



机器学习与数据挖掘

线性回归



- 目标

找到一组参数 $\{w_k\}_{k=1}^m$ 使得预测值

$$\hat{y} = f(x_1, x_2, \dots, x_m)$$

在训练集中，尽可能接近所有数据样本的真实值 y

	大小 (英尺) x_1	# 卧室数 x_2	# 楼层 x_3	# 房龄 (年) x_4	价格 (\$ 1000) y
样本 1	2104	5	1	45	460
样本 2	1416	3	2	40	232
样本 3	1534	3	2	30	315
样本 4	852	2	1	36	178

课程大纲

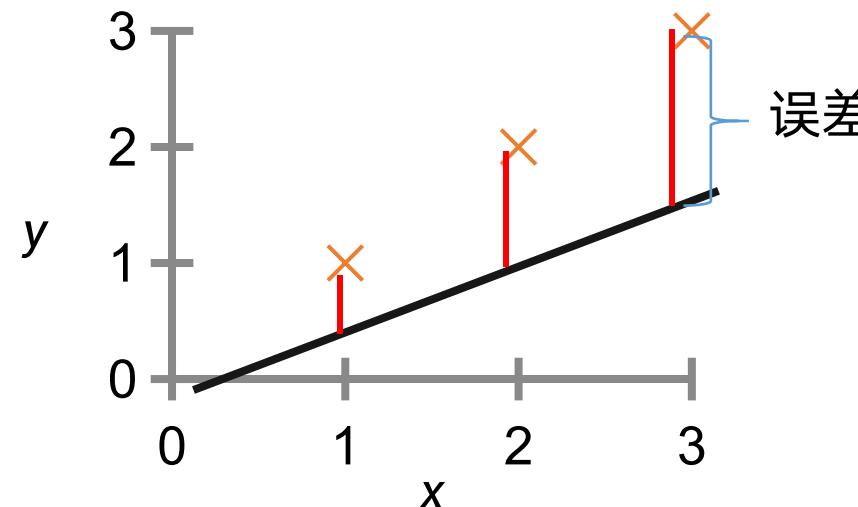
- 引言
- 单特征情况
- 多特征情况
- 数值优化

代价/损失函数

- 从数学上讲，我们的目标可以被表述为最小化代价（损失）函数

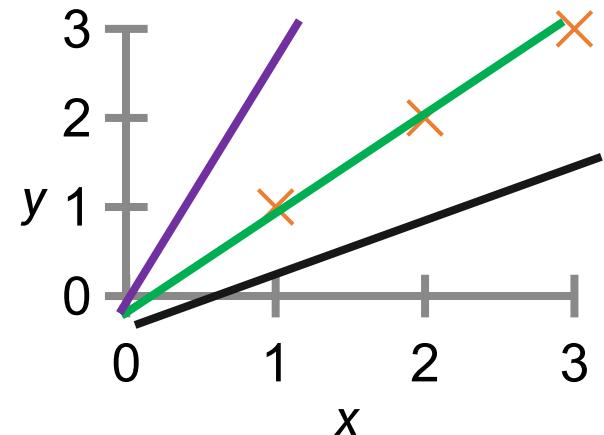
$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (f(x^{(i)}) - y^{(i)})^2$$

其中 $x^{(i)}$ 和 $y^{(i)}$ 分别表示第 i 个特征和目标值； n 是训练样本的数量



- 将 $f(x^{(i)}) = w_0 + w_1 x^{(i)}$ 代入 $L(w_0, w_1)$ 得到：

$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)})^2$$



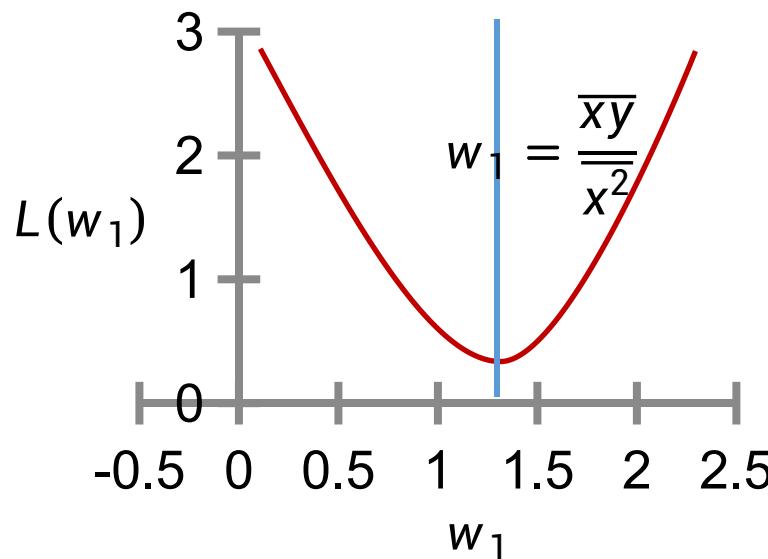
备注：为了更好地理解这个代价函数，我们通过令 $w_0 = 0$ 来简化它

- 然后，代价函数变为：

$$L(w_1) = \overline{x^2} w_1^2 - 2\overline{xy}w_1 + \overline{y^2}$$

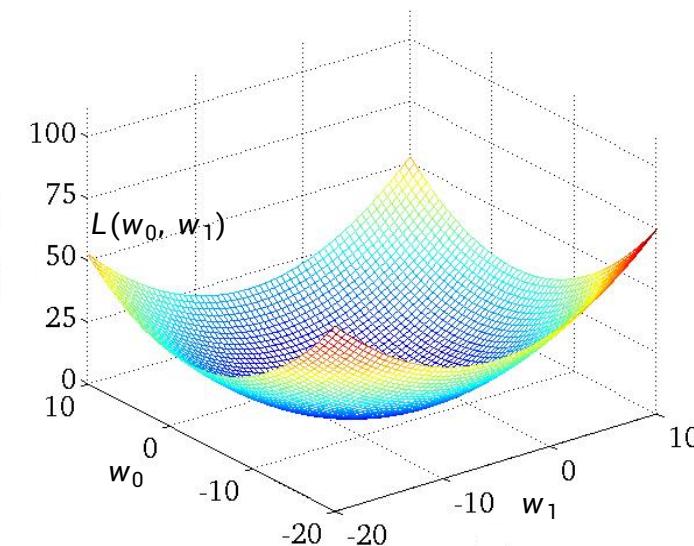
其中 $\overline{x^2} = \frac{\sum_{i=1}^n (x^{(i)})^2}{n}$, $\overline{xy} = \frac{\sum_{i=1}^n x^{(i)} y^{(i)}}{n}$ 且 $\overline{y^2} = \frac{\sum_{i=1}^n (y^{(i)})^2}{n}$

- 代价函数是关于 w_1 的二次函数



$$L(w_1) = \overline{x^2}w_1^2 - 2\overline{xy}w_1 + \overline{y^2}$$

- 如果将 w_0 也考虑在内，代价函数 $L(w_0, w_1)$ 仍然是一个二次函数，但变成了二维的



$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)})^2$$

- 最优的 w_0 和 w_1 可以通过将导数置为零来找到

$$\frac{\partial L}{\partial w_0} = \frac{2}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)}) = 0$$

$$\frac{\partial L}{\partial w_1} = \frac{2}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)} = 0$$

- 通过求解该线性方程组，可以得到最优的 w_0 和 w_1

$$w_0 = \frac{\bar{xy}\bar{x} - \bar{x}^2\bar{y}}{\bar{x}^2 - \bar{x}^2}$$

$$w_1 = \frac{\bar{x}\bar{y} - \bar{x}\bar{y}}{\bar{x}^2 - \bar{x}^2}$$

课程大纲

- 引言
- 单特征情况
- 多特征情况
- 数值优化

- 一般线性回归的函数是：

$$f(x_1, x_2 \dots x_m) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_m x_m$$

- x_i 是第 i 个特征
- 使用标量形式很繁琐。将其重新表述为矩阵形式，得到：

$$f(\mathbf{x}) = \mathbf{x}\mathbf{w}$$

- $\mathbf{x} = [1, x_1, x_2, \dots, x_m]$ 是特征行向量
- $\mathbf{w} = [w_0, w_1, w_2, \dots, w_m]^T$ 是参数列向量

通过将 \mathbf{x} 的第一个元素设置为 1，可以将 w_0 像其他参数 w_k 一样处理

代价函数

- 目标仍然是找到一个 w ，使得预测值

$$f(\mathbf{x}^{(i)}) = \mathbf{x}^{(i)} \mathbf{w}$$

尽可能接近真实值 $y^{(i)}$ ，其中 $\mathbf{x}^{(i)}$ 和 $y^{(i)}$ 分别是第 i 个特征向量和目标值

大小 (英尺)	# 卧室数	# 楼层	# 房龄 (年)	价格 (\$ 1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
....

- 因此，代价函数可以表示为：

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} \mathbf{w} - y^{(i)})^2$$

- 代价函数可以进一步写为：

$$L(\mathbf{w}) = \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

其中 \mathbf{X} 是特征矩阵，定义为：

$$\mathbf{X} \triangleq \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(n)} \end{bmatrix}$$

	大小 (英尺)	# 卧室数	# 楼层	# 房龄 (年)	价格 (\$ 1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1

\mathbf{X} \mathbf{y}

- 代价函数关于 w 的梯度为：

$$\frac{\partial L(w)}{\partial w} = \frac{2}{n} X^T (Xw - y)$$

- 由于 $L(w)$ 是一个凸函数，其最优解可以通过令其导数（梯度）为零来找到：

$$\frac{\partial L(w)}{\partial w} = \frac{2}{n} X^T (Xw - y) = 0$$

- 求解该方程可得：

$$w^* = (X^T X)^{-1} X^T y$$

- 可以验证，当特征数量为 1 时，该结果可以简化为：

$$w_0 = \frac{\bar{xy} - \bar{x}^2 \bar{y}}{\bar{x}^2 - \bar{x}^2}, \quad w_1 = \frac{\bar{x}\bar{y} - \bar{xy}}{\bar{x}^2 - \bar{x}^2}$$

补充：函数关于向量或矩阵的梯度

- 函数关于向量或矩阵的梯度的含义

► $L(\cdot)$ 是一个标量函数

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \triangleq \begin{bmatrix} \frac{\partial L(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial L(\mathbf{w})}{\partial w_m} \end{bmatrix} \quad \frac{\partial L(\mathbf{X})}{\partial \mathbf{X}} \triangleq \begin{bmatrix} \frac{\partial L(\mathbf{X})}{\partial x_{11}} & \cdots & \frac{\partial L(\mathbf{X})}{\partial x_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L(\mathbf{X})}{\partial x_{m1}} & \cdots & \frac{\partial L(\mathbf{X})}{\partial x_{mm}} \end{bmatrix}$$

函数关于向量或矩阵的梯度，只是函数关于每个元素的梯度的紧凑记法

► $L(\cdot)$ 可以是任何函数，例如范数 $\|\cdot\|^2$ 、求和 $\sum_{i=1}^m w_i$ 、迹 $\text{trace}(\cdot)$ 、行列式 $\det(\cdot)$ 等

$$\frac{\partial \|\mathbf{w}\|^2}{\partial \mathbf{w}} = 2\mathbf{w}$$

$$\frac{\partial \|\mathbf{Xw} - \mathbf{y}\|^2}{\partial (\mathbf{Xw} - \mathbf{y})} = 2(\mathbf{Xw} - \mathbf{y})$$

- 当 $L(\cdot) = [L_1(\mathbf{w}), \dots, L_p(\mathbf{w})]$ 是一个行向量函数时

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \triangleq \begin{bmatrix} \frac{\partial L_1(\mathbf{w})}{\partial \mathbf{w}} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial \mathbf{w}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L_1(\mathbf{w})}{\partial w_1} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial w_1} \\ \vdots & \dots & \vdots \\ \frac{\partial L_1(\mathbf{w})}{\partial w_m} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial w_m} \end{bmatrix}$$

它仍然是一种紧凑记法

- 然后，我们可以看到

链式法则

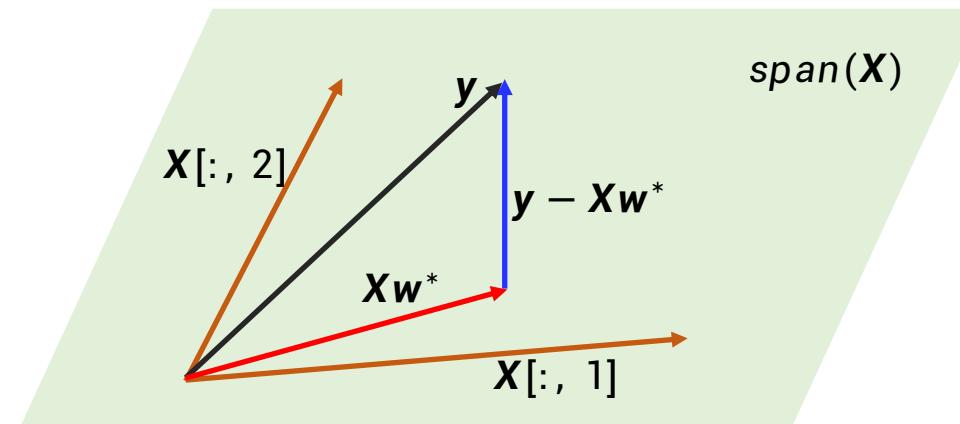
$$\frac{\partial \|X\mathbf{w} - \mathbf{y}\|^2}{\partial \mathbf{w}} = \boxed{\frac{\partial (X\mathbf{w} - \mathbf{y})^T}{\partial \mathbf{w}} \frac{\partial \|X\mathbf{w} - \mathbf{y}\|^2}{\partial (X\mathbf{w} - \mathbf{y})}} = 2X^T(X\mathbf{w} - \mathbf{y})$$

几何解释

- 从 $\mathbf{X}^T(\mathbf{X}\mathbf{w}^* - \mathbf{y}) = \mathbf{0}$ 这一条件，我们可以看出

$$\mathbf{y} - \mathbf{X}\mathbf{w}^* \perp \text{span}(\mathbf{X})$$

- 这个结果表明， $\mathbf{X}\mathbf{w}^*$ 可以被理解为 \mathbf{y} 在由 \mathbf{X} 张成的空间上的投影



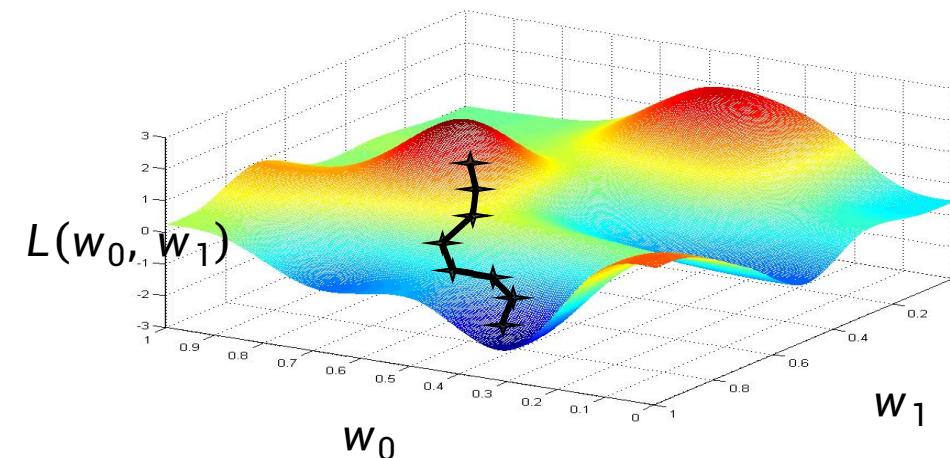
课程大纲

- 引言
- 单特征情况
- 多特征情况
- 数值优化

梯度下降

- 解析解**并非总是存在**，或者计算解析式的成本**过于高昂**
- 在这种情况下，我们可以求助于数值方法，例如梯度下降

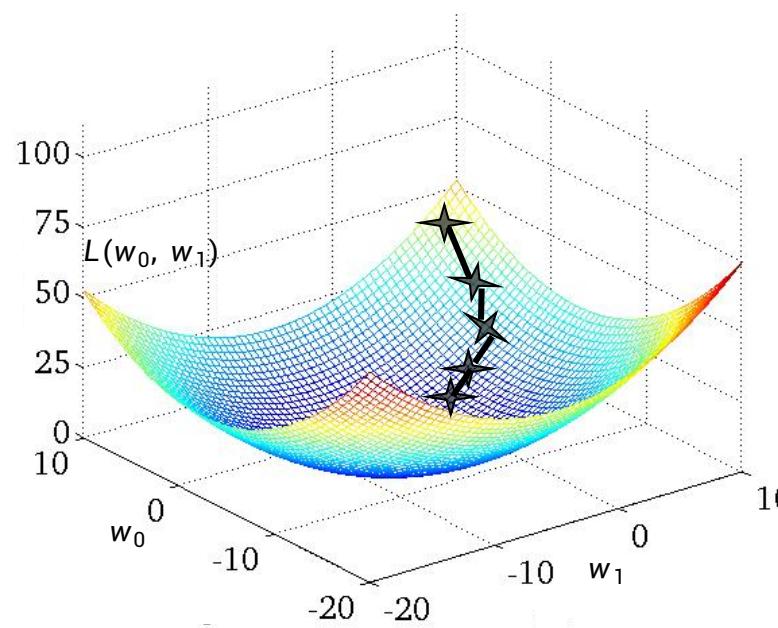
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - r \cdot \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}^{(t)}} \quad - r: \text{学习率}$$



- 现在考虑同时包含 w_0 和 w_1 的情况

$$w_0^{(t+1)} = w_0^{(t)} - r \cdot \frac{\partial L(w_0, w_1)}{\partial w_0} \Big|_{w_0 = w_0^{(t)}, w_1 = w_1^{(t)}}$$

$$w_1^{(t+1)} = w_1^{(t)} - r \cdot \frac{\partial L(w_0, w_1)}{\partial w_1} \Big|_{w_0 = w_0^{(t)}, w_1 = w_1^{(t)}}$$



随机梯度下降

- GD 算法需要在每一次迭代中计算关于模型参数 w 的损失梯度
- 通常，梯度具有以下形式：

$$\frac{\partial L(w)}{\partial w} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \ell(w, x^{(i)}, y^{(i)})}{\partial w}$$

- 每一次迭代都要求在训练数据集上的所有数据样本上计算梯度

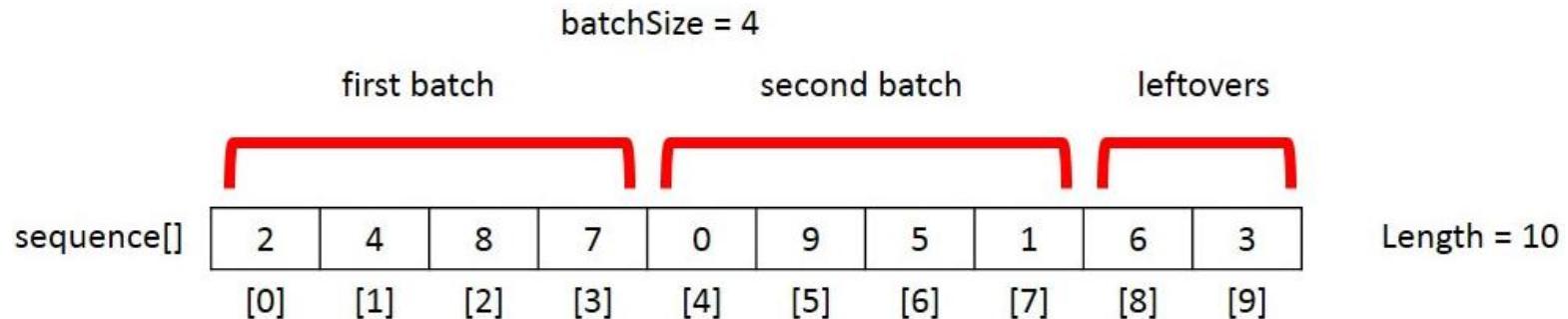
对于大型数据集，其复杂度会极高

- 为了降低复杂度，我们可以使用数据集的一小部分（即小批量）来估计梯度 $\frac{\partial L(w)}{\partial w}$

- 如何获取小批量数据？

➤ 乱序

➤ 分段



- 更新：

对真实梯度的带噪声
估计

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + r \cdot \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, \mathbf{y}^{(i)})}{\partial \mathbf{w}}$$

其中 \mathcal{B}_t 是第 t 次迭代时数据集的一个小批量

- 问题：随机梯度与下面的真实梯度有什么关系？

$$\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

与下面的真实梯度又有什么关系？

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

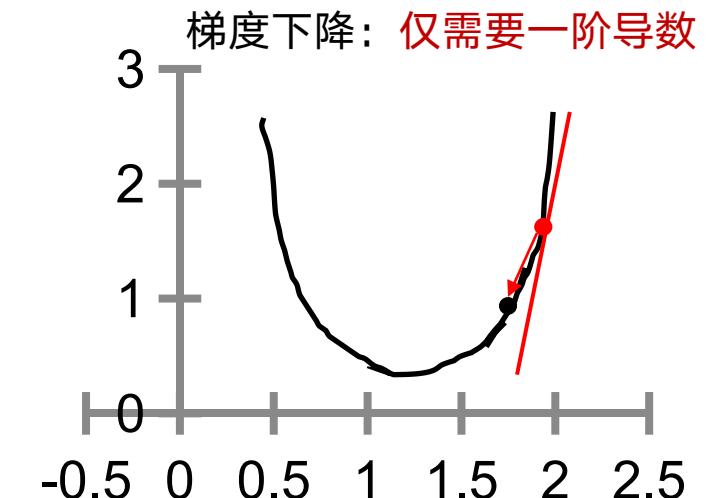
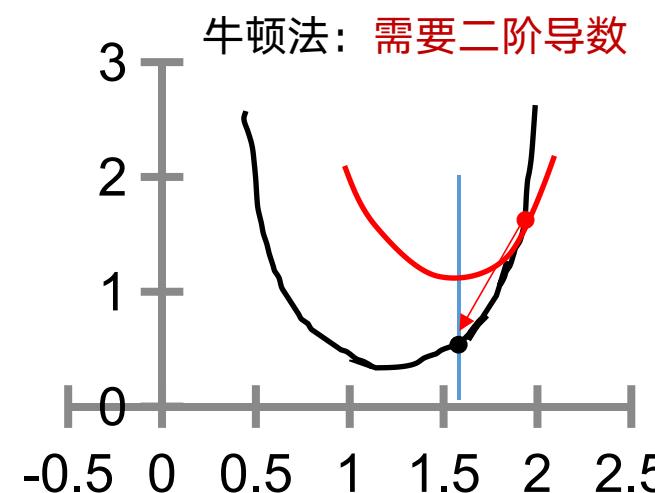
- $\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$ 是对真实梯度 $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$ 的无偏估计，即：

$$\mathbb{E}_{\mathcal{B}_t} \left[\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}} \right] = \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$$

其他优化方法

- 还有许多其他优化方法

1) 牛顿法



优点

- 无需手动选择学习率
- 收敛速度更快

缺点

- 计算成本更高

2) 拟牛顿法

3) 共轭梯度法

4) 坐标下降法

⋮

这些方法通常比梯度下降法收敛得更快，但计算成本更高



机器学习与数据挖掘

线性分类器

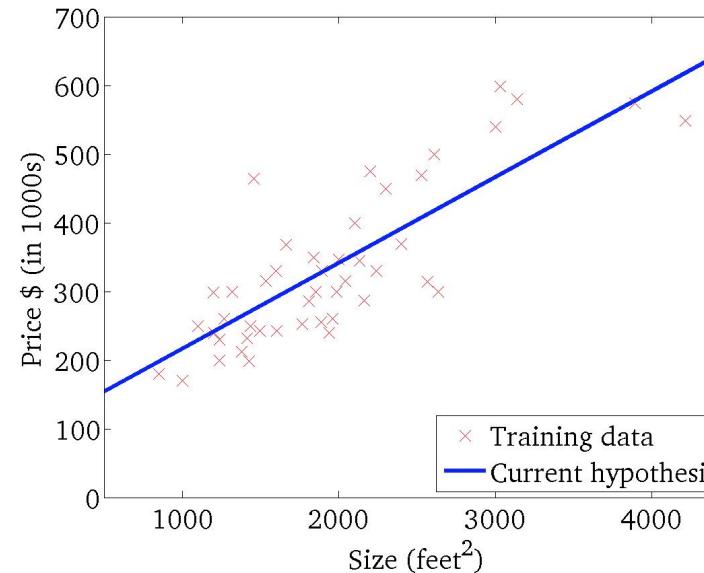


课程大纲

- 二分类
- 多分类

逻辑回归

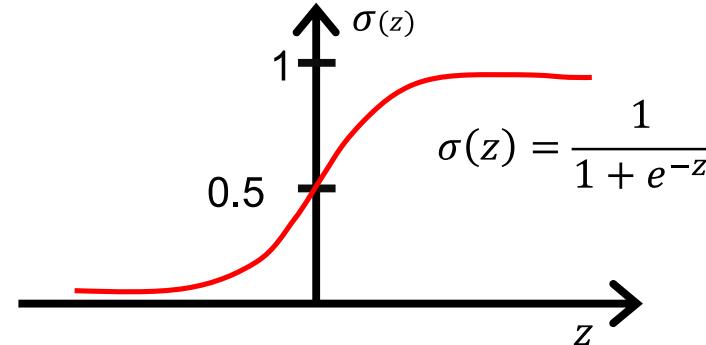
- 在分类问题中，目标变量 $y \in \{0, 1\}$
- 在线性回归中，输出 $f(\mathbf{x}) = \mathbf{x}\mathbf{w}$ 的取值范围是 $[-\infty, +\infty]$



- 线性回归的输出值与分类任务中的目标值不兼容

- Sigmoid/logistic 函数

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



- 逻辑回归

$$f(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{w})$$

- 线性回归

$$f(\mathbf{x}) = \mathbf{x}\mathbf{w}$$

- 输出范围从 $[-\infty, +\infty]$ 转换为 $[0, 1]$

代价函数

- 目标
 - 如果真实标签 $y = 1$, 我们希望 $f(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{w})$ 接近于1
 - 如果真实标签 $y = 0$, 我们希望 $f(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{w})$ 接近于0
- 为了实现这个目标, 我们可以定义一个类似于回归中的代价函数:

$$L(\mathbf{w}) = (\sigma(\mathbf{x}\mathbf{w}) - y)^2$$

- 或者，我们也可以寻求最小化：

$$L(\mathbf{w}) = \begin{cases} -\log(\sigma(\mathbf{x}\mathbf{w})) & \text{if } y = 1 \\ -\log(1 - \sigma(\mathbf{x}\mathbf{w})) & \text{if } y = 0 \end{cases}$$

- 上面的目标可以等价地写成：

$$L(\mathbf{w}) = -y \log(\sigma(\mathbf{x}\mathbf{w})) - (1-y) \log(1 - \sigma(\mathbf{x}\mathbf{w}))$$

如果 $y = 1$, $L(\mathbf{w})$ 简化为 $L(\mathbf{w}) = -\log(\sigma(\mathbf{x}\mathbf{w}))$;

否则, 如果 $y = 0$, $L(\mathbf{w})$ 简化为 $L(\mathbf{w}) = -\log(1 - \sigma(\mathbf{x}\mathbf{w}))$

- 上面的损失函数被称为**交叉熵损失**

- 交叉熵损失：

$$L(\mathbf{w}) = -y \log(\sigma(\mathbf{x}\mathbf{w})) - (1-y) \log(1 - \sigma(\mathbf{x}\mathbf{w}))$$

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

- 二阶导数（海森矩阵）：

$$H = \frac{\partial^2 L(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^\top} = \mathbf{x}\mathbf{x}^\top \sigma(\mathbf{x}\mathbf{w})(1 - \sigma(\mathbf{x}\mathbf{w}))$$

- 判断半正定性：

$$\mathbf{v}^\top H \mathbf{v} = (\mathbf{x}^\top \mathbf{v})^2 \sigma(\mathbf{x}\mathbf{w})(1 - \sigma(\mathbf{x}\mathbf{w})) \geq 0$$

- 交叉熵损失函数的海森矩阵是半正定的，所以该函数是一个凸函数

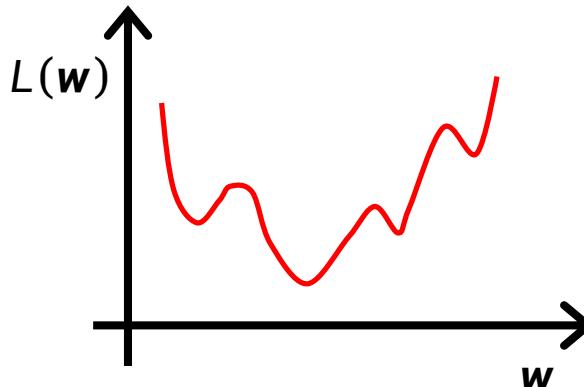
- 哪个代价函数更好？

平方误差: $L(w) = (\sigma(xw) - y)^2$

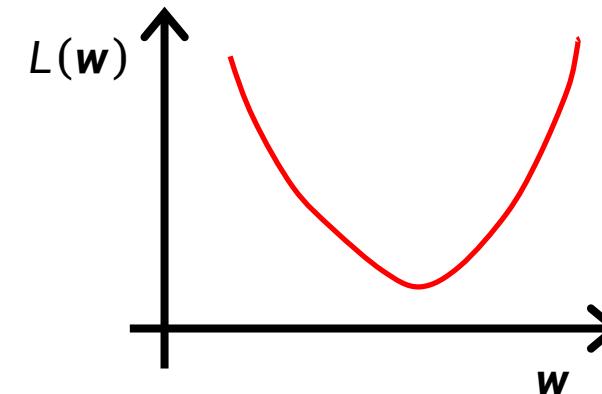
函数 $f(z) = \log(\sigma(z))$ 的形状
是什么样的?

交叉熵: $L(w) = -[y \log(\sigma(xw)) + (1-y) \log(1-\sigma(xw))]$

➤ 平方损失是非凸的



➤ 交叉熵是凸的



凸函数更易于优化

- 在下一次课中，我们将从**精确建模**的角度，展示使用交叉熵损失的另一个优点

- 交叉熵损失

$$L(\mathbf{w}) = -y \log(\sigma(\mathbf{x}\mathbf{w})) - (1-y) \log(1 - \sigma(\mathbf{x}\mathbf{w}))$$

- 交叉熵损失的梯度

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_{\ell=1}^N [\sigma(\mathbf{x}^{(\ell)}\mathbf{w}) - y^{(\ell)}] \mathbf{x}^{(\ell)T}$$

- 求解 $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = 0$ 可以得到最优的 \mathbf{w}^* 。将 Sigmoid 函数代入，我们得到：

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_{\ell=1}^N \left[\frac{1}{1 + e^{-\mathbf{x}^{(\ell)}\mathbf{w}}} - y^{(\ell)} \right] \mathbf{x}^{(\ell)T} = \mathbf{0}$$

- 这个方程是关于 \mathbf{w} 的一个非线性方程，无法直接求解这个方程，不能得到一个解析解

梯度下降

- 交叉熵损失的梯度

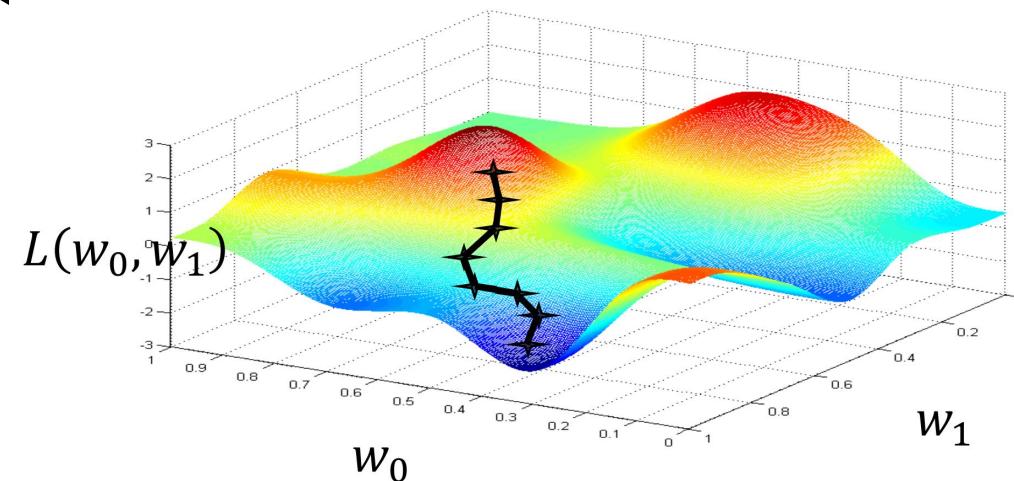
$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_{\ell=1}^N [\sigma(\mathbf{x}^{(\ell)} \mathbf{w}) - y^{(\ell)}] \mathbf{x}^{(\ell)T}$$

- 求解 $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = 0$ 可以得到最优的 \mathbf{w}^* 。但在这里，*解析解不存在*
- 因此，我们只能求助于数值方法
 - 梯度下降
 - 牛顿法
 - 坐标下降
 -

- 梯度下降法
- 从一个初始的 w_0 开始，沿着梯度的反方向迭代更新参数，以逐步逼近最小值点

$$w_{t+1} = w_t - r \cdot \frac{\partial L(w)}{\partial w}$$

- r 是学习率，控制每一步更新的幅度
- 收敛性：由于交叉熵损失是凸函数 梯度下降法保证能收敛到最优值 w^*



- 梯度的直观理解

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_{\ell=1}^N \left[\underbrace{\sigma(\mathbf{x}^{(\ell)} \mathbf{w}) - y^{(\ell)}}_{\text{预测误差}} \right] \mathbf{x}^{(\ell)T}$$

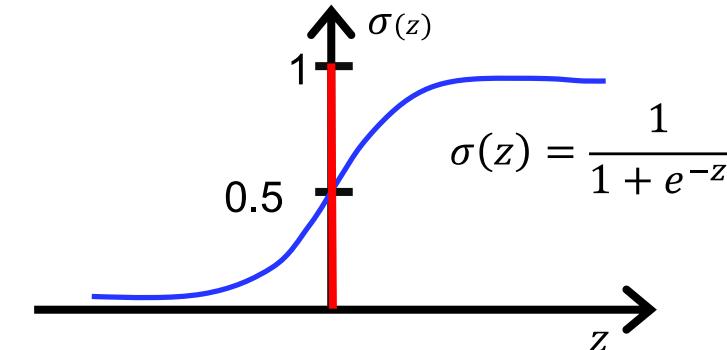
可以发现梯度下降法 (GD) 总是试图减小预测误差

- 如果 $y^{(\ell)} = 1$, 算法会驱使 $\sigma(\mathbf{x}^{(\ell)} \mathbf{w})$ 趋近于 1
- 如果 $y^{(\ell)} = 0$, 算法会驱使 $\sigma(\mathbf{x}^{(\ell)} \mathbf{w})$ 趋近于 0

决策边界

- 决策边界是特征空间中的一个“边界”，它将不同类别的点分隔开
- 样本按以下方式被分为 1 和 0 两类：

$$\hat{y} = \begin{cases} 1, & \text{if } \sigma(\mathbf{x}\mathbf{w}) \geq 0.5 \\ 0, & \text{if } \sigma(\mathbf{x}\mathbf{w}) < 0.5 \end{cases}$$

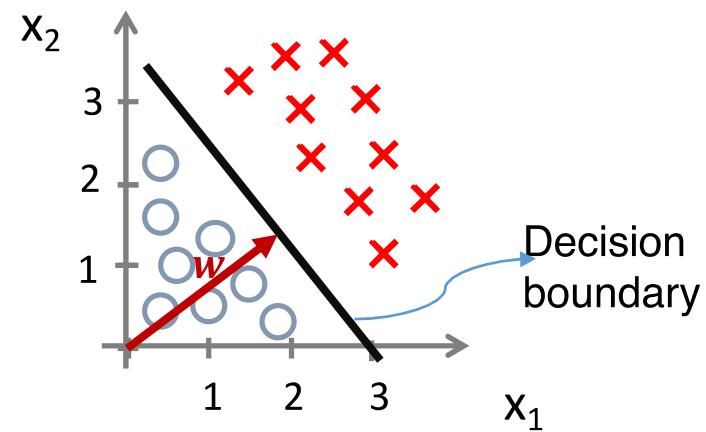


- 这等价于按以下方式对样本进行分类：

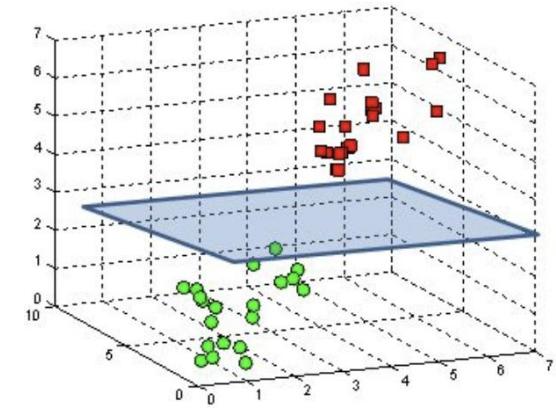
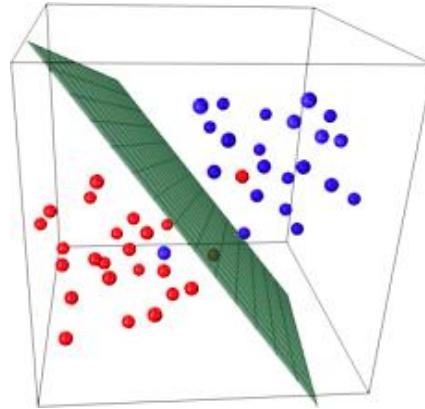
$$\hat{y} = \begin{cases} 1, & \text{if } \mathbf{x}\mathbf{w} \geq 0 \\ 0, & \text{if } \mathbf{x}\mathbf{w} < 0 \end{cases}$$

- 决策边界由满足 $\mathbf{x}\mathbf{w} = 0$ 的 \mathbf{x} 组成

- 由于 w 是一个向量，所有满足 $xw = 0$ 的 x 构成一个与 w 正交的空间



- 在二维情况下，这个空间是一条直线
- 在三维情况下，这个空间是一个平面

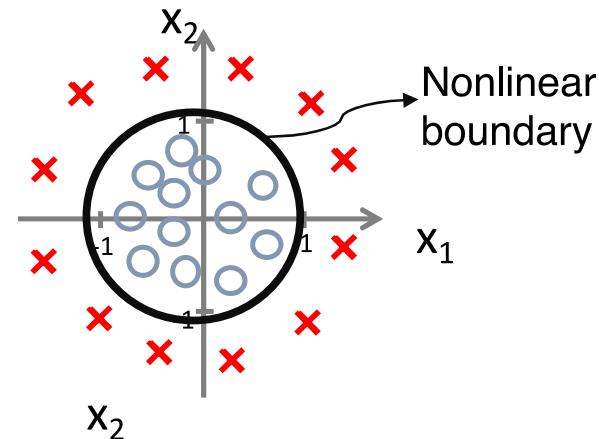


- 对于一个固定的向量 $w \in \mathbb{R}^K$, 点集

$$x \in \{x | xw = 0\}$$

构成一个 $(K - 1)$ 维的超平面

- 超平面永远无法表示非线性的决策边界, 例如:

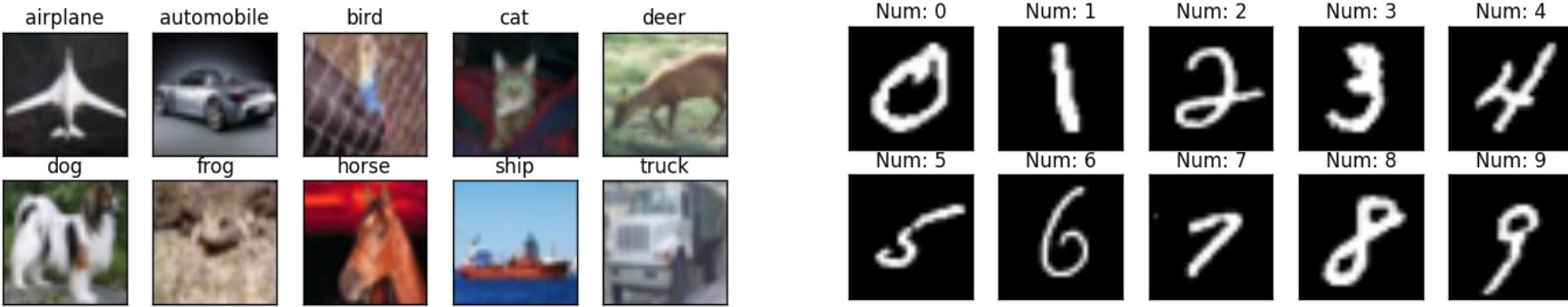


这就是为什么逻辑回归被称为线性分类器的原因

课程大纲

- 二分类
- 多分类

- 许多应用包含多于 2 个类别



- 处理多类别分类的两种方法：

➤ One-vs-All

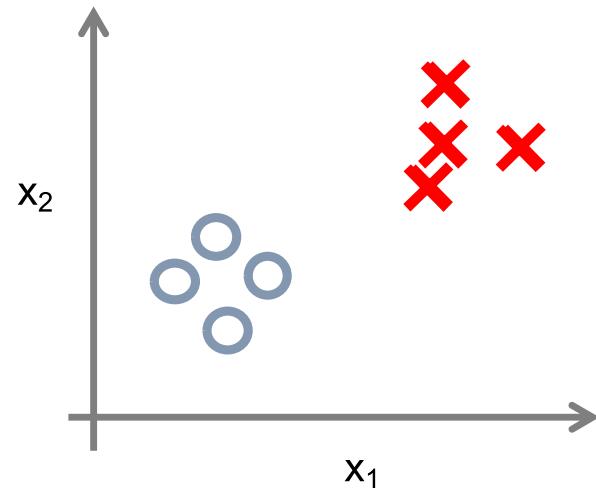
将多类别问题转化为多个二元分类问题

➤ Softmax 函数

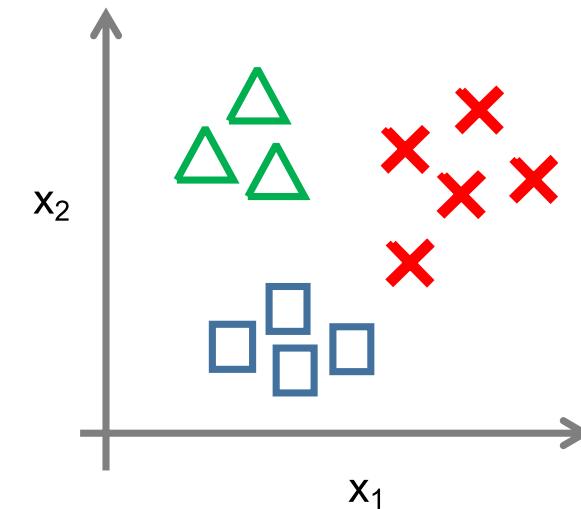
直接将样本分到其中一个类别

One-vs-All

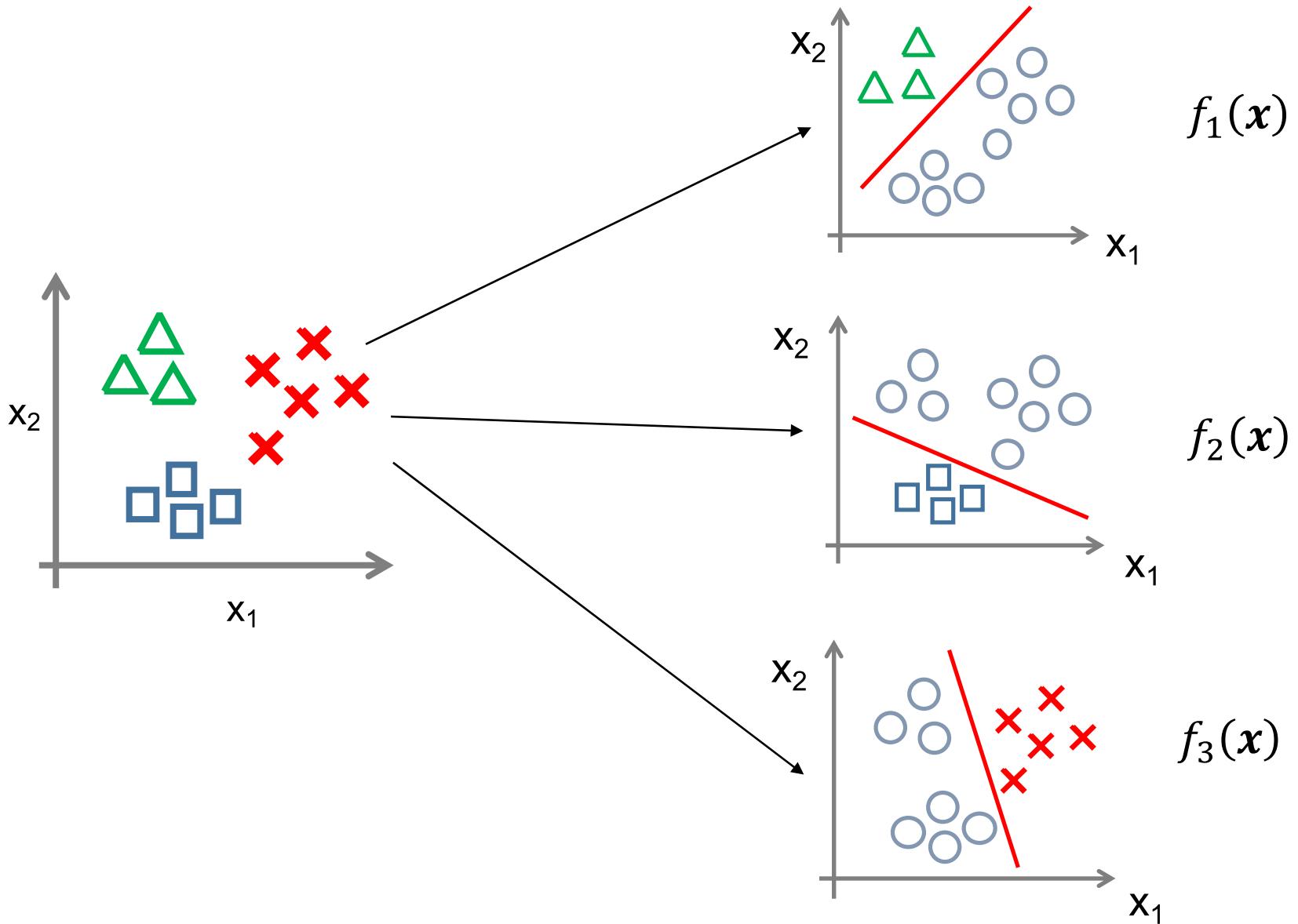
二分类



多分类

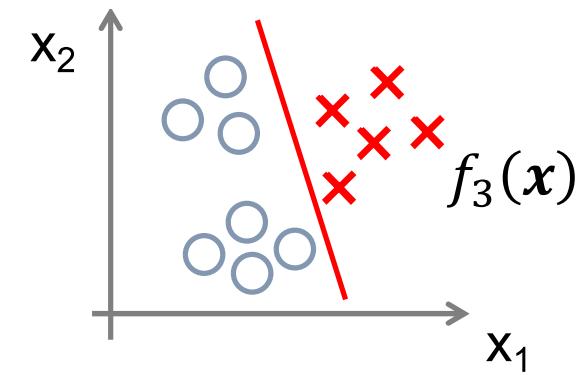
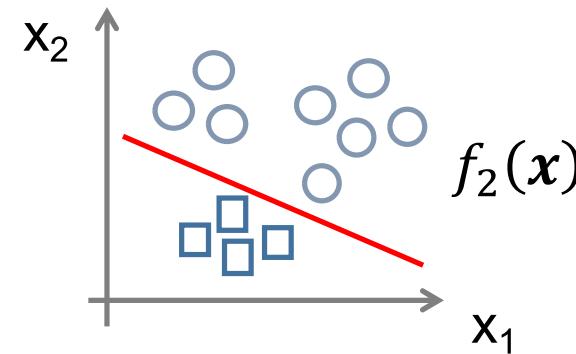
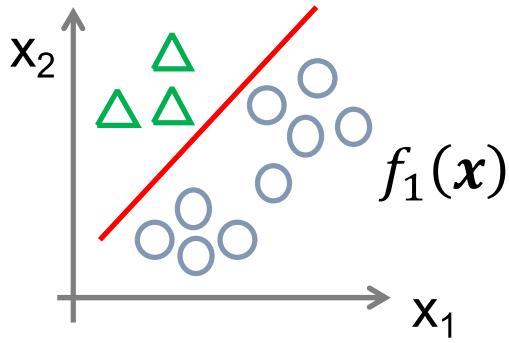


- 为每个类别训练一个分类器



- 为了预测新样本 x 的类别，选择满足以下条件的类别：

$$k = \arg \max_i f_i(x)$$

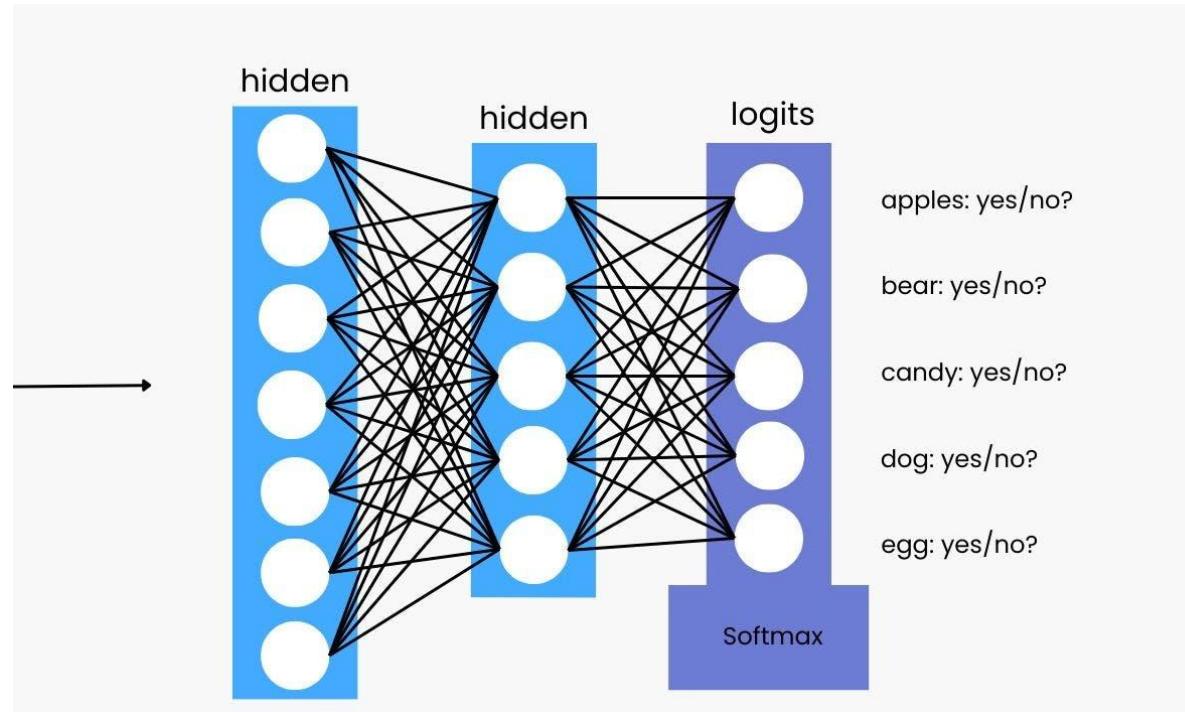


Softmax 函数

- Softmax 函数

$$\text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

可以看出, $\sum_{i=1}^K \text{softmax}_i(\mathbf{z}) = 1$



$$\text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

- 数据 \mathbf{x} 被分类到第 i 个类别的概率是

$$f_i(\mathbf{x}) = \text{softmax}_i(\mathbf{x}\mathbf{W}) = \frac{e^{\mathbf{x}\mathbf{w}_i}}{\sum_{k=1}^K e^{\mathbf{x}\mathbf{w}_k}}$$

其中 $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K]$

- 如果 \mathbf{x} 属于第 i 个类别，模型应该促使 $f_i(\mathbf{x})$ 尽可能地大

- 当 $K = 2$ 时，Softmax 函数等价于 logistic 函数

➤ 在二分类情况下，我们有：

$$\begin{aligned} softmax_1(\mathbf{x}\mathbf{w}) &= \frac{e^{\mathbf{x}\mathbf{w}_1}}{e^{\mathbf{x}\mathbf{w}_1} + e^{\mathbf{x}\mathbf{w}_2}} \\ &= \frac{1}{1 + e^{-\mathbf{x}(\mathbf{w}_1 - \mathbf{w}_2)}} \end{aligned}$$

$$\begin{aligned} softmax_2(\mathbf{x}\mathbf{w}) &= \frac{e^{\mathbf{x}\mathbf{w}_2}}{e^{\mathbf{x}\mathbf{w}_1} + e^{\mathbf{x}\mathbf{w}_2}} \\ &= \frac{e^{-\mathbf{x}(\mathbf{w}_1 - \mathbf{w}_2)}}{1 + e^{-\mathbf{x}(\mathbf{w}_1 - \mathbf{w}_2)}} \end{aligned}$$

➤ 可以看出：

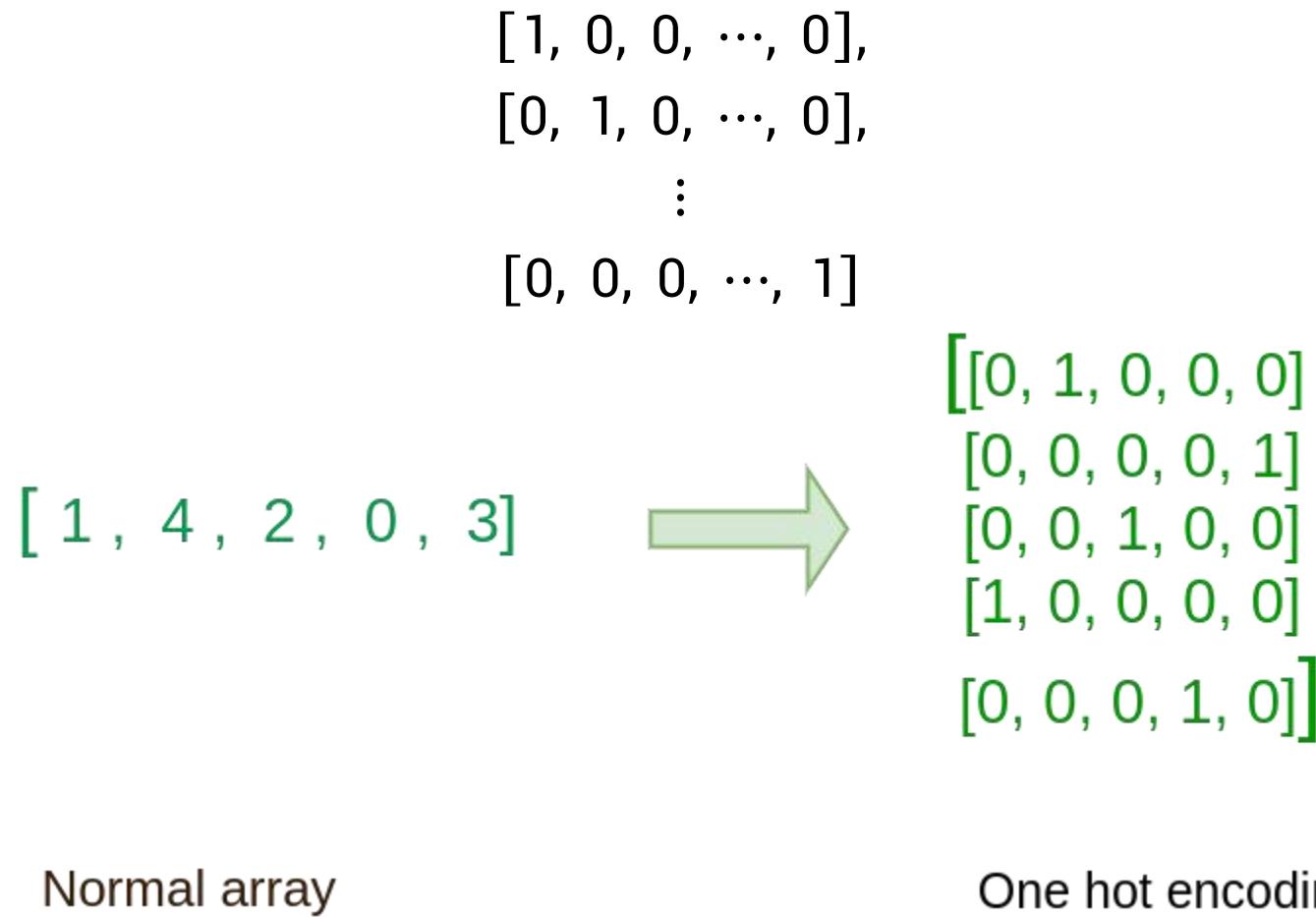
$$softmax_1(\mathbf{x}\mathbf{w}) = \sigma(\mathbf{x}(\mathbf{w}_1 - \mathbf{w}_2))$$

$$softmax_2(\mathbf{x}\mathbf{w}) = 1 - \sigma(\mathbf{x}(\mathbf{w}_1 - \mathbf{w}_2))$$

二分类 Softmax 分类等价于逻辑回归，其模型参数为 $\mathbf{w}_1 - \mathbf{w}_2$

代价函数

- 对于一个有 K 个类别的训练数据集，其标签 y 用一个**独热向量** (one-hot vector)



- 目标是最大化相应的概率 $f_i(\mathbf{x})$ 。因此，代价函数可以写为

$$L(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K) = -\frac{1}{N} \sum_{\ell=1}^N \sum_{k=1}^K y_k^{(\ell)} \log [\text{softmax}_k(\mathbf{x}^{(\ell)} \mathbf{w})]$$

— $y_k^{(\ell)}$ 是 $\mathbf{y}^{(\ell)}$ 的第 k 个元素

Cross-entropy 损失

梯度下降

- 关于 w_j 的梯度是

$$\frac{\partial L(w_1, w_2, \dots, w_K)}{\partial w_j} = \frac{1}{N} \sum_{\ell=1}^N \left(\underbrace{\text{softmax}_j(\mathbf{x}^{(\ell)} \mathbf{W}) - y_j^{(\ell)}}_{\text{预测误差}} \right) \mathbf{x}^{(\ell)T}$$

注意，对于 $j = 1, \dots, K$ ，所有的 w_j 都应该同时更新

$$\frac{\partial L(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K)}{\partial \mathbf{w}_j} = \frac{1}{N} \sum_{\ell=1}^N \left(\underbrace{\text{softmax}_j(\mathbf{x}^{(\ell)} \mathbf{W}) - \mathbf{y}_j^{(\ell)}}_{\text{预测误差}} \right) \mathbf{x}^{(\ell)T}$$

- 将 \mathbf{W} 表示为 $[\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K]$, 我们有:

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} = \frac{1}{N} \sum_{\ell=1}^N \mathbf{x}^{(\ell)T} (\text{softmax}(\mathbf{x}^{(\ell)} \mathbf{W}) - \mathbf{y}^{(\ell)})$$

- $\text{softmax}(\mathbf{x}^{(\ell)} \mathbf{W}) = [\text{softmax}_1(\mathbf{x}^{(\ell)} \mathbf{W}), \dots, \text{softmax}_K(\mathbf{x}^{(\ell)} \mathbf{W})]$ 是一个行向量
- 更新: $\mathbf{W}_{t+1} = \mathbf{W}_t - r \cdot \frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_t}$



机器学习与数据挖掘

线性回归与分类的概率视角



课程大纲

- 介绍
- 从概率角度看回归
- 从概率角度看分类

条件概率视角

- 回归和分类的目标是在给定输入数据 x 的情况下，预测可能的输出 y

$$x \xrightarrow{\text{预测}} y$$

- 在回归和分类中，预测是通过确定性函数进行的

$$\text{Regression: } f(x) = \mathbf{x}w$$

$$\text{Classification: } f(x) = \sigma(\mathbf{x}w)$$

从概率的角度来看，为了在给定 x 的情况下预测输出 y ，我们只需要建模条件概率

$$p(y|x)$$

- 有了条件概率 $p(y|x)$, 输出可以按以下方式预测:

$$\text{均值: } \hat{y} = \int y p(y|x) dy$$

或

$$\text{最大后验: } \hat{y} = \arg \max_y p(y|x)$$

均值: 更侧重于预测输出的平均结果, 适用于需要连续预测或期望值估计的场景

最大后验: 更侧重于选择最有可能的类别或参数, 适用于分类问题和参数估计

回归和分类的目标都与条件概率紧密相关

课程大纲

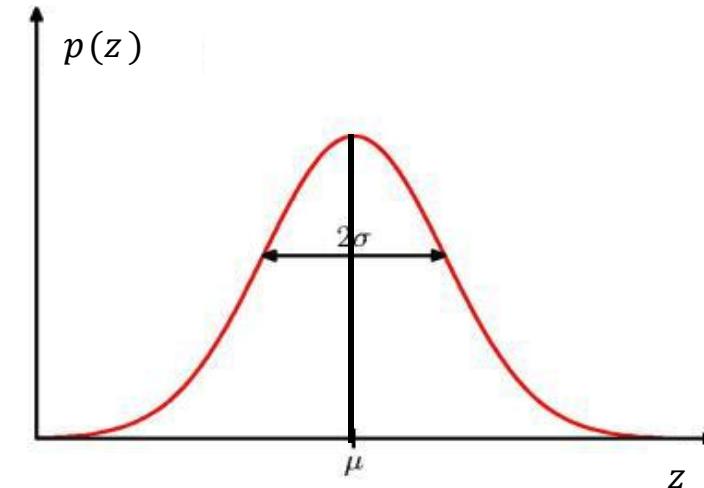
- 介绍
- 从概率角度看回归
- 从概率角度看分类

高斯分布

- 一元高斯分布

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2} \frac{(z - \mu)^2}{\sigma^2}\right] \triangleq \mathcal{N}(z; \mu, \sigma^2)$$

- μ 是均值
- $\sigma^2 = E[(z - \mu)^2]$ 是方差
- σ 是标准差



- μ 是分布的峰值和中心
- σ 决定了分布的离散程度（或宽度）

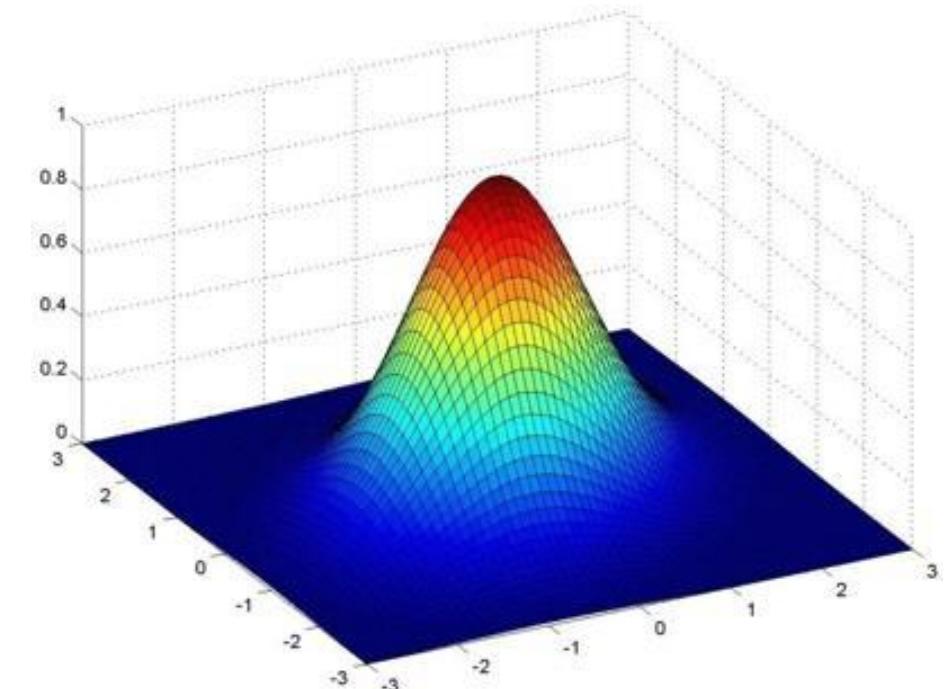
钟形

高斯分布

- 多元高斯分布

$$p(\mathbf{z}) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{z} - \boldsymbol{\mu}) \right\} \triangleq \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \Sigma)$$

- D 是维度
 - $\boldsymbol{\mu} \in R^D$ 是均值向量
 - $\Sigma \in R^{D \times D}$ 是标准差
- $\boldsymbol{\mu}$ 是分布的**峰值**和**中心**
- Σ 决定了分布的离散程度 (或宽度)

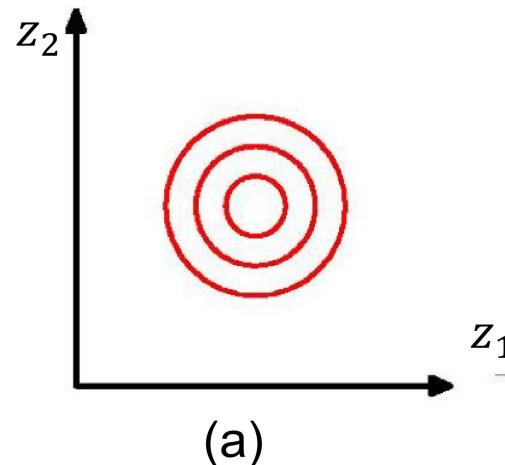


高斯分布

- 不同 Σ 类型下的分布形状

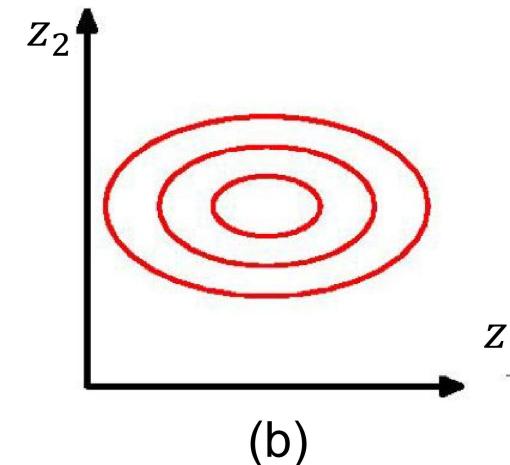
$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$$\sigma_1^2 = \sigma_2^2$$

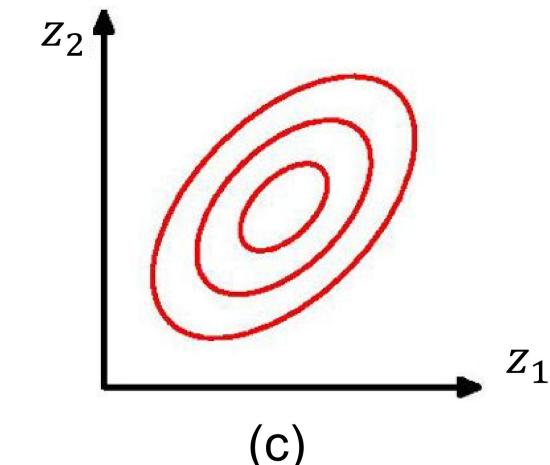


$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$$\sigma_1^2 > \sigma_2^2$$



$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho \\ \rho & \sigma_2^2 \end{bmatrix}$$



- 无论 Σ 如何变化, 峰值总是位于 μ (单峰)

高斯分布

- 对于每一个协方差矩阵 Σ ，可以分解为 $\Sigma = \mathbf{U}\Lambda\mathbf{U}^T$
 - \mathbf{U} 是一个正交矩阵，其中 $\mathbf{U}\mathbf{U}^T = \mathbf{I}$
 - Λ 是一个对角矩阵
- 使 $\mathbf{z}' = \mathbf{U}^T \mathbf{z}$ ， $\boldsymbol{\mu}' = \mathbf{U}^T \boldsymbol{\mu}$ ，分布可以表示为：

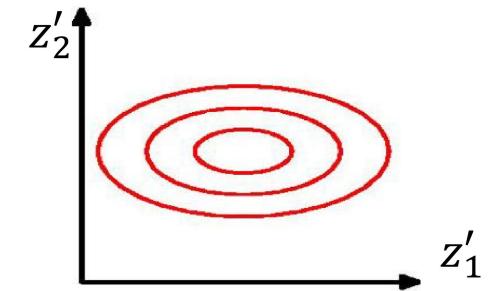
$$p(\mathbf{z}') = \frac{1}{(2\pi)^{D/2} |\Lambda|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{z}' - \boldsymbol{\mu}')^T \Lambda^{-1} (\mathbf{z}' - \boldsymbol{\mu}') \right\}$$

高斯分布

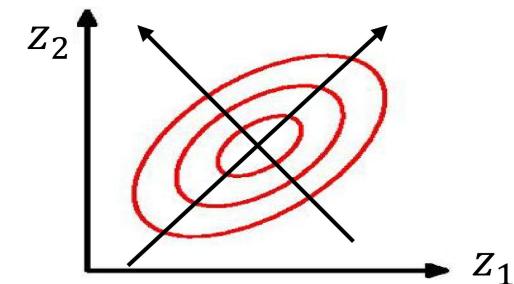
- 使 $\mathbf{z}' = \mathbf{U}^T \mathbf{z}$, $\boldsymbol{\mu}' = \mathbf{U}^T \boldsymbol{\mu}$, 分布可以表示为:

$$p(\mathbf{z}') = \frac{1}{(2\pi)^{D/2} |\Lambda|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{z}' - \boldsymbol{\mu}')^T \Lambda^{-1} (\mathbf{z}' - \boldsymbol{\mu}') \right\}$$

因此 $p(\mathbf{z}')$ 的形状看起来是:



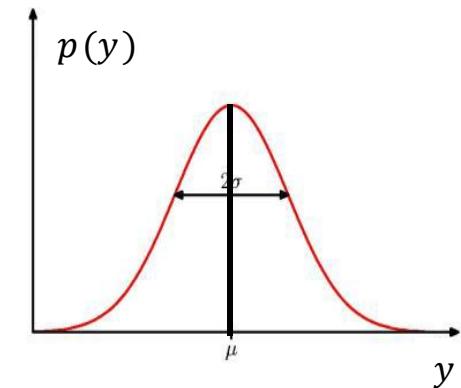
而 $p(\mathbf{z})$ 的形状被 \mathbf{U} 旋转了:



线性回归

- 从概率视角来看，为了进行预测，我们只需要指定条件概率分布 $p(y|x)$ 。对于回归问题，我们假设这个分布是正态分布

$$\begin{aligned} p(y|x; \mathbf{w}) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(y - \mathbf{x}\mathbf{w})^2}{\sigma^2}\right] \\ &= \mathcal{N}(y; \mathbf{x}\mathbf{w}, \sigma^2) \end{aligned}$$



- 我们使用分布的均值进行预测，即：

$$\hat{y} = \mathbf{x}\mathbf{w}$$

在这里得到的 \mathbf{w} 与传统回归中得到的 \mathbf{w} 相同吗？

- 模型训练的目标是找到使对数概率最大化的参数 \mathbf{w} , 即:

$$\max_{\mathbf{w}} \log p(y|\mathbf{x}; \mathbf{w})$$

对数似然函数

- 从 $p(y|\mathbf{x}; \mathbf{w})$ 的表达式中, 我们得到:

$$\log p(y|\mathbf{x}; \mathbf{w}) = -\frac{1}{2} \frac{(y - \mathbf{x}\mathbf{w})^2}{\sigma^2} + \text{constant}$$

因此, 最大化对数似然 $p(y|\mathbf{x}; \mathbf{w})$ 等价于最小化

$$\min_{\mathbf{w}} (y - \mathbf{x}\mathbf{w})^2,$$

这与回归中使用的损失函数是相同的

- 对于 N 个训练样本 $(\mathbf{x}^{(i)}, y^{(i)})$, 通过假设它们是 **独立同分布 (i.i.d.)** 的, 可以得到它们的联合条件概率密度函数:

$$p(y^{(1)}, \dots, y^{(N)} | \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2} \frac{(y^{(i)} - \mathbf{x}^{(i)} \mathbf{w})^2}{\sigma^2}\right]$$

- 对数似然函数是:

$$\log p(y^{(1)}, \dots, y^{(N)} | \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = -\frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - \mathbf{x}^{(i)} \mathbf{w})^2 + \text{constant}$$

- 最大化对数似然 $\log p(y^{(1)}, \dots, y^{(N)} | \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)})$ 等价于最小化

$$L(\mathbf{w}) = \sum_{i=1}^N (y^{(i)} - \mathbf{x}^{(i)} \mathbf{w})^2,$$

你能想出为什么这里会出现求和符号 Σ 吗?

这与回归中使用的损失函数是相同的

- 从概率建模的角度来看，线性回归实际上等价于：

- 建模：假设条件分布是对角高斯分布
 - 训练：通过最大化对数似然来训练模型

课程大纲

- 介绍
- 从概率角度看回归
- 从概率角度看分类

伯努利分布

- 伯努利分布与其他分布的关系

- 与二项分布的关系:

伯努利分布是二项分布的一种特殊情况

- 与高斯分布的关系:

在某些情况下，伯努利分布可以视为高斯分布的一个特例，
特别是当高斯分布的协方差矩阵是对角矩阵时

这是因为在这种情况下，高斯分布退化为伯努利分布，其中协方差矩阵的对角线元素表示变量间不相关，即每个变量独立地取值为 0 或 1

伯努利分布

- 伯努利分布

$$p(z) = \begin{cases} \pi, & \text{if } z = 1 \\ 1 - \pi, & \text{if } z = 0 \end{cases}$$

其中 $\pi \in [0, 1]$ 是 z 等于 1 的概率

- $p(z)$ 可以简洁地表示为:

$$p(z) = \pi^z \cdot (1 - \pi)^{1-z}$$

其中 $z = 0$ 或 1

二分类

- 为了实现二元分类，我们假设条件概率是伯努利分布

$$p(y|x) = (\sigma(xw))^y \cdot (1 - \sigma(xw))^{1-y}$$

其中 $\pi = \sigma(xw)$; 且 $y = 0$ 或 1

- 训练目标是最大化对数似然函数

$$\log p(y|x) = y \log \sigma(xw) + (1 - y) \log (1 - \sigma(xw))$$

二分类

回想一下，逻辑回归最小化：

cross entropy

$$\triangleq -y \log \sigma(\mathbf{x} \mathbf{w}) - (1 - y) \log (1 - \sigma(\mathbf{x} \mathbf{w}))$$

最大化 $\log p(y|\mathbf{x})$ 等价于最小化交叉熵

- 逻辑回归等价于：
 - 建模：假设输出服从伯努利条件分布
 - 训练：通过最大化对数似然来训练模型

类别分布

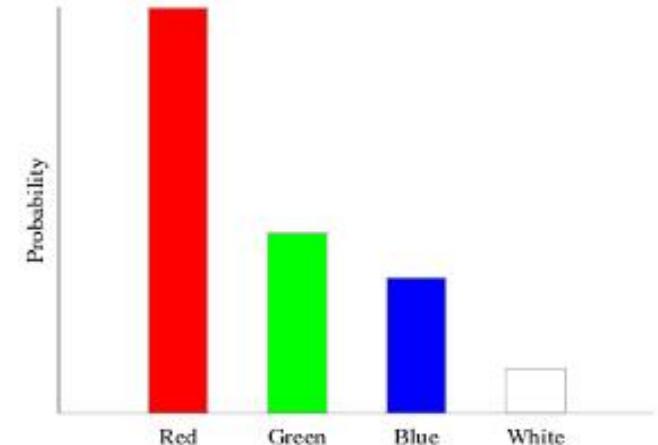
- 类别分布

$$p(\mathbf{z} = \text{one hot}_k) = \pi_k$$

- 其中 $\text{one hot}_i = [0, \dots, 0, 1, 0, \dots, 0]$ 是一个独热向量，其中第 i 个元素是唯一非零的元素 1
- $\sum_{k=1}^K \pi_k = 1$
- 该分布可以等价地写为：

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}$$

其中 \mathbf{z} 是一个独热向量



多分类

- 建模：通过将概率 π_k 设置

$$\pi_k = \text{softmax}_k(\mathbf{x} \mathbf{W}),$$

假设条件概率分布服从类别分布

$$p(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^K [\text{softmax}_k(\mathbf{x} \mathbf{W})]^{y_k}$$

多分类

条件概率分布服从类别分布: $p(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^K [\text{soft max}_k(\mathbf{x}\mathbf{W})]^{y_k}$

- 训练: 给定一个训练样本 (\mathbf{x}, \mathbf{y}) , 模型通过最大化对数似然函数进行训练

$$\begin{aligned} & \log p(\mathbf{y}|\mathbf{x}) \\ &= \sum_{k=1}^K y_k \cdot \log (\text{soft max}_k(\mathbf{x}\mathbf{W})) \\ & \quad \textcolor{red}{=- \text{cross entropy}} \end{aligned}$$

总结

- 回归、逻辑回归和多类别回归可以被归纳到同一个通用框架下

1) 建模: 为输出 y 假设不同的条件概率密度函数

- 回归: 高斯分布
- 逻辑回归: 伯努利分布
- 多类别逻辑回归: 分类分布

2) 训练: 最大化对数似然函数



机器学习与数据挖掘

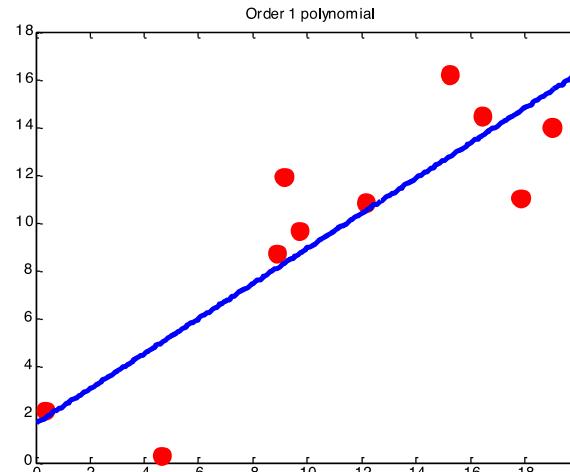
模型非线性化、过拟合与正则化



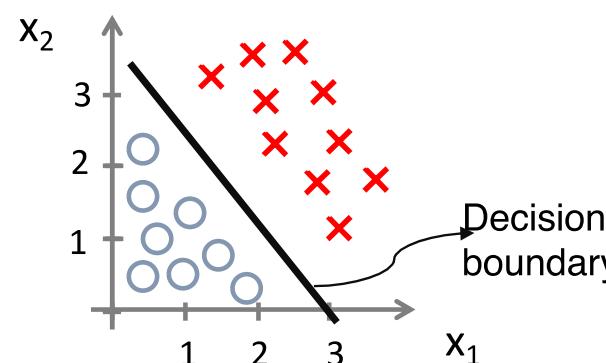
课程大纲

- 模型非线性化
- 过拟合
- 模型选择
- 正则化

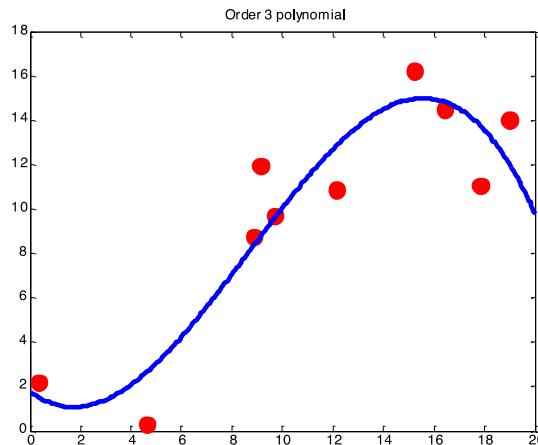
- 线性回归只能对输入 x 和输出 y 之间的线性关系进行建模



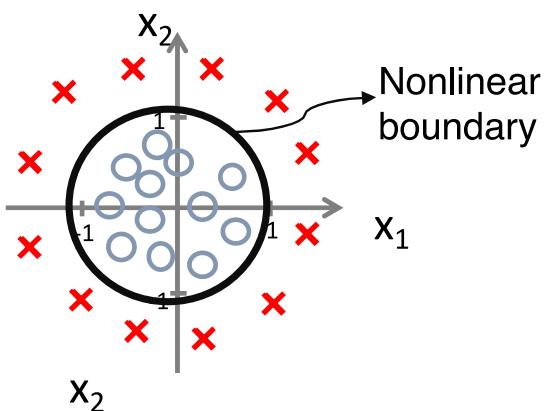
- 对于线性分类器，其决策边界只能是线性的



- 对于更复杂的任务，模型应该具备更多的能力以捕获：
 - 非线性输入-输出关系

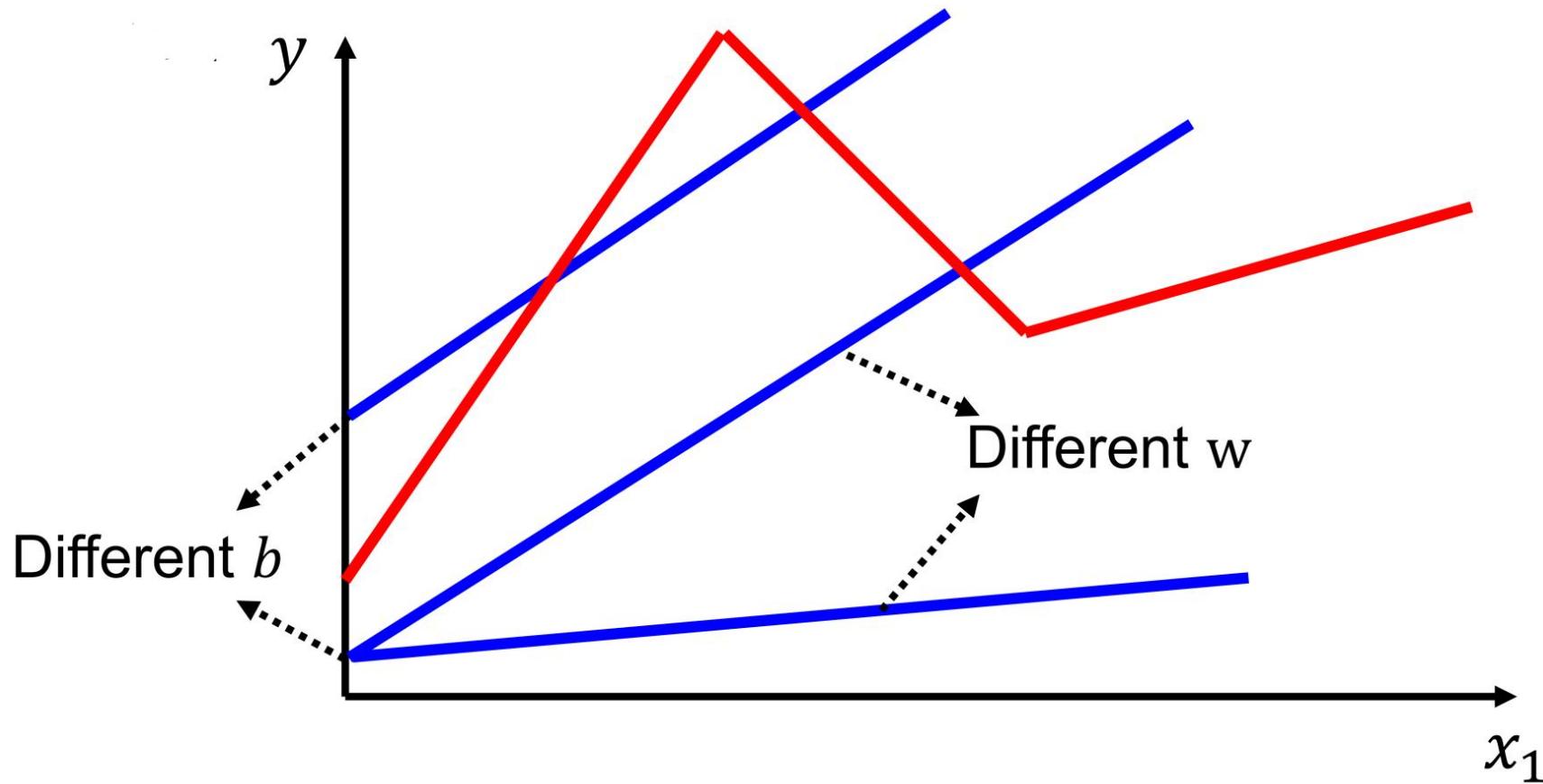


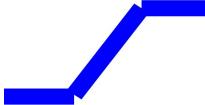
- 非线性决策边界

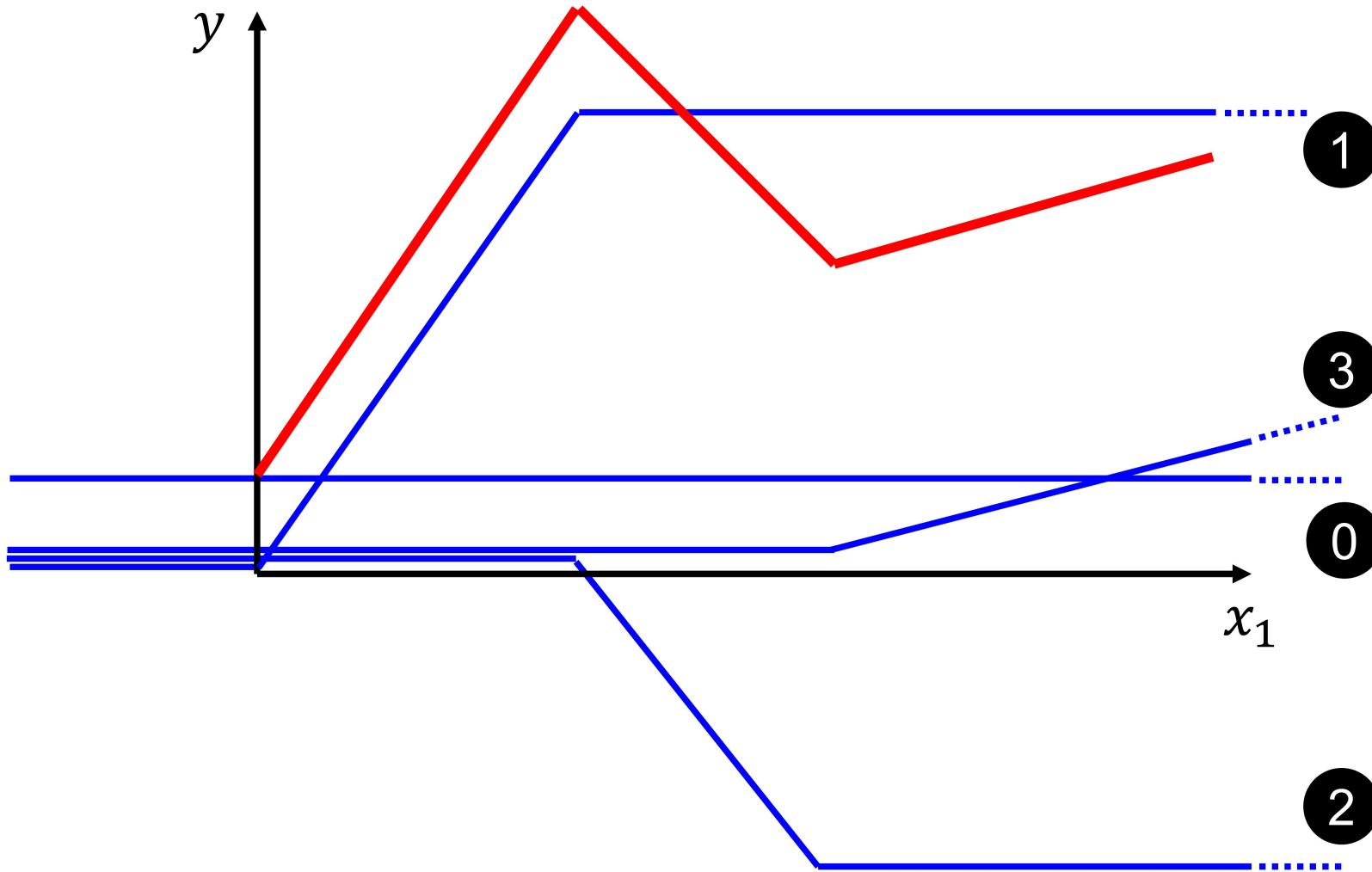


线性模型过于简单.....我们需要更复杂的模型

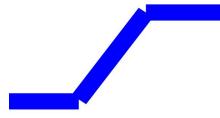
线性模型有严重的局限性：更灵活的模型！——bias(偏差)



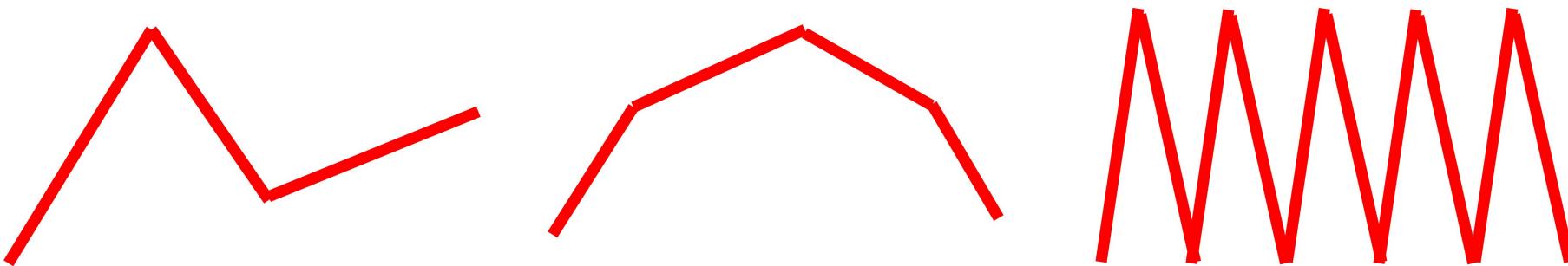
红色的折线 = 常数 + 一组  的和



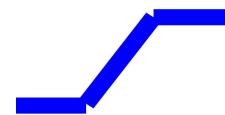
所有分段线性曲线 = 常数 + 一组



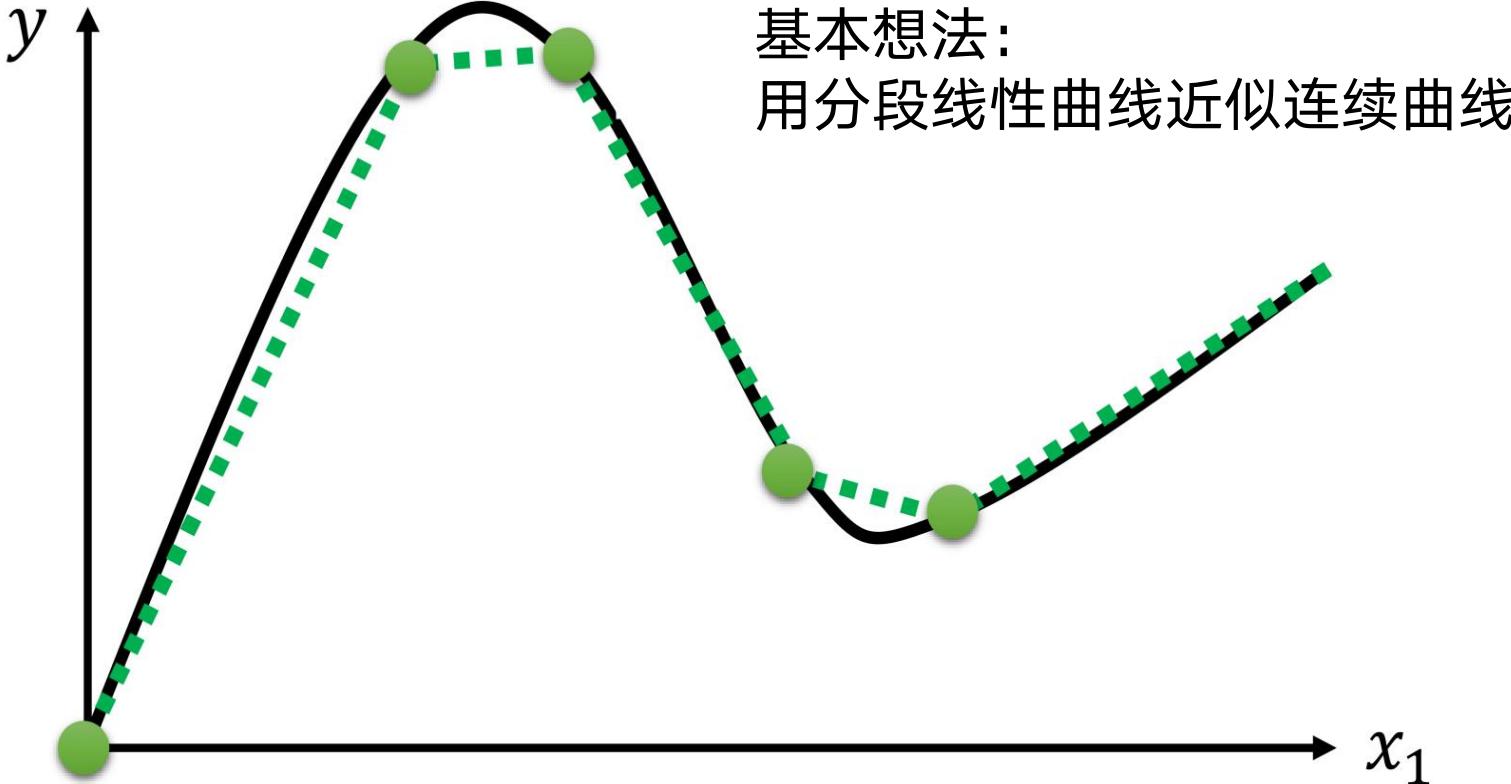
的和



曲线更多段就需要更多的



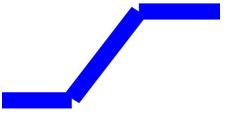
超越分段线性?



如何构建具备非线性建模能力的模型？

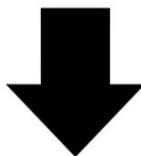
基本思想：使用基函数将线性模型非线性化

$$[x] \rightarrow [x, x^2, x^3]$$

红色的折线 = 常数 + 一组  的和

如何表示这个函数？

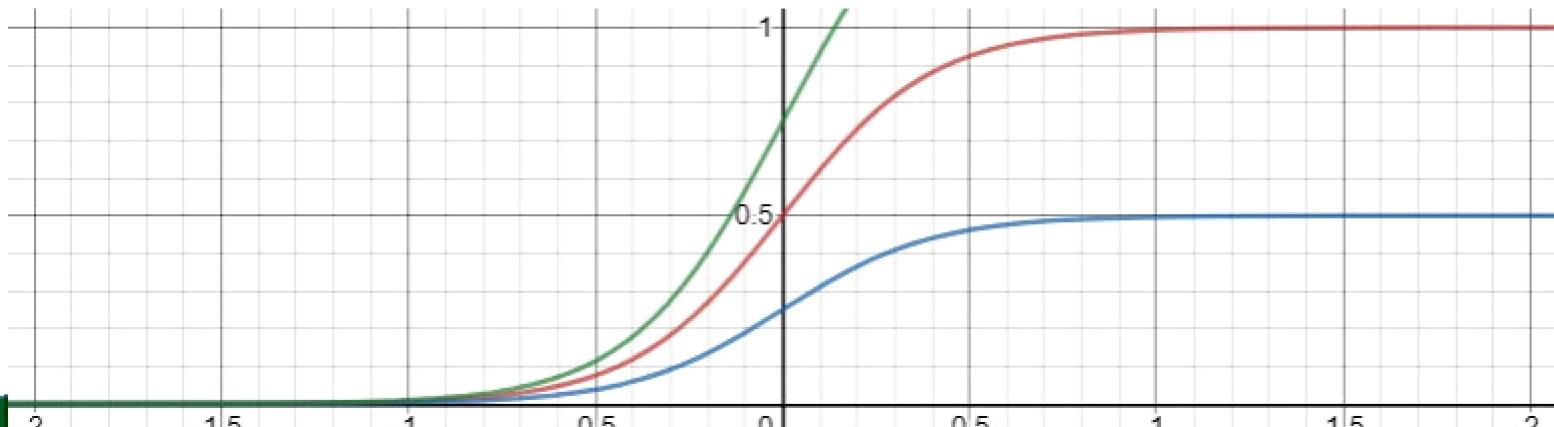
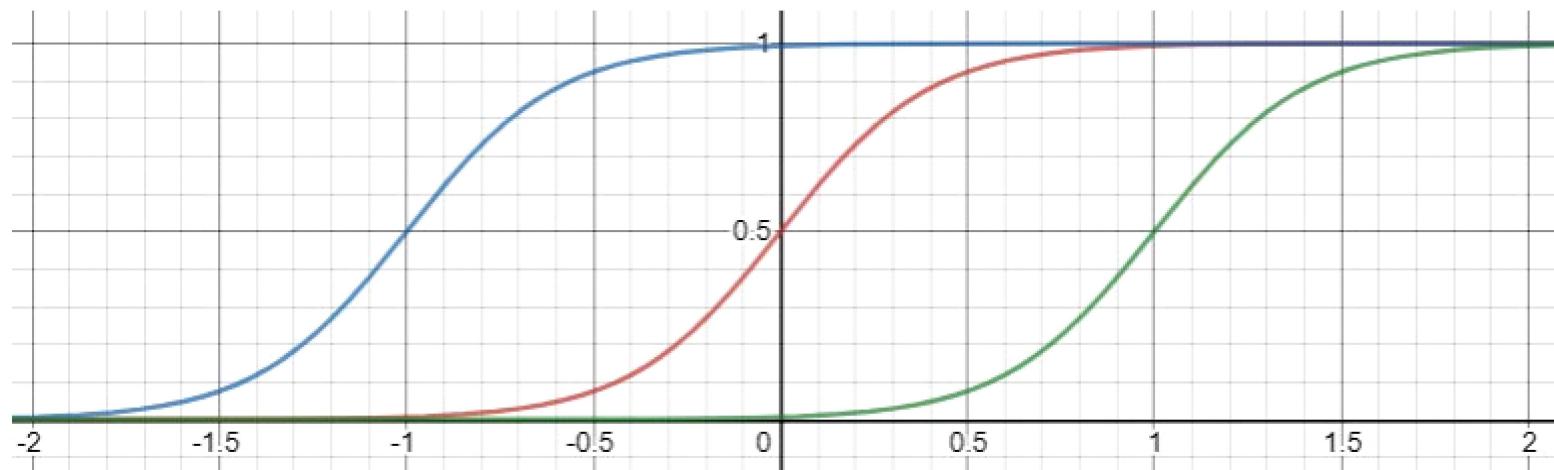
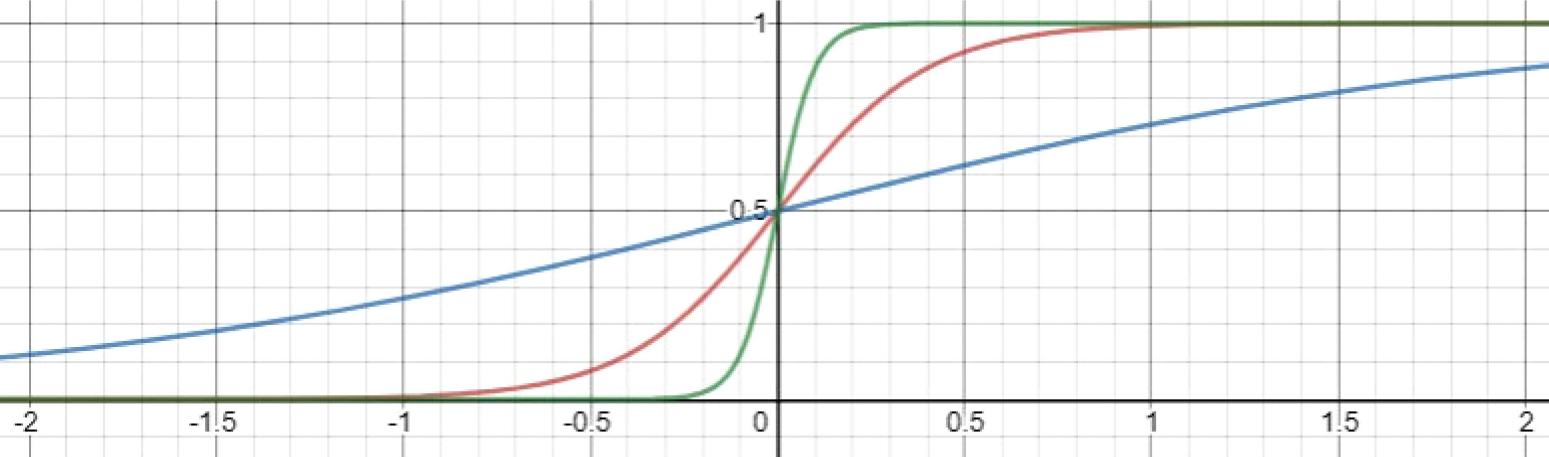
Hard Sigmoid



sigmoid函数

$$\begin{aligned}y &= \frac{1}{1 + e^{-(\textcolor{red}{b} + \textcolor{blue}{w}x_1)}} \\&= \text{sigmoid}(\textcolor{red}{b} + \textcolor{blue}{w}x_1)\end{aligned}$$





通过基函数进行非线性化

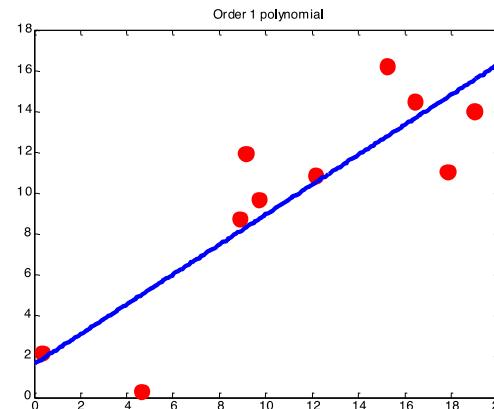
- 通过多项式变换特征：

$$[x] \rightarrow [x, x^2, x^3]$$

单个特征被扩展为 3 个特征

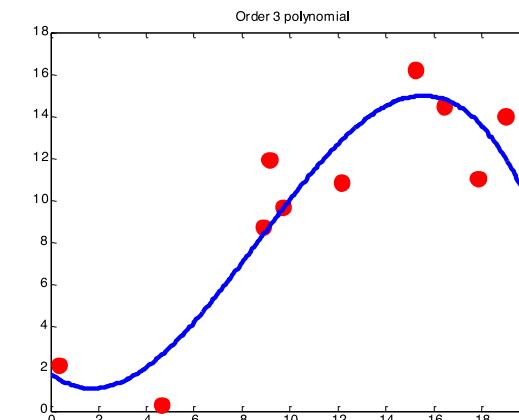
- 使用原始特征的模型

$$\begin{aligned}f(x) &= w_0 + w_1 x \\&= [1, x]w\end{aligned}$$



- 使用扩展特征的模型

$$\begin{aligned}f(x) &= w_0 + w_1 x + w_2 x^2 + w_3 x^3 \\&= \Phi(x)w\end{aligned}$$



- 通常，这种变换可以表示为：

$$[x_1, x_2, \dots, x_m] \rightarrow [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x})] \triangleq \boldsymbol{\phi}(\mathbf{x})$$

$\phi_k(\mathbf{x})$ 可以是任何能够产生有意义特征的函数，例如：

$$\sqrt{x}, \quad \log x, \quad \frac{1}{x}, \quad x_1 + x_2, \quad x_1 - x_2, \quad x_1 x_2$$

- 非线性化后的模型现在变为：

$$f(\mathbf{x}) = \phi(\mathbf{x})\mathbf{w}$$

这被称为基函数模型

基函数模型关于 \mathbf{x} 是非线性的，但关于模型参数 \mathbf{w} 仍然是线性的

- 使用非线性变换后的特征 $\Phi(\mathbf{x})$, 回归问题的最优模型参数 \mathbf{w}^* 通过优化损失函数获得:

$$L(\mathbf{w}) = \frac{1}{N} \|\Phi(\mathbf{X})\mathbf{w} - \mathbf{y}\|^2$$

其中 $\Phi(\mathbf{X}) \triangleq \begin{bmatrix} \Phi(\mathbf{x}^{(1)}) \\ \vdots \\ \Phi(\mathbf{x}^{(N)}) \end{bmatrix}$

- 使用符号 $\Phi = \Phi(X)$ ，最优模型参数 w^* 为：

$$w^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

与线性回归相同，**只是将 X 替换为 Φ**

- 我们也可以使用数值方法（例如梯度下降）来获得最优解

- 对于使用基函数的分类问题，交叉熵损失变为：

$$L(\mathbf{W}) = -\frac{1}{N} \sum_{\ell=1}^N \sum_{k=1}^K y_k^{(\ell)} \log [softmax_k(\phi(x^{(\ell)})\mathbf{W})]$$

最优的 \mathbf{W}^* 只能通过数值方法获得

- 将 $\phi(x^{(\ell)})$ 表示为 $\phi^{(\ell)}$ ，可以推导出梯度等于

$$\frac{\partial L(\mathbf{W})}{\partial w_j} = \frac{1}{N} \sum_{\ell=1}^N (\text{softmax}_j(\phi^{(\ell)}\mathbf{W}) - y_j^{(\ell)}) \phi^{(\ell)T}$$

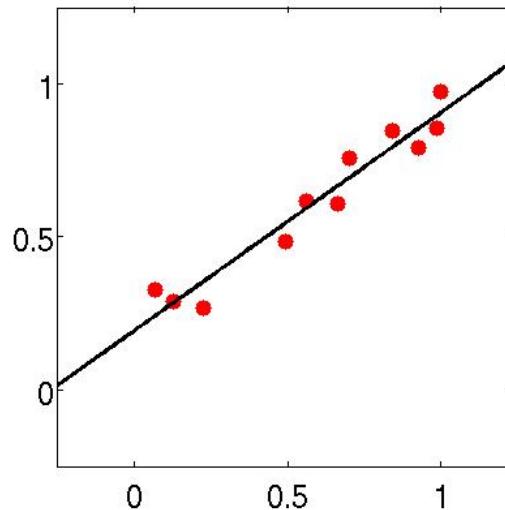
与多类别逻辑回归相同，只是将 $x^{(\ell)}$ 替换为 $\phi^{(\ell)}$

课程大纲

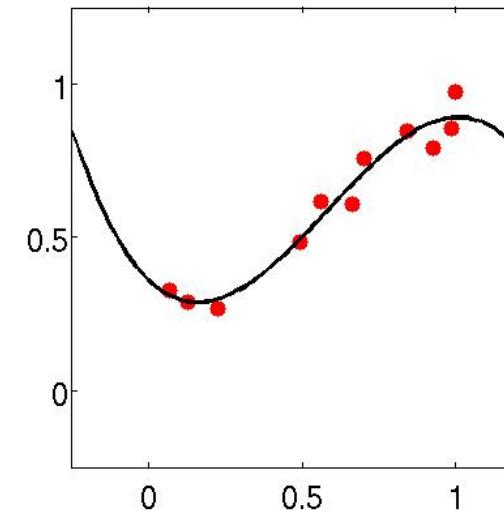
- 模型非线性化
- 过拟合
- 模型选择
- 正则化

过拟合

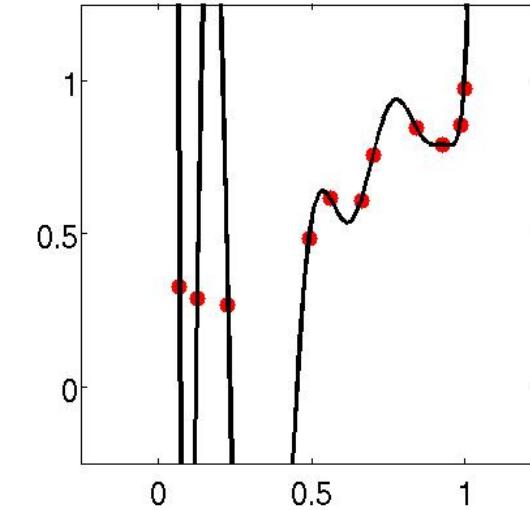
- 高维特征 $\phi(x)$ 往往能更好地拟合训练数据



1阶



3阶

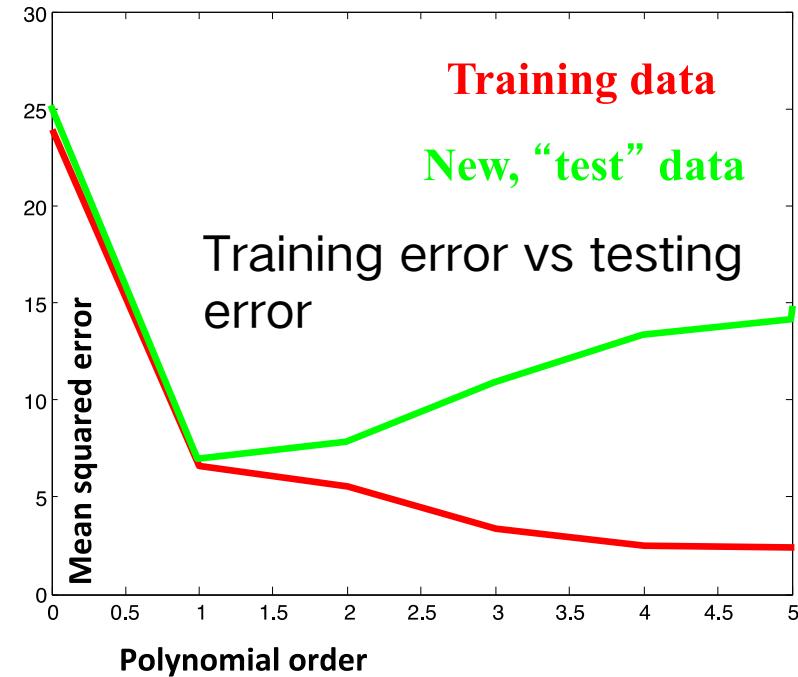
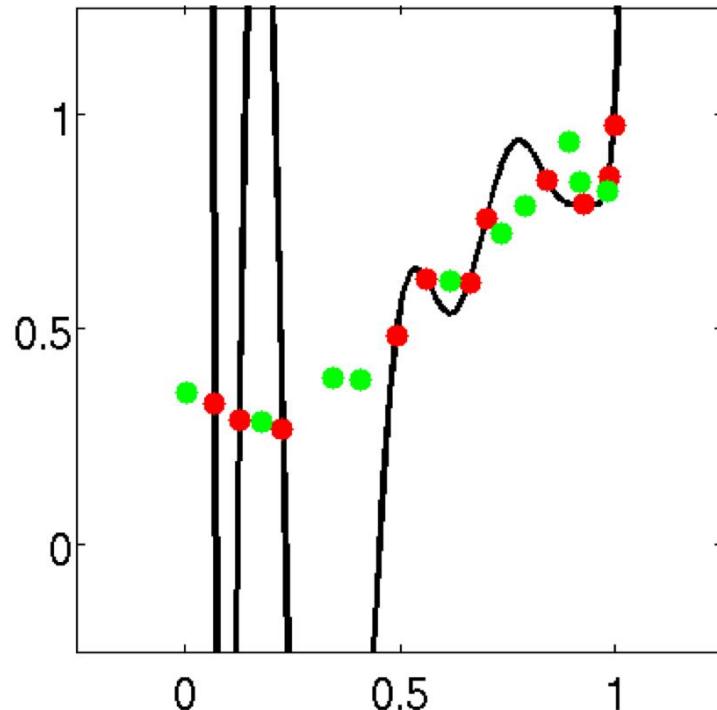


5阶

哪个模型更好？？

- 从拟合训练数据的角度来看，模型阶数越高，拟合效果看起来肯定越好

- 但是高阶模型在测试数据上的表现可能很差



模型在新数据上表现良好的能力被称为模型的泛化能力

过拟合

训练数据上损失小，测试数据上损失大
为什么？

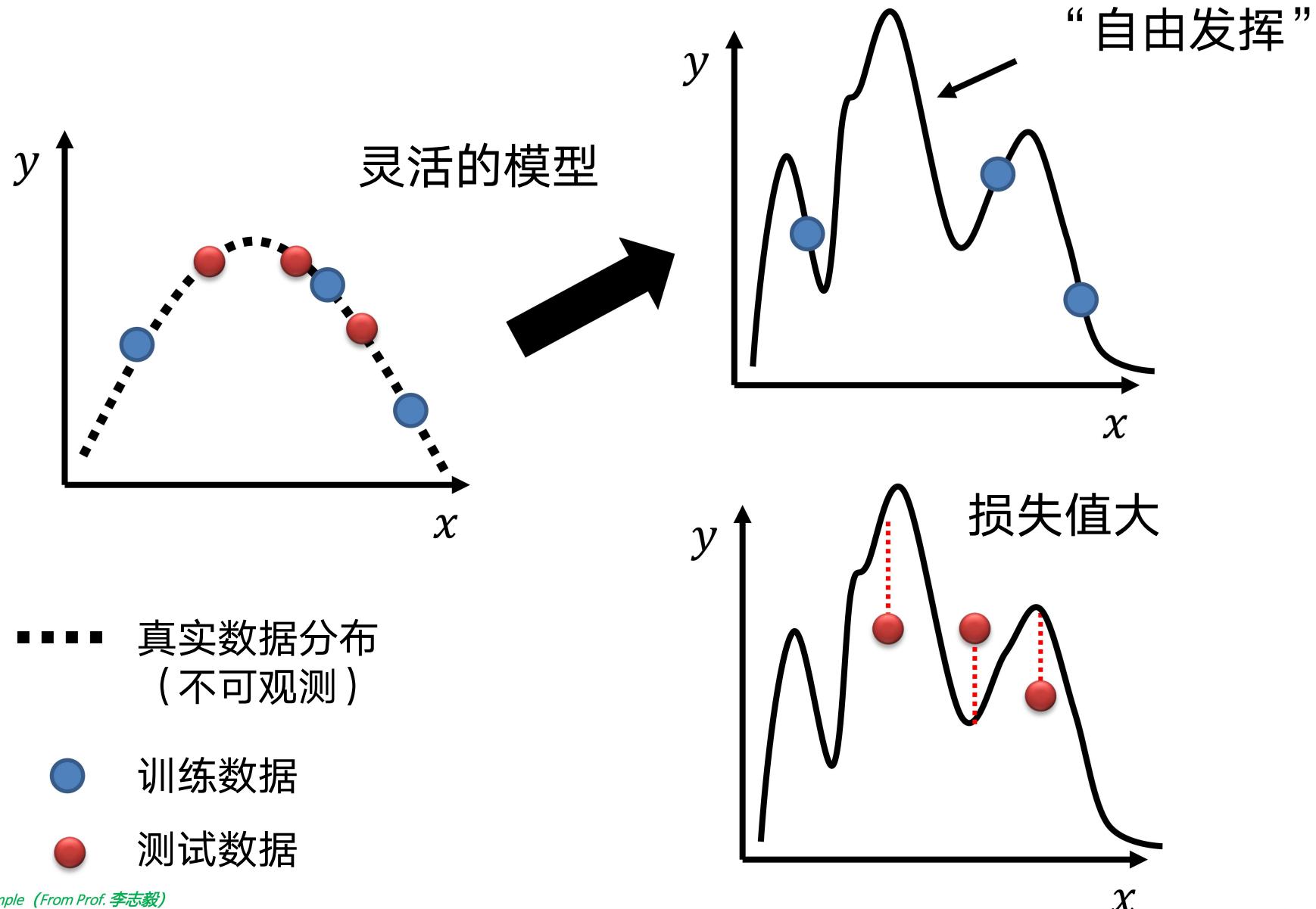
一个极端的例子：

Training data: $\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \dots, (x^N, \hat{y}^N)\}$

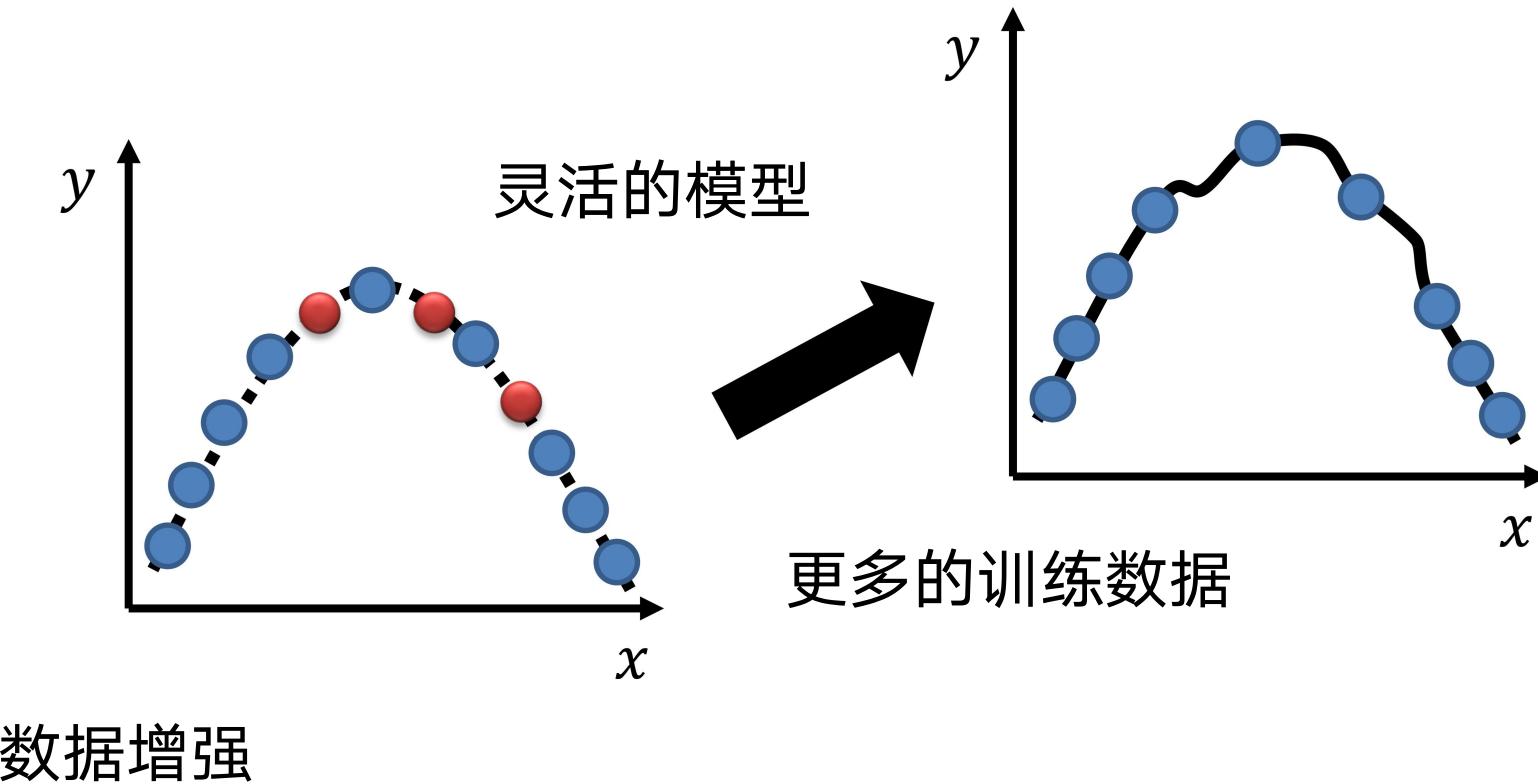
$$f(x) = \begin{cases} \hat{y}^i & \exists x^i = x \\ random & otherwise \end{cases} \quad \text{Less than useless ...}$$

此函数的训练损失为零，但测试损失较大

过拟合 Overfitting

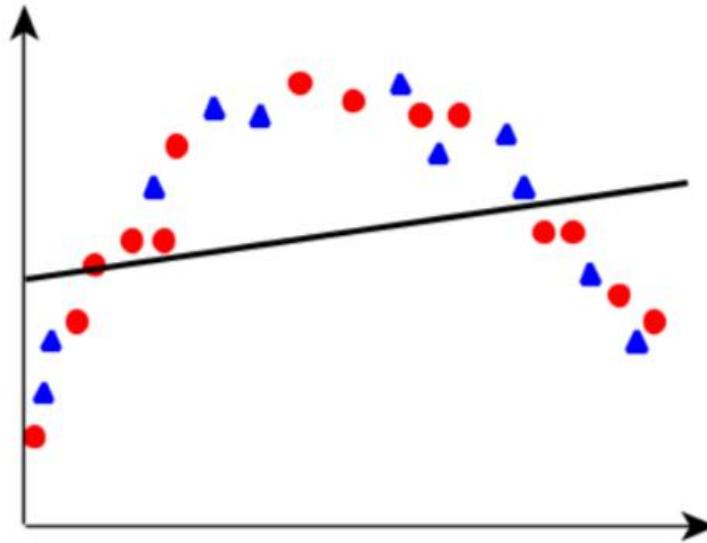


过拟合 How to Solve Overfitting

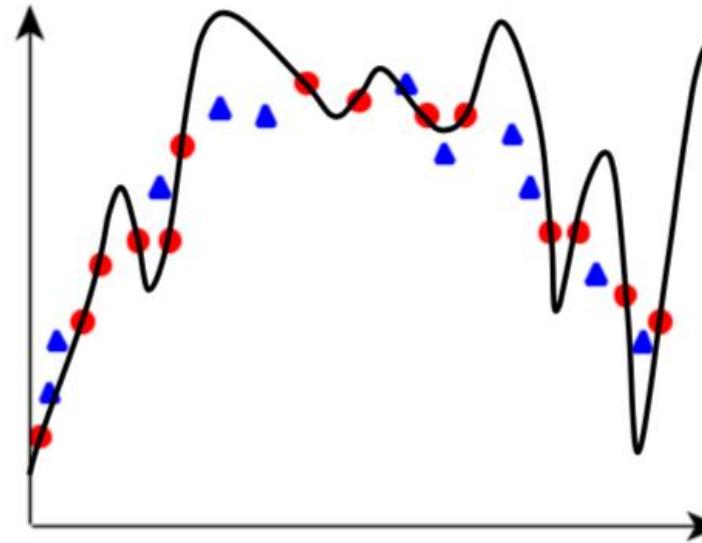


Basic Example (From Prof. 李志毅)

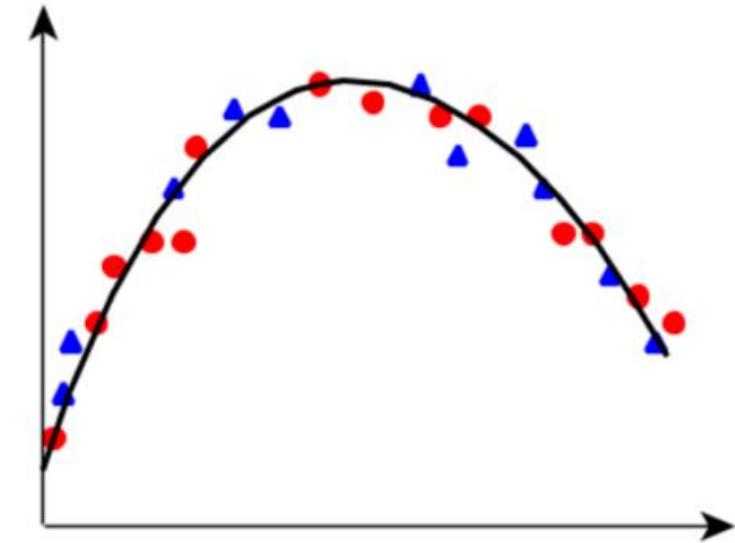
欠拟合



欠拟合

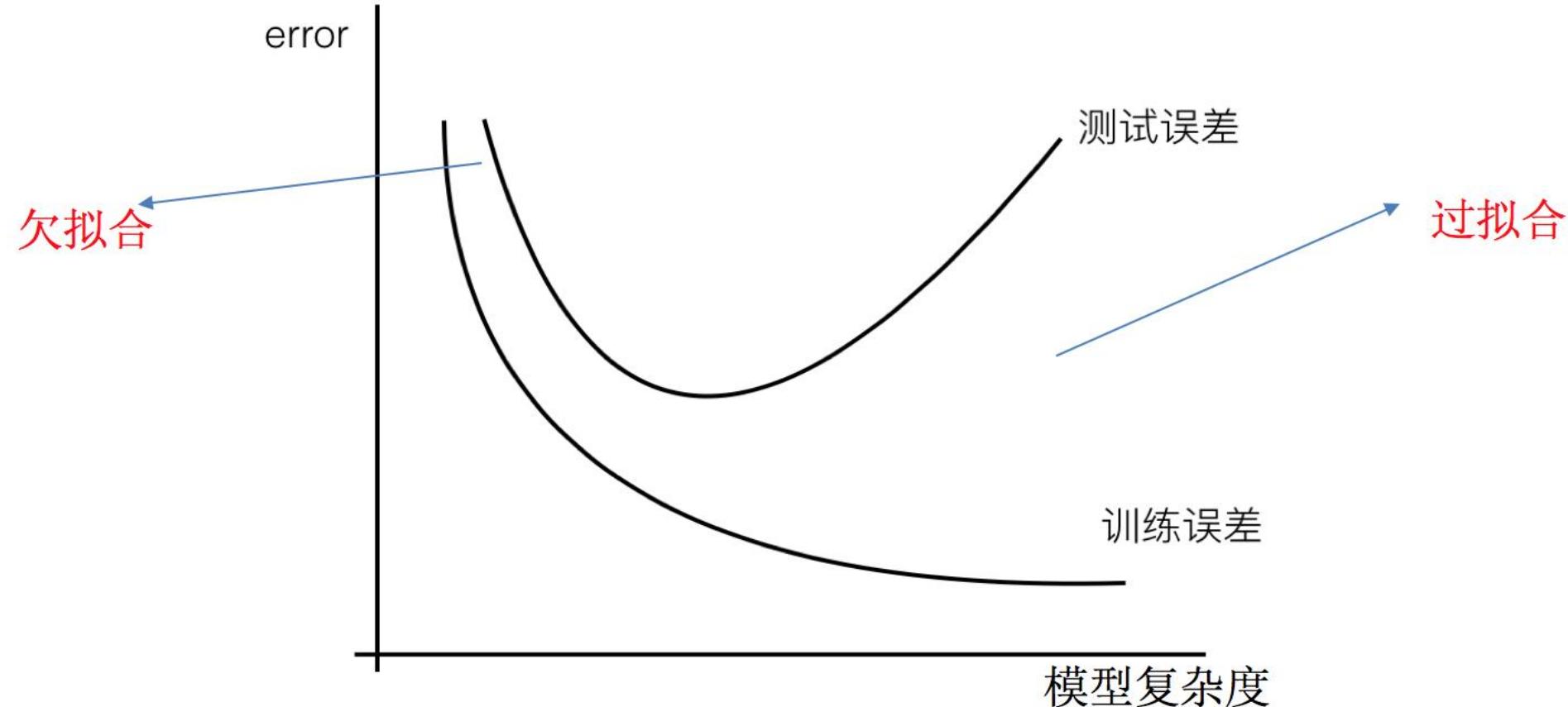


过拟合



好的拟合

拟合情况与模型复杂度



模型复杂度

- 每个模型都对应着一个复杂度
- 但很难给出一个精确的表达式来描述模型复杂度
- 一般来说，模型复杂度取决于参数的数量，参数越多，模型越复杂
- 为了让模型表现良好，我们应该平衡模型复杂度和表征能力

课程大纲

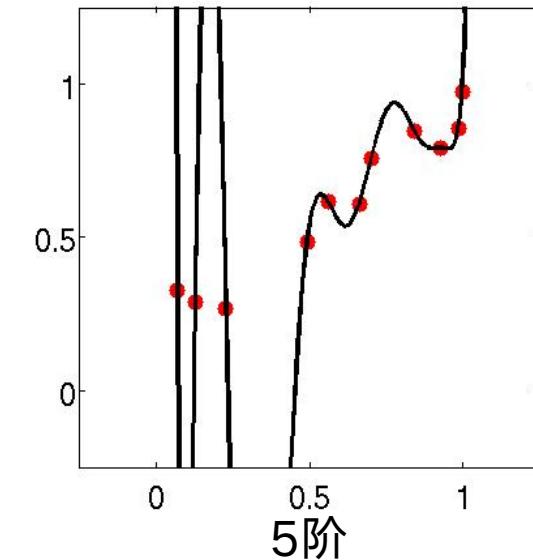
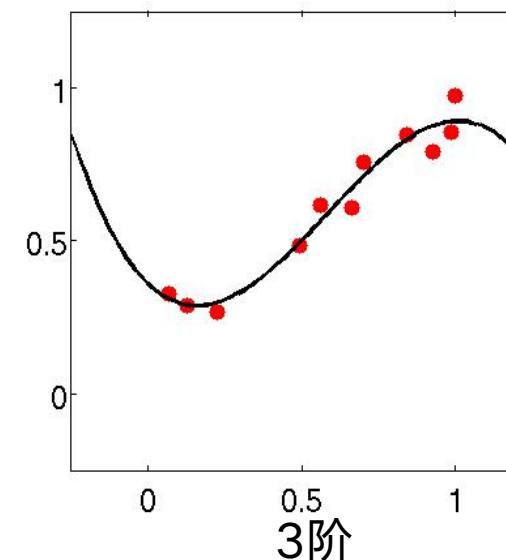
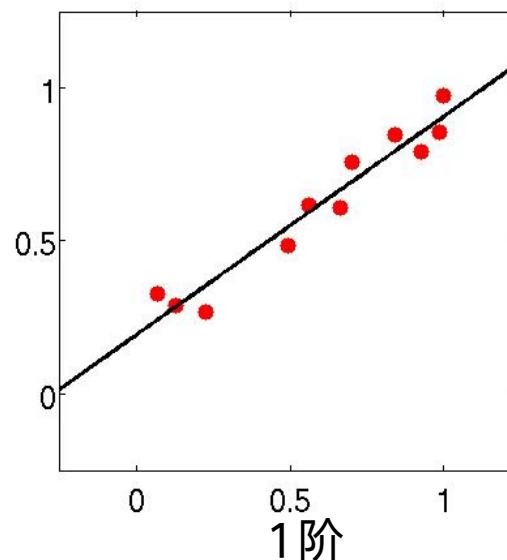
- 模型非线性化
- 过拟合
- 模型选择
- 正则化

模型选择

- 模型选择：给定一组模型 $\{\mathcal{M}_1, \dots, \mathcal{M}_m\}$ ，选择在未见过的测试数据上表现最好的那个

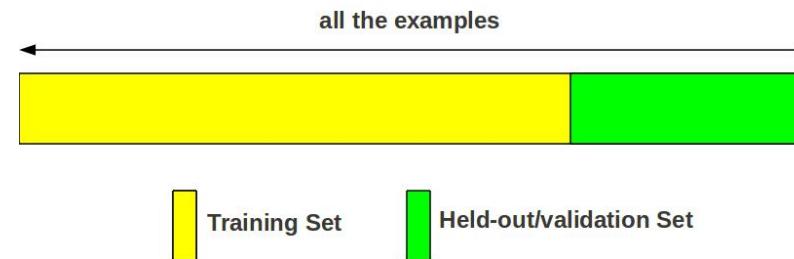
候选模型可以是同一种类型，也可以是不同类型

不能基于模型在训练数据上的表现来选择模型



验证集

- 拿出训练数据的一部分（20% ~ 30%）作为验证集，剩下的作为训练数据

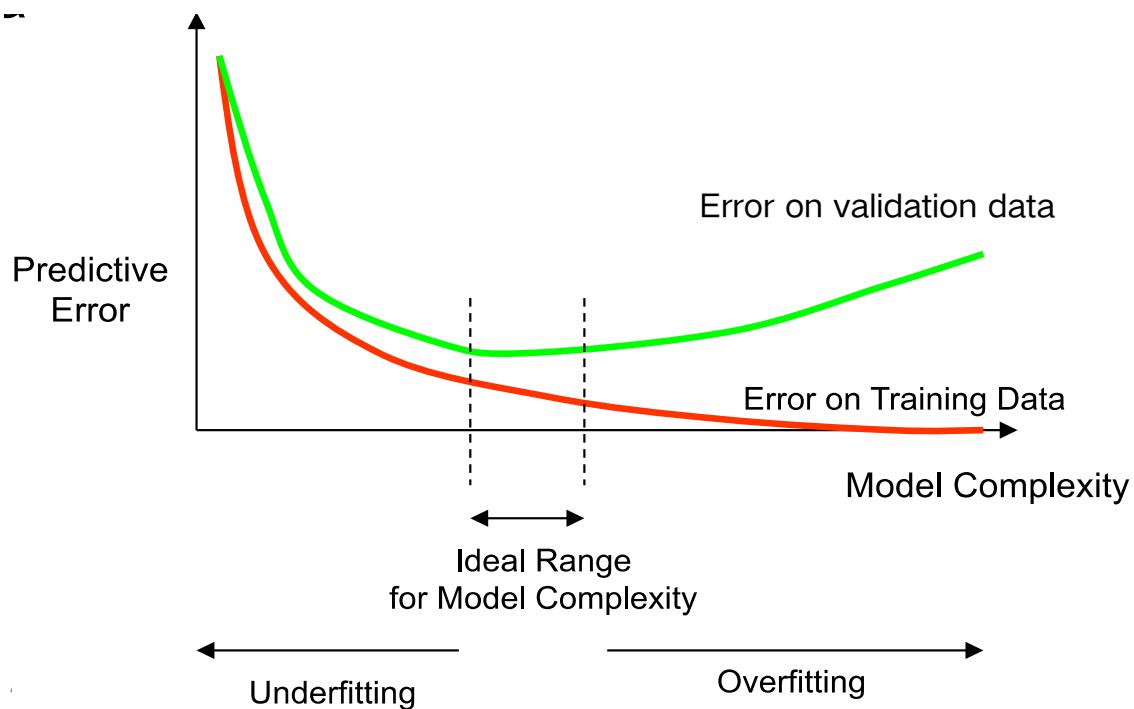


训练集和验证集都**不能包含测试样本**

验证集不能太小。为什么？

- 在训练集上训练模型，同时在留出的验证集上评估模型
- 选择在验证集上表现最好的模型

- 训练集和验证集上的预测误差



- 如果随着模型复杂度的增加，验证误差减小，则表明模型欠拟合
- 否则，则意味着模型过拟合

留一验证

- 普通验证方法存在的问题

训练数据通常是稀缺的。如果留出很大一部分用于验证，就没有足够的数据样本用于训练

- 一个折中方案：**留一法验证**

对于数据集中的每个样本，执行以下步骤：

- 将该样本作为验证集，其余所有样本作为训练集，训练模型并评估其在验证集上的性能
- 重复上述步骤，直到每个样本都被用作一次验证集
- 由于几乎整个数据集都用于训练（除了一个样本），因此模型可以充分利用所有数据，这通常可以提供更稳定的性能估计，适用于小数据集。



交叉验证

- 普通验证方法存在的问题

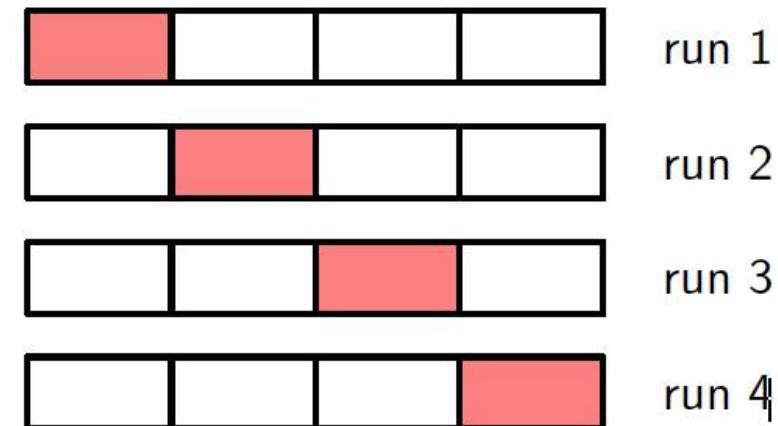
训练数据通常是稀缺的。如果留出很大一部分用于验证，就没有足够的数据样本用于训练

- 一个常用方案：*K 折交叉验证*

 - 将整个训练数据集平均划分成 K 个子集

 - 在 $(K - 1)$ 个子集上进行训练，
在剩下的一个子集上进行评估

 - 重复上述步骤 K 次，每次都使用不同的子集进行验证



信息准则

- Akaike 信息量准则 (AIC)

$$AIC = 2M - 2\log(l)$$

- M 是参数数量
 - l 是对数似然

- 贝叶斯信息量准则 (BIC)

$$AIC = M\log N - 2\log(l)$$

- N 是训练数据样本的数量

这些准则只能用于概率模型，因为它们需要用到对数似然 l

课程大纲

- 模型非线性化
- 过拟合
- 模型选择
- 正则化

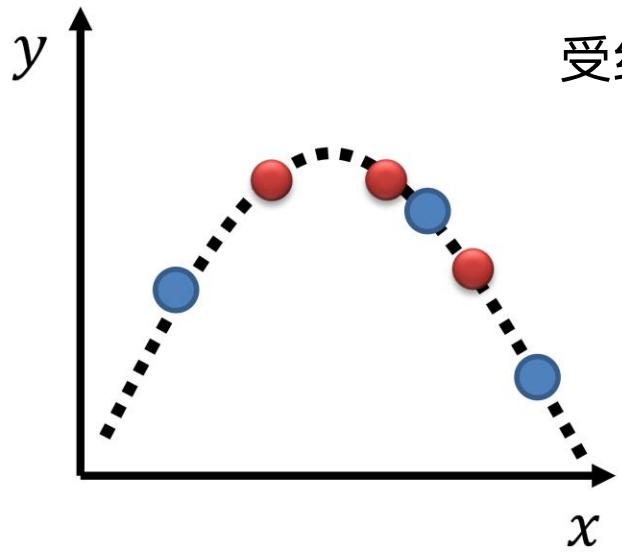
- 除了拟合训练数据之外，对参数施加一些先验偏好，例如：

$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

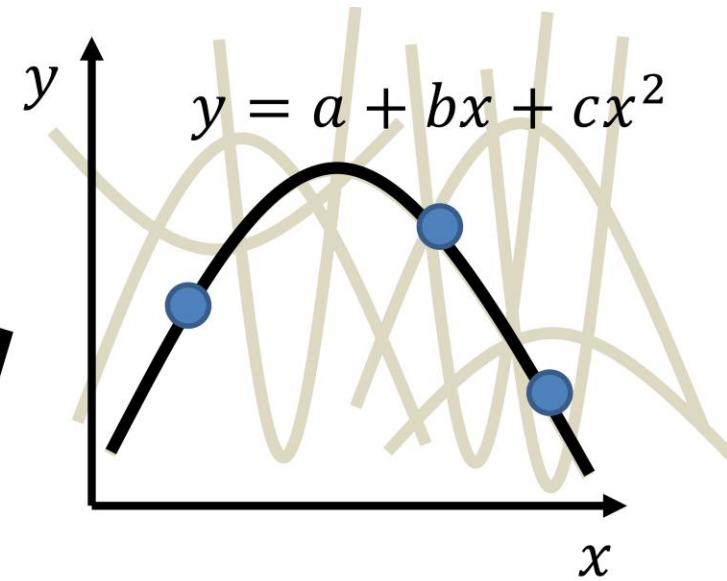
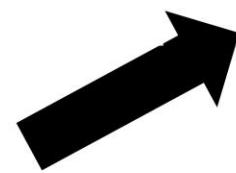
- $L(\mathbf{w})$ 是原始的回归或分类损失
- $\|\mathbf{w}\|_2 = (\sum_{k=1}^K w_k^2)^{\frac{1}{2}}$ 是 L_2 范数
- λ 是用于控制 $\|\mathbf{w}\|^2$ 影响的超参数

L_2 正则化

- L_2 正则化的特性
 - 倾向于将模型参数向零收缩——如果 w 小于 1，那么 w^2 会更小
 - λ 越大，对较小 w 值的偏好就越强
 - 在实现上， L_2 正则化通常被称为权重衰减（Weight Decay），在框架如 PyTorch 中可以通过设置优化器的“weight_decay”参数轻松实现



受约束模型

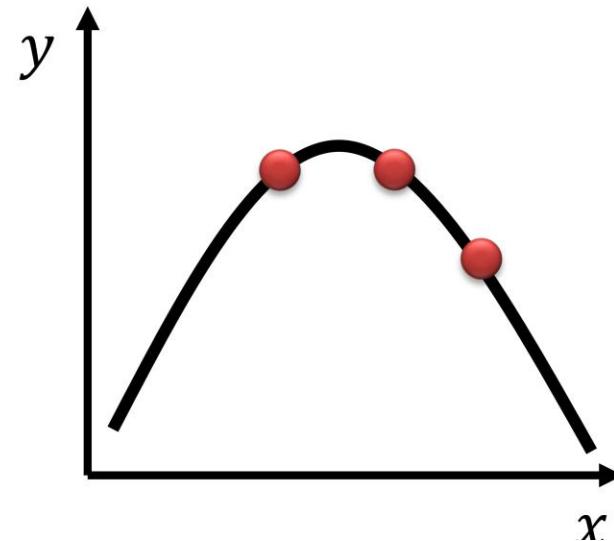


····· 真实数据分布
(不可观测)

● 训练数据

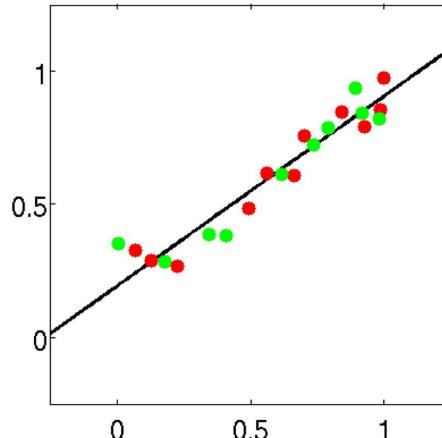
● 测试数据

Basic Example (From Prof. 李志毅)

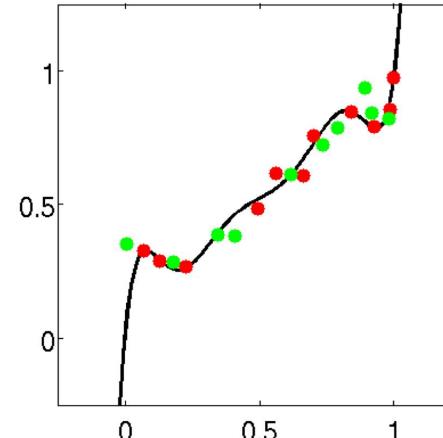


- 正则化影响的可视化

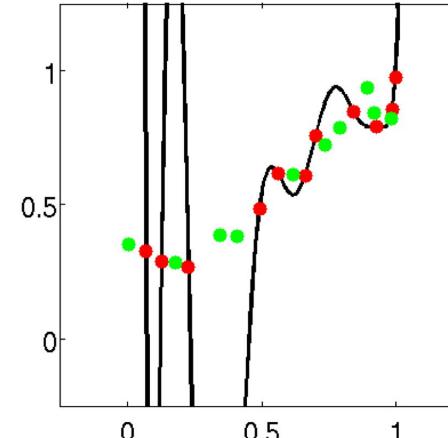
- 无正则化



1阶

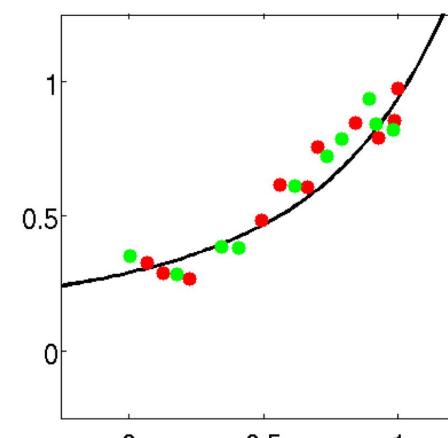
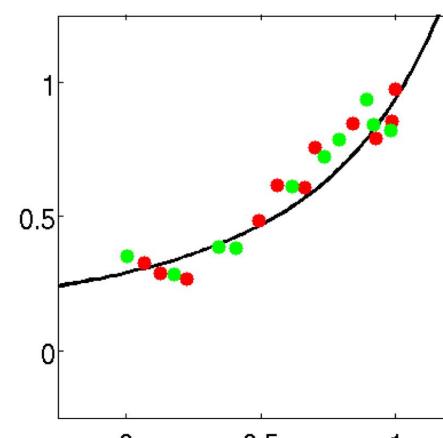
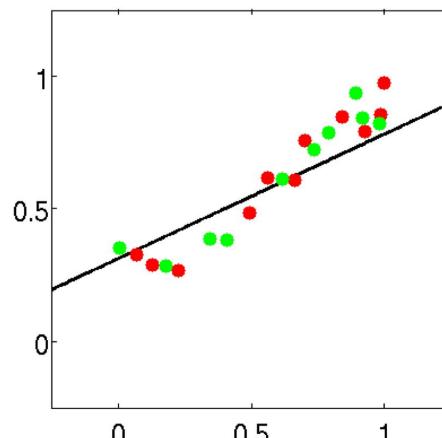


3阶



5阶

- 采用 $\lambda = 1$ 的 L_2 正则化

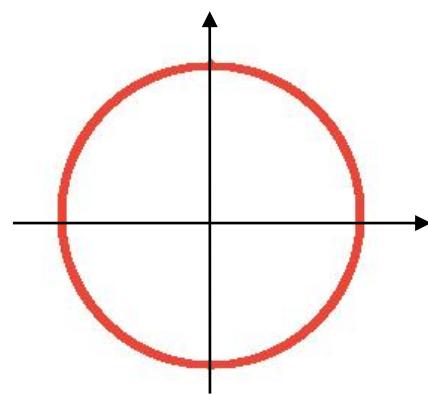


- 另一种广泛使用的正则化是 L_1 正则化

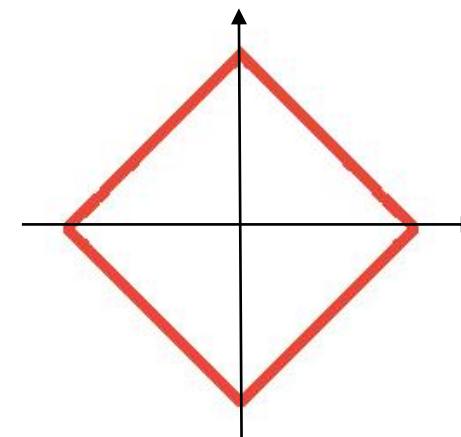
$$\tilde{L}(\mathbf{w}) = L(\mathbf{w}) + \lambda \|\mathbf{w}\|_1$$

其中 $\|\mathbf{w}\|_1 \triangleq \sum_{k=1}^K |w_k|$ 是 L_1 范数

- L_2 范数和 L_1 范数的等高线

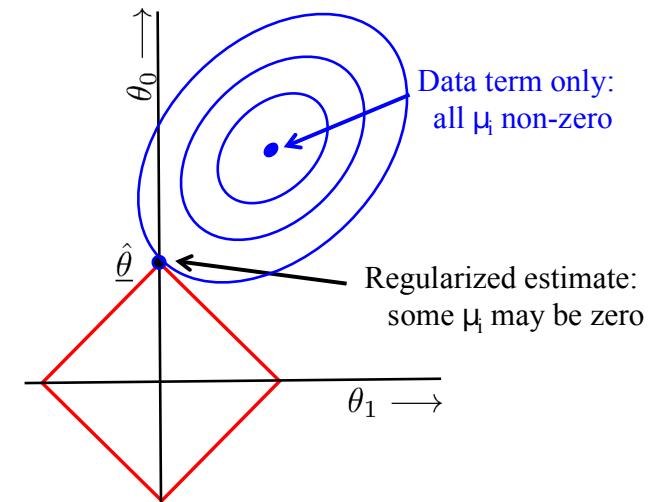
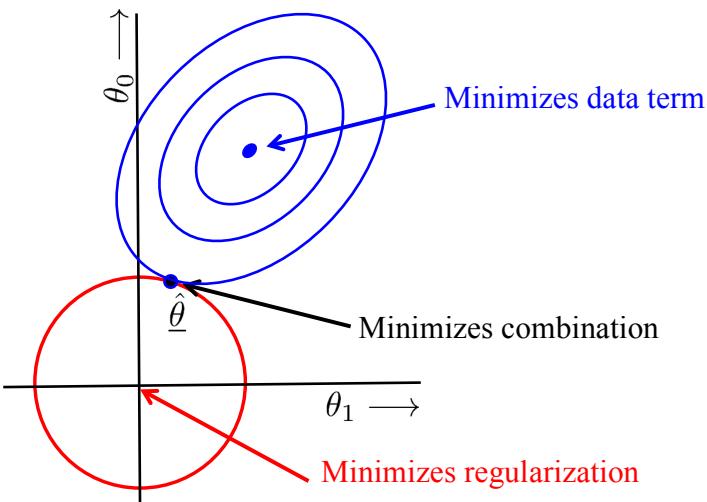


L_2 范数



L_1 范数

- 与 L_2 正则化类似， L_1 正则化也倾向于使模型参数具有较小的值
- 但是， L_1 正则化通常会**导致 w 出现稀疏解**，即 w 中的许多元素为零





机器学习与数据挖掘

支持向量机

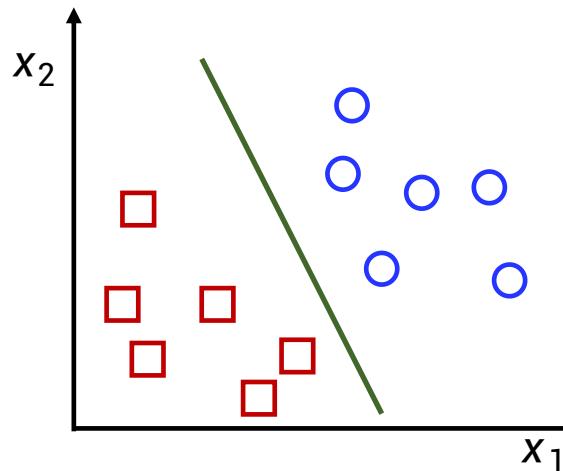


课程大纲

- 线性分类器的决策边界
- 线性最大间隔分类器
- 软线性最大间隔分类器
- 支持向量机
- 与逻辑回归的关系

线性分类器中的决策边界

- 在线性分类器中，决策边界总是一个超平面。目标是找到能够分离不同类型样本的超平面

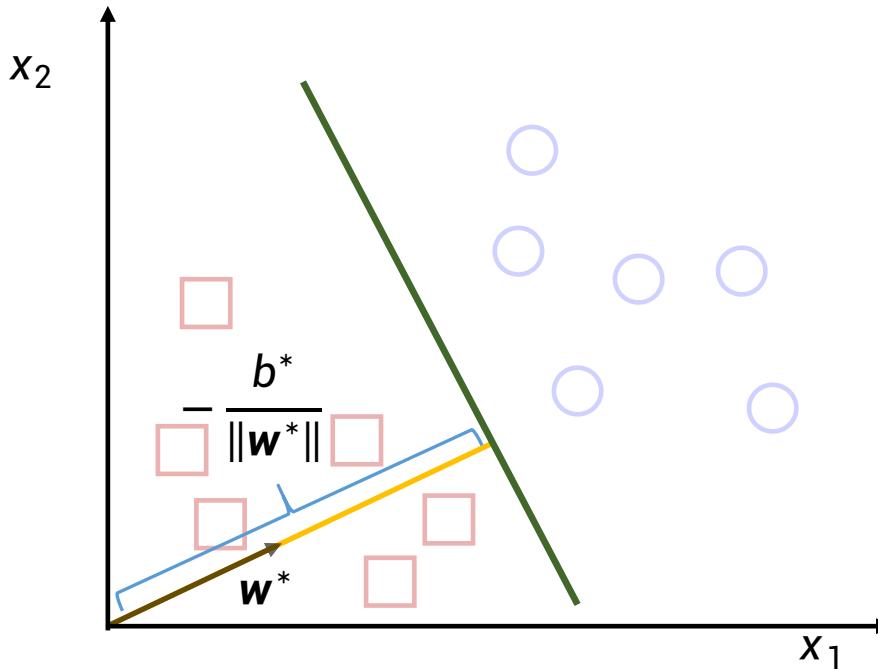


- 逻辑回归
 - 决策边界超平面是通过最小化交叉熵损失来找到的

$$L(\mathbf{w}, b) = -y \log(\sigma(\mathbf{w}^T \mathbf{x} + b)) - (1 - y) \log(1 - \sigma(\mathbf{w}^T \mathbf{x} + b))$$

➤ 当得到最优的 w^* 和 b^* 时，超平面由下式的 x 组成：

$$\{x | w^{*T}x + b^* = 0\}$$



1) 超平面与向量 w^* 垂直

2) 从原点到平面的距离是 $-\frac{b^*}{\|w^*\|}$

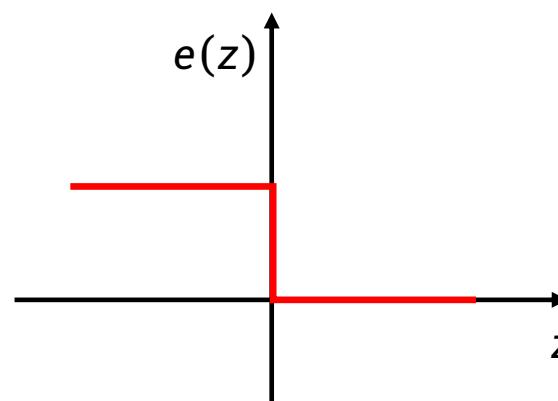
- 理想分类器

- 超平面是通过最小化损失来确定的

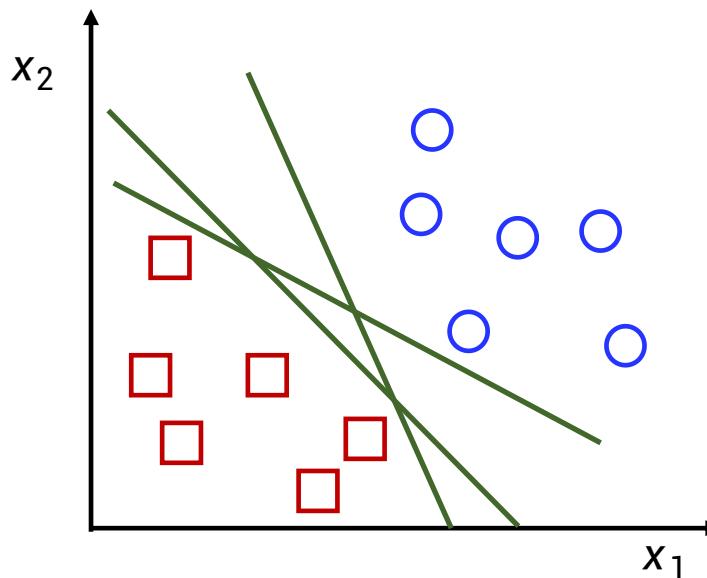
$$L(\mathbf{w}, b) = \sum_{\ell=1}^N e(y^{(\ell)}(\mathbf{w}^T \mathbf{x}^{(\ell)} + b))$$

$L(\mathbf{w}, b)$ 代表被错误分类的样本数量

- $y \in \{-1, 1\}$
- 如果 $z \geq 0$, 则 $e(z) = 0$; 否则 $E[z] = 1$



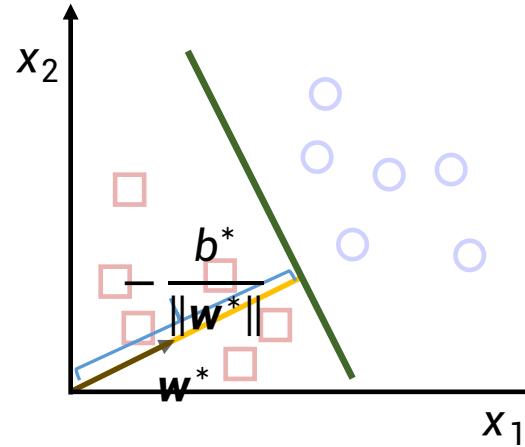
- 如果样本是线性可分的，将会有无数个理想分类器，它们由 w^* 和 b^* 决定
- 每一对 w^* 和 b^* 都对应一个超平面



上述所有超平面都能使损失降为零

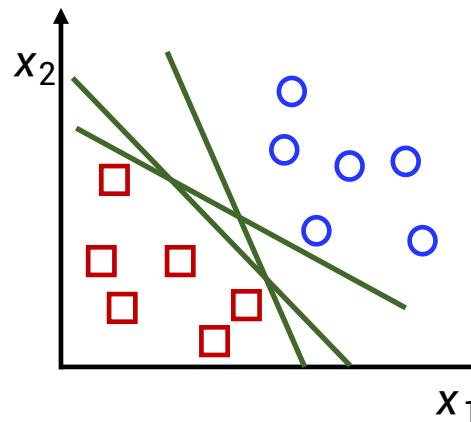
哪个超平面是最好的？

- 下面的超平面从最小化交叉熵损失的角度来看是最优的



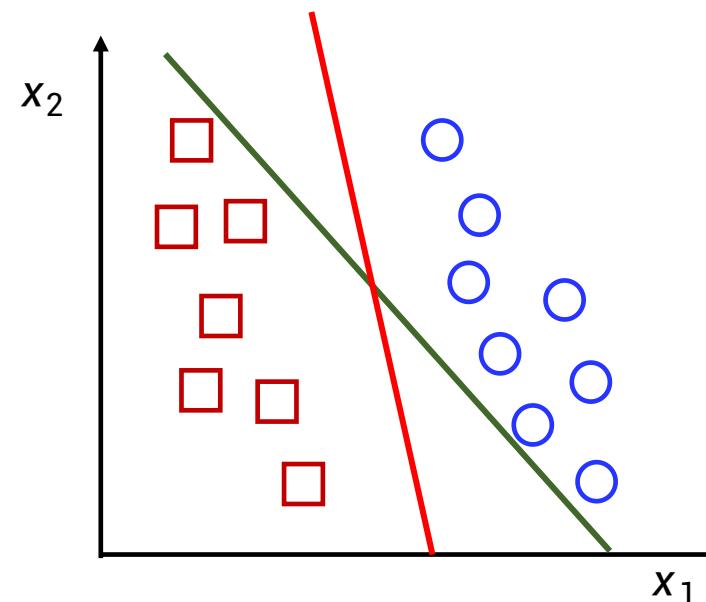
哪个超平面是最好的？

- 下面的超平面从**最小化误分类样本数量**的角度来看是最优的



- 这些超平面都是在 **训练样本** 上进行评估的
- 但我们真正需要的是在 **(未见过的) 测试数据** 上的表现

根据我们的直觉，下面的哪个超平面更有可能在测试数据上产生更好的结果？

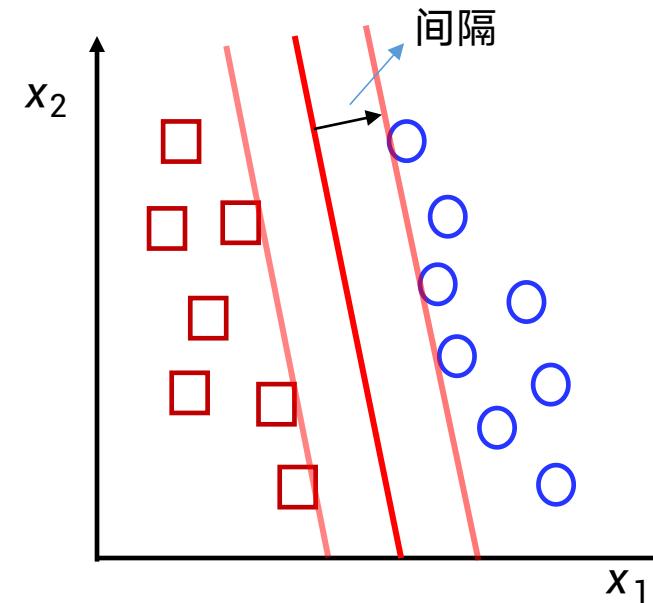


课程大纲

- 线性分类器的决策边界
- 线性最大间隔分类器
- 软线性最大间隔分类器
- 支持向量机
- 与逻辑回归的关系

最大间隔目标

- 为了在未见过的数据上表现良好，直觉是找到一个能尽可能扩大间隔的超平面



当间隔较大时，我们可以预期未见过的样本有更高的几率被正确分类

如何表示间隔？

- 样本 x 到超平面 \mathcal{H} 的距离。记 $w^T x + b = h(x)$

➤ 每个 x 都可以被分解为：

$$x = m_1 + m_2$$

- m_1 在超平面 \mathcal{H} 上，即 $w^T m_1 + b = 0$
- $m_2 \perp \mathcal{H}$ 及 $m_2 \parallel w$

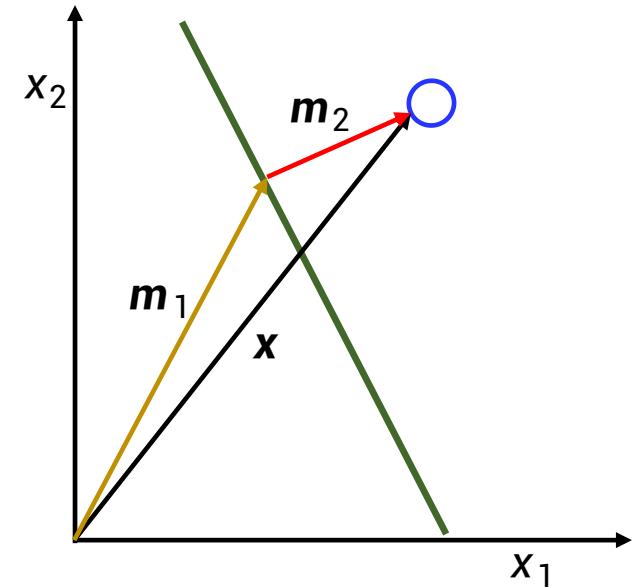
➤ 因此，我们有：

$$w^T x + b = w^T(m_1 + m_2) + b = w^T m_2 = h(x)$$

➤ 由于 $m_2 \parallel w$ ，我们可以写成：

$$m_2 = \gamma \cdot \frac{w}{\|w\|},$$

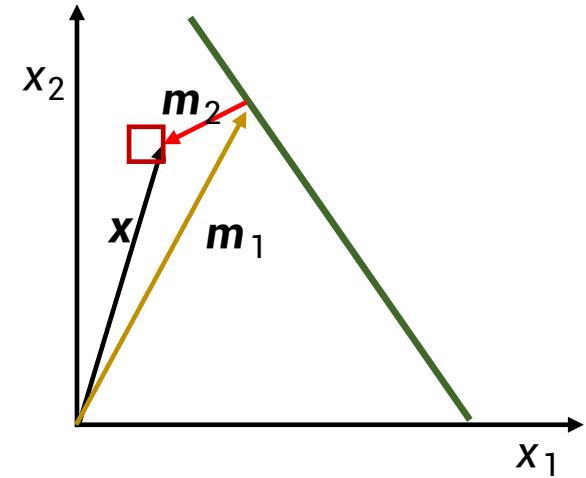
其中 $|\gamma|$ 代表 m_2 的长度



➤ 将 $m_2 = \gamma \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$ 代入 $h(\mathbf{x}) = \mathbf{w}^T \mathbf{m}_2$ 得到：

$$h(\mathbf{x}) = \gamma \cdot \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \quad \Rightarrow \quad \gamma = \frac{h(\mathbf{x})}{\|\mathbf{w}\|}$$

- 位于超平面上方的样本 \mathbf{x} 到超平面的距离为 $\frac{h(\mathbf{x})}{\|\mathbf{w}\|}$
- 位于超平面下方的样本 \mathbf{x} 到超平面的距离为 $-\frac{h(\mathbf{x})}{\|\mathbf{w}\|}$



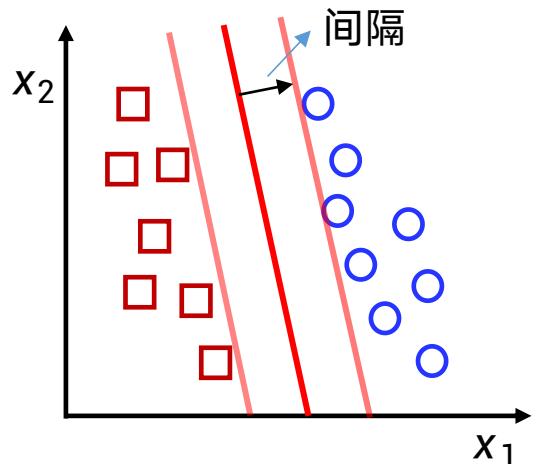
➤ 样本 (\mathbf{x}, y) 到超平面的距离由下式给出：

$$\frac{y \cdot h(\mathbf{x})}{\|\mathbf{w}\|} = \frac{y \cdot (\mathbf{w}^T \mathbf{x} + b)}{\|\mathbf{w}\|}$$

其中 $y \in \{-1, 1\}$

- 一个超平面在一个数据集上的间隔由最小距离给出，即：

$$\text{Margin} = \min_{\ell} \frac{y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b)}{\|\mathbf{w}\|}$$



- 因此，最大间隔分类器的目标是找到最优的 \mathbf{w}^* 和 b^* 来使间隔最大化，即：

$$\mathbf{w}^*, b^* = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{\ell} [y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b)] \right\}$$

但是，如何优化是未知的

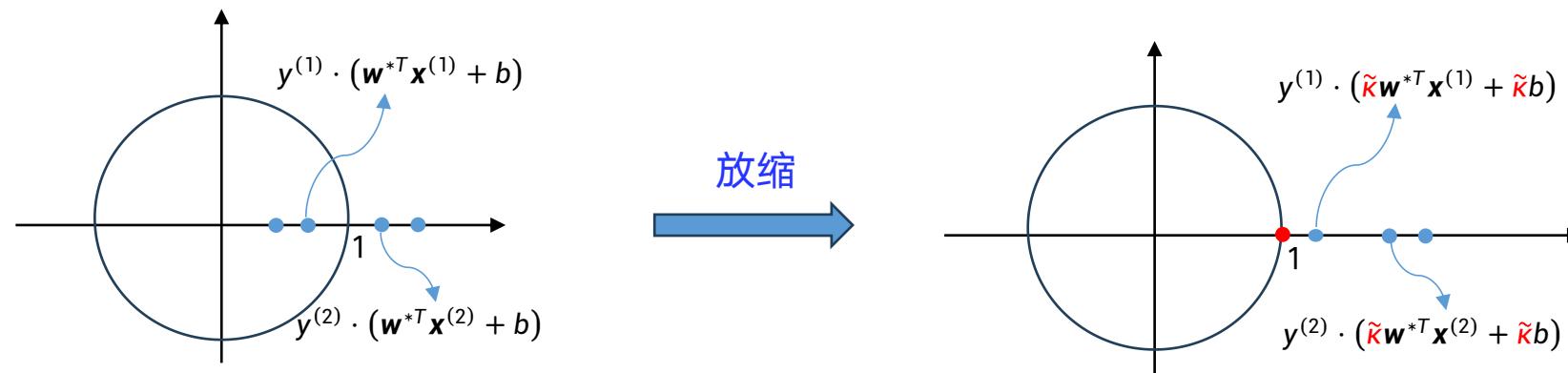
变换后的目标函数

- **基本思想：**优化一个与原问题具有相同最优解的目标函数

- 假设 w^* 和 b^* 是 $\frac{1}{\|w\|} \min_{\ell} [y^{(\ell)} \cdot (w^T x^{(\ell)} + b)]$ 的最优解。那么，对于所有 $\kappa \neq 0$ ， κw^* 和 κb^* 也必定是最优解
- 此外，总存在一个特定的 $\tilde{\kappa}$ 使得

对于所有 $\ell = 1, 2, \dots, n$ ， $y^{(\ell)} \cdot (\tilde{\kappa} w^{*\top} x^{(\ell)} + \tilde{\kappa} b^*) \geq 1$

并且至少有一个等号成立



➤ 因此，可以通过求解无约束优化问题来找到 $\frac{1}{\|\mathbf{w}\|} \min_{\ell} [y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b)]$ 的最大值：

$$\max_{\mathbf{w}, b} \left[\frac{1}{\|\mathbf{w}\|} \min_{\ell} [y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b)] \right]$$

或者通过求解带约束的优化问题来找到：

$$\max_{\widetilde{\mathbf{w}}, \tilde{b}} \left[\frac{1}{\|\widetilde{\mathbf{w}}\|} \min_{\ell} [y^{(\ell)} \cdot (\widetilde{\mathbf{w}}^T \mathbf{x}^{(\ell)} + \tilde{b})] \right]$$

$$s. t.: \text{对于所有 } \ell = 1, 2, \dots, n, \quad y^{(\ell)} \cdot (\widetilde{\mathbf{w}}^T \mathbf{x}^{(\ell)} + \tilde{b}) \geq 1$$

并且至少有一个等号成立

- 最优解 \mathbf{w}^* 和 $\widetilde{\mathbf{w}}^*$ 可能不相同，但它们所得到的 $\frac{1}{\|\mathbf{w}^*\|} \min_{\ell} [y^{(\ell)} \cdot (\mathbf{w}^{*T} \mathbf{x}^{(\ell)} + b^*)]$ 和 $\frac{1}{\|\widetilde{\mathbf{w}}^*\|} \min_{\ell} [y^{(\ell)} \cdot (\widetilde{\mathbf{w}}^{*T} \mathbf{x}^{(\ell)} + \tilde{b}^*)]$ 的值必定相等

- 在第二个优化问题中，由于对于所有 $\ell = 1, 2, \dots, n$ ， $y^{(\ell)} \cdot (\tilde{\mathbf{w}}^T \mathbf{x}^{(\ell)} + \tilde{b}) \geq 1$ 并且至少有一个等号成立，我们可以很容易地得到：

$$\min_{\ell} [y^{(\ell)} \cdot (\tilde{\mathbf{w}}^T \mathbf{x}^{(\ell)} + \tilde{b})] = 1$$

因此，优化目标可以简化为 $\frac{1}{\|\tilde{\mathbf{w}}\|}$

- 最大化 $\frac{1}{\|\tilde{\mathbf{w}}\|}$ 可以等价于最小化 $\|\tilde{\mathbf{w}}\|^2$ 。因此，该问题可以等价地写为：

$$\min_{\tilde{\mathbf{w}}, \tilde{b}} \|\tilde{\mathbf{w}}\|^2$$

$$\text{s. t. } \text{对于所有 } \ell = 1, 2, \dots, n, \quad y^{(\ell)} \cdot (\tilde{\mathbf{w}}^T \mathbf{x}^{(\ell)} + \tilde{b}) \geq 1$$

至少有一个等号成立

- 当最小化 $\|\tilde{\mathbf{w}}\|^2$ 时，“**至少有一个等号成立**”的约束条件将自动满足（为什么？？）。因此，可以在不影响结果的情况下将其移除

- 因此，最大间隔超平面可以通过求解下面的优化问题来找到：

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$s.t.: y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b) \geq 1, \text{ for } \ell = 1, 2, \dots, N$$

- 这是一个二次规划问题。它的最优解可以高效地通过**数值方法**找到

- 这是一个二次规划问题。它的最优解可以高效地通过**数值方法**找到
二次规划问题：

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{2} x^T Q x + c^T x \text{ subject to } Ax \leq b, \quad Ex = d, \quad l \leq x \leq u$$

$$\frac{1}{2} \|w\|^2$$

x 是决策变量的向量

Q 是一个 $n \times n$ 的对称正定矩阵，定义了二次项

c 是线性项的系数向量

A 和 b 定义了不等式约束

E 和 d 定义了等式约束

l 和 u 定义了变量的下界和上界

- 这是一个二次规划问题。它的最优解可以高效地通过**数值方法**找到

数值方法：牛顿法，

梯度下降法（神经网络学习），

共轭梯度下降法，

.....

- 当得到最优的 w^* 和 b^* 时，一个未见过的数据 x 可以被分类为：

$$\hat{y}(x) = \text{sign}(\mathbf{w}^{*T} \mathbf{x} + b^*)$$

等价对偶形式

- 每个凸优化问题都对应一个等价的对偶形式

本节内容摘自**凸优化**这一学科

- 原始优化问题的拉格朗日函数

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{\ell=1}^N \mathbf{a}_\ell (y^{(\ell)} (\mathbf{w}^\top \mathbf{x}^{(\ell)} + b) - 1),$$

其中，拉格朗日乘子 \mathbf{a}_ℓ 必须满足 $\mathbf{a}_\ell \geq 0$

- 拉格朗日对偶函数

$$g(\mathbf{a}) = \min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b, \mathbf{a})$$

- 原始优化问题的对偶形式

$$\begin{aligned} & \max_{\mathbf{a}} g(\mathbf{a}) \\ & \text{s. t.: } \mathbf{a} \geq \mathbf{0} \end{aligned}$$

- 推导函数 $g(\mathbf{a})$ 的闭式表达式

将梯度 $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ 和 $\frac{\partial \mathcal{L}}{\partial b}$ 设置为 0，得到：

$$\mathbf{w} = \sum_{\ell=1}^N a_\ell y^{(\ell)} \mathbf{x}^{(\ell)} \quad \sum_{\ell=1}^N a_\ell y^{(\ell)} = 0$$

$$\mathcal{L} = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{\ell=1}^N a_\ell (y^{(\ell)} (\mathbf{w}^T \mathbf{x}^{(\ell)} + b) - 1)$$

将它们代入 $\mathcal{L}(\mathbf{w}, b, \mathbf{a})$ ，得到 $g(\mathbf{a}) = \min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b, \mathbf{a})$ 为：

$$g(\mathbf{a}) = \underbrace{\sum_{\ell=1}^N a_\ell - \frac{1}{2} \sum_{\ell=1}^N \sum_{j=1}^N a_\ell a_j y^{(\ell)} y^{(j)} \mathbf{x}^{(\ell)T} \mathbf{x}^{(j)}}_{= \mathbf{1}^T \mathbf{a} - \frac{1}{2} \mathbf{a}^T \mathbf{M} \mathbf{a}}$$

其中 $[\mathbf{M}]_{\ell j} \triangleq y^{(\ell)} y^{(j)} \mathbf{x}^{(\ell)T} \mathbf{x}^{(j)}$

- 因此，对偶优化问题变为：

$$\begin{aligned} & \max_{\mathbf{a}} g(\mathbf{a}) \\ \text{s. t.: } & \mathbf{a} \geq \mathbf{0} \text{ 及 } \sum_{\ell=1}^N a_\ell y^{(\ell)} = 0 \end{aligned}$$

其中 $g(\mathbf{a}) = \underbrace{\sum_{\ell=1}^N a_\ell - \frac{1}{2} \sum_{\ell=1}^N \sum_{j=1}^N a_\ell a_j y^{(\ell)} y^{(j)} \mathbf{x}^{(\ell)T} \mathbf{x}^{(j)}}_{= \mathbf{1}^T \mathbf{a} - \frac{1}{2} \mathbf{a}^T \mathbf{M} \mathbf{a}}$

它也是一个二次规划问题，可以通过**数值方法** 高效地求解

- 最优解 w^* 、 b^* 与最优解 a^* 之间的关系

➤ 给定最优解 a^* ，根据 $w = \sum_{\ell=1}^N a_\ell y^{(\ell)} x^{(\ell)}$ ，最优的 w^* 可以等价地表示为：

$$w^* = \sum_{\ell=1}^N a_\ell^* y^{(\ell)} x^{(\ell)}$$

➤ 由于对于所有位于间隔边界上的样本 $(x^{(\ell)}, y^{(\ell)})$ ，都有 $y^{(\ell)}(w^{*T} x^{(\ell)} + b) = 1$ ，我们可以推导出：

$$b^* = \frac{1}{N_S} \sum_{n \in S} \left(y^{(n)} - \sum_m a_m^* y^{(m)} x^{(n)T} x^{(m)} \right)$$

其中 S 表示位于间隔边界上的样本集合

- 最大间隔分类器

- 原问题形式

$$\hat{y}(\mathbf{x}) = \text{sign}(\mathbf{w}^{*T}\mathbf{x} + b^*)$$

- 对偶问题形式

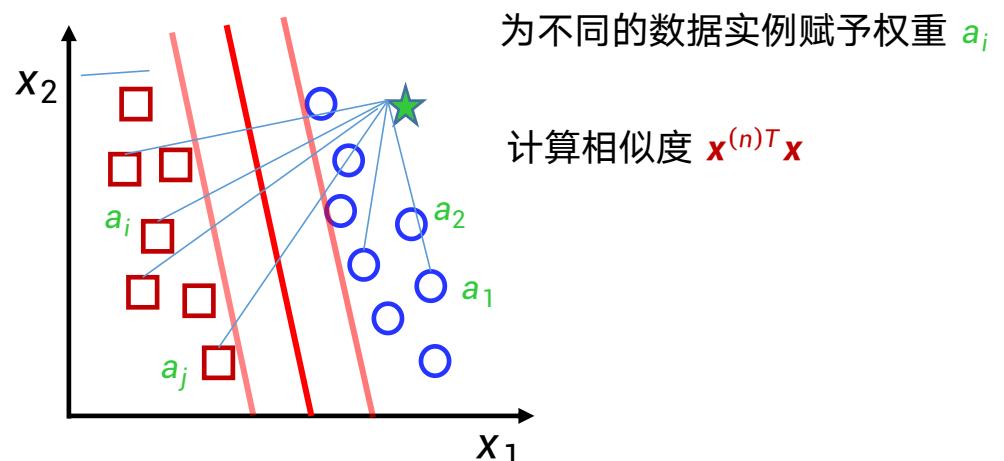
将 $\mathbf{w}^* = \sum_{n=1}^N a_n^* y^{(n)} \mathbf{x}^{(n)}$ 代入原问题形式，得到：

$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N a_n^* y^{(n)} \mathbf{x}^{(n)T} \mathbf{x} + b^* \right)$$

这两个分类器是等价的

$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N a_n^* \cdot (\mathbf{x}^{(n)T} \mathbf{x}) \cdot y^{(n)} + b^* \right)$$

- 如何理解对偶最大间隔分类器？
 - 对于一个测试样本 \mathbf{x} , 通过计算它与所有训练样本 $\mathbf{x}^{(n)}$ ($n = 1, \dots, N$) 的相似度 $\mathbf{x}^{(n)T} \mathbf{x}$
 - 将所有标签 $y^{(n)}$ 按样本相似度 $\mathbf{x}^{(n)T} \mathbf{x}$ 和乘子 a_n^* 进行加权求和



原问题与对偶问题比较

- 优化复杂度

原问题

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{s. t.: } y^{(\ell)} \cdot (\mathbf{w}^T \mathbf{x}^{(\ell)} + b) \geq 1,$$

for $\ell = 1, 2, \dots, N$

要优化的参数数量：特征维度

对偶问题

$$\max_{\mathbf{a}} g(\mathbf{a})$$

$$\text{s. t.: } \mathbf{a} \geq \mathbf{0}$$

$$\sum_{\ell=1}^N a_\ell y^{(\ell)} = 0$$

要优化的参数数量：训练样本数量

在高维特征情况下，求解对偶问题更高效

- 测试复杂度

原问题

$$\hat{y}(\mathbf{x}) = \text{sign}(\mathbf{w}^{*T} \mathbf{x} + b^*)$$

只需要一次内积运算

$$\mathbf{w}^{*T} \mathbf{x}$$

对偶问题

$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N a_n^* (\mathbf{x}^{(n)T} \mathbf{x}) y^{(n)} + b^* \right)$$

需要 N 次内积运算 $\mathbf{x}^{(n)T} \mathbf{x}$, 其中
 $n = 1, 2, \dots, N$

乍一看, 对偶分类器看起来比原分类器昂贵得多

- 幸运的是, 可以证明绝大多数 a_n^* 都为 0

拉格朗日乘子 a^* 的稀疏性

- 对于任何凸优化问题，最优解都满足 *KKT 条件*，对于我们这个问题，KKT 条件为：

KKT条件：

1. 原始可行性：约束被满足
2. 对偶可行性：拉格朗日乘子不小于零
3. 互补松弛性：拉格朗日乘子与约束的乘积为零
4. 梯度条件：最优点的梯度为零

$$a_n^* \geq 0$$

$$y^{(n)}(\mathbf{w}^{*T}\mathbf{x}^{(n)} + b^*) - 1 \geq 0$$

$$a_n^*[y^{(n)}(\mathbf{w}^{*T}\mathbf{x}^{(n)} + b^*) - 1] = 0$$

前两个条件来自原始的原问题和对偶问题

拉格朗日乘子 a^* 的稀疏性

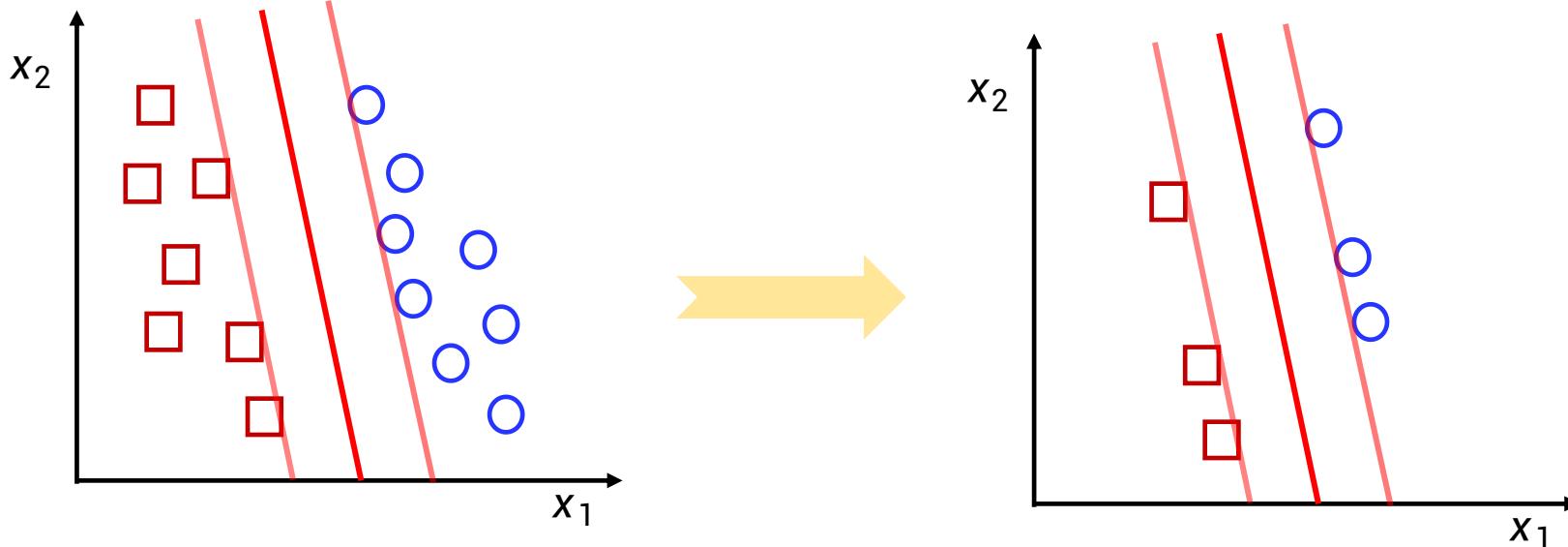
- 从最后一个条件可以看出，只有当 $y^{(n)}(\mathbf{w}^{*T}\mathbf{x}^{(n)} + b^*) = 1$ 时， $a_n^* \neq 0$
- 如果 $\mathbf{x}^{(n)}$ 满足 $y^{(n)}(\mathbf{w}^{*T}\mathbf{x}^{(n)} + b^*) = 1$ ，这意味着它位于间隔边界上

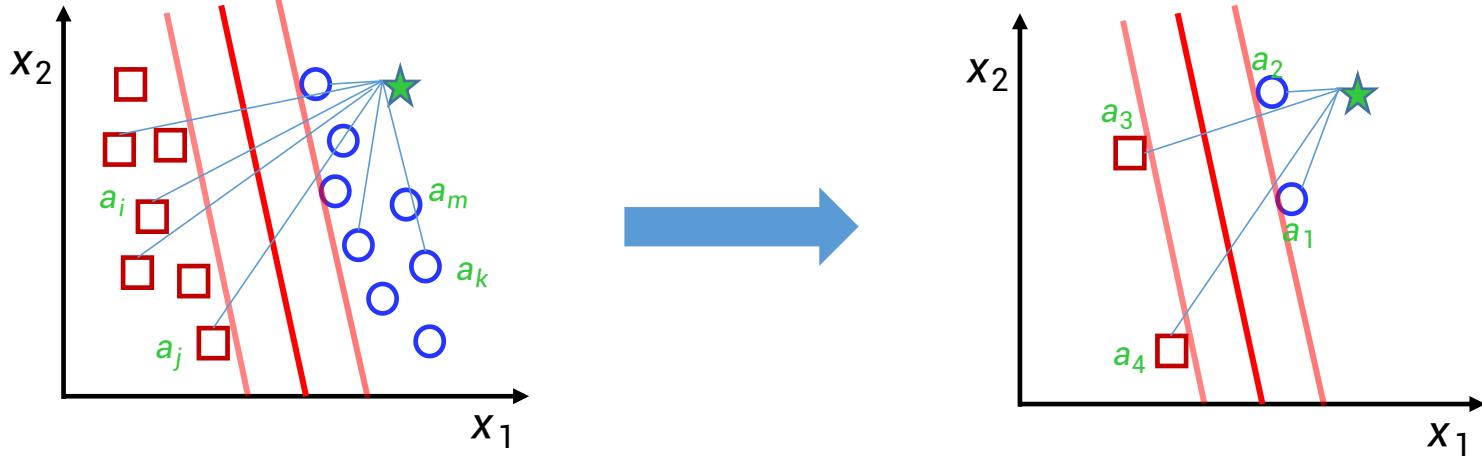
这类样本被称为支持向量

- 因此，当我们对一个未见过的样本 \mathbf{x} 进行分类时：

$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_n a_n^* y^{(n)} \mathbf{x}^{(n)T} \mathbf{x} + b^* \right),$$

我们只需要评估 \mathbf{x} 与支持向量（样本）之间的相似度 $\mathbf{x}^{(n)T} \mathbf{x}$





$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N \left(a_n^* (\mathbf{x}^{(n)T} \mathbf{x}) \right) \cdot y^{(n)} + b^* \right)$$

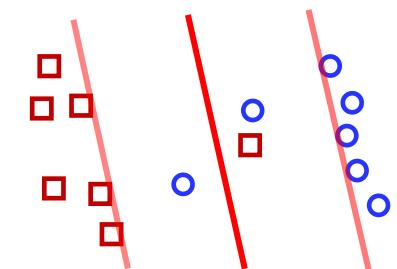
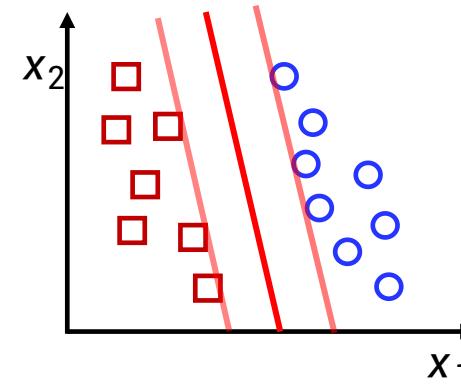
课程大纲

- 线性分类器的决策边界
- 线性最大间隔分类器
- 软线性最大间隔分类器
- 支持向量机
- 与逻辑回归的关系

非线性可分类别

- 之前最大间隔分类器中隐含的假设

训练样本是线性可分的！！！



非线性可分类别

- 在这种情况下，优化问题会发生什么？

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{s. t.: } y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1, \text{ for } n = 1, 2, \dots, N$$

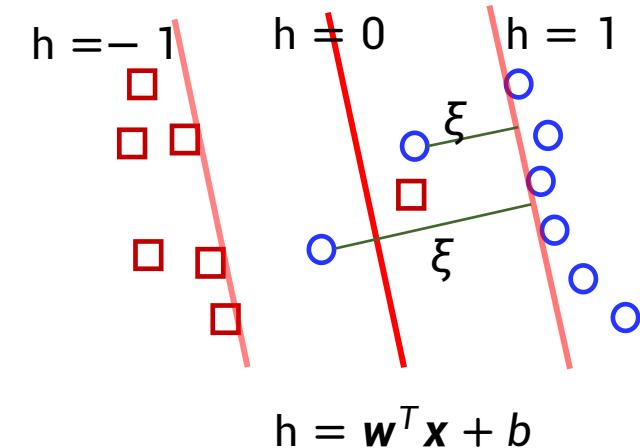
- 该优化问题没有可行解。也就是说，**不存在这样的超平面**

软最大间隔

- 为了解决这个问题，我们不再要求对于所有 $n = 1, \dots, N$ 都必须满足 $y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1$ ，而是只要求：

$$y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1 - \xi_n$$

其中 ξ_n 是一个松弛变量，并且 $\xi_n \geq 0$



- 目标不再仅仅是最小化 $\frac{1}{2} \|\mathbf{w}\|^2$ ，还需要最小化 ξ_n 的总和，从而得到以下目标函数：

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

其中 C 用于控制相对重要性

- 现在的优化问题变为：

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

$$s.t.: \quad y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1 - \xi_n,$$

$$\xi_n \geq 0, \quad \text{for } n = 1, 2, \dots, N$$

- 使用之前相同的方法，可以推导出对偶形式为：

$$\max_{\mathbf{a}} g(\mathbf{a})$$

$$\text{s. t.: } a_n \geq 0, \quad a_n \leq C$$

当 $a_n > C$ 时，可以证明 $g(\mathbf{a}) = -\infty$

$$\sum_{n=1}^N a_n y^{(n)} = 0$$

$$\text{其中 } g(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m y^{(n)} y^{(m)} \mathbf{x}^{(n)T} \mathbf{x}^{(m)}$$

- 当得到最优的 w^* 和 b^* 时，样本 x 被分类为：

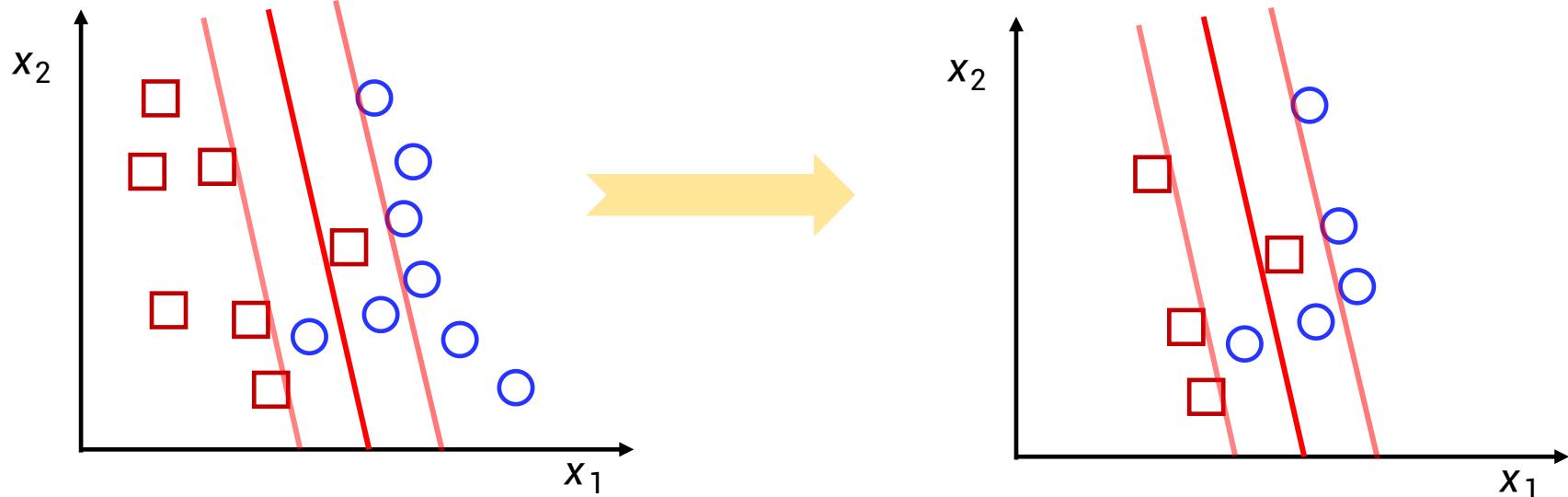
$$\hat{y}(x) = \text{sign}(w^{*T}x + b^*)$$

- 当得到最优的 a^* 时，样本 x 被分类为：

$$\hat{y}(x) = \text{sign} \left(\sum_{n=1}^N a_n^* y^{(n)} x^{(n)T} x + b^* \right)$$

此外，上面的两个分类器是等价的

- 最优解 a^* 是稀疏的，只有间隔边界内的元素是非零的



课程大纲

- 线性分类器的决策边界
- 线性最大间隔分类器
- 软线性最大间隔分类器
- 支持向量机
- 与逻辑回归的关系

非线性化

- 到目前为止，最大间隔分类器仍然是线性的
- 为了将模型非线性化，我们可以通过基函数将原始数据 x 变换到特征空间

$$\phi: x \rightarrow \phi(x)$$

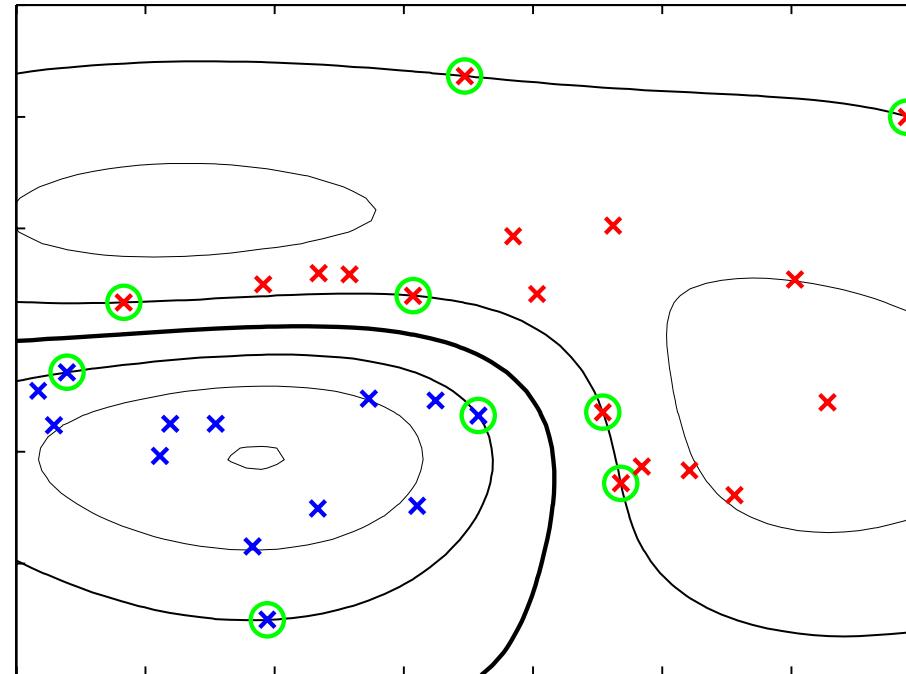
- 那么，原最大间隔优化问题变为：

$$\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n$$

$$s. t.: y^{(n)} \cdot (w^T \phi(x^{(n)}) + b) \geq 1 - \xi_n,$$

$$\xi_n \geq 0, \quad \text{for } n = 1, 2, \dots, N$$

$$\text{分类器: } \hat{y}(\mathbf{x}) = \text{sign}(\mathbf{w}^{*T} \boldsymbol{\phi}(\mathbf{x}^{(n)}) + b^*)$$



- 直观上，数据在高维空间中更容易被分离
- 为了获得更好的性能，我们应该将变换后特征空间 $\phi(x^{(n)})$ 的维度设置得尽可能高
- 然而，基函数 $\phi(x)$ 的维度不能设置得太高，因为原问题的计算成本会变得非常昂贵

$$\begin{aligned}
 & \min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n \\
 \text{s. t.: } & y^{(n)} \cdot (w^T \phi(x^{(n)}) + b) \geq 1 - \xi_n, \\
 & \xi_n \geq 0, \quad \text{for } n = 1, 2, \dots, N
 \end{aligned}$$

- 该问题可以通过其对偶形式求解

$$\begin{aligned} & \max_{\mathbf{a}} g(\mathbf{a}) \\ \text{s. t.: } & a_n \geq 0, \quad a_n \leq C \\ & \sum_{n=1}^N a_n y^{(n)} = 0 \end{aligned}$$

其中 $g(\mathbf{a}) = \underbrace{\sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m y^{(n)} y^{(m)} \boldsymbol{\phi}(\mathbf{x}^{(n)})^T \boldsymbol{\phi}(\mathbf{x}^{(m)})}_{= \mathbf{1}^T \mathbf{a} - \frac{1}{2} \mathbf{a}^T \mathbf{M} \mathbf{a}}$

分类器: $\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N a_n^* y^{(n)} \boldsymbol{\phi}(\mathbf{x}^{(n)})^T \boldsymbol{\phi}(\mathbf{x}) + b^* \right)$

- \mathbf{a} 的维度与 $\boldsymbol{\phi}(\cdot)$ 的维度无关, 因此对偶形式能够在非常大的特征空间 $\boldsymbol{\phi}(\cdot)$ 中工作

对偶形式需要计算内积

$$\phi(x^{(n)})^T \phi(x),$$

这在高维情况下计算成本很高

这个问题可以通过使用 **核技巧** 来解决

核函数

- 核函数是一个双变量函数 $k(\mathbf{x}, \mathbf{x}')$, 可以表示为某个函数 $\phi(\cdot)$ 的内积

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

显然, $\mathbf{x}^T \mathbf{x}'$ 和 $\phi(\mathbf{x})^T \phi(\mathbf{x}')$ 是核函数

- 默瑟定理: 如果一个函数 $k(\mathbf{x}, \mathbf{x}')$ 是对称正定的, 即

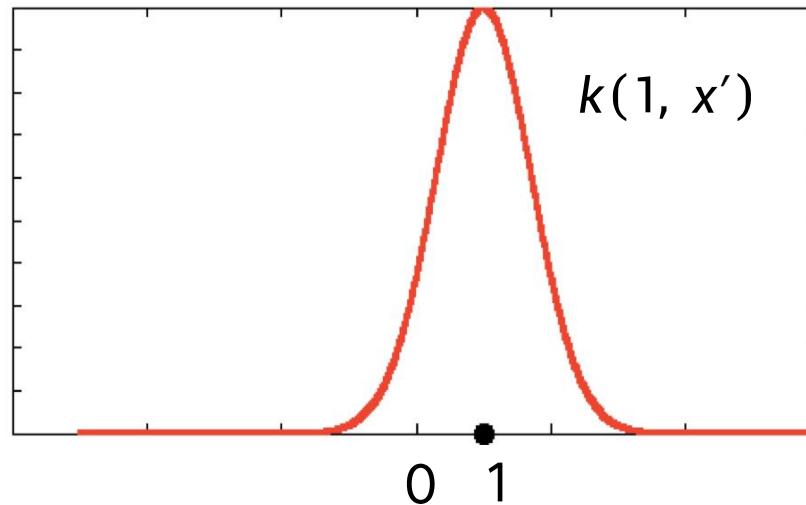
$$\int \int g(\mathbf{x}) k(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0, \quad \forall g(\cdot) \in L^2,$$

那么必然存在一个函数 $\phi(\cdot)$ 使得 $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$

如果一个函数 $k(\mathbf{x}, \mathbf{x}')$ 满足对称正定条件, 那么它必然是一个核函数

- 最广泛使用的核函数之一是高斯核，其形式为：

$$k(\mathbf{x}, \mathbf{x}') = \exp \left\{ -\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{x}'\|^2 \right\}$$



➤ 高斯核的函数 $\phi(\cdot)$ 具有无限维度

$$\phi(x) = e^{-x^2/2\sigma^2} \left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \dots \right]^T$$

核技巧

- 借助核函数，对偶最大间隔分类器可以等价地重写为：

$$\max_{\mathbf{a}} g(\mathbf{a})$$

$$\text{s. t.: } a_n \geq 0, \quad a_n \leq C$$

$$\sum_{n=1}^N a_n y^{(n)} = 0$$

$$\text{其中 } g(\mathbf{a}) = \underbrace{\sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m y^{(n)} y^{(m)} k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})}_{= \mathbf{1}^T \mathbf{a} - \frac{1}{2} \mathbf{a}^T \mathbf{M} \mathbf{a}}$$

- 导出的分类器

$$\hat{y}(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^N a_n^* y^{(n)} k(\mathbf{x}^{(n)}, \mathbf{x}) + b^* \right)$$

核技巧：用核函数 $k(\mathbf{x}, \mathbf{x}')$ 替代 $\Phi(\mathbf{x})^T \Phi(\mathbf{x}')$

- 结论可以总结如下：

- 如果 $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$, 这是一个线性最大间隔分类器
- 如果 $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$, 这是一个基于基函数的**有限维**非线性最大间隔分类器
- 如果 $k(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\right\}$, 这是一个**无限维**非线性最大间隔分类器

课程大纲

- 线性分类器的决策边界
- 线性最大间隔分类器
- 软线性最大间隔分类器
- 支持向量机
- 与逻辑回归的关系

- 在逻辑回归中，我们最小化损失：

$$L(\mathbf{w}, b) = - \sum_{n=1}^N \left[\tilde{y}^{(n)} \log \sigma(h^{(n)}) + (1 - \tilde{y}^{(n)}) \log (1 - \sigma(h^{(n)})) \right] + \lambda \|\mathbf{w}\|^2$$

$$= \sum_{n=1}^N \log (1 + \exp (-y^{(n)} h^{(n)})) + \lambda \|\mathbf{w}\|^2$$

注： $\tilde{y} \in \{0, 1\}$, $y \in \{-1, 1\}$

$$= \sum_{n=1}^N E_{LR}(y^{(n)} h^{(n)}) + \lambda \|\mathbf{w}\|^2$$

其中， $E_{LR}(z) = \log (1 + \exp (-z))$

- 在理想分类器中，我们最小化损失

$$L(\mathbf{w}, b) = \sum_{n=1}^N E_{Ideal}(y^{(n)} h^{(n)}) + \lambda \|\mathbf{w}\|^2$$

注: $y \in \{-1, 1\}$,
 $h \in (-\infty, \infty)$

其中, 如果 $z \geq 0$, $E_{Ideal}(z) = 0$; 否则为 1

公式定义: $E_{Ideal}(z) = 0 \text{ if } z \geq 0 \text{ else } 1$

阶跃函数, 不可导

- 回顾“线性最大间隔分类器”：

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{s. t.: } y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1, \text{ for } n = 1, 2, \dots, N$$

- 在线性最大间隔分类器中，我们等价地最小化损失：

$$L(\mathbf{w}, b) = \sum_{n=1}^N E_\infty(y^{(n)} h^{(n)} - 1) + \frac{1}{2} \|\mathbf{w}\|^2$$

其中，如果 $z \geq 0$ ，则 $E_\infty(z) = 0$ ；否则为 $+\infty$

硬约束：所有样本都满足约束，否则损失无穷大

公式定义： $E_\infty(z) = 0 \text{ if } z \geq 0 \text{ else } +\infty$

也为阶跃函数，可以发现仍然不可导

- 回顾“软线性最大间隔分类器”：

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

$$s.t.: y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1 - \xi_n,$$

$$\xi_n \geq 0, \quad \text{for } n = 1, 2, \dots, N$$

- 从约束推导损失函数：

$$\xi_n \geq 1 - y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b),$$

$$\xi_n \geq 0,$$

$$\xi_n = \max(0, 1 - y^{(n)} \cdot (\mathbf{w}^T \mathbf{x}^{(n)} + b))$$

注： $\max(0, 1 - z)$

- 在软线性最大间隔分类器中，我们等价地最小化损失：

$$L(\mathbf{w}, b) = C \sum_{n=1}^N E_{SV}(y^{(n)} \mathbf{h}^{(n)}) + \frac{1}{2} \|\mathbf{w}\|^2$$

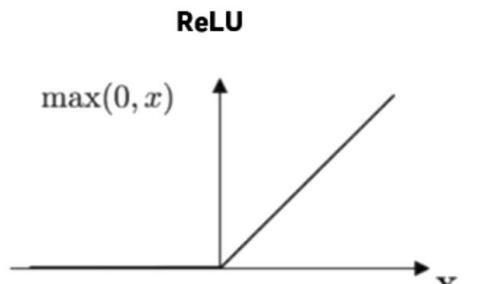
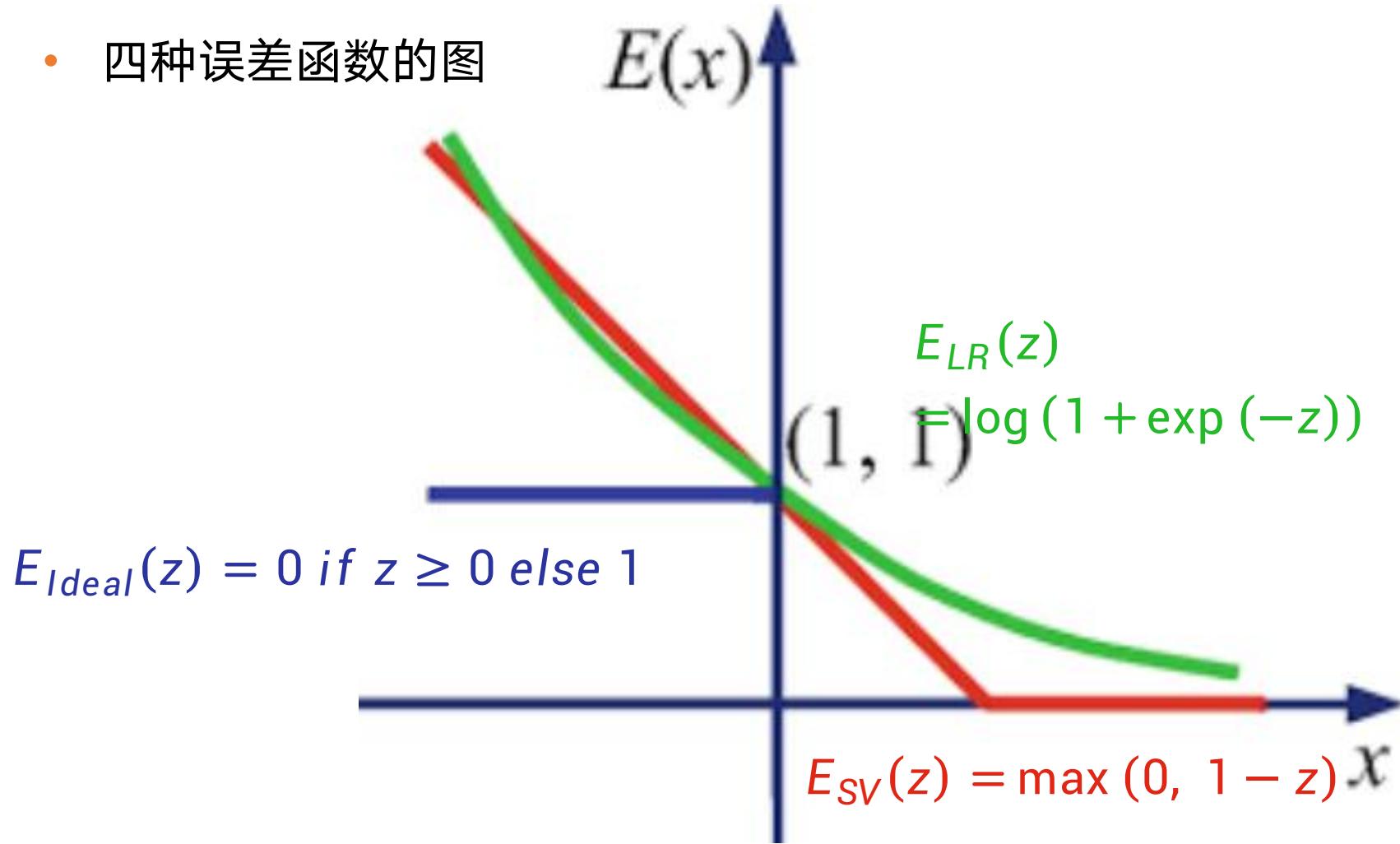
$$= \sum_{n=1}^N E_{SV}(y^{(n)} \mathbf{h}^{(n)}) + \lambda \|\mathbf{w}\|^2$$

其中， $E_{SV}(z) = \max(0, 1 - z)$ ，这被称为 *合页损失 (hinge loss)*

理想损失像“台阶”——对了 0，错了 1，不可导；
而在软线性最大间隔分类器里，我们用 hinge 函数把它磨平：
 $\max(0, 1 - z)$ —— 训练时可导，测试时近似理想

- 我们可以看到，这四种分类器可以在同一个框架下进行建模，唯一的区别在于所选择的误差函数

- 四种误差函数的图



注: $E_{SV}(z)$ 类似 ReLU
训练时可导:
可以使用次梯度求导

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$



机器学习与数据挖掘

神经网络



课程大纲

- 神经网络
- 反向传播
- 典型神经网络
 - 卷积神经网络 (CNN)
 - 循环神经网络 (RNN)
 - 基于 Transformer 的神经网络 (BERT, GPT)
 - 图神经网络

动机

- 许多应用需要富有表达能力的非线性模型

- 现有非线性化方法

- a) 使用基函数 $\phi(\cdot)$ 进行特征变换

$$\phi: x \rightarrow \phi(x)$$

- b) 核方法，可以理解为由无限维基函数 $\phi(\cdot)$ 表示的一种变换

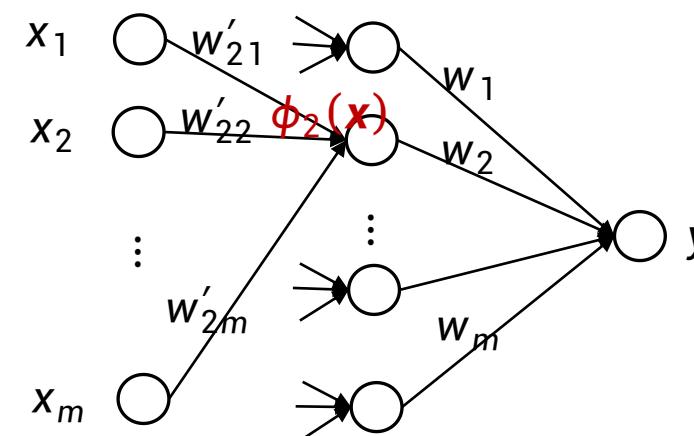
在这两种方法中，基函数都是固定的，并且不能根据数据的特性进行
自适应调整

神经网络

- 之前的非线性化过程可以如下图所示：



- 为了增加灵活性，我们将函数 $\phi_i(x)$ 变成可学习的



- 函数 $\phi_i(\mathbf{x})$ 可以设置为：

$$\phi_i(\mathbf{x}) = \mathbf{a}\left(\sum_{\ell=1}^m w'_{i\ell} x_\ell\right)$$

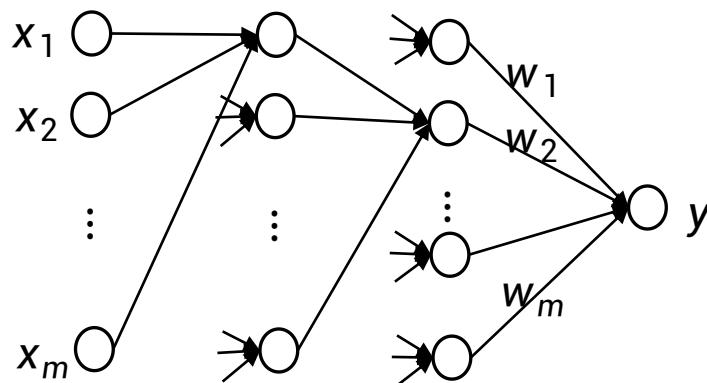
其中， $\mathbf{a}(\cdot)$ 是激活函数

$\mathbf{a}(\cdot)$ 不能被去掉，否则输出 y 将与输入 \mathbf{x} 呈线性关系

- 输出 y 可以简洁地写为：

$$\hat{y}(\mathbf{x}) = \mathbf{w}_2^T \underbrace{\mathbf{a}(\mathbf{W}_1 \mathbf{x})}_{\phi(\mathbf{x})}$$

- 进一步增加层数

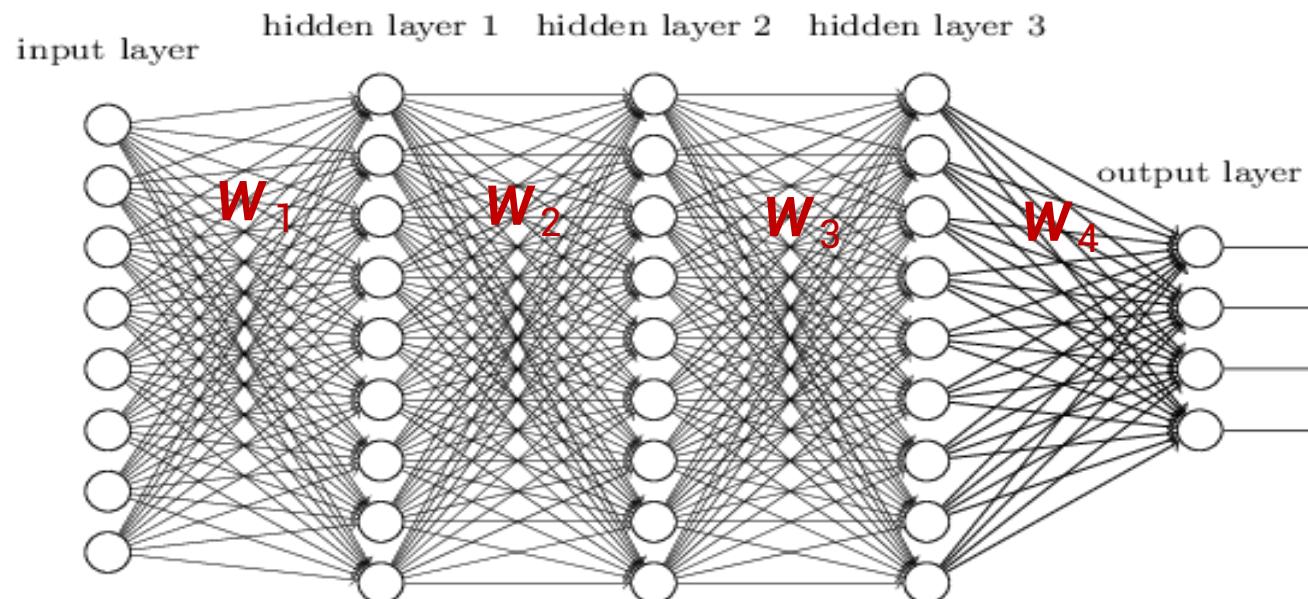


- 一般来说， L 层神经网络的输出可以表示为：

回归： $\hat{y}(\mathbf{x}) = \mathbf{W}_L a(\cdots a(\mathbf{W}_2 a(\mathbf{W}_1 \mathbf{x})))$

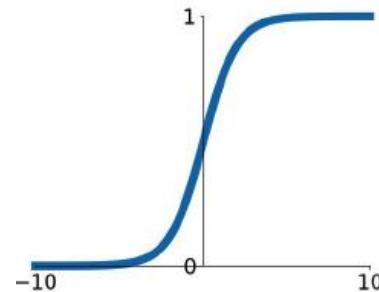
分类： $\hat{y}(\mathbf{x}) = \text{softmax}\left(\mathbf{W}_L a(\cdots a(\mathbf{W}_2 a(\mathbf{W}_1 \mathbf{x})))\right)$

其中 \mathbf{W}_ℓ 是第 ℓ 层的参数



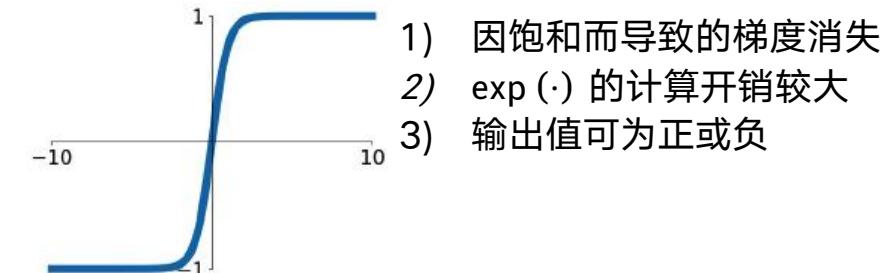
• 激活函数

Sigmoid: $a(x) = \frac{1}{1+e^{-x}}$



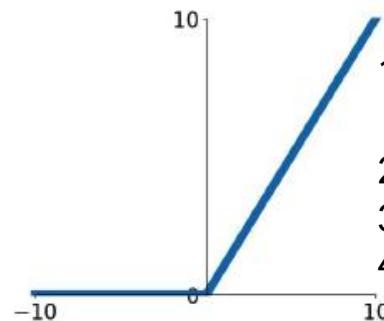
- 1) 因饱和而导致的梯度消失
- 2) 输出值只为正
- 3) $\exp(\cdot)$ 的计算开销较大

Tanh: $a(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$



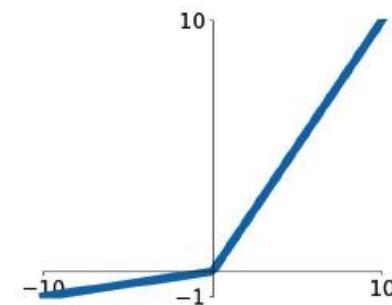
- 1) 因饱和而导致的梯度消失
- 2) $\exp(\cdot)$ 的计算开销较大
- 3) 输出值可为正或负

ReLU: $a(x) = \max(0, x)$



- 1) 在 $x > 0$ 时没有梯度消失问题
- 2) 在 $x < 0$ 时梯度为 0
- 3) 计算高效
- 4) 输出值只为正

Leaky ReLU: $a(x) = \max(0.1x, x)$



- 1) 对于所有 x 都没有梯度消失问题
- 2) 计算高效

损失函数

- 给定一个训练数据集 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, 回归和分类的训练目标分别是:

1) 回归损失

$$l_r(\boldsymbol{\theta}) = \frac{1}{N} \sum_{(\mathbf{x}, y) \in \mathcal{D}} |y - \hat{y}(\mathbf{x})|^2$$

2) 多类别分类损失

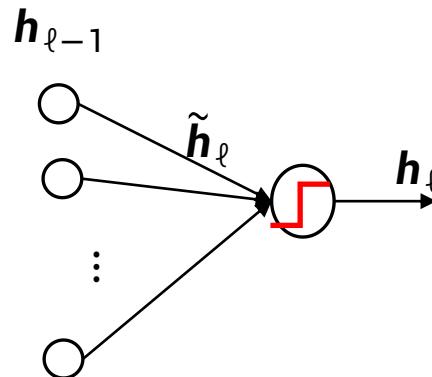
$$l_c(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \sum_{k=1}^K y_k \log \hat{y}_k(\mathbf{x})$$

课程大纲

- 神经网络
- 反向传播
- 典型神经网络

梯度

- 为了训练神经网络，我们需要损失 \mathcal{L} 关于 \mathbf{W}_ℓ 的梯度，即 $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\ell}$
- 为此，我们将神经网络的输出用递归方式表示

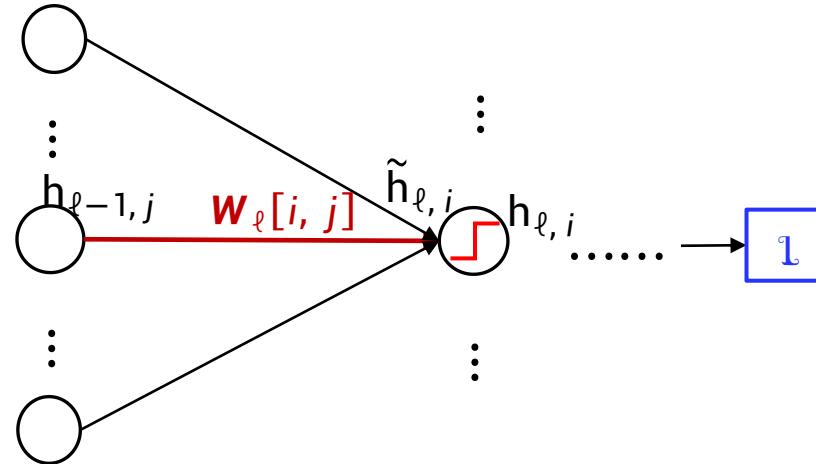


$$\tilde{\mathbf{h}}_l = \mathbf{W}_l \mathbf{h}_{l-1}, \quad \mathbf{h}_l = a(\tilde{\mathbf{h}}_l), \quad \mathbf{h}_l = a(\mathbf{W}_l \mathbf{h}_{l-1})$$

➤ 输出层

$$\hat{y} = \tilde{\mathbf{h}}_L \text{ 或 } softmax(\tilde{\mathbf{h}}_L)$$

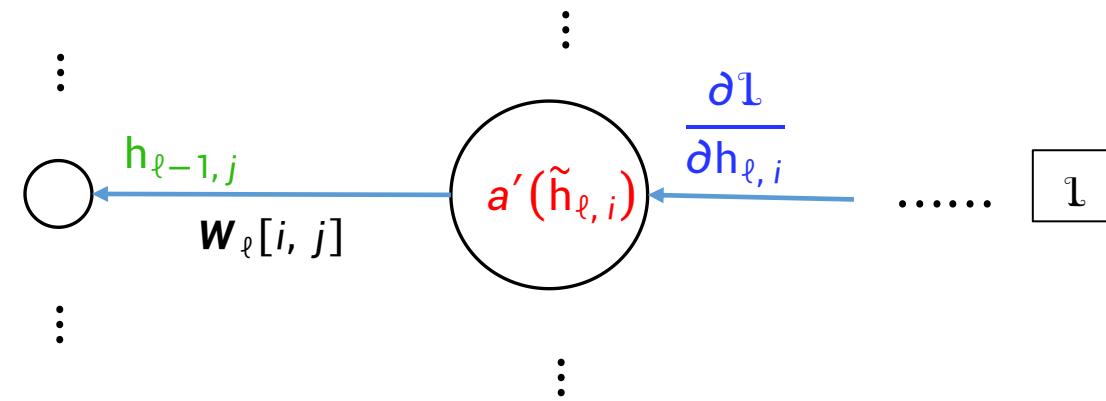
- 损失 $\mathbb{1}$ 关于 $W_\ell[i, j]$ 的导数，即 $\frac{\partial \mathbb{1}}{\partial W_\ell[i, j]}$



$$\begin{aligned}
 \frac{\partial \mathbb{1}}{\partial W_\ell[i, j]} &= \frac{\partial \mathbb{1}}{\partial h_{\ell, i}} \cdot \boxed{\frac{\partial h_{\ell, i}}{\partial W_\ell[i, j]}} \\
 &= \frac{\partial \mathbb{1}}{\partial h_{\ell, i}} \cdot \boxed{\frac{\partial h_{\ell, i}}{\partial \tilde{h}_{\ell, i}}} \cdot \boxed{\frac{\partial \tilde{h}_{\ell, i}}{\partial W_\ell[i, j]}} \\
 &= \boxed{\frac{\partial \mathbb{1}}{\partial h_{\ell, i}}} \cdot a'(\tilde{h}_{\ell, i}) \cdot h_{\ell-1, j}
 \end{aligned}$$

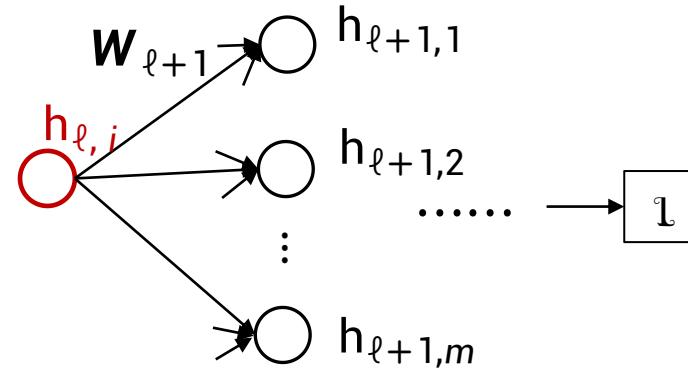
➤ 导数结构示意图

$$\frac{\partial \mathcal{L}}{\partial w_{\ell}[i,j]} = \frac{\partial \mathcal{L}}{\partial h_{\ell,i}} \cdot a'(\tilde{h}_{\ell,i}) \cdot h_{\ell-1,j}$$



为了计算 $\frac{\partial \mathcal{L}}{\partial w_{\ell}[i,j]}$, 关键是计算导数 $\frac{\partial \mathcal{L}}{\partial h_{\ell,i}}$

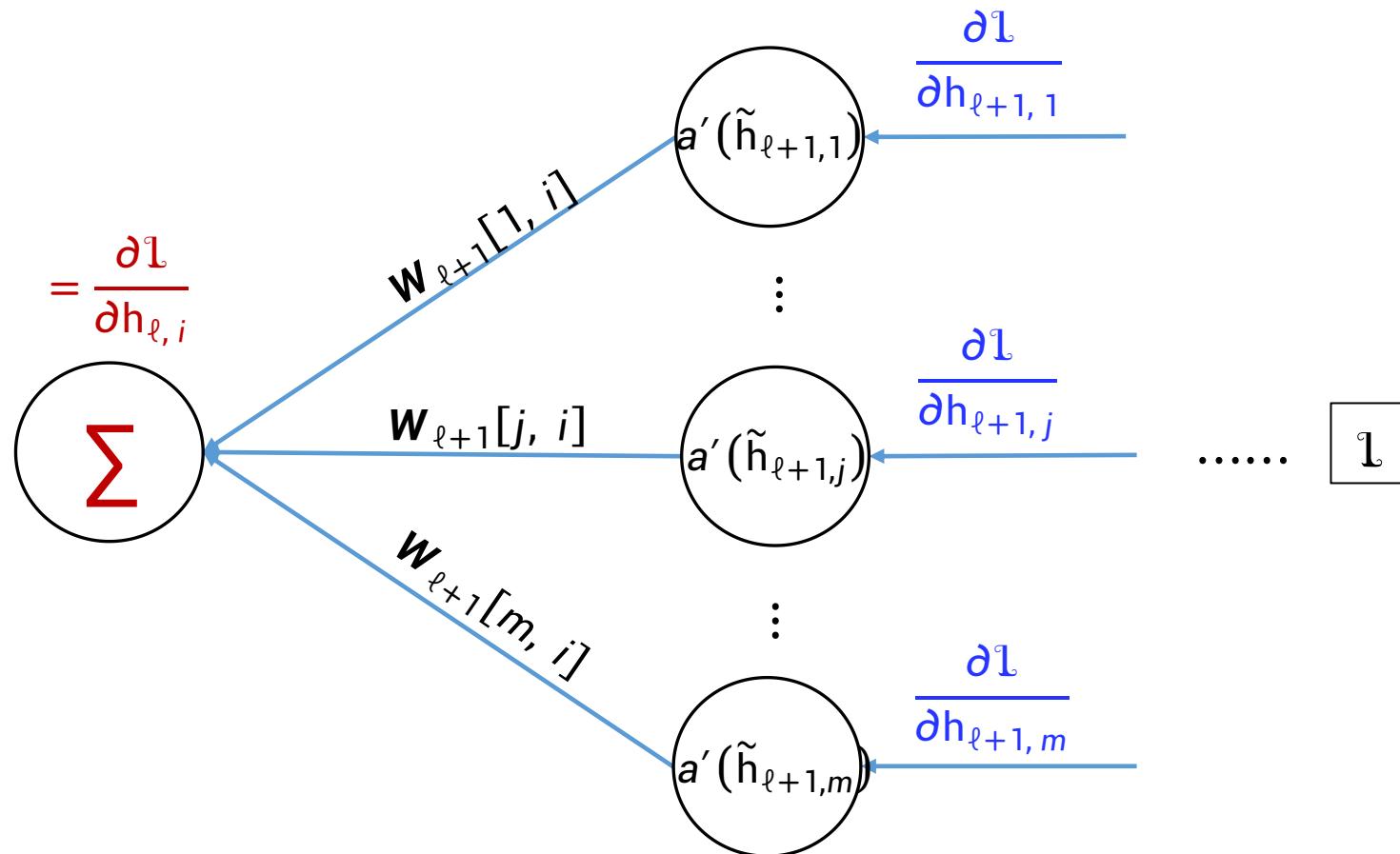
- 损失 l 关于 $h_{\ell, i}$ 的导数，即 $\frac{\partial l}{\partial h_{\ell, i}}$



$$\begin{aligned}
 \frac{\partial l}{\partial h_{\ell, i}} &= \sum_{j=1}^m \left[\frac{\partial l}{\partial h_{\ell+1, j}} \cdot \frac{\partial h_{\ell+1, j}}{\partial h_{\ell, i}} \right] \\
 &= \sum_{j=1}^m \left[\frac{\partial l}{\partial h_{\ell+1, j}} \cdot \frac{\partial h_{\ell+1, j}}{\partial \tilde{h}_{\ell+1, j}} \cdot \frac{\partial \tilde{h}_{\ell+1, j}}{\partial h_{\ell, i}} \right] \\
 &= \sum_{j=1}^m \left[\frac{\partial l}{\partial h_{\ell+1, j}} \cdot a'(\tilde{h}_{\ell+1, j}) \cdot w_{\ell+1}[j, i] \right]
 \end{aligned}$$

➤ 表达式中的递归结构

$$\frac{\partial L}{\partial h_{\ell, i}} = \sum_{j=1}^m \left[\frac{\partial L}{\partial h_{\ell+1, j}} \cdot a'(\tilde{h}_{\ell+1, j}) \cdot W_{\ell+1}[j, i] \right]$$



➤ 损失关于最后一层输出 h_L 的初始导数

1) 回归损失: $\mathcal{L} = \frac{1}{2} |y - h_L|^2$

$$\frac{\partial \mathcal{L}}{\partial h_L} = (h_L - y)$$

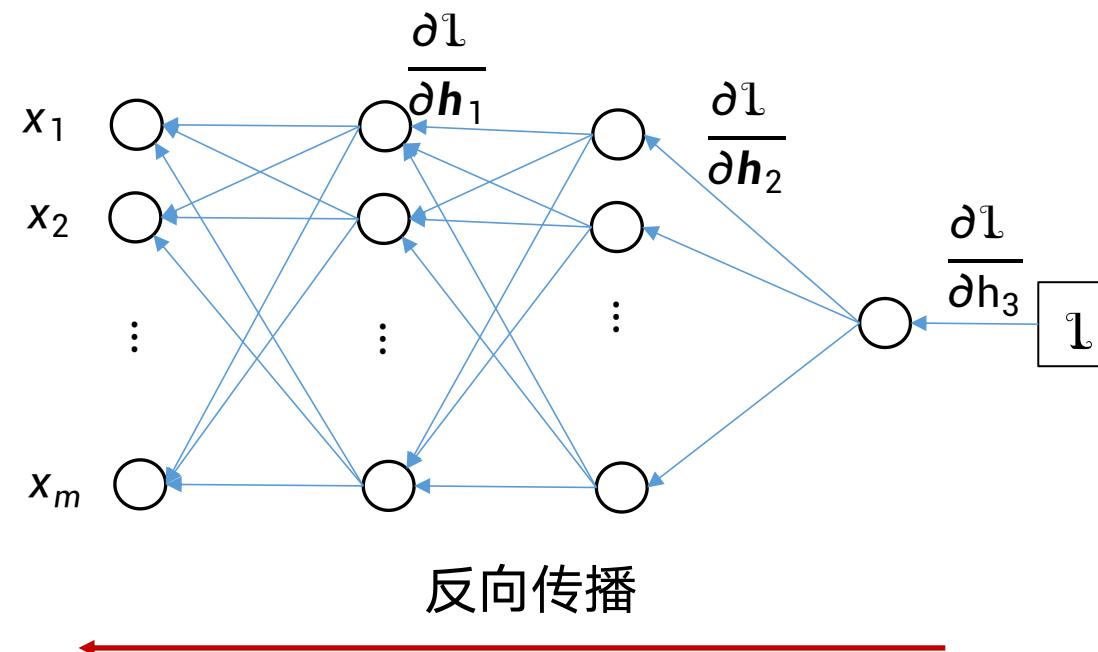
2) 交叉熵损失: $\mathcal{L} = -y \log h_L - (1 - y) \log (1 - h_L)$

$$\frac{\partial \mathcal{L}}{\partial h_L} = \frac{h_L - y}{h_L}$$

梯度完整计算图

- 两个步骤

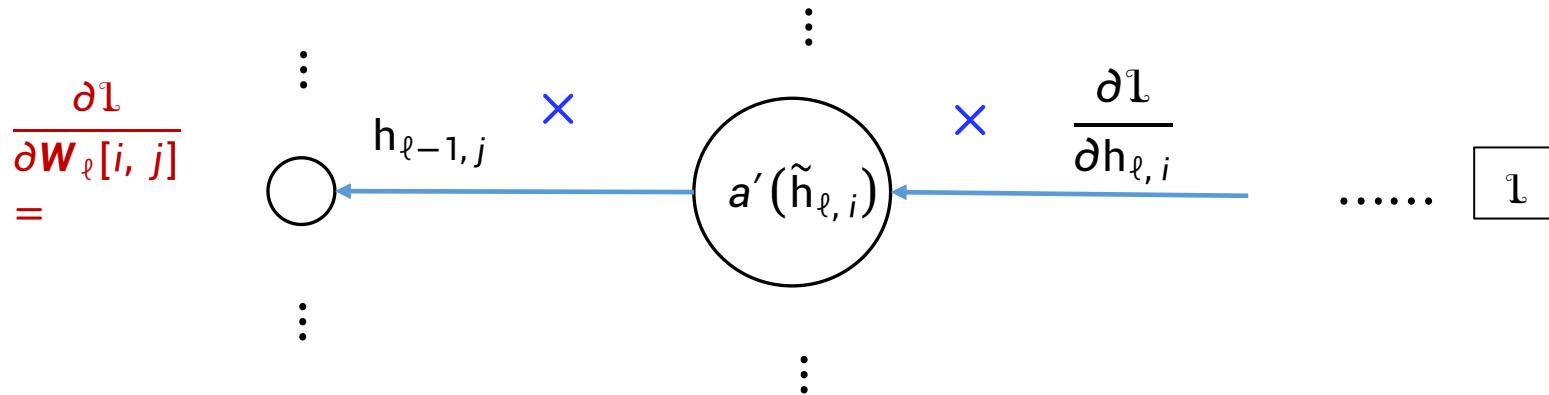
➤ 步骤 1：计算 $\frac{\partial l}{\partial h_\ell}$



$$\frac{\partial l}{\partial h_\ell} = \mathbf{W}_{\ell+1}^T \left(\tilde{\mathbf{g}}_{\ell+1} \odot \frac{\partial l}{\partial h_{\ell+1}} \right)$$

其中 $\tilde{\mathbf{g}}_{\ell+1} = \left[\frac{\partial h_{\ell+1,1}}{\partial \tilde{h}_{\ell+1,1}}, \dots, \frac{\partial h_{\ell+1,m}}{\partial \tilde{h}_{\ell+1,m}} \right]^T$

➤ 步骤 2：计算 $\frac{\partial \mathbf{l}}{\partial \mathbf{W}_\ell}$



$$\begin{aligned}\frac{\partial \mathbf{l}}{\partial \mathbf{W}_\ell[i, j]} \\ = \frac{\partial \mathbf{l}}{\partial h_{\ell, i}} \times a'(\tilde{h}_{\ell, i}) \times h_{\ell-1, j}\end{aligned}$$

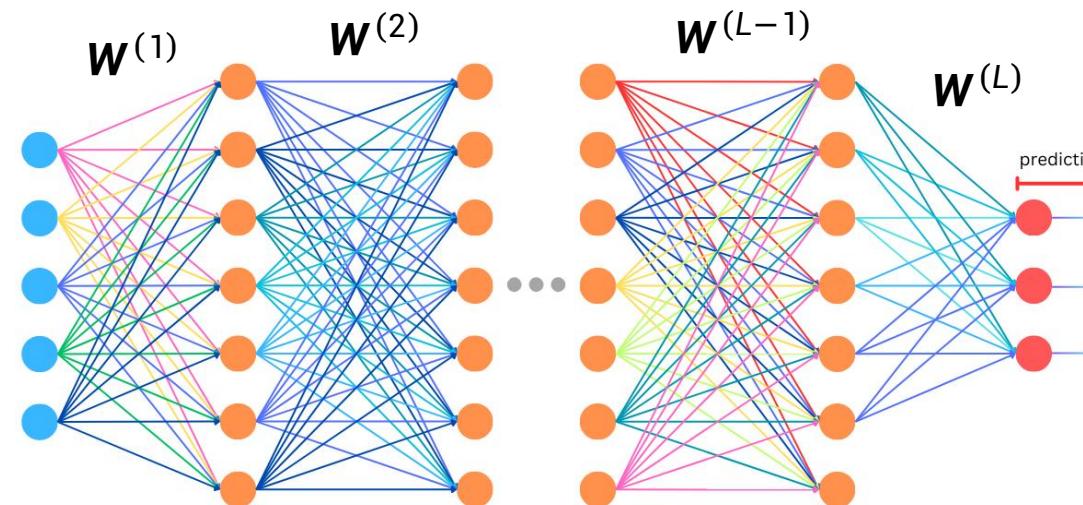
用矩阵形式写为：

$$\frac{\partial \mathbf{l}}{\partial \mathbf{W}_\ell} = \left(\tilde{\mathbf{g}}_\ell \odot \frac{\partial \mathbf{l}}{\partial \mathbf{h}_\ell} \right) \mathbf{h}_{\ell-1}^T$$

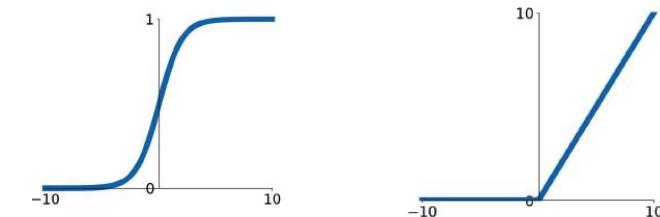
课程大纲

- 神经网络
- 反向传播
- 典型神经网络
 - 卷积神经网络 (CNN)
 - 循环神经网络 (RNN)
 - 基于 Transformer 的神经网络 (BERT, GPT)
 - 图神经网络

多层感知机MLP的局限



$$h^{(\ell+1)} = \sigma(W^{(\ell)} h^{(\ell)})$$



$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

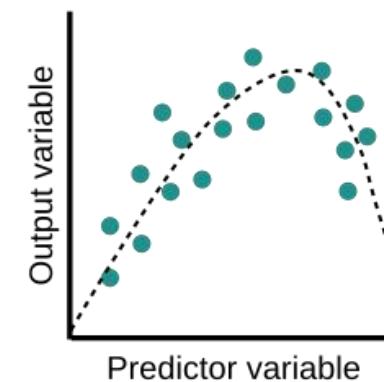
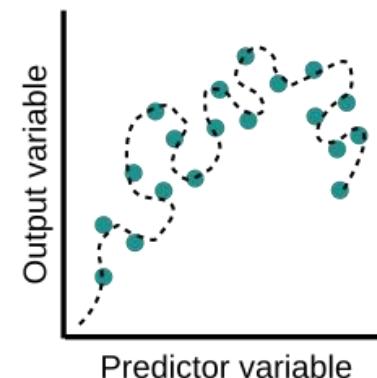
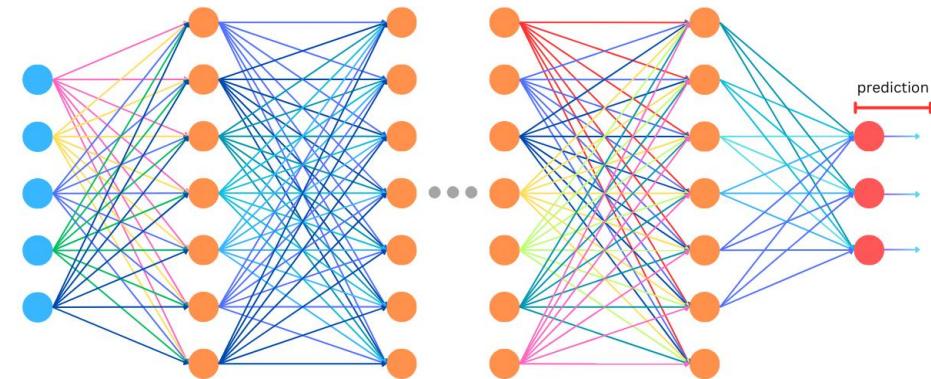
问题：当输入的图像像素是 200×200 时，网络的第一层包含多少参数？

$$(200 \times 200 \times 3)^2 = 14.4 \text{ Billion}$$

多层感知机MLP的局限

模型参数过多会导致什么问题？

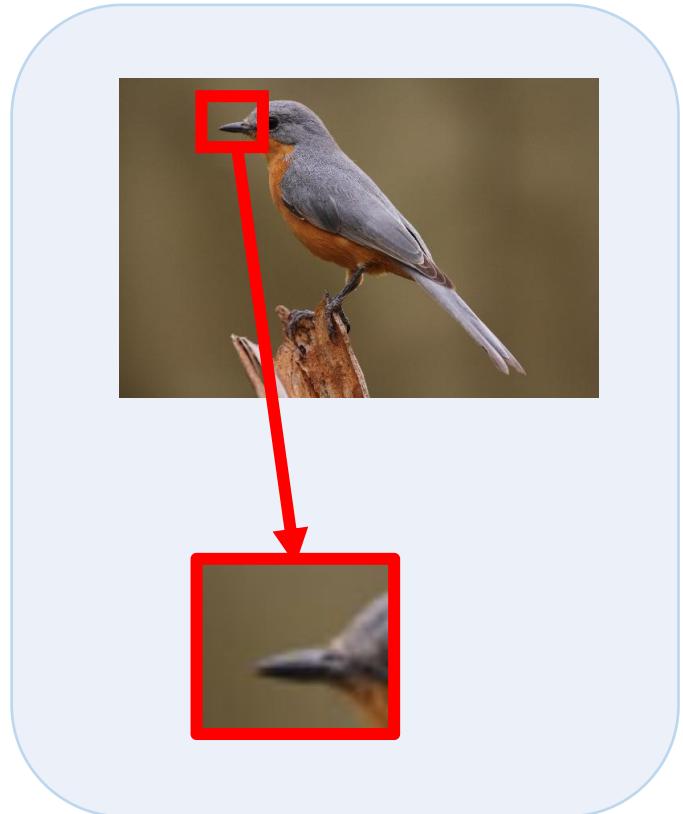
- 模型训练成本非常高
- 模型及其容易过拟合



对于图像数据，该如何设计神经网络？

图像数据的独特性

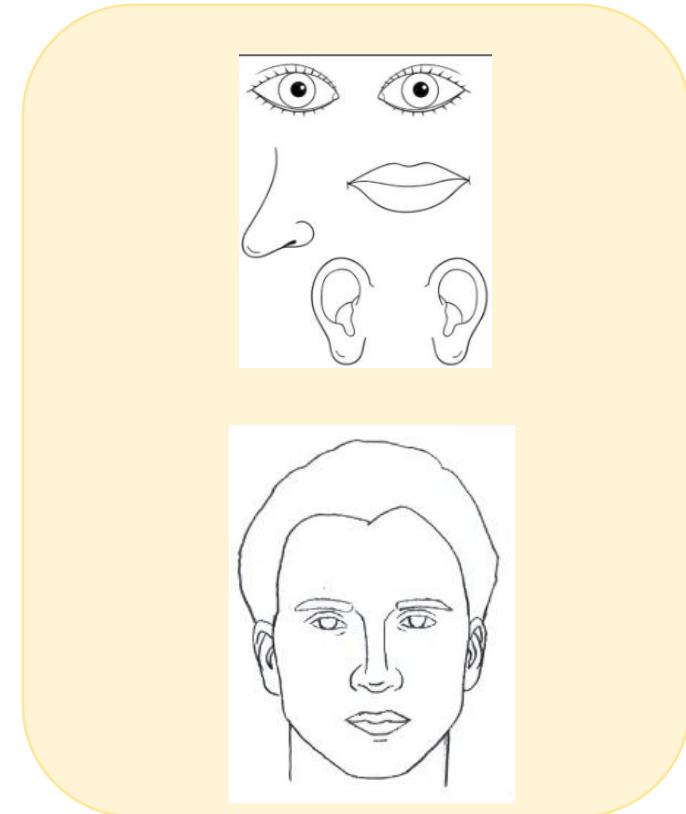
局部结构



平移不变性

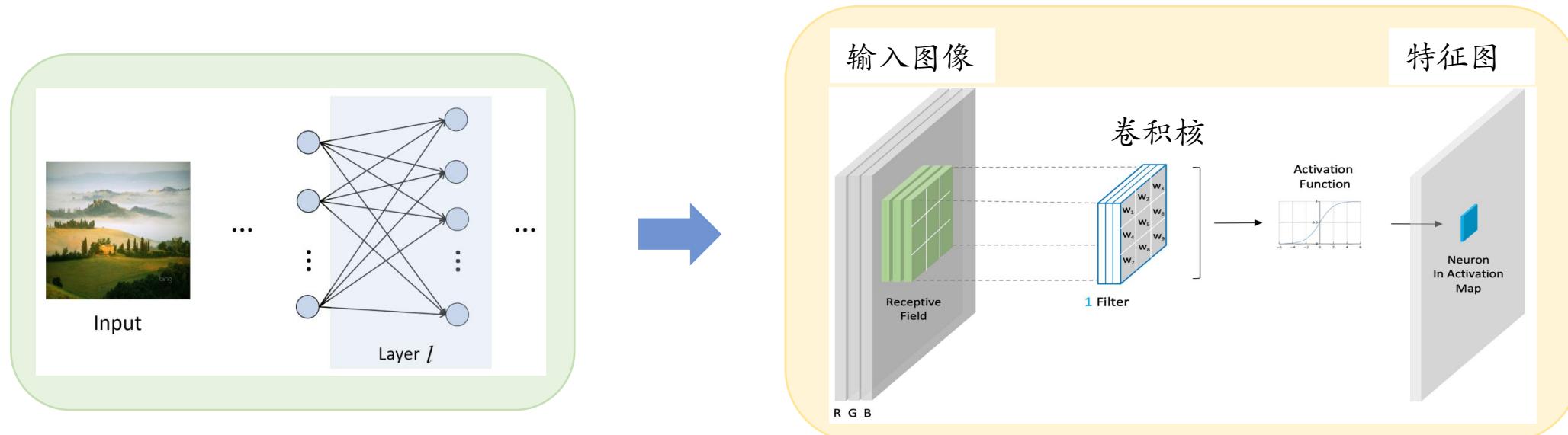


局部组合形成整体



如何设计网络结构，以便更好利用图像数据这些独特的特点？

从全链接到卷积链接



与全连接的神经网络相比，卷积链接具有如下特点

- ✿ 局部连接性：每个神经元只连接到前一层的一部分神经元
- ✿ 参数共享：不同神经元使用相同的一组参数

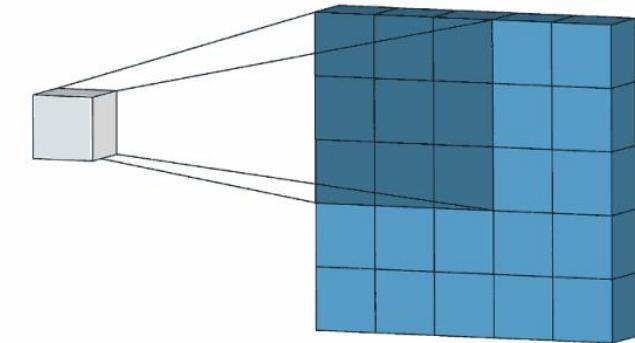
特征图计算过程

$$\begin{matrix} -1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & -1 & 1 \end{matrix} \otimes \begin{matrix} \text{卷积核} \\ -1 & 1 \\ 1 & -2 \end{matrix} \rightarrow \begin{matrix} 7 \\ \\ \end{matrix}$$

$$\begin{matrix} -1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & -1 & 1 \end{matrix} \otimes \begin{matrix} \text{卷积核} \\ -1 & 1 \\ 1 & -2 \end{matrix} \rightarrow \begin{matrix} 7 & -4 \\ \\ \end{matrix}$$

$$\begin{matrix} 1 & -1 & 1 \\ 1 & -2 & 1 \\ 1 & -1 & 1 \end{matrix} \otimes \begin{matrix} \text{卷积核} \\ -1 & 1 \\ 1 & -2 \end{matrix} \rightarrow \begin{matrix} 7 & -4 \\ 0 \\ \end{matrix}$$

$$\begin{matrix} 1 & -1 & 1 \\ 1 & -2 & 1 \\ 1 & -1 & 1 \end{matrix} \otimes \begin{matrix} \text{卷积核} \\ -1 & 1 \\ 1 & -2 \end{matrix} \rightarrow \begin{matrix} 7 & 4 \\ 0 & 0 \end{matrix}$$



在图像什么区域容易产生较大的输出值？



图像区域值与卷积核相匹配的时候

卷积核扫描图像的过程可以怎么理解？



搜寻图像中某一特定的特征

卷积核功能展示



原始图像



锐化(Sharpen)



边缘检测
(Edge Detect)

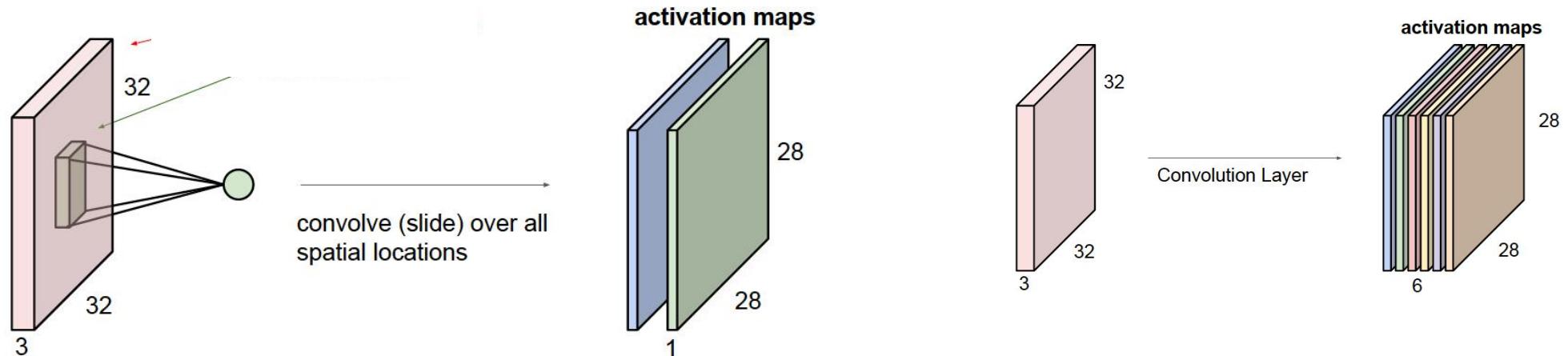


强边缘检测
(Strong Edge Detect)

图像中的不同特征或模式可用不同卷积核去检测！

多卷积核结构

可使用多个卷积核，每个负责识别不同模式



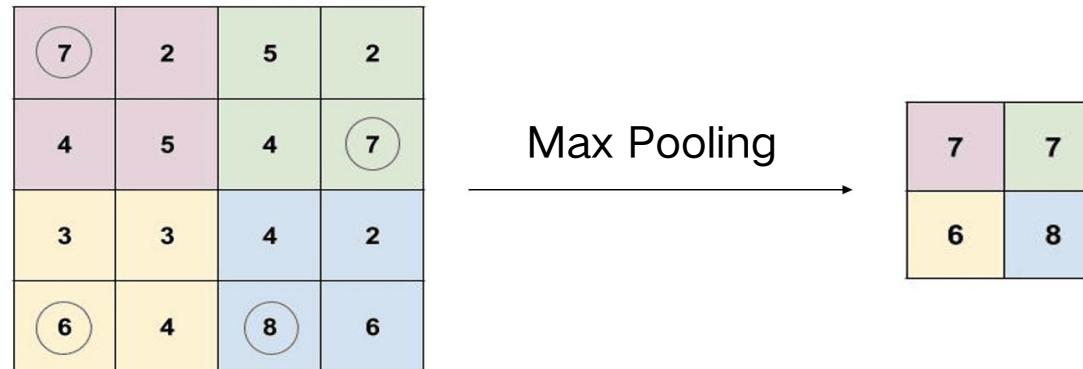
每个卷积核具体功能通过模型训练获得

问题：假设卷积核大小为 5×5 ，图像有3个通道，输出200个特征图，模型包含多少个参数？

$$5 \times 5 \times 3 \times 200 = 15000$$

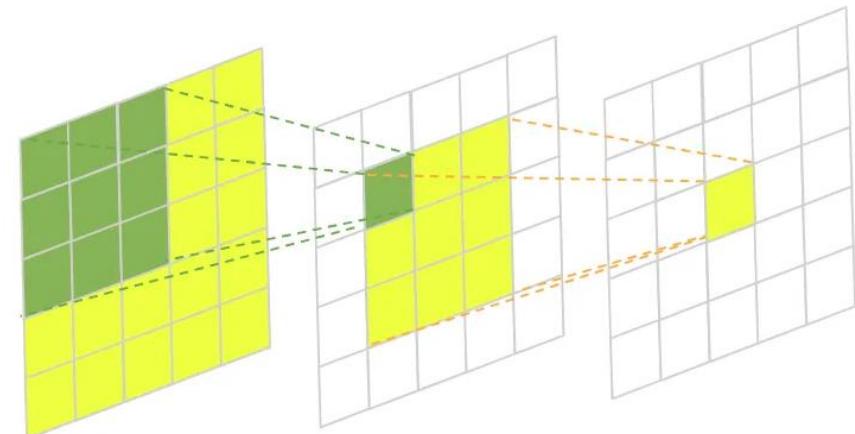
参数量远远小于MLP网络的参数。计算量呢？

池化操作 (Pooling)



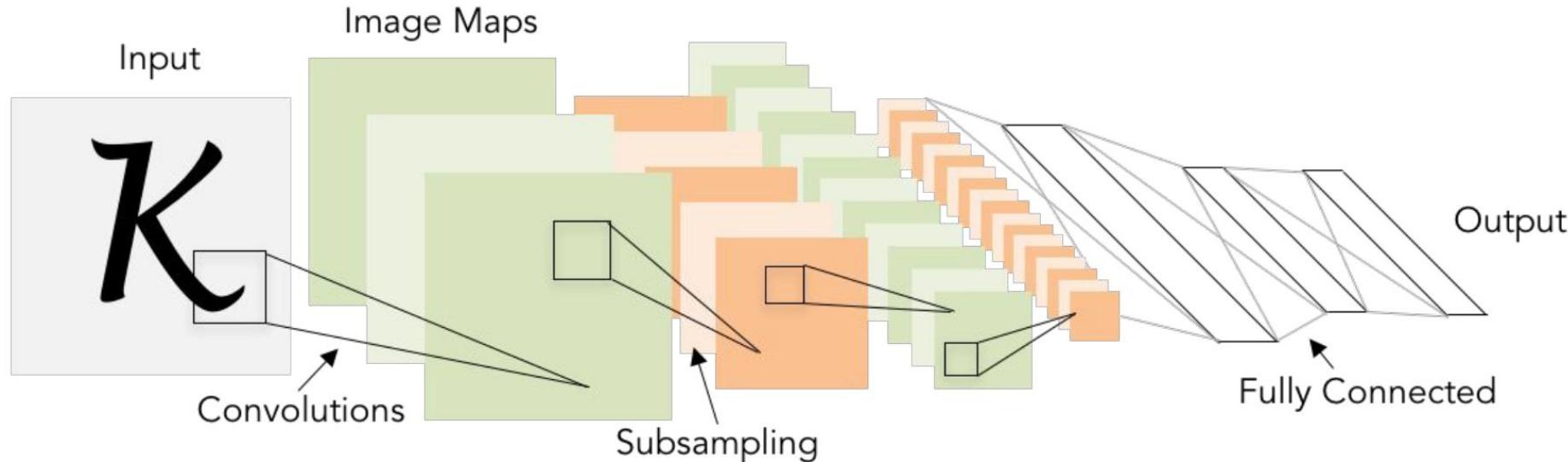
池化操作的作用

- 提升平移、旋转不变性
- 降低特征图的维度，减少计算量
- 增大每个特征图元素的感受野，提升输出特征的全局性



经典卷积神经网络

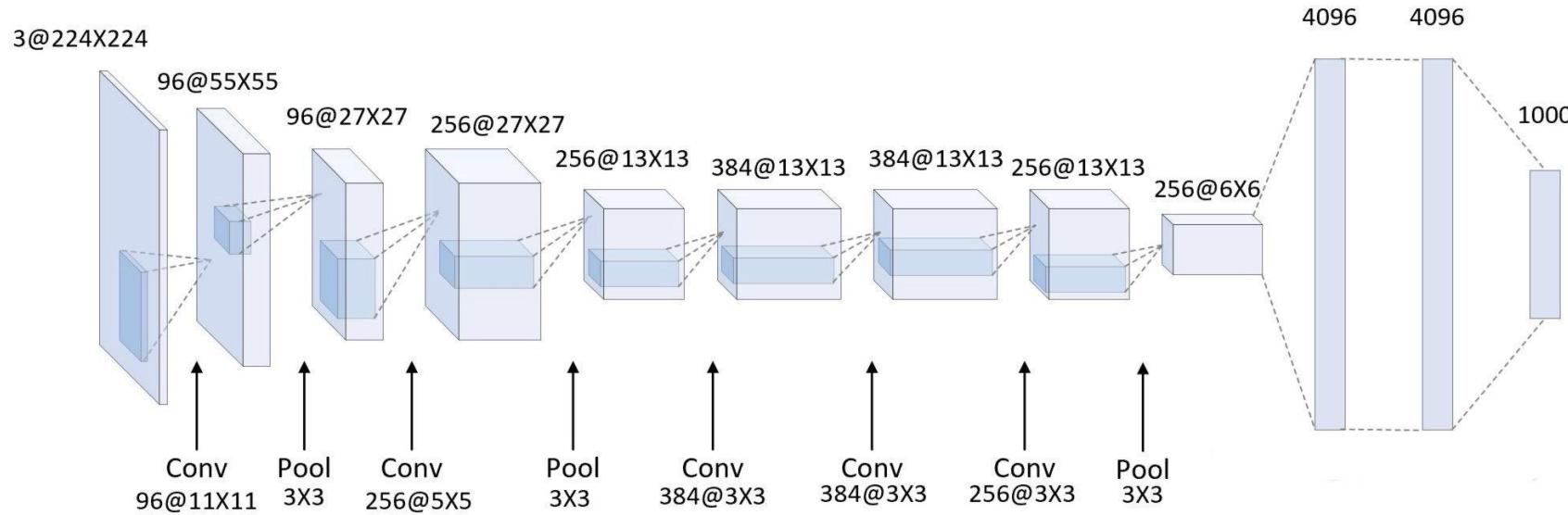
□ LeNet



- LeNet是第一个被广泛应用于数字图像识别的神经网络之一，其由Yann LeCun 在1989年提出，旨在解决手写数字识别问题
- 具体网络结构：卷积层-池化层-卷积层-池化层-全连接层-全连接层

经典卷积神经网络

□ AlexNet

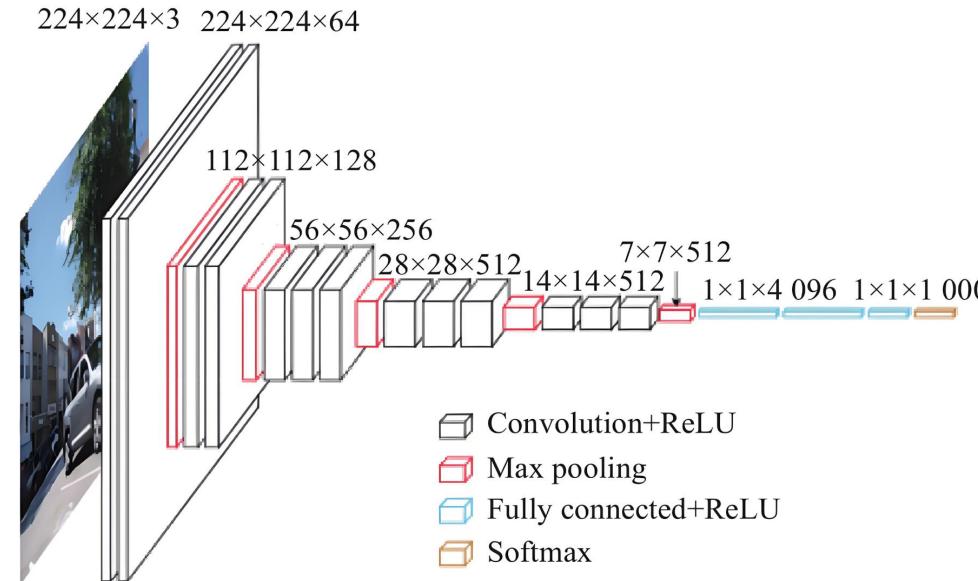


- AlexNet是由Alex、Hinton等人于2012年ImageNet图像分类竞赛中提出的一种经典网络
- AlexNet包含5个卷积层、3个池化层、3个全连接层



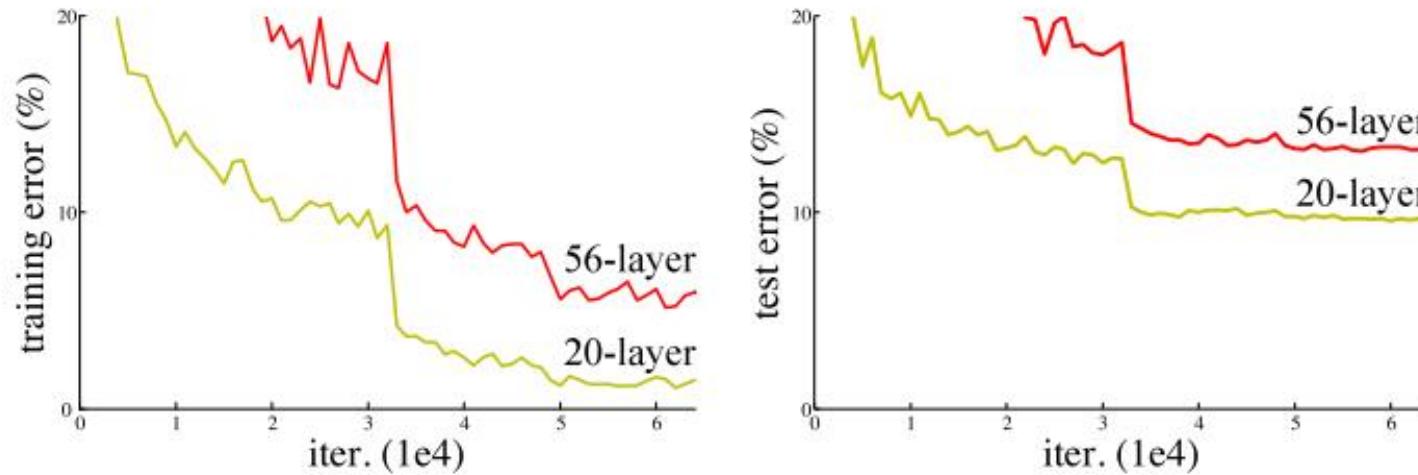
经典卷积神经网络

□ VGG Net

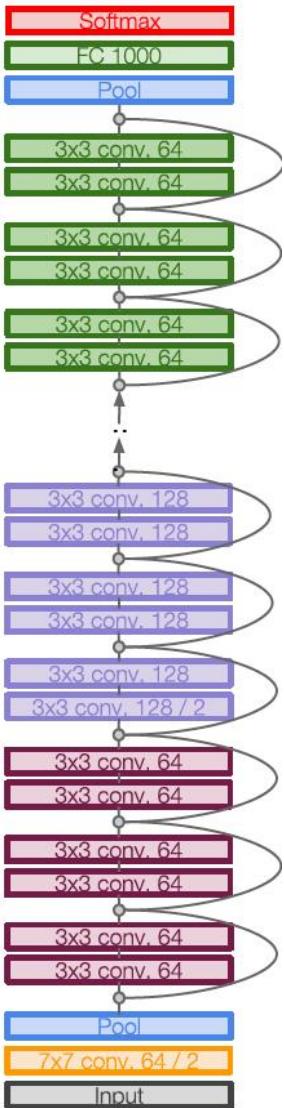
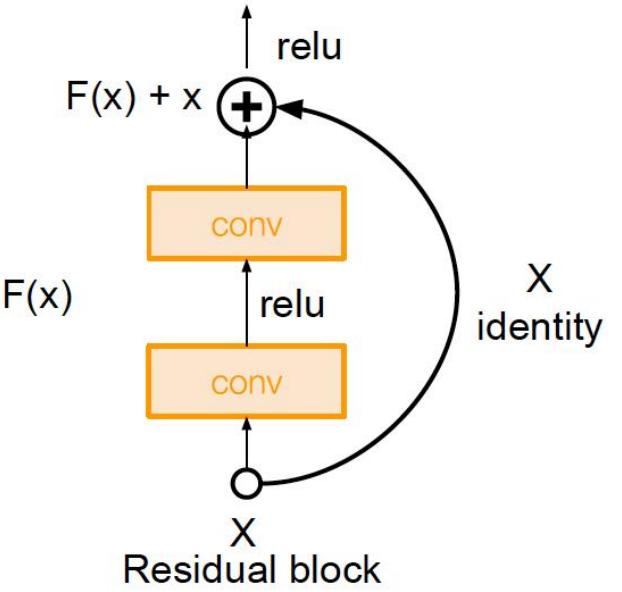


- VGG-16 是一种深度卷积神经网络（CNN），由牛津大学视觉几何组（Visual Geometry Group，简称VGG）的 Karen 等人在 2014 年提出
- VGG-16 因其结构的简洁性和一致性而著名，它主要由堆叠的小型 3×3 卷积核和 2×2 的最大池化层构成

- ResNet
 - 简单地堆叠层并不能带来性能提升。相反，它可能会损害性能



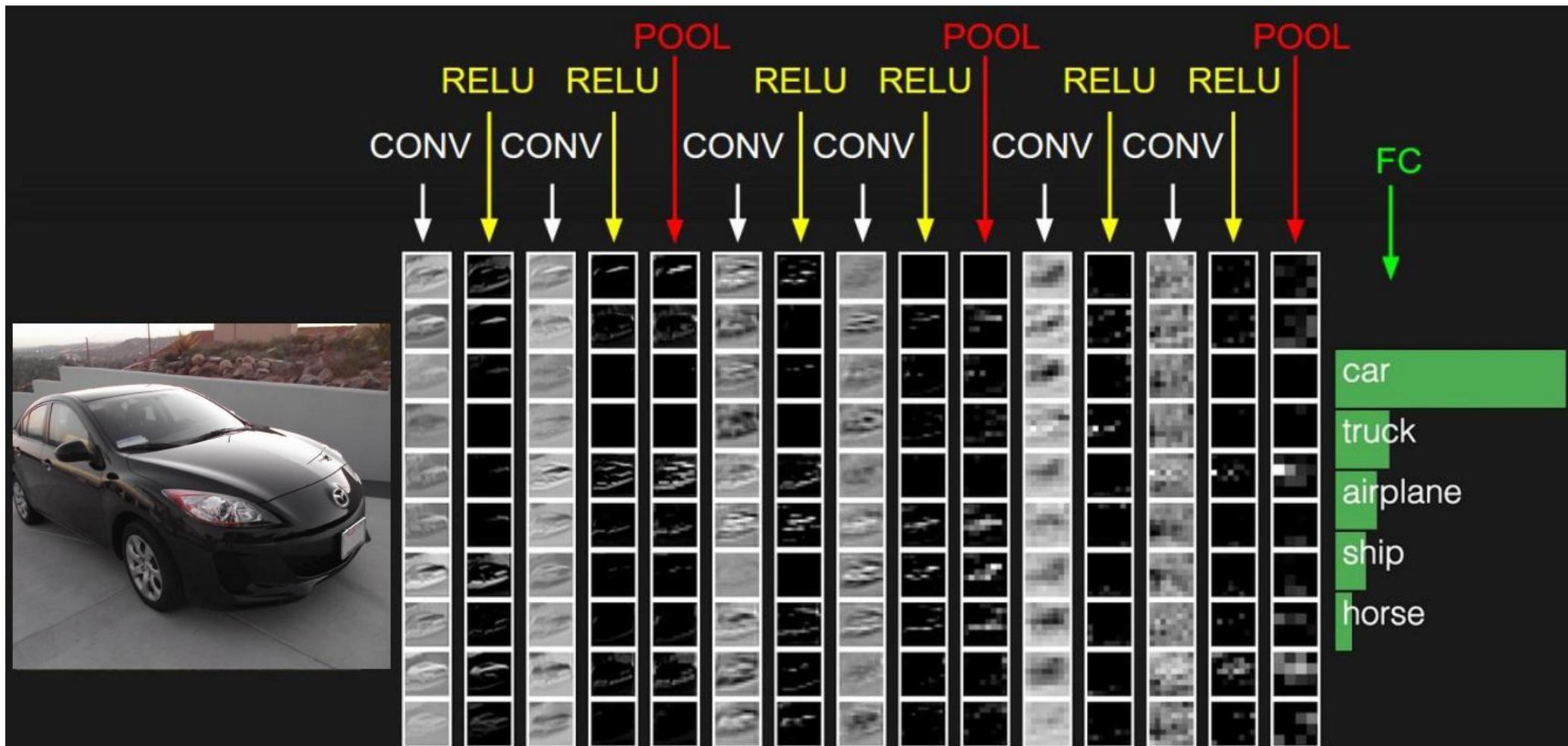
- 其中一个原因是更深的模型更难训练
- ResNet 通过引入层与层之间的“shortcut”连接来缓解训练困难



赢得 ILSVRC 2015 分类冠军的 ResNet 包含了多达 152 层!

卷积神经网络特征图可视化

■ 利用卷积神经网络进行图像分类



分类之外的典型应用

Semantic Segmentation



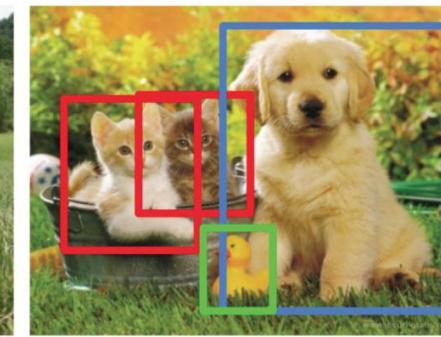
CAT

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK

原始循环神经网络

- 原始循环神经网络可以用两个方程来描述：

1) 隐藏状态更新方程

$$h_t = f_h(h_{t-1}, x_t)$$

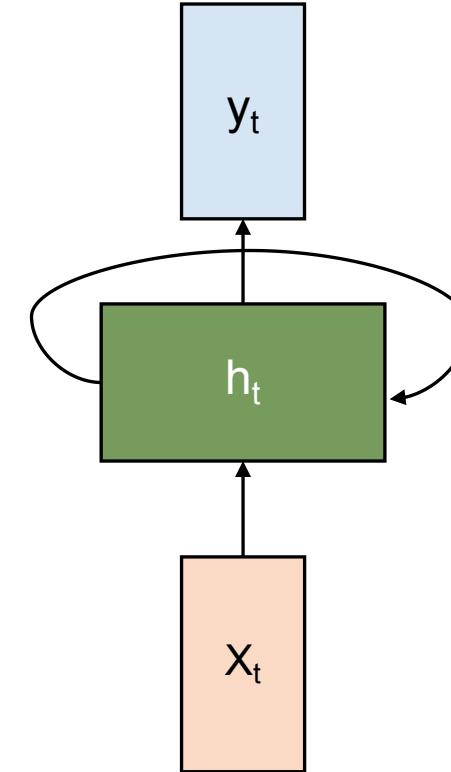
2) 输出

$$y_t = f_o(h_t)$$

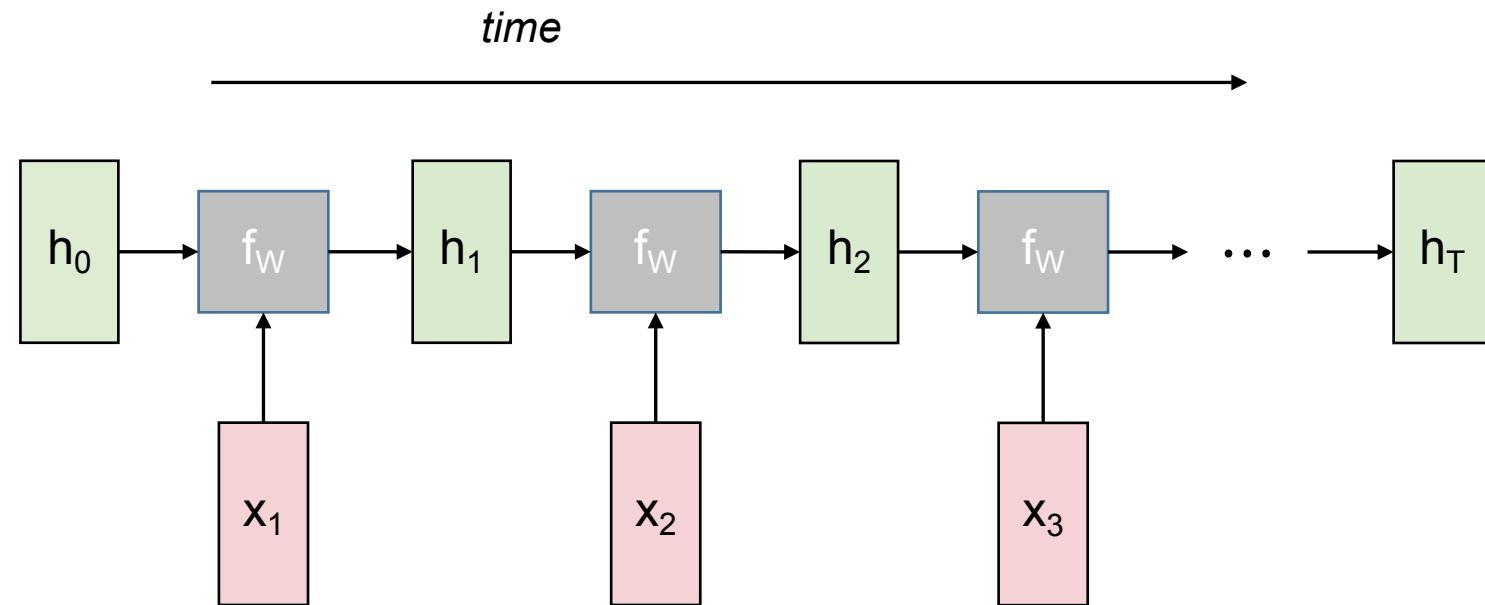
- 例如， $f_h(\cdot)$ 和 $f_o(\cdot)$ 可以是：

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t \text{ 或 } y_t = \text{softmax}(W_{hy}h_t)$$

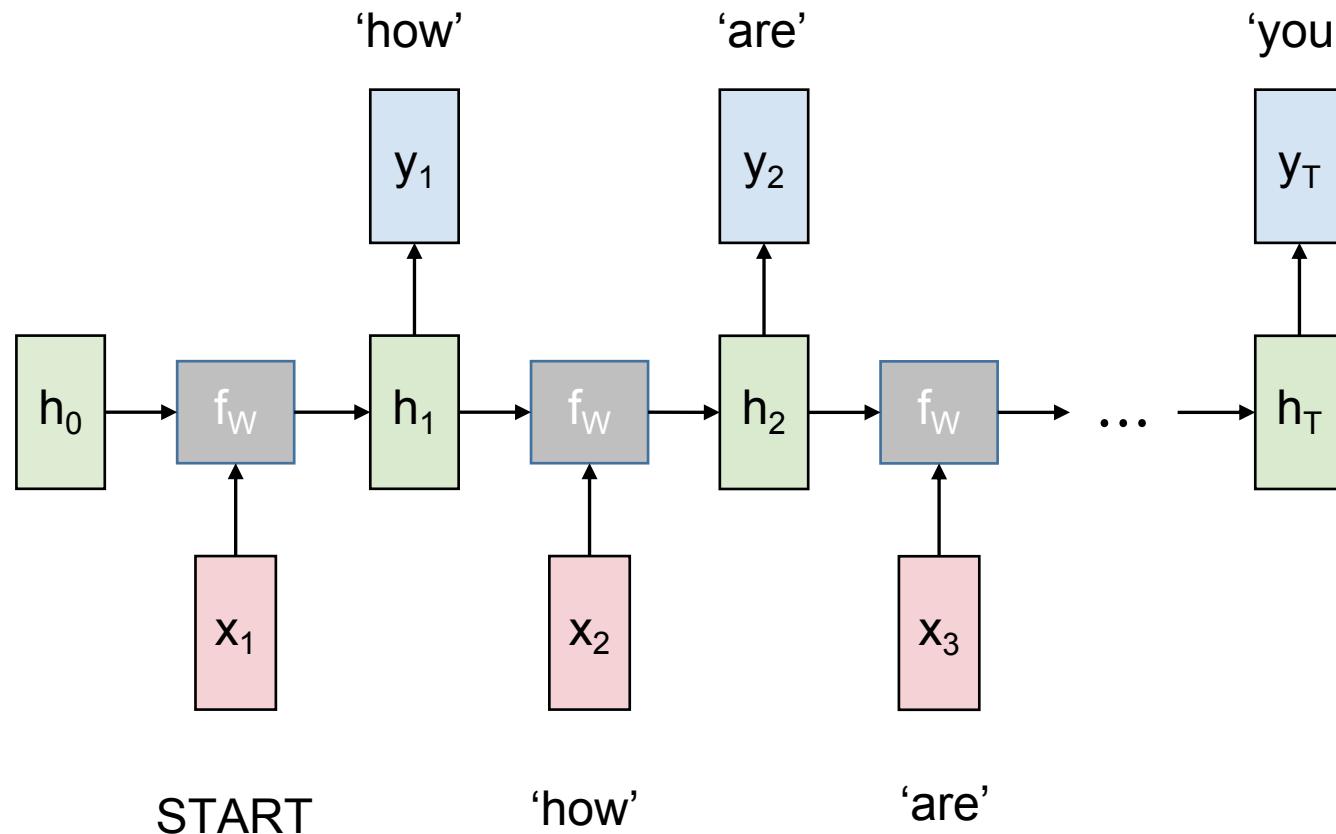


- 按时间展开循环神经网络



输出类型

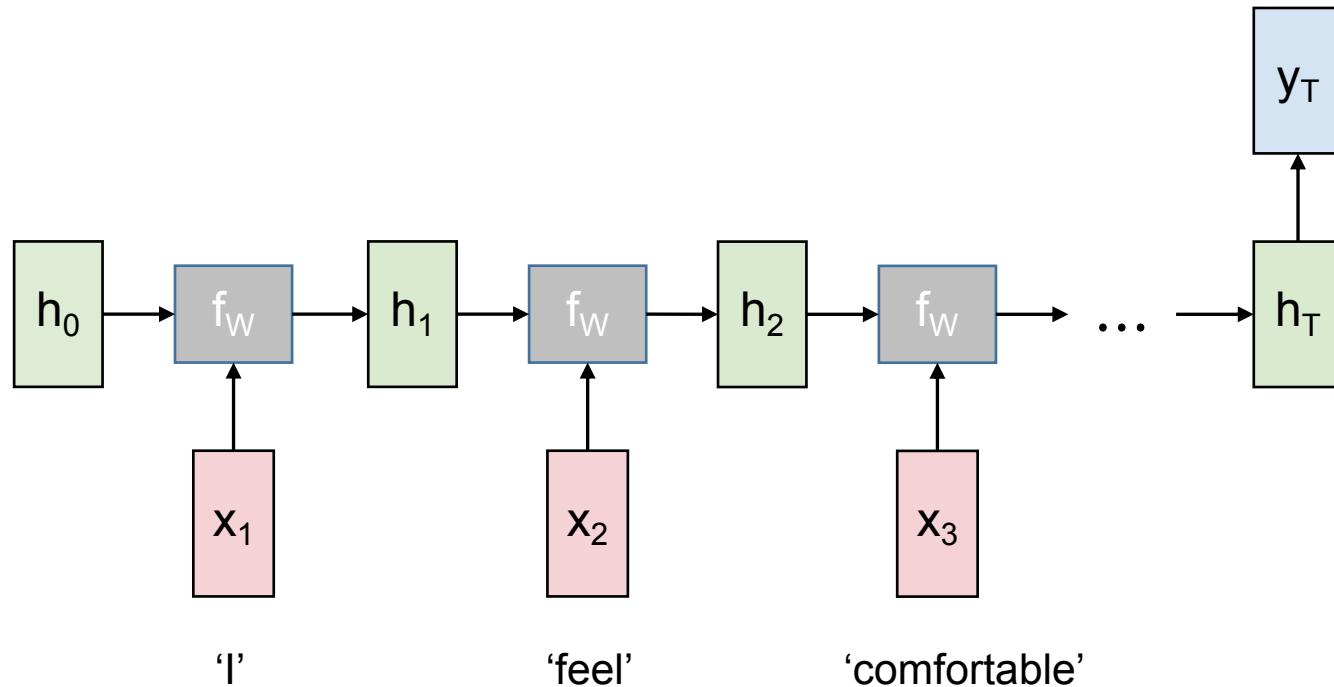
- 多对多



应用：语言模型、视频中的帧分类等

- 多对一

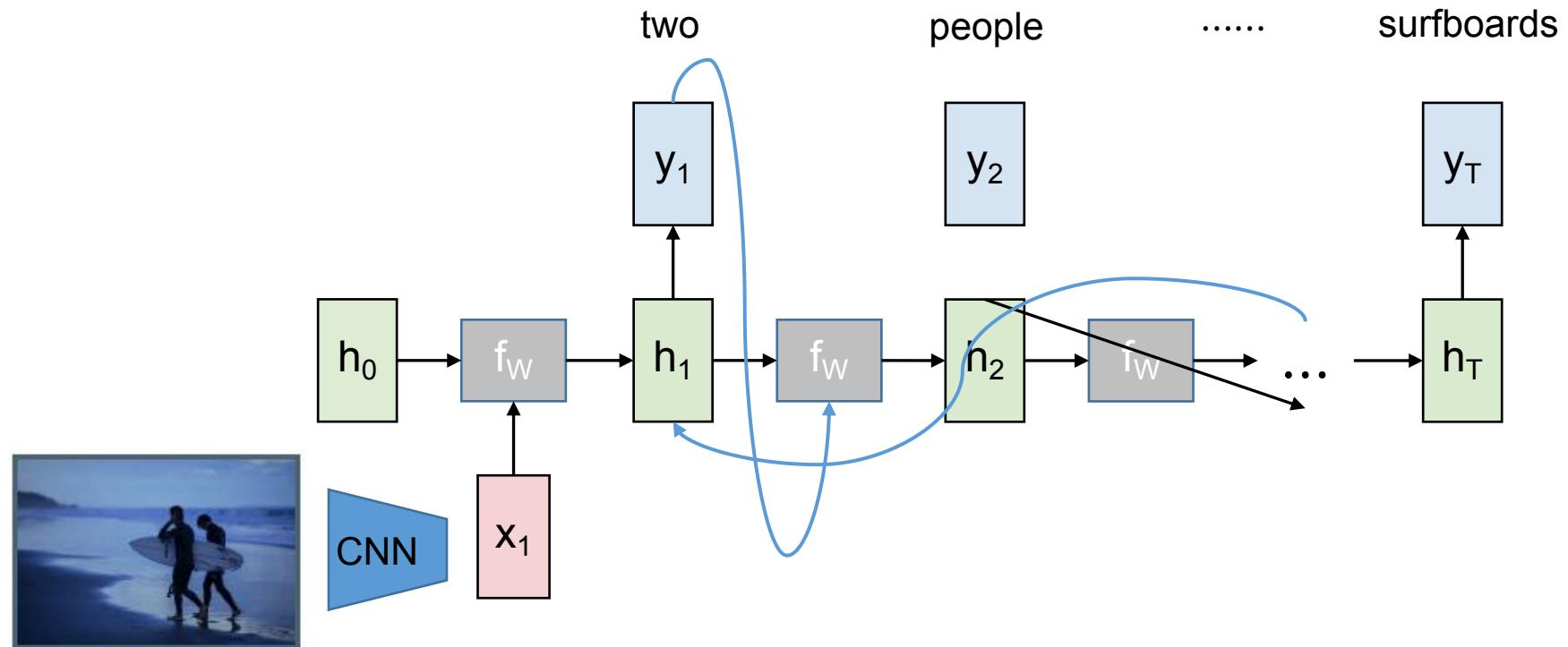
‘positive or negative’



应用：序列分类、情感分析等

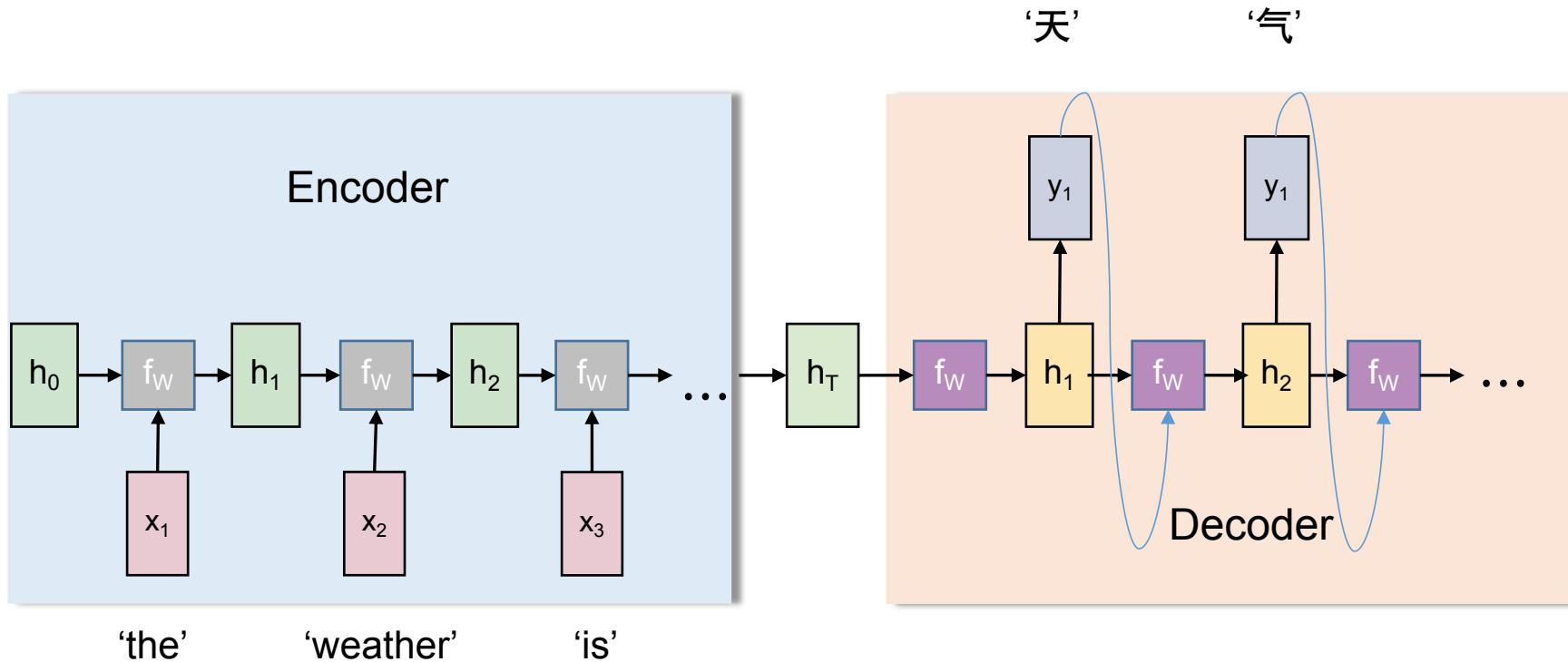
- 一对多

Two people walking on the beach with surfboards



应用：图像描述生成等

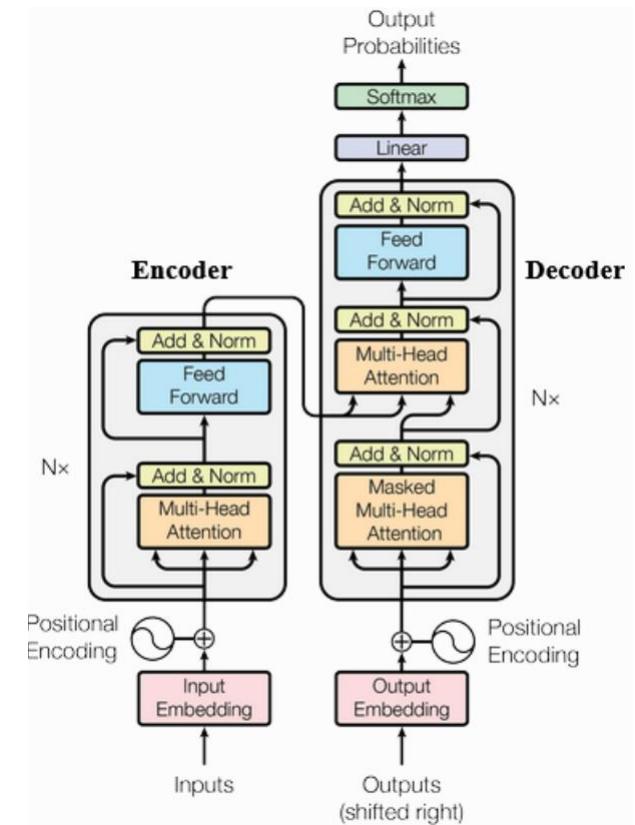
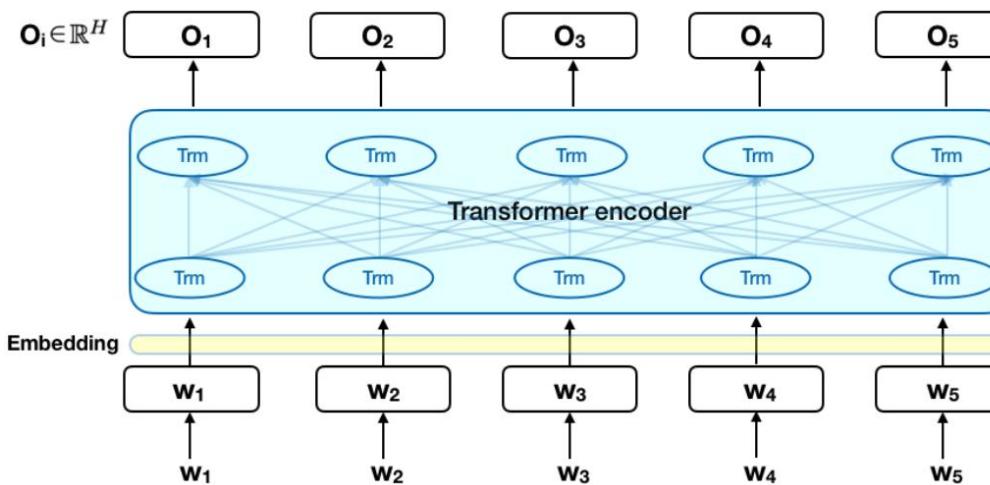
- 多对一 + 一对多

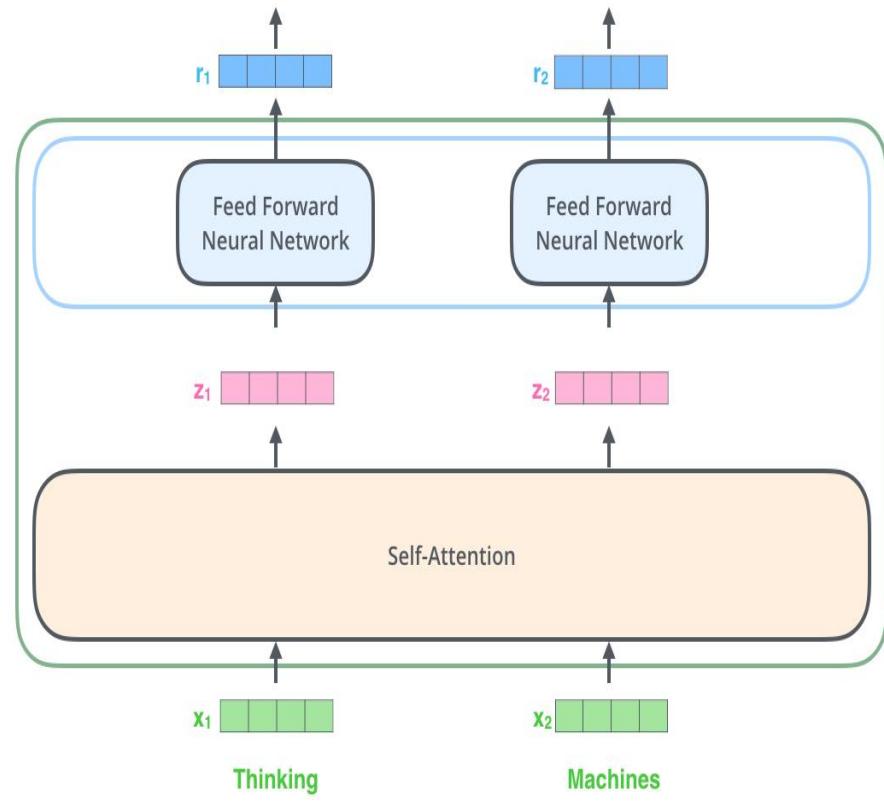
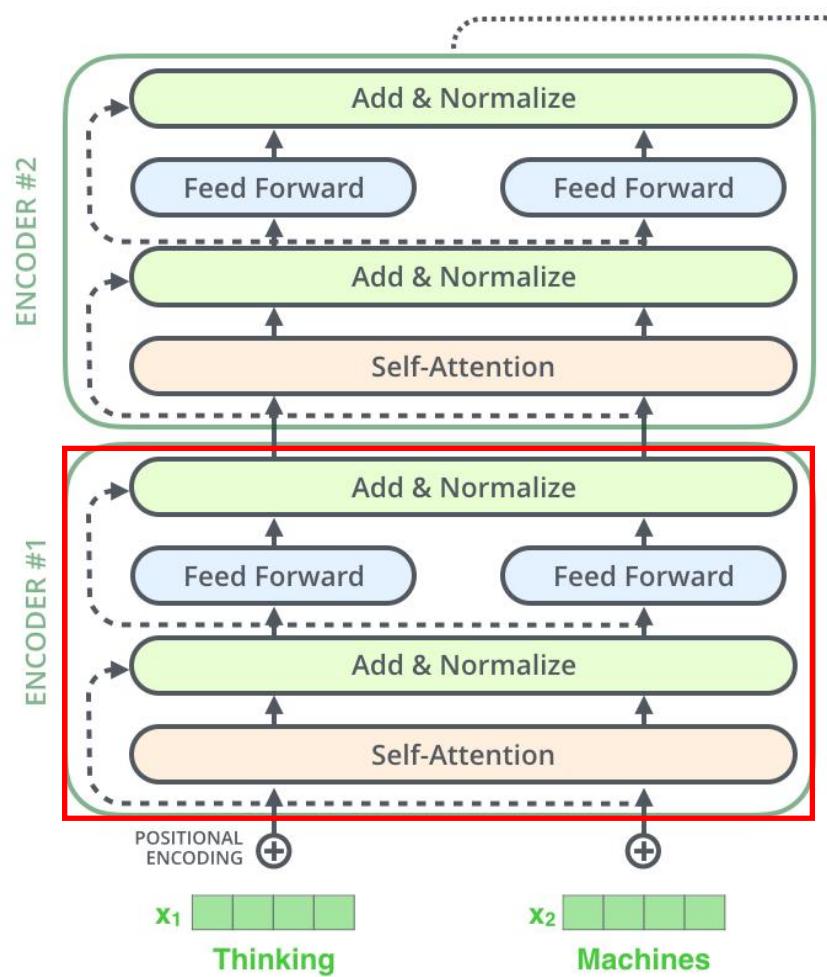


应用：机器翻译、句子特征提取等

基于 Transformer 的神经网络

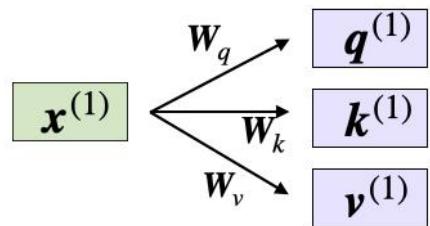
- 循环神经网络的问题
 - 1) 难以捕获文档中的长程依赖
 - 2) 只允许序列化执行，无法利用 GPU 中的并行计算资源
- BERT 和 GPT 模型都采用了基于 Transformer 的结构



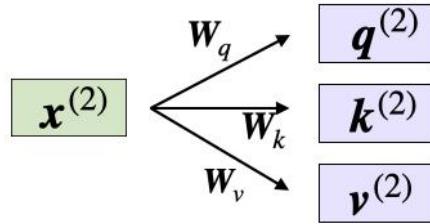
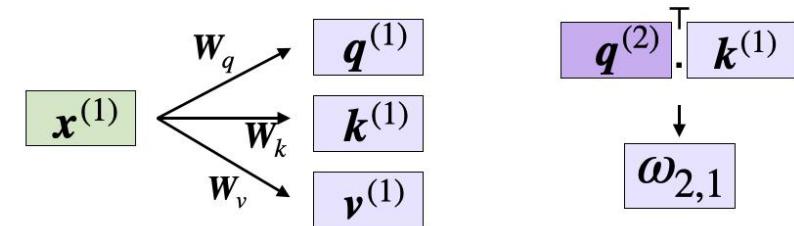


- 自注意力机制

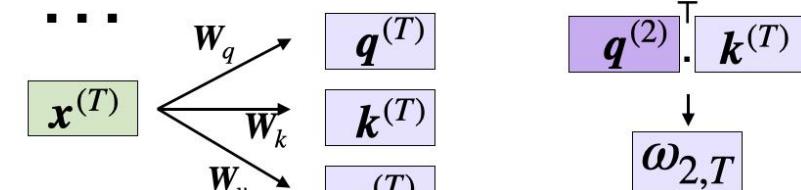
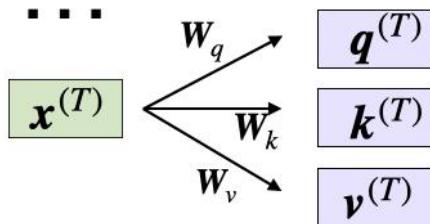
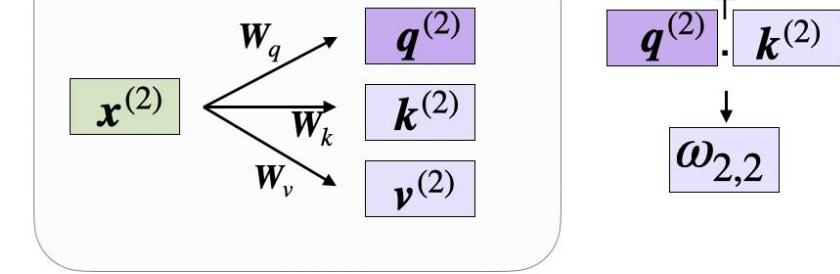
(1) query, key, value



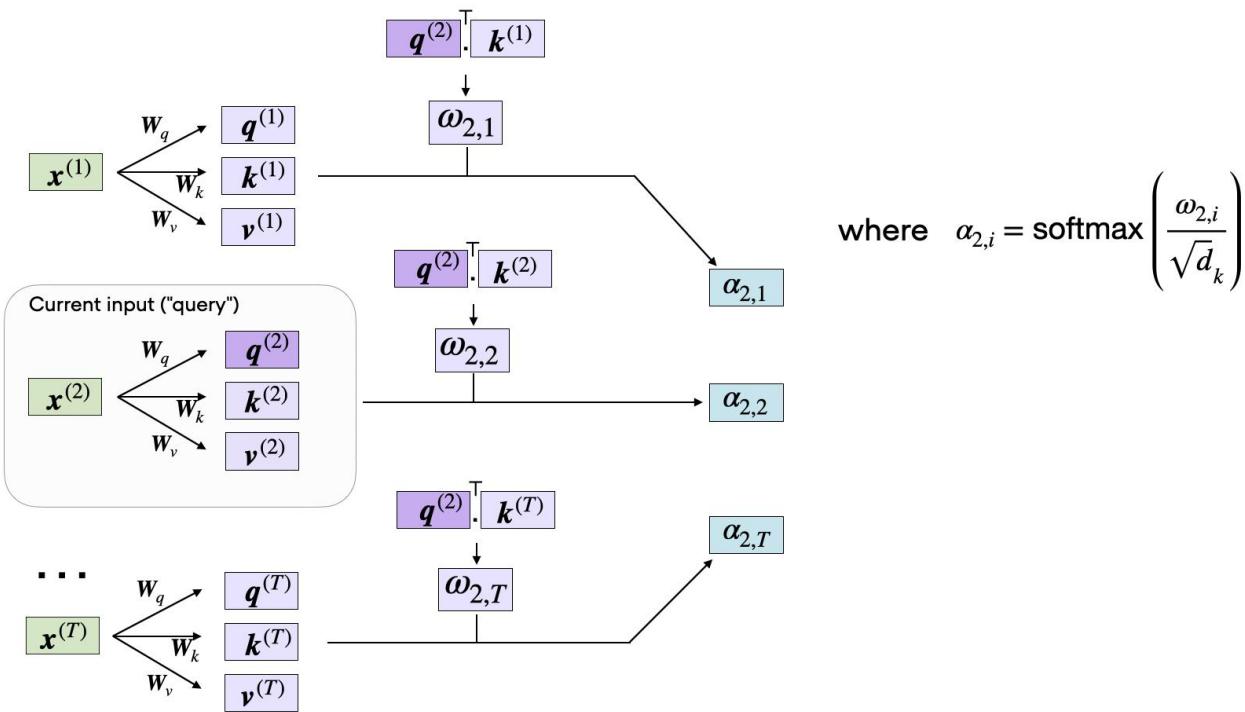
(2) query-key 相似度



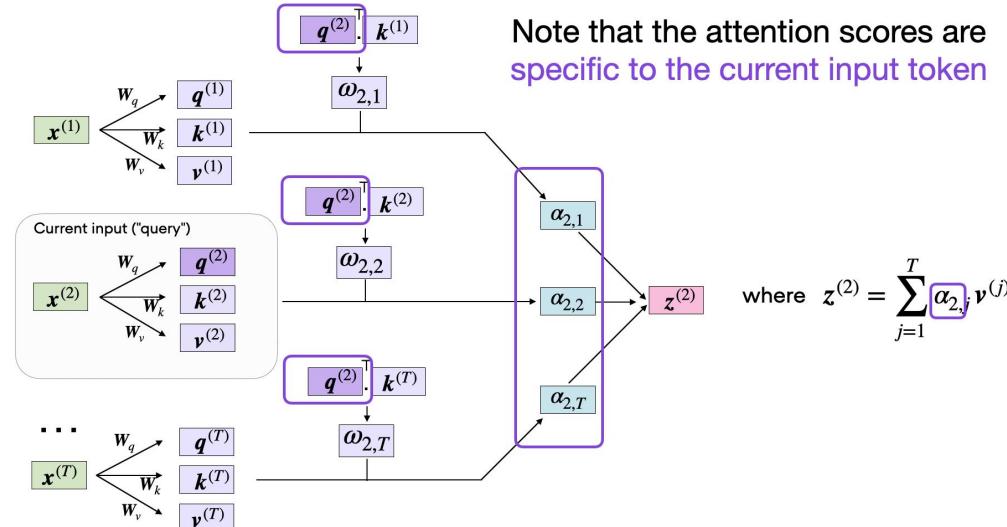
Current input ("query")



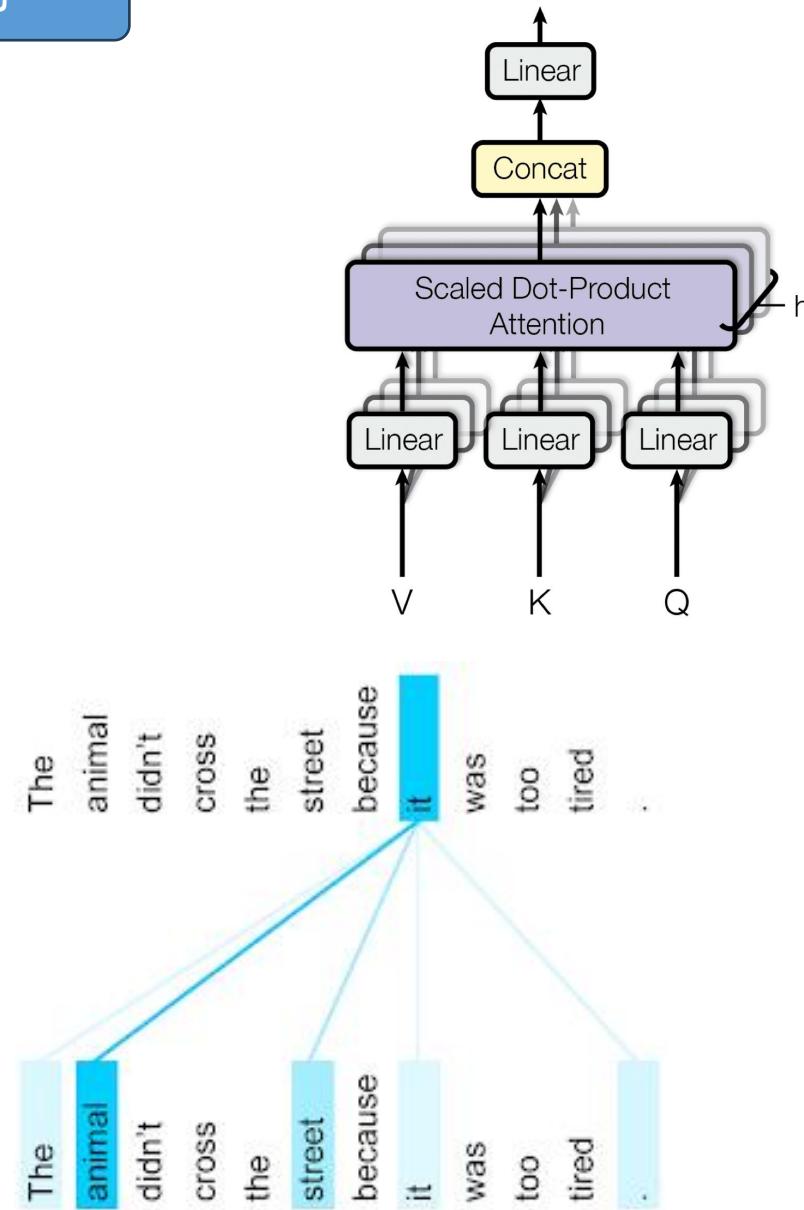
(3) 注意力权重



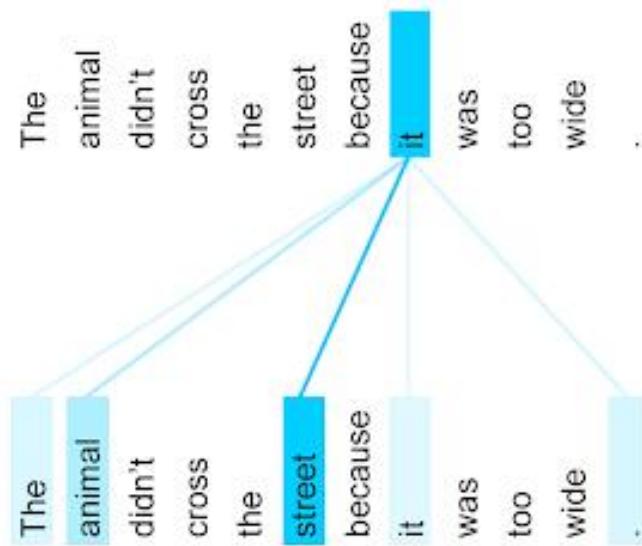
(4) 输出



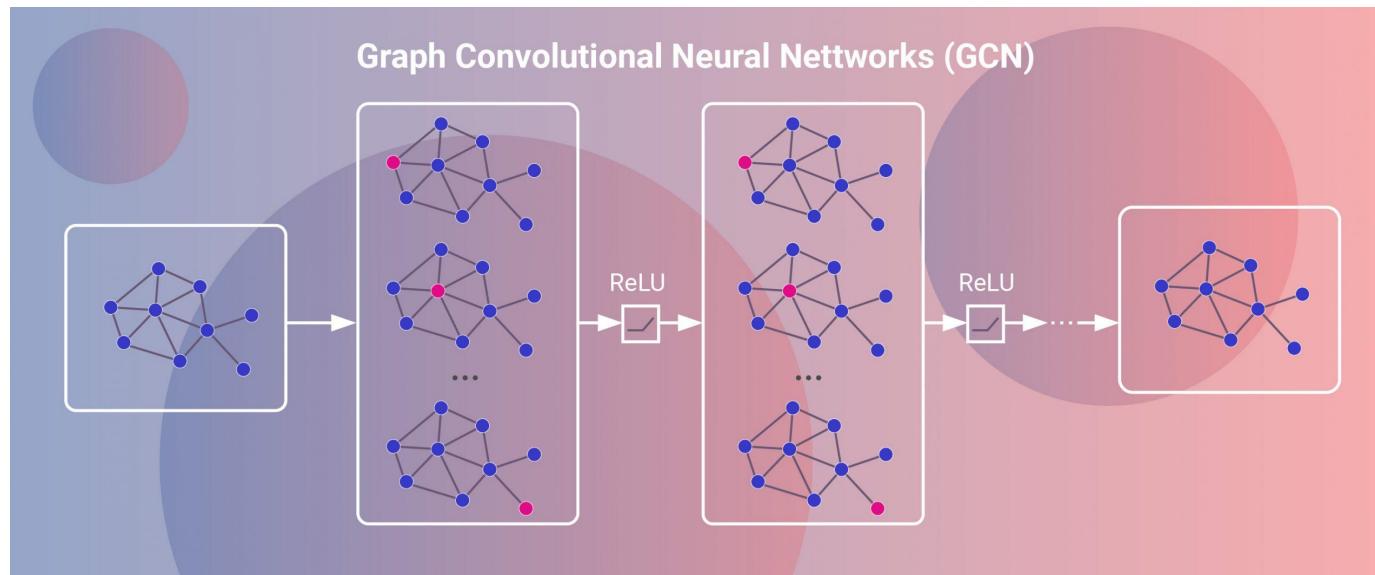
(5) 多头注意力



每个头负责捕获不同的含义



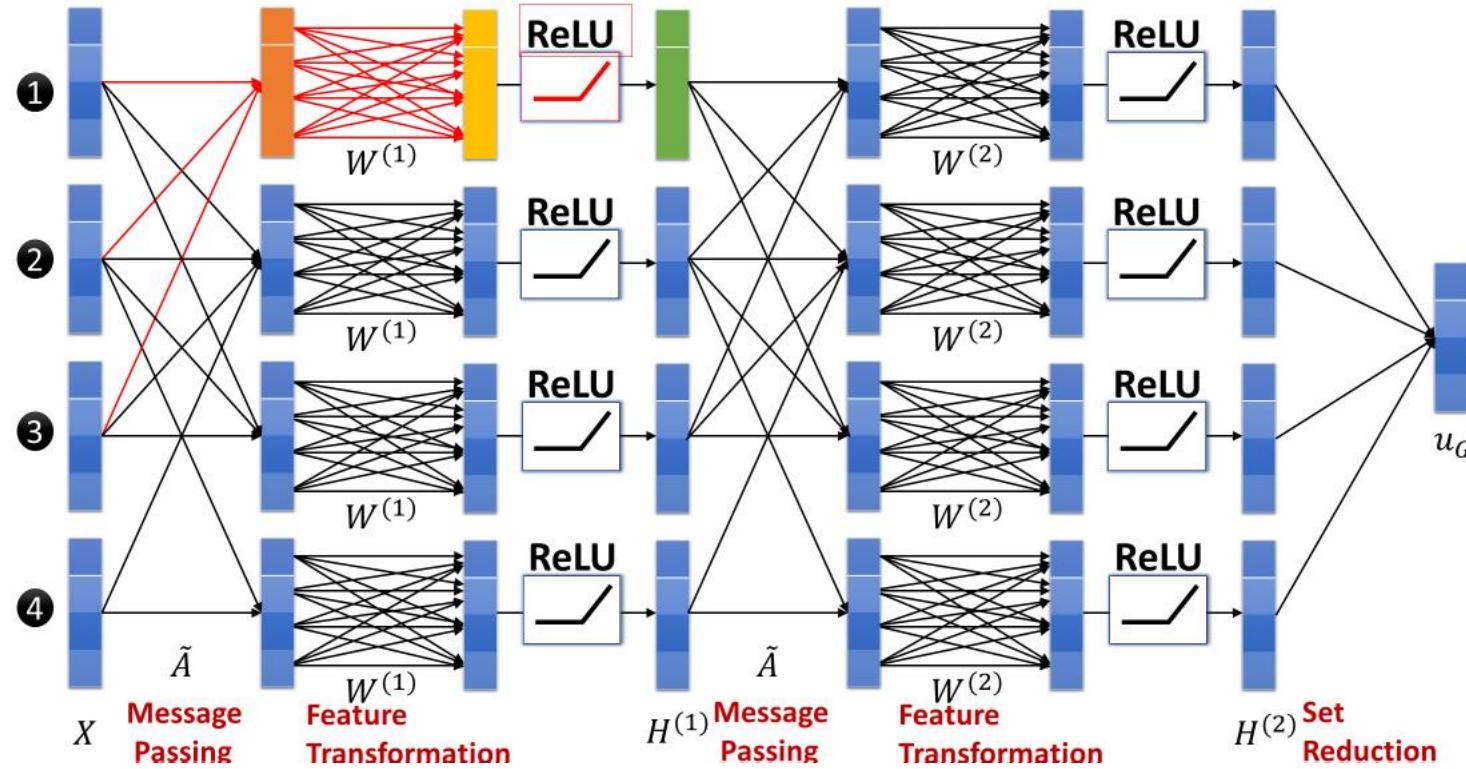
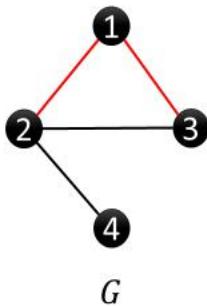
图神经网络



第 $(\ell + 1)$ 层中第 i 个节点的隐藏表示可以表示为：

$$\mathbf{h}_v^{\ell+1} = \sigma \left(\mathbf{W}^{\ell+1} \cdot \sum_{u \in \mathcal{N}(v) \cup v} \frac{\mathbf{h}_u^\ell}{\sqrt{|\mathcal{N}(u)| |\mathcal{N}(v)|}} \right)$$

一个在4节点图上的2层GCN的简单示例





机器学习与数据挖掘

神经网络的训练与优化



课程大纲

- ◆ 神经网络训练的总体框架
- ◆ 随机梯度下降算法

随机梯度下降 (SGD)

- 假设 w 是要优化的参数。随机梯度下降 (SGD) 中的更新方式是：

$$w_{t+1} = w_t - lr * \nabla f(w_t)$$

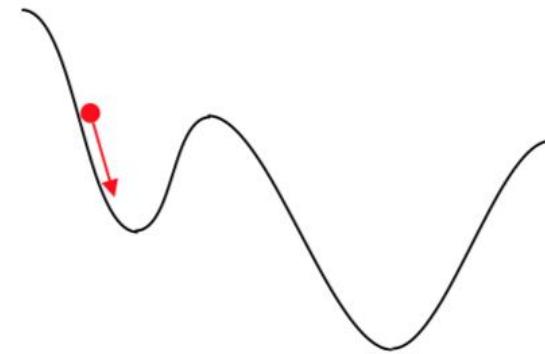
- $\nabla f(w)$ 是损失 l 的随机梯度，仅使用部分训练数据进行估计
- lr 是学习率



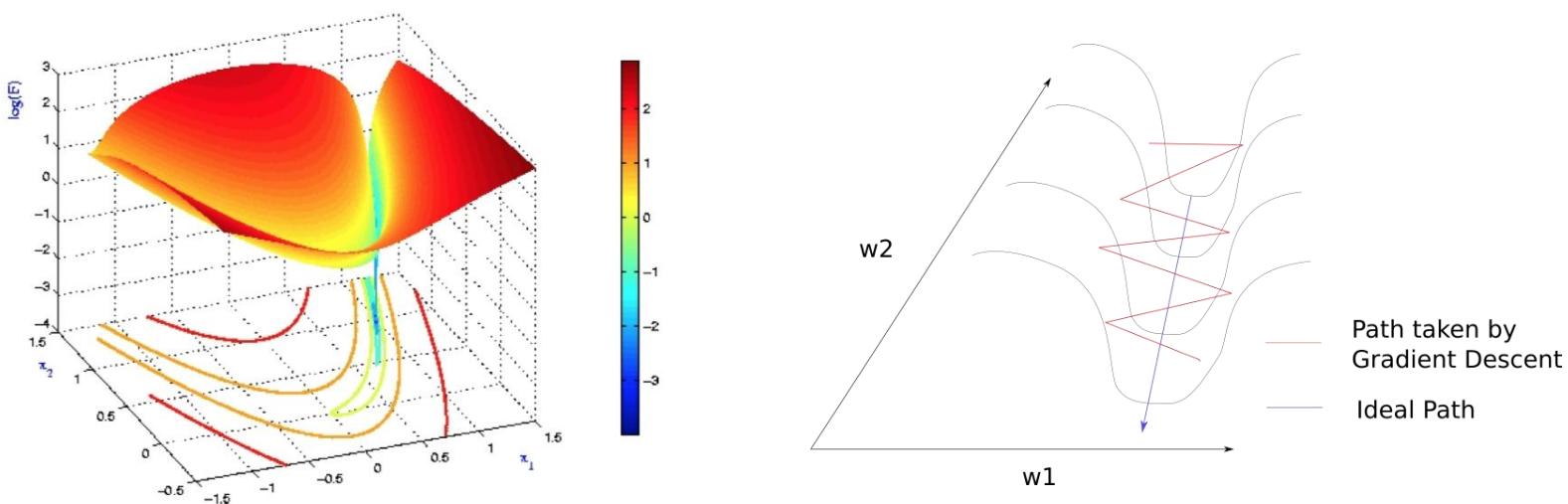
想象一步步地沿着**山坡**行走以降低高度，
最终走到山脉的最低处

- 随机梯度下降存在的问题

- 非常容易陷入局部最优



- 由于神经网络 (NNs) 中存在**病态曲率**, 导致收敛缓慢

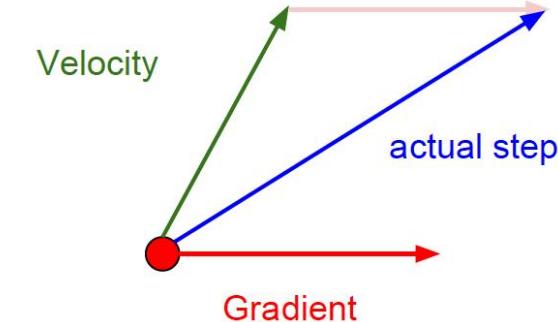


SGD + 动量

- 更新方程

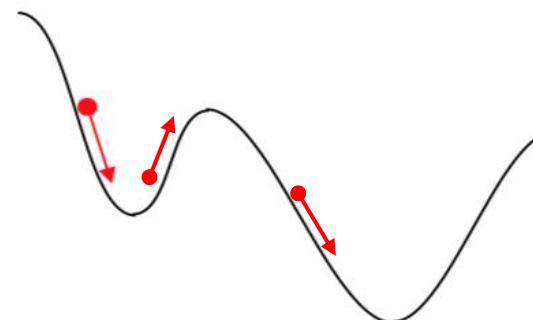
$$\mathbf{v}_t = \rho \mathbf{v}_{t-1} + \nabla f(\mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - lr \cdot \mathbf{v}_t$$



其中 $\rho \in (0, 1)$ 是衰减率，通常选择为 0.9, 0.95, 0.99

这个更新过程可以理解为一个球从崎岖的山坡上滚下，其摩擦力与 ρ 成正比



RMSProp

- 更新方程

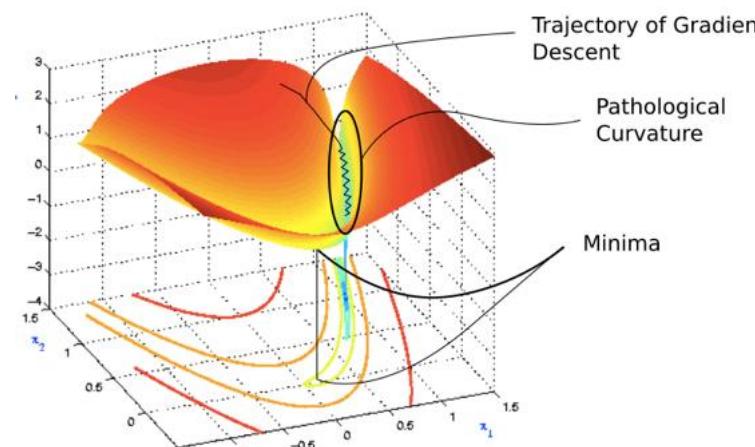
滑动平均

$$\mathbf{s}_t = \rho \cdot \mathbf{s}_{t-1} + (1 - \rho)(\nabla f(\mathbf{w}_t))^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - lr \cdot \nabla f(\mathbf{w}_t) \oslash \sqrt{\mathbf{s}_t}$$

其中 \mathbf{s}_t 是截至当前时间步的平方梯度加权平均； \oslash 表示逐元素除法

它将不同维度的梯度重新缩放到相同的水平，从而缓解了病态曲率问题



补充：滑动平均

$$\mathbf{s}_1 = \rho \cdot \mathbf{s}_0 + (1 - \rho) (\nabla f(\mathbf{w}_1))^2$$

$$\mathbf{s}_2 = \rho^2 \cdot \mathbf{s}_0 + \rho(1 - \rho) (\nabla f(\mathbf{w}_1))^2 + (1 - \rho) (\nabla f(\mathbf{w}_2))^2$$

加权

$$\mathbf{s}_3 = \rho^3 \cdot \mathbf{s}_0 + \rho^2(1 - \rho) (\nabla f(\mathbf{w}_1))^2 + \rho(1 - \rho) (\nabla f(\mathbf{w}_2))^2 + (1 - \rho) (\nabla f(\mathbf{w}_3))^2$$

⋮

对于较早的记录，其权重会衰减

可以验证，所有权重之和为 1，即：

$$(1 - \rho) + \rho(1 - \rho) + \rho^2(1 - \rho) + \rho^3(1 - \rho) + \dots$$

$$= \frac{1 - \rho^n}{1 - \rho} (1 - \rho) \rightarrow 1$$

Adam

- 更新方程

$$\mathbf{m}_t = \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \nabla f(\mathbf{w}_t)$$

$$\mathbf{s}_t = \beta_2 \cdot \mathbf{s}_{t-1} + (1 - \beta_2) (\nabla f(\mathbf{w}_t))^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - lr \cdot \mathbf{m}_t \oslash \sqrt{\mathbf{s}_t}$$

与 RMSProp 相同,
除了 \mathbf{m}_t

其中 \mathbf{m}_t 是过去梯度的滑动平均

它是 RMSProp 的一个改进版本, **用梯度的滑动平均 \mathbf{m}_t 替代了精确梯度 $\nabla f(\mathbf{w}_t)$**

- 默认设置: $\beta_1 = 0.9$, $\beta_2 = 0.999$ 和 $lr = 10^{-3}$
- 通过将 $\beta_1 = 0$, Adam 退化为 RMSProp

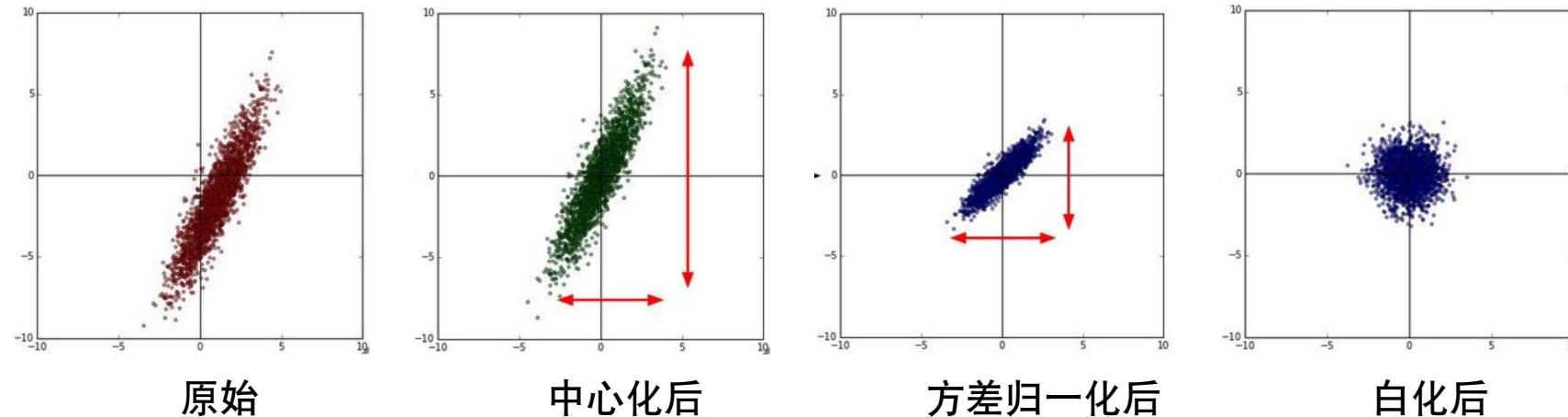
课程大纲

- 神经网络训练的总体框架
- 随机梯度下降算法

数据预处理

- 常用的数据预处理方法：

- 1) 数据**中心化**，即减去数据的均值
- 2) **方差标准化**，即对每个维度进行方差归一化
- 3) 数据**白化**



- 根据数据的特性，我们可以使用上述一种或几种方法来预处理数据

参数初始化

1) 通过从固定的高斯分布中随机抽样来初始化所有层的权重，例如：

$$W \sim \mathcal{N}(0, 0.01^2)$$

- 这种方法对于不那么深的神经网络效果不错（例如，层数小于 8）
- 如果使用 ReLU 作为激活函数，有些人也建议使用截断高斯分布，即：

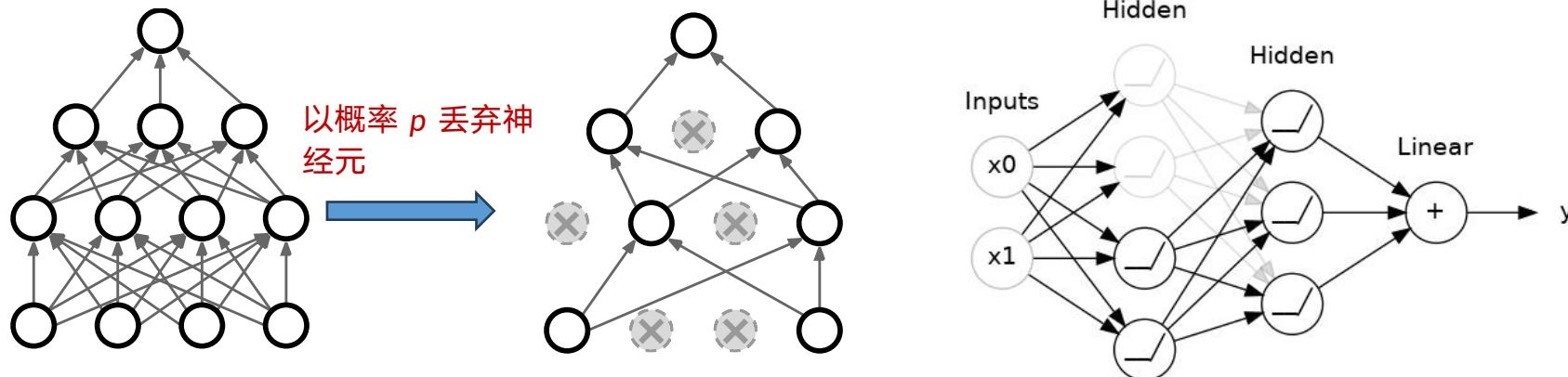
$$W \sim \mathcal{N}_T(0, 0.01^2)$$

2) 当网络变得更深时，可以通过一个高斯分布 $W \sim \mathcal{N}(0, \sigma_i^2)$ 进行初始化，其中每层的方差与前一层神经元的数量成反比，即：

$$\sigma_i = \frac{1}{\sqrt{\# \text{ neurons in } (i-1)\text{th layer}}}$$

Dropout

- 在每次前向传播中，随机将一些神经元设置为零
 - 丢弃神经元的概率是一个超参数；0.5 是一个常用的值

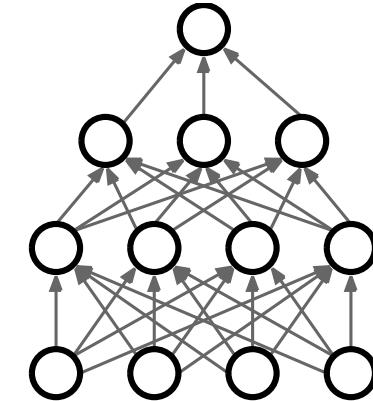


Dropout 可以有效缓解深度神经网络中常见的过拟合问题

- 测试时的 Dropout

➤ 在测试期间，我们倾向于让模型对一个输入输出一个固定的值

➤ 因此，在测试时应该移除神经元上的随机性，将它们全部打开



➤ 为了保持每个神经元在测试时的输入与训练时一致，我们需要用丢弃概率 p 来重新缩放这个值

$$\mathbf{h}_{\ell+1} = \sigma(\mathbf{W}_\ell \mathbf{h}_{\ell-1})$$

w/o dropout



$$\mathbf{h}_{\ell+1} = \sigma(p \cdot \mathbf{W}_\ell \mathbf{h}_{\ell-1})$$

w/ dropout

批量归一化

- 虽然好的初始化可以让网络在一开始处于正常状态，但其影响会随着训练的进行而减弱
- 一种更有效的方法是，在训练过程中有目的地将隐藏状态 $\{\tilde{h}_\ell^{(i)}\}_{i=1}^N$ 归一化到标准正态分布，如下所示：

$$\hat{h}_\ell^{(i)} = (\tilde{h}_\ell^{(i)} - \mu_\ell) \odot \sigma_\ell$$

其中

$$\mu_\ell = \frac{1}{N} \sum_{i=1}^N \tilde{h}_\ell^{(i)} \quad \sigma_\ell^2 = \frac{1}{N} \sum_{i=1}^N (\tilde{h}_\ell^{(i)} - \mu_\ell)^2$$

- 然后，归一化后的 $\hat{h}_\ell^{(i)}$ 会通过一个缩放系数 γ_ℓ 和一个偏移系数 β_ℓ 进行进一步变换

$$z_\ell^{(i)} = \gamma_\ell \odot \hat{h}_\ell^{(i)} + \beta_\ell$$

- 可以很容易地验证，如果 γ_ℓ 和 β_ℓ 分别被设置为 σ_ℓ 和 μ_ℓ ，那么 $z_\ell^{(i)}$ 将恢复到输入状态 $\tilde{h}_\ell^{(i)}$
- 系数 γ_ℓ 和 β_ℓ 与模型权重同时进行学习
- 计算精确的均值 μ_ℓ 和方差 σ_ℓ^2 成本很高。在实践中，它们是通过当前的小批量 B 来估计的

$$\mu_\ell = \frac{1}{|B|} \sum_{i \in B} \tilde{h}_\ell^{(i)} \quad \sigma_\ell^2 = \frac{1}{|B|} \sum_{i \in B} (\tilde{h}_\ell^{(i)} - \mu_\ell)^2$$

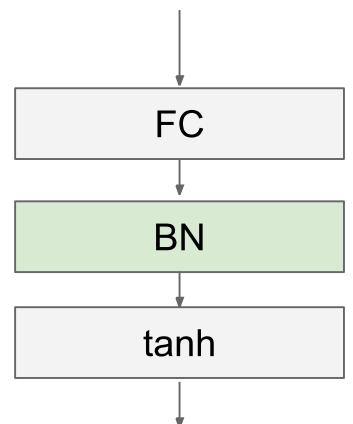
- 在测试阶段，我们不希望看到输出结果依赖于输入与哪个小批量相关联

- 有两种解决方案：

- 在整个训练数据集上计算精确的均值 $\mu_\ell = \frac{1}{N} \sum_{i=1}^N \tilde{h}_\ell^{(i)}$ 和方差 $\sigma_\ell^2 = \frac{1}{N} \sum_{i=1}^N (\tilde{h}_\ell^{(i)} - \mu_\ell)^2$ 。只需计算一次
- 使用滑动平均来根据基于小批量的均值和方差，估计出精确的均值和方差

- 批量归一化可以被看作一个用 $BN(\cdot)$ 表示的层，即：

$$\gamma_\ell \odot (\tilde{h}_\ell^{(i)} - \mu_\ell) \oslash \sigma_\ell + \beta_\ell = BN(\tilde{h}_\ell^{(i)})$$



- 批量归一化在 MLP 和 CNN 中的区别

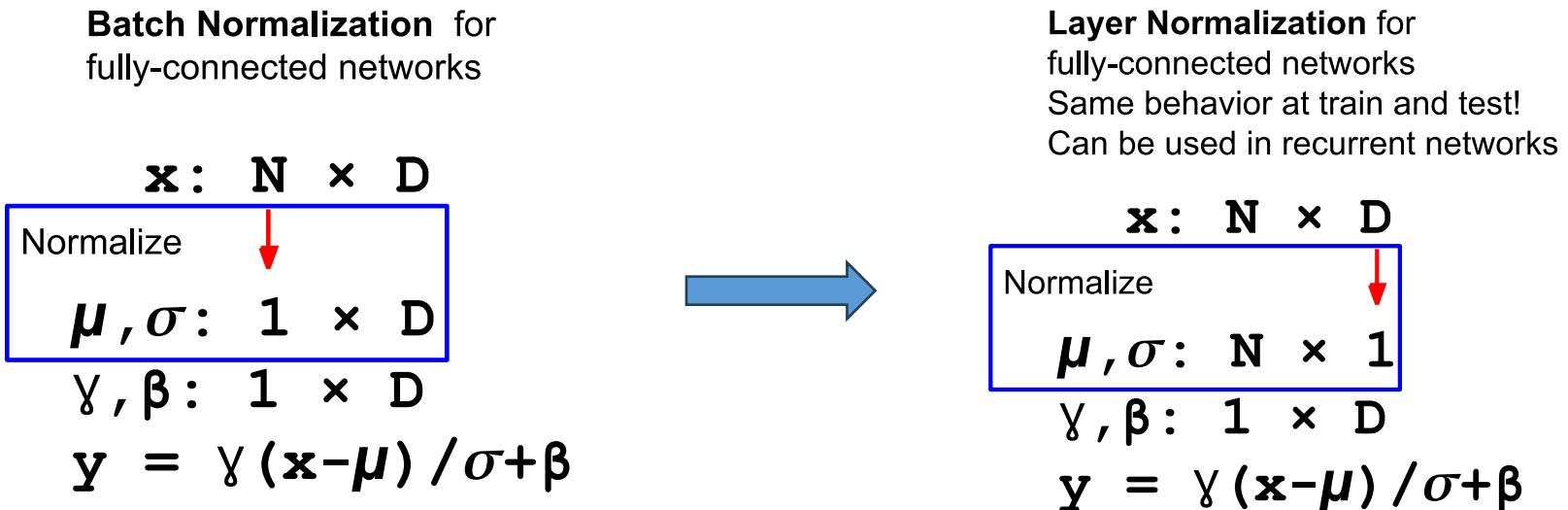
Batch Normalization for
fully-connected networks

$$\begin{aligned} \mathbf{x}: & N \times D \\ \text{Normalize} & \downarrow \\ \boldsymbol{\mu}, \sigma: & 1 \times D \\ \gamma, \beta: & 1 \times D \\ \mathbf{y} = & \gamma(\mathbf{x}-\boldsymbol{\mu}) / \sigma + \beta \end{aligned}$$

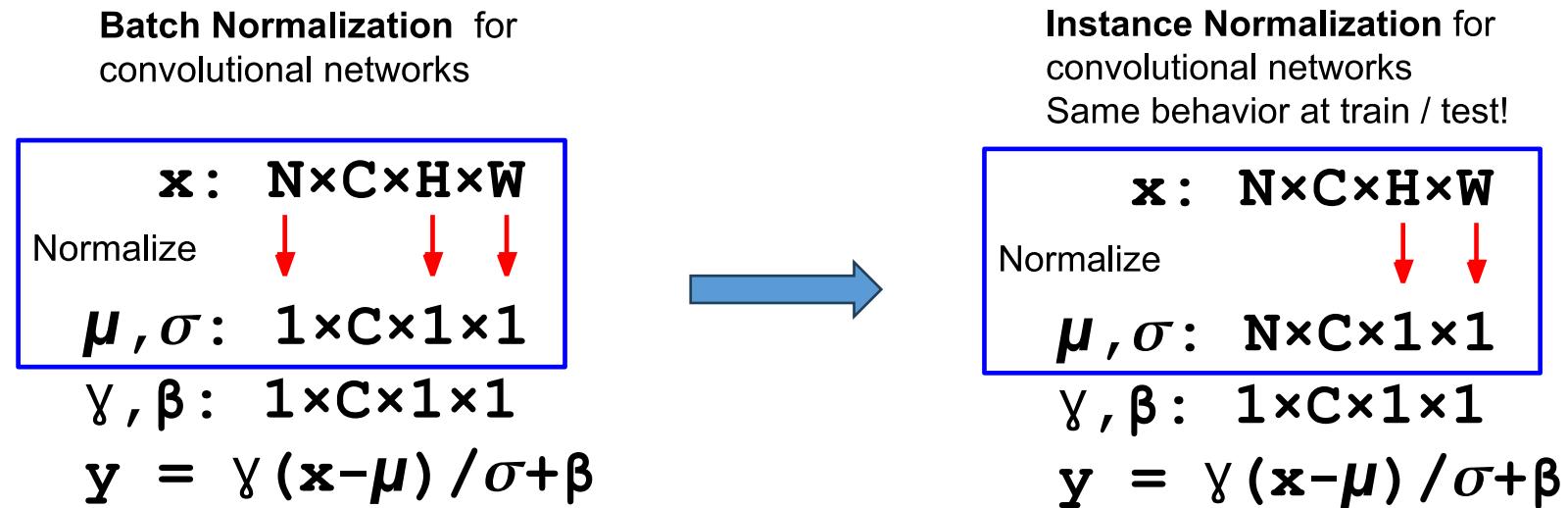
Batch Normalization for
convolutional networks
(Spatial Batchnorm, BatchNorm2D)

$$\begin{aligned} \mathbf{x}: & N \times C \times H \times W \\ \text{Normalize} & \downarrow \quad \downarrow \quad \downarrow \\ \boldsymbol{\mu}, \sigma: & 1 \times C \times 1 \times 1 \\ \gamma, \beta: & 1 \times C \times 1 \times 1 \\ \mathbf{y} = & \gamma(\mathbf{x}-\boldsymbol{\mu}) / \sigma + \beta \end{aligned}$$

- 层归一化

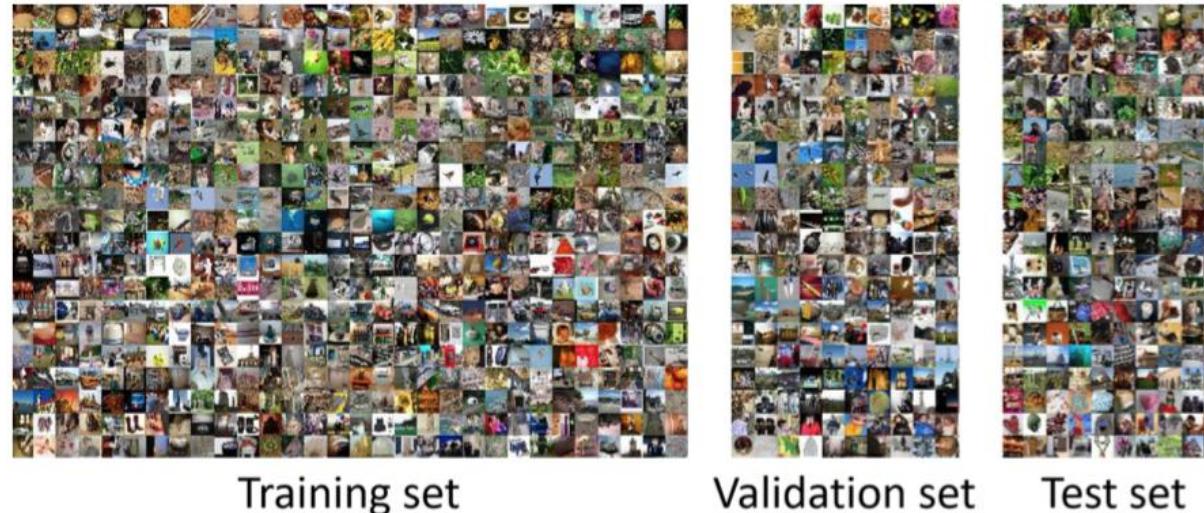


- 实例归一化



超参数调优

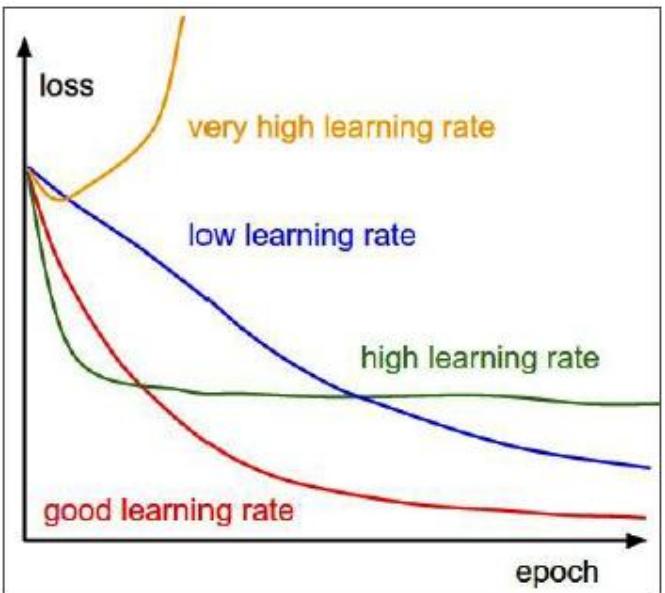
- 在开始训练之前，有很多参数需要设置，例如学习率、RMSProp 和 Adam 中的衰减参数、丢弃率等
- 为了给这些超参数设置合适的值，我们需要将整个数据集分为三部分，即训练集、验证集和测试集



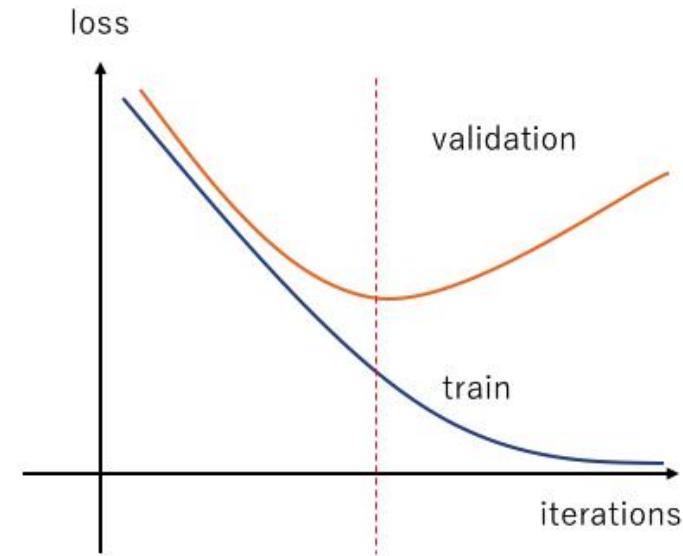
- 在训练集上训练模型
- 根据在验证集上的表现调优超参数
- 在测试集上测试性能，并将其作为最终性能进行报告

- 超参数的调整不是盲目的，而是可以遵循一些指导原则

- 首先尝试一些在许多任务上表现良好的默认值
- 根据训练过程中观察到的现象来选择值



训练损失曲线暗示了良好学习率的可能范围



训练集和验证集上的表现关系表明模型过于灵活



决策树与集成方法

主讲人：林惊



决策树课程大纲

- 介绍
- 选择扩展属性的准则
- 决策树学习

什么是决策树？

- 给定下面的数据集，我们希望基于属性来预测其结果

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 与之前的分类器不同，决策树通过构建属性树将数据分到不同的类别中

- 决策树（DT）的结构

- 内部节点对应于属性

- 叶子节点对应于结果

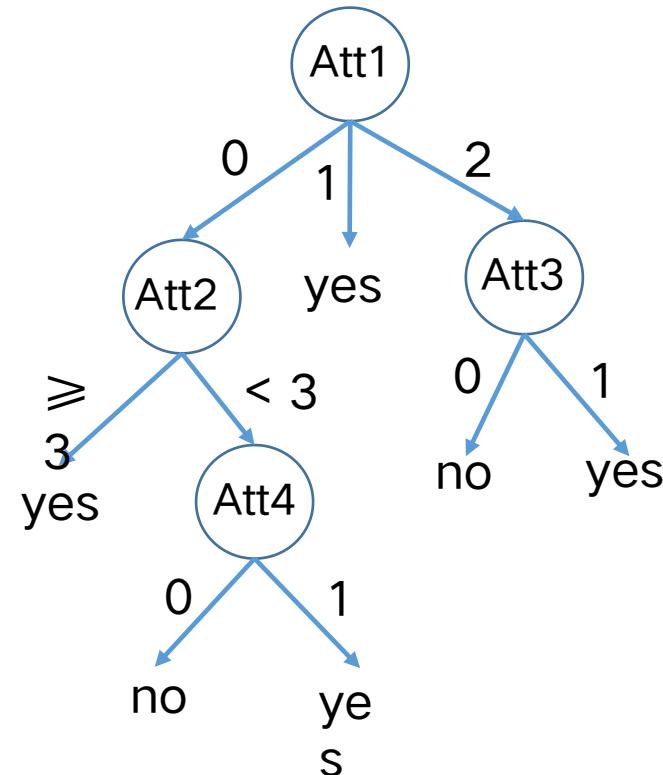
- 边对应于属性值

- 学习决策树

- 在每个节点使用哪个属性？

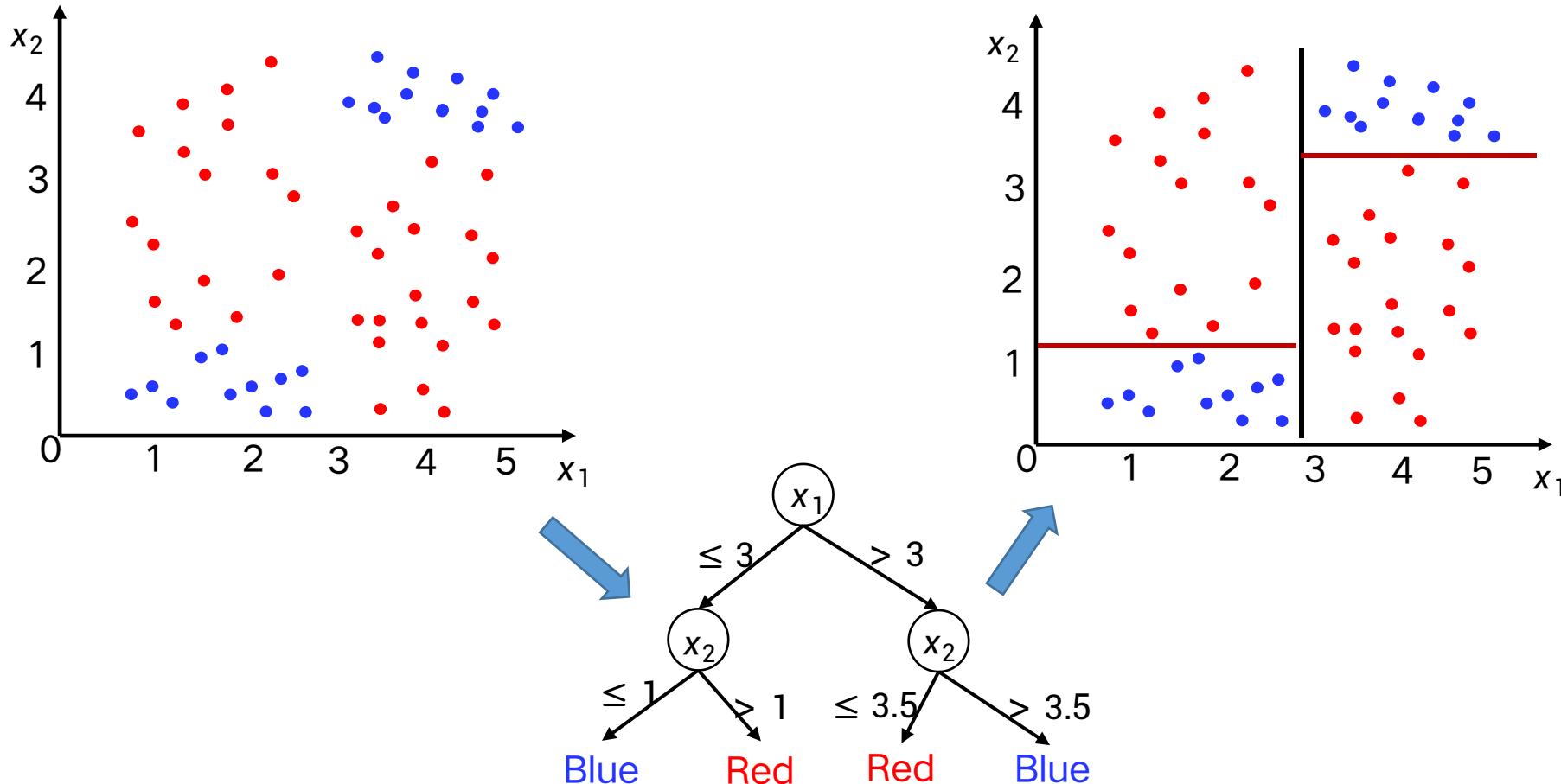
- 如何划分属性值？

- 何时停止扩展树？

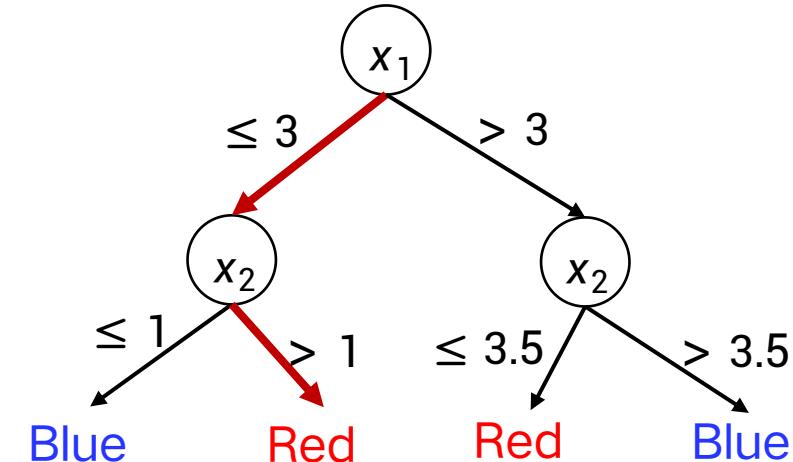
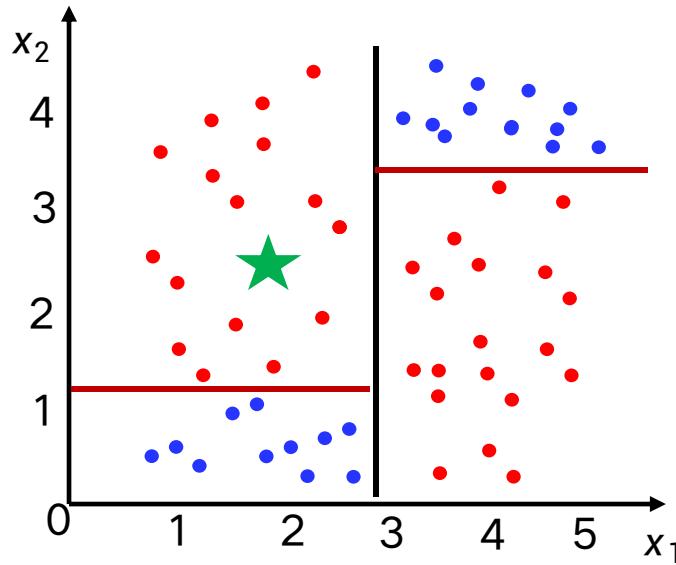


决策树学习：示例1

- 给定一个包含两个属性 x_1 和 x_2 的数据集，如下图所示，我们希望构建一个决策树来对实例进行分类



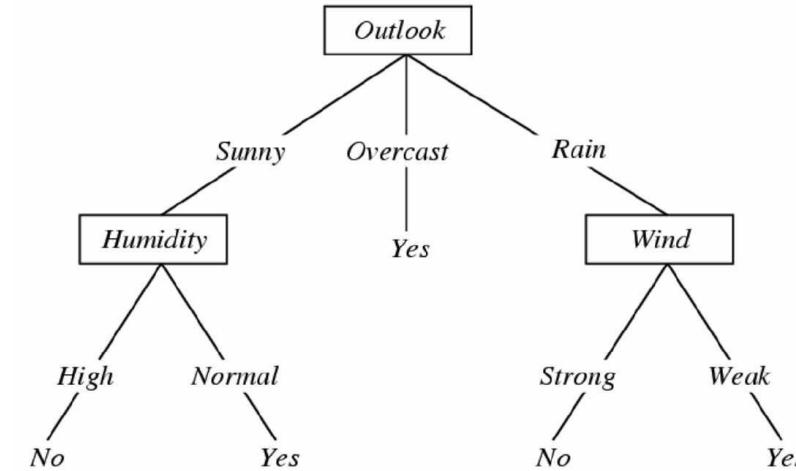
- 根据这个例子，为了构建这棵树，我们需要决定：
 - 在每一步使用哪个属性
 - 如何划分属性值
 - 当树已经构建好之后，对于一个新的输入数据，可以根据其属性值找到一条从根节点到叶子节点的路径，并从中读取其类别



决策树学习：示例2

- 给定下面的数据集，构建一个决策树，根据四个属性值来决定是否去打球

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



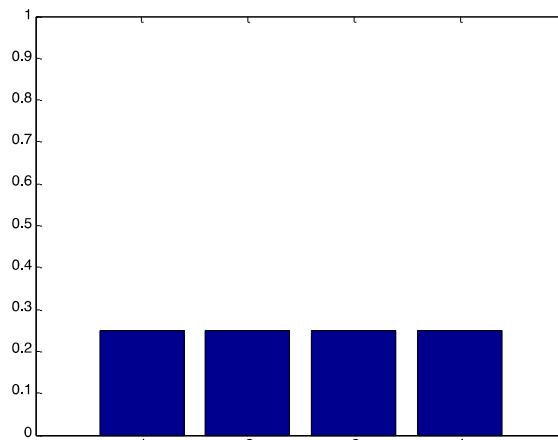
- 问题
 - a) 为什么首先选择 Outlook，然后是 Humidity 和 Wind？
 - b) 为什么在 Humidity 和 Wind 之后不再进一步扩展？
 - c) 为什么 Temperature 属性没有出现在树中？

课程大纲

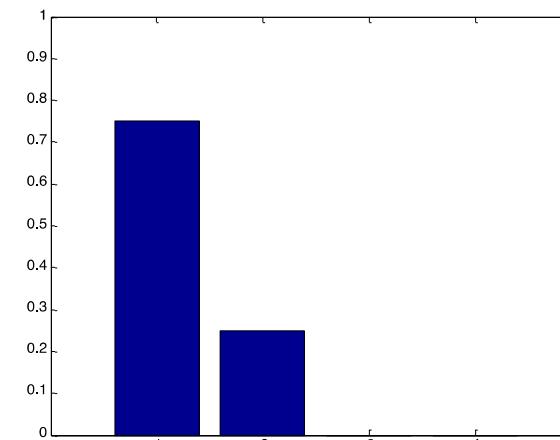
- 介绍
- 选择扩展属性的准则
- 决策树学习

- 在所有属性中，选择包含**信息量最多**的那个来扩展
- 如何衡量一个属性所包含的信息量？

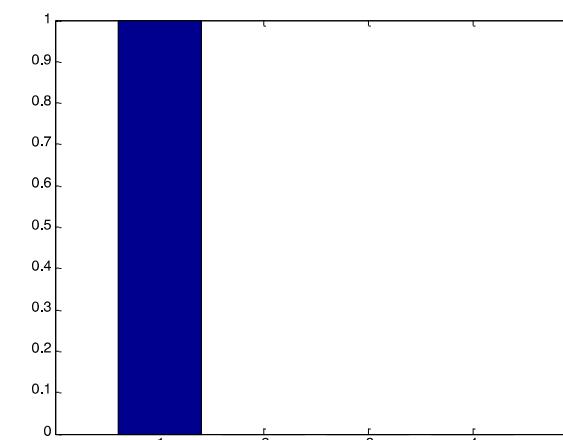
我们首先需要衡量一个随机变量的**不确定性**



(a)



(b)



(c)

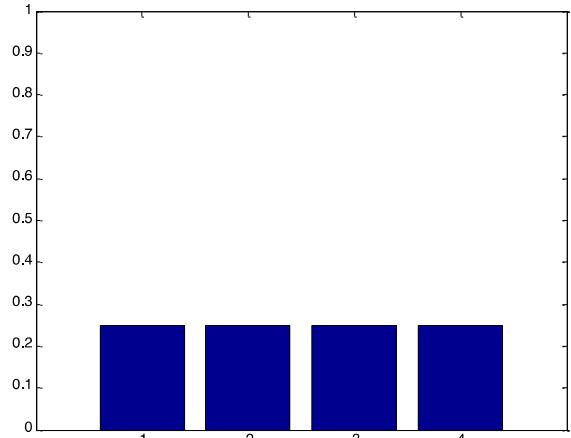
- 在数学中，一个随机变量的不确定性是通过**熵**来衡量的
- 定义：给定一个服从分布 $p(z)$ 的随机变量 Z ，其熵被定义为：

$$H(Z) = - \sum_{z \in \mathcal{C}} p(z) \log_2 p(z)$$

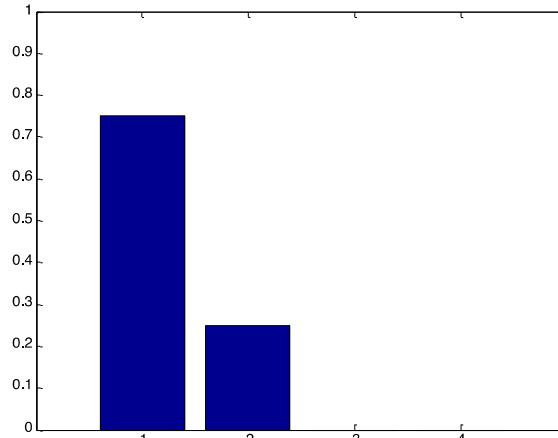
- \mathcal{C} 是随机变量 Z 的所有可能取值的集合

- **示例**

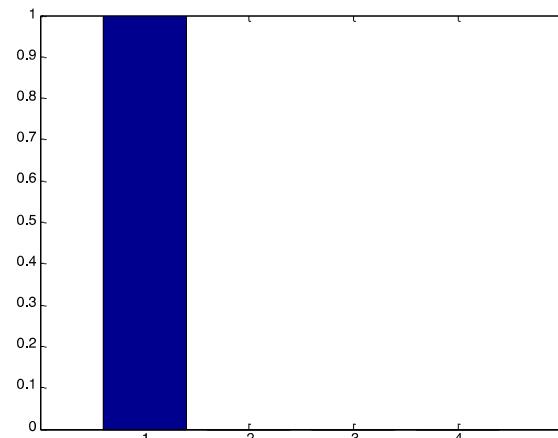
下面哪个分布的熵最大？



(a)



(b)



(c)

a) $H(Z) = -4 \times 0.25 \log_2 0.25 = 2 \text{ bits}$

b) $H(Z) = -0.75 \log_2 0.75 - 0.25 \log_2 0.25 \approx 0.8133 \text{ bits}$

c) $H(Z) = -1 \log_2 1 = 0 \text{ bits}$

分布 a) 的熵最大，而 c) 的熵最小

- 示例：二分类问题（伯努利熵）

- 伯努利分布：

$$P(Z = 1) = p; \quad P(Z = 0) = 1 - p$$

- 根据熵的定义：

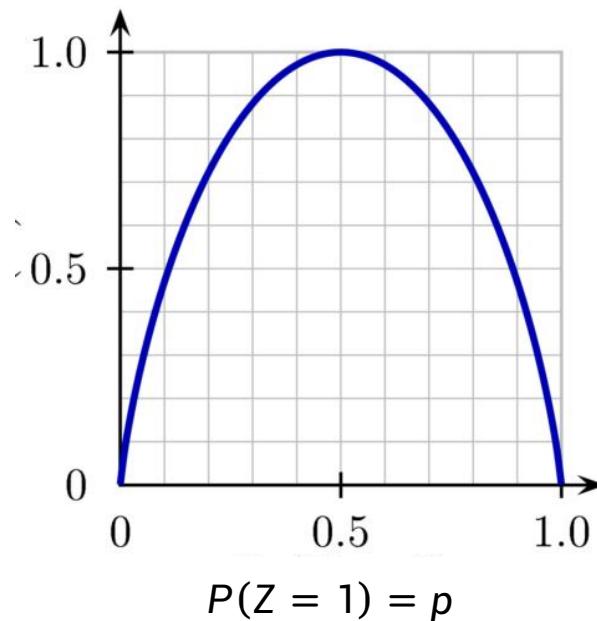
$$H(Z) = - \sum_{z \in \mathcal{C}} p(z) \log_2 p(z)$$

- 伯努利熵：

$$H(Z) = - [p \log_2 p + (1 - p) \log_2 (1 - p)]$$

- 伯努利随机变量的熵作为概率 $P(Z = 1)$ 的函数

$$H(Z) = -[p \log_2 p + (1-p) \log_2 (1-p)]$$



当 $p = 0.5$ 时，熵最大，不确定性最大；当 $p = 0$ 或者 $p = 1$ 时，不确定性为0

- 熵与我们的直觉是一致的，即：

分布越平坦，不确定性就越大

条件熵

- 条件熵 $H(Z|Y)$: 在已知随机变量 Y 的值后, 随机变量 Z 的熵

$$H(Z|Y = y) = - \sum_{z \in \mathcal{C}} p(z|y) \log p(z|y)$$

$$H(Z|Y) = \sum_{y \in \mathcal{T}} P(Y = y) H(Z|Y = y)$$

Y	Z
t	t
t	t
t	t
t	t
f	t
f	f

➤ 示例

$$p(Z = t|Y = t) = 1 \text{ 且 } p(Z = f|Y = t) = 0 \quad \Rightarrow \quad H(Z|Y = t) = 0$$

$$p(Z = t|Y = f) = 0.5 \text{ 且 } p(Z = f|Y = f) = 0.5 \quad \Rightarrow \quad H(Z|Y = f) = 1$$

$$p(Y = t) = 4/6 \text{ 且 } p(Y = f) = 2/6$$

$$\Rightarrow \quad H(Z|Y) = \frac{4}{6} \times 0 + \frac{2}{6} \times 1 = \frac{2}{6}$$

- 条件熵 $H(Z|Y)$ 与熵 $H(Z)$ 是不同的

对于给定的例子，可以看出：

$$\begin{aligned} H(Z) &= -p(z = t) \log p(z = t) - p(z = f) \log p(z = f) \\ &= -\frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6} \\ &= -\frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6} \\ &\approx 0.65 \end{aligned}$$

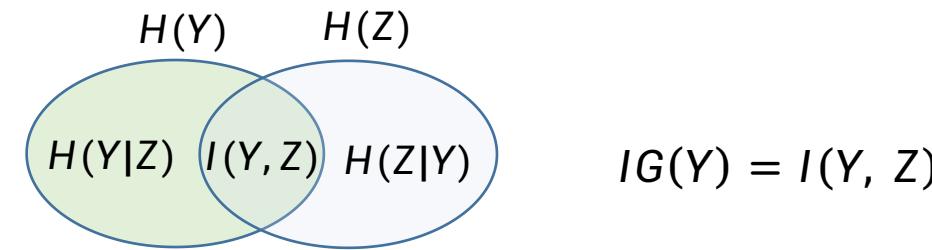
显然，它大于条件熵 $H(Z|Y) \approx 0.33$

实际上，不等式 $H(Z) \geq H(Z|Y)$ 总是成立的

信息增益

- 随机变量 Y 的信息增益是在已知其值之后，随机变量 Z 的 **熵减少的量**

$$IG(Y) = H(Z) - H(Z|Y)$$



- 对于上面给出的例子，随机变量 Y 的信息增益为：

$$IG(Y) = 0.65 - 0.33 = 0.32$$

- Y 的信息增益意味着，如果已知其值，则 **平均可以减少的不确定性**

课程大纲

- 介绍
- 选择扩展属性的准则
- 决策树学习

选择根节点

- 结果变量“Liked?”的熵

$$P(Like = yes) = 2/3 \text{ 且 } P(Like = no) = 1/3 \Rightarrow H(Like) = 0.91$$

- 在给定属性 (Type, Length, Director 和 Actors) 后，结果的条件熵

$$H(Like|Type) = 0.61$$

$$H(Like|Length) = 0.61$$

$$H(Like|Director) = 0.36$$

$$H(Like|Actor) = 0.85$$

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 信息增益

$$IG(\text{Type}) = H(\text{Like}) - H(\text{Like}|\text{Type}) = 0.3$$

$$IG(\text{Length}) = H(\text{Like}) - H(\text{Like}|\text{Length}) = 0.3$$

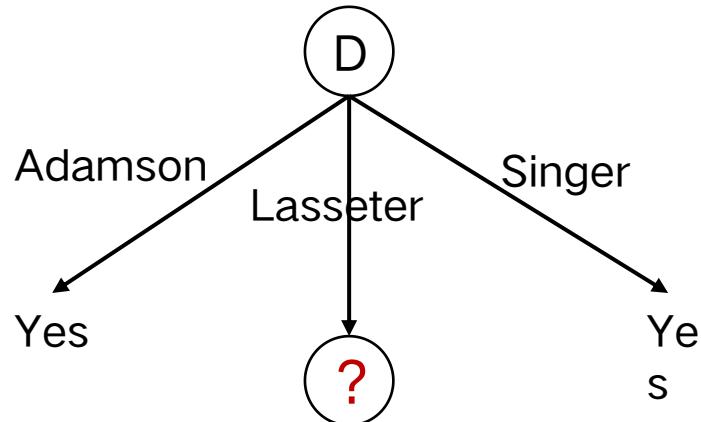
$$IG(\text{Actor}) = H(\text{Like}) - H(\text{Like}|\text{Actor}) = 0.06$$

$$IG(\text{Director}) = H(\text{Like}) - H(\text{Like}|\text{Director}) = 0.55$$

} \Rightarrow Director 应该作为根节点

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 构建树



	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 由于来自“Adamson”和“Singer”分支的所有结果都是“Yes”，因此我们无需进一步扩展这两个分支
- 问题是如何为“Lasseter”分支选择属性

继续扩展

- 选择导演“Lasseter”后，剩余的数据是

	Type	Length	Director	Famous actors	Liked?
2	Animated	Short	Lasseter	No	No
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
9	Drama	Medium	Lasseter	No	Yes

- 重新计算熵和条件熵得到：

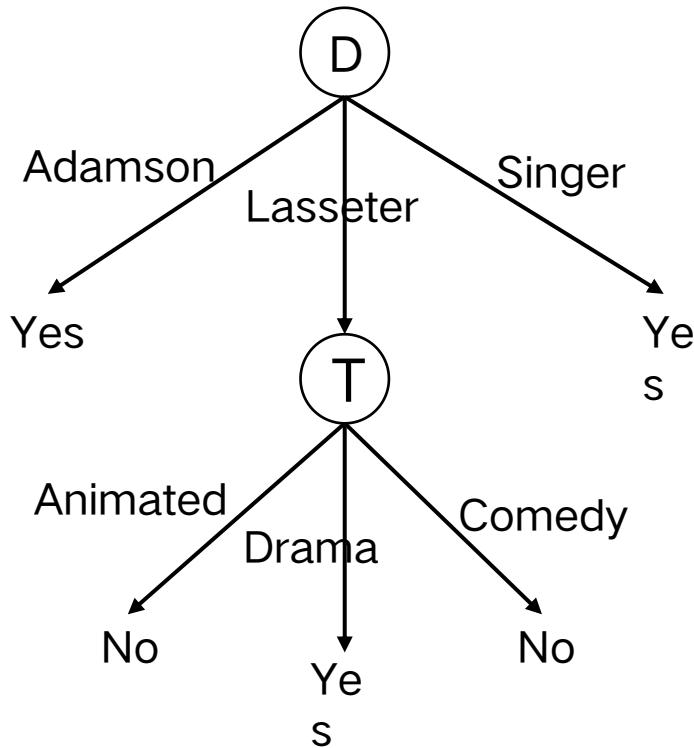
$$H(Like) = 0.81 \quad H(Like|Type) = 0 \quad H(Like|Length) = 0 \quad H(Like|Actor) = 0.5$$

- 因此，信息增益为

$$IG(Type) = 0.81 \quad IG(Length) = 0.81 \quad IG(Actor) = 0.31$$

因此，我们应该选择属性“Type”或“Length”来扩展

- 构建树



这是最终的决策树！！

	Type	Length	Director	Famous actors	Liked?
2	Animated	Short	Lasseter	No	No
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
9	Drama	Medium	Lasseter	No	Yes

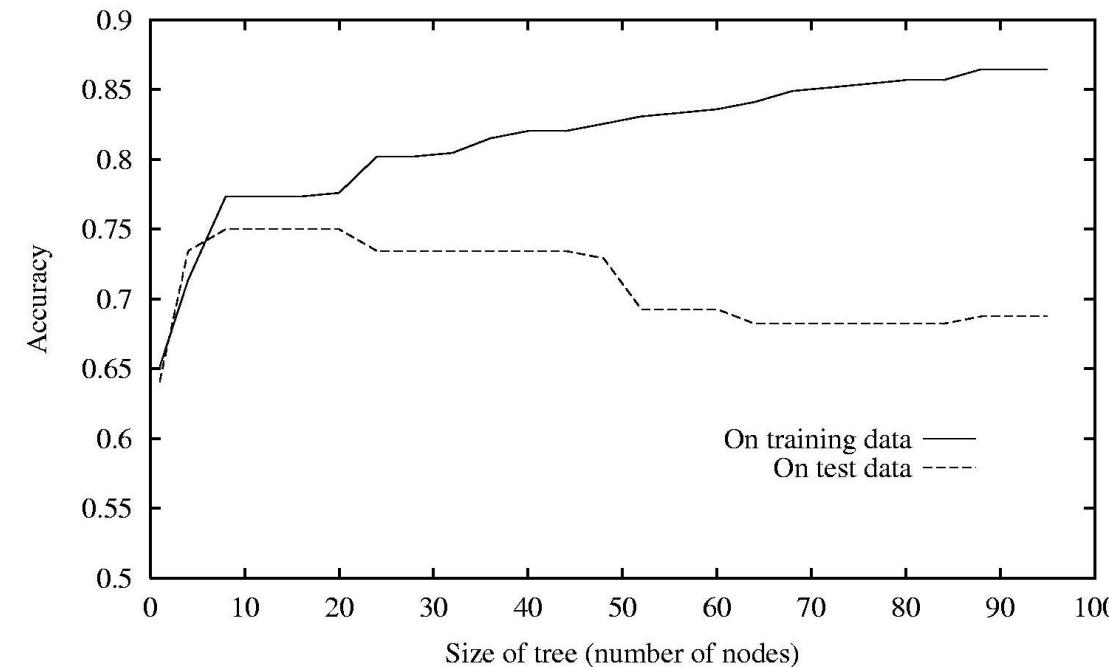
- 我们停止进一步扩展这棵树，因为在训练数据中，对于每条从根节点到叶节点的路径，都只有一种可能的结果

停止扩展的准则

- 树不能无限扩展，应该在某个点停止。以下是阐述的一些停止准则：
 - 所有剩余的实例都具有相同的标签
 - 我们用完了所有的属性
 - 树的深度达到最大限制
 - 信息增益小于某个阈值
 -

过拟合问题

- 如果树太大或过于复杂，它在训练数据上会表现得很好，但在测试数据上表现可能很差

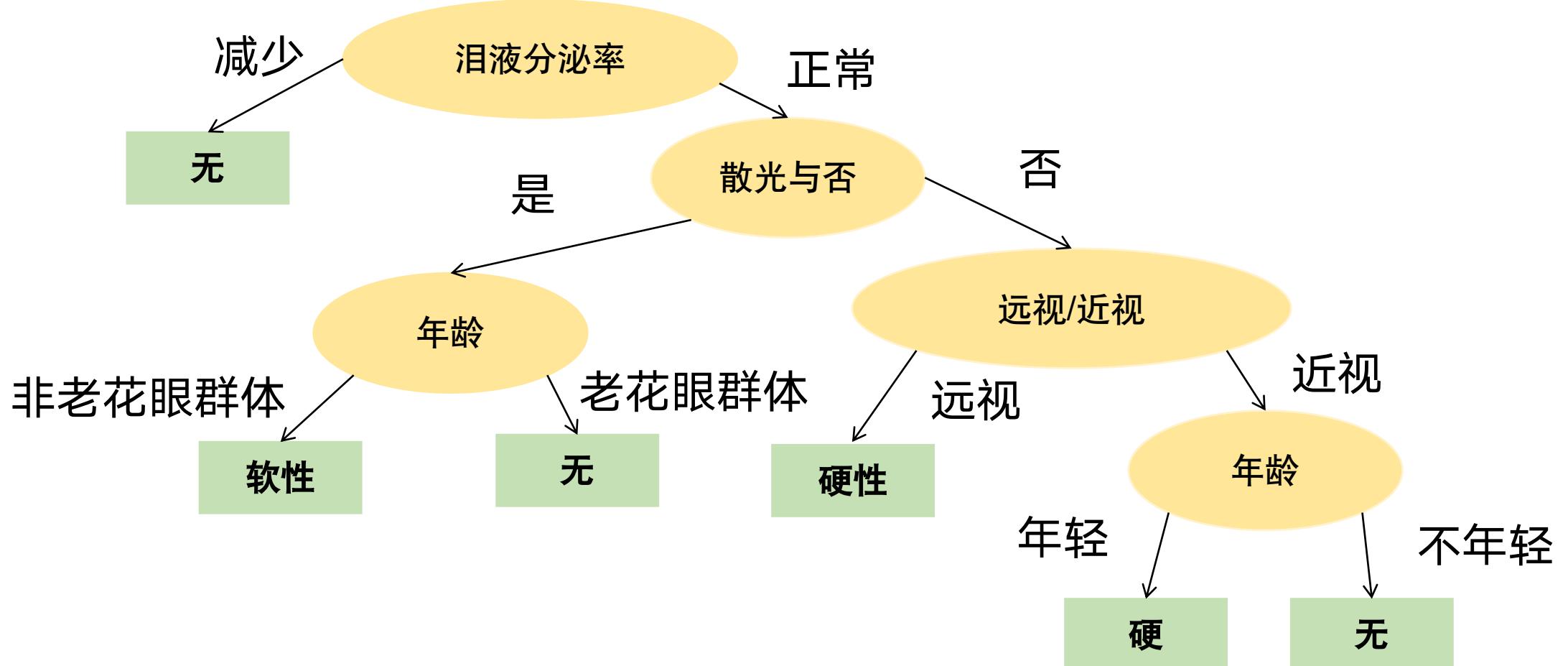


树剪枝

- 为了控制决策树的复杂度，我们可以：
 - 在学习树时进行剪枝
 - 在学习树后进行剪枝
- 基于一个验证数据集，我们可以：
 - 剪掉那些不损害**验证集**准确率的节点
 - 贪婪地移除对**验证准确率**提升最小的节点
 - 当**验证集**准确率开始下降时停止

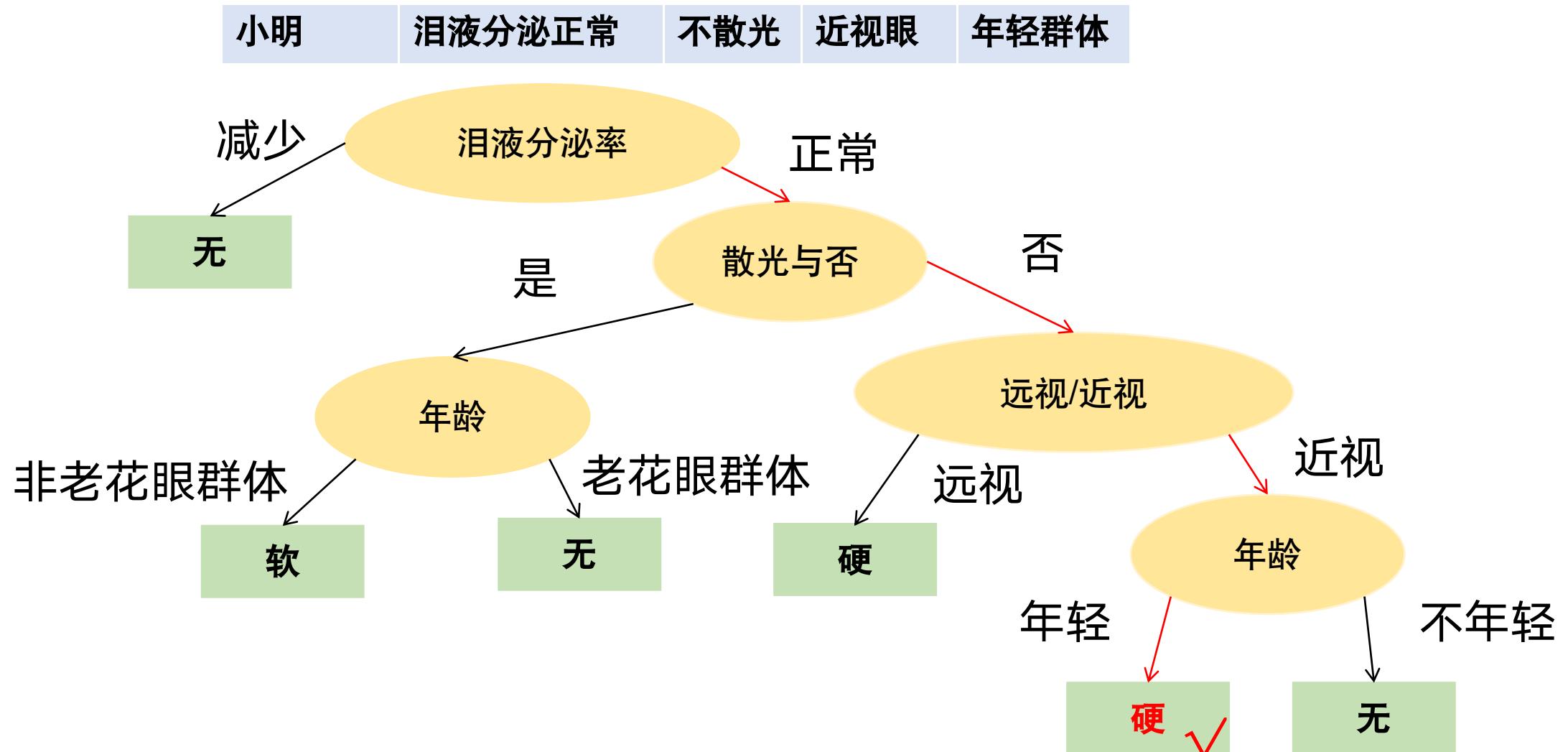
树剪枝

- 以下是一个例子，我们构建了一颗确定一个人应该戴何种隐形眼镜的决策树：



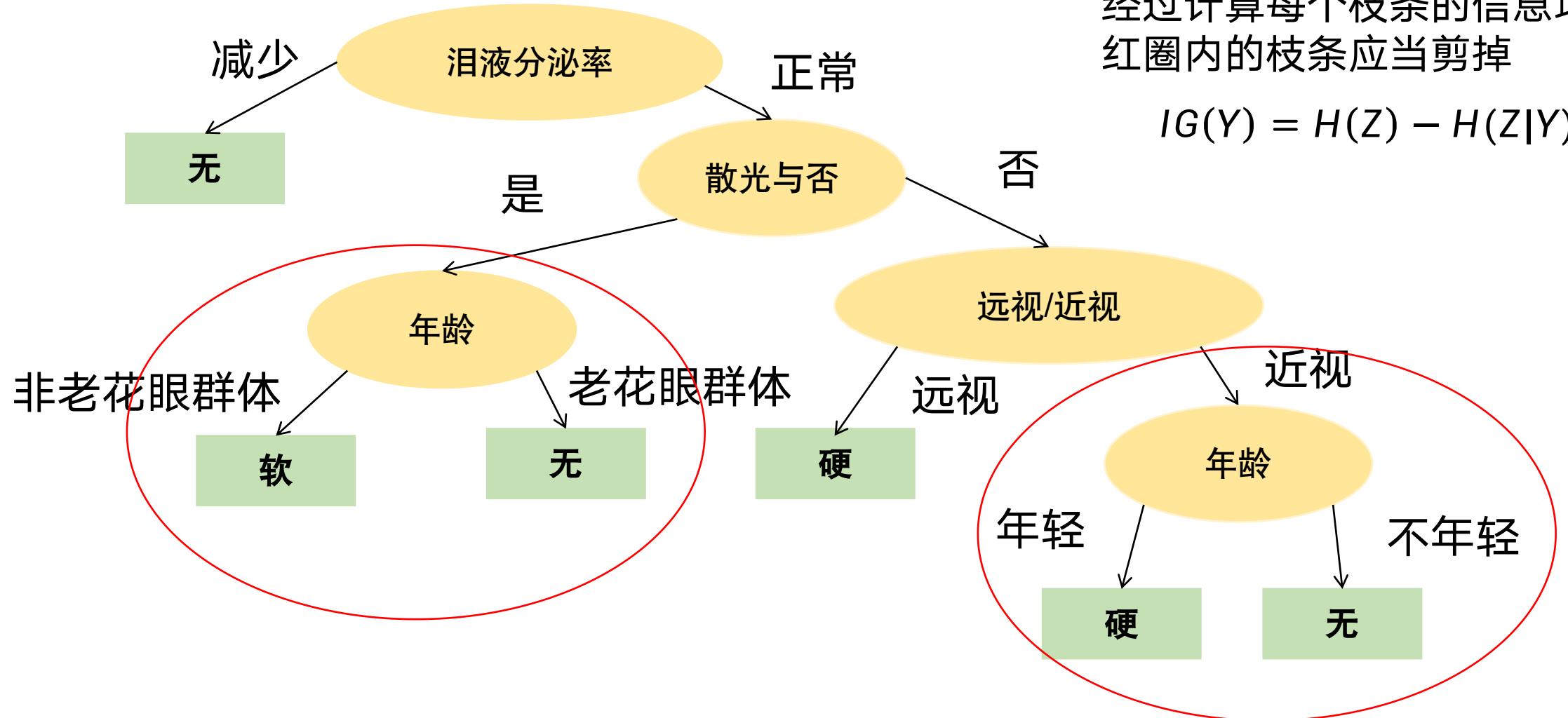
树剪枝

- 接下来，我们用这颗树对一个实例进行评估：



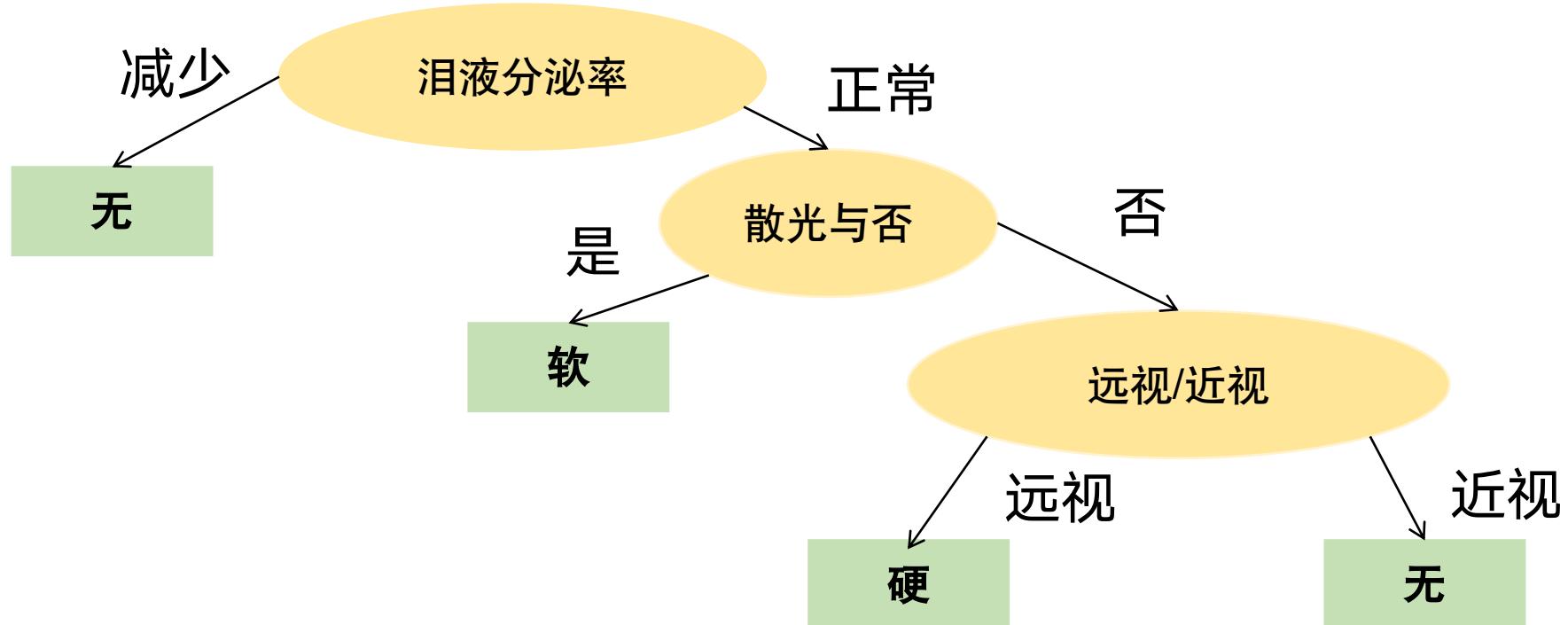
树剪枝

- 这棵树被分得太细，有过拟合风险，我们使用**基于信息增益的剪枝**：



树剪枝

- 最后的决策树：



小明 | 泪液分泌正常 | 不散光 | 近视眼 | 年轻群体

课堂小问：剪枝后小明还是戴硬性眼镜吗？剪枝会带来什么样的结果？

决策树小结

- 什么是决策树
- 决策树构建相关概念：熵、条件熵、互信息、信息增益
- 决策树的过拟合、剪枝问题

集成方法概述

- 很难学习到一个总是能正确分类实例的**强分类器**

- 但很容易学习到很多**弱分类器**

一个弱分类器可能在整个数据集上表现不佳，但可能在部分样本上表现良好，例如，有些可能善于识别“猫”，有些可能善于识别“狗”

- 如果弱分类器在不同的样本子集上表现良好，就有可能通过以适当的方式组合这些**弱分类器来得到一个强分类器**

- 两个问题

- 1) 如何生成这些弱分类器？
- 2) 如何组合这些弱分类器？

两种组合方法

1) 无加权平均

- 多数投票法

2) 加权平均

- 给予更好的分类器更大的权重

对于一个二分类问题，例如分类标签为 $\{-1, 1\}$ 的问题，考虑两个基本分类器：

$$\text{两个基本分类器} \quad \hat{y}_1 \quad \hat{y}_2 = \text{sign}(f_2(\mathbf{x}))$$

$$= \text{sign}(f_1(\mathbf{x}))$$

$$\text{最终分类器} \quad \hat{y}_e = \text{sign}(a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}))$$

备注：这些弱分类器可以是任何类型，例如决策树、支持向量机 (SVM)、神经网络、逻辑回归等

集成方法课程大纲

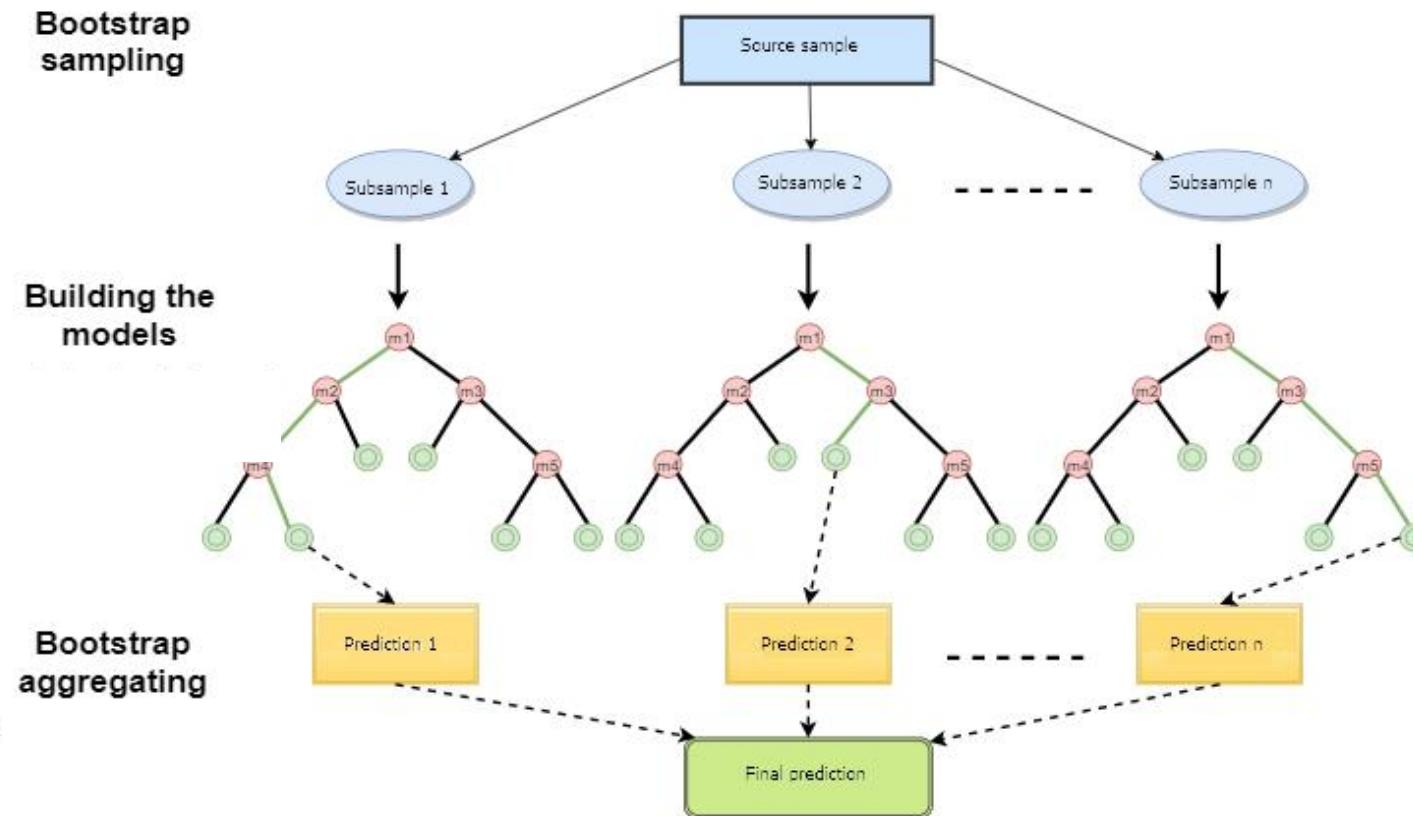
- 集成方法

- Bagging (多数投票法)
- Boosting (加权平均法)

得到弱分类器

- 如何得到在不同样本子集上表现良好的分类器是未知的
- 我们可以尝试获得预测误差 **相互独立的** 分类器。例如：
 - 1) 通过自举法 (bootstrapping) 创建训练数据集的子集
 - 从 N 个样本的训练数据集中 **有放回地** 随机抽取 N' 个样本
 - 重复上述过程 K 次，生成子集 S_1, S_2, \dots, S_K
 - 2) 在每个子集 S_k 上训练一个决策树

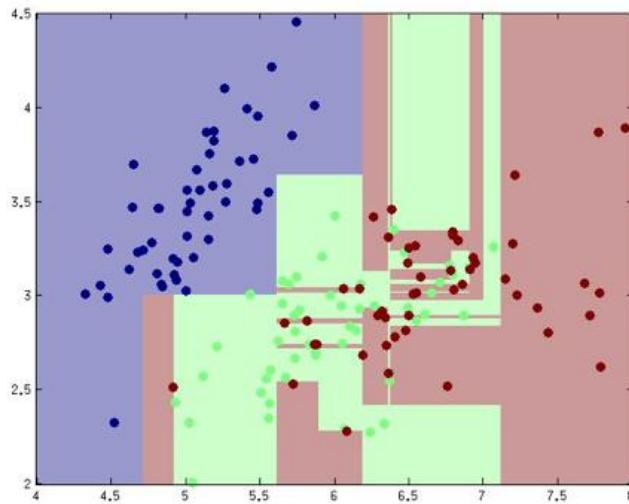
3) 通过多数投票法将 K 个决策树组合成一个



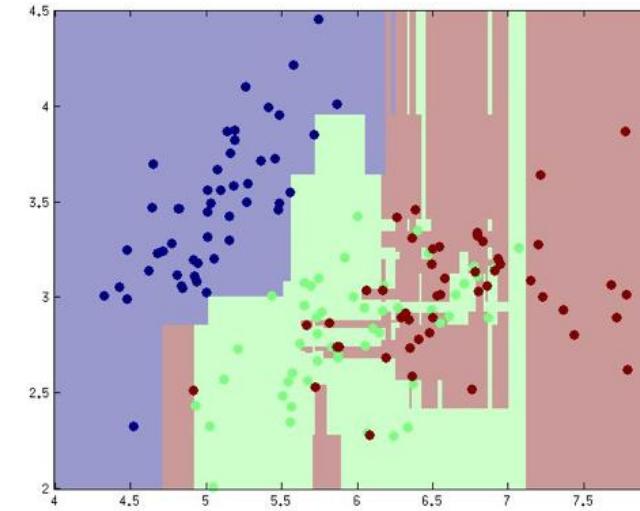
在测试时，将测试数据通过所有 K 个分类器，并使用多数投票的结果作为最终预测

示例：Bagged 决策树

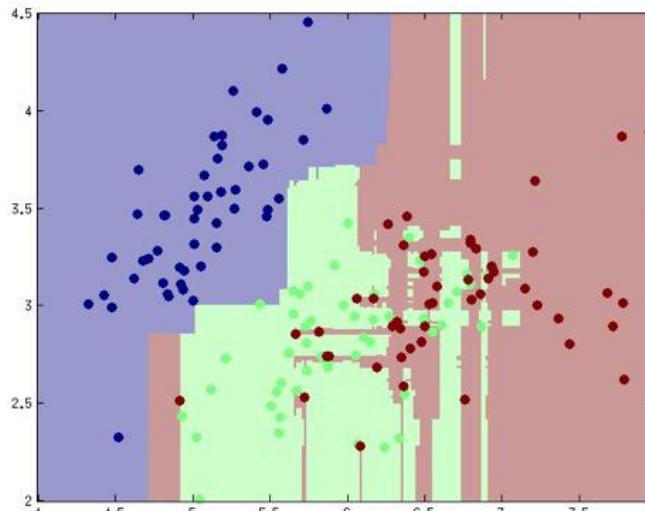
单棵树



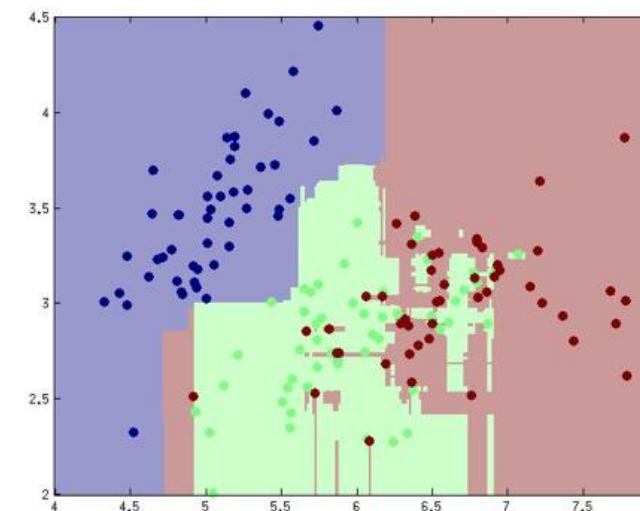
5棵树平均



25棵树平均



100棵树平均



随机森林

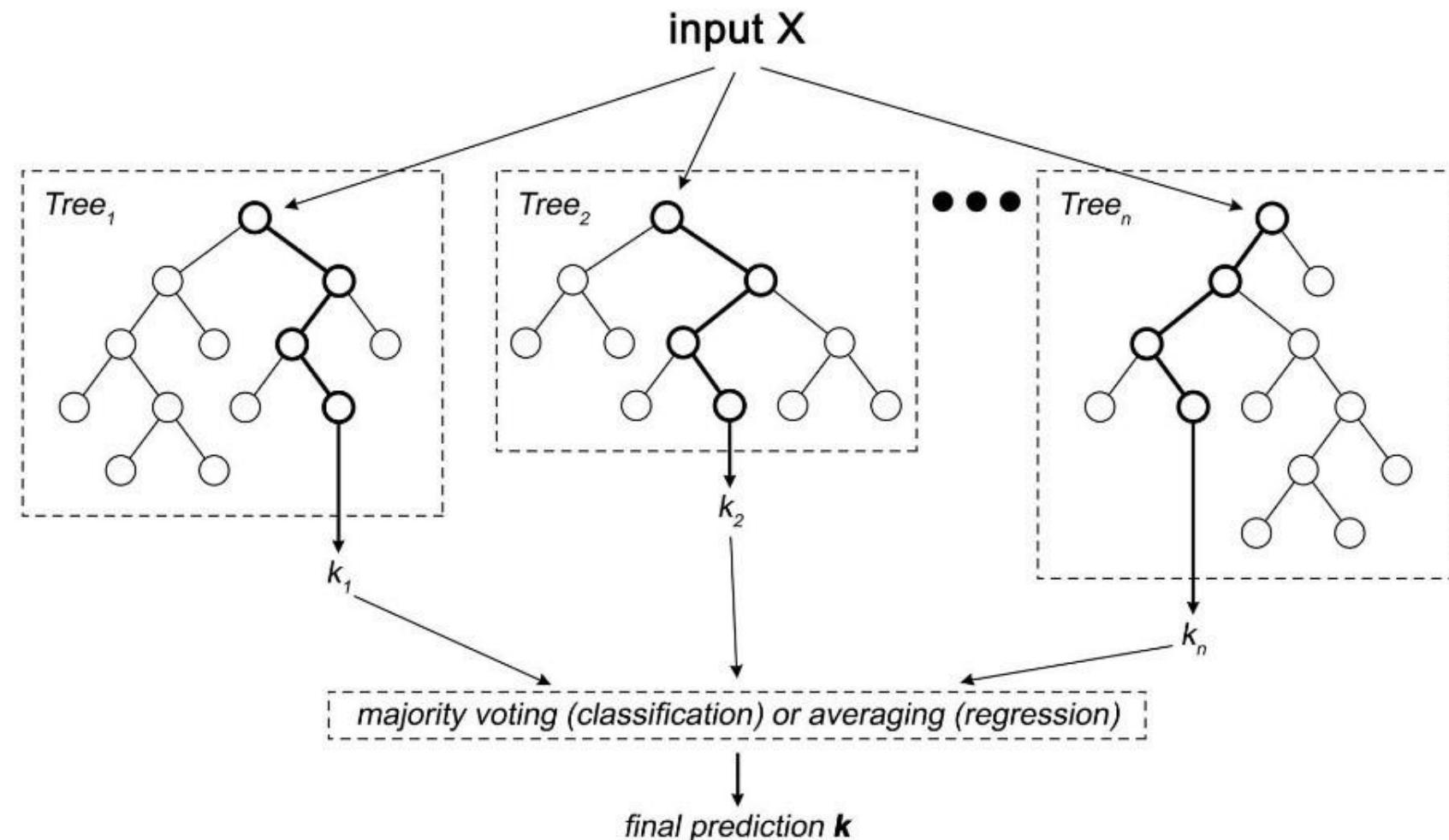
- 如果前面得到的分类器的预测误差 **仍然高度相关**, 会发生什么?

⇒ 仅仅通过多数投票法组合它们可能对最终性能没有太大帮助

- 补救措施:** 在决策树的学习过程中引入额外的随机性

仅使用一个随机选择的属性子集来构建决策树

- 随机森林图解



课程大纲

- 集成方法

- Bagging (多数投票法)
- Boosting (加权平均法)

提升法 (Boosting) 概述

- 弱分类器来源

重复以下步骤数次：

- 1) 识别被错误分类的样本
- 2) 通过**对错误分类的样本赋予更大的权重** 来重新训练分类器

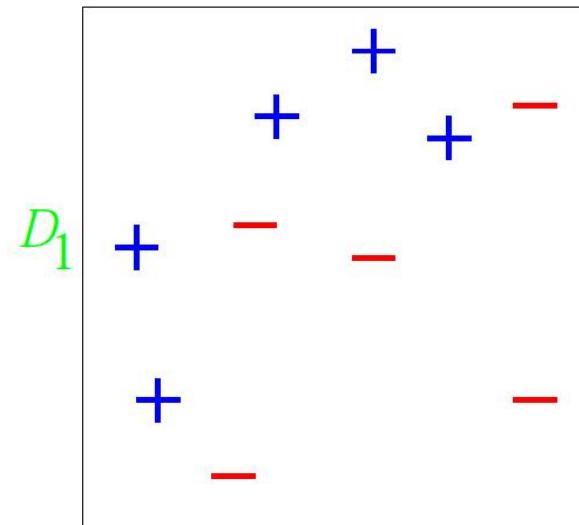
- 组合

通过**加权平均** 组合每个分类器的预测结果

如何**对样本和预测结果进行加权** 是关键所在

Adaboost

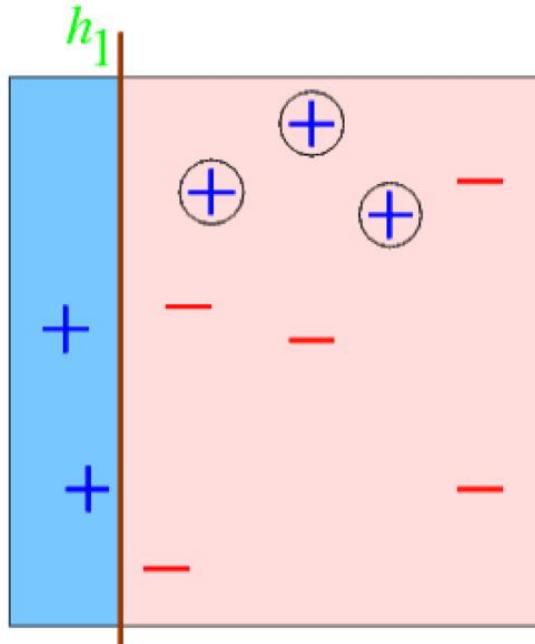
- 考虑一个包含10个训练样本的二分类问题



- 为简单起见，仅考虑决策边界与坐标轴平行的弱分类器，即：

$$\hat{y} = \text{sign}(x_1 + b) \text{ or } \hat{y} = \text{sign}(x_2 + b)$$

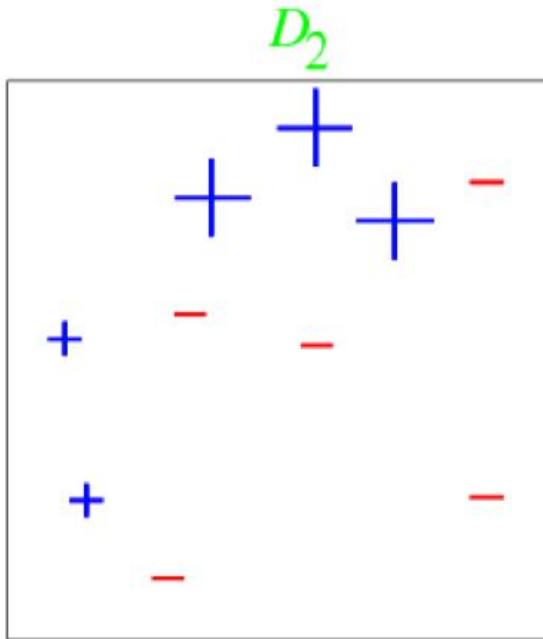
- 第一次迭代



- 第一个分类器 h_1 的错误率: $\epsilon_1 = 0.3$
- 分类器 h_1 的权重: $a_1 = \frac{1}{2} \ln \left(\frac{1-\epsilon_1}{\epsilon_1} \right) = 0.42$

$$\frac{1 - \epsilon_1}{\epsilon_1} = \frac{\text{正确率}}{\text{错误率}}$$

⇒ 权重与分类器的性能呈正比



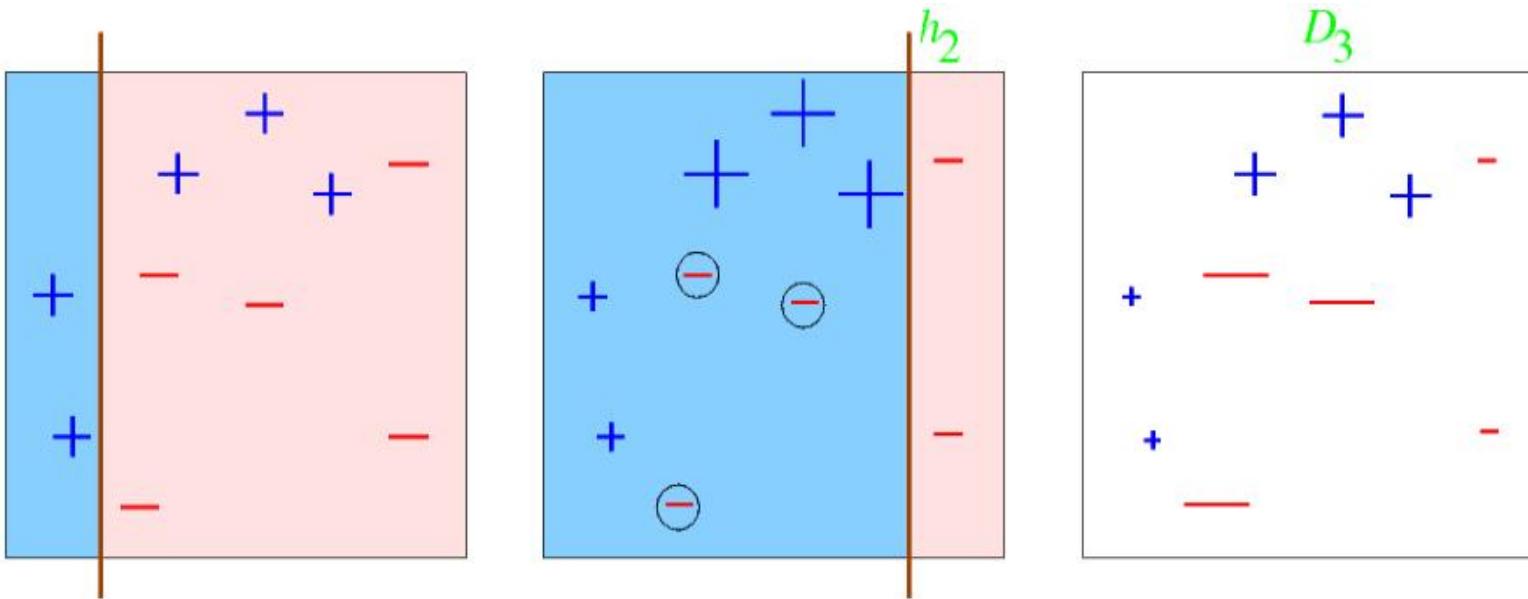
➤ 被错误分类 样本的权重被 e^{a_1} 放大

$$e^{a_1} = \sqrt{\frac{1 - \epsilon_1}{\epsilon_1}} = \sqrt{\frac{\text{正确率}}{\text{错误率}}}$$

➤ 被正确分类 样本的权重被 e^{-a_1} 削弱

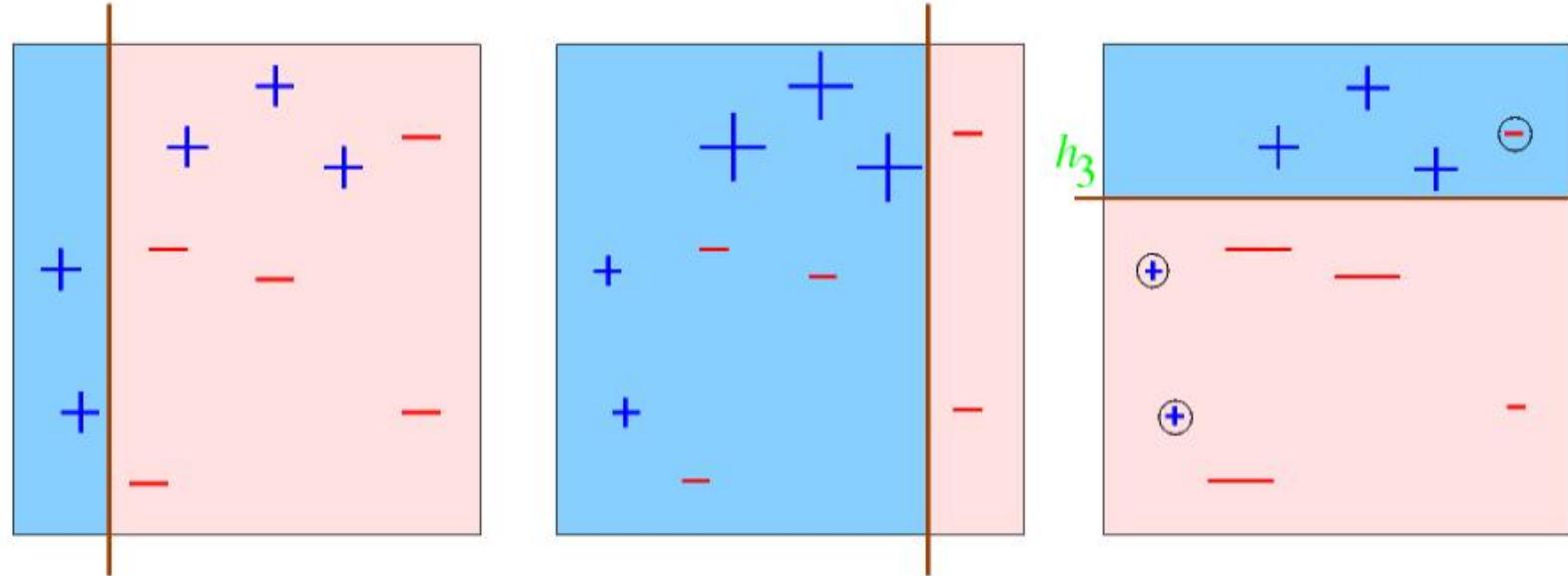
⇒ 分类器权重: $\ln \left(\sqrt{\frac{\text{正确率}}{\text{错误率}}} \right)$, 样本权重: $\sqrt{\frac{\text{正确率}}{\text{错误率}}}$

- 第二次迭代



- 第二个分类器 h_2 的错误率: $\epsilon_2 = 0.21$
- 分类器 h_2 的权重: $a_2 = \frac{1}{2} \ln \left(\frac{1-\epsilon_2}{\epsilon_2} \right) = 0.65$
- 被错误分类样本的权重被 $e^{a_2} = \sqrt{\frac{1-\epsilon_2}{\epsilon_2}}$ 放大
- 被正确分类样本的权重被 $e^{-a_2} = \sqrt{\frac{\epsilon_2}{1-\epsilon_2}}$ 削弱

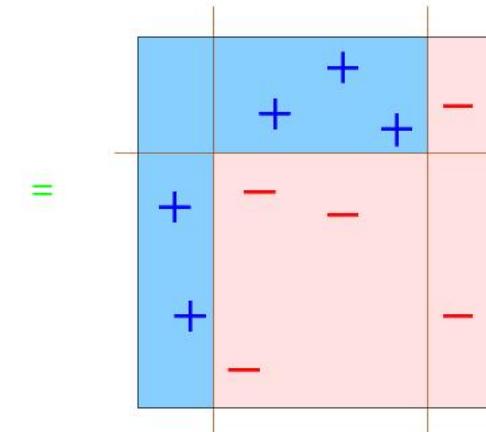
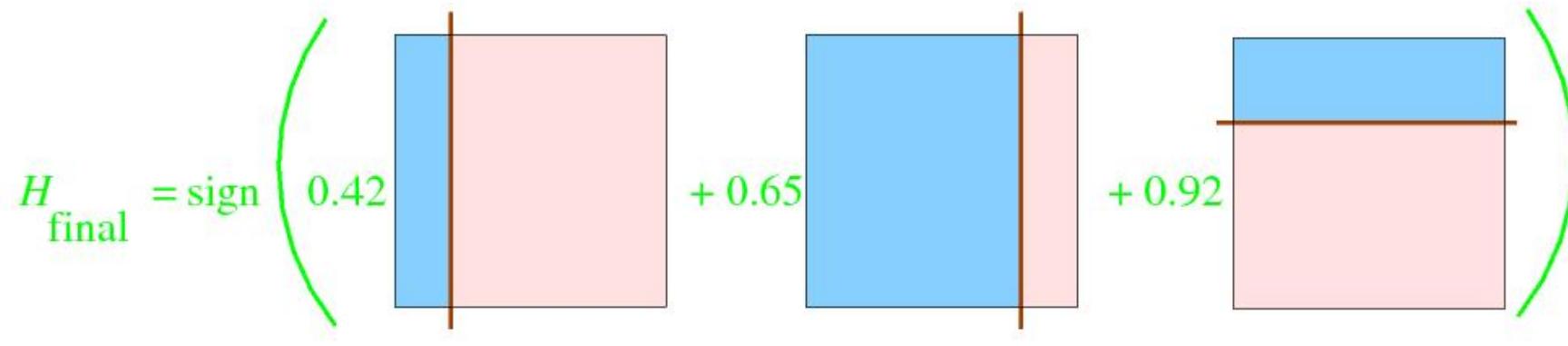
- 第三次迭代



- 第二个分类器 h_3 的错误率: $\epsilon_3 = 0.14$
- 分类器 h_3 的权重: $a_3 = \frac{1}{2} \ln \left(\frac{1-\epsilon_3}{\epsilon_3} \right) = 0.92$
- 停止迭代

- 最终分类器

最终分类器通过线性组合来组合这三个分类器



AdaBoost 算法

- 1) 初始化样本权重, $\omega_0^{(n)} = \frac{1}{N}$, 其中 $n = 1, \dots, N$
- 2) 对于第 k 次迭代, 使用按 $w_{k-1}^{(n)}$ 加权的训练样本训练一个分类器 $h_k(x)$
- 3) 评估加权分类错误:

$$\epsilon_k = \frac{\sum_{n=1}^N \omega_{k-1}^{(n)} I(y_i \neq h_k(x_n))}{\sum_{n=1}^N \omega_{k-1}^{(n)}}$$

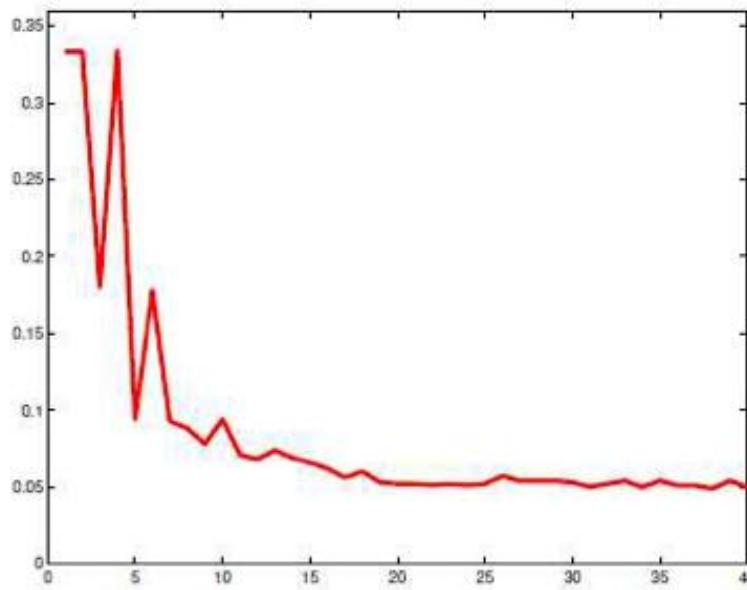
- 4) 确定第 k 个分类器的投票权重:

$$a_k = \frac{1}{2} \ln \left(\frac{1-\epsilon_k}{\epsilon_k} \right)$$

- 5) 更新样本权重:

$$\omega_k^{(n)} = \omega_{k-1}^{(n)} \exp \{ -y_i h_k(x_i) a_k \}$$

- 典型的错误率曲线，以弱分类器数量为函数



- 典型的弱分类器：
 - 决策树
 - 逻辑回归
 - 神经网络

AdaBoost 背后的目标函数

- 定义以下**指数损失**:

$$l = \sum_{n=1}^N \exp \{-y^{(n)} h_{\text{combine}}(\mathbf{x}^{(n)})\}$$

其中 $\mathbf{x}^{(n)}$ 是输入, $y^{(n)} \in \{-1, 1\}$ 是标签, $h_{\text{combine}}(\cdot)$ 是组合分类器:

$$h_{\text{combine}}(\mathbf{x}) = a_1 h_1(\mathbf{x}) + \cdots + a_K h_K(\mathbf{x})$$

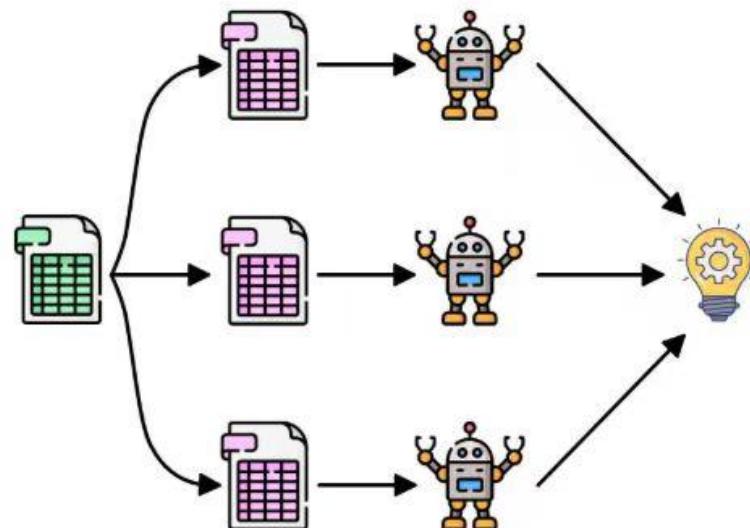
$h_k(\mathbf{x})$ 代表第 k 个组成分类器, 例如 $h_k(\mathbf{x}) = \text{sign}(\mathbf{w}_k^T \mathbf{x} + b_k)$

- 可以证明, AdaBoost 算法**等价于**以**序列式**的方式**最小化指数损失**

Bagging和Boosting的本质区别

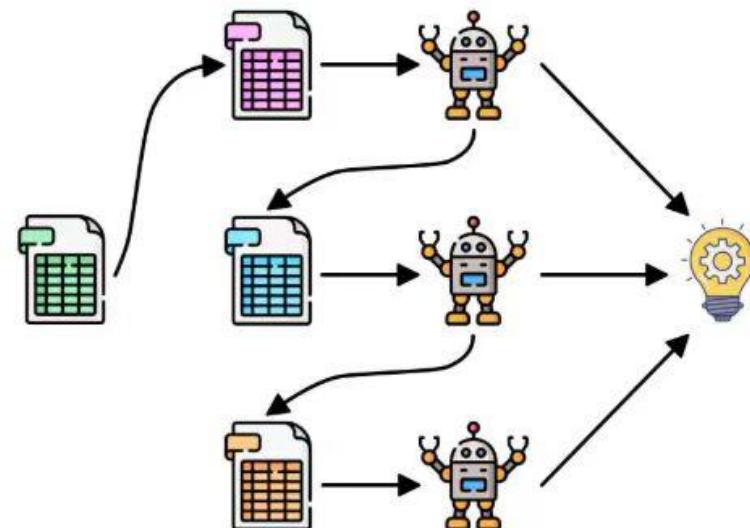
- 本质区别是并行和串行：

Bagging



Parallel

Boosting



Sequential

集成方法小结

- 什么是集成学习？
- Bagging（多数投票法）
- Boosting（加权平均法）



聚类 K-Means、线性降维 PCA

主讲人：林惊

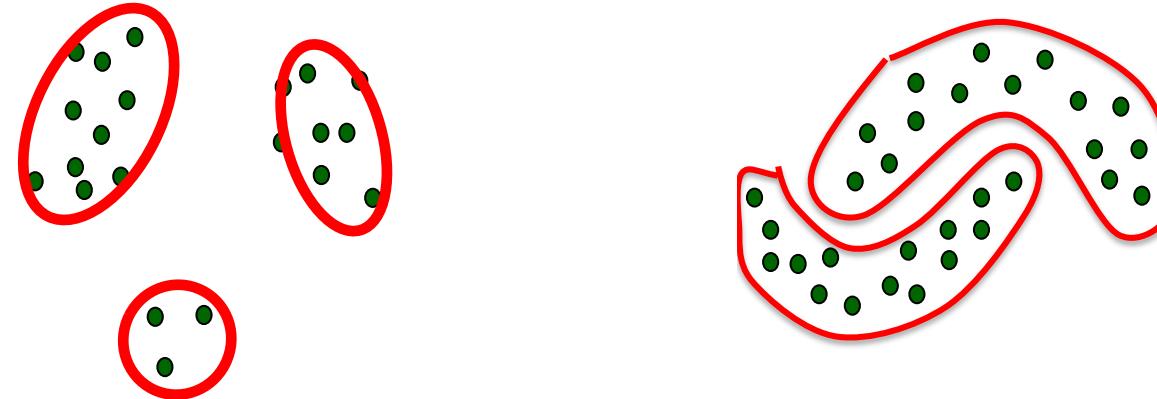


聚类大纲

- 聚类简介
- K -Means

什么是聚类?

- 给定一组数据实例 $\{\mathbf{x}^{(i)}\}_{i=1}^N$, 聚类就是如何将它们分成不同的簇



- 目标:
 - 簇内实例具有高相似度
 - 簇间实例具有低相似度

相似性准则很重要

- 不同的相似性准则可能导致不同的结果



相似还是不相似？

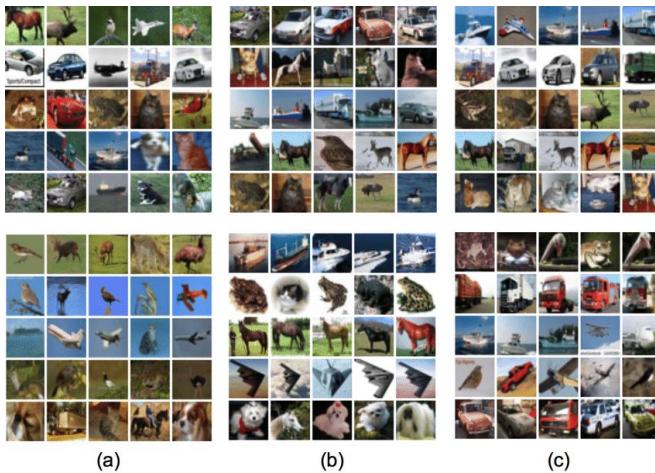


准则1：身份

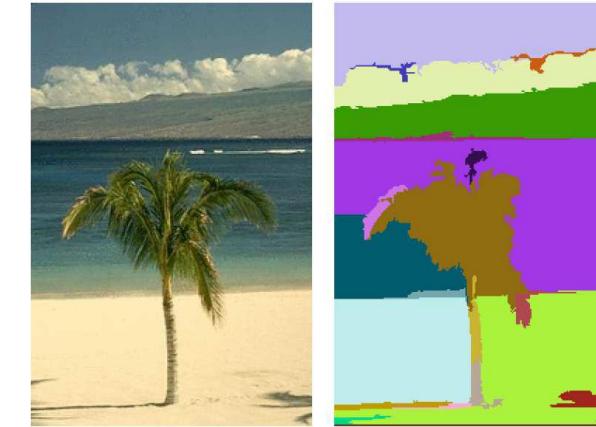
准则2：眼镜

实际应用

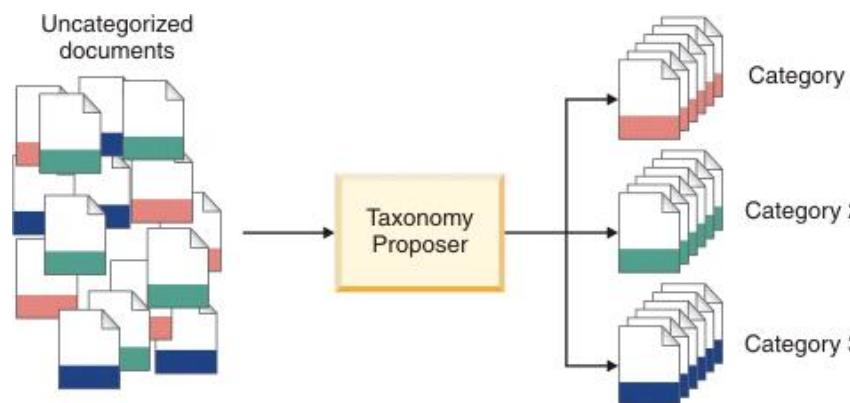
- 图像分组



- 图像分割



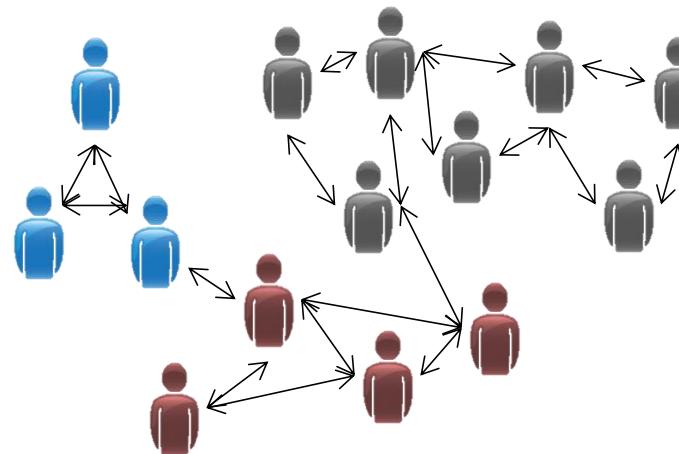
- 自动将语义相似的文档分组在一起



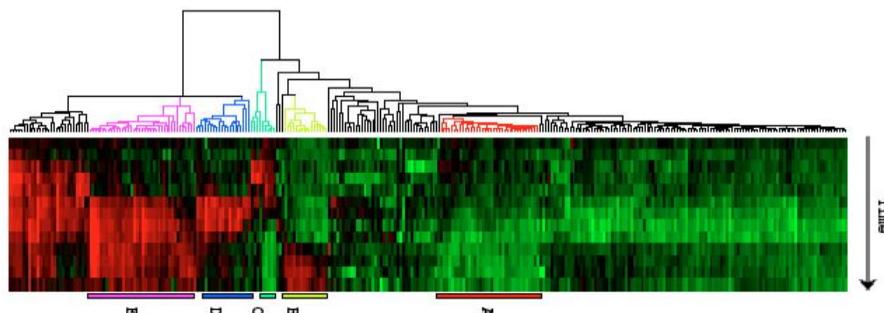
• 网络搜索结果聚类



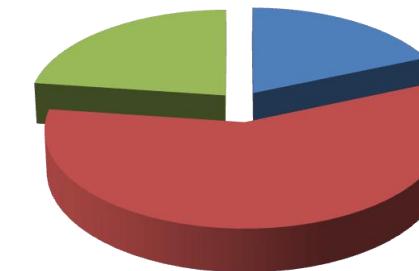
• 社交网络分析



• 基因表达数据聚类



• 市场细分

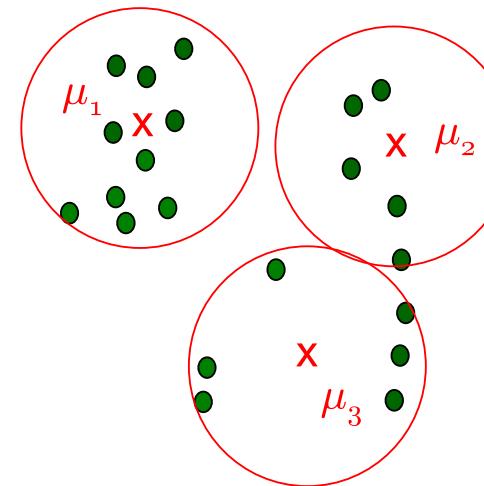


聚类大纲

- 聚类简介
- K-Means

K-均值算法

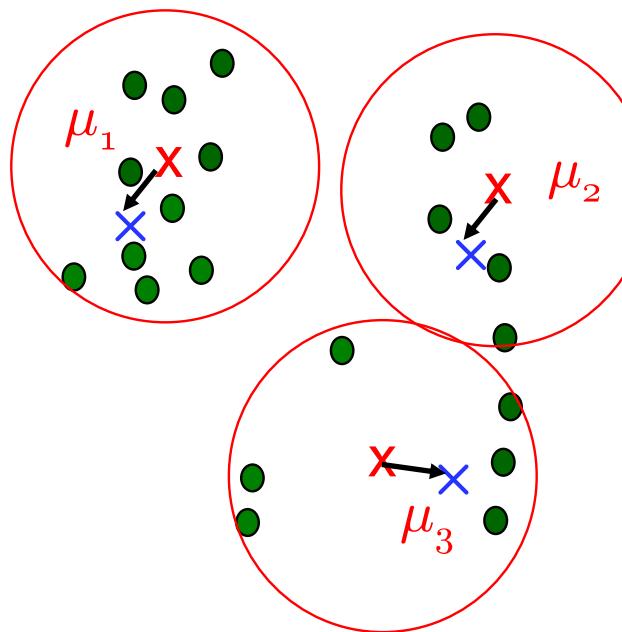
- 指定 K 个中心 μ_k , 其中 $k = 1, \dots, K$, 然后评估每个数据点 $x^{(n)}$ 与所有中心 μ_k 之间的距离



- 数据点 $x^{(n)}$ 被分配给距离其最近的簇 k

$$r_{nk} = \begin{cases} 1, & \text{if } k = \arg \min_j \|x^{(n)} - \mu_j\|^2 \\ 0, & \text{otherwise} \end{cases}$$

- 使用簇内样本的均值来更新中心



两个问题：

- 1) 这个算法到底在做什么？
- 2) 这个算法能保证收敛吗？

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$$

- 重复上述分配和中心更新步骤

收敛性保证

- 定义一个目标函数，它是所有数据实例与其对应中心之间距离的总和：

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x^{(n)} - \mu_k\|^2$$

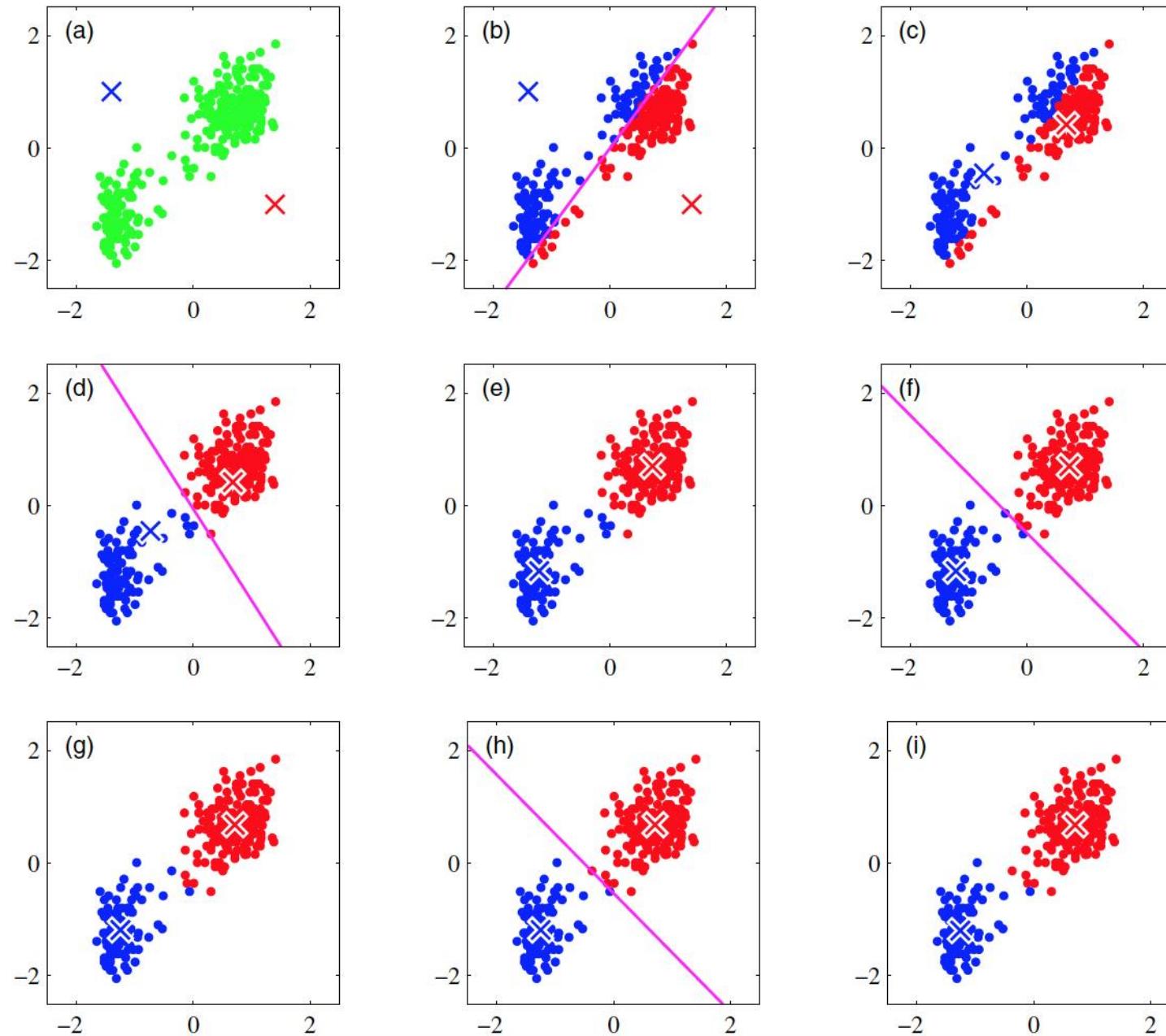
- K -均值可以通过以交替方式更新 r_n 和 μ_k 的以下优化问题得到：

$$\min_{r_n, \mu_k} J$$

$$\text{s. t. : } r_n \in \text{onehot vector} \quad \forall n & k$$

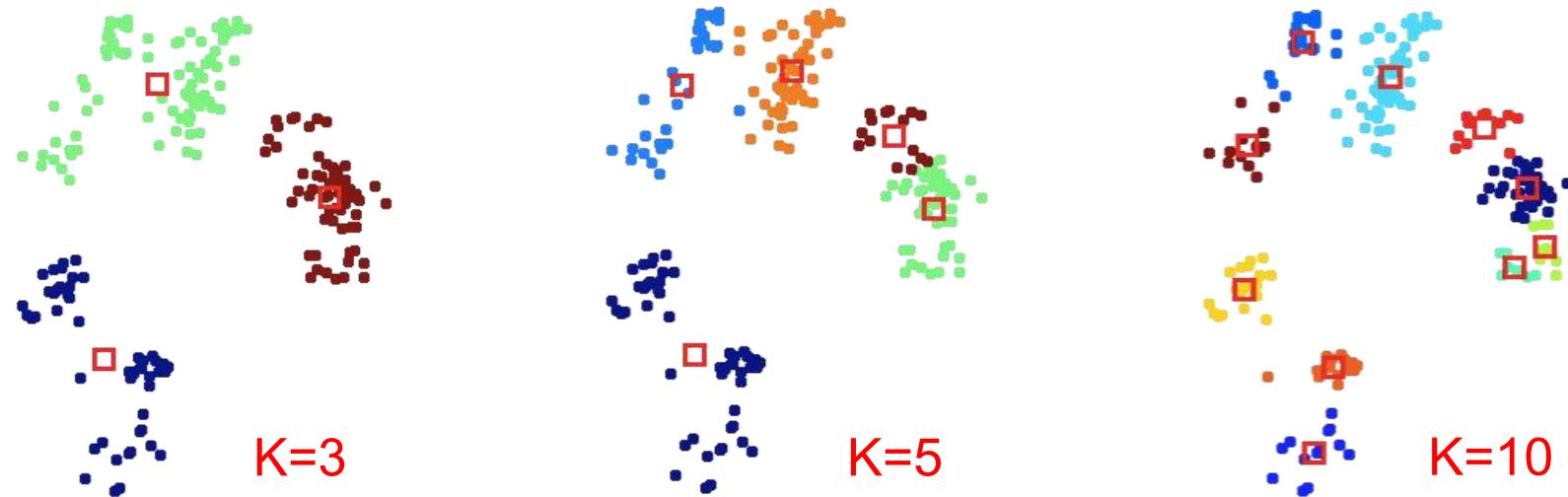
其中， $r_n \triangleq [r_{n1}, r_{n2}, \dots, r_{nK}]$ 需为独热向量

- 总距离 J 单调递减，因此可以保证 K -均值算法收敛

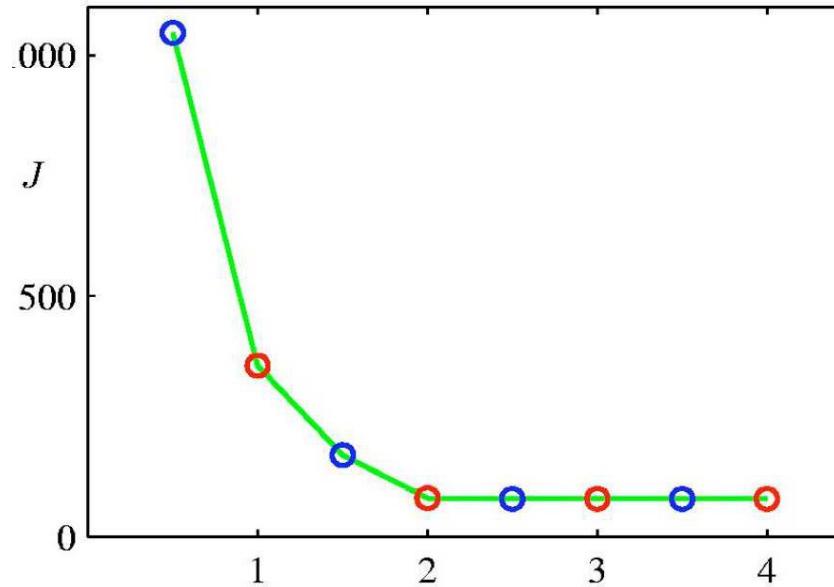


问题：簇的数量

- 如何设置 K 的值对于最终的聚类结果至关重要



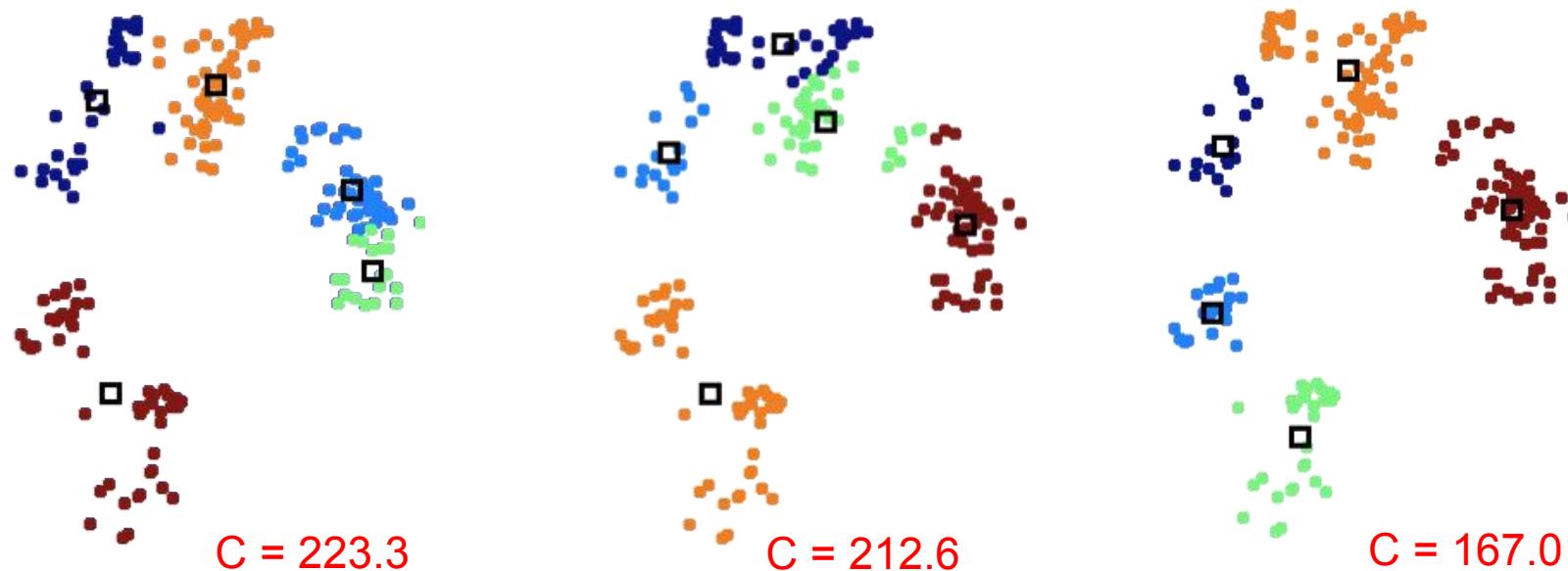
- 随着簇数量 K 的增加，距离 J 被确定为递减。因此， K 不能通过最小化 J 来确定



- 1) 一种可能的方法是选择拐点（这里 $K=2$ ）
- 2) 另一种可能的方法是根据下游应用的性能来确定最佳的 K 值

问题：初始化

- K -均值的性能也高度依赖于初始中心点



1) 随机方法

- 随机选择数据实例作为初始化
- 问题：可能会选择相邻的实例

2) 基于距离的方法

- 从一个随机数据实例开始
- 选择距离现有中心最远的点
- 问题：可能会选择离群值

3) 随机 + 距离方法

- 从一个随机数据实例开始
- 从其余距离现有中心较远的实例中随机选择下一个中心

问题：硬分配

- 硬分配

一个数据实例要么属于某个簇，要么不属于，这种分配方式是确定性的，即 r_n 必须是一个独热向量

- 软 K -均值

软 K -均值不是确定性地将 $\mathbf{x}^{(n)}$ 分配给某个簇，而是以软性方式进行分配

β 控制了分布的尖锐程度

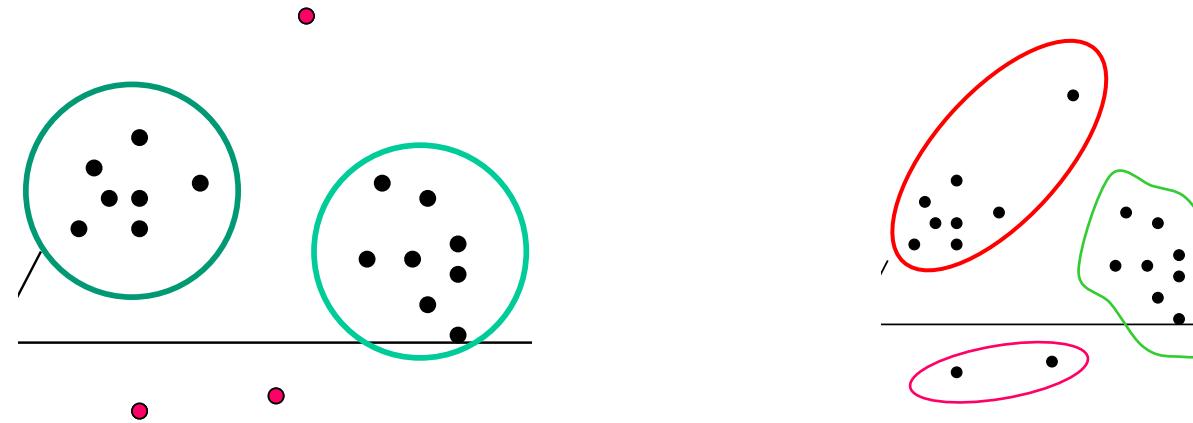
$$r_{nk} = \frac{e^{-\beta \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\|^2}}{\sum_{i=1}^K e^{-\beta \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_i\|^2}}$$

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{n=1}^N r_{nk} \mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$$

r_{nk} 可以被解释为数据点 $\mathbf{x}^{(n)}$ 属于簇 k 的概率

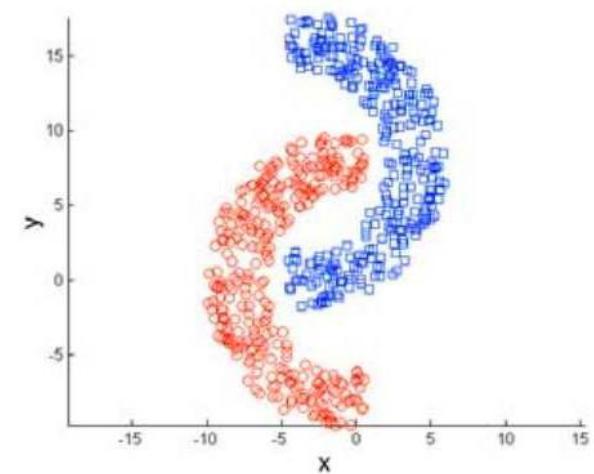
其他问题

- 对离群值敏感



- 圆形形状

欧氏距离意味着边界只能是球形的。当簇具有不规则形状时，性能会很差



课堂小结

- 聚类的目标——簇内高相似、簇间低相似
- K-Means的算法
 - 算法过程
 - 等价的优化目标函数
 - 存在的问题，如：簇数量的确定、初始化、硬分配等
- 软K-Means

线性降维PCA大纲

- 动机
- 视角 1: 最小化重构误差
- 视角 2: 最大化方差
- 视角 3: 奇异值分解 (SVD)
- PCA的其他应用

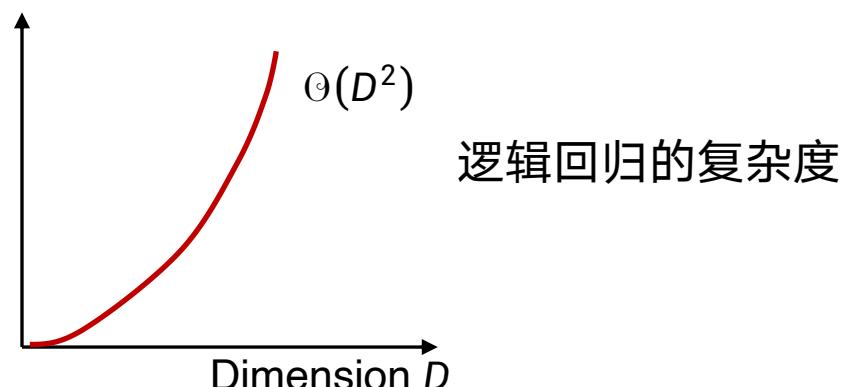
动机

- 许多类型数据的维度非常高，例如下面图像的维度高达

$$256 \times 256 = 65536$$

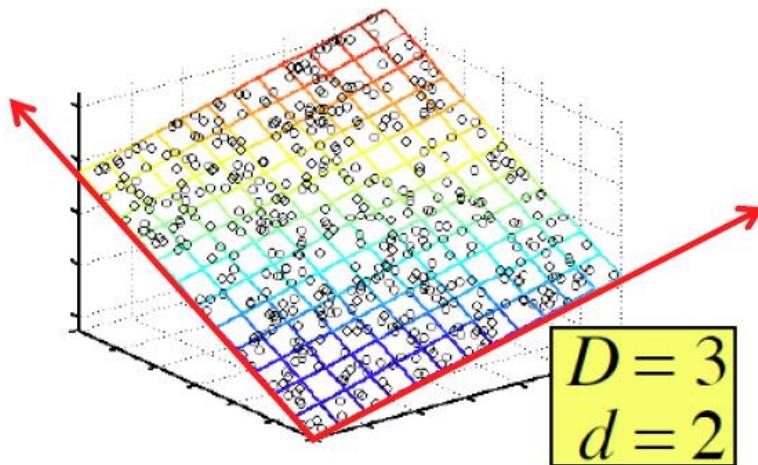


- 如果我们直接在原始数据上工作，后续任务（例如，分类）的复杂性可能会非常高。

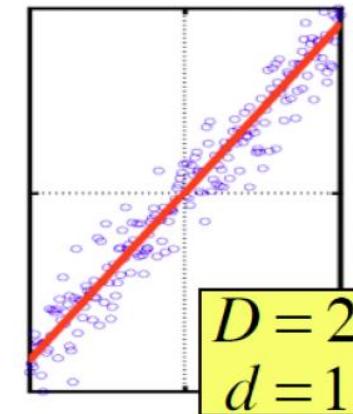


降低数据的维度**为什么可行**？

- 高维数据通常近似存在于一个低维的内在空间上



3维数据近似存在于一个2维平面上

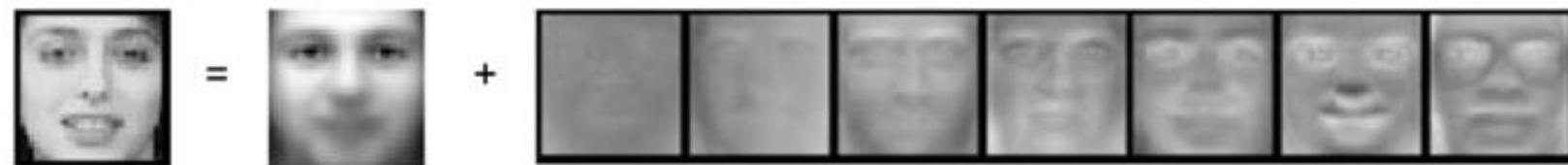


2维数据近似存在于一个1维直线上

关键在于如何找到**主方向**，从而使得数据样本的**表示误差尽可能小**

- 对于真实世界的数据，也可以用低维来表示

例如：如果找到合适的“方向”，人脸可以用很少的几个值很好地表示



$$x \approx \mu_0 + a_1\mu_1 + \cdots + a_7\mu_7$$

具有65536个值的原始图像 x 可以仅用7个值 a_1, \dots, a_7 表示

线性降维PCA大纲

- 动机
- 视角 1：最小化重构误差
- 视角 2：最大化方差
- 视角 3：奇异值分解（SVD）
- PCA的其他应用

新方向下的样本重表示

- 如何在高维空间中表示正交方向？

一组满足以下条件的向量 \mathbf{u}_i

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$

如果 $i = j$, $\delta_{ij} = 1$; 否则 $\delta_{ij} = 0$

定理：在给定 M 个正交方向 $\{\mathbf{u}_i\}_{i=1}^M$ 的情况下，对数据样本 \mathbf{x} 的**最佳近似**为

$$\tilde{\mathbf{x}} = a_1 \mathbf{u}_1 + a_2 \mathbf{u}_2 + \cdots + a_M \mathbf{u}_M$$

其中 a_i 等于

$$a_i = \mathbf{u}_i^T \mathbf{x}$$

证明：

$$\begin{aligned}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2 &= \left\| \mathbf{x} - \sum_{i=1}^M a_i \mathbf{u}_i \right\|^2 \\ &= \|\mathbf{x}\|^2 - 2 \sum_{i=1}^M a_i \mathbf{u}_i^T \mathbf{x} + \sum_{i=1}^M a_i^2\end{aligned}$$

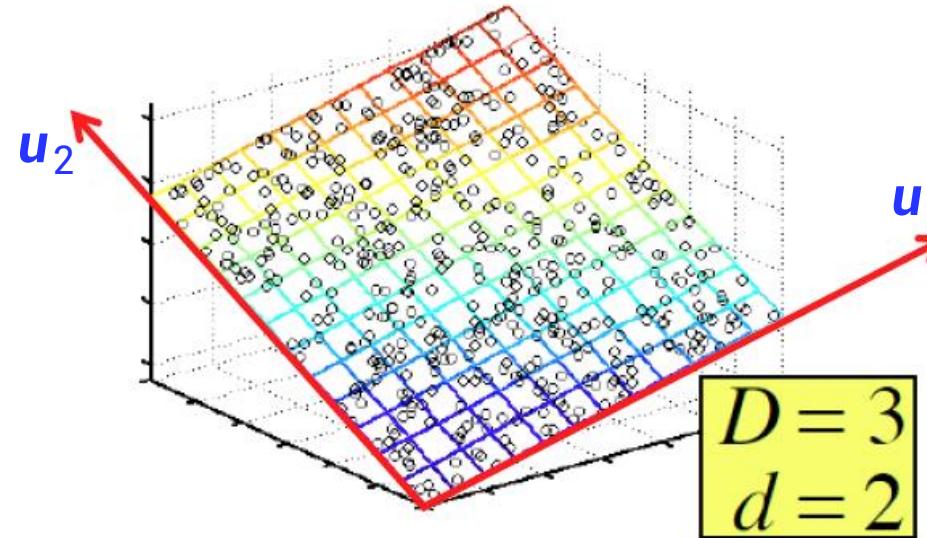
若 $i \neq j$, $\mathbf{u}_i^T \mathbf{u}_j = 0$; 若 $i = j$, $\mathbf{u}_i^T \mathbf{u}_j = 1$

这是一个二次函数，当 $a_i = \mathbf{u}_i^T \mathbf{x}$ 时可以最小化

给定方向 $\{\mathbf{u}_i\}_{i=1}^M$, 最佳系数为 $a_i = \mathbf{u}_i^T \mathbf{x}$ 。但是如何找到最佳方向仍然未知

寻找最佳方向

- **目标:** 给定来自 \mathbb{R}^D 的数据样本 $\{\mathbf{x}^{(n)}\}_{n=1}^N$ ，找到 M 个正交方向 \mathbf{u}_i ，使得原始数据可以在这些方向下得到最佳表示



$$\mathbf{x}^{(n)} \approx \sum_{i=1}^M a_i^{(n)} \mathbf{u}_i$$

- 如果假设给定了最佳方向 $\{\mathbf{u}_i\}_{i=1}^M$ ，那么最佳系数 $a_i^{(n)}$ 是什么？

$$a_i^{(n)} = \mathbf{u}_i^T \mathbf{x}^{(n)}$$

不直接表示数据 $\mathbf{x}^{(n)}$ ，我们首先将数据中心化到原点，即从每个数据点 $\mathbf{x}^{(n)}$ 中减去其均值

$$\mathbf{x}^{(n)} - \bar{\mathbf{x}}, \quad \text{其中 } \bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}^{(n)}$$

- 现在的目标可以描述为——最小化 $\mathbf{x}^{(n)} - \bar{\mathbf{x}}$ 与其在方向 $\{\mathbf{u}_i\}_{i=1}^M$ 下的最佳表示 $\sum_{i=1}^M a_i^{(n)} \mathbf{u}_i$ 之间的误差

$$E \triangleq \frac{1}{N} \sum_{n=1}^N \|(\mathbf{x}^{(n)} - \bar{\mathbf{x}}) - \sum_{i=1}^M a_i^{(n)} \mathbf{u}_i\|^2$$

即：解如下优化问题

$$\underset{a_i^{(n)}, \mathbf{u}_i}{\operatorname{Min}} E,$$

其中系数 $a_i^{(n)}$ 的最佳取值已经知道等于

$$a_i^{(n)} = \mathbf{u}_i^T (\mathbf{x}^{(n)} - \bar{\mathbf{x}})$$

- 化简重建误差 E

a) 注意到 $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$, 重建误差 $E = \frac{1}{N} \sum_{n=1}^N \|(\mathbf{x}^{(n)} - \bar{\mathbf{x}}) - \sum_{i=1}^M a_i^{(n)} \mathbf{u}_i\|^2$ 可表示成

$$E = \frac{1}{N} \left(\sum_{n=1}^N \|\mathbf{x}^{(n)} - \bar{\mathbf{x}}\|^2 - 2 \sum_{n=1}^N \sum_{i=1}^M a_i^{(n)} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T \mathbf{u}_i + \sum_{n=1}^N \sum_{i=1}^M (a_i^{(n)})^2 \right)$$

b) 代入 $a_i^{(n)} = \mathbf{u}_i^T (\mathbf{x}^{(n)} - \bar{\mathbf{x}})$, 可得

$$E = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \bar{\mathbf{x}}\|^2 - \sum_{i=1}^M \mathbf{u}_i^T \underbrace{\frac{1}{N} \sum_{n=1}^N (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T}_{\triangleq \mathbf{S}} \mathbf{u}_i$$

$\triangleq \mathbf{S}$

c) 将其写成矩阵形式给出

$$E = \frac{1}{N} \|\mathbf{X} - \bar{\mathbf{X}}\|_F^2 - \sum_{i=1}^M \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i$$

其中 $\mathbf{X} \triangleq [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}]$, $\|\cdot\|_F$ 表示 Frobenius 范数

- 在约束 $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$ 下, 最小化 $E = \frac{1}{N} \|\mathbf{X} - \bar{\mathbf{X}}\|_F^2 - \frac{1}{N} \sum_{i=1}^M \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i$ 可等价表示为

$$\max_{\mathbf{u}_1, \dots, \mathbf{u}_M} \sum_{i=1}^M \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i$$

$$\text{s. t. : } \mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$

- 考虑 $M = 1$ 的简单情况。问题简化为：

$$\max_{\mathbf{u}_1} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

$$\text{s. t.: } \mathbf{u}_1^T \mathbf{u}_1 = 1$$

➤ 这等价于最大化

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 - \lambda (\mathbf{u}_1^T \mathbf{u}_1 - 1)$$

➤ 对 \mathbf{u}_1 求导并将其设置为 0 给出

$$\mathbf{S} \mathbf{u}_1 = \lambda \mathbf{u}_1,$$

从中我们可以看到 \mathbf{u}_1 应该是 \mathbf{S} 的特征向量

➤ 可以进一步验证，对于最大特征值的特征向量可以最大化 $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$

- 对于 $M = 2$ 的情况，问题变为

$$\max_{\mathbf{u}_1, \mathbf{u}_2} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2$$

$$\text{s. t.: } \mathbf{u}_1^T \mathbf{u}_1 = 1, \mathbf{u}_2^T \mathbf{u}_2 = 1, \mathbf{u}_1^T \mathbf{u}_2 = 0$$

➤ 这等价于最大化

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 - \lambda_1 (\mathbf{u}_1^T \mathbf{u}_1 - 1) + \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2 - \lambda_2 (\mathbf{u}_2^T \mathbf{u}_2 - 1)$$

约束为 $\mathbf{u}_1^T \mathbf{u}_2 = 0$

➤ 对 \mathbf{u}_1 和 \mathbf{u}_2 求导并将其设置为 0 给出

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1, \quad \mathbf{S} \mathbf{u}_2 = \lambda_2 \mathbf{u}_2,$$

⇒ \mathbf{u}_1 和 \mathbf{u}_2 必须是 \mathbf{S} 的特征向量

⇒ 可以看出，为了最大化 $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2$ ， \mathbf{u}_1 和 \mathbf{u}_2 必须是对应于两个最大特征值的特征向量

对于 $M > 1$ 的情况，方向 \mathbf{u}_i 是 \mathbf{S} 对应于最大的 M 个特征值的特征向量

问题：对于 $i \neq j$ ， \mathbf{S} 的特征向量 \mathbf{u}_i 是否满足 $\mathbf{u}_i^T \mathbf{u}_j = 0$ ？

$$\mathbf{A} = \mathbf{X} - \bar{\mathbf{X}}$$

- 对于任何 $D \times D$ 实对称矩阵（例如 $\mathbf{S} \triangleq \mathbf{A}\mathbf{A}^T$ ），它都有 D 个彼此正交的特征向量
- 对于每个 $\mathbf{S} \triangleq \mathbf{A}\mathbf{A}^T$ ，它可以分解为

$$\mathbf{S} = \mathbf{U} \Lambda \mathbf{U}^T$$

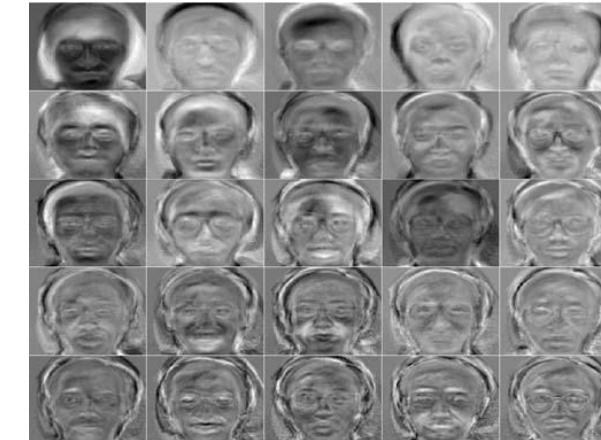
其中 \mathbf{U} 由 \mathbf{S} 的特征向量组成，且 $\mathbf{U}\mathbf{U}^T = I$; Λ 是由 \mathbf{S} 的特征值组成的对角矩阵

示例

输入数据：每张人脸
图像是一个数据点



前 25 个主方向



$$\begin{matrix} \text{face image} \\ = \end{matrix} \quad \begin{matrix} \text{mean face} \\ + \end{matrix} \quad \begin{matrix} \text{7 eigenfaces} \end{matrix}$$

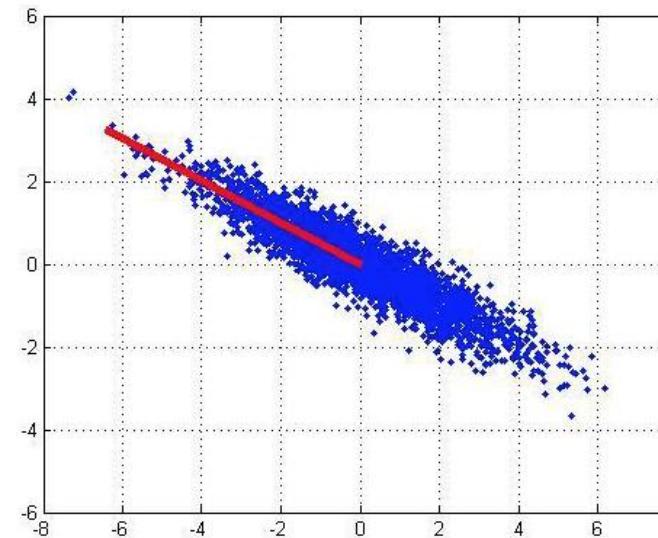
$$x \approx \bar{x} + a_1 u_1 + \cdots + a_7 u_7$$

线性降维PCA大纲

- 动机
- 视角1：最小化重构误差
- 视角2：最大化方差
- 视角3：奇异值分解（SVD）
- 主成分分析（PCA）的其他应用

问题公式化

- **目标:** 给定一个数据集 $\{\mathbf{x}^{(n)}\}_{n=1}^N$ ，找到正交方向 $\{\mathbf{u}_k\}_{k=1}^M$ ，使得数据在这些方向上的投影的方差最大化



最大化方差相当于尽可能多地保留原始数据的信息

- 对于第一个方向 \mathbf{u}_1 , 我们希望沿方向 \mathbf{u}_1 投影的数据的方差最大化, 即 $\{\mathbf{u}_1^T \mathbf{x}^{(n)}\}_{n=1}^N$ 最大化

➤ 方差表达式

$$\begin{aligned} var &= \frac{1}{N} \sum_{n=1}^N (\mathbf{u}_1^T (\mathbf{x}^{(n)} - \bar{\mathbf{x}}))^2 \\ &= \mathbf{u}_1^T \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^{(n)} - \bar{\mathbf{x}}) (\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T \mathbf{u}_1 \\ &= \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 \end{aligned}$$

- 如先前证明的, 在满足 $\mathbf{u}_1^T \mathbf{u}_1 = 1$ 的约束条件下, 当 \mathbf{u}_1 是 \mathbf{S} 对应于最大特征值的特征向量时, 方差最大化

- 对于第二个方向 \mathbf{u}_2 ，它也应该最大化方差

$$var = \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2,$$

但应满足约束条件 $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$ ，即：

$$\mathbf{u}_2^T \mathbf{u}_2 = 1 \quad \mathbf{u}_1^T \mathbf{u}_2 = 0$$

- 由于 \mathbf{u}_1 是对应于最大特征值的特征向量，可以证明 \mathbf{u}_2 是 \mathbf{S} 对应于第二大特征值的特征向量

\mathbf{u}_i 是 $\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$ 的对应于第 i 大特征值的特征向量

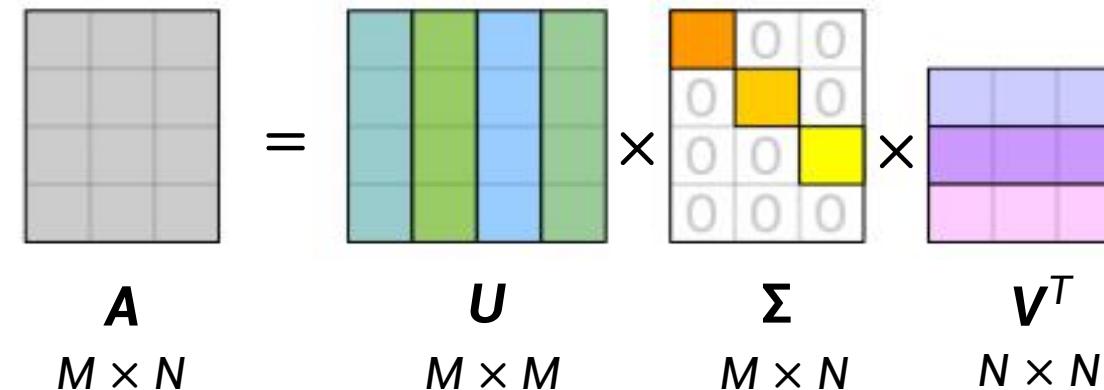
线性降维PCA大纲

- 动机
- 角度 1：最小化重构误差
- 角度 2：最大化方差
- 视角 3：奇异值分解（SVD）
- PCA的其他应用

奇异值分解 (SVD)

- 对于任何 $M \times N$ 矩阵 A ，它总是可以分解为

$$A = U\Sigma V^T$$



- $U = [u_1, \dots, u_M]$ 、 $V = [v_1, \dots, v_N]$ ，其中 u_i 和 v_i 分别是 AA^T 和 A^TA 的第 i 个特征向量，并且 $u_i^T u_j = \delta_{ij}$, $v_i^T v_j = \delta_{ij}$
- Σ 仅在对角线上有非零值，这些值是 AA^T 或 A^TA 的特征值的平方根（ AA^T 和 A^TA 的非零特征值相同）

对角元素 Σ_{ii} 称为**奇异值**，并按降序排列

- 因为 Σ 仅在对角线上有非零值，所以 A 可以表示为

$$A = \sum_{i=1}^r \Sigma_{ii} \mathbf{u}_i \mathbf{v}_i^T = \mathbf{U}' \Sigma' \mathbf{V}'^T$$

其中 \mathbf{u}_i 和 \mathbf{v}_i 是 \mathbf{U} 和 \mathbf{V} 的第 i 列； r 是 Σ 中非零对角元素的数量

$$\mathbf{A} = \mathbf{U}' \times \Sigma' \times \mathbf{V}'^T$$

$$\mathbf{A} \quad \mathbf{U}' \quad \Sigma' \quad \mathbf{V}'^T$$

$$M \times N \quad M \quad r \quad r$$

$$\times r \quad \times r \quad \times N$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

- 在 \mathbf{A} 的奇异值分解 (SVD) 中，向量 \mathbf{u}_i 是矩阵 \mathbf{AA}^\top 对应于其第 i 大特征值的特征向量
- 通过设置 $\mathbf{A} = \tilde{\mathbf{X}}$ ，其中 $\tilde{\mathbf{X}} \triangleq [\mathbf{x}^{(1)} - \bar{\mathbf{x}}, \mathbf{x}^{(2)} - \bar{\mathbf{x}}, \dots, \mathbf{x}^{(N)} - \bar{\mathbf{x}}]$ ，我们可以看到

$$\begin{aligned}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T &= \sum_{n=1}^N (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T \\ &= N \cdot \mathbf{S}\end{aligned}$$

$\Rightarrow \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ 的特征向量与矩阵 \mathbf{S} 相同

通过对 $\tilde{\mathbf{X}}$ 执行 SVD，可以直接获得数据 $\{\mathbf{x}^{(n)}\}_{n=1}^N$ 的主方向，这些主方向是矩阵 \mathbf{U} 的列

- 问题:

给定 $\tilde{\mathbf{X}}$ 的 SVD 分解如下所示， $\tilde{\mathbf{x}}^{(n)}$ 的主方向和系数 a_i 's 是什么？

$$\tilde{\mathbf{X}} = \mathbf{U} \times \Sigma \times \mathbf{V}^T$$

如果只保留前两个方向，系数 a_i 's 是什么？

$$\tilde{\mathbf{X}} \approx \mathbf{U} \times \Sigma \times \mathbf{V}^T$$

线性降维PCA大纲

- 动机
- 视角 1：最小化重构误差
- 视角 2：最大化方差
- 视角 3: 奇异值分解 (SVD)
- PCA的其他应用

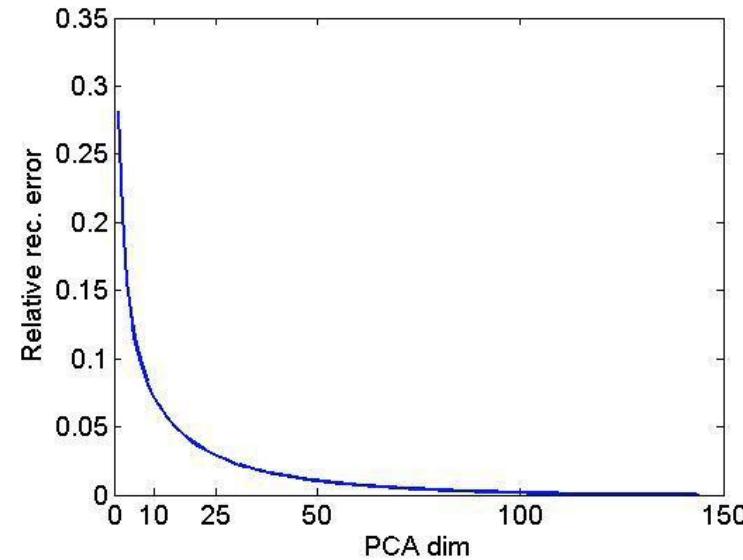
图像压缩

将一个 372×492 的图像分割成多个 12×12 的图像块

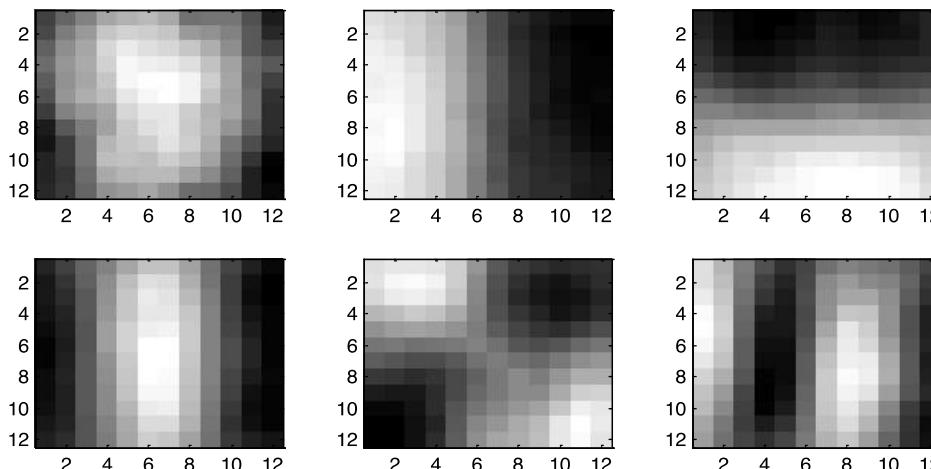
- 每个图像块被视为一个数据实例
- 在这些图像块上执行 PCA



重构误差 vs PCA 分量数



前 6 个 PCA 分量图示





使用前 60 个分量进行重构



使用前 16 个分量进行重构

图像去噪

噪声图像



去噪图像



由前 15 个主成分重构

课堂小结

- PCA的最小重建误差视角
- PCA的方差最大化视角
- PCA的矩阵分解（SVD）视角



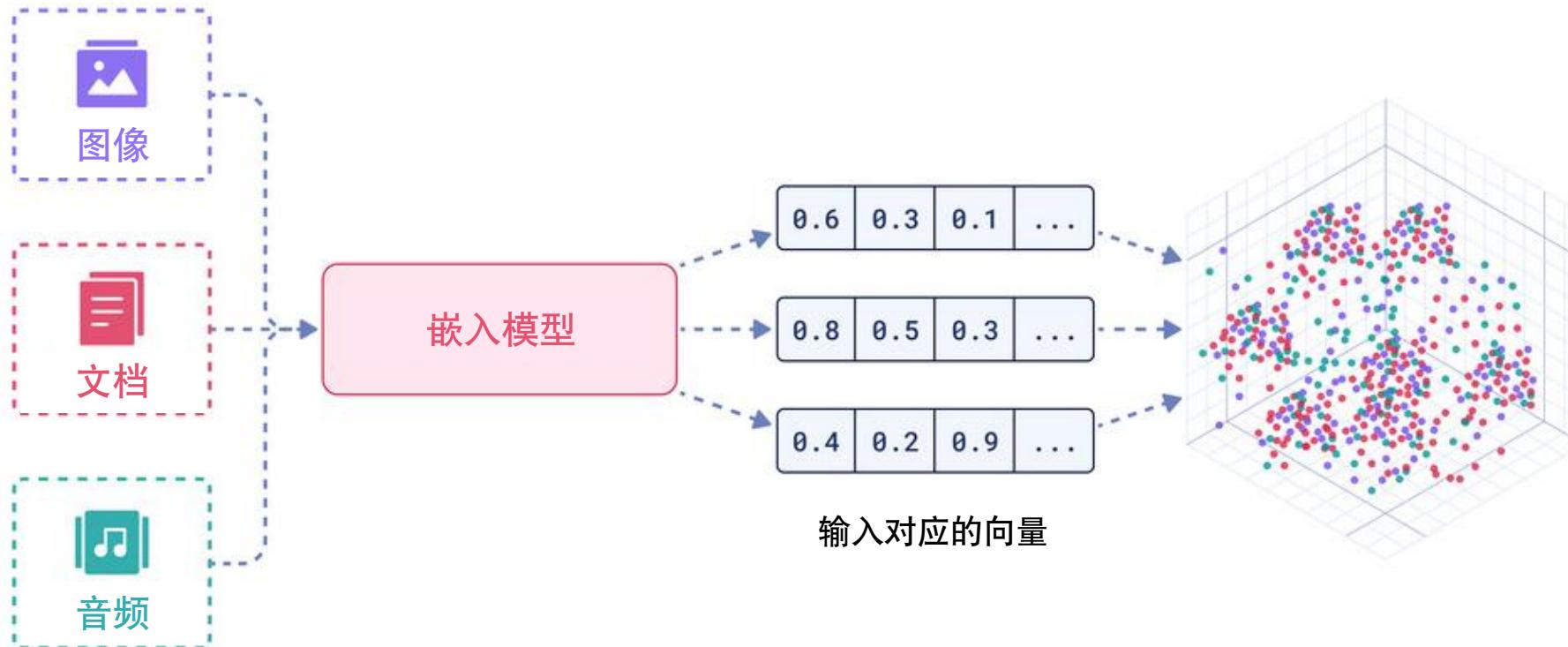
机器学习与数据挖掘

视觉表示学习



什么是表示学习？

表示学习旨在将每个数据实例映射到 \mathbb{R}^d 空间中紧凑的向量



关键在于如何得到这个映射!!!

表示学习的目标

尽可能多地保留原始数据的语义信息



0.6	0.3	0.1	...
-----	-----	-----	-----



0.8	0.5	0.3	...
-----	-----	-----	-----



0.4	0.2	0.9	...
-----	-----	-----	-----

表示学习的目标

尽可能多地保留原始数据的语义信息



0.6	0.3	0.1	...
-----	-----	-----	-----



0.8	0.5	0.3	...
-----	-----	-----	-----



0.4	0.2	0.9	...
-----	-----	-----	-----

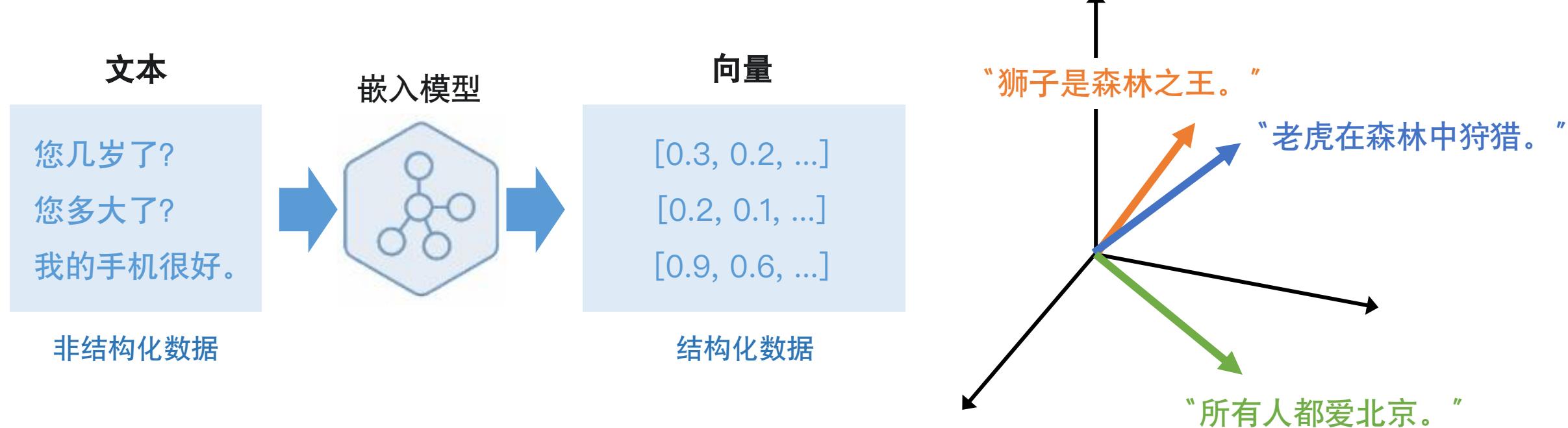
推远

拉近

表示学习的目标

文本表示学习

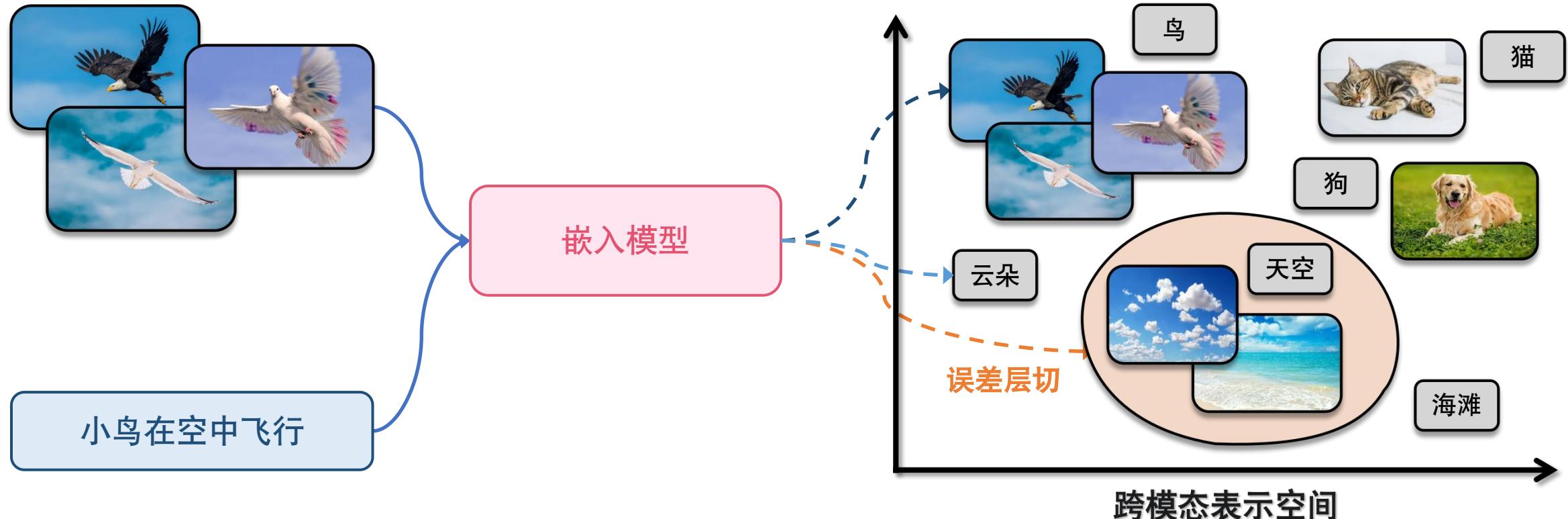
将文本映射为向量，向量间的某种距离能够反映语义相似性



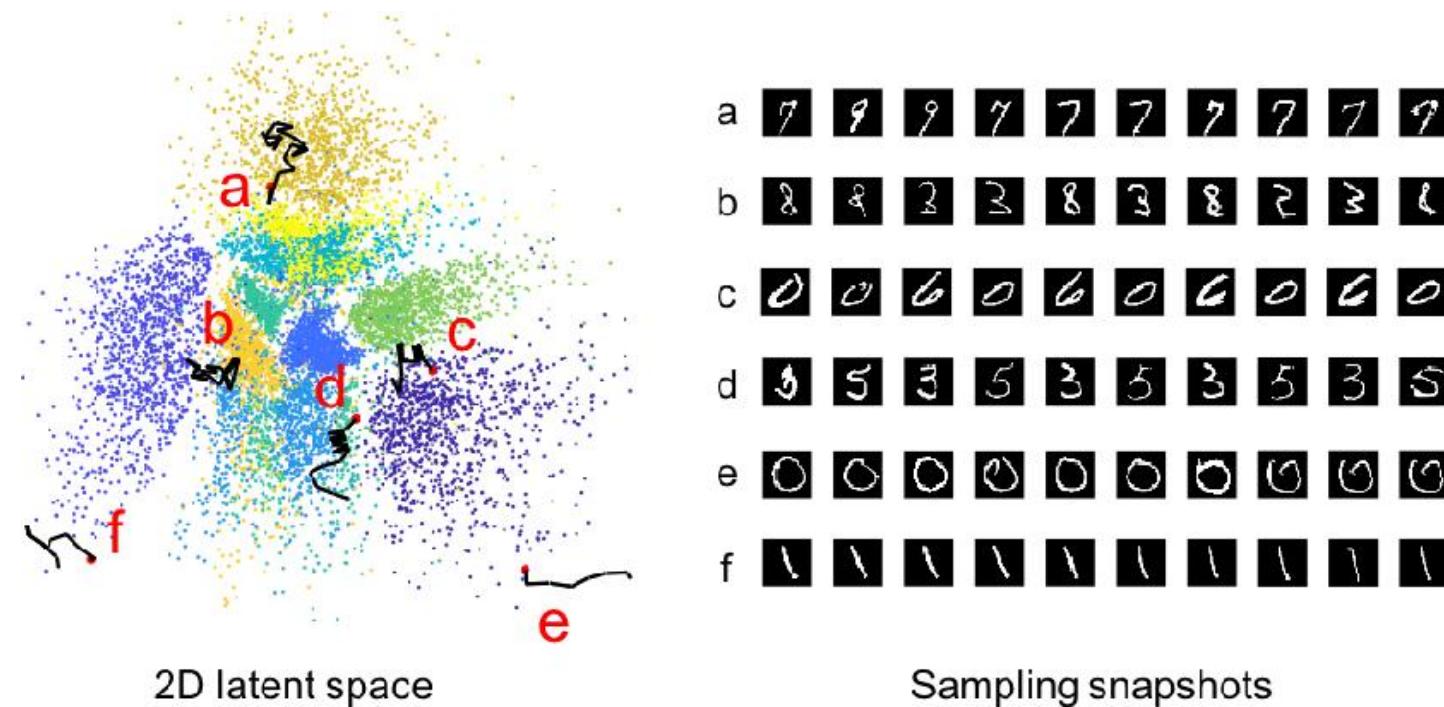
表示学习的目标

跨模态表示学习

学习共享的表示空间，对齐不同模态的语义信息



例子：用 \mathbb{R}^2 中的向量表示手写数字，为了获得良好的表示，来自同一数字的图像的表示应该彼此接近。



- 什么样的表示才算好？

好的表示应该尽可能保留原始数据的语义信息，而不是所有信息。

为什么需要表示学习？

- 与原始数据相比，表示对于下游应用更加友好，这是因为表示具有以下特点：
 - ◆ 丢弃无关信息，同时保留重要的语义信息
 - ◆ 更加紧凑

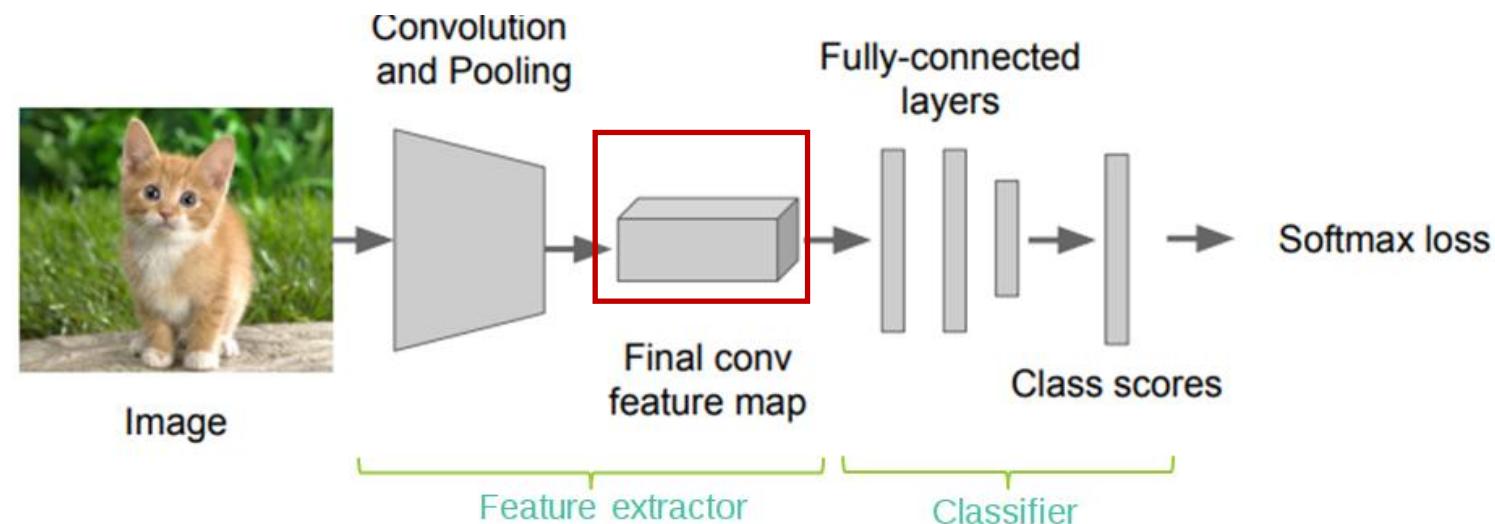
- 表示可以应用于各种下游应用，例如：
 - ◆ 分类
 - ◆ 聚类
 - ◆ 相似性搜索
 - ◆ 异常检测

大纲

- 基于迁移学习的方法
- 自编码器
- 对比学习
- ViT & MAE预训练

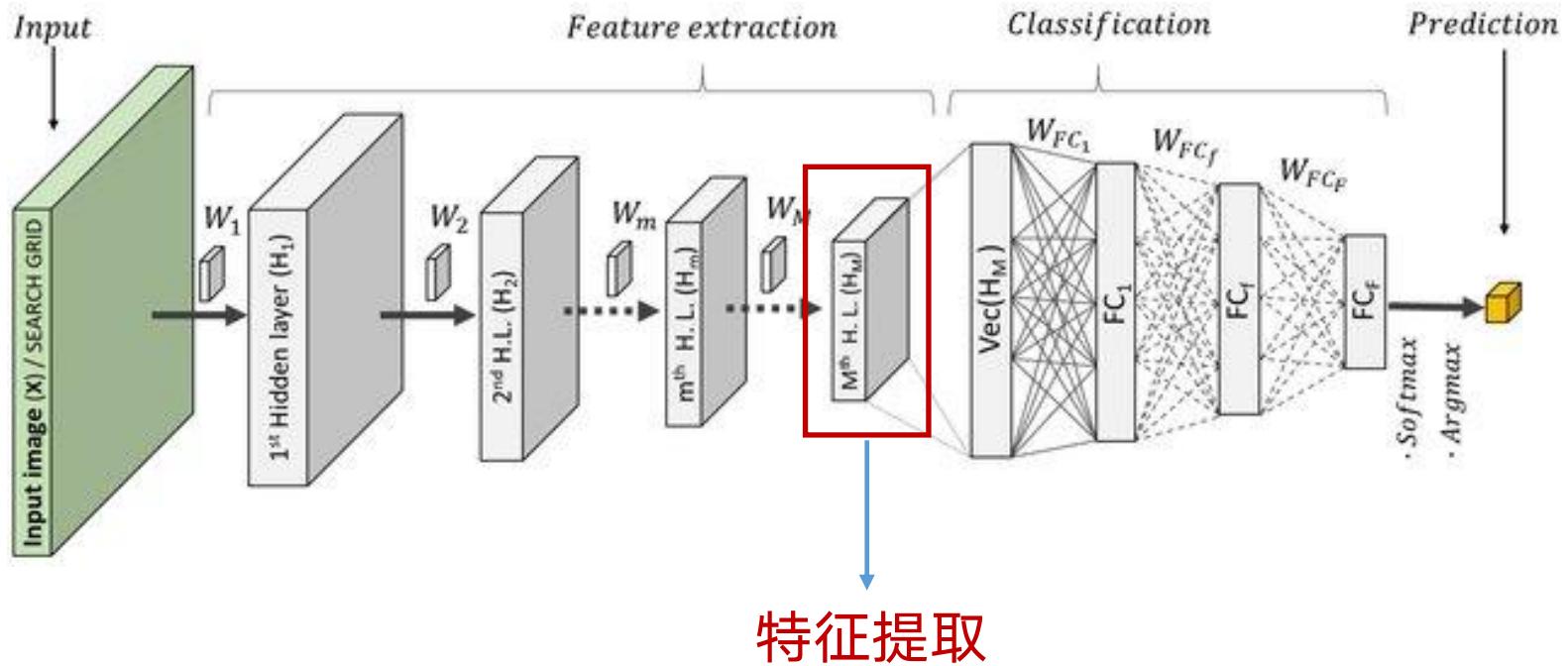
监督学习案例

- 给定一组图像-标签对 $\{x_i, y_i\}_{i=1}^N$ ，我们在给定的图像-标签对上训练一个分类器



- 最后一个卷积层的特征图输出可以看作是输入图像的表示

为什么??



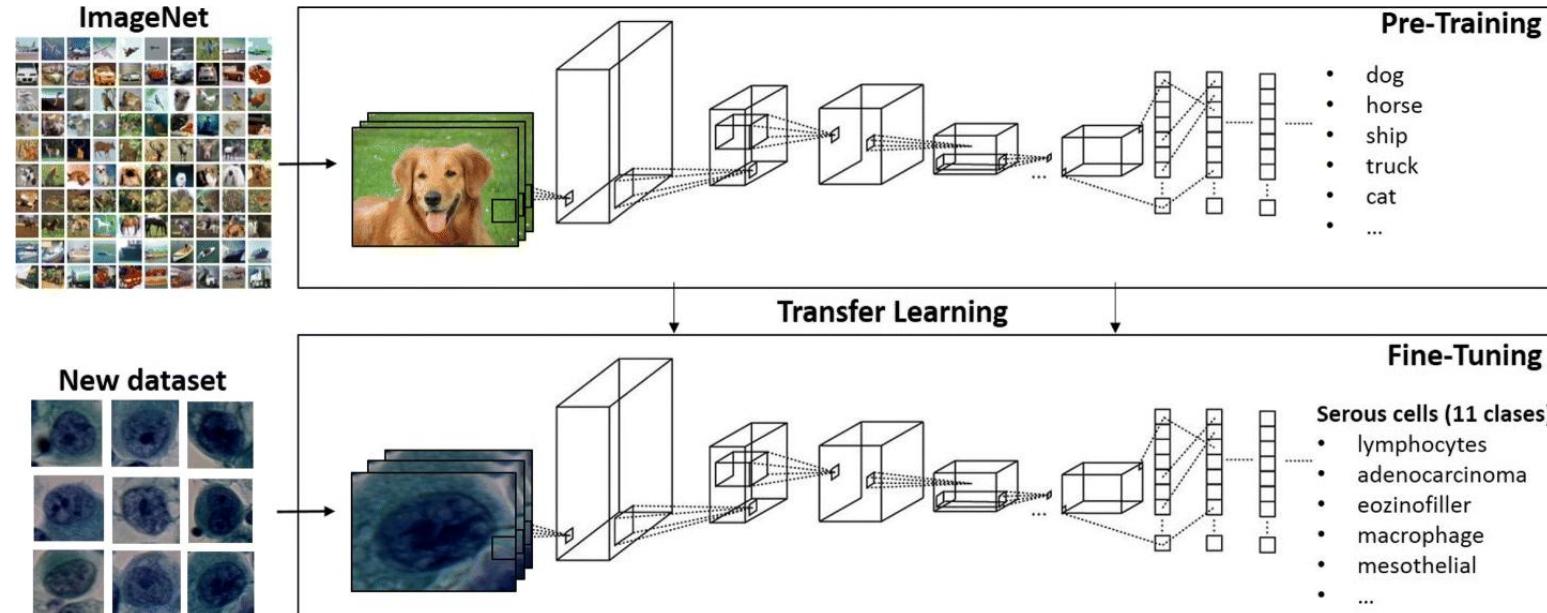
除了卷积特征图，全连接倒数第二层的输出有时也被用作表示

大纲

- 基于迁移学习的方法
- 自编码器
- 对比学习
- ViT & MAE预训练

无标签场景

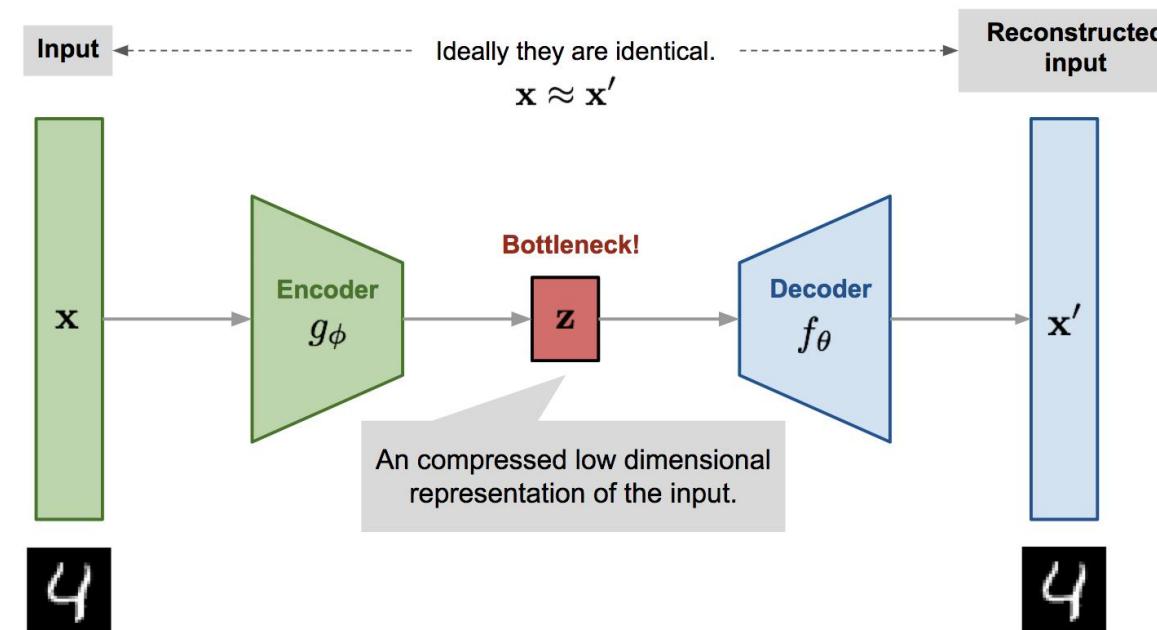
- 当给定的数据集不包含标签时
 - 在一个带有标签的辅助数据集（例如，ImageNet）上训练 CNN）
 - 可以通过将新数据集的图像传递到预训练的 CNN 来获得图像的表示



预训练的 CNN 可以被看作是一个特征提取器

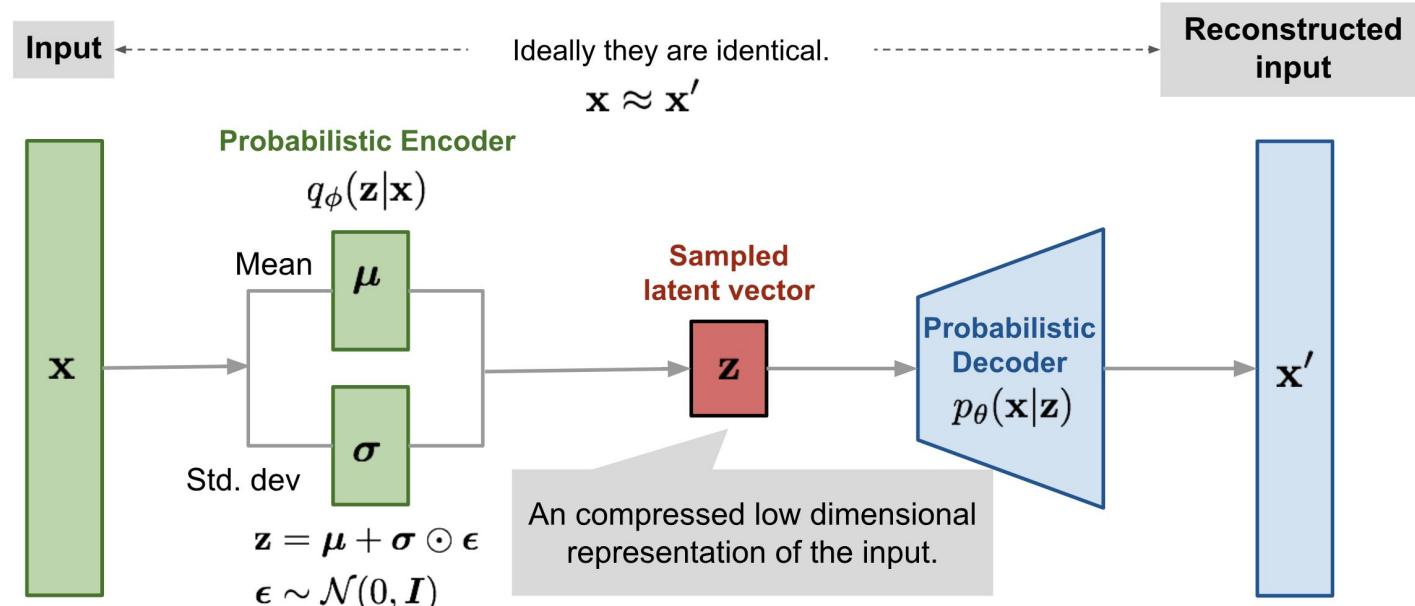
无标签场景&辅助数据集

- 在这种情况下，我们使用无监督学习
- 一个直接的方法是在给定的无标签数据 $\{x_i\}_{i=1}^N$ 上训练一个自编码器，然后使用编码器来提取表示



PCA 可以被看作是一个线性自编码器

- 我们也可以在无标签数据 $\{x_i\}_{i=1}^N$ 上训练**生成模型**，并使用生成模型来提取表示



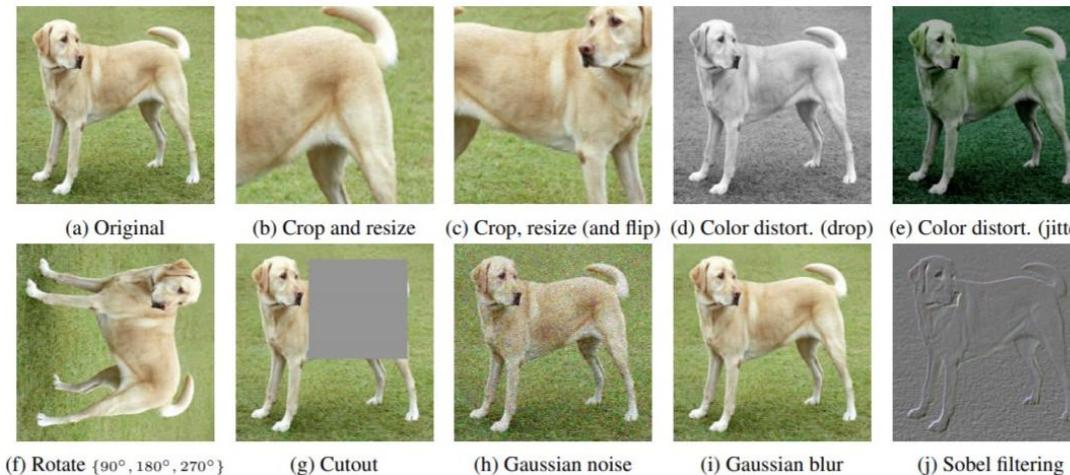
使用生成方法，各种先验知识可以很容易地施加到学习过程中

大纲

- 基于迁移学习的方法
- 自编码器
- 对比学习
- ViT & MAE预训练

对比学习：SimCLR [Chen et al., 2020]

- 数据增强



➤ 通过应用上述一种或多种数据增强方法来扩增原始图像 \mathbf{x}_i ，得到

$$\mathbf{x}_i^+ = \text{aug}(\mathbf{x}_i)$$

- SimCLR 的基本思想是拉近 \mathbf{x}_i 和 \mathbf{x}_i^+ 的表示，同时将 \mathbf{x}_i 的表示推离其他样本 \mathbf{x}_j 和 \mathbf{x}_j^+ 的表示，其中 $j \neq i$

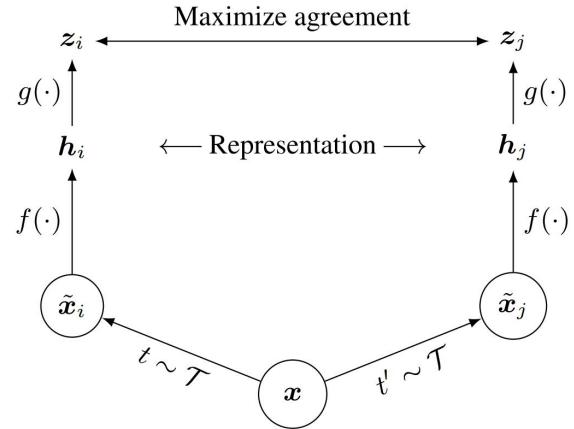
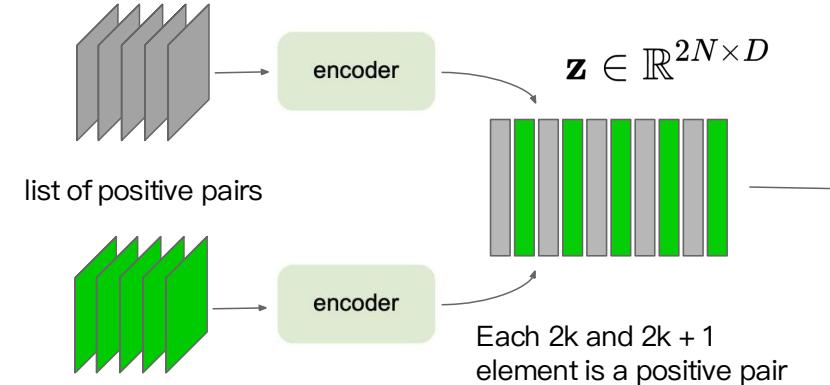


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation h for downstream tasks.



➤ 所有的 z_i 都来自同一个神经网络

$$\text{Loss} = -\log \frac{e^{sim(z_i, z_i^+)/\tau}}{e^{sim(z_i, z_i^+)/\tau} + \sum_{j \neq i} e^{sim(z_i, z_j)/\tau} + \sum_{j \neq i} e^{sim(z_i, z_j^+)/\tau}}$$

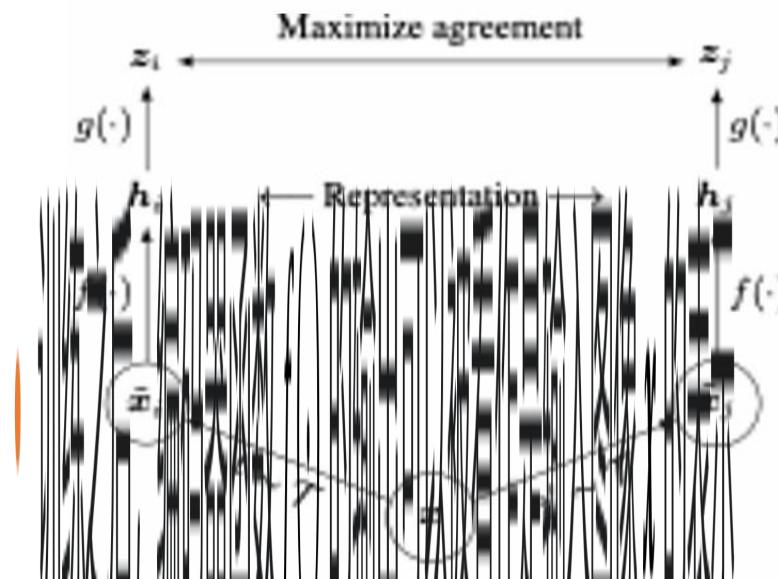
其中 $z = g(f(x))$, $sim(z_i, z_j) = z_i^T z_j / \|z_i\| \|z_j\|$

InfoNCE loss, 和互信息紧密相关

- 训练之后，编码器函数 $f(\cdot)$ 的输出可以被看作是输入图像 x 的表示

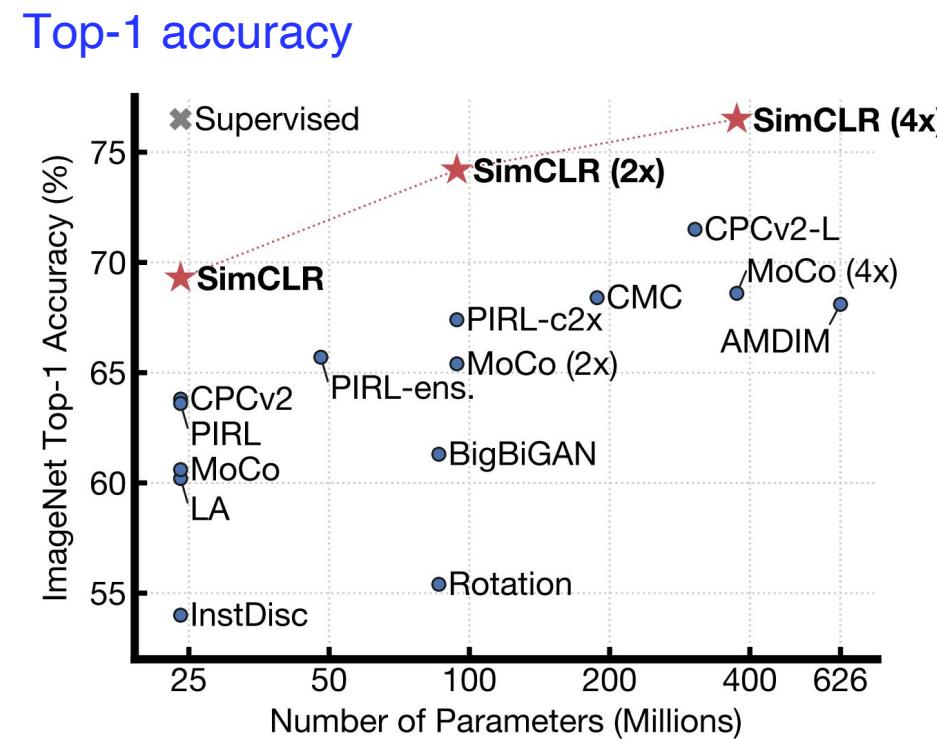
$$h = f(x)$$

$g(\cdot)$ 被称为投影头，它通常由浅 MLP 实现



评估表示的质量

- 对学习到的特征进行线性探测
 - 在 ImageNet 的训练集上用 SimCLR 训练特征编码器
 - 冻结特征编码器，在输出特征之上**训练线性分类器**



- SimCLR 特征上的半监督学习

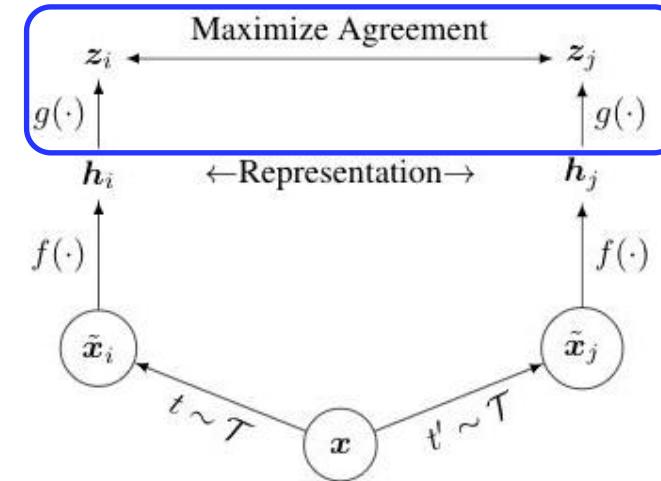
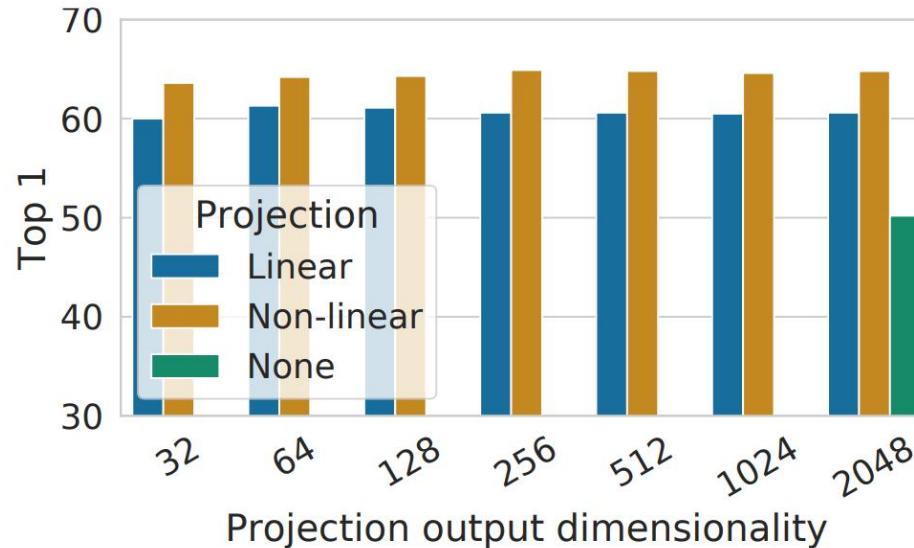
- 在 ImageNet 的训练集上用 SimCLR 训练特征编码器
- 在 ImageNet 上用 1% / 10% 的标记数据微调编码器

Method	Architecture	Label fraction	
		1%	10%
Top 5			
Supervised baseline	ResNet-50	48.4	80.4
<i>Methods using other label-propagation:</i>			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
<i>Methods using representation learning only:</i>			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(*)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	85.8	92.6

Table 7. ImageNet accuracy of models trained with few labels.

SimCLR 设计选择

- 投影头

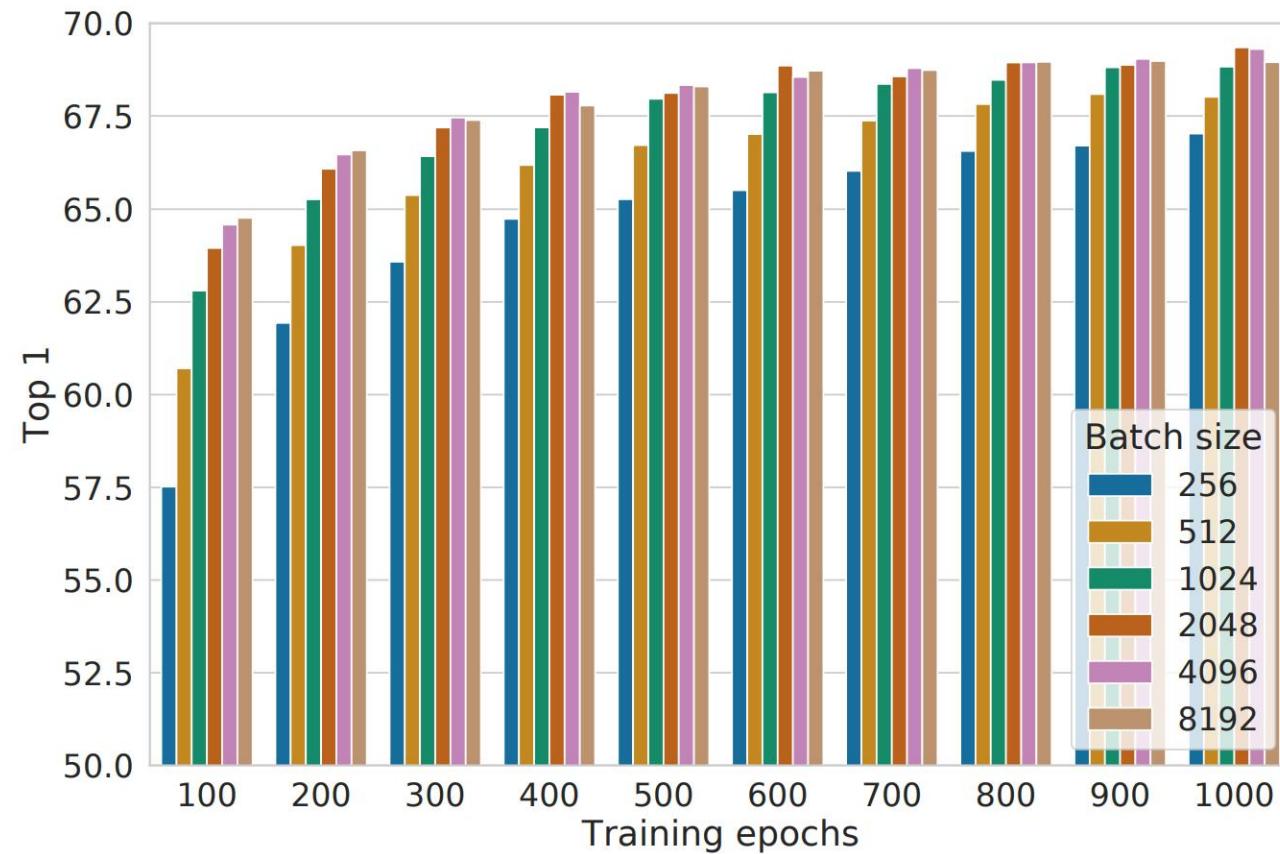


线性/非线性投影头大幅提高了表示质量。可能的原因是：

- 1) 空间 z 更像是一个专门为 SimCLR 代理任务优化的空间
- 2) 通过利用投影头 $g(\cdot)$ ，更多的信息可以保留在表示空间 h 中

- 使用大批量

- 大的训练批量大小对于SimCLR 至关重要
- 但是大的训练批量也会导致大的内存占用，从而对硬件提出更高的要求



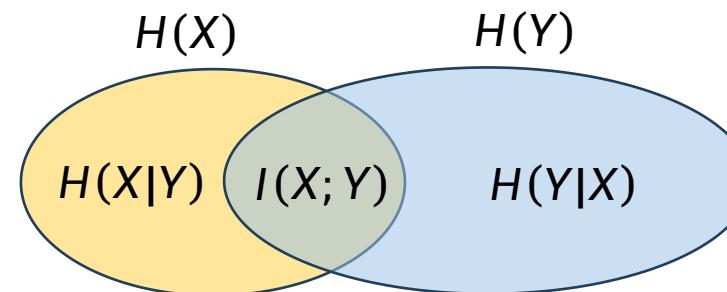
与互信息的联系

- 对于服从联合分布 $p(x, y)$ 的两个随机变量 X 和 Y ，它们之间的互信息定义为

$$I(X; Y) \triangleq \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \quad \longleftrightarrow \quad KL(p(x, y) \| p(x)p(y))$$

$$= \int p(x, y) \log \frac{p(y|x)}{p(y)} dx dy \quad \longleftrightarrow \quad H(Y) - H(Y|X)$$

$$= \int p(x, y) \log \frac{p(x|y)}{p(x)} dx dy \quad \longleftrightarrow \quad H(X) - H(X|Y)$$



- 互信息的下界被证明为

$$I(X; Y) \geq \log K + \mathbb{E}_{p(x,y) \prod_{k=1}^{K-1} p(y_k)} \left[\log \frac{e^{s(x,y)}}{e^{s(x,y)} + \sum_{k=1}^{K-1} e^{s(x,y_k)}} \right] \triangleq L$$

其中 $p(y) = \int p(x, y) dx$ 是 $p(x, y)$ 的边缘分布； $s(x, y)$ 可以是任何形式。

实践中， $s(x, y)$ 被训练以获得尽可能紧的下界

- 从 $p(x, y)$ 中抽取一个样本 (x, y) ，并从 $p(y)$ 中抽取 $K - 1$ 个独立样本 $\{y_k\}_{k=1}^{K-1}$ ，我们可以将下界估计为

$$L \approx \log K + \log \frac{e^{s(x,y)}}{e^{s(x,y)} + \sum_{k=1}^{K-1} e^{s(x,y_k)}}$$

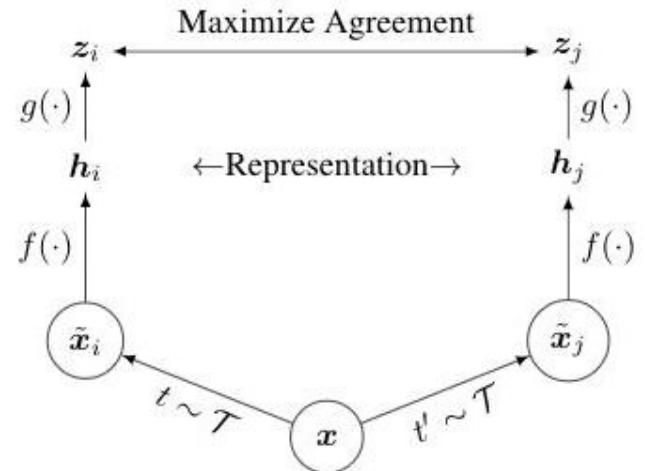
Poole, Ben, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. "On variational bounds of mutual information." In *International Conference on Machine Learning*, pp. 5171-5180, 2019.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." *arXiv preprint arXiv:1807.03748* (2018).

- 与 InfoNCE 损失的联系

➤ 定义两个随机变量 Z 和 Z^+ 如下

$$(z, z^+) = (g(f(\mathbf{T}(x))), g(f(\mathbf{T}(x^+))))$$



其中 $x \sim p(x)$, 表示训练样本的分布; 而 $\mathbf{T}(\cdot)$ 是随机增强算子

➤ 将样本 (z, z^+) 的分布表示为 $p(z, z^+)$

➤ 那么, Z 和 Z^+ 之间的互信息可以被限制为如下

$$I(Z; Z^+) \geq \log N + \log \frac{e^{sim(z, z^+)}}{e^{sim(z, z^+)} + \sum_{k=1}^{N-1} e^{sim(z, z_k^+)}}$$

其中 $(z, z^+) \sim p(z, z^+)$, $z_k^+ \sim p(z^+)$

➤ 边缘分布 $p(z)$ 和 $p(z^+)$ 是相同的，因此我们可以

- 将 (z_i, z_i^+) 视为从联合分布 $p(z, z^+)$ 中抽取的样本
- 将所有其他样本 $\{z_j, z_j^+\}_{j \neq i}$ 视为从 $p(z^+)$ 中独立抽取的样本

那么，下界可以写成如下形式

$$I(Z; Z^+) \geq \log(2N - 1) + \log \frac{e^{sim(z_i, z_i^+)}}{e^{sim(z_i, z_i^+)} + \sum_{j \neq i} e^{sim(z_i, z_j)} + \sum_{j \neq i} e^{sim(z_i, z_j^+)}}$$

它是 InfoNCE 损失的负数

因此，最小化 InfoNCE 损失可以理解为最大化 Z 和 Z^+ 之间的互信息

- 动量对比学习 [He et al., 2019]
 - 负样本的数量在对比学习中非常重要
 - 如何解决这个问题？-> 记忆库

- MoCo also optimizes contrastive learning objective

$$\mathcal{L}_{q,k^+, \{k^-\}} = -\log \frac{\exp(q \cdot k^+ / \tau)}{\exp(q \cdot k^+ / \tau) + \sum_{k^-} \exp(q \cdot k^- / \tau)}$$

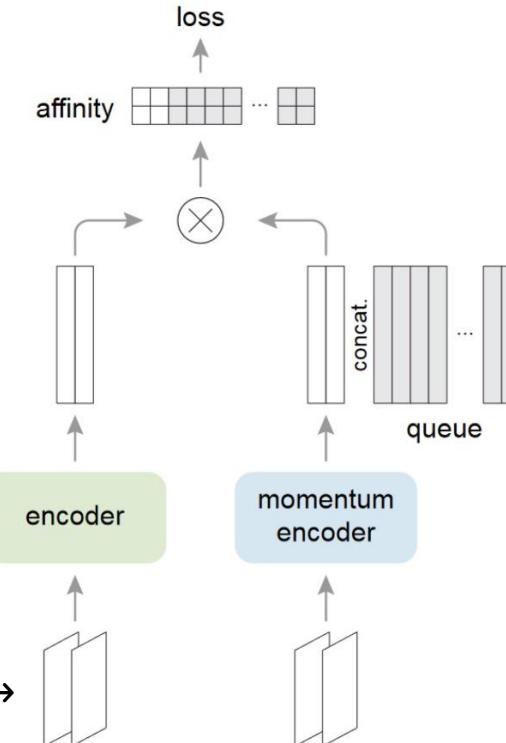
- After encoder is updated,

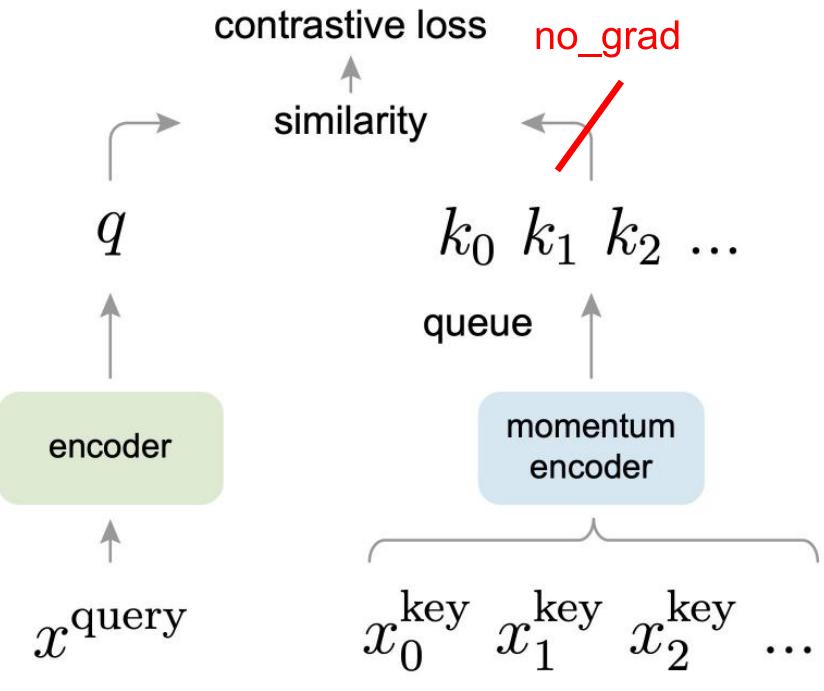
- Momentum encoder is updated by

$$\theta_{\text{momentum}} \leftarrow m\theta_{\text{momentum}} + (1 - m)\theta$$

- Add the current positive keys k^+ into the queue

Randomly augmented samples →





Key differences to SimCLR:

- Keep a running **queue** of keys (negative samples).
- Compute gradients and update the encoder **only through the queries**.
- Decouple min-batch size with the number of keys: can support **a large number of negative samples**.
- The key encoder is **slowly progressing** through the momentum update rules:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q$$

Source: [He et al., 2020](#)

“MoCo V2”

Improved Baselines with Momentum Contrastive Learning

Xinlei Chen Haoqi Fan Ross Girshick Kaiming He

Facebook AI Research (FAIR)

A hybrid of ideas from SimCLR and MoCo:

- **From SimCLR:** non-linear projection head and strong data augmentation.
- **From MoCo:** momentum-updated queues that allow training on a large number of negative samples (no TPU required!).

Source: [Chen et al., 2020](#)

- 该模型进行了消融实验，研究了非线性投影头、强数据增强和训练轮数的影响

MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

case	unsup. pre-train				ImageNet acc.	VOC detection		
	MLP	aug+	cos	epochs		AP ₅₀	AP	AP ₇₅
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	82.5	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	800	71.1	82.5	57.4	64.0

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). “MLP”: with an MLP head; “aug+”: with extra blur augmentation; “cos”: cosine learning rate schedule.

MoCo vs. SimCLR vs. MoCo V2

case	MLP	aug+	cos	unsup. pre-train epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	67.5
<i>results of longer unsupervised training follow:</i>						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	71.1

Table 2. **MoCo vs. SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. “aug+” in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	5.0G	53 hrs
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G [†]	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. [†]: based on our estimation.

Key takeaways:

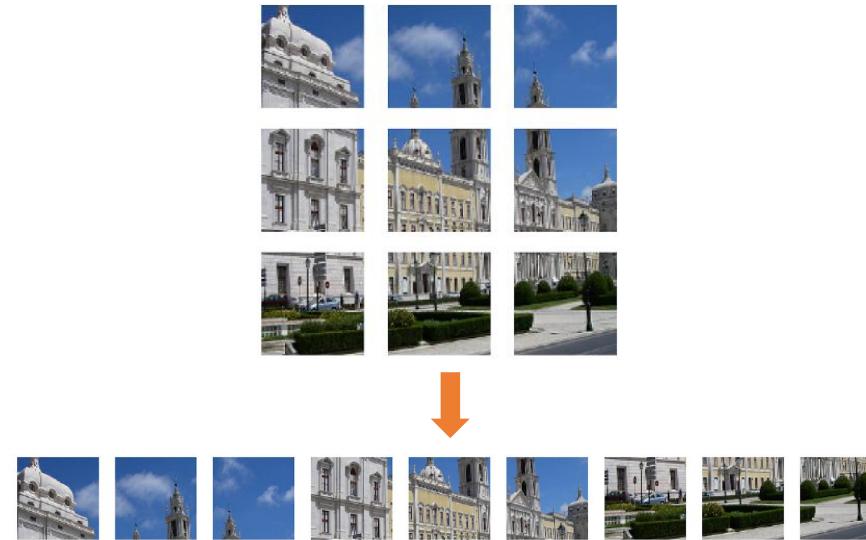
- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).
- ... all with much smaller memory footprint! (“end-to-end” means SimCLR here)

大纲

- 基于迁移学习的方法
- 自编码器
- 对比学习
- ViT & MAE预训练

视觉Transformer (ViT)

- 使用类似 BERT 的编码器进行图像表示学习
 - 通过将图像分割成 N 个图像块的序列来创建“词”的概念



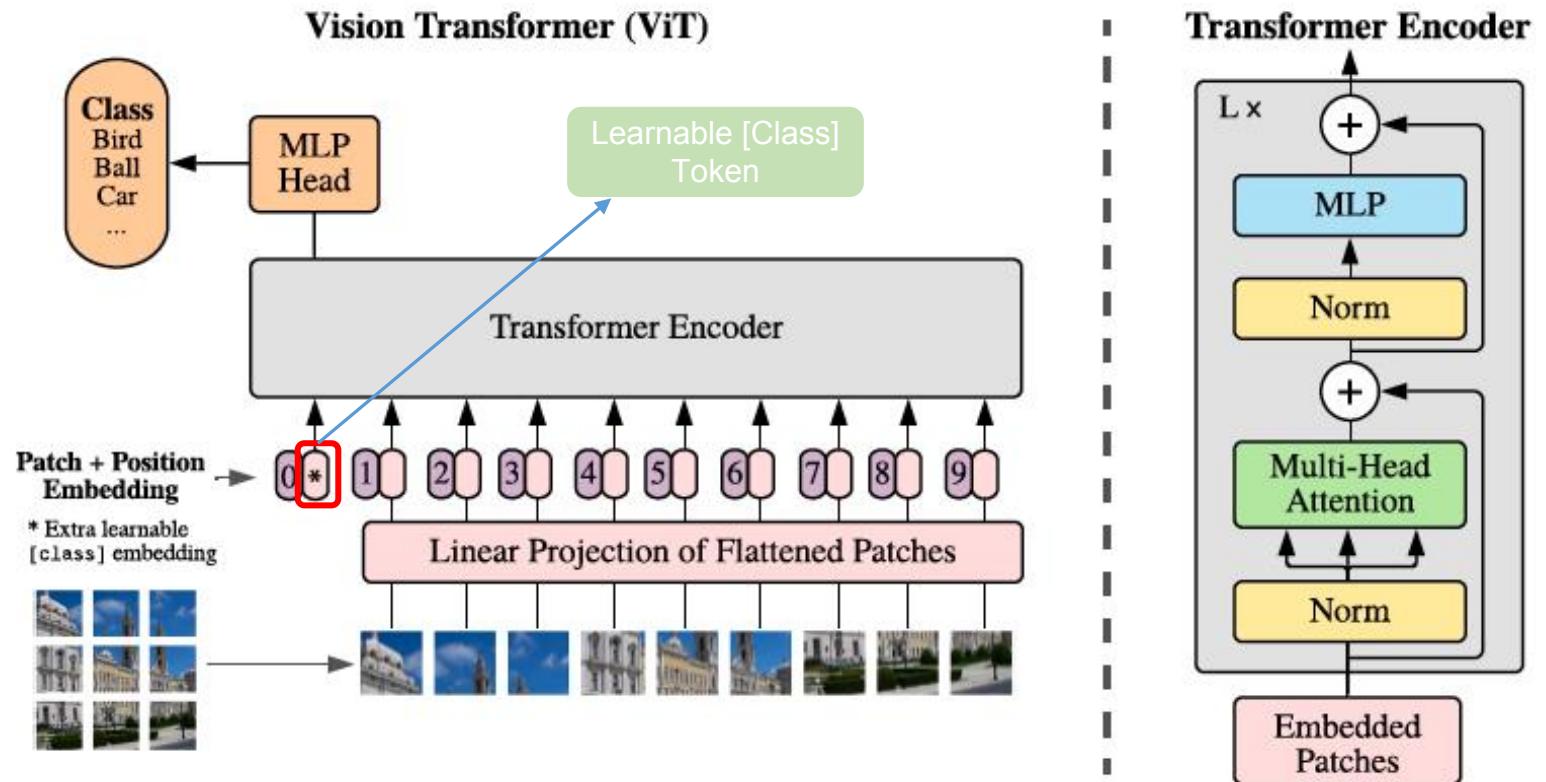
- 将压平的图像块进行线性投影 $\mathbf{x}_p^i \in \mathbb{R}^{16^2 \cdot 3}$ for $i = 1, 2, \dots N$

$$\mathbf{z}_0^i = \mathbf{x}_p^i \mathbf{E},$$

其中 $\mathbf{E} \in \mathbb{R}^{16^2 \cdot 3 \times D}$ 是线性嵌入矩阵

Alexey Dosovitskiy et. al, An Image is Worth 16 x 16 Words: Transformers for Image Reconstruction at Scale, ICLR 2021

- 引入一个可学习的 [class] 标记，表示为 z_0^0 ，类似于 BERT 的 [CLS] 标记
- 为位置 $0, 1, 2, \dots, N$ 引入可学习的一维位置嵌入，可以表示为 $E_{pos} \in \mathbb{R}^{(N+1) \times D}$



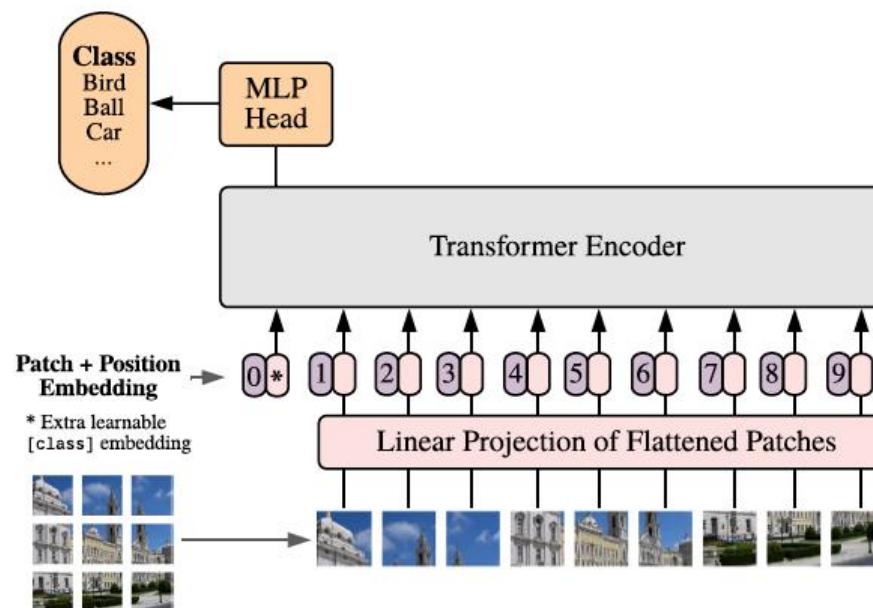
➤ 更新规则

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1 \mathbf{E}; \mathbf{x}_p^2 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{pos}$$

$$\mathbf{z}'_\ell = MSA(LN(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}$$

$$\mathbf{z}_\ell = MLP(LN(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell$$

其中 $MSA(\cdot)$ 表示多头注意力



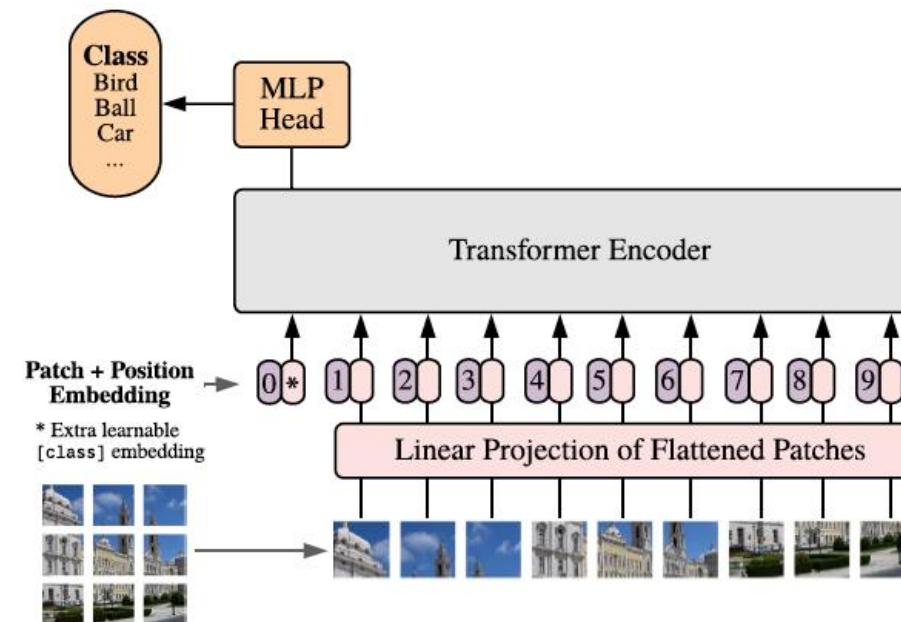
- ViT 变体的详细信息

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

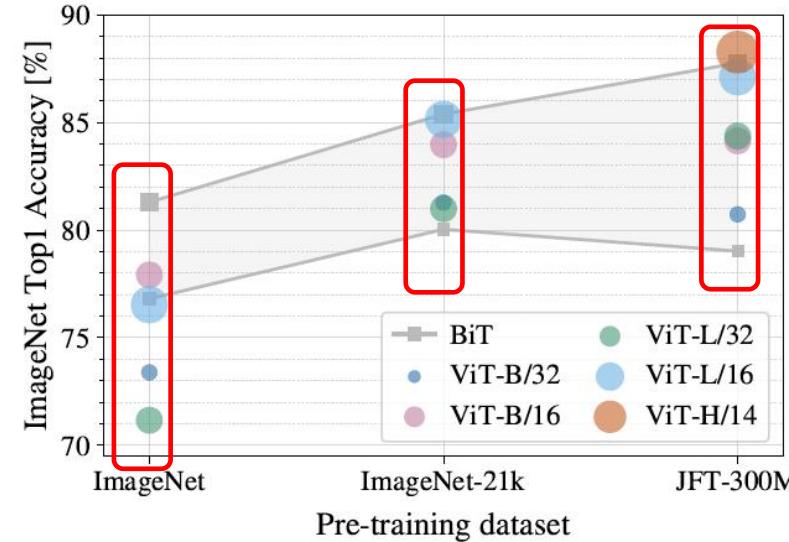
ViT 的监督预训练

JFT300M包含 3 亿张带标签的图像，是谷歌为内部使用而开发的专有数据集

JFT300M比著名的 ImageNet 数据集大得多

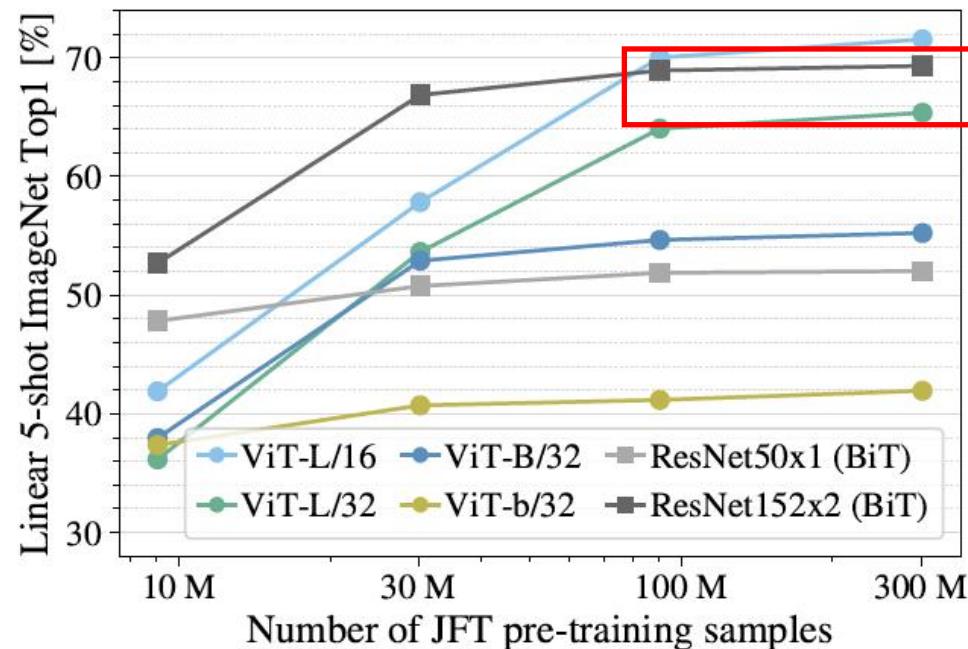


- 在不同的数据集上进行监督预训练，然后在 ImageNet 上进行微调以进行评估
 - ViT-L/16 意味着图像块大小设置为 16×16
 - BiT 指的是在 ImageNet 和辅助数据集上训练的 ResNet



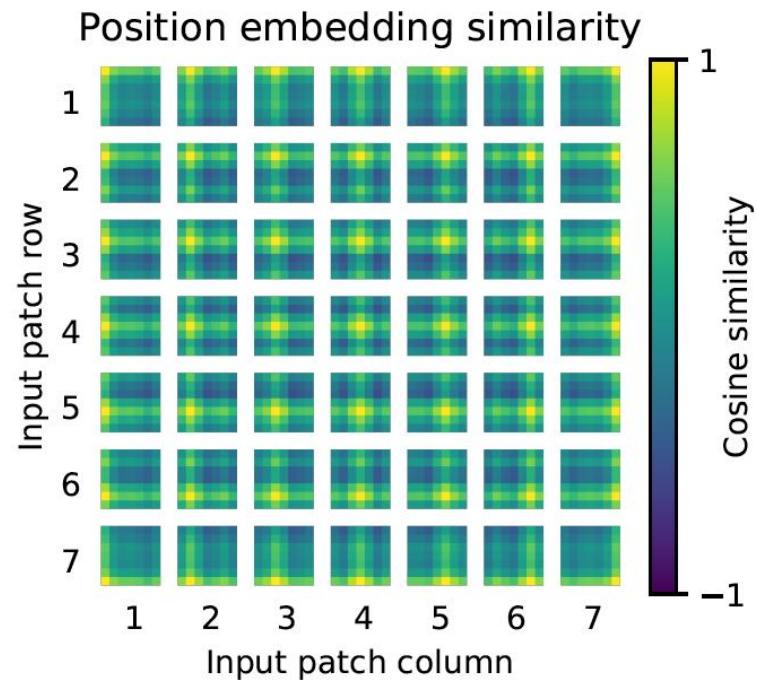
- ✓ 对于小型预训练数据集，ResNet 表现更好，ViT-B 表现优于 ViT-L，这可能是由于 CNN 对图像的归纳偏置以及大型 ViT 容易过拟合
- ✓ 对于大型数据集，ViT-L 和 ViT-H 的优势比 ResNet 和 ViT-B 更为突出
- ✓ 小图像块 (16×16) 比大图像块 (32×32) 更受欢迎

- 在不同比例的 JFT-300 数据集上进行监督预训练，然后在 ImageNet 上对输出表示进行线性探测



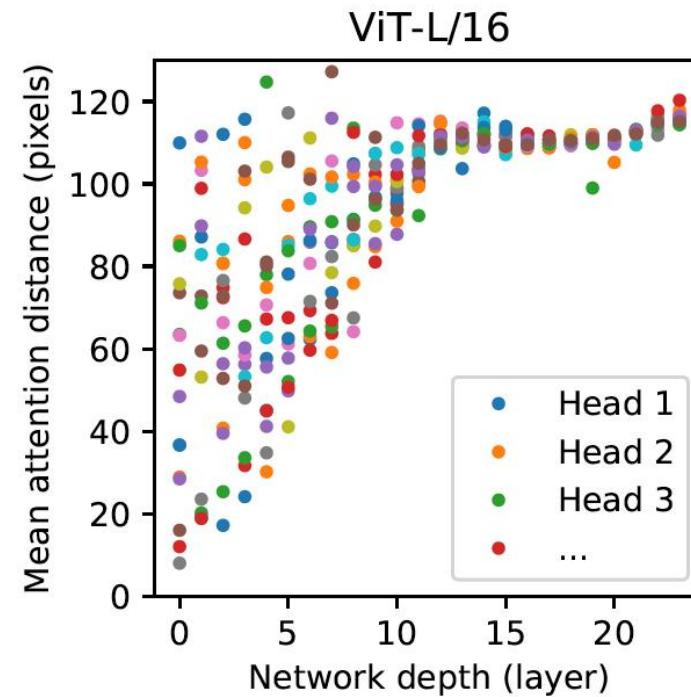
- ✓ 结果表明，当预训练数据集足够大时，ViT-L 学习到的表示比 ResNet 更好
 - 从海量数据中学到的知识压倒了 CNN 的归纳偏置

➤ ViT-L/32 的位置嵌入的相似性



- 学习到的位置嵌入可以自己发现相关的行和列

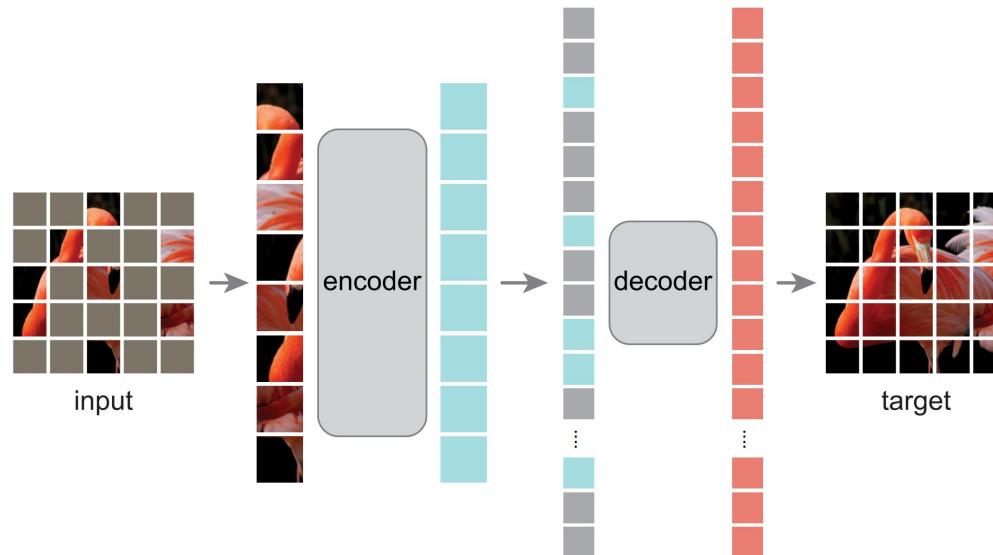
➤ 每个头和网络深度对关注区域大小的影响。图中每个点代表16个注意力头中的一个，并显示了其在所有图像上的平均注意力距离。



- 浅层：一些头关注局部图像块，而一些头关注远距离图像块
- 深层：大多数关注远距离图像块

ViT 的 MAE 预训练

模型结构



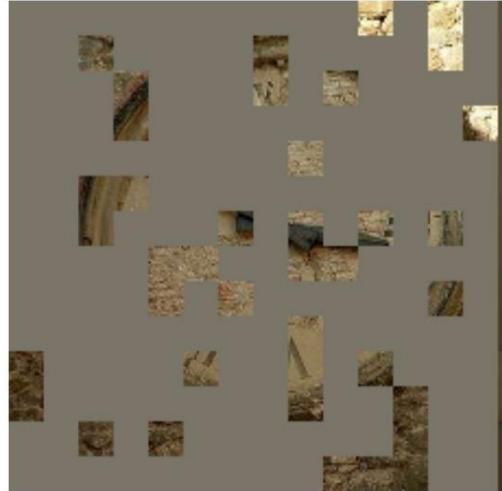
- 基于 Transformer 的编码器和解码器
- 非对称编码器-解码器架构，即轻量级解码器
- 高掩蔽率，75% 的图像块是随机掩蔽的

训练目标：基于未掩蔽的图像块预测掩蔽的图像块

事实证明，这种预训练方法比在 ViT 上进行有监督的预训练更有效率，**所需的数据更少**

Kaiming He, et al., Masked Autoencoders are Scalable Vision Learners, CVPR 2022

输入图像块



预测结果



真实图像



设计选择

- 轻量级解码器就足够了

blocks	ft	lin
1	84.8	65.5
2	84.9	70.0
4	84.9	71.9
8	84.9	73.5
12	84.4	73.3

(a) **Decoder depth.** A deep decoder can improve linear probing accuracy.

dim	ft	lin
128	84.9	69.1
256	84.8	71.3
512	84.9	73.5
768	84.4	73.1
1024	84.3	73.1

(b) **Decoder width.** The decoder can be narrower than the encoder (1024-d).

- 解码器的深度和宽度对通过微调提高性能的影响较小
- 只有单个 transformer 块的解码器可以通过微调表现出色

- 在编码器中跳过 mask token [M]，并稍后在解码器中应用

case	ft	lin	FLOPs
encoder w/ [M]	84.2	59.6	3.3×
encoder w/o [M]	84.9	73.5	1×

(c) **Mask token.** An encoder without mask tokens is more accurate and faster (Table 2).

- 使用逐图像块归一化预测像素可提高准确性
- MAE 在仅使用裁剪增强时效果很好

case	ft	lin
pixel (w/o norm)	84.9	73.5
pixel (w/ norm)	85.4	73.9
PCA	84.6	72.3
dVAE token	85.3	71.6

(d) **Reconstruction target.** Pixels as reconstruction targets are effective.

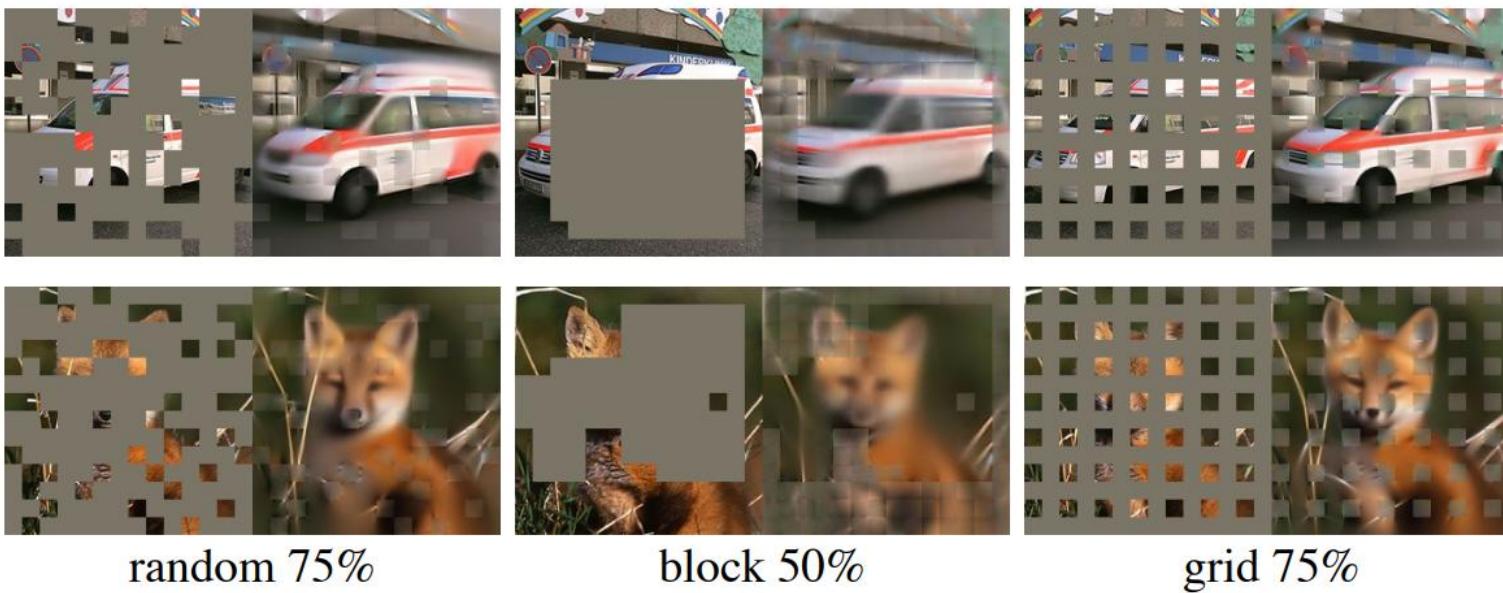
case	ft	lin
none	84.0	65.7
crop, fixed size	84.7	73.1
crop, rand size	84.9	73.5
crop + color jit	84.3	71.9

(e) **Data augmentation.** Our MAE works with minimal or no augmentation.

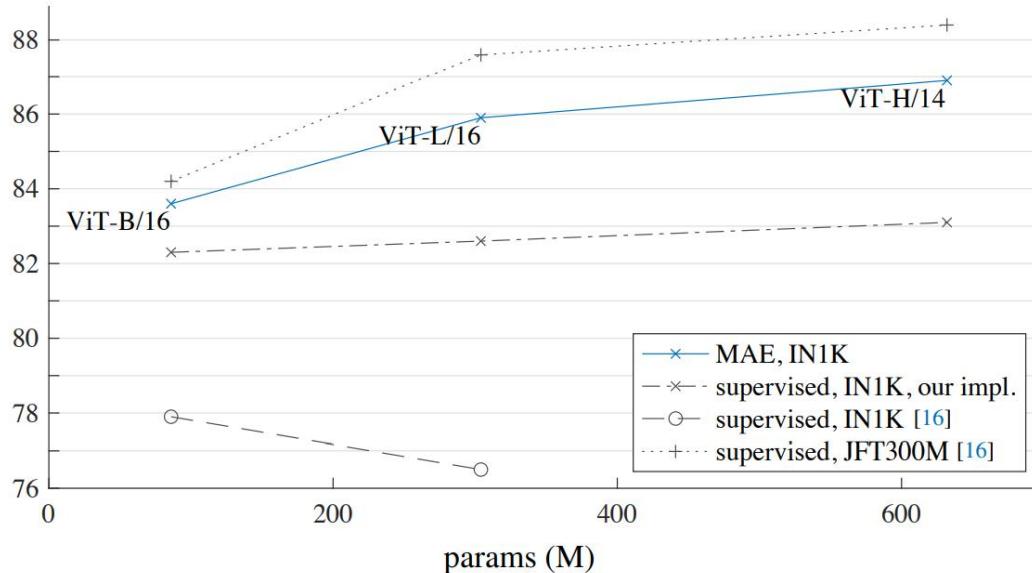
- 随机图像块掩蔽优于块状和网格状掩蔽

case	ratio	ft	lin
random	75	84.9	73.5
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0

(f) **Mask sampling.** Random sampling works the best. See Figure 6 for visualizations.



监督预训练 vs MAE 预训练



1N1K 是 ImageNet-1K，
包含约 128 万张图像

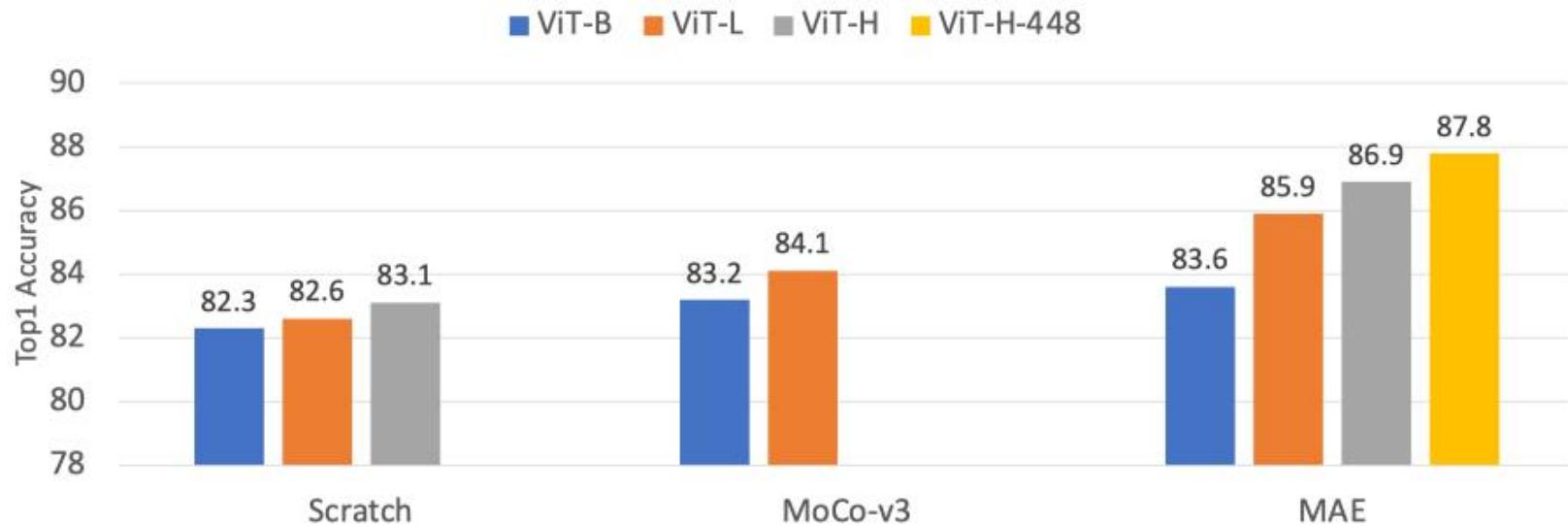
JFT300M 包含谷歌为内
部使用而开发的 3 亿张带
标签的图像

- ViT-L 和 ViT-H 非常大，当以监督方式直接在小型数据集 ImageNet-1K 上训练时，很容易导致过拟合
- 使用 MAE 预训练 ViT-large 可以有效缓解过拟合问题
- 在 ImageNet-1K 上使用 MAE 预训练的 ViT 表现出与在 JFT 300M 上使用监督预训练相似的性能

- MAE 预训练比从头开始在 ImageNet 上进行监督训练效果更好
- 在 ImageNet-1K 上进行的 MAE 预训练比在 JFT-300M 数据集上进行的监督预训练落后 4%
 - MAE 预训练中使用的图像数量远小于监督预训练中使用的图像数量 (~ 1M vs 300M)
 - MAE 预训练不需要使用任何标签信息，而监督预训练需要
 - 监督预训练优于 MAE 的性能也可能表明，在预训练阶段利用一些监督信息是有益的

- MAE 预训练 vs 其他无监督预训练，例如，MoCo 预训练
 - 在没有使用标签的 ImageNet-1K 训练数据集上进行预训练
 - 在具有标签的同一数据集上进行微调

下面显示了评估性能，其中 Scratch 表示直接从标签进行训练而不使用预训练



MAE 预训练表现最好，并且随着模型尺寸的增加，性能提升变得更大



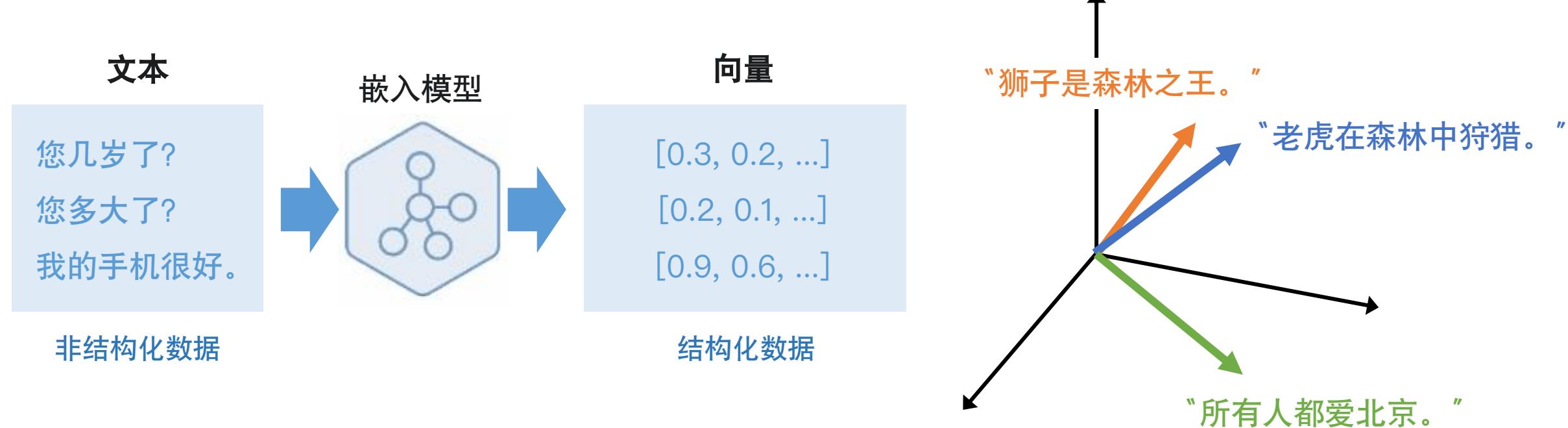
文本表示与理解

主讲人：苏勤亮



文本表示的目标

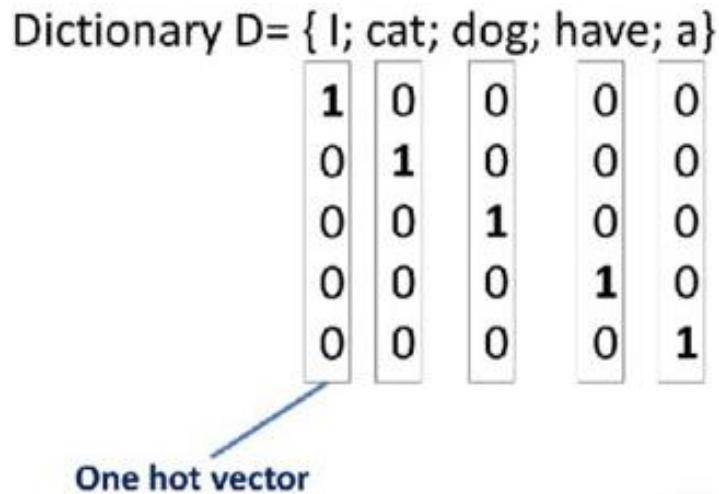
将文本映射为向量，向量间的某种距离能够反映语义相似性



大纲

- 基于词频的文本表示
- Word2Vec词嵌入
- 文本理解的预训练模型BERT
- 基于BERT 微调的文本理解

- 表示文本数据最简单的方式，是用一个 one-hot 向量来编码词汇表中的每个词，例如



- 之后，句子被表示为 one-hot 向量的连接

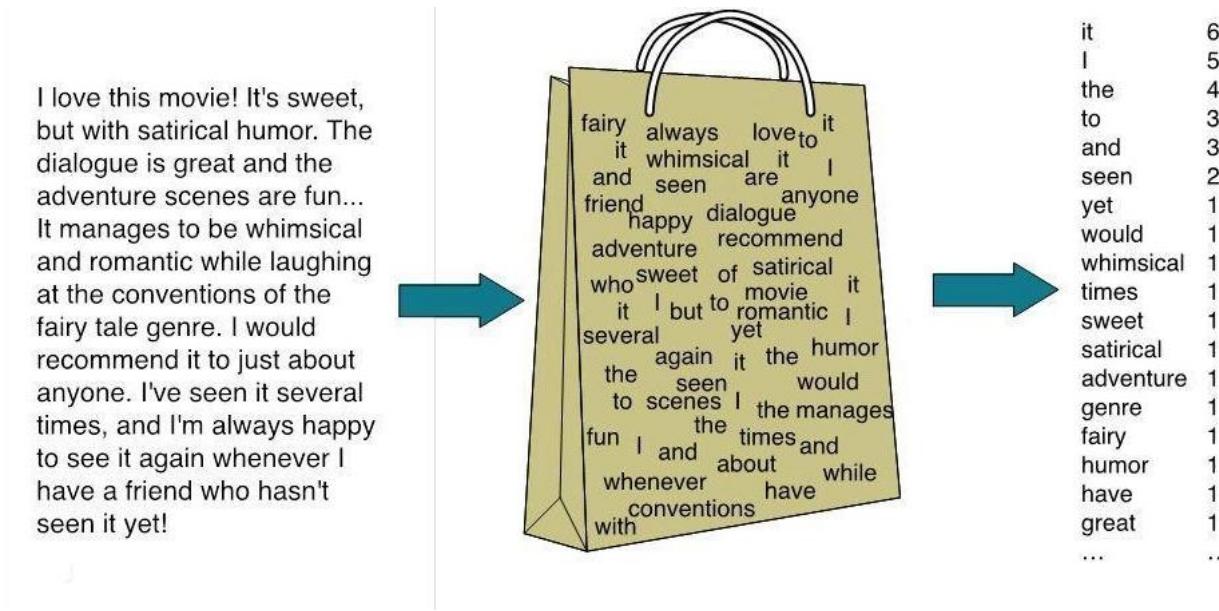
例如，“I have a dog” 可以表示为右边的矩阵

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

➤ 表示句子为一个矩阵是低效的，尤其是当句子很长的时候

词袋 (BOW)

词袋 (BOW) 丢弃了词语顺序的信息，只记录一个文档中每个词的计数



- 缺点：每个词都被认为具有同等的重要性。然而，不同词语所传达的信息量是不相等的

例如，像 'the', 'a' 这样的词语的信息量远不如像 'classroom', 'football' 等词语

TFIDF

- Term Frequency – Inverse Document Frequency (TFIDF)试图反映不同词语的不等重要性
具体来说，它是通过两个统计数据的乘积计算得出的，即：

$$tfidf(w, d, \mathcal{D}) = tf(w, d) \times idf(w, \mathcal{D})$$

- w 表示词语； d 表示第 d 个文档； \mathcal{D} 表示语料库（文档的集合）
- $tf(w, d)$ 表示词语 w 在文档 d 中的频率

$$tf(w, d) = \frac{\# \text{ of word } w \text{ in } d}{\# \text{ of all words in } d}$$

- $idf(w, \mathcal{D})$ 是语料库 \mathcal{D} 中包含词语 w 的文档比例的对数

$$idf(w, \mathcal{D}) = \log \frac{\# \text{ of documents in } \mathcal{D}}{\# \text{ of documents containing word } w \text{ in } \mathcal{D}}$$

- $idf(w, D)$ 衡量了词语 w 包含多少信息。 $idf(w, D)$ 的值越大，词语的信息量就越大
- 因此，TFIDF 特征 $tfidf(w, d, D) = tf(w, d) \times idf(w, D)$ 同时考虑了词语在文档中的频率和其在语料库中的信息量

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

		blue	bright	can	see	shining	sky	sun	today
Document 1	1	0.301	0	0	0	0	0.151	0	0
Document 2	2	0	0.0417	0	0	0	0	0.0417	0.201
Document 3	3	0	0.0417	0	0	0	0.100	0.0417	0
Document 4	4	0	0.0209	0.100	0.100	0.100	0	0.0417	0

每一行代表一个文档的 TFIDF 特征

局限性

- One-hot 词语表示
 - 无法反映词语的语义相似性，例如 'football' 和 'soccer'
 - 当表示一个文档时，过于低效
 - 维度非常高，高达 20000
 - 非常稀疏
- BOW 和 TFIDF 文档表示
 - 无法反映词语的语义相似性
 - 不包含任何词语顺序信息
 - 维度非常高
 - 非常稀疏



但比 one-hot 表示好得多

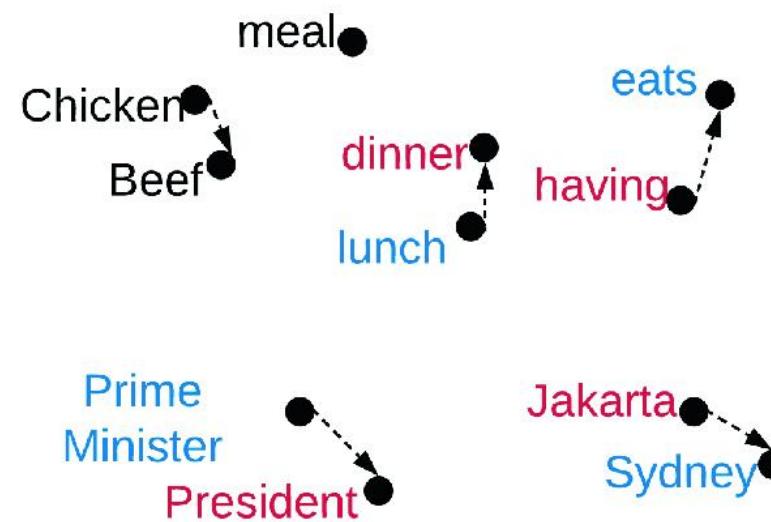
大纲

- 基于词频的文本表示
- Word2Vec词嵌入
- 文本理解的预训练模型BERT
- 基于BERT 微调的文本理解

Word2Vec 词嵌入表示

- 目标：用具有以下特征的实值向量表示单词

- 1) 稠密 (低维度)
- 2) 反映单词之间的语义相似性



- 句子嵌入可以随后从单词嵌入中构建

如何学习 Word2Vec 嵌入？

- 观察：单词的语义相似性隐含地反映在现有句子中
例如，如果两个词经常同时出现，它们可能会保留相似的含义

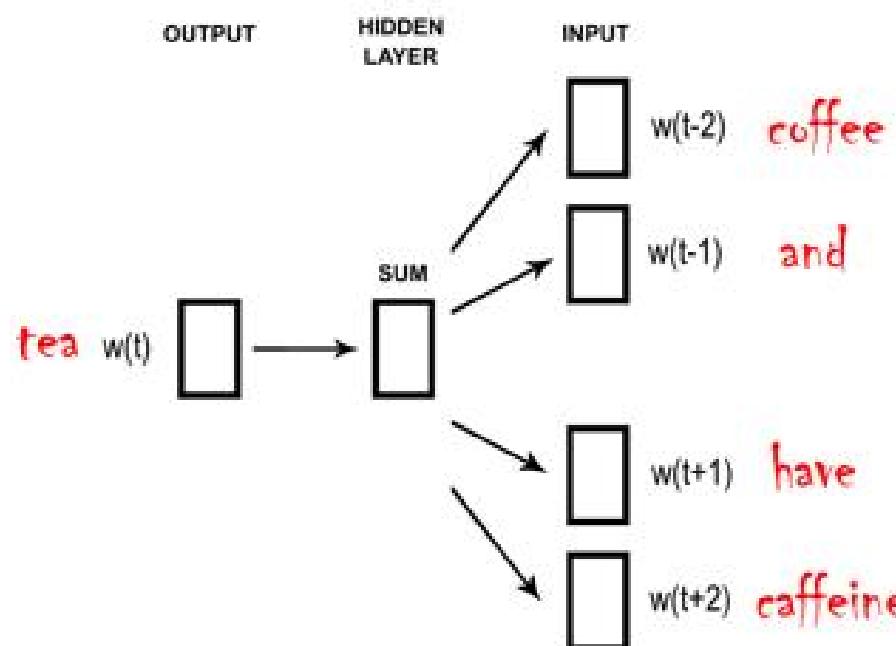
A cup of **tea**
A cup of **coffee**
Tea or coffee?
Coffee and **tea** have caffeine
Let's go for a **coffee**
Let's get a **tea**
Coffee vs Tea: Which is Best?
I avoid adding sugar to my **tea**
I drink **coffee** with two spoons of sugar



- 因此，可以从大量现有的句子中学习保留语义的嵌入

Skip-gram

- 两个基本方法
 - 1) Skip-grams
 - 2) 连续词袋 (CBOW)
- Skip-gram: 使用中心词预测上下文



- 首先，用一个随机向量 $u \in \mathbb{R}^m$ 初始化每个词的嵌入
- 然后，使用第 t 个词语 w_t 来预测其左侧和右侧的词语。目标函数是

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

The quick brown fox jumps over the lazy dog. \rightarrow (the, quick)
 (the, brown)

The quick brown fox jumps over the lazy dog. \rightarrow (quick, the)
 (quick, brown)
 (quick, fox)

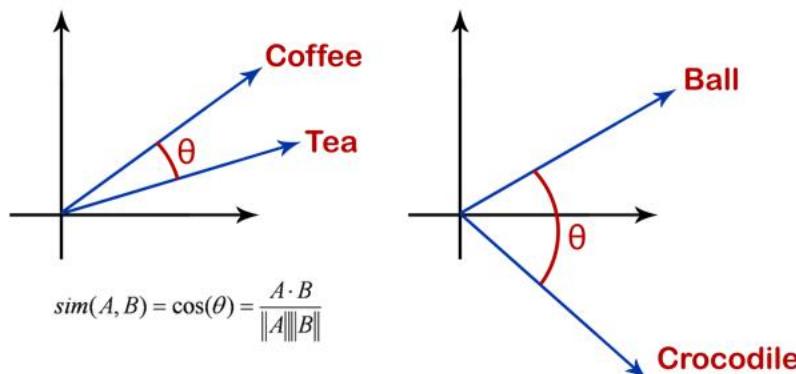
The quick brown fox jumps over the lazy dog. \rightarrow (brown, the)
 (brown, quick)
 (brown, fox)
 (brown, jumps)

The quick brown fox jumps over the lazy dog. \rightarrow (fox, quick)
 (fox, brown)
 (fox, jumps)
 (fox, over)

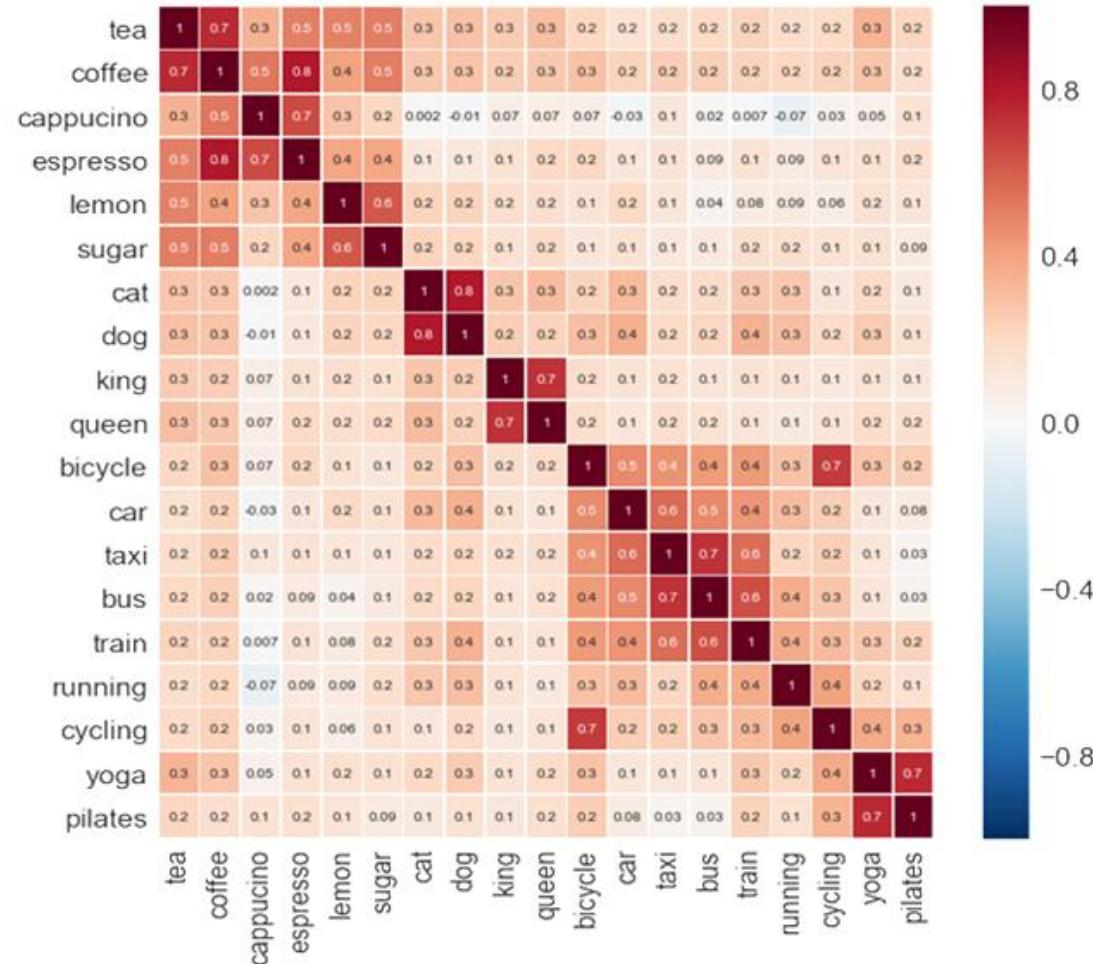
- 预测概率 $P(w_{t+j}|w_t; \theta)$ 建模为

$$P(w_{t+j}|w_t; \theta) = \frac{\exp(u_{w_t}^T u_{w_{t+j}})}{\sum_{w \in \mathcal{V}} \exp(u_{w_t}^T u_w)}$$

- \mathcal{V} 是词汇表中的词语集合
- u_w 表示词语 w 的嵌入，需要被优化
- 在一个大型语料库上训练之后，单词的语义相似性反映在它们嵌入的余弦相似度中

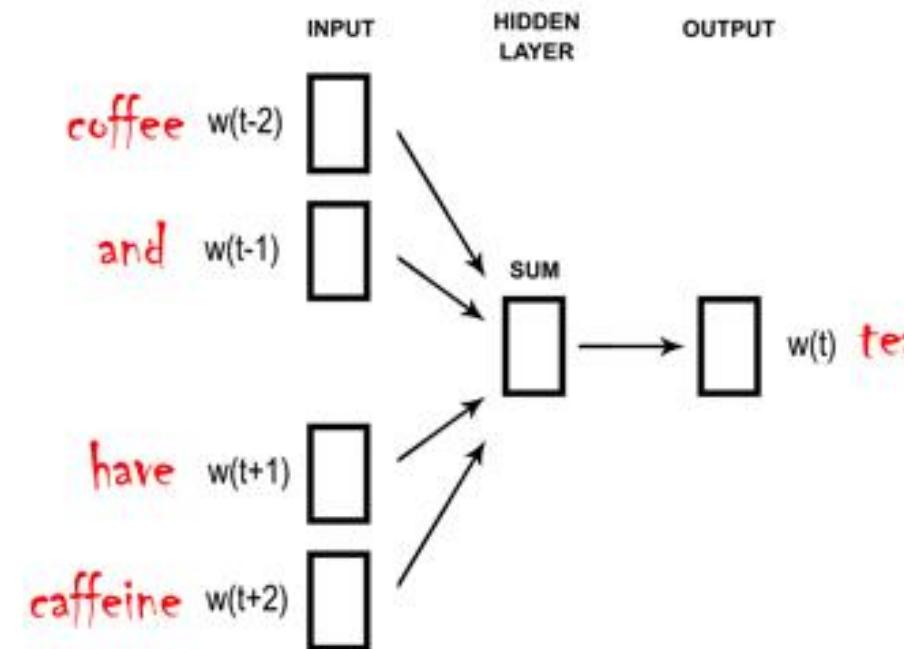


- 在大语料库上训练的词嵌入上评估的余弦相似性示例



连续词袋模型 (Continuous Bag-of-Words)

- 与 skip-gram 不同，CBOW 使用左侧和右侧的词来预测中心词



句子嵌入

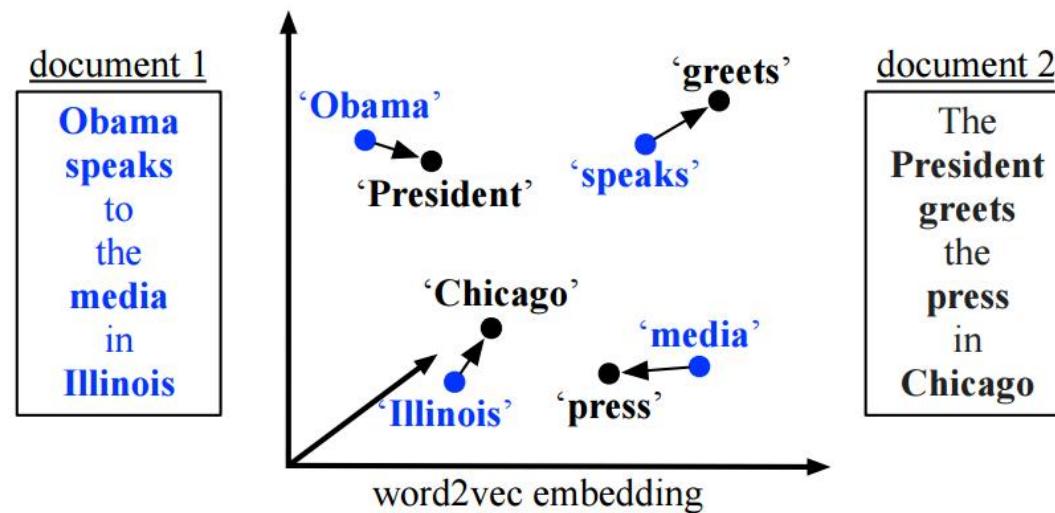
- 句子嵌入可以通过多种不同的方式从单词嵌入中获得

1) 平均

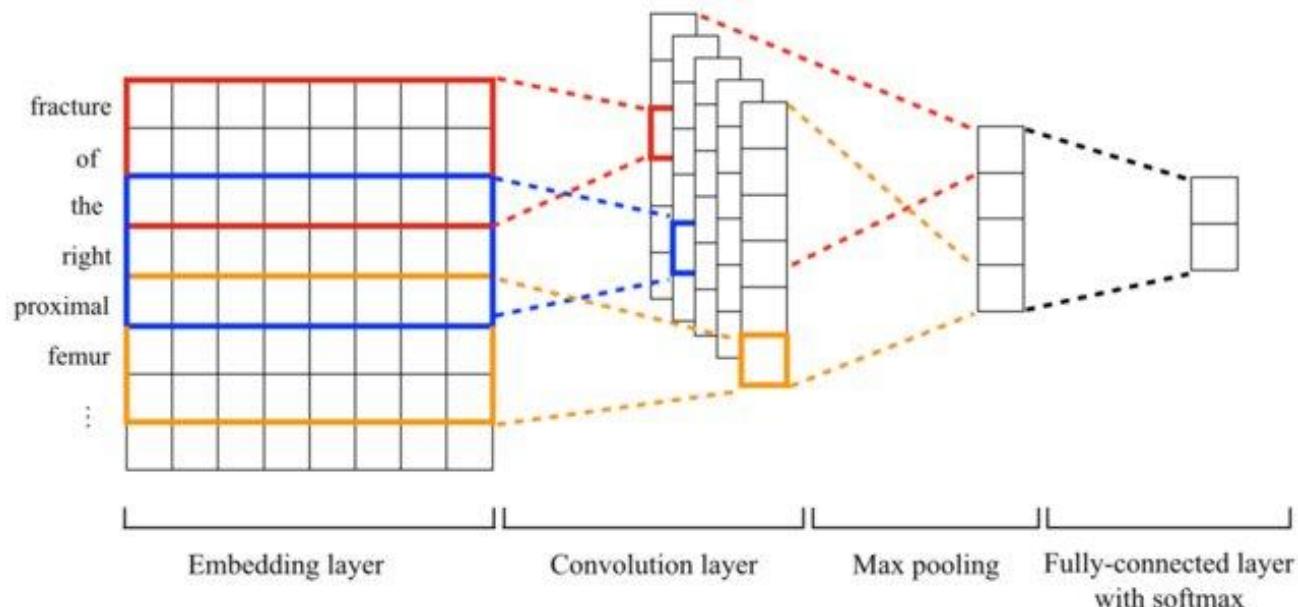
Sentence	Embeddings
coffee	0.2 0.4 0.32 0.89 ... 0.77 0.12 0.11 0.99
or	0.54 0.8 0.35 0.34 ... 0.56 0.22 0.56 0.43
tea	0.23 0.39 0.55 0.91 ... 0.6 0.2 0.61 0.8
?	0.7 0.45 0.56 0.43 ... 0.22 0.16 0.33 0.5
Average	0.42 0.51 0.45 0.64 ... 0.54 0.17 0.4 0.68

2) 拼接

Sentence	Embeddings concatenated																																				
coffee or tea ?	<table border="1"><tr><td>0.2</td><td>0.4</td><td>0.32</td><td>0.89</td><td>...</td><td>0.77</td><td>0.12</td><td>0.11</td><td>0.99</td></tr><tr><td>0.54</td><td>0.8</td><td>0.35</td><td>0.34</td><td>...</td><td>0.56</td><td>0.22</td><td>0.56</td><td>0.43</td></tr><tr><td>0.23</td><td>0.39</td><td>0.55</td><td>0.91</td><td>...</td><td>0.6</td><td>0.2</td><td>0.61</td><td>0.8</td></tr><tr><td>0.7</td><td>0.45</td><td>0.56</td><td>0.43</td><td>...</td><td>0.22</td><td>0.16</td><td>0.33</td><td>0.5</td></tr></table> <p>#Tokens</p> <p>Dimension of word embeddings</p>	0.2	0.4	0.32	0.89	...	0.77	0.12	0.11	0.99	0.54	0.8	0.35	0.34	...	0.56	0.22	0.56	0.43	0.23	0.39	0.55	0.91	...	0.6	0.2	0.61	0.8	0.7	0.45	0.56	0.43	...	0.22	0.16	0.33	0.5
0.2	0.4	0.32	0.89	...	0.77	0.12	0.11	0.99																													
0.54	0.8	0.35	0.34	...	0.56	0.22	0.56	0.43																													
0.23	0.39	0.55	0.91	...	0.6	0.2	0.61	0.8																													
0.7	0.45	0.56	0.43	...	0.22	0.16	0.33	0.5																													

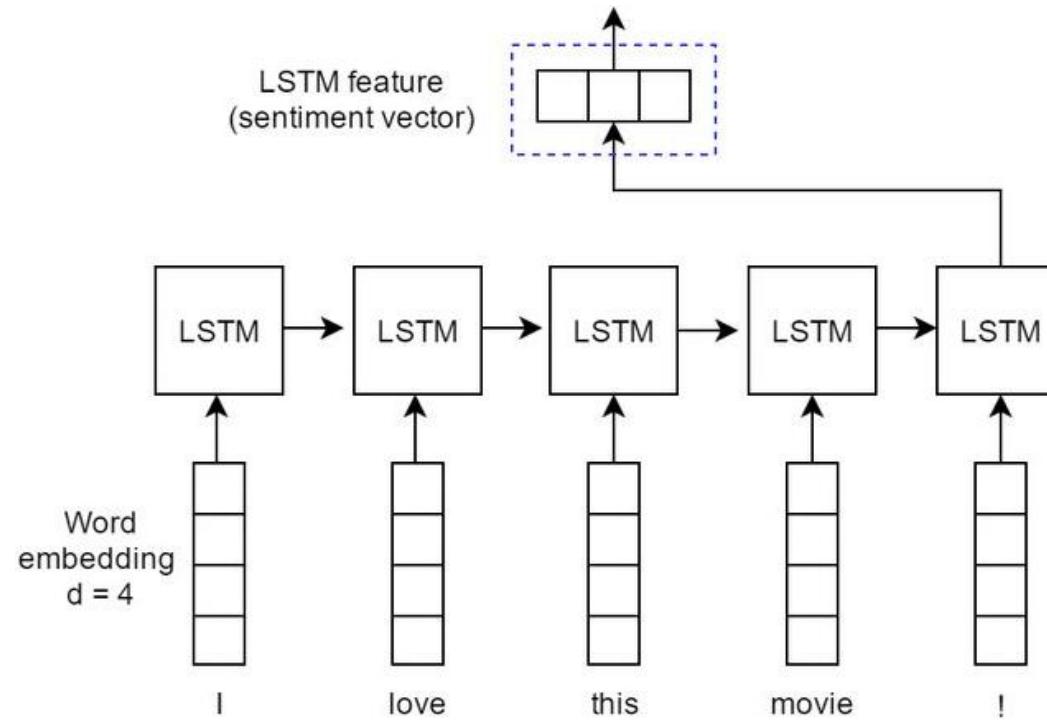


3) 使用 CNN 提取句子嵌入(文本 CNN)



CNN 的参数是使用下游任务优化的

4) 使用 RNN 提取句子嵌入



RNN 的参数是使用下游任务优化的

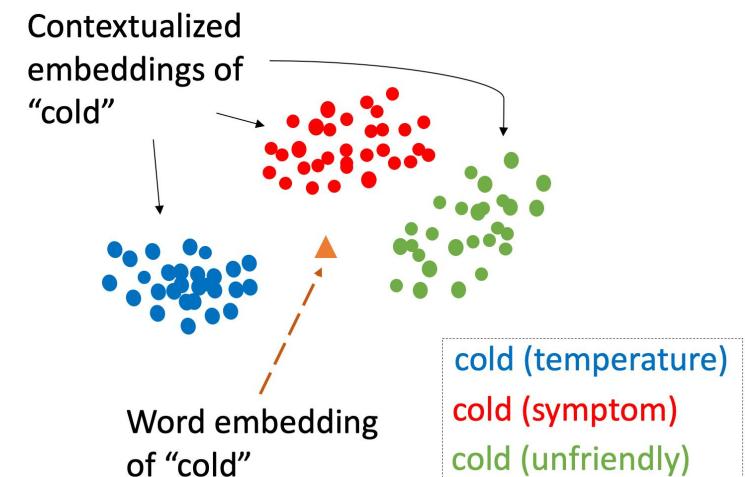
语境感知的词嵌入

- Word2Vec 嵌入的问题
 - Word2Vec 为每个词分配一个固定的嵌入
 - 但是，单词在不同的语境下通常表现出不同的含义，例如：

1) open a *bank* account

2) on the river *bank*

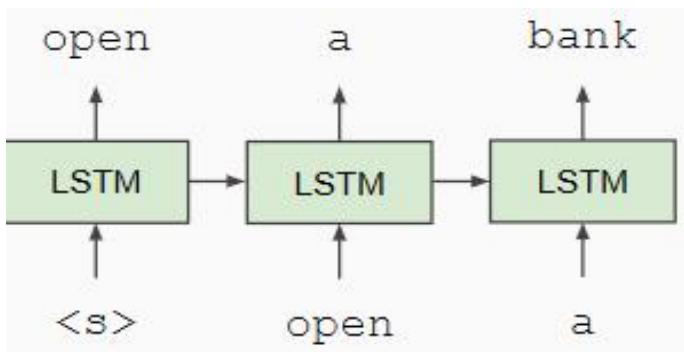
- 单词的嵌入应该根据其所在的上下文不同而变化



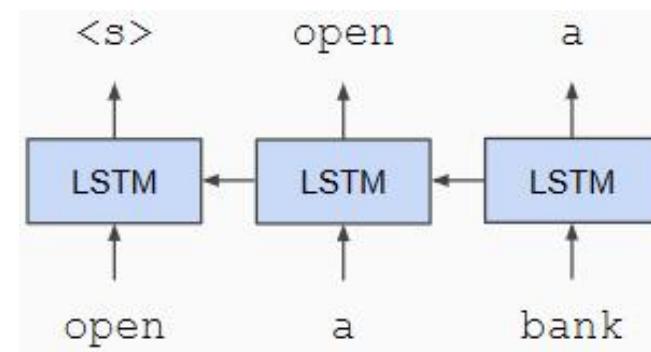
- ELMo

1) 首先，在一个大型语料库上训练一个双向 RNN

2) 然后，将感兴趣的句子通过预训练的 RNN

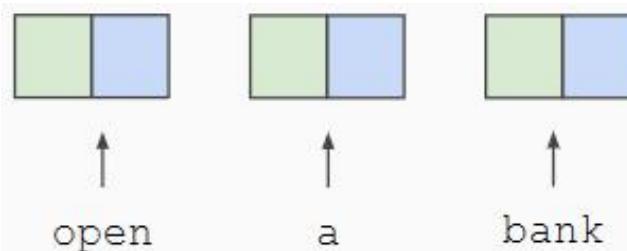


从左到右



从右到左

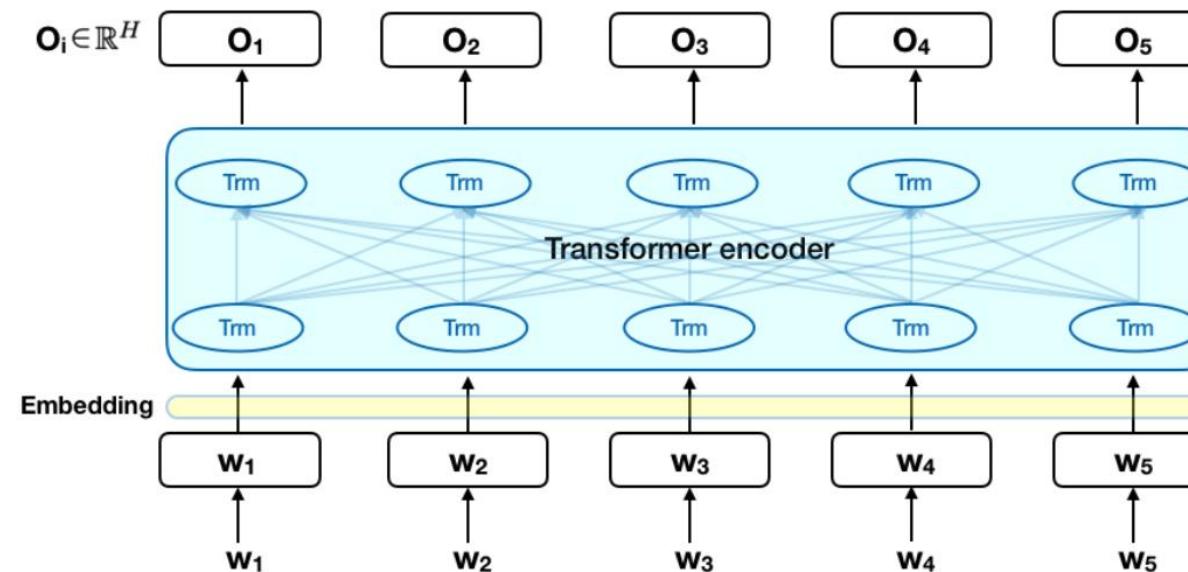
3) 最终的单词嵌入是作为预训练 RNN 的隐藏状态的连接获得的



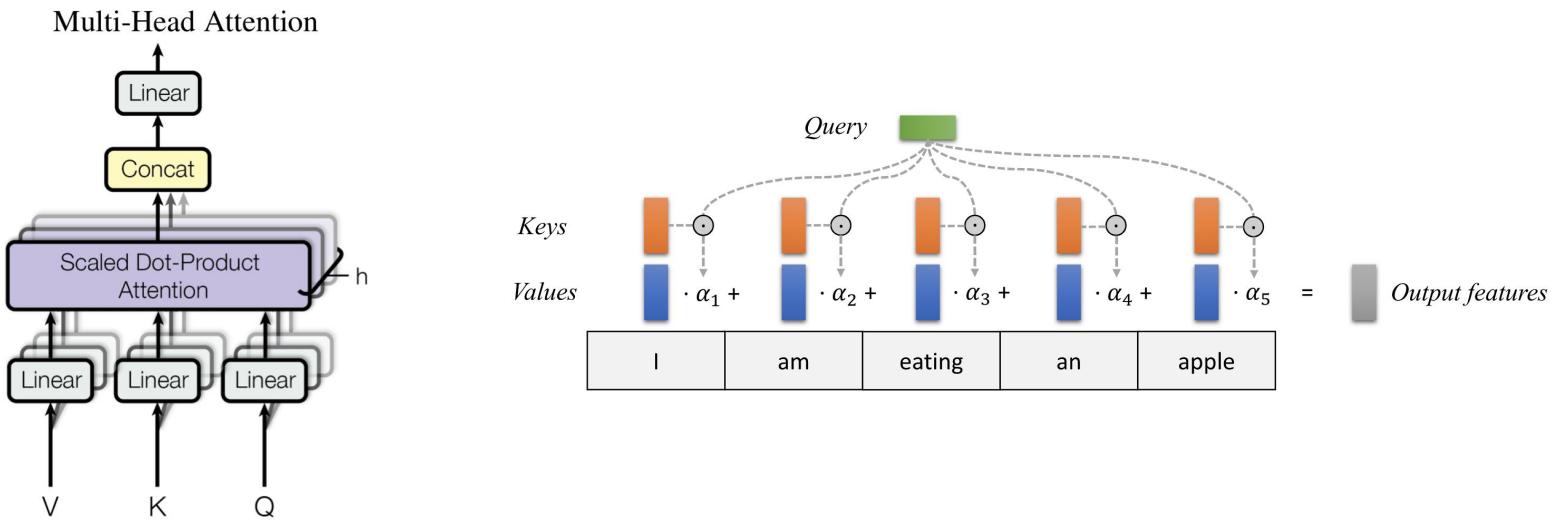
大纲

- 基于词频的文本表示
- Word2Vec词嵌入
- 文本理解的预训练模型BERT
- 基于BERT 微调的文本理解

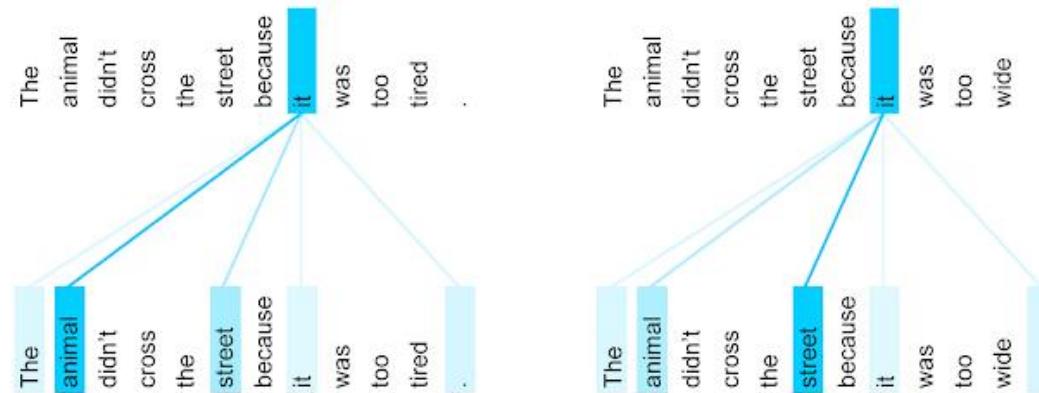
- ELMo 嵌入的问题
 - 1) RNN 难以捕捉文档中词语的**长期依赖关系**
 - 2) RNN 只允许顺序执行，这使得它难以利用 GPU 中的**并行计算资源**
- Bidirectional Encoder Representations from Transformers (BERT) 没有使用顺序结构，而是采用**基于 Transformer 的“全连接”结构**



- 与 MLP 中的全连接不同，这里的全连接结构建立在一个名为“transformer”的模块上



➤ 例子：注意单词“it”的相关单词的变化



预训练任务

- 该模型在一个非常大的语料库上进行预训练
- 1) 预训练任务 1：掩码语言模型 (Masked Language Model)
 - 对于每个输入序列，随机选择 15% 的 token
 - 用 [MASK] 标记替换 80% 的选定 token

went to the store → went to the [MASK]
 - 用一个随机词替换 10% 的选定 token

went to the store → went to the running
 - 保持剩余 10% 的选定 token 不变

went to the store → went to the store

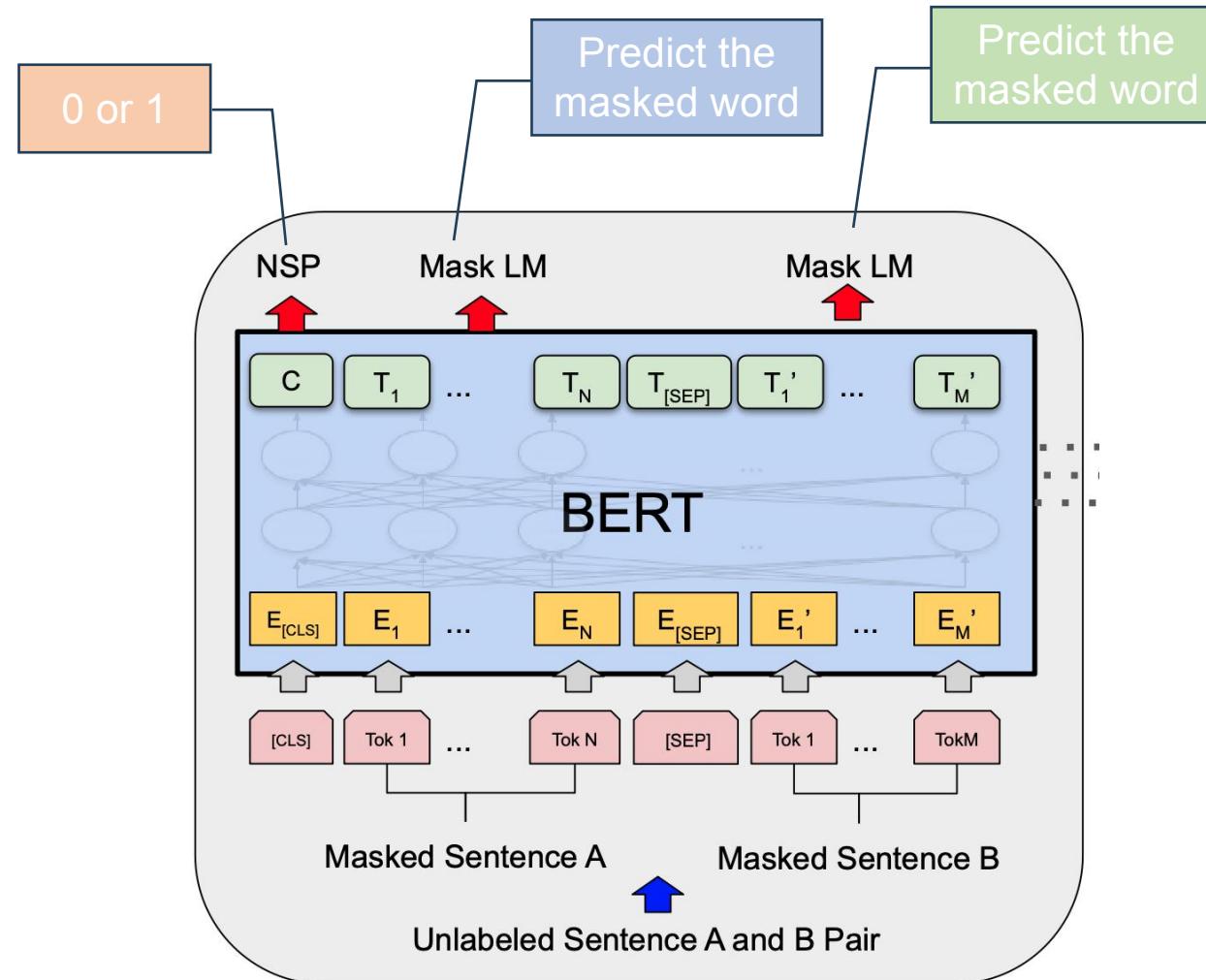
- 目标：训练 BERT 模型来预测掩码 token 位置的真实 token

2) 预训练任务 2: 下一句预测 (Next Sentence Prediction, NSP)

预测句子 B 是否是句子 A 的下一句

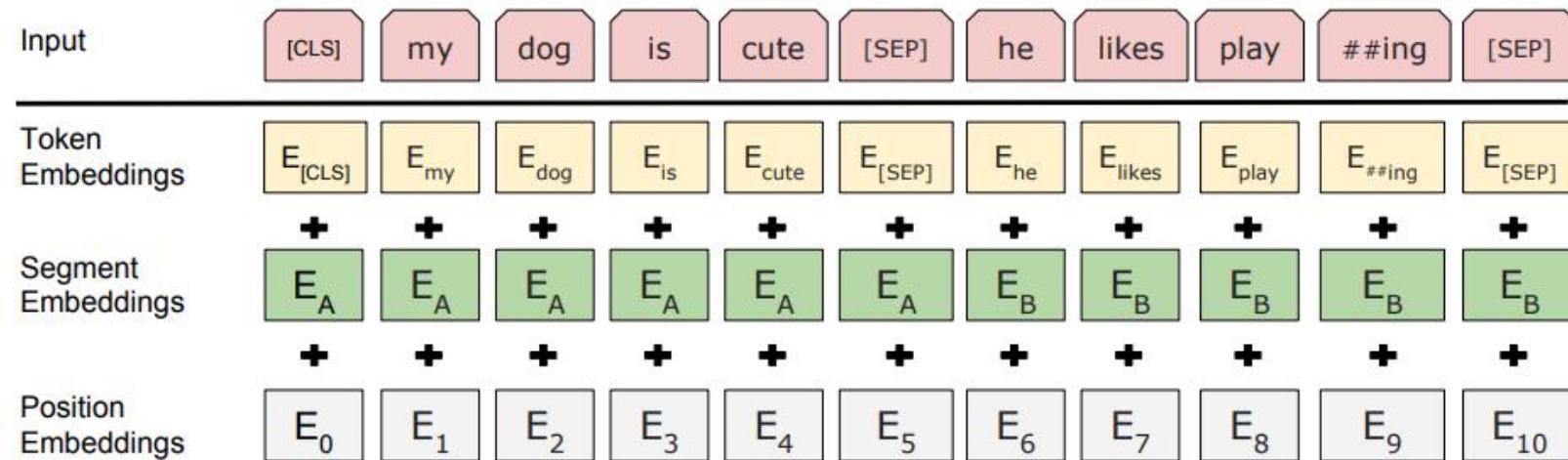
- 50% 的机会让句子 B 成为句子 A 的真实下一句

- 将预训练任务放在一起



- BERT 模型的输入

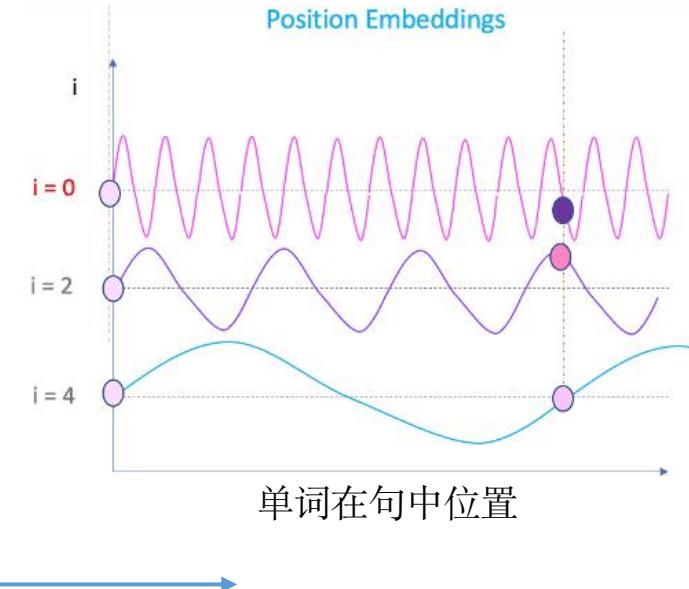
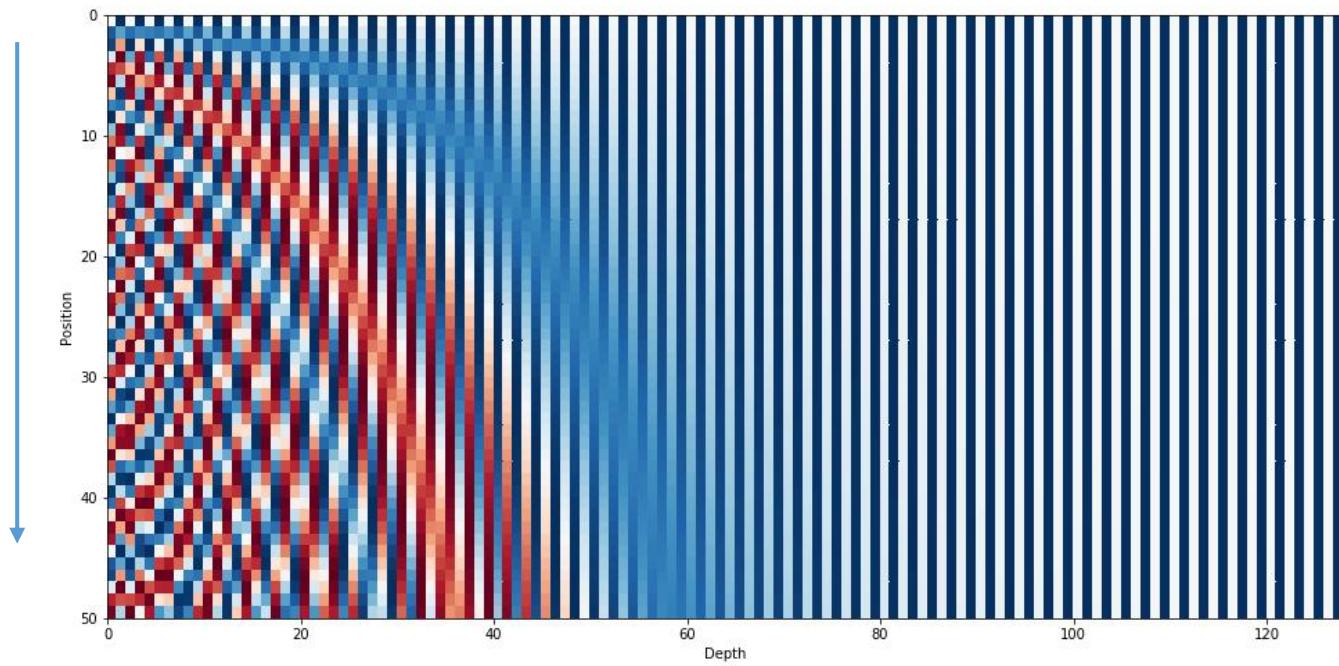
将三个 token 嵌入相加作为模型的最终嵌入输入



- 位置嵌入

$$PE(pos, i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

单词位置

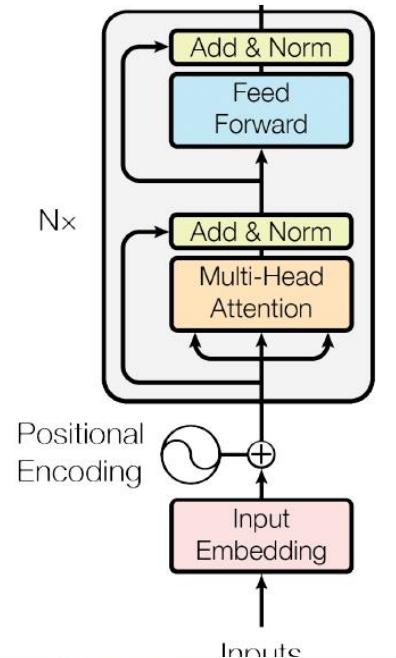


- 训练细节

- 训练语料库: Wikipedia (2.5B) + BooksCorpus (0.8B)
- 训练序列长度: 512 个单词片段
- Batch size: 128K
- 训练了一百万步

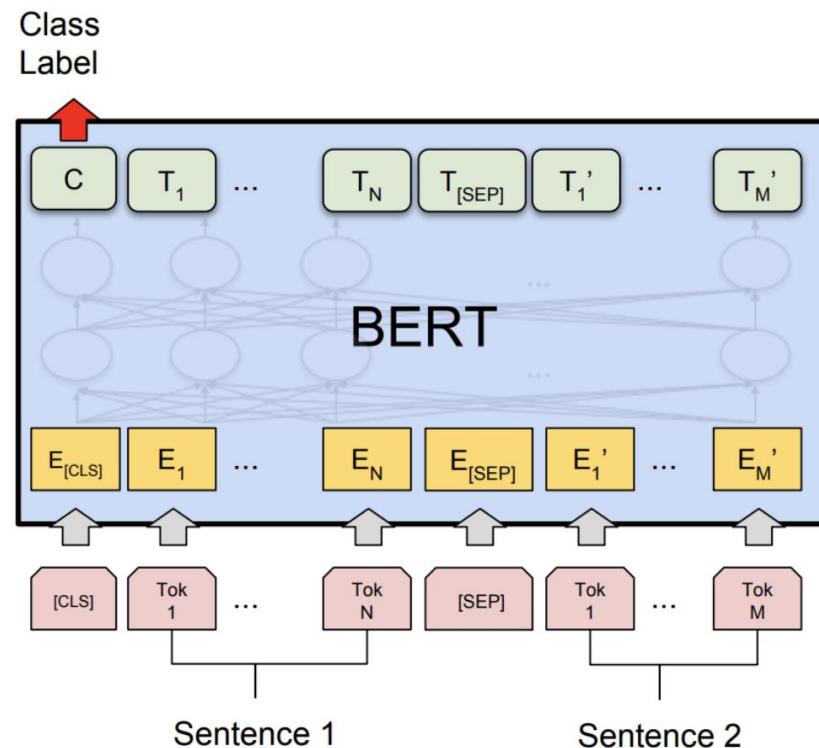
- BERT 模型的大小

- **BERT-base**: 12 层，768 隐藏层大小，12 个注意力头，1.1 亿参数
- **BERT-large**: 24 层，1024 隐藏层大小，16 个注意力头，3.4 亿参数



Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12,288	96	175B	694GB	?
Gopher	80	16,384	128	280B	10.55 TB	4096x TPUv3 (38 days)

- 为了获得单词和句子嵌入，将感兴趣的句子传递给预训练的 BERT 模型



- BERT 的输出是单词和句子的语境化嵌入

大纲

- 基于词频的文本表示
- Word2Vec词嵌入
- 文本理解的预训练模型BERT
- 基于BERT 微调的文本理解

GLUE: 通用语言理解评估基准，包括 6 个句子对和 2 个单句级别任务

- 句子级别任务例子

MNLI

Premise: A soccer game with multiple males playing

Hypothesis: Some men are playing a sport



(entailment, contradiction, neutral)

QQP

Q1: Where can I learn to invest in stocks?

Q2: How can I learn more about stocks?



(duplicate, not duplicate)

SST2

Rich veins of funny stuff in this movie



(positive, negative)

- Token 级别任务

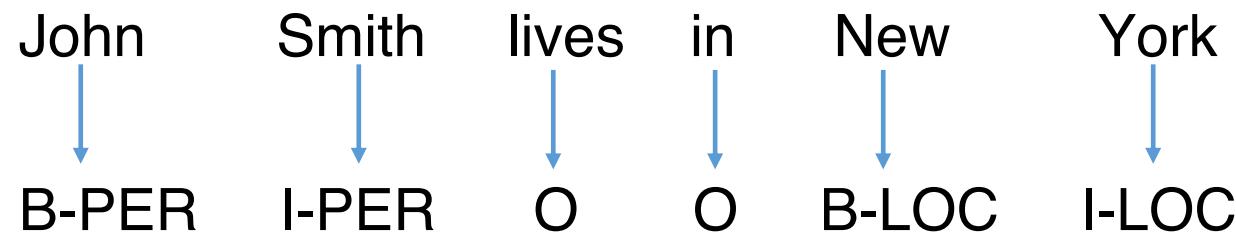
- 问题回答(SQuAD)

Question: The New York Giants and the New York Jets play at which stadium in NYC ?

Context: The city is represented in the National Football League by the New York Giants and the New York Jets , although both teams play their home games at MetLife Stadium in nearby East Rutherford , New Jersey , which hosted Super Bowl XLVIII in 2014 .

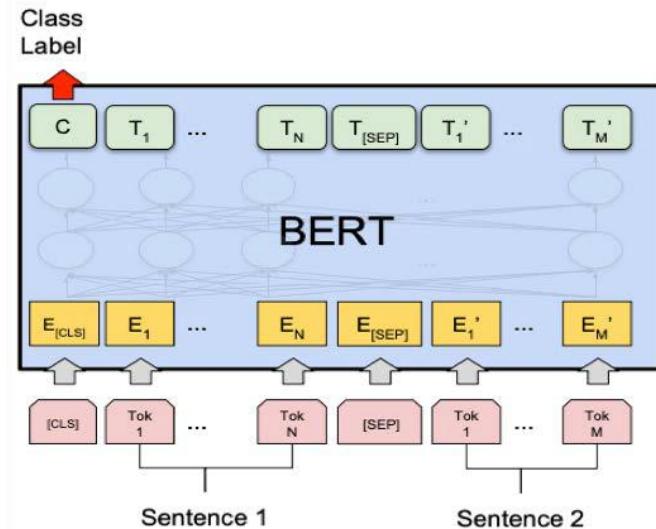
(Training example 29,883)

- 命名实体识别

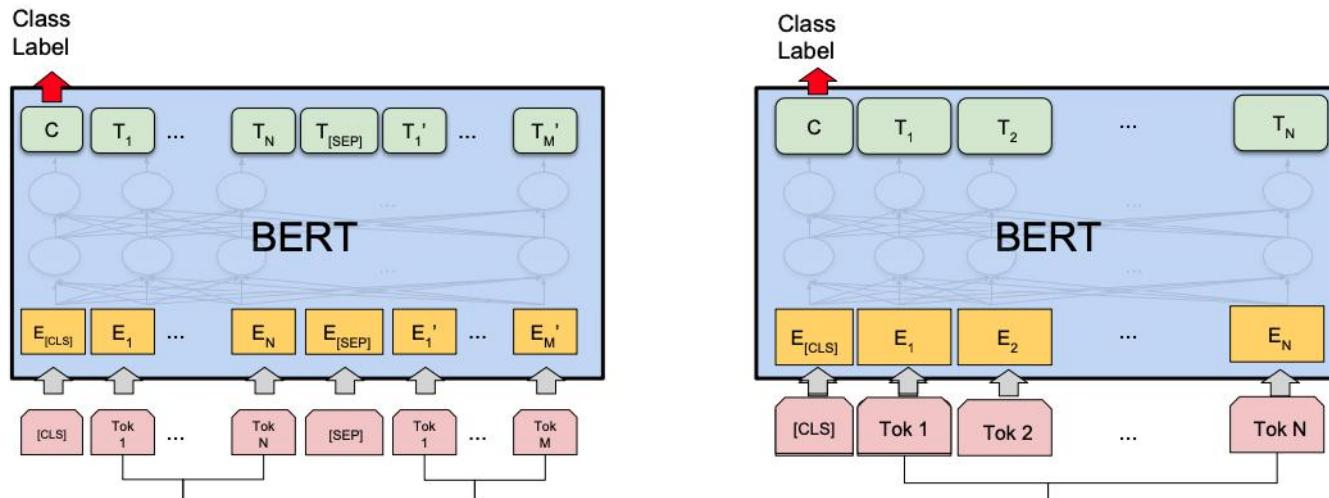


- 微调 BERT——对于句子对任务，使用 [SEP] 来分隔带有分段嵌入的两个段落

➤ 在 [CLS] 表示之上添加一个线性分类器



➤ 对于 token 级别预测任务，在输出层的token表示之上添加独立的线性分类器



• GLUE实验结果

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

• 消融研究

➤ 更大的模型通常会带来更好的性能

# layers	hidden size	# of heads	Hyperparams		Dev Set Accuracy		
			#L	#H	#A	LM (ppl)	MNLI-m
3	768	12	5.84			77.9	79.8
6	768	3	5.24			80.6	82.2
6	768	12	4.68			81.9	84.8
12	768	12	3.99			84.4	86.7
12	1024	16	3.54			85.7	86.9
24	1024	16	3.23			86.6	87.8
							93.7

课堂小结

- Bag of Words, TFIDF——基于词频的文本表示
- 词嵌入——基于语义的文本表示
- BERT——文本理解的基本模型
- 基于BERT微调范式的各种下游文本理解



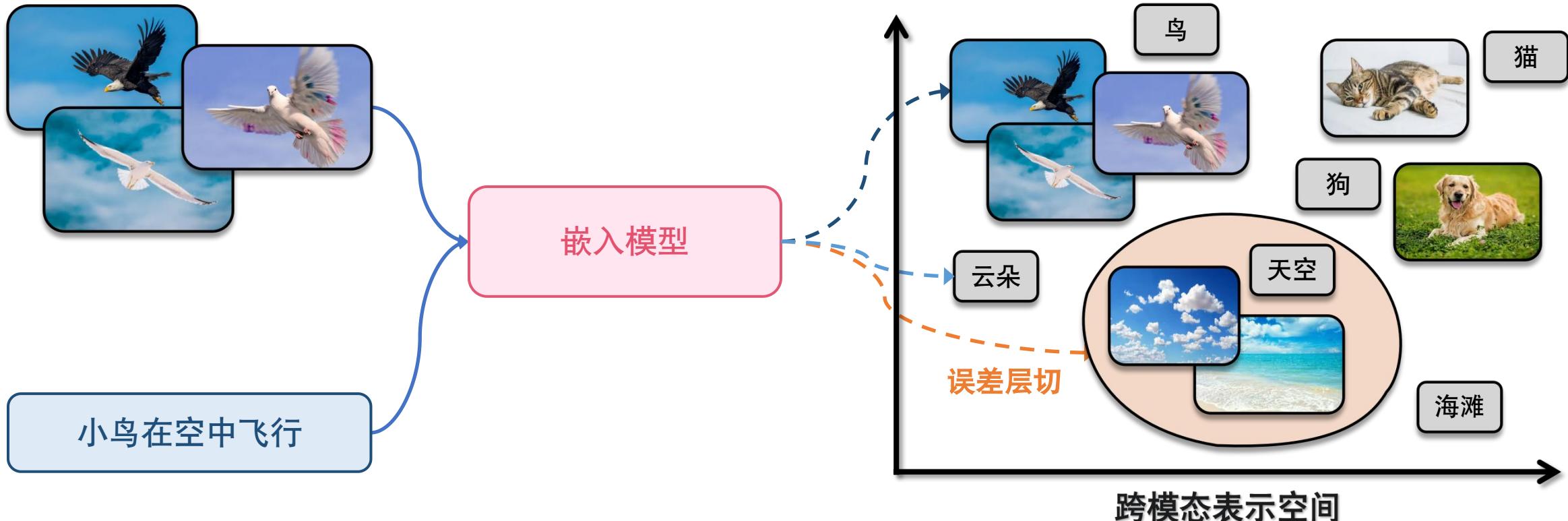
机器学习与数据挖掘

多模态感知与理解



跨模态表示学习的目标

学习共享的表示空间，对齐不同模态的语义信息

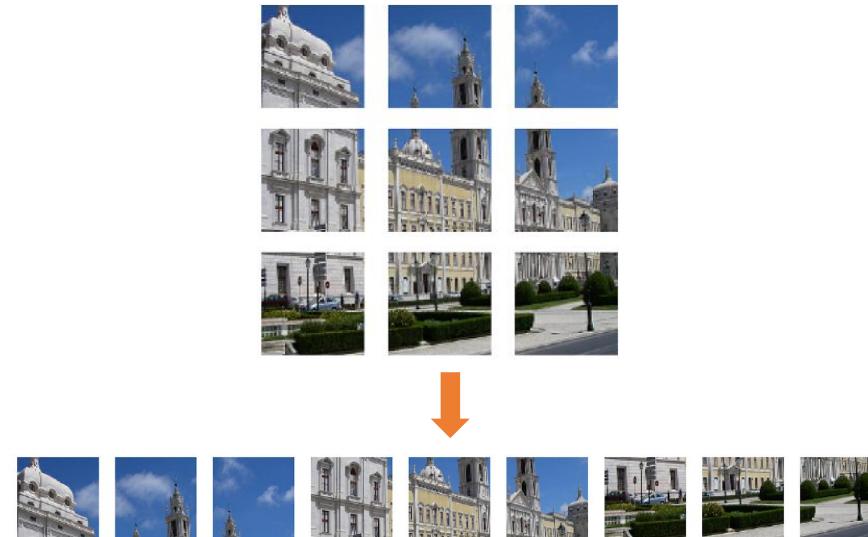


大纲

- 基于Transformer的视觉处理架构ViT
- CLIP
- BLIP

视觉与文本统一网络架构

- 思路：利用BERT类似的模型来处理视觉数据
 - 通过将图像分割成 N 个图像块的序列来创建“词”的概念



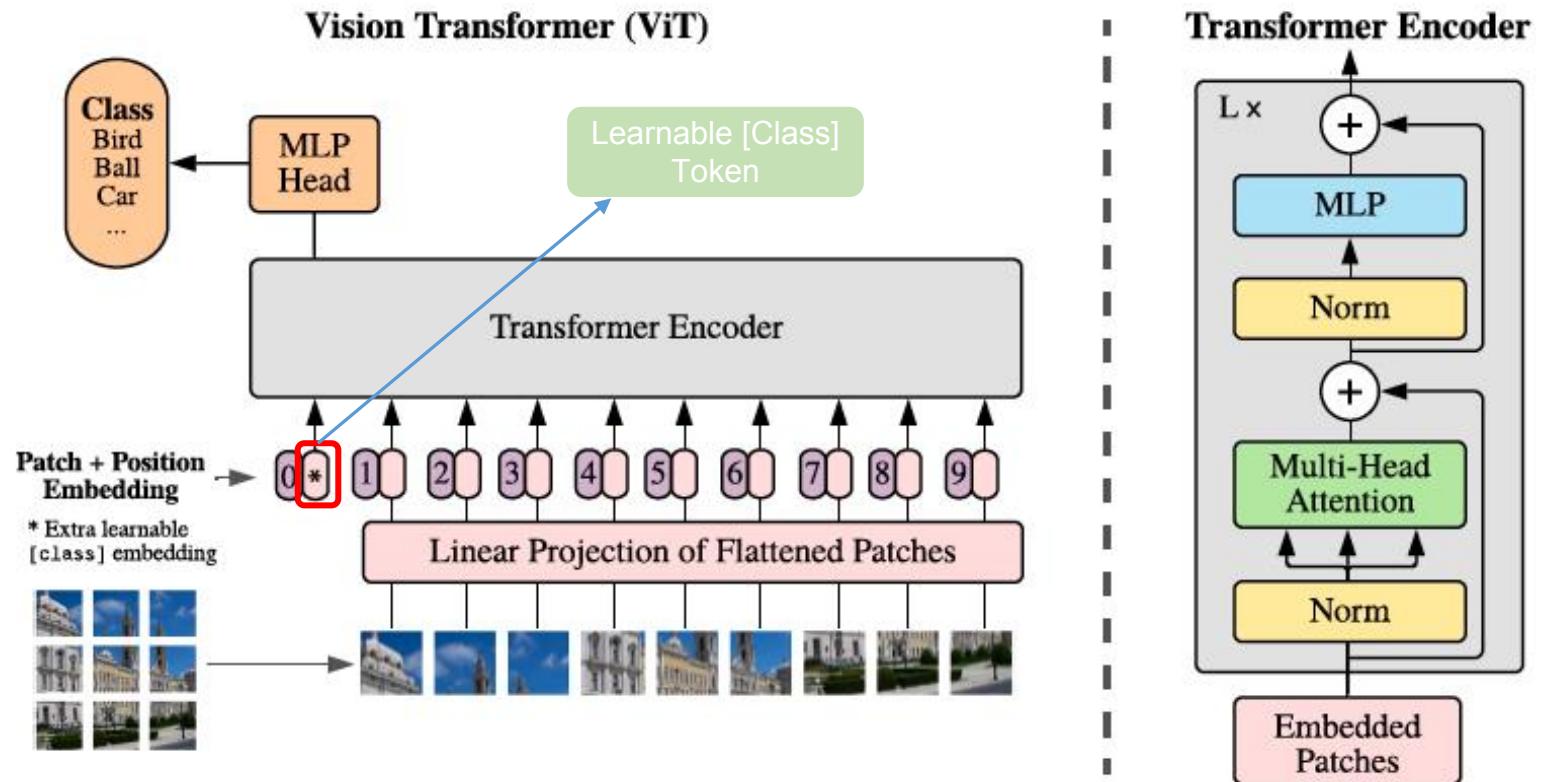
- 将压平的图像块进行线性投影 $\mathbf{x}_p^i \in \mathbb{R}^{16^2 \cdot 3}$ for $i = 1, 2, \dots N$

$$\mathbf{z}_0^i = \mathbf{x}_p^i \mathbf{E},$$

其中 $\mathbf{E} \in \mathbb{R}^{16^2 \cdot 3 \times D}$ 是线性嵌入矩阵

Alexey Dosovitskiy et. al, An Image is Worth 16 x 16 Words: Transformers for Image Reconstruction at Scale, ICLR 2021

- 引入一个可学习的 [class] 标记，表示为 z_0^0 ，类似于 BERT 的 [CLS] 标记
- 为位置 $0, 1, 2, \dots, N$ 引入可学习的一维位置嵌入，可以表示为 $E_{pos} \in \mathbb{R}^{(N+1) \times D}$



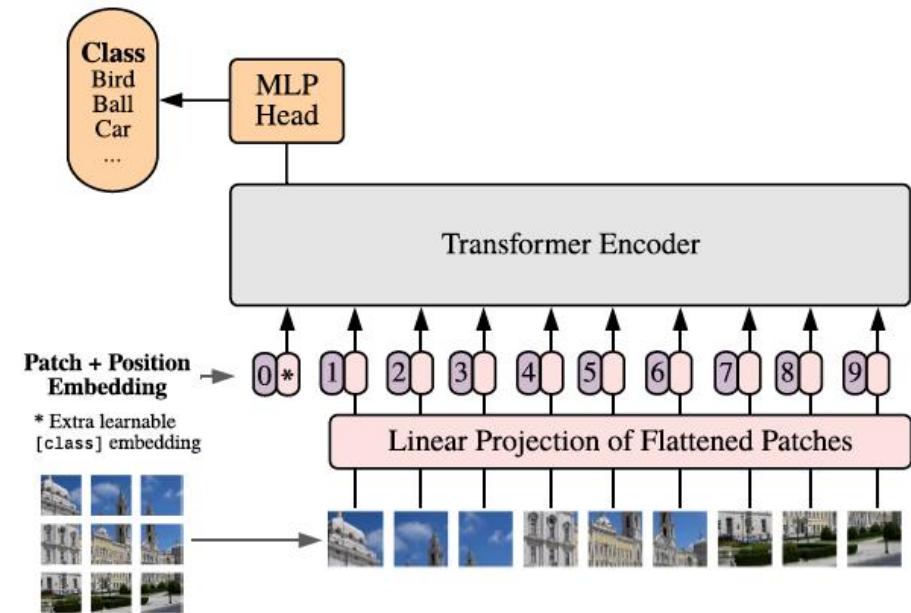
➤ 更新规则

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1 \mathbf{E}; \mathbf{x}_p^2 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{pos}$$

$$\mathbf{z}'_\ell = MSA(LN(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}$$

$$\mathbf{z}_\ell = MLP(LN(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell$$

其中 $MSA(\cdot)$ 表示多头注意力



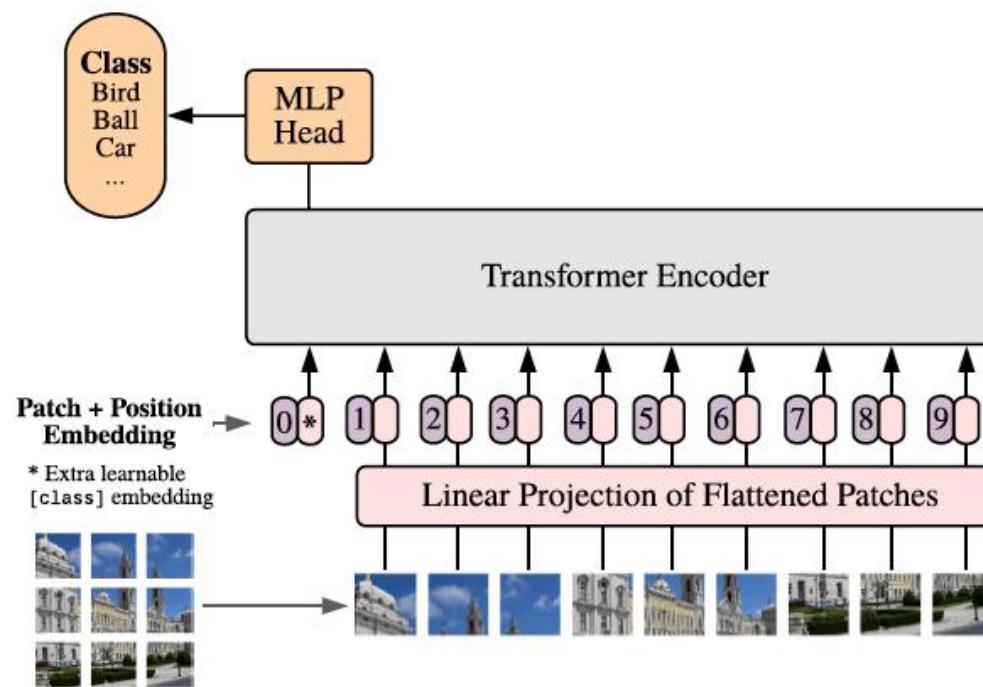
- ViT模型不同版本（Base, Large, Huge）的模型参数信息

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

ViT的预训练——监督方式

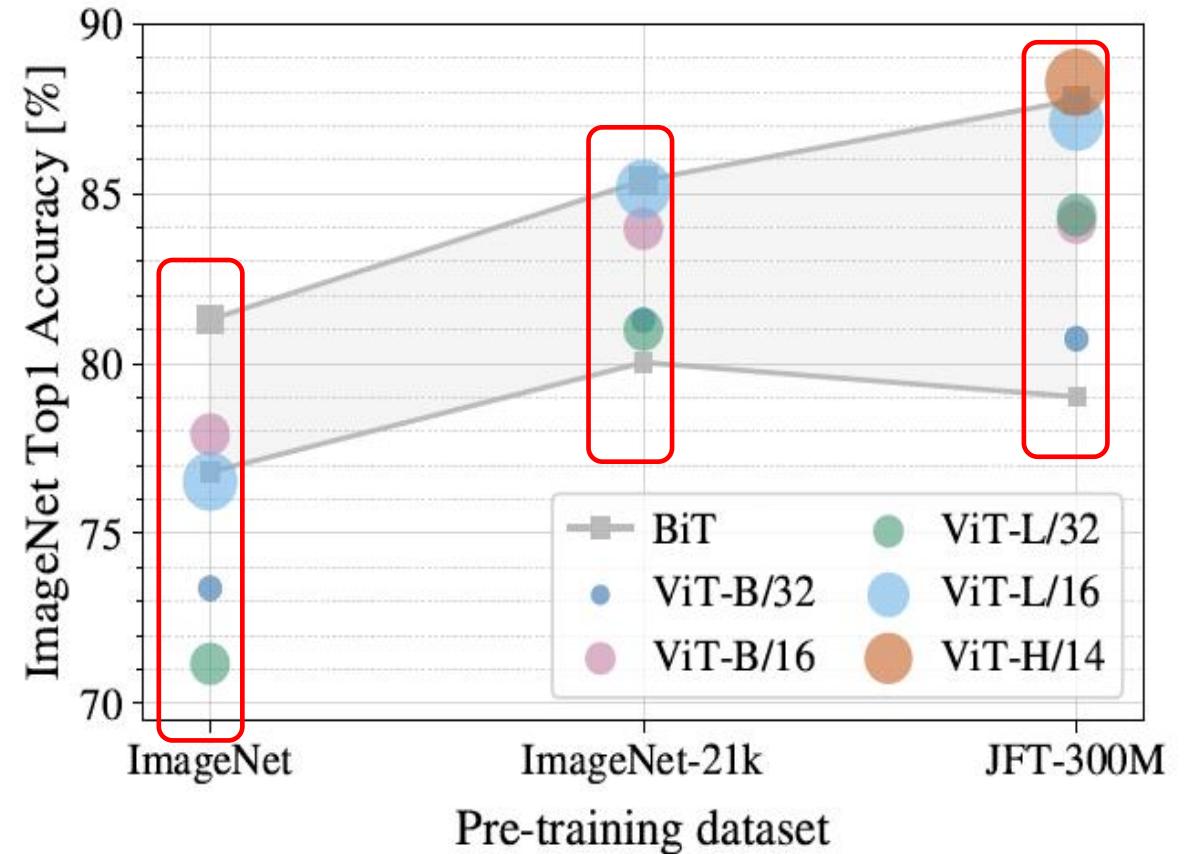
JFT300M包含 3 亿张带标签的图像，是谷歌为内部使用而开发的专有数据集

JFT300M比著名的 ImageNet 数据集大得多



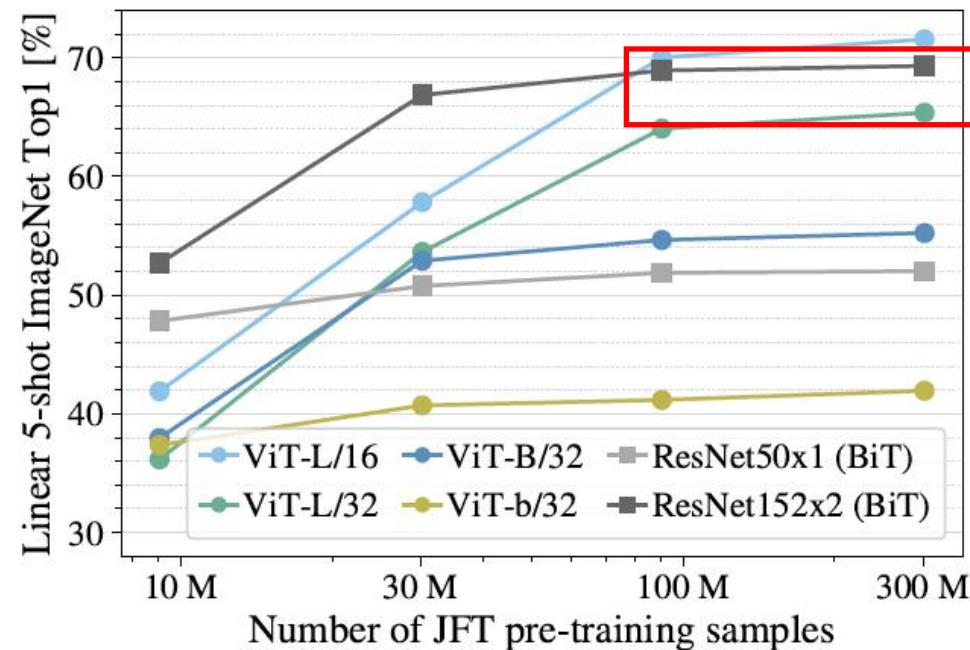
- 在不同的数据集上进行监督预训练，然后在 ImageNet 上进行微调以进行评估

- ✓ 对于小型预训练数据集，ResNet 表现更好，ViT-B 表现优于 ViT-L，这可能是由于 CNN 对图像的归纳偏置以及大型 ViT 容易过拟合
- ✓ 对于大型数据集，ViT-L 和 ViT-H 的优势比 ResNet 和 ViT-B 更为突出
- ✓ 小图像块 (16×16) 比大图像块 (32×32) 更受欢迎



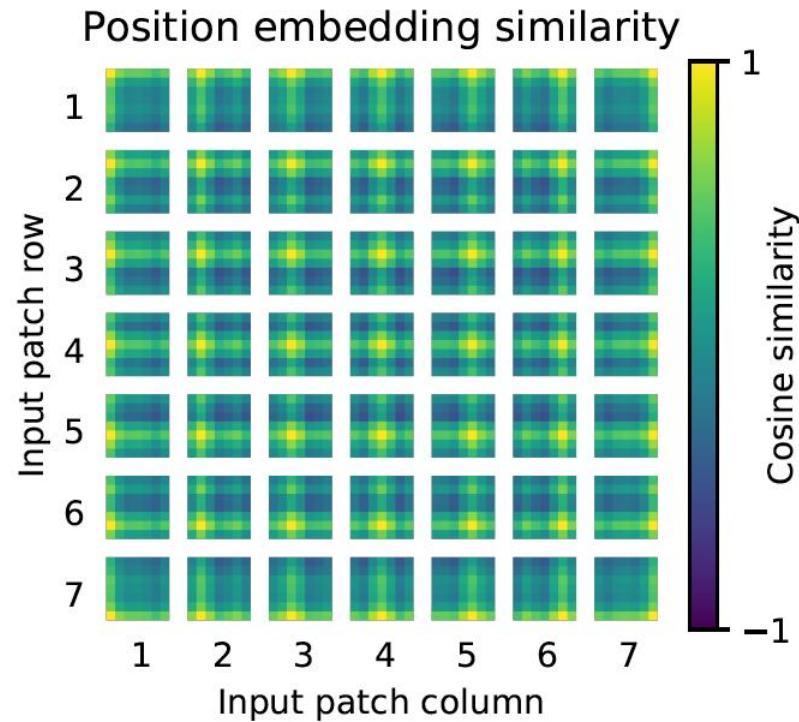
- ViT-L/16 意味着图像块大小设置为 16×16
- BiT 指在 ImageNet 和辅助数据上训练的 ResNet

- 在不同比例的 JFT-300 数据集上进行监督预训练，然后在 ImageNet 上对输出表示进行线性探测

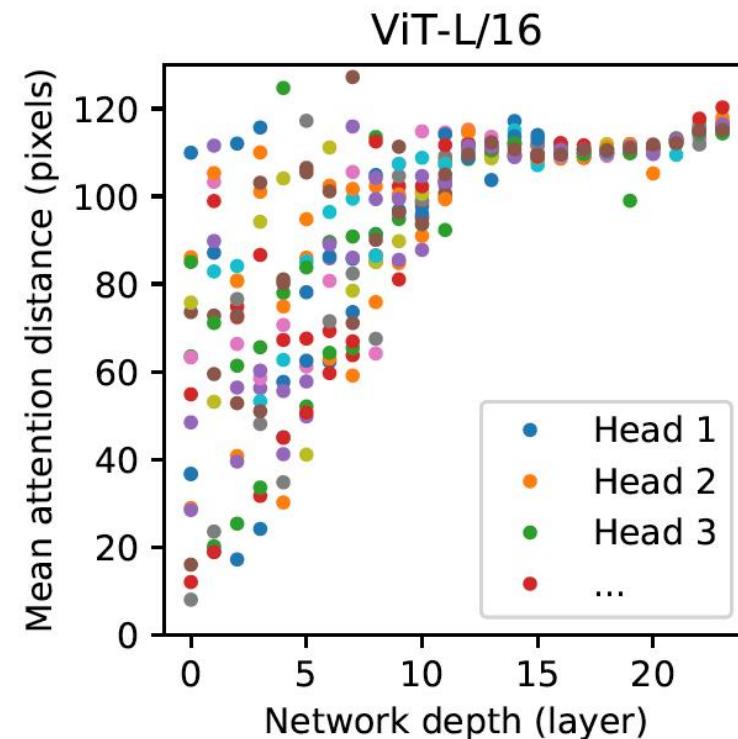


- ✓ 结果表明，当预训练数据集足够大时，ViT-L 学习到的表示比 ResNet 更好
 - 从海量数据中学到的知识压倒了 CNN 的归纳偏置

➤ ViT-L/32 的位置嵌入的相似性



➤ 每个头和网络深度对关注区域大小的影响。图中每个点代表16个注意力头中的一个，并显示了其在所有图像上的平均注意力距离。



— 学习到的位置嵌入可以自己发现相关的行和列

— 浅层：一些头关注局部图像块，而一些头关注远距离图像块
— 深层：大多数关注远距离图像块

大纲

- 基于Transformer的视觉处理架构ViT
- CLIP
- BLIP

- 使用标签作为监督的局限性
 - 1) 标注图像成本高昂
 - 2) 难以获取网络规模的数据集
 - 3) 仅使用一个标签来描述图像既不准确也不完整



如何只用一个标签来定义该图像表达的信息？

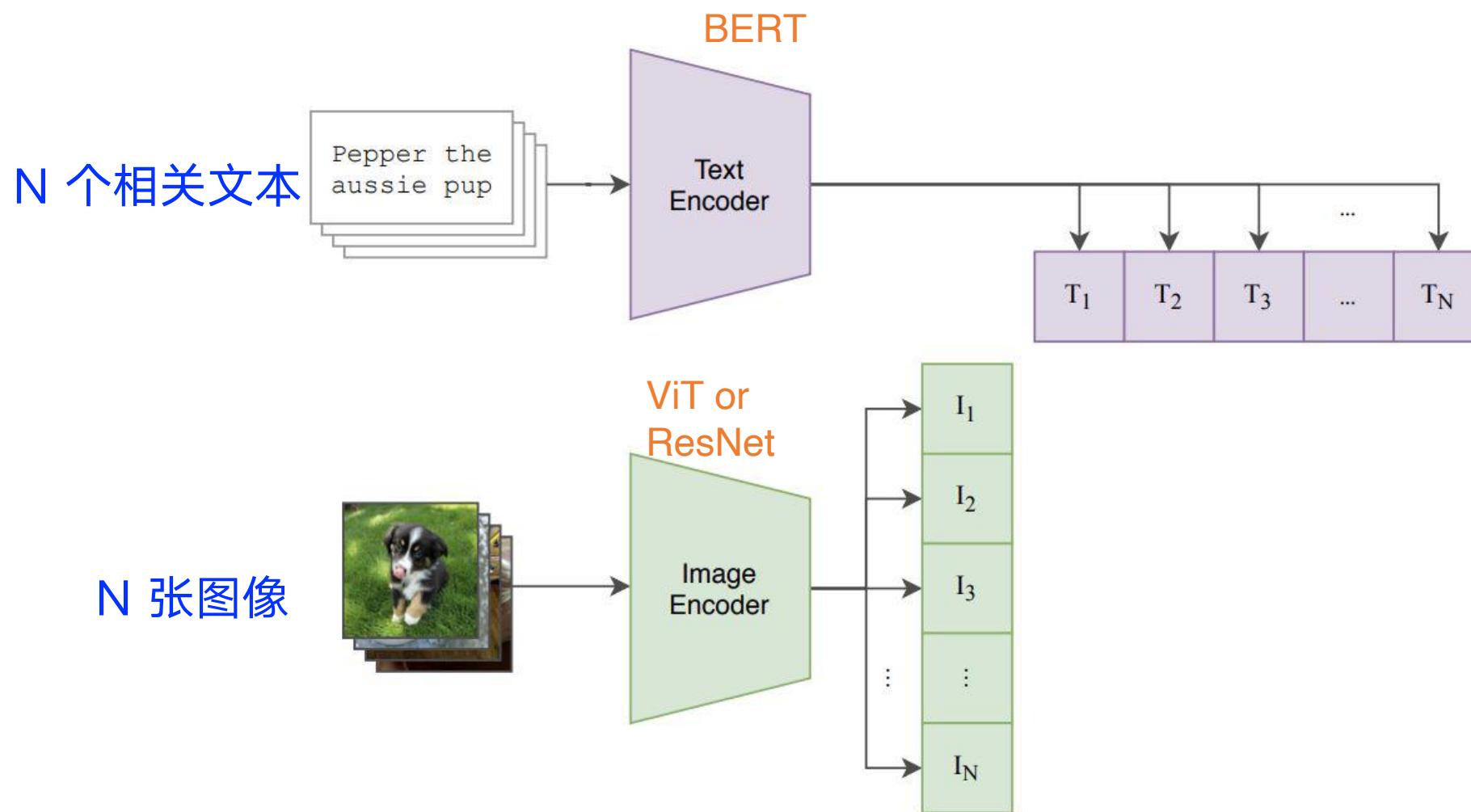
- 4) 从不利用或对齐自然语言中的标签名称

从自然语言监督中学习可转移的视觉模型

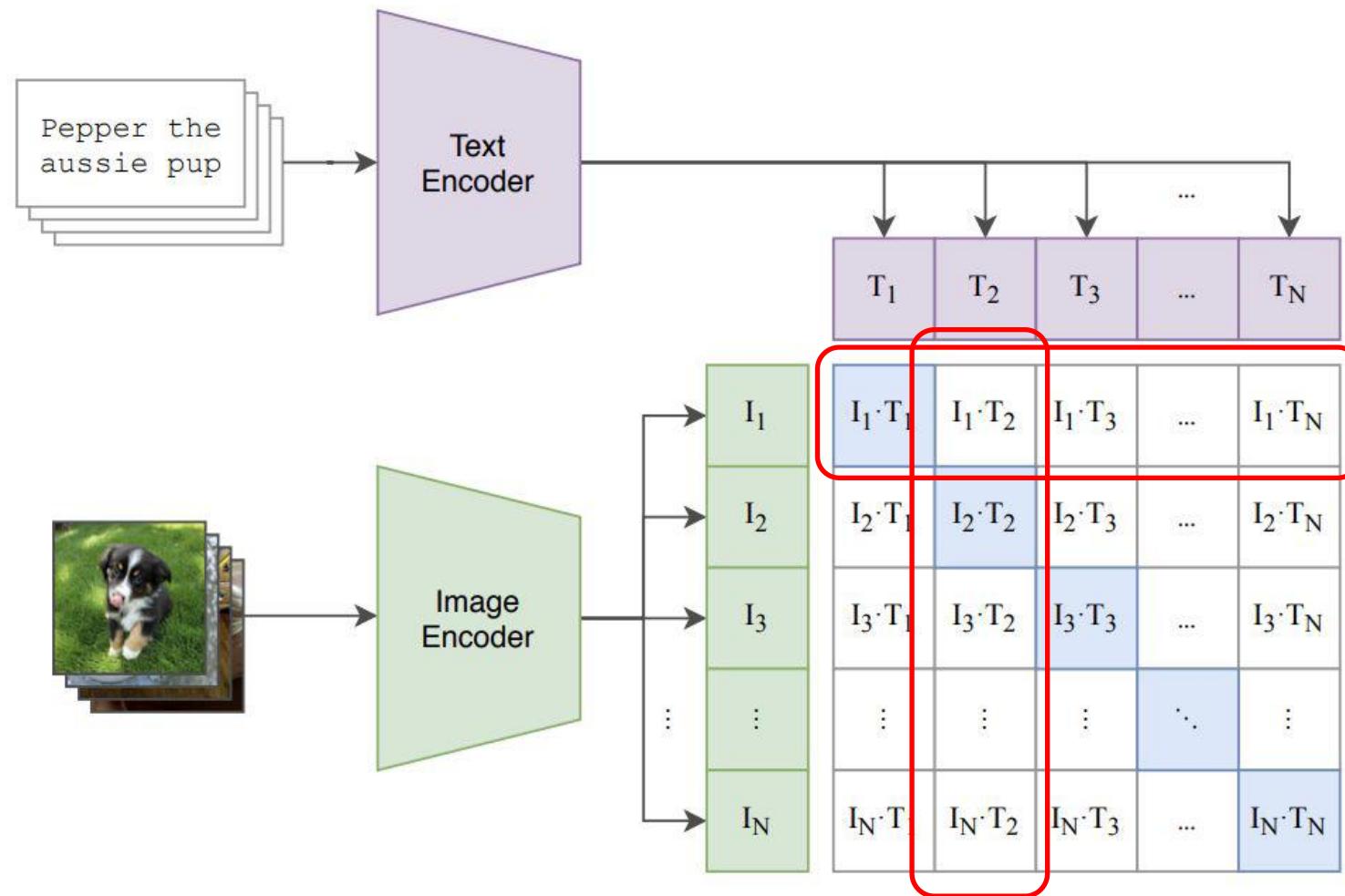
Radford et. al, Learning Transferable Visual Models From Natural Language Supervision, ICML 2021

- 构建训练数据集——WebImageText (WIT)
 - 从互联网上各种公开可用的来源收集了 4 亿个 (图像, 文本) 对
 - 进行一些预处理以使数据集涵盖尽可能广泛的视觉概念，并实现大致平衡的类别分布
- 基于收集到的<图像, 文本>数据集可能的预训练方式
 - 1) 方式一：图像 -> 确切的文本 (Image captioning)
 - 2) 方式二：图像 -> 相关文本的词袋 (预测某个单词是否出现, 图像二分类)
 - 3) 方式三：图像和文本之间的对比学习

- 方式三：通过图像和相关文本之间的对比学习进行预训练



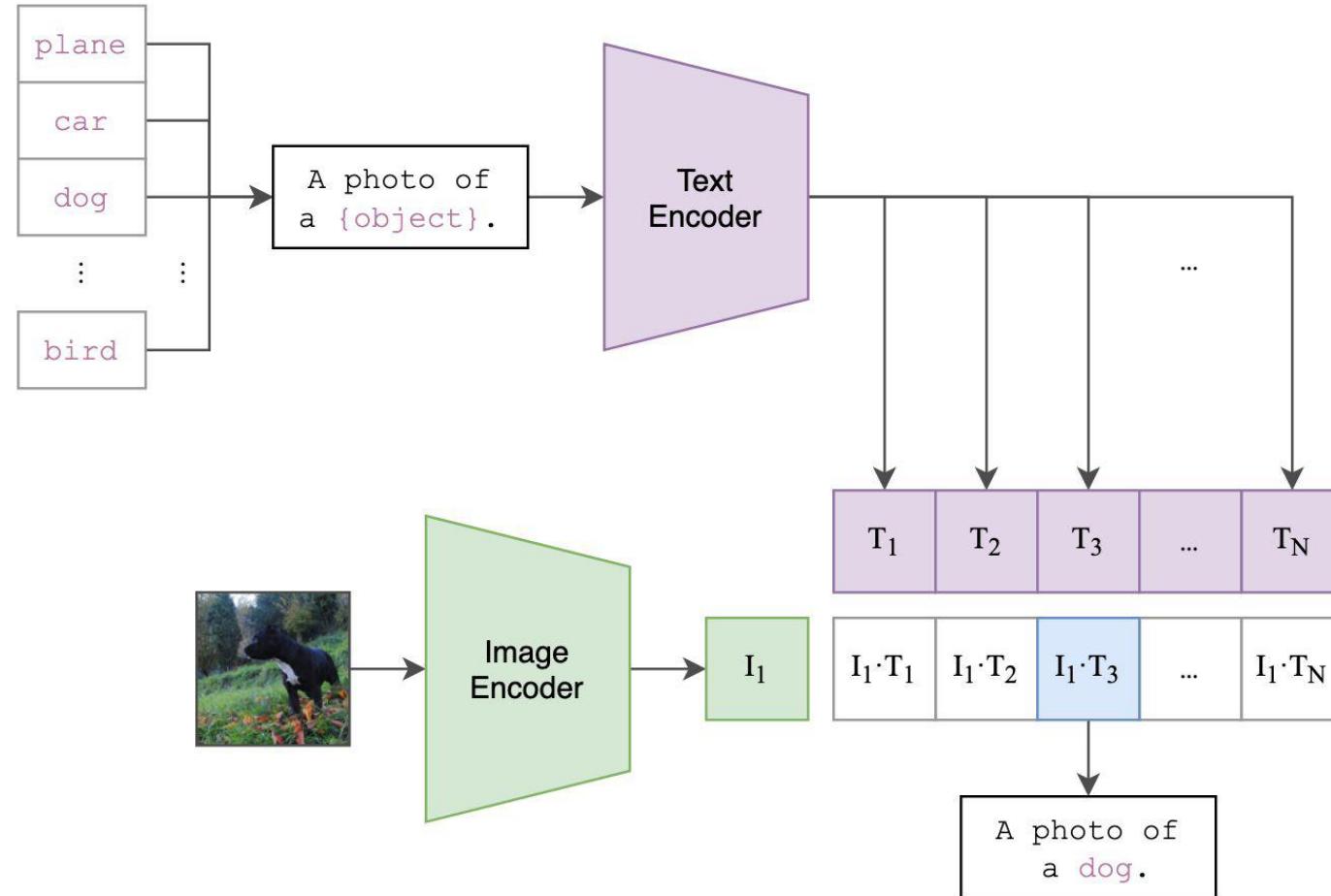
➤ 分别沿文本和图像维度进行对比训练



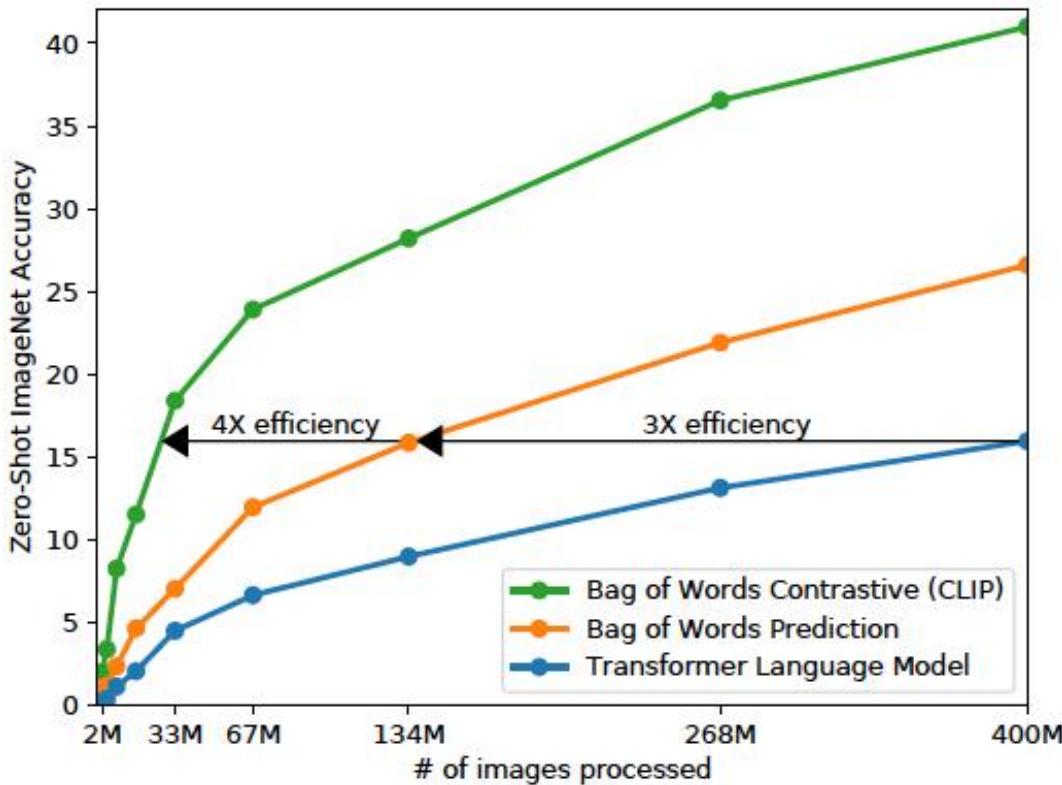
- 训练参数说明

- 使用来自互联网的 4 亿个图像-文本对进行训练
- Batch size 为 32768
- 在数据集上进行 32 个 epoch
- 余弦学习率衰减

- 如何使用 CLIP 进行零样本学习



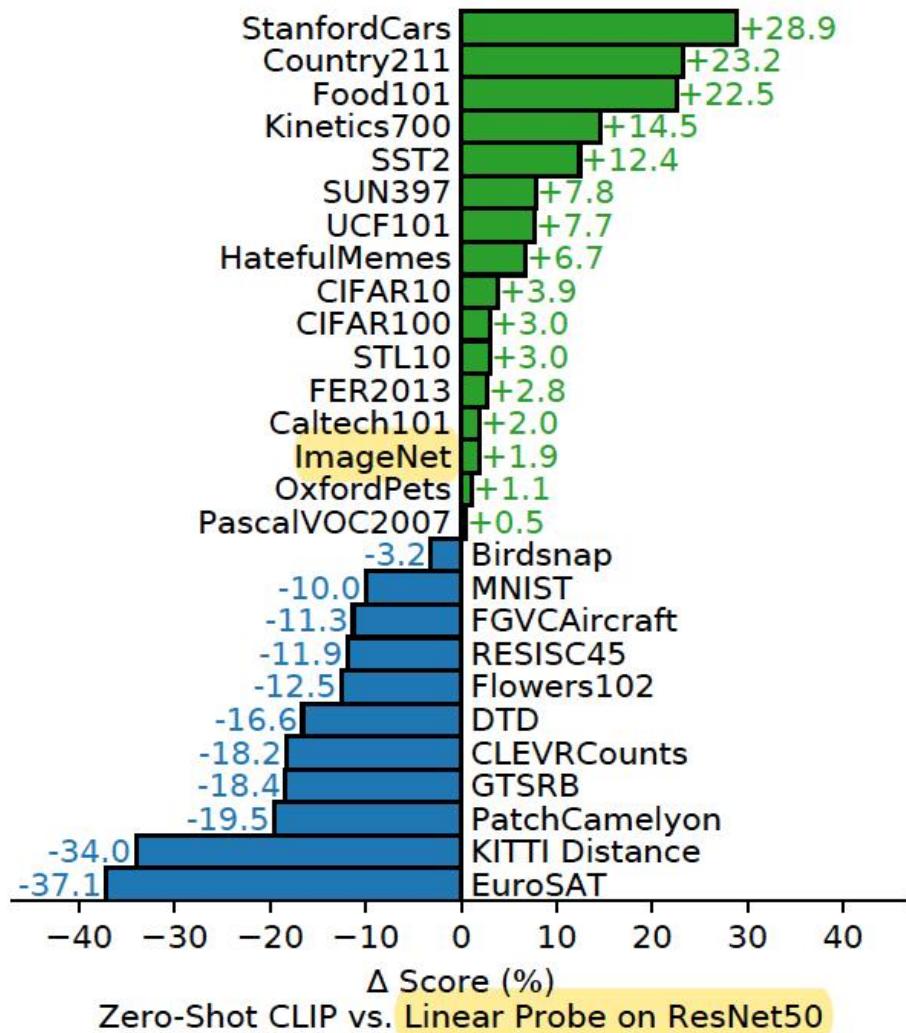
- 在 ImageNet 准确率的零样本性能上比较三种预训练方法



- Transformer Language Model: 预训练方式一
- Bag of Words Prediction: 预训练方式二
- Bag of Words Contrastive: 预训练方式三

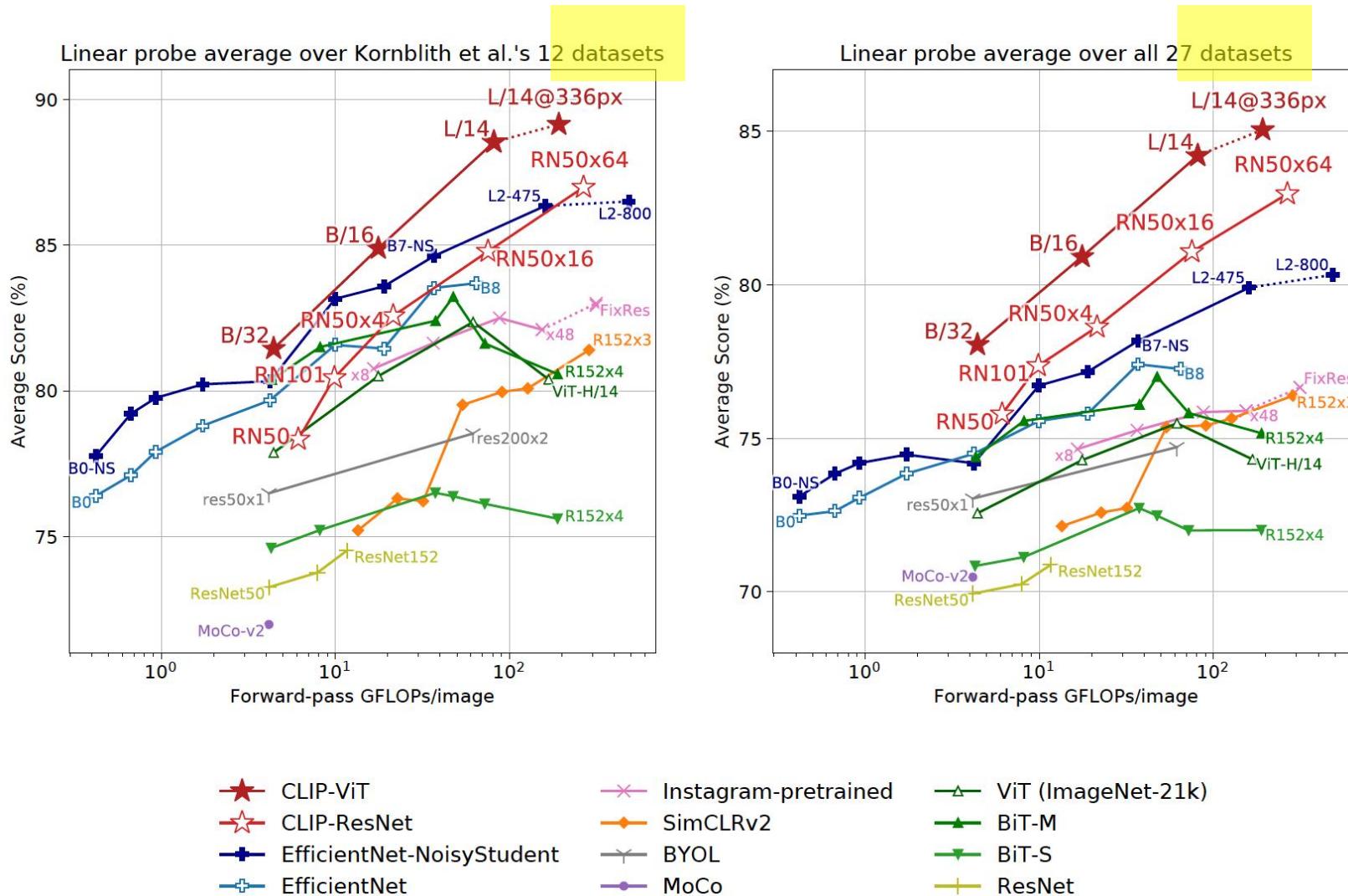
对比学习预训练方式在收敛速度和最终性能方面都展现出突出优势

- 零样本 CLIP vs ResNet 50 特征上完全监督的线性探针



- 在大多数自然图像数据集上，零样本训练的 CLIP 性能超过 ResNet 50 特征的完全监督线性探测性能，包括 ImageNet 数据集
- CLIP 只在在几个特殊数据集上表现较弱，例如 卫星图像分类、淋巴结肿瘤检测、合成场景中 物体的计数、德国交通标志识别

- CLIP 学习到的视觉表示的线性探测



- 对数据分布偏移的强鲁棒性

	Dataset Examples			ImageNet	Zero-Shot	ResNet101	CLIP	Δ Score
ImageNet				76.2	76.2	0%		
ImageNetV2				64.3	70.1	+5.8%		
ImageNet-R				37.7	88.9	+51.2%		
ObjectNet				32.6	72.3	+39.7%		
ImageNet Sketch				25.2	60.2	+35.0%		
ImageNet-A				2.7	77.1	+74.4%		

大纲

- 基于Transformer的视觉处理架构ViT
- CLIP
- BLIP

- CLIP 模型的局限
 - 模型视角
 - 仅采用基于编码器的结构，使其难以应用于涉及文本生成的任务，如Image captioning
 - 数据视角
 - 从互联网上抓取的 (图像，文本) 对是嘈杂的，对学习到的表示产生负面影响



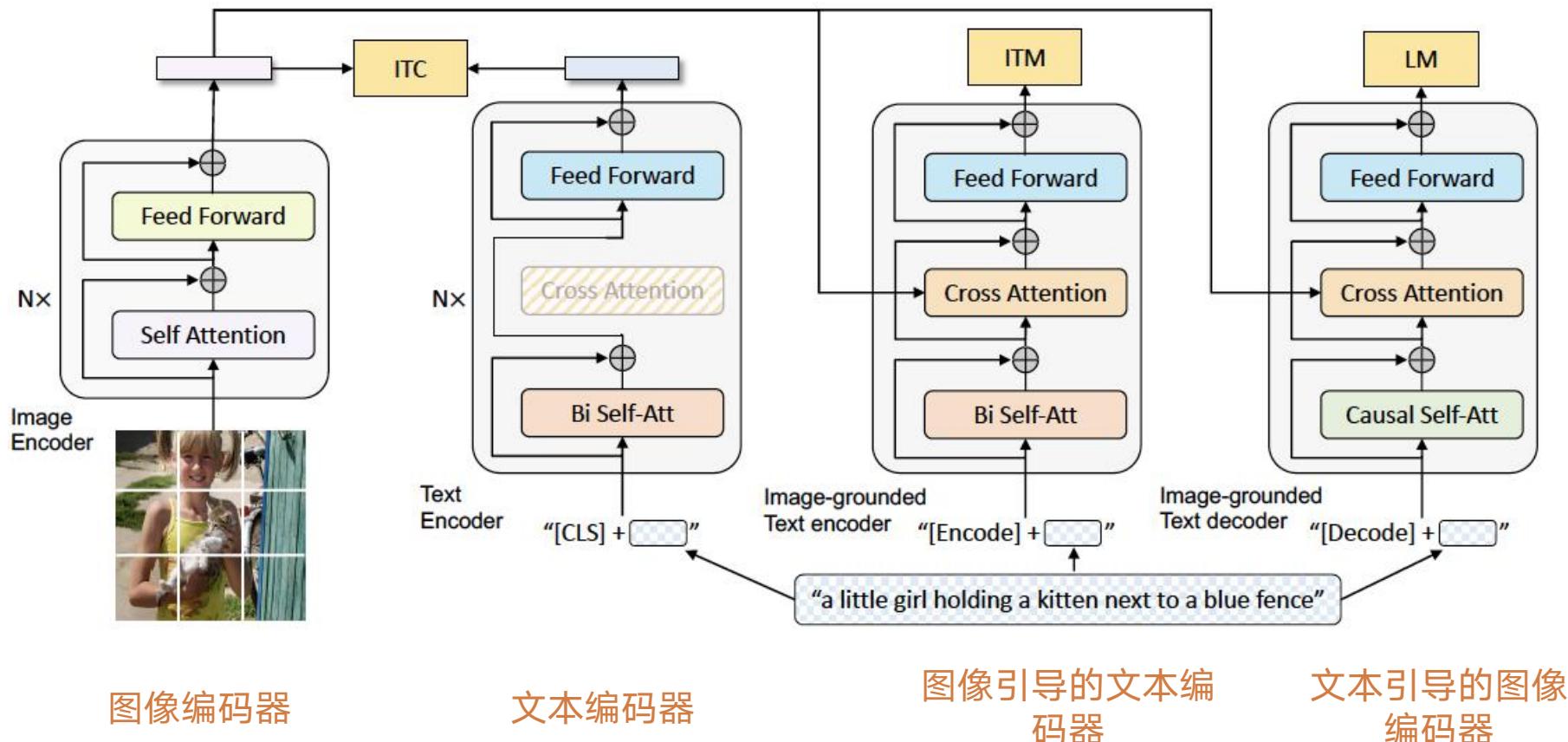
from bridge
near my
house



blue sky bakery
in sunset park

- 对于第一个问题，提出一个具有多个组件的结构

- 图像编码器
- 图像引导的文本编码器
- 文本编码器
- 文本引导的图像编码器

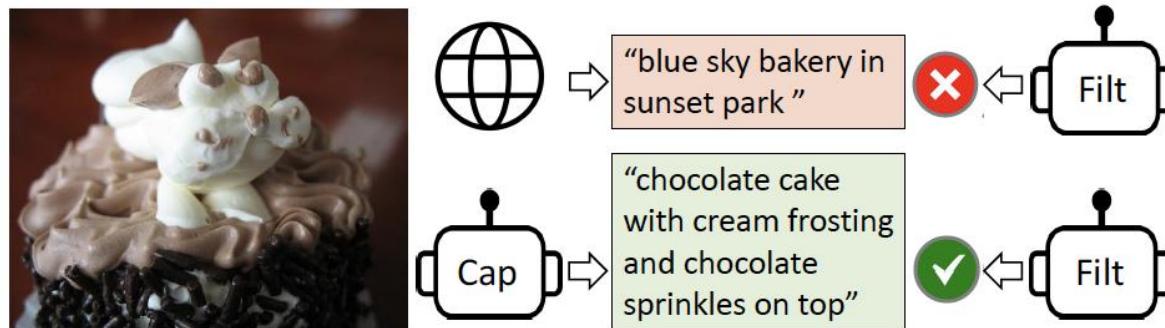


- 模型结构规格
 - 图像编码器：默认为 ViT-B
 - 文本编码器：默认为 BERT_{base}
- 训练数据集
 - 总共 14M 张图像 (主要)
 - 两个人工注释的数据集：COCO 和 Visual Genome
 - 两个网络数据集：Conceptual 12M, SBU 字幕
 - 一个额外的具有 1.15 亿张图像的网络数据集 (可选)

BLIP训练数据集的大小、训练batch size、模型参数量都比 CLIP 小得多

- 对于第二个问题，提出一种字幕和过滤 (CapFilt) 机制

- 过滤不正确的字幕
- 用生成的字幕替换它们



T_w : "from bridge near my house"

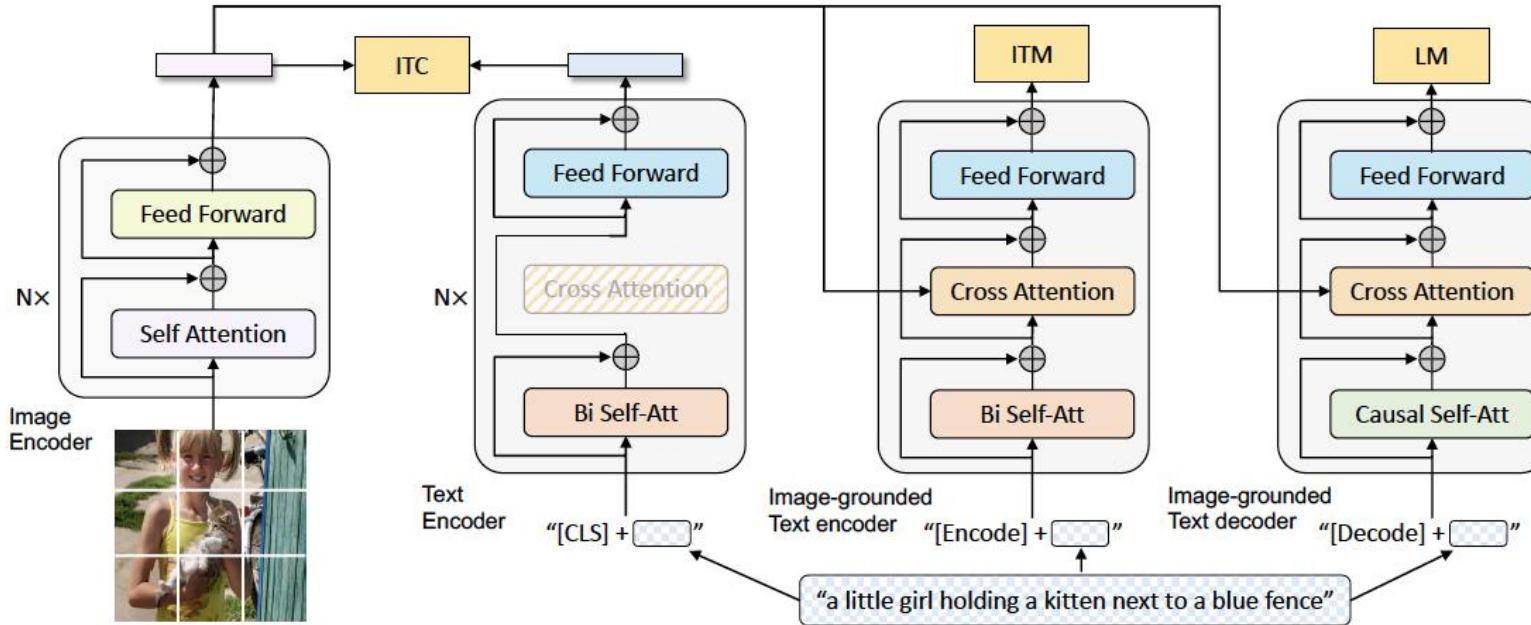
T_s : "a flock of birds flying over a lake at sunset"



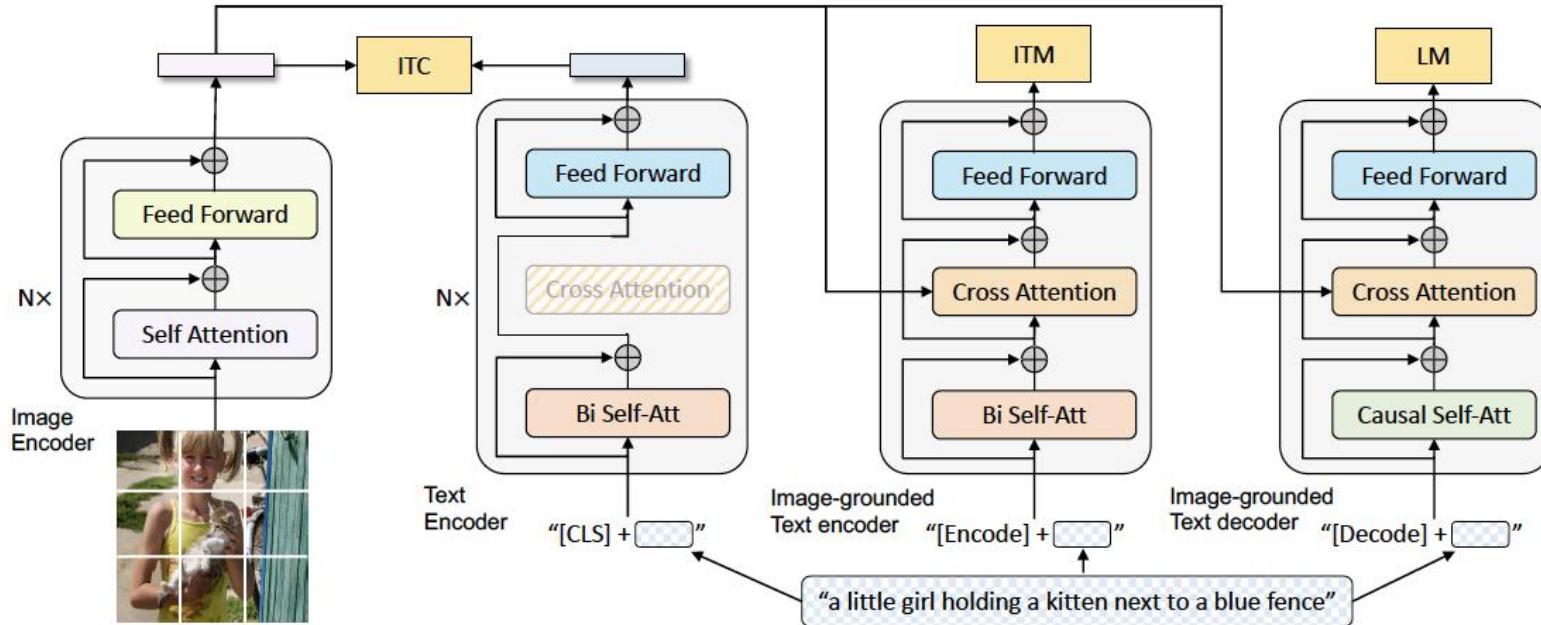
T_w : "in front of a house door in Reichenfels, Austria"

T_s : "a potted plant sitting on top of a pile of rocks"

- 预训练目标



- **图像-文本对比损失 (Image-Text Contrastive Loss, ITC):** 通过类似 CLIP 的对比学习来对齐视觉 transformer 和文本 transformer 的特征空间
- **图像-文本匹配损失 (Image-Text Matching Loss, ITM):** 用于预测图像-文本对是否匹配的二元分类器。它学习图像相关的文本表示，该表示通过交叉注意力机制 (cross attention) 涉及文本和图像。



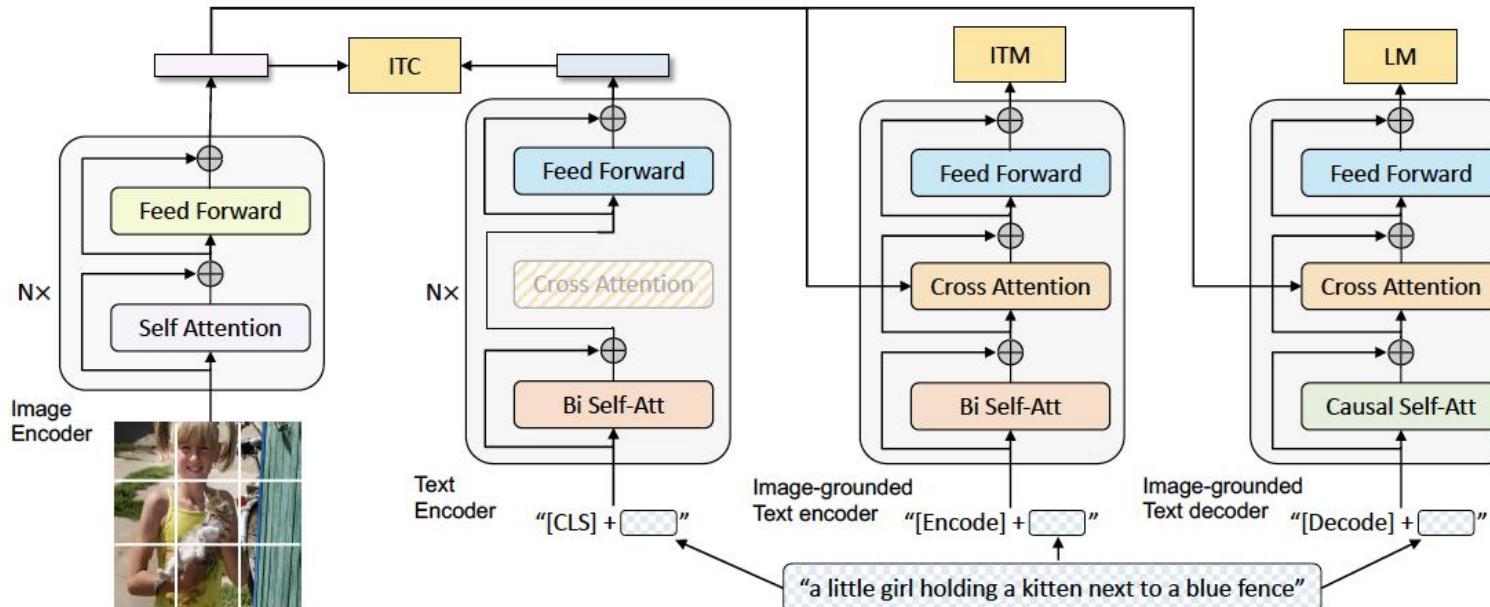
- 语言建模损失 (Language Modeling Loss, LM): 生成给定图像的文本描述。它通过最大化自回归方式的文本的对数似然来训练模型。

- 字幕 & 过滤 (CapFilt)

两个图像-文本对数据集

少量高质量的人工标注对 $\{(I_h, T_h)\}$
大量网络爬取的对 $\{(I_w, T_w)\}$

- 字幕生成器是以图像为依据的文本生成器，它会在高质量的 $\{(I_h, T_h)\}$ 数据集上进行进一步的微调
- $\{(I_h, T_h)\}$ 过滤器是以图像为依据的文本分类器，它会在高质量的 $\{(I_h, T_h)\}$ 数据集上进行进一步的微调。



- CapFilt 的效果通过零样本检索任务进行评估

Pre-train dataset	Bootstrap		Vision backbone	Retrieval-FT (COCO)		Retrieval-ZS (Flickr)		Caption-FT (COCO)		Caption-ZS (NoCaps)	
	C	F		TR@1	IR@1	TR@1	IR@1	B@4	CIDEr	CIDEr	SPICE
COCO+VG +CC+SBU (14M imgs)	X	X	ViT-B/16	78.4	60.7	93.9	82.1	38.0	127.8	102.2	13.9
	X	✓ _B		79.1	61.5	94.1	82.8	38.1	128.2	102.7	14.0
	✓ _B	X		79.7	62.0	94.4	83.6	38.4	128.9	103.4	14.2
	✓ _B	✓ _B		80.6	63.1	94.8	84.9	38.6	129.7	105.1	14.4
COCO+VG +CC+SBU +LAION (129M imgs)	X	X	ViT-B/16	79.6	62.0	94.3	83.6	38.8	130.1	105.4	14.2
	✓ _B	✓ _B		81.9	64.3	96.0	85.0	39.4	131.4	106.3	14.3
	✓ _L	✓ _L		81.2	64.1	96.0	85.5	39.7	133.3	109.6	14.7
	X	X		80.6	64.1	95.1	85.5	40.3	135.5	112.5	14.7
	✓ _L	✓ _L	ViT-L/16	82.4	65.1	96.7	86.7	40.4	136.7	113.2	14.8

Table 1. Evaluation of the effect of the captioner (C) and filter (F) for dataset bootstrapping. Downstream tasks include image-text retrieval and image captioning with finetuning (FT) and zero-shot (ZS) settings. TR / IR@1: recall@1 for text retrieval / image retrieval. ✓_{B/L}: captioner or filter uses ViT-B / ViT-L as vision backbone.

- 图像-文本检索结果，其中文本和图像编码器在 COCO 和 Flickr30K 数据集上进行了微调

Method	Pre-train # Images	COCO (5K test set)						Flickr30K (1K test set)					
		TR			IR			TR			IR		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
UNITER (Chen et al., 2020)	4M	65.7	88.6	93.8	52.9	79.9	88.0	87.3	98.0	99.2	75.6	94.1	96.8
VILLA (Gan et al., 2020)	4M	-	-	-	-	-	-	87.9	97.5	98.8	76.3	94.2	96.8
OSCAR (Li et al., 2020)	4M	70.0	91.1	95.5	54.0	80.8	88.5	-	-	-	-	-	-
UNIMO (Li et al., 2021b)	5.7M	-	-	-	-	-	-	89.4	98.9	99.8	78.0	94.2	97.1
ALIGN (Jia et al., 2021)	1.8B	77.0	93.5	96.9	59.9	83.3	89.8	95.3	99.8	100.0	84.9	97.4	98.6
ALBEF (Li et al., 2021a)	14M	77.6	94.3	97.2	60.7	84.3	90.5	95.9	99.8	100.0	85.6	97.5	98.9
BLIP	14M	80.6	95.2	97.6	63.1	85.3	91.1	96.6	99.8	100.0	87.2	97.5	98.8
BLIP	129M	81.9	95.4	97.8	64.3	85.7	91.5	97.3	99.9	100.0	87.3	97.6	98.9
BLIP _{CapFilt-L}	129M	81.2	95.7	97.9	64.1	85.8	91.6	97.2	99.9	100.0	87.5	97.7	98.9
BLIP _{ViT-L}	129M	82.4	95.4	97.9	65.1	86.3	91.8	97.4	99.8	99.9	87.6	97.7	99.0

Table 5. Comparison with state-of-the-art image-text retrieval methods, finetuned on COCO and Flickr30K datasets. BLIP_{CapFilt-L} pre-trains a model with ViT-B backbone using a dataset bootstrapped by captioner and filter with ViT-L.

- 图像字幕 (Image Captioning) 结果

Method	Pre-train #Images	NoCaps validation								COCO Caption	
		in-domain		near-domain		out-domain		overall		Karpathy test	
		C	S	C	S	C	S	C	S	B@4	C
Enc-Dec (Changpinyo et al., 2021)	15M	92.6	12.5	88.3	12.1	94.5	11.9	90.2	12.1	-	110.9
VinVL [†] (Zhang et al., 2021)	5.7M	103.1	14.2	96.1	13.8	88.3	12.1	95.5	13.5	38.2	129.3
LEMON _{base} [†] (Hu et al., 2021)	12M	104.5	14.6	100.7	14.0	96.7	12.4	100.4	13.8	-	-
LEMON _{base} [†] (Hu et al., 2021)	200M	107.7	14.7	106.2	14.3	107.9	13.1	106.8	14.1	40.3	133.3
BLIP	14M	111.3	15.1	104.5	14.4	102.4	13.7	105.1	14.4	38.6	129.7
BLIP	129M	109.1	14.8	105.8	14.4	105.7	13.7	106.3	14.3	39.4	131.4
BLIP _{CapFilt-L}	129M	111.8	14.9	108.6	14.8	111.5	14.2	109.6	14.7	39.7	133.3
LEMON _{large} [†] (Hu et al., 2021)	200M	116.9	15.8	113.3	15.1	111.3	14.0	113.4	15.0	40.6	135.7
SimVLM _{huge} (Wang et al., 2021)	1.8B	113.7	-	110.9	-	115.2	-	112.2	-	40.6	143.3
BLIP _{ViT-L}	129M	114.9	15.2	112.1	14.9	115.3	14.4	113.2	14.8	40.4	136.7

Table 7. Comparison with state-of-the-art image captioning methods on NoCaps and COCO Caption. All methods optimize the cross-entropy loss during finetuning. C: CIDEr, S: SPICE, B@4: BLEU@4. BLIP_{CapFilt-L} is pre-trained on a dataset bootstrapped by captioner and filter with ViT-L. VinVL[†] and LEMON[†] require an object detector pre-trained on 2.5M images with human-annotated bounding boxes and high resolution (800×1333) input images. SimVLM_{huge} uses 13 \times more training data and a larger vision backbone than ViT-L.

视觉问答与自然语言视觉推理 (Natural Language Visual Reasoning, NLVR²) 的结果

Method	Pre-train #Images	VQA		NLVR ²	
		test-dev	test-std	dev	test-P
LXMERT	180K	72.42	72.54	74.90	74.50
UNITER	4M	72.70	72.91	77.18	77.85
VL-T5/BART	180K	-	71.3	-	73.6
OSCAR	4M	73.16	73.44	78.07	78.36
SOHO	219K	73.25	73.47	76.37	77.32
VILLA	4M	73.59	73.67	78.39	79.30
UNIMO	5.6M	75.06	75.27	-	-
ALBEF	14M	75.84	76.04	82.55	83.14
SimVLM _{base} †	1.8B	77.87	78.14	81.72	81.77
BLIP	14M	77.54	77.62	82.67	82.30
BLIP	129M	78.24	78.17	82.48	83.08
BLIP _{CapFilt-L}	129M	78.25	78.32	82.15	82.24

Table 8. Comparison with state-of-the-art methods on VQA and NLVR². ALBEF performs an extra pre-training step for NLVR². SimVLM† uses 13× more training data and a larger vision backbone (ResNet+ViT) than BLIP.

课堂小结

- 基于Transformer的视觉处理模型-ViT
- 视觉-文本多模态表示与理解模型CLIP
- 视觉-文本多模态表示与理解模型BLIP



隐变量模型、EM算法和 变分自编码器

主讲人：林惊



无监督概率建模

- 在监督学习中，回归和分类都可以理解为是在学习条件概率分布

$$p(y|x; w)$$

- 在回归任务中，条件概率密度函数的形式通常假设为

$$p(y|x; w) = \mathcal{N}(y; w^T x, \sigma^2)$$

- 在分类任务中，条件概率密度函数的形式通常假设为

二分类: $p(y|x) = (\sigma(xw))^y \cdot (1 - \sigma(xw))^{1-y}$

多分类: $p(\mathbf{y}|x) = \prod_{k=1}^K [\text{softmax}_k(Wx)]^{y_k}$

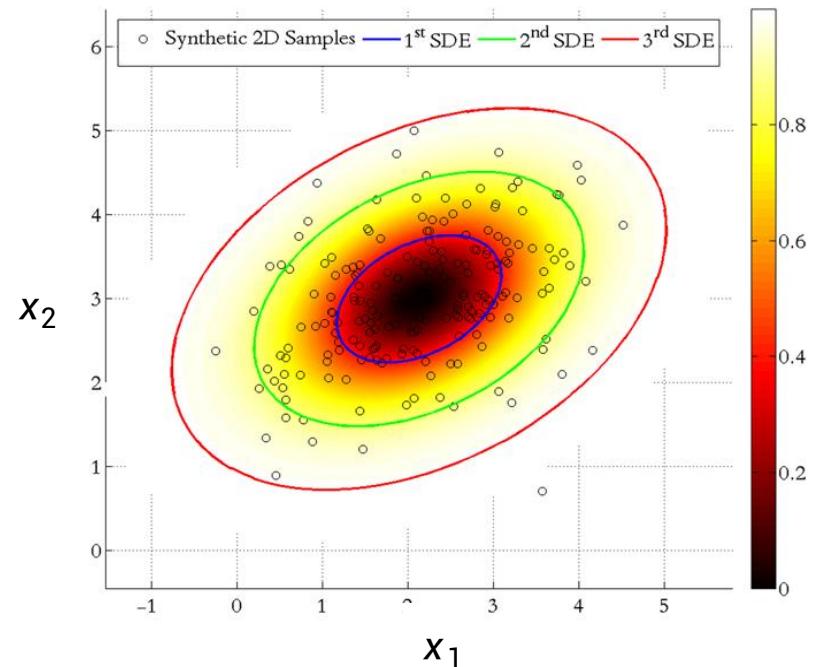
- 无监督学习也可以从学习概率分布的角度来理解。但它只关注输入数据 \mathbf{x} 的分布

$$p(\mathbf{x}; \mathbf{w})$$

- 建模 \mathbf{x} 要比标签 y 困难得多。一种朴素的方法是将 $p(\mathbf{x}; \mathbf{w})$ 限制为高斯形式

$$p(\mathbf{x}; \mathbf{w}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

通过优化 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 来最优化地描述数据点 $\{\mathbf{x}^{(n)}\}_{n=1}^N$

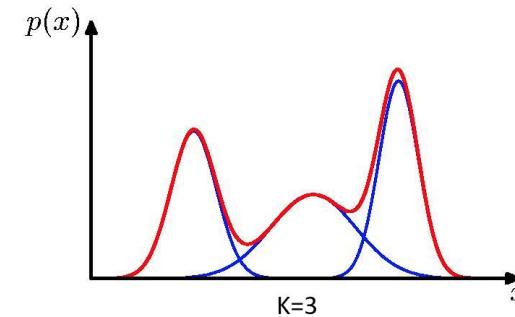


显然，该模型的表达能力非常有限

为什么需要隐变量？

- 理由1：通过组合简单模型构建更具表达能力的模型
 - 假设存在一个简单的类别分布 $p(z) = Cat(K, \pi)$ 和一个高斯分布 $p(x) = \mathcal{N}(x|\mu, \sigma^2)$
 - 如果单独使用它们，只能对简单的统计关系进行建模
 - 但如果我们将它们组合成 $p(x, z) = p(x|z)p(z)$ ，所得到的边缘分布 $p(x)$ 表达能力强得多

$$p(x) = \sum_z p(x|z)p(z) = \sum_{k=1}^K \pi_k \mathcal{N}(x|\mu_k, \sigma_k^2)$$



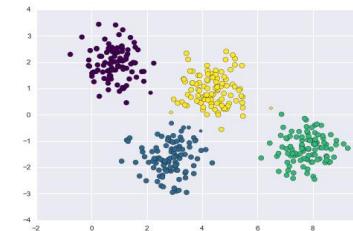
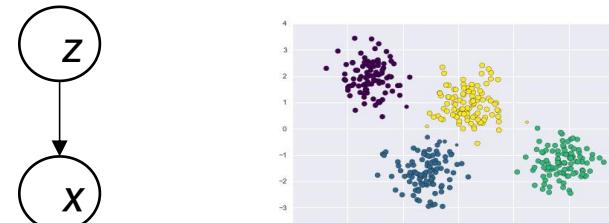
- 理论上，它能够表示任何复杂的分布

- 理由2：数据中隐藏的结构

- 1) 具有隐藏簇状结构的数据

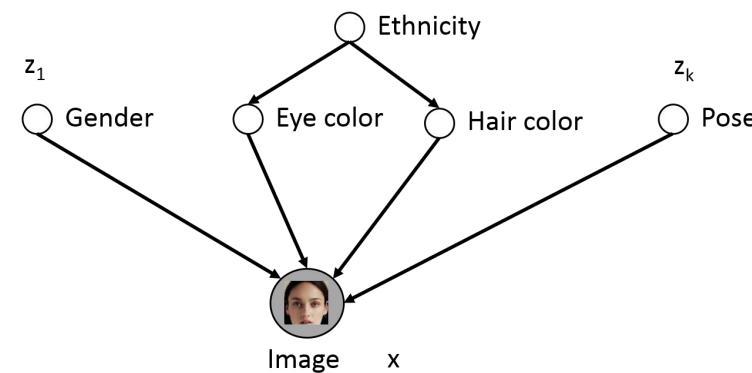
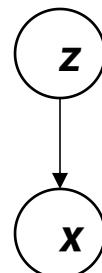
$$z_n \sim \text{Cat}(K, \pi)$$

$$x_n \sim \mathcal{N}(x | \mu_{z_n}, \Sigma_{z_n})$$



- 2) 面向文档的主题模型

- 3) 图像建模



- 在上面的例子中，隐变量 z 通常对应于高层级特征
- 如果考虑到这种隐藏结构，就可以获得更具可解释性的模型

隐变量模型的一般形式

- 隐变量模型 (LVMs) : 含有隐变量的概率模型

$$p(x, z)$$

- x 是我们感兴趣的随机变量
 - z 是隐变量 (Latent Variable)
- 有时可能存在多个隐变量 z_1, z_2, \dots, z_K

$$p(x, z_1, z_2, \dots, z_K)$$

- 关于我们感兴趣的变量 x 的概率模型是

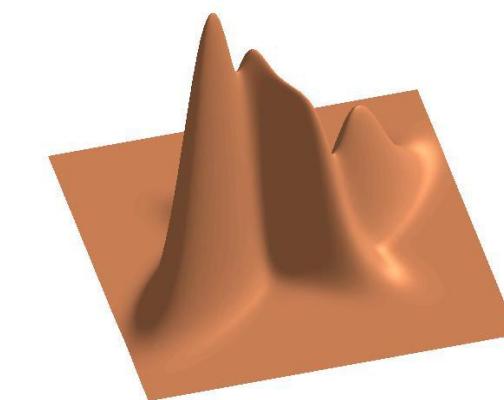
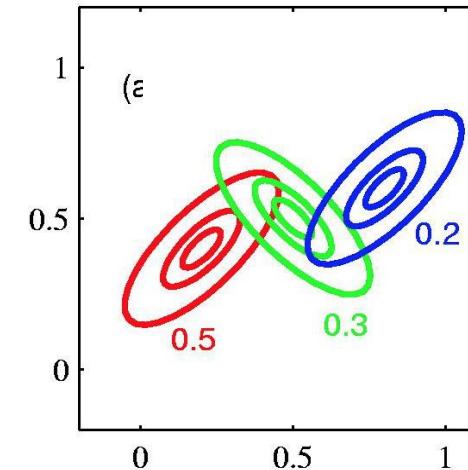
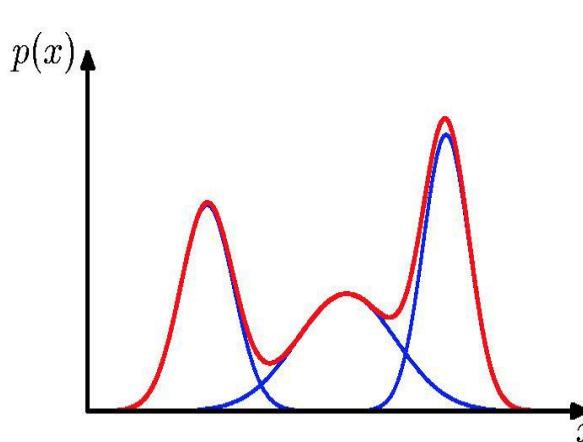
$$\begin{aligned} p(x) &= \int_z p(x, z) dz \quad \text{或} \quad p(x) = \int_{z_1 \dots z_K} p(x, z_1, \dots, z_K) dz_1 \dots dz_K \end{aligned}$$

高斯混合分布

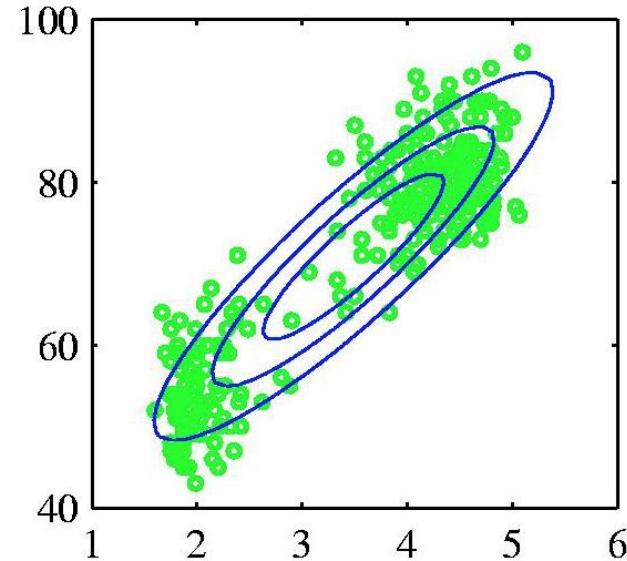
- 高斯混合分布的表达式

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

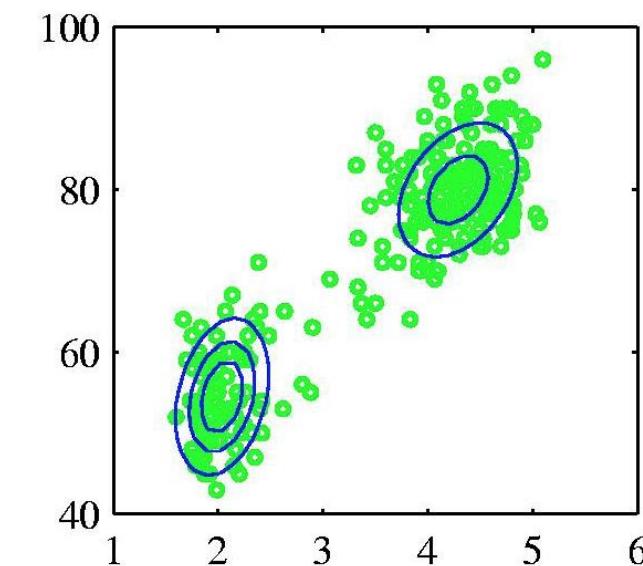
- K 表示高斯分布的数量
- π_k 是第 k 个分布的权重，满足 $\sum_{k=1}^K \pi_k = 1$
- $\boldsymbol{\mu}_k$ 和 $\boldsymbol{\Sigma}_k$ 分别是第 k 个高斯分布的均值向量和协方差矩阵



- 这些绿色的点很难用单个高斯分布建模



- 但如果用两个高斯分布的混合来建模，效果会好得多



将高斯混合分布表示为隐变量模型

- 对于一个隐变量模型 $p(\mathbf{x}, \mathbf{z})$ ，如果我们将其条件分布 $p(\mathbf{x}|\mathbf{z})$ 和先验分布 $p(\mathbf{z})$ 设为

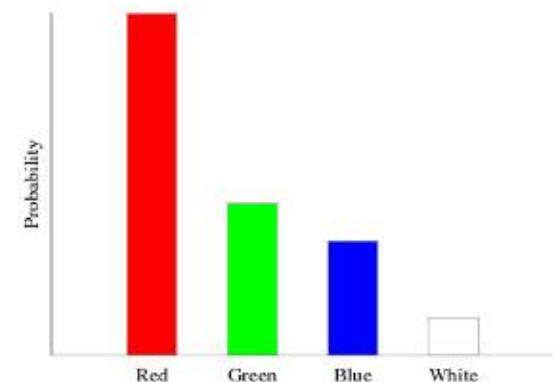
$$p(\mathbf{x}|\mathbf{z} = \mathbf{1}_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\mathbf{z} = \mathbf{1}_k) = \pi_k$$

- \mathbf{z} 只能是一个 one-hot 向量，其中 $\mathbf{1}_k$ 表示第 k 个元素为 1
- 由于 $p(\mathbf{z} = \mathbf{1}_k) = \pi_k$, $p(\mathbf{z})$ 实际上表示一个类别分布，即

$$p(\mathbf{z}) = Cat(\mathbf{z}; \boldsymbol{\pi})$$

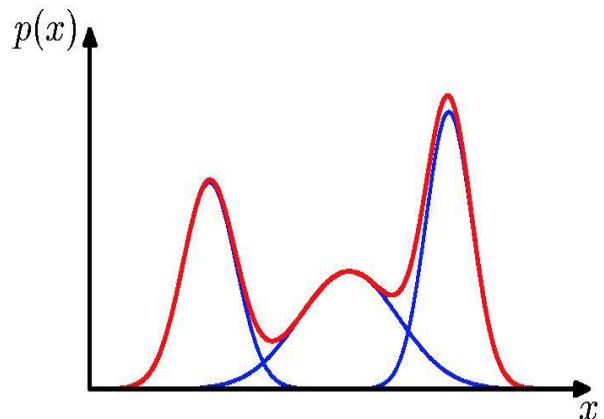
其中 $Cat(\mathbf{z} = \mathbf{1}_k; \boldsymbol{\pi}) = \pi_k$ 且 $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_K]$



- 由于 $p(\mathbf{x}) = \sum_z p(\mathbf{x}, \mathbf{z})$, 我们可以很容易得到

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

这正是高斯混合分布



- 因此，高斯混合分布也可以等价用隐变量模型来表示，其中 $p(\mathbf{x}|\mathbf{z} = \mathbf{1}_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $p(\mathbf{z} = \mathbf{1}_k) = \pi_k$

高斯混合分布的隐变量分布 $p(\mathbf{x}, \mathbf{z})$ 可以写成如下形式

$$p(\mathbf{x}, \mathbf{z}) = \prod_{k=1}^K [\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_k}$$

训练目标：最大化边缘概率分布

- 给定一组训练数据 $\{\mathbf{x}^{(n)}\}_{n=1}^N$ ， 目标是学习分布的参数

$$\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K \triangleq \boldsymbol{\theta}$$

- 假设数据点 $\mathbf{x}^{(n)}$ 是独立同分布的，因此我们可以将联合分布写为

$$p(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = \underbrace{\prod_{n=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{p(\mathbf{x}^{(n)})}$$

未使用隐变量形式

- 对于概率模型，训练目标是**最大化对数似然函数**，即

$$\log p(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

针对问题的一般形式

- 给定联合分布

$$p(x, z; \theta),$$

其中 x 是观测变量， z 是隐变量，我们需要最大化关于 θ 的对数似然，即

$$\theta = \arg \max_{\theta} \log p(x; \theta),$$

其中

$$p(x; \theta) = \sum_z p(x, z; \theta)$$

我们拥有的是 联合概率密度函数 $p(x, z; \theta)$ ，但需要优化的是边缘概率密度函数
 $p(x; \theta)$

一般隐变量模型的训练

- 根据边缘分布与联合分布的关系，可知

$$p_{\theta}(x_n) = \int_{z_n} p_{\theta}(x_n, z_n) dz_n$$

训练目标就是最大化对数似然函数

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log p_{\theta}(x_n)$$

- 但是，由于积分的存在，很多时候，获得 $p_{\theta}(x_n)$ 的表达式是很困难的
- 因此，对于一般隐变量模型，计算对数似然的梯度 $\frac{d \log p_{\theta}(x)}{d \theta}$ 是很困难的，使用梯度下降法来训练模型就变得非常具有挑战性

对数似然的重表示

- 对数似然可以重写为

$$\begin{aligned}\log p(\mathbf{x}; \boldsymbol{\theta}) &= \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{x}) && \forall \text{ 分布 } q(\mathbf{z}) \\ &= \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}) q(\mathbf{z})} \\ &= \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} + \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})}}_{KL(q||p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}))} \\ &= \mathcal{L}(q, \boldsymbol{\theta}) + KL(q||p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})), \quad \text{for } \forall \boldsymbol{\theta}, q(\mathbf{z})\end{aligned}$$

注：KL 散度用于衡量两个分布 q 和 p 之间的距离，定义为

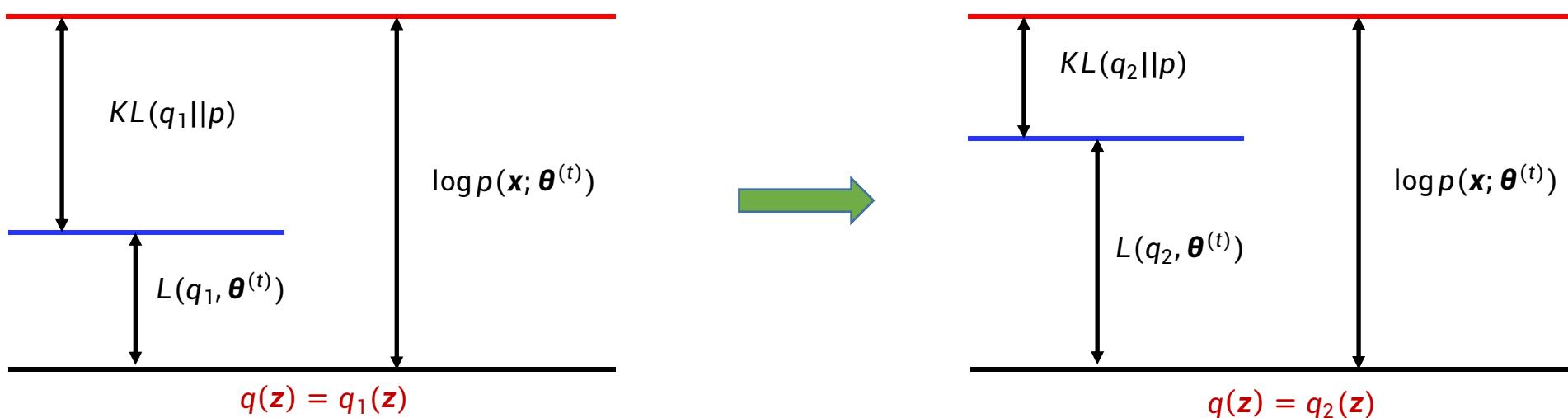
$$KL(q||p) \triangleq \int q(z) \log \frac{q(z)}{p(z)} dz \geq 0$$

- 因此，我们将第 t 次迭代的参数记为 $\theta^{(t)}$ ，可得

$$\log p(\mathbf{x}; \theta^{(t)}) = \mathbb{L}(q, \theta^{(t)}) + KL(q||p(\mathbf{z}|\mathbf{x}; \theta^{(t)}))$$

该等式对任何分布 $q(\mathbf{z})$ 都成立

- 不同的 $q(\mathbf{z})$ 会导致 $\log p(\mathbf{x}; \theta^{(t)})$ 的不同分解



EM 算法的理论依据

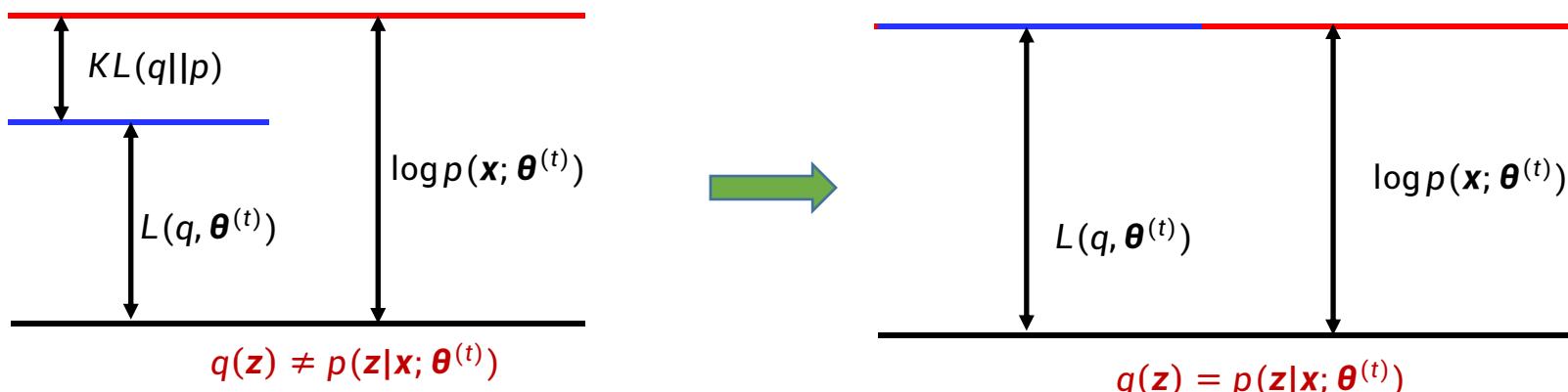
$$\log p(\mathbf{x}; \boldsymbol{\theta}^{(t)}) = \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}^{(t)})}{q(\mathbf{z})} + \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)})}$$

- 如果我们设 $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)})$, 那么可以得到

$$KL(q||p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)})) = 0$$

因此, 我们有

$$\begin{aligned}\log p(\mathbf{x}; \boldsymbol{\theta}^{(t)}) &= L(p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t)}) \\ &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)}) \log \frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}^{(t)})}{p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)})}\end{aligned}$$



$$\log p(\mathbf{x}; \boldsymbol{\theta}^{(t)}) = \mathbb{1}(p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t)}) = \sum_z p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)}) \log \frac{p(\mathbf{x}, z; \boldsymbol{\theta}^{(t)})}{p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)})}$$

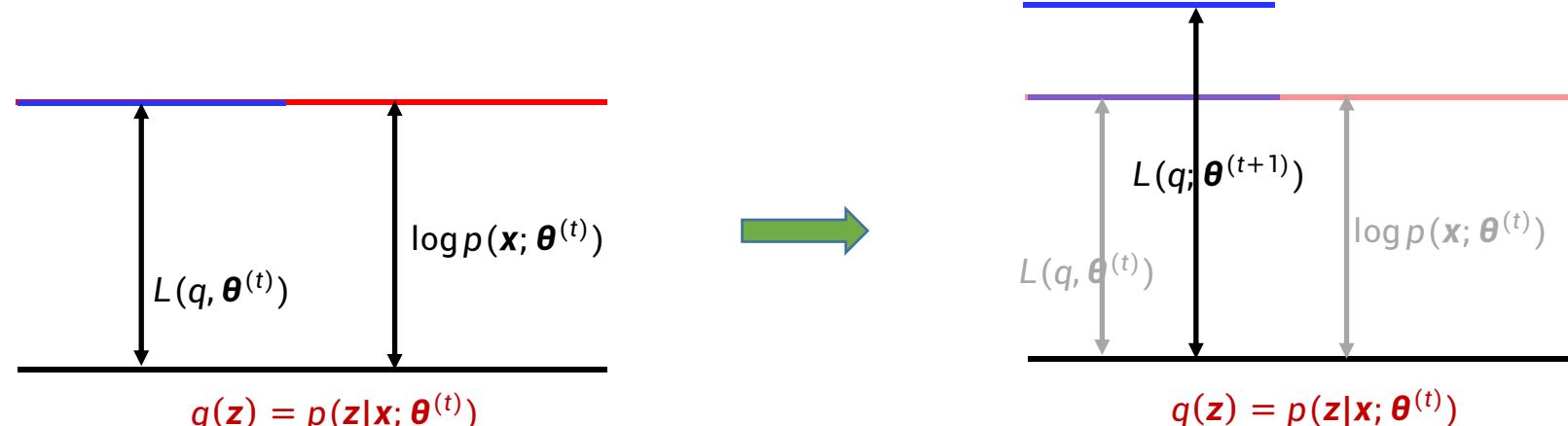
- 如果我们更新 $\boldsymbol{\theta}$ 为

$$\boldsymbol{\theta}^{(t+1)} = \arg \max_{\boldsymbol{\theta}} \mathbb{1}(p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}),$$

那么我们必然得到如下关系

$$\mathbb{1}(p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t+1)}) \geq \underbrace{\mathbb{1}(p(z|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t)})}_{= \log p(\mathbf{x}; \boldsymbol{\theta}^{(t)})}$$

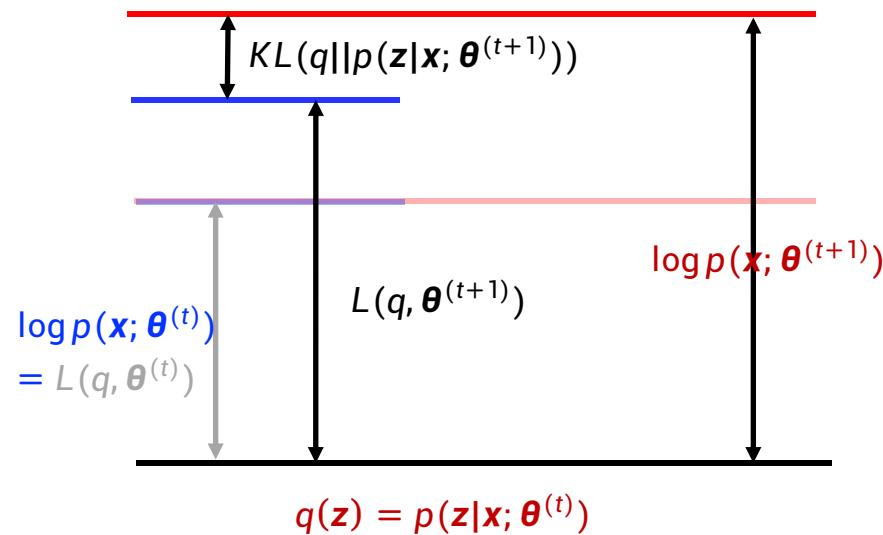
$$= \log p(\mathbf{x}; \boldsymbol{\theta}^{(t)})$$



- 通过设置 $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)})$, 我们得到

$$\log p(\mathbf{x}; \boldsymbol{\theta}^{(t+1)}) = \underbrace{\mathcal{L}(p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t+1)})}_{\geq \log p(\mathbf{x}; \boldsymbol{\theta}^{(t)})} + \underbrace{KL(p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)}))||p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t+1)})}_{\geq 0}$$

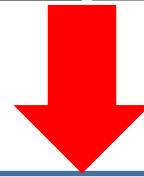
KL 散度始终为非负



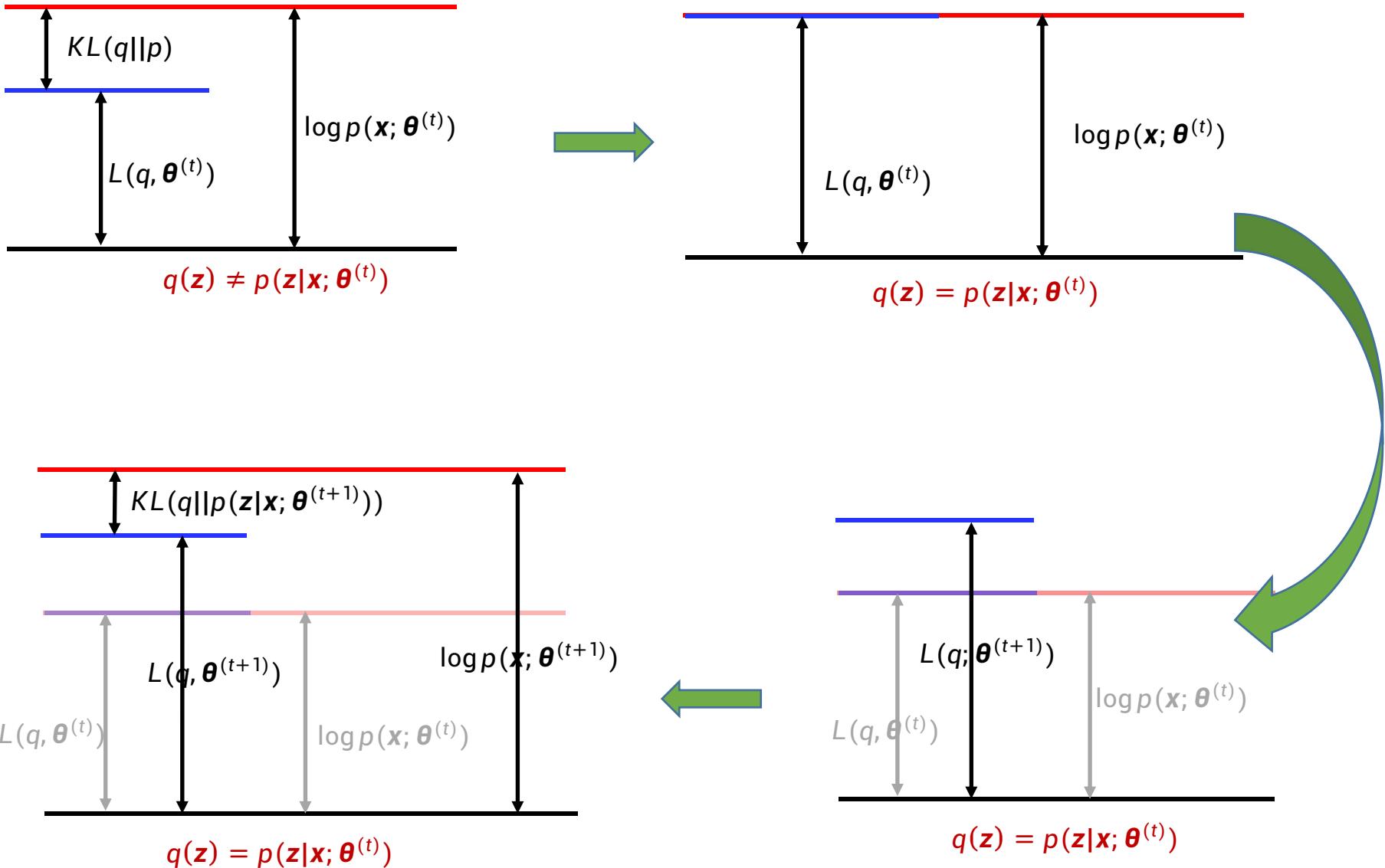
- 因此，我们可以看到

$$\log p(\mathbf{x}; \boldsymbol{\theta}^{(t+1)}) \geq \log p(\mathbf{x}; \boldsymbol{\theta}^{(t)})$$

KL 散度始终为非负



$\max_{\boldsymbol{\theta}} \text{L}(p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}^{(t)}), \boldsymbol{\theta})$ 能够保证每一步的似然函数值都会增加



EM 算法

- 算法

E 步: 计算期望

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{p(z|x; \boldsymbol{\theta}^{(t)})} [\log p(x, z; \boldsymbol{\theta})]$$

M 步: 更新参数 *EM* 算法可以保证每一步的似然函数值都会增加

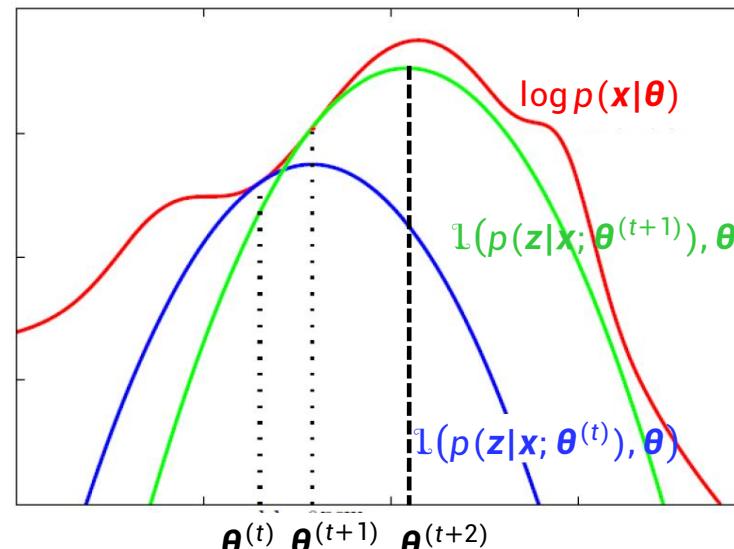
$$\boldsymbol{\theta}^{(t+1)} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$$

- EM 算法的关键要素

- 1) 后验分布 $p(z|x; \boldsymbol{\theta}^{(t)})$
- 2) 联合分布对数 $\log p(x, z; \boldsymbol{\theta})$ 关于后验概率的期望
- 3) 最大化

参数空间视角

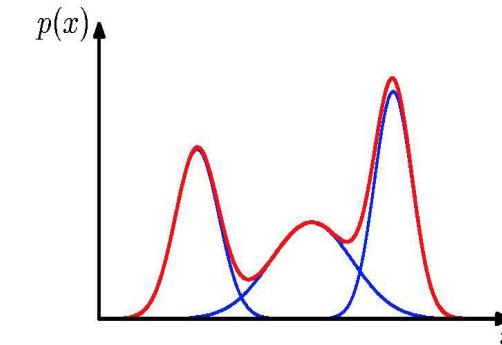
- 1) E 步 (t): 给定模型参数 $\theta^{(t)}$, 推导出 $\mathbb{L}(p(z|x; \theta^{(t)}), \theta)$ 的表达式
- 2) M 步 (t): 计算最优值 $\theta^{(t+1)} = \arg \max_{\theta} \mathbb{L}(p(z|x; \theta^{(t)}), \theta)$
- 3) E 步 ($t + 1$): 给定模型参数 $\theta^{(t+1)}$, 推导出 $\mathbb{L}(p(z|x; \theta^{(t+1)}), \theta)$ 的表达式
- 4) 重复以上过程直至收敛



高斯混合模型回顾

- 对于一个高斯混合分布，即

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$



它可以表示为联合分布的边缘分布

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

$$= \prod_{k=1}^K [\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_k}$$

- $\mathbf{z} = [z_1, z_2, \dots, z_K]$ 服从参数为 π 的类别分布

EM 算法的两个步骤

- 这是一个隐变量模型，因此我们可以使用 EM 算法来优化它

注: $\max_{\theta} \mathbb{L}(p(z|x; \theta^{(t)}), \theta)$ 等价于 $\max_{\theta} Q(\theta; \theta^{(t)})$

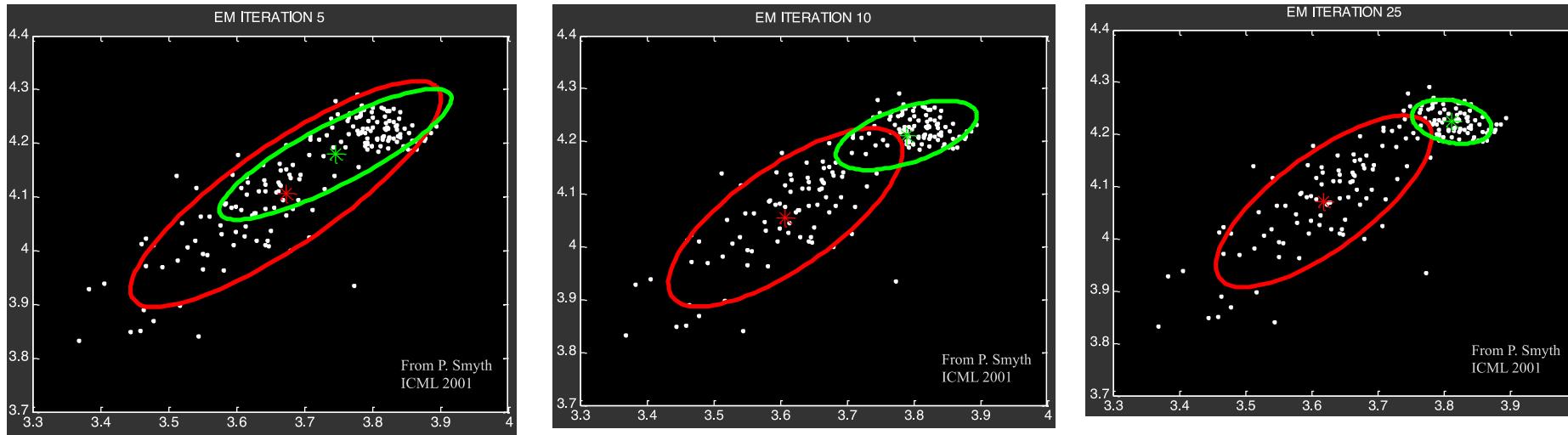
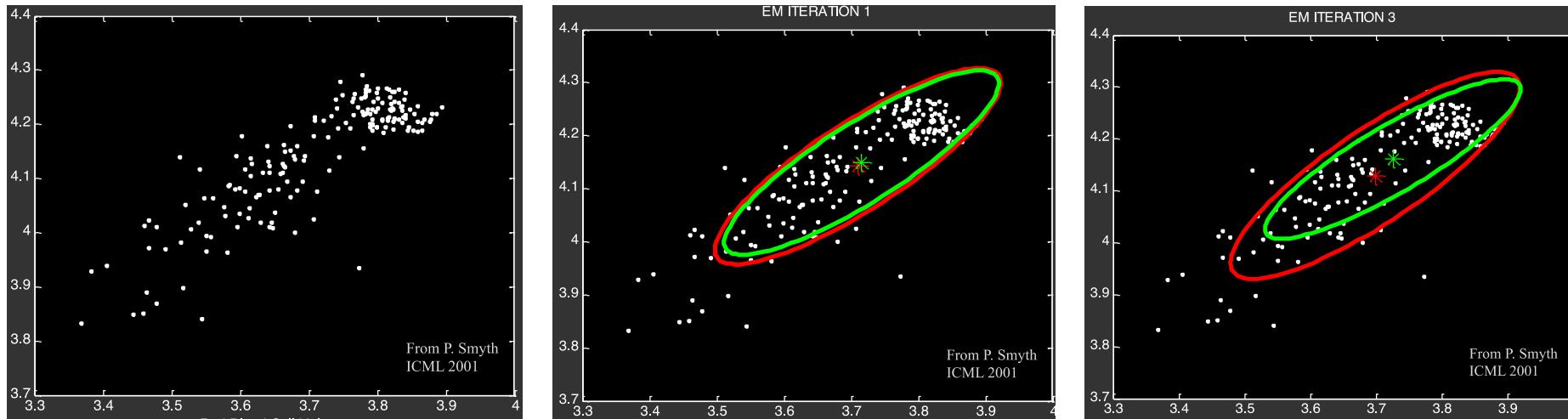
- 回顾: EM 算法的关键要素

➤ E 步: 计算关于后验概率 $p(z|x; \theta^{(t)})$ 的期望

$$Q(\theta; \theta^{(t)}) = \frac{1}{N} \sum_{n=1}^N \mathbb{E}_{p(z^{(n)}|x^{(n)}; \theta^{(t)})} [\log p(x^{(n)}, z^{(n)}; \theta)]$$

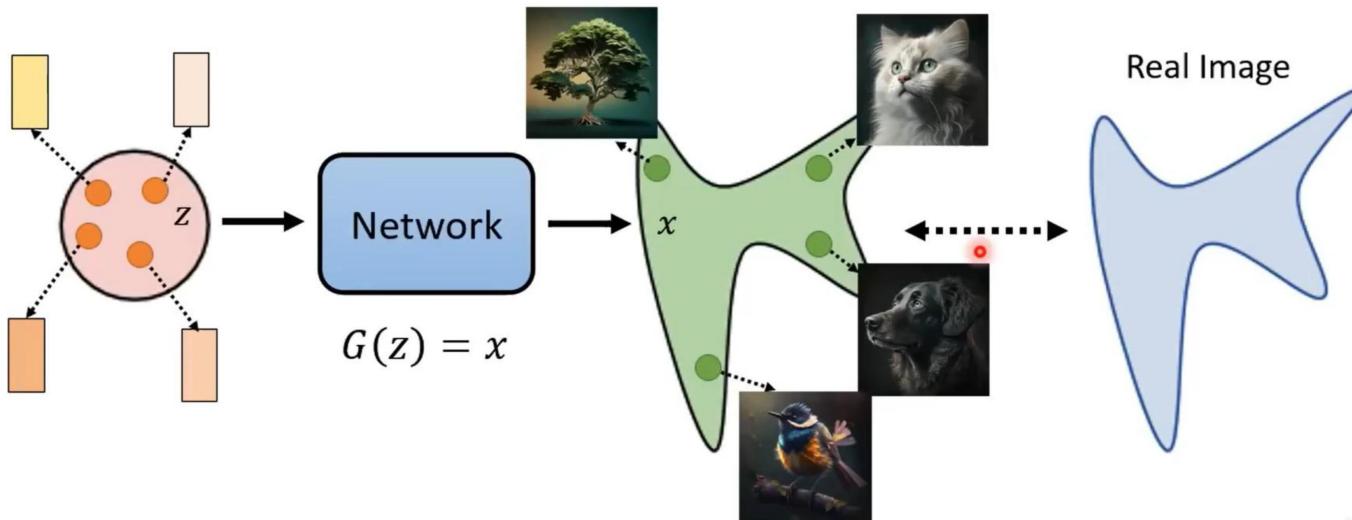
➤ M 步: 最大化

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta; \theta^{(t)})$$



从 EM 算法到变分自编码器

- 变分自编码器（Variational AutoEncoder, VAE）的直接数学目标就是：
输入一个向量 z , 经过模型得到 $G(z) = x$, 希望这个得到的 x 的分布 $p_\theta(x)$
与实际已有数据的分布 $p_{data}(x)$ 尽可能接近



- 优化目标:

$$\theta^* = \arg \min_{\theta} KL(p_\theta(x), p_{data}(x)) = \arg \max_{\theta} \mathbb{E}_{x \sim p(data)} [p_\theta(x)]$$

从 EM 算法到变分自编码器

- 优化目标：

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{x \sim p(\text{data})} [p_{\theta}(x)]$$

- EM算法最核心的地方在于，为隐变量 z 引入了一个分布 $q(z)$ ：

$$\begin{aligned} L(\theta) &= \log p_{\theta}(x) = \int_z q(z) \log p_{\theta}(x|z) dz \\ &= \int_z q(z) \log \left(\frac{p(x, z|\theta)}{p(z|x, \theta)} \cdot \frac{q(z)}{q(z)} \right) dz \\ &= \int_z q(z) \log \frac{p(x|z, \theta) \cdot p(z)}{q(z)} dz \\ &= \underbrace{\mathbb{E}_{z \sim q(z)} [\log p_{\theta}(x|z)]}_{ELBO} + \underbrace{KL(q(z) \| p_{\theta}(z|x))}_{\text{KL散度}} \end{aligned}$$

从 EM 算法到变分自编码器

- 优化目标：

$$L(\theta) = \underbrace{\mathbb{E}_{z \sim q(z)} [\log p_\theta(x|z)] - KL(q(z)\|p(z))}_{ELBO} + \underbrace{KL(q(z)\|p_\theta(z|x))}$$

- EM算法的基本思路：先让KL散度取0，这样只剩一个ELBO项，最后让ELBO项最大。不断迭代直到收敛

*E步：*令 $q^{(t+1)}(z) = p_\theta(z|x)$ ，计算期望：

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{z \sim p_\theta(z|x)} [\log p_\theta(x|z)]$$

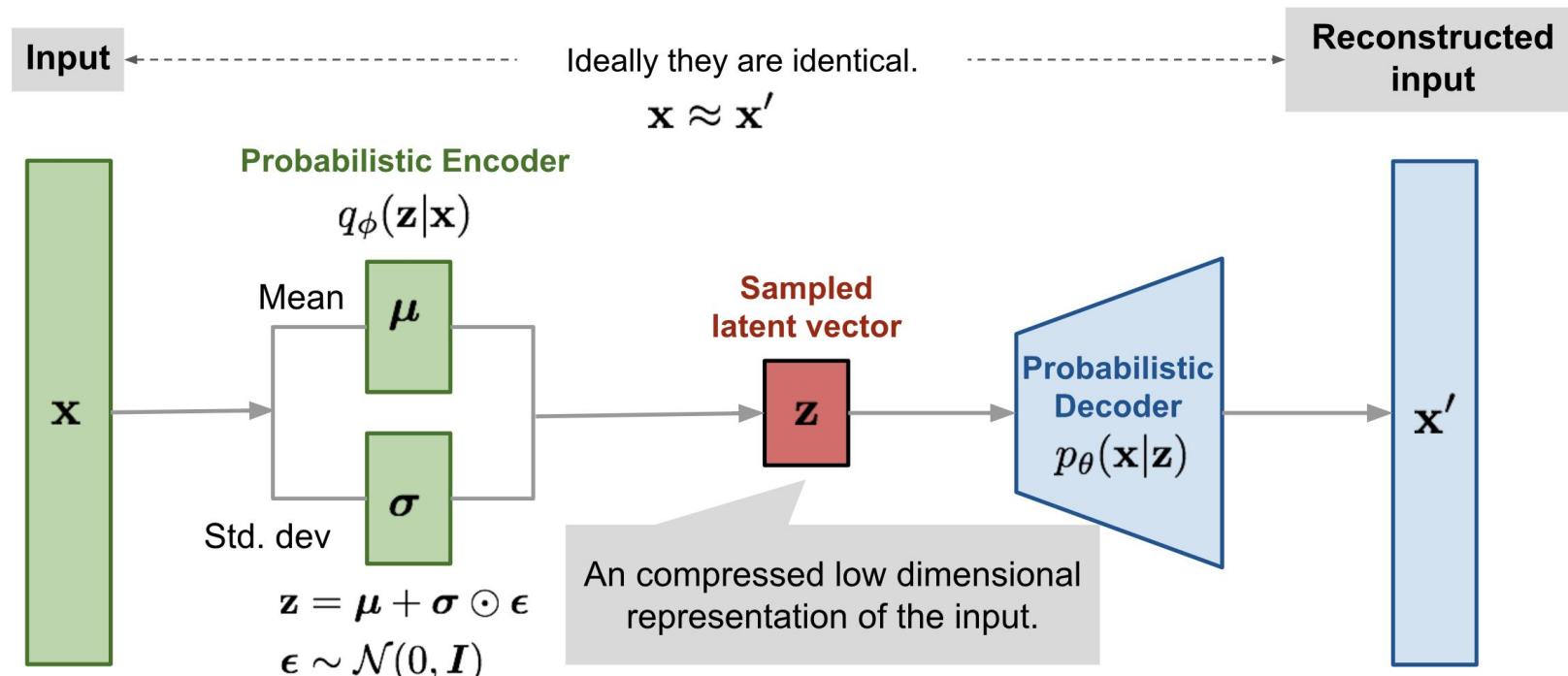
*M步：*更新参数：

$$\boldsymbol{\theta}^{(t+1)} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$$

变分自编码器

$$L(\theta) = \underbrace{\mathbb{E}_{z \sim q(z)} [\log p_\theta(x|z)] - KL(q(z)\|p(z))}_{ELBO} + \underbrace{KL(q(z)\|p_\theta(z|x))}$$

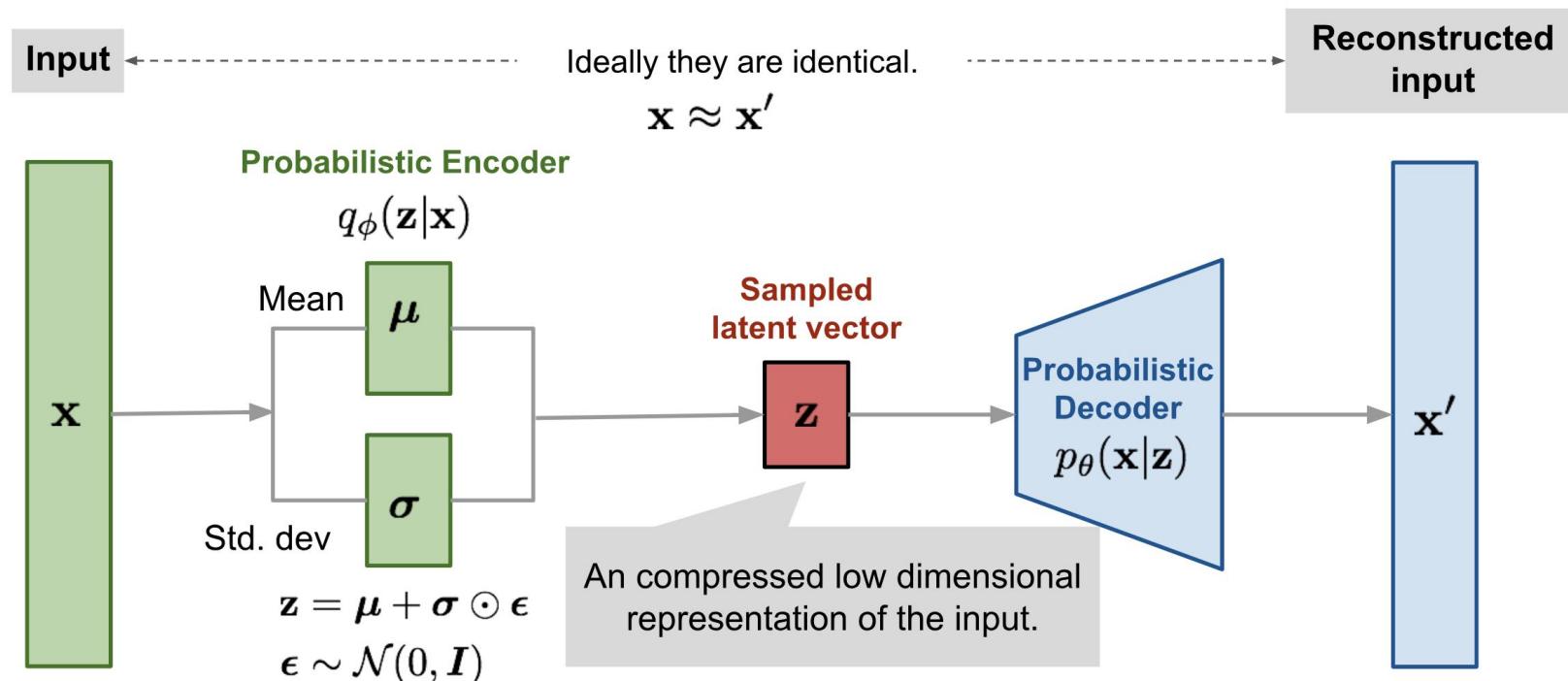
- 第一个改进：合并E-step和M-step，同时优化 $q(z), \theta$ 来达到一步到位 $\max_{q,\theta} ELBO$
- 第二个改进：对于 $q(z)$ 这个很难预测的分布，用深度学习模型 $q_\phi(z|x)$ 来进行控制



变分自编码器

- 变分自编码器的优化目标：

$$L(\theta) = \underbrace{\mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)]}_{\text{重建项}} + \underbrace{KL(q_\phi(z|x) \| p(z))}_{\text{正则项}}$$





扩散模型

主讲人：苏勤亮

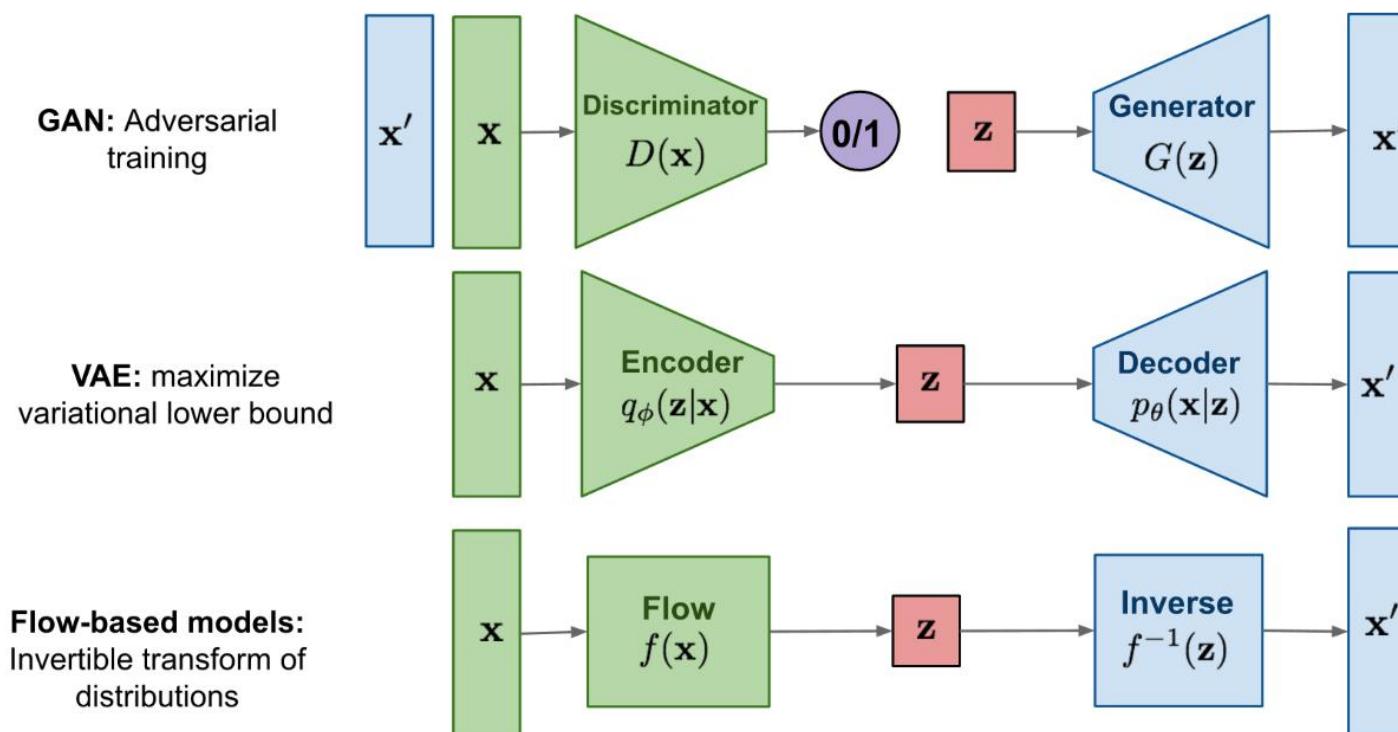


Outline

- 背景
- 扩散模型原理总览
- DDPM原理剖析

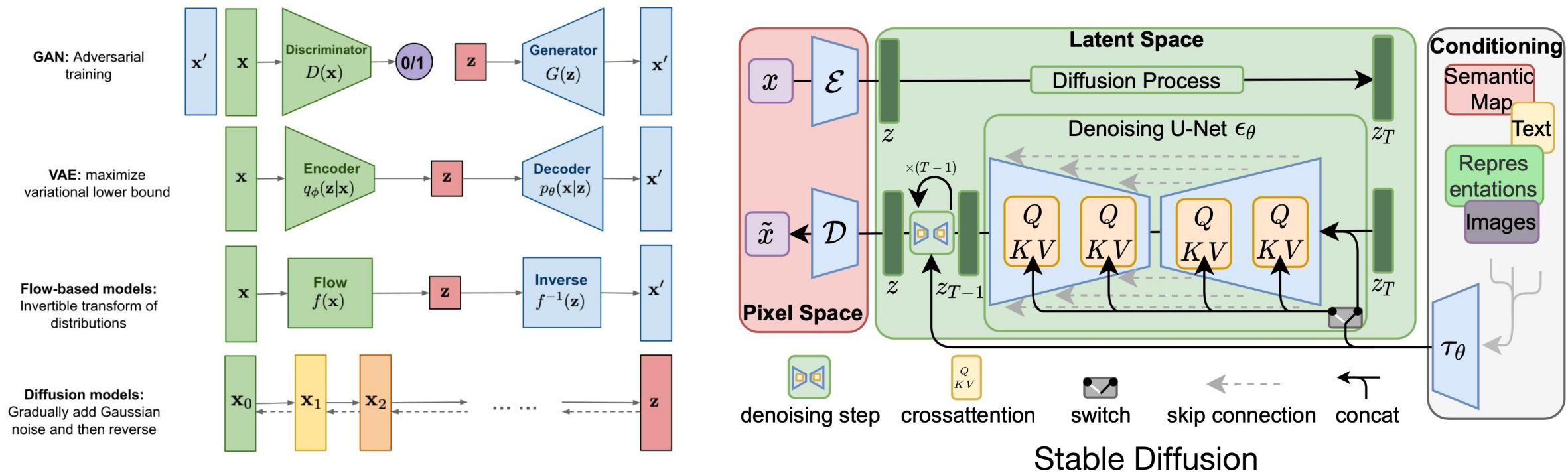
□ 生成模型

- GAN: 训练不稳定, 极难达到纳什均衡, 对超参数敏感, 训练容易崩塌
- VAE: 生成图像模糊, 均方误差和变分近似导致生成的细节丢失, 清晰度不如GAN
- Flow-based Models: 需要专门设计的可逆网络结构, 参数量巨大, 生成高维数据困难



□ 扩散模型的优势

- **训练稳定性**: 不同于GAN的对抗训练，扩散模型的训练本质上是回归问题
- **生成质量**: 扩散模型的生成质量已超越GAN，能够生成极其精细的纹理和复杂的场景
- **可扩展性**: DALL-E 3, Stable Diffusion, Sora 等现代AIGC应用的基石



Outline

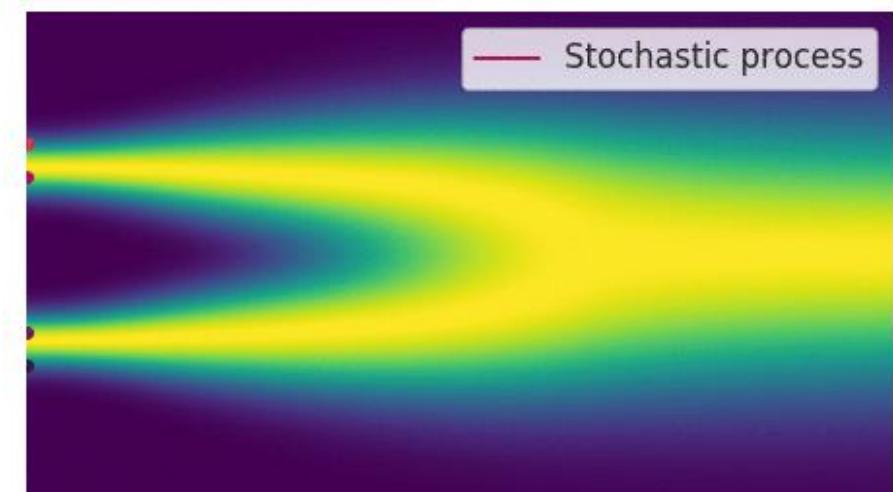
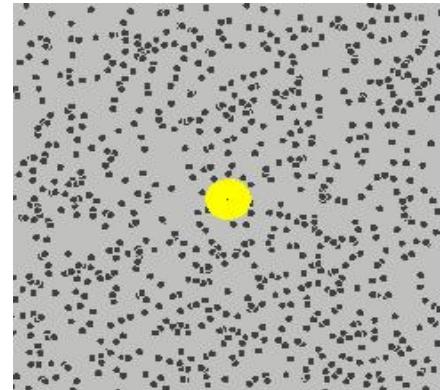
- 背景
- 扩散模型原理总览
- DDPM原理剖析

扩散过程

- 物理学中，扩散过程用于描述分子从高浓度区域向低浓度区域自发迁移、最终达到动态平衡的过程
- 统计学中，扩散过程指描述随机变量随时间演化的过程

$$X_{t+\Delta t} - X_t = \mu(X_t, t)\Delta t + \sigma(X_t, t)\Delta t \cdot \epsilon_t$$

- ϵ_t : 高斯随机噪声 $\epsilon_t \sim \mathcal{N}(0, I)$
- $\mu(X_t, t)$: 漂移项，表示系统的平均趋势



图像扩散过程

1000步加噪过程



给定图像 \mathbf{x} ，对其逐渐添加高斯噪声，同时减弱其强度

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t \quad \text{其中 } \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$$

其中， β_t 表示每次加入噪声的强度，值非常小，如：0.01； $t = 1, 2, \dots, 1000$

想一想，随着时间的增加， \mathbf{x}_t 会逐渐变成什么？

高斯噪声，即： $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$

扩散模型的基本原理

加噪过程



$$\text{前向扩散过程: } \mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t$$

训练一个模型，用于近似扩散过程的逆过程

去噪过程



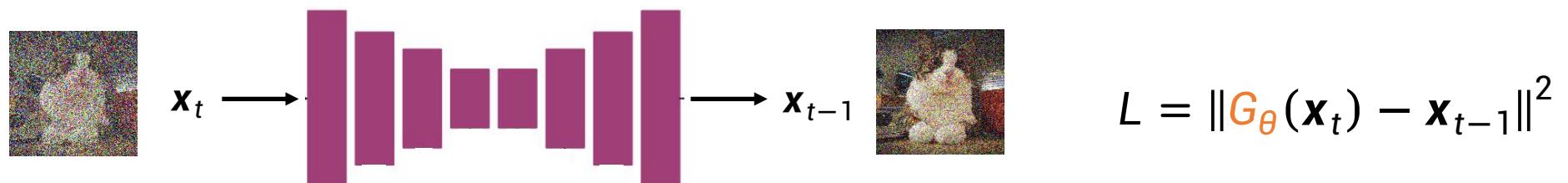
$$\text{近似逆扩散过程: } \hat{\mathbf{x}}_{t-1} = G_{\theta}(\mathbf{x}_t)$$

扩散模型的启发式训练方法 (I)



近似逆扩散过程: $\hat{\mathbf{x}}_{t-1} = G_{\theta}(\mathbf{x}_t)$

□ 思路一: 训练一个神经网络, 用 \mathbf{x}_t 直接预测 \mathbf{x}_{t-1}



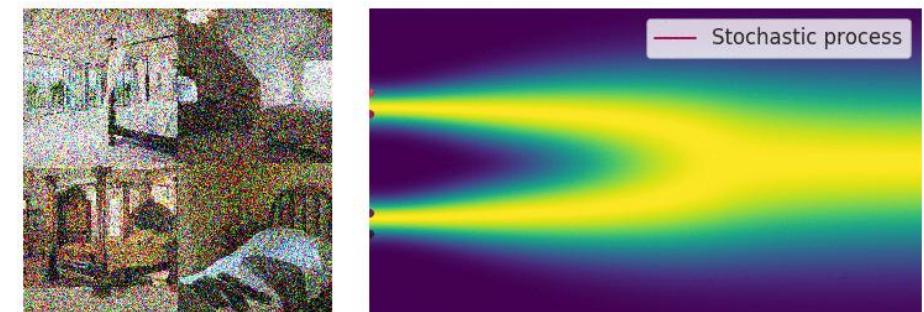
存在的问题: 没有利用扩散过程蕴含的结构信息, 模型被主要用于预测扩散过程中不可预测的随机性, 导致最后效果不好

扩散模型的启发式训练方法 (II)

□ 思路二：充分利用扩散过程所蕴含的结构信息

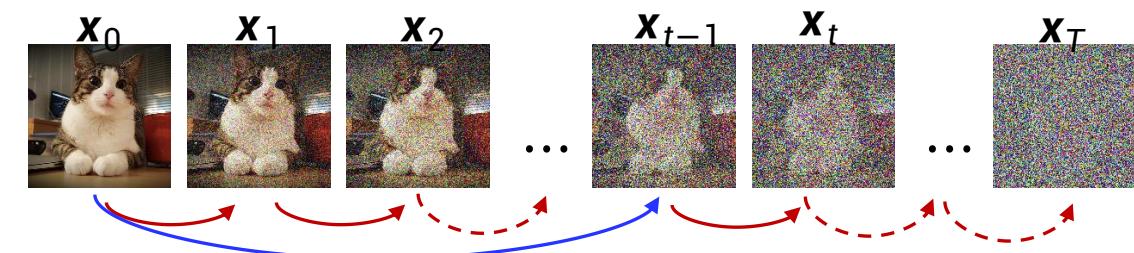
- 扩散过程 $\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t$ 实质上描述的是一个马尔科夫随机过程

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t I)$$



- 0到 $t-1$ 时刻的条件转移概率

$$q(\mathbf{x}_{t-1} | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{a}_{t-1}} \mathbf{x}_0, (1 - \bar{a}_{t-1}) I)$$



其中 $\bar{a}_{t-1} \triangleq \prod_{s=1}^{t-1} (1 - \beta_s)$

扩散模型的启发式训练方法 (III)

扩散过程只指定了如下前向加噪过程



$$q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{a}_{t-1}}\mathbf{x}_0, (1 - \bar{a}_{t-1})I)$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t I)$$

逆扩散过程：根据贝叶斯定理，在 t 时刻的状态 \mathbf{x}_t 已知时， $t - 1$ 时刻状态 \mathbf{x}_{t-1} 分布为

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0) \times q(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_0)} = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I)$$

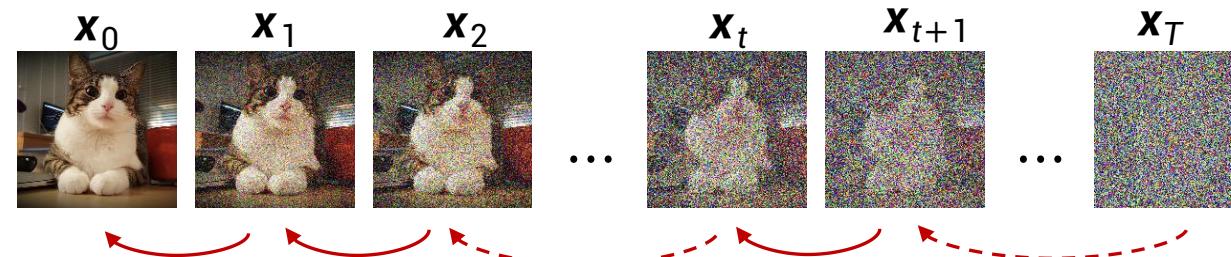
其中 $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1 - \bar{a}_t} \mathbf{x}_0 + \frac{\sqrt{1 - \beta_t}(1 - \bar{a}_{t-1})}{1 - \bar{a}_t} \mathbf{x}_t$ $\tilde{\beta}_t = \frac{1 - \bar{a}_{t-1}}{1 - \bar{a}_t} \beta_t$

扩散模型的启发式训练方法 (IV)

后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t I)$ 揭示了扩散过程逆过程的采用过程

$$\mathbf{x}_{t-1} = \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) + \tilde{\boldsymbol{\beta}}_t \cdot \boldsymbol{\epsilon}_t$$

- $\boldsymbol{\epsilon}_t$: 高斯随机噪声 $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$



思考：是否可以使用上述采用过程来生成数据呢？

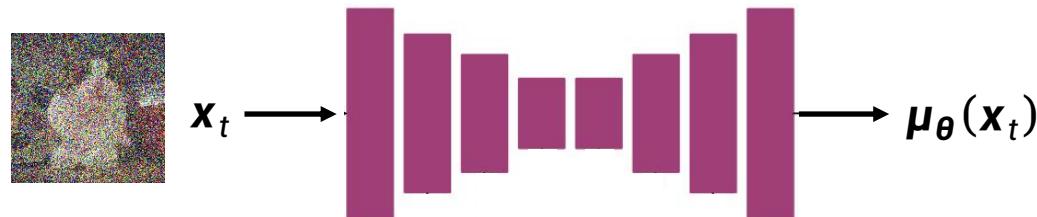
不可以， $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ 含数据 \mathbf{x}_0 ，而在数据生成阶段，只有 \mathbf{x}_t 可使用

如何解决？



扩散模型的启发式训练方法 (V)

解决思路：训练一个输入为 x_t 的神经网络，要求其输出值 $\mu_\theta(x_t)$ 尽可能逼近 $\tilde{\mu}_t(x_t, x_0)$



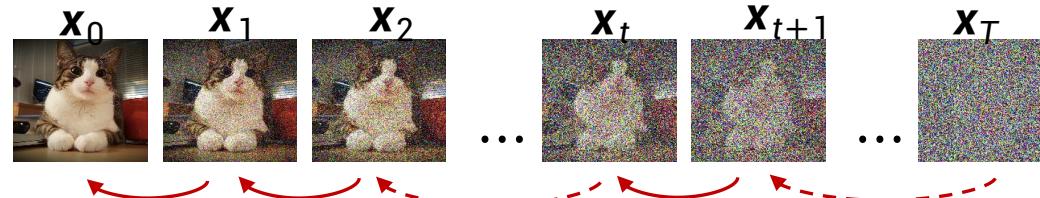
$$\text{Loss} = \sum_{x_0 \in \mathcal{D}} \|\mu_\theta(x_t) - \tilde{\mu}_t(x_t, x_0)\|^2$$

用 $\mu_\theta(x_t)$ 替换 $\tilde{\mu}_t(x_t, x_0)$ 实现数据的采样

$$x_{t-1} = \tilde{\mu}_t(x_t, x_0) + \tilde{\beta}_t \cdot \epsilon_t$$



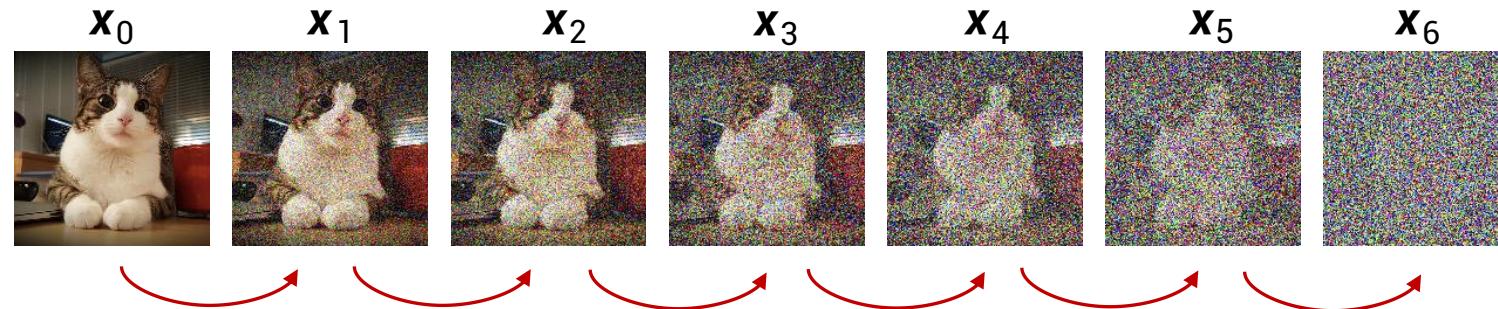
$$x_{t-1} = \mu_\theta(x_t) + \tilde{\beta}_t \cdot \epsilon_t$$



Outline

- 背景
- 扩散模型原理总览
- DDPM原理剖析

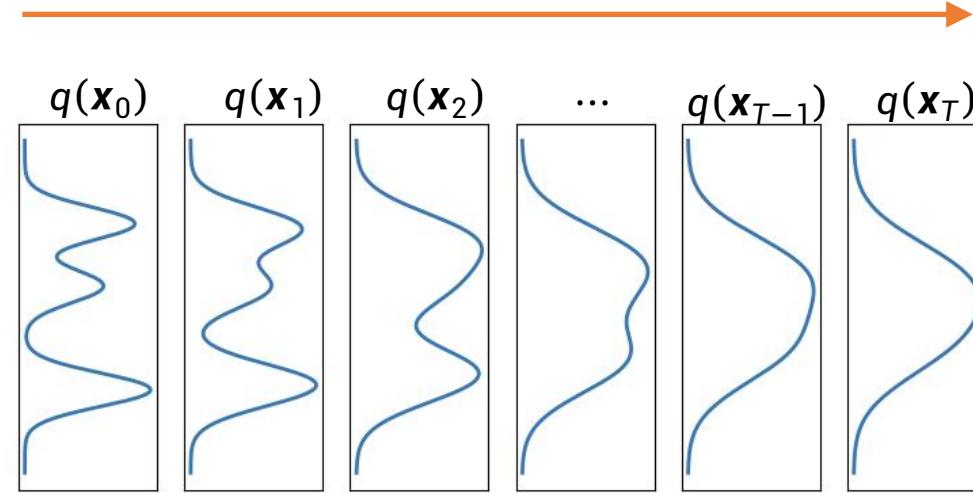
前向扩散过程



- 假设初始的数据分布为 $q(\mathbf{x}_0) = p_{data}(\mathbf{x})$
- 扩散过程可以由转移概率分布来描述

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

其中 β_t 通常是一个非常小的值，例如 0.01。



□ 若已知 \mathbf{x}_{t-1} ，则样本 $\mathbf{x}_t \sim q(\mathbf{x}_t | \mathbf{x}_{t-1})$ 可通过以下采样过程来获得：

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t \quad \text{其中 } \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$$

问题 1：为什么我们要在 \mathbf{x}_{t-1} 的前面乘上系数 $\sqrt{1 - \beta_t}$? (防止方差爆炸)

问题 2：当 T 趋于无穷大时， $q(\mathbf{x}_T)$ 是什么分布? (正态分布)

□ 若已知 x_{t-1} ，则样本 $x_t \sim q(x_t|x_{t-1})$ 可通过以下采样过程来获得：

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t \quad \text{其中 } \epsilon_t \sim \mathcal{N}(0, I)$$

问题 1：为什么我们要在 x_{t-1} 的前面乘上系数 $\sqrt{1 - \beta_t}$ ？

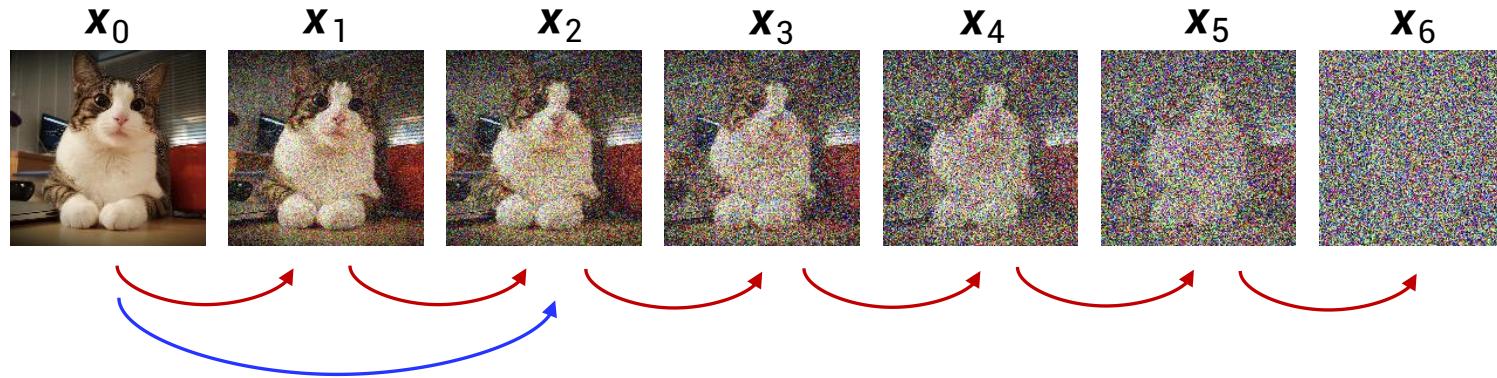
在扩散模型的前向过程中，我们在每一步都向图像中添加高斯噪声。为了确保数据在经过多次加噪后数值范围不会无限扩大（即防止方差爆炸），我们需要对原始信号 x_{t-1} 进行缩放

假设 x_{t-1} 的方差已经归一化为 1，如果我们希望 x_t 的方差保持为 1，计算如下

$$\begin{aligned} \text{Var}(x_t) &= \text{Var}(\sqrt{1 - \beta_t} x_{t-1}) + \text{Var}(\sqrt{\beta_t} \epsilon_t) \\ &= (\sqrt{1 - \beta_t})^2 \text{Var}(x_{t-1}) + (\sqrt{\beta_t})^2 \text{Var}(\epsilon_t) \\ &= (1 - \beta_t) \cdot 1 + \beta_t \cdot 1 = 1 \end{aligned}$$

系数 $\sqrt{1 - \beta_t}$ 的作用是抵消加入噪声 $\sqrt{\beta_t} \epsilon_t$ 后带来的方差增加。如果不乘这个系数，随着 t 的增加，图像的像素值方差会越来越大（方差爆炸）

$$\text{已知 } q(\mathbf{x}_1|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_1; \sqrt{1-\beta_1}\mathbf{x}_0, \beta_1\mathbf{I})$$



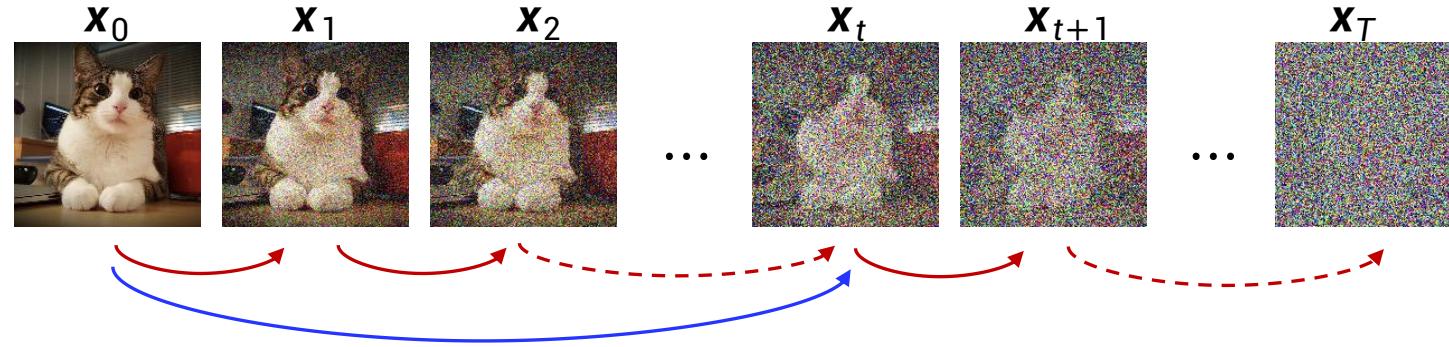
分布 $q(\mathbf{x}_2|\mathbf{x}_0)$ 的形式是什么？

$$\begin{aligned}\mathbf{x}_2 &= \sqrt{1-\beta_2}\mathbf{x}_1 + \sqrt{\beta_2}\boldsymbol{\epsilon}_2 \\ &= \sqrt{1-\beta_2}(\sqrt{1-\beta_1}\mathbf{x}_0 + \sqrt{\beta_1}\boldsymbol{\epsilon}_1) + \sqrt{\beta_2}\boldsymbol{\epsilon}_2 \\ &= \sqrt{(1-\beta_2)(1-\beta_1)}\mathbf{x}_0 + \sqrt{(1-\beta_2)\beta_1}\boldsymbol{\epsilon}_1 + \sqrt{\beta_2}\boldsymbol{\epsilon}_2\end{aligned}$$

$$\rightarrow \mathbb{E}[\mathbf{x}_2] = \sqrt{(1-\beta_2)(1-\beta_1)}\mathbf{x}_0 \quad \text{Var}[\mathbf{x}_2] = 1 - (1-\beta_2)(1-\beta_1)$$

$$\rightarrow q(\mathbf{x}_2|\mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_2; \sqrt{(1-\beta_1)(1-\beta_2)}\mathbf{x}_0, (1-(1-\beta_2)(1-\beta_1))I \right)$$

那么分布 $q(\mathbf{x}_t|\mathbf{x}_0)$ 是什么形式?



$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_t; \sqrt{\prod_{s=1}^t (1-\beta_s)} \mathbf{x}_0, \left(1 - \prod_{s=1}^t (1-\beta_s) \right) I \right)$$

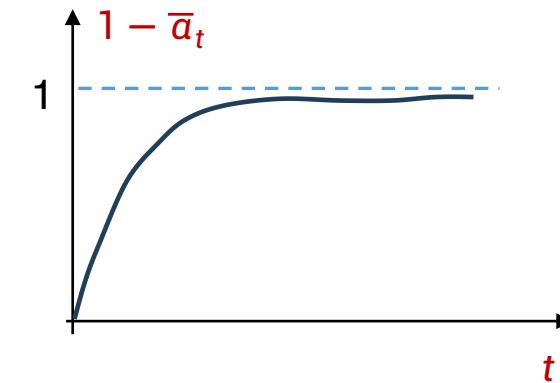
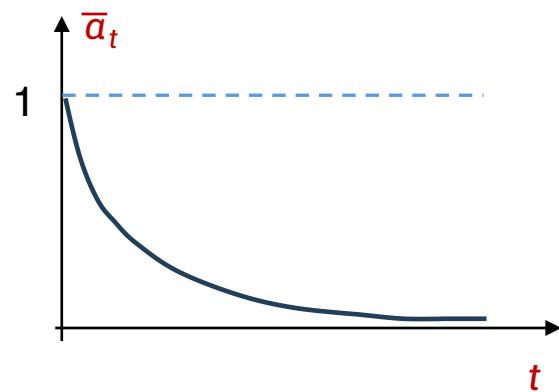
$$= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{a}_t} \mathbf{x}_0, (1 - \bar{a}_t) I) \quad \text{其中 } \bar{a}_t \triangleq \prod_{s=1}^t (1 - \beta_s)$$

➡ $\mathbf{x}_t = \sqrt{\bar{a}_t} \mathbf{x}_0 + \sqrt{1 - \bar{a}_t} \epsilon$ 其中 $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$

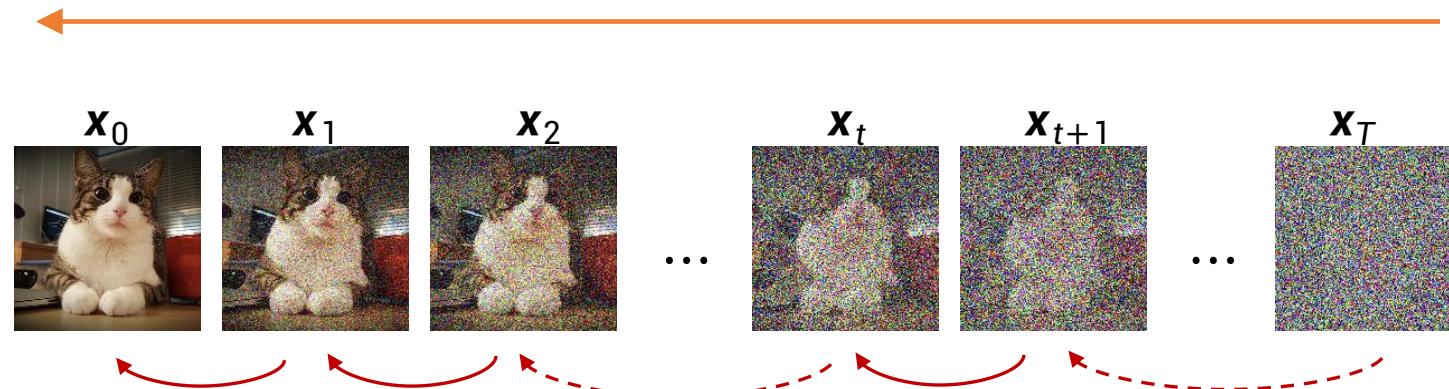
\bar{a}_t 的含义是什么？

$1 - \bar{a}_t$ 代表图像 \mathbf{x}_t 中噪声的强度（方差）

\bar{a}_t 代表图像 \mathbf{x}_t 中原图像 \mathbf{x}_0 的强度



通过逆转扩散过程进行生成



扩散过程规定了一个联合分布：

$$q(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T) = q(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)\cdots q(\mathbf{x}_t|\mathbf{x}_{t-1})\cdots q(\mathbf{x}_T|\mathbf{x}_{T-1})$$

由于其马尔可夫结构，该分布可以等价地以逆形式表示

$$q(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T) = q(\mathbf{x}_T)q(\mathbf{x}_{T-1}|\mathbf{x}_T)\cdots q(\mathbf{x}_{t-1}|\mathbf{x}_t)\cdots q(\mathbf{x}_0|\mathbf{x}_1)$$

$$q(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T) = q(\mathbf{x}_T)q(\mathbf{x}_{T-1}|\mathbf{x}_T)\cdots q(\mathbf{x}_{t-1}|\mathbf{x}_t)\cdots q(\mathbf{x}_0|\mathbf{x}_1)$$

已知当 $T \rightarrow +\infty$ 时， $q(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, I)$ 。因此，为了从分布 $q(\mathbf{x}_0)$ 中采集样本，我们可以通过以下过程进行采样：

- 1) 初始化 $\mathbf{x}_T \sim q(\mathbf{x}_T)$
- 2) 采样 $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 对于 $t = T, T-1, \dots, 1$
- 3) 保留 \mathbf{x}_0 ，同时舍弃样本 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$

问题是如何得到转移概率分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

- 但是 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 是未知的，因为分布 $q(\mathbf{x}_0)$ 是未知的

$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 分布

- 尽管 $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 未知，但在进一步给定 \mathbf{x}_0 条件下，我们可以得到分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$
- 根据 $q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{a}_{t-1}}\mathbf{x}_0, (1 - \bar{a}_{t-1})I)$ 及 $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t I)$ ，我们可以得到联合分布：

$$q(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t I) \cdot \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{a}_{t-1}}\mathbf{x}_0, (1 - \bar{a}_{t-1})I)$$

- 从该联合概率密度函数中，我们可以得到后验分布（推导过程见后页补充材料）：

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I)$$

其中 $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1 - \bar{a}_t} \mathbf{x}_0 + \frac{\sqrt{1 - \beta_t}(1 - \bar{a}_{t-1})}{1 - \bar{a}_t} \mathbf{x}_t, \quad \tilde{\beta}_t = \frac{1 - \bar{a}_{t-1}}{1 - \bar{a}_t} \beta_t$

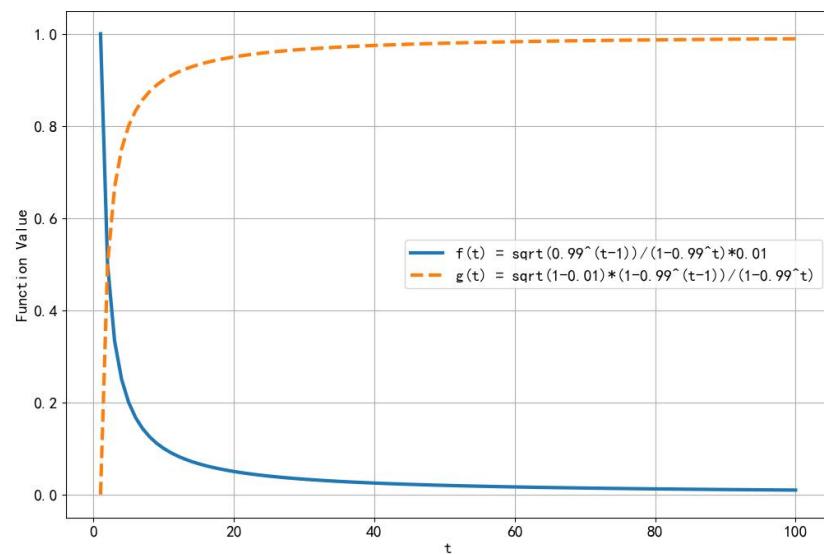
因此，后验分布的形式为

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t I)$$

其中

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t}\mathbf{x}_t \quad \tilde{\beta}_t = \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t$$

- 下图显示了当设置 $\beta_t = 0.01$ 时，系数 $\frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t}$ 和 $\frac{\sqrt{1-\beta_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t}$ 随 t 变化的函数关系



- 当 $t \geq 20$ 时，后验均值 $\tilde{\boldsymbol{\mu}}_t$ 主要由 \mathbf{x}_t 决定
- 当 $t < 20$ 时，后验均值 $\tilde{\boldsymbol{\mu}}_t$ 由 \mathbf{x}_t 和 \mathbf{x}_0 共同决定

补充材料：使用“完全平方”技巧推导后验概率

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &\propto \exp\left\{-\frac{1}{2\beta_t}\left(\mathbf{x}_t - \sqrt{1-\beta_t}\mathbf{x}_{t-1}\right)^2\right\} \cdot \exp\left\{-\frac{1}{2(1-\bar{a}_{t-1})}\left(\mathbf{x}_{t-1} - \sqrt{\bar{a}_{t-1}}\mathbf{x}_0\right)^2\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left[\left(\frac{1-\beta_t}{\beta_t} + \frac{1}{1-\bar{a}_{t-1}}\right)\mathbf{x}_{t-1}^2 - 2\left(\frac{\sqrt{1-\beta_t}\mathbf{x}_t}{\beta_t} + \frac{\sqrt{\bar{a}_{t-1}}\mathbf{x}_0}{1-\bar{a}_{t-1}}\right)\mathbf{x}_{t-1}\right]\right\} \end{aligned}$$

→ 后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 的方差和均值为：

$$Var(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\frac{1-\beta_t}{\beta_t} + \frac{1}{1-\bar{a}_{t-1}}} = \boxed{\frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t}$$

$$E[\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0] = \frac{\frac{\sqrt{1-\beta_t}\mathbf{x}_t}{\beta_t} + \frac{\sqrt{\bar{a}_{t-1}}\mathbf{x}_0}{1-\bar{a}_{t-1}}}{\frac{1-\beta_t}{\beta_t} + \frac{1}{1-\bar{a}_{t-1}}} = \boxed{\frac{\sqrt{1-\beta_t}(1-\bar{a}_{t-1})\mathbf{x}_t}{1-\bar{a}_t} + \frac{\sqrt{\bar{a}_{t-1}}\beta_t\mathbf{x}_0}{1-\bar{a}_t}}$$

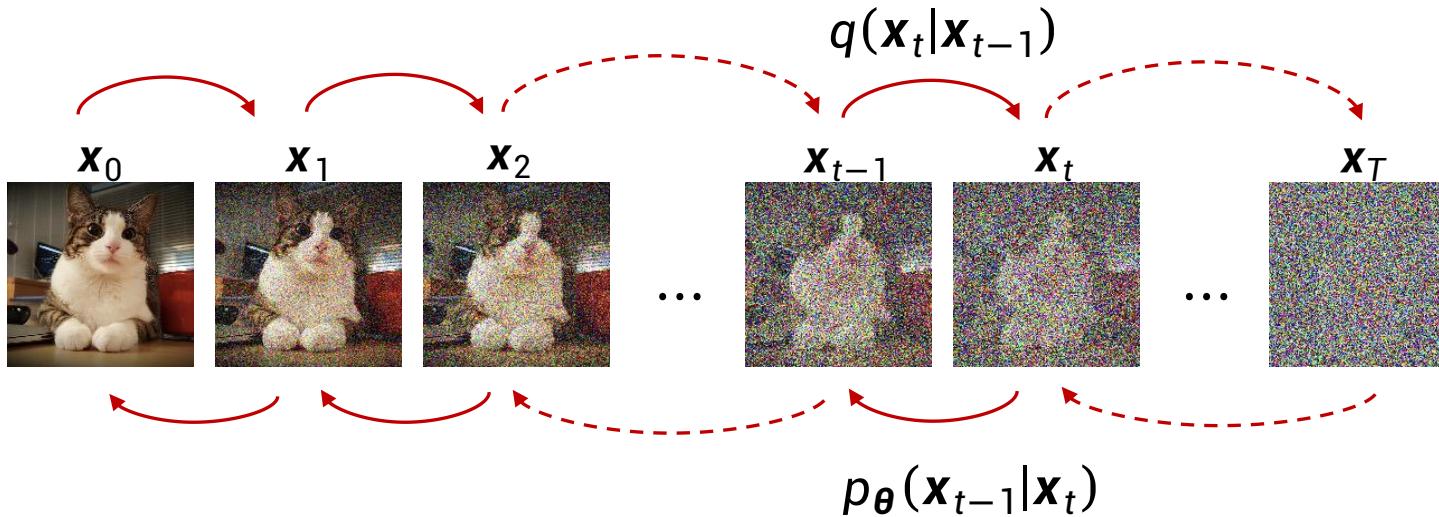
一种启发式生成方法

- 由于无法获得真实图像 \mathbf{x}_0 ，我们不能使用真实的后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ 来生成图像
- 一个直观的想法
 - 对于所有 $\mathbf{x}_0 \sim q(\mathbf{x})$ ，学习一个分布 $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 来近似真实的后验分布 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ ，其中 $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 设定为如下形式
$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I})$$
 - 使用 $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 来生成图像
- 为最小化 $KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))$ ，可以验证等价于 $\min \|\boldsymbol{\mu}_{\theta}(\mathbf{x}_t) - \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)\|^2$

能否更严谨一点？



一个更严谨的视角



将 $t \geq 1$ 时的 \mathbf{x}_t 视为隐变量，扩散生成模型就是一个多层隐变量模型，其分布为

$$p_{\theta}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T) = p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) \cdots p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdots p_{\theta}(\mathbf{x}_{T-1}|\mathbf{x}_T) p(\mathbf{x}_T)$$

其中 $p(\mathbf{x}_0|\mathbf{z}) \triangleq p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)$ 并且 $p(\mathbf{z}) \triangleq p_{\theta}(\mathbf{x}_1|\mathbf{x}_2) \cdots p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdots p_{\theta}(\mathbf{x}_{T-1}|\mathbf{x}_T) p(\mathbf{x}_T)$

- 对于一个生成模型，一个广泛使用的训练目标是最大化对数似然

$$\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}_0 \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_0)$$

或其变分下界（仅考虑单个样本 \mathbf{x}_0 ）

$$\begin{aligned} \mathcal{L} &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_0) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_0, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_0)} d\mathbf{z} \\ &= \int q_{\boldsymbol{\phi}}(\mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{x}_0) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1) \cdots p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdots p_{\boldsymbol{\theta}}(\mathbf{x}_{T-1}|\mathbf{x}_T) p(\mathbf{x}_T)}{q_{\boldsymbol{\phi}}(\mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{x}_0)} d\mathbf{z} \end{aligned}$$

其中隐变量 $\mathbf{z} \triangleq \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$

- 直接将近似后验分布 $q(\mathbf{x}_1, \dots, \mathbf{x}_T | \mathbf{x}_0)$ 设定为如下扩散过程的分布

$$q(\mathbf{x}_1, \dots, \mathbf{x}_T | \mathbf{x}_0) = q(\mathbf{x}_1 | \mathbf{x}_0) \cdots q(\mathbf{x}_t | \mathbf{x}_{t-1}) \cdots q(\mathbf{x}_T | \mathbf{x}_{T-1}),$$

- 变分下界可以写成如下形式

$$\mathcal{L} = \int q(\mathbf{x}_1 | \mathbf{x}_0) \cdots q(\mathbf{x}_T | \mathbf{x}_{T-1}) \log \frac{p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1) \cdots p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) \cdots p_{\theta}(\mathbf{x}_{T-1} | \mathbf{x}_T) p(\mathbf{x}_T)}{q(\mathbf{x}_1 | \mathbf{x}_0) \cdots q(\mathbf{x}_T | \mathbf{x}_{T-1})} d\mathbf{x}$$

- 这里，后验分布是固定且不可学习的，我们只能学习生成模型中的参数 θ

跟之前学习的VAE有什么区别？

变分下界1的表达式

- 通过一定概率变换操作，可推导变分下界的表达式为（具体见[下页补充材料](#)）：

$$l = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}[KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t))]}_{l_{t-1}} - KL(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))$$

- 将 $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I})$ 和 $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \tilde{\boldsymbol{\beta}}_t \mathbf{I})$ 代入 KL 散度，可以得到

$$l_{t-1} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|^2 \right] + C$$

- 因此，最大化变分下界等价于最大化

$$\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^{T-1} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|^2 \right]$$

- 该目标与前述启发式训练目标（最小化 $\|\mu_{\boldsymbol{\theta}}(\mathbf{x}_t) - \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)\|^2$ ）非常类似，除了两点区别
 - 直观方法不包含数据项 $\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)]$
 - 直观方法没有指定如何选择 \mathbf{x}_t ，而变分下界目标明确要求 \mathbf{x}_t 来自于 $q(\mathbf{x}_t|\mathbf{x}_0)$ ，即 $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$

补充材料：变分下界的推导过程

- 重写后验分布

$$\begin{aligned} q(\mathbf{x}_1, \dots, \mathbf{x}_T | \mathbf{x}_0) &= q(\mathbf{x}_1 | \mathbf{x}_0) \cdots q(\mathbf{x}_t | \mathbf{x}_{t-1}) \cdots q(\mathbf{x}_T | \mathbf{x}_{T-1}) \\ &= q(\mathbf{x}_T | \mathbf{x}_0) q(\mathbf{x}_{T-1} | \mathbf{x}_T, \mathbf{x}_0) \cdots q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \cdots q(\mathbf{x}_1 | \mathbf{x}_2, \mathbf{x}_0) \end{aligned}$$

$q(\mathbf{x}_T | \mathbf{x}_0)$ 和 $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ 的分布是什么？

$$\begin{aligned} q(\mathbf{x}_T | \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_T; \sqrt{\bar{a}_T} \mathbf{x}_0, (1 - \bar{a}_T) \mathbf{I}) \text{ 其中 } \bar{a}_T \triangleq \prod_{s=1}^T (1 - \beta_s) \\ &\approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \end{aligned}$$

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I})$$

补充材料：变分下界的推导过程 Cont'

将后验分布代入变分下界，我们得到

$$L = \int q(\mathbf{x}_1|\mathbf{x}_2, \mathbf{x}_0) \cdots q(\mathbf{x}_{T-1}|\mathbf{x}_T, \mathbf{x}_0) q(\mathbf{x}_T|\mathbf{x}_0) \times \log \frac{p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)p_{\theta}(\mathbf{x}_1|\mathbf{x}_2)\cdots p_{\theta}(\mathbf{x}_{T-1}|\mathbf{x}_T)p(\mathbf{x}_T)}{q(\mathbf{x}_1|\mathbf{x}_2, \mathbf{x}_0) \cdots q(\mathbf{x}_{T-1}|\mathbf{x}_T, \mathbf{x}_0) q(\mathbf{x}_T|\mathbf{x}_0)} d\mathbf{x}_1 \cdots d\mathbf{x}_T$$

$$= \int q(\mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{x}_0) \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) d\mathbf{X} \quad \Leftrightarrow \quad \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)]$$

$$- \sum_{t=2}^T \int q(\mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{x}_0) \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)} d\mathbf{X} \quad \Leftrightarrow \quad \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))]$$

$$- \int q(\mathbf{x}_1, \dots, \mathbf{x}_T|\mathbf{x}_0) \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p(\mathbf{x}_T)} d\mathbf{X} \quad \Leftrightarrow \quad KL(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))$$

- 因此，变分下界可以写为

$$L = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [KL(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))] - KL(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))$$

重写 $\tilde{\mu}_t(x_t, x_0)$ 的表达式

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t}x_0 + \frac{\sqrt{1-\beta_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t}x_t$$

- 回顾 $x_t = \sqrt{\bar{a}_t}x_0 + \sqrt{1-\bar{a}_t}\epsilon$ 其中 $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$ 。将 $x_0 = \frac{x_t - \sqrt{1-\bar{a}_t}\epsilon}{\sqrt{\bar{a}_t}}$ 代入 $\tilde{\mu}_t(x_t, x_0)$ 得到

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon \right)$$

- 因此，我们可以将估计的均值 $\mu_{\theta}(x_t, t)$ 显式地约束为如下形式

$$\mu_{\theta}(x_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon_{\theta}(x_t, t) \right)$$

- 也就是说，为估计 $\tilde{\mu}_t$ ，我们只需训练一个神经网络 $\epsilon_{\theta}(x_t, t)$ 来估计所添加的噪声 ϵ

重新表示 l_{t-1}

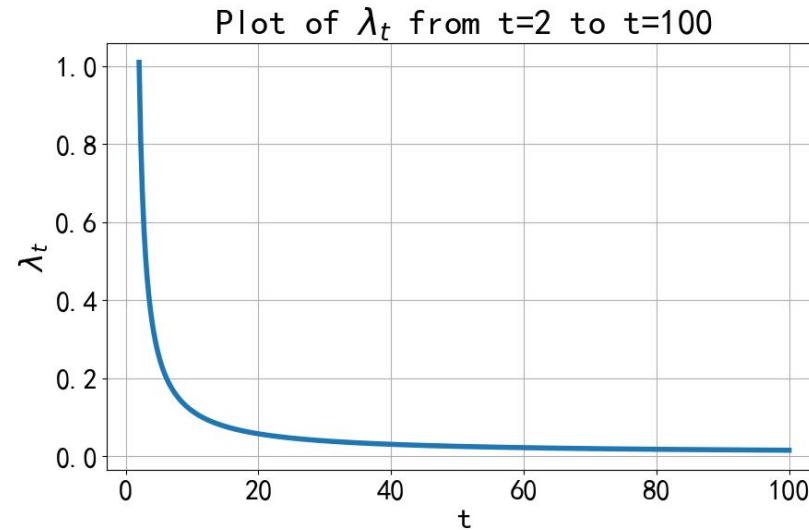
- 将 $\mu_\theta(x_t, t)$ 和 $\tilde{\mu}_t(x_t, x_0)$ 的表达式代入 l_{t-1} 得到

$$l_{t-1} = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\lambda_t \left\| \epsilon - \epsilon_\theta \left(\underbrace{\sqrt{\bar{a}_t} x_0 + \sqrt{1 - \bar{a}_t} \epsilon, t}_{x_t} \right) \right\|^2 \right] + C$$

其中 $\lambda_t \triangleq \frac{\beta_t^2}{2\tilde{\beta}_t(1-\beta_t)(1-\bar{a}_t)} = \frac{\beta_t}{(1-\beta_t)(1-\bar{a}_{t-1})}$

当最小化损失函数 $\tilde{l} = \sum_{t=1}^T l_{t-1}$ 时，我们实际上是在尝试最大化对数似然 $\log p_\theta(x_0)$

系数 λ_t 过分重视了较小的 t 值

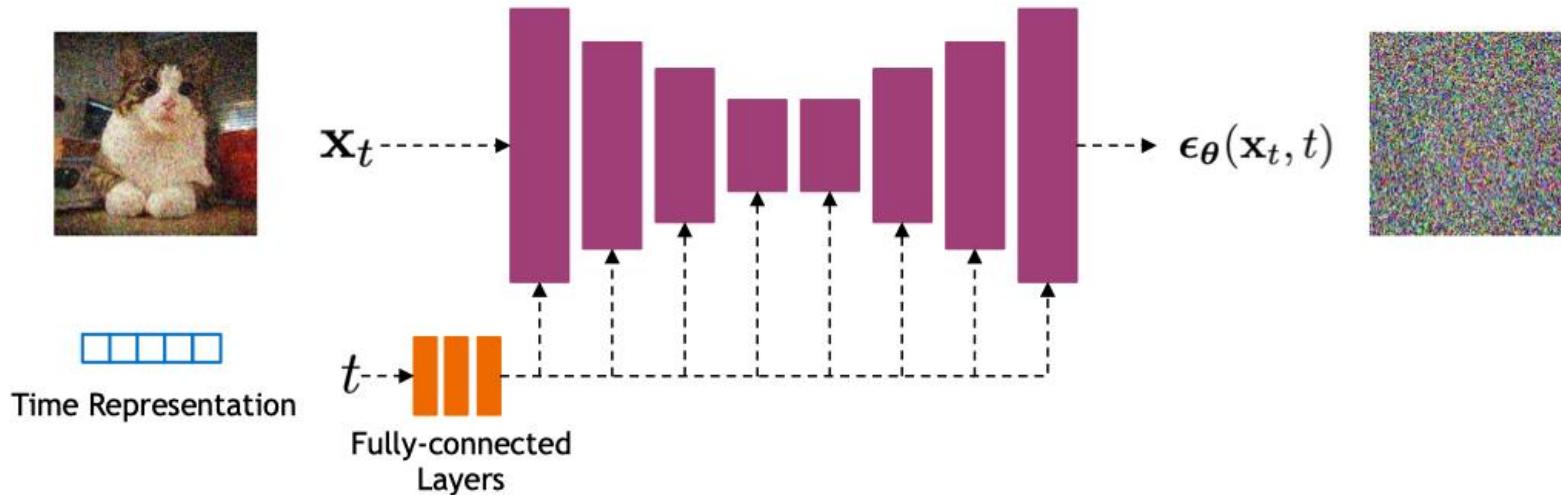


研究发现，通过简单地将所有 t 的 λ_t 设置为 1，我们可以生成更高质量的样本。
也就是说，最大化以下损失函数

$$l_{simple} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, I), t \sim U(1, T)} \left[\|\epsilon - \epsilon_{\theta}(\sqrt{\bar{a}_t} \mathbf{x}_0 + \sqrt{1 - \bar{a}_t} \epsilon, t)\|^2 \right]$$

$\epsilon_{\theta}(x_t, t)$ 的神经网络架构

使用包含 ResNet 模块的 U-Net 架构来表示 $\epsilon_{\theta}(x_t, t)$



- 时间步编码：正弦函数位置编码或随机傅里叶特征
- 时间特征通过空间加法或自适应组归一化（AGN）输入到残差块中

训练过程

Algorithm 1 Training

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
      
$$\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$$

6: until converged
```

采样过程

Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sqrt{\tilde{\beta}_t} \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

课堂小结

- 扩散模型原理总览
- DDPM原理剖析