机器学习

苏州大学计算机科学与技术学院
自然语言处理实验室
主讲：周夏冰
邮箱：zhouxiabing@suda.edu.cn

01

BP神经网络

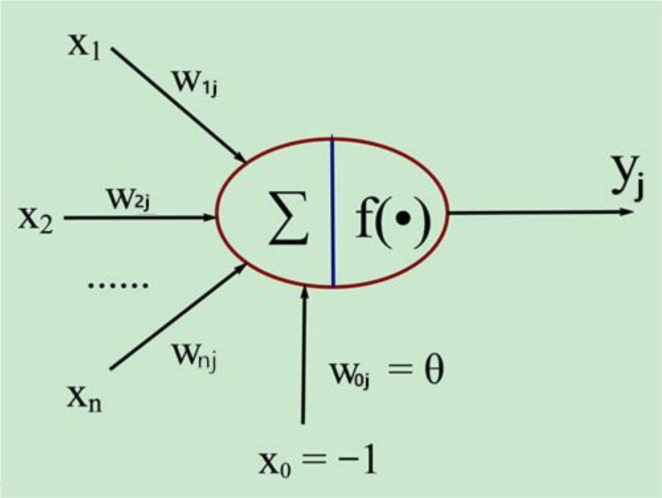
神经网络

◆模拟人脑神经系统的结构和功能，运用大量简单处理单元经广泛连接而组成的人工网络系统。

◆神经网络方法是一种**隐式**的知识表示方法

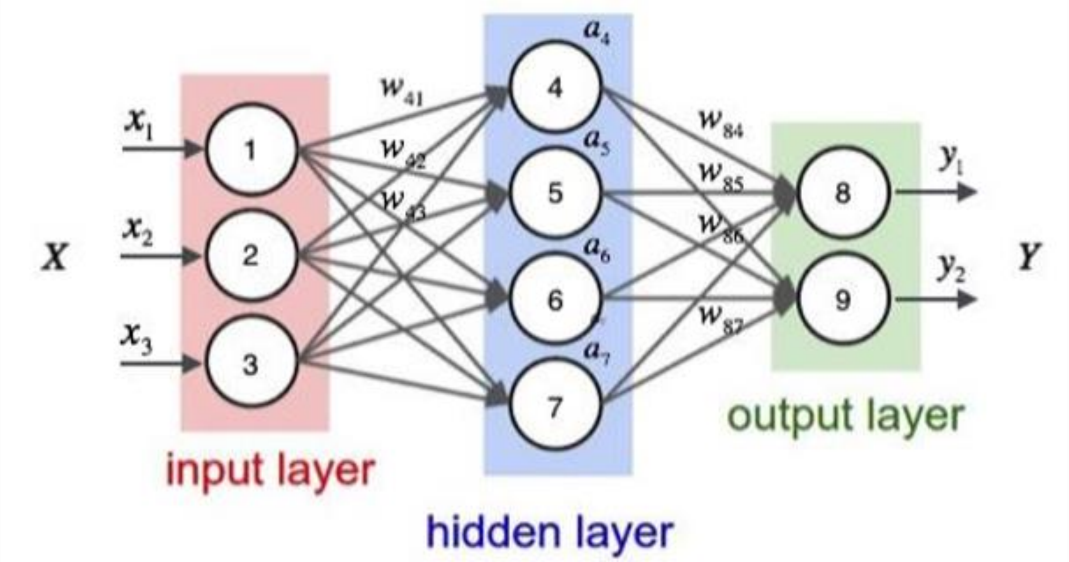
◆最早的神经网络的思想起源于1943年的MCP人工神经元模型

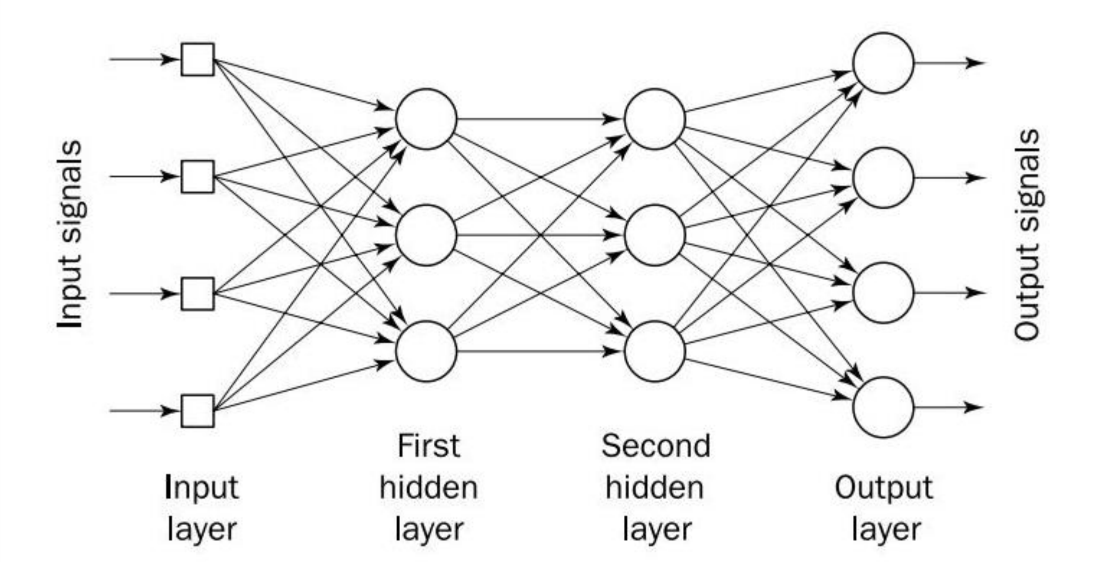
M-P模型



神经网络

- 第一次打破非线性诅咒的当属现代DL大牛Hinton，其在1986年发明了适用于多层感知器（MLP）的BP算法，并采用Sigmoid进行非线性映射，有效解决了非线性分类和学习的问题





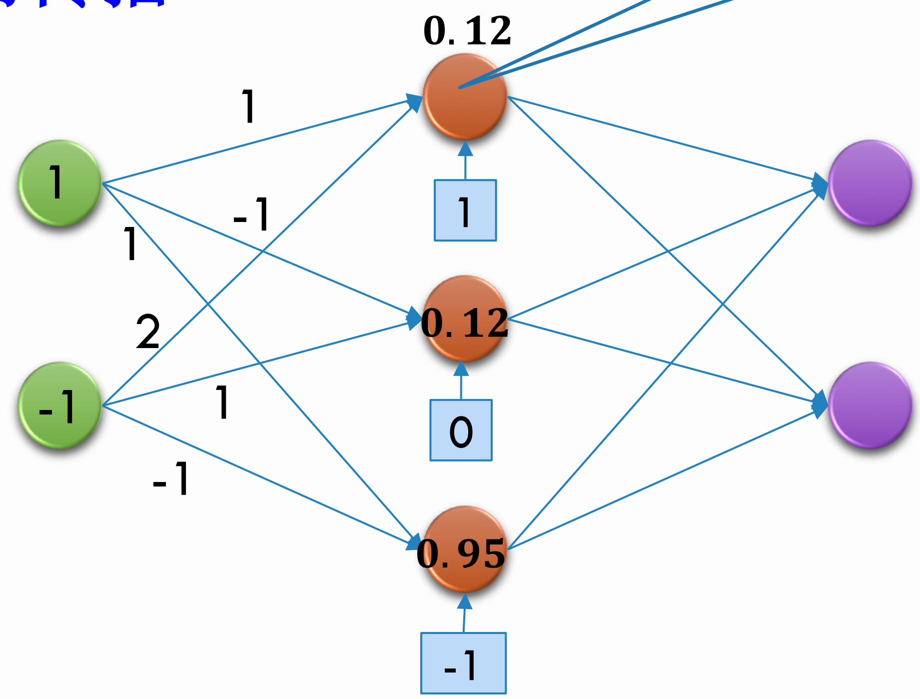
BP神经网络

- 连接主义的神经网络有着多种多样的网络结构以及学习方法，虽然早期模型强调模型的生物可解释性（Biological Plausibility），但后期更关注于对某种特定认知能力的模拟，比如物体识别、语言理解等. 尤其在引入误差反向传播来改进其学习能力之后，神经网络也越来越多地应用在各种机器学习任务上.
- BP（back-propagation）神经网络就是多层前向网络，利用反向误差传播来进行学习

BP学习算法

- 前向传播

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



$$1\times1+(-1)\times(-1)+1=3$$

$$\frac{1}{1+e^{-3}}=0.95$$

$$1\times1+(-1)\times2-1=-2$$

$$1\times(-1)+(-1)\times1-0=-2$$

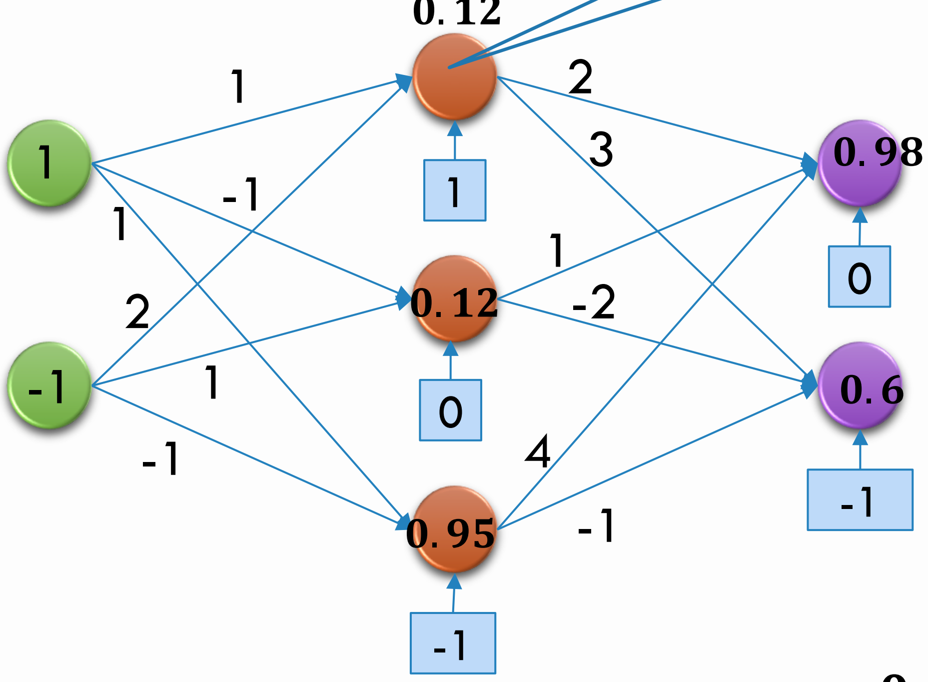
$$\frac{1}{1+e^{2}}=0.12$$

$$\frac{1}{1+e^{2}}=0.12$$

BP学习算法

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

- **前向传播**



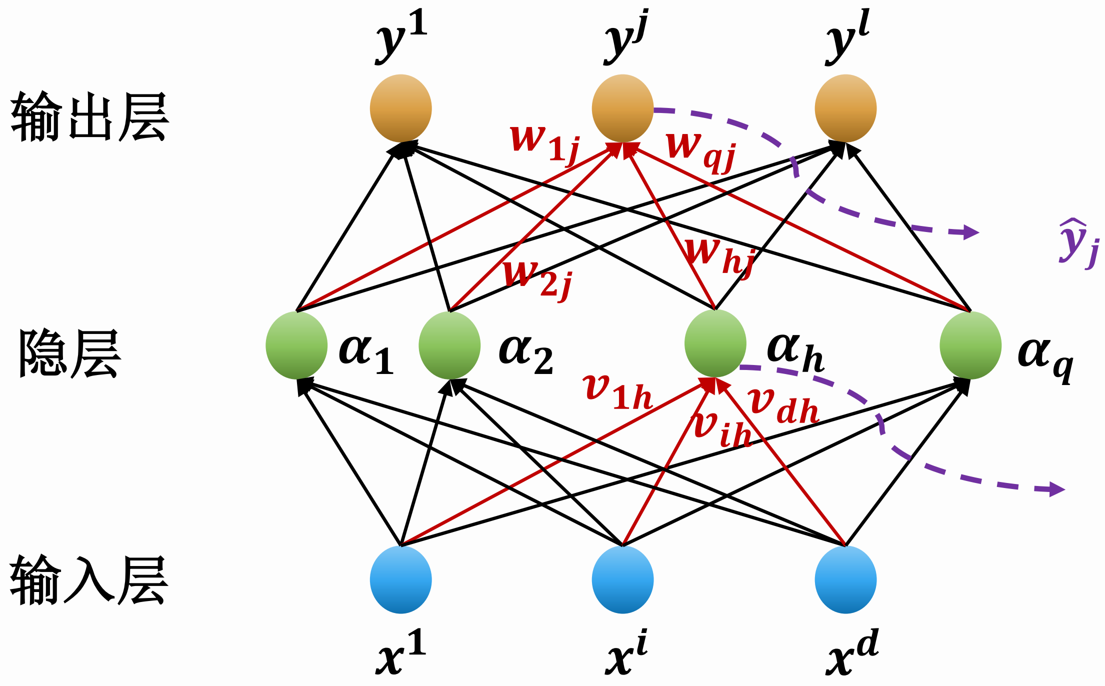
$$0.12\times 2+0.12\times 1 +0.95\times 4-0=4.16$$

$$\frac{1}{1+e^{-4.16}}=0.98$$

$$0.12\times 3+0.12\times(-2) +0.73\times(-1)+1=0.39$$

$$\frac{1}{1+e^{-0.39}}=0.6$$

BP学习算法



$$\hat{y}_j = f\left(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2\right)$$

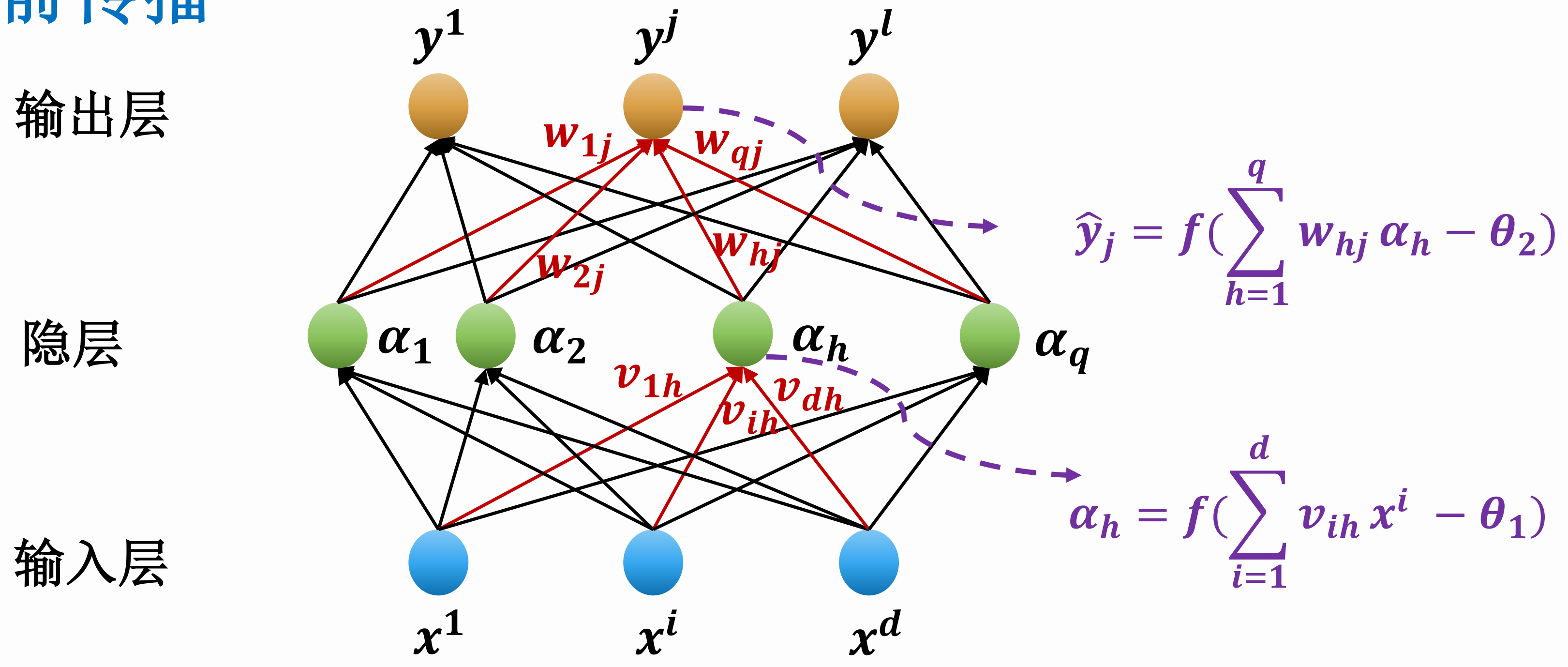
$$\alpha_h = f\left(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1\right)$$

BP学习算法

- 误差反向传播
  - $\widehat{w_i} \leftarrow w_i - \eta \frac{\Delta E}{\Delta w_i}$

BP学习算法

■向前传播



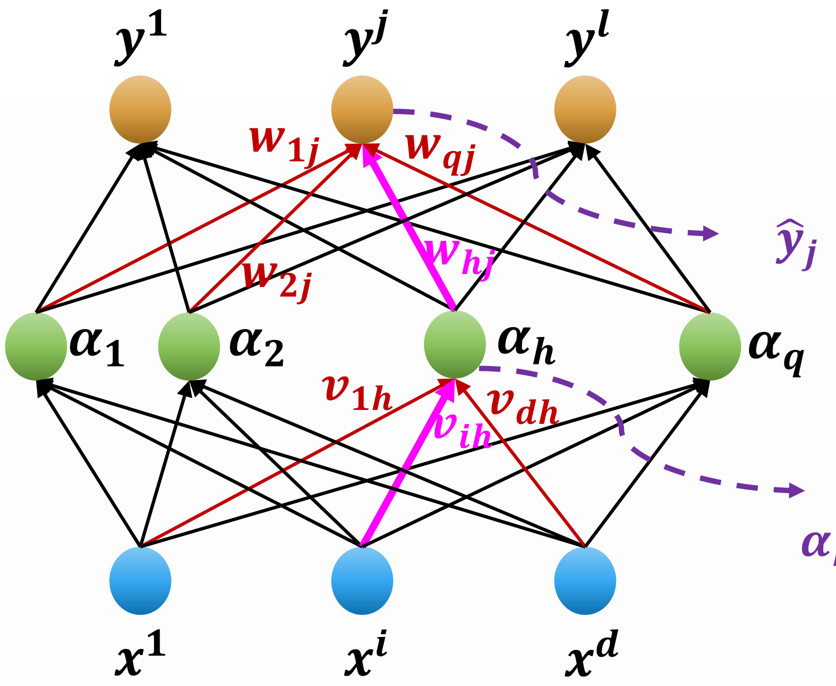
■ 对于训练样例：(x_k, y_k)

➢ 均方误差：$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2$

BP学习算法

■反向传播

➢梯度下降



$$\hat{y}_j = f\left(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2\right)$$

$$\alpha_h = f\left(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1\right)$$

$$f = \frac{1}{1+e^{-z}}$$

$$f' = -\frac{1}{(1+e^{-z})^2}\cdot(-e^{-z})$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1+e^{-z}-1}{(1+e^{-z})^2}$$

$$= \frac{1}{1+e^{-z}}\cdot\frac{1+e^{-z}-1}{1+e^{-z}}$$

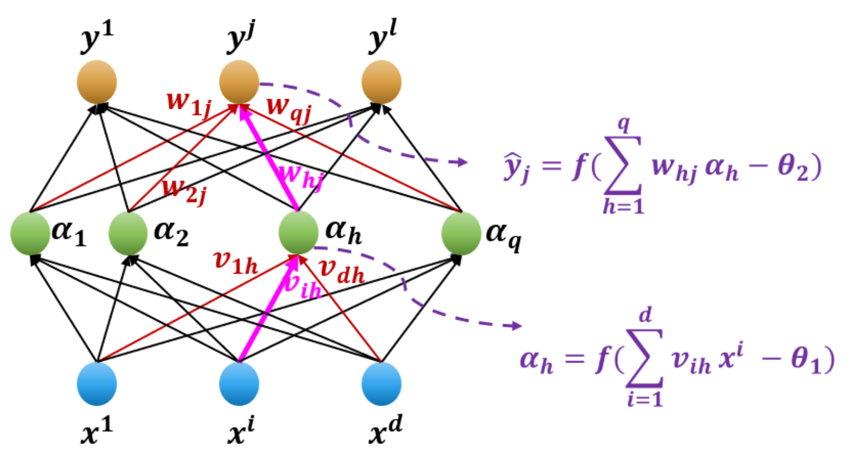
$$= f\cdot(1-f)$$

■输出层损失函数

➢$L = \frac{1}{2}(\hat{y}_k^j - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}\Big(f\big(\overbrace{\boxed{\textstyle\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2}}^{\beta_j}\big) - y_k^j\Big)^2$

➢$\frac{\partial L}{\partial w_{hj}} = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial w_{hj}} = (\hat{y}_k^j - y_k^j)\cdot\boxed{f(\beta_j)(1-f(\beta_j))}\cdot\alpha_h$

BP学习算法



$$\beta_j = \sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2$$

$$\gamma_h = \sum_{i=1}^{d} v_{ih}\,x^i - \theta_1$$

$$\frac{\partial L}{\partial w_{hj}} = \frac{\partial L}{\partial \hat{y}_k^j} \cdot \frac{\partial \hat{y}_k^j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} = \underbrace{\boxed{(\hat{y}_k^j - y_k^j) \cdot f(\beta_j)(1 - f(\beta_j))}}_{\delta_j^L} \cdot \alpha_h$$

$$\delta_j^L = \frac{\partial L}{\partial \hat{y}_k^j} \cdot \frac{\partial \hat{y}_k^j}{\partial \beta_j} = \frac{\partial L}{\partial \beta_j}$$

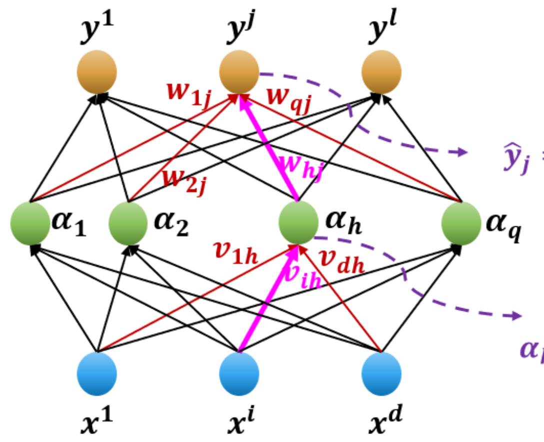
$$\mathrm{L} = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\beta_j) - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2) - y_k^j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\gamma_h)) - y_k^j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1) - \theta_2) - y_k^j)^2$$

$$\frac{\partial L}{\partial v_{ih}} = \sum_{j=1}^{l}(\hat{y}_k^j - y_k^j) \cdot f(\beta_j)(1 - f(\beta_j)) \cdot w_{hj} \cdot f(\gamma_h)(1 - f(\gamma_h)) \cdot x^i$$

BP学习算法



$$\hat{y}_j = f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2)$$

$$\alpha_h = f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1)$$

$$\beta_j = \sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2$$

$$\gamma_h = \sum_{i=1}^{d} v_{ih}\,x^i - \theta_1$$

$$\frac{\partial L}{\partial w_{hj}} = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial w_{hj}} = \underbrace{\boxed{(\hat{y}_k^j - y_k^j)\cdot f(\beta_j)(1-f(\beta_j))}}_{\delta_j^L}\cdot\alpha_h$$

$$\delta_j^L = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j} = \frac{\partial L}{\partial \beta_j}$$

$$\mathrm{L} = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\beta_j) - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2) - y_k^j)^2$$

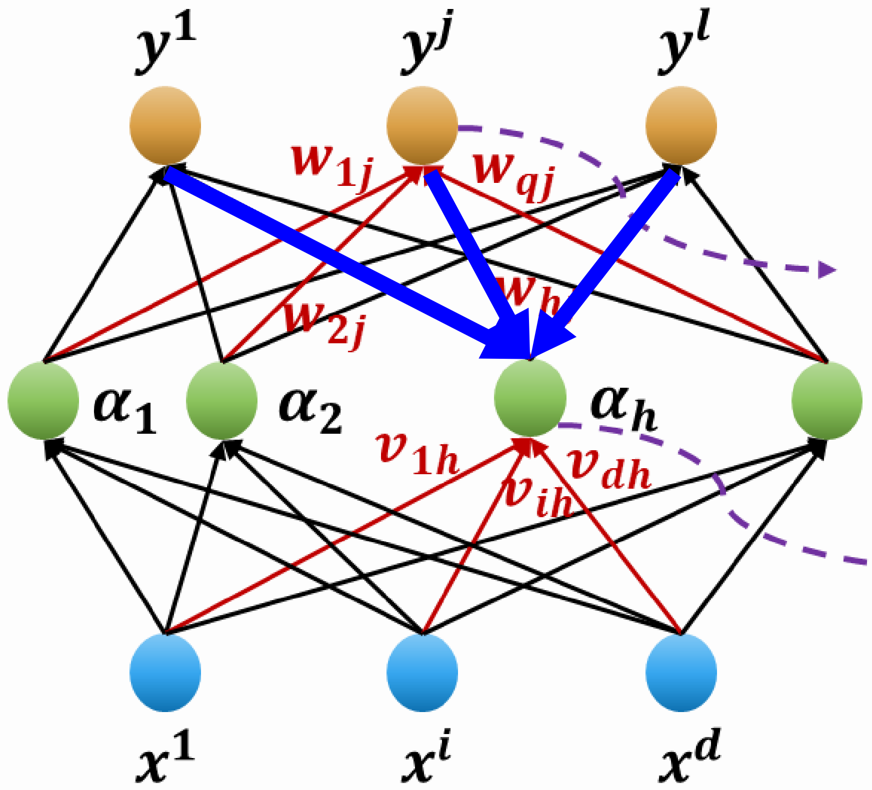
$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\gamma_h)) - y_k^j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1) - \theta_2) - y_k^j)^2$$

$$\frac{\partial L}{\partial v_{ih}} = \underbrace{\boxed{\sum_{j=1}^{l}\delta_j^L\cdot w_{hj}\cdot f(\gamma_h)(1-f(\gamma_h))}}_{\delta_h^{L-1}}\cdot x^i$$

$$\delta_h^{L-1} = \sum_{j=1}^{l}\delta_j^L\cdot w_{hj}\cdot\frac{\partial \alpha_h}{\partial \gamma_h}$$

BP学习算法

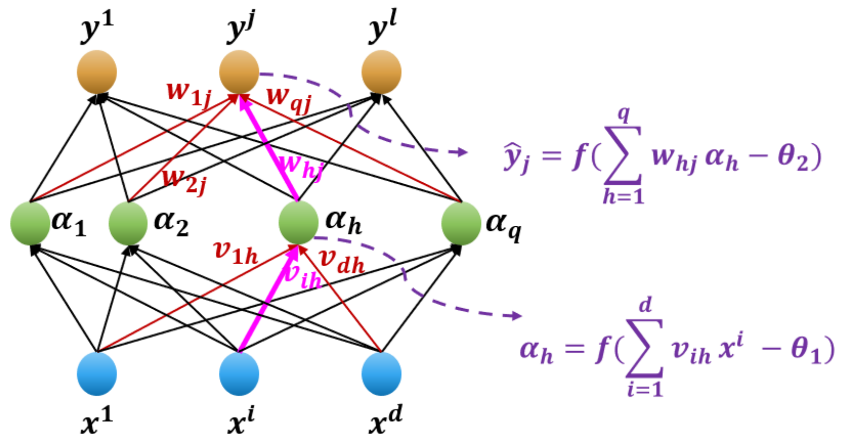


$$\hat{y}_j = f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2)$$

$$\alpha_h = f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1)$$

$$\delta_h^{L-1} = \sum_{j=1}^{l} \delta_j^L \cdot w_{hj} \cdot \frac{\partial \alpha_h}{\partial \gamma_h}$$

BP学习算法



$$\beta_j = \sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2$$

$$\gamma_h = \sum_{i=1}^{d} v_{ih}\,x^i - \theta_1$$

$$\frac{\partial L}{\partial w_{hj}} = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial w_{hj}} = \underbrace{\boxed{(\hat{y}_k^j - y_k^j)\cdot f(\beta_j)(1-f(\beta_j))}}_{\delta_j^L}\cdot\alpha_h$$

$$\delta_j^L = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j} = \frac{\partial L}{\partial \beta_j}$$

$$\mathrm{L} = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\beta_j) - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2) - y_k^j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\gamma_h)) - y_k^j)^2$$

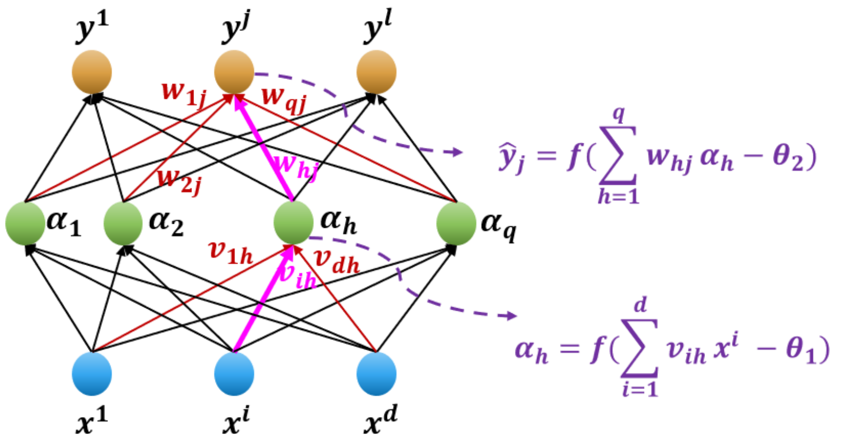
$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1) - \theta_2) - y_k^j)^2$$

$$\frac{\partial L}{\partial v_{ih}} = \underbrace{\boxed{\sum_{j=1}^{l}\delta_j^L\cdot w_{hj}\cdot f(\gamma_h)(1-f(\gamma_h))}}_{\delta_h^{L-1}}\cdot x^i$$

$$\delta_h^{L-1} = \sum_{j=1}^{l}\delta_j^L\cdot w_{hj}\cdot\frac{\partial \alpha_h}{\partial \gamma_h} = \sum_{j=1}^{l}\frac{\partial L}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial \alpha_h}\cdot\frac{\partial \alpha_h}{\partial \gamma_h} = \frac{\partial L}{\partial \gamma_h}$$

$$= \frac{\partial L}{\partial \alpha_h}\frac{\partial \alpha_h}{\partial \gamma_h}$$

BP学习算法



$$\beta_j = \sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2$$

$$\gamma_h = \sum_{i=1}^{d} v_{ih}\,x^i - \theta_1$$

前提 $L = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2$

$$\frac{\partial L}{\partial w_{hj}} = \frac{\partial L}{\partial \hat{y}_k^j}\cdot\frac{\partial \hat{y}_k^j}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial w_{hj}} = \underbrace{\boxed{(\hat{y}_k^j - y_k^j)\cdot f(\beta_j)(1-f(\beta_j))}}_{\delta_j^L}\cdot\alpha_h = \delta_j^L\cdot\alpha_h$$

$$\mathrm{L} = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_k^j - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\beta_j) - y_k^j)^2 = \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,\alpha_h - \theta_2) - y_k^j)^2$$

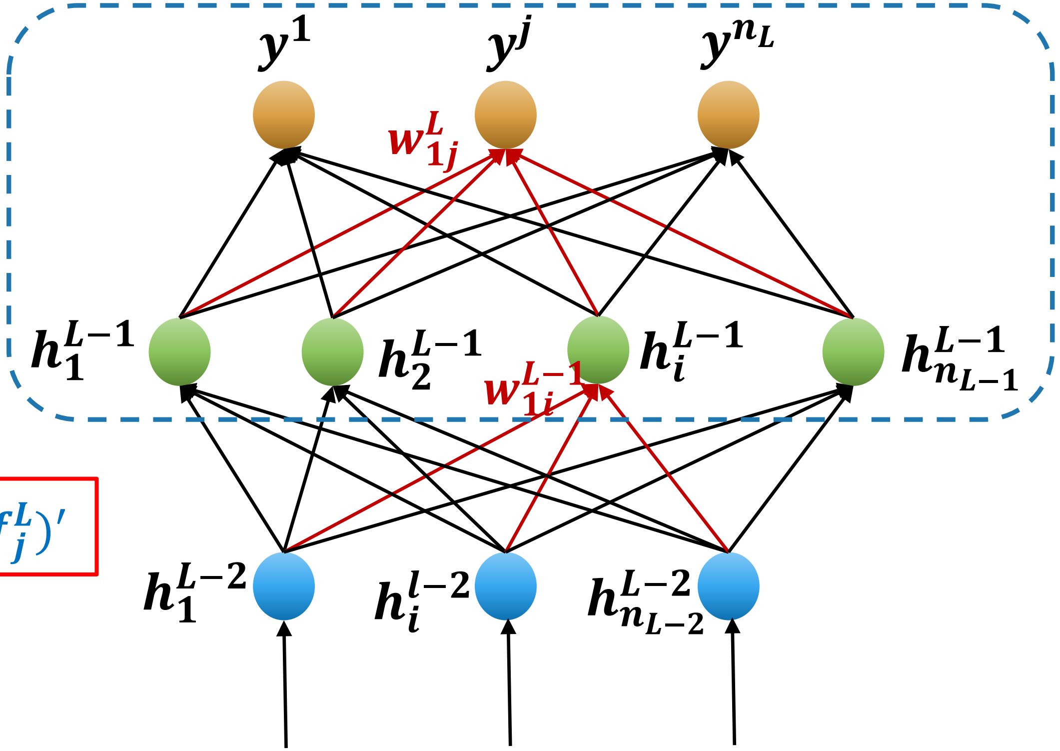
$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\gamma_h)) - y_k^j)^2$$

$$= \frac{1}{2}\sum_{j=1}^{l}(f(\sum_{h=1}^{q} w_{hj}\,(f(\sum_{i=1}^{d} v_{ih}\,x^i - \theta_1) - \theta_2) - y_k^j)^2$$

$$\frac{\partial L}{\partial v_{ih}} = \delta_h^{L-1}\cdot x^i \qquad \delta_h^{L-1} = \sum_{j=1}^{l}\delta_j^L\cdot w_{hj}\cdot f(\gamma_h)(1-f(\gamma_h))$$

BP学习算法

- 共有L层
- 第l层神经元共有n_l
- 第$l-1$层到l层的权重为：w_{ij}^{l}
- 第l层的输入为：h_i^l



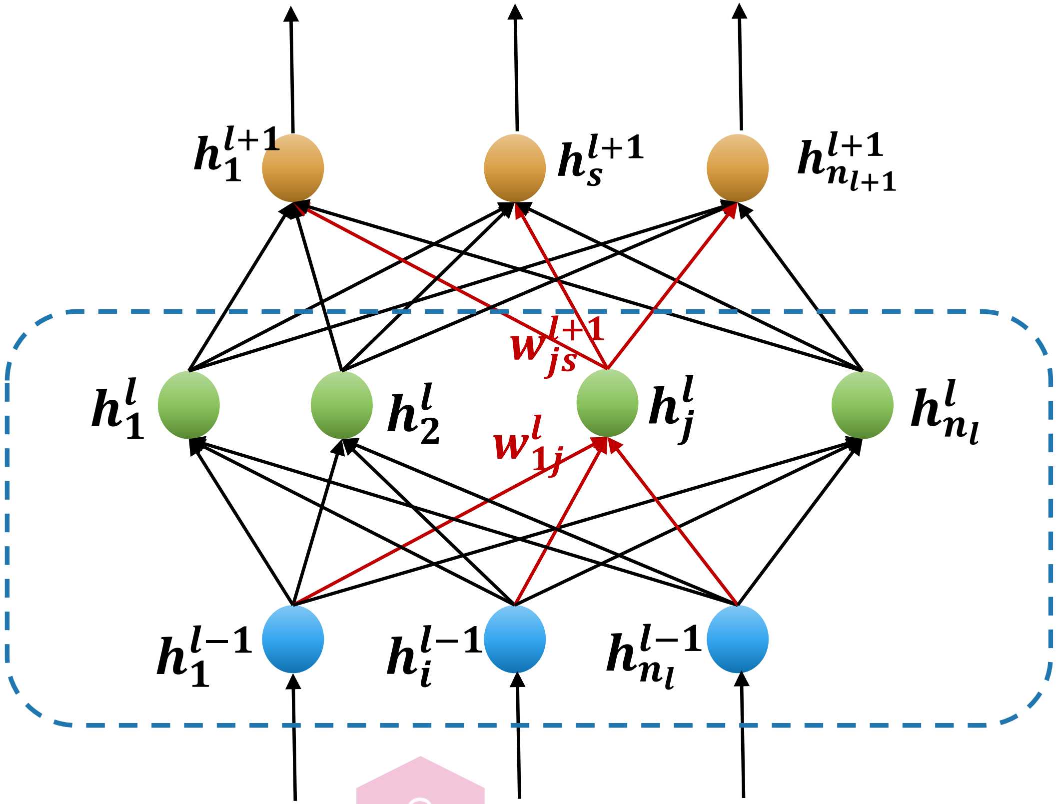
$$\boxed{\frac{\partial E}{\partial w_{ij}^{L}} = \delta_j^L \cdot h_i^{L-1}}$$

$$\boxed{\delta_j^L = (\hat{y}_k^j - y_k^j)(f_j^L)'}$$

BP学习算法

$$\frac{\partial E}{\partial w_{ij}^{l}} = \delta_j^l \cdot h_i^{l-1}$$

$$\delta_j^l = \sum_{s=1}^{n_{l+1}} \delta_s^{l+1} \cdot w_{js}^{l+1} \cdot (f_j^l)'$$



BP学习算法

所有参数初始化w, θ

for iter=1 to max （迭代次数）

for k =1 to M: (所有样本都要计算）

前向传播过程，计算出所有的输出值$\hat{y}_k^1, \cdots, \hat{y}_k^{n_L}$,并且保留所有隐层输入值h

//计算每一层的Δw $\frac{\partial E_k}{\partial \theta_j^L} = \delta_j^L$

$\frac{\partial E_k}{\partial w_{ij}^L} = \delta_j^L \cdot h_i^{L-1}; \delta_j^L = (\hat{y}_k^j - y_k^j)(f_j^L)'; f_j^L = sigmoid(\sum_{d=1}^{n_{L-1}} h_d^{L-1} w_{dj}^L - \theta_j^L)$

for l = l-1 to 2 $\frac{\partial E}{\partial \theta_j^l} = \delta_j^l$

$\frac{\partial E}{\partial w_{ij}^l} = \delta_j^l \cdot h_i^{l-1}; \delta_j^l = \sum_{s=1}^{n_{l+1}} \delta_s^{l+1} \cdot w_{js}^{l+1} \cdot (f_j^l)'; f_j^l = sigmoid(\sum_{d=1}^{n_{l-1}} h_d^{l-1} w_{dj}^l - \theta_j^l)$

//更新每一层的权值

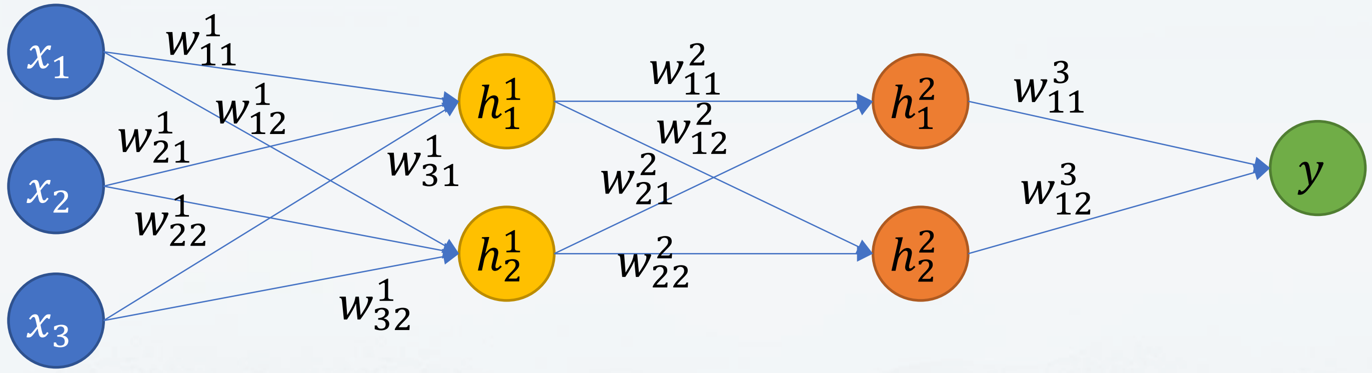
for l = 2 to L

$w_{ij}^l = w_{ij}^l - \eta \sum_{k=1}^{M} \frac{\partial E_k}{\partial w_{ij}^l}$

课堂练习

- 如下的BP神经网络，学习步长$\eta=1$，各点的阈值$\theta=0$, 输入层到隐层的激活函数为

$f(x)=\max(1,x)$, 隐层到隐层以及隐层到输出层的激活函数为$f(x)=\frac{1}{1+e^{-x}}$



$w_{11}^1=1$
$w_{12}^1=0$
$w_{21}^1=1$
$w_{22}^1=0.5$
$w_{31}^1=1$
$w_{32}^1=1.5$

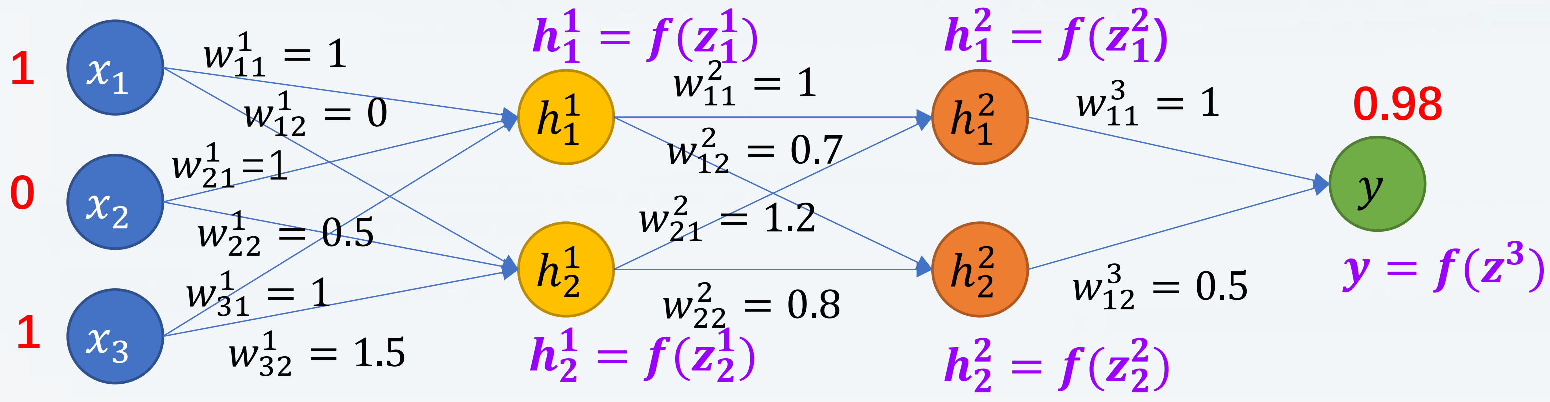
$w_{11}^2=1$
$w_{12}^2=0.7$
$w_{21}^2=1.2$
$w_{22}^2=0.8$

$w_{11}^3=1$
$w_{12}^3=0.5$

设输入样本$x_1=1, x_2=0$, $x_3=1$，输出节点的期望输出$y=0.98$

利用预测误差$E=\frac{1}{2}(\hat{y}-y)^2$对**连接权进行调整**（只调整一轮，阈值更新不计算）

课堂练习



$$w_{11}^1 = 1 \quad w_{12}^1 = 0 \quad w_{21}^1 = 1 \quad w_{22}^1 = 0.5 \quad w_{31}^1 = 1 \quad w_{32}^1 = 1.5$$

$$w_{11}^2 = 1 \quad w_{12}^2 = 0.7 \quad w_{21}^2 = 1.2 \quad w_{22}^2 = 0.8$$

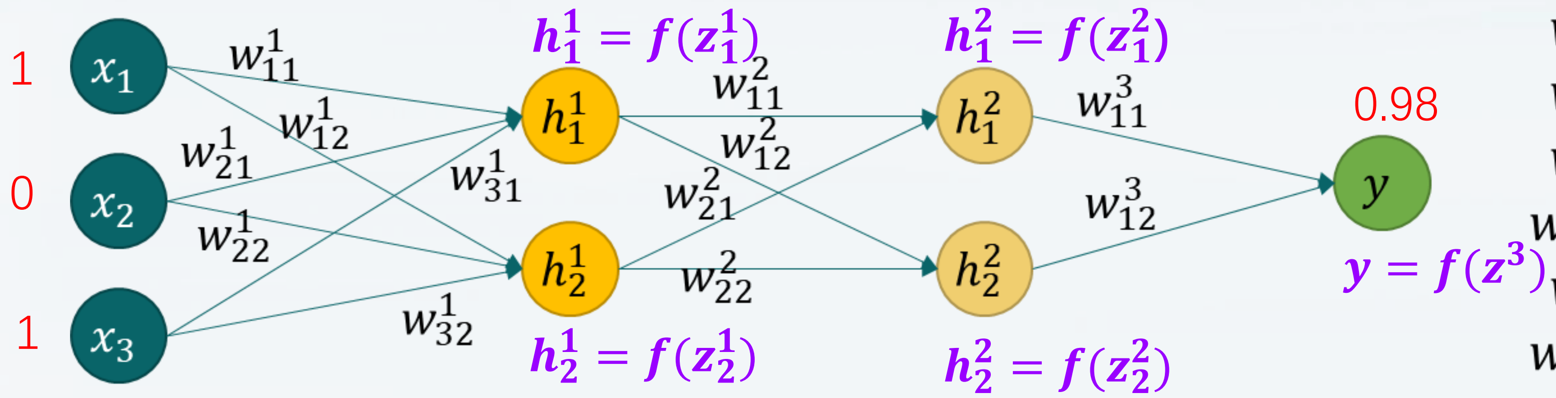
$$w_{11}^3 = 1 \quad w_{12}^3 = 0.5$$

- 前向传播
  - $z_1^1 = 1 * 1 + 0 * 1 + 1 = 2, \quad z_2^1 = 1 * 0 + 0 * 0.5 + 1 * 1.5 = 1.5$
  - $h_1^1 = \max(1,2) = 2, \quad h_2^1 = \max(1,1.5) = 1.5$
  - $z_1^2 = 2 * 1 + 1.5 * 1.2 = 3.8, \quad z_2^2 = 2 * 0.7 + 1.5 * 0.8 = 2.6$
  - $h_1^1 = \frac{1}{1+e^{-3.8}} = 0.9781, \quad h_2^1 = \frac{1}{1+e^{-2.6}} = 0.9309$

$$z^3 = 0.9781 * 1 + 0.9309 * 0.5 = 1.4436$$

$$\hat{y} = \frac{1}{1 + e^{-1.4436}} = 0.809$$

课堂练习



$w^1_{11} = 1$
$w^1_{12} = 0$
$w^1_{21} = 1$
$w^1_{22} = 0.5$
$w^1_{31} = 1$
$w^1_{32} = 1.5$

$w^2_{11} = 1$
$w^2_{12} = 0.7$
$w^2_{21} = 1.2$
$w^2_{22} = 0.8$

$w^3_{11} = 1$
$w^3_{12} = 0.5$

- 反向传播

$$f = \frac{1}{1+e^{-z}}$$

- $\delta^L_j = \frac{\partial L}{\partial z^L_j} = \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial z^L_j}$

$$\delta^3_1 = (\hat{y} - y) * \boxed{f'(z^3)} = (0.809 - 0.98) * \underbrace{\boxed{0.809 * (1 - 0.809)}}_{}\overset{}{} = -0.0264$$

$f \cdot (1 - f)$

- $\delta^l_j = \frac{\partial L}{\partial z^l_j}$

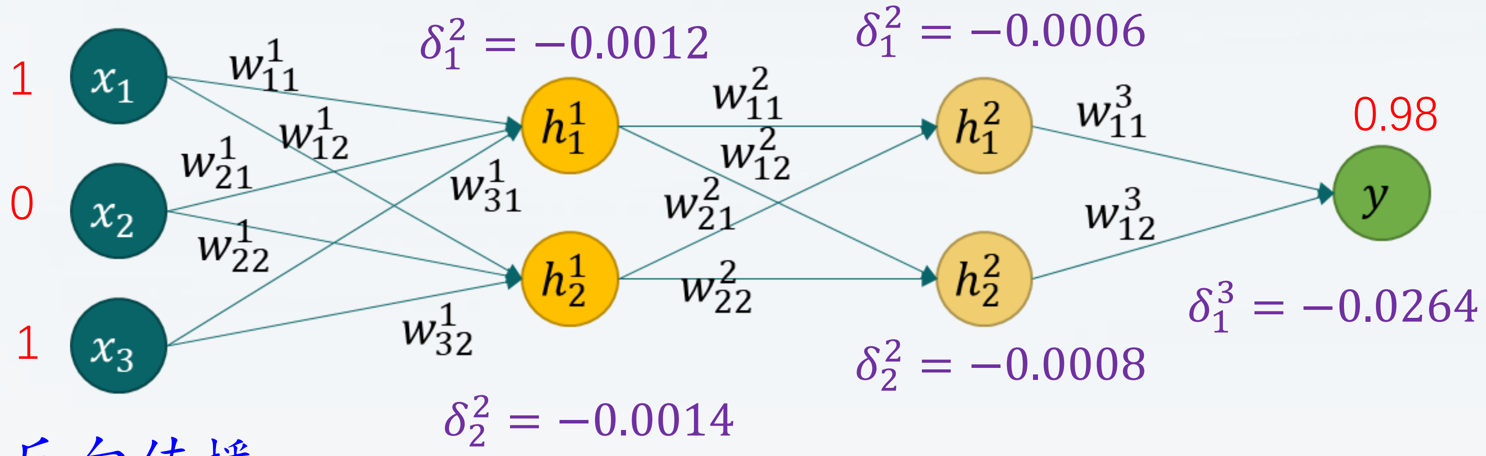
$$\delta^2_1 = \delta^3_1 * w^3_{11} * f'(z^2_1) = -0.0264 * 1 * 0.9781 * (1 - 0.9781) = -0.0006$$

$$\delta^2_2 = \delta^3_1 * w^3_{12} * f'(z^2_2) = -0.0264 * 0.5 * 0.9309 * (1 - 0.9309) = -0.0008$$

$$\delta^2_1 = (\delta^2_1 * w^2_{11} + \delta^2_2 * w^2_{12}) * f'(z^1_1) = (-0.0006 * 1 - 0.0008 * 0.7) * 1 = -0.0012$$

$$\delta^2_2 = (\delta^2_1 * w^2_{21} + \delta^2_2 * w^2_{22}) * f'(z^1_2) = (-0.0006 * 1.2 - 0.0008 * 0.8) * 1 = -0.0014$$

课堂练习



$w^1_{11} = 1$
$w^1_{12} = 0$
$w^1_{21} = 1$
$w^1_{22} = 0.5$
$w^1_{31} = 1$
$w^1_{32} = 1.5$

$w^2_{11} = 1$
$w^2_{12} = 0.7$
$w^2_{21} = 1.2$
$w^2_{22} = 0.8$

$w^3_{11} = 1$
$w^3_{12} = 0.5$

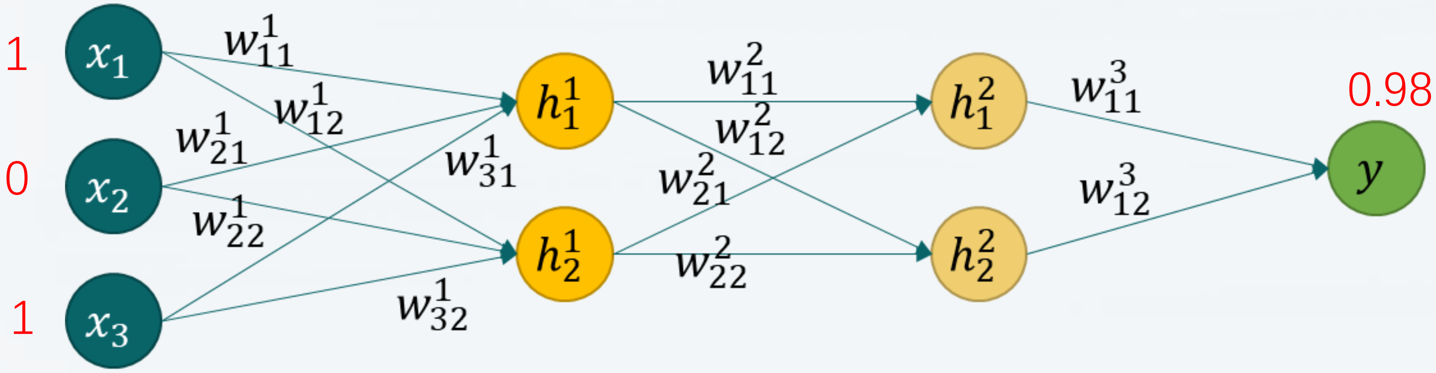
- 反向传播

$$\Delta w^3_{11} = \delta^3_1 * h^2_1 = -0.0264 * 0.9781 = -0.0258$$
$$\Delta w^3_{12} = \delta^3_1 * h^2_2 = -0.0264 * 0.9309 = -0.0246$$

$$\Delta w^2_{11} = \delta^2_1 * h^1_1 = -0.0006 * 2 = -0.0012$$
$$\Delta w^2_{12} = \delta^2_2 * h^1_1 = -0.0008 * 2 = -0.0016$$
$$\Delta w^2_{21} = \delta^2_1 * h^1_1 = -0.0006 * 1.5 = -0.0009$$
$$\Delta w^2_{22} = \delta^2_2 * h^1_2 = -0.0008 * 1.5 = -0.0012$$

$$\Delta w^1_{11} = \delta^1_1 * x_1 = -0.0012 * 1 = -0.0012$$
$$\Delta w^1_{12} = \delta^1_2 * x_1 = -0.0014 * 1 = -0.0014$$
$$\Delta w^1_{21} = \delta^1_1 * x_2 = -0.0012 * 0 = -0.0012$$
$$\Delta w^1_{22} = \delta^1_2 * x_2 = -0.0014 * 0. = -0.0014$$
$$\Delta w^1_{31} = \delta^1_1 * x_3 = -0.0012 * 1 = -0.0012$$
$$\Delta w^1_{32} = \delta^1_2 * x_3 = -0.0014 * 1 = -0.0014$$

课堂练习



$w_{11}^1 = 1$
$w_{12}^1 = 0$
$w_{21}^1 = 1$
$w_{22}^1 = 0.5$
$w_{31}^1 = 1$
$w_{32}^1 = 1.5$

$w_{11}^2 = 1$
$w_{12}^2 = 0.7$
$w_{21}^2 = 1.2$
$w_{22}^2 = 0.8$

$w_{11}^3 = 1$
$w_{12}^3 = 0.5$

- 反向传播

$w_{11}^1 = 1 + 1 * 0.0012 = 1.0012$
$w_{12}^1 = 0 + 0.0014 = 0.0014$
$w_{21}^1 = 1 + 0 = 1$
$w_{22}^1 = 0.5 + 0 = 0.5$
$w_{31}^1 = 1 + 0.0012 = 1.0012$
$w_{32}^1 = 1.5 + 0.0014 = 1.5014$

$w_{11}^2 = 1 + 0.0012 = 1.0012$
$w_{12}^2 = 0.7 + 0.0016 = 0.7016$
$w_{21}^2 = 1.2 + 0.0009 = 1.2009$
$w_{22}^2 = 0.8 + 0.0012 = 0.8012$

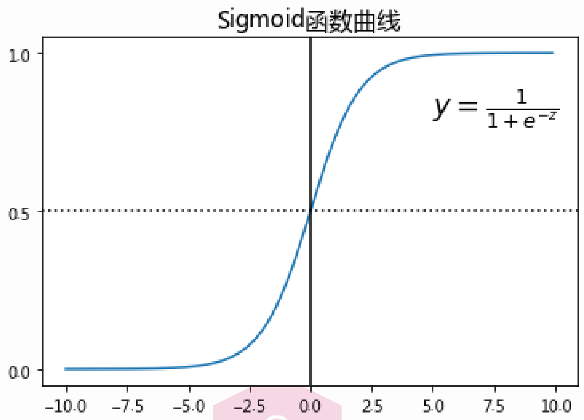
$w_{11}^3 = 1 + 0.0258 = 1.0258$
$w_{12}^3 = 0.5 + 0.0246 = 0.5246$

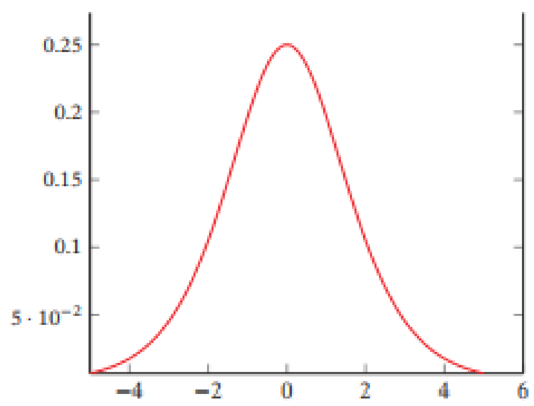
神经网络

- 在1989年以后由于没有特别突出的方法被提出，且NN一直缺少相应的严格的数学理论支持，神经网络的热潮渐渐冷淡下去。冰点来自于1991年，BP算法被指出存在梯度消失问题

Sigmoid函数： $\sigma(x) = \frac{1}{1+\exp(-x)}$

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





饱和区的导数接近0

误差传递不断衰减

梯度消失

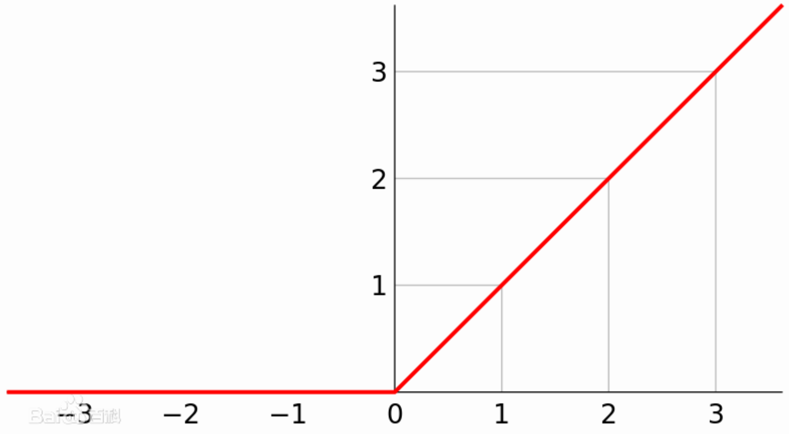
神经网络

- 在1989年以后由于没有特别突出的方法被提出，且NN一直缺少相应的严格的数学理论支持，神经网络的热潮渐渐冷淡下去。冰点来自于1991年，BP算法被指出存在梯度消失问题
- 2006年，Hinton提出了深层网络训练中梯度消失问题的解决方案
- 2011年，ReLU激活函数被提出，该激活函数能够有效的抑制梯度消失问题

激活函数

- ReLU函数

  - $ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$



1) 计算更加高效
2) 求导在x>0时为1，一定程度缓解梯度消失现象

如果出现不恰当更新，某个神经元不被激活，自身梯度永远为0：死亡ReLU问题

Optimization of New Model

$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

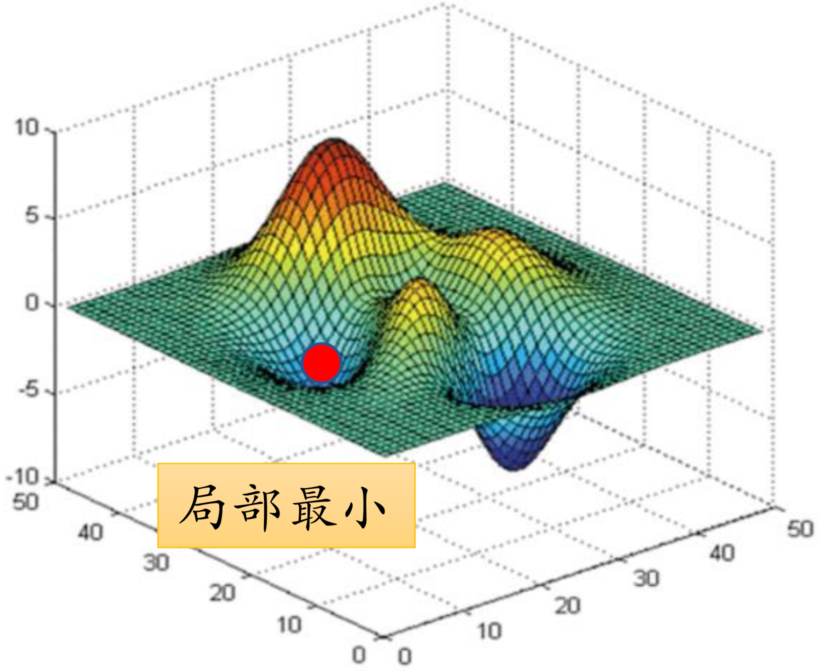
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

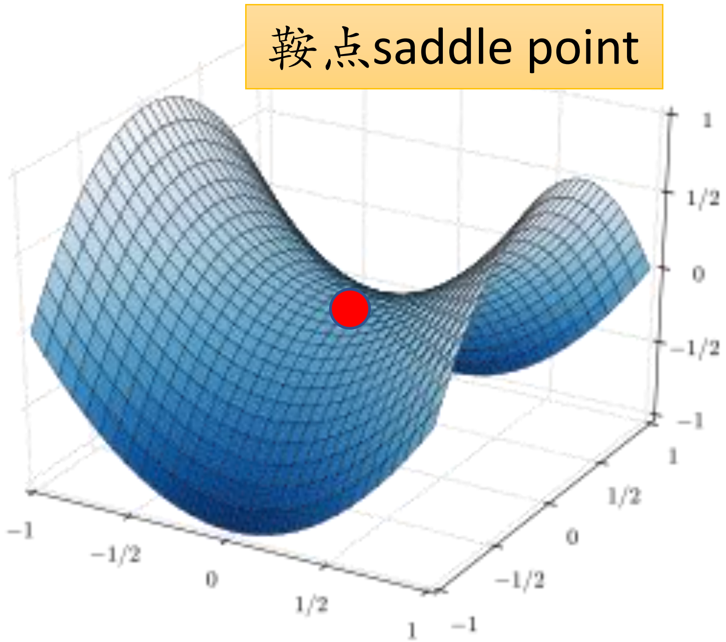
$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$$

- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$$

Optimization of New Model





critical point

Batch

Optimization of New Model

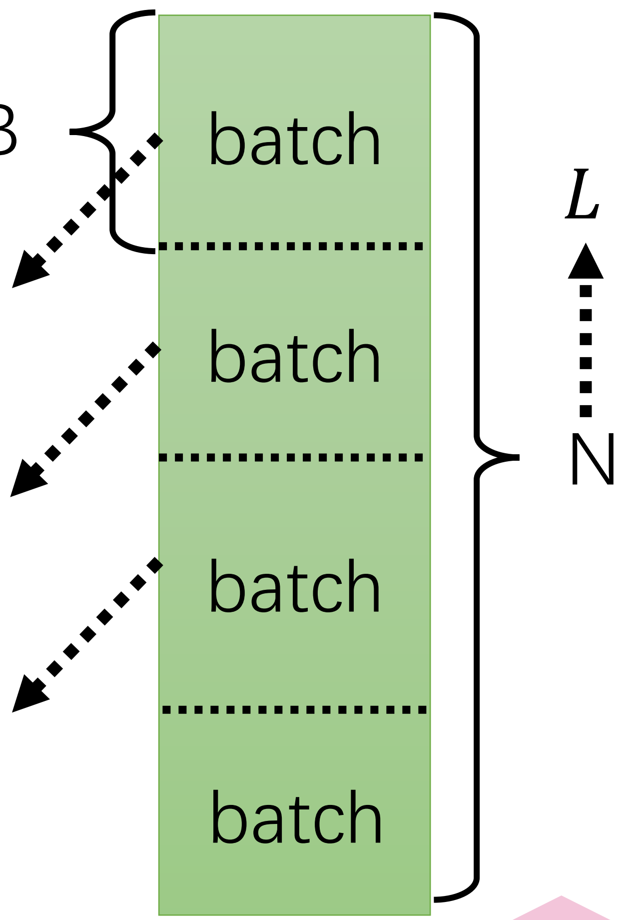
$$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$ L^1

  update $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^2(\boldsymbol{\theta}^1)$ L^2

  update $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^3(\boldsymbol{\theta}^2)$ L^3

  update $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$



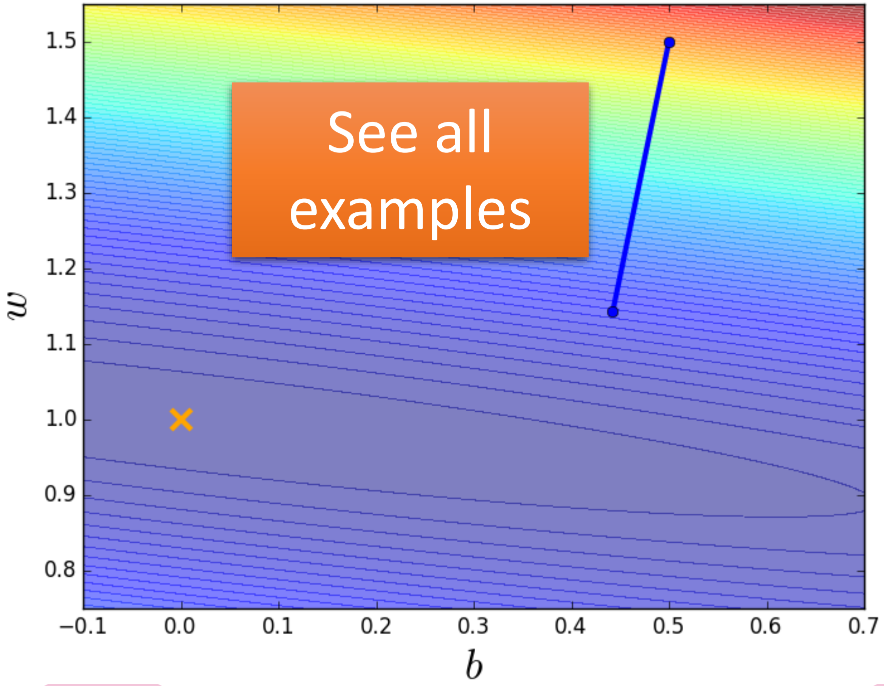
1 **epoch** = see all the batches once

Batch

Consider 20 examples (N=20)

Batch size = N (Full batch)

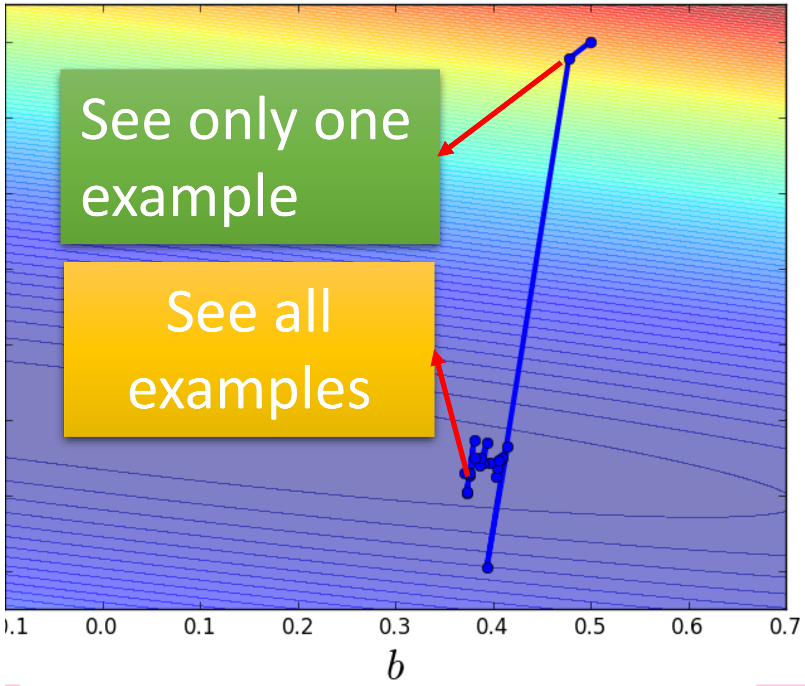
Update after seeing all the 20 examples



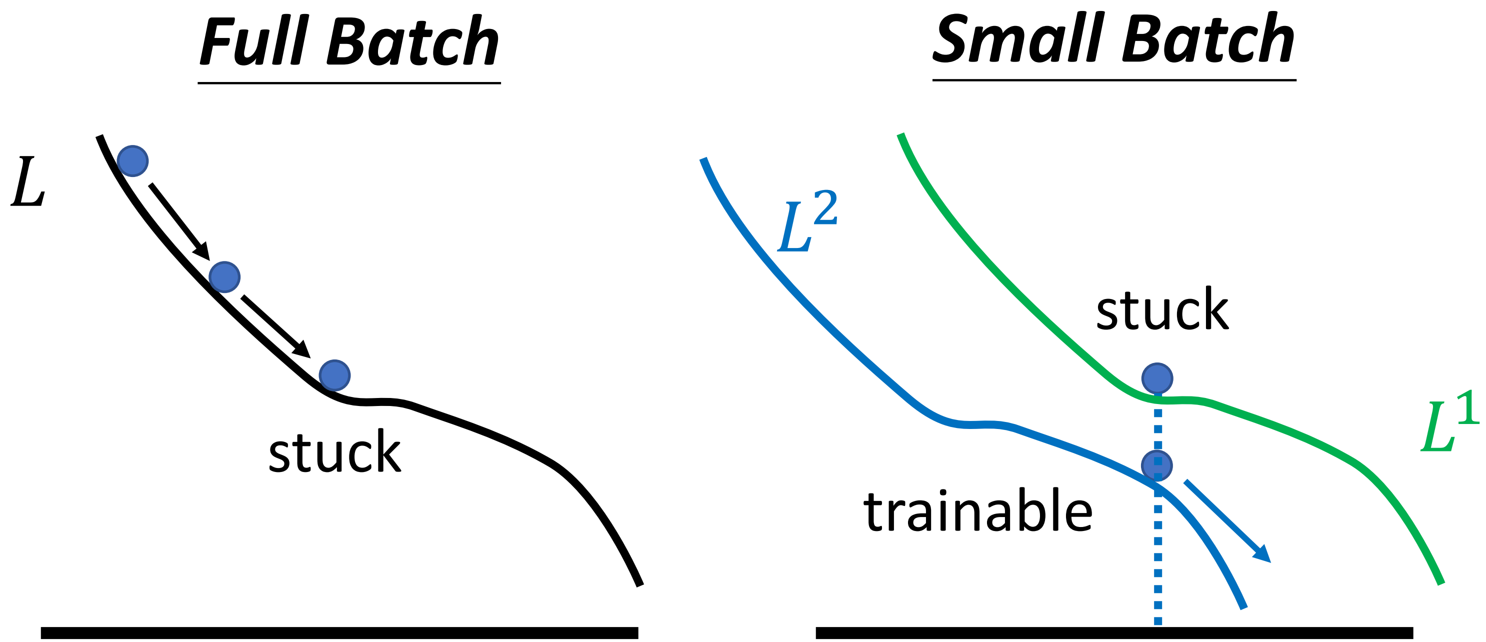
Batch size = 1

Update for each example

Update 20 times in an epoch

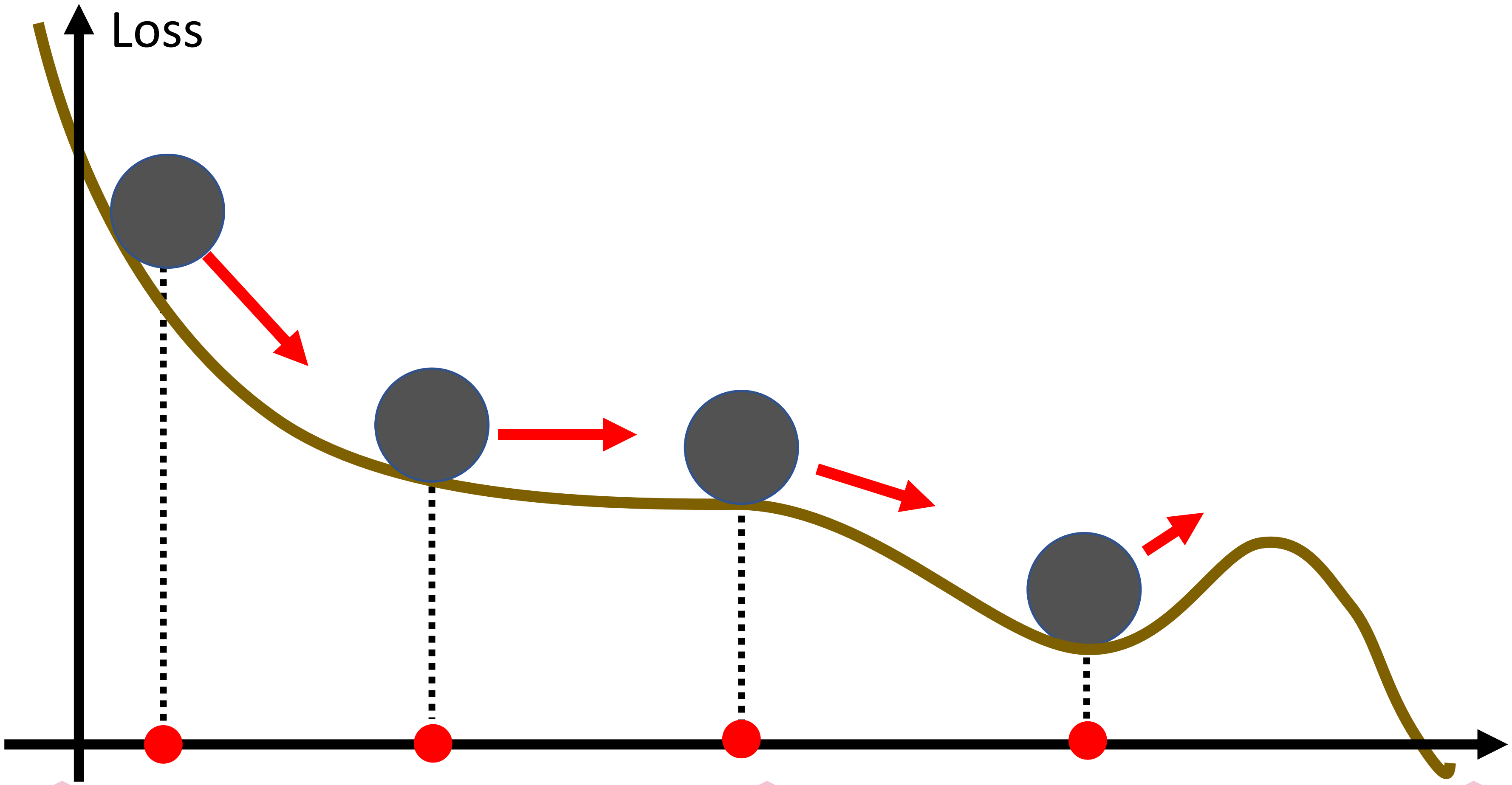


Batch

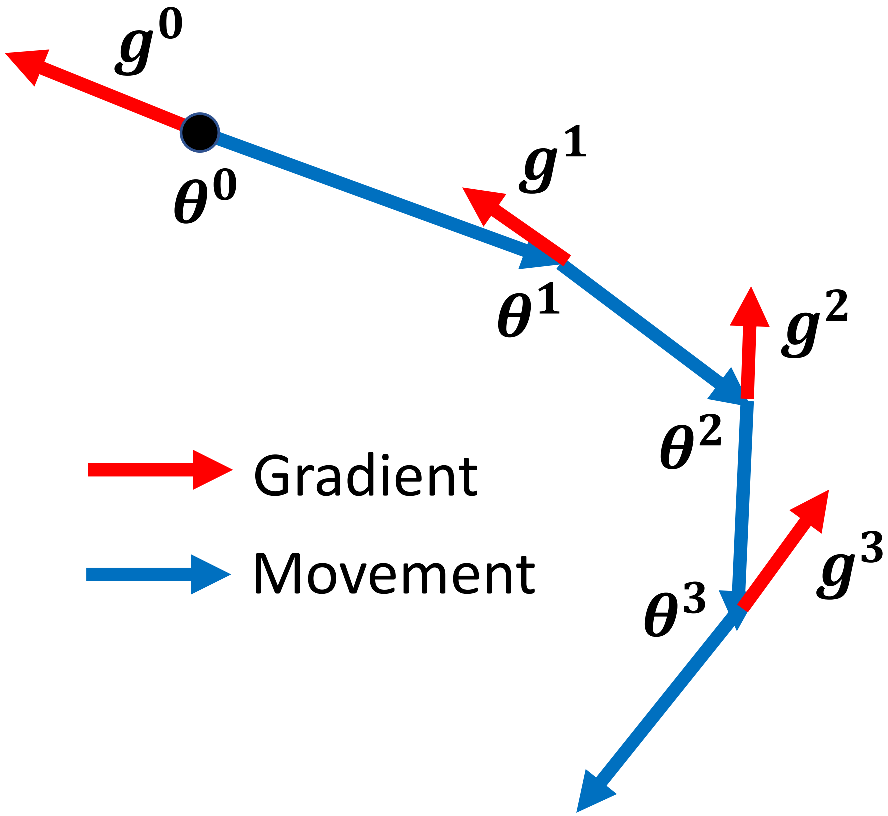


Momentum

Momentum



(Vanilla) Gradient Descent



初始 $\boldsymbol{\theta}^0$

梯度计算$\boldsymbol{g}^0$

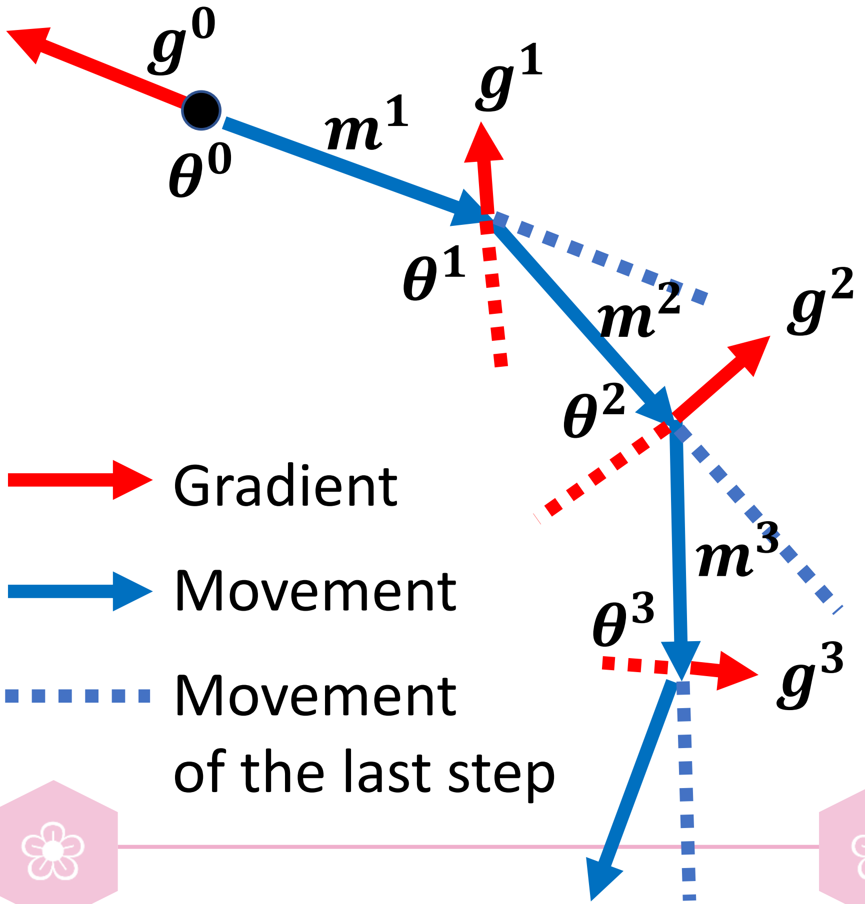
$\boldsymbol{\theta}^1 = \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$

梯度计算$\boldsymbol{g}^1$

$\boldsymbol{\theta}^2 = \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^1$

⋮

Gradient Descent + Momentum



初始 θ^0

设$m^0 = 0$

梯度计算 g^0

$m^1 = \lambda m^0 - \eta g^0$

$\theta^1 = \theta^0 + m^1$

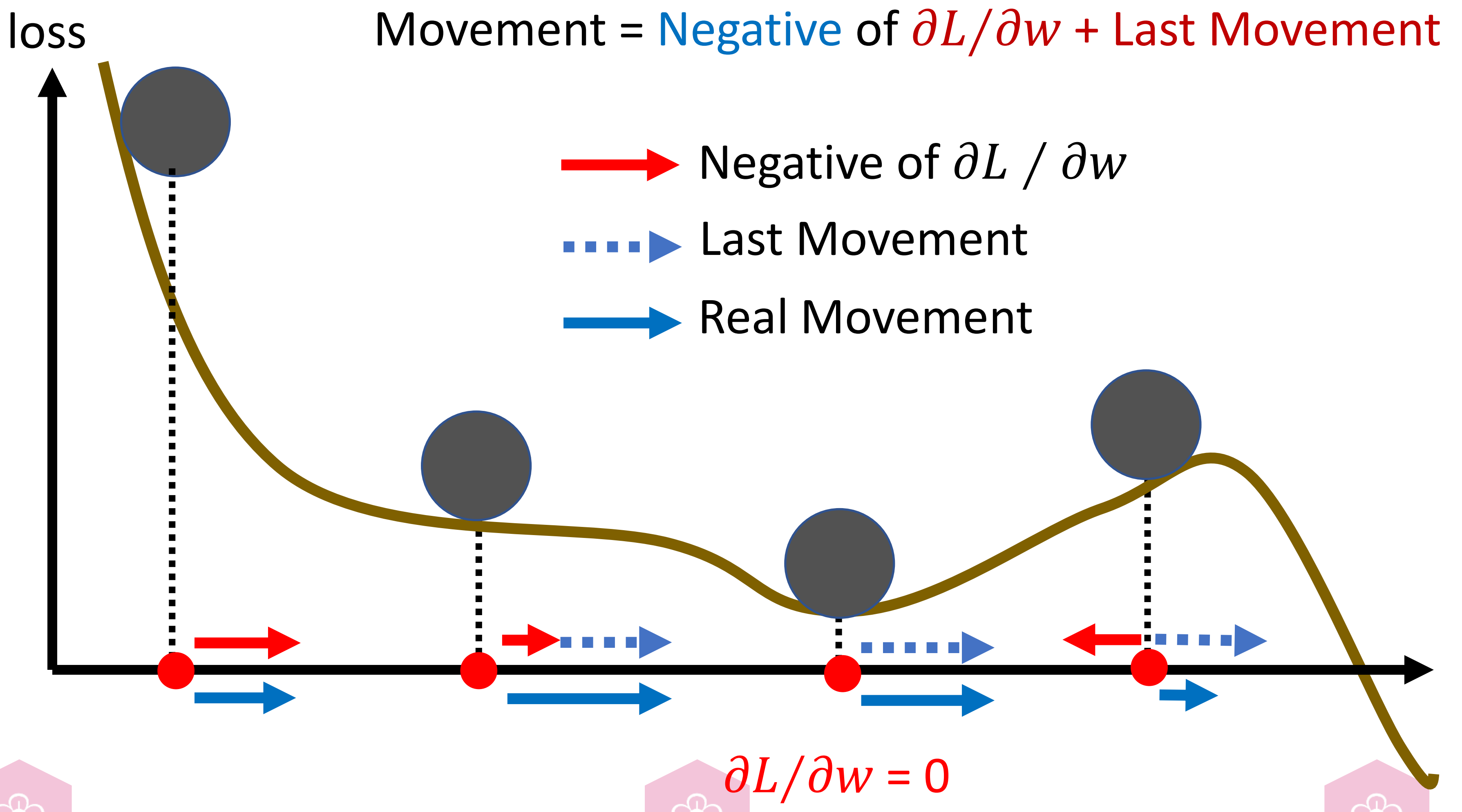
梯度计算g^1

$m^2 = \lambda m^1 - \eta g^1$

$\theta^2 = \theta^1 + m^2$

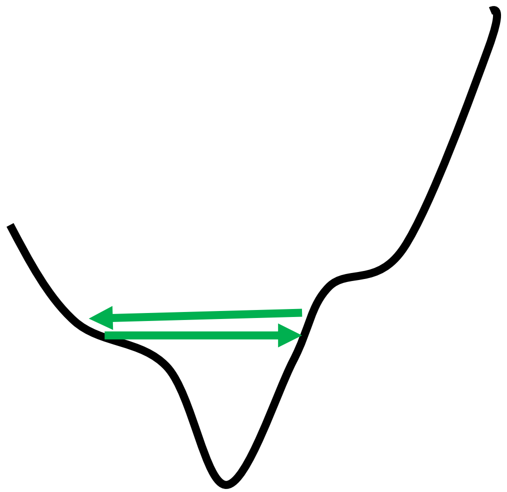
Gradient Descent + Momentum

Movement = Negative of $\partial L / \partial w$ + Last Movement

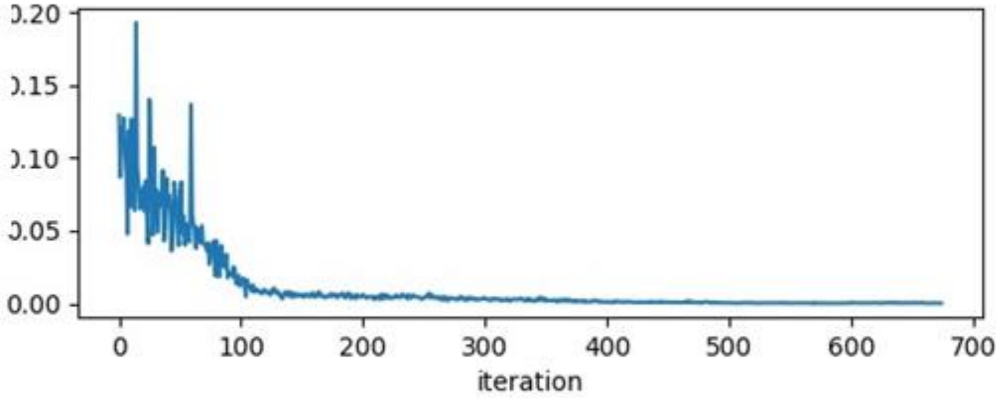


Learning rate

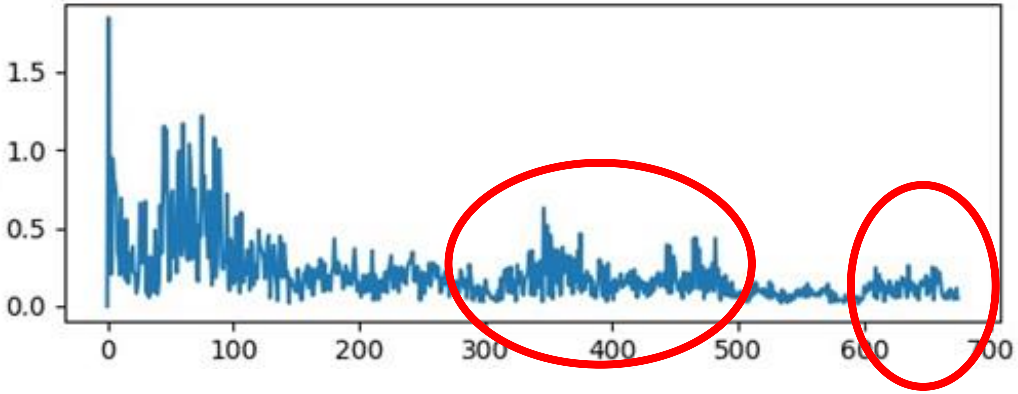
Training stuck ≠ Small Gradient

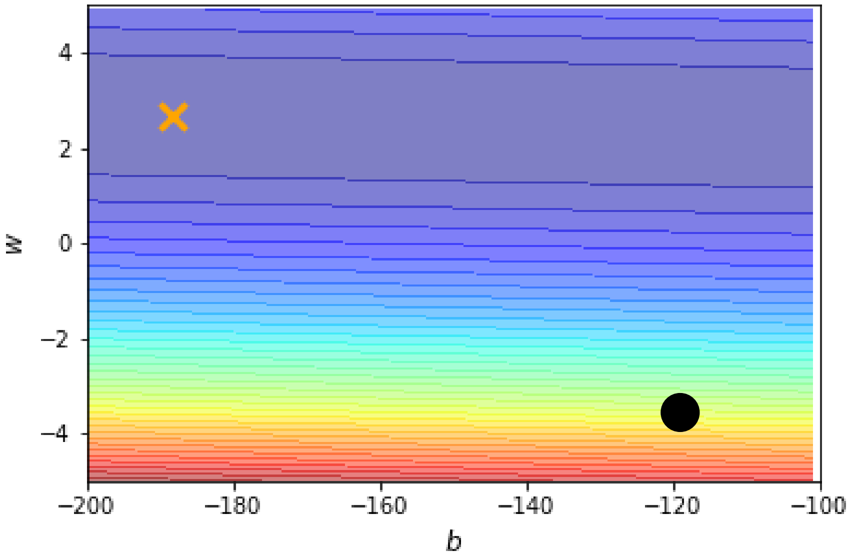


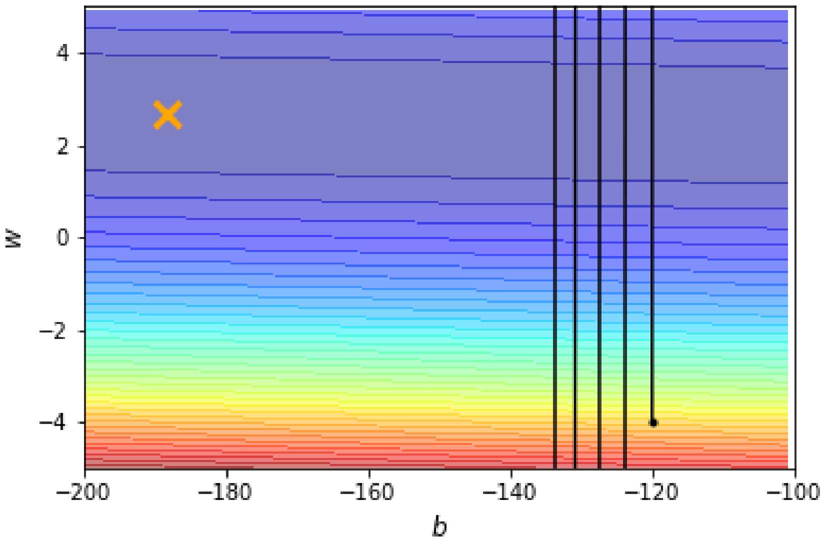
loss



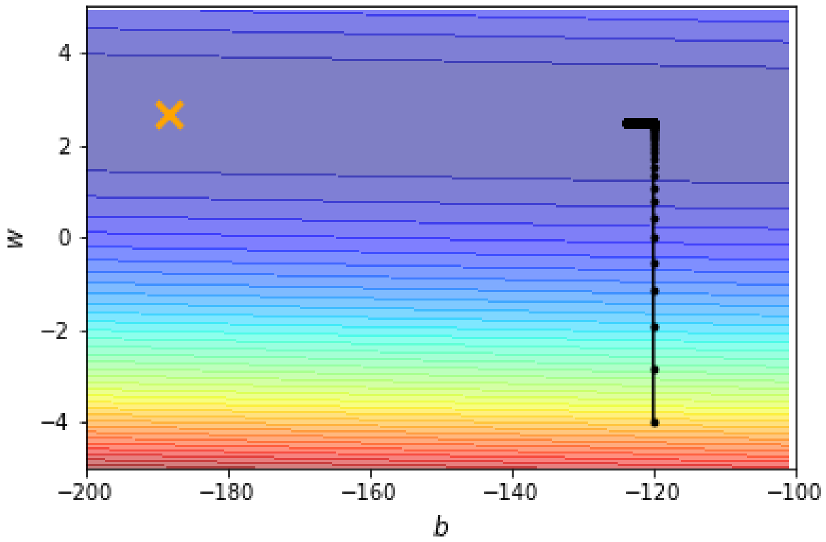
norm of gradient





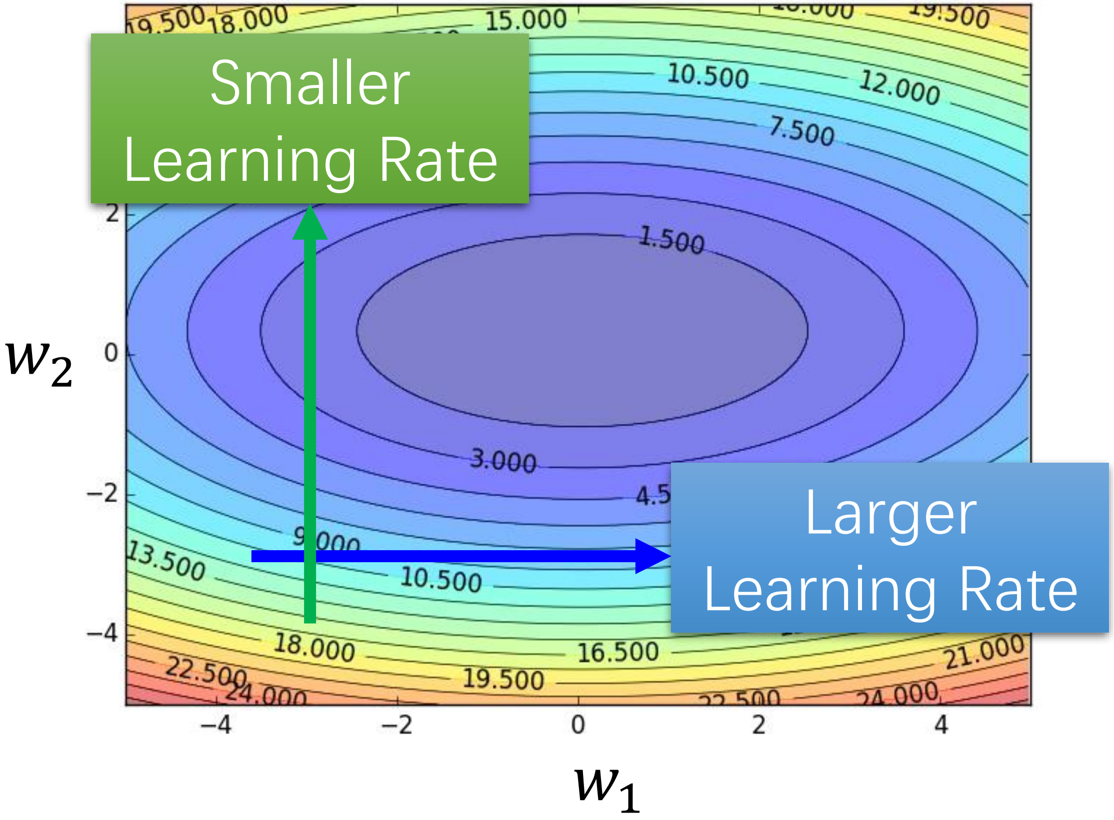


$\eta = 10^{-2}$



$\eta = 10^{-7}$

Learning rate



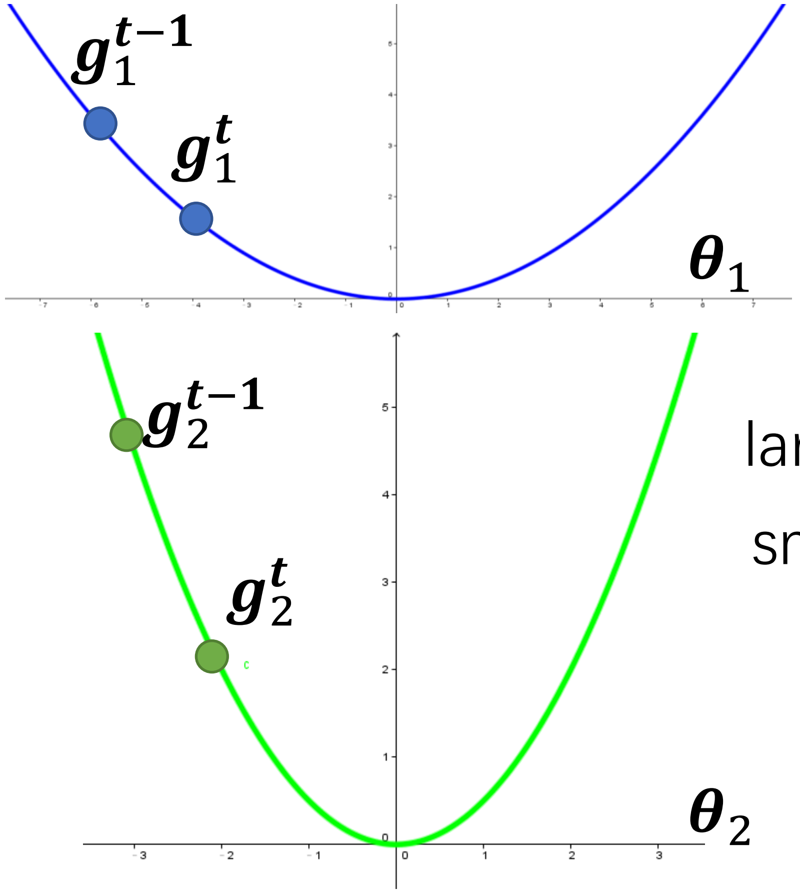
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\eta}\, \boldsymbol{g}_i^{t}$$

$$\boldsymbol{g}_i^{t} = \frac{\partial L}{\partial \boldsymbol{\theta}_i} \big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t}$$

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}}\, \boldsymbol{g}_i^{t}$$

Learning rate

- $\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^{t}$
  - $\sigma_i^t = \sqrt{\frac{1}{t+1} \sum_{i=0}^{t} \left(\boldsymbol{g}_i^t\right)^2}$
    - **Adagrad**



smaller σ_1^t

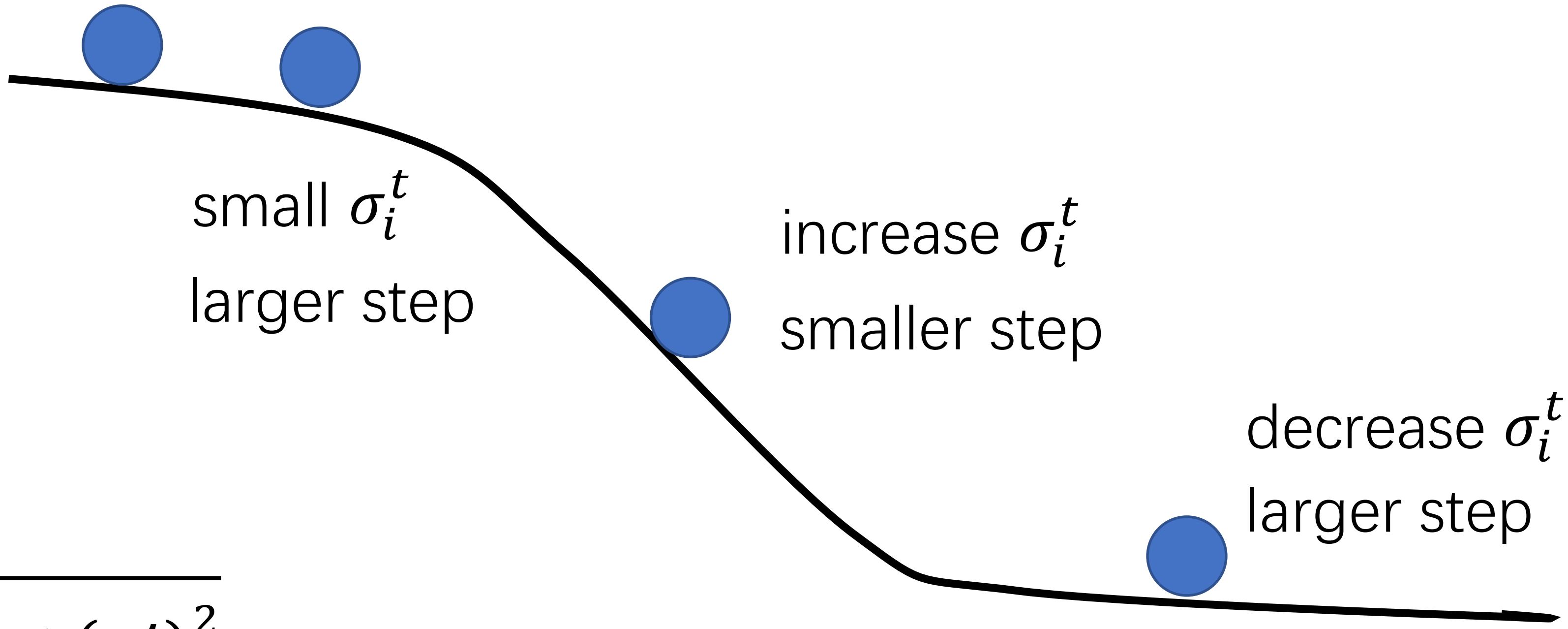
larger step

larger σ_2^t

smaller step

Learning rate

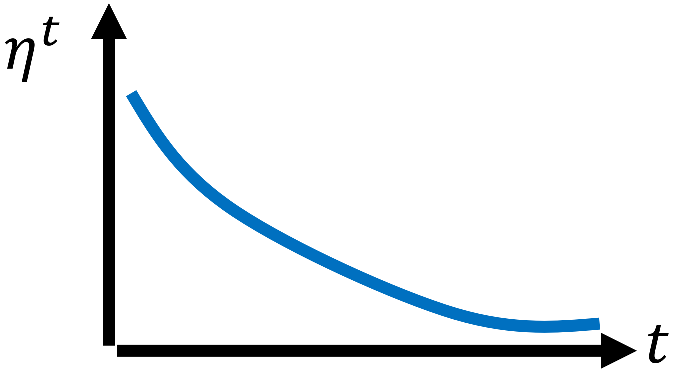
- $\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta}{\sigma_i^t}\boldsymbol{g}_i^t$
  - $\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}\left(\boldsymbol{g}_i^t\right)^2}$
    - **Adagrad**
  - $\sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^t\right)^2}$
    - **RMSProp**



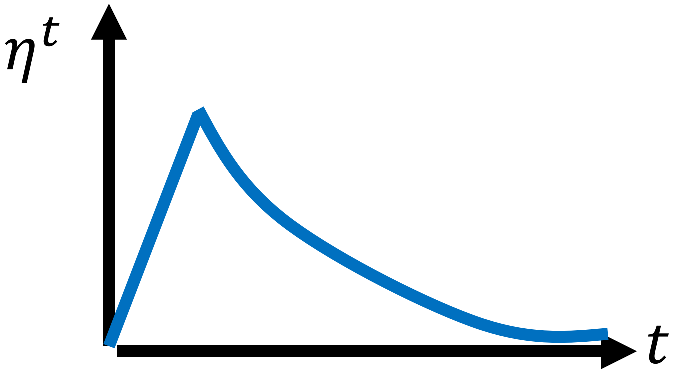
Adam:结合了动量梯度下降和RMSProp两种算法的优点

Learning rate

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^t$$



Learning Rate Decay



Warm Up

Summary of Optimization

(Vanilla) Gradient Descent

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \eta \boldsymbol{g}_i^{t}$$

Various Improvements

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta^t}{\sigma_i^t} \boldsymbol{m}_i^{t}$$

Learning rate scheduling

Momentum: weighted sum of the previous gradients

确定方向

root mean square of the gradients

确定大小