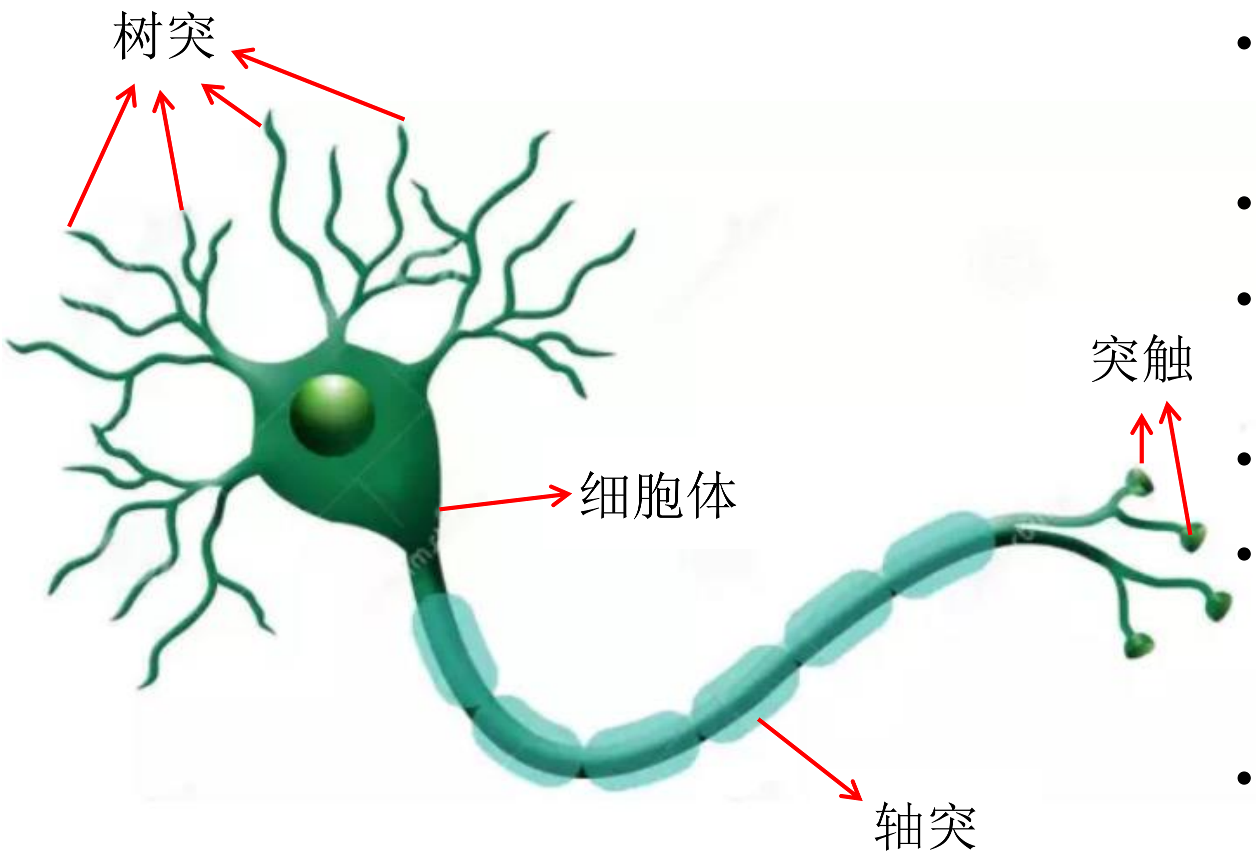
感知机、Adaline与逻辑回归

李 波

神经元模型

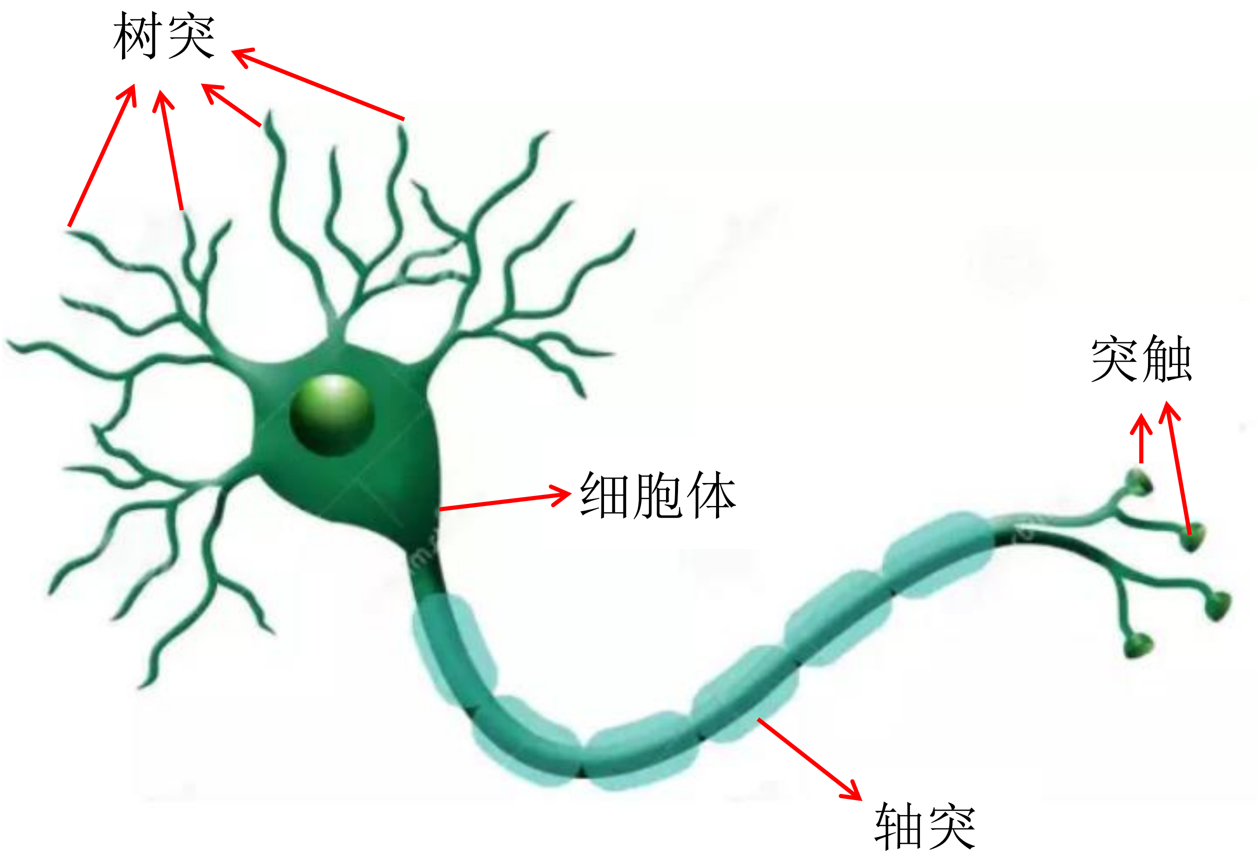
神经元



- 一个神经元细胞由树突、细胞体、轴突和突触构成
- 一个神经元有多个树突，每个树突接收来自其他神经元传递的信号
- 多个树突接收到的信号在细胞体内进行积累
- 当积累的信号超过一个门限时，轴突产生一个信号，轴突产生的信号为二元信号
- 轴突产生的信号通过突触传递给其他神经元
- 首次提出：W.S.McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5(4):115-133, 1943.
- 此神经元模型也被称为MCP神经元模型

The picture is from https://www.icode9.com/content-4-652630.html.

神经元模型



- 树突接收到的信号：$x_1, x_2, \cdots, x_m$
- 细胞体内积累的信号 $\sum_{i=1}^{m} x_i$
- 积累信号与门限值比较。令门限值为θ。如果$\sum_{i=1}^{m} x_i > \theta$，轴突输出1；否则，轴突输出0。

$$\text{轴突输出} = \begin{cases} 1 & \text{如果 } \sum_{i=1}^{m} x_i > \theta \\ 0 & \text{如果 } \sum_{i=1}^{m} x_i \le \theta \end{cases}$$

$$\Downarrow$$

$$\text{轴突输出} = \begin{cases} 1 & \text{如果 } \sum_{i=1}^{m} x_i - \theta > 0 \\ 0 & \text{如果 } \sum_{i=1}^{m} x_i - \theta \le 0 \end{cases}$$

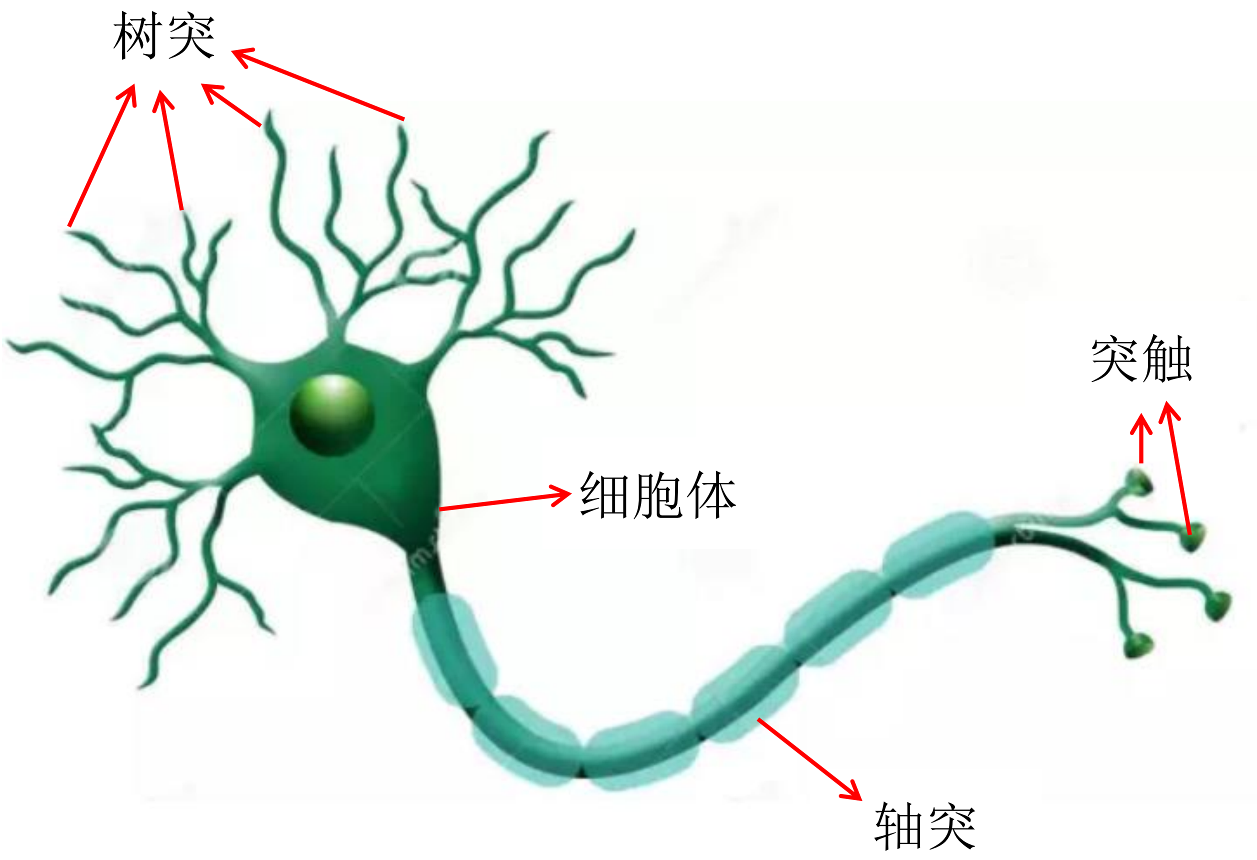
$$\Downarrow$$

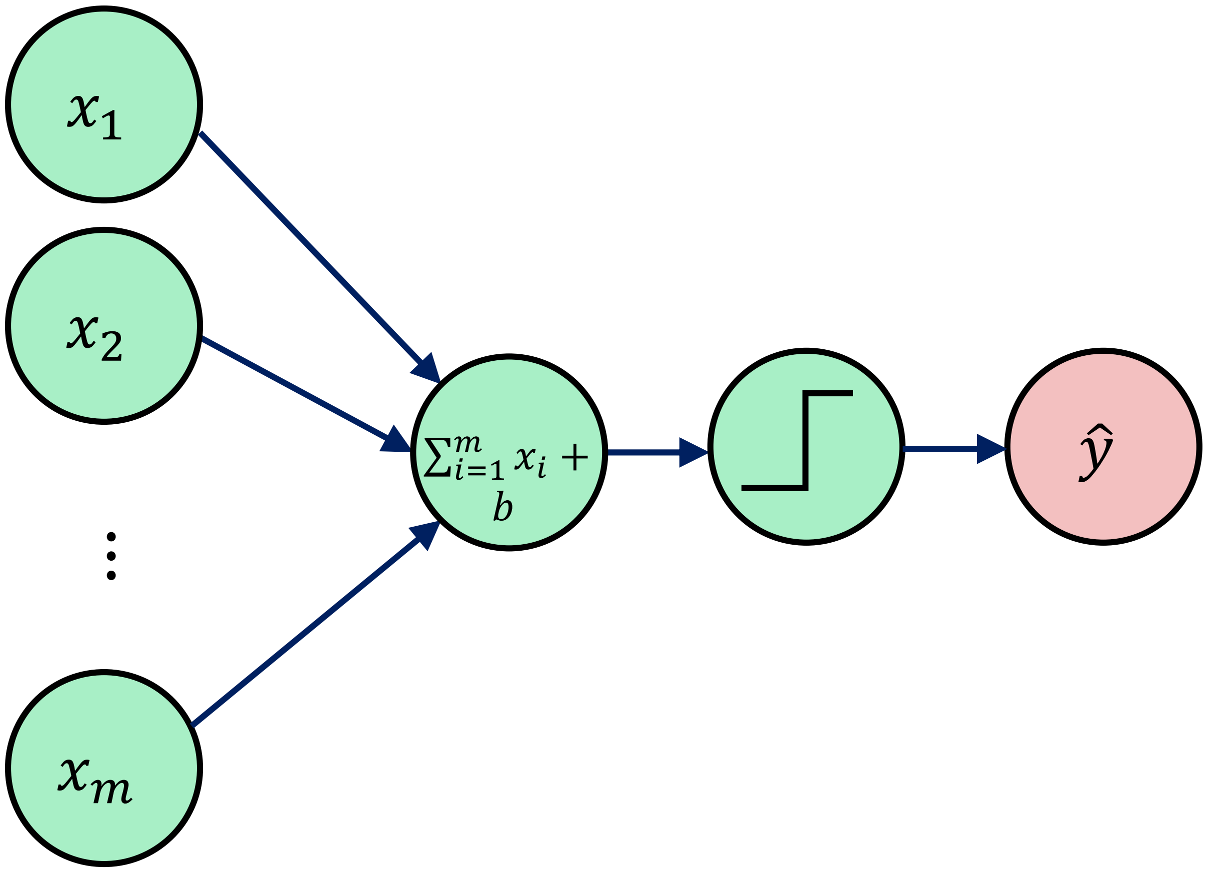
$b = -\theta$

$$\text{轴突输出} = \begin{cases} 1 & \text{如果 } \sum_{i=1}^{m} x_i + b > 0 \\ 0 & \text{如果 } \sum_{i=1}^{m} x_i + b \le 0 \end{cases}$$

The picture is from https://www.icode9.com/content-4-652630.html.

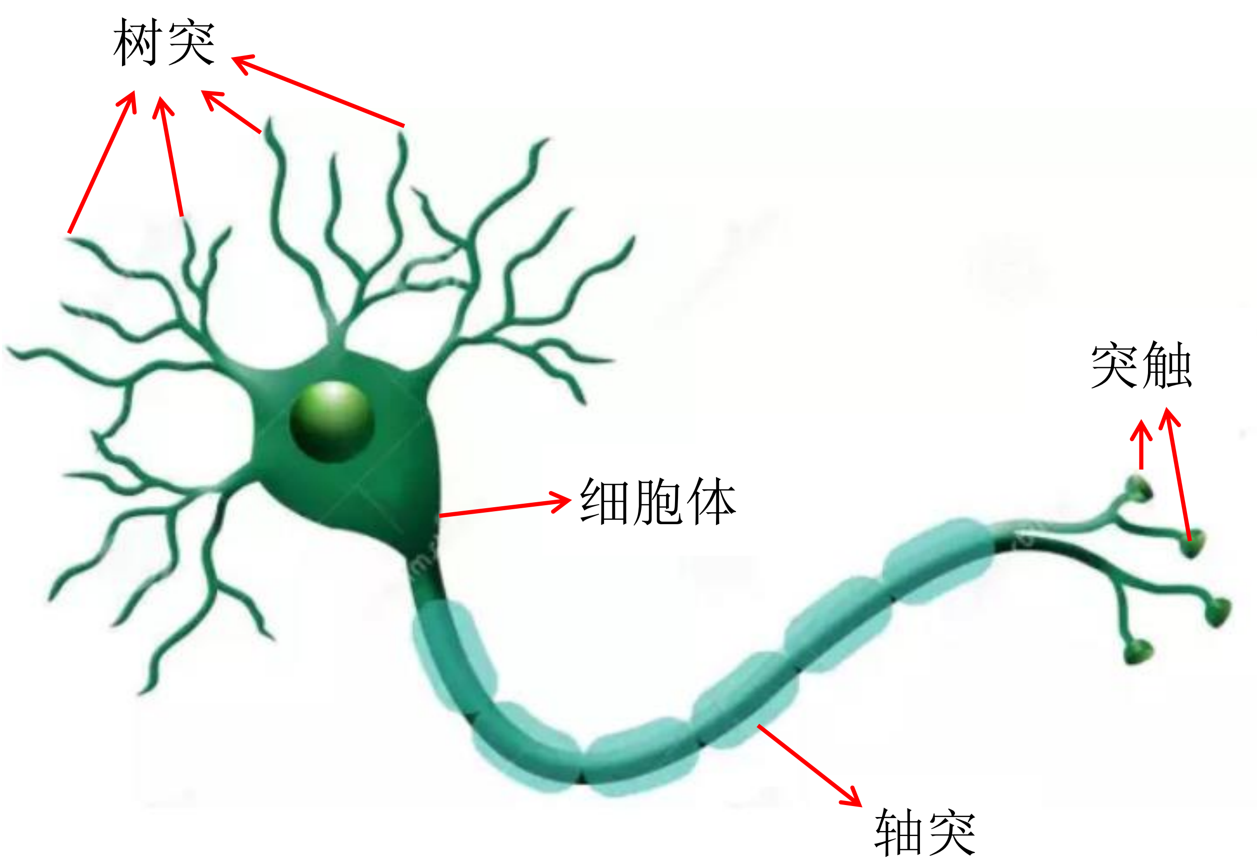
神经元模型

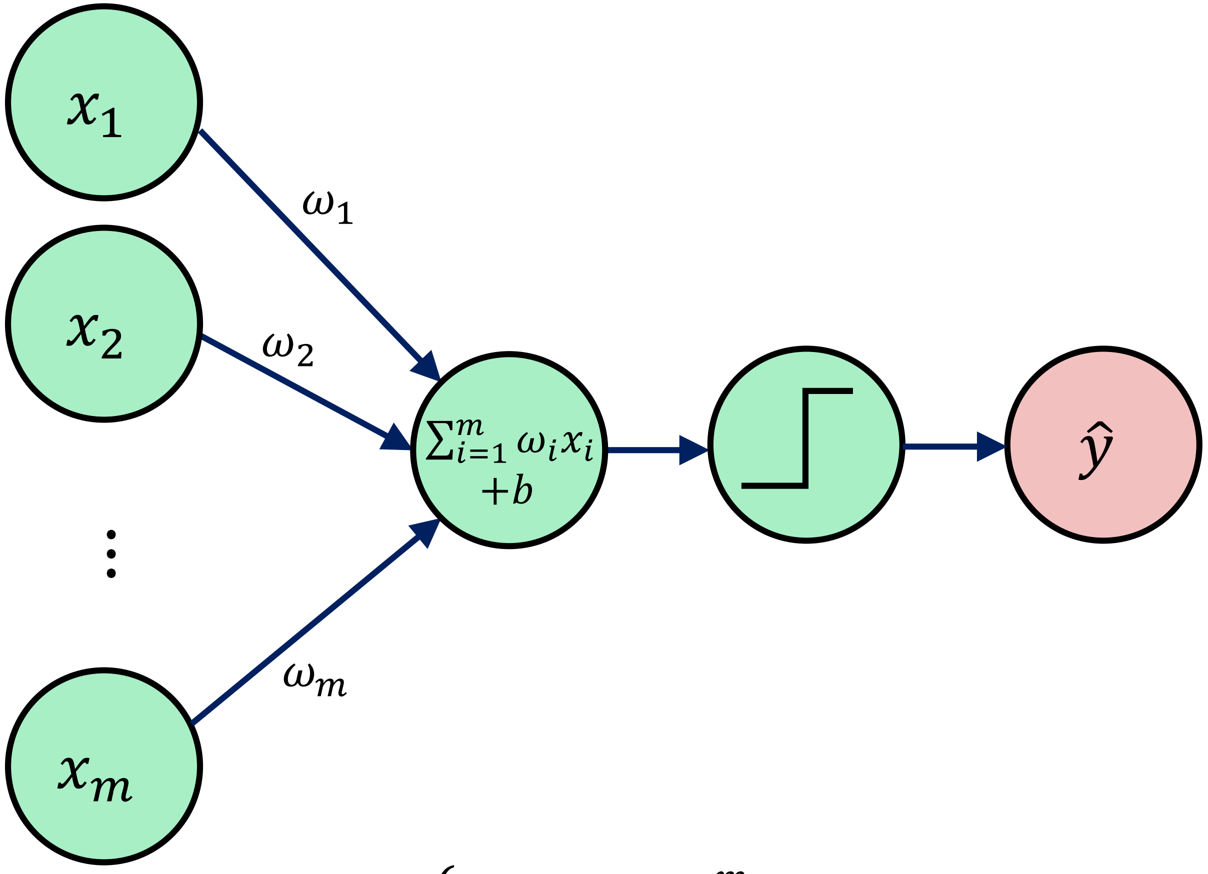




The picture is from https://www.icode9.com/content-4-652630.html.

神经元模型





$$\hat{y} = \begin{cases} 1 & \text{如果} \sum_{i=1}^{m} \omega_i x_i + b > 0 \\ 0 & \text{如果} \sum_{i=1}^{m} \omega_i x_i + b \leq 0 \end{cases}$$

The picture is from https://www.icode9.com/content-4-652630.html.

神经元模型

$$\hat{y} = \sigma(\boldsymbol{\omega}^T \boldsymbol{x} + b)$$

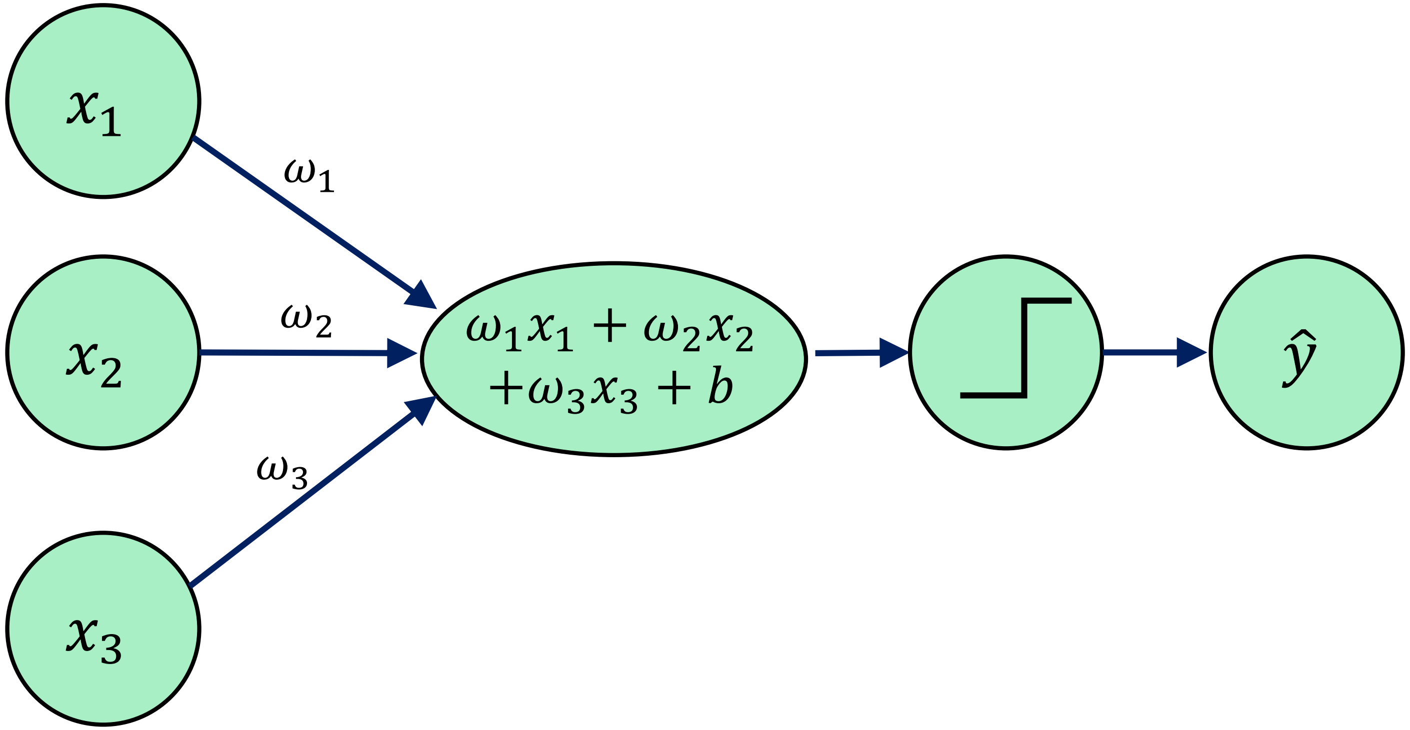
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

- 线性变换加偏置项

$$z = \boldsymbol{\omega}^T \boldsymbol{x} + b$$

- 非线性变换

$$\hat{y} = \sigma(z)$$



感知机

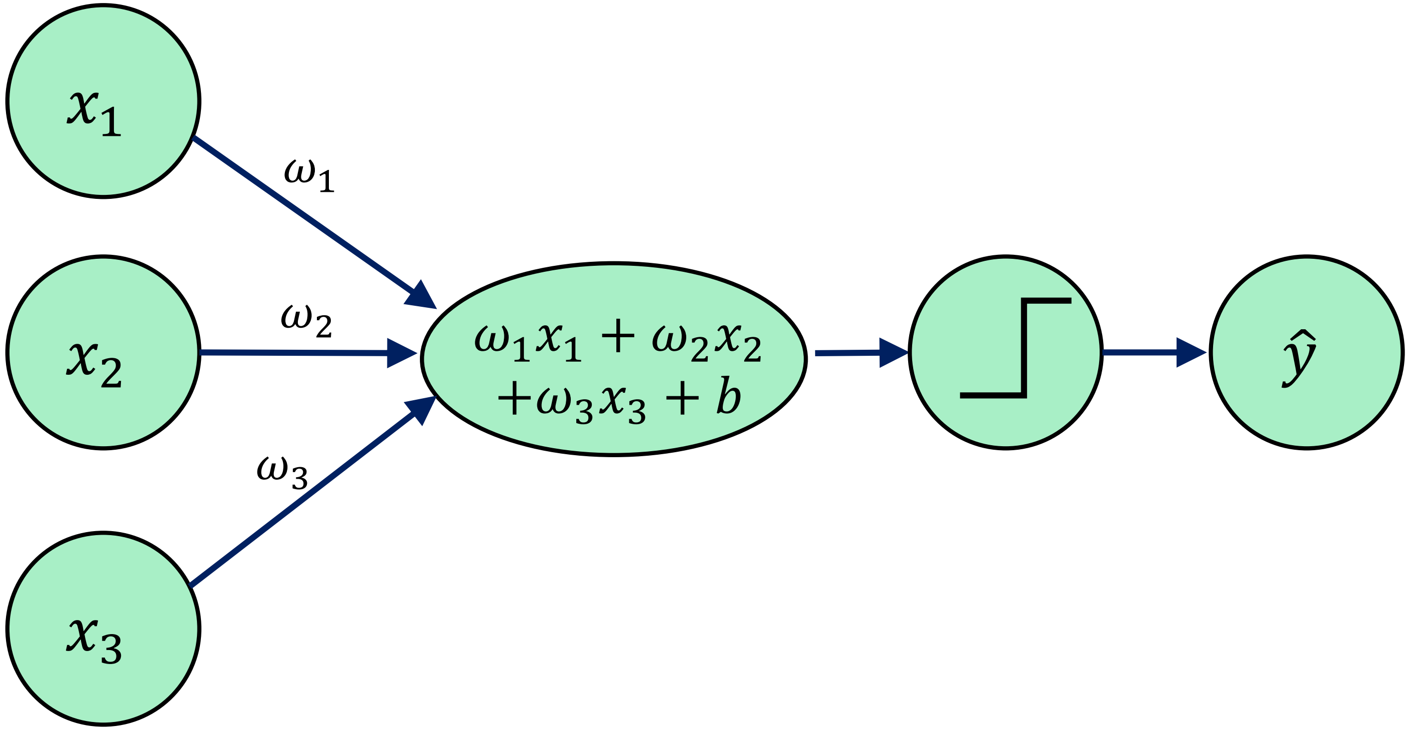
- 单个神经元模型也被称为感知机。

- 感知机根据每个样本，进行如下参数更新

$$\boldsymbol{\omega} = \boldsymbol{\omega} + \Delta\boldsymbol{\omega}$$
$$b = b + \Delta b$$

，其中η为学习率，

$$\Delta\boldsymbol{\omega} = \eta(y_i - \hat{y}_i)\boldsymbol{x}_i$$
$$\Delta b = \eta(y_i - \hat{y}_i)$$



感知机

感知机参数更新

$$\boldsymbol{\omega} = \boldsymbol{\omega} + \Delta\boldsymbol{\omega} \qquad \Delta\boldsymbol{\omega} = \eta(y_i - \hat{y}_i)\boldsymbol{x}_i$$

$$b = b + \Delta b \qquad \Delta b = \eta(y_i - \hat{y}_i)$$

$$\hat{y} = \begin{cases} 1 & \text{如果} \sum_{i=1}^{m} \omega_i x_i + b > 0 \\ 0 & \text{如果} \sum_{i=1}^{m} \omega_i x_i + b \leq 0 \end{cases}$$

- 如果感知机分类正确，即 $y_i = \hat{y}_i$

$$\Delta\boldsymbol{\omega} = \eta(y_i - \hat{y}_i)\boldsymbol{x}_i = 0$$
$$\Delta b = \eta(y_i - \hat{y}_i) = 0$$

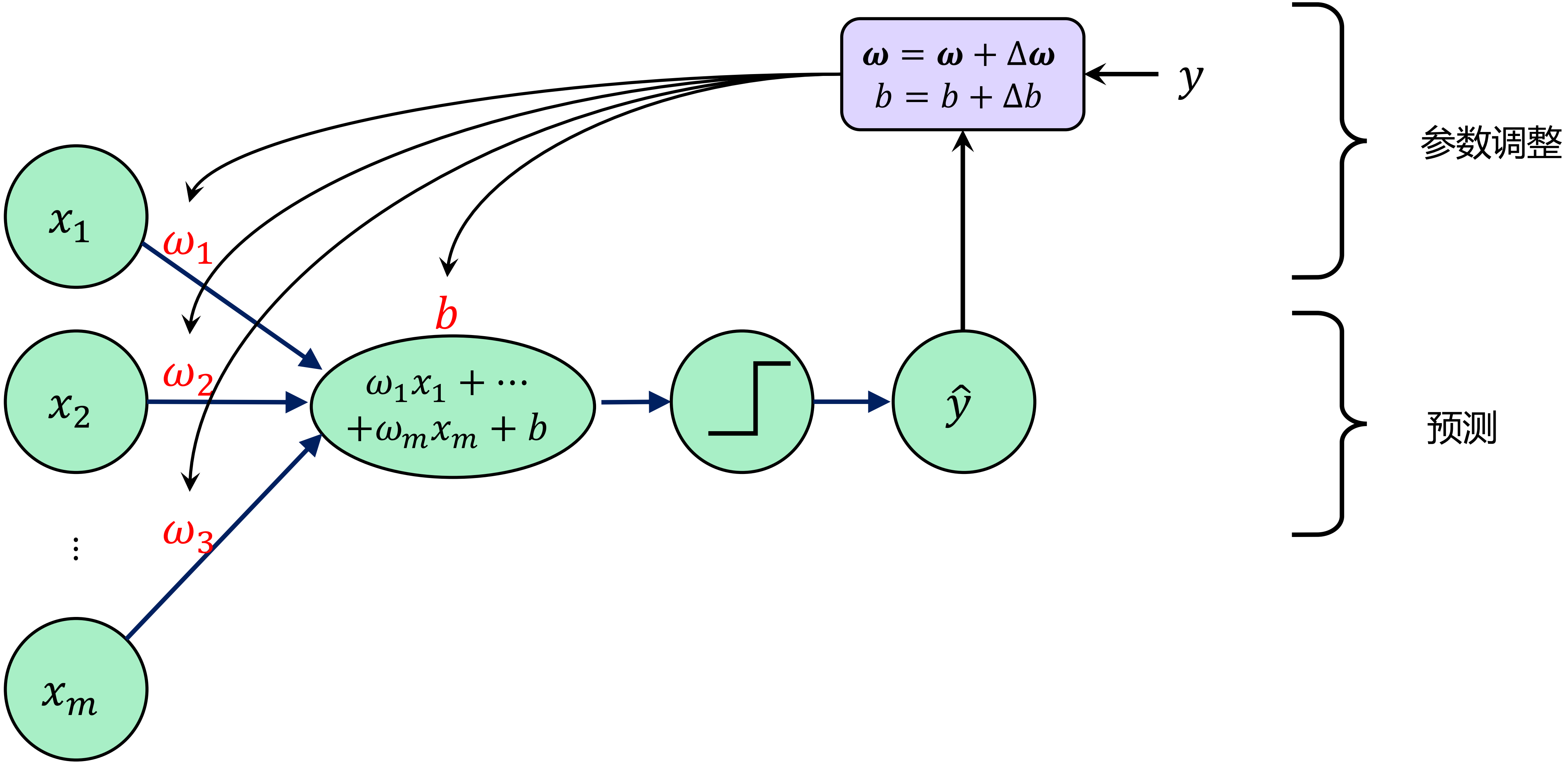
- 如果感知机分类错误， $y_i = 1$， $\hat{y}_i = 0$，此时$\boldsymbol{\omega}^T\boldsymbol{x}_i + b < 0$

$$\Delta\boldsymbol{\omega} = \eta(y_i - \hat{y}_i)\boldsymbol{x}_i = \eta\boldsymbol{x}_i$$
$$\Delta b = \eta(y_i - \hat{y}_i) = \eta$$

此时

$$\Delta\boldsymbol{\omega}^T\boldsymbol{x}_i = \eta\boldsymbol{x}_i^T\boldsymbol{x}_i > 0$$
$$\Delta b = \eta > 0$$
$$(\boldsymbol{\omega} + \Delta\boldsymbol{\omega})^T\boldsymbol{x}_i + (b + \Delta b) = \boldsymbol{\omega}^T\boldsymbol{x}_i + b + \Delta\boldsymbol{\omega}^T\boldsymbol{x}_i + \Delta b > \boldsymbol{\omega}^T\boldsymbol{x}_i + b$$

感知机



$\boldsymbol{\omega} = \boldsymbol{\omega} + \Delta\boldsymbol{\omega}$
$b = b + \Delta b$
y
x_1
x_2
$\vdots$
x_m
ω_1
ω_2
ω_3
b
$\omega_1 x_1 + \cdots$
$+\omega_m x_m + b$
$\hat{y}$
参数调整
预测

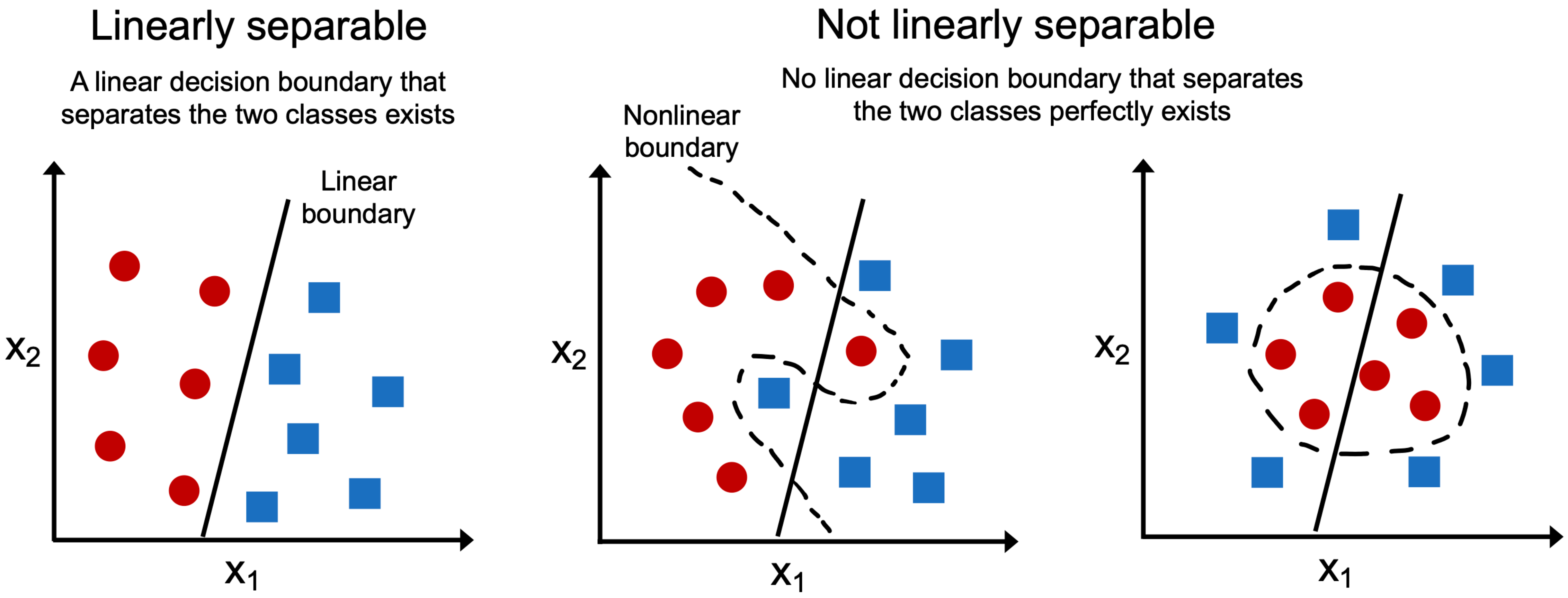
感知机

感知机训练伪代码

输入: 训练数据$\boldsymbol{X}$, $\boldsymbol{y}$, epoches
输出: 参数$\boldsymbol{\omega}$, b.

（1）计算样本个数n，初始化$\boldsymbol{\omega}$, b
（2） for epoch in epochs:
（3） 所有样本乱序
（4） for i in $\{1,2,\cdots n\}$:
（5） $\boldsymbol{\omega} = \boldsymbol{\omega} + \eta(y_i - \hat{y}_i)\boldsymbol{x}_i$
（6） $b = b + \eta(y_i - \hat{y}_i)$
（7）返回$\boldsymbol{\omega}$, b.

感知机



感知机在线性不可分数据，永远不收敛。

The pic is from the book titled *Machine Learning with Pytorch and Scikit-Learn*.

自适应线性神经元

- 自适应线性神经元（Adaptive Linear Neuron, Adaline）

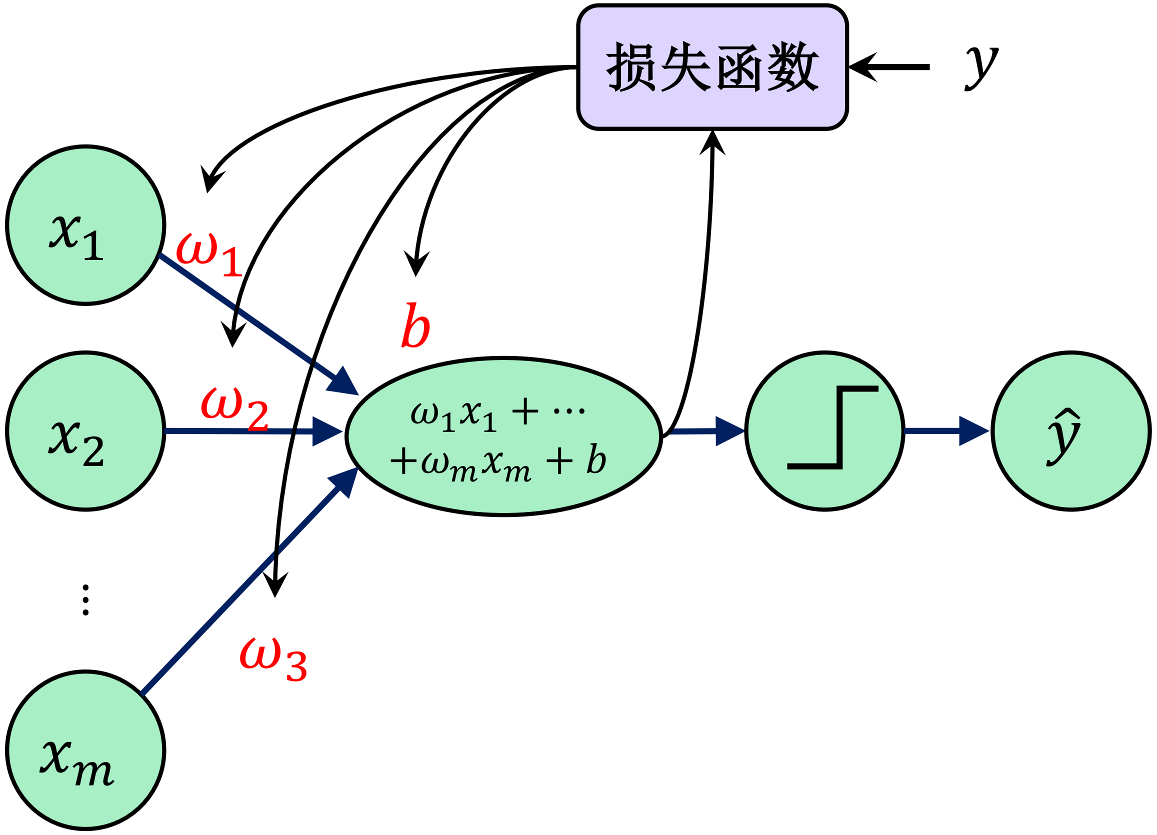
- Adaline结构与感知机结构一致。

- 与感知机的唯一区别在于，使用门限函数的输入，而非输出，调整参数

- 定义损失函数为

$$L(\boldsymbol{\omega}, b) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T\boldsymbol{x}_i - b\right)^2$$

- 寻找$\boldsymbol{\omega}, b$，使损失函数最小。



自适应线性神经元

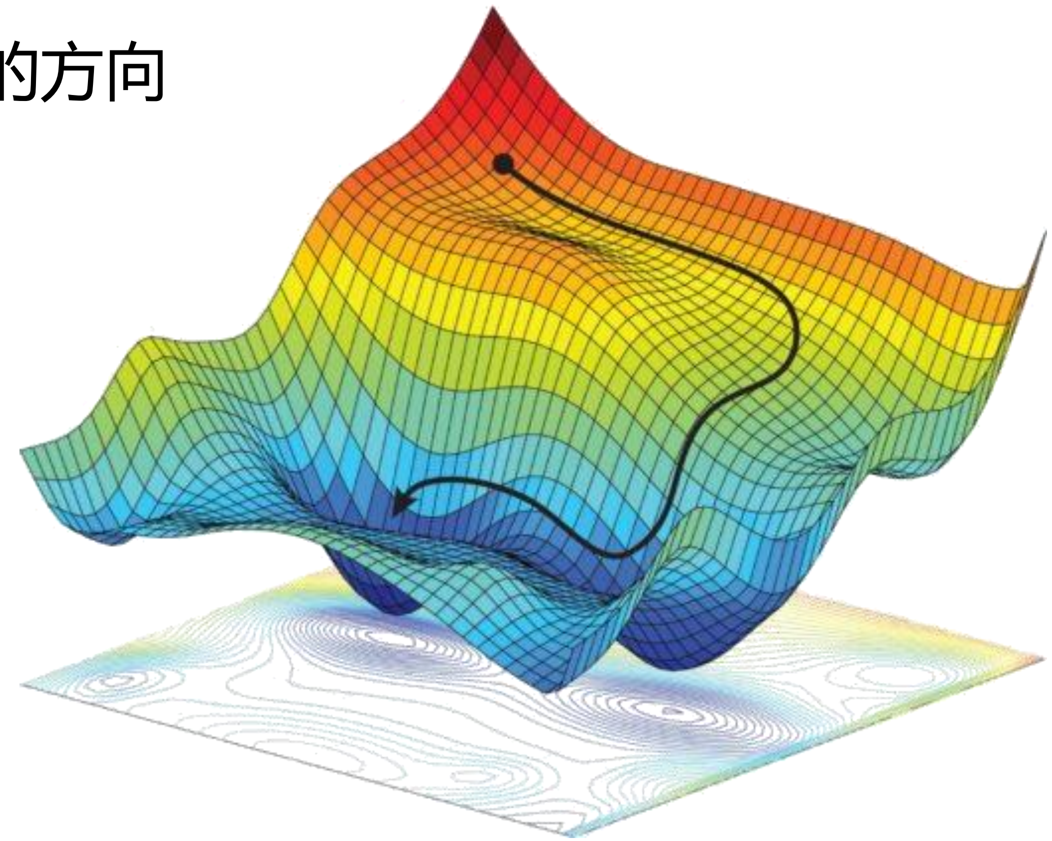
梯度下降: 一种基于迭代更新的方法。

$\frac{dL(\omega,b)}{d\omega}$ 也被称为梯度。梯度是函数值增加最快的方向

$$\omega_{i+1} \leftarrow \omega_i - \gamma \left.\frac{dL(\omega,b)}{d\omega}\right|_{\omega=\omega_i,b=b_i}$$

$$b_{i+1} \leftarrow b_i - \gamma \left.\frac{dL(\omega,b)}{db}\right|_{\omega=\omega_i,b=b_i}$$

γ 为步长



梯度下降法参数更新路径

The pic for "trajectory by gradient descent" is from https://machinelearningknowledge.ai/keras-optimizers-explained-with-examples-for-beginners/.

自适应线性神经元

梯度计算

- 损失函数为

$$L(\boldsymbol{\omega}, b) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right)^2$$

- 梯度为

$$\frac{\partial L(\boldsymbol{\omega}, b)}{\partial \boldsymbol{\omega}} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right)\boldsymbol{x}_i$$

$$\frac{\partial L(\boldsymbol{\omega}, b)}{\partial b} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right)$$

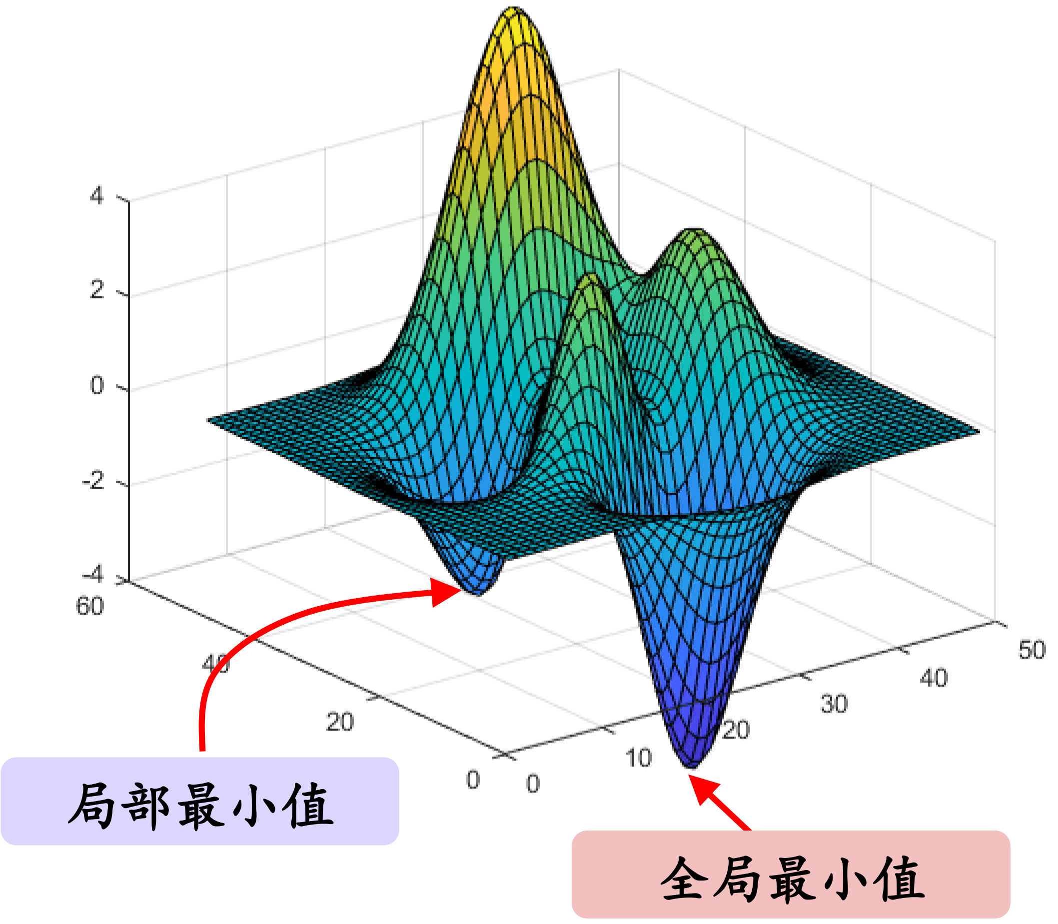
- 使用梯度下降，参数更新为：

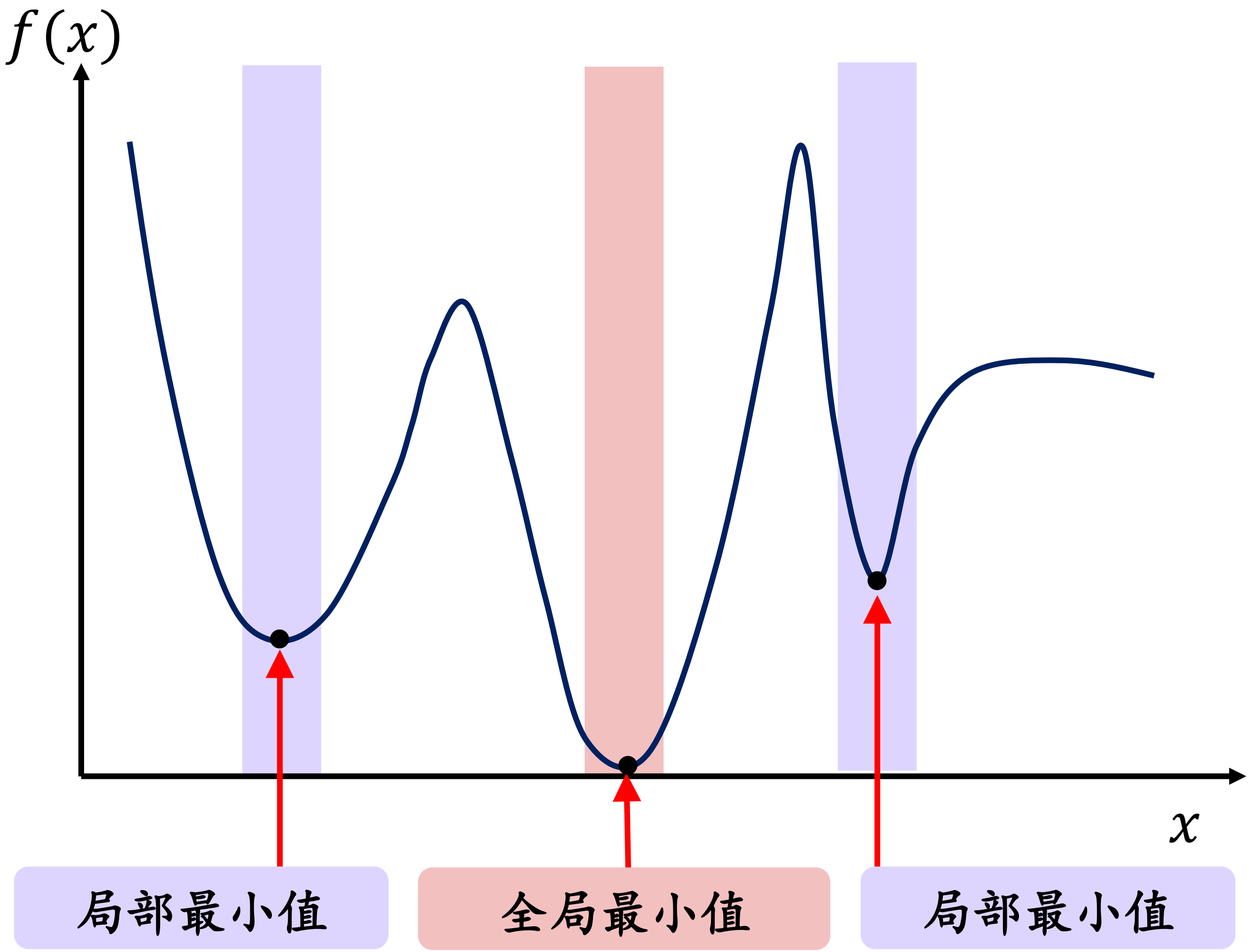
$$\boldsymbol{\omega}_{i+1} \leftarrow \boldsymbol{\omega}_i - \gamma\frac{(-1)}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right)\boldsymbol{x}_i$$

$$b_{i+1} \leftarrow b_i - \gamma\frac{(-1)}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right)$$

自适应线性神经元

- **全局最小值与局部最小值**



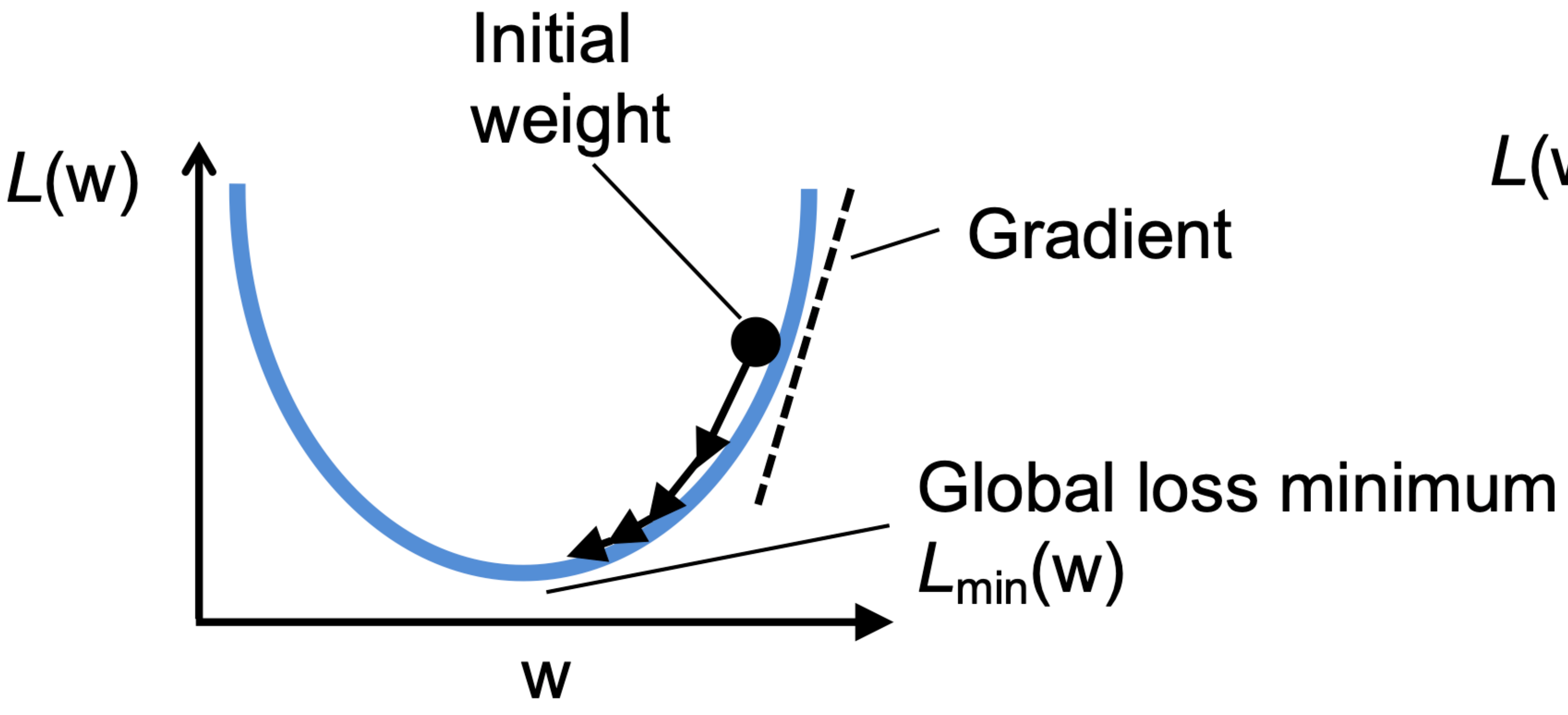


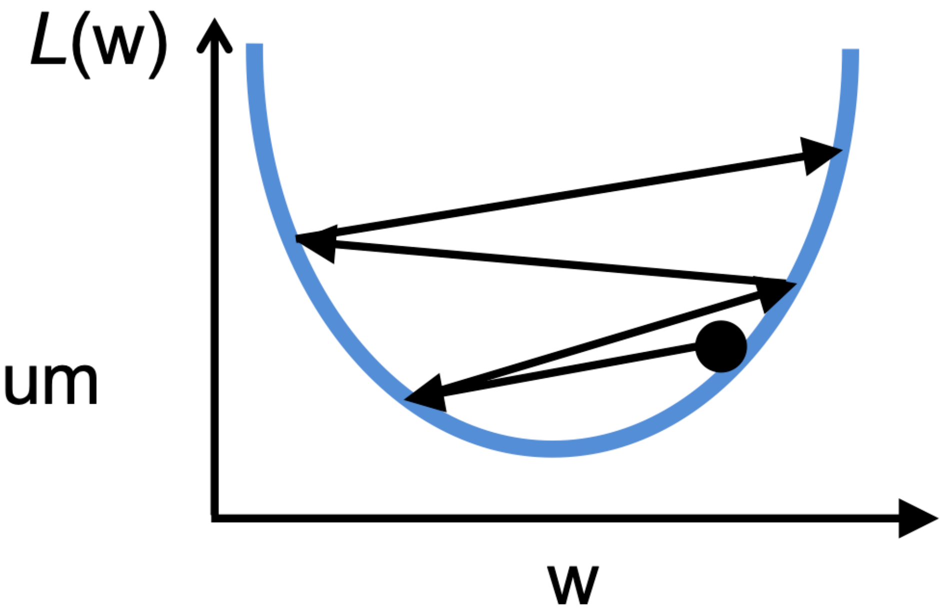
Pic on left is from https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/.

自适应线性神经元

步长的选择

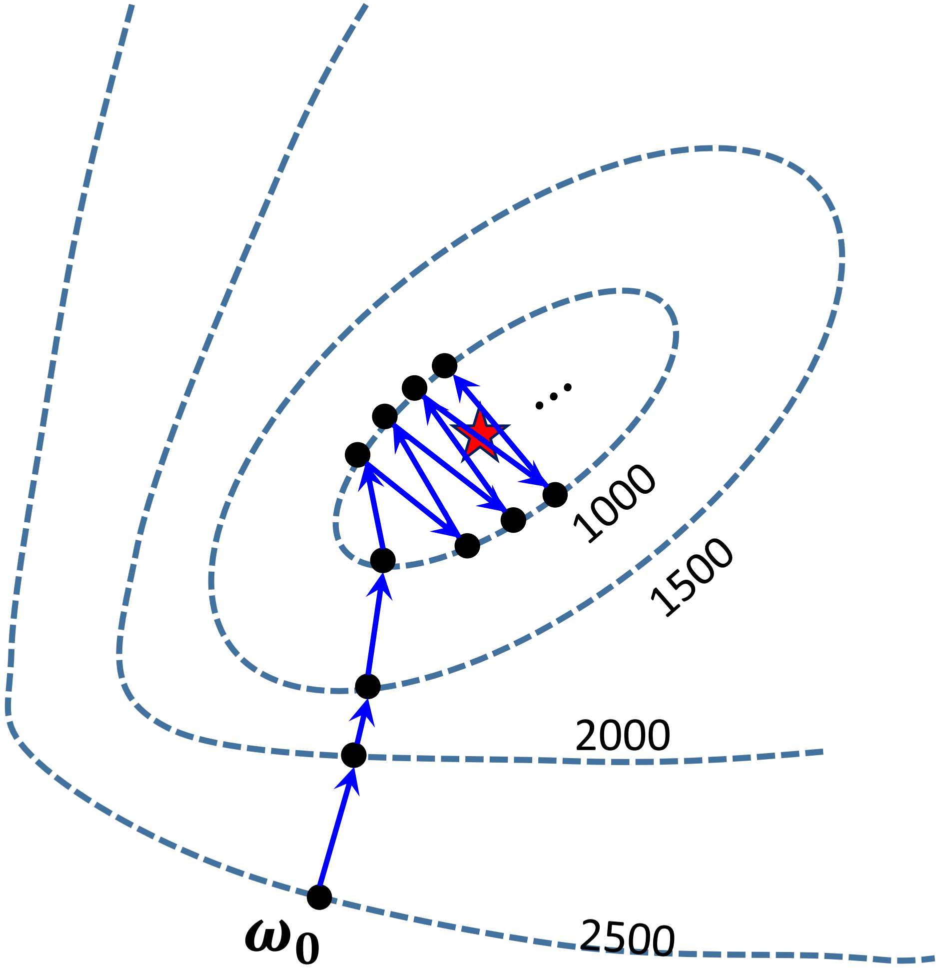
- 步长小，可以保证算法收敛到最小值处，但收敛速度会很慢
- 步长大，会很快找到最小值附近，但会在最小值附近震荡。



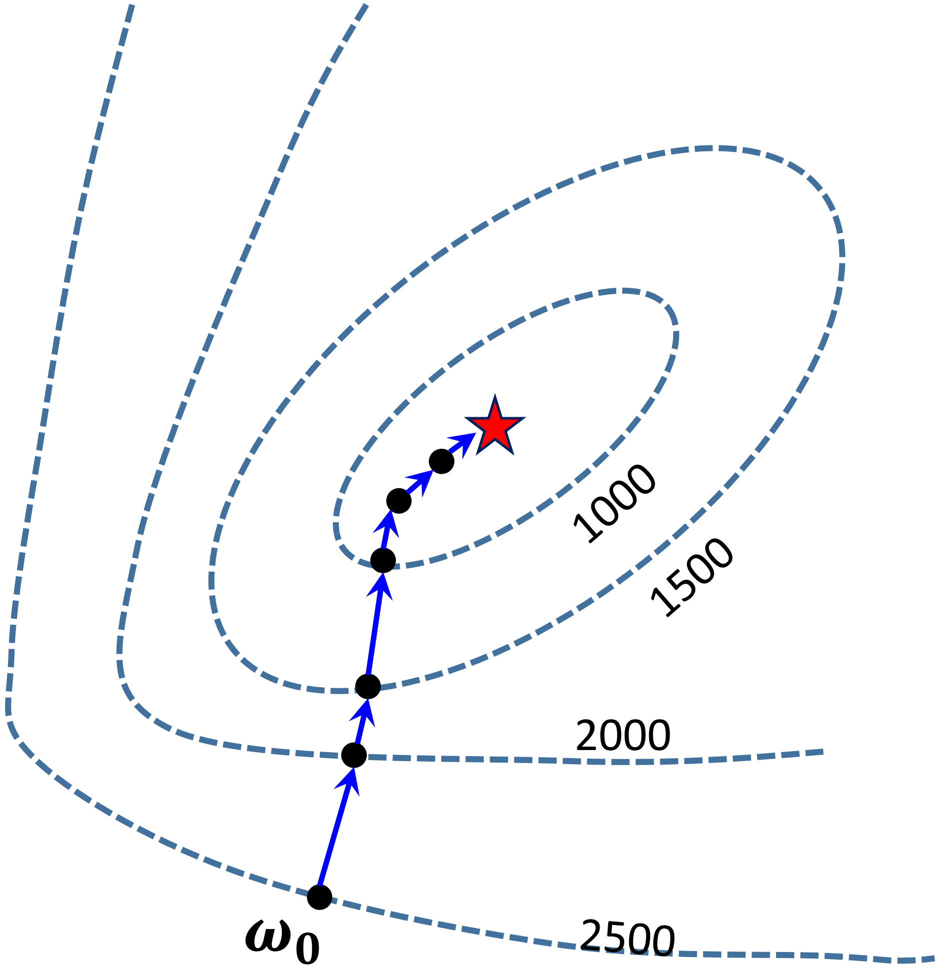


自适应线性神经元

- **固定步长 vs 衰减步长**



固定步长



衰减步长

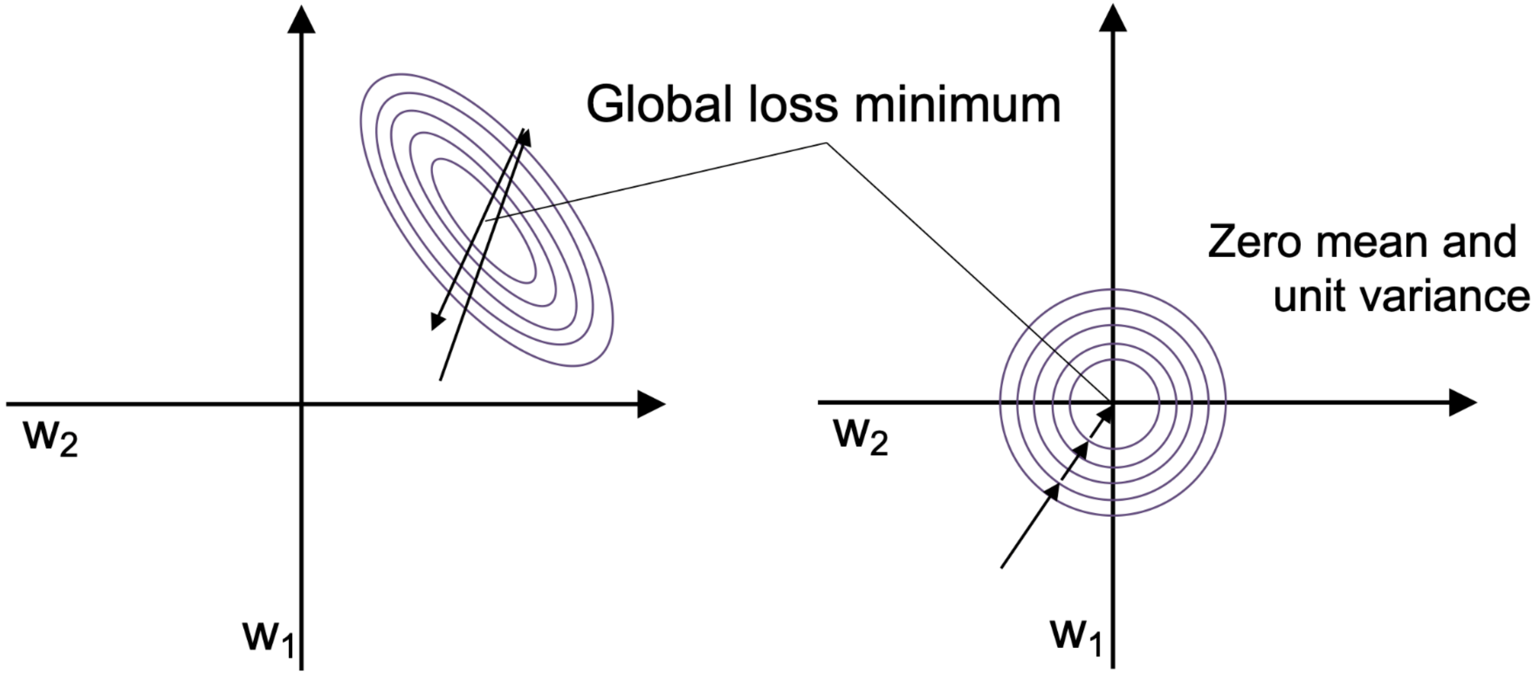
自适应线性神经元

特征的影响

- 如果特征之间相关性大，而且幅度差别很大，那么在每个特征方向使用相同的步长不合适
- **解决方案**：特征标准化

$$\boldsymbol{X}_{.i} = \frac{\boldsymbol{X}_{.i} - \mu_i}{\sigma_i}$$

其中μ_i， σ_i为第i个特征的均值和方差。



自适应线性神经元

随机梯度下降(stochastic gradient descent)

- 梯度下降利用所有训练样本计算梯度。如果训练样本大，计算量巨大。

$$\boldsymbol{\omega}_{i+1} \leftarrow \boldsymbol{\omega}_i - \gamma \frac{(-1)}{n} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right) \boldsymbol{x}_i$$

- 随机梯度下降在每一次迭代中，随机抽取一个训练样本用于估计梯度。

$$\boldsymbol{\omega}_{i+1} \leftarrow \boldsymbol{\omega}_i - \gamma (-1) \left(y^{(t)} - \boldsymbol{\omega}^T \boldsymbol{x}_t - b\right) \boldsymbol{x}_t$$

, 其中t为随机抽选样本的索引。

用于估计梯度

自适应线性神经元

小批随机梯度下降(mini-batch stochastic gradient descent)

- 梯度下降利用所有训练样本计算梯度。如果训练样本大，计算量巨大。

$$\boldsymbol{\omega}_{i+1} \leftarrow \boldsymbol{\omega}_i - \gamma \frac{(-1)}{n} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right) \boldsymbol{x}_i$$

- 小批随机梯度下降在每一次迭代中，随机抽取一部分训练样本用于估计梯度。

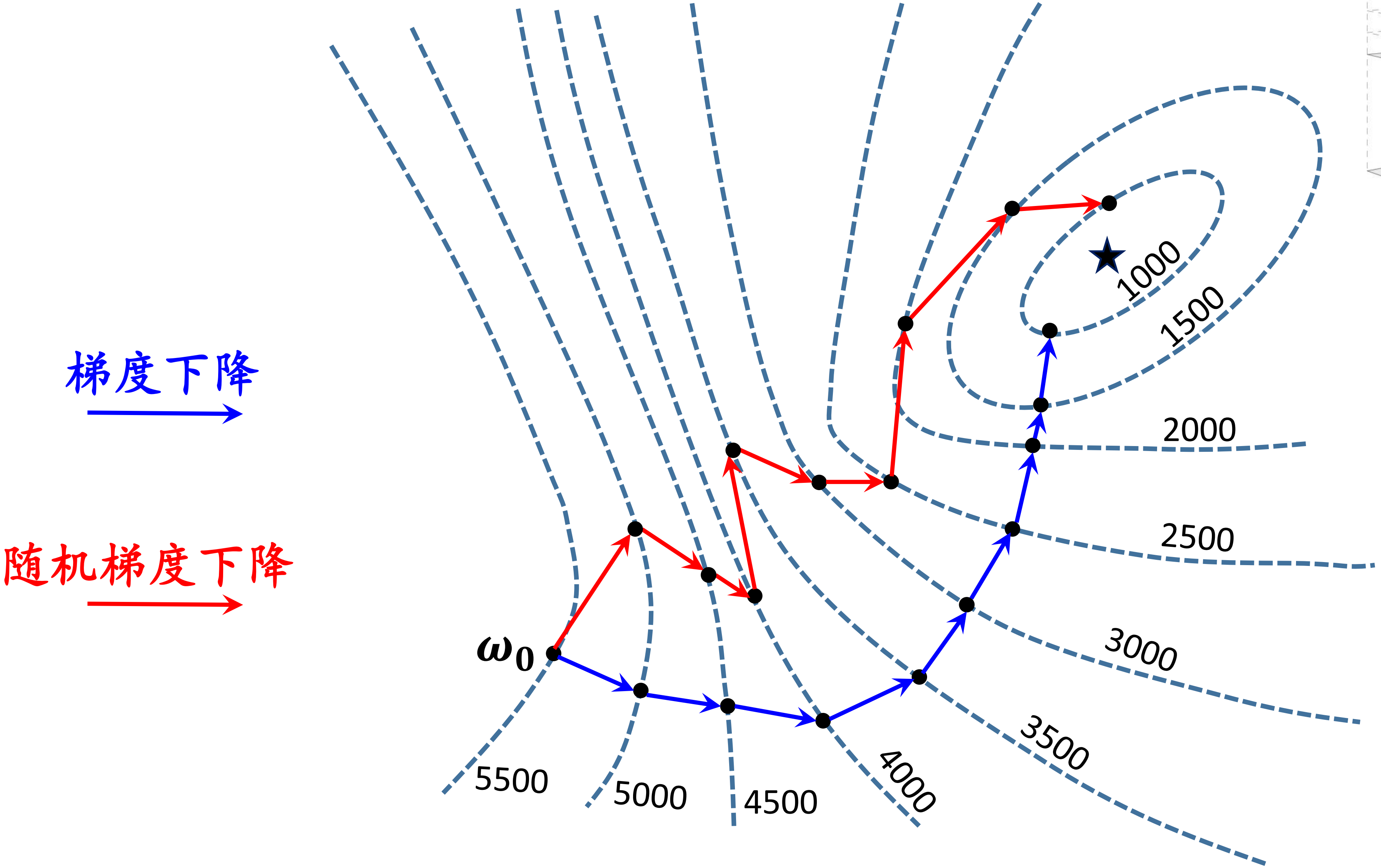
$$\boldsymbol{\omega}_{i+1} \leftarrow \boldsymbol{\omega}_i - \gamma \frac{(-1)}{|\Gamma|} \sum_{i \in \Gamma} \left(y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b\right) \boldsymbol{x}_i$$

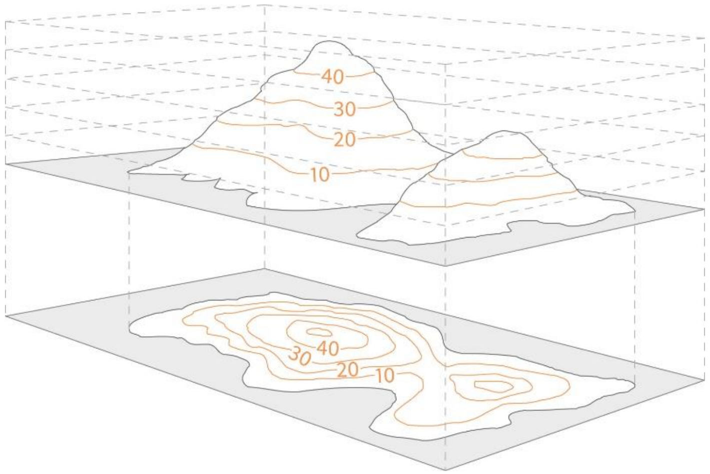
用于估计梯度

, 其中 Γ为随机抽选样本索引的集合。 $|\Gamma|$ 为集合元素个数。

自适应线性神经元

- **梯度下降 vs 小批量随机梯度下降**

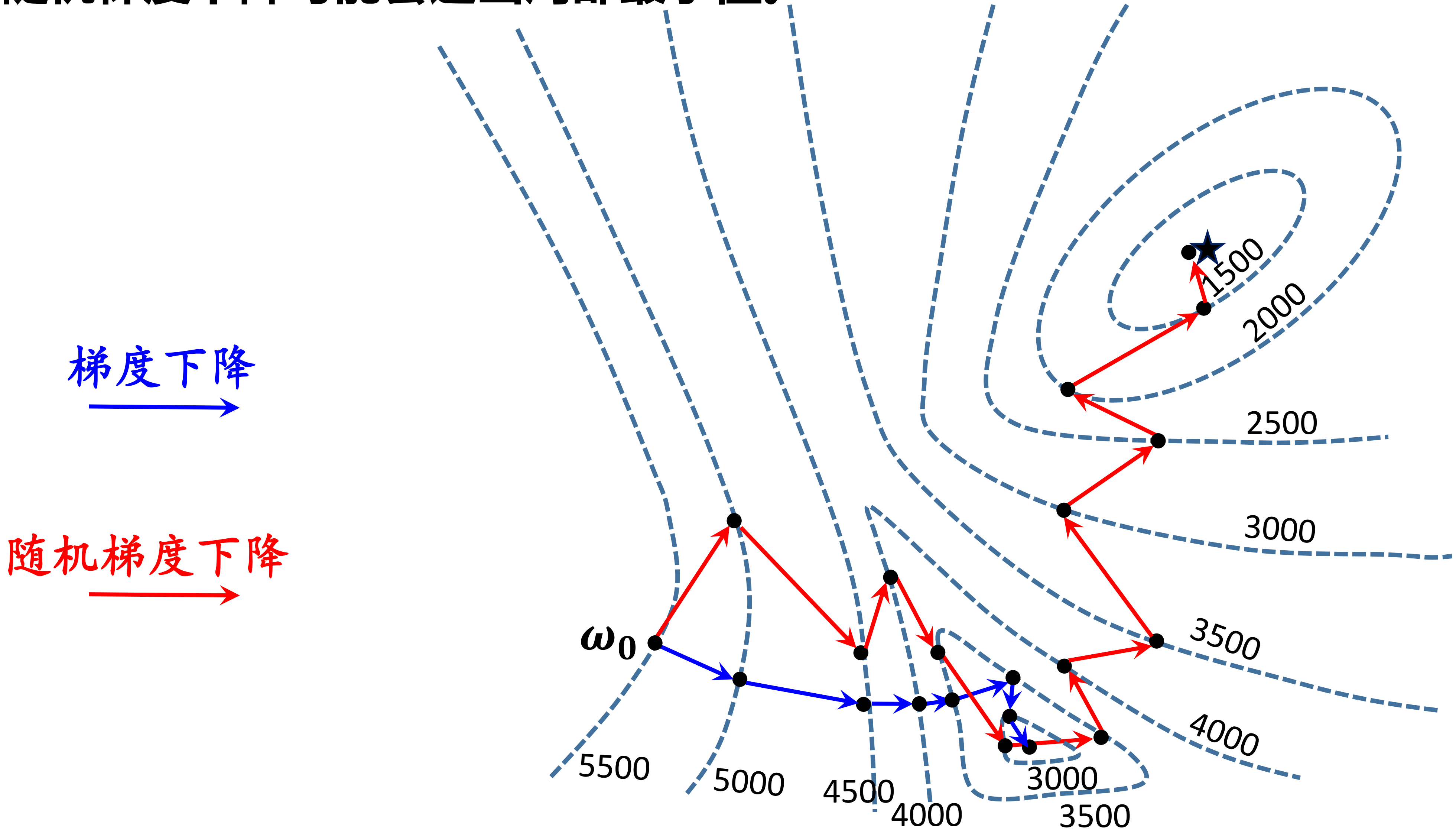




等高线示意图
https://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/

自适应线性神经元

随机梯度下降可能会逃出局部最小值。



自适应线性神经元

Adaline训练伪代码

输入: 训练数据$\boldsymbol{X}, \boldsymbol{y}$
输出: 参数$\boldsymbol{\omega}, b, k$.

（1）计算样本个数n，初始化$\boldsymbol{\omega}, b$

（2） 运行以下代码直至停止:

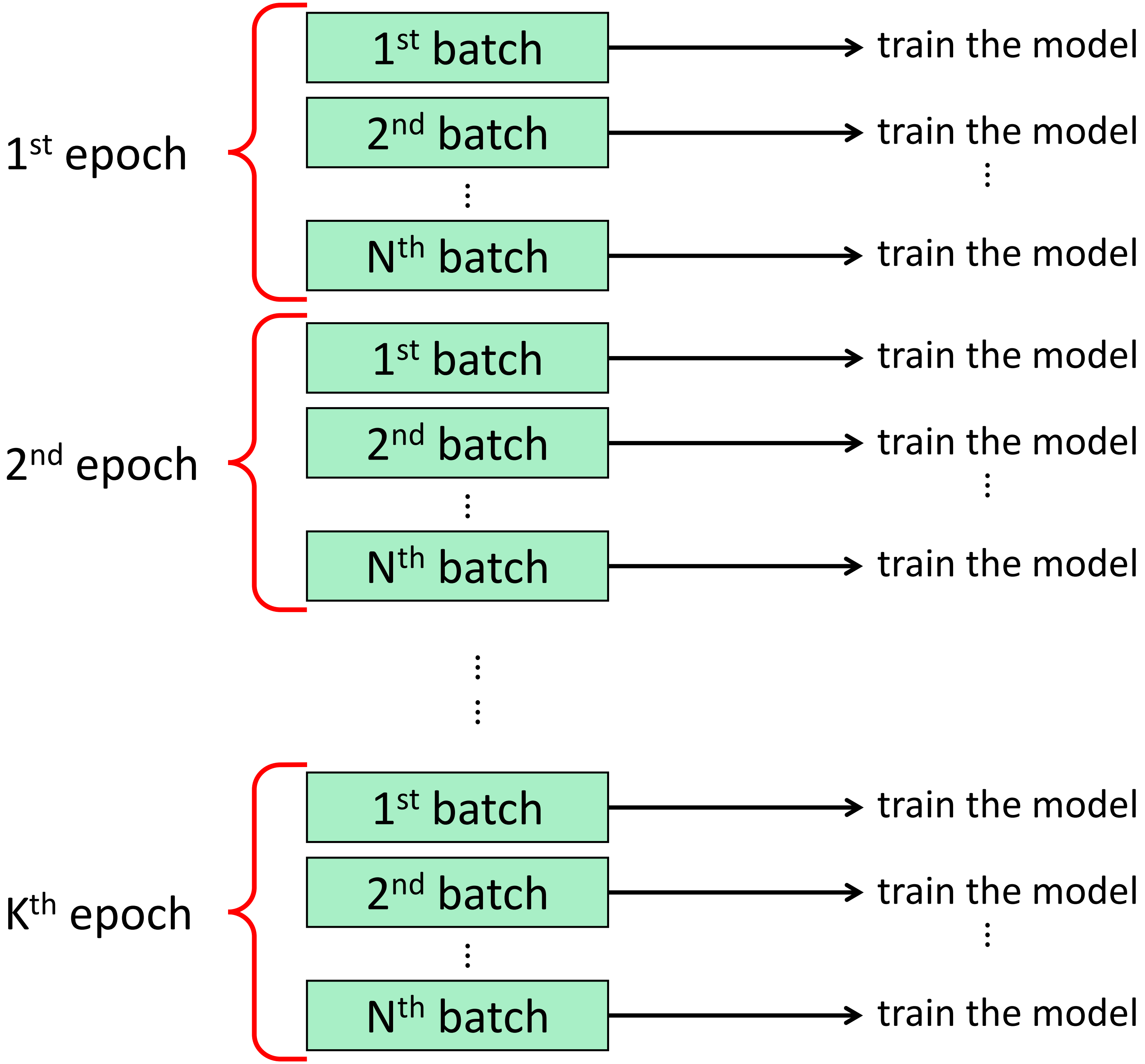
（3） 随机选取集合中$\{1,2,\cdots n\}$的k个元素组成集合Γ

（4） $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \gamma \frac{(-1)}{|\Gamma|} \sum_{i \in \Gamma} (y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b) \boldsymbol{x}_i$

（5） $b \leftarrow b - \gamma \frac{(-1)}{|\Gamma|} \sum_{i \in \Gamma} (y^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}_i - b)$

（6）返回$\boldsymbol{\omega}, b$.

训练过程

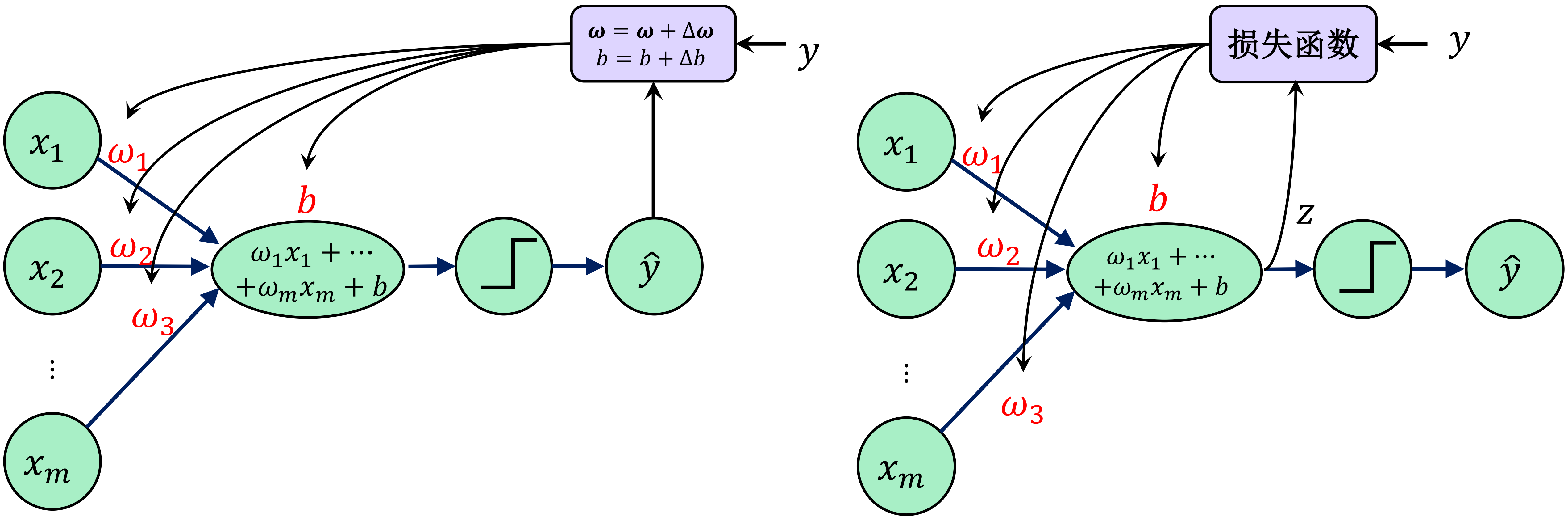


1st epoch
1st batch
train the model
2nd batch
train the model
Nth batch
train the model
2nd epoch
1st batch
train the model
2nd batch
train the model
Nth batch
train the model
Kth epoch
1st batch
train the model
2nd batch
train the model
Nth batch
train the model

训练过程

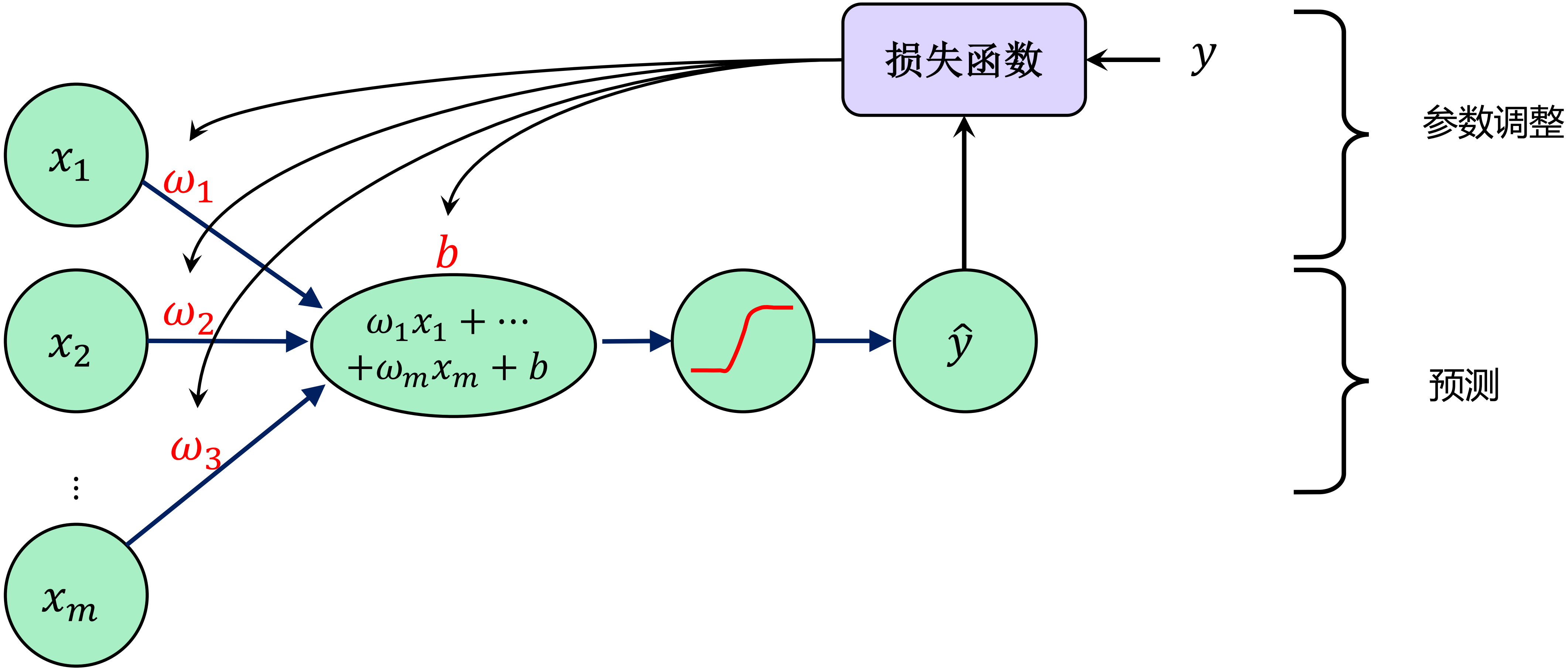
```
num_epochs = K
num_batches = N
for epoch in range(num_epoches):
        shuffle training data
        for batch in range(num_batches):
                X, y = Xtrain[batch, :], ytrain[batch]
                y_hat = model(X)
                loss_value = loss(y, y_hat)
                gradient descent to update parameters of the model
```

逻辑回归



- 感知机使用门限函数输出计算损失函数，由于门限函数不可微，无法执行梯度下降
- Adaline算法使用门限函数输入计算损失函数，损失函数可微，但损失函数与线性回归相同。

逻辑回归

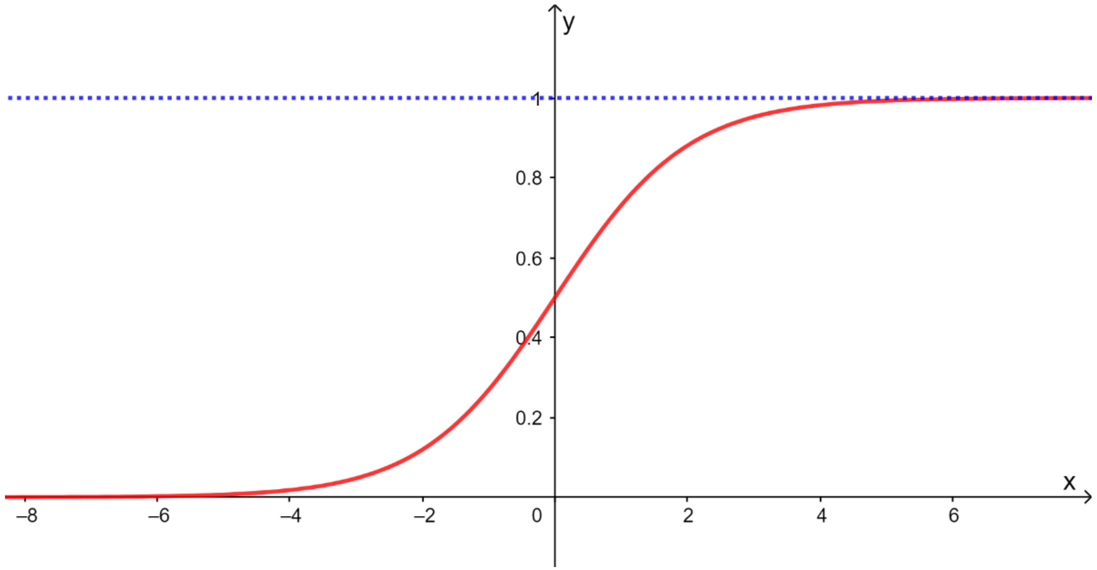


- 逻辑回归仍然使用门限函数输出计算损失函数
- 逻辑回归使用一个平滑可微函数代替门限函数
- 逻辑回归使用一种新的损失函数——交叉熵损失函数

逻辑回归

- 激活函数：逻辑函数

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



- $0 < \sigma(x) < 1$
- $\sigma(-\infty) = 0$, $\sigma(+\infty) = 1$, $\sigma(0) = 0.5$
- 逻辑函数适合表示概率

逻辑回归

假设：标签为二项式分布

二项式分布：n 个相互独立实验成功次数服从二项式分布。

$$X \sim \text{Bin}(n, p)$$

其中n 为实验次数， p 为一次实验成功的概率。

伯努利分布式二项式分布的一个特例，当只有一次实验时，二项式分布为伯努利分布。

例子：令 Y 为投一次硬币正面朝上的次数。 硬币正面朝上概率为 p。有

$$p(Y = 1) = p$$

$$p(Y = 0) = 1 - p$$

逻辑回归

- 标签 Y 为伯努利分布的随机变量，概率为

$$p(Y=1|\boldsymbol{x};\boldsymbol{\omega})=\sigma(\omega_1 x_1+\omega_2 x_2+\cdots+\omega_m x_m+b)=\sigma(\textstyle\sum_{i=1}^{m}\omega_i x_i+b)$$

$$\boldsymbol{x} \leftarrow [x_1, x_2, \cdots x_m, 1]^T$$
$$\boldsymbol{\omega} \leftarrow [\omega_1, \omega_2, \cdots \omega_m, b]^T$$

$$\Rightarrow \sum_{i=1}^{m}\omega_i x_i+b=\sum_{i=1}^{m}\omega_i x_i+\omega_{m+1}x_{m+1}=\sum_{i=1}^{m+1}\omega_i x_i=\boldsymbol{\omega}^T\boldsymbol{x}$$

- 所以有

$$p(Y=1|\boldsymbol{x};\boldsymbol{\omega})=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$
$$p(Y=0|\boldsymbol{x};\boldsymbol{\omega})=1-p(y=1|\boldsymbol{x},\boldsymbol{\omega})=1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

Logistic Regression（逻辑回归）

利用逻辑函数表示给定参数和样本标签的条件概率（似然概率）

$$p(Y=1|\boldsymbol{x};\boldsymbol{\omega})=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=0|\boldsymbol{x};\boldsymbol{\omega})=1-p(y=0|\boldsymbol{x},\boldsymbol{\omega})=1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=y|\boldsymbol{x};\boldsymbol{\omega})=p(Y=1|\boldsymbol{x};\boldsymbol{\omega})^y p(Y=0|\boldsymbol{x};\boldsymbol{\omega})^{1-y}=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})^y(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}))^{1-y}$$

$$p(Y_1=y_1,Y_2=y_2,\cdots,Y_n=y_n|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n;\boldsymbol{\omega})=\prod_{i=1}^{n}p(Y_i=y_i|\boldsymbol{x}_i;\boldsymbol{\omega})$$

所有训练样本独立同分布

$$=\prod_{i=1}^{n}\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)^{y_i}(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i))^{1-y_i}$$

Logistic Regression（逻辑回归）

利用逻辑函数表示给定参数和样本标签的条件概率（似然概率）

$$p(Y=1|\boldsymbol{x};\boldsymbol{\omega})=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=0|\boldsymbol{x};\boldsymbol{\omega})=1-p(y=0|\boldsymbol{x};\boldsymbol{\omega})=1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=y|\boldsymbol{x};\boldsymbol{\omega})=p(Y=1|\boldsymbol{x};\boldsymbol{\omega})^y p(Y=0|\boldsymbol{x};\boldsymbol{\omega})^{1-y}=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})^y(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}))^{1-y}$$

$$p(Y_1=y_1,Y_2=y_2,\cdots,Y_n=y_n|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n;\boldsymbol{\omega})=\prod_{i=1}^{n}p(Y_i=y_i|\boldsymbol{x}_i;\boldsymbol{\omega})$$

$$=\prod_{i=1}^{n}\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)^{y_i}(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i))^{1-y_i}$$

选择一个 ω 使这个条件概率大还是小?

Logistic Regression（逻辑回归）

利用逻辑函数表示给定参数和样本标签的条件概率（似然概率）

$$p(Y=1|\boldsymbol{x};\boldsymbol{\omega})=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=0|\boldsymbol{x};\boldsymbol{\omega})=1-p(y=0|\boldsymbol{x};\boldsymbol{\omega})=1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=y|\boldsymbol{x};\boldsymbol{\omega})=p(Y=1|\boldsymbol{x};\boldsymbol{\omega})^{y}p(Y=0|\boldsymbol{x};\boldsymbol{\omega})^{1-y}=\sigma(\boldsymbol{\omega}^T\boldsymbol{x})^{y}(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}))^{1-y}$$

$$p(Y_1=y_1,Y_2=y_2,\cdots,Y_n=y_n|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n;\boldsymbol{\omega})=\prod_{i=1}^{n}p(Y_i=y_i|\boldsymbol{x}_i;\boldsymbol{\omega})$$

$$=\prod_{i=1}^{n}\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)^{y_i}(1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i))^{1-y_i}$$

① 大概率事件必然发生。
② 小概率事件基本不发生。

这个条件概率的值应该大还是小?

Logistic Regression（逻辑回归）

条件概率的负log变换

$$NLL(\boldsymbol{\omega}) = -\log\big(P(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\omega})\big) = -\sum_{i=1}^{n} y_i \log[\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)] + (1-y_i)\log[1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]$$

$$= -\sum_{i=1}^{n} y_i \log\left(\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}_i}}\right) + (1-y_i)\log\left(1-\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}_i}}\right)$$

$$= \sum_{i=1}^{n} -y_i \log\left(\frac{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}}{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}+1}\right) - (1-y_i)\log\left(\frac{1}{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}+1}\right)$$

$$= \sum_{i=1}^{n} -y_i\boldsymbol{\omega}^T\boldsymbol{x}_i + \log\left(1+e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}\right)$$

这个也被称作经验损失函数$L(\omega)$

寻找一个 ω 使经验损失函数最小

Logistic Regression（逻辑回归）

- 经验损失函数 $\sum_{i=1}^{n} -y_i\boldsymbol{\omega}^T\boldsymbol{x}_i + \log\left(1+e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}\right)$ 对应损失函数为交叉熵损失函数
- k分类问题交叉熵损失函数定义如下

$$-\sum_{j=0}^{k-1} y_j \log\left(\hat{y}_j\right)$$

- 对于逻辑回归有 $\hat{y}_1 = p(Y=1|\boldsymbol{x},\boldsymbol{\omega}) = \sigma(\boldsymbol{\omega}^T\boldsymbol{x})$， $\hat{y}_0 = p(Y=0|\boldsymbol{x},\boldsymbol{\omega}) = 1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$

$$\begin{aligned}
-\sum_{j=0}^{1} y_j \log\left(\hat{y}_j\right) &= -y_1\log(\hat{y}_1) - y_0\log(\hat{y}_0) \\
&= -y_1 \log[\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)] - (1-y_1)\log[1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)] \\
&= -y_1 \log\left(\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}}}\right) - (1-y_1)\log\left(1-\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}}}\right) \\
&= -y_1\boldsymbol{\omega}^T\boldsymbol{x} + \log\left(1+e^{\boldsymbol{\omega}^T\boldsymbol{x}}\right)
\end{aligned}$$

Logistic Regression（逻辑回归）

$$L(\boldsymbol{\omega}) = -y_1\boldsymbol{\omega}^T\boldsymbol{x} + \log\left(1 + e^{\boldsymbol{\omega}^T\boldsymbol{x}}\right)$$

- $\boldsymbol{\omega}$为参数，最小化损失函数

$$\min_{\boldsymbol{\omega}} \frac{1}{n}\sum_{i=1}^{n} -y_i\boldsymbol{\omega}^T\boldsymbol{x}_i + \log\left(1 + e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}\right)$$

- 损失函数的导数为

$$\frac{dL(\boldsymbol{\omega})}{d\boldsymbol{\omega}} = \frac{1}{n}\sum_{i=1}^{n} -y_i\boldsymbol{x}_i + \frac{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}}{1 + e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}}\boldsymbol{x}_i = \frac{1}{n}\sum_{i=1}^{n}[-y_i + \sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]\boldsymbol{x}_i$$

- 令损失函数导数为0，求解参数$\boldsymbol{\omega}$

$$\frac{1}{n}\sum_{i=1}^{n}[-y_i + \sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]\boldsymbol{x}_i = \boldsymbol{0}$$

Logistic Regression（逻辑回归）

小批量随机梯度下降(mini-batch stochastic gradient descent)

- 梯度下降利用所有训练样本计算梯度。如果训练样本大，计算量巨大。
- 随机梯度下降在每一次迭代中，随机抽取一部分训练样本用于估计梯度。

$$\frac{1}{n}\sum_{i=1}^{n}[-y_i+\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]\boldsymbol{x}_i \approx \frac{1}{|\Gamma|}\sum_{i\in\Gamma}[-y_i+\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]\boldsymbol{x}_i$$

$$\boldsymbol{\omega}_{i+1} \leftarrow \ \boldsymbol{\omega}_i - \gamma\frac{1}{|\Gamma|}\sum_{i\in\Gamma}[-y_i+\sigma(\boldsymbol{\omega}^T\boldsymbol{x}_i)]\boldsymbol{x}_i$$

, 其中 Γ为随机抽选样本索引的集合。 $|\Gamma|$ 为集合元素个数。

Logistic Regression（逻辑回归）

逻辑回归模型训练伪代码

输入： 训练数据$\boldsymbol{X}, \boldsymbol{y}$，整数$k$，步长$\gamma$。
输出： 逻辑回归模型参数.

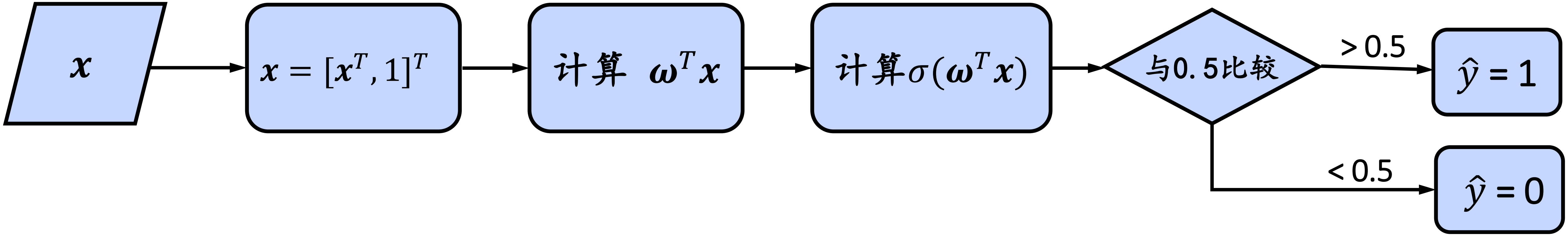
(1) **初始化：** 随机初始化 $\boldsymbol{\omega}_{\mathbf{0}}, i = 1$.
(2) $\boldsymbol{X} = [\boldsymbol{X}, \mathbf{1}_n]$
(3) 循环以下部分直至结束：
(4) 从 $\{1,2,\cdots,n\}$ 中无放回地随机抽取k个数构成集合Γ.
(5) $\boldsymbol{\omega}_i \leftarrow \boldsymbol{\omega}_{i-1} - \gamma \frac{1}{|\Gamma|} \sum_{j\in\Gamma} \left[-y_j + \sigma\left(\boldsymbol{\omega}^T \boldsymbol{x}_j\right)\right] \boldsymbol{x}_j$
(6) $i \leftarrow i + 1$
(7) 返回 $\boldsymbol{\omega}_i$

Logistic Regression（逻辑回归）

逻辑回归模型测试伪代码

输入： 线性模型参数$\boldsymbol{\omega}$，测试样本$\boldsymbol{x}$。
输出： 测试样本预测标签。

（1） $\boldsymbol{x} = [\boldsymbol{x}^T, 1]^T$。
（2） 计算$\boldsymbol{\omega}^T\boldsymbol{x}$。
（3） 计算$\sigma(\boldsymbol{\omega}^T\boldsymbol{x})$。
（4） 如果$\sigma(\boldsymbol{\omega}^T\boldsymbol{x}) > 0.5$，$\hat{y} = 1$;如果$\sigma(\boldsymbol{\omega}^T\boldsymbol{x}) < 0.5$，$\hat{y} = 0$。
（5） 返回$\hat{y}$。

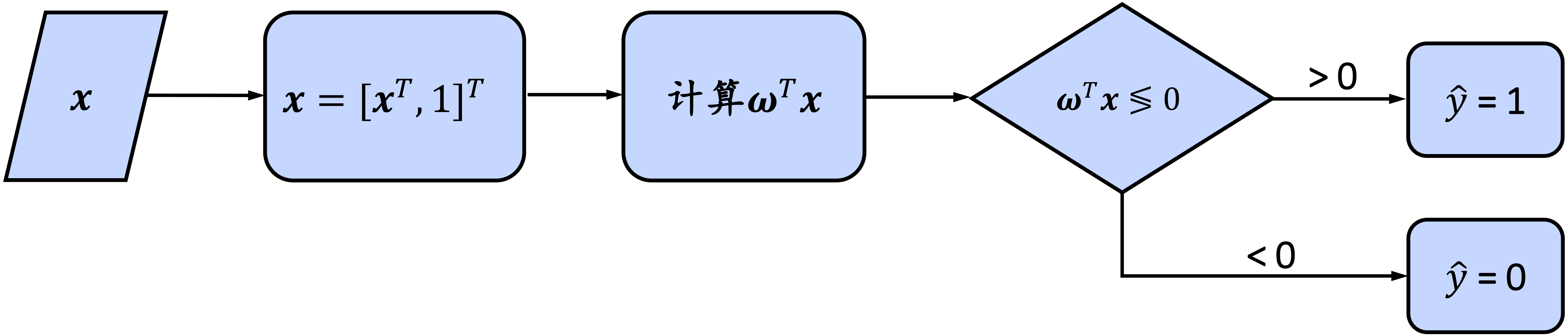


Logistic Regression（逻辑回归）

逻辑回归模型测试伪代码

输入: 线性模型参数$\boldsymbol{\omega}$，测试样本$\boldsymbol{x}$。
输出: 测试样本预测标签。

(1) $\boldsymbol{x} = [\boldsymbol{x}^T, 1]^T$。
(2) 计算$\boldsymbol{\omega}^T\boldsymbol{x}$。
(3) 如果$\boldsymbol{\omega}^T\boldsymbol{x} > 0$，$\hat{y} = 1$；如果$\boldsymbol{\omega}^T\boldsymbol{x} < 0$，$\hat{y} = 0$。
(4) 返回$\hat{y}$。



Logistic Regression（逻辑回归）

$$p(Y=1|\boldsymbol{x},\boldsymbol{\omega}) = \sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$p(Y=0|\boldsymbol{x},\boldsymbol{\omega}) = 1 - \sigma(\boldsymbol{\omega}^T\boldsymbol{x})$$

$$\frac{p(Y=1|\boldsymbol{x},\boldsymbol{\omega})}{p(Y=0|\boldsymbol{x},\boldsymbol{\omega})} = \frac{\sigma(\boldsymbol{\omega}^T\boldsymbol{x})}{1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x})} = \frac{\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}}}}{1-\frac{1}{1+e^{-\boldsymbol{\omega}^T\boldsymbol{x}}}} = e^{\boldsymbol{\omega}^T\boldsymbol{x}}$$

$$\log\left(\frac{p(Y=1|\boldsymbol{x},\boldsymbol{\omega})}{p(Y=0|\boldsymbol{x},\boldsymbol{\omega})}\right) = \boldsymbol{\omega}^T\boldsymbol{x}$$

Log odd (logit function) 与样本特征具有线性关系，所以算法名字为Logistic Regression

Logistic Regression（逻辑回归）

- 计算梯度会遇到的计算问题: **浮点数溢出**

如果 $\boldsymbol{\omega}^T\boldsymbol{x}_i$ 太大，$e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}$ 会更大，导致计算机无法表示这么大的数，会发生浮点数溢出。

$$\frac{dL(\boldsymbol{\omega})}{d\boldsymbol{\omega}} = \sum_{i=1}^{n} -y_i\boldsymbol{x}_i + \frac{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}}{1 + e^{\boldsymbol{\omega}^T\boldsymbol{x}_i}}\boldsymbol{x}_i$$

- **解决方案**

$$\frac{dL(\boldsymbol{\omega})}{d\boldsymbol{\omega}} = \sum_{i=1}^{n} -y_i\boldsymbol{x}_i + \frac{e^{\boldsymbol{\omega}^T\boldsymbol{x}_i-\beta}}{e^{-\beta} + e^{\boldsymbol{\omega}^T\boldsymbol{x}_i-\beta}}\boldsymbol{x}_i$$

其中 $\beta = \max_i \boldsymbol{\omega}^T\boldsymbol{x}_i$.

分子分母中的指数函数的指数项都很小

Multinomial Logistic Regression（多类别逻辑回归）

类别数目为 k。 k 大于 2。

- 训练数据为 $(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$ ，其中 $\boldsymbol{x}_i \in \boldsymbol{R}^{m\times 1}$, $\boldsymbol{y}_i \in \boldsymbol{R}^{k\times 1}$。
- 标签$\boldsymbol{y}_i$为一个独热向量（one-hot vector）。比如，类别标签为0， $\boldsymbol{y}_i$表达式为

$$\boldsymbol{y}_i \in \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- x_{ij}代表第i个输入向量的第j个特征。
- y_{ij}代表第i个标签向量的第j个元素。

Multinomial Logistic Regression（多类别逻辑回归）

- **Softmax函数**为一个多输入多输出的函数.

$$\boldsymbol{\sigma}\left(\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}\right) = \begin{bmatrix} \frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}} \\ \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}} \\ \vdots \\ \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \end{bmatrix}$$

- $\boldsymbol{z} = [z_1 \quad z_2 \quad \cdots \quad z_k]^T$，令 $\sigma_i(\boldsymbol{z})$ 为softmax函数第 i个输出，有

$$\sigma_i(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Softmax函数输出值非负，而且和为1，所以softmax函数适合做概率。

Multinomial Logistic Regression（多类别逻辑回归）

方法：利用softmax函数表示条件概率

- 首先，对输入向量添加一个人工特征

$$\boldsymbol{x} \leftarrow \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}$$

- 然后，对添加人工特征的输入向量进行k次线性变换

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}, \quad \boldsymbol{W} = \begin{bmatrix} \boldsymbol{\omega}_1^T \\ \boldsymbol{\omega}_2^T \\ \vdots \\ \boldsymbol{\omega}_k^T \end{bmatrix} \qquad \boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\omega}_1^T \\ \boldsymbol{\omega}_2^T \\ \vdots \\ \boldsymbol{\omega}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\omega}_1^T \boldsymbol{x} \\ \boldsymbol{\omega}_2^T \boldsymbol{x} \\ \vdots \\ \boldsymbol{\omega}_k^T \boldsymbol{x} \end{bmatrix}$$

$$z_1 = \boldsymbol{\omega}_1^T \boldsymbol{x}, \quad z_2 = \boldsymbol{\omega}_2^T \boldsymbol{x}, \quad \cdots, \quad z_k = \boldsymbol{\omega}_k^T \boldsymbol{x}$$

- 最后，一个样本标签的条件概率（即预测标签）可以利用softmax函数表示为

$$p(Y_1 = y_1 | \boldsymbol{z}) = \sigma_1(\boldsymbol{z}),\ p(Y_2 = y_2 | \boldsymbol{z}) = \sigma_2(\boldsymbol{z}), \cdots, p(Y_k = y_k | \boldsymbol{z}) = \sigma_k(\boldsymbol{z})$$

，其中$Y = [Y_1 \quad Y_2 \quad \cdots \quad Y_k]^T$为表示标签向量的随机变量。

Multinomial Logistic Regression（多类别逻辑回归）

方法：利用softmax函数表示条件概率

$$
\begin{cases}
p(Y_1 = y_1|\boldsymbol{z}) = \sigma_1(\boldsymbol{z}) \\
p(Y_2 = y_2|\boldsymbol{z}) = \sigma_2(\boldsymbol{z}) \\
\quad\quad\quad \cdots \\
p(Y_k = y_k|\boldsymbol{z}) = \sigma_k(\boldsymbol{z})
\end{cases}
\Rightarrow
$$

$$
p(Y = \boldsymbol{y}|\boldsymbol{x};\boldsymbol{W}) = \sigma_1(\boldsymbol{z})^{y_1}\sigma_2(\boldsymbol{z})^{y_2}\cdots\sigma_k(\boldsymbol{z})^{y_k}
$$

$$
= \prod_{i=1}^{k}\sigma_i(\boldsymbol{z})^{y_i}
$$

Multinomial Logistic Regression（多类别逻辑回归）

- 所有训练样本的联合条件概率为

$$p(\boldsymbol{Y}_1=\boldsymbol{y}_1,\boldsymbol{Y}_2=\boldsymbol{y}_2,\cdots,\boldsymbol{Y}_n=\boldsymbol{y}_n,|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n;\boldsymbol{W})=\prod_{i=1}^{n}p(\boldsymbol{Y}_i=\boldsymbol{y}_i|\boldsymbol{x}_i;\boldsymbol{W})=\prod_{i=1}^{n}\prod_{j=1}^{k}\sigma_j(\boldsymbol{z}_i)^{y_{ij}}$$

- 联合条件概率的负对数变换为

$$NNL=-\log\big(p(\boldsymbol{Y}_1=\boldsymbol{y}_1,\boldsymbol{Y}_2=\boldsymbol{y}_2,\cdots,\boldsymbol{Y}_n=\boldsymbol{y}_n,|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n,\boldsymbol{\omega}_1,\boldsymbol{\omega}_2,\cdots,\boldsymbol{\omega}_k)\big)$$

$$=-\sum_{i=1}^{n}\sum_{j=1}^{k}y_{ij}\log\left(\sigma_j(\boldsymbol{z}_i)\right)$$

$$=-\sum_{i=1}^{n}\sum_{j=1}^{k}y_{ij}\log\left(\frac{e^{z_{ij}}}{\sum_{t=1}^{k}e^{z_{it}}}\right)$$

$$=-\sum_{i=1}^{n}\sum_{j=1}^{k}y_{ij}\left[z_{ij}-\log\left(\sum_{t=1}^{k}e^{z_{it}}\right)\right]$$

$$=-\sum_{i=1}^{n}\left[\sum_{j=1}^{k}y_{ij}z_{ij}-\left(\sum_{j=1}^{k}y_{ij}\right)\log\left(\sum_{t=1}^{k}e^{z_{it}}\right)\right]$$

$$=\sum_{i=1}^{n}\left[\log\left(\sum_{t=1}^{k}e^{z_{it}}\right)-\sum_{j=1}^{k}y_{ij}z_{ij}\right]$$

$$z_{it}=\boldsymbol{\omega}_t^T\boldsymbol{x}_i$$

经验损失函数

Multinomial Logistic Regression（多类别逻辑回归）

$$L(\boldsymbol{W}) = \mathrm{NNL} = \sum_{i=1}^{n}\left[\log\left(\sum_{t=1}^{k} e^{z_{it}}\right) - \sum_{j=1}^{k} y_{ij}\, z_{ij}\right]$$

经验损失函数偏导数为

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_s} = \sum_{i=1}^{n}\left[\frac{e^{z_{is}}}{\sum_{t=1}^{k} e^{z_{it}}}\frac{\partial z_{is}}{\partial \boldsymbol{\omega}_s} - y_{is}\frac{\partial z_{is}}{\partial \boldsymbol{\omega}_s}\right] = \sum_{i=1}^{n}[\sigma_s(\boldsymbol{z}_i) - y_{is}]\frac{\partial z_{is}}{\partial \boldsymbol{\omega}_s} = \sum_{i=1}^{n}[\sigma_s(\boldsymbol{z}_i) - y_{is}]\boldsymbol{x}_i$$

，因为 $z_{is} = \boldsymbol{\omega}_s^T \boldsymbol{x}_i$

Multinomial Logistic Regression（多类别逻辑回归）

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_s} = \sum_{i=1}^{n} [\sigma_s(\boldsymbol{z}_i) - y_{is}]\boldsymbol{x}_i \qquad \boldsymbol{W} = \begin{bmatrix} \boldsymbol{\omega}_1^T \\ \boldsymbol{\omega}_2^T \\ \vdots \\ \boldsymbol{\omega}_k^T \end{bmatrix}$$

梯度为

$$\frac{dL(\boldsymbol{W})}{d\boldsymbol{W}} = \begin{bmatrix} \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_1^T} \\ \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_2^T} \\ \vdots \\ \vdots \\ \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_k^T} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} [\sigma_1(\boldsymbol{z}_i) - y_{i1}]\, \boldsymbol{x}_i^T \\ \sum_{i=1}^{n} [\sigma_2(\boldsymbol{z}_i) - y_{i2}]\, \boldsymbol{x}_i^T \\ \vdots \\ \sum_{i=1}^{n} [\sigma_k(\boldsymbol{z}_i) - y_{ik}]\, \boldsymbol{x}_i^T \end{bmatrix} = \sum_{i=1}^{n} (\boldsymbol{\sigma}(\boldsymbol{z}_i) - \boldsymbol{y}_i)\boldsymbol{x}_i^T = \sum_{i=1}^{n} (\hat{\boldsymbol{y}}_i - \boldsymbol{y})\boldsymbol{x}_i^T$$

Multinomial Logistic Regression（多类别逻辑回归）

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_s} = \sum_{i=1}^{n} [\sigma_s(\boldsymbol{z}_i) - y_{is}]\boldsymbol{x}_i \qquad \boldsymbol{W} = \begin{bmatrix} \boldsymbol{\omega}_1^T \\ \boldsymbol{\omega}_2^T \\ \vdots \\ \boldsymbol{\omega}_k^T \end{bmatrix}$$

梯度为

$$\frac{dL(\boldsymbol{W})}{d\boldsymbol{W}} = \begin{bmatrix} \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_1^T} \\ \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_2^T} \\ \vdots \\ \vdots \\ \frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{\omega}_k^T} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} [\sigma_1(\boldsymbol{z}_i) - y_{i1}]\, \boldsymbol{x}_i^T \\ \sum_{i=1}^{n} [\sigma_2(\boldsymbol{z}_i) - y_{i2}]\, \boldsymbol{x}_i^T \\ \vdots \\ \sum_{i=1}^{n} [\sigma_k(\boldsymbol{z}_i) - y_{ik}]\, \boldsymbol{x}_i^T \end{bmatrix} = \sum_{i=1}^{n} (\boldsymbol{\sigma}(\boldsymbol{z}_i) - \boldsymbol{y}_i)\boldsymbol{x}_i^T = \sum_{i=1}^{n} (\widehat{\boldsymbol{y}}_i - \boldsymbol{y})\boldsymbol{x}_i^T$$

$$(\widehat{\boldsymbol{Y}} - \boldsymbol{Y})^{\boldsymbol{T}}\boldsymbol{X}$$

Multinomial Logistic Regression（多类别逻辑回归）

多类别逻辑回归模型训练伪代码.

输入： $\boldsymbol{X}$, $\boldsymbol{Y}$, k, γ.
输出： 模型参数.

(1) **初始化：** 随机初始化 $\boldsymbol{W}^{(0)}$, $i = 1$.
(2) $\boldsymbol{X} = [\boldsymbol{X}, \boldsymbol{1}_n]$.
(3) 循环以下代码直至停止：
(4) 从 $\{1,2,\cdots,n\}$ 中无放回地随机抽取k个数构成集合Γ.
(5) $\boldsymbol{z}_j = \boldsymbol{W}\boldsymbol{x}_j, \hat{\boldsymbol{y}}_j = \boldsymbol{\sigma}(\boldsymbol{z}_j)$, $j \in \Gamma$.
(6) $\boldsymbol{G} = \sum_{j\in\Gamma}(\hat{\boldsymbol{y}}_j - \boldsymbol{y}_j)\boldsymbol{x}_i^T$.
(7) $\boldsymbol{W}^{(i)} \leftarrow \boldsymbol{W}^{(i-1)} - \gamma \frac{n}{|\Gamma|}\boldsymbol{G}$.
(8) $i \leftarrow i + 1$.
(9) 返回$\boldsymbol{W}^{(i)}$.

Multinomial Logistic Regression（多类别逻辑回归）

多类别逻辑回归模型测试伪代码

输入： 模型参数$\boldsymbol{W}$，测试样本$\boldsymbol{x}$.
输出： 测试样本预测标签.

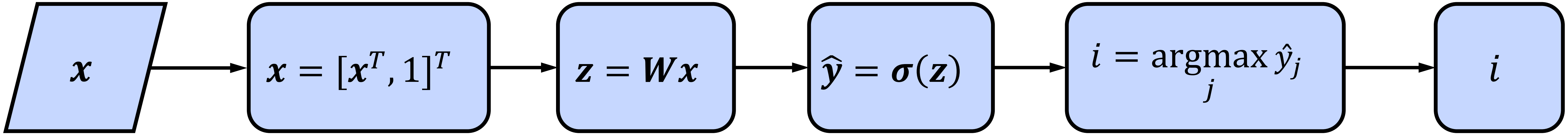
(1) $\boldsymbol{x} = [\boldsymbol{x}^T, 1]^T$.

(2) $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$.

(3) $\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{z})$

(4) $i = \underset{j}{\operatorname{argmax}}\, \hat{y}_j$.

(5) 返回i.



Multinomial Logistic Regression（多类别逻辑回归）

多类别逻辑回归模型测试伪代码

输入： 模型参数$\boldsymbol{W}$，测试样本$\boldsymbol{x}$.
输出： 测试样本预测标签.

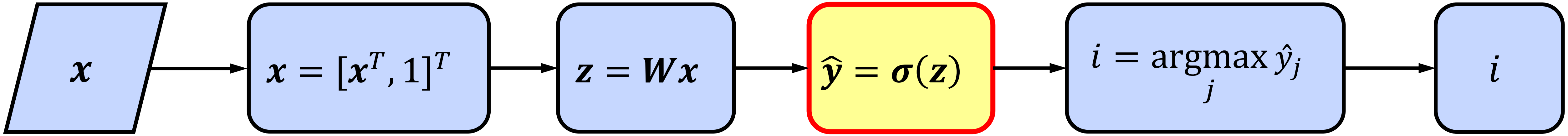
(1) $\boldsymbol{x} = [\boldsymbol{x}^T, 1]^T$.

(2) $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$.

(3) $\boxed{\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{z})}$

(4) $i = \underset{j}{\operatorname{argmax}}\, \hat{y}_j$.

(5) 返回i.



Multinomial Logistic Regression（多类别逻辑回归）

多类别逻辑回归模型测试伪代码

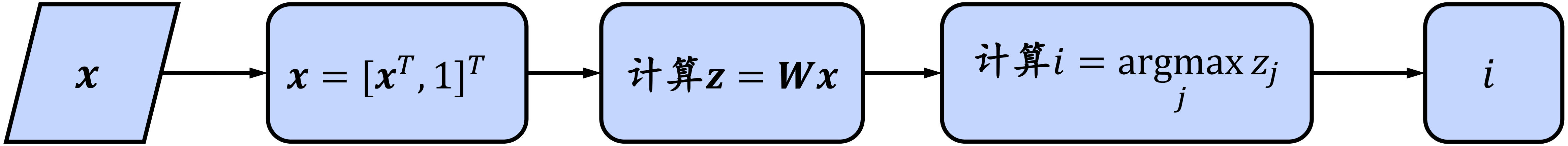
输入： 模型参数$\boldsymbol{W}$，测试样本$\boldsymbol{x}$.
输出： 测试样本预测标签.

（1） $\boldsymbol{x} = [\boldsymbol{x}^T, 1]^T$.

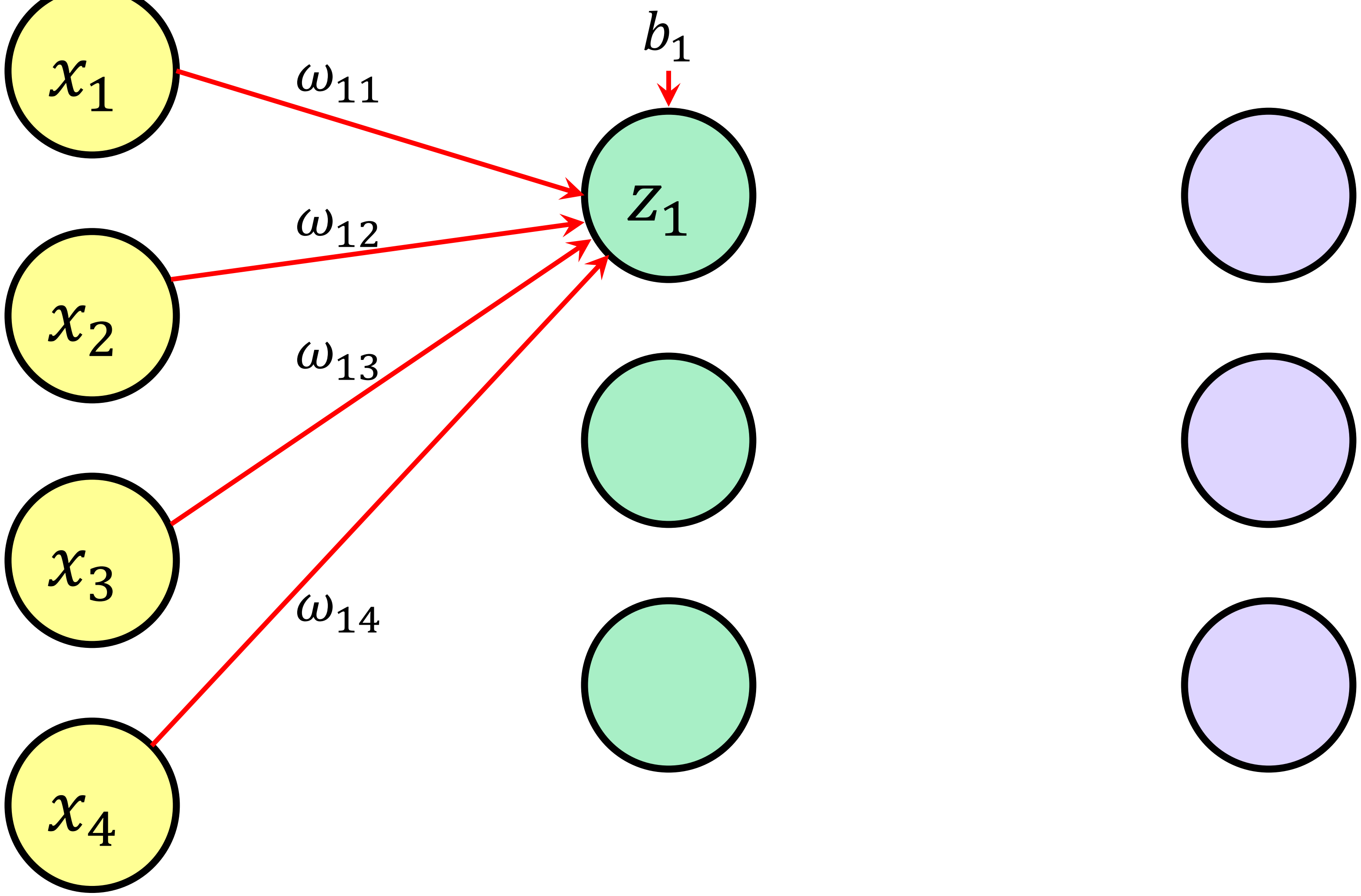
（2） $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$.

（3） $i = \operatorname*{argmax}_j z_j$.

（4） 返回i.

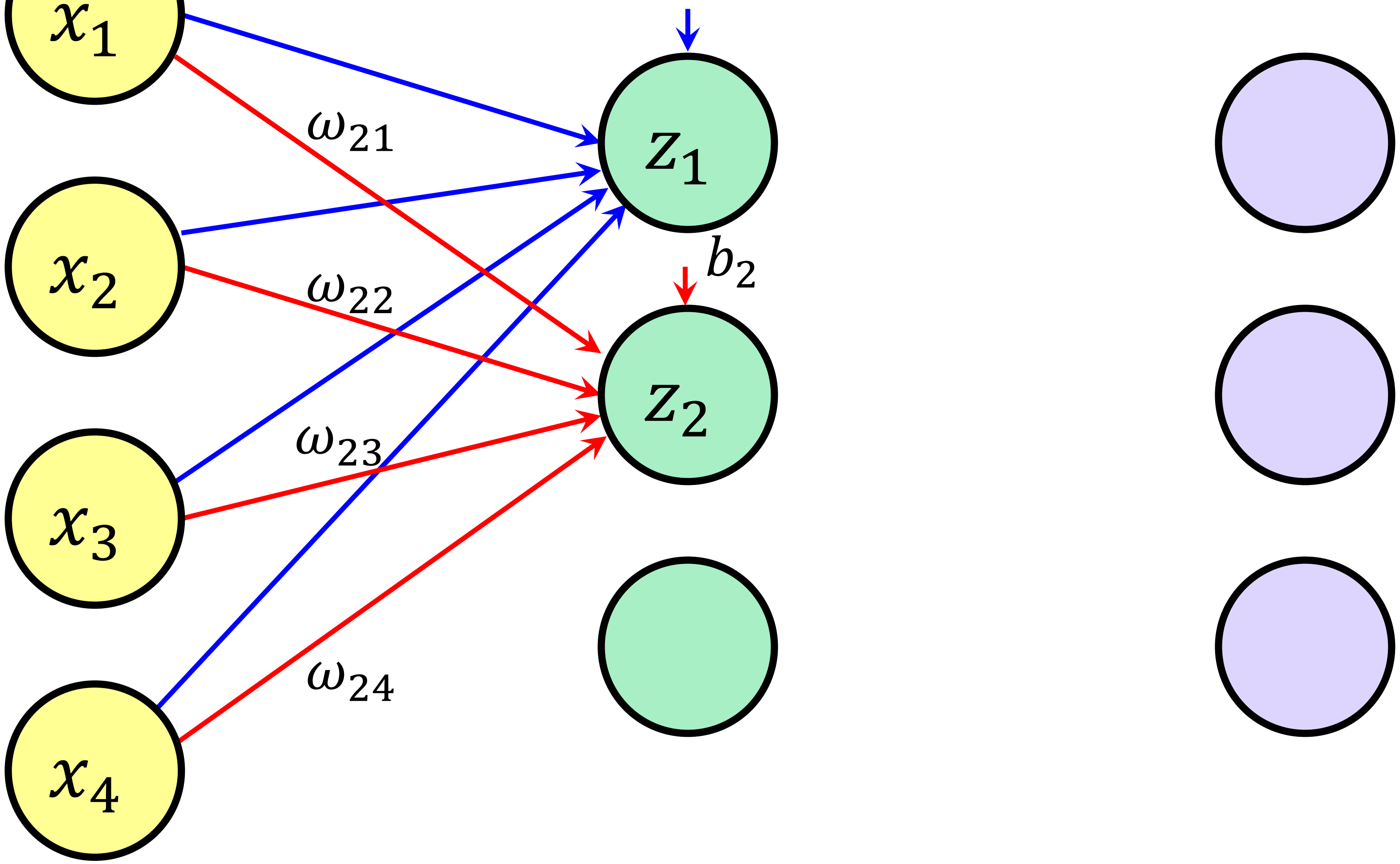


Multinomial Logistic Regression（多类别逻辑回归）



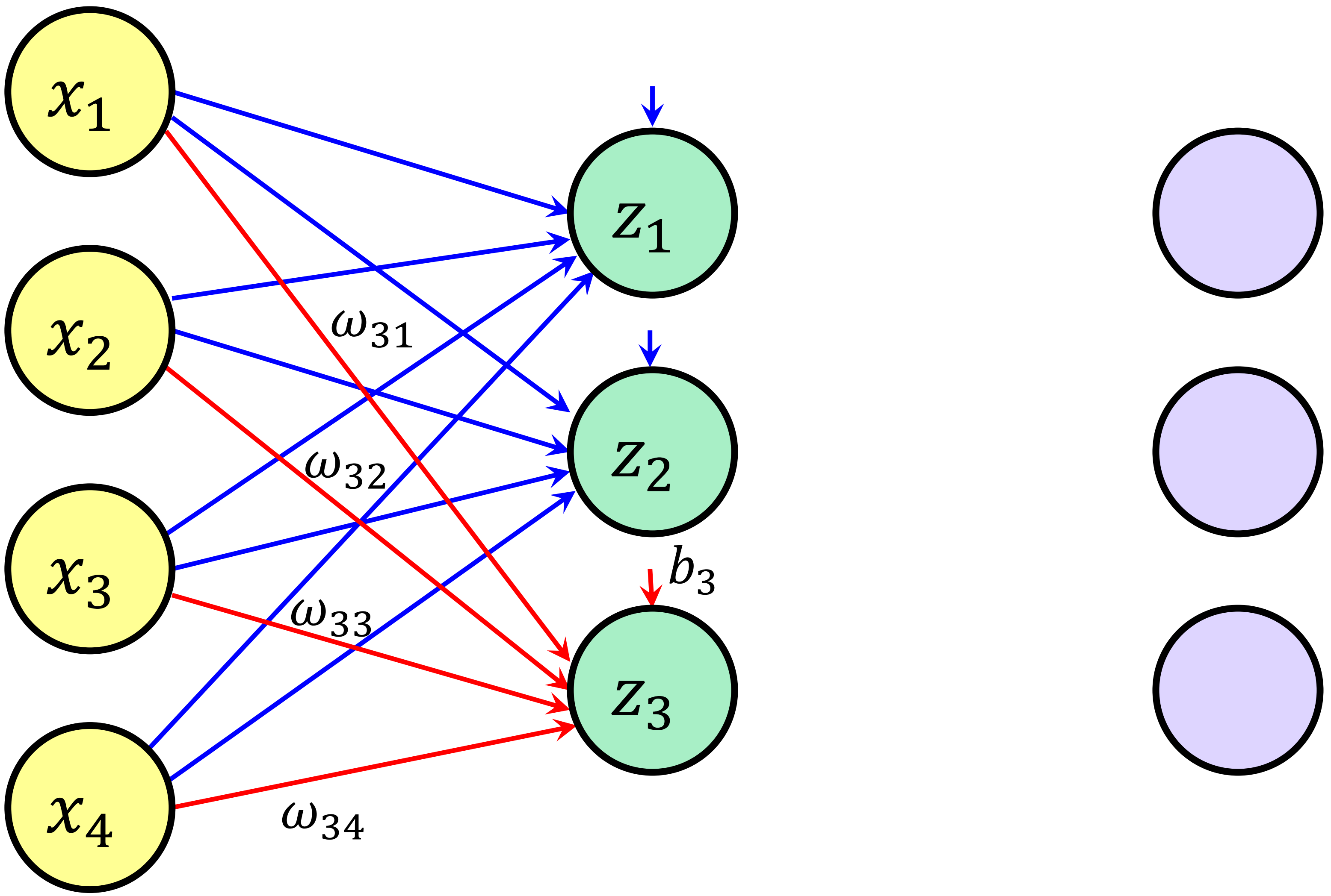
$$z_1 = \omega_{11}x_1 + \omega_{12}x_2 + \omega_{13}x_3 + b_1$$

Multinomial Logistic Regression（多类别逻辑回归）



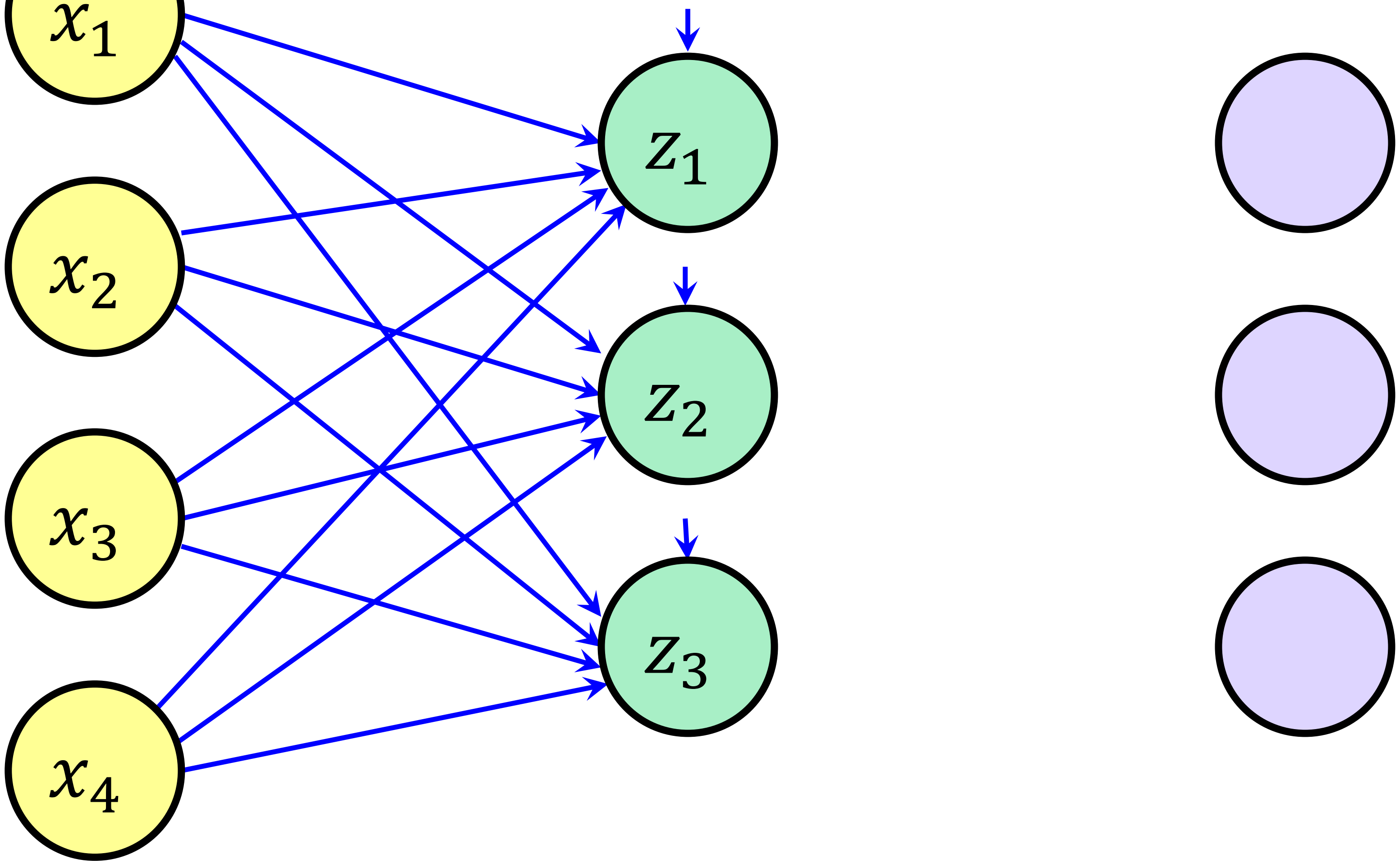
$$z_2 = \omega_{21}x_1 + \omega_{22}x_2 + \omega_{23}x_3 + b_2$$

Multinomial Logistic Regression（多类别逻辑回归）

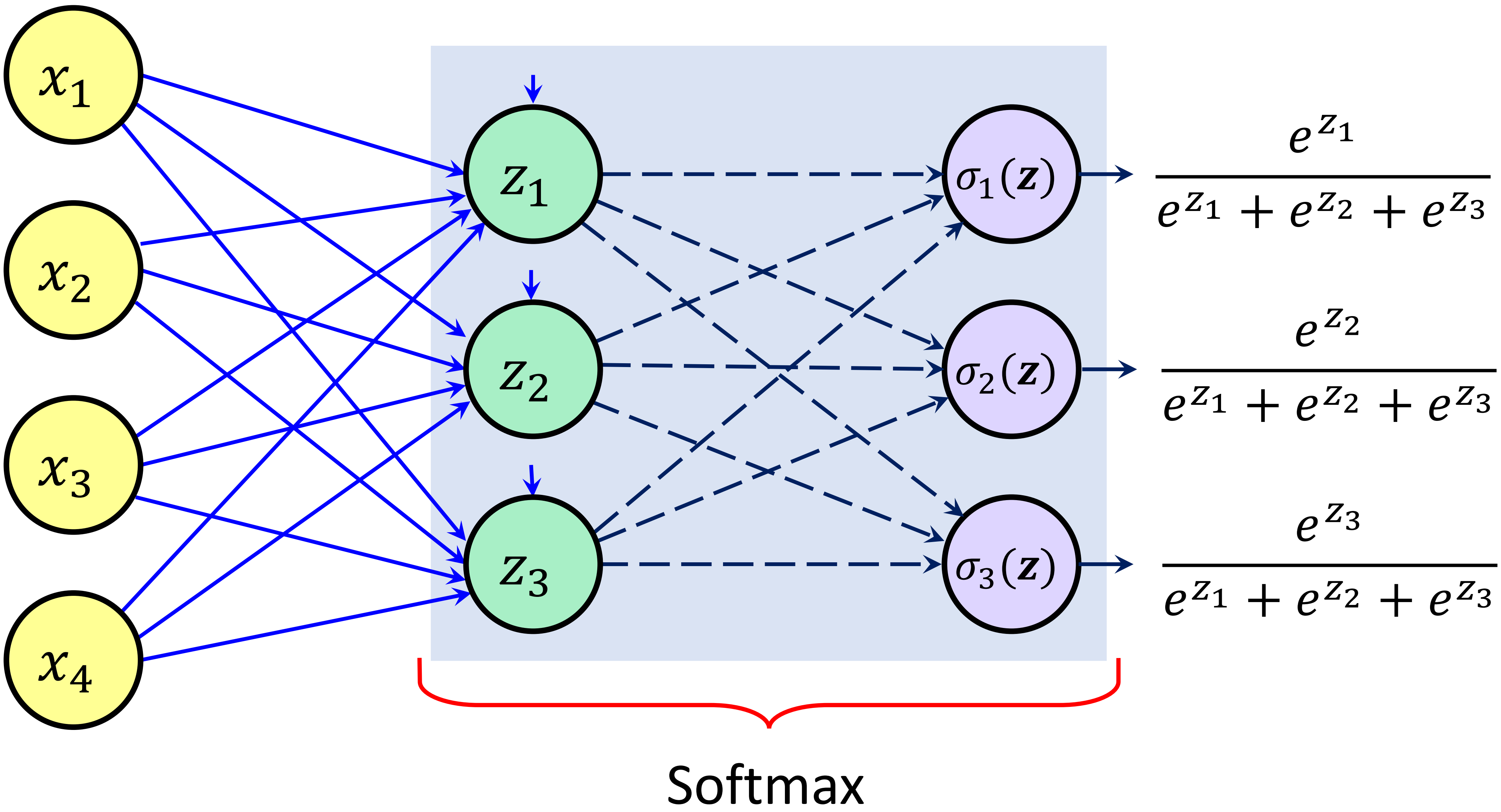


$$z_3 = \omega_{31}x_1 + \omega_{32}x_2 + \omega_{33}x_3 + b_3$$

Multinomial Logistic Regression（多类别逻辑回归）

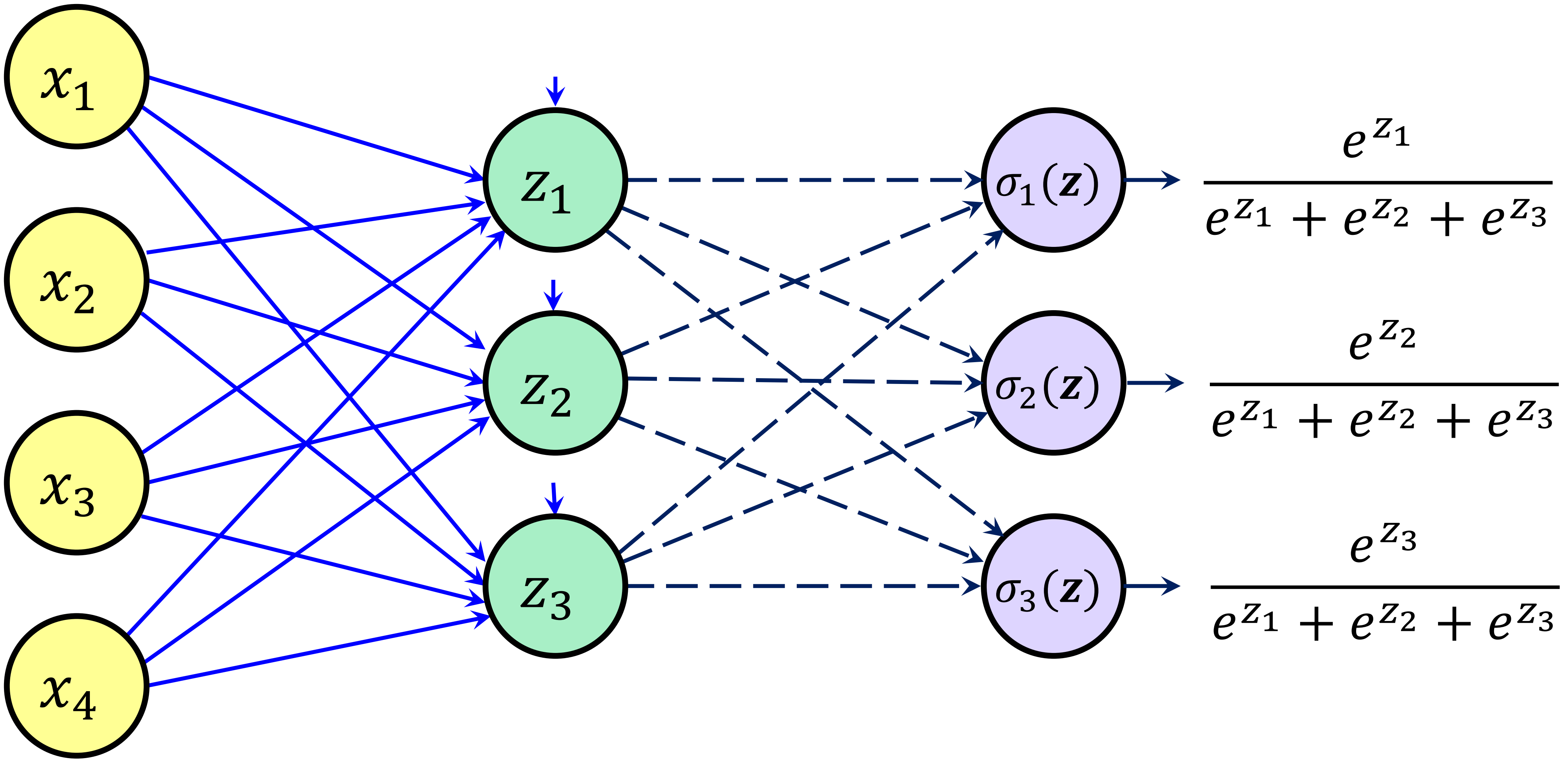


Multinomial Logistic Regression（多类别逻辑回归）

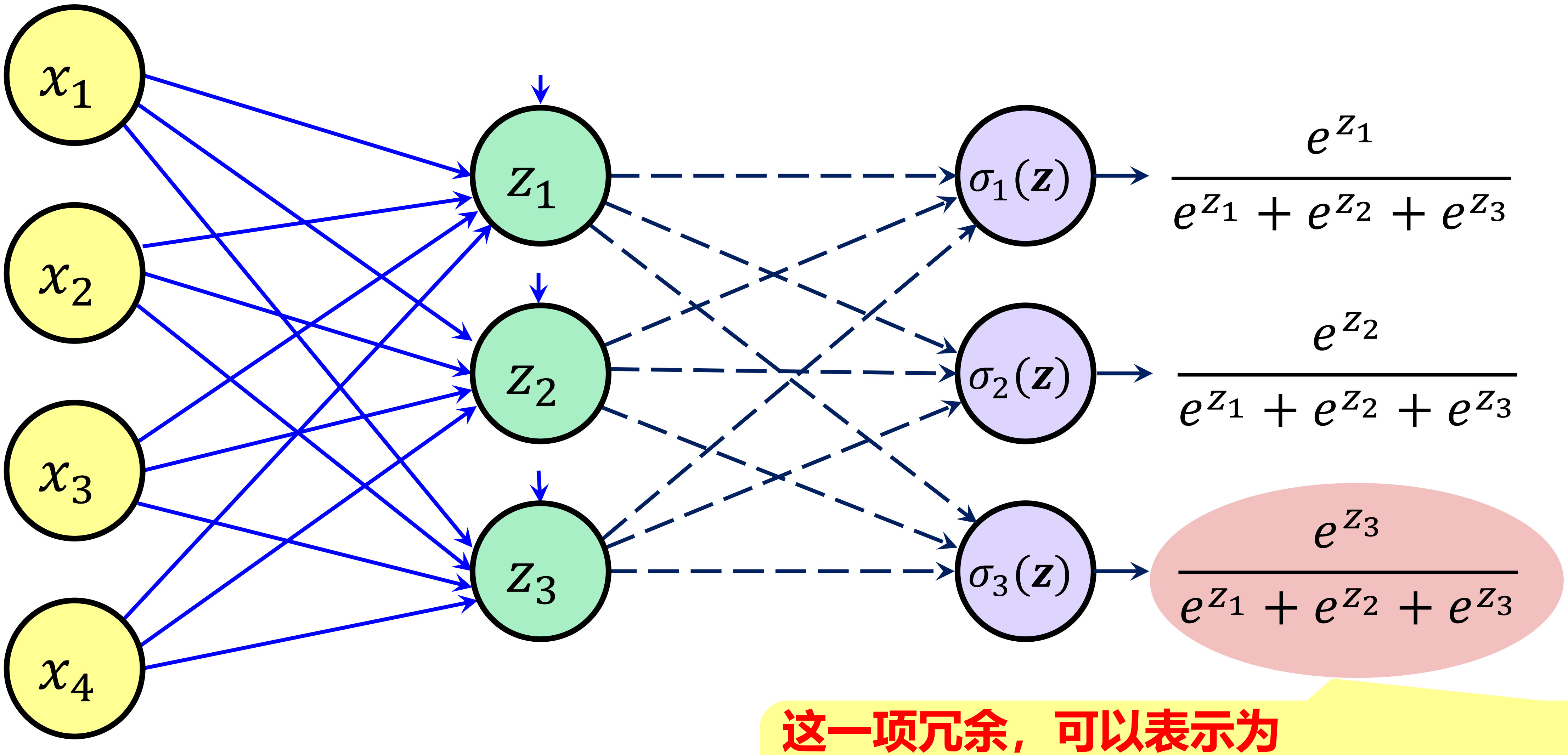


Multinomial Logistic Regression（多类别逻辑回归）

多类别逻辑回归一般用于神经网络最后一层。



Multinomial Logistic Regression（多类别逻辑回归）



这一项冗余，可以表示为

$$1-\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}-\frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

Multinomial Logistic Regression（多类别逻辑回归）