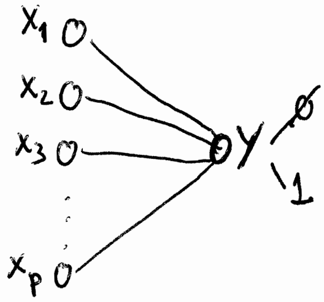
1

1 perceptron



2

X	T	Y
x^1	t_1	y_1
x^2	t_2	y_2
x^3	t_3	y_3
x^i	t_i	y_i
x^N	t_N	y_N

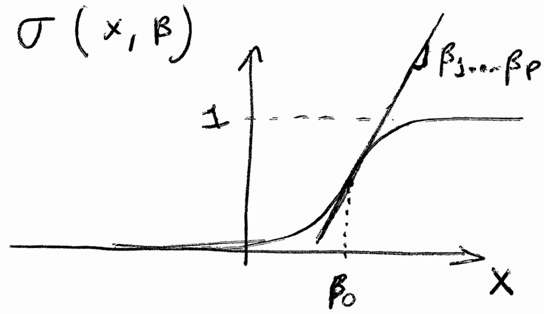
data

target label

predicted label

3 $y = \sigma(\underline{x}, \beta)$

$\sigma(x, \beta)$



$$\sigma(x, \beta) = \frac{1}{1 + \exp(\beta_0 + \beta \cdot x)}$$

$$\sigma'(x, \beta) = \sigma(x, \beta) \cdot (1 - \sigma(x, \beta))$$

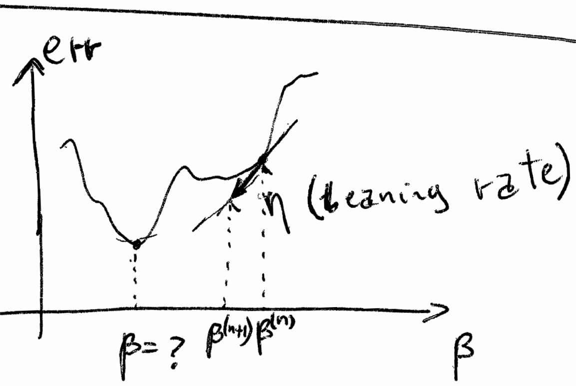
4 Error function

$$err = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2$$

$$err(\beta) = \frac{1}{N} \sum_{i=1}^{N} (t_i - y(x^i, \beta))^2$$

5 Finding the minimum of the error function

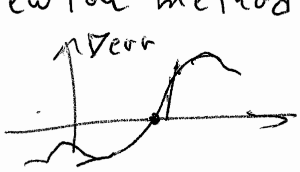
Gradient descent



Other methods:

· Golden section search (greedy)

Newton method (zero of the gradient)

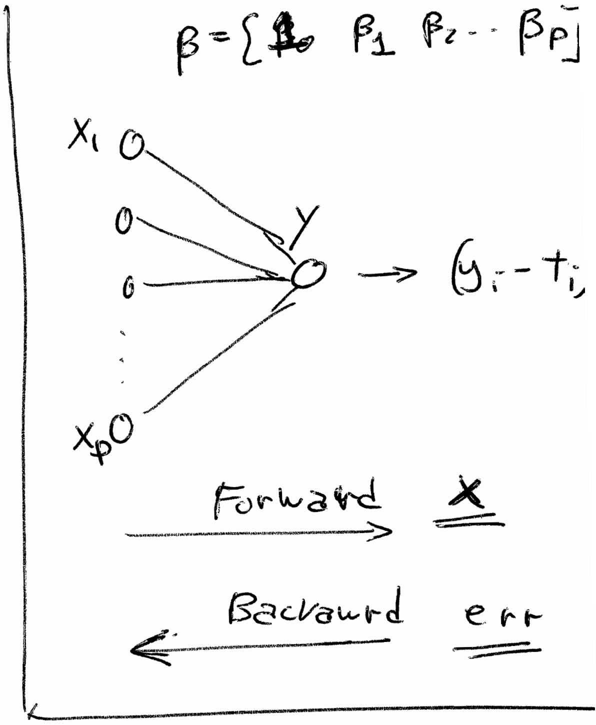


Update rule:

$$\beta^{(n+1)} = \underbrace{\beta^{(n)}}_{\text{previous point}} + \underbrace{\eta}_{\text{learning rate}} \underbrace{\nabla err(\beta^{(n)})}_{\text{Tangent}}$$

Training

- shuffle data randomly
- initiate $\cancel{\ }\beta$ randomly



(1) Forward:

$$y_i = \sigma(x^i, \beta)$$

(2) Backward

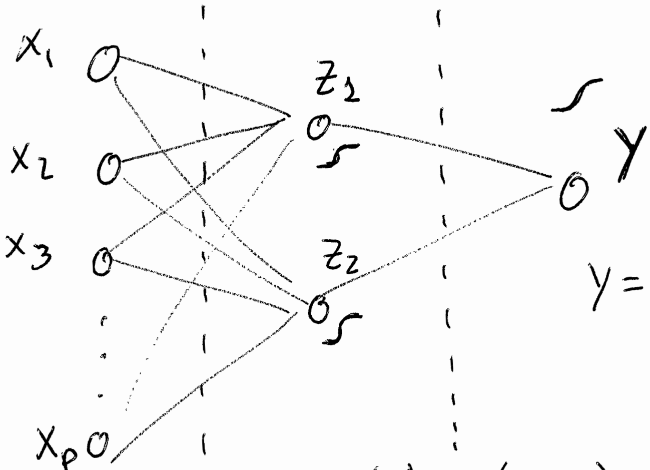
Gradient:

$$\Delta err_\beta = (t_i - y_i) \cdot y_i (1 - y_i)$$

Update:

$$\beta_j^{(n+1)} = \beta_j^{(n)} + \eta \cdot \Delta err_\beta \cdot x_j^i$$

(6)



$y = \sigma(z, \beta)$

$y = \sigma(\sigma(x, \alpha), \beta)$

$z_1(x) = \sigma(x, \alpha)$

---

(7)

$$err(\alpha, \beta) = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2$$

$$err(\alpha, \beta) = \frac{1}{N} \sum_{i=1}^{N} (t_i - y(x^i, \alpha, \beta))$$

---

(8)

$$\begin{cases} \alpha^{(n+1)} = \alpha^{(n)} + \eta \sum_{i=1}^{N} \frac{\partial err(\alpha^{(n)}, \beta^{(n)}, x^i)}{\partial \alpha} \\ \beta^{(n+1)} = \beta^{(n)} + \eta \sum_{i=1}^{N} \frac{\partial err(\alpha^{(n)}, \beta^{(n)}, x^i)}{\partial \beta} \end{cases}$$

$\boxed{\text{gradient descent}}$

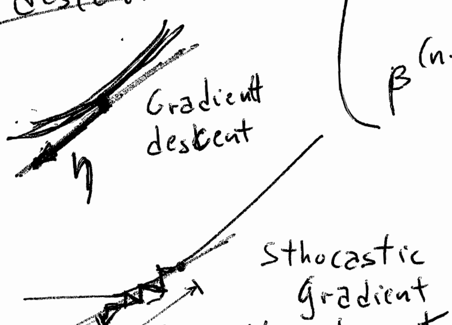
∇err is robust

$\downarrow$ faster

$\boxed{\text{Sthocastic gradient descent}}$

(9)

$$\begin{cases} \alpha^{(n+1)} = \alpha^{(n)} + \eta \frac{\partial err(\alpha^{(n)}, \beta^{(n)}, x^i)}{\partial \alpha} \\ \beta^{(n+1)} = \beta^{(n)} + \eta \frac{\partial err(\alpha^{(n)}, \beta^{(n)}, x^i)}{\partial \beta} \end{cases}$$



Training

- Shuffle data randomly
- initialize α's and β's randomly $[-0.1, +0.1]$

for each sample i

① Forward

$$z_1^i = \sigma(x^i, \alpha_1)$$

$$z_2^i = \sigma(x^i, \alpha_2)$$

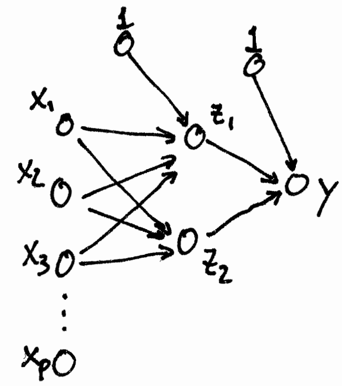
$$y^i = \sigma(z^i, \beta) \qquad z^i = [1 \; z_1^i \; z_2^i]$$

② Backward

$$\text{Gradients} = \begin{cases} \dfrac{\partial err}{\partial \beta} = (t_i - y_i) \cdot y_i \cdot (1 - y_i) \\[2ex] \dfrac{\partial err}{\partial \alpha_1} = z_1^i (1 - z_1^i) \cdot \beta_1 \cdot \dfrac{\partial err}{\partial \beta} \\[2ex] \dfrac{\partial err}{\partial \alpha_2} = z_2^i (1 - z_2^i) \cdot \beta_2 \cdot \dfrac{\partial err}{\partial \beta} \end{cases}$$

$$\eta = 0.5$$

$$\text{Updates} = \begin{cases} \beta^{(n+1)} = \beta^{(n)} + \eta \cdot \dfrac{\partial err}{\partial \beta} \cdot z^i \\[2ex] \alpha_1^{(n+1)} = \alpha_1^{(n)} + \eta \cdot \dfrac{\partial err}{\partial \alpha_1} \cdot x^i \\[2ex] \alpha_2^{(n+1)} = \alpha_2^{(n)} + \eta \cdot \dfrac{\partial err}{\partial \alpha_2} \cdot x^i \end{cases}$$



$$x = [x_1 \; \dots \; x_p]$$

$$z_1 = \sigma(x, \alpha_1)$$

$$z_2 = \sigma(x, \alpha_2)$$

$$Y = \sigma(z = [1 \; z_1 \; z_2], \beta)$$

$$\alpha_1 = [\alpha_{10} \; \alpha_{11} \; \cdots \; \alpha_{1p}]$$

$$\alpha_2 = [\alpha_{20} \; \alpha_{21} \; \cdots \; \alpha_{2p}]$$

$$\beta = [\beta_0 \; \beta_1 \; \beta_2]$$