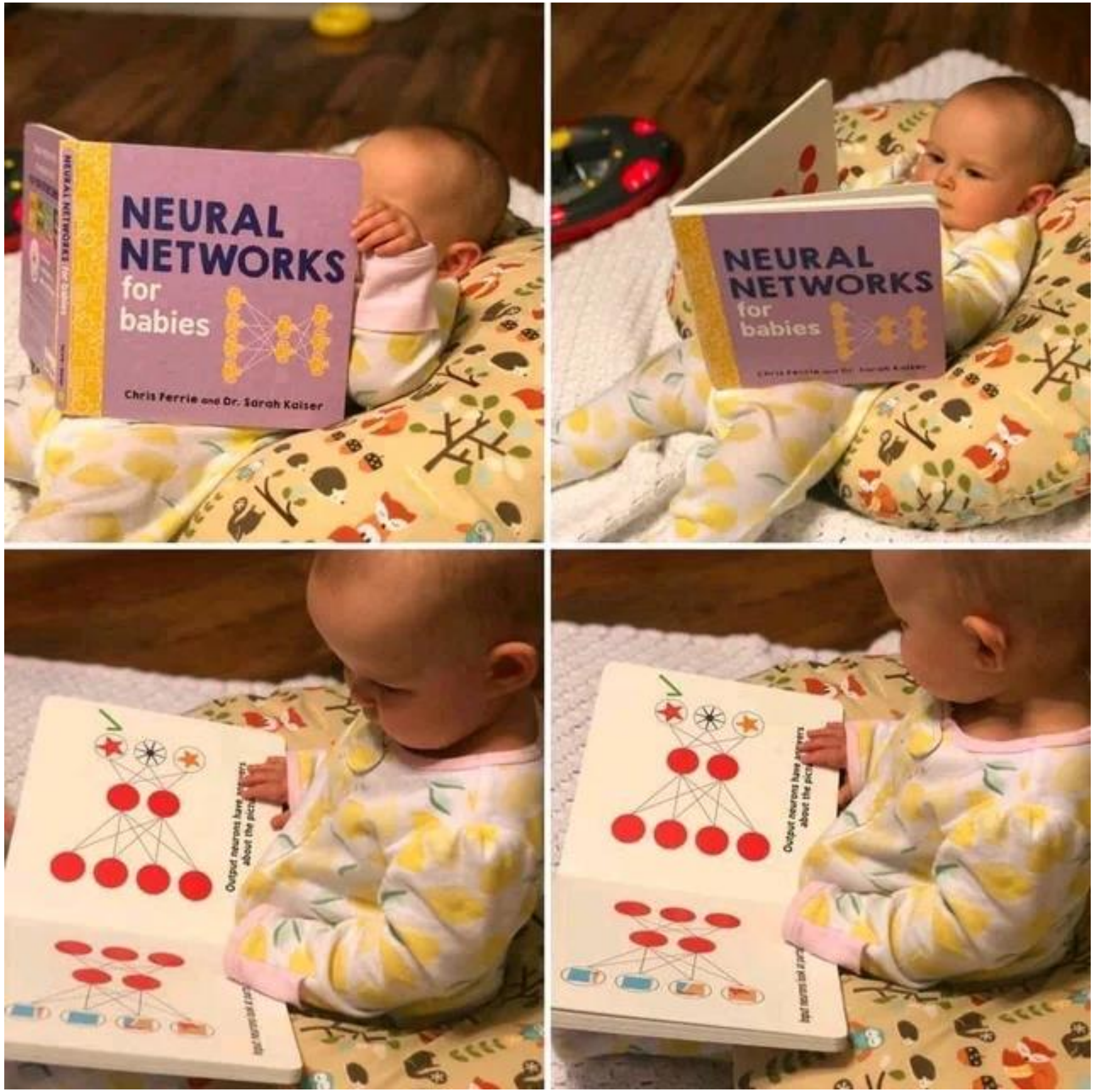
微生物学实验

体验机器学习：无监督算法和有监督算法

前沿交叉学科研究院 高开

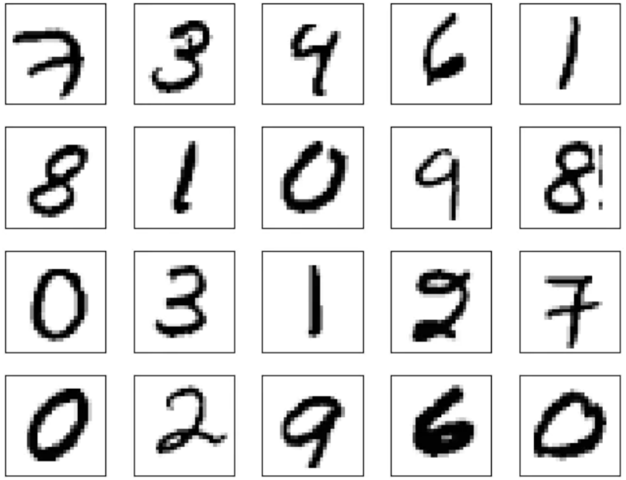
深度学习社区的小朋友们热爱ANN



无监督学习、监督学习

案例1

6 9 8 0



MNIST

案例2

Stanford Dogs Dataset

Summary:

- 120 dog breeds
- ~150 images per class
- Total images: 20,580

无监督学习、监督学习

无监督学习：直接从数据中学习某些模式
监督学习：从数据中学习提前指定的模式

自主学习、科学创新：无监督学习，没有标准答案，凭好奇心发现世界的科学规律

训练做题、考高分：监督学习，所有问题都有标准答案，做对的越多越好

无监督学习、监督学习

本次课程涉及的算法:

PCA：最常见、最简单的一类无监督学习算法，可以对数据进行降维，本质是基变换

CNN：比较常见的监督学习算法，可用于图像分类

无监督学习：PCA

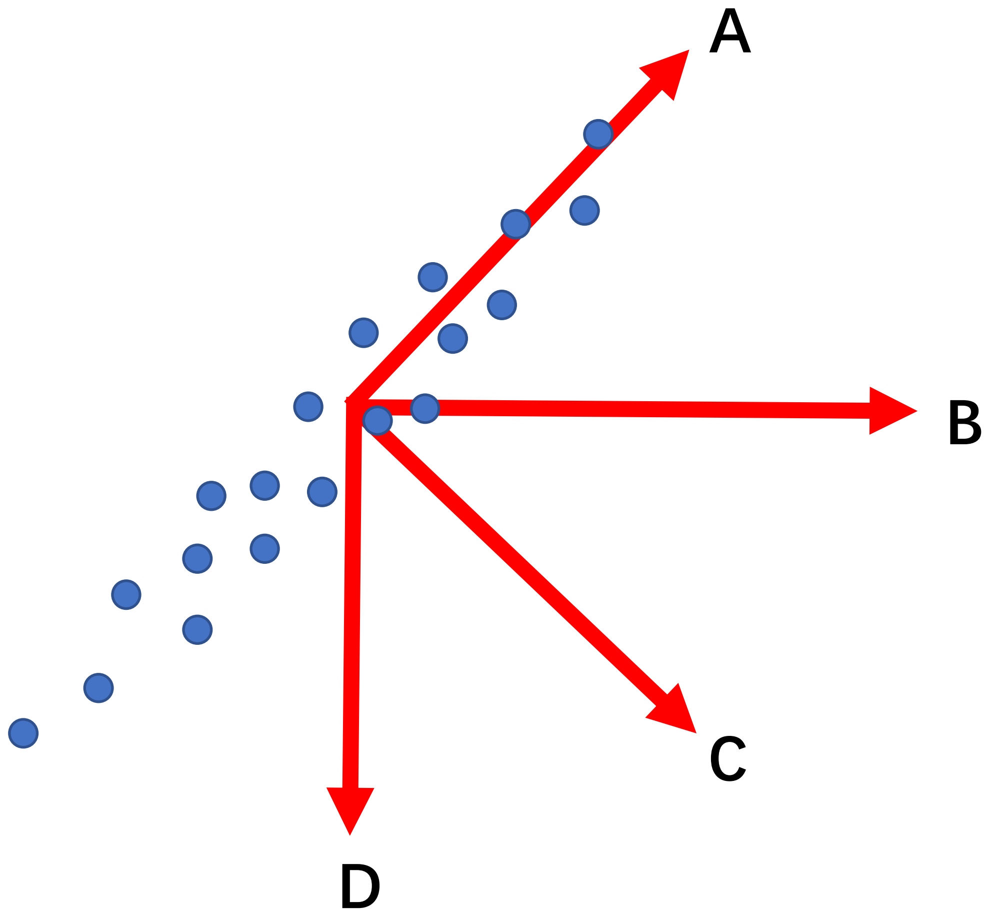
多维数据可视作线性空间中的一个向量
例：数据（0.1, 0.5, 0.2, 0.4）是R^4中的一个向量

PCA的目标：在这个线性空间中寻找一组有如下特点的正交基
1）这组基的第一个向量的方向就是使得所有数据在上面**投影的方差最大**的那个方向
2）这组基的第n至最后一个向量张成线性空间V’，所有数据点投射到V’后组成点集P’，第n个向量就是所有V’中向量里使得P’在其上的**投影方差最大**的那个向量

这组基中的每个向量都被称为一个主成分

无监督学习：PCA

如图是一组二维数据，在箭头所示的向量里，第一个主成分是哪个？**A**
第二个主成分是哪个？**C**



主成分分析可以去除数据多余的维数，将最明显的特征放于更低的维度上。

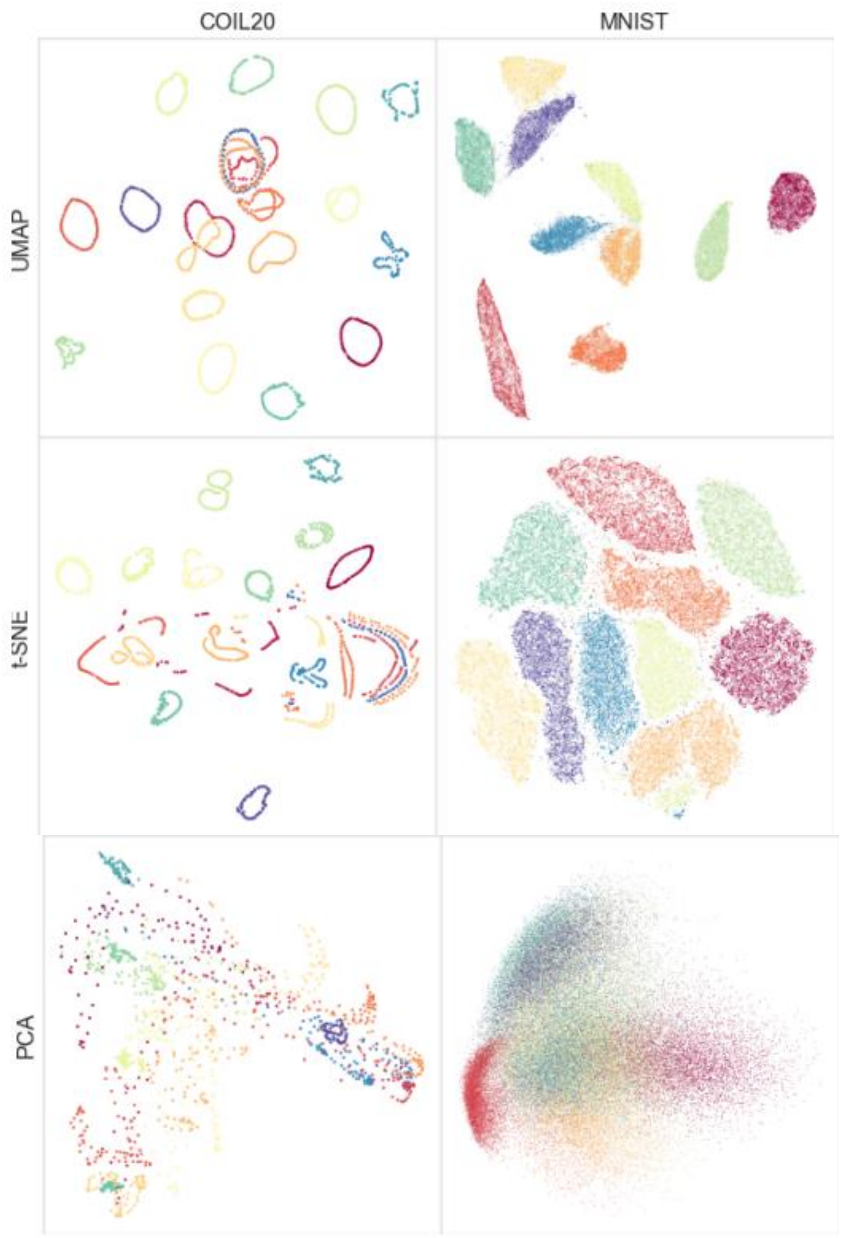
1. 降维
2. 降噪
3. 复杂算法的预处理步骤，如聚类的预处理
4. 提取特征
5. 体现数据的高维特性

流形学习

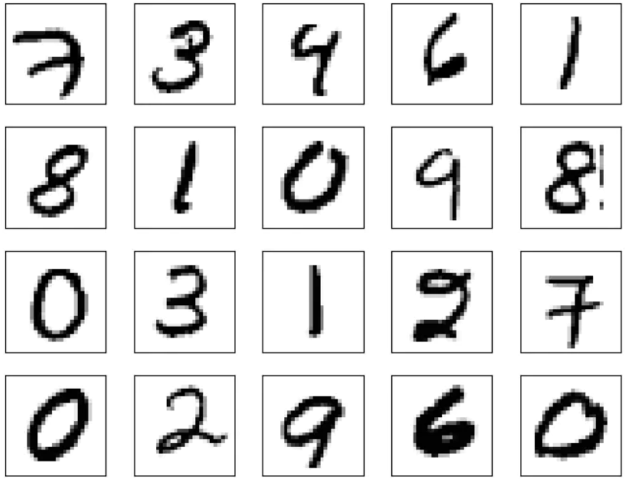
常见的流形学习算法

ISOMAP
LLE
t-SNE
UMAP

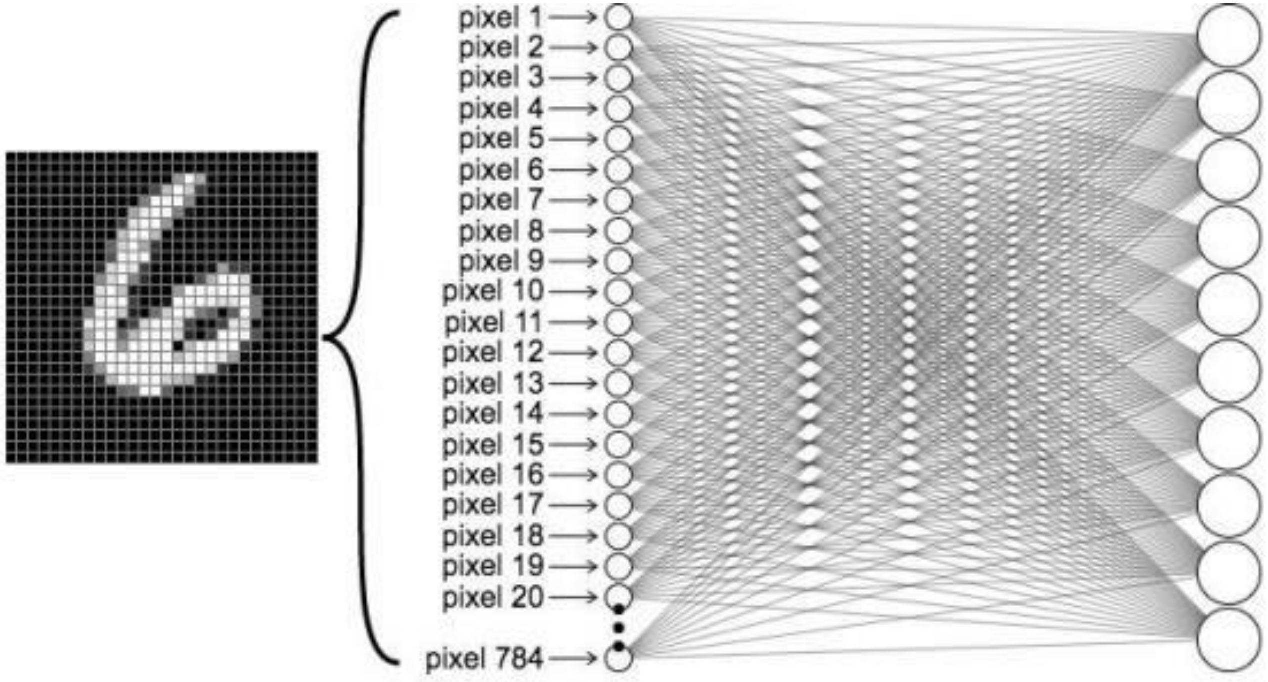
相比于PCA，流形学习更进一步地假设所有数据分布在高维空间中的一个**低维流形**上，基于这个假设得到的降维效果更好，但可解释性也变差了



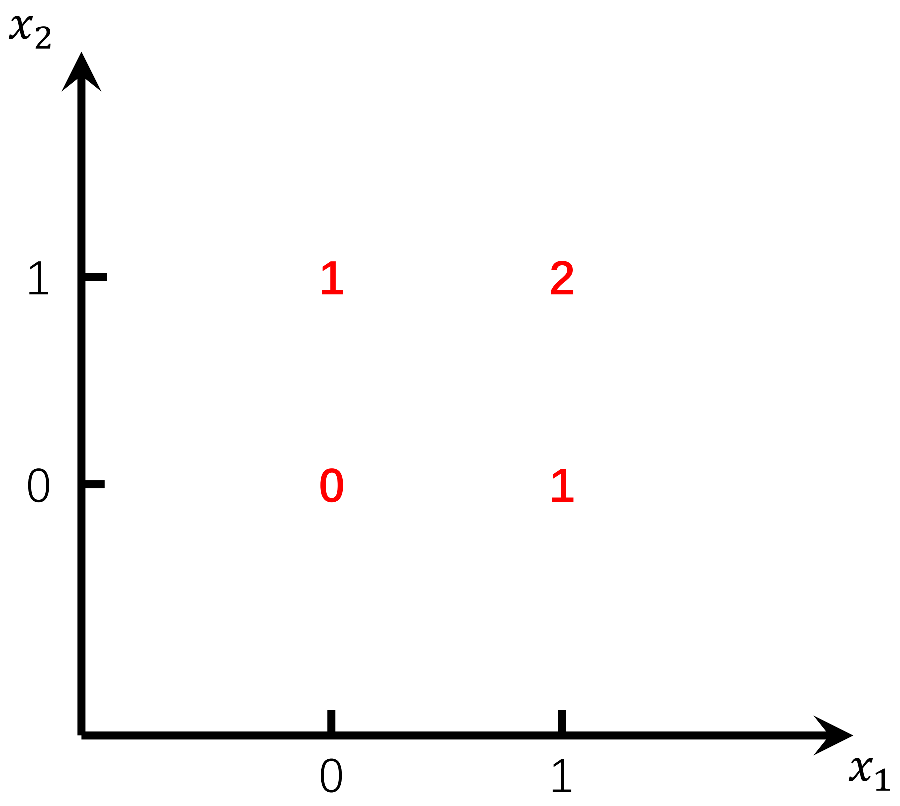
全连接网络（多层感知机，Multi-Layer Perceptron, MLP）



MNIST



目标：建立全连接的模型解释如下的数据

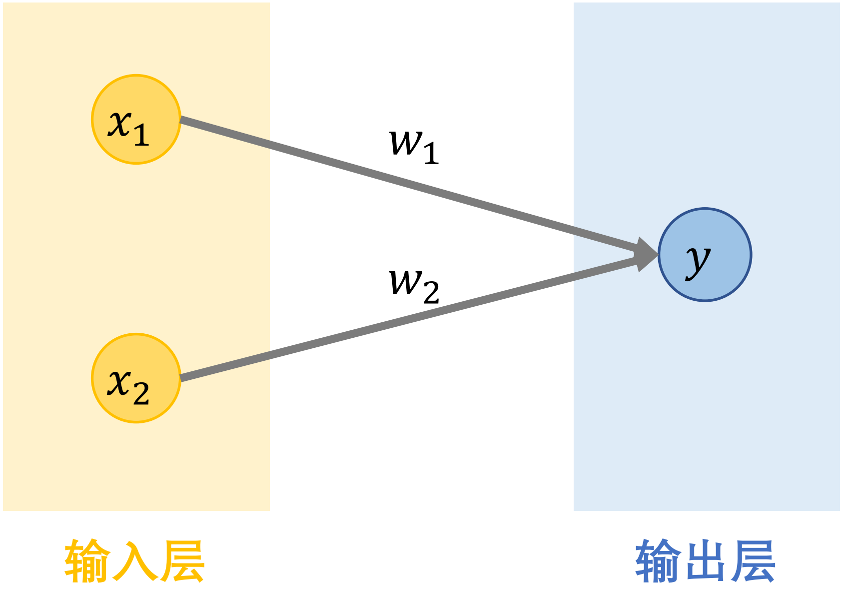


x	y
(0, 0)	0
(1, 0)	1
(0, 1)	1
(1, 1)	2

目标是给定x预测y:

x	y
(0, 0)	0
(1, 0)	1
(0, 1)	1
(1, 1)	2

模型:



初等数学语言: $\hat{y} = w_1 x_1 + w_2 x_2 + b$

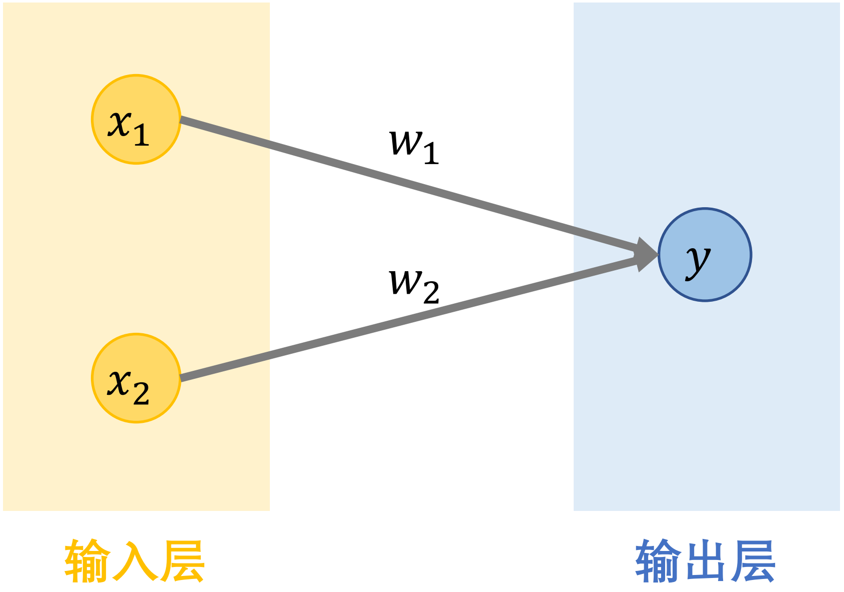
线性代数语言: $\hat{y} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} + b$

$$\hat{y} = \boldsymbol{x}^T \boldsymbol{w} + b$$

目标是给定x预测y:

x	y
(0, 0)	0
(1, 0)	1
(0, 1)	1
(1, 1)	2

模型:



$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w} + b$$

权重矩阵　　偏置参数

$$\begin{pmatrix} \widehat{y_1} \\ \widehat{y_2} \\ \widehat{y_3} \\ \widehat{y_4} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} + b$$

用MSE作为**损失函数**: $J = \frac{1}{4}\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|^2$

目标: $\underset{\boldsymbol{w},b}{\operatorname{argmin}}\ J(\boldsymbol{w}, b)$

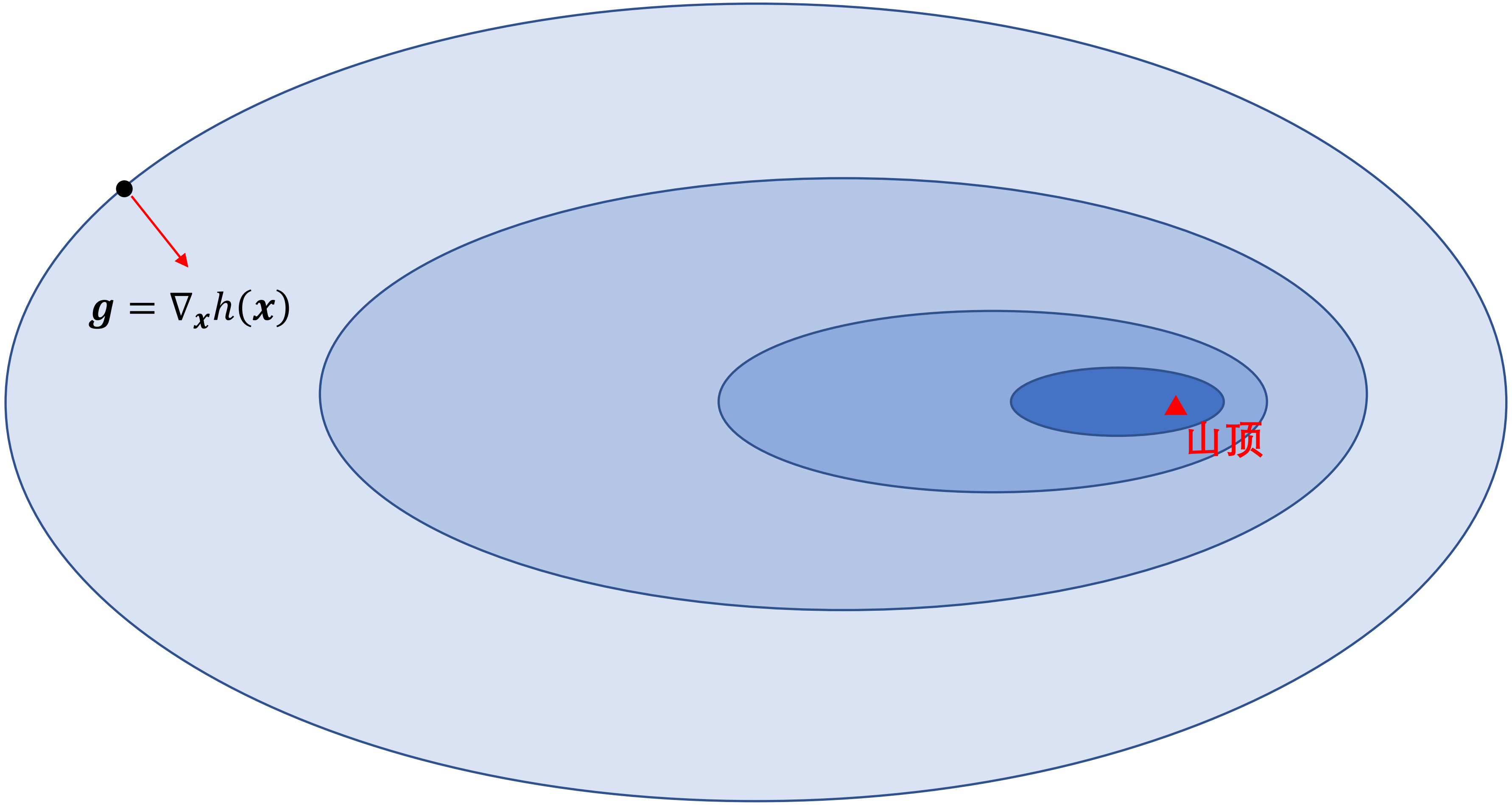
盲人爬山法（梯度下降/上升）

给定　$h = h(\boldsymbol{x})$

目标：　$\underset{\boldsymbol{x}}{\operatorname{argmax}}\, h(\boldsymbol{x})$

$$\boldsymbol{g} = \nabla_{\boldsymbol{x}} h(\boldsymbol{x}) = \begin{pmatrix} \dfrac{\partial h}{\partial x_1} \\ \dfrac{\partial h}{\partial x_2} \end{pmatrix}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \underbrace{\varepsilon}_{\text{学习率}} \boldsymbol{g}$$



$y = \boldsymbol{x}^T \boldsymbol{w} + b$

用MSE当损失函数:　　$J = \frac{1}{4}\left\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\right\|^2$

目标:　　$\underset{\boldsymbol{w},b}{\operatorname{argmin}}\ J(\boldsymbol{w}, b)$

梯度下降

初始化一个 $(\boldsymbol{w}, b)_0^T$

按照下列公式迭代

$$\begin{pmatrix}\boldsymbol{w}\\ b\end{pmatrix}_n = \begin{pmatrix}\boldsymbol{w}\\ b\end{pmatrix}_{n-1} - \varepsilon \nabla_{\boldsymbol{w},b} J(\boldsymbol{w}, b)$$

演示（请有兴趣的同学验证）

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\boldsymbol{w} + b$$

$$J = \frac{1}{4}\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\right\|^2$$

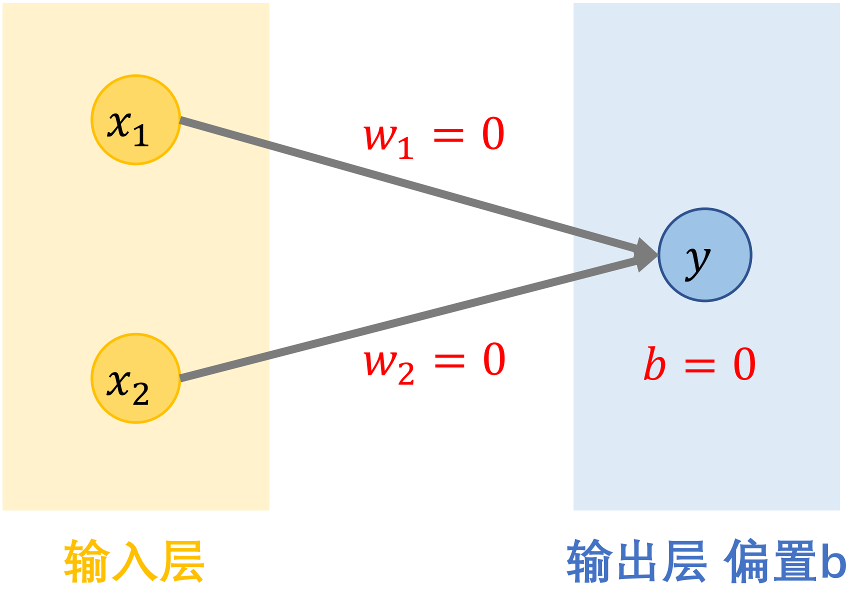
$\longrightarrow$

$$\boxed{\nabla_{\boldsymbol{w}} J(\boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} + b - \boldsymbol{Y})}$$

$$\boxed{\nabla_b J(\boldsymbol{w}, b) = \frac{1}{2}\sum_{i=1}^{4}(\boldsymbol{x}_i^T\boldsymbol{w} + b - y_i)}$$

x	y
(0, 0)	0
(1, 0)	1
(0, 1)	1
(1, 1)	2

初始化

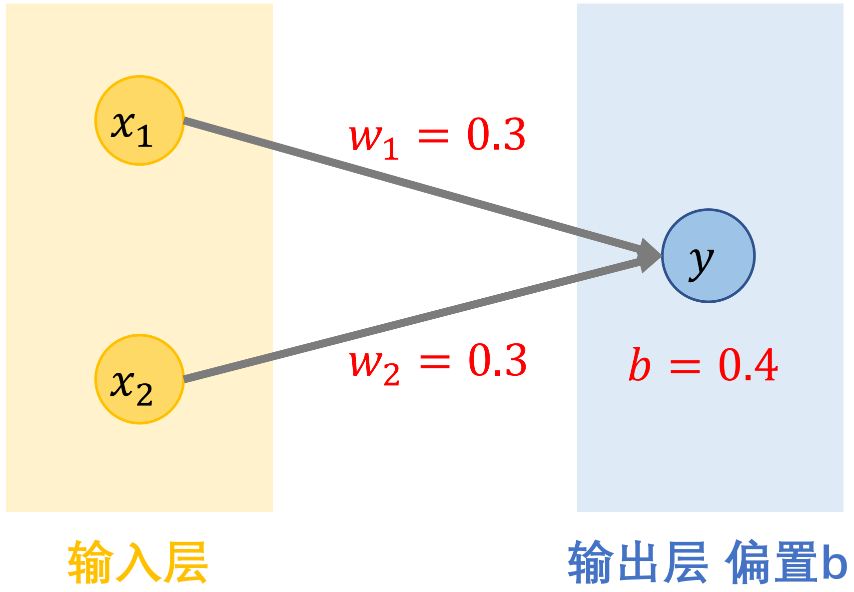


$$\hat{\boldsymbol{Y}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix} + 0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Loss: $J(\boldsymbol{w}, b) = \frac{1}{4}(0^2 + 1^2 + 1^2 + 2^2) = \frac{6}{4} = 1.5$

Accuracy: 25%

第1轮（学习率0.2）



$$\nabla_{\boldsymbol{w},b} J(\boldsymbol{w}, b) = \begin{pmatrix} -1.5 \\ -1.5 \\ -2 \end{pmatrix}$$

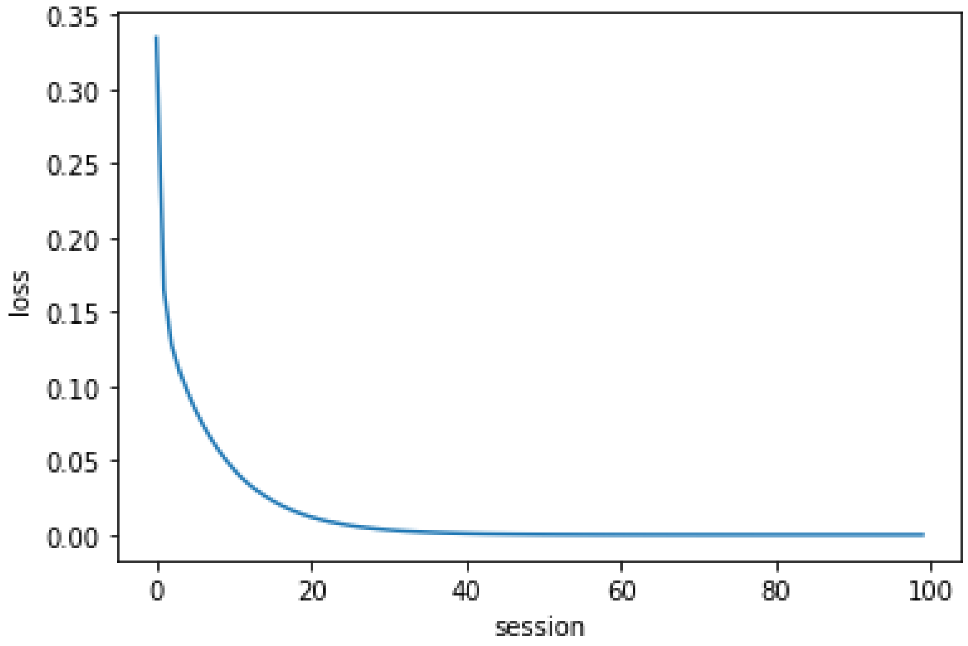
$$\begin{pmatrix} w_1 \\ w_2 \\ b \end{pmatrix}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} - 0.2 \times \begin{pmatrix} -1.5 \\ -1.5 \\ -2 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.4 \\ 0.3 \end{pmatrix}$$

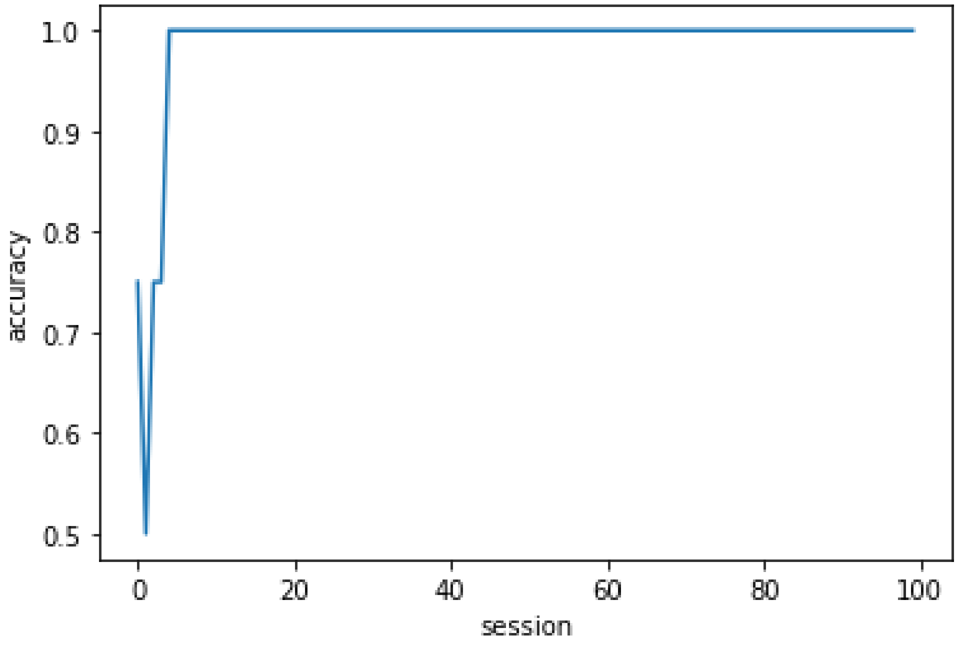
$$\widehat{\boldsymbol{Y}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0.3 \\ 0.3 \end{pmatrix} + 0.4 = \begin{pmatrix} 0.4 \\ 0.7 \\ 0.7 \\ 1.0 \end{pmatrix}$$

Loss: $J(\boldsymbol{w}, b) = \frac{1}{4}(0.4^2 + 0.3^2 + 0.3^2 + 1^2) = 0.35$

Accuracy: 75%

100轮（100 sessions / epochs）之后（学习率0.2）

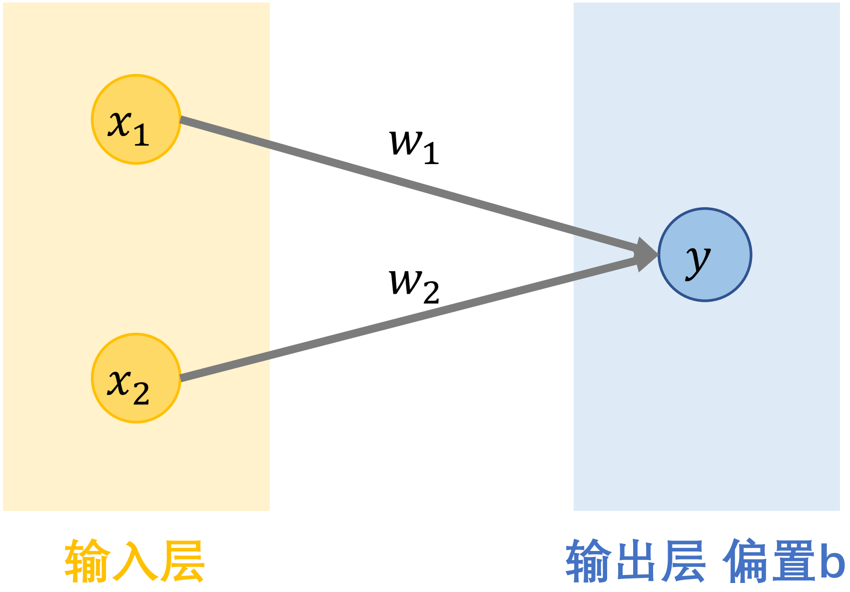


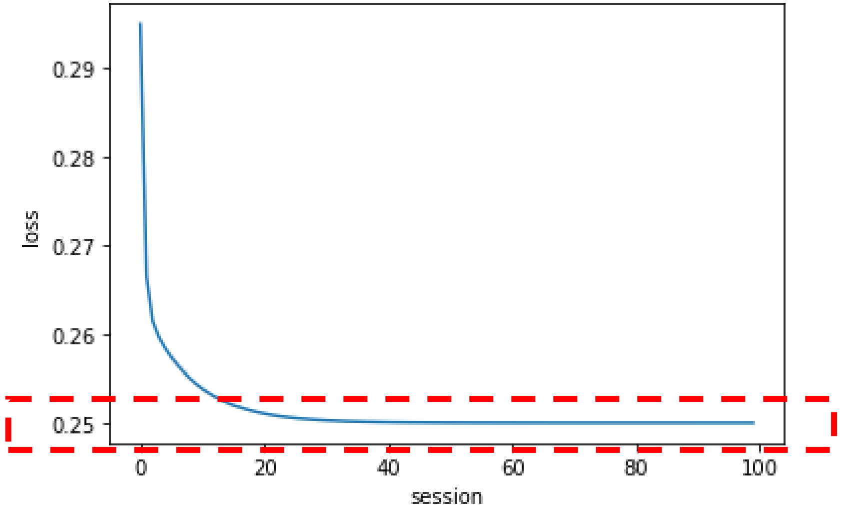


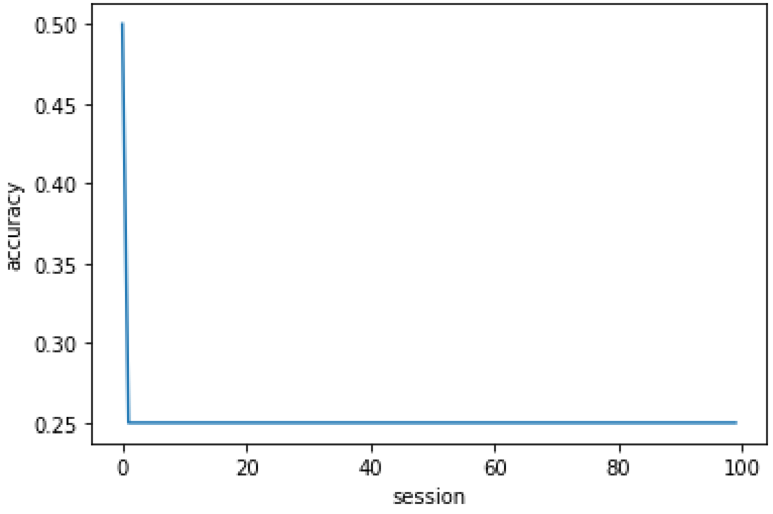
学习异或:

x	y
(0, 0)	0
(1, 0)	1
(0, 1)	1
(1, 1)	0

模型:

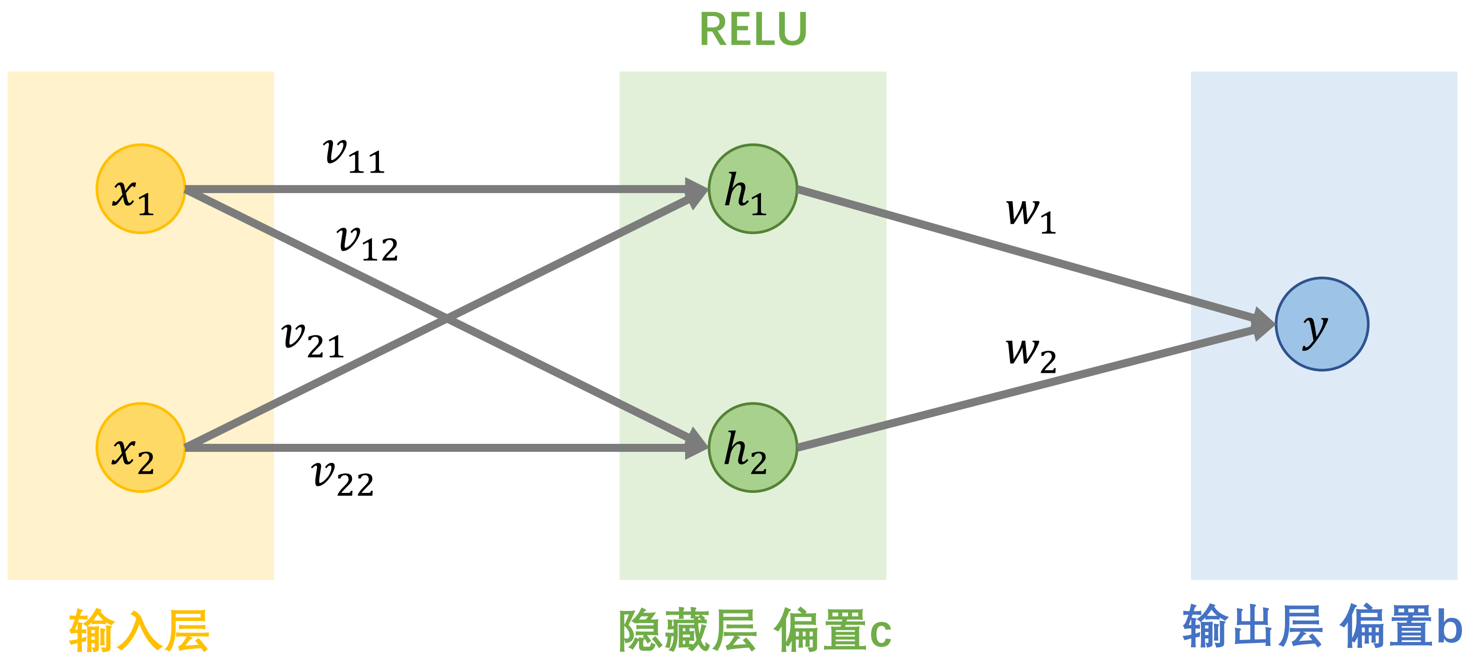






学习异或:

为了使模型成为非线性，在隐藏层增加激活函数RELU（线性整流函数）



$$RELU(x) = \max\{0, x\}$$

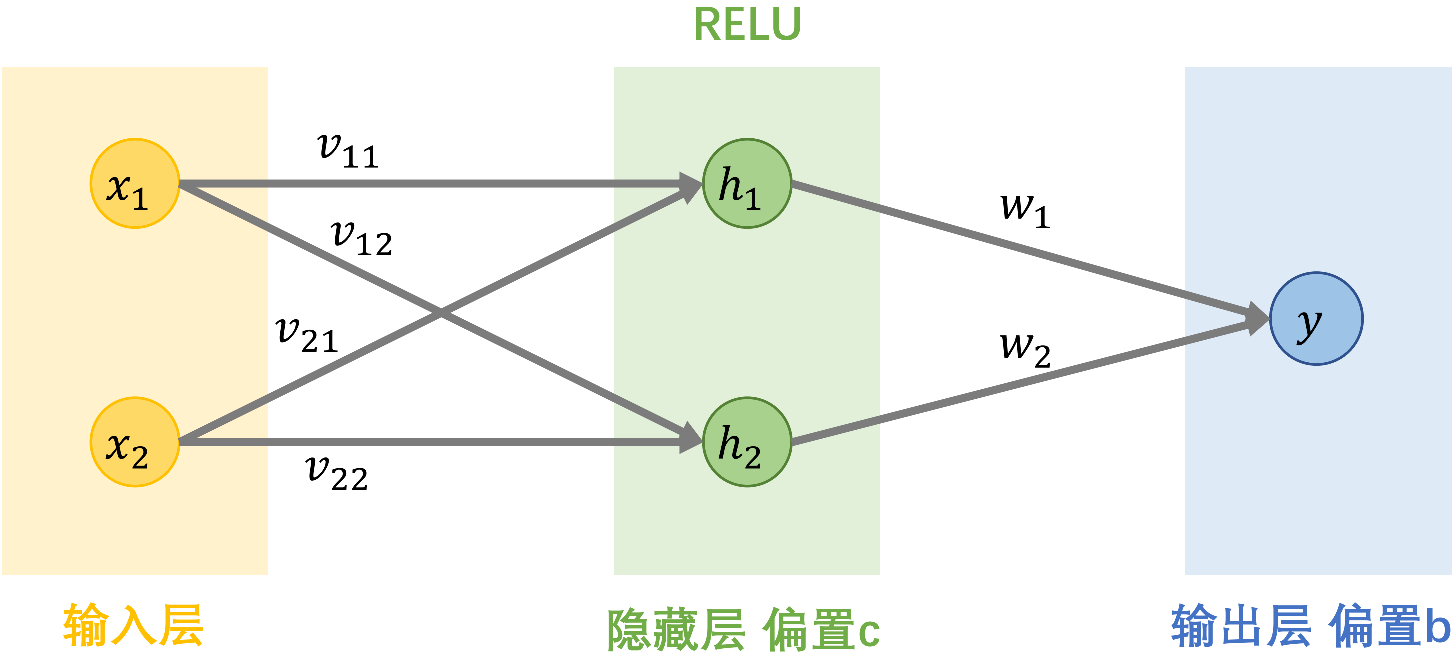
输出层常用的输出函数softmax： $o_i = \frac{e^{y_i}}{\sum e^{y_j}}$

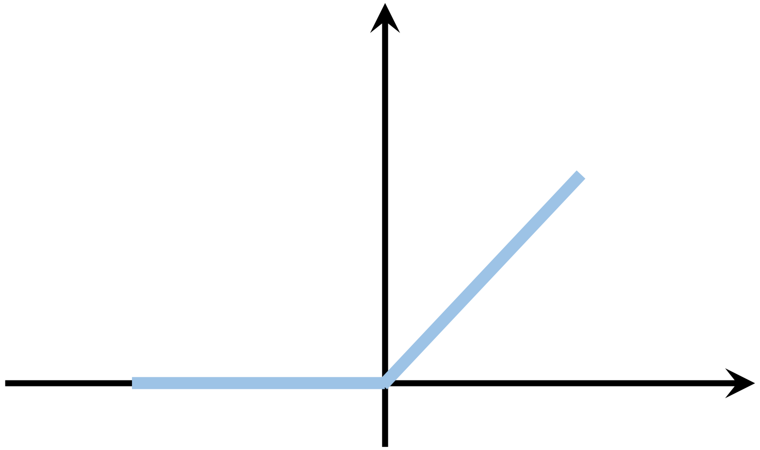
它有很好的性质，参见Deep Learning一书115-117页

反向传播

设 $Z = Z[\boldsymbol{y}(\boldsymbol{x})]$，根据链式法则： $\nabla_{\boldsymbol{x}} Z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^T \nabla_{\boldsymbol{y}} Z$

雅可比阵



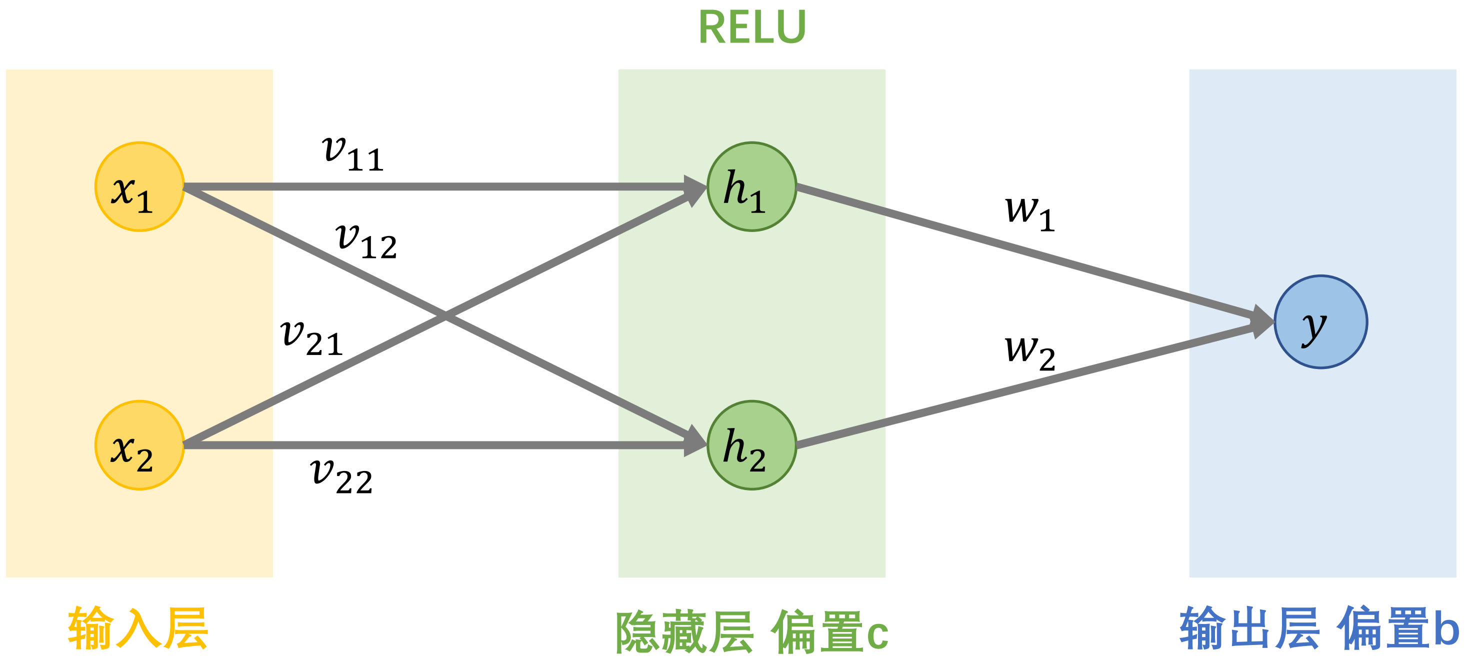


RELU在x=0不可导

这种情况可取其左导数

$$\frac{\partial RELU(x)}{\partial x} = \max\{0, \operatorname{sgn} x\}$$

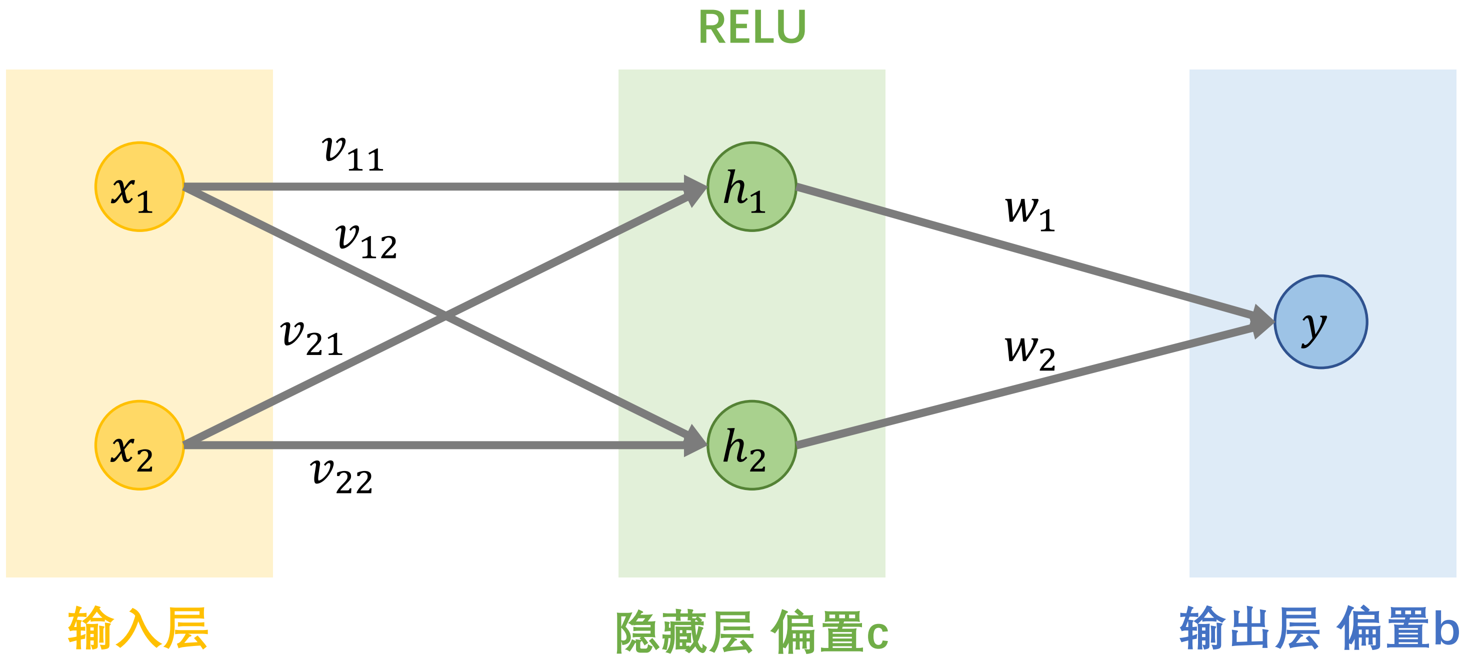
学习异或:



损失函数：MSE $$J = \frac{1}{4}\left\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\right\|^2 = \frac{1}{4}\sum_{i=1}^{4}(y_i - \widehat{y_i})^2$$

对于一组输入/输出的输出层梯度: $$\nabla_y J = \frac{1}{2}(y - \hat{y})$$

学习异或:



输出层梯度: $\nabla_{\hat{y}} J = \frac{1}{2}(y - \hat{y})$ 无激活函数

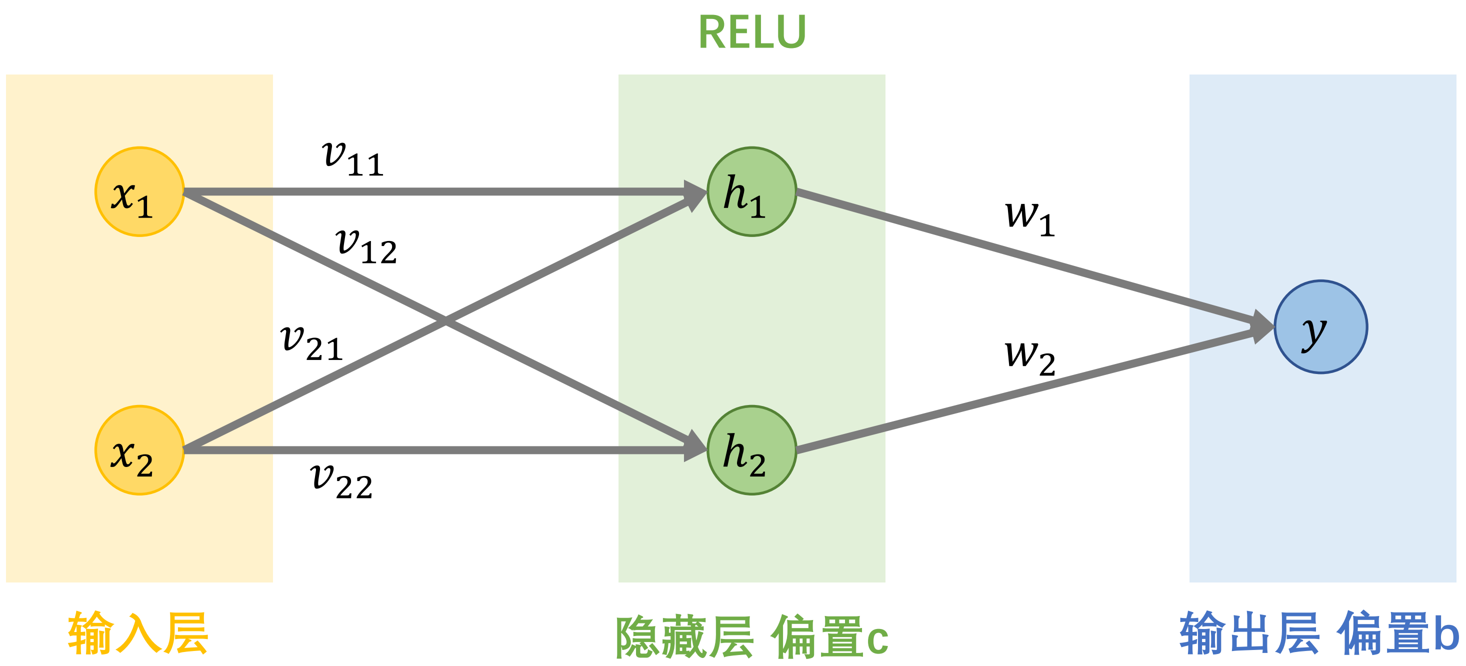
权重梯度: $\nabla_{w} J = \left(\frac{\partial \hat{y}}{\partial \boldsymbol{w}}\right)^T \nabla_{\hat{y}} J = \frac{1}{2}(y - \hat{y})\boldsymbol{h}$

偏置梯度: $\nabla_{b} J = \left(\frac{\partial \hat{y}}{\partial b}\right)^T \nabla_{\hat{y}} J = \frac{1}{2}(y - \hat{y})$

梯度传播到隐藏层

$$\nabla_{h} J = \left(\frac{\partial \hat{y}}{\partial \boldsymbol{h}}\right)^T \nabla_{\hat{y}} J = \frac{1}{2}(y - \hat{y})\boldsymbol{w}$$

学习异或:



隐藏层梯度: $\nabla_h J = \frac{1}{2}(y-\hat{y})\boldsymbol{w}$

有激活函数,运用链式法则: $\nabla_a J = \nabla_{\boldsymbol{a}} RELU(\boldsymbol{a}) \odot \nabla_h J = \frac{1}{2}(y-\hat{y}) \max\{0, \operatorname{sgn} \boldsymbol{a}\} \odot \boldsymbol{w}$

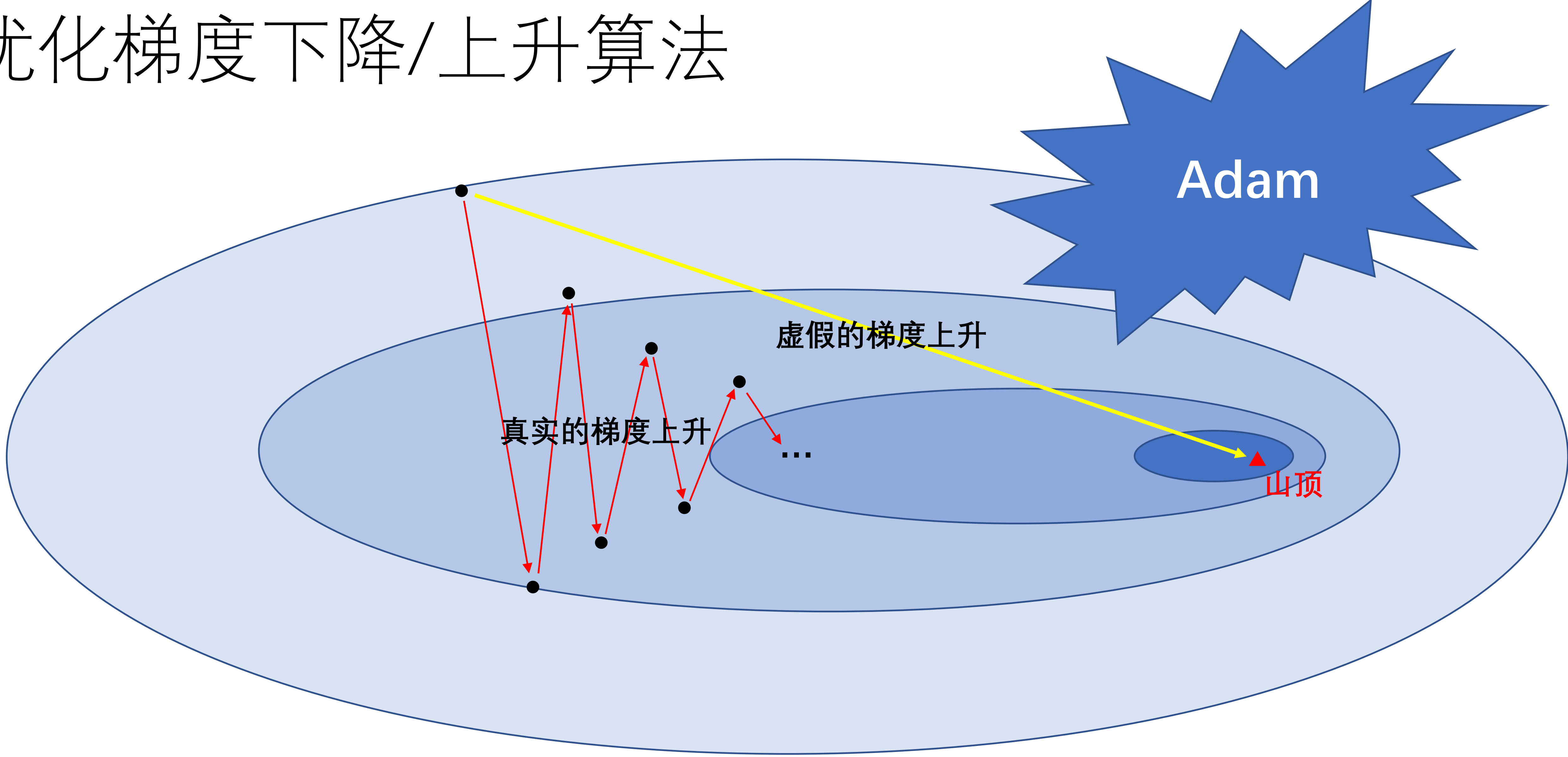
$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \odot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_2 b_2 \end{pmatrix}$$

权重梯度: $\nabla_v J = \left(\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{v}}\right)^T \nabla_a J = \boldsymbol{v}^T \nabla_a J$

偏置梯度: $\nabla_c J = \left(\frac{\partial \boldsymbol{a}}{\partial c}\right)^T \nabla_a J = \nabla_a J$

梯度传播到输入层,传播结束

优化梯度下降/上升算法



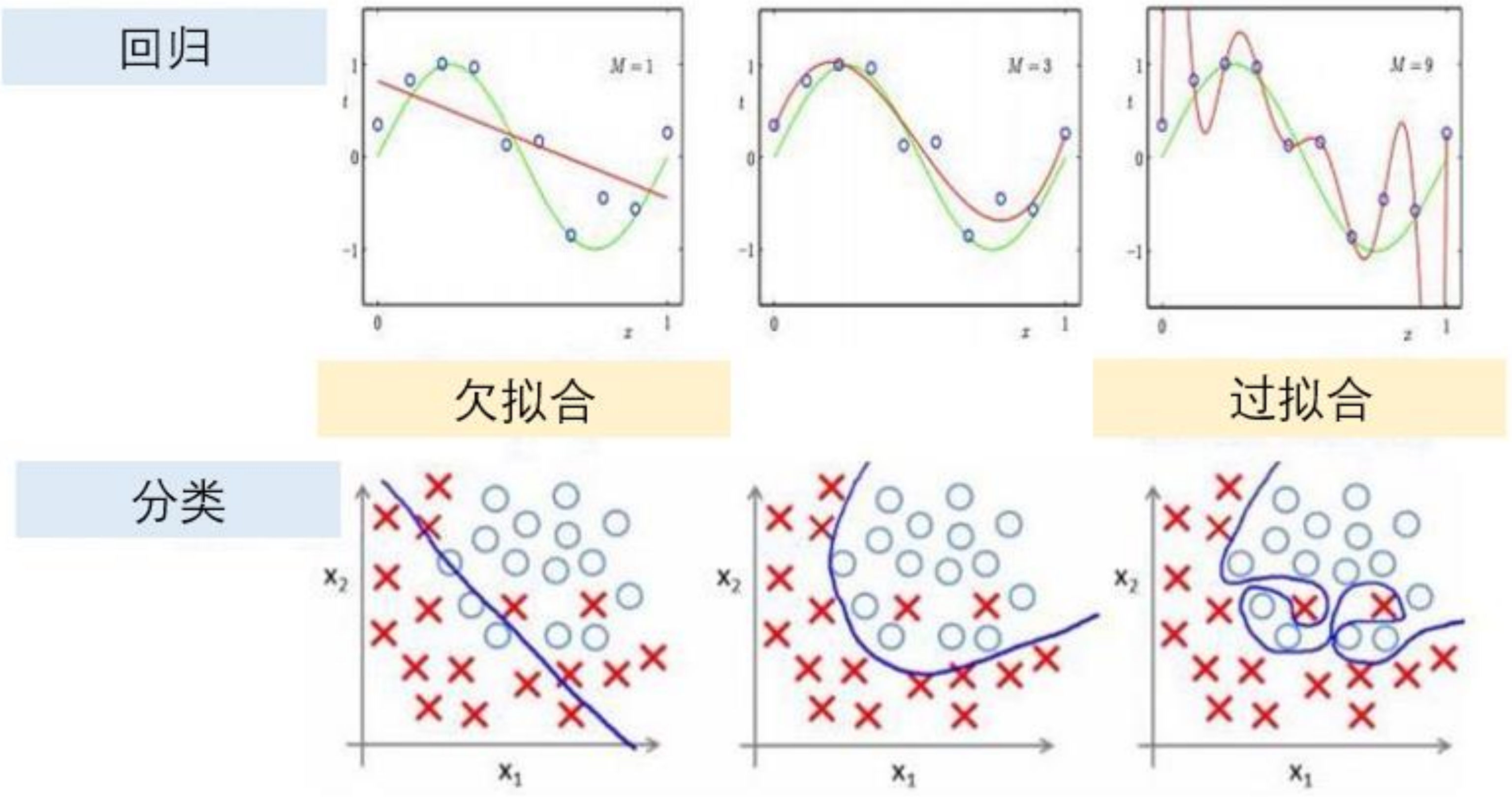
人们利用动量方法和可调节的学习率开发了各种比SGD更强大的优化器。

分批训练： batch

小batch： 更强的泛化能力，更优的解，但是训练和收敛可能慢。示例代码的batch size是100，偏大，同学们可以尝试10-50的batch size

过大的batch： 降低梯度下降的随机性，可能陷入局部最优解；占内存

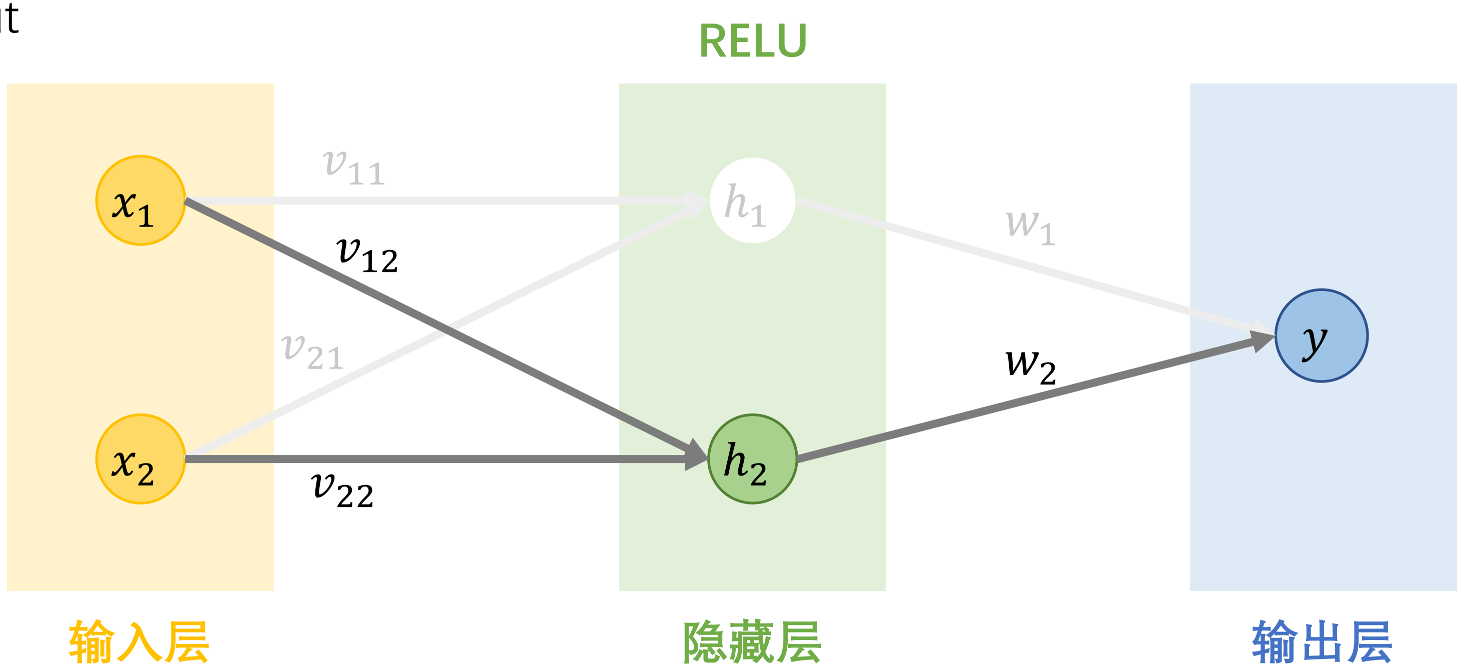
欠拟合（underfit）、过拟合（overfit）



正则化以防止过拟合

正则化方法举例

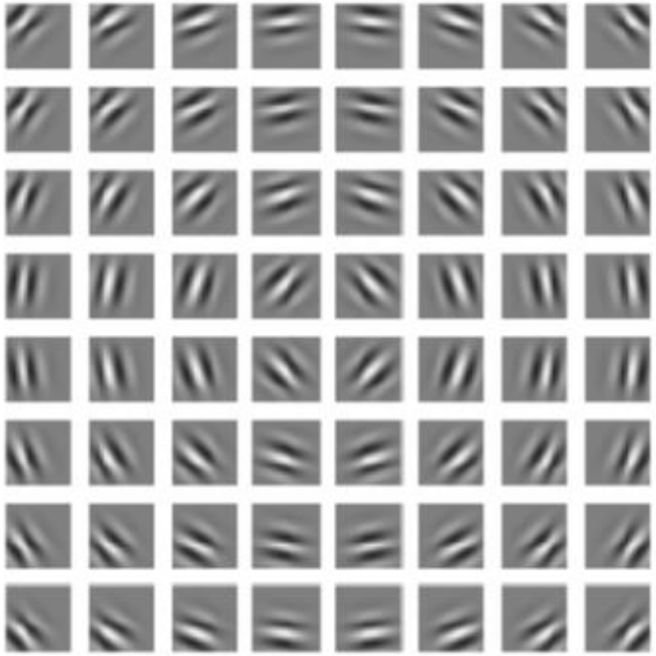
1. Dropout

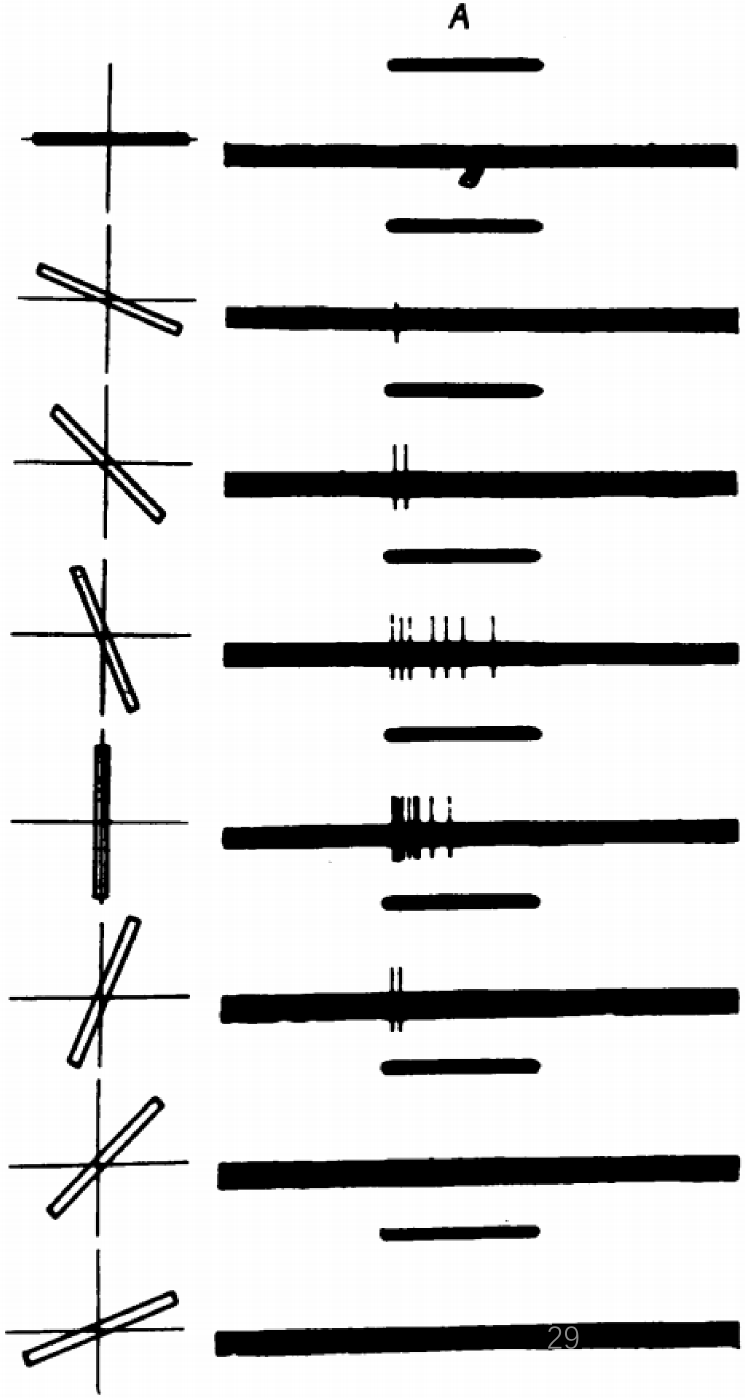


2. 目标函数加正则化项：$J = L + \lambda\Omega$
L1和L2正则化

视觉和卷积

1958, Hubel & Wiesel





卷积

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 3 \\ 2 & 1 & 2 & 0 & 2 \\ 1 & 0 & 0 & 3 & 1 \\ 0 & 1 & 2 & 1 & 4 \\ 0 & 0 & 3 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 8 & 7 \\ 3 & 7 & 5 \\ 3 & 7 & 10 \end{bmatrix}$$

V　　　　卷积核 K　　　　Z

$$Z_{i,j} = \sum_{m,n} V_{i+m-1,j+n-1} K_{m,n}$$

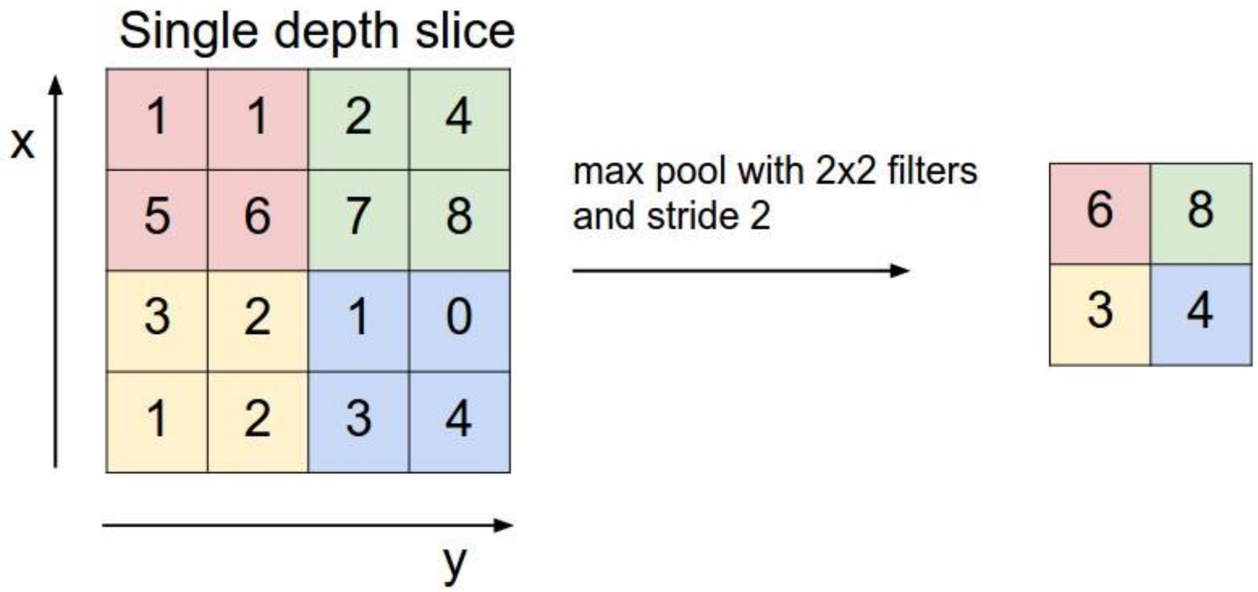
实际CNN网络中的卷积： $Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1}\left(K_{i,l,m,n} + b_i\right)$

卷积核　　**偏置参数**

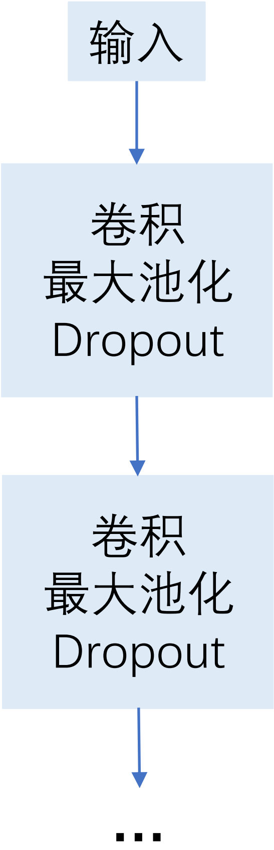
CNN目标：寻找合适的卷积核
方法：反向传播的梯度下降，详见花书第九章

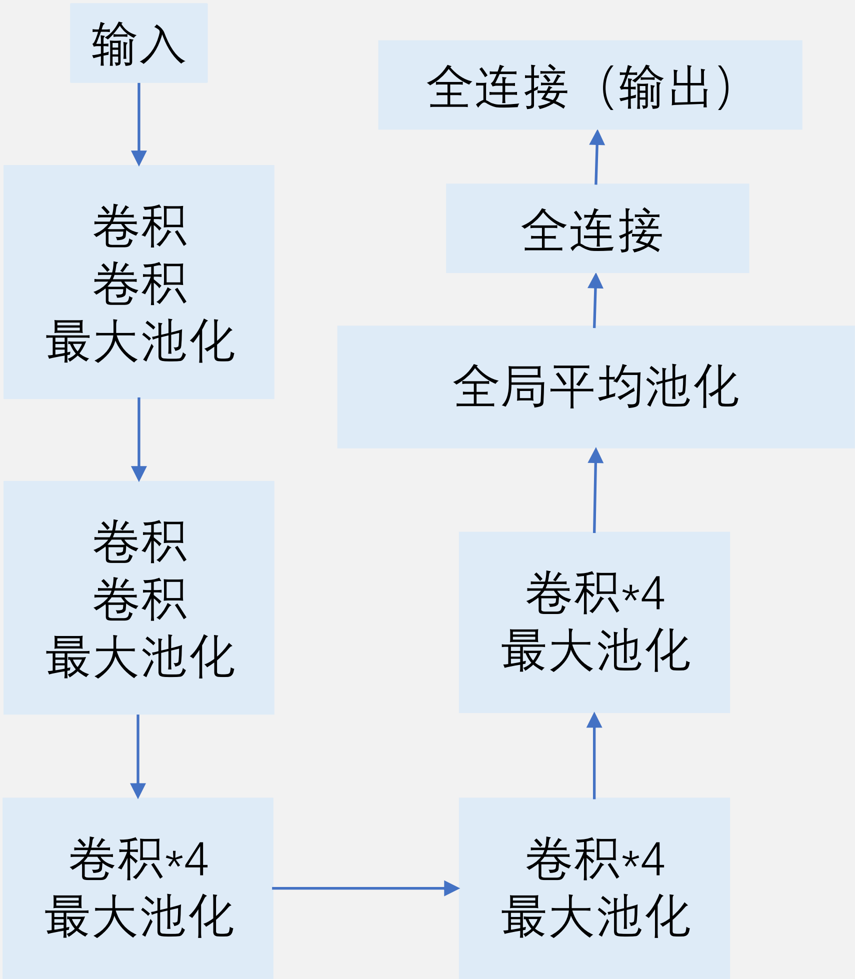
最大池化

池化的作用：减少不必要的计算、提高训练速度、扩大感受野、降低对于微小位移的敏感性



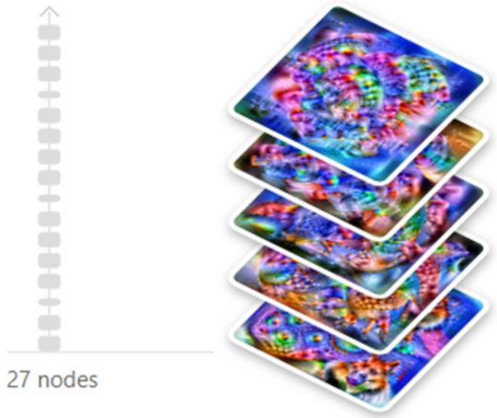
典型的卷积神经网络结构





VGG 19

Introduced in 2014, this network is simpler than Inception variants, using only 3x3 convolutions and no branches.



别人以为的AI vs 大多数情况下的AI

认为AI会统治世界的人	我训练的CNN

2021/5/16 干实验清单

下载和安装MATLAB，版本要高于 2020a

请勾选视觉工具箱！！！

湿实验清单

测量两种藻类的UV-vis光谱

拍摄约3分钟长度的两种藻类的40倍镜视频，要保证玻片清洁，焦距合适。拍摄过程要缓慢移动玻片，以便在保证每一帧都清楚的前提下，尽可能多地覆盖更大地玻片范围。