ESTR 3108 人工智能基础

> Fundamentals of Artificial Intelligence (ESTR3108)

①②③④⑤⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳㉑㉒㉓㉔㉕㉖㉗㉘㉙㉚㉛㉜㉝㉞㉟㊱㊲㊳㊴㊵㊶㊷㊸㊹㊺㊻㊼㊽㊾㊿

Lecture 1 简介

> 2025.9.2

课程网站:

https://www.cse.cuhk.edu.hk/~qdou/csci_3230/

Lecture 计 attendance, Tutorial 没有 attendance.

数据划分

> Data Split

机器学习的核心任务是训练一个模型（Model），模型通过从数据中学习模式，用于后续预测（Prediction）. 因此，数据的划分很重要.

训练集 + 测试集

> Training Data Set + Test Data Set

训练集：占大部分，如 80%.

测试集：用于在模型训练完成后评估性能，独立于训练集，防止过拟合.

> 较大的测试集能让结果更可靠.

训练集 + 验证集 + 测试集

验证集（Validation Data Set）：有时额外划分出来，用于调节超参数.

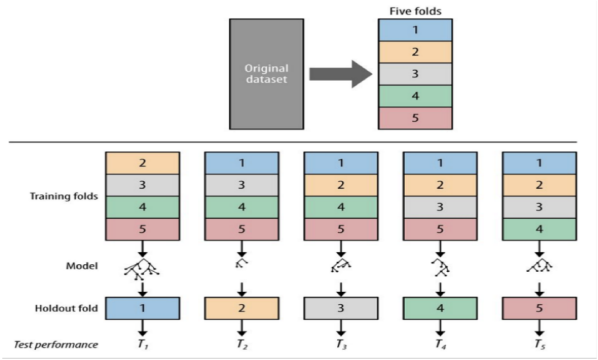
> 训练集训练 - 验证集调参 - 测试集检验.

交叉验证

> Cross-validation
>
> 见 `CSCI 3320 机器学习 9.3 验证集`.

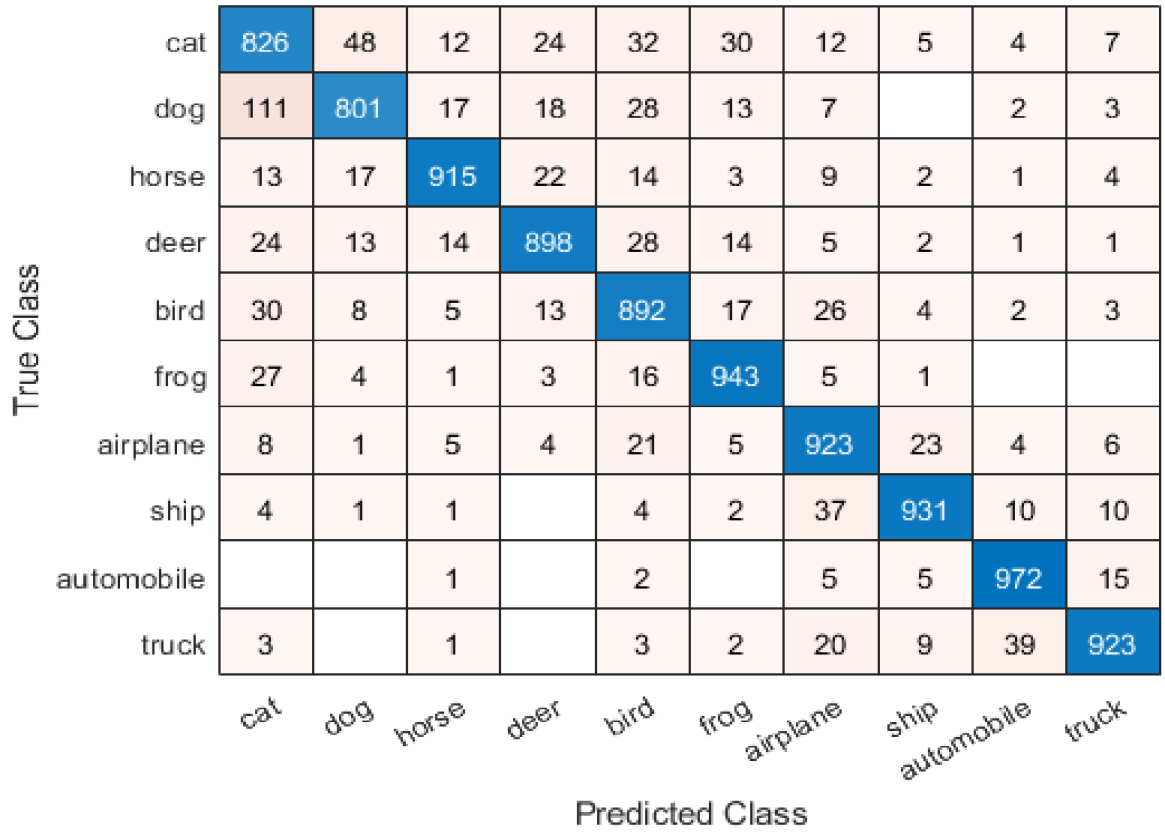
Cross-validation is a popular method used to evaluate AI models on a small-scale data set. The general procedure is as follows:

- Shuffle the dataset randomly and split the dataset into k groups.
- For each unique group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
- Summarize the skill of the model using all the model evaluation scores.



混淆矩阵

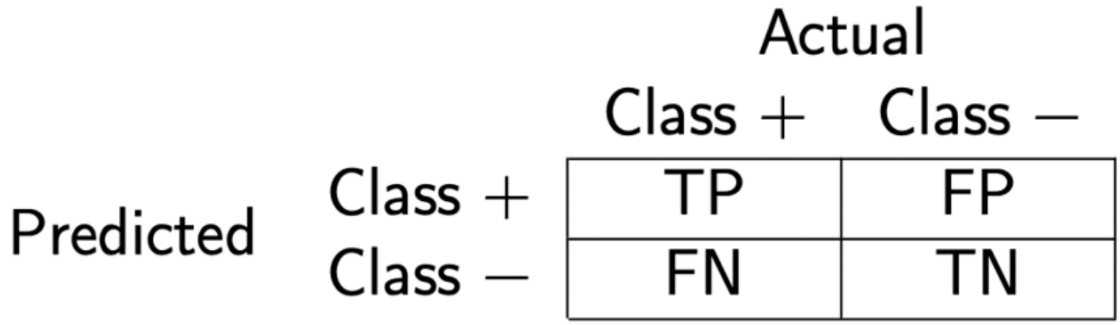
> Confusion Matrix



混淆矩阵是一个特殊的列联表，用来直观展示分类算法的预测效果. 它有两个维度，一个维度表示预测类别（Predicted Class），一个维度表示真实类别（True Class）.

TP FP TN FN

二分类模型的混淆矩阵只有四个单元格，对应四个概念：



- TP is the number of True Positives

  > 实际为正，正确预测为正.

- FP is the number of False Positives

  > 实际为负，错误预测为正.
  >
  > 误报，Type I error.

- TN is the number of ture negatives

  > 实际为负，正确预测为负.

- FN is the number of false negatives

  > 实际为正，错误预测为负.
  >
  > 漏报，Type II error.

准确率 精确率 召回率

组合 TP FP TN FN 可以得出另外一些有意义的概念：

准确率（Accuracy）：$\frac{TP+TN}{TP+FP+TN+FN}$.

> 所有样本被正确预测的比例.

精确率（Precision）：$\frac{TP}{TP+FP}$.

> 所有正预测是正确预测的比例.

召回率（Recall / sensitivity）：$\frac{TP}{TP+FN}$.

> 所有正样本被正确预测的比例.

精确率是针对预测结果，高精确率 means if you test positive, you're probably positive. 召回率是针对样本，召回率越高，正样本越容易被检测出, which means you're not missing many positives.

三类学习

> Types of Learning

There are mainly three types of learning.

监督学习

> Supervised Learning

无监督学习

> Unsupervised Learning

强化学习

> Reinforcement Learning

Lecture 2 线性回归

> Linear Regression
>
> 见 `大二 term 2 CSCI 3320 机器学习 6.3 线性回归`. 本章与机器学习课的符号记法略有出入，但内容一致.

回归是一种统计方法，用于估计因变量与自变量之间的关系.

因变量（dependent variable / outcome / response variable）：想要预测的.

自变量（independent variable / predictor / covariate / explanatory variable / feature）：影响因变量的因素.

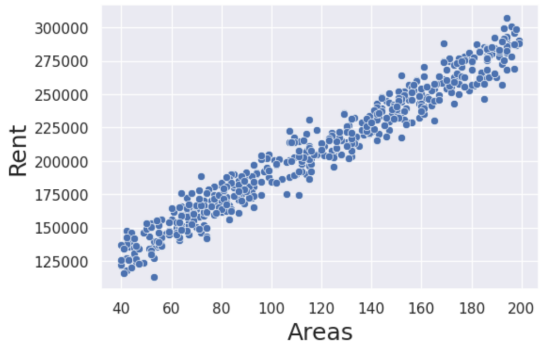
线性回归（Linear Regression）是一种回归模型，它假设因变量与自变量的关系是线性的.

> Intuitively, graphs of linear functions are straight lines / planes （或者更高维的超平面）.
>
> Mathmatically, a function $f(x)$ is linear if and only if $f(u+v)=f(u)+f(v)$ and $f(cu)=cf(u)$.
>
> 线性回归很简单，也很常见，因此放在第一个作介绍.

2.1 房租预测

例：基于面积预测房租.

Index	Areas (m^2)	Rent (HKD)
1	40	134072
2	92	182241
3	37	134731
4	124	204325
5	88	187375
…	…	…



$f: X \to Y$, where $X \subset \mathbb{R}^n$ is the domain of input and $Y \subset \mathbb{R}$ is the domain of output. 这里 X is the domain of apartment areas (仅一个特征) and Y is the domain of rent.

目标：take the given data as training examples and try to find a general mapping $\hat{f}: X \to Y$, which is close to true f as much as possible.

2.2 单对训练数据

2.2.1 单特征

单变量线性函数：

> Univariate Linear Function
>
> 这里的 "单" 指自变量的特征数，不是说只有一对训练数据.

$$\hat{y} = \hat{f}_{\theta_0,\theta_1}(x) = \theta_1 x + \theta_0.$$

如果能找到最佳的 θ_1 和 θ_0，即能用线性回归模型基于面积预测租金.

2.2.2 多特征

如果有多个 features 影响租金，需要引入多变量线性函数：

> Multivariate Linear Function
>
> 这里的 "多" 指自变量的特征数，不是说有多对训练数据.

$$\hat{y} = \hat{f}_{\theta_0,\theta_1,\ldots,\theta_n}(x_1,\ldots,x_n) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n.$$

> 注意，需要训练的参数数量比特征数大 1，因为超平面不总是过原点（截距 / 偏置 / Bias Term 是额外的一项待训练参数）.

写成矩阵形式：

$$\hat{y} = \hat{f}_{\Theta,\theta_0}(X) = X^T\Theta + \theta_0.$$

where

$$\Theta = \begin{pmatrix}\theta_1\\ \vdots \\ \theta_n\end{pmatrix} \text{ and } X = \begin{pmatrix}x_1\\ \vdots \\ x_n\end{pmatrix}.$$

2.3 m 对训练数据

接 `2.2.2 多特征映射`. 对于 m 对 samples / 训练数据，记

$$X^{(1)} = \begin{pmatrix}x_1^{(1)}\\ \vdots \\ x_n^{(1)}\end{pmatrix}, X^{(2)} = \begin{pmatrix}x_1^{(2)}\\ \vdots \\ x_n^{(2)}\end{pmatrix}, \cdots, X^{(m)} = \begin{pmatrix}x_1^{(m)}\\ \vdots \\ x_n^{(m)}\end{pmatrix}$$

with respective labels $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$.

2.3.1 数据矩阵 / 目标向量

把 m 个自变量合并为数据矩阵（data matrix）$\mathbf{X} \in \mathbb{R}^{m\times n}$，$m$ 个因变量 / 标签合并为目标向量 $Y \in \mathbb{R}^m$（target vector）.

$$\mathbf{X} = \begin{pmatrix}X^{(1)^T}\\ \vdots \\ X^{(m)^T}\end{pmatrix} = \begin{pmatrix}x_1^{(1)} & \cdots & x_n^{(1)}\\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)}\end{pmatrix}, Y = \begin{pmatrix}y^{(1)}\\ \vdots \\ y^{(m)}\end{pmatrix}.$$

目标：找到一个最佳的线性映射（权重向量 $\Theta = \begin{pmatrix}\theta_1\\ \vdots \\ \theta_n\end{pmatrix}$ 和偏置项 θ_0），使得 $\mathbf{X}$ 经过该映射得到的预测向量 $\hat{Y}$ 尽可能接近目标向量 Y.

此时，多变量线性函数可以写成矩阵形式：

> Matrix Representation of Linear Function

$$\hat{Y} = \hat{f}_{\Theta,\theta_0}(\mathbf{X}) = \mathbf{X}\Theta + \theta_0.$$

where

$$\Theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_n \end{pmatrix} \text{ and } \hat{Y} = \begin{pmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(m)} \end{pmatrix}.$$

> 其实就是把 m 个预测函数写到一起.

注意：

$$\mathbf{X}\Theta = \theta_1 \begin{pmatrix} x_1^{(1)} \\ \vdots \\ x_1^{(m)} \end{pmatrix} + \theta_2 \begin{pmatrix} x_2^{(1)} \\ \vdots \\ x_2^{(m)} \end{pmatrix} + \cdots + \theta_n \begin{pmatrix} x_n^{(1)} \\ \vdots \\ x_n^{(m)} \end{pmatrix}$$

可以理解为特征列的加权求和（不同特征对结果的影响不同）.

2.3.2 偏置项

> bias term

为简化起见，通常把 bias term θ_0 吸收到 Θ 中，并给每一个输入样本添加一个常数项偏置 $x_0 = 1$.

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}, \ \Theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}.$$

得到简化后的映射函数：

> Simplified Matrix Representation of Linear Function

$$\hat{Y} = f_\Theta(\mathbf{X}) = \mathbf{X}\Theta.$$

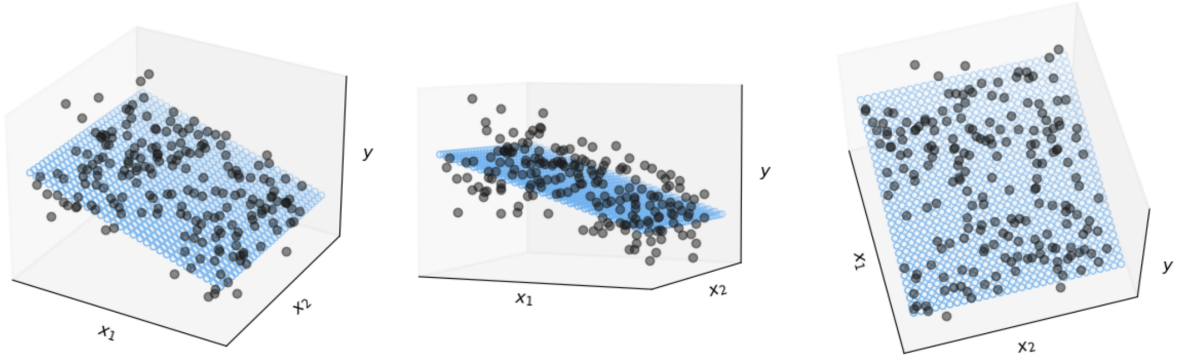
这使得后续的优化推导更加方便.

2.4 几何理解

> Geometry of Linear Regression

m 个训练数据 $\{(x_1^{(1)}, \dots, x_n^{(1)}, y^{(1)}), \dots, (x_1^{(m)}, \dots, x_n^{(m)}, y^{(m)})\}$ 分布在 $(n+1)$ 维空间，The "fitting lines" for n-dimensional feature space are n-dimensional hyperplanes.

> 注意这里 $x_0 = 1$ 不计入特征维度.



在 $(n+1)$ 维空间里，每个线性函数 $y = X\Theta + \theta_0$ 定义了一个 n 维超平面.

注意，不需要证明 n 维特征空间经过线性映射会落在同一个超平面上，因为超平面本身就是由线性映射函数对应的非零线性方程的解集定义的.

线性回归目标的几何理解：找到最佳超平面，使得该超平面与训练数据整体分布最接近.

2.5 优化方法

> Optimizing Linear Regression

"最接近 / 最佳" 都是定性概念，如何转化为定量?

直觉上，对于某对特定样本，估计值和真实标签的距离越小，估计越好. 但是有多对训练数据（多个标签），因此要从总体考虑.

2.5.1 目标函数

> Objective Function

记 $J(\Theta)$ 为 Objective / Cost / Loss Function.

Goal: find the Θ^* that minimizes the function $J(\Theta)$.

2.5.2 残差平方和

> Residual Sum of Squares (RSS)

① 定义

线性回归通常定义代价函数 $J(\Theta)$ 为残差平方和:

$$J(\Theta) = \sum_{i=1}^{m} \left(\hat{f}_\Theta \left(X^{(i)} \right) - y^{(i)} \right)^2 = \| \hat{f}_\Theta(\mathbf{X}) - Y \|_2^2.$$

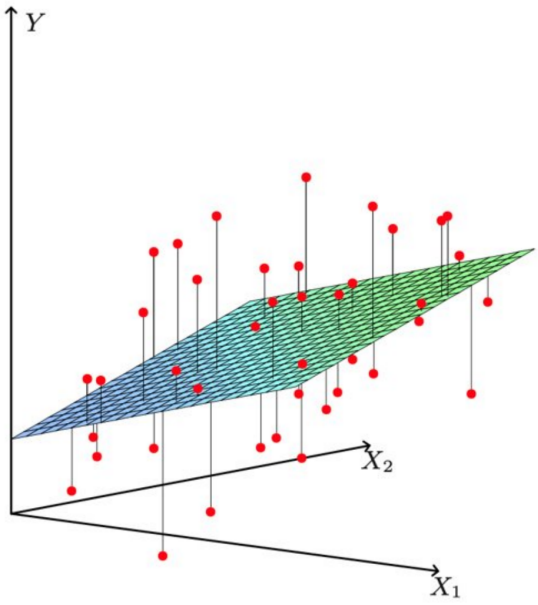
> $\| \cdot \|_2^2$ 表示欧几里得范数的平方.
>
> 这里除以 m 就是 MSE，均方误差. 但分母 m 对优化结果没有影响.

注意，代价函数不是必须设置成残差平方和. 线性回归中采用最小化 MSE 进行优化，出于以下原因:

- 目标函数可导 / 是凸函数，矩阵形式下可以直接导出解析解.
- MSE 对应高斯噪声——假设误差服从正态分布，极大似然估计等价于最小化 MSE.

② 几何理解

> Geometry of RSS



对于每个样本，"residual" means the difference between the estimated value (the plane) and the corresponding training label (red points).

2.5.3 普通最小二乘法

当我们把 $J(\Theta)$ 定义为 RSS，即用最小化 RSS / MSE 的方式来估计参数 Θ，这种方法叫做**普通最小二乘法**（Ordinary Least Squares）. 核心思想：找到一个拟合曲线（非线性回归） / 直线，使得所有训练数据的预测值和真实值之间的误差平方和最小.

> 普通是相对正则化方法而言的. 这里没有考虑惩罚项.

2.5.4 求解

① 单特征

> 见 `2.2.1 单特征` .

RSS for univariate linear regression:

$$
\begin{aligned}
J(\theta_0, \theta_1) &= \sum_{i=1}^{m} \left(\hat{f}_{\theta_0, \theta_1} \left(x^{(i)} \right) - y^{(i)} \right)^2 \\
&= \sum_{i=1}^{m} \left(\theta_1 x^{(i)} + \theta_0 - y^{(i)} \right)^2
\end{aligned}
$$

After expanding the quadratic term and reduction, we have

$$
\begin{aligned}
J(\theta_0, \theta_1) = & \sum_{i=1}^{m} \left(x^{(i)} - \overline{x} \right)^2 \left(\theta_1 - \frac{\sum_{i=1}^{m} \left(x^{(i)} - \overline{x} \right) \left(y^{(i)} - \overline{y} \right)}{\sum_{i=1}^{m} \left(x^{(i)} - \overline{x} \right)^2} \right)^2 \\
& + m(\theta_0 - (\overline{y} - \theta_1 \overline{x}))^2 \\
& + \sum_{i=1}^{m} \left(y^{(i)} - \overline{y} \right)^2 \\
& - \frac{\left(\sum_{i=1}^{m} \left(x^{(i)} - \overline{x} \right) \left(y^{(i)} - \overline{y} \right) \right)^2}{\sum_{i=1}^{m} \left(x^{(i)} - \overline{x} \right)^2}
\end{aligned}
$$

仅前两项与 θ_0, θ_1 有关. 因为前两项非负，to minimize $J(\theta_0, \theta_1)$, let first two terms equal to 0.

$$\begin{cases} 0 = \sum\limits_{i=1}^{m} \left(x^{(i)} - \overline{x}\right)^2 \left(\theta_1 - \frac{\sum_{i=1}^{m}\left(x^{(i)}-\overline{x}\right)\left(y^{(i)}-\overline{y}\right)}{\sum_{i=1}^{m}\left(x^{(i)}-\overline{x}\right)^2}\right)^2 \\ 0 = m(\theta_0 - (\overline{y} - \theta_1\overline{x}))^2 \end{cases}$$

解得

$$\begin{cases} \theta_1 = \frac{\sum_{i=1}^{m}\left(x^{(i)}-\overline{x}\right)\left(y^{(i)}-\overline{y}\right)}{\sum_{i=1}^{m}\left(x^{(i)}-\overline{x}\right)^2} \\ \theta_0 = \overline{y} - \theta_1\overline{x} \end{cases}$$

> 这里 $\overline{x}$，$\overline{y}$ 计算时要注意取平均.

详细步骤

$$\begin{aligned} J(\theta_0, \theta_1) &= \sum_{i=1}^{m} \left(\hat{f}_{\theta_0,\theta_1}\left(x^{(i)}\right) - y^{(i)}\right)^2 \\ &= \sum_{i=1}^{m} \left(\theta_1 x^{(i)} + \theta_0 - y^{(i)}\right)^2 \\ &= \sum_{i=1}^{m} \left(\theta_1 x^{(i)} + \theta_0 - y^{(i)} + (\overline{y} - \theta_1\overline{x}) - (\overline{y} - \theta_1\overline{x})\right)^2 \\ &= \sum_{i=1}^{m} \left((\theta_0 - (\overline{y} - \theta_1\overline{x})) + \left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right)\right)^2 \\ &= \sum_{i=1}^{m} \left[(\theta_0 - (\overline{y} - \theta_1\overline{x}))^2 + \left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right)^2 + 2\left(\theta_0 - (\overline{y} - \theta_1\overline{x})\right)\left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right)\right] \end{aligned}$$

注意到:

$$\sum_{i=1}^{m}(x^{(i)} - \overline{x}) = 0, \quad \sum_{i=1}^{m}(y^{(i)} - \overline{y}) = 0$$

所以上式的交叉项

$$\sum_{i=1}^{m} 2\left(\theta_0 - (\overline{y} - \theta_1\overline{x})\right)\left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right) = 0$$

于是

$$J(\theta_0, \theta_1) = m(\theta_0 - (\overline{y} - \theta_1\overline{x}))^2 + \sum_{i=1}^{m}\left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right)^2$$

等式右侧第一项不动，第二项继续变形.

记 $S_{xx} = \sum\limits_{i=1}^{m}(x^{(i)} - \overline{x})^2$，$S_{yy} = \sum\limits_{i=1}^{m}(y^{(i)} - \overline{y})^2$，$S_{xy} = \sum\limits_{i=1}^{m}(x^{(i)} - \overline{x})(y^{(i)} - \overline{y})$

$$\begin{aligned} \sum_{i=1}^{m}\left(\theta_1\left(x^{(i)} - \overline{x}\right) - \left(y^{(i)} - \overline{y}\right)\right)^2 &= \sum_{i=1}^{m}\left(\theta_1^2\left(x^{(i)} - \overline{x}\right)^2 + \left(y^{(i)} - \overline{y}\right)^2 - 2\theta_1\left(x^{(i)} - \overline{x}\right)\left(y^{(i)} - \overline{y}\right)\right) \\ &= S_{xx}\left(\theta_1^2 - \frac{2S_{xy}}{S_{xx}}\theta_1\right) + S_{yy} \\ &= S_{xx}\left(\theta_1 - \frac{S_{xy}}{S_{xx}}\right)^2 - \frac{S_{xy}^2}{S_{xx}} + S_{yy} \end{aligned}$$

证毕.

② 多特征

> 见 `2.2.2 多特征`.

$$\hat{\Theta} = \Theta^* = \arg\min_{\Theta} J(\Theta)$$

解得

$$\hat{\Theta} = \Theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$

> 正规方程.

推导过程：

> 把偏置项 θ_0 吸收到权重向量，并给每一个输入样本添加一个常数项偏置 $x_0 = 1$.

$$\begin{aligned} J(\Theta) &= \|\hat{f}_\Theta(\mathbf{X}) - Y\|_2^2 \\ &= (\mathbf{X}\Theta - Y)^T(\mathbf{X}\Theta - Y) \\ &= \Theta^T\mathbf{X}^T\mathbf{X}\Theta - Y^T\mathbf{X}\Theta - \Theta^T\mathbf{X}^T Y + Y^T Y \end{aligned}$$

$$\begin{aligned} \text{Let} \quad \frac{\partial J(\Theta)}{\partial \Theta} &= 2\mathbf{X}^T\mathbf{X}\Theta - \mathbf{X}^T Y - \mathbf{X}^T Y \\ &= 2\mathbf{X}^T(\mathbf{X}\Theta - Y) \\ &= \mathbf{0} \\ \Rightarrow \hat{\Theta} = \Theta^* &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y \end{aligned}$$

关于二阶导是否大于 0，在下方 `2.5.5` 中详细说明.

2.5.5* 解的存在性

> ESTR content

已知线性回归解析解：

$$\Theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$

当 $(\mathbf{X}^T\mathbf{X})^{-1}$ 存在，即 $\mathbf{X}^T\mathbf{X}$ 可逆时，该解存在.

讨论：$\mathbf{X}^T\mathbf{X}$ 什么时候可逆？

根据线性代数知识，$\text{rank}(\mathbf{X}) = d + 1$（满秩）时 $\mathbf{X}^T\mathbf{X}$ 可逆.

> $d + 1$ 是包含 $x_0 = 1$ 的特征数量. d 是输入数据的真实特征数量.
>
> 这是充要条件，即 $\mathbf{X}^T\mathbf{X}$ 可逆当且仅当 $\mathbf{X}$ 列满秩.

讨论：Hessian 矩阵正定/半正定与解析解存在性的关系

对于一个二次型函数，

如果 $\mathbf{X}$ 不满秩，则黑塞矩阵是 Positive Semi-Definite，该函数是凸函数，有无穷多个全局最小值点（此时 $\Theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$ 不存在）；

如果 $\mathbf{X}$ 满秩，则黑塞矩阵 Positive Definite，则函数是严格凸函数，有全局唯一最小值点 $\Theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$

2.5.6* 几何理解

> ESTR content

Y 落在特征空间内：训练样本恰好在同一个超平面内. 此时 Y 本身可以由 X 的特征列线性组合而成（组合方式即 $\hat{\Theta}$），完美拟合，不需要投影.

Y 落在特征空间外（绝大多数情况）：训练样本不在同一个超平面内. 此时 Y 在特征空间的每一个投影 $\hat{Y}$ 对应一种拟合方式，$||\hat{Y} - Y||_2^2$ 表示残差平方和. 其中，正交投影使得残差平方和最小，对应普通最小二乘法得到的解.

2.6 偏差与方差

> Bias and Variance

Due to the noise from observation, $y = f(X) + \epsilon$, where $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$.

> 假设每次观测总存在一个不可避免的随机误差，来模拟现实世界的测量误差，记作 ϵ.
>
> ϵ 表示即使我们知道真实的函数关系 $f(X)$, 观测值 y 依然包含无法避免的波动.

2.6.1 期望预测误差

> Expected Prediction Error

For any fixed X with label y, 定义 the expected prediction error at X 为

$$EPE(X) = \mathbb{E}\left(\left(y - \hat{f}(X)\right)^2\right)$$

2.6.2 偏差-方差分解

> Bias-variance decomposition

$$
\begin{aligned}
EPE(X) &= \mathbb{E}\left(\left(y-\hat{f}(X)\right)^2\right) \\
&= \mathbb{E}\left(\left(f(X)+\epsilon-\hat{f}(X)\right)^2\right) \\
&= \mathbb{E}\left(\left(f(X)-\hat{f}(X)\right)^2+\epsilon^2+2\epsilon\left(f(X)-\hat{f}(X)\right)\right) \\
&= \mathbb{E}\left(\left(f(X)-\hat{f}(X)\right)^2\right)+\sigma^2 \\
&= \mathbb{E}\left(\left(\left(f(X)-\mathbb{E}\left(\hat{f}(X)\right)\right)-\left(\hat{f}(X)-\mathbb{E}\left(\hat{f}(X)\right)\right)\right)^2\right)+\sigma^2 \\
&= \mathbb{E}\left(\left(f(X)-\mathbb{E}\left(\hat{f}(X)\right)\right)^2\right)+Var(\hat{f}(X))+\sigma^2 \\
&= \left(f(X)-\mathbb{E}\left(\hat{f}(X)\right)\right)^2+Var(\hat{f}(X))+\sigma^2
\end{aligned}
$$

记 $Bias(\hat{f}(X)) = \mathbb{E}\left(\hat{f}(X)\right) - f(X)$

$Var(\hat{f}(X)) = \mathbb{E}\left(\left(\hat{f}(X)-\mathbb{E}\left[\hat{f}(X)\right]\right)^2\right)$

则期望预测误差可以分解为：

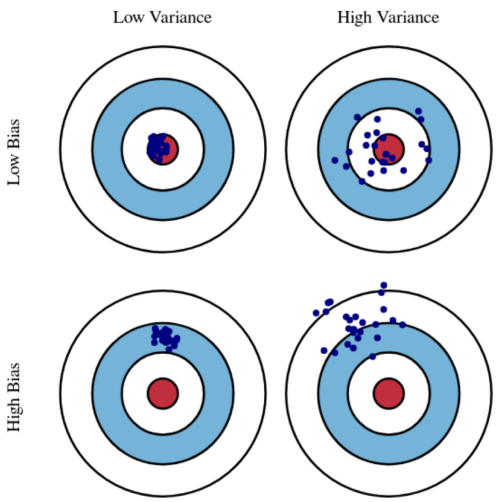
$$EPE(X) = Bias(\hat{f}(X))^2 + Var(\hat{f}(X)) + \sigma^2$$

其中，σ^2 是噪声带来的固有的不可约误差. 那么，对于一定的 $EPE(X)$，$Bias(\hat{f}(X))$ 和 $Var(\hat{f}(X))$ 一个小一个就大.

The expected prediction error:

$$\mathrm{EPE}(X) = \boxed{\mathrm{Bias}(\hat{f}(X))^2 + \mathrm{Var}(\hat{f}(X))} + \sigma^2$$

- The expected prediction error is composed of bias, variance and irreducible errors.



2.6.3 偏差-方差权衡

> Bias-variance tradeoff

If simultaneously minimizing bias and variance, we can achieve a very considerable EPE.

However, in general, low variance will cause high bias, while low bias will result in high variance.

Think about we repeat the training process on randomly sampled data for many times.

- For each training, if the model perfectly fits the training data, the prediction bias is very low but the variance will be very high since the model will vary significantly among different training data.
- If the model is constant among different training data, the prediction variance is zero but definitely the prediction bias is very high.

Thus, we need to make a tradeoff between minimizing bias and minimizing variance.

2.6.4 过拟合 / 欠拟合

> Overfitting and underfitting

过拟合（Overfitting）：Low bias high variance

欠拟合（Underfitting）：High bias low variance

2.6.5 收缩法

> Shrinkage methods

Ordinary least squares prone to learn a low bias, high variance model.

有过拟合风险.

To improve the EPE, sometimes we would like to sacrifice some bias to reduce the variance.

Generally, tuning model complexity is a way to achieve better EPE.

Shrinkage methods are batch of methods for automatically model complexity tuning, e.g., Ridge regression and Lasso regression.

2.7 岭回归

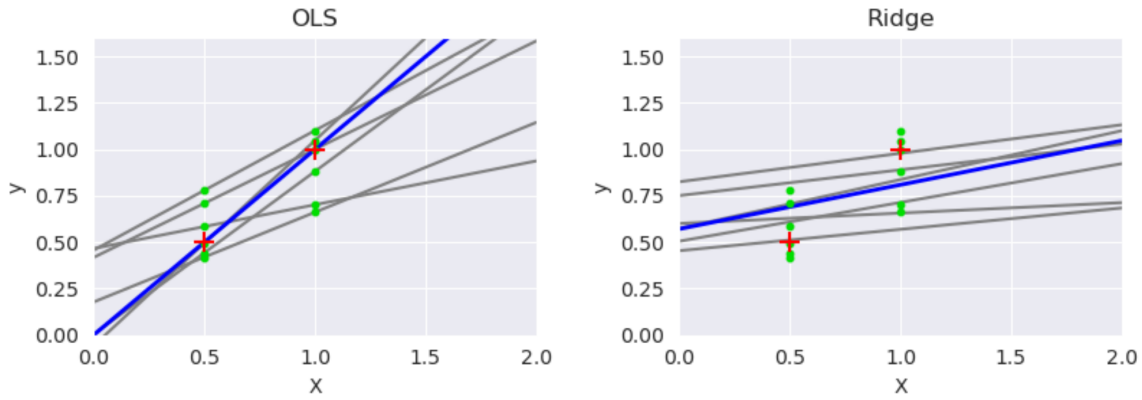
> Ridge Regression
>
> 引入 L2 正则.

$$\widehat{\mathbf{\Theta}}^{\text{ridge}} = \arg\min_{\mathbf{\Theta}} \|\mathbf{X}\mathbf{\Theta} + \theta_0 - \mathbf{Y}\|_2^2 + \lambda\|\mathbf{\Theta}\|_2^2$$

其中，$\|\Theta\|_2^2 = \sum_{i=1}^{n} \theta_i^2$ 叫做 L2 Norm，乘以一个正系数 λ 作为惩罚项.

> λ 是超参数.

Each time OLS is good at fitting training data but varies tremendously. On contrary, ridge regression suppresses the variation of models but fails to give a perfect fitting on training data.



例（Assignment 1 Q2）：

Recall that we have learned ridge regression which is a shrinkage method to regularize the coefficients in linear models.

Suppose we have M samples $\mathbf{X} = \begin{pmatrix} \mathbf{X}^{(1)} \\ \mathbf{X}^{(2)} \\ \vdots \\ \mathbf{X}^{(\mathbf{m})} \end{pmatrix}$. The ridge regression penalizes L2 norm of the model parameters:

$\widehat{\mathbf{\Theta}}^{\text{ridge}} = \arg\min_{\Theta} \|\mathbf{X}\mathbf{\Theta} + \theta_0 - \mathbf{Y}\|_2^2 + \lambda\|\mathbf{\Theta}\|_2^2$, with λ as a positive scalar ($\lambda > 0$). Let's find the analytic solution for the ridge regression.

(a) First of all, in order to be more convenient for the derivation, we rewrite the error function as:

$$J(\mathbf{\Theta}, \theta_0) = \sum_{i=1}^{m} \left[\mathbf{X}^{(\mathbf{i})}\mathbf{\Theta} + \theta_0 - \mathrm{Y}^{(\mathbf{i})}\right]^2 + \lambda\|\mathbf{\Theta}\|_2^2$$

Set $\overline{\mathbf{X}}$ be the mean vector of all the raw vectors of $\mathbf{X}$, then, if we change the form of $J(\mathbf{\Theta}, \theta_0)$ into the following rewritten function:

$$J(\mathbf{\Theta}, \theta_0) = \sum_{i=1}^{m} \left[\left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\mathbf{\Theta} + \theta_0 + \overline{\mathbf{X}}\mathbf{\Theta} - \mathrm{Y}^{(\mathbf{i})}\right]^2 + \lambda\|\mathbf{\Theta}\|_2^2$$

Prove that, when $\theta_0 = \overline{\mathrm{Y}} - \overline{\mathbf{X}}\mathbf{\Theta}$, we can get the minimal value for $J(\mathbf{\Theta}, \theta_0)$.

Solution:

$$\begin{aligned}
& J(\mathbf{\Theta}, \theta_0) \\
=& \sum_{i=1}^{m} \left[\left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\mathbf{\Theta} + \theta_0 + \overline{\mathbf{X}}\mathbf{\Theta} - \mathrm{Y}^{(\mathbf{i})}\right]^2 + \lambda\|\mathbf{\Theta}\|_2^2 \\
=& \sum_{i=1}^{m} \left[\left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\mathbf{\Theta}\right]^2 \rightarrow \text{not related to } \theta_0 \\
&+ \sum_{i=1}^{m} \left(\theta_0 + \overline{\mathbf{X}}\mathbf{\Theta} - \mathrm{Y}^{(\mathbf{i})}\right)^2 \\
&+ \sum_{i=1}^{m} 2\left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\mathbf{\Theta}\theta_0 \rightarrow 2\theta_0 \left[\sum_{i=1}^{m} \left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\right]\mathbf{\Theta} \rightarrow 0 \rightarrow \text{not related to } \theta_0 \\
&+ \sum_{i=1}^{m} 2\left(\mathbf{X}^{(\mathbf{i})} - \overline{\mathbf{X}}\right)\mathbf{\Theta}\left(\overline{\mathbf{X}}\mathbf{\Theta} - \mathrm{Y}^{(\mathbf{i})}\right) \rightarrow \text{not related to } \theta_0 \\
&+ \lambda\|\mathbf{\Theta}\|_2^2 \rightarrow \text{not related to } \theta_0
\end{aligned}$$

By analysis, only the second term $\sum_{i=1}^{m} \left(\theta_0 + \overline{\mathbf{X}}\mathbf{\Theta} - \mathrm{Y}^{(\mathbf{i})}\right)^2$ is related to θ_0.

$$
\begin{aligned}
&\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\mathrm{Y}^{(\mathrm{i})}\right)^2 \\
=&\sum_{i=1}^{m}\left[\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)-\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right)\right]^2 \\
=&\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2+\sum_{i=1}^{m}\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right)^2-\sum_{i=1}^{m}2\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right) \\
=&\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2+\sum_{i=1}^{m}\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right)^2-2\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)\sum_{i=1}^{m}\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right) \\
=&\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2+\sum_{i=1}^{m}\left(\mathrm{Y}^{(\mathrm{i})}-\overline{Y}\right)^2
\end{aligned}
$$

Only the first term $\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2$ is related to θ_0.

$$
\begin{cases}
\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2\geq 0, \\
\sum_{i=1}^{m}\left(\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\overline{Y}\right)^2=0 \text{ if } \theta_0=\overline{\mathrm{Y}}-\overline{\mathbf{X}}\mathbf{\Theta}.
\end{cases}
$$

$\Rightarrow$ When $\theta_0=\overline{\mathrm{Y}}-\overline{\mathbf{X}}\mathbf{\Theta}$, we can get the minimal value for $J\left(\mathbf{\Theta},\theta_0\right)$.

(b) Next, let's define the centered input as $\mathbf{X}_c^{(i)}=\mathbf{X}^{(\mathrm{i})}-\overline{\mathbf{X}}$, and the corresponding centered label as $\mathrm{Y}_c^{(i)}=\mathrm{Y}^{(\mathrm{i})}-\overline{\mathrm{Y}}$. Plug the above θ_0 into the loss function, try to derive that:

$$
J\left(\mathbf{\Theta},\theta_0\right)=J_c(\mathbf{\Theta})=\sum_{i=1}^{m}\left[\mathbf{X}_c^{(i)}\mathbf{\Theta}-\mathrm{Y}_c^{(i)}\right]^2+\lambda\|\mathbf{\Theta}\|_2^2 \qquad (5\%)
$$

Solution:

By plugging $\theta_0=\overline{\mathrm{Y}}-\overline{\mathbf{X}}\mathbf{\Theta}$ into the loss function, we have

$$
\begin{aligned}
J\left(\mathbf{\Theta},\theta_0\right)&=\sum_{i=1}^{m}\left[\left(\mathbf{X}^{(\mathrm{i})}-\overline{\mathbf{X}}\right)\mathbf{\Theta}+\theta_0+\overline{\mathbf{X}}\mathbf{\Theta}-\mathrm{Y}^{(\mathrm{i})}\right]^2+\lambda\|\mathbf{\Theta}\|_2^2 \\
&=\sum_{i=1}^{m}\left[\left(\mathbf{X}^{(\mathrm{i})}-\overline{\mathbf{X}}\right)\mathbf{\Theta}+\left(\overline{\mathrm{Y}}-\overline{\mathbf{X}}\mathbf{\Theta}\right)+\overline{\mathbf{X}}\mathbf{\Theta}-\mathrm{Y}^{(\mathrm{i})}\right]^2+\lambda\|\mathbf{\Theta}\|_2^2 \\
&=\sum_{i=1}^{m}\left[\mathbf{X}_c^{(i)}\mathbf{\Theta}-\mathrm{Y}_c^{(i)}\right]^2+\lambda\|\mathbf{\Theta}\|_2^2
\end{aligned}
$$

(c) Finally, with calculating the first-order derivatives of $J_c(\mathbf{\Theta})$, try to find the analytic solution $\widehat{\mathbf{\Theta}}$ for the ridge regression.

Solution:

$$
\begin{aligned}
J_c(\mathbf{\Theta})&=\sum_{i=1}^{m}\left[\mathbf{X}_c^{(i)}\mathbf{\Theta}-\mathrm{Y}_c^{(i)}\right]^2+\lambda\|\mathbf{\Theta}\|_2^2 \\
&=\|\mathbf{X}_c\mathbf{\Theta}-\mathrm{Y}_c\|_2^2+\lambda\|\mathbf{\Theta}\|_2^2 \\
&=\left(\mathbf{X}_c\mathbf{\Theta}-\mathrm{Y}_c\right)^T\left(\mathbf{X}_c\mathbf{\Theta}-\mathrm{Y}_c\right)+\lambda\mathbf{\Theta}^T\mathbf{\Theta} \\
&=\mathbf{\Theta}^T\mathbf{X}_c^T\mathbf{X}_c\mathbf{\Theta}-\mathrm{Y}_c^T\mathbf{X}_c\mathbf{\Theta}-\mathbf{\Theta}^T\mathbf{X}_c^T\mathrm{Y}_c+\mathrm{Y}_c^T\mathrm{Y}_c+\lambda\mathbf{\Theta}^T\mathbf{\Theta}
\end{aligned}
$$

Calculate the first-order derivatives of $J_c(\mathbf{\Theta})$. We have

$$
\begin{aligned}
\frac{\partial}{\partial\mathbf{\Theta}}J_c(\mathbf{\Theta})&=\frac{\partial}{\partial\mathbf{\Theta}}\left(\mathbf{\Theta}^T\mathbf{X}_c^T\mathbf{X}_c\mathbf{\Theta}-\mathrm{Y}_c^T\mathbf{X}_c\mathbf{\Theta}-\mathbf{\Theta}^T\mathbf{X}_c^T\mathrm{Y}_c+\mathrm{Y}_c^T\mathrm{Y}_c+\lambda\mathbf{\Theta}^T\mathbf{\Theta}\right) \\
&=2\mathbf{X}_c^T\mathbf{X}_c\mathbf{\Theta}-\mathbf{X}_c^T\mathrm{Y}_c-\mathbf{X}_c^T\mathrm{Y}_c+2\lambda\mathbf{\Theta} \\
&=2\mathbf{X}_c^T\mathbf{X}_c\mathbf{\Theta}-2\mathbf{X}_c^T\mathrm{Y}_c+2\lambda\mathbf{\Theta}
\end{aligned}
$$

By letting $\frac{\partial}{\partial\mathbf{\Theta}}J_c(\mathbf{\Theta})=0$, we have

$$
\begin{aligned}
2\mathbf{X}_c^T\mathbf{X}_c\widehat{\mathbf{\Theta}} - 2\mathbf{X}_c^T\mathrm{Y}_c + 2\lambda\widehat{\mathbf{\Theta}} &= 0 \\
\mathbf{X}_c^T\mathbf{X}_c\widehat{\mathbf{\Theta}} - \mathbf{X}_c^T\mathrm{Y}_c + \lambda\widehat{\mathbf{\Theta}} &= 0 \\
(\mathbf{X}_c^T\mathbf{X}_c + \lambda\mathrm{I})\widehat{\mathbf{\Theta}} &= \mathbf{X}_c^T\mathrm{Y}_c \\
\widehat{\mathbf{\Theta}} &= \left(\mathbf{X}_c^T\mathbf{X}_c + \lambda\mathrm{I}\right)^{-1}\mathbf{X}_c^T\mathrm{Y}_c
\end{aligned}
$$

ps: $\mathbf{X}_c^T\mathbf{X}_c$ is positive semidefinite. When $\lambda > 0$, the matrix $\mathbf{X}_c^T\mathbf{X}_c + \lambda\mathrm{I}$ must be positive definite and therefore invertible.

2.8 套索回归

> Lasso Regression
>
> 引入 L1 正则.
>
> Lasso 回归对过拟合的缓解能力更胜一筹，因为它可以将某些系数精确地压缩到 0

$$
\widehat{\mathbf{\Theta}}^{\text{lasso}} = \arg\min_{\mathbf{\Theta}} \|\mathbf{X}\mathbf{\Theta} + \theta_0 - \mathbf{Y}\|_2^2 + \lambda\|\mathbf{\Theta}\|_1
$$

其中 $\|\Theta\|_1 = \sum_{i=1}^{n} |\theta_i|$ (a.k.a. L1 Norm), which is designed for measuring the sparsity of parameters.

Sparsity implies the number of zeros in Θ. If many elements in Θ are around zero, less features will be used, namely, the model complexity is reduced.

Lecture 3 逻辑回归

> Logistic Regression

3.1 分类问题

> Classification Problem

生活中有很多分类问题

例 1：Spam Detection - 垃圾邮件检测（二分类，是或不是）

例 2：Image Classification - 图像分类（二分类 / 多分类）

例 3：Cancer Detection - 癌症检测（二分类，有或没有）

目标：Given a set of samples, with different categories, learn a classifier. Then, the classifier can automatically recognize new samples within the set of given categories.

> 这里考虑监督学习 & Closed Set Recognition，即 Categories 有哪几个是已知的（pre-defined）.

> 与之相对的是 Open Set Recognition.

对比：分类 vs 回归

- The output of classification is a discrete value.
- The output of regression is a continuous value.

3.1.1 样本形式

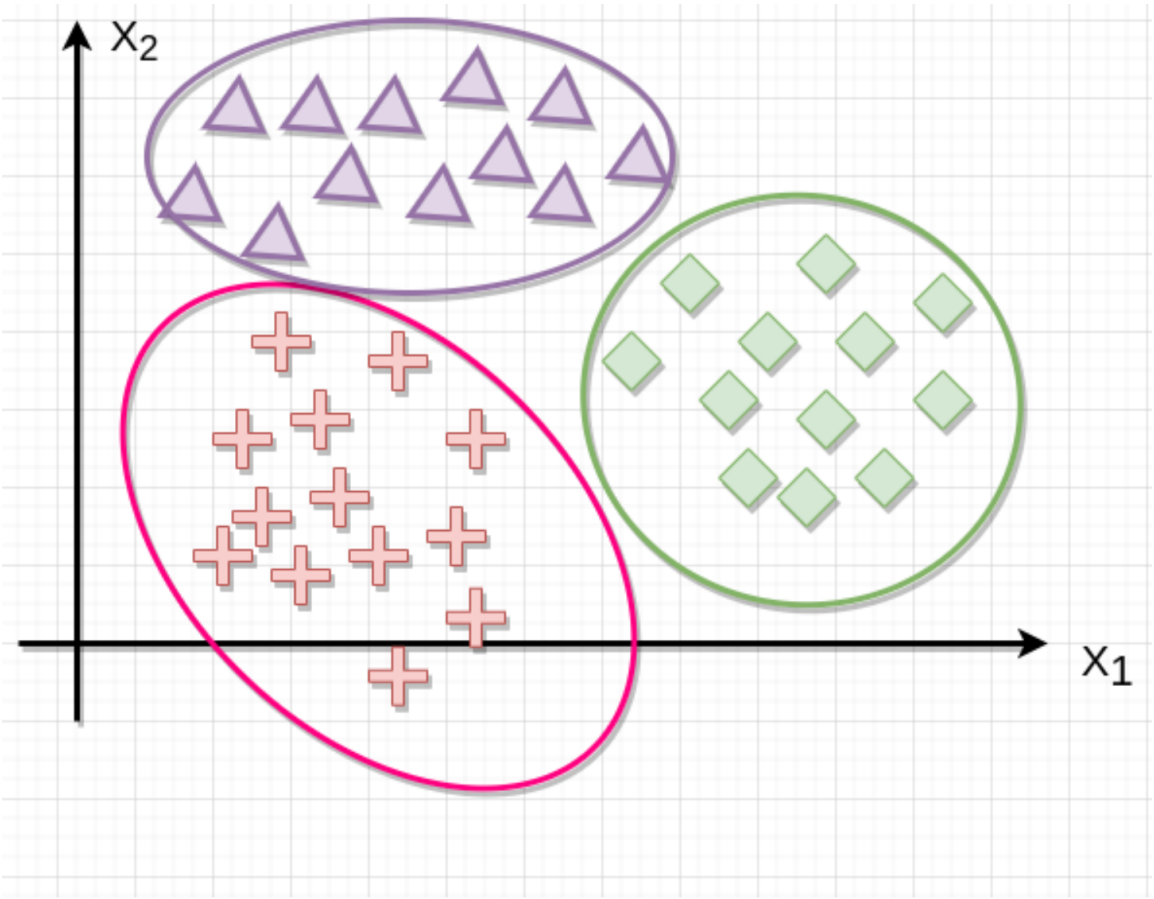
与回归问题类似，样本以 $(X^{(i)}, y^{(i)})$ 的形式成对出现. 其中，$X^{(i)}$ 仍然是该样本的特征向量（feature vectors），但 $y^{(i)}$ 是类别标签（category labels），而非回归问题中的连续值（如房租）.

Suppose we have m samples, each has n-dimensional features. We can write these samples as:

$$
\begin{aligned}
X^{(1)} &= (x_1^{(1)}, \ldots, x_n^{(1)}) \\
X^{(2)} &= (x_1^{(2)}, \ldots, x_n^{(2)}) \\
&\vdots \\
X^{(m)} &= (x_1^{(m)}, \ldots, x_n^{(m)})
\end{aligned}
$$

with their corresponding category labels as $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$.

例：An illustration of classification for samples with 2-dimensional features.



3.1.2 二分类 vs 多分类

> Binary vs. multi-class classification

二分类（Binary Classification）：Only two categories, usually, use $0, 1$ to label the two categories (0 for negative samples and 1 for positive samples).

例：

- Whether an email is spam or normal? (0 for spam, 1 for normal)
- Whether a photo contains a cat? (0 for no, 1 for yes)
- Whether you will be admitted to CUHK?

多分类（Multi-class classification）：More than two categories, usually, use $1, 2, 3, \ldots$, to label categories.

例：

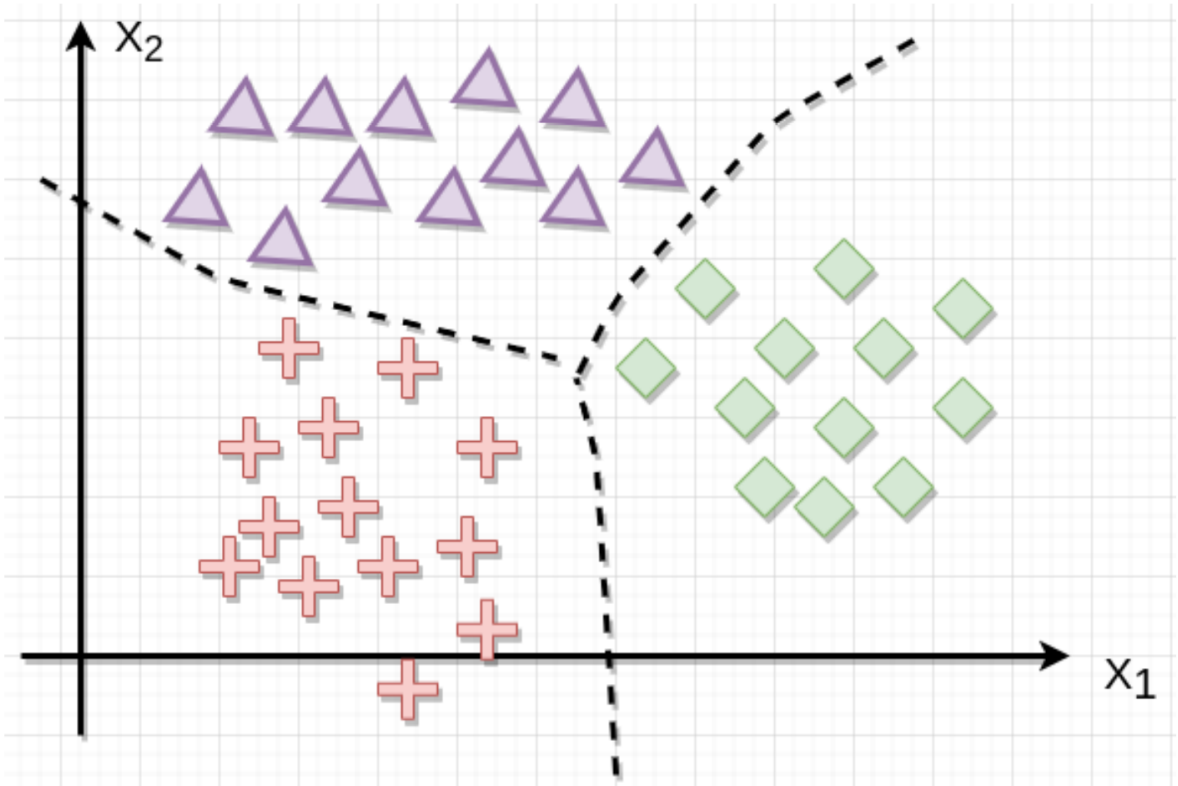
- Which animal is in the photo, cat / dog / pig / duck?
- Which CUHK College is a student associated with?

3.1.3 决策边界

> Decision boundary

Usually consider partitioning the entire feature space into multiple parts, one for each category (class). Data points associated to the same category lie in the same partition.

Intuitively, decision boundary is a union of curves or surfaces (多特征时) that partition the feature space.

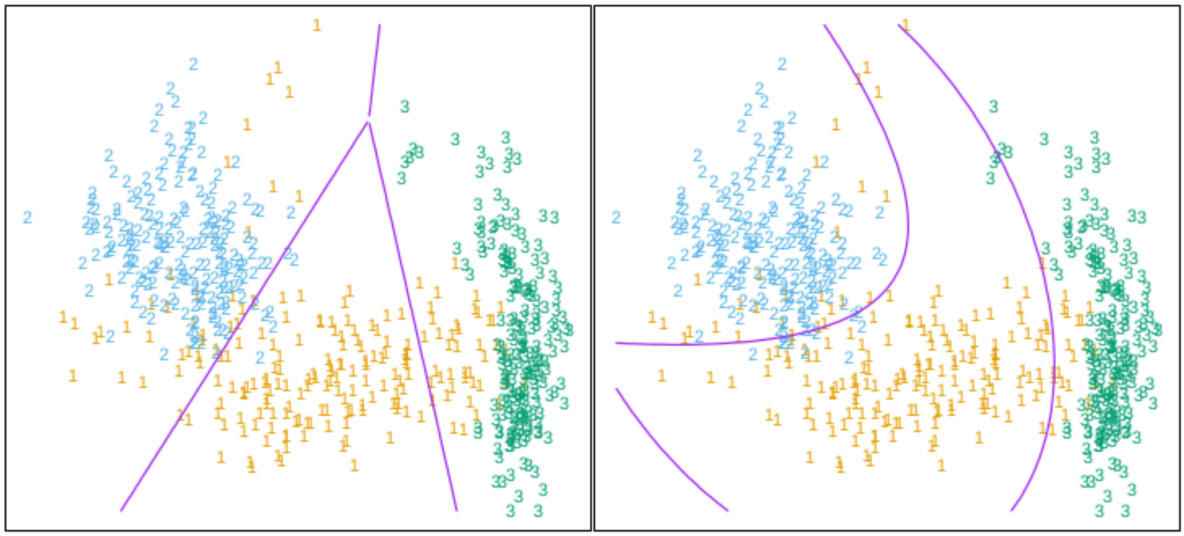


3.1.4 线性分类器

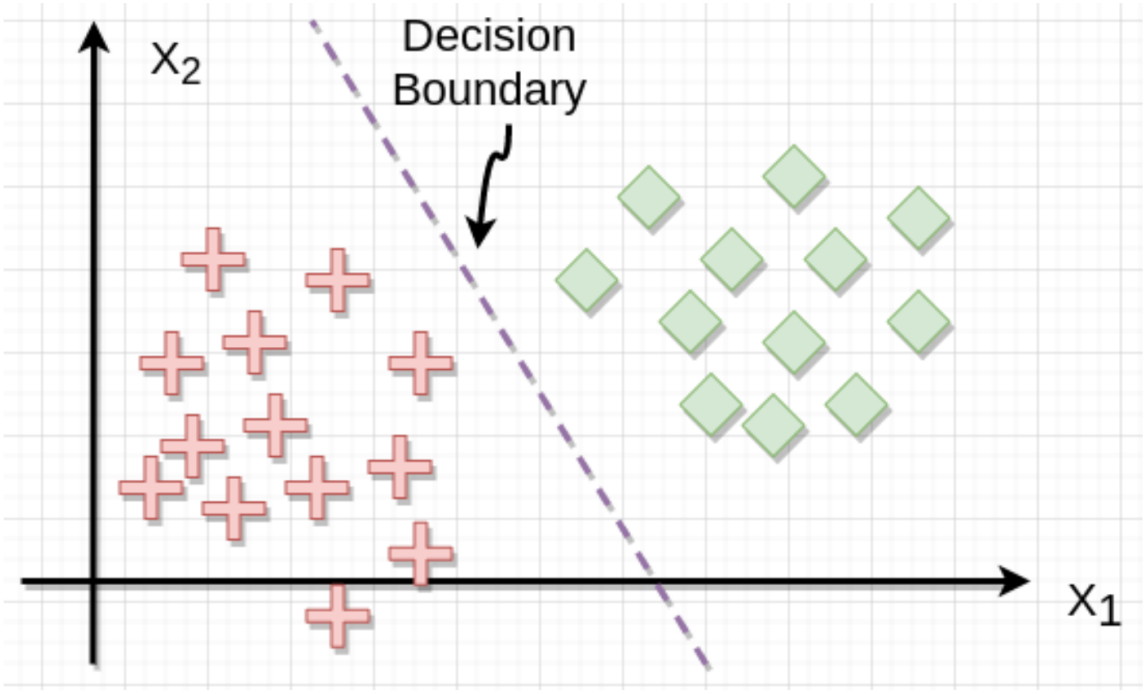
> Linear Classifier

Linear classifier is a kind of linear models, which classifies data based on a linear combination of input features.

Decision boundary of linear classifiers is a union of straight lines / hyperplanes, e.g., left figure.



For a linear binary classifier, its decision boundary is a straight line / hyperplane that partitions the feature space into two half-spaces.



3.2 逻辑回归

> Logistic regression for classification

见 `大二 term 2 CSCI 3320 机器学习 6.4 逻辑回归`.

Logistic regression models the probabilities for classification problems with two possible outcomes. 假设逻辑回归模型输出的值可以解释为目标变量为 1 的真实概率.

> 逻辑回归考虑使用 `Sigmoid` 激活函数解决二分类问题. 如果扩展到多分类，则用 `Softmax` 激活函数，称为 Softmax 回归. Softmax 回归可以被视为广义上的逻辑回归.
>
> 关于 Softmax 回归，见 `6.3.1 Softmax` .

简化起见，考虑 Linear Binary Classification.

Given an input n-dimensional feature $X \in \mathbb{R}^n$, and its possible category $y \in \{0, 1\}$.

By definition, for binary classification, we would like to construct a model to estimate $P(\hat{y} = 1|X)$, which is the probability that our prediction is 1 by giving X.

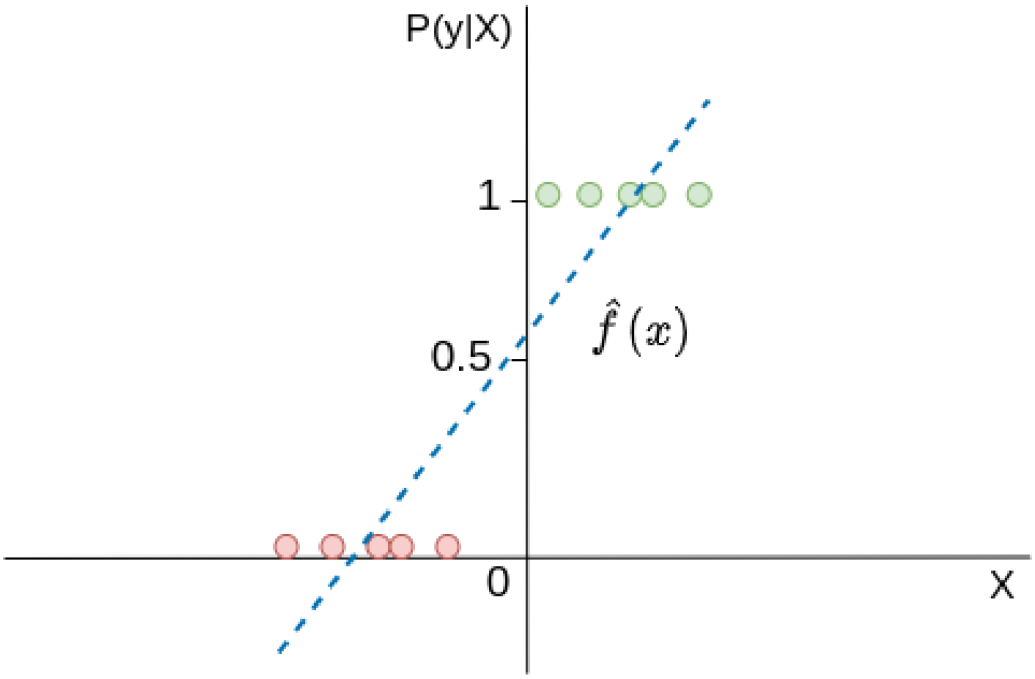
由全概率公式，得

$$P(\hat{y} = 0|X) + P(\hat{y} = 1|X) = 1$$

若 $P(\hat{y} = 1|X) \geq P(\hat{y} = 0|X)$，$y = 1$ 更可能发生，the sample with feature X is predicted to label 1 (i.e. $\hat{y} = 1$). Otherwise, the sample is labelled 0.

Equivalently, $P(\hat{y} = 1|X) \geq 0.5$，预测为 1; otherwise, 预测为 0.

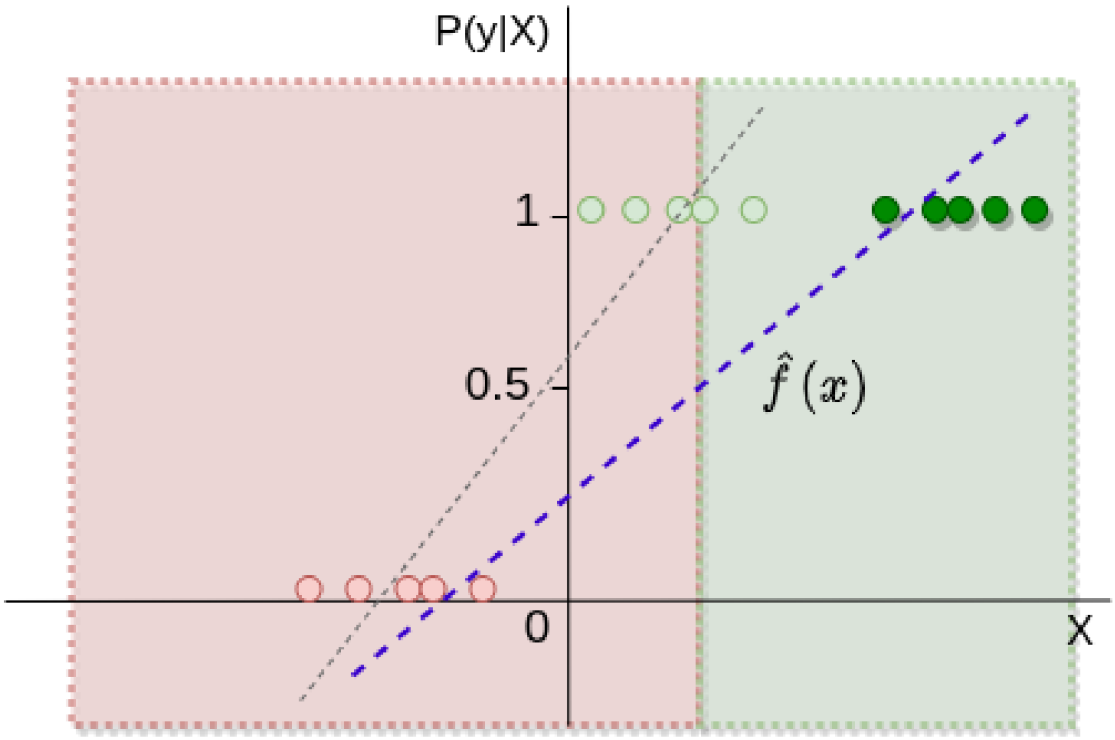
尝试用线性回归拟合：



显然失败，概率的值域是 $[0, 1]$，但线性回归的值域是 $(-\infty, +\infty)$.

考虑：一刀切（cut-off rule），若 $\hat{f}(X) \geq 0.5$，$\hat{y} = 1$; 若 $\hat{f}(X) < 0.5$, $\hat{y} = 0$.

似乎可行，但是受异常值影响太大：



一刀切是一种 map 值域 $(-\infty, +\infty) \to (0, 1)$ 的方式，但是太粗暴. 接下来引入一些数学处理方法让 mapping 更丝滑.

3.2.1 发生比 / 胜算 / 赔率

> Odds

已知对于 Event A，$P(A) = p$，$P(A^C) = 1 - p$.

The odds（发生比 / 胜算 / 赔率）of event A is defined as

$$\text{odds}(p) = \frac{P(A)}{P(A^C)} = \frac{p}{1-p}$$

The range of odds is $(0, +\infty)$.

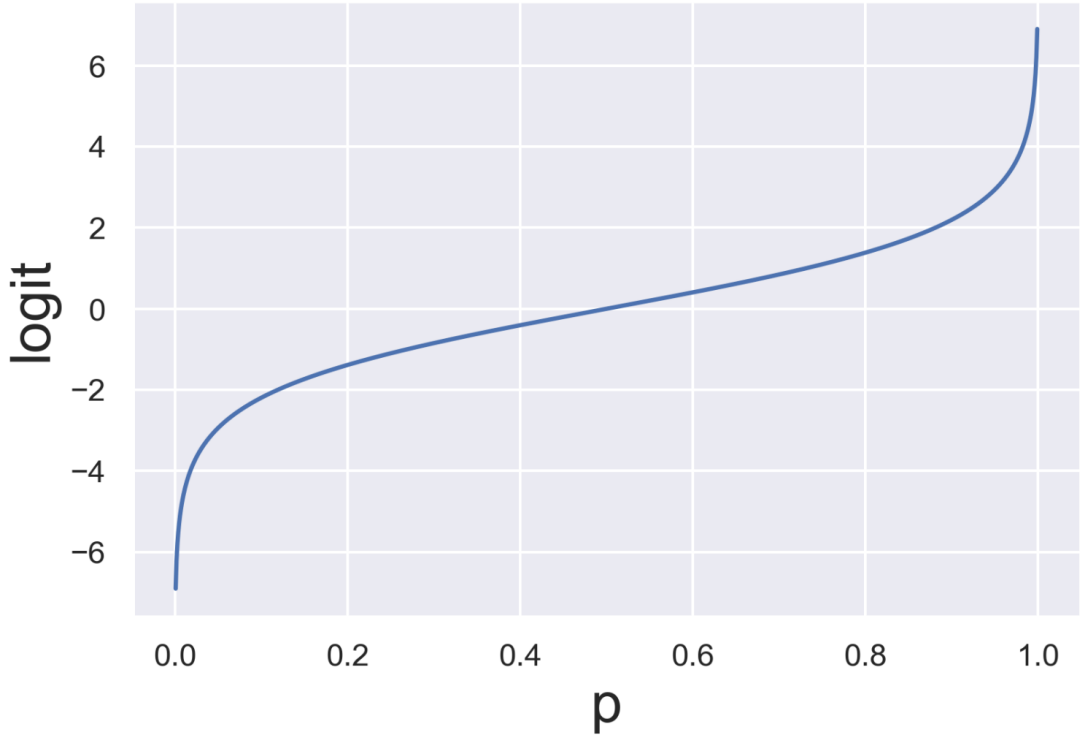
3.2.2 Logit 变换

> Logit transformation

We define logit (a.k.a. log-odds) of an event A as

$$\text{logit}(p) = \ln \text{odds}(p) = \ln\left(\frac{p}{1-p}\right)$$

whose range is $(-\infty, +\infty)$.



The $\text{logit}(p)$ is called logit transformation, which maps probability in range $(0, 1)$ to a value in $(-\infty, +\infty)$.

反向操作，即取反函数 $\text{logit}^{-1}(\cdot)$，就能得到另一种 mapping 值域 $(-\infty, +\infty) \to (0, 1)$ 的方式：

记映射前的预测值为 z. $\forall z \in (-\infty, +\infty)$, $\exists\, p \in (0, 1)$ such that

$$
\begin{aligned}
z = \text{logit}(p) &= \ln\left(\frac{p}{1-p}\right) \\
-z &= \ln\left(\frac{1-p}{p}\right) \\
e^{-z} &= \frac{1}{p} - 1 \\
p &= \frac{1}{1+e^{-z}}
\end{aligned}
$$

得 $p = \text{logit}^{-1}(z) = \frac{1}{1+e^{-z}}$.

Recall 线性回归模型：

$$\hat{f}_{\Theta,\theta_0}(X) = X^T\Theta + \theta_0.$$

对预测值 apply logit transformation 使其具有概率意义：

$$P(\hat{y} = 1|X) = \text{logit}^{-1}\left(X^T\Theta + \theta_0\right) = \frac{1}{1+e^{-(X^T\Theta+\theta_0)}}.$$

这样就能用来建模二分类问题的线性分类器了.

这样的回归方式叫逻辑回归（Logistic Regression），得到的决策边界是线性的（引入多项式特征后可变为非线性决策边界）.

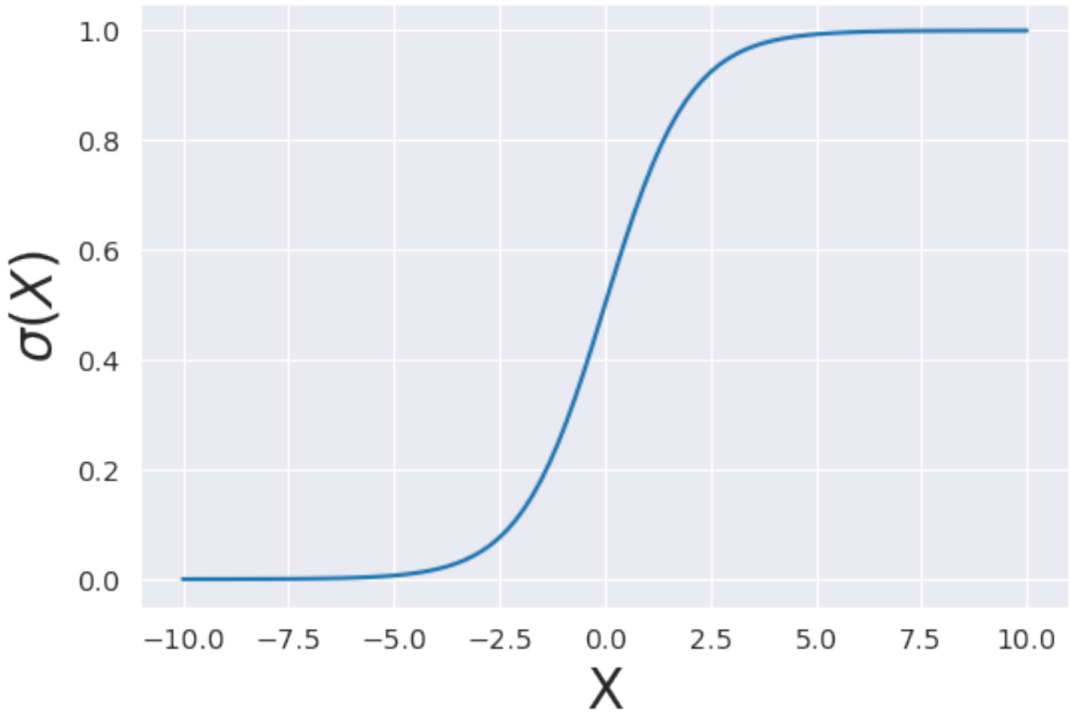
3.2.3 Sigmoid 函数

> Logistic Sigmoid Function

Logit 变换的反函数 $\text{logit}^{-1}(\cdot)$ 也叫做 Sigmoid 函数：

$$\sigma(x) = \text{logit}^{-1}(x) = \frac{1}{1+e^{-x}}$$

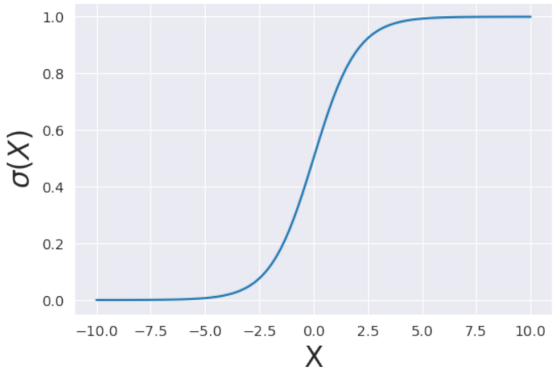
Sigmoid 函数的图像是 S 形 —— Sigmoid 函数也叫 S 型函数.

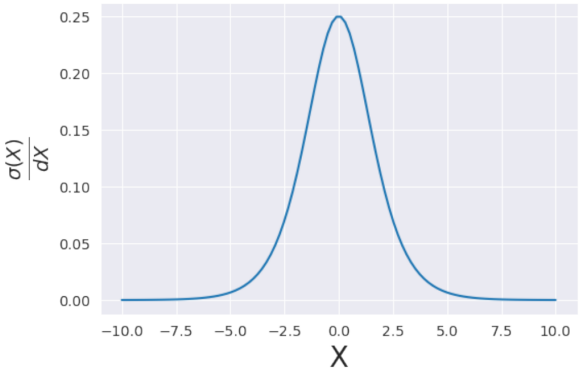


性质:

- $\sigma(0) = 0.5$;
- $\lim\limits_{x \to +\infty} \sigma(x) = 1$;
- $\lim\limits_{x \to -\infty} \sigma(x) = 0$;
- $\sigma(x)$ 单调递增（monotonic），它的一阶导是钟形（bell shaped）.

> 见 `Appendix 1.1 一阶导`.





3.2.4 偏置项

> bias term
>
> 见 `2.3.2 偏置项`.

Logistic Regression:

$$P(\hat{y} = 1|X) = \sigma\left(X^T\Theta + \theta_0\right) = \frac{1}{1 + e^{-(X^T\Theta + \theta_0)}}.$$

- 若 $P(\hat{y} = 1|X) \geq 0.5 \Rightarrow \hat{y} = 1$;
- 若 $P(\hat{y} = 1|X) < 0.5 \Rightarrow \hat{y} = 0$.

- Θ and θ_0 are parameters learned from training data.

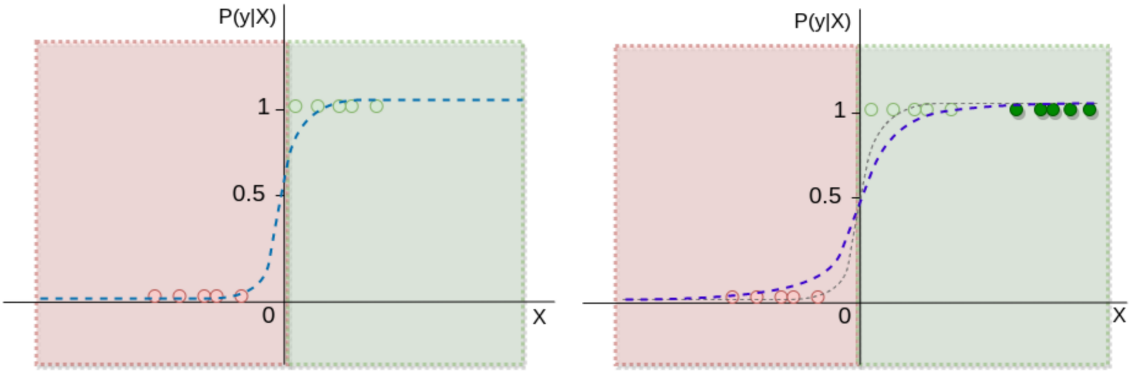
和线性回归的操作类似，简化起见，we absorb the θ_0 into Θ.

$$P(\hat{y}=1|X)=\sigma\left(X^T\Theta\right)=\frac{1}{1+e^{-(X^T\Theta)}}.$$

3.2.5 讨论

逻辑回归的优点：

- Logistic regression exactly outputs the probability of $y=1$ conditioned on the known input X.
- $P(\hat{y}=1|X)$ 在决策边界 $X^T\Theta=0$ 附近变化显著，这一特性能很好区分决策边界附近不同 labels 的样本.
- Logistic regression is more robust to outliers.



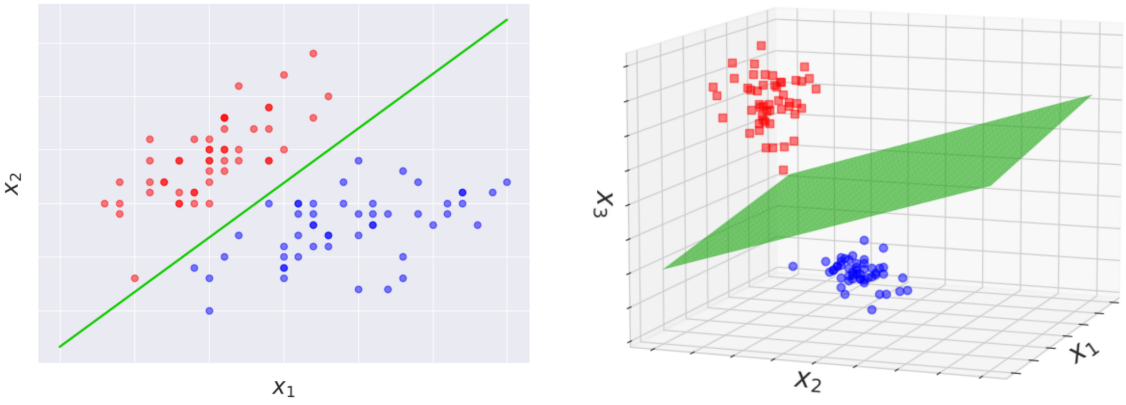
3.2.6 广义线性分类器

> Logistic regression is a (generalized) linear classifier

决策边界 $\sigma(X^T\Theta)=0.5\Leftrightarrow X^T\Theta=0$.

- $X^T\Theta\geq 0\Rightarrow\hat{y}=1$;
- $X^T\Theta<0\Rightarrow\hat{y}=0$.

几何上，$X^T\Theta=0$ 是一个超平面. $X^T\Theta\geq 0$ is the space above the hyperplane, while $X^T\Theta<0$ is the space below the hyperplane.



因此，逻辑回归模型得到的分类器是一种广义的线性分类器. 它预测的概率值和特征向量 X 不是线性关系，但是二分类的决策边界是线性的.

3.3 优化方法

> Optimizing logistic regression
>
> 见 `大二 term 2 CSCI 3320 机器学习 6.4.4 训练误差 & 6.4.5 梯度下降`.
>
> 这里的标签是 0 和 1，会推导出交叉熵形式的损失函数；CSCI 3320 采用 ± 1 标签，数学处理和导出的公式略有不同. 但是思路相似.

如何通过训练数据找到最优的 Θ? Recall logistic regression model:

$$P(\hat{y}=1|X)=\sigma\left(X^T\Theta\right)=\frac{1}{1+e^{-(X^T\Theta)}}.$$

> MLE，待完成.

Given that we have training data:

$$\{(x_1^{(1)},\dots,x_n^{(1)},y^{(1)}),\dots,(x_1^{(m)},\dots,x_n^{(m)},y^{(m)})\}$$

where $y^{(i)}\in\{0,1\}$ is the given corresponding training label.

For each sample $X^{(i)}=(x_1^{(i)},\dots,x_n^{(i)})$, we have that

$$P(\hat{y}^{(i)}=1|X^{(i)})=\sigma\left(X^{(i)^T}\Theta\right)=\frac{1}{1+e^{-\left(X^{(i)^T}\Theta\right)}}.$$

监督标准:

- 若 $y^{(i)}=1$，我们希望 $P(\hat{y}^{(i)}=1|X^{(i)})$ 尽可能接近 1;
- 若 $y^{(i)}=0$，我们希望 $P(\hat{y}^{(i)}=1|X^{(i)})$ 尽可能接近 0.

We can write the probability that the label is correctly predicted as:

$$p_i\overset{\text{def}}{=}P\left(\hat{y}^{(i)}=1|X^{(i)}\right)^{y^{(i)}}\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)^{1-y^{(i)}}$$

> 注意这里只是把分类讨论组合起来，不是在写全概率公式.

- When $y^{(i)}=0\Rightarrow p_i=P\left(\hat{y}^{(i)}=0|X^{(i)}\right)$;
- When $y^{(i)}=1\Rightarrow p_i=P\left(\hat{y}^{(i)}=1|X^{(i)}\right)$.

考虑 joint probability for all the training samples, and maximize it to learn Θ.

Since the samples are **independent to each other**, the joint probability can be written as:

$$\begin{aligned}p(\Theta)&\overset{\text{def}}{=}P\left(\hat{y}^{(1)}=y^{(1)},\dots,\hat{y}^{(m)}=y^{(m)}|X^{(1)},\dots,X^{(m)}\right)\\&=P\left(\hat{y}^{(1)}=y^{(1)}|X^{(1)}\right)\cdots P\left(\hat{y}^{(m)}=y^{(m)}|X^{(m)}\right)\\&=\prod_{i=1}^{m}P\left(\hat{y}^{(i)}=1|X^{(i)}\right)^{y^{(i)}}\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)^{1-y^{(i)}}\\&=\prod_{i=1}^{m}p_i\end{aligned}$$

3.3.1 最大似然估计

$p(\Theta)$ is also known as the likelihood function w.r.t. Θ. The method that estimates the parameters of a probabilistic model by maximizing a likelihood function is called **maximum likelihood estimation (MLE)**.

3.3.2 交叉熵损失

> Cross-entropy error function

目标：最大化似然 $p(\Theta)$.

Recall $p(\Theta)$:

$$\begin{aligned} p(\Theta) &\overset{\text{def}}{=} P\left(\hat{y}^{(1)}=y^{(1)},\ldots,\hat{y}^{(m)}=y^{(m)}|X^{(1)},\ldots,X^{(m)}\right) \\ &= P\left(\hat{y}^{(1)}=y^{(1)}|X^{(1)}\right)\cdots P\left(\hat{y}^{(m)}=y^{(m)}|X^{(m)}\right) \\ &= \prod_{i=1}^{m} P\left(\hat{y}^{(i)}=1|X^{(i)}\right)^{y^{(i)}}\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)^{1-y^{(i)}} \\ &= \prod_{i=1}^{m} p_i \end{aligned}$$

Maximizing $p(\Theta)$ is equivalent to minimizing $-\ln p(\Theta)$.

$$\begin{aligned} E(\Theta) &= -\ln p(\Theta) \\ &= -\ln\left(\prod_{i=1}^{m} p_i\right) \\ &= -\sum_{i=1}^{m}\ln p_i \\ &= -\sum_{i=1}^{m}\ln\left(P\left(\hat{y}^{(i)}=1|X^{(i)}\right)^{y^{(i)}}\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)^{1-y^{(i)}}\right) \\ &= -\sum_{i=1}^{m}\left[y^{(i)}\ln P\left(\hat{y}^{(i)}=1|X^{(i)}\right)+\left(1-y^{(i)}\right)\ln\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)\right] \end{aligned}$$

> 通常来说，求和相比乘积更好优化，因为它对于求导是线性的. 这里取 $\ln$ 是一个数学技巧，把乘积化为求和.

For each single sample, we define $E_i(\Theta)$ as

$$E_i(\Theta) = -y^{(i)}\ln P\left(\hat{y}^{(i)}=1|X^{(i)}\right)-\left(1-y^{(i)}\right)\ln\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right)$$

thus we rewrite $E(\Theta)=\sum_{i=1}^{m}E_i(\Theta)$.

Each $E_i(\Theta)$ is exactly in the form of **cross-entropy error function**:

$$\text{cross-entropy error} = -y\ln P(y)-(1-y)\ln\left(1-P(y)\right),\quad y\in\{0,1\}$$

> 交叉熵是信息论（information theory）的概念.

In our case, $P(y)=P(\hat{y}^{(i)}=1|X^{(i)})$, which is w.r.t. Θ.

When $y^{(i)}=0$,

$$E_i(\Theta) = -\ln\left(1-P\left(\hat{y}^{(i)}=1|X^{(i)}\right)\right) = -\ln\left(1-\frac{1}{1+e^{-X^{(i)T}\Theta}}\right)$$

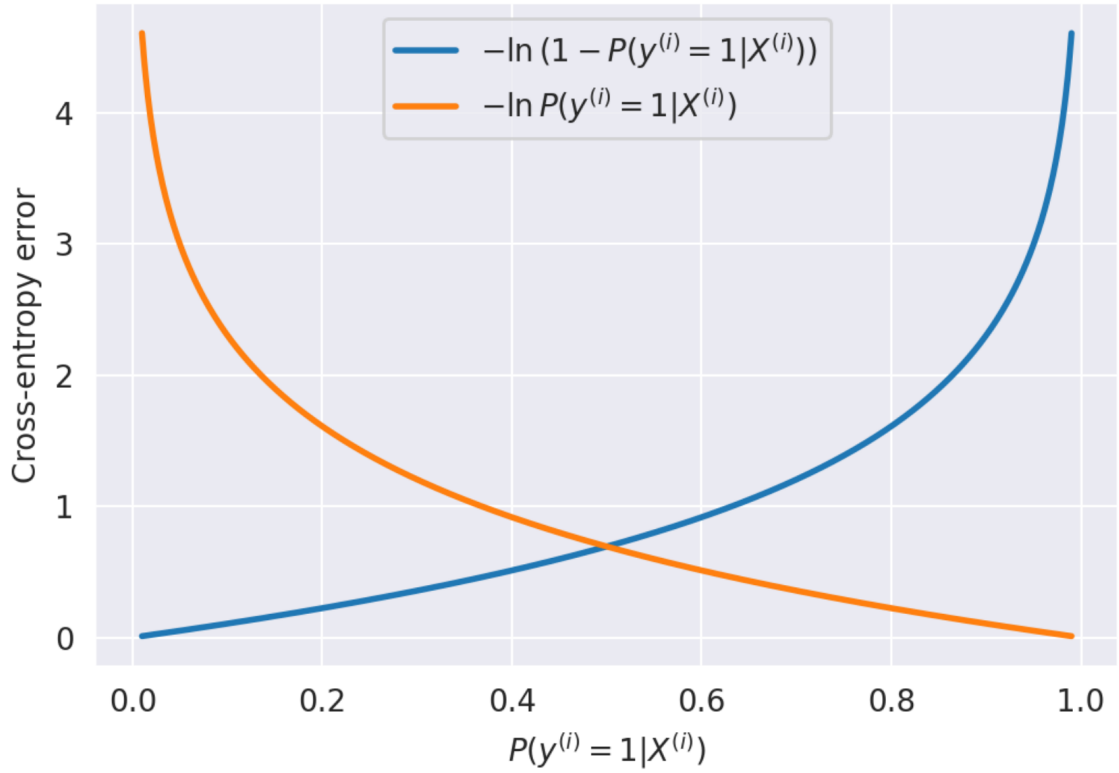
When $y^{(i)}=1$,

$$E_i(\Theta) = -\ln P\left(\hat{y}^{(i)}=1|X^{(i)}\right) = -\ln\left(\frac{1}{1+e^{-X^{(i)T}\Theta}}\right)$$

① 凸性分析

> Convex

作出 $E_i(\Theta)$ 的图像，which shows a good property of cross-entropy error function: **convex**.



3.3.3 梯度下降

> gradient descent

① 梯度

For a twice differentiable function $f:\mathbb{R}^n\to\mathbb{R}$, the gradient $\nabla f(\Theta)$ points in the direction of the steepest ascent or descent at Θ.

The gradient $\nabla f(\Theta)$ is a vector defined as:

$$\nabla f(\Theta)=\frac{\partial f(\Theta)}{\partial\Theta}=\left(\frac{\partial f(\Theta)}{\partial\theta_1},\dots,\frac{\partial f(\Theta)}{\partial\theta_n}\right)$$

where θ_i is the i-th element (dimension) of Θ.

方向导数（directional derivative）：一个标量场在某点沿着某个向量方向上的方向导数，描绘了该点附近标量场沿着该向量方向变动时的瞬时变化率.

By definition, the derivatives of $f(\Theta)$ along an arbitary direction is

$$D_{\mathbf{v}}f(\Theta)=\lim_{h\to0}\frac{f(\Theta+h\mathbf{v})-f(\Theta)}{h}$$

where $\mathbf{v}\in\mathbb{R}^n$, $\|\mathbf{v}\|_2=1$ is a unit vector along that direction.

如果一个标量场在某点沿任意方向的方向导数都存在，则其中必有最大的一个. 由柯西不等式可知，方向导数的最大值等于其梯度的范数，当且仅当沿着其梯度的方向时取到.

> 标量场某点梯度的方向是函数瞬时变化率最大（增长最快）的方向.

证明：见 `Appendix 2. 梯度最大化方向导数` .

② 梯度下降算法

> Gradient descent algorithm

梯度下降沿梯度的反方向更新参数，通过迭代最小化目标函数 $f(\Theta)$.

> 反方向：下降最快的方向.

Gradient descent

Ensure: $\alpha > 0$

Initialize $\Theta \leftarrow \Theta_0$ randomly

while not converge **do**

$\Theta \leftarrow \Theta - \alpha \nabla f(\Theta)$

end while

α：学习率（learning rate），超参数，人为设置.

> 太小：学太慢.
>
> 太大：难以收敛.
>
> 自适应学习率：Allowed to change at every iteration.
>
> 注意，写代码时一定要看学习率有没有按要求设置. 有时复制代码会忘记改学习率.

③ 优化交叉熵损失

> Find the optimal Θ for logistic regression

Recall `3.3.2 交叉熵损失`. For each single sample, we define $E_i(\Theta)$ as

$$E_i(\Theta) = -y^{(i)} \ln P\left(\hat{y}^{(i)} = 1 | X^{(i)}\right) - \left(1 - y^{(i)}\right) \ln\left(1 - P\left(\hat{y}^{(i)} = 1 | X^{(i)}\right)\right)$$

thus we rewrite $E(\Theta) = \sum_{i=1}^{m} E_i(\Theta)$.

The analytic expression of $E(\Theta)$'s gradient is

$$\nabla E(\Theta) = \frac{\partial E(\Theta)}{\partial \Theta} = \sum_{i=1}^{m} \left(\frac{1}{1 + e^{-X^{(i)T}\Theta}} - y^{(i)} \right) X^{(i)}$$

It seems we cannot find an explicit solution to this equation...

代入梯度下降公式，得到逻辑回归的梯度下降优化算法：

Gradient descent for logistic regression

Ensure: $\alpha > 0$

Initialize Θ randomly

while not converge **do**

$\Theta \leftarrow \Theta - \alpha \sum_{i=1}^{m} \left(\frac{1}{1+e^{-X^{(i)T}\Theta}} - y^{(i)} \right) X^{(i)}$

end while

Lecture 4 支持向量机

> Support Vector Machine (SVM)

4.1 线性分类器

> Linear classifier
>
> 见 `3.1.4 线性分类器`.

分类（Classification）：识别、区分和理解想法与对象的过程. 目标是学习一个分类器，能够自动识别属于给定类别的新样本.

例：将邮件分类为 “垃圾邮件” 或 “非垃圾邮件”.

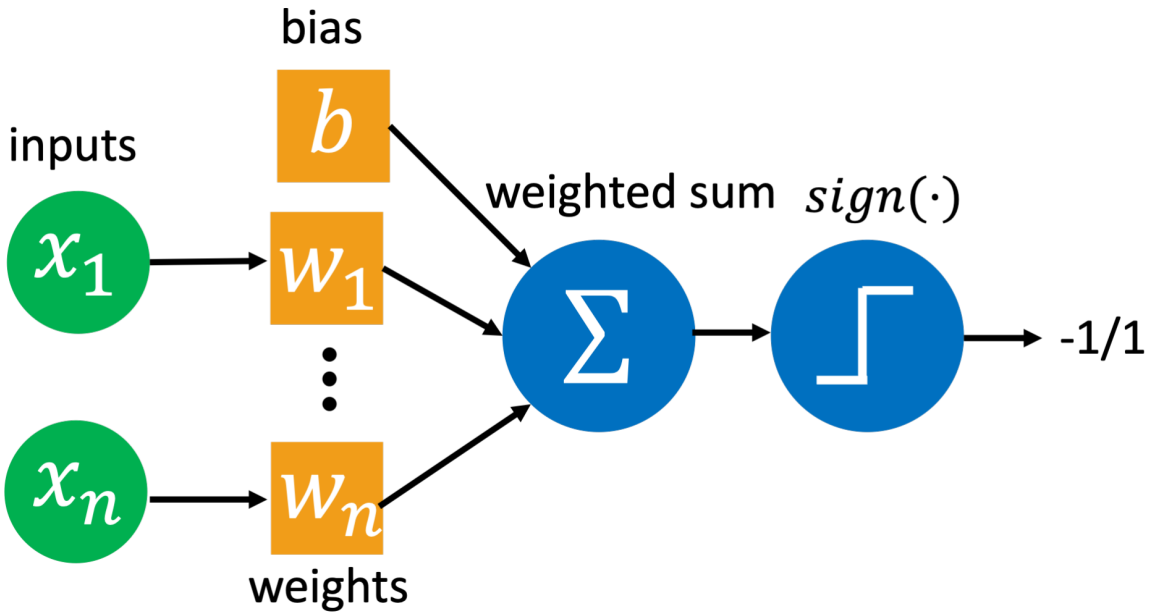
线性分类器：一种基于输入特征的线性组合进行数据分类的模型. 它利用一个决策边界（decision boundary，一个超平面）将输入数据划分为两个集合，分别对应一个类别.

> 注意，基于逻辑回归的分类器也是线性分类器. 虽然逻辑回归模型本身预测的概率值和输入特征 X 并非线性关系，但是决策边界是线性的.

Example - 一个简单的二元线性分类器

使用方程 $h(x) = \mathbf{w}^T\mathbf{x} + b = 0$ 定义超平面，预测结果由 $\mathbf{w}^T\mathbf{x} + b$ 的正负号决定. 大于 0 为正样本，小于 0 为负样本.

$$\hat{y} = \text{sign}(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b > 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \end{cases}$$



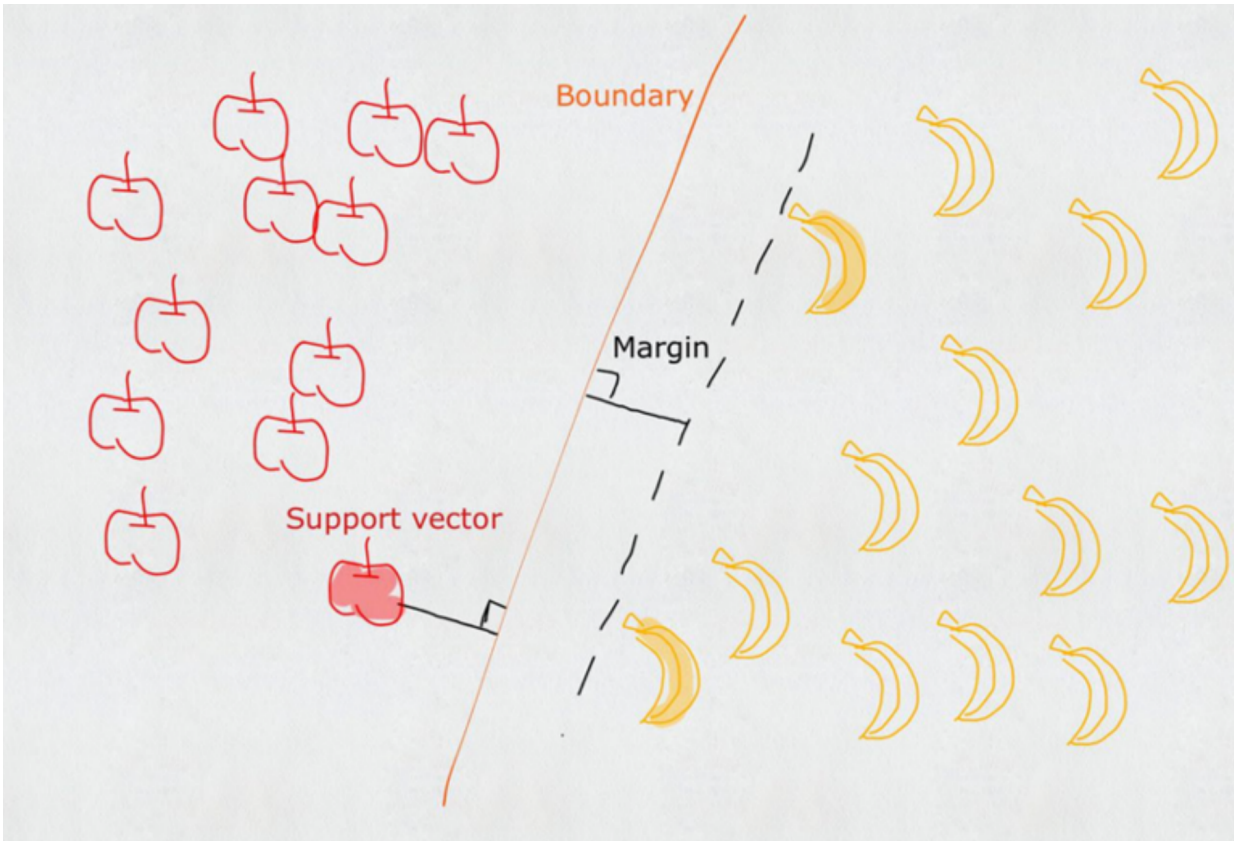
4.2 支持向量机

> Support Vector Machine (SVM)

SVM 目标：寻找一个作为决策边界的分离超平面. 对于线性可分的数据，存在多个超平面，SVM 的目标是找到 "最胖" 的超平面.

最大化间隔（Margin）："最胖" 的超平面意味着它到最近数据点的距离最大. 这个距离（的两倍）被称为间隔（Margin）. 最大化间隔可以提高分类器的鲁棒性和容噪能力（noise tolerance）.

> 在后面的定义中，Margin 定义为到最近点距离的两倍，指的是两个间隔边界之间的距离.



4.2.1 点到超平面距离公式

Assume a hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that can correctly classify the samples.

Give an example point $\mathbf{x}_i$，the distance between $\mathbf{x}_i$ and the hyperplane is

$$\gamma = \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

where $\mathbf{x}_i$ and $\mathbf{w}$ are both d-dimensional vectors.

> $\mathbf{x}_i$ 不是 $\mathbf{x}$ 在某个维度的分量，而是一个 d 维向量.
>
> 推导见 `Appendix 3. 点到超平面距离公式` .

SVM 找最大间隔超平面，实际上是一个优化问题：

$$(\mathbf{w}^*, b^*) = \arg\max_{\mathbf{w},b} \left(\min_i \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|} \right)$$

但是有一些困难：

① 难以求解；

> 优化目标无法简化，无法应用对偶理论和核技巧.

② 优化目标具有冗余自由度.

> 就算找到了某个 “最优解” $(\mathbf{w}^*, b^*)$，把它们同步缩放任意倍数到 $(\alpha\mathbf{w}^*, \alpha b^*)$，超平面保持不变，仍是一个 “最优解”. 即 "最优解" 有无数个，但它们实际上表示同一个超平面. 这样，由于这种尺度不变性，优化器可以沿着 $(\mathbf{w}^*, b^*)$ 的尺度伸缩方向任意移动，计算效率低下，难以收敛，解缺乏唯一性.

4.2.2 归一化约束

为了解决上述两个困难，我们引入归一化约束.

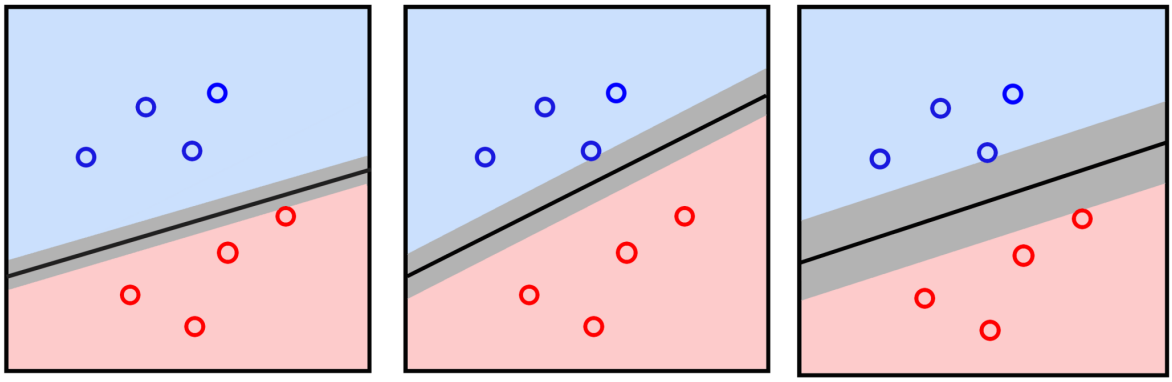
我们知道优化目标具有冗余自由度，即 $(\mathbf{w}, b) \to (\alpha\mathbf{w}, \alpha b)$ 不改变超平面，也不改变各个点到超平面的距离. 但是，仔细观察距离公式：

$$\gamma = \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|}$$

$(\mathbf{w}, b) \to (\alpha\mathbf{w}, \alpha b)$ 时，点到超平面距离是因为分子分母同时缩放相互抵消，才不变的. 那么，如果我们能通过某些特殊点限制分子或分母，使得 $(\mathbf{w}, b)$ 在变化时必须满足限制，就能把冗余的复杂度去掉了.

对于线性可分的数据，任找一个能作为决策边界的分离超平面（不一定最优）. 先不考虑角度问题，从平移的角度，如果 $+1$ 和 -1 的最近点到它的距离不同，那么一定能通过平移找到一个更优的解. 所以最优解一定满足 $+1$ 和 -1 的最近点到它的距离相同.

> 注意，最优解满足这个约束，但满足这个约束的不一定最优.



把这个距离计为 $\gamma_{\min} = \frac{|\mathbf{w}^T\mathbf{x}_+ + b|}{\|\mathbf{w}\|} = \frac{|\mathbf{w}^T\mathbf{x}_- + b|}{\|\mathbf{w}\|}$. 把 $(\mathbf{x}_+, y_+)$ 和 $(\mathbf{x}_-, y_-)$ 称为支持向量（support vector）.

> 注意，到这里都还没有进行归一化约束，即使满足这个很弱的 "$+1$ 和 -1 的最近点到它的距离相同" 约束，最优解还是可以任意伸缩.
>
> 支持向量有两个，把 $+1$ 标签的那个计为 $(\mathbf{x}_+, y_+)$，叫做正支持向量（positive support vector）；-1 标签的那个计为 $(\mathbf{x}_-, y_-)$，叫做负支持向量（negative support vector）. 课件把它们统一写作 $(\mathbf{x}_s, y_s)$.

接下来添加第二个约束来消除 $\mathbf{w}$ 和 b 的冗余自由度：

对于任意 $\gamma_{\min}$，直接约束 $|\mathbf{w}^T\mathbf{x}_+ + b| = |\mathbf{w}^T\mathbf{x}_- + b| = 1$. 注意，这个约束不是说最优解必然满足，而是我们只要满足这个约束的最优解（其他最优解即通过缩放约束 $|\mathbf{w}^T\mathbf{x}_+ + b| = |\mathbf{w}^T\mathbf{x}_- + b| = \alpha$ 得到，但是都等价，没有意义）.

此时，$\gamma_{\min} = \frac{1}{\|\mathbf{w}\|}$，它表示最近点到超平面的距离. 对于任意数据点 $(\mathbf{x}_i, y_i)$，必然满足 $|\mathbf{w}^T\mathbf{x}_i + b| \geq 1$. 定义间隔（margin）为 $\gamma = 2\gamma_{\min} = \frac{2}{\|\mathbf{w}\|}$.

因为这是一个线性分类器，即

$$y_i = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x}_i + b > 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x}_i + b < 0 \end{cases}$$

那么由于上述限制，这个分类器等价于

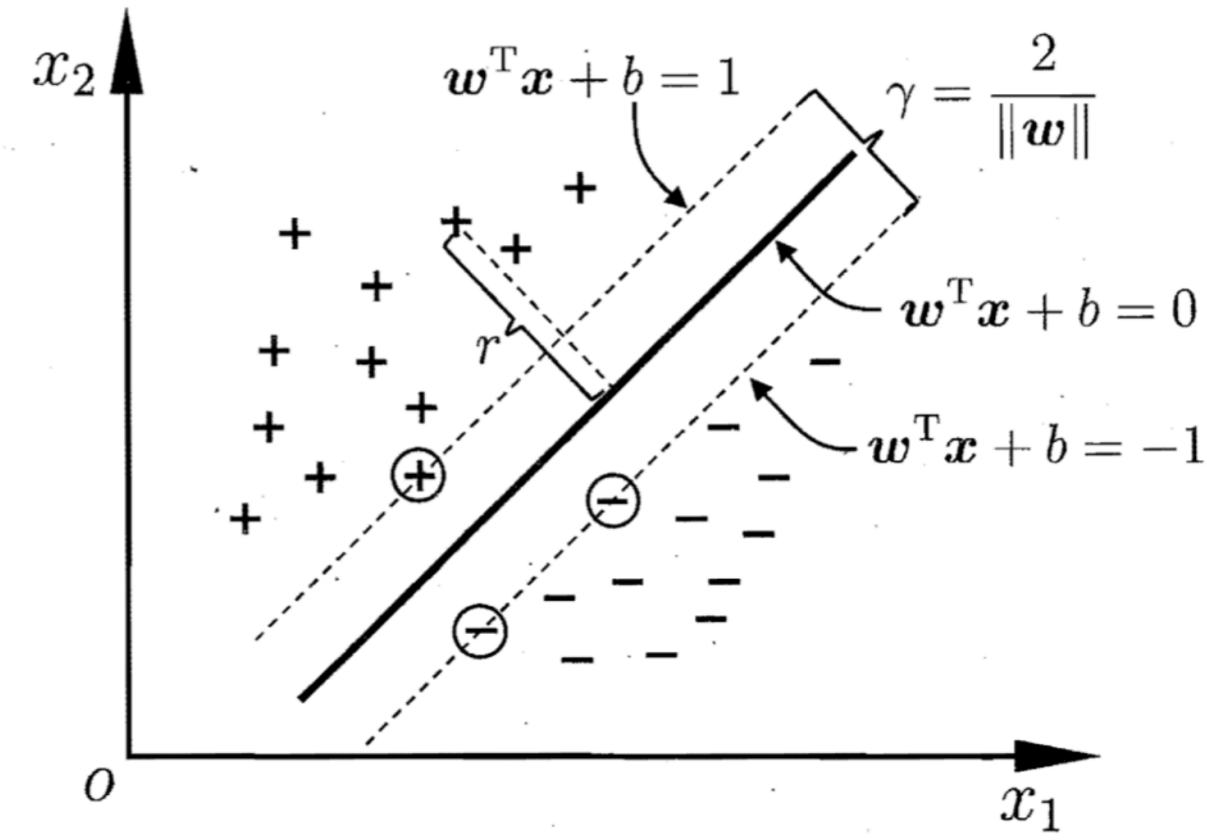
$$y_i = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x}_i + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x}_i + b \leq -1 \end{cases}$$

> 因为在这个限制下，只要 $\mathbf{w}^T\mathbf{x}_i + b > 0$ 就一定有 $\mathbf{w}^T\mathbf{x}_i + b \geq 1$，只要 $\mathbf{w}^T\mathbf{x}_i + b < 0$ 就一定有 $\mathbf{w}^T\mathbf{x}_i + b \leq -1$.
>
> 至于 $(-1, 1)$ 之间，不需要定义，因为这之间没有数据点.

添加这两个限制之后，优化问题变为

$$\begin{aligned} (\mathbf{w}^*, b^*) &= \arg\max_{\mathbf{w},b} \left(\min_i \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|} \right) \\ &= \arg\max_{\mathbf{w},b} \left(\frac{1}{\|\mathbf{w}\|} \right) \\ &= \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \end{aligned}$$

4.2.3 几何意义



4.2.4 优化目标

通过数学推导可以把上述约束整合为标准形式. 优化目标为:

$$\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1$$

> 注意，这个优化目标和约束可以推出最优解满足 `4.2.2 归一化约束` 中的两个约束. 详细说明待补充.

4.2.5 拉格朗日乘子法

> Lagrange multipliers
>
> 见 `大二 term2 CSCI 3320 机器学习 9.2.3 拉格朗日乘子法`.

优化目标:

$$\begin{aligned}&\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1\\ \Leftrightarrow&\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b)\leq 0, i=1,\dots,m\end{aligned}$$

> m 个训练数据点.

这是不等式约束，构造拉格朗日函数（Lagrangian）：

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\|\mathbf{w}\|^2+\sum_{i=1}^{m}\alpha_i(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))$$

For each sample $(\mathbf{x}_i, y_i)$, it has a corresponding Lagrange multiplier $\alpha_i \geq 0$.

> 当某些样本约束不紧，即 $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0$ 时，它们对应的拉格朗日乘子 α_i 为 0（互补松弛性）.
>
> 几何意义为，如果我们找到某个最优解，对于某个非支持向量的样本点，显然它带来的约束不起作用，即最优解并不在这个约束的边界上（但是仍然满足这个约束，可以理解为在边界之内）. 那么该约束为非紧绷约束，不需要参与非负线性组合锥的张成.
>
> 对于多约束，最优解时目标函数的梯度必须落在紧绷约束的梯度的非负线性组合锥中，否则可以进一步下降.

最优解满足拉格朗日量对优化变量的偏导为 0：

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m}\alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

$$\begin{cases} \frac{\partial L}{\partial \mathbf{w}} = 0 \\ \frac{\partial L}{\partial b} = 0 \end{cases} \Rightarrow \begin{cases} \mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i \\ 0 = \sum_{i=1}^{m}\alpha_i y_i \end{cases}$$

> $0 = \sum_{i=1}^{m}\alpha_i y_i$ 称为平衡约束（Balance Constraint），即如果存在正支持向量，那么一定存在负支持向量（数量不一定相等），反之亦然.

4.2.6 对偶问题

注意，上面令拉格朗日量对 $\mathbf{w}$ 和 b 偏导为 0 并没有完全解决问题，因为拉格朗日乘子的值还未知. 把上述含拉格朗日乘子的解代回拉格朗日量，经过数学处理，得到对偶问题（dual problem）

$$\max_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$
$$\text{s.t.} \sum_{i=1}^{m}\alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

> 见 `Appendix 4. 对偶问题`.

This optimization problem is a quadratic programming (QP，二次规划) problem. There are many effective algorithm can solve this problem, such as SMO (Sequential Minimal Optimization)

对于对偶问题，使用 KKT 条件：

$$\begin{cases} \alpha_i \geq 0 \\ y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \\ \alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) = 0 \end{cases}$$

- 第一条：拉格朗日乘子非负.
- 第二条：所有点满足约束. 在支持向量处取等号.
- 第三条：互补松弛性，非支持向量对应的乘子为 0，支持向量对应的乘子 > 0.

> 注意，数学上，位于边界的点（支持向量）的 α_i 可以为 0，但是 SVM 惯例上，我们只将 $\alpha_i > 0$ 的点称为支持向量. 在大多数教科书和实现中，位于边界的支持向量的拉格朗日乘子一定大于 0.

在优化对偶问题

$$\max_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$
$$\text{s.t.} \sum_{i=1}^{m}\alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

得到最优解 α^*（这是一个向量，有 m 个分量，m 是训练数据点的个数）后，可以代入 $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ 得到最优 $\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i$. 然后选取任意一个支持向量 $(\mathbf{x}_s, y_s)$，由 $y_s(\mathbf{w}^T \mathbf{x}_s + b) = 1$ 得到 $b^* = \frac{1}{y_s} - \mathbf{w}^{*T} \mathbf{x}_s$, where $s \in S$ and $S = \{i : a_i > 0, i = 1, 2, \ldots, m\}$（the set of indices of support vector）.

> 支持向量很好找，只要找 α 中的非零项，它们对应的点就是支持向量.

Pratically, we suggest averaging over all eligible support vectors to alleviate numerical error:

$$b^* = \frac{1}{|S|} \sum_{s \in S} (\frac{1}{y_s} - \mathbf{w}^{*T} \mathbf{x}_s)$$

综上，得到最优超平面：

$$\begin{cases} \mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i \\ b^* = \frac{1}{|S|} \sum_{s \in S} (\frac{1}{y_s} - \mathbf{w}^{*T} \mathbf{x}_s) \end{cases} \Rightarrow \mathbf{w}^{*T} \mathbf{x} + b^* = 0$$

4.2.7 SMO 算法

> 待补充.

现在唯一的问题只剩下：如何优化对偶问题

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$
$$\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

得到最优解 α^*.

介绍序列最小优化（SMO）算法.

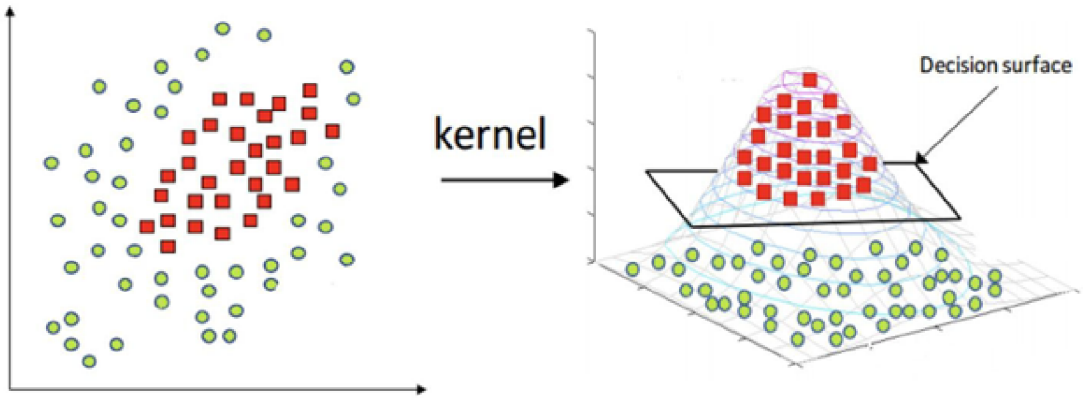
> 待补充.

本课只要求调用 SMO 的 toolbox 来得到最优解 α^*.

4.3 非线性可分数据集的 SVM

> SVM for non-linear datasets

对于非线性可分的数据，如果增加维度，可能可以变为线性可分：



有很多方法来增加特征维度. 下面给出用**核函数**进行变换的方法.

4.3.1 高维变换

对于线性不可分的二维数据集 $\begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix}$，可以使用变换转三维：

$$\phi(\mathbf{x}) = \begin{pmatrix} (x^{(1)})^2 \\ (x^{(2)})^2 \\ \sqrt{2}x^{(1)}x^{(2)} \end{pmatrix}$$

数据集在这个三维空间中可能就是线性可分的了.

> 注意：
>
> - 该变换的系数和形式是为了满足后续的核技巧精心计算出来的；如果我们任意定义一个映射，就无法使用核技巧.
> - 二维转三维并不能保证一定可分，这只是一个例子. 但是，由于核函数和核技巧通用性，可以将数据不断升维，直到一个线性可分的维度为止.

经过该变换，对偶问题变为

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$
$$\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

又有新的问题：very large feature spaces have potential issue of memory and computational cost. The “kernel trick” can help alleviate this issue.

4.3.2 核技巧

> Kernel Trick

如果我们对原始数据显式地进行变换，即一一计算：

$$\phi(\mathbf{x}) = \begin{pmatrix} (x^{(1)})^2 \\ (x^{(2)})^2 \\ \sqrt{2}x^{(1)}x^{(2)} \end{pmatrix}$$

那么，一旦变换后的维度很高，计算量将非常大.

解决方法：引入核技巧.

因为实际上的优化问题为

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$
$$\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

即 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ 的计算都是成对进行的. 如果我们直接对 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ 进行定义，即

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

which means $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the function of $\mathbf{x}_i$ and $\mathbf{x}_j$, and we don't need to calculate $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ explicitly（显式地）. The $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is called kernel function.

It allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.

对偶问题转化为：

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m$$

> 注意，不是所有 function of $\mathbf{x}_i$ and $\mathbf{x}_j$ 都可以作为核函数.
>
> 待补充：Mercer 定理.

用 SMO toolbox 解得最优 α^*，（高维变换后的）最优解为

$$\begin{cases} \mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \phi(\mathbf{x}_i) \\ b^* = \frac{1}{|S|} \sum_{s \in S} (\frac{1}{y_s} - \mathbf{w}^{*T} \phi(\mathbf{x}_s)) \end{cases}$$

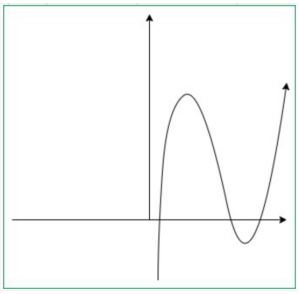
最优超平面为 $\mathbf{w}^{*T} \phi(\mathbf{x}) + b^* = 0$，即

$$\sum_{i=1}^{m} \alpha_i^* y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + \frac{1}{|S|} \sum_{s \in S} (\frac{1}{y_s} - \sum_{i=1}^{m} \alpha_i^* y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_s)) = 0$$
$$\Rightarrow \sum_{i=1}^{m} \alpha_i^* y_i \kappa(\mathbf{x}_i, \mathbf{x}) + \frac{1}{|S|} \sum_{s \in S} (\frac{1}{y_s} - \sum_{i=1}^{m} \alpha_i^* y_i \kappa(\mathbf{x}_i, \mathbf{x}_s)) = 0$$

> 注意，最优超平面也不需要显式计算 $\phi(\cdot)$

一些常用核函数：

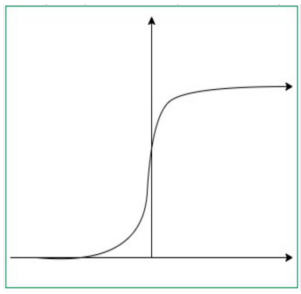
Polynomial Kernel：



$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^T \boldsymbol{x_j})^d$

It represents the similarity of vectors in training set of data in a feature space over polynomials of the original variables used in kernel.

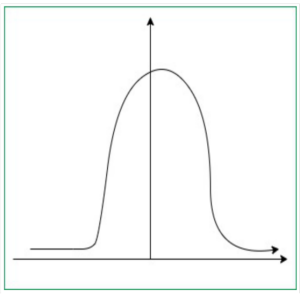
Sigmoid Kernel：



$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = tahn(\beta \boldsymbol{x_i}^T \boldsymbol{x_j} + \theta)$

This function is equivalent to a two-layer, perceptron model of neural network, which is used as activation function for artificial neurons.

Gaussian Kernel：



$\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\sigma^2})$

It is used to perform transformation, when there is no prior knowledge about data.

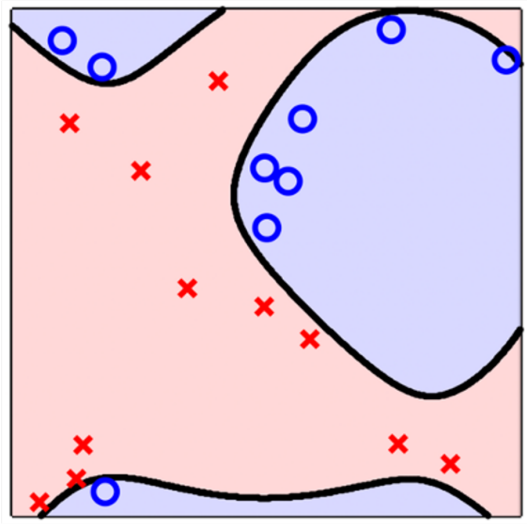
4.4 软间隔

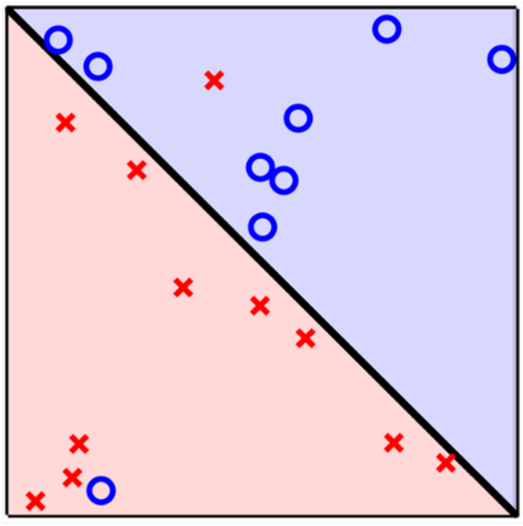
> Soft-margin SVM

Sometimes it is still **difficult** to find a proper kernel function to make the data **completely separable**.

Even though we find a kernel function to make the data separable, it may be faced with **overfitting** problem.

In this case, we can give up some noisy examples.





软间隔：允许一些 “带噪声” 的样本点违反分离条件.

4.4.1 Hinge Loss

优化目标：在最小化 $\frac{1}{2}\|\mathbf{w}\|^2$ 的同时，增加一个惩罚项来惩罚被错误分类或违反间隔的样本. 常用的惩罚函数是 Hinge Loss（铰链损失），将优化目标重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

> 满足约束的点，不惩罚；不满足约束的点，惩罚. 且越不满足惩罚越大.
>
> C 是人为设定的超参数. C 很大，则惩罚很大，模型倾向于不让数据点违规，但是可能导致过拟合；C 很小，则模型主要目标是最大化间隔，几乎忽略了误分类的点.
>
> 我们需要权衡二者，来找到合适的 C.

4.5* 拉格朗日乘子法

> Lagrange multipliers
>
> ESTR content

4.6* 对偶性

> Duality
>
> ESTR content

4.7* KKT 条件

> Karush-Kuhn-Tucker (KKT) conditions
>
> ESTR content

4.8* 优化 SVM

> Optimizing SVM
>
> ESTR content

Lecture 5 聚类算法

> Clustering Algorithms

聚类算法 (Clustering Algorithms) 是一种**无监督学习** (Unsupervised Learning) 技术，用于将数据集中的对象或数据点根据它们的相似性 (similarity) 组织和分组，形成簇 (clusters).

简单来说，聚类算法就是寻找数据中天然的、隐藏的分组或模式. 它不需要事先知道数据点的标签（即无监督），而是通过分析数据点之间的特征或距离来决定哪些点应该归为一类.

目标：确保同一簇的数据点高度相似，不同簇的数据点高度不相似.

相似性度量：通常使用距离（如欧几里得距离）来衡量. 两个数据点距离越近，相似度越高.

应用场景：

- 市场细分（Market Segmentation）：根据购买行为或人口统计信息将客户分组.
- 异常检测（Anomaly Detection）：将不属于任何主要簇的数据点识别为异常值（Outliers）
- 图像处理：图像分割等.
- 探索性数据分析（Exploratory Data Analysis）：发现数据中未知的结构和关系.

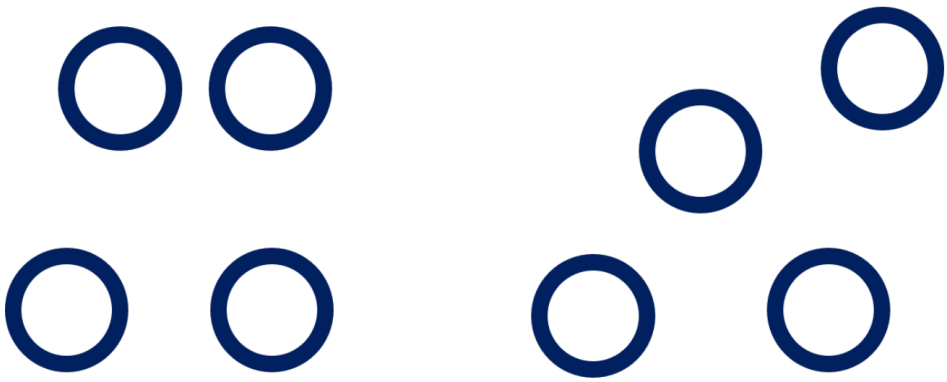
5.1 基本概念

> Clustering basics

5.1.1 样本 / 标签 / 聚类

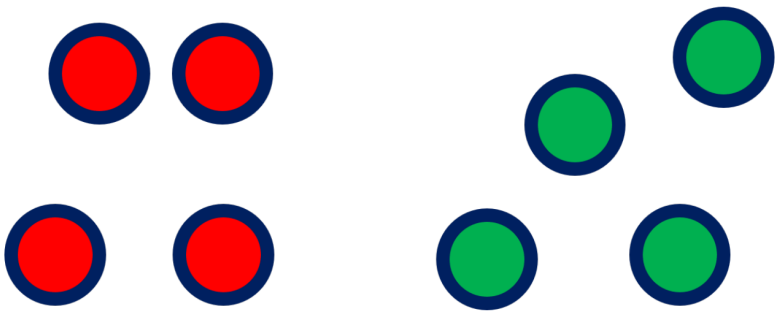
样本（Samples）

- Samples represent a set of individuals or objects collected or selected from a statistical population by a defined procedure.
- Each sample can be called a Data Point.



标签（Labels）

- A category or class applied to an object or a thing.
- For example, there are two classes (indicated by two colors, red for Class 0 and green for Class 1):

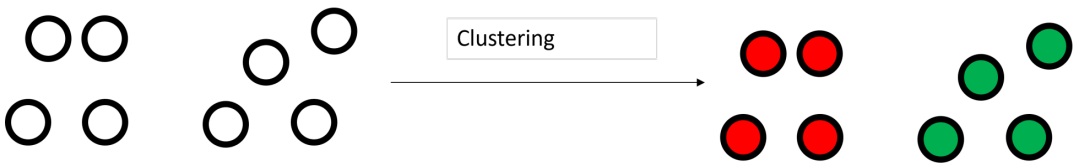


聚类（Clustering）

- Group a set of objects so that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).
- High intra-cluster similarity and low inter-cluster similarity.

  > 高簇内相似性，低簇间相似性.

- Clustering belongs to unsupervised learning (i.e., having samples but without using labels).



- In comparison, classfication belongs to supervised learning (i.e., need both samples and corresponding labels).

5.1.2 相似性度量

> Measure of similarity

衡量数据点相似性有多种方法.

Given two data points $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{R}^n$.

① 欧几里得距离

> Euclidean distance

$$\mathrm{dist}(\boldsymbol{a},\boldsymbol{b})=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}$$

② 曼哈顿距离

> Manhattan distance

$$\mathrm{dist}(\boldsymbol{a},\boldsymbol{b})=\sum_{i=1}^{n}|a_i-b_i|$$

③ 切比雪夫距离

> Chebyshev distance

$$\mathrm{dist}(\boldsymbol{a},\boldsymbol{b})=\max_i(|a_i-b_i|)$$

④ 闵可夫斯基距离

> Minkowski distance

$$\mathrm{dist}(\boldsymbol{a},\boldsymbol{b})=\left(\sum_{i=1}^{n}|a_i-b_i|^p\right)^{\frac{1}{p}}$$

⑤ 余弦相似度

> Cosine similarity

5.1.3 聚类算法

> Clustering algorithms

5.2 K-Means 聚类

一种**基于中心点**的划分聚类算法. 需要用户手动指定簇的数量 K.

5.2.1 迭代过程

① 初始化 - 初始化 K 个随机中心点（centroid）.

> 理想的初始化范围应该在数据集的分布范围内，否则可能导致收敛速度变慢、聚类结果不稳定（可能陷入局部最优解）.
>
> 例如，初始化 2 个中心点，第一个在数据集附近，第二个离所有数据点都很远. 在第一次分配时，所有数据点都会分配给第一个中心点， 导致第二个中心点在第一次更新后仍然处于空置状态. 这样可能陷入局部最优解，无法得到最优的聚类结果（无法最小化误差平方和 SSE）.

② 分配 - 将每个数据点分配给**最近**的中心点.

③ 更新 - 将每个簇的中心点更新为其所有成员数据点的平均值.

④ 重复 ②-③，直到满足收敛条件.

5.2.2 收敛条件

- No (or minimum) reassignment of data points to different clusters, or
- no (or minimum) change of centroids, or
- minimum decrease in the sum of squared error (SSE)

5.2.3 误差平方和

误差平方和（Sum of Squared Error, SSE）：

> 待完成.

5.2.4 复杂度分析

复杂度分析：

> 待完成.

5.2.5 优缺点

> advantages & disadvantages

优点：

- 易于理解和实现.
- 效率高，易拓展，保证收敛.

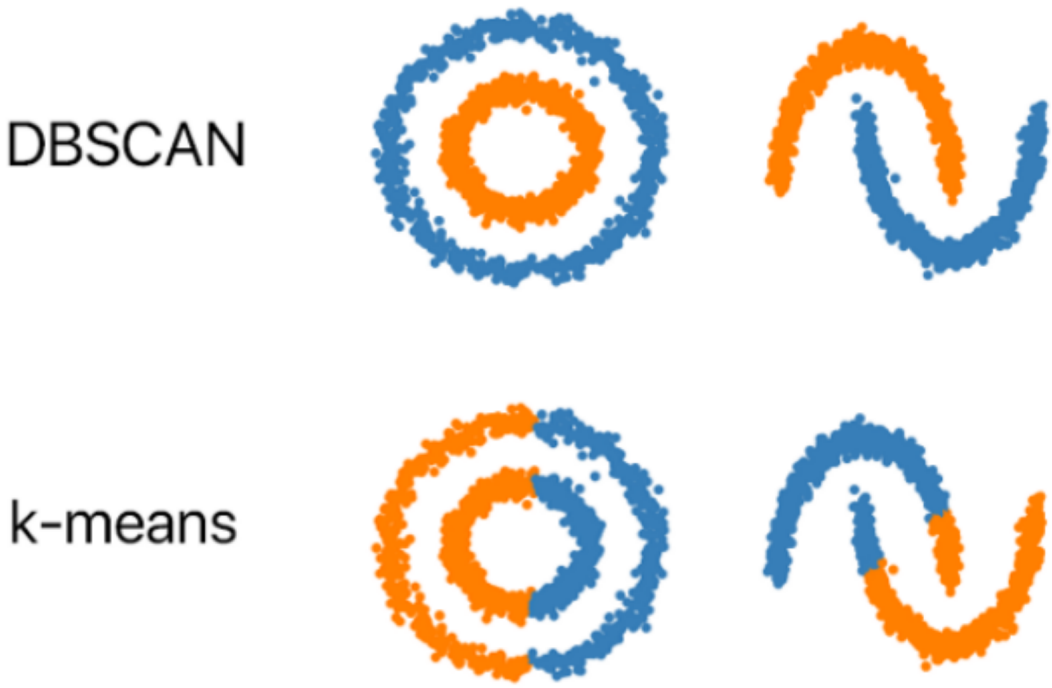
缺点

- 必须手动设置 K 值.
- 对初始中心点的选择敏感.
- 对异常值敏感.
- 对非球形数据的聚类效果不佳.

5.3 基于密度聚类

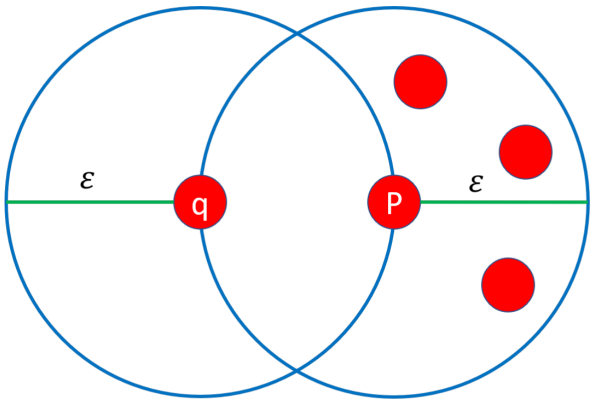
> DBSCAN

基本原理：将簇视为数据空间中的密集区域. 它能发现任意形状和大小的簇，并将低密度区域的点视为噪声.



5.3.1 DBSCAN

DBSCAN: Density-based spatial clustering of applications with noise



$MinPts = 4$

$$\epsilon - Neightborhood\ of\ p$$
$$\epsilon - Neightborhood\ of\ q$$
$$Density\ pf\ p\ is"\ \text{high}"\ (MinPts = 4)$$
$$Density\ pf\ q\ is\ "\text{low}"\ (MinPts = 4)$$

核心参数：

- ϵ：邻域的最大半径.

  ϵ-Neighbor: data points within a radius of ϵ from a data point (including the point itself)
- $MinPts$: minimum number of points required in an ϵ-Neighbor.

密度（Density）

- Density = number of points within a specified radius ϵ

- "high density": data point's ϵ-Neighbor contains at least $MinPts$ data

数据点分类：基于密度定义，点被分为核心点、边界点和噪声点

- Core point
- Border point
- Noise point

5.3.2 密度可达性

> Density-reachable

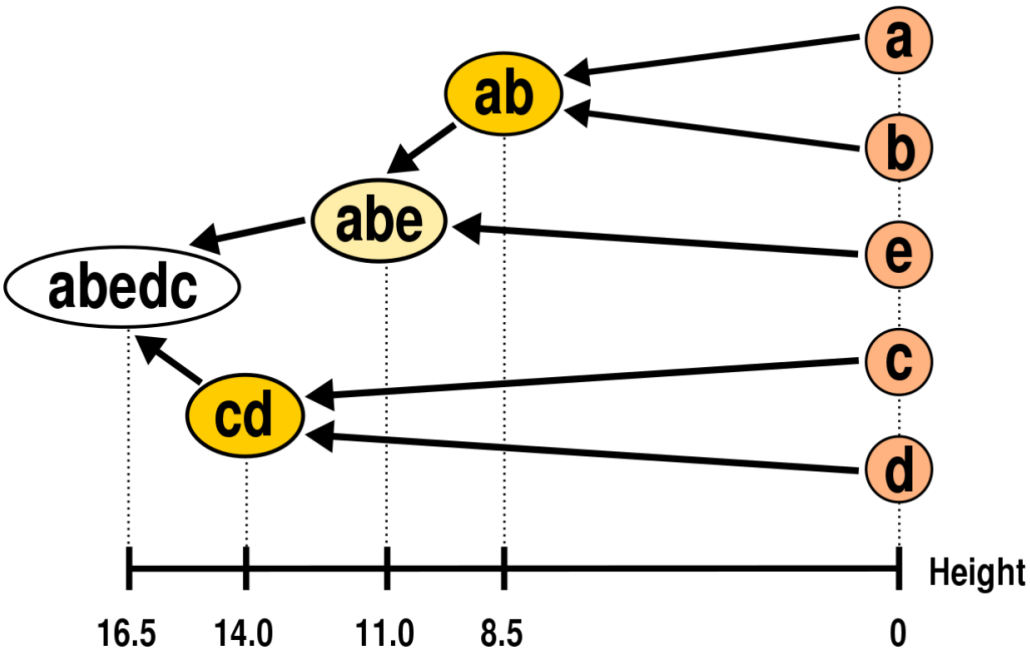
5.4 凝聚型层次聚类

> Agglomerative Hierarchical Clustering

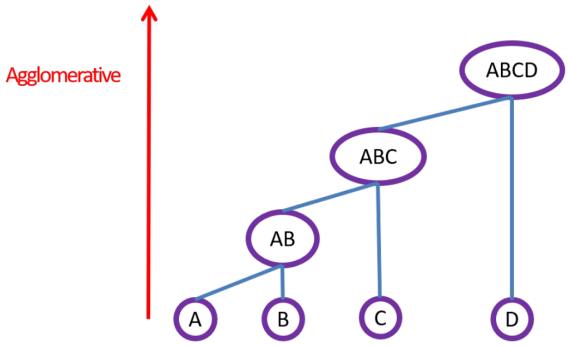
层次聚类（Hierarchical clustering）：旨在构建一个簇的层次结构.

- 聚合式（Agglomerative，自底向上）：每个观测值从自己的簇开始，然后将簇对合并，直到所有簇融合在一起.
- 分裂式（Divisive，自顶向下）：所有观测值从一个簇开始，然后递归地进行分裂.

The result of hierarchical clustering usually presented in dendrogram（树状图）.



这里主要介绍凝聚型层次聚类：



Basic algorithm is straightforward

1. Compute the distance matrix
2. Let each data point be a cluster
3. **Repeat**
4. **Merge the two closest clusters**
5. **Update the proximity matrix**
6. **Until** only a single cluster remains

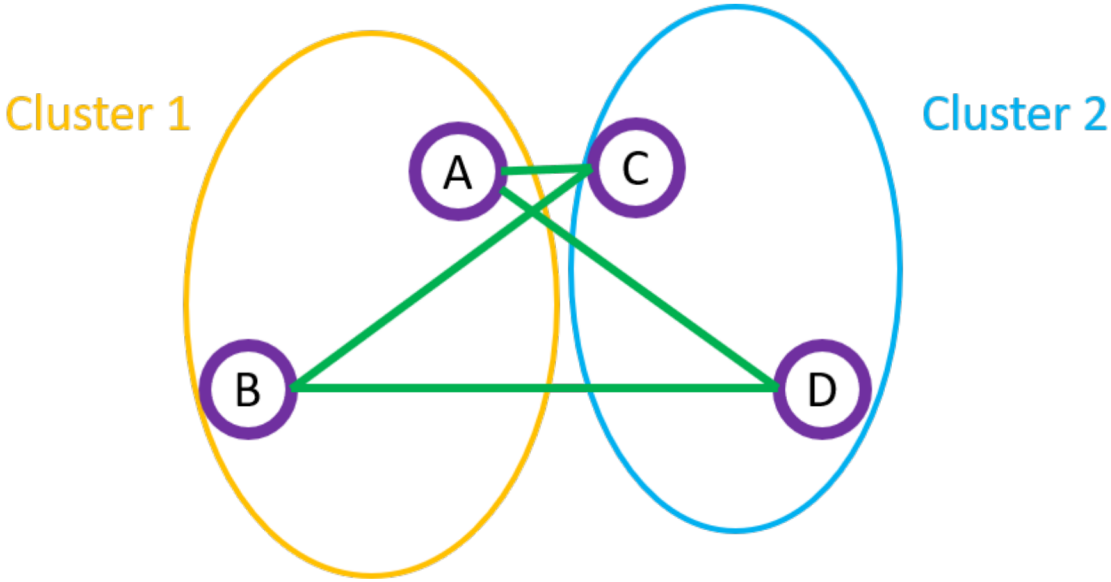
5.4.1 距离矩阵

> Distance matrix / proximity matrix

记录簇之间的距离.

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{pmatrix}
0.0 & & & & \\
2.0 & 0.0 & & & \\
6.0 & 5.0 & 0.0 & & \\
10.0 & 9.0 & 4.0 & 0.0 & \\
9.0 & 8.0 & 5.0 & 3.0 & 0.0
\end{pmatrix}
\end{array}
$$

平均距离（Group average distance / Average Linkage）：



Lecture 6 神经网络

> Neural Network Basics

前面讲的 `Lecture 2 线性回归` 解决连续值预测问题、`Lecture 3 逻辑回归` 解决二分类问题，实际上可以被视为最简单形式的神经网络——单个神经元（或称为感知机）.

- 线性回归：一层输入层（节点数等于输入特征数加一个偏置），一个神经元（激活函数 $g(x)=x$ 或视为没有激活函数，把输入直接输出）作为输出层，没有隐藏层. 输入层各节点的值经过加权平均作为神经元的输入，然后神经元不作处理直接输出. 这样只能捕捉特征之间的线性关系.
- 逻辑回归：一层输入层（节点数等于输入特征数加一个偏置），一个神经元（激活函数为 Sigmoid 函数 $g(x)=\frac{1}{1+e^{-x}}$）作为输出层，没有隐藏层. 输入层各节点的值经过加权平均作为神经元的输入，然后神经元把输入经过激活函数后得到输出. 这里虽然将输入特征的线性组合通过 Sigmoid 非线性地映射到了概率值 p，但二分类问题的决策边界仍是一个线性方程（超平面），即仍然无法捕捉特征之间的非线性关系用于分类. 逻辑回归通常被归类为线性分类器.

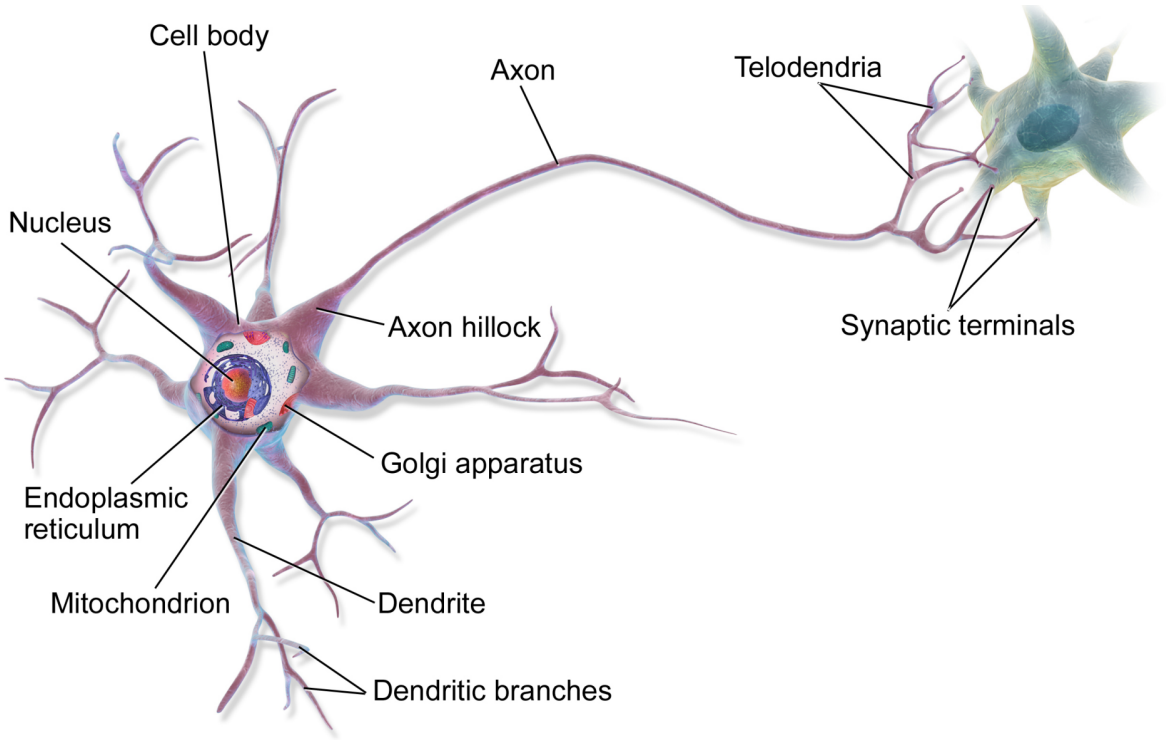
本章则把神经元拓展为网络，通过引入隐藏层和非线性激活函数，多层神经网络可以学习复杂的、非线性的决策边界.

如果一个模型只包含输入层和输出层（如逻辑回归），被称为浅层模型，虽然勉强也可视为神经网络（单个神经元），但不属于深度学习，只是普通的机器学习任务. 一个只包含输入层、少数隐藏层（例如仅一层）和输出层的多层感知机（MLP），通常也认为是浅层学习，而不是深度学习. 当网络足够深（通常指超过 2-3 层隐藏层）时，其学习任务才被称为深度学习.

6.1 神经网络架构

> Neural network architecture

启发：人类的大脑神经元.

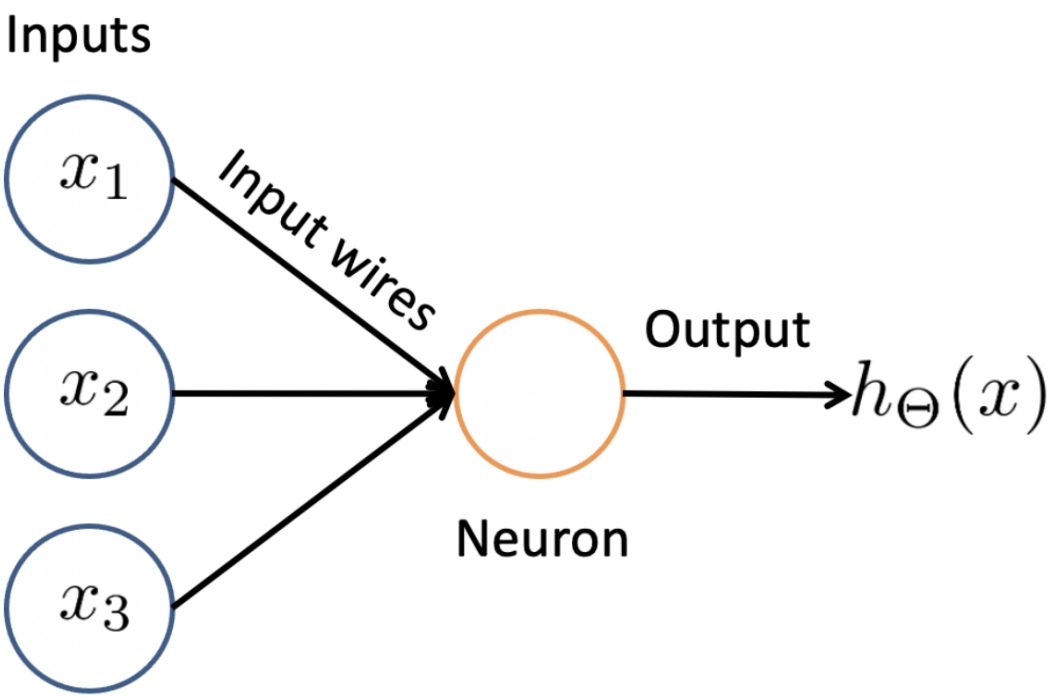


- The basic computational unit of the brain is a neuron.
- 86 billion neurons can be found in the human nervous system and they are connected with 10^{14}-10^{15} approximately synapses（突触）.
- Each neuron receives input signals from its dendrites（树突）and produces output signals along its (single) axon（轴突） connections.

6.1.1 神经元模型

> Neuron model: logistic unit

Here we model a neuron as a logistic unit.



- We have several inputs, e.g. x_1, x_2, x_3.
- We call the connection from the input to the neuron "input wires".
- We call the connection from the neuron to the output "output wire".

> 一个神经元有多条输入线和一条输出线.

- Here, $h_\Theta(x) = \frac{1}{1+e^{-x\Theta}}$ is an activation function, where $x = [x_1, x_2, x_3]$, $\Theta = [\theta_1, \theta_2, \theta_3]^T$.
- Θ is often called the "weights" of model or the "parameters" of model.

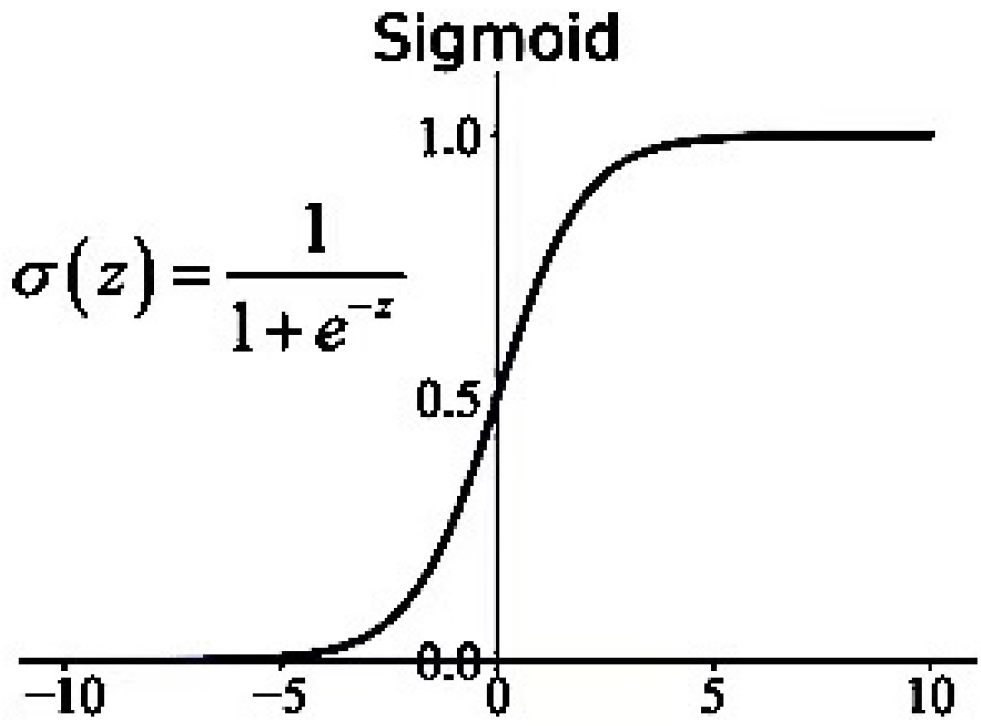
6.1.2 激活函数

> Activation functions (non-linear functions)

注意：

- 不同的隐藏层可以使用不同的激活函数，不过通常为了简化设计是使用同一种激活函数（对于某些复杂架构，使用不同激活函数也是合理的）.
- 同一层隐藏层的不同神经元理论上可以配置不同的激活函数，但是实际应用中罕见且不推荐.
- 输出层的激活函数可以和隐藏层不同，输出层使用的激活函数是由任务类型决定的.
- 常见的情况：使用 ReLU 或 Tanh 作为所有隐藏层的激活函数；在输出层，使用 Sigmoid（二分类任务）、Softmax（多分类任务）或不使用激活函数（连续值预测任务，也可以视为使用恒等函数 $g(x) = x$ 作为激活函数）.

① Sigmoid

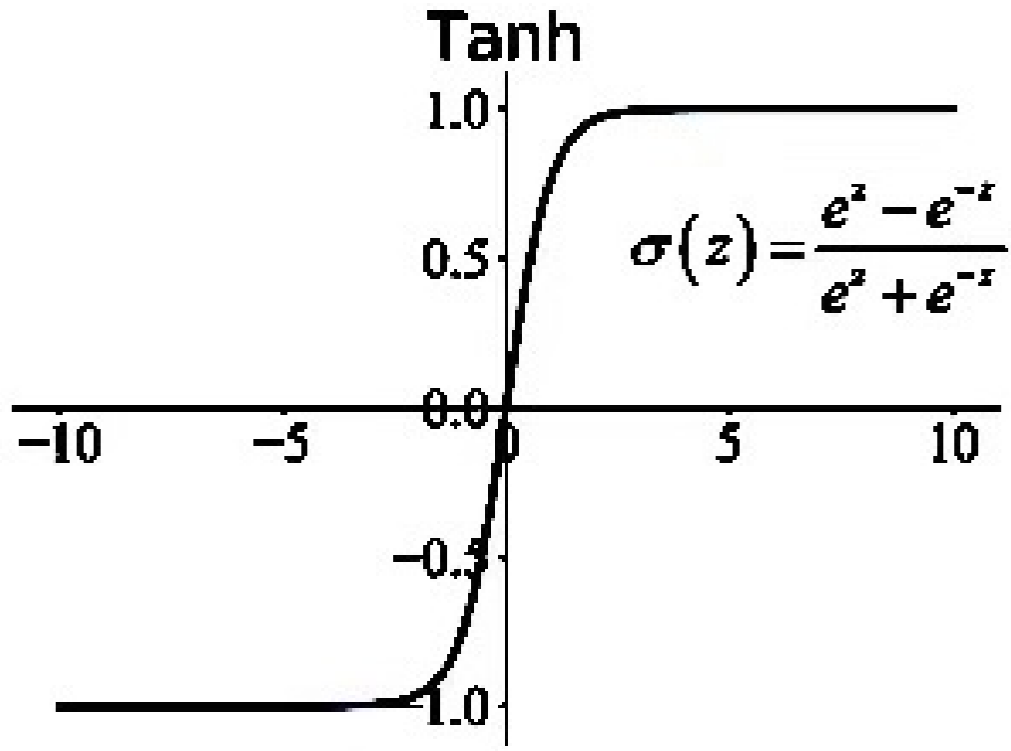


- Sigmoid with the mathematical from $\sigma(x) = \frac{1}{1+e^{-x}}$, which squashes real numbers to range between $[0, 1]$.
- Advantages
  - Continuous function for easy derivation
  - Limited output range, easy to optimize, can be used as output layer.
- Disadvantages
  - When $x \to \pm\infty$, the gradient will be 0（梯度消失，gradient vanish）
  - The computational complexity is high (exponential form)

- The output mean value is not centered at 0, resulting in a decrease in the convergence speed.

> 无论输入神经元的值是什么，经过 Sigmoid 激活后，输出值永远是正数. 当一个隐藏层中的所有神经元的输出都是正数时，我们说这一层的激活值没有以 0 为中心.
>
> 想象一个简单的场景：对于一层神经元来说，它的输入是上一层的输出. 如果上一层的输出全部是正数，那么在反向传播计算梯度时，根据链式法则，在每一次梯度更新中，所有权重要么全部向正方向更新，要么全部向负方向更新.
>
> 但最优解是有正有负的，这样梯度的更新方向就受到了限制，不能直接向最优解前进，而是以 Z 字形接近，收敛速度慢.

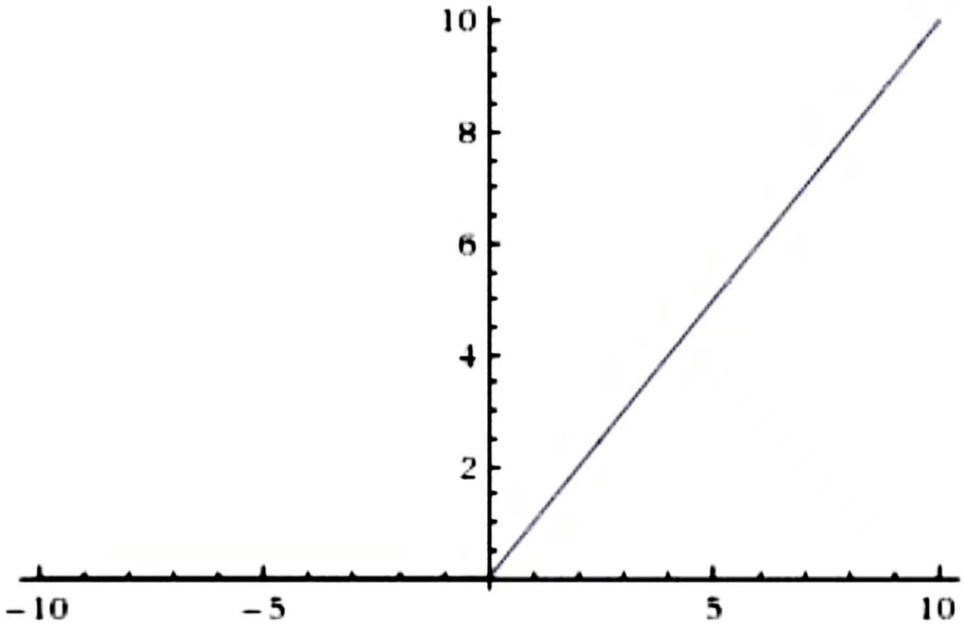
② Tanh



- Tanh with the mathematical form $\tanh(x) = 2\sigma(x) - 1$, which squashes real numbers to range between $[-1, 1]$.
- Advantages
  - The output mean value is centered at 0, resulting in converging faster than sigmoid.
- Disadvantages
  - When $x \to \pm\infty$, the gradient will be 0 (梯度消失，gradient disappearance)
  - The computational complexity is high (exponential form)

③ ReLU

> Rectified Linear Unit，整流线性单位函数 / 修正线性单元

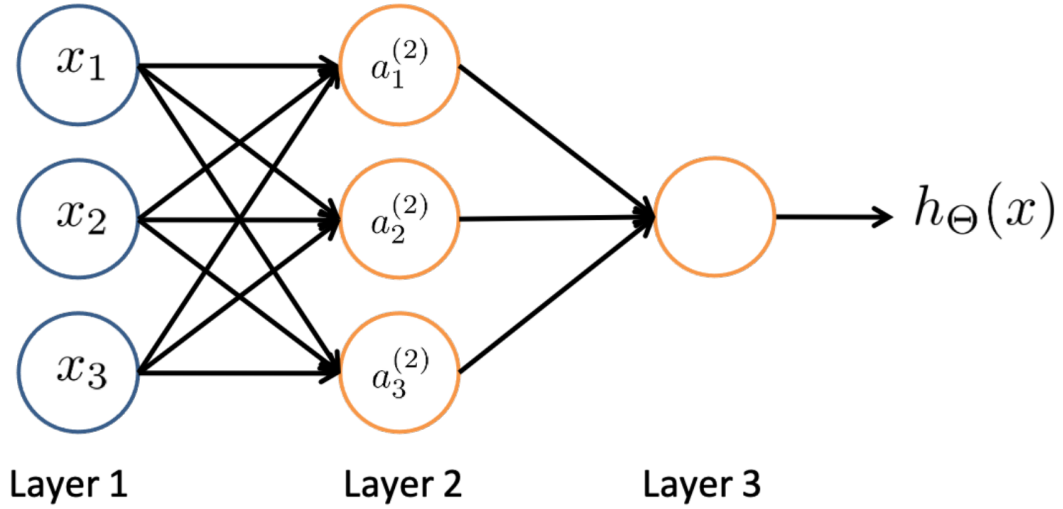


- ReLU with the mathmatical form $f(x) = \max(0, x)$, which is zero when $x < 0$ and then linear with slope 1 when $x \geq 0$.
- Advantages
  - The gradient is not saturated（梯度不饱和，意味着当神经元输入值非常大或非常小时，其梯度不会趋近于零） and the convergence speed is fast. Compared with sigmoid / tanh, it greatly improves the problem of gradient disappearance.
  - No exponential calculation is required, so the calculation speed is fast and the complexity is low.
- Disadvantages
  - When the input $x < 0$, the gradient is 0 and the parameters are not updated (a.k.a neuron death).
  - The output mean value is greater than 0, which affects the convergence of the network.

6.1.3 前馈神经网络

> Feed forward neural network

A neuron network is just groups of these neurons strung together.



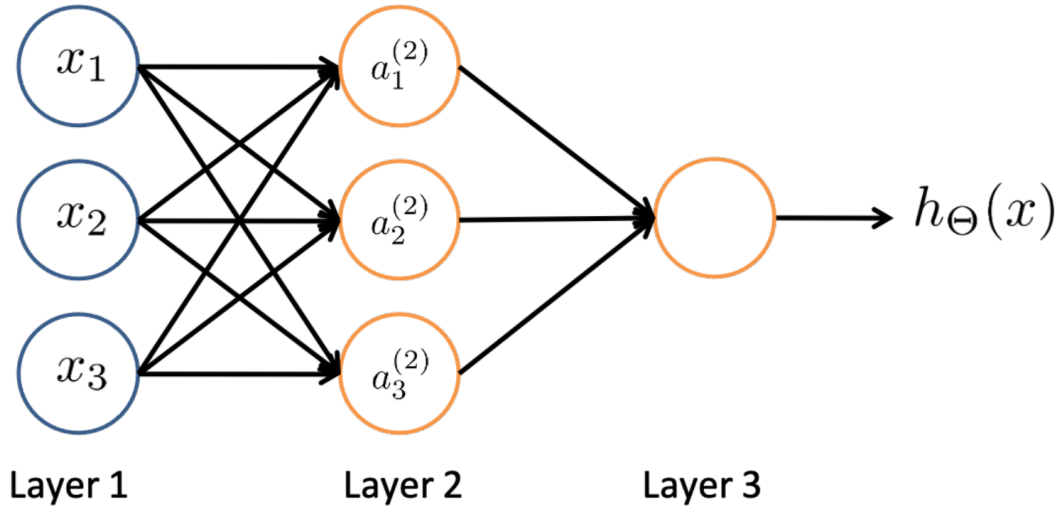
前馈神经网络由多组神经元连接而成. 通常包括输入层（Layer 1）、隐藏层（Layer 2）和输出层（Layer 3）.

> - The first layer is called the input layer because this is where the features x_1, x_2, x_3 are input.
> - The final layer is called the output layer because that layer has the neuron that outputs the final value computed.
> - The layer between the input and output layer is called the hidden layer as its value can not be seen.

注意，输入层通常认为只由一组节点组成，没有神经元，不执行任何计算，只是作为网络的起点. 隐藏层和输出层都由神经元组成.

Here we have input units x_1, x_2, x_3, three neurons, and the last neuron to compute $h_\Theta(x)$, where $a_i^{(j)}$ denotes the "activation" of unit i in layer j.

- $a_i^{(j)} =$ "activation" of unit i in layer j.
- $\Theta^{(j)} =$ matrix of weights controlling function mapping from layer j to layer $j+1$.
- $\Theta_{mn}^{(j)} =$ weight controlling function mapping from unit m in layer j to unit n in layer $j+1$.



- $a_1^{(2)} = g(\Theta_{11}^{(1)}x_1 + \Theta_{21}^{(1)}x_2 + \Theta_{31}^{(1)}x_3)$
- $a_2^{(2)} = g(\Theta_{12}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{32}^{(1)}x_3)$
- $a_3^{(2)} = g(\Theta_{13}^{(1)}x_1 + \Theta_{23}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$

where g is the sigmoid activation function.

- $h_\Theta(x) = a_1^{(3)} = g(\Theta_{11}^{(2)} a_1^{(2)} + \Theta_{21}^{(2)} a_2^{(2)} + \Theta_{31}^{(2)} a_3^{(2)})$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_j \times s_{j+1}$.

写成矩阵形式，即：

$$\begin{aligned}\mathbf{a}^{(j+1)} &= g(\mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}) \\ &= g\left(\begin{bmatrix}\Theta_{11}^{(j)} & \Theta_{12}^{(j)} & \cdots & \Theta_{1s_{j+1}}^{(j)} \\ \Theta_{21}^{(j)} & \Theta_{22}^{(j)} & \cdots & \Theta_{2s_{j+1}}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{s_j1}^{(j)} & \Theta_{s_j2}^{(j)} & \cdots & \Theta_{s_js_{j+1}}^{(j)}\end{bmatrix}^T \begin{bmatrix}a_1^{(j)} \\ a_2^{(j)} \\ \vdots \\ a_{s_j}^{(j)}\end{bmatrix}\right)\end{aligned}$$

这里忽略了偏置项. 注意，这里有两步操作，第 j 层的输入 $\mathbf{\Theta}^{(j-1)^T}\mathbf{a}^{(j-1)}$ 经过激活函数 g 后得到第 j 层的输出 $\mathbf{a}^{(j)} = g\left(\mathbf{\Theta}^{(j-1)^T}\mathbf{a}^{(j-1)}\right)$，第 j 层的输出（即激活函数的值）要经过加权平均才是第 $j+1$ 层的输入 $\mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}$，然后再经过激活函数得到第 $j+1$ 层的输出 $\mathbf{a}^{(j+1)} = g\left(\mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}\right)$. 即每层的输入是上一层输出的加权平均，每层的输出是该层输入经过激活函数的结果.

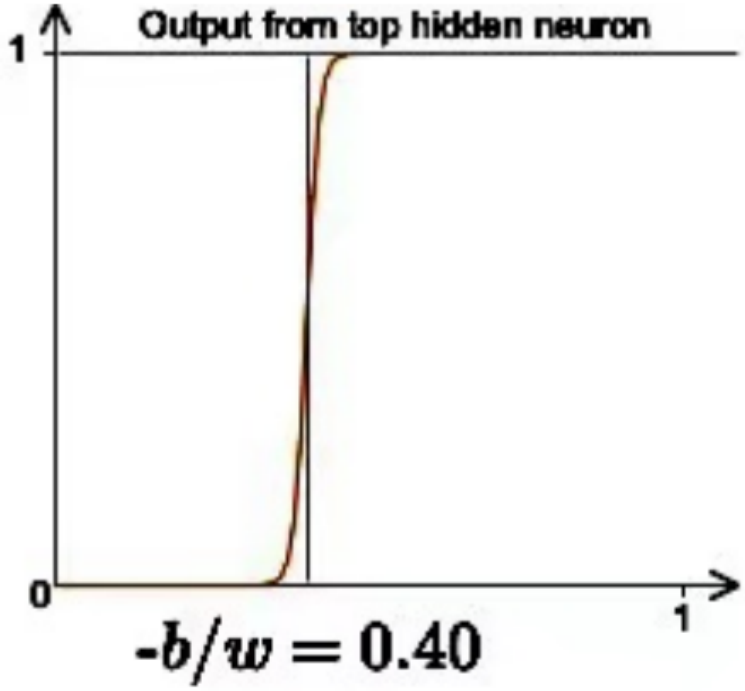
在神经网络的计算中，通常会引入一个中间变量 $\mathbf{z}^{(j)}$ 表示第 j 层激活函数的加权输入，即 $\mathbf{z}^{(j+1)} = \mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}$. 这样会使反向传播的梯度计算更简洁.

其中，输入层（第一层）和输出层（最后一层）比较特殊：输入层可以看作没有激活函数，或者激活函数 $g(x) = x$，但是输入层到第一个隐藏层也要经过加权平均；输出层如果是二分类或者回归问题，通常仅包含一个神经元，如果是多分类则需要 K 个神经元（K 是类别数）. 其输出为隐藏层最后一层的输出加权平均后作为输入， 并经过激活函数的结果. 但这个结果不需要再乘以权重了.

考虑单输入 x. 当 hidden neuron 使用 Sigmoid 作为输出，即

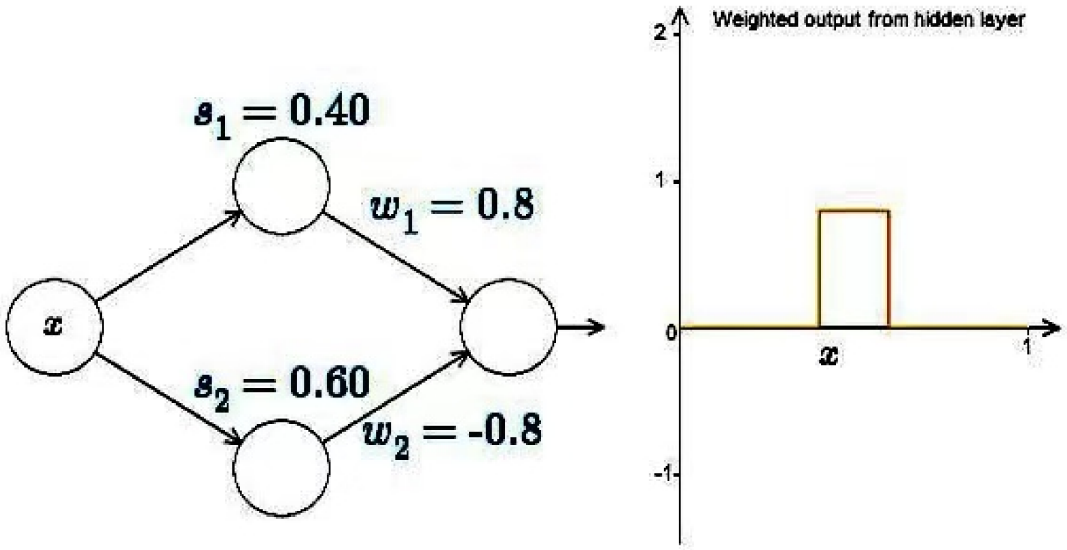
$$y = f(x) = \frac{1}{1+e^{-(wx+b)}}$$

为讨论 “激活” 的含义，我们简化模型，考虑比值 $-\frac{b}{w}$. 函数的整体形状类似在 $x = -\frac{b}{w}$ 处的阶跃函数.



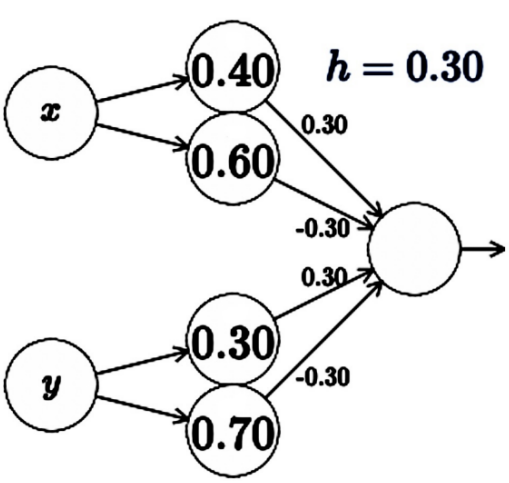
> 就像 x 到 $-\frac{b}{w}$ 时该神经元突然被激活了.

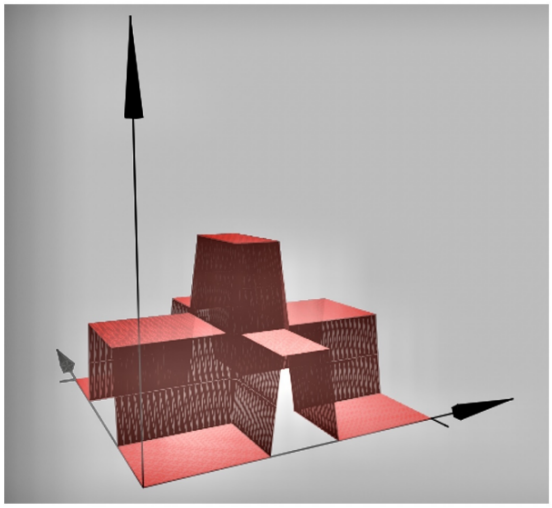
假设 hidden layer 有一层、两个神经元. top neuron 的阶跃点 $s_1 = 0.40$，bottom neuron 的阶跃点 $s_2 = 0.60$. 记这两个神经元的输出分别为 a_1，a_2，则通过对 hidden layer 到最终输出 neuron 之间的权重进行设计，可以实现如下的 $w_1 a_1 + w_2 a_2$ 的输出图像：



> 类似于 top neuron 负责激活，bottom neuron 负责抑制.
>
> 当 $x \in (0.4, 0.6)$ 时，top neuron 激活，bottom neuron 未激活（抑制未生效），整体输出为 0.8.
>
> 当 $x < 0.4$，两个 neurons 都未激活；当 $x > 0.6$，top neuron 激活，但被 bottom neuron 抑制，整个层等同于未激活.

增加 input 数量，图像的维度增加：





> 实际上 inputs 一般写作 x_1 和 x_2.

6.2 反向传播

> Backpropagation

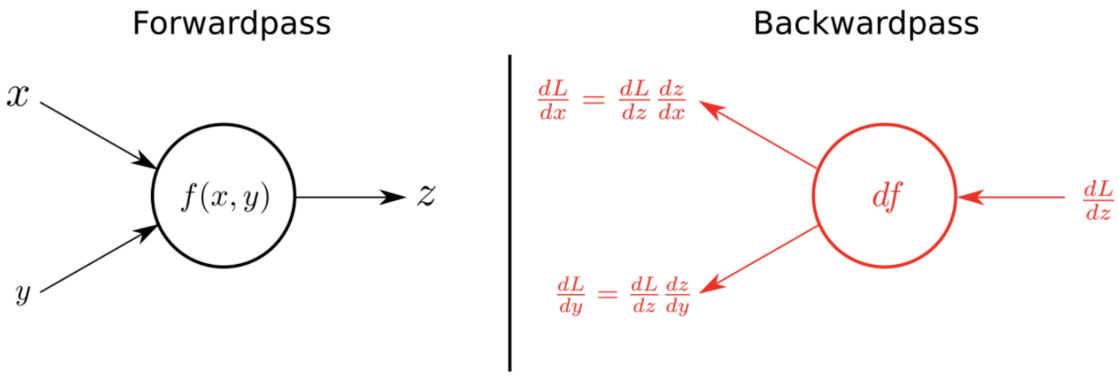
除了架构之外，权重本身的数值也是神经网络的重要部分. 如果架构设计很好，但是权重是乱写的，效果也不好.

In practice, randomly initialize the parameters to small values

> e.g. normally distributed around zero: $N(0, 0.1)$

然后我们开始训练（train）神经网络，来优化（optimize）参数.

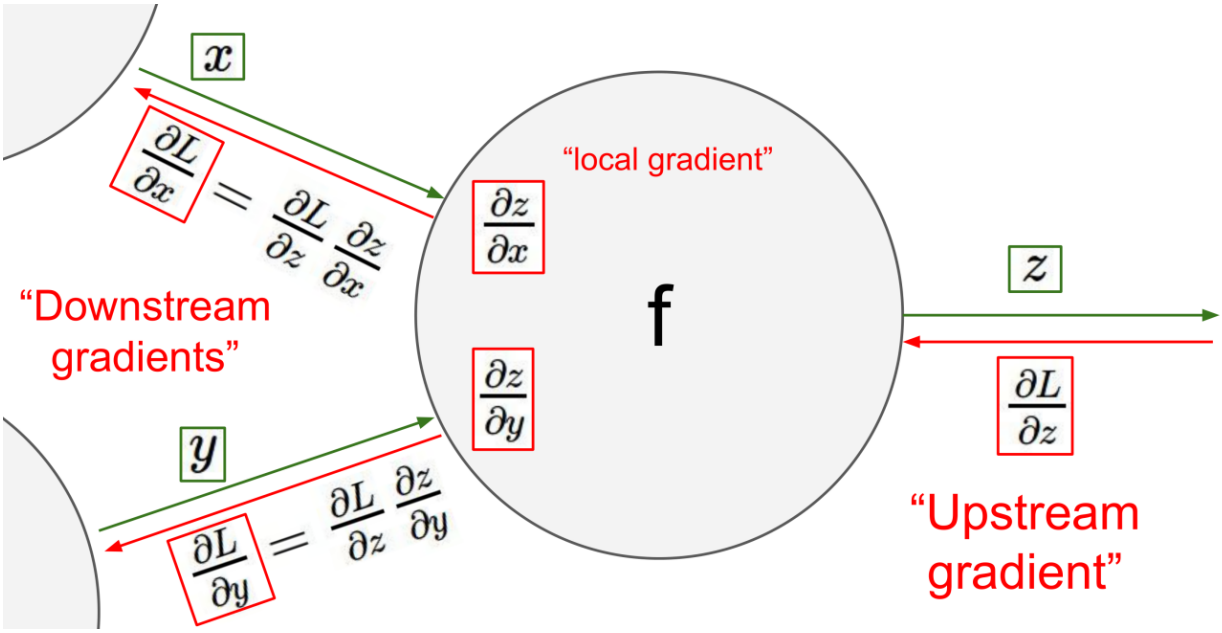
The **backpropagation** algorithm 通过梯度下降法，在权重空间中微调权重来最小化代价函数.



每层的局部梯度通常需要代入局部的输入输出值（中间变量值）进行计算. 因此，在前向传播时，除了计算当前层输出 z 并传给下一层，还会把后续反向传播时计算局部梯度要用到的中间变量值（例如输入 x, y，激活值 z，以及相关权重值）都缓存起来，使得反向传播的运行非常高效.

至于反向传播时的每个局部梯度的解析公式，都是程序员在编写深度学习库时，基于微积分预先写死在代码里的. 实际的调包训练过程只是纯粹的数值流动.

上述图示仅体现了梯度从上游到下游 “传播” 的过程，实际的权重更新是另外的计算. 例如，假设一个简单的线性神经元 $z = wx + b$，实际的参数更新公式是 $w \leftarrow w - \alpha \frac{\partial L}{\partial w}$. 我们保存了所有的当前权重和中间变量值，实际只需要计算 $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w} = \frac{\partial L}{\partial z} \cdot x$. 其中 x 是前向传播时保存的中间变量值，$\frac{\partial L}{\partial z}$ 即上游传下来的梯度. 所以真正重要的就是传播的这些梯度值，$\frac{\partial L}{\partial w}$ 总能由这些传播的梯度和中间变量值组合计算出来.



6.2.1* 解析推导

以单个样本 $(\mathbf{x}, \mathbf{y})$ 为例，推导反向传播时局部梯度的解析公式.

> 若多样本，则使用批量梯度下降（对所有样本求平均）. 每个样本的推导逻辑一致.

注意，输出层（最后一层）的局部梯度的解析形式和输出层所选的激活函数以及损失函数有关. 隐藏层（中间层）的局部梯度解析形式只和那一层所选的激活函数有关. 而输出层的激活函数和损失函数通常和学习任务有关. 例如，连续值预测 - 线性（无激活）- 均方误差；二分类 - Sigmoid - 交叉熵损失；多分类 - Softmax - 交叉熵损失.

通用符号：

- $\mathbf{z}^{(j)}$ 是第 j 层隐藏层的加权输入，$\mathbf{a}^{(j)}$ 是第 j 层隐藏层的激活输出，维度为 $s_j \times s_{j+1}$ 的矩阵 $\mathbf{\Theta}^{(j)}$ 是第 j 层隐藏层的权重矩阵. $j = L$ 时表示输出层，$s_L = 1$，$\mathbf{a}^{(L)}$ 是输出层的激活输出.
- $J(\mathbf{\Theta})$ 是损失函数.
- 定义误差项 $\delta^{(j)} = \dfrac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}} = \begin{bmatrix} \frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}} \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_2^{(j)}} \\ \vdots \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_{s_j}^{(j)}} \end{bmatrix}$，$\delta^{(L)} = \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(L)}}$

> 注意这里 $\delta^{(j)}$ 是向量，但是 Markdown 无法渲染 δ 的粗体.

Recall: `6.1.3 前馈神经网络`

$$
\begin{aligned}
\mathbf{a}^{(j+1)} &= g(\mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}) \\
&= g\left(\begin{bmatrix} \Theta_{11}^{(j)} & \Theta_{12}^{(j)} & \cdots & \Theta_{1s_{j+1}}^{(j)} \\ \Theta_{21}^{(j)} & \Theta_{22}^{(j)} & \cdots & \Theta_{2s_{j+1}}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{s_j1}^{(j)} & \Theta_{s_j2}^{(j)} & \cdots & \Theta_{s_js_{j+1}}^{(j)} \end{bmatrix}^T \begin{bmatrix} a_1^{(j)} \\ a_2^{(j)} \\ \vdots \\ a_{s_j}^{(j)} \end{bmatrix}\right)
\end{aligned}
$$

反向传播的核心是计算 $J(\mathbf{\Theta})$ 对每个权重参数 $\Theta_{ik}^{(j)}$ 的梯度 $\frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{ik}^{(j)}}$. 这里 $\Theta_{ik}^{(j)}$ 表示第 j 层隐藏层第 i 个神经元的激活输出 $a_i^{(j)}$ 到第 $j+1$ 层隐藏层第 k 个神经元的权重.

已知 $\mathbf{z}^{(j+1)} = \mathbf{\Theta}^{(j)^T}\mathbf{a}^{(j)}$，即 $z_k^{(j+1)} = \begin{bmatrix} \Theta_{1k}^{(j)} \\ \Theta_{2k}^{(j)} \\ \vdots \\ \Theta_{s_jk}^{(j)} \end{bmatrix}^T \begin{bmatrix} a_1^{(j)} \\ a_2^{(j)} \\ \vdots \\ a_{s_j}^{(j)} \end{bmatrix}$ 是第 $j+1$ 隐藏层第 k 个神经元的加权输入.

由链式法则，即计算 $\frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{ik}^{(j)}} = \frac{\partial J(\mathbf{\Theta})}{\partial z_k^{(j+1)}} \cdot \frac{\partial z_k^{(j+1)}}{\partial \Theta_{ik}^{(j)}} = \delta_k^{(j+1)} \cdot a_i^{(j)}$

定义标量对矩阵求导:

$$
\begin{aligned}
\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{\Theta}^{(j)}} &= \begin{bmatrix} \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{11}^{(j)}} & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{12}^{(j)}} & \cdots & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{1s_{j+1}}^{(j)}} \\ \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{21}^{(j)}} & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{22}^{(j)}} & \cdots & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{2s_{j+1}}^{(j)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{s_j1}^{(j)}} & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{s_j2}^{(j)}} & \cdots & \frac{\partial J(\mathbf{\Theta})}{\partial \Theta_{s_js_{j+1}}^{(j)}} \end{bmatrix} \\
&= \begin{bmatrix} \delta_1^{(j+1)} \cdot a_1^{(j)} & \delta_2^{(j+1)} \cdot a_1^{(j)} & \cdots & \delta_{s_{j+1}}^{(j+1)} \cdot a_1^{(j)} \\ \delta_1^{(j+1)} \cdot a_2^{(j)} & \delta_2^{(j+1)} \cdot a_2^{(j)} & \cdots & \delta_{s_{j+1}}^{(j+1)} \cdot a_2^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1^{(j+1)} \cdot a_{s_j}^{(j)} & \delta_2^{(j+1)} \cdot a_{s_j}^{(j)} & \cdots & \delta_{s_{j+1}}^{(j+1)} \cdot a_{s_j}^{(j)} \end{bmatrix} \\
&= \mathbf{a}^{(j)}\left(\delta^{(j+1)}\right)^T
\end{aligned}
$$

因为 $\mathbf{a}^{(j)}$ 是第 j 层隐藏层的激活输出，在前向传播时保存了，我们只需要计算误差项 $\delta^{(j+1)}$.

对于隐藏层，如果我们知道 $\delta^{(j+1)}$，可计算出 $\delta^{(j)}$，即反向传播:

$$
\delta^{(j)} = \left(\mathbf{\Theta}^{(j)}\delta^{(j+1)}\right) \odot \left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)
$$

> 见 `Appendix 6. 反向传播`
>
> 这里 $\mathbf{z}^{(j)}$ 也在前向传播时保存了，激活函数求导的解析形式根据具体的激活函数求导得到.

因此唯一需要计算的是反向传播的起点，即输出层的 $\delta^{(L)}$

对于输出层 $\delta^{(L)} = \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(L)}}$，分类讨论:

① 回归

对于回归问题（连续值预测）：

- $\mathbf{y}$ 通常是一个标量，因此后续用细体 y 表示；
- 输出层只有 1 个神经元，通常没有激活函数（把加权输入直接输出），或视激活函数为 $g(x)=x$. 因此 $\mathbf{a}^{(j)}=\mathbf{z}^{(j)}$
- 使用 Squared Error $J(\mathbf{\Theta})=\frac{1}{2}\|h_{\mathbf{\Theta}}(\mathbf{x})-y\|^2$，其中 $h_{\mathbf{\Theta}}(\mathbf{x})=\mathbf{a}^{(L)}=a^{(L)}$. 也是标量，用细体表示.
- 输出层的加权输入也是标量，用 $z^{(L)}$ 表示.
- 误差项 $\delta^{(L)}=\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(L)}}=\frac{\partial J(\mathbf{\Theta})}{\partial z^{(L)}}$ 也是标量.

> 不过 δ 粗体显示不出，因此全部显示为细体.

此时，

$$\begin{aligned}\delta^{(L)}&=\frac{\partial J(\mathbf{\Theta})}{\partial z^{(L)}}\\&=\frac{\partial J(\mathbf{\Theta})}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\\&=a^{(L)}-y\end{aligned}$$

> 因为 $a^{(L)}=z^{(L)}$，所以 $\frac{\partial a^{(L)}}{\partial z^{(L)}}=1$.
>
> 或者写作 $\hat{y}-y$，实际上就是预测值减去真实值.

② 二分类

对于二分类问题：

- $\mathbf{y}$ 通常是一个标量（0 或 1），因此后续用细体 y 表示；
- 输出层只有 1 个神经元，使用 Sigmoid 激活函数输出概率，即 $g(x)=\frac{1}{1+e^{-x}}$. 因此 $\mathbf{a}^{(j)}=g\left(\mathbf{z}^{(j)}\right)$
- 使用二元交叉熵 Cross-entropy error $J(\mathbf{\Theta})=-\left[y\ln P(y)+(1-y)\ln\left(1-P(y)\right)\right]$，其中 $P(y)=P(\hat{y}=1|\mathbf{x})=a^{(L)}$ 是标量，表示模型预测该样本标签为 1 的概率.
- 输出层的加权输入也是标量，用 $z^{(L)}$ 表示.
- 误差项 $\delta^{(L)}=\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(L)}}=\frac{\partial J(\mathbf{\Theta})}{\partial z^{(L)}}$ 也是标量.

此时，

$$\begin{aligned}\delta^{(L)}&=\frac{\partial J(\mathbf{\Theta})}{\partial z^{(L)}}\\&=\frac{\partial J(\mathbf{\Theta})}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\\&=a^{(L)}-y\end{aligned}$$

证明：

注意到 Sigmoid 函数有一个性质，即

$$g'(x)=g(x)\left(1-g(x)\right)$$

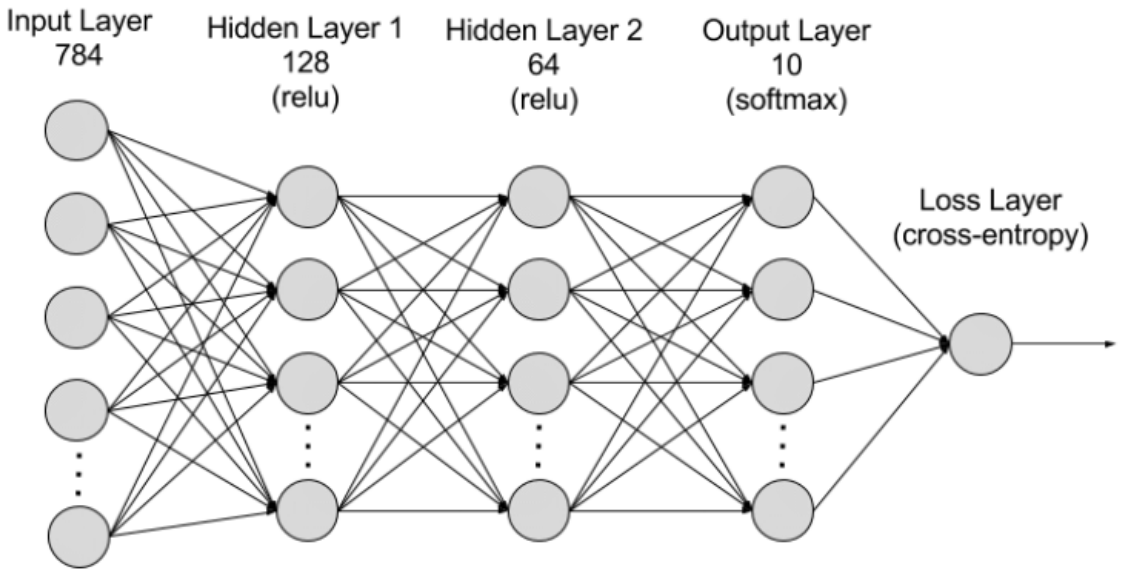
因此，

$$\begin{aligned}J(\mathbf{\Theta})&=-\left[y\ln P(y)+(1-y)\ln\left(1-P(y)\right)\right]\\&=-\left[y\ln a^{(L)}+(1-y)\ln\left(1-a^{(L)}\right)\right]\\\frac{\partial J(\mathbf{\Theta})}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}&=-\left[\frac{y}{a^{(L)}}-\frac{1-y}{1-a^{(L)}}\right]a^{(L)}\left(1-a^{(L)}\right)\\&=-y\left(1-a^{(L)}\right)+(1-y)a^{(L)}\\&=a^{(L)}-y\end{aligned}$$

这里虽然激活函数比较复杂，但是通过它的求导性质巧妙地化简为预测值（0 到 1 之间，预测标签为 1 的概率）减去真实值（样本的实际标签，0 或 1）

③ 多分类

以 MNIST 手写数字识别为例：



对于多分类问题：

- $\mathbf{y}$ 是一个 $s_L \times 1$ 的 one-hot 向量，其中 s_L 是类别数；
- 输出层有 s_L 个神经元，s_L 等于类别数量.
- 使用 Softmax 激活函数输出概率，即 $a_i^{(L)} = g\left(z_i^{(L)}\right) = \frac{e^{z_i^{(L)}}}{\sum_{j=1}^{s_L} e^{z_j^{(L)}}}$. 简写为 $\mathbf{a}^{(L)} = g\left(\mathbf{z}^{(L)}\right)$. 注意 Softmax 函数比较特殊，它有跨神经元的特性，因此这里不能简单看作把 $\mathbf{z}^{(L)}$ 的每个分量分别输入进去得到结果，再排成一列，而是把整个向量都输入进去，计算得到 s_L 个结果，排成一列. 每个神经元的计算都依赖所有神经元的输入.
- 使用交叉熵损失 Cross-entropy error $J(\mathbf{\Theta}) = -\sum_{i=1}^{s_L} y_i \ln P(y_i)$，其中 $P(y_i) = P(\hat{y}_i = 1|\mathbf{x}) = a_i^{(L)}$ 表示模型预测该样本为第 i 类的概率.
- 输出层的加权输入是向量，用 $\mathbf{z}^{(L)}$ 表示.
- 误差项 $\delta^{(L)} = \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(L)}}$ 也是向量.

可得 $\delta^{(L)} = \mathbf{a}^{(L)} - \mathbf{y}$

> 待补充.

多分类可以维护一个 Loss Layer.

> 待补充.

6.3 网络训练与验证

> Network training and validation

6.3.1 Softmax

> Softmax 是 Sigmoid 在多类别上的推广.

多分类问题中，The softmax function is often used as the last activation function of a neural netwalk to normalize the output of a netwalk η to a pobability distribution ϕ over predicted output classes.

数学形式：

$$\phi_i = \frac{e^{\eta_i}}{\sum\limits_{j=1}^{k} e^{\eta_j}}$$

它将一个 k 维的向量 $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$ 转换为一个 k 维的概率分布 $\phi = (\phi_1, \phi_2, \ldots, \phi_k)$，其中 $\sum\limits_{j=1}^{k} \phi_j = 1$. Softmax 通常用于多分类问题（$k > 2$）.

对比：Softmax vs Sigmoid

对比	**Softmax** $\phi_i = \frac{e^{\eta_i}}{\sum\limits_{j=1}^{k} e^{\eta_j}}$	**Sigmoid** $\sigma(x) = \frac{1}{1+e^{-x}}$
输入	一个向量 η，有 k 个元素 η_i	一个标量 x
输出	一个向量 ϕ，有 k 个元素 ϕ_i	一个标量 $\sigma(x)$
输出范围	每个 $\phi_i \in (0,1)$，且 $\sum\limits_{i=1}^{k} \phi_i = 1$	$\sigma(x) \in (0,1)$
用途	主要用于多分类的输出层	主要用于二分类的输出层或作为隐藏层的激活函数

Softmax 在 $k = 2$ 时退化为 Sigmoid：

当 $k = 2$，输入为 (η_1, η_2). Softmax for ϕ_1：

$$\phi_1 = \frac{e^{\eta_1}}{e^{\eta_1} + e^{\eta_2}} = \frac{1}{1 + e^{-(\eta_1 - \eta_2)}}$$

因此，对于二分类，Softmax 函数输出的第一个概率 ϕ_1 就等于 Sigmoid 函数作用于两个原始输入差值 $\eta_1 - \eta_2$ 的结果. 在二分类中，通常只需要计算第一个概率 ϕ_1，因为 $\phi_2 = 1 - \phi_1$.

Sigmoid 在多分类时推广为 Softmax 的推广思路：

Let's consider a multi-class classification problem in which the response variable y can take on any one of k values, so $y \in \{1, 2, \ldots, k\}$.

To parameterize a multinomial over k possible outcomes, we could use k parameters $\phi_1, \phi_2, \ldots, \phi_k$ specifying the probability of each of the outcomes.

However, they would bot be independent since knowing any $k-1$ of the ϕ_i's uniquely determines the last one, satisfying $\sum\limits_{i=i}^{k} \phi_i = 1$.

So we will instead parameterize the multinomial with only $k-1$ parameters, $\phi_1, \phi_2, \ldots, \phi_{k-1}$, where $\phi_i = p(y = i; \phi)$, and $p(y = k; \phi) = 1 - \sum\limits_{i=1}^{k-1} \phi_i$. For notational convenience, let $\phi_k = 1 - \sum\limits_{i=1}^{k-1} \phi_i$.

Here, we introduce an indicator function $1\{\cdot\}$ takes on a value of 1 if its argument is true, and 0 otherwise. For example, $1\{2 = 3\} = 0$, and $1\{3 = (5-2)\} = 1$.

We would like to define $T(y) \in \mathbb{R}^{k-1}$ as follows:

$$T(1)=\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, T(2)=\begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix}, \cdots, T(k-1)=\begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}, T(k)=\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$$

We will write

6.4* 迁移学习

> Transfer learning
>
> ESTR 内容.

- Training every model from scratch is time-consuming and expensive.
- But there are extensive existing knowledge. Can we reuse them?

Definition: research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

6.4.1 CNN 中的迁移学习

> Transfer learning in CNNs

CNN 的神经网络越靠前的层学到的特征越通用，越靠后的层学到的特征越针对特定的任务或数据集.

- Bottom / earlier layers: general learners ("low-level" concepts such as edges, colors, shapes)
  > 这些特征对于任何图像识别任务都通用，因此在迁移学习中通常会保留这些层的权重不变（或只进行微调）.
- Top / later layers: specific learners ("high-level" features such as object parts, semantics, object categories)
  > 这些特征是针对原始训练任务的，因此在迁移学习中，通常会替换或重新训练这些层的权重，以适应新的目标任务.

6.4.2 微调

> fine-tune

Lecture 7 卷积神经网络

> Convolutional Neural Networks (CNN)

7.1 基本概念

> Basic concepts

发展历史：LeNet (1998) and AlexNet (2012)

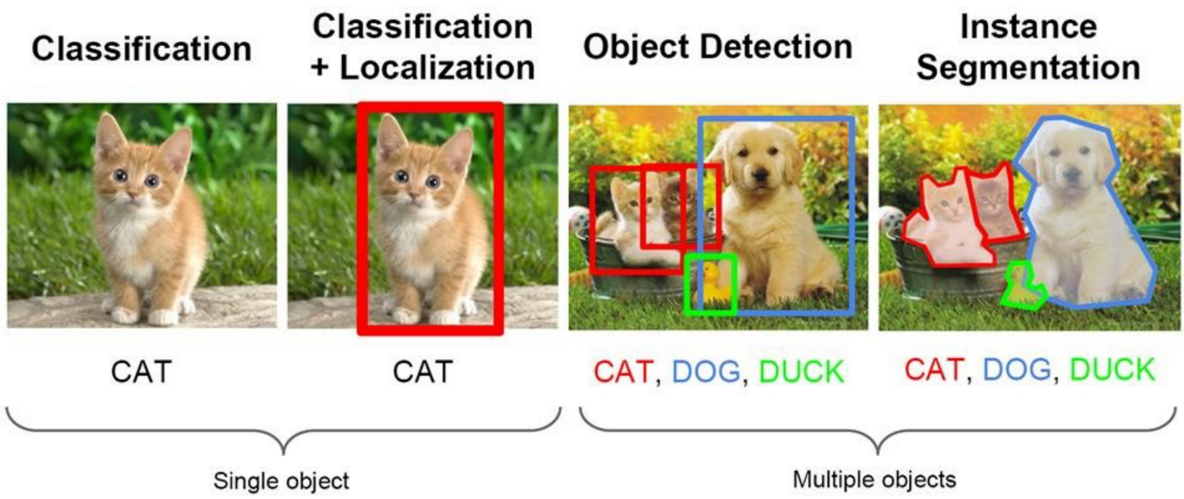
四种计算机视觉任务：分类（Classfication）、分类加定位（Classification + Localization）、目标检测（Object Detection）、实例分割（Instance Segmentation）.

分类（Classification）

- 目标：识别图像中的主要物体是什么，并将其归类到一个预定义的类别中.
- 输出：图像的单一类别标签（如：“猫”、"狗"、"卡车"）

分类加定位（Classification + Localization）

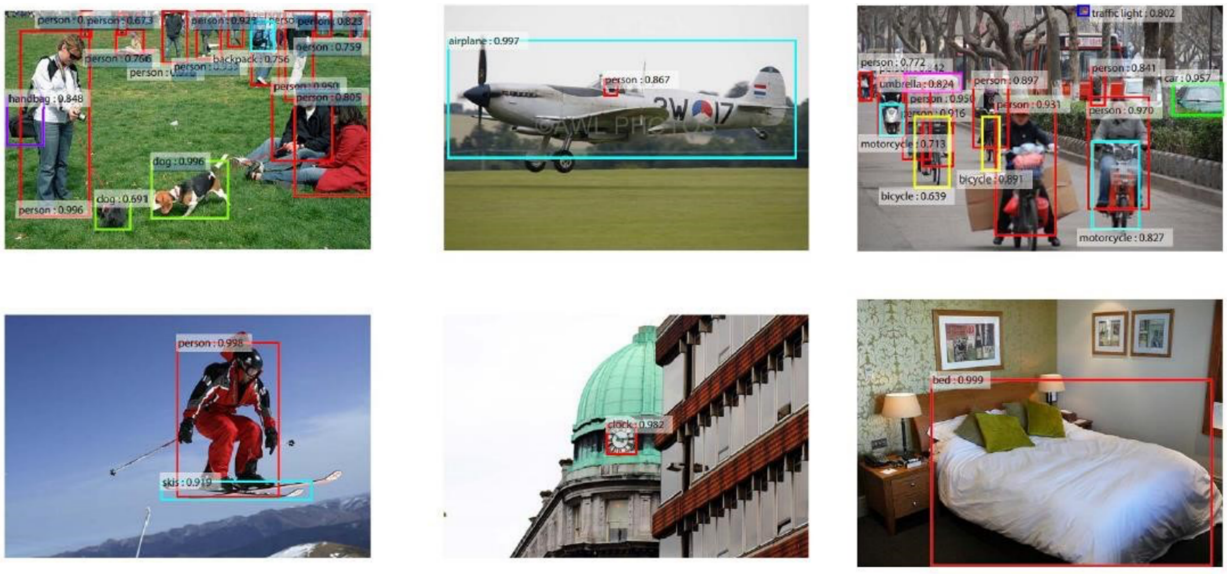
- 目标：在对图像进行分类的同时，用一个边界框（Bounding Box）精确地标出该物体在图像中的位置.



- 输出：单一类别标签 + 一个边界框（通常由四个顶点坐标组成）.

目标检测（Object Detection）

- 目标：识别图像中所有感兴趣的物体，并为每个物体提供类别标签和精确的边界框.



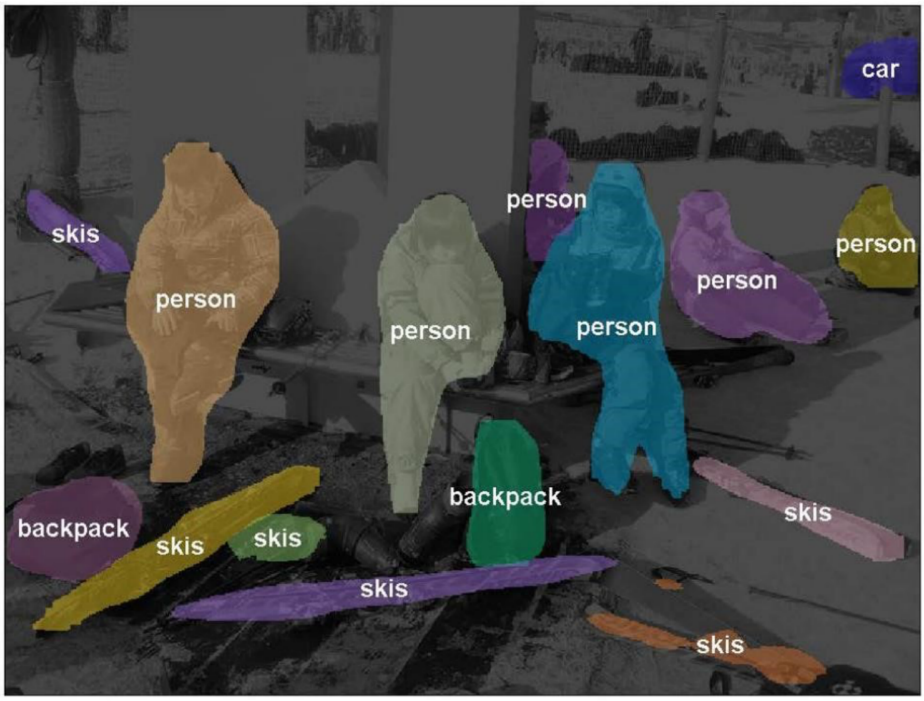
Results from Faster R-CNN, Ren et al 2015

- 输出：多个边界框，每个边界框伴随一个类别标签.

实例分割（Instance Segmentation）

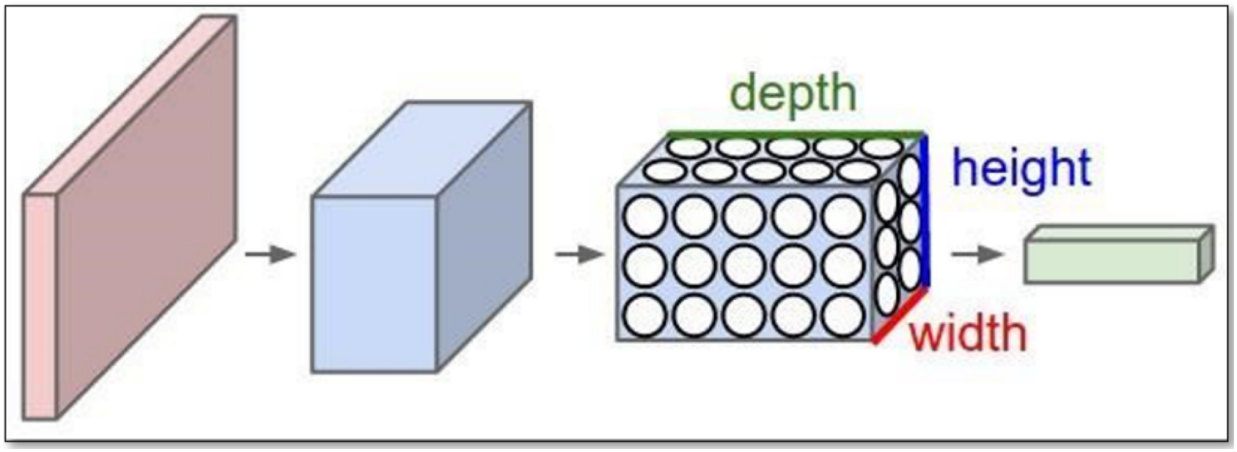
- 目标：识别图像中的所有物体，并为每个物体在像素级别进行精确的分割，即将属于该物体的每一个像素都标记出来.
- 输出：多个像素掩码（Pixel Masks），每个掩码对应一个类别标签和一个物体实例.

input



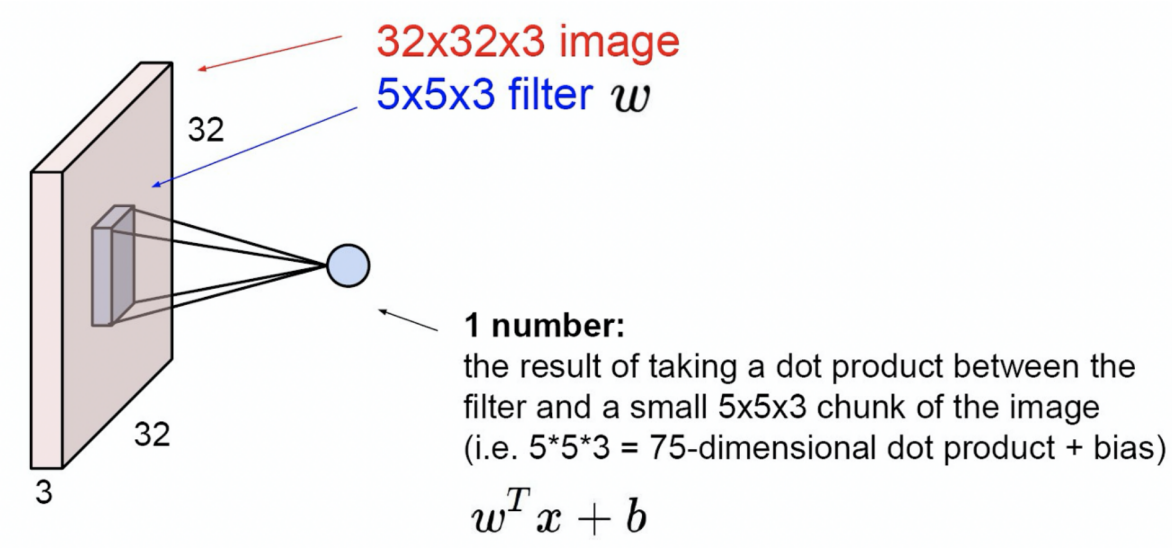
7.2 网络架构

常规神经网络（NN）以层状排列，而 CNN 将神经元排列成三维（3D 输出张量）.



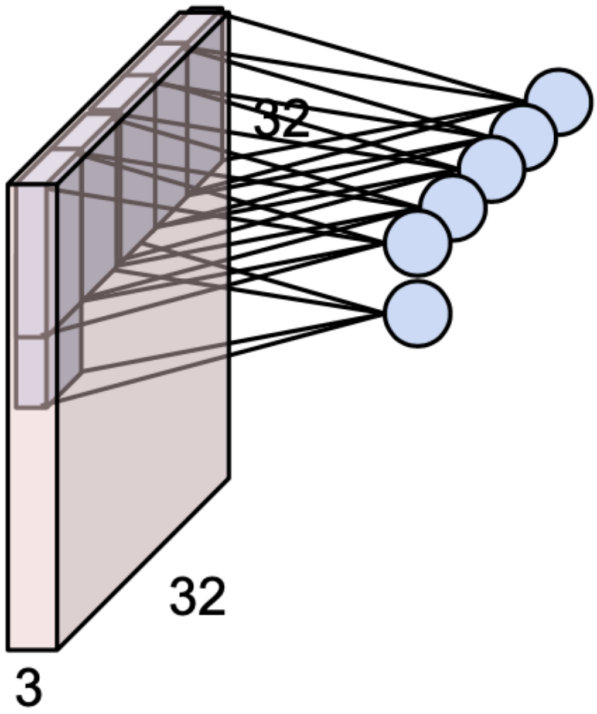
- width × height：特征图（Feature Map）的尺寸，可以看作单个特征图的平面.
- depth / channel：深度 / 通道数，等于该层所使用的滤波器（Filter / Kernel）数量. 每个滤波器会产生一个独立的、具有 width × height 尺寸的特征图.

7.2.1 卷积层

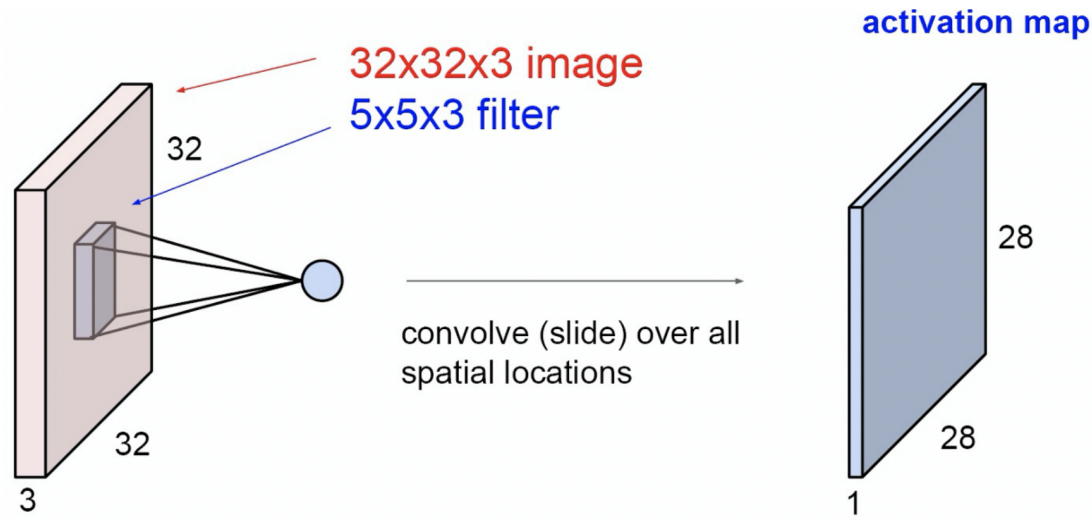


每个卷积核从左到右、从上到下进行卷积操作. 每次卷积的乘法次数是卷积核的体积大小，然后把乘法结果相加，再加1个偏置，经过激活函数得到特征图上一个单位的数.

卷积操作很简单，就是对应位置相乘，然后把所有乘法结果相加.



一个卷积核完成所有卷积操作后（从左上开始，到右下结束），把卷积结果加上偏置，经过激活函数，生成一个特征图：



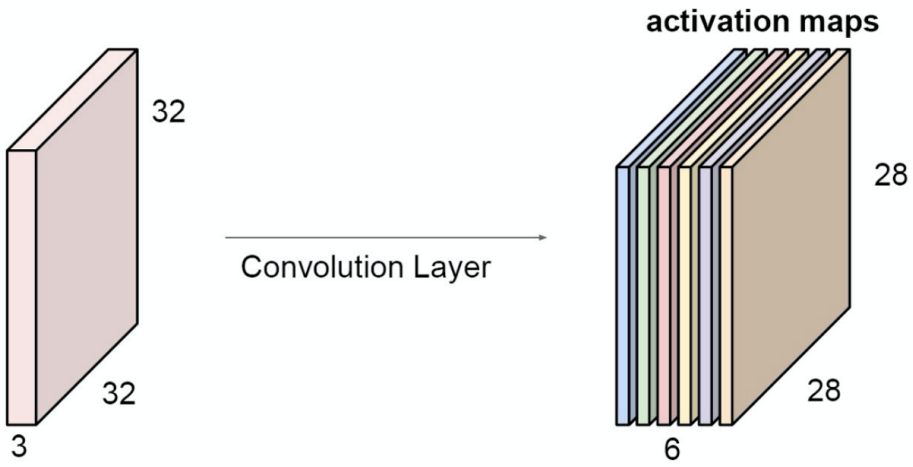
Activation map is also called feature map.

注意，卷积核的高度和宽度是超参数，但是深度必须和被卷积的层（输入特征图）一致. 这是因为卷积的目的在于沿着深度方向对信息进行聚合（压缩），将所有通道的信息结合起来生成一个单一的输出特征图. 如果卷积核深度小于输入特征图，生成的单个特征图深度就大于 1，这没有意义.

每个卷积核生成一个深度为 1 的特征图，把多个卷积核生成的特征图沿深度方向叠加，就得到总的输出特征图（如果后面还有卷积层，就作为下一个卷积层的输入特征图）.

因此，输出特征图的深度数量取决于使用的卷积核数量，和输入特征图的深度无关.

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

The output dimension is also known as channel.

输出维度 / 通道数 / 输出特征图的深度都是指同一个量，即这一层卷积层使用的卷积核数量.

① 四个超参数

注意到输出特征图的高宽深和输入特征图不一定相同. 它被五个参数影响：

- 滤波器（卷积核）数量 K：影响深度.
- 输入特征图高宽 N，滤波器大小 F（通常是正方形，边长记为 F；深度必须和输入一致，不用考虑深度）、步长（S，表示 stride）、零填充宽度（P，表示 padding）共同影响输出特征图的高宽.

其中 K, F, S, P 是卷积层的四个超参数.

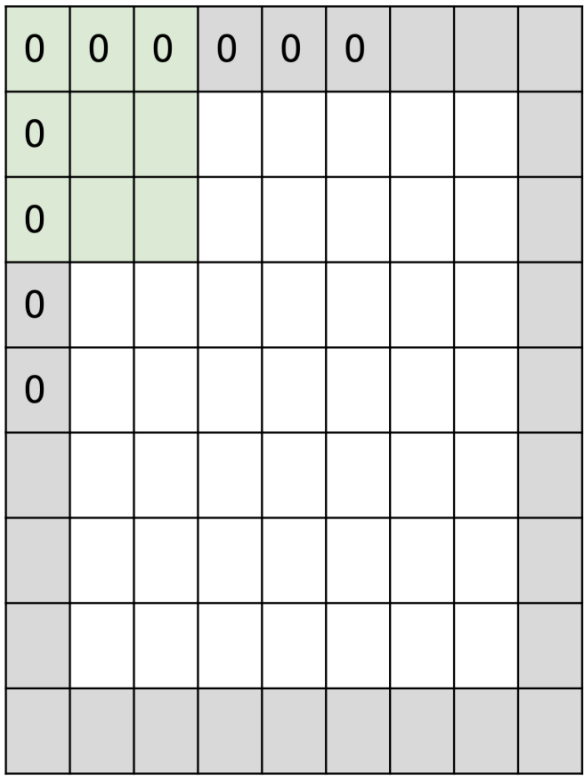
zero-padding：实践中，通常使用 0 来扩充边界. 各个方向都要填充. 若 padding 为 3，则数学上看边长实际增加了 6.

输出特征图边长：$\frac{N+2P-F}{\text{stride}}+1$

> 加一是因为起始位置就能生成一个值，占一个位置.
>
> 如果不能整除，就下取整（跳过最后超出边界的一步）. 不过通常可以通过恰当的零填充让它可以整除.

输出特征图深度：K

例：input 7×7 image, use 3×3 filter, convolve with stride 1, padding with 1 pixel border. What is the output size?



$$\frac{N+2P-F}{\text{stride}}+1=\frac{7+2\times 1-3}{1}+1=7$$

output size: 7×7

实践中，为保持输入输出大小不变（不是必要），通常使用 zero-padding with $\frac{F-1}{2}$. stride of 1. 这样能让 $\frac{N+2P-F}{\text{stride}}+1=N$

② 训练权重参数

每层卷积层的权重参数通过训练得到，每个卷积核的每个位置都是一个权重参数. 每个偏置也算一个权重参数.

每层的权重参数数量由几个参数决定：

- 滤波器（卷积核）数量 K：参数量和 K 成正比. 注意每个卷积核除了自身单元里的参数，还额外带一个偏置.
- 滤波器大小 F（滤波器越大，参数越多）
- 滤波器深度 d（它和输入特征图的深度相等，因此通常会隐藏在题目中）

参数数量计算：$(F\times F\times d+1)\times K$

> $F\times F\times d$ 是单个滤波器体积，$+1$ 表示每个滤波器额外带一个偏置参数，$\times K$ 表示共 K 个滤波器.

例：

For a convolutional layer:

input volume: $224 \times 224 \times 3$

50 个 8×8 filters with stride 2, pad 3

Number of parameters in this layer?

$$(F \times F \times d + 1) \times K = (8 \times 8 \times 3 + 1) \times 50 = 9650$$

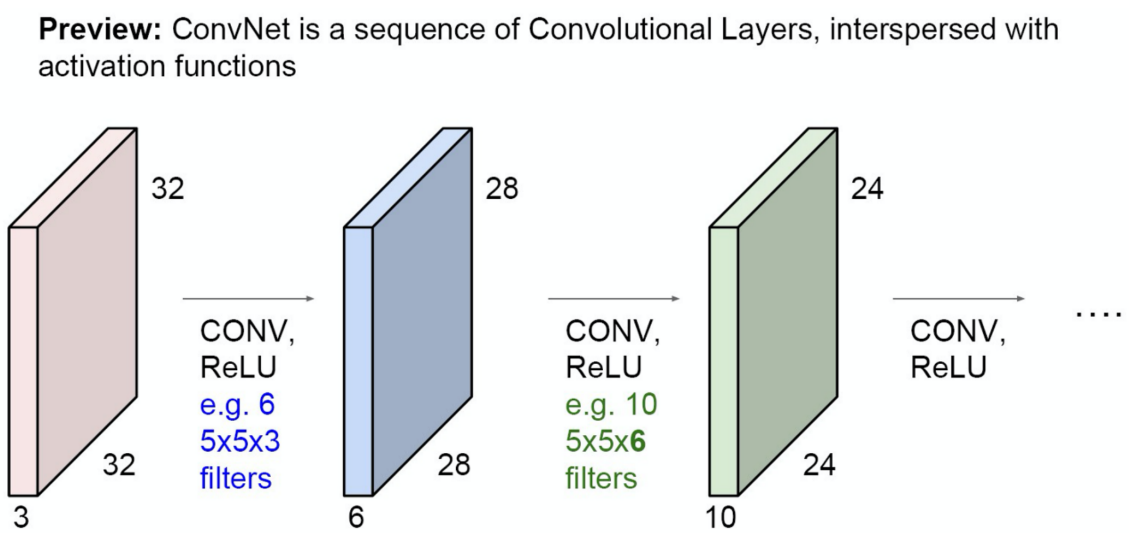
注意，这里 $d = 3$ 隐藏在输入特征图的深度中了.

总结:

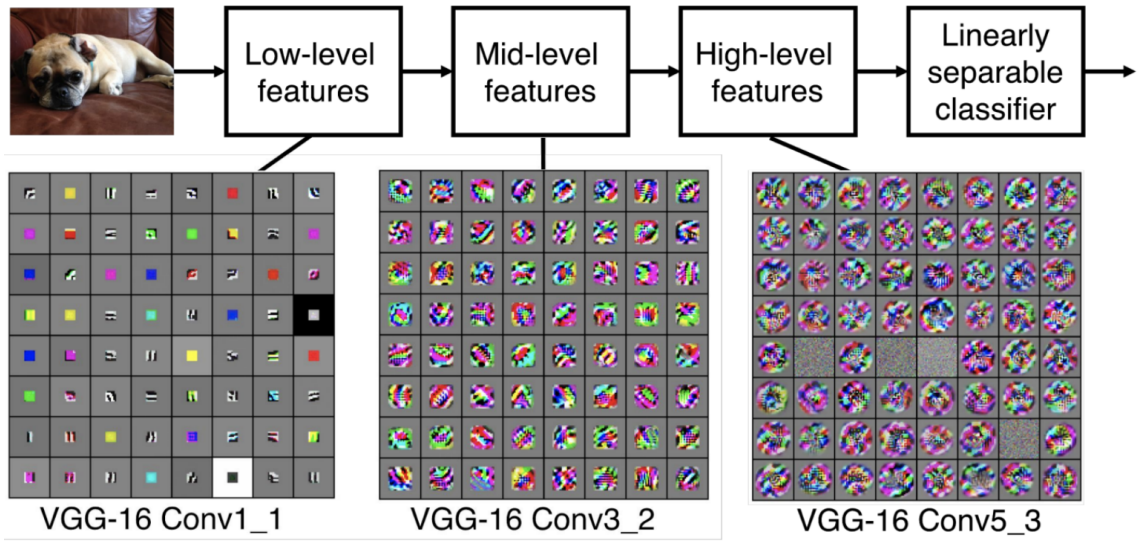
- Receives a 3D volume of size $W_1 \times H_1 \times D_1$ and requires four hyper-parameters:
  - number of filters K
  - filter spatial extent F
  - the convolution stride S
  - the amount of zero padding P
- Produces a volume of size $W_2 \times H_2 \times D_2$ with:
  - $W_2 = (W_1 + 2P - F)/S + 1$
  - $H_2 = (H_1 + 2P - F)/S + 1$
  - $D_2 = K$
- It introduces $F \times F \times D_1$ weights in each filter, in total, for a convolutional layer, it has $(F \times F \times D_1) \times K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

③ 卷积+偏置+激活

一般提到卷积层，作为整体模块，包含卷积操作、加偏置、经过激活函数:



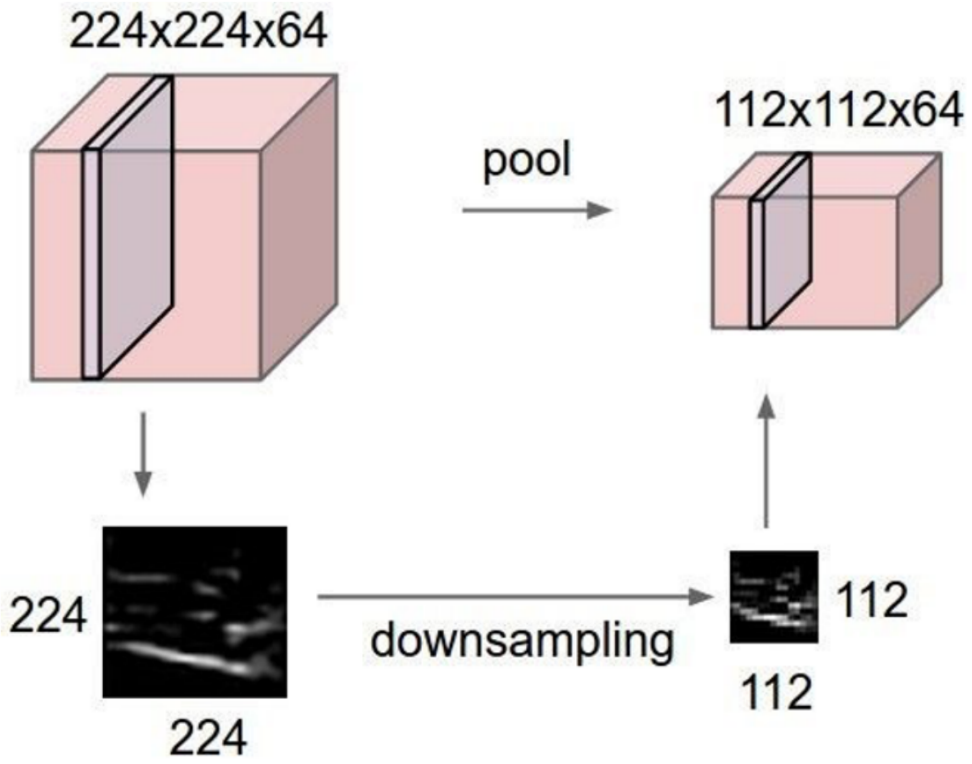
卷积层越往后，捕捉的特征越抽象、复杂：



7.2.2 池化层

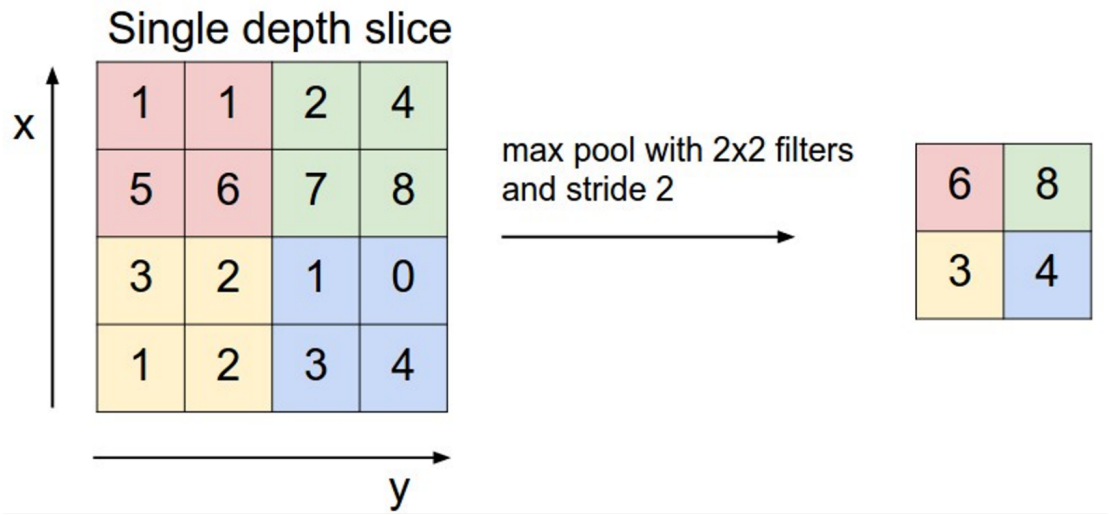
> Pooling layer

本质：降采样，减少参数量，节省计算资源.



通常在若干个卷积层之后跟一个池化层.

常见池化：Max Pooling



池化层对输入特征图的每一层（多层叠加）进行降采样，然后再叠起来，因此深度不变，只改变大小.

注意：池化层**不包含参数**，因为它只是一种指定的过滤方式.

记 input $W_1 \times H_1 \times D$

池化层需要两个超参数：F_p 和 S_p，表示池化过滤器的大小和步长. 池化的输出为 $W_2 \times H_2 \times D$:

- $W_2 = \frac{W_1 - F_p}{S_p} + 1$
- $H_2 = \frac{H_1 - F_p}{S_p} + 1$

Number of parameters: 0

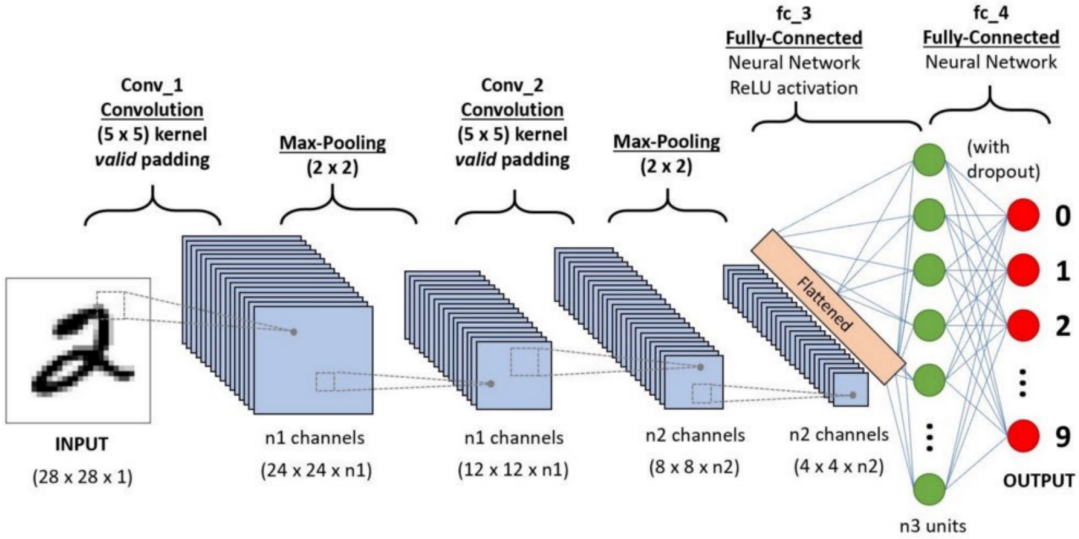
注意：最大池化在反向传播时需要结合梯度路由，把梯度传回之前选择的位置，其他被过滤掉的位置无法更新（梯度稀疏性）.

7.2.3 全连接层

> Fully-connected layer

最后一层卷积层的输出经过池化，然后展平，得到一系列值，这一个长向量就是全连接层的输入.

"全连接" 的含义：输入神经元与前一层的所有激活值都完全连接，用于最终的分类或回归任务.



全连接层就是一个普通的神经网络，或者说多层感知机. 它可以完成回归、分类等任务.

7.3 著名架构

待补充.

7.4 残差网络

> ResNet

退化问题（Degradation）：Kaiming 发现简单地增加网络深度，表现会先变好，再变差（无论是训练集还是测试集）.

如果训练集持续变好，但测试集越深越差，这可能是过拟合；但反常现象在于，训练集在一定深度之后，也会越深越差.

这说明某些原因导致 "overly deep" 带来更高的训练误差.

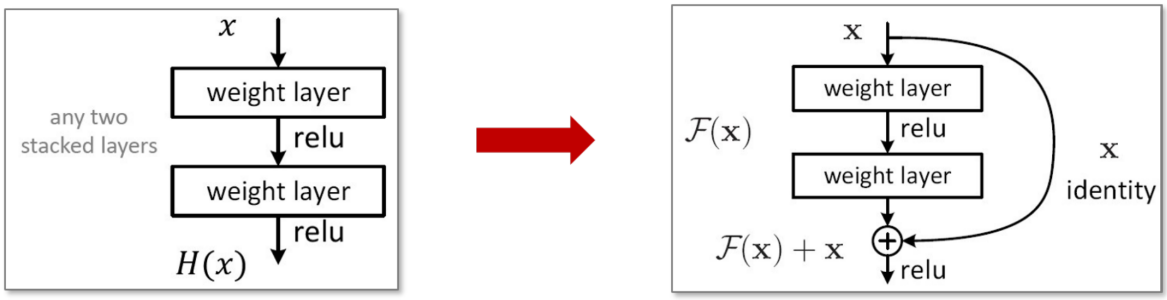
反常的点在于：理想情况下，控制超参数一致（性能一致），一个深层网络的表现应该不低于一个浅层网络，因为它可以简单地复制某个已训练好的浅层网络的层，并在中间添加一些恒等映射层实现相同效果.

问题在于：恒等映射在目前架构下，容易被习得吗？答案是否定的.

恒等映射 $H(x)=x$，看似很简单，实际上：

- x 是一个输入特征图，是一个立方体.
- 学习恒等映射，该卷积层的超参数要设置为 $F=1$，$P=0$，$S=0, K=D_x$，其中 D_x 是输入特征图的深度. 偏置强制设置为 0
- 然后，要通过训练，把第 i 个卷积核的权重训练为：第 i 通道权重为 1，其他通道权重为 0，这样，它与输入特征图的第 i 个通道卷积的结果就是整个输出特征图的第 i 层，相当于把输入特征图的第 i 层复制到输出特征图的第 i 层. 然而，要通过梯度下降的方式训练出这样的权重分布是很困难的.
- 注意到，把所有卷积核按顺序排在一起，权重可以组成一个单位矩阵. 左上到右下为 1，其余为 0.
- 如果是多层连续的恒等，每层都要学到这样的权重分布. 很困难.

换一种思路：对于优化器（如 SGD），将所有权重推向 0，比学习一个复杂的权重分布容易得多. 如果我们在输出中添加输入，那么学习的映射就变为：



这里，$H(x)=ReLU(w_2ReLU(w_1x+b_1)+b_2)$ 是普通网络学习的非线性映射，$ReLU(F(x)+x)$ 是残差块学习的非线性映射. $F(x)+x=x$ 比 $w_2ReLU(w_1x+b_1)+b_2=x$ 好学得多. 只需要让
$F(x)=w_2ReLU(w_1x+b_1)+b_2\approx 0$ 即可.

注意：如果要实现残差连接，必须保证残差块，输出特征的高度宽度深度和输入完全相同（逐元素相加）.

> 实际实践中，如果无法保证维度相同，可能使用投影连接（而非这里的恒等连接）.

7.5 训练技巧

7.5.1 数据增强

以图像分类任务为例，对每个训练数据保持标签不变，输入图像进行反转、旋转、裁切、遮挡、变色、模糊、加噪等操作，然后全部添加进训练集.

优点：Adding more training data into the models.

- Alleviating data scarcity for learning better models.
- Reducing data overfitting (i.e., the learned network corresponds too closely to a limited set of data points). Augmentations help create variability in data.
- Helping resolve class imbalance issues in classification（例如，训练集中猫的照片是狗的两倍，可以对所有狗照片反转加入训练集. 否则当模型难以判断时，理想情况应该输出概率 0.5，但模型可能偏向于更常见的猫）.
- Reducing costs of collecting and labeling data.
- Increasing generalization ability of the models.

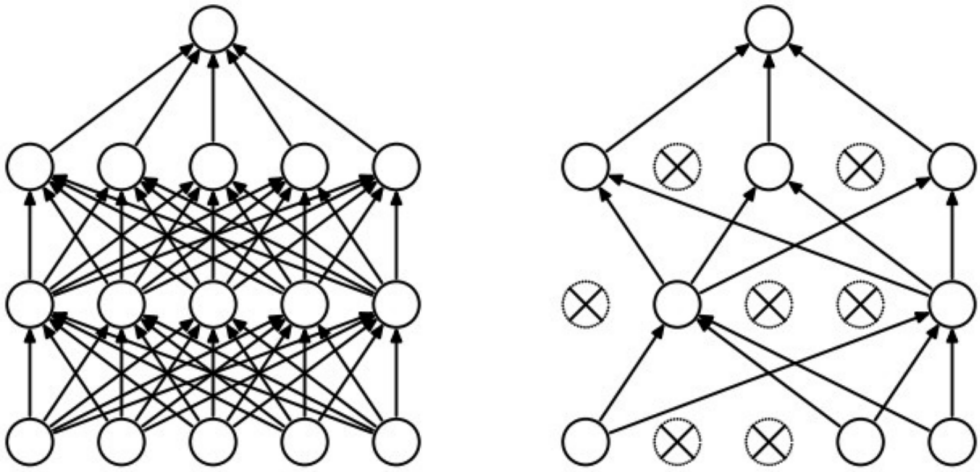
7.5.2 Dropout

> 暂退法，正则化技术

Dropout is an extremely effective yet simple regularization technique by Srivastava et al.

While training, dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter, usually $p=0.5$)

This means that a neuron is temporally removed in the forward pass and any weight updates are not applied to this neuron on the backward pass.



(a) Standard Neural Net (b) After applying dropout.

关闭神经元可以简单地将激活输出设置为 0. 对于卷积层，也可以关闭神经元，即把特征图某些像素点设置为 0.

优点：

- Preventing all neurons in a layer from synchronously optimizing their weights. Random dropout prevents all the neurons from converging to the same goal (or called co-adaptation).
- In other words, if neurons are randomly “turned-off” during network training, the remaining neurons will have to step in and handle the representation required to make predictions for the missing neurons.
- Dropout prevents co-adaptation by making the presence of other neurons unreliable. Thus, a neuron cannot rely on other specific neurons to make predictions. Instead, it must perform well in a wide variety of different contexts provided by other neurons. Therefore, dropout improves model generalization capability.
- Dropout, in practice, as a regularization of the deep neural networks to reduce network overfitting.

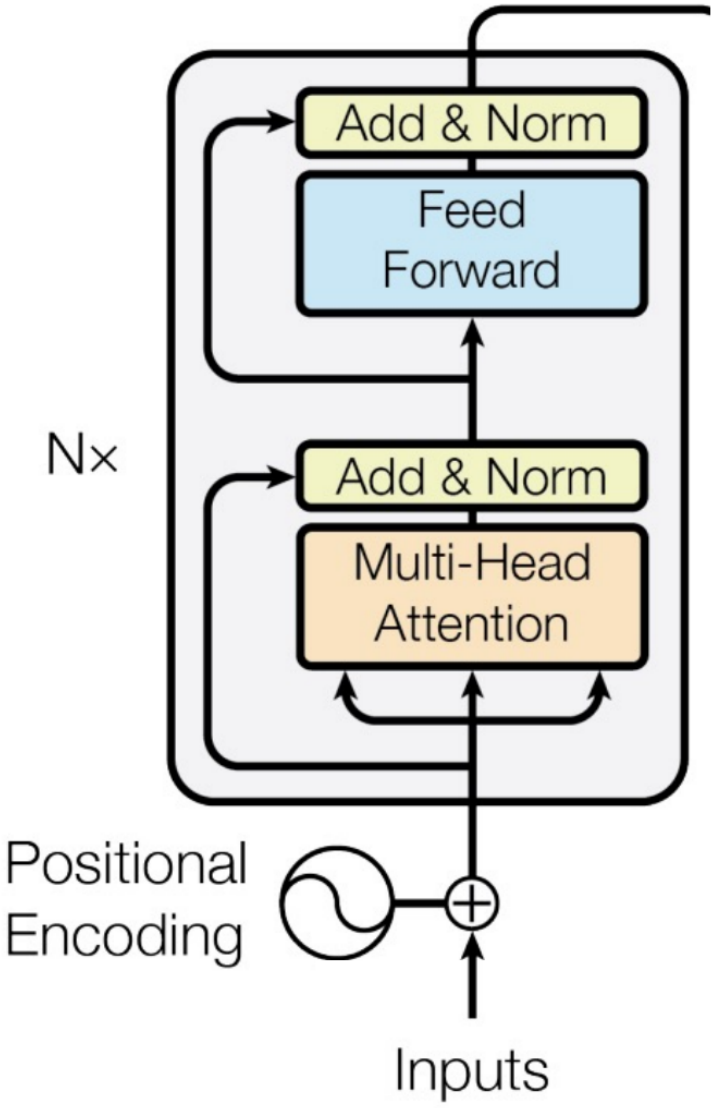
Lecture 8 Transformer

> 暂时没有标准的翻译，可以译为 “变换器”（或 “变形金刚”）.

The Transformer is a neural network architecture initially designed for sequential modeling in natural language processing (NLP), and now applied to fields like vision and audio.

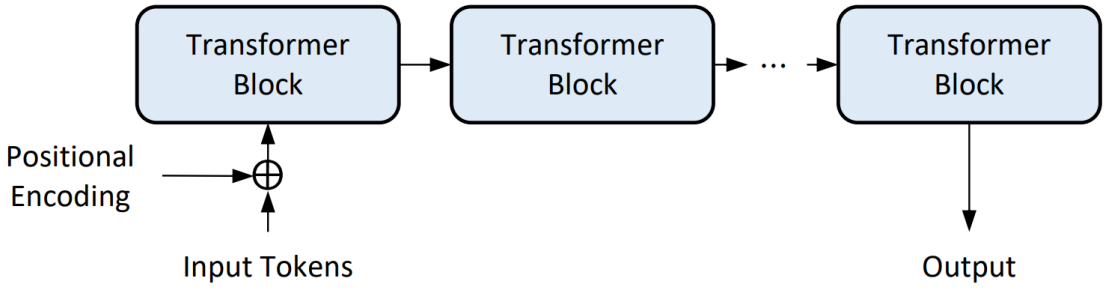
最初用于翻译任务.

Structurally, a Transformer is composed of linear layers, normalization layers and activation functions.



Add：残差连接.

Norm：层归一化.



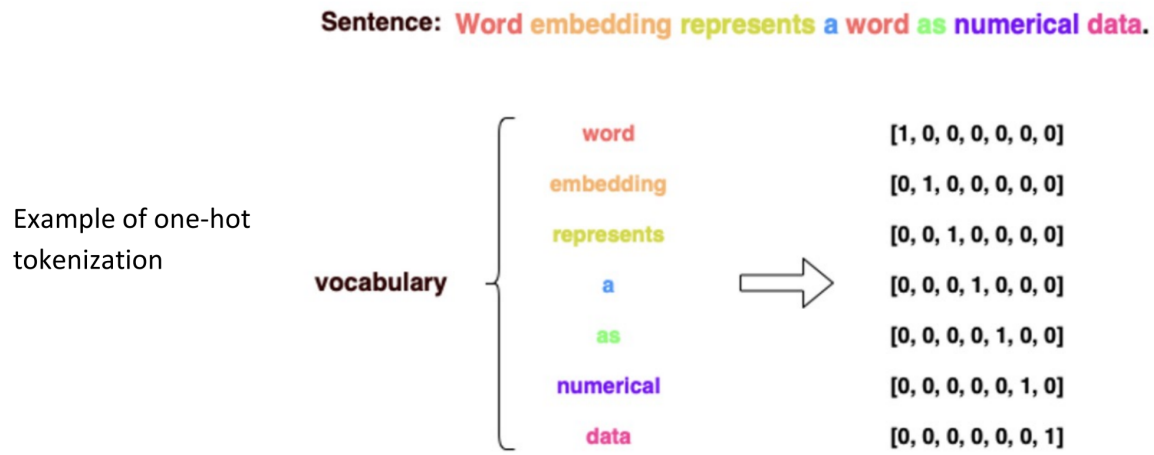
越后面，生成的特征嵌入越有代表性，越和本次输入文本的实际含义有关.

8.1 分词

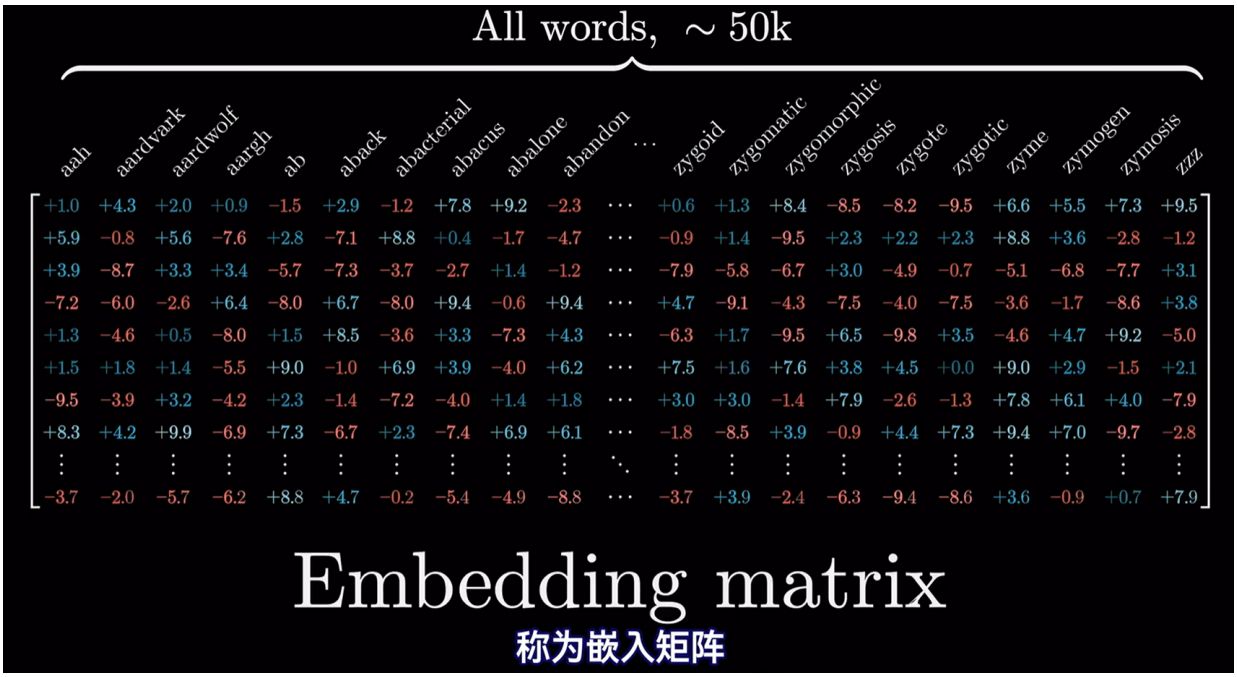
> Tokenization

Before feeding into Transformer, the input sentence should be first converted into a sequence of **tokens**

- Each token is an 1D vector
- The initial tokens are obtained by looking up a dictionary that stores the mapping between words and tokens



现代 Transformer 通常不用 one-hot 嵌入，而是用嵌入层（Embedding）将离散的 token ID 转换为连续的稠密向量.



嵌入矩阵（Embedding Matrix）**不是单独训练的**，而是作为 Transformer 模型整体结构的一部分，**与整个 Transformer 模型（包括所有注意力层、前馈网络和输出层）一起进行**端到端（End-to-End）的联合训练.

用嵌入矩阵，可以包含语义相似度信息.

Transformer 最后一层会对最右侧向量进行解嵌入. 得到 Logits 分布，经过 Softmax 之后得到概率分布.

Transformer 的上下文窗口固定，如果输入序列的 Token 数量小于最大上下文长度，模型会在序列末尾填充特殊的 Padding Token，把序列扩展到刚好最大上下文. 然后模型会生成一个注意力掩码，在计算注意力分数时，所有指向 Padding Token 的注意力权重都会被设置为一个极小的负数（接近负无穷），经过 Softmax 后，这些权重会接近于 0. 模型计算注意力时会忽略这些填充 Token，确保它们不会对真实输入 Token 产生影响.

填充 Token 也有位置编码.

如果输入序列的 Token 数量超过最大上下文长度，模型必须截断（Truncation）或分块处理（Chunking）.

8.2 位置编码

位置编码（PE）以**向量**形式存在和运算，并且必须与 Token 嵌入的维度保持**一致**.

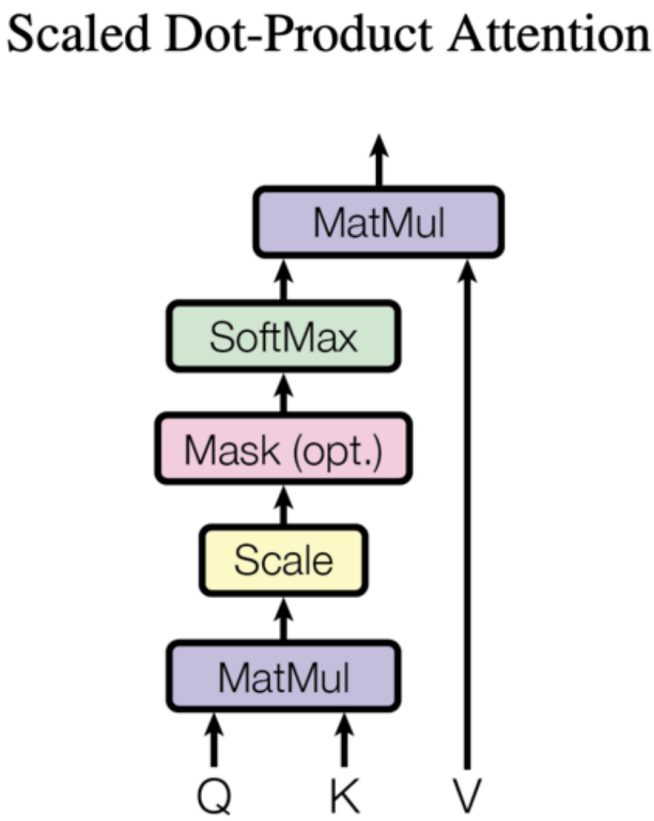
原始 Transformer 论文中，PE 是用正弦和余弦函数生成的，根据 token 在序列中的位置和向量的维度计算出一个确定值.

输入：原始输入加上位置编码作为第一层的输入.

可恢复性：如果 $Z = A + B$，已知 Z 和 B 可以还原出 A. 而位置编码是已知的，因此相加不会抹去原始语义信息.

8.3 注意力机制

在生成式 Tranformer 中，通常讨论的是自注意力（self-attention），即同一个序列的查询向量和键向量两两相乘.

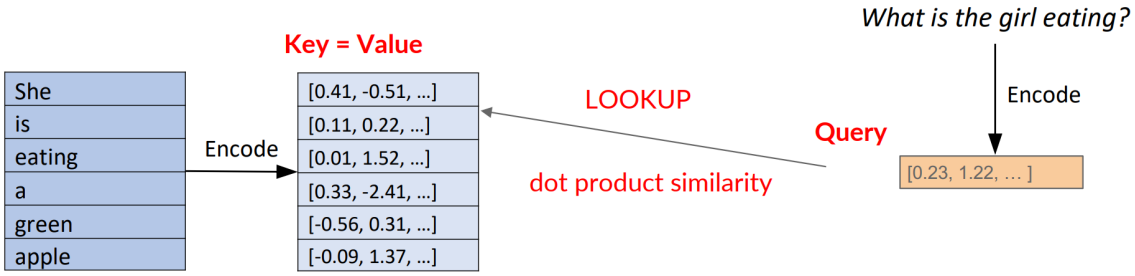


8.3.1 单头注意力

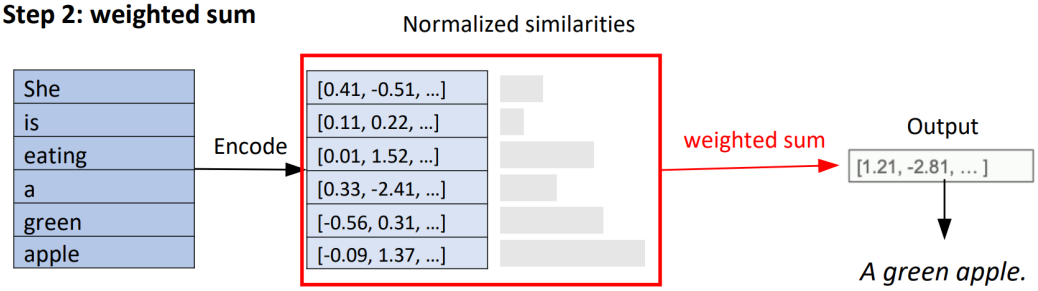
The attention mechanism is based on three types of vectors as a retrieval system: query, key, value.

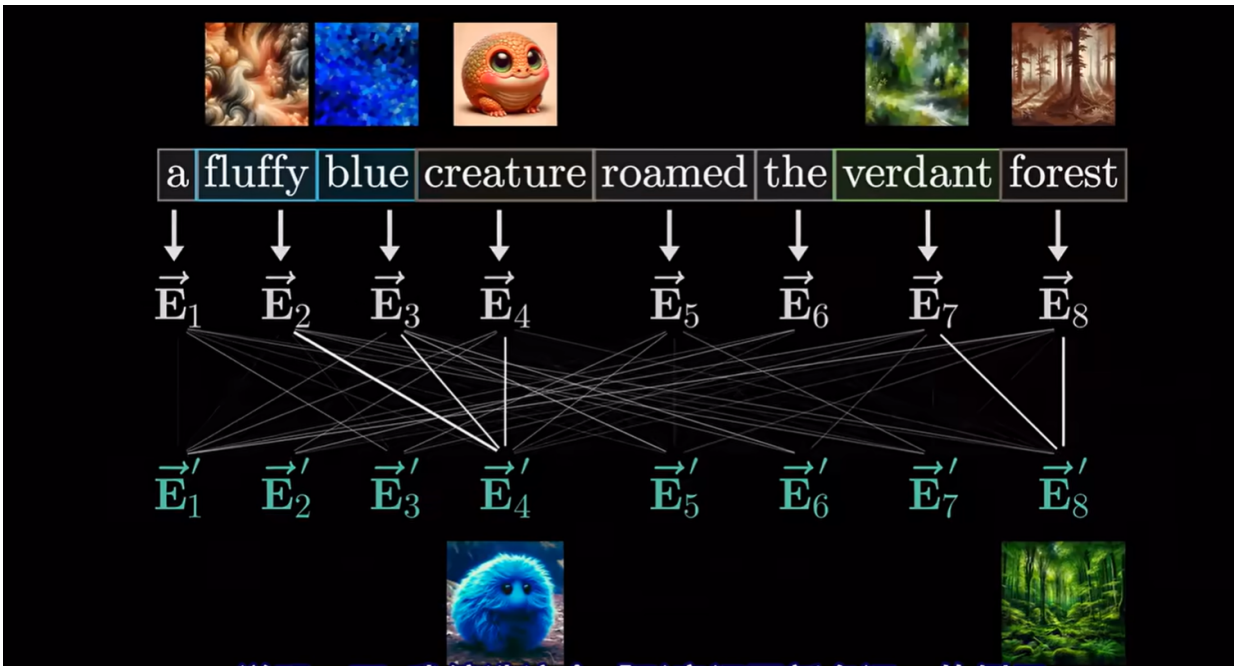
The attention mechanism operates on a find-then-retrieve paradigm.

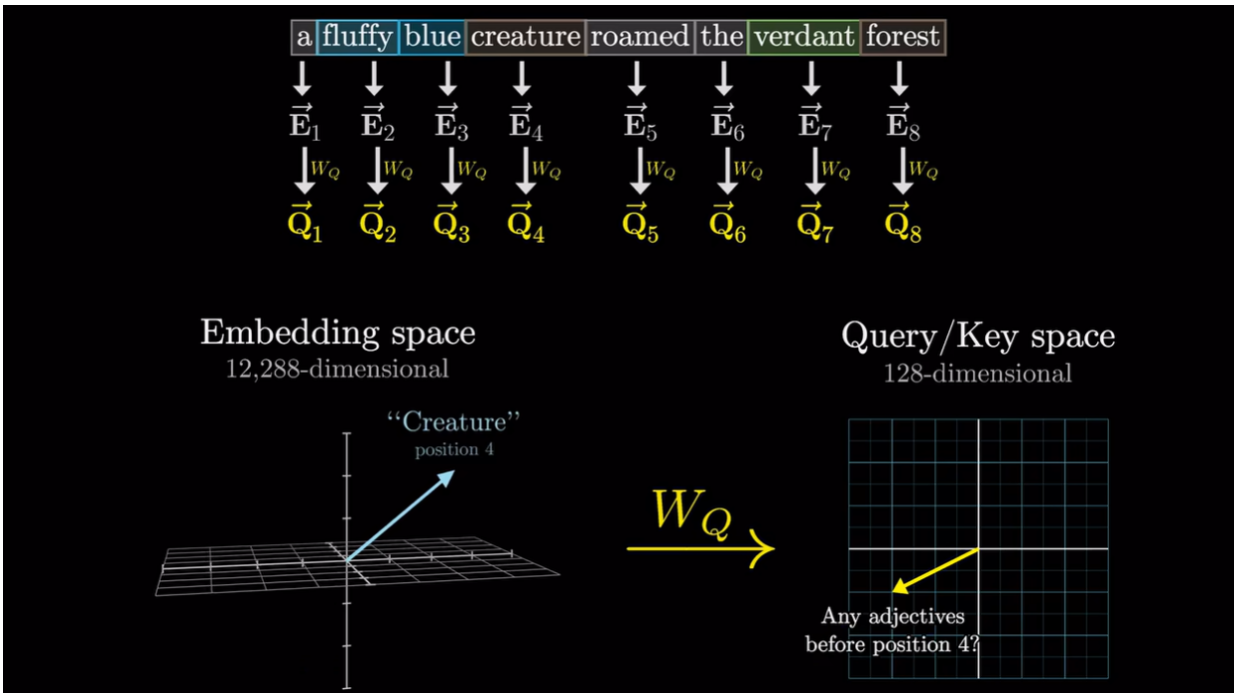
Step 1: similarity calculation



Step 2: weighted sum







查询矩阵和嵌入向量相乘，可以给每个向量生成它们的问题（查询向量）. 本质上就是将 Token 的原始输入空间（Embedding Space）线性映射到了一个新的查询空间（Query Space）

同样，键矩阵和嵌入向量相乘，给每个向量生成它们的键向量. 键向量和查询向量的维度对齐.

然后，把键向量和查询向量做点积，衡量向量相似度. 点积越大越匹配.

点积的结果是注意力分数. 之后要经过缩放和 Softmax 得到注意力权重.

注意，键矩阵和查询矩阵在操作上一致，但是可以分辨，因为注意力分数计算完后，是对每个查询向量列进行 Softmax. 该查询向量和每个键的注意力分数被转化为概率分布（权重分布）. 这个权重分布对值向量加权求和就得到需要的值.

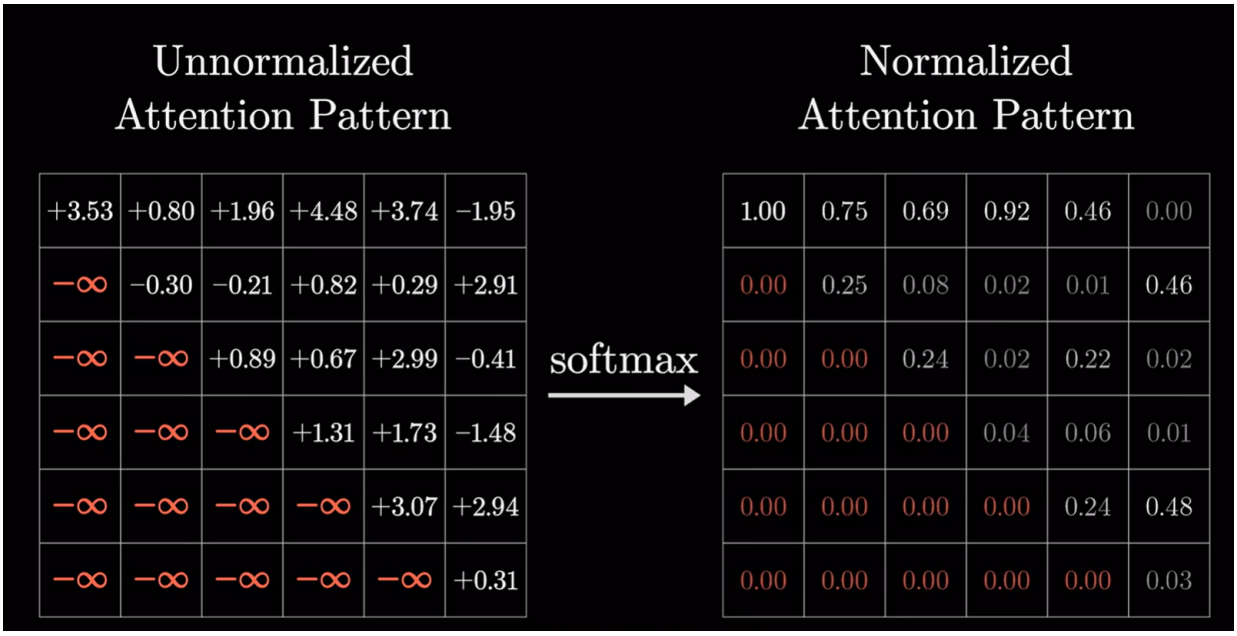
键向量的作用是充当信息地址或标签，它被设计用于与 Q 向量进行高效的匹配和比较. 因此，W^K 矩阵学习的是如何将 Token 映射成最适合比较的表示. 值向量的作用是充当信息内容. 它代表了 Token 实际携带的、需要被提取和聚合的语义信息. 因此，W^V 矩阵学习的是如何将 Token 映射成最适合聚合的表示.

把键向量和值向量分开，可以解耦地址和内容.

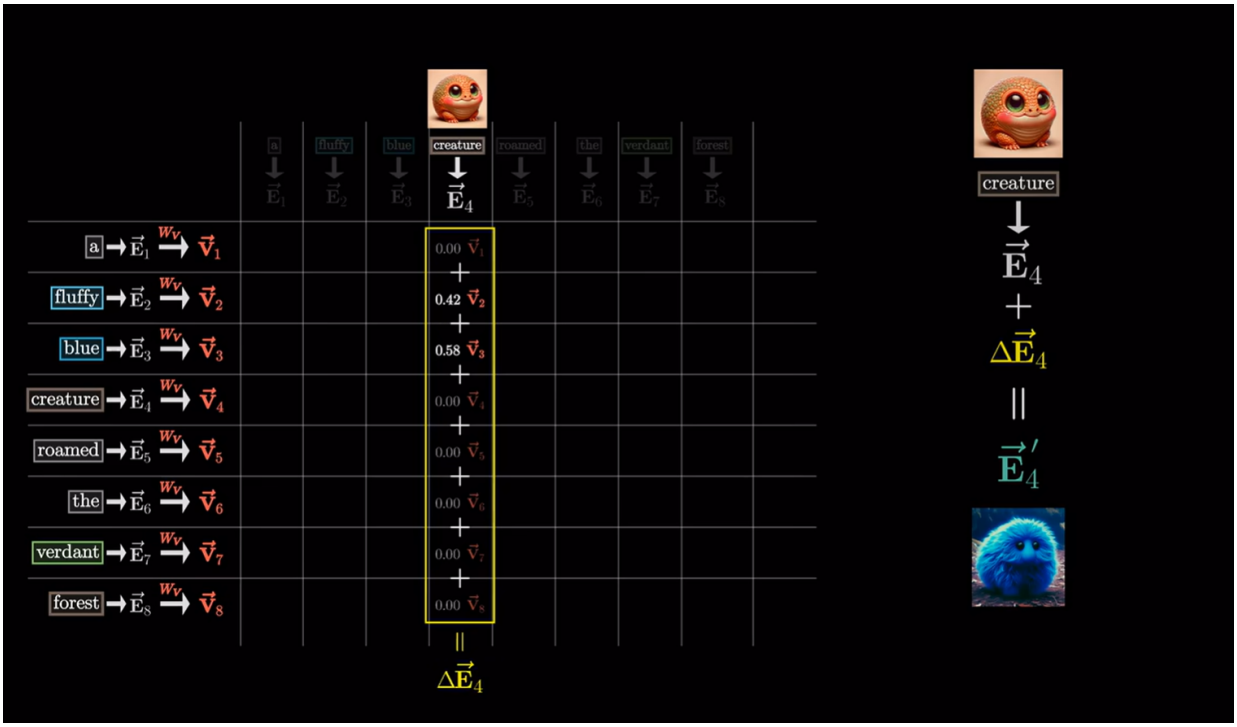
$$\text{Attention}(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

这里 $\sqrt{d_k}$ 是键向量的维度，因为 softmax 在 0 附近才有显著变化，如果太高维，不缩放的话，梯度过小.

Masking（可选）：对于 GPT 类型的模型，因为一条训练文本我们对每个位置都进行预测和算损失，因此我们为防作弊，不能让预测位置之前的词注意到预测位置及之后的词.



计算一次注意力需要进行上下文长度平方次点乘，因此随上下文窗口增大计算量迅速增加.



这里就是残差连接.

以上讨论的都是自注意力机制. 其他模型如翻译，键矩阵和查询矩阵作用在不同的数据上（不同语言），叫做交叉注意力，不masking.

8.3.2 多头注意力

并行执行不同的单头注意力（不同的键、查询、值矩阵），把所有单头注意力产生的加权向量 ΔE 全部求和，再加到初始嵌入 E. 这个和就是多头注意力模块输出的一列.

目的是，不同单头注意力可以学习不同的上下文改变语义的方式.

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\dots,\text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

这里 Q,K,V 是指输入给多头注意力的三个相同的嵌入向量序列.

Concat 是拼接，如果每个头的值向量维度是 d_{model}/h，拼接后整个序列的维度就是 $N \times d_{model}$（单个 Token 的嵌入向量通常表示为行向量，N 是 Token 数，即上下文窗口）. W^O 是可学习的权重矩阵，它对来自 h 个头拼接而成的 d_{model} 维向量进行线性变换. 它的维度是 $(h \cdot d_v) \times d_{model}$，标准情况下是 $d_{model} \times d_{model}$，非标准情况下可以把 $N \times (h \cdot d_v)$ 的序列值向量拼接映射到 $N \times d_{model}$ 的可用于残差连接的序列向量.

引入 W_O 的优势：如果只是简单地让每个头生成 d_{model} 维向量，再用一个 h 维向量对它们加权平均，这样 head_1 的第 100 维特征和 head_2 的第 50 维特征就无法进行组合.

8.4 层归一化

在 Transformer 的每个子层（多头自注意力层和前馈网络层）的**残差连接（Add）之后**.

计算：独立地对 Token 序列中的每个 Token 的嵌入向量进行操作. 求平均、求方差、归一化，仿射变换.

作用：确保每层神经元的输入分布保持稳定，从而加速收敛，并允许模型使用更高的学习率. 通过这种方式，Transformer 中的每个 Token 向量在进入下一个子层之前，都会被独立地、稳定地调整到一个相似的数值范围，这对于深度 Transformer 网络的稳定训练至关重要

8.5 前馈神经网络

自注意力捕捉的都是token间的非线性关系，所以单个token的向量内部各分量之间的非线性关系需要经过FFN来捕捉.

Add & Norm 之后的向量是信息的混合体，将 Token 的原始表示与从整个上下文中聚合到的新语义信息（即 MultiHead Attention 的输出）简单地线性叠加在一起（相加和归一化）.

模型需要一个非线性的、高维度的空间来真正解析、融合与提炼这些混合信息.

每个 Transformer 块，输入在经过多头注意力、残差连接和层归一化后，都要传入第二个子层 FFN（Feed-Forward Network）. FFN 是一个标准的两层全连接神经网络，它独立地对每个 Token 进行非线性变换. 它包括两个线性变换和一个激活函数（引入非线性）.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

第一层 $xW_1 + b_1$ 是扩展维度，给模型提供一个更宽广的空间来学习复杂的非线性特征.

第二层 $(\cdot)W_2 + b_2$ 是将扩展后的高维向量映射回原始的维度（即嵌入空间的维度）

注意：进入 FFN 后的维度比嵌入向量维度还高，但是出来的时候会降维.

在FFN的变换就是模型把原始聚合信息的向量在高维空间中移动，然后再投影回嵌入空间，现在它的位置就精确地对应了融合新语义信息的token了.

举个例子："a big green apple". 我们考虑 apple 对整个 token 序列的查询，得到多头注意力返回的 "big" 和 "green" 向量（有的头关注大小，有的头关注颜色）. 然后，把它们拼接在一起，表示 "big green". 但是这是简单地首尾相接，因此需要经过 W^O 得到准确的表示 "big and green" 的一个向量. 然后，把 "apple" 和 "big and green" 相加（残差连接），但这只是简单地线性叠加，无法真正融合语义，因此传入 FFN，嵌入高维空间，激活，再投影回原始嵌入空间. 只要模型训练得当，经过这样的升维和降维，重新得到的向量就是表示 “big and green apple” 的混合了新语义 "big and green" 的原始 token "apple" 了（模型通过学习 FFN 的两个权重矩阵实现这一点）.

共享权重：权重矩阵是为了学习特征转换逻辑，它不应该随 Token 的具体值而变化，而是应该对所有的嵌入向量都能够用同一个权重矩阵转为非线性. 这样可以减少参数数量.

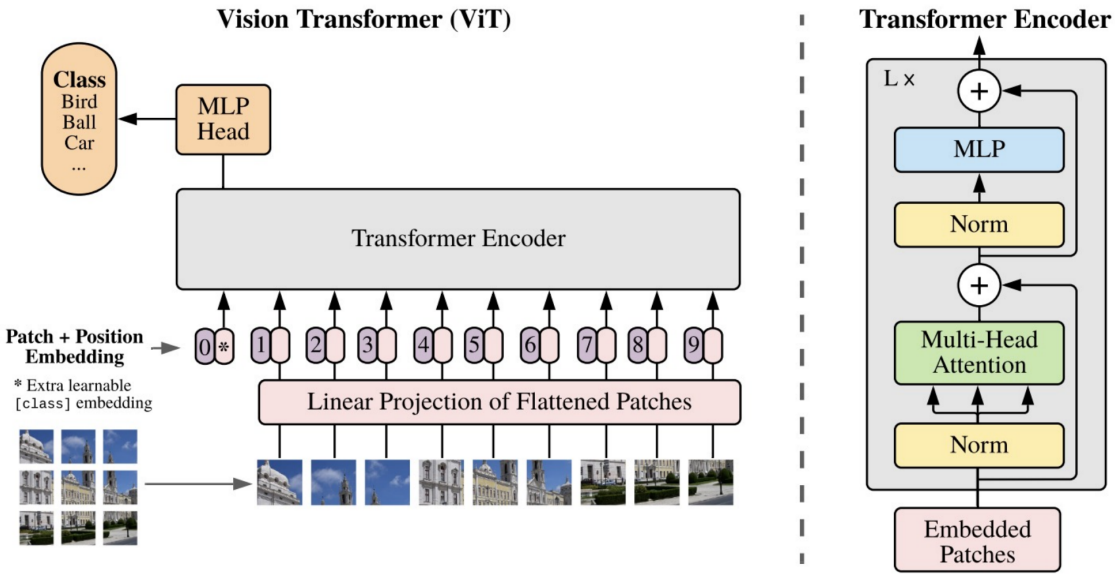
8.6 训练

预训练直接用训练文本，不需要人工标记. 以 GPT 为例，任务是预测序列中的下一个 Token（自回归任务）. 损失函数：交叉熵损失. 以GPT的预训练为例，损失函数是对每个未预测token的交叉熵损失求和.

8.7 视觉 Transformer

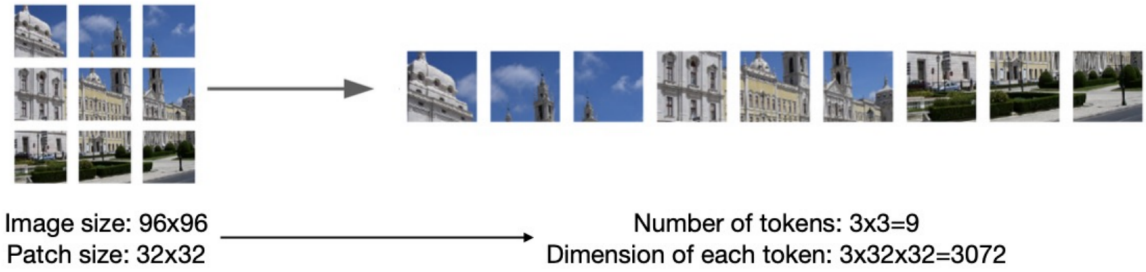
> Vision Transformers, ViT

由于Transformer是为序列建模设计的，ViT通过将2D图像分割成一系列非重叠的**图像块（patches）**，将每个图像块视为一个token序列.



8.7.1 分词

分词：每个图像块通过线性层或卷积层投影成一维的token嵌入，然后与位置编码相加后输入到Transformer层.



Tokenization of ViT is formulated as $f: \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{D}$, which converts each image patch to a vector (i.e. patch embedding)

In practice, we use convolution layers for linear projection

如果用 CNN 来嵌入 Patch，再用 Tranformer 处理，相当于把CNN 展平后的一维向量作为输入，后面的多层感知机换成 Transformer. 这结合了两个模型的优点：

- CNN 的局部感受野（卷积核）和权值共享天然适合捕捉图像的局部、空间性的特征（如边缘、形状、纹理）等.
- Transformer 的自注意力机制实现全局建模和整合信息，能够捕捉特征图上任意两个局部特征 Token 之间的远程依赖关系.
- Transformer 如果把每个 RGB 像素（映射到嵌入空间）看作一个 token，那么对于高分辨率图像，序列长度呈 $O(N^2)$ 增长（N 是图像的像素宽度/高度）. 而图像中相邻的像素往往高度相似，模型会浪费大量计算资源处理微小的差异. 同时，图像中，“眼睛”、“嘴巴”等语义存在于一个局部区域整体，而不是单个像素点，单个像素携带的语义信息太少.
- CNN 和 Transformer 的结合是互补的：CNN 解决了 Transformer 在处理高分辨率图像时 $O(N^2)$ 的效率问题，而 Transformer 弥补了 CNN 难以捕捉长距离依赖的缺点

例：For an input RGB image of size 224×224 with patch size of 16×16. The hidden size of Transformer layer is 768. How should we configure the linear projection layer for patch embedding?

Use linear layers：

The number of tokens: $\frac{224 \times 224}{16 \times 16} = 196$

The input channel of the linear layer: $16 \times 16 \times 3 = 768$

> 注意 RGB 自带三维.

The output channel of the linear layer is 768

Use convolution layers, No overlapping among patches

目标：让一个卷积操作实现 “提取一个 16×16 像素块并将其投影到 $d_{model} = 768$” 的功能.

The number of tokens is 196

The input channel of the convolution layer is 3

The kernel size of the convolution layer is 16

The stride of the convolution layer is 16

> The stride should be 16 for non-overlapping patch embedding

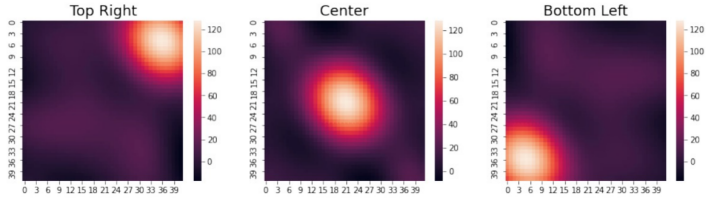
The output channel: 768

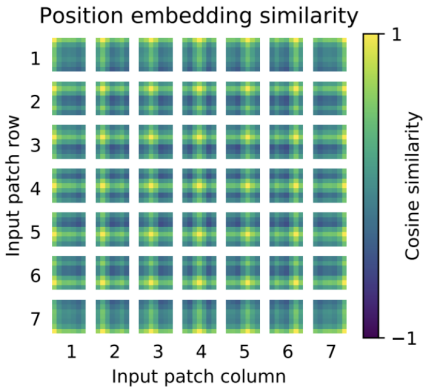
把每个patch看作一个token，CNN 对完整图像进行滑动卷积，每次滑动和卷积就生成一个token的某个维度的值，然后用768个卷积核的结果作为一个token的嵌入向量.

因此 stride 要为 patch 的宽度，保证每次都对新的 patch 进行卷积（不重合）.

8.7.2 位置编码

Like positional embeddings in NLP, the positional embeddings in ViT ensure that spatially closer tokens have higher dot-product similarities.





Similarity maps of pixels at different locations

8.8 Scaling Law

The impressive scaling laws of Transformers motivate the development of large models.

Given large model size and vast amounts of data, Transformer shows incredible performance and generalization

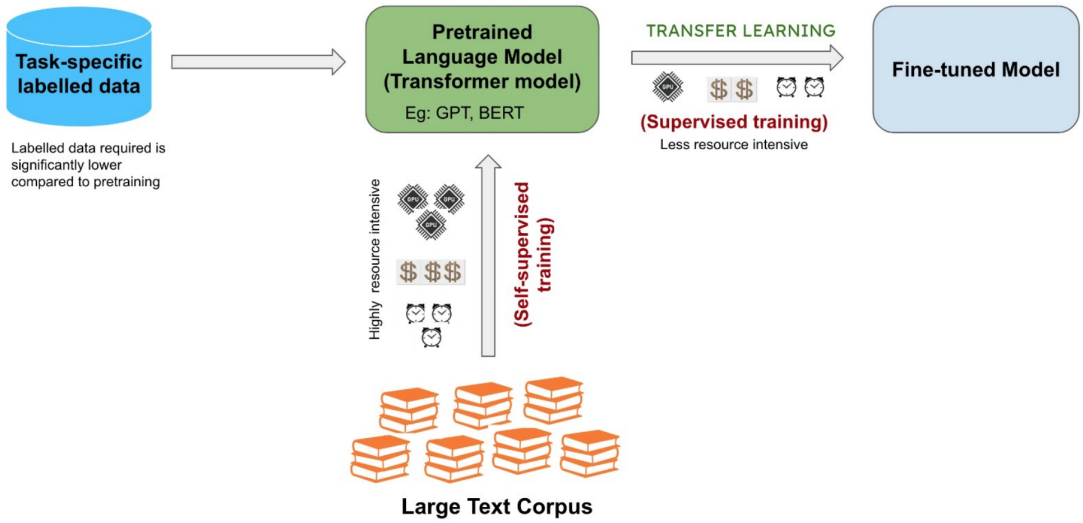
These large-scale pre-trained models are called foundation model

8.9 Transformer 变体

> Variants of Transformers

The pretraining and finetuning paradigm.

- Pretrain a foundation model on large unlabeled data
- Finetune with small labeled data on a specific downstream task



8.9.1 基础模型

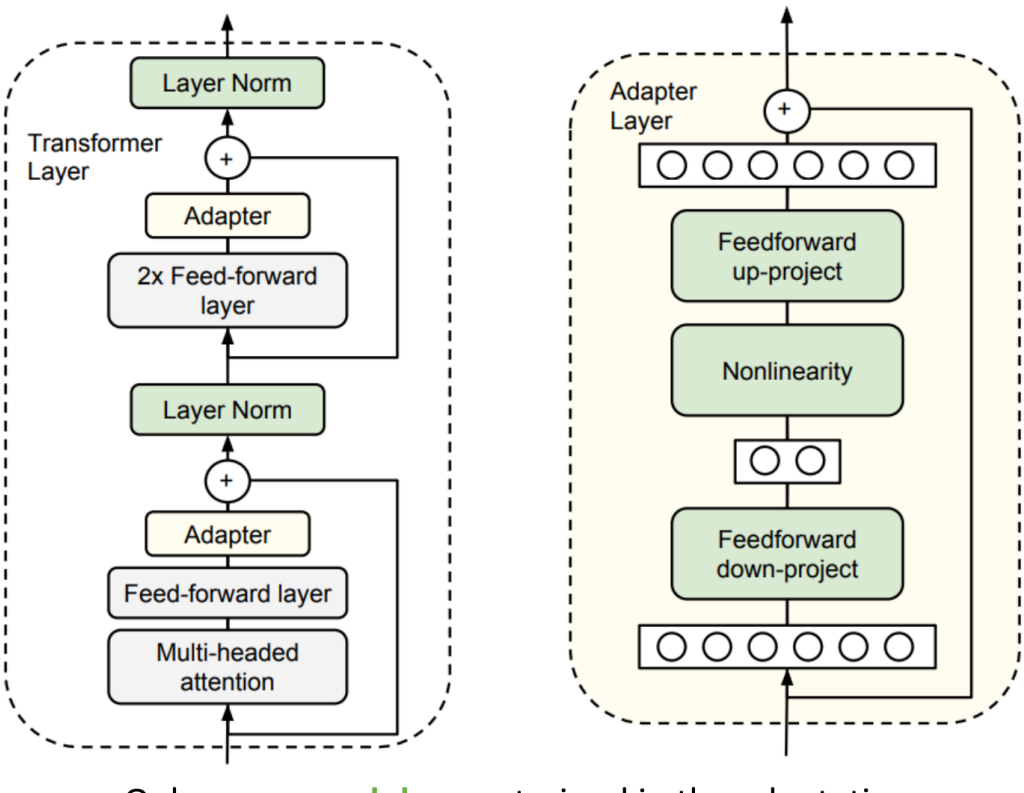
> Foundation Model

Transformer的缩放定律（Scaling Law）表明，大规模模型和海量数据能够带来出色的性能和泛化能力

8.9.2 参数高效微调

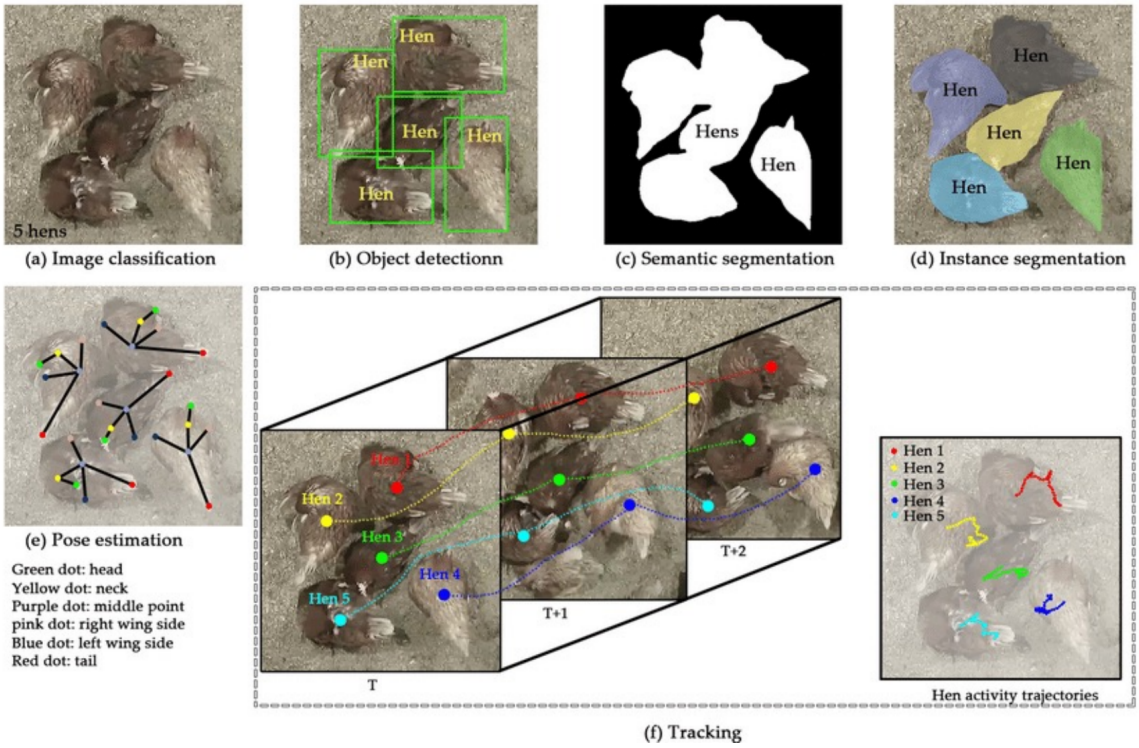
> Parameter-efficient fine-tuning: Adapter

针对大型模型微调成本高、微调数据量不足（可能过拟合）等问题，**Adapter** 等方法被提出，通过在 Transformer 块中插入少量参数（通常少于原 Transformer 的 2%）的新层并仅训练这些层，实现了高效的微调，性能可与训练所有参数相媲美



Only **green modules** are trained in the adaptation

Large pretrained ViTs can be transferred to various downstream tasks through fine-tuning.



8.9.3 密集预测：线性注意力

ViT finds it challenging to handle dense prediction tasks such as segmentation, object detection, depth estimation due to the high computation cost of self-attention.

密集预测：模型要具备像素级的理解能力，例如预测图像中每个像素属于哪个预定义类别，或者预测每个像素到相机的距离（深度估计）.

ViT 通过 Patch Embedding 将图像压缩成短序列，丢失了像素级别的细节. 为了保留细节，需要把 patch 取得很小，这样，

- 序列长度就会随图像大小呈 $O(N)$ 增长
- 序列长度随图像边长呈 $O(N^2)$ 增长
- 自注意力随序列长度呈 $O(N^2)$ 增长
- 自注意力随图像大小呈 $O(N^2)$ 增长
- 自注意力随图像边长呈 $O(N^4)$ 增长

The computational cost of a global multi-head self-attention module with hidden channel C of an image of $h \times w$ patches is $4hwC^2 + 2(hw)^2C$

- $4hwC^2$：线性投影成本，包括 $QW^Q, KW^K, VW^V, \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$ 的计算.
- $2(hw)^2C$：注意力成本，包括注意力分数 QK^T 和加权求和 $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

这里 C 即嵌入向量的维度 d_{model}，hw 即序列长度 N

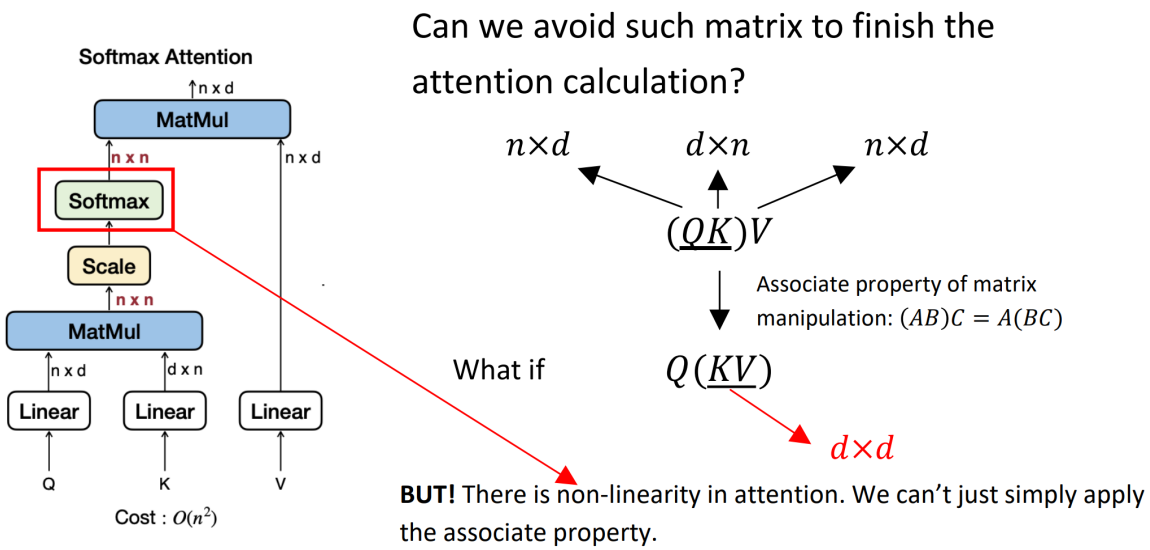
增长过快，需要更高效地处理长序列，才能在密集预测任务中表现良好.

On one hand, patch encoding reduces the resolution of the image with a scale of patch size. The information of original resolution is hard to recover for pixel-wise prediction.

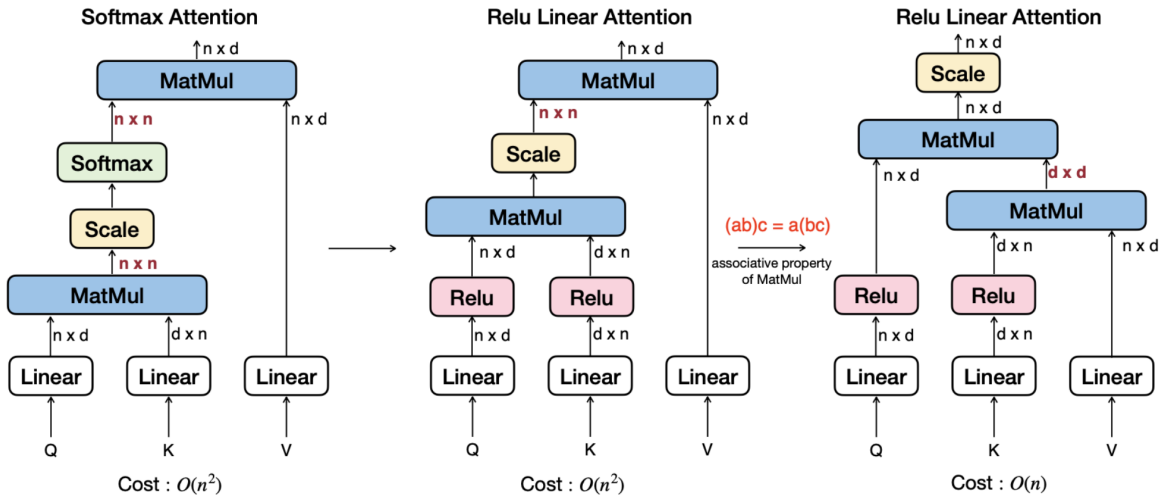
On the other hand, Its intractable for ViT on high-resolution images due to its quadratic computational complexity to image size

引入线性注意力 Linear Attention (EfficientViT)

The quadratic complexity of attention comes from the maintenance and matrix manipulation of a $n \times n$ matrix



Then we can apply the associate property, and get linear attention with complexity of $O(n)$



However, linear attention cannot produce sharp distributions, which may sacrifice performance.

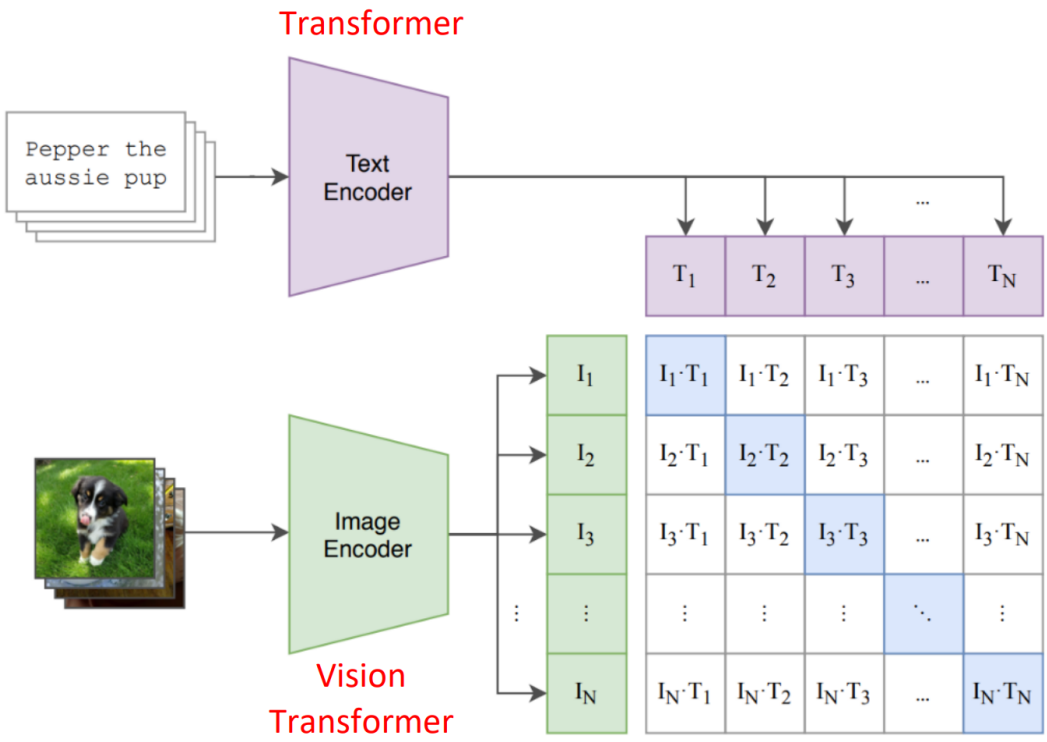
Thus, it is good at capturing global context information but bad at capturing local information

8.9.4 多模态 Transformer：CLIP

As a universal architecture for both NLP and CV, transformer acts as a bridge connecting multiple modalities for collaboration.

CLIP (Contrastive Language-Image Pre-Training) models have two transformer branches: a vision branch and a text branch

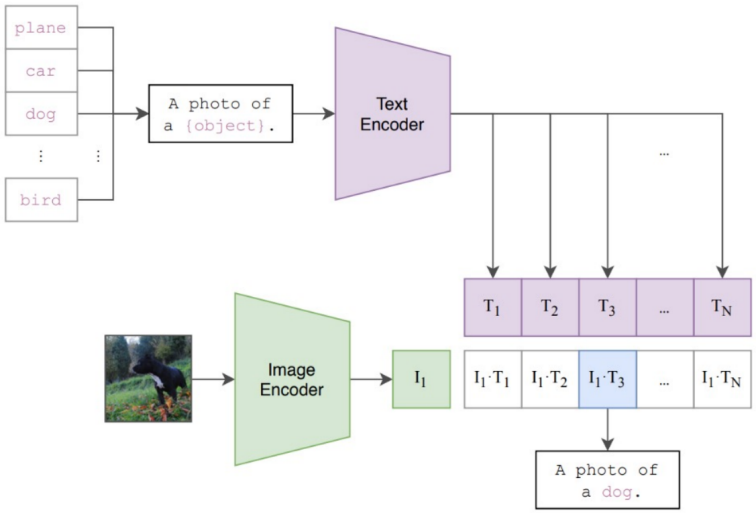
The two branches encode the an image/sentence into a vector



CLIP is trained with text supervision by contrastive learning. The training dataset consists of image-text pairs, with each image accompanied by a descriptive text. The training brings the encoded image embedding closer to its paired text embedding, and pushing it away from other text embeddings (This can be achieved with the cross-entropy loss)

- CLIP enables zero-shot classification without any finetuning
  - Prepare a text description for each class
  - Do classification by matching the image to be classified with the text of all classes

Inference:



Lecture 9 非启发式搜索

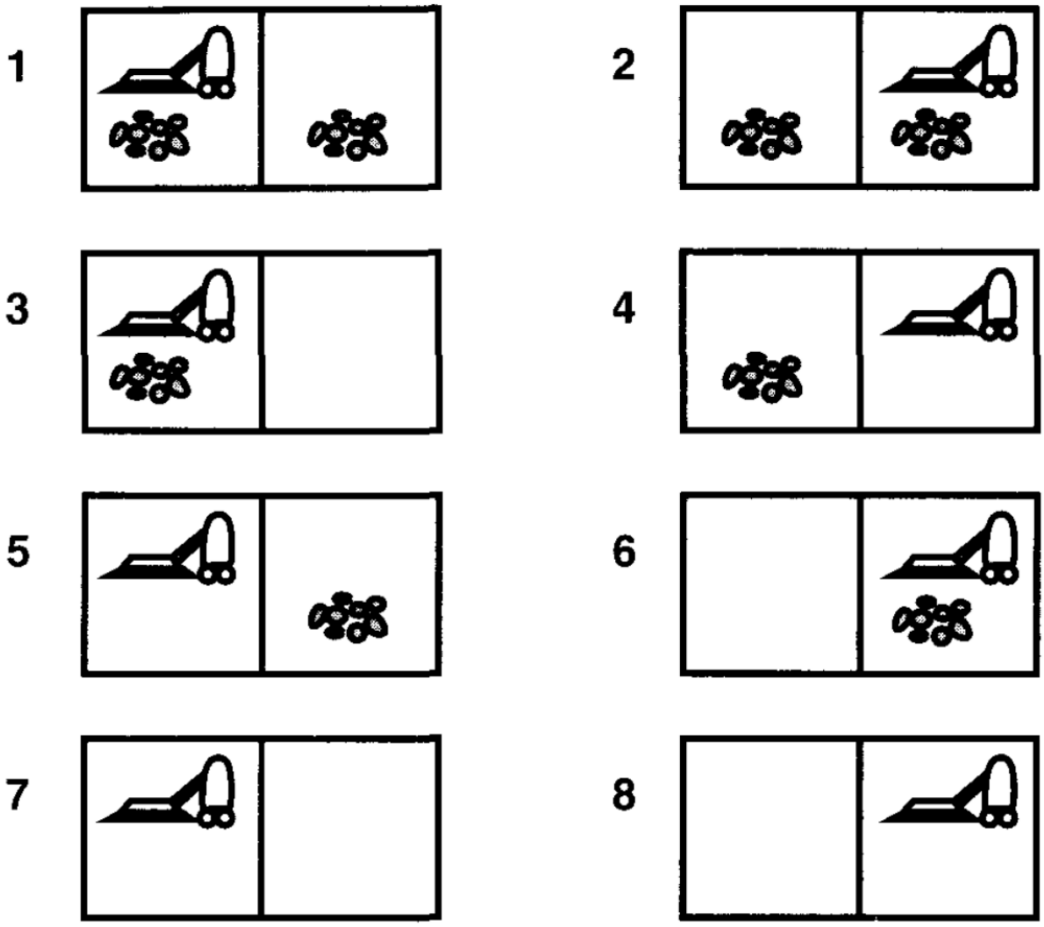
> Uninformed Search

9.1 搜索算法

搜索算法：解决搜索问题的一种算法，它旨在从数据结构中检索信息，或在问题的搜索空间中进行计算.

应用：最短路径，搜索引擎，视频游戏.

例：真空吸尘器问题



8 个可能状态，3 个可能动作（向左、向右、吸尘）

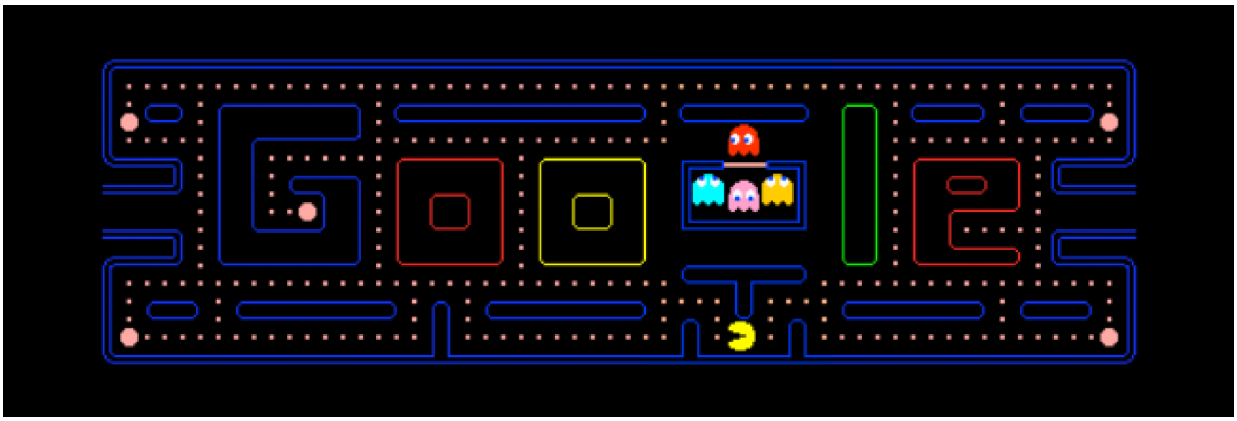
问题类型

- Single-state Problem：智能体拥有完整的世界状态知识和完整的动作知识.
- Multiple-state Problem：智能体状态知识不完整（如，没有传感器）
- 偶然问题（Contigency Problem）：动作的效果不确定.
- 探索问题（Exploration Problem）：状态空间和动作效果均未知.

9.2 搜索问题

> Search Problems

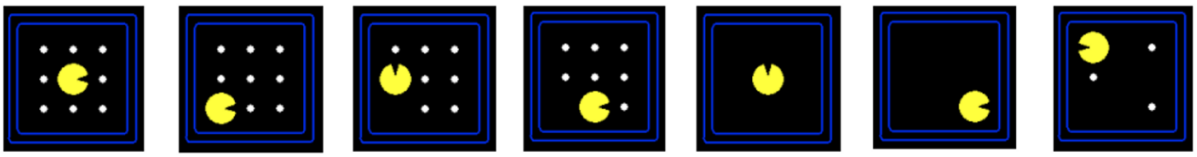
以经典游戏吃豆人为例：



游戏目标：eat all yellow dots

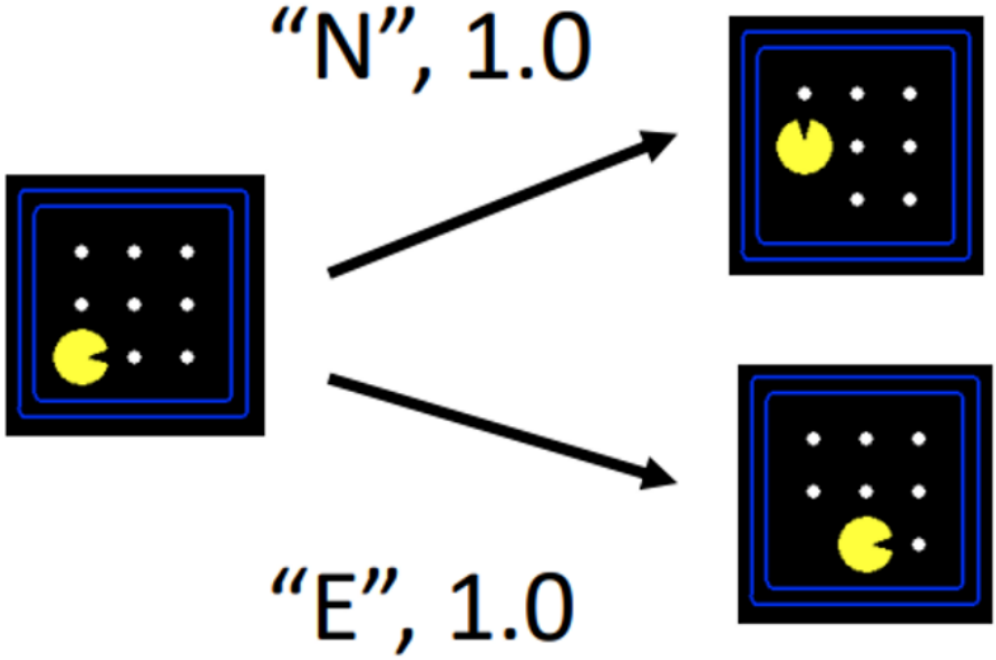
一个搜索问题由以下要素构成：

- 状态空间（State space）：从初始状态可到达的所有状态的集合.



- 操作集合（Operator set）：the set of possible actions available

- 后继函数（Successor function）：给定一个状态 x，返回通过单个动作可到达的状态集合.



- 起始状态（Start state）和目标测试（Goal test）：用于确定一个状态是否为目标状态.

> Start state (initial state): indicate which state that the agent starts in.
>
> State space: the set of all states reachable from the initial state by any sequence of actions.

> Path: a path in the state space is simply any sequence of actions leading from one state to another.
>
> The goal test: the agent can apply the goal test to a single state to determine if it is a goal state. For example, does Pac-Man already eat all dots?
>
> Sometimes, there is an explicit set of possible goal states, and the test simply checks to see if the agent have reached one of them.
>
> Sometimes, the goal is specified by an abstract property rather than an explicitly itemized set of states. For example, in chess, the goal is to reach a state called “checkmate”（将死）

- 路径成本函数（Path cost function）：路径上单个动作成本的总和. 成本可以是时间、距离等，取决于实际问题.

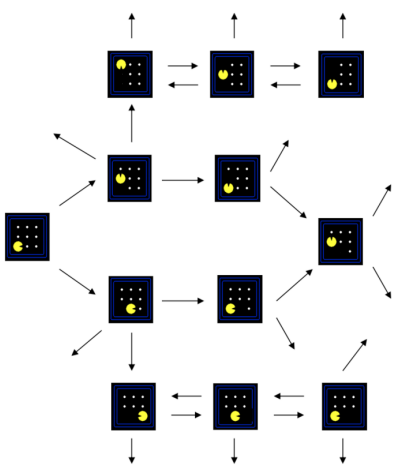
These information will be the input to the search algorithms. The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test

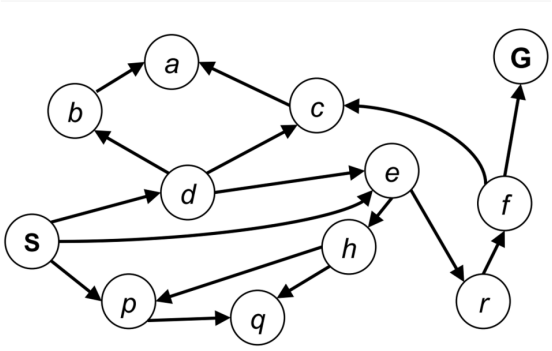
9.3 状态空间图

> 类似自动机的状态转移图.

State space graph: mathematical representation of a search problem

- Nodes are instances of states
- Arcs represent successors (action results)
- The goal test is a (set of) goal node(s)
- Each state occurs only once

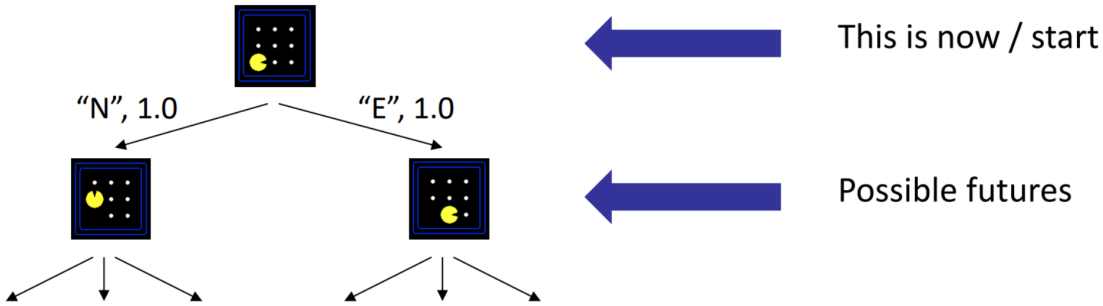




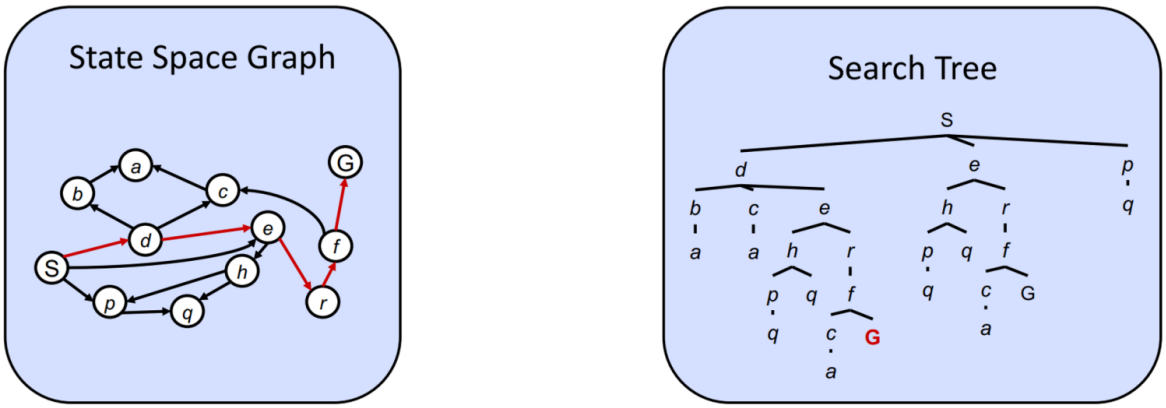
9.4 搜索树

搜索问题可以通过搜索树来寻找解决方案，树的根节点对应于初始状态，搜索算法在每一步选择一个叶节点进行扩展.

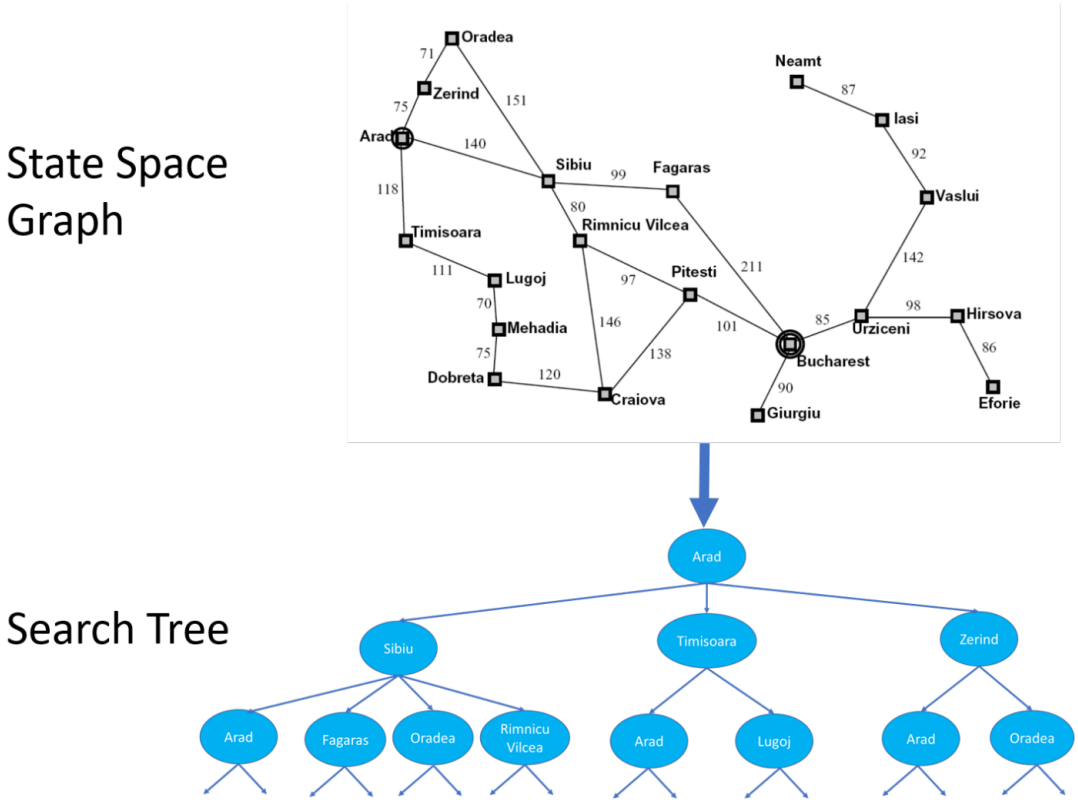
> 搜索树描述动态的建树过程，而不是一开始就完整的树.



搜索树的每个节点，对应一条状态空间图中 S 到该节点的路径.



从状态空间图也可以构建搜索树.



一般搜索树的关键概念：

- Fringe（边缘节点）: the collection of nodes that are waiting to be expanded
- Expansion: pick one out of fringe and expand it
- Search strategy: determine the order in which nodes are expanded. In other words, it is a function that selects the next node to be expanded from the fringe.

搜索树建树伪代码：

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candicates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

The output of a problem-solving algorithm is either failure or a solution. (Note that some algorithms might get stuck in an infinite loop and never return an output.)

9.5 搜索策略

决定搜索树建树顺序的策略. 评估搜索策略有四项标准：

完备性（Completeness）： 是否保证在有解时找到解

最优性（Optimality）： 是否找到最低路径成本的最优解

时间复杂度（Time complexity）： 找到解所需的时间

空间复杂度（Space complexity）： 搜索所需的内存

搜索策略分为 Uninformed search 和 informed search.

- Uninformed search：除了上述定义，对不同状态没有额外信息. 只能产生后继（决定拓展哪个节点）以及检查 goal state.
- Informed search：Strategies that know whether one non-goal state is “more promising” than another are called informed search or heuristic search strategies.

本课介绍 5 个 Uninformed search：

- 广度优先搜索（breadth-first search）
- 一致代价搜索（uniform cost search）
- 深度优先搜索（depth-first search）
- 深度受限搜索（depth-limited search）
- 迭代加深深度优先搜索（iterative deepening search）

9.6 非启发式搜索

策略	描述	完备性	最优性	时间复杂度	空间复杂度（Space Complexity）
广度优先搜索 (BFS)	先扩展最浅层的节点	是	当所有操作成本相同时是	$O(b^d)$	$O(b^d)$
一致代价搜索 (UCS)	扩展路径成本最低的节点	是（当每一步成本 $\geq$ 正常数时）	是（当每一步成本 $\geq$ 正常数时）	$O(b^{C^*/\epsilon})$	$O(b^{C^*/\epsilon})$
深度优先搜索 (DFS)	总是扩展搜索树中最深层的节点	仅在防止循环时是	否	$O(b^m)$	$O(bm)$
深度受限搜索 (DLS)	设定了预定深度限制 l 的 DFS	如果 $l < d$ 则不完备	如果 $l > d$ 则不最优	$O(b^l)$	$O(b^l)$
迭代加深深度优先搜索 (IDDFS)	从 $l = 0$ 开始，逐步增加深度限制 l，直到找到目标	是（分支因子有限时）	是（路径成本是节点深度的非递减函数时）	$O(b^d)$	$O(bd)$

9.6.1 广度优先搜索

all the nodes at depth d in the search tree are expanded, before the nodes at depth $d+1$

待补充.

9.6.2 一致代价搜索

> Uniform-cost search

注意，一致代价搜索 $\neq$ 贪心.

一致代价搜索每一步执行时，扩展所有未扩展节点中，从起点到自身总路径成本最低的节点.

9.6.3 深度优先搜索

待补充

9.6.4 深度受限搜索

给深度优先搜索一个最大深度的限制.

待补充.

9.6.5 迭代加深深度优先搜索

迭代执行的深度受限搜索. 深度限制从 $0, 1, 2, \ldots$ 逐渐增大，直到找到目标节点.

兼具 DFS 和 BFS 的优点：既能有 DFS 的小空间复杂度，又能有 BFS 的完备性和特定条件下的最优性.

Lecture 10 启发式搜索

非启发式搜索通过系统地扩展新状态来寻找解决方案，但在大多数情况下效率较低 。启发式搜索（Informed search）利用超出问题定义本身的问题特定知识，因此能比非启发式搜索更有效地找到解决方案.

10.1 启发式函数

> Heuristics

$h(n)$ 用于**估计**一个状态 n 距离目标有多近.

> 注意，我们不能精确计算出状态 n 距离目标的成本，如果可以，说明我们找到了一条最短路径，问题直接解决了.
>
> $h(n)$ 只是一种估计，例如用地图中的直线距离来估计状态离目标状态的成本.

如果 $h(n)$ 从不高估到达目标的实际成本 $h^*(n)$，即 $0 \leq h(n) \leq h^*(n)$，则它是可采纳的（Admissible）.

10.2 启发式搜索

10.2.1 贪婪搜索

尝试扩展最接近目标的节点，以求快速找到解决方案.

不完备、不保证最优.

10.2.2 A^* 搜索

评估函数：$f(n)=g(n)+h(n)$

> $g(n)$ 是起点到节点 n 的准确成本.
>
> $h(n)$ 是节点 n 到目标点的估计成本.
>
> A^* 搜索可以看作一致代价搜索和贪婪搜索的结合.

Assignments

Assignment 1

Q1. 线性回归

(a) 易错点：$\overline{x}$, $\overline{y}$ 忘记取平均.

> 考试时最好用卡西欧打表，不要手算平均数.

Appendix

1. Sigmoid 函数

> 见 `3.2.3 Sigmoid 函数`.

$$\sigma(x)=\operatorname{logit}^{-1}(x)=\frac{1}{1+e^{-x}}$$

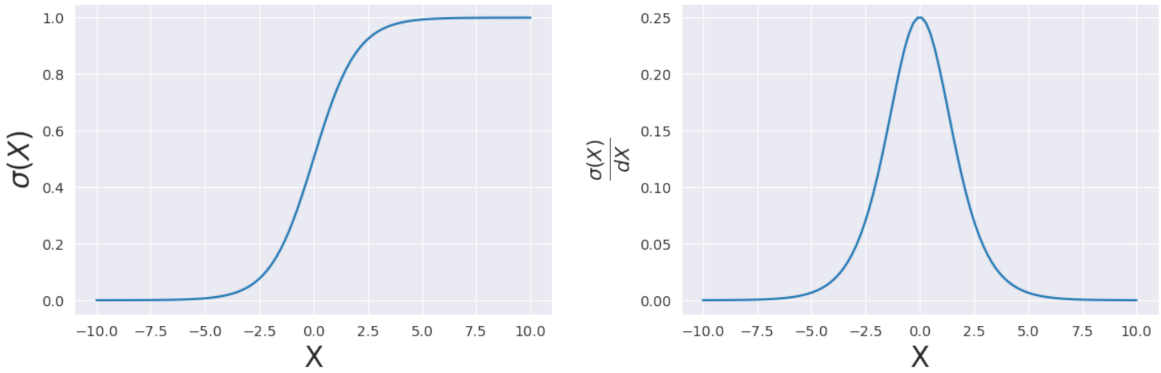
1.1 一阶导

> first derivative

$$
\begin{aligned}
\sigma'(x) &= \frac{d}{dx}\sigma(x) \\
&= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) \\
&= -\frac{1}{(1+e^{-x})^2}\cdot\frac{d}{dx}(1+e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2}
\end{aligned}
$$

注意到 $\sigma(x)=\frac{1}{1+e^{-x}} \Rightarrow 1-\sigma(x)=\frac{e^{-x}}{1+e^{-x}}$，有

$$
\sigma'(x)=\sigma(x)(1-\sigma(x))
$$



2. 梯度最大化方向导数

3. 点到超平面距离公式

Assume a hyperplane $\mathbf{w}^T\mathbf{x}+b=0$ that can correctly classify the samples.

Give an example point $\mathbf{x}_i$，the distance between $\mathbf{x}_i$ and the hyperplane is

$$
\gamma=\frac{|\mathbf{w}^T\mathbf{x}_i+b|}{||\mathbf{w}||}
$$

where $\mathbf{x}_i$ and $\mathbf{w}$ are both d-dimensional vectors.

证明：

首先证明，$\mathbf{w}$ 是超平面 $\mathbf{w}^T\mathbf{x}+b=0$ 的法向量之一.

超平面上任意一个向量 $\mathbf{v}$ 都可以由超平面上两点之差来表示：

$$
\mathbf{v}=\mathbf{x}_1-\mathbf{x}_2
$$

因为 $\mathbf{w}^T\mathbf{x}_1=\mathbf{w}^T\mathbf{x}_2=-b$（点在超平面上），所以

$$
\mathbf{w}^T\mathbf{v}=\mathbf{w}^T(\mathbf{x}_1-\mathbf{x}_2)=0
$$

因为 $\mathbf{w}$ 与超平面 $\mathbf{w}^T\mathbf{x}+b=0$ 上的任意一个向量点积都为零，这意味着 $\mathbf{w}$ 垂直于超平面，因此 $\mathbf{w}$ 是超平面 $\mathbf{w}^T\mathbf{x}+b=0$ 的法向量之一.

点 $\mathbf{x}_i$ 到超平面距离 $\mathbf{w}^T\mathbf{x}+b=0$ 公式：

任取超平面上一点 $\mathbf{x}_p$，满足 $\mathbf{w}^T\mathbf{x}_p+b=0$.

点 $\mathbf{x}_i$ 到超平面距离即差向量 $\mathbf{x}_i-\mathbf{x_p}$ 在法向量 $\mathbf{w}$ 上投影长度：

$$\begin{aligned}\gamma &= \frac{|(\mathbf{x}_i-\mathbf{x_p})^T\mathbf{w}|}{\|\mathbf{w}\|}\\ &= \frac{|\mathbf{x}_i^T\mathbf{w}-\mathbf{x}_p^T\mathbf{w}|}{\|\mathbf{w}\|}\\ &= \frac{|\mathbf{w}^T\mathbf{x}_i-\mathbf{w}^T\mathbf{x}_p|}{\|\mathbf{w}\|}\\ &= \frac{|\mathbf{w}^T\mathbf{x}_i+b|}{\|\mathbf{w}\|}\end{aligned}$$

4. 对偶问题

Recall：SVM 中，原始优化目标为

$$(\mathbf{w}^*,b^*)=\arg\max_{\mathbf{w},b}\left(\min_i\frac{|\mathbf{w}^T\mathbf{x}_i+b|}{\|\mathbf{w}\|}\right)$$

经过归一化约束，优化问题转化为

$$\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1$$

应用拉格朗日乘子法，构造拉格朗日量并令偏导为 0 得到

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\|\mathbf{w}\|^2+\sum_{i=1}^m\alpha_i(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))$$

$$\begin{cases}\frac{\partial L}{\partial \mathbf{w}}=0\\ \frac{\partial L}{\partial b}=0\end{cases}\Rightarrow\begin{cases}\mathbf{w}=\sum\limits_{i=1}^m\alpha_i y_i\mathbf{x}_i\\ 0=\sum\limits_{i=1}^m\alpha_i y_i\end{cases}$$

下面给出将优化问题转换为对偶问题（dual problem）的过程.

将结果代回拉格朗日量，得

$$\begin{aligned}L(\mathbf{w},b,\alpha)&=\frac{1}{2}\|\mathbf{w}\|^2+\sum_{i=1}^m\alpha_i(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))\\ &=\frac{1}{2}\|\mathbf{w}\|^2+\sum_{i=1}^m\alpha_i-\sum_{i=1}^m\alpha_i y_i\mathbf{w}^T\mathbf{x}_i-b\sum_{i=1}^m\alpha_i y_i\\ &=\frac{1}{2}\|\mathbf{w}\|^2+\sum_{i=1}^m\alpha_i-\|\mathbf{w}\|^2-0\\ &=\sum_{i=1}^m\alpha_i-\frac{1}{2}\|\mathbf{w}\|^2\\ &=\sum_{i=1}^m\alpha_i-\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m\alpha_i\alpha_j y_i y_j\mathbf{x}_i^T\mathbf{x}_j\end{aligned}$$

带约束的原始优化问题可以转为无约束的优化问题：

$$\begin{aligned}&\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t. } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b)\le 0\\ \Leftrightarrow &\min_{\mathbf{w},b}\max_{\alpha\ge\mathbf{0}}L(\mathbf{w},b,\alpha)\end{aligned}$$

> $\alpha\ge\mathbf{0}$ 是向量不等式，意思是向量 α 的每个分量都 ≥ 0.
>
> 无约束不太准确，准确地说是将对原始变量 $\mathbf{w}$ 和 b 的复杂约束，转换为对拉格朗日乘子 α 的简单约束上.
>
> 解释：拉格朗日乘子是用来惩罚某些点对约束的违反的. 如果某个点违反约束，即 $1-y_i(\mathbf{w}^T\mathbf{x}_i+b)>0$，那么 $\max$ 会直接让这个违反约束的点的拉格朗日乘子取 ∞，则整个结果就是 ∞，显然不是最优解. 因此，最优解要让每个点都在约束之内，即 $1-y_i(\mathbf{w}^T\mathbf{x}_i+b)\le 0$. 其中，对于 $1-y_i(\mathbf{w}^T\mathbf{x}_i+b)<0$ 的点，$\max$ 会让这些点的拉格朗日乘子取 0. 对于 $1-y_i(\mathbf{w}^T\mathbf{x}_i+b)=0$ 的点（支持向量），$\max$ 不会影响 α 的取值. 综上所述，原始带复杂约束的优化问题可以转为带简单约束的优化问题.

优化问题：

$$\min_{\mathbf{w},b}\max_{\alpha\ge\mathbf{0}}L(\mathbf{w},b,\alpha)$$

对偶问题：

$$\max_{\alpha\ge 0}\min_{\mathbf{w},b}L(\mathbf{w},b,\alpha)$$

> 待补充.

弱对偶性定律：对于任何优化问题（无论凸性），原始问题的最优值总是大于等于对偶问题的最优值.

> 这个关系称为弱对偶性. 对偶问题的最优解是原始问题最优解的一个下限估计.
>
> 待补充.

强对偶性：强对偶性是指原始问题的最优值等于对偶问题的最优值. 对于一般的优化问题，强对偶性并不总是成立. 但是，对于凸优化问题，在满足某些条件（如 Slater's Condition）时，强对偶性成立. 由于 SVM 是一个凸优化问题，并且对于线性可分的数据集，我们总是可以找到一个严格分离的超平面. 这意味着 Slater's Condition 总是成立. 因此，对于硬间隔 SVM，强对偶性成立.

> 待补充.

因此，结合偏导为 0 时的拉格朗日量，最终对偶问题为：

$$\begin{aligned}&\max_{\alpha\ge 0}\min_{\mathbf{w},b}L(\mathbf{w},b,\alpha)\\ =&\max_{\alpha}\sum_{i=1}^m\alpha_i-\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m\alpha_i\alpha_j y_i y_j\mathbf{x}_i^T\mathbf{x}_j\\ &\text{s.t. }\sum_{i=1}^m\alpha_i y_i=0,\alpha_i\ge 0,i=1,2,\dots,m\end{aligned}$$

5. 矩阵运算 & 矩阵微积分

5.1 Hadamard 积

> $\odot$

Hadamard 积是一种特殊的矩阵/向量乘法，要求参与运算的两个矩阵/向量**维度完全相同**. 运算时，将对应位置的元素相乘得到与原矩阵/向量维度相同的新矩阵/新向量.

对于两个 $m \times n$ 矩阵 A 和 B，$C = A \odot B \Rightarrow C_{ik} = A_{ik}B_{ik}$

5.2 求导

5.1.1 标量对向量

① 一阶导：分母布局

设 $\mathbf{x}$ 是一个 n 维列向量，y 是一个标量.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad y = f(\mathbf{x})$$

神经网络中，标量对向量求导通常使用分母布局，即

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

例：线性回归中，求正规方程

> 见 `2.5.4`
>
> 把偏置项 θ_0 吸收到权重向量，并给每一个输入样本添加一个常数项偏置 $x_0 = 1$.

$$\begin{aligned} J(\Theta) &= \|\hat{f}_\Theta(\mathbf{X}) - Y\|_2^2 \\ &= (\mathbf{X}\Theta - Y)^T(\mathbf{X}\Theta - Y) \\ &= \Theta^T\mathbf{X}^T\mathbf{X}\Theta - Y^T\mathbf{X}\Theta - \Theta^T\mathbf{X}^TY + Y^TY \end{aligned}$$

$$\begin{aligned} \text{Let} \quad \frac{\partial J(\Theta)}{\partial \Theta} &= 2\mathbf{X}^T\mathbf{X}\Theta - \mathbf{X}^TY - \mathbf{X}^TY \\ &= 2\mathbf{X}^T(\mathbf{X}\Theta - Y) \\ &= \mathbf{0} \\ \Rightarrow \hat{\Theta} = \Theta^* &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^TY \end{aligned}$$

这里就用到了标量对向量求导，它的结果排布方式和分母一致（即一个列向量）.

② 二阶导：Hessian 矩阵

标量对向量求二阶导较特殊，通常采用 Hessian 矩阵来书写. Hessian 矩阵基于分母布局的对 Θ 一阶导，用分子布局对 Θ^T 再求一次导得到（混合布局）.

分子布局：列方向同分子分布，行方向同分母分布.

$$
\begin{aligned}
\mathbf{H} &= \frac{\partial^2 J(\Theta)}{\partial \Theta \partial \Theta^T} \\
&= \frac{\partial}{\partial \Theta^T} \begin{pmatrix} \frac{\partial^2 J(\Theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial^2 J(\Theta)}{\partial \theta_n} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial^2 J(\Theta)}{\partial \theta_1 \partial \theta_1} & \cdots & \frac{\partial^2 J(\Theta)}{\partial \theta_1 \partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\Theta)}{\partial \theta_n \partial \theta_1} & \cdots & \frac{\partial^2 J(\Theta)}{\partial \theta_n \partial \theta_n} \end{pmatrix}
\end{aligned}
$$

其中，$H_{ij} = \frac{\partial^2 J}{\partial \theta_i \partial \theta_j}$

这么写是因为：

- 通过分析 Hessian 矩阵的性质可以判定一阶导为 $\mathbf{0}$ 时是否存在唯一最优解.
- 二阶导是一阶导再求一次导，而一阶导按照分母布局结果是向量，则再求一次导实际上是向量对向量求导，转到 `5.1.3` 采用分子布局写雅各比矩阵.

注意：黑塞矩阵实际上就是一阶导的雅各比矩阵.

5.1.2 标量对矩阵

5.1.3 向量对向量：分子布局

向量 $\mathbf{y}$ 对向量 $\mathbf{x}$ 的导数 $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ 被定义为雅可比矩阵（Jacobian Matrix）.

设 $\mathbf{x}$ 是一个 n 维列向量，$\mathbf{y}$ 是一个 m 维列向量.

$$
\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}
$$

神经网络中，雅可比矩阵通常使用分子布局（Numerator Layout），即一个 $m \times n$ 矩阵，其 (i, j) 元素定义为分子 $\mathbf{y}$ 的第 i 个分量对分母 $\mathbf{x}$ 的第 j 个分量的偏导数：

$$
\mathbf{J} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}
$$

> 分子布局的意思是，结果在列方向上的顺序和分子一致.
>
> 这里按照 `5.1.1` 的写法，应该是 $\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T}$. 但更重要的是知道雅各比矩阵的结果是列方向按照分子的索引布局，行方向按照分母的索引布局.

例（默认分子布局）：

① $\mathbf{z}^{(j+1)}=\boldsymbol{\Theta}^{(j)^T}\mathbf{a}^{(j)} \Rightarrow \frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{a}^{(j)}}=\boldsymbol{\Theta}^{(j)^T}$

证明：

$$\mathbf{z}^{(j+1)}=\boldsymbol{\Theta}^{(j)^T}\mathbf{a}^{(j)}=\begin{bmatrix}\Theta_{11}^{(j)} & \Theta_{12}^{(j)} & \cdots & \Theta_{1s_{j+1}}^{(j)}\\ \Theta_{21}^{(j)} & \Theta_{22}^{(j)} & \cdots & \Theta_{2s_{j+1}}^{(j)}\\ \vdots & \vdots & \ddots & \vdots\\ \Theta_{s_j1}^{(j)} & \Theta_{s_j2}^{(j)} & \cdots & \Theta_{s_js_{j+1}}^{(j)}\end{bmatrix}^T\begin{bmatrix}a_1^{(j)}\\ a_2^{(j)}\\ \vdots\\ a_{s_j}^{(j)}\end{bmatrix}$$

使用分子布局的雅可比矩阵为

$$\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{a}^{(j)}}=\begin{bmatrix}\frac{\partial z_1^{(j+1)}}{\partial a_1^{(j)}} & \frac{\partial z_1^{(j+1)}}{\partial a_2^{(j)}} & \cdots & \frac{\partial z_1^{(j+1)}}{\partial a_{s_j}^{(j)}}\\ \frac{\partial z_2^{(j+1)}}{\partial a_1^{(j)}} & \frac{\partial z_2^{(j+1)}}{\partial a_2^{(j)}} & \cdots & \frac{\partial z_2^{(j+1)}}{\partial a_{s_j}^{(j)}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial a_1^{(j)}} & \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial a_2^{(j)}} & \cdots & \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial a_{s_j}^{(j)}}\end{bmatrix}=\boldsymbol{\Theta}^{(j)^T}$$

② $\mathbf{a}^{(j)}=g^{(j)}\left(\mathbf{z}^{(j)}\right) \Rightarrow \frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}}=\mathbf{D}^{(j)}=\operatorname{diag}\left(\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)\right)$

> 见 Appendix 6. 反向传播

证明：分子布局时，列方向同分子分布，行方向同分母分布，有

$$\frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}}=\begin{bmatrix}\frac{\partial a_1^{(j)}}{\partial z_1^{(j)}} & 0 & \cdots & 0\\ 0 & \frac{\partial a_2^{(j)}}{\partial z_2^{(j)}} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \frac{\partial a_{s_j}^{(j)}}{\partial z_{s_j}^{(j)}}\end{bmatrix}$$

其中，每个对角项满足

$$a_i^{(j)}=g^{(j)}\left(z_i^{(j)}\right) \Rightarrow \frac{\partial a_i^{(j)}}{\partial z_i^{(j)}}=\left(g^{(j)}\right)'\left(z_i^{(j)}\right)$$

对角项抽出来排成一列即

$$\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)$$

5.1.4 矩阵求导

① 含矩阵的标量对向量求导

i) 记 $\Theta \in \mathbb{R}^{n\times 1}$，$A \in \mathbb{R}^{n\times n}$，则 $\frac{\partial}{\partial \Theta}\Theta^T A\Theta=(A+A^T)\Theta$

推导：

$$\begin{aligned}\Theta^T A\Theta &= \begin{pmatrix}\theta_1 & \dots & \theta_n\end{pmatrix}\begin{pmatrix}A_{11} & \dots & A_{1n}\\ \vdots & \ddots & \vdots\\ A_{n1} & \dots & A_{nn}\end{pmatrix}\begin{pmatrix}\theta_1\\ \vdots\\ \theta_n\end{pmatrix}\\ &= \theta_1(A_{11}\theta_1+\cdots+A_{1n}\theta_n)+\cdots+\theta_n(A_{n1}\theta_1+\cdots+A_{nn}\theta_n)\end{aligned}$$

标量对向量求一阶导，采用分母布局：

$$\begin{aligned}\frac{\partial}{\partial\Theta}\Theta^T A\Theta &= \begin{pmatrix}\frac{\partial}{\partial\theta_1}\theta_1(A_{11}\theta_1+\cdots+A_{1n}\theta_n)+\cdots+\theta_n(A_{n1}\theta_1+\cdots+A_{nn}\theta_n)\\ \vdots\\ \frac{\partial}{\partial\theta_n}\theta_1(A_{11}\theta_1+\cdots+A_{1n}\theta_n)+\cdots+\theta_n(A_{n1}\theta_1+\cdots+A_{nn}\theta_n)\end{pmatrix}\\ &= \begin{pmatrix}(2A_{11}\theta_1+A_{12}\theta_2+\cdots+A_{1n}\theta_n)+\theta_2A_{21}+\cdots+\theta_nA_{n1}\\ \vdots\\ (2A_{nn}\theta_n+A_{n1}\theta_1+\cdots+A_{(n-1)n}\theta_{n-1})+\theta_1A_{1n}+\cdots+\theta_{n-1}A_{n(n-1)}\end{pmatrix}\\ &= (A+A^T)\Theta\end{aligned}$$

若 $A=A^T$（如线性回归中，$A=X^TX$），则可以写作 $2A\Theta$.

ii) 记 $\Theta\in\mathbb{R}^{n\times1}$，$A\in\mathbb{R}^{1\times n}$，则 $\frac{\partial}{\partial\Theta}A\Theta=A^T$

推导：

$$A\Theta=\begin{pmatrix}A_1 & \dots & A_n\end{pmatrix}\begin{pmatrix}\theta_1\\ \vdots\\ \theta_n\end{pmatrix}=A_1\theta_1+\cdots+A_n\theta_n$$

采用分母布局，易得 $\frac{\partial}{\partial\Theta}A\Theta=A^T$

iii) 记 $\Theta\in\mathbb{R}^{n\times1}$，$A\in\mathbb{R}^{n\times1}$，则 $\frac{\partial}{\partial\Theta}\Theta^TA=A$

推导：

$$\Theta^TA=\begin{pmatrix}\theta_1 & \dots & \theta_n\end{pmatrix}\begin{pmatrix}A_1\\ \vdots\\ A_n\end{pmatrix}=A_1\theta_1+\cdots+A_n\theta_n$$

采用分母布局，易得 $\frac{\partial}{\partial\Theta}\Theta^TA=A$

② 含矩阵的向量对向量求导

ii) 记 $\Theta\in\mathbb{R}^{n\times1}$，$A\in\mathbb{R}^{n\times n}$，则 $\frac{\partial}{\partial\Theta}A\Theta=A$

推导：

$$\begin{aligned}A\Theta &= \begin{pmatrix}A_{11} & \dots & A_{1n}\\ \vdots & \ddots & \vdots\\ A_{1n} & \cdots & A_{nn}\end{pmatrix}\begin{pmatrix}\theta_1\\ \vdots\\ \theta_n\end{pmatrix}\\ &= \begin{pmatrix}A_{11}\theta_1+\cdots+A_{1n}\theta_n\\ \vdots\\ A_{1n}\theta_1+\cdots+A_{nn}\theta_n\end{pmatrix}\end{aligned}$$

向量对向量求导采用分子布局（雅各比矩阵），易得 $\frac{\partial}{\partial\Theta}A\Theta=A$

应用：线性回归中，一阶导求得

$$\frac{\partial J(\Theta)}{\partial\Theta}=2\mathbf{X}^T\mathbf{X}\Theta-\mathbf{X}^TY-\mathbf{X}^TY$$

易得二阶导为 $2\mathbf{X}^T\mathbf{X}$.

③

5.3 链式法则

情况很多，这里用本课中出现的具体例子分析：

① $\delta^{(j)}=\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}}=\left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j+1)}}=\left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T\delta^{(j+1)}$

由 `5.1.1`，标量对向量求导采用分母布局：

$$\delta^{(j)}=\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}}=\begin{bmatrix}\frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}}\\ \frac{\partial J(\mathbf{\Theta})}{\partial z_2^{(j)}}\\ \vdots\\ \frac{\partial J(\mathbf{\Theta})}{\partial z_{s_j}^{(j)}}\end{bmatrix},\delta^{(j+1)}=\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j+1)}}=\begin{bmatrix}\frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j+1)}}\\ \frac{\partial J(\mathbf{\Theta})}{\partial z_2^{(j+1)}}\\ \vdots\\ \frac{\partial J(\mathbf{\Theta})}{\partial z_{s_{j+1}}^{(j+1)}}\end{bmatrix}$$

由 `5.1.3`，向量对向量求导采用分子布局：

$$\begin{aligned}\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}&=\begin{bmatrix}\frac{\partial z_1^{(j+1)}}{\partial z_1^{(j)}} & \frac{\partial z_1^{(j+1)}}{\partial z_2^{(j)}} & \cdots & \frac{\partial z_1^{(j+1)}}{\partial z_{s_j}^{(j)}}\\ \frac{\partial z_2^{(j+1)}}{\partial z_1^{(j)}} & \frac{\partial z_2^{(j+1)}}{\partial z_2^{(j)}} & \cdots & \frac{\partial z_2^{(j+1)}}{\partial z_{s_j}^{(j)}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial z_1^{(j)}} & \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial z_2^{(j)}} & \cdots & \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial z_{s_j}^{(j)}}\end{bmatrix}\\ &=\begin{bmatrix}\left(\frac{\partial z_1^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T\\ \left(\frac{\partial z_2^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T\\ \vdots\\ \left(\frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T\end{bmatrix}\end{aligned}$$

注意，标量对向量求导通常用分母布局，而向量对向量求导常用分子布局. 把向量对向量求导看作分子向量的每个分量（标量）分别对分母向量求导，每个结果转置后排一行.

因此，

$$\left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j+1)}} = \begin{bmatrix} \frac{\partial z_1^{(j+1)}}{\partial \mathbf{z}^{(j)}} & \frac{\partial z_2^{(j+1)}}{\partial \mathbf{z}^{(j)}} & \cdots & \frac{\partial z_{s_{j+1}}^{(j+1)}}{\partial \mathbf{z}^{(j)}} \end{bmatrix} \begin{bmatrix} \frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j+1)}} \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_2^{(j+1)}} \\ \vdots \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_{s_{j+1}}^{(j+1)}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}} \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_2^{(j)}} \\ \vdots \\ \frac{\partial J(\mathbf{\Theta})}{\partial z_{s_j}^{(j)}} \end{bmatrix}$$

$$= \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}}$$

> 转化为已知的标量对标量求导的链式法则.

注意，这里虽然用的是熟悉的标量对标量求导的链式法则，但是属于多元函数的链式法则，复习一下：

以 $\frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}}$ 的第一项 $\frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}}$ 为例，这里以 $\mathbf{z}^{(j+1)}$ 为中间变量，但是不能写作 $\frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}} = \frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j+1)}} \frac{\partial z_1^{(j+1)}}{\partial z_1^{(j)}}$，而是要写作

$$\frac{\partial J(\mathbf{\Theta})}{\partial z_1^{(j)}} = \sum_{i=1}^{s_{j+1}} \frac{\partial J(\mathbf{\Theta})}{\partial z_i^{(j+1)}} \frac{\partial z_i^{(j+1)}}{\partial z_1^{(j)}}$$

6. 反向传播

> 见 `6.2.1* 解析推导`

推导：

$$\delta^{(j)} = \left(\mathbf{\Theta}^{(j)} \delta^{(j+1)}\right) \odot \left(g^{(j)}\right)' \left(\mathbf{z}^{(j)}\right)$$

由定义和链式法则得，

$$\begin{aligned} \delta^{(j)} &= \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}} \\ &= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j+1)}} \\ &= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \delta^{(j+1)} \end{aligned}$$

> 这里链式法则的推导见 `Appendix 5.3`

其中，雅可比矩阵 $\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}$ 可进一步分解：

$$\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}} = \frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{a}^{(j)}} \frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}}$$

其中，第 $j+1$ 层的加权输入由第 j 层的激活输出加权得到：

$$\mathbf{z}^{(j+1)} = {\mathbf{\Theta}^{(j)}}^T \mathbf{a}^{(j)} \Rightarrow \frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{a}^{(j)}} = {\mathbf{\Theta}^{(j)}}^T$$

> 见 `Appendix 5.1.3` 例 ①.

第 j 层的激活输出由第 j 层的加权输入经过激活函数得到：

$$\mathbf{a}^{(j)} = g^{(j)}\left(\mathbf{z}^{(j)}\right)$$

> 这里函数以向量 $\mathbf{z}^{(j)}$ 作为自变量，即 $\mathbf{z}^{(j)}$ 的每个分量传入，然后结果重新排成一列.

可得 $\frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}}$ 是一个对角矩阵，记为

$$\frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}} = \mathbf{D}^{(j)} = \operatorname{diag}\left(\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)\right)$$

$\operatorname{diag}\left(\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)\right)$ 意思是一个 $s_j \times s_j$ 对角矩阵，对角线元素分别是向量 $\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)$ 各分量（共 s_j 个分量）.
$\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)$ 意思是把 $\mathbf{z}^{(j)}$ 各分量传入激活函数的导函数，然后把结果排成一列.

> 见 `Appendix 5.1.3 向量对向量`

因此，

$$\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}} = \frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{a}^{(j)}} \frac{\partial \mathbf{a}^{(j)}}{\partial \mathbf{z}^{(j)}} = {\boldsymbol{\Theta}^{(j)}}^T \mathbf{D}^{(j)}$$

代回误差项公式，

$$\begin{aligned}
\delta^{(j)} &= \frac{\partial J(\boldsymbol{\Theta})}{\partial \mathbf{z}^{(j)}} \\
&= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \frac{\partial J(\boldsymbol{\Theta})}{\partial \mathbf{z}^{(j+1)}} \\
&= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \delta^{(j+1)} \\
&= \left({\boldsymbol{\Theta}^{(j)}}^T \mathbf{D}^{(j)}\right)^T \delta^{(j+1)} \\
&= {\mathbf{D}^{(j)}}^T {\boldsymbol{\Theta}^{(j)}} \delta^{(j+1)} \\
&= \mathbf{D}^{(j)} \boldsymbol{\Theta}^{(j)} \delta^{(j+1)}
\end{aligned}$$

这里，

$\mathbf{D}^{(j)} = \operatorname{diag}\left(\left(g^{(j)}\right)'\left(\mathbf{z}^{(j)}\right)\right)$ 是 $s_j \times s_j$ 的对角矩阵

$\boldsymbol{\Theta}^{(j)} = \begin{bmatrix} \Theta_{11}^{(j)} & \Theta_{12}^{(j)} & \cdots & \Theta_{1s_{j+1}}^{(j)} \\ \Theta_{21}^{(j)} & \Theta_{22}^{(j)} & \cdots & \Theta_{2s_{j+1}}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{s_j1}^{(j)} & \Theta_{s_j2}^{(j)} & \cdots & \Theta_{s_js_{j+1}}^{(j)} \end{bmatrix}$ 是 $s_j \times s_{j+1}$ 的权重矩阵

$\delta^{(j+1)} = \frac{\partial J(\boldsymbol{\Theta})}{\partial \mathbf{z}^{(j+1)}} = \begin{bmatrix} \frac{\partial J(\boldsymbol{\Theta})}{\partial z_1^{(j+1)}} \\ \frac{\partial J(\boldsymbol{\Theta})}{\partial z_2^{(j+1)}} \\ \vdots \\ \frac{\partial J(\boldsymbol{\Theta})}{\partial z_{s_{j+1}}^{(j+1)}} \end{bmatrix}$ 是 $s_{j+1} \times 1$ 的下一层误差项

因为对角矩阵除主对角线以外都是 0，可以利用 `5.1 Hadamard 积` 简化（在实际训练中可以理解为减少内存和计算次数，因为 0 没用）：

$$
\begin{aligned}
\delta^{(j)} &= \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j)}} \\
&= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{z}^{(j+1)}} \\
&= \left(\frac{\partial \mathbf{z}^{(j+1)}}{\partial \mathbf{z}^{(j)}}\right)^T \delta^{(j+1)} \\
&= \left({\mathbf{\Theta}^{(j)}}^T \mathbf{D}^{(j)}\right)^T \delta^{(j+1)} \\
&= {\mathbf{D}^{(j)}}^T {\mathbf{\Theta}^{(j)}}^T \delta^{(j+1)} \\
&= \mathbf{D}^{(j)} \mathbf{\Theta}^{(j)} \delta^{(j+1)} \\
&= \left(\mathbf{\Theta}^{(j)} \delta^{(j+1)}\right) \odot \left(g^{(j)}\right)' \left(\mathbf{z}^{(j)}\right)
\end{aligned}
$$

这里还利用了矩阵乘法满足结合律，后两项 $\mathbf{\Theta}^{(j)}\delta^{(j+1)}$ 看作一个整体. 此时这个整体为 $s_j \times 1$ 的向量，它有实际意义：既然前向传播时，加权输入从第 j 层到第 $j+1$ 层需要经过 ${\mathbf{\Theta}^{(j)}}^T$ 的变换，那么反向传播时，误差项从第 $j+1$ 层反向传播回第 j 层也应当经过 $\mathbf{\Theta}^{(j)}$ 的变换. 加权变换互为转置也是合理的，因为一个从 s_j 转 s_{j+1}，一个从 s_{j+1} 转 s_j.

当然中间还经过激活函数，因此应用链式法则的完整传播路径为

前向传播：

$$
\mathbf{z}^{(j)} \xrightarrow{\text{传入激活函数}} \mathbf{a}^{(j)} = g(\mathbf{z}^{(j)}) \xrightarrow{{\mathbf{\Theta}^{(j)}}^T\text{加权}} \mathbf{z}^{(j+1)} = {\mathbf{\Theta}^{(j)}}^T \mathbf{a}^{(j)}
$$

反向传播：

$$
\delta^{(j+1)} \xrightarrow{\mathbf{\Theta}^{(j)}\text{加权}} \frac{\partial J(\mathbf{\Theta})}{\partial \mathbf{a}^{(j)}} = \mathbf{\Theta}^{(j)}\delta^{(j+1)} \xrightarrow{\text{链式法则反向传入激活函数}} \delta^{(j)}
$$

Project

> AI in Language

Overview

If you've ever had a comment taken down on Reddit and wondered "why?", you're not alone. Each subreddit（子版块）has its own set of guidelines, and trying to understand individual subreddit moderation（版主） can feel like chaos.

In this competition, you'll bring some 'comment sense' to the table and work with real data to build models that predict which rule (if any) a comment may have broken.

Description

Your task is to create a **binary classifier** that predicts **whether a Reddit comment broke a specific rule**. The dataset comes from a large collection of moderated comments, with a range of subreddit norms, tones, and community expectations.

The rules you'll be working with are based on actual subreddit guidelines, but the dataset itself is drawn from older, unlabeled content. A small labeled dev set has been created to help you get started.

This is a chance to explore how machine learning can support real-world content moderation, particularly in communities with unique rules and norms.

Background

Inspired by the work of our colleagues Deepak Kumar, Yousef AbuHashem, and Zakir Durumeric where large language models were deployed to try to guess the reasons that moderators used to remove comments. This work builds upon the work of Eshwar Chandrasekharan and Eric Gilbert which collected a set of millions of moderated comments.

This several-year-old dataset is **unlabeled**. It is accompanied by a list of hypothetical rules—derived from real rules on a variety of subreddits—to help identify potential comment violations.

未标注：这指的是 Eshwar Chandrasekharan 和 Eric Gilbert 几年前收集的**数百万条**被版主删除的原始评论数据. 这些原始数据只有评论内容和删除记录，但**没有**一个明确的字段说明它具体违反了"**哪条**"规则.

由于原始数据没有明确的规则标签，主办方**基于**各种 Reddit 子版块的**真实规则**，人工**制定**或**整理**了一份规则清单（例如："No Adver"、"No legal advice"）.

主办方随后用这份"假设规则"列表去**人工标注**或**自动匹配**了一小部分原始数据，从而创建了您手中的 `train.csv`

"假设"的意思是，这些规则是**为了竞赛目的而创建的**，它们**模仿**了现实中的规则，但它们不是原始数据中的版主亲自使用的标签.

Rules Classification

Participants have access to a small subset of the data, which can be used as a dev resource（即调试数据，2000 条训练数据和 10 条测试数据，和云端的数据无关，这是用来本地调试的）. This information is suitable for use as training data or for few-shot examples. The remainder of the labels will be used, in a 30%:70% to form the public and private test sets.

数据集

The dataset provides instances of comments that may or may not have violated a specific rule on a subreddit.

The training dataset contains only two rules. The test dataset contains *additional* rules that models must be able to generalize to. (The number of unseen rules is not specified as part of the competition.)

在云端评分中，测试集又分为 Public（30%） 和 Private（70%）集. 在比赛过程中，参赛者只能看到自己在 Public 集的得分. 比赛结束时，Kaggle 会用 Private 集重新打分，并生成最终的排行榜. 这样操作的目的是，防止参赛者根据 Public 的表现来调整参数，使得模型过拟合 Public 集.

Files

- train.csv

  - the training dataset

    - `body` - the text of the comment
    - `rule` - the rule the comment is judged to be in violation of
    - `subreddit` - the forum the comment was made in
    - `positive_example_{1,2}` - examples of comments that violate the rule
    - `negative_example_{1,2}` - examples of comments that do not violate the rule
    - `rule_violation` - the binary target

- **test.csv** - the test dataset; your objective is to predict the probability of a `rule_violation`. **NOTE**: The test dataset contains additional rules that are not seen in the in the training data, so models must be flexible to unseen rules.
- **sample_submission.csv** - a sample submission file in the correct format.

调试数据

主办方提供了约 2000 条训练数据和 10 条测试数据.

这些数据用来本地运行，和最终评分使用的训练集和测试集无关. 调试数据的训练集和测试集都只有 2 种规则（因为不能泄露另外的规则是什么），但云端评分时，测试集会有额外规则.

数据格式：

row_id	body	rule	subreddit	positive_example_1	positive_example_2	negative_example_1	negative_example_2	rule_violation
0	Banks don't want you to know this! Click here to know more!	No Advertising: Spam, referral links, unsolicited advertising, and promotional content are not allowed.	Futurology	If you could tell your younger self something different about sex, what would that be?	hunt for lady for jack off in neighbourhood http://url.inmusi.com/gakq	Watch Golden Globe Awards 2017 Live Online in HD Coverage without ADS (VIP STREAMS)	DOUBLE CEE x BANDS EPPS - "BIRDS" DOWNLOAD/STREAM: http://music.theblacksmithed.com/download/birds/	0

关键：不能只看评论 `body`，必须将所有列的信息结合起来进行预测.

- **特征组合**：模型需要将 `body`、`rule`、`subreddit`、四个示例（`positive_example_1`、`positive_example_2`，`negative_example_1`，`negative_example_2`）作为联合输入.
- 泛化挑战：模型需要学会给定规则和示例，推断新的评论是否违规，而不仅仅是记住训练集.
- 内存限制.

第一名 Solution

作者解说

这份方案的核心思路是**利用大型语言模型（LLM）的少量样本（Few-shot）能力**，结合**高效的内存优化微调**和**精细的推理后处理**，来应对数据稀疏和规则泛化的挑战

Validation strategy：在线微调

What we know:

- train.csv contains two rules with comments (bodies).
- test.csv includes six rules, four of them are new.
- The four new rules have already been identified.

> 参赛者自发在社区中合作进行排行榜探测并公示结果，对抗主办方对私自探测者的不作为：https://www.kaggle.com/competitions/jigsaw-agile-community-rules/discussion/607118
>
> 结果：https://www.kaggle.com/competitions/jigsaw-agile-community-rules/discussion/607941

Public Rule 0	**no advertising: spam, referral links, unsolicited advertising, and promotional content are not allowed.**
Public Rule 1	no legal advice: do not offer or request legal advice.
Private Rule 0	no financial advice: we do not permit comments that make personal recommendations for investments, taxes, or careers.
Private Rule 1	no medical advice: do not offer or request specific medical advice, diagnoses, or treatment recommendations.
Private Rule 2	no promotion of illegal activity: do not encourage or promote illegal activities, such as drug-related activity, violence, exploitation, theft, or other criminal behavior.
Private Rule 3	no spoilers: do not reveal important details that would limit people's ability to enjoy a show or movie.

However, to build a strong local validation, we still need to generate realistic comments for the four new rules — a nontrivial task.

实际上第一名并没有使用探测到的新规则.

I believe the Public LB is a better choice for validation because of the following:

- The host mentioned that the public/private LB split is random, which means the public LB serves as an unbiased estimator of the private LB.
- The total test data size is not small, with the public LB representing about 30% of it. This ensures low variance in score estimation.

Therefore, all models were finetuned online and validated using the public LB.

所有模型都在线进行微调（Finetuning）和验证，以 Public LB 分数作为模型性能的唯一可靠指标.

Data processing

This notebook provides an excellent baseline for data processing and finetuning. The idea is to use positive and negative examples in the test.csv as training data. The main modifications I made to the data processing pipeline are as follows:

- Used the LLM built-in chat templates.

  > 将数据转为类似对话的格式.
  >
  > System Prompt: `Reddit moderation: Does the comment violate the rule? Answer 'Yes' or 'No' only`.
  >
  > User：Comment: {comment_text}\n\nrule:{rule_text}
  >
  > Assistant：Yes / No（从标签 `rule_violation` 转换）

- I excluded the subreddit column before deduplication, since including it can create unnecessary duplicates across different subreddits. The performance improvement is significant after this change!

  > 在对评论文本、规则和示例进行去重之前，排除了 subreddit 字段. 原因：包含 subreddit 会错误地将同一条评论 / 规则组合在不同子版块下的数据视为不同，造成不必要的重复. 排除后性能显著提升.

Finetuning

Initially, I used vanilla transformers + pytorch code, which works great, achieved 0.931 on public LB early with an ensemble of Qwen3-4b-instruct-2507, llama3.2-3b, and phi-4-mini-instruct. However, I couldn't scale beyond 7B models due to **memory limitations**. Later, I switched to **Unsloth**, which is much more memory-efficient — it can easily fit a 14B model within 16GB of GPU memory.

Before finetuning, I upsampled examples from test.csv once to increase their weight, and then finetuned for one epoch. This is equivalent to one epoch for train.csv and two epochs for test.csv.

Chat templates can introduce unnecessary tokens before and after the the "Yes" or "No" targets, such as "\n\n\n\n" in Qwen3 models, as well as one or more tokens following the "Yes" or "No" targets. By default, finetuning will also apply loss to these extra tokens which is unnecessary and can slow down training.

To address this, I manually excluded these tokens from the loss computation, ensuring that only the "Yes" or "No" token contributes to the loss. An additional benefit of this step is that it standardizes the training behavior across different models, leading to more consistent convergence speeds and allowing me to use a single set of hyperparameters across models.

Inference

Since the last linear layer maps to the whole vocabulary, even after finetuning on the Yes/No tokens, there could still be a non-trival probabilitiy that the probs will be scattered on other tokens. So I constructed a candidate set of first tokens for many variants:

- ["Yes","YES","Y","yes","True"] -> Yes_ids
- ["No","NO","N","no","False"] -> No_ids
- plus their space-prefixed forms.

> 不需要LLM生成答案，只需要看它即将输出前的各token的原始值（原始分数，Logits）

The variants are obtained by local experiments outputting the most frequent topK predicted tokens.
For scoring, I output probs equivalent to the sigmoid of log-odds of Yes_ids vs No_ids.

Other tricks to speed up inference is:

- Sort test data by length

- Use forward() only. Avoids all the other operations during generation because all we need is the logits of the last token.

Postprocessing

For each model's output, I computed per-rule rankings, normalized the scores to the range [0, 1], and then ensembled based on these normalized values. This postprocessing step provided a consistent improvement, increasing the public LB score from 0.929 to 0.931 in early tests.

Model performance and final ensemble

- Larger models generally performed better.
- Qwen3 is better than the other common LLMs.
- The final ensemble is a weighted average of these six models.

model	Public LB	Private LB
Qwen3-14b	0.9297	0.9239
Qwen2.5-14b	0.9287	0.9232
Qwen3-8b	0.9272	0.9236
Qwen3-4b-instruct-2507	0.9258	0.9198
llama3.1-8b	0.9257	0.9202
Ettin-400M	0.8991	0.8944
ensemble	0.9344	0.9290

代码

运行流程：

- 运行 `train_qwen3_14b.py`