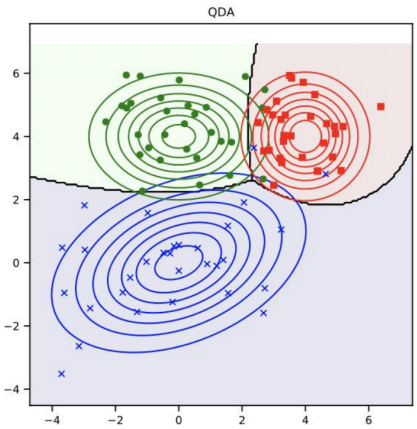
$$\log\frac{\pi_c}{\pi_{c'}} - \tfrac{1}{2}\log\frac{|\Sigma_c|}{|\Sigma_{c'}|} - \tfrac{1}{2}\left[(\mathbf{x}-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}-\mu_c) - (\mathbf{x}-\mu_{c'})^T\Sigma_{c'}^{-1}(\mathbf{x}-\mu_{c'})\right] = 0.$$

输入为 2D 的例子：









10.4.2 LDA

> 高斯判别分析的特例.
>
> 注意，LDA 中二次项仍然存在，但是对于所有类都一样，因此计算决策边界时会抵消，决策边界变为线性.

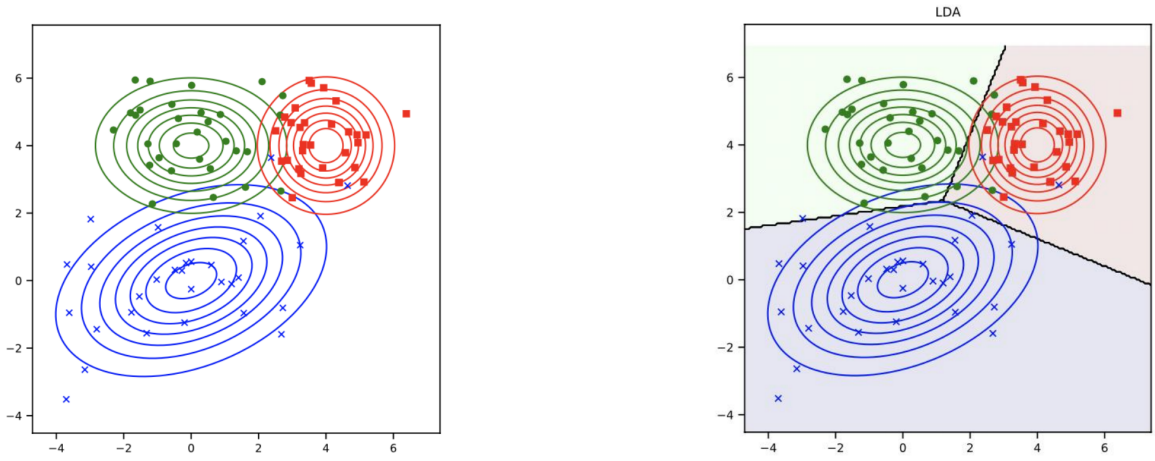
In LDA we assume $\Sigma_c = \Sigma$ for every class c. Then the discriminant function simplifies to

$$\begin{aligned}\log p(y=c \mid \mathbf{x}, \theta) &= \log \pi_c - \tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma^{-1}(\mathbf{x}-\mu_c) + \text{const} \\ &= \log \pi_c - \frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c + \mathbf{x}^T\Sigma^{-1}\mu_c + \text{const} - \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} \\ &= \gamma_c + \mathbf{x}^T\beta_c + \kappa.\end{aligned}$$

Let $\log p(y=c \mid \mathbf{x}, \theta) = \log p(y=c' \mid \mathbf{x}, \theta)$, we have

$$\mathbf{x}^T\Sigma^{-1}(\mu_c - \mu_{c'}) - \tfrac{1}{2}\left(\mu_c^T\Sigma^{-1}\mu_c - \mu_{c'}^T\Sigma^{-1}\mu_{c'}\right) + \log\frac{\pi_c}{\pi_{c'}} = 0.$$

> 这是线性的.



> 等概率线（判别函数的等值点构成的线）还是二次，只是边界变成了线性.

This method is called linear discriminant analysis or LDA.

10.4.3 GDA 优化

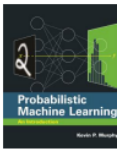
How to fit a GDA model using MLE?

$$\text{Cat}(y|\boldsymbol{\theta}) \triangleq \prod_{c=1}^{C} \theta_c^{\mathbb{I}(y=c)}$$
$$\sum_{c=1}^{C} \theta_c = 1$$

Cat() categorical distribution, page 53

- The likelihood function is as follows

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \text{Cat}(y_n|\boldsymbol{\pi}) \prod_{c=1}^{C} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)^{\mathbb{I}(y_n=c)}$$



$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N}\sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \pi_c\right] + \sum_{c=1}^{C}\left[\sum_{n:y_n=c} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\right]$$

We can optimize the class prior and the θ respectively.

24

How to fit a GDA model using MLE?

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \left[\sum_{n=1}^{N}\sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \pi_c\right] + \sum_{c=1}^{C}\left[\sum_{n:y_n=c} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\right]$$

We omit the derivation steps here,
and it will be part of the assignment!

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{n:y_n=c} \boldsymbol{x}_n$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{n:y_n=c} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^{\mathsf{T}}$$

indicator function:
1 if $y_n = c$, otherwise 0.

25

推导:

(c.1) The likelihood function of GDA is

$$p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} \text{Cat}(y_n \mid \pi) \prod_{c=1}^{C} \mathcal{N}(\mathbf{x}_n \mid \mu_c, \Sigma_c)^{\mathbb{I}(y_n=c)},$$

Take log, we have

$$\log p(\mathcal{D} \mid \theta) = \sum_{n=1}^{N} \left[\log \text{Cat}(y_n \mid \pi) + \sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \mathcal{N}(\mathbf{x}_n \mid \mu_c, \Sigma_c)\right].$$

Since $\text{Cat}(y_n \mid \pi) = \prod_{c=1}^{C} \pi_c^{\mathbb{I}(y_n=c)}$, we have

$$\begin{aligned}\log p(\mathcal{D} \mid \theta) &= \sum_{n=1}^{N}\left[\sum_{c=1}^{C}\mathbb{I}(y_n = c)\log \pi_c + \sum_{c=1}^{C}\mathbb{I}(y_n = c)\log \mathcal{N}(\mathbf{x}_n \mid \mu_c, \Sigma_c)\right] \\ &= \left[\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_n = c)\log \pi_c\right] + \sum_{c=1}^{C}\left[\sum_{n:y_n=c}\log N(\mathbf{x}_n|\mu_\mathbf{c}, \mathbf{\Sigma}_c)\right] \\ &= \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C}\left[\sum_{n:y_n=c}\log N(\mathbf{x}_n|\mu_\mathbf{c}, \mathbf{\Sigma}_c)\right].\end{aligned}$$

(c.2) To get the maximum likelihood estimator $\hat{\theta}_c = \hat{\pi}, \hat{\mu_c}, \hat{\Sigma}_c$, we need to get

i) MLE $\hat{\pi}$ for π

Focus on the part of the log likelihood that depends on $\pi = \pi_1, \ldots, \pi_C$:

$$\sum_{c=1}^{C} N_c \log \pi_c$$

Notice constraint $\sum_{c=1}^{C} \pi_c = 1$, by Lagrange Multiplier, set up the Lagrangian

$$\mathcal{L}(\pi, \lambda) = \sum_{c=1}^{C} N_c \log \pi_c + \lambda\left(1 - \sum_{c=1}^{C}\pi_c\right).$$

Differentiate with respect to π_c and let the derivative be 0, we have

$$\frac{\partial \mathcal{L}}{\partial \pi_c} = \frac{N_c}{\pi_c} - \lambda = 0 \Rightarrow \pi_c = \frac{N_c}{\lambda}.$$

Substituting into the constraint, we have

$$\sum_{c=1}^{C}\pi_c = \sum_{c=1}^{C}\frac{N_c}{\lambda} = \frac{N}{\lambda} = 1 \Rightarrow \lambda = N$$

Hence, we have

$$\hat{\pi}_c = \frac{N_c}{N} \Rightarrow \hat{\pi} = \frac{N_1}{N}, \frac{N_2}{N} \ldots, \frac{N_C}{N}.$$

This estimator means that the prior probability of each class is its frequency in the training set.

ii) MLE $\hat{\mu_c}$ for μ_c

Focus on the part of the log likelihood that depends on μ_c:

$$\sum_{n:y_n=c}\log N(\mathbf{x}_n \mid \mu_c, \Sigma_c) = \sum_{n:y_n=c} -\tfrac{d}{2}\log(2\pi) - \tfrac{1}{2}\log|\Sigma_c| - \tfrac{1}{2}(\mathbf{x} - \mu_c)^T\Sigma_c^{-1}(\mathbf{x} - \mu_c).$$

Ignoring constants independent of μ_c, we need to maximize

$$-\frac{1}{2}\sum_{n:y_n=c}(\mathbf{x}_n - \mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n - \mu_c).$$

Take the derivative with respect to μ_c and set it to zero:

$$\frac{\partial}{\partial \mu_c}\left[-\tfrac{1}{2}\sum_{n:y_n=c}(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c)\right]=\mathbf{0}$$
$$\Rightarrow \sum_{n:y_n=c}\Sigma_c^{-1}(\mathbf{x}_n-\mu_c)=\mathbf{0}.$$

Multiplying on the left by Σ_c gives

$$\sum_{n:y_n=c}\mathbf{x}_n - N_c\,\mu_c=\mathbf{0},$$

so that the maximum likelihood estimator is

$$\hat{\mu}_c=\frac{1}{N_c}\sum_{n:y_n=c}\mathbf{x}_n.$$

iii) MLE $\hat{\Sigma_c}$ for Σ_c

Focus on the part of the log likelihood that depends on Σ_c:

$$\sum_{n:y_n=c}\log N(\mathbf{x}_n\mid\hat{\mu}_c,\Sigma_c)=\sum_{n:y_n=c}-\tfrac{d}{2}\log(2\pi)-\tfrac{1}{2}\log|\Sigma_c|-\tfrac{1}{2}(\mathbf{x}-\hat{\mu}_c)^T\Sigma_c^{-1}(\mathbf{x}-\hat{\mu}_c).$$

Ignoring constants independent of Σ_c, we need to maximize

$$-\frac{1}{2}\sum_{n:y_n=c}\left[\log|\Sigma_c|+(\mathbf{x}_n-\hat{\mu}_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\hat{\mu}_c)\right].$$

Take the derivative with respect to Σ_c and set it to zero:

$$\frac{\partial}{\partial\Sigma_c}\left[-\frac{1}{2}\sum_{n:y_n=c}\left[\log|\Sigma_c|+(\mathbf{x}_n-\hat{\mu}_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\hat{\mu}_c)\right]\right]=\mathbf{0}$$
$$\Rightarrow -\frac{N_c}{2}\Sigma_c^{-1}+\frac{1}{2}\sum_{n:y_n=c}\Sigma_c^{-1}(\mathbf{x}_n-\hat{\mu}_c)(\mathbf{x}_n-\hat{\mu}_c)^T\Sigma_c^{-1}=\mathbf{0}$$

Multiply both sides by Σ_c, we have

$$\hat{\Sigma_c}=\frac{1}{N_c}\sum_{n:y_n=c}(\mathbf{x}_n-\hat{\mu}_c)(\mathbf{x}_n-\hat{\mu}_c)^T$$

10.4.4 最近质心分类器

> Nearest centroid classifier

基于 LDA 继续简化，假设所有标签先验相等.

$$\pi_c=\frac{1}{C}\Rightarrow \log\pi_c \text{ 是常数项，可以忽略}$$

LDA 中，判别函数为

$$\begin{aligned}\log p(y=c\mid\mathbf{x},\theta)&=\log\pi_c-\tfrac{1}{2}(\mathbf{x}-\mu_c)^T\Sigma^{-1}(\mathbf{x}-\mu_c)+\text{const}\\&=\log\pi_c-\frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c+\mathbf{x}^T\Sigma^{-1}\mu_c+\text{const}-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\\&=\gamma_c+\mathbf{x}^T\beta_c+\kappa.\end{aligned}$$

现在，只剩下一项：

$$\hat{y}(\mathbf{x}) = \arg\max_c \log p(y=c|\mathbf{x},\theta) = \arg\min_c (\mathbf{x}-\mu_c)^T \Sigma^{-1} (\mathbf{x}-\mu_c)$$

求最大后验 = 求最小马氏距离（Mahalanobis distance）

> 当 $\Sigma = I$ 时退化为欧氏距离.

看离哪个质心最近（马氏距离），就判定为哪类.

10.4.5 朴素贝叶斯分类器

> Naïve Bayes Classifiers

给定类别标签 y，假设数据的各个特征条件独立.

Naïve Bayes Assumption: features are conditionally independent given the class label.

$$p(\boldsymbol{x}|y=c,\boldsymbol{\theta}) = \prod_{d=1}^{D} p(x_d|y=c,\boldsymbol{\theta}_{dc})$$

The parameters for the Class conditional density For class c and feature d

$$p(y=c|\boldsymbol{x},\boldsymbol{\theta}) = \frac{p(y=c|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|y=c,\boldsymbol{\theta}_{dc})}{\sum_{c'} p(y=c'|\boldsymbol{\pi}) \prod_{d=1}^{D} p(x_d|y=c',\boldsymbol{\theta}_{dc'})}$$

naive Bayes classifier

$P(X \mid Y, Z) = P(X \mid Z)$

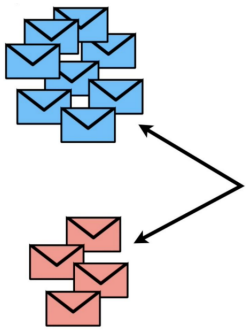
$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$

29

例子：垃圾邮件（Spam）分类器

An example: Spam classifier

- Specifying the features x_j used to represent an email.
- Representing an email via a feature vector (length = the number of words in the dictionary).
- If the dictionary size is 50000, clearly too many parameters!



...and we wanted to filter out the **spam** messages.

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}$$

30

- 二值：$x_d \in \{0,1\}$
- 用伯努利建模生成过程 $p(\mathbf{x}|y=c,\theta)$

如果是实值：$x_d \in \mathbb{R}$

使用单变量高斯（univariate Gaussian）的联合：

$$p(\mathbf{x}|y=c,\theta) = \prod_{d=1}^{D} N(x_d|\mu_{dc},\sigma_{dc}^2)$$

在这种情况下，如何估计参数 μ_{dc}，σ_{dc}^2？MLE.

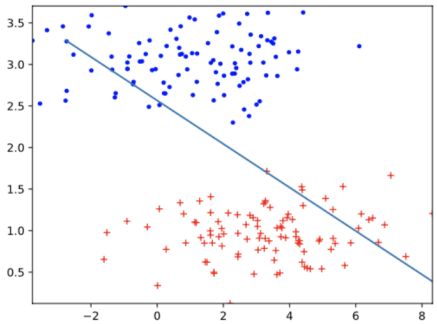
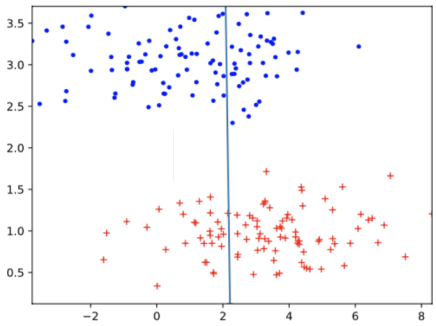
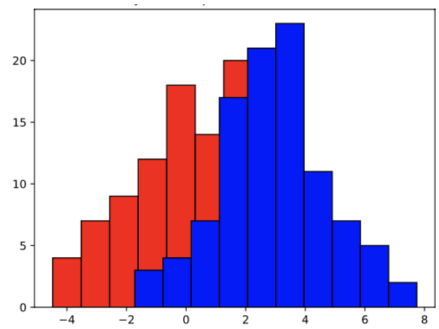
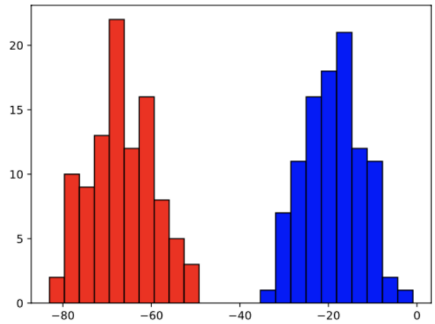
10.5 Fisher's LDA

> 是一个判别式学习的投影方法，但基于生成模型原理推导

假设你有两个类别的数据，每个样本是一个 50 维向量. 如果你能找到一个方向（线性组合），使得两个类别在这个方向上的投影尽可能拉开，那你就可以只在这个方向上进行分类，大大简化问题.

目标：从高维空间 $\mathbb{R}^D$ 中找到一个投影方向，把数据映射到低维（甚至一维），使得

- 不同类别之间的可分性尽量大.
- 同一类别内部尽量紧凑.

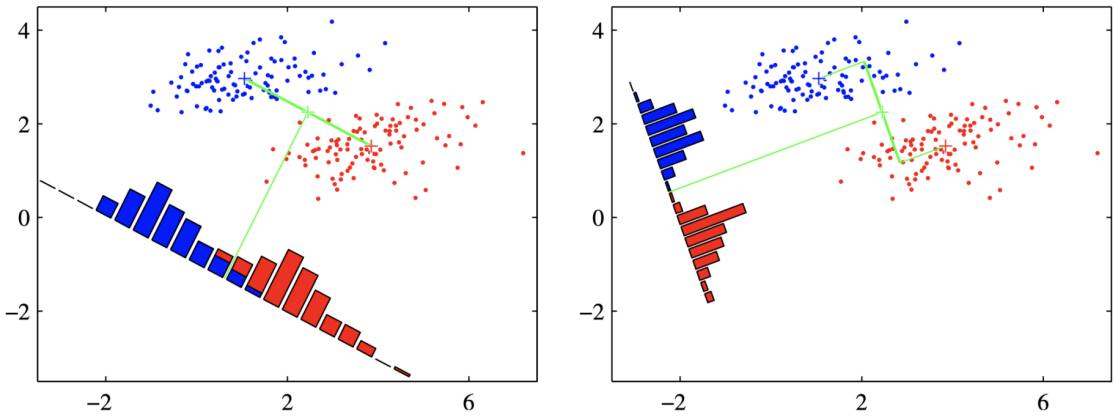





> 右边比左边好.

10.5.1 二分类

> Fisher's LDA for two classes

讨论：如何在二分类问题中找到一个最优方向 $\mathbf{w}$，把高维数据投影到一维（一条线），从而实现分类.



The left plot shows samples from two classes along with the histograms resulting from projection onto the line joining the class means.

The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

已知

- 两类的均值向量：$\mu_1 = \frac{1}{N_1} \sum_{n:y_n=1} \mathbf{x}_n, \mu_2 = \frac{1}{N_2} \sum_{n:y_n=2} \mathbf{x}_n$

> 注意我们已经有 $N_1 + N_2$ 条训练数据.

- 均值投影：$m_k = \mathbf{w}^T \mu_k$
- 样本投影：$z_n = \mathbf{w}^T \mathbf{x}_n$

> 注意，投影后我们不关心方向，因为往一条直线上投影，用投影值（有正负）就可以找到阈值并分类.

我们希望类间距离尽量大，也希望类内方差尽量小（聚拢，避免重叠）.

构造 Fisher 比例：

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

优化目标：最大化 $J(\mathbf{w})$.

Fisher's LDA for two classes

- The derivation:

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^\mathsf{T}\mathbf{S}_B\boldsymbol{w}}{\boldsymbol{w}^\mathsf{T}\mathbf{S}_W\boldsymbol{w}}$$

$$\boldsymbol{w}^\mathsf{T}\mathbf{S}_B\boldsymbol{w} = \boldsymbol{w}^\mathsf{T}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\mathsf{T}\boldsymbol{w} = (m_2 - m_1)(m_2 - m_1)$$

$$\begin{aligned}\boldsymbol{w}^\mathsf{T}\mathbf{S}_W\boldsymbol{w} = &\sum_{n:y_n=1} \boldsymbol{w}^\mathsf{T}(\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\mathsf{T}\boldsymbol{w} + \\ &\sum_{n:y_n=2} \boldsymbol{w}^\mathsf{T}(\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^\mathsf{T}\boldsymbol{w} \\ = &\sum_{n:y_n=1} (z_n - m_1)^2 + \sum_{n:y_n=2} (z_n - m_2)^2\end{aligned}$$

Reference: Page 328 in the book of Probabilistic Machine Learning

$$\mathbf{S}_B = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\mathsf{T}$$

$$\mathbf{S}_W = \sum_{n:y_n=1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\mathsf{T} + \sum_{n:y_n=2} (\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^\mathsf{T}$$

> 类内散度矩阵，Within-class scatter matrix：$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$，未投影前，两类的类内协方差矩阵之和.
>
> 类间散度矩阵，Between-class scatter matrix

优化结果：

$$\mathbf{w}^* \propto \mathbf{S}_W^{-1}(\mu_2 - \mu_1)$$

> 可以归一化. 我们只关注方向，模长无所谓.

Lecture 11 无监督学习

> Unsupervised Learning

不考.

Lecture 12 大间隔分类器

> Large Margin Classifier

不考.

附录

1. Assignment 1

Question 5

Consider a learning problem where there are k possible hypotheses $h_1, h_2, \ldots, h_k$, and you have a sample of N independent and identically distributed (i.i.d.) examples. Let μ_i be the true probability that hypothesis h_i makes a correct prediction on an example, and v_i be the fraction of correct predictions made by hypothesis h_i on the sample. We are interested in applying Hoeffding's inequality to bound the probability that v_i deviates from μ_i by more than ϵ.

The Hoeffding's inequality for a single hypothesis is:

$$\mathbb{P}\left[|v_i - \mu_i| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}$$

where ϵ is a positive deviation threshold, v_i is the observed fraction of correct predictions for hypothesis h_i, and μ_i is the true probability for h_i.

Now, consider the following:

(a) Using Hoeffding's inequality, please calculate the probability that the error of hypothesis h_1 deviates from the true probability μ_1 by more than $\epsilon = 0.05$, given that the sample size is $N = 500$.

(b) Suppose we have $k = 5$ hypotheses, and we want to ensure that for each hypothesis h_i, the deviation is bounded by $\epsilon = 0.05$ with probability at least $1 - \delta$. What sample size N is required for $\delta = 0.01$? Please use Hoeffding's inequality to derive the minimum value of N.

(c) Consider a situation where we have $k = 5$ hypotheses, and you want to ensure that with probability at least $1 - \delta$, the maximum deviation over all hypotheses is bounded by $\epsilon = 0.05$. What sample size N is required to achieve this, given $\delta = 0.01$? Please provide an explanation for why this differs from part (b).

Solution:

(a) By Hoeffding's inequality,

$$\begin{aligned}\mathbb{P}\left[|v_1 - \mu_1| > 0.05\right] &\leq 2e^{-2 \cdot 0.05^2 \cdot 500} \\ &= 2e^{-2.5} \\ &\approx 0.1642.\end{aligned}$$

So the probability that the error of hypothesis h_1 deviates from the true probability μ_1 by more than $\epsilon = 0.05$ is at most 16.42%.

(b) We want for each hypothesis h_i,

$$\mathbb{P}\left[|v_i - \mu_i| \leq \epsilon\right] \geq 1 - \delta \Leftrightarrow \mathbb{P}\left[|v_i - \mu_i| > \epsilon\right] \leq \delta.$$

By Hoeffding's inequality,

$$\mathbb{P}\left[|v_i - \mu_i| > \epsilon\right] \leq 2e^{-2\epsilon^2 N} \leq \delta.$$

Solve for N,

$$N \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta} = \frac{1}{2 \cdot 0.05^2} \ln \frac{2}{0.01} \approx 1059.66.$$

Since N must be an integer, the minimum value of N is 1060.

(c) We want,

$$\mathbb{P}\left[\max_{i=1,\ldots,k} |v_i - \mu_i| \leq \epsilon\right] \geq 1 - \delta$$

By using the union bound and Hoeffding's inequality,

$$\begin{aligned}1 - \mathbb{P}\left[\max_{i=1,\ldots,k} |v_i - \mu_i| \leq \epsilon\right] &= \mathbb{P}\left[\bigcup_{i=1}^{5} |v_i - \mu_i| > \epsilon\right] \\ &\leq \sum_{i=1}^{k} \mathbb{P}\left[|v_i - \mu_i| > \epsilon\right] \\ &\leq 2ke^{-2\epsilon^2 N} \\ &\leq \delta\end{aligned}$$

Solve for N,

$$N \geq \frac{1}{2\epsilon^2} \ln \frac{2k}{\delta} = \frac{1}{2 \cdot 0.05^2} \ln \frac{2 \cdot 5}{0.01} \approx 1381.55.$$

Since N must be an integer, the minimum value of N is 1382.

Explanation:

In part (b) we ensured that **each individual hypothesis meets the deviation bound** with probability at least 0.99 (i.e., an error probability of 0.01 per hypothesis). However, when we want the bound to **hold simultaneously for all 5 hypotheses** (i.e., the maximum deviation is bounded), the failure probabilities add up (via the union bound), making it more likely that at least one hypothesis fails the criterion. **To maintain the overall confidence level of 0.99 for all hypotheses together, we need a larger sample size**. This is why in part (c) the required sample size is greater (approximately 1382 samples) compared to 1060 samples in part (b).

2. 范数

在公式 $E_{in}(\mathbf{w}) = \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$ 中，$\|\cdot\|_2$ 是欧几里得范数（Euclidean norm），也叫 L_2 范数，定义为：

$$\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2}$$

其中 $\mathbf{v}$ 是一个向量. 注意该向量并不限制维度.

而 $\|\cdot\|_2^2$ 代表欧几里得范数的平方：

$$\|\mathbf{v}\|_2^2 = \sum_i v_i^2$$

即去掉了平方根，只保留平方项.

直观理解：

- $\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2}$：向量的欧几里得长度（原点到点 $\mathbf{v}$ 的距离）.
- $\|\mathbf{v}\|_2^2 = \sum_i v_i^2$：向量长度的平方.

在最小二乘法中，我们更倾向于最小化平方误差，而不是误差本身，这样可以：

- 避免平方根的计算（计算更简单）.
- 保持导数连续（便于优化），因为平方根的导数在某些情况下不稳定.

附：`6.3.3 训练误差` 公式推导：

$$
\begin{aligned}
E_{in}(\mathbf{w}) &= \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \\
&= \frac{1}{N}\|X\mathbf{w} - \mathbf{y}\|_2^2 \\
&= \frac{1}{N}(X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y}) \\
&= \frac{1}{N}(\mathbf{w}^T X^T - \mathbf{y}^T)(X\mathbf{w} - \mathbf{y}) \\
&= \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - \mathbf{y}^T X\mathbf{w} - \mathbf{w}^T X^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \\
&= \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{y}^T\mathbf{y})
\end{aligned}
$$

其中：

- 列向量范数的平方等价于转置乘自己.
- $\mathbf{y}^T X\mathbf{w}$ 和 $\mathbf{w}^T X^T\mathbf{y}$ 是等价的，因为 $\mathbf{y}$ 和 $X\mathbf{w}$ 是相同维度的列向量. 如果 $X\mathbf{w}$ 写作 $\hat{\mathbf{y}}$ 结果更显然：$\mathbf{y}^T\hat{\mathbf{y}} = \hat{\mathbf{y}}^T\mathbf{y}$.

3. 线性回归梯度计算

给定线性回归的均方误差（MSE, Mean Squared Error）：

$$
E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2
$$

使用矩阵形式重写为：

$$
\begin{aligned}
E_{in}(\mathbf{w}) &= \frac{1}{N}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \\
&= \frac{1}{N}\|X\mathbf{w} - \mathbf{y}\|_2^2 \\
&= \frac{1}{N}(\mathbf{w}^T X^T X\mathbf{w} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{y}^T\mathbf{y})
\end{aligned}
$$

计算梯度：

$$
\nabla E_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T\mathbf{y})
$$

推导如下：

① 二次型 $\mathbf{x}^TA\mathbf{x}$（标量）对 $\mathbf{x}$（列向量）求导

$$\begin{aligned}\frac{\partial}{\partial\mathbf{x}}(\mathbf{x}^TA\mathbf{x}) &= (A+A^T)\mathbf{x} \\ &= 2A\mathbf{x} \qquad\qquad \text{如果 } A \text{ 是对称矩阵}\end{aligned}$$

② $\mathbf{x}^TA$ 对 $\mathbf{x}$ 求导

$$\frac{\partial}{\partial\mathbf{x}}(\mathbf{x}^TA) = A$$

综上，

$$\begin{aligned}\nabla E_{in}(\mathbf{w}) &= \frac{\partial E_{in}}{\partial\mathbf{w}} \\ &= \frac{\partial}{\partial\mathbf{w}}\frac{1}{N}(\mathbf{w}^TX^TX\mathbf{w} - 2\mathbf{w}^TX^T\mathbf{y} + \mathbf{y}^T\mathbf{y}) \\ &= \frac{1}{N}(2X^TX\mathbf{w} - 2X^T\mathbf{y}) \\ &= \frac{2}{N}(X^TX\mathbf{w} - X^T\mathbf{y})\end{aligned}$$

4. 交叉熵损失

> cross-entropy loss
>
> 见 `6.4.4 训练误差`.

本课程中，逻辑回归使用的标签是 $y \in \{-1, +1\}$. 此时损失函数为

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1 + e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right).$$

> 见 `6.4.4 训练误差`.

而我们也可以使用标签 $y \in \{0, 1\}$，此时损失函数推导如下：

当 $y_n = 1$ 时，$P(y_n|\mathbf{x}_n) = \theta(\mathbf{w}^T\mathbf{x}_n)$；

当 $y_n = 0$ 时，$P(y_n|\mathbf{x}_n) = 1 - \theta(\mathbf{w}^T\mathbf{x}_n)$.

综上，$P(y_n|\mathbf{x}_n) = \theta(\mathbf{w}^T\mathbf{x}_n)^{y_n}(1 - \theta(\mathbf{w}^T\mathbf{x}_n))^{1-y_n}$.

$$
\begin{aligned}
& \max \quad \prod_{n=1}^{N} P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \max \quad \ln\left(\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right) \\
\Leftrightarrow \quad & \max \quad \sum_{n=1}^{N} \ln P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \min \quad -\frac{1}{N}\sum_{n=1}^{N} \ln P(y_n|\mathbf{x}_n) \\
\Leftrightarrow \quad & \min \quad -\frac{1}{N}\sum_{n=1}^{N} \ln \theta(\mathbf{w}^T\mathbf{x}_n)^{y_n}(1-\theta(\mathbf{w}^T\mathbf{x}_n))^{1-y_n} \\
\Leftrightarrow \quad & \min \quad -\frac{1}{N}\sum_{n=1}^{N} \left(y_n \ln \theta(\mathbf{w}^T\mathbf{x}_n) + (1-y_n)\ln(1-\theta(\mathbf{w}^T\mathbf{x}_n))\right) \\
\Leftrightarrow \quad & \min \quad \frac{1}{N}\sum_{n=1}^{N} \left(-y_n \ln \theta(\mathbf{w}^T\mathbf{x}_n) - (1-y_n)\ln(1-\theta(\mathbf{w}^T\mathbf{x}_n))\right)
\end{aligned}
$$

其中 $\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum\limits_{n=1}^{N}[-y_n \log h(\mathbf{x}_n) - (1-y_n)\log(1-h(\mathbf{x}_n))]$ 叫做交叉熵损失. 由于推导步骤完全相同（都是最大似然），可知交叉熵损失和标签为 $\{-1,+1\}$ 时的负对数损失是同一种损失的不同形式.

数学证明如下：

将标签从 $\{0,1\}$ 映射到 $\{-1,+1\}$：令 $y'_n = 2y_n - 1 \Leftrightarrow y_n = \frac{y'_n+1}{2}$.

则交叉熵损失变为

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}[-y_n \log h(\mathbf{x}_n) - (1-y_n)\log(1-h(\mathbf{x}_n))] \\
&= \frac{1}{N}\sum_{n=1}^{N}[-\frac{y'_n+1}{2}\log h(\mathbf{x}_n) - (1-\frac{y'_n+1}{2})\log(1-h(\mathbf{x}_n))]
\end{aligned}
$$

当 $y'_n = +1$ 时，$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum\limits_{n=1}^{N} -\log h(\mathbf{x}_n) = \frac{1}{N}\sum\limits_{n=1}^{N}\log(1+e^{-\mathbf{w}^T\mathbf{x}_n})$;

当 $y'_n = -1$ 时，$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum\limits_{n=1}^{N} -\log(1-h(\mathbf{x}_n)) = \frac{1}{N}\sum\limits_{n=1}^{N}\log(1+e^{\mathbf{w}^T\mathbf{x}_n})$.

> $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$.
>
> $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.
>
> $\theta(-s) = \frac{e^{-s}}{1+e^{-s}} = \frac{1}{1+e^s} = \frac{1+e^s-e^s}{1+e^s} = 1-\theta(s)$.

综上，

$$
\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\log\left(1+e^{-y'_n\mathbf{w}^T\mathbf{x}_n}\right).
$$