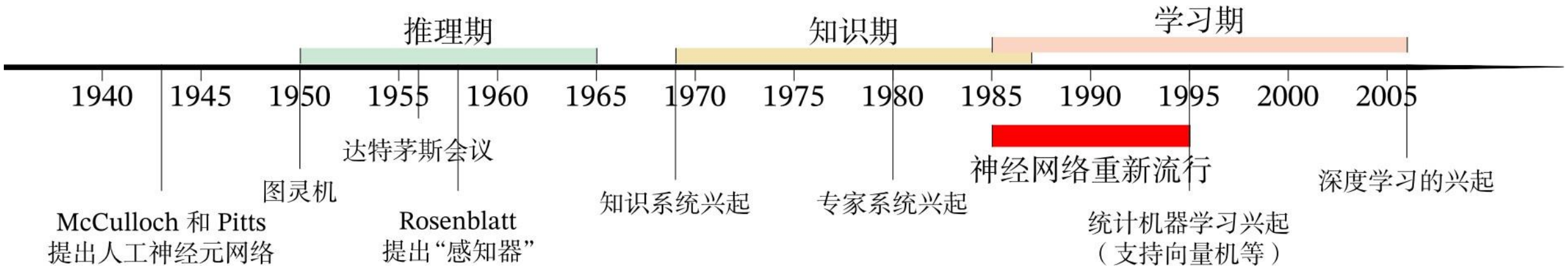
NEURAL NETWORKS AND DEEP LEARNING

神经网络与深度学习

Neural Networks

神经网络基础

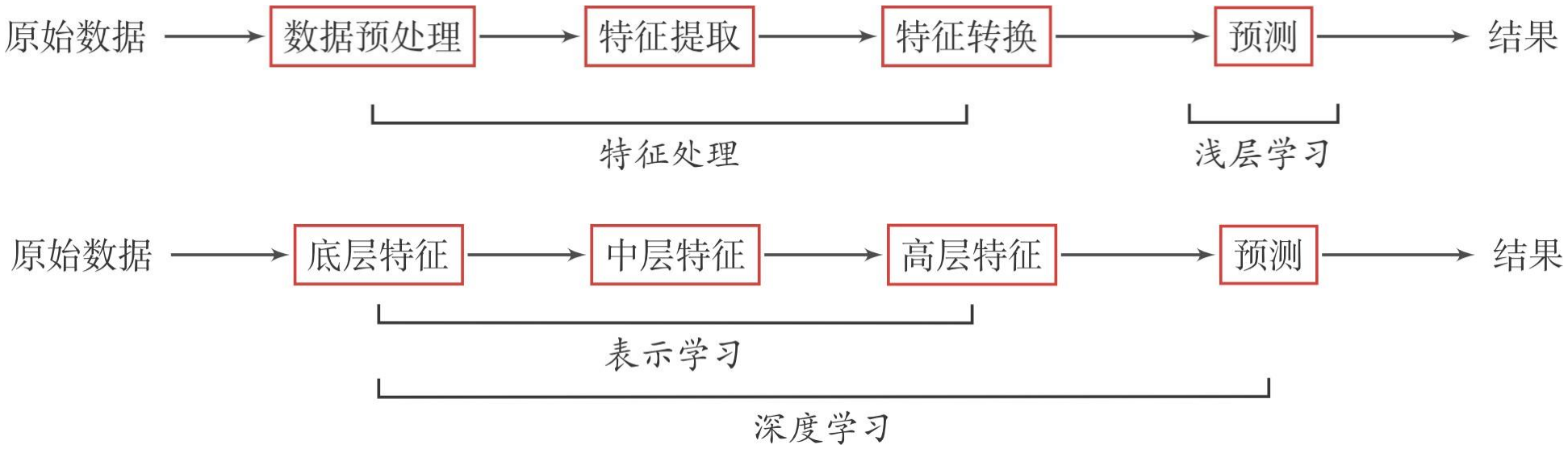
历史


推理期
知识期
学习期
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
达特茅斯会议
图灵机
神经网络重新流行
深度学习的兴起
知识系统兴起
专家系统兴起
统计机器学习兴起
（支持向量机等）
McCulloch 和 Pitts
提出人工神经元网络
Rosenblatt
提出“感知器”

历史

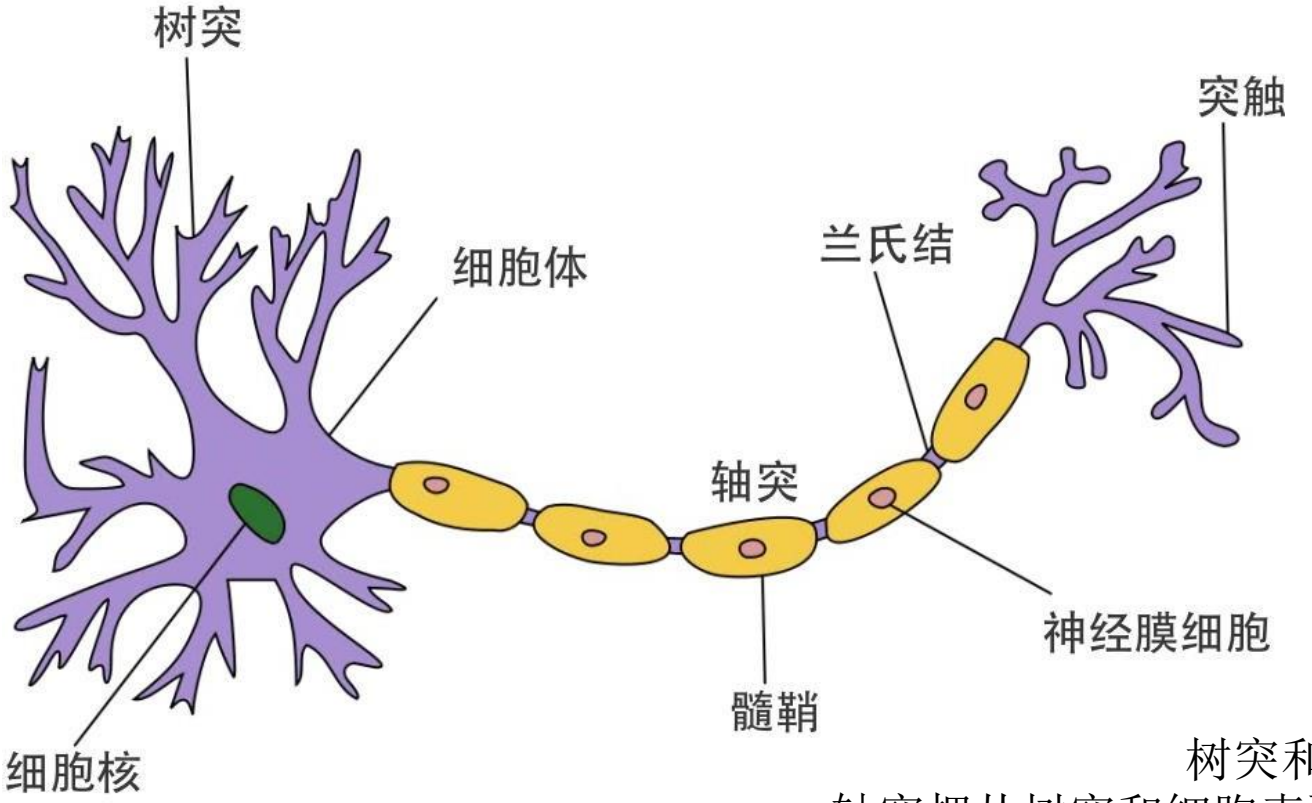
机器学习 → 表示学习

分阶段学习 → 端到端学习



神经元

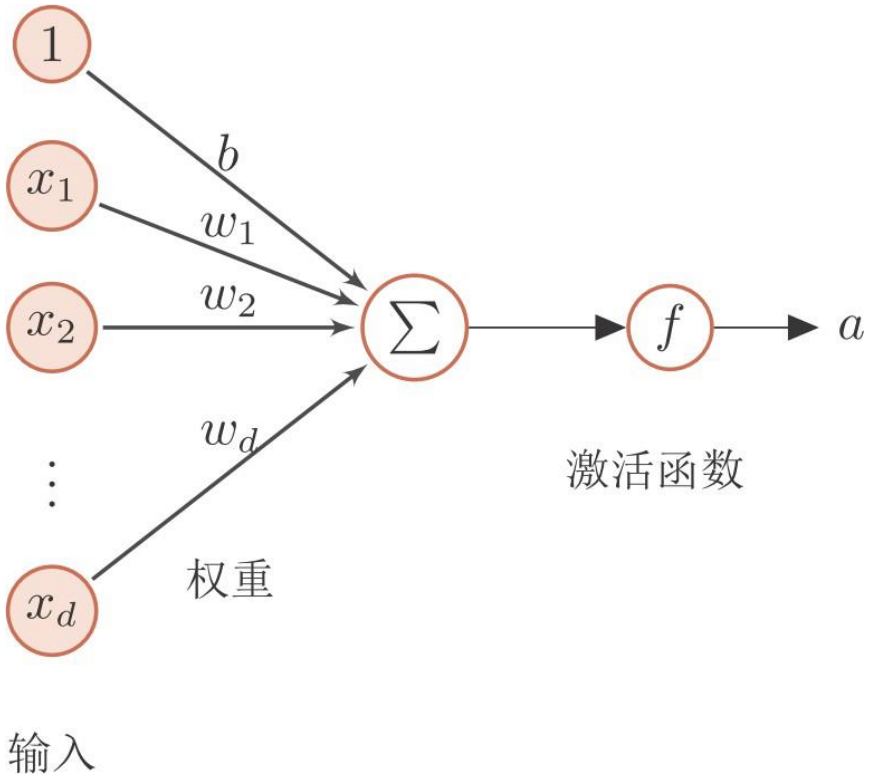
每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个“阈值”(threshold)，那么它就会被激活，即“兴奋”起来，向其他神经元发送化学物质.



树突和细胞体的表膜都有接受刺激的功能
轴突把从树突和细胞表面传入细胞体的神经冲动传出到其他神经元或效应器

神经元

[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。

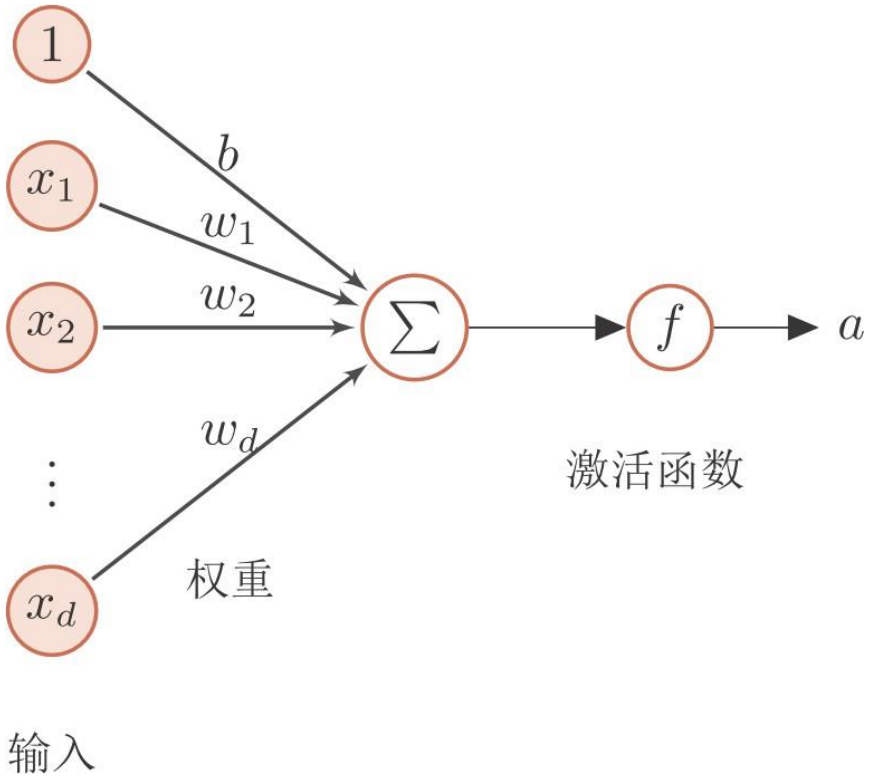


★ 阶跃型激活函数（在阈值处不可导，不是连续可微的，i.e.,得不到梯度）

$$\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)=\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\geq 0 & \rightarrow 1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b<0 & \rightarrow 0\end{cases}$$

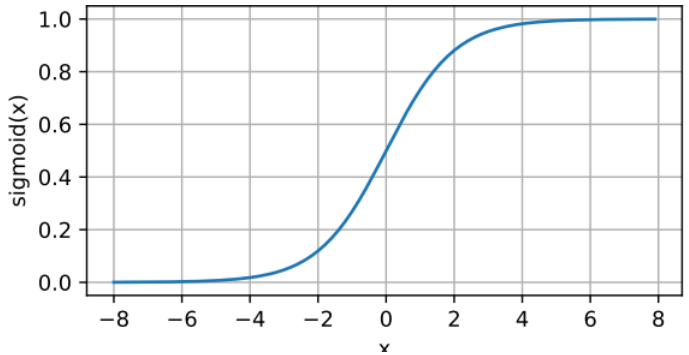
神经元

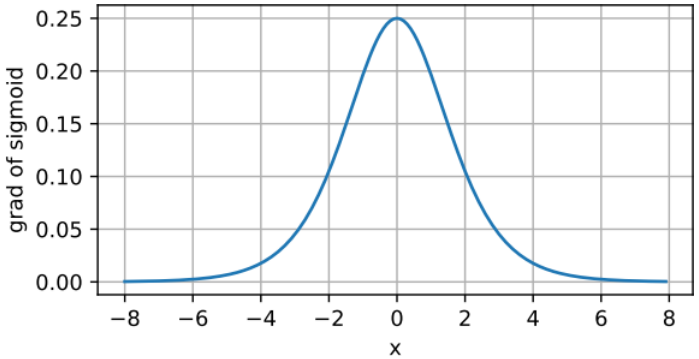
[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



★ Sigmoid 型激活函数

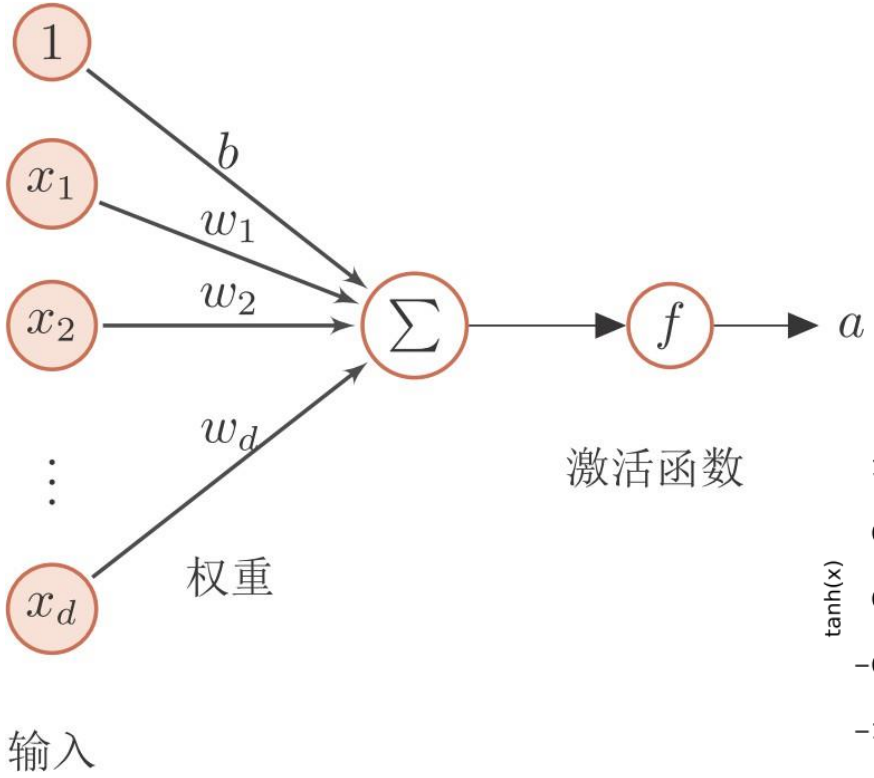
$$\sigma(\boldsymbol{w}^{\top}\boldsymbol{x}+b)=\frac{1}{1+\exp(-(\boldsymbol{w}^{\top}\boldsymbol{x}+b))}$$





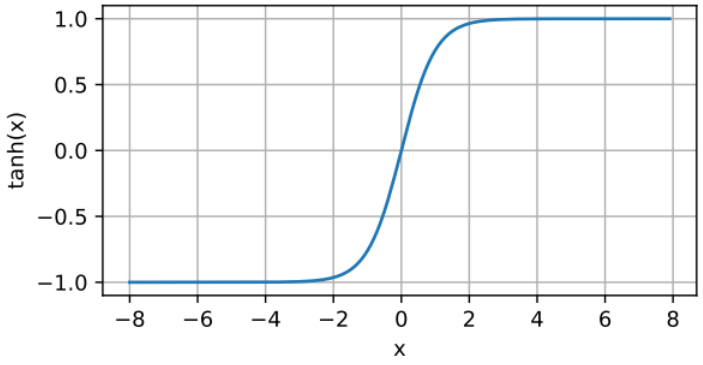
神经元

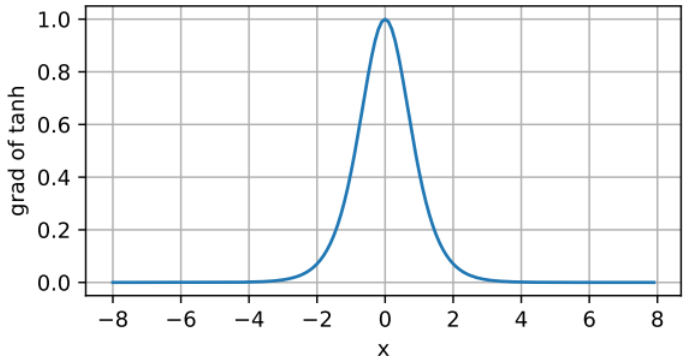
[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



★ 双曲正切型激活函数

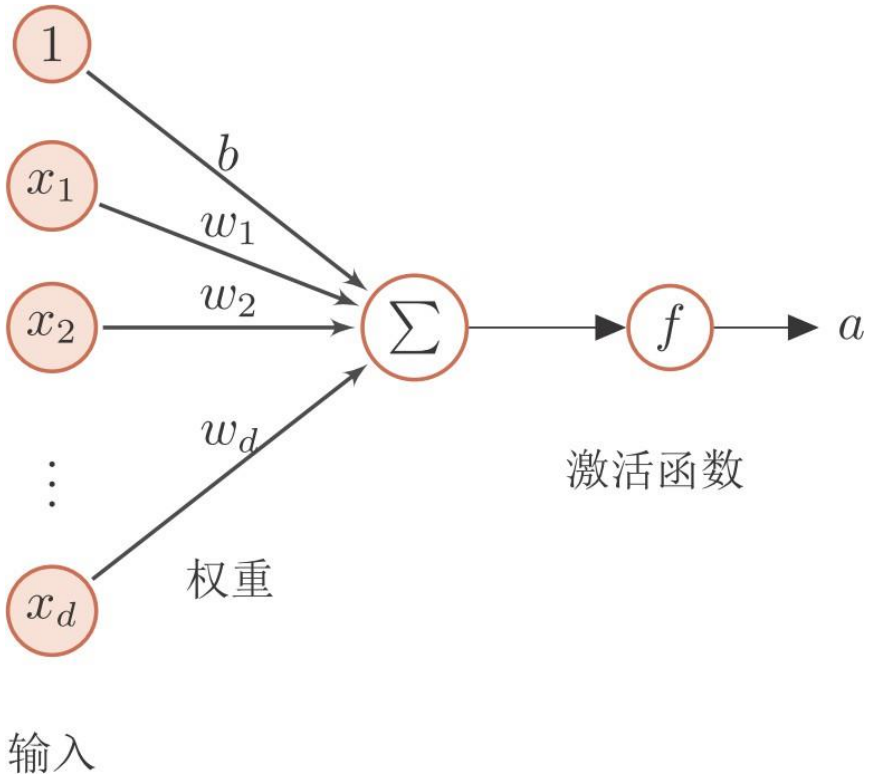
$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$





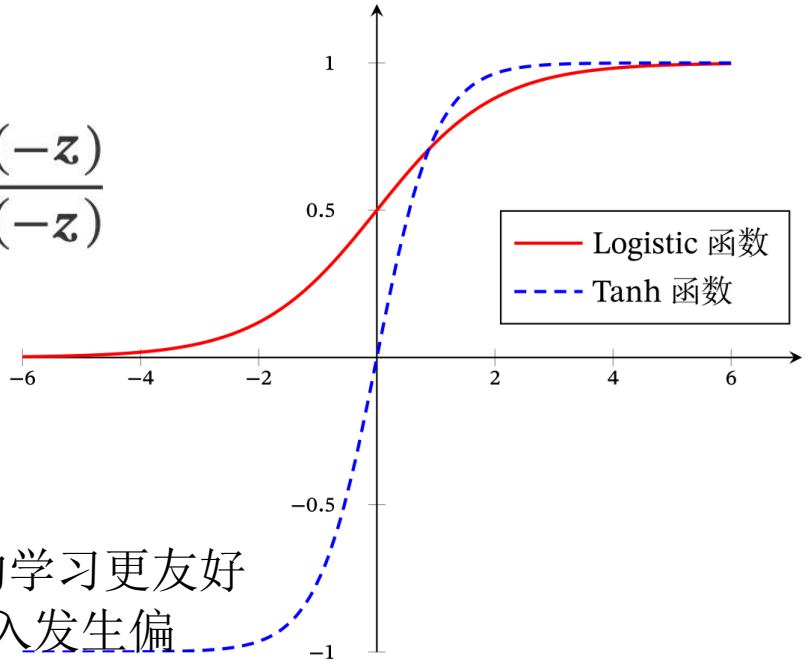
神经元

[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



★ 双曲正切型激活函数

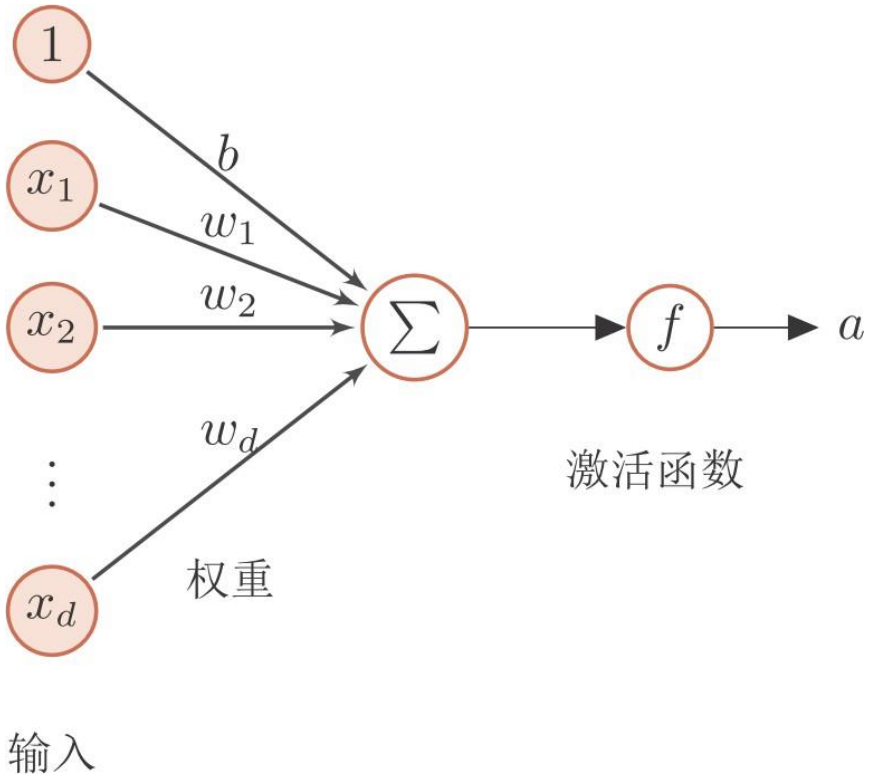
$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$



零中心化的输出将对神经网络的学习更友好
（否则会导致下一层神经元的输入发生偏移，影响梯度更新收敛速度和效率）

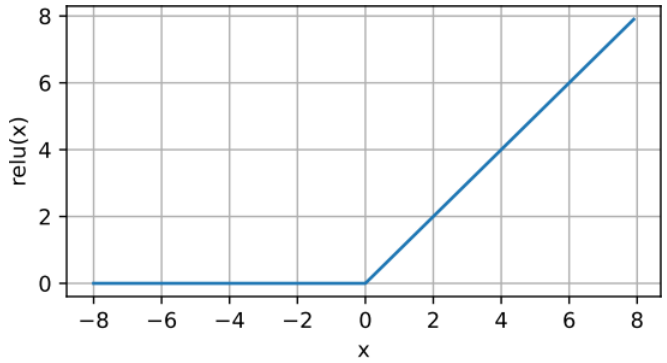
神经元

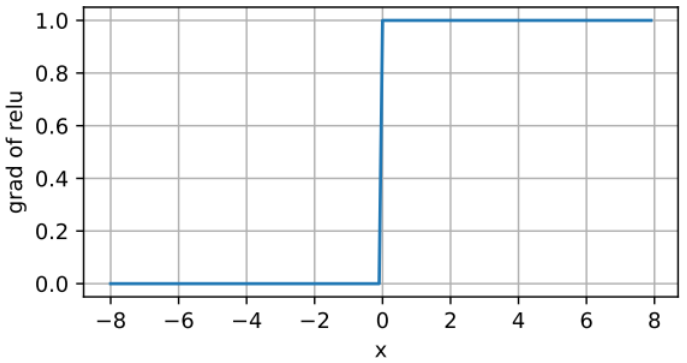
[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



★ 修正线性单元（Rectified Linear Unit，ReLU）

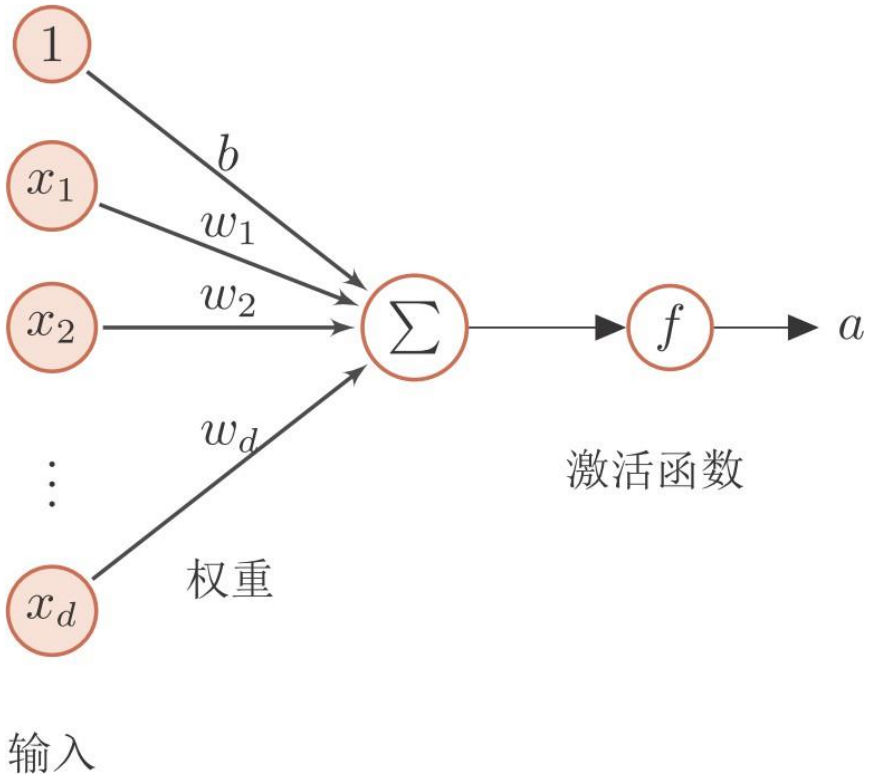
$$\mathrm{ReLU}(z) = \max(0, z)$$





神经元

[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。

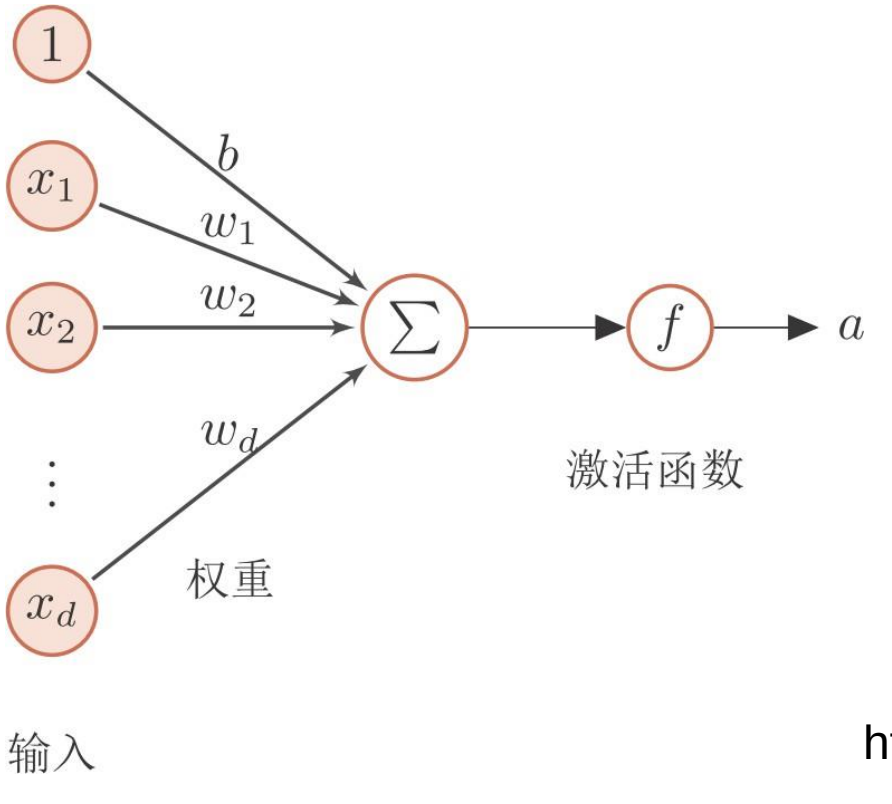


★ 带泄露的ReLU（LeakyReLU）

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma x & \text{if } x \leq 0 \end{cases}$$
$$= \max(0, x) + \gamma \min(0, x)$$

神经元

[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



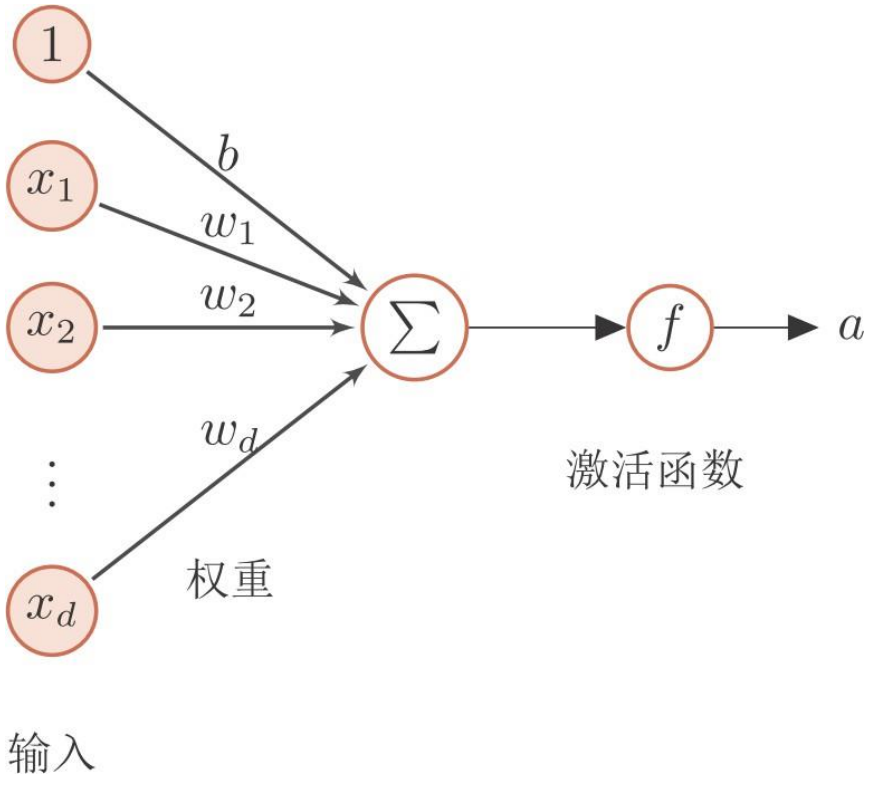
★ 指数线性单元（Exponential Linear Unit，ELU）

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \gamma(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$
$$= \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

https://pytorch.org/docs/stable/generated/torch.nn.ELU.html

神经元

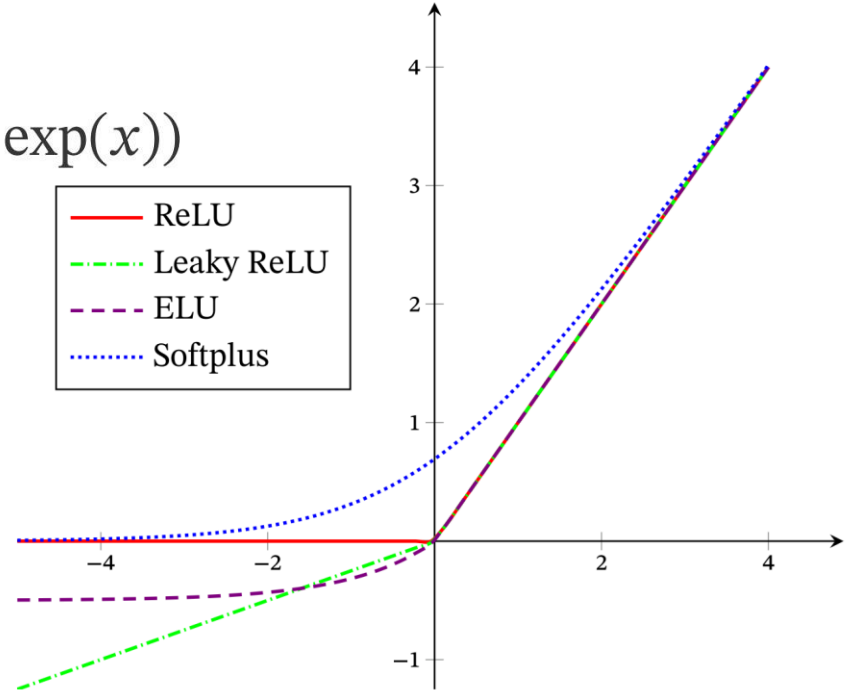
[McCulloch and Pitts, 1943] 设计了“M-P 神经元模型”来模拟生物神经系统的神经元功能.
★偏置 bias 有时候也会被用阈值来刻画，两者等价。



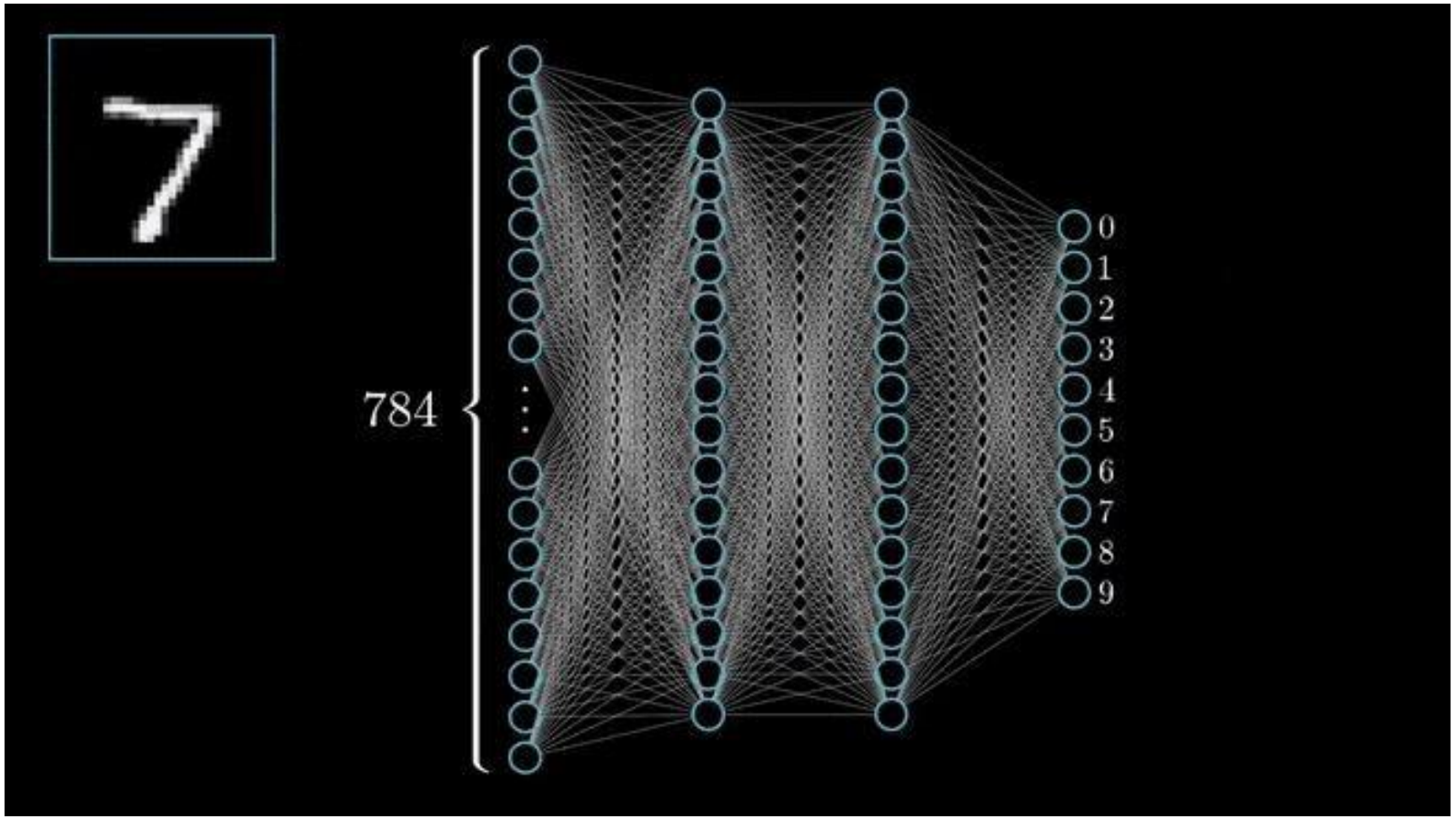
SoftPlus is a smooth approximation to the ReLU function and can be used to constrain the output of a machine to always be positive.

★ Softplus 函数

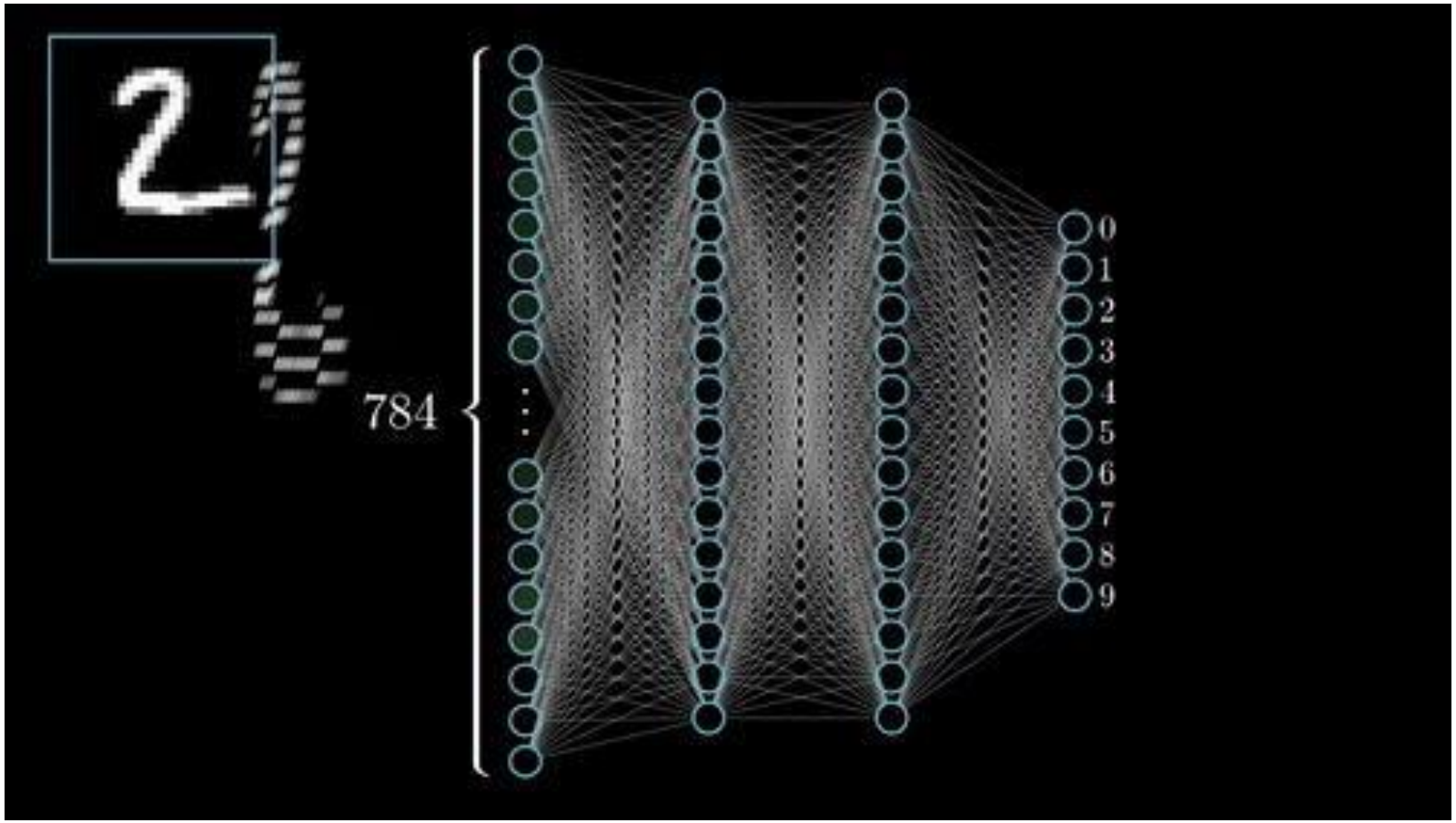
$$\text{Softplus}(x) = \log(1 + \exp(x))$$



全连接前馈神经网络

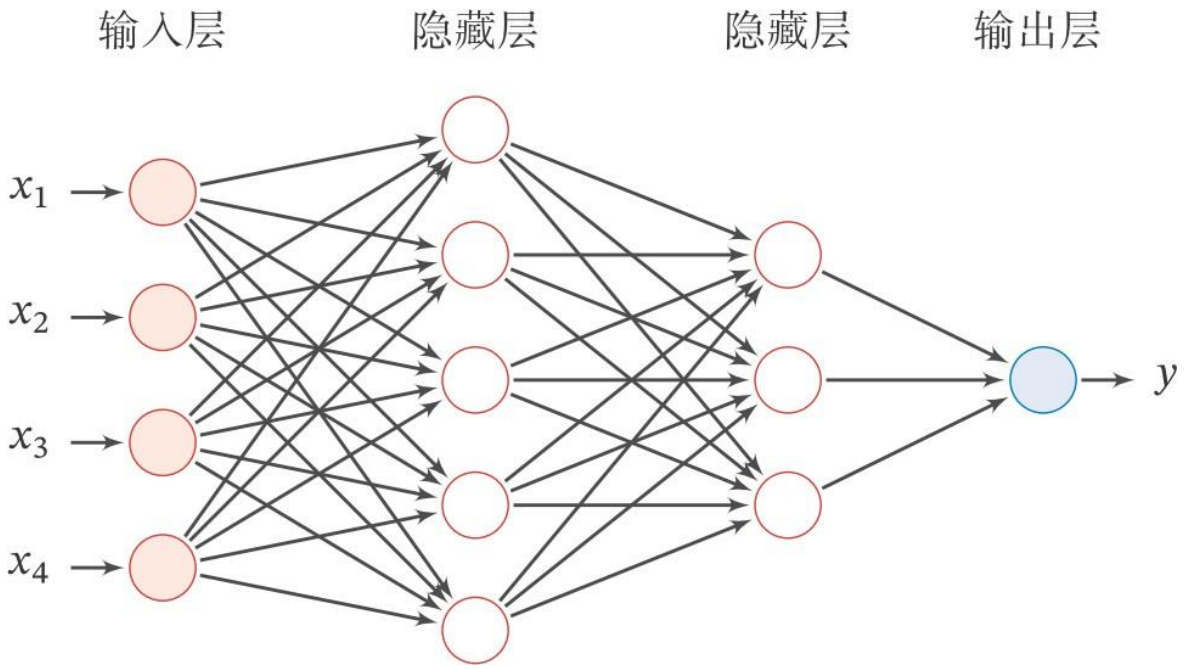


全连接前馈神经网络



全连接前馈神经网络

和上一层fully-connected, 层内神经元相互独立



L	神经网络的层数
M_l	第 l 层神经元的个数
$f_l(\cdot)$	第 l 层神经元的激活函数
$\boldsymbol{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 l 层的权重矩阵
$\boldsymbol{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 l 层的偏置
$\boldsymbol{z}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的净输入（净活性值）
$\boldsymbol{a}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的输出（活性值）

$$\boldsymbol{a}^{(0)} = \boldsymbol{x}$$

$$\boldsymbol{a}^{(l)} = f_l(\boldsymbol{z}^{(l)})$$

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$$

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \rightarrow \boldsymbol{z}^{(1)} \rightarrow \boldsymbol{a}^{(1)} \rightarrow \boldsymbol{z}^{(2)} \rightarrow \cdots$$

$$\rightarrow \boldsymbol{a}^{(L-1)} \rightarrow \boldsymbol{z}^{(L)} \rightarrow \boldsymbol{a}^{(L)} = \phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) \longrightarrow \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$$

本质就是在做矩阵运算！需要用梯度反向传播去更新模型参数（参数矩阵）

反向传播

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \rightarrow \boldsymbol{z}^{(1)} \rightarrow \boldsymbol{a}^{(1)} \rightarrow \boldsymbol{z}^{(2)} \rightarrow \cdots \rightarrow \boldsymbol{a}^{(L-1)} \rightarrow \boldsymbol{z}^{(L)} \rightarrow \boldsymbol{a}^{(L)} \rightarrow \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$$

$$\delta^{(1)} \leftarrow \cdots \cdots \cdots \cdots \cdots \cdots \cdots \leftarrow \delta^{(L-1)} \leftarrow \delta^{(L)}$$

（1） 前馈计算每一层的净输入 $\boldsymbol{z}^{(l)}$ 和激活值 $\boldsymbol{a}^{(l)}$,直到最后一层;

（2） 反向传播计算每一层的误差项 $\delta^{(l)}$;

（3） 计算每一层参数的偏导数,并更新参数.

自动微分（Automatic Differentiation, AD)

通过编程语言中的函数来表示数学函数，然后利用链式法则自动计算导数。这种方法的优点是可以直接在计算机上实现复杂的函数和多变量函数的导数计算，而不需要人工进行符号推导。

神经网络Python实践

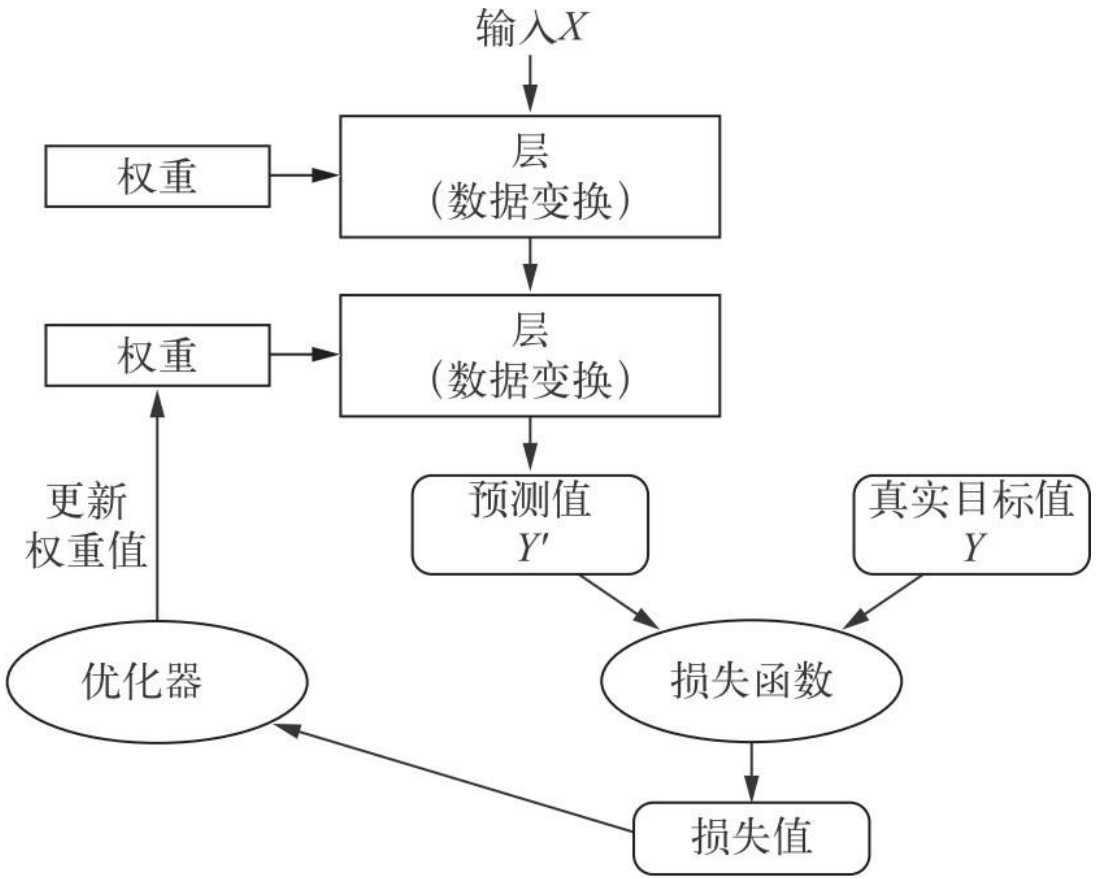
神经网络Python实践

- 步骤

- 准备数据
- 定义由层+结构刻画的网络
- 设定学习细节：

1 损失函数
2 优化方法
3 评估指标

- 网络学习
- 网络使用与存储



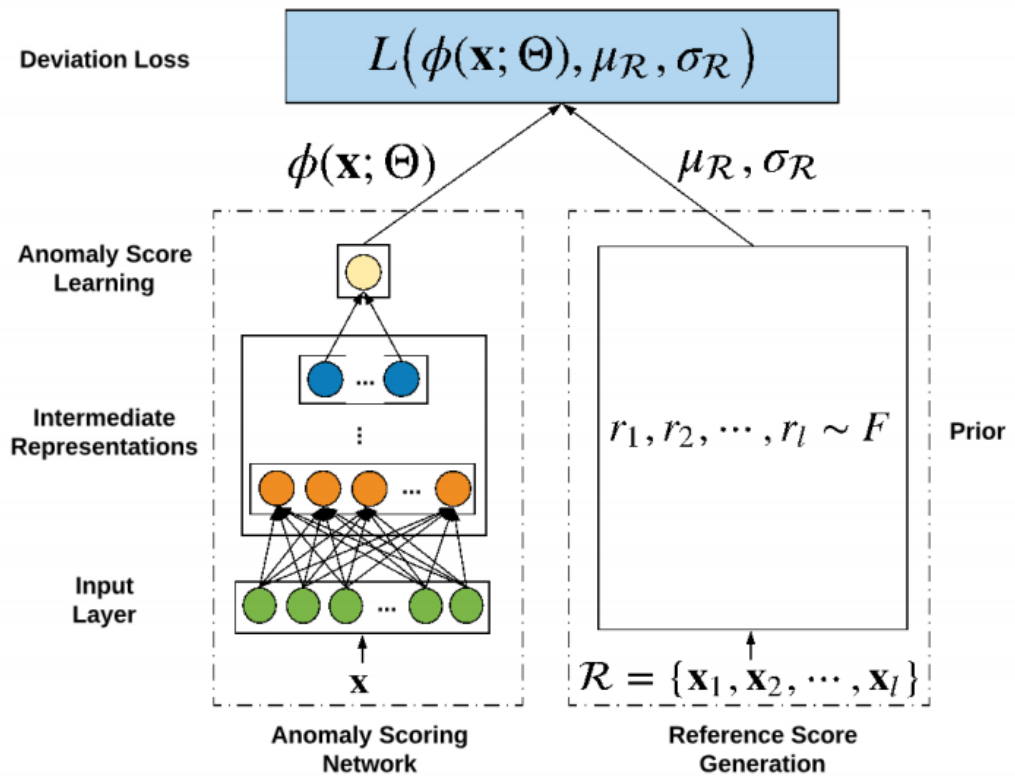
神经网络Python实践

代码示例：

deep-learning-with-python-notebooks

异常检测：能否进一步提升？——end-to-end structure

DevNet, Guansong Pang et al. KDD 2019



- 另外一种对于异常检测进行建模的方式在于构建端到端（end-to-end）的模型，这种方法训练神经网络直接学习映射f(X)=score，而不是采用auto-encoder或者GANomaly的two-stage方法，先重构，再计算score。
- End-to-end方式理论上能获得更好的效果，模型的训练效果与检测性能关联性更强。

异常检测：能否进一步提升？——end-to-end structure

Table 1: AUC-ROC and AUC-PR Performance (with ± standard deviation) of DevNet and Four Competing Methods. #obj. is the overall data size, D is the dimensionality size, f_1 and f_2 denote the percentage that the labeled anomalies respectively comprise in the training data and the total anomalies. D in *URL* and *news20*, i.e., '3M' and '1M', are short for 3,231,961 and 1,355,191, respectively. The best performance is boldfaced.

Data Characteristic					AUC-ROC Performance					AUC-PR Performance				
Data	#obj.	D	f_1	f_2	**DevNet**	**REPEN**	**DSVDD**	**FSNet**	**iForest**	**DevNet**	**REPEN**	**DSVDD**	**FSNet**	**iForest**
donors	619,326	10	0.01%	0.08%	**1.000**±0.000	0.975±0.005	0.995±0.005	0.997±0.002	0.874±0.015	**1.000**±0.000	0.508±0.048	0.846±0.114	0.994±0.002	0.221±0.025
census	299,285	500	0.01%	0.16%	0.828±0.008	0.794±0.005	**0.835**±0.014	0.732±0.020	0.624±0.020	**0.321**±0.004	0.164±0.003	0.291±0.008	0.193±0.019	0.076±0.004
fraud	284,807	29	0.01%	6.10%	**0.980**±0.001	0.972±0.003	0.977±0.001	0.734±0.046	0.953±0.002	**0.690**±0.002	0.674±0.004	0.688±0.004	0.043±0.021	0.254±0.043
celeba	202,599	39	0.02%	0.66%	**0.951**±0.001	0.894±0.005	0.944±0.003	0.808±0.027	0.698±0.020	**0.279**±0.009	0.161±0.006	0.261±0.008	0.085±0.012	0.065±0.006
backdoor	95,329	196	0.04%	1.29%	**0.969**±0.004	0.878±0.007	0.952±0.018	0.928±0.019	0.752±0.021	**0.883**±0.008	0.116±0.003	0.856±0.016	0.573±0.167	0.051±0.005
URL	89,063	3M	0.04%	1.69%	**0.977**±0.004	0.842±0.006	0.908±0.027	0.786±0.047	0.720±0.032	**0.681**±0.022	0.103±0.003	0.475±0.040	0.149±0.076	0.066±0.012
campaign	41,188	62	0.10%	0.65%	**0.807**±0.006	0.723±0.006	0.748±0.019	0.623±0.024	0.731±0.015	**0.381**±0.008	0.330±0.009	0.349±0.023	0.193±0.012	0.328±0.022
news20	10,523	1M	0.37%	5.70%	**0.950**±0.007	0.885±0.003	0.887±0.000	0.578±0.050	0.328±0.016	**0.653**±0.009	0.222±0.004	0.253±0.001	0.082±0.010	0.035±0.002
thyroid	7,200	21	0.55%	5.62%	**0.783**±0.003	0.580±0.016	0.749±0.011	0.564±0.017	0.688±0.020	**0.274**±0.011	0.093±0.005	0.241±0.009	0.116±0.014	0.166±0.017
				Average	**0.916**±0.004	0.838±0.006	0.888±0.011	0.750±0.028	0.708±0.018	**0.574**±0.008	0.263±0.010	0.473±0.025	0.270±0.037	0.140±0.015
				P-value	-	0.004	0.023	0.004	0.004	-	0.004	0.004	0.004	0.004

金融风控案例：用DevNet有效检测信用卡欺诈案例

频率	时间维度	变量维度
最大	交易前10天	交易笔数
平均	交易前5天	入账交易笔数
累计	交易前1天	出账交易笔数
	交易当天至交易时刻	同交易对手交易笔数
	交易前10小时	同交易对手入账笔数
	交易前5小时	同交易对手出账笔数
	交易前3小时	交易对手个数
	交易前1小时	入账交易对手个数
	交易前30分钟	出账交易对手个数
	交易前10分钟	交易金额
	交易前1分钟	入账交易金额
		出账交易金额
		同交易对手交易金额
		同交易对手入账金额
		同交易对手出账金额

模型的输入数据可以用构建衍生变量的形式捕捉宝贵的时序信息，也可以直接利用时序数据，结合能够捕捉序列信息的模型（例如RNN以及Transformer），为下游异常检测任务提供强有力的backbone。

已知的欺诈案例往往是少量、宝贵的，数据类别极端不平衡。合适的损失函数（例如deviation loss）能够显著提升模型效果！

异常检测：能否进一步提升？——end-to-end structure

Anomaly Detection with Score Distribution Discrimination, KDD 2023

https://arxiv.org/pdf/2306.14403