# Deep Learning


Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir

https://www.aparat.com/mehran.safayani

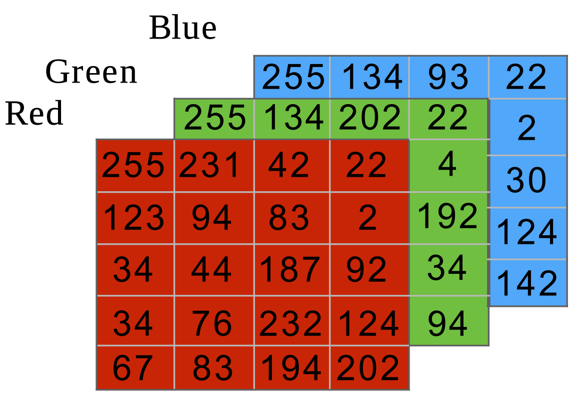
https://github.com/safayani/deep_learning_course

Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran


# Basics of Neural Network Programming

## Binary Classification

# Binary classification




$$\vec{x} = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 254 \\ 253 \\ 250 \\ 220 \end{bmatrix} \quad 64 \times 64 \times 3 = \underbrace{12288}_{n=n_x}$$

$$\vec{x} \rightarrow \boxed{\text{model}} \rightarrow \hat{y}$$

- Notation

$(\vec{x}, y) \qquad x \in R^{n_x}, \;\; y \in \{0,1\}$

# Binary classification

- m training example: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}, \ldots, (x^{(m)}, y^{(m)})\}$
- $$X = \begin{array}{cc} \left[\begin{matrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots\ldots\ldots & x^{(\mathrm{m})} \\ | & | & & | \end{matrix}\right] & \mathrm{m} \\ n_x & \end{array}$$

$\mathrm{X} \in \mathbb{R}$

$\mathrm{X.shape} = (n_x, m)$

$\mathrm{Y} = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]$

$\mathrm{Y} \in \mathbb{R}^{1 \times m}$

$\mathrm{Y.shape} = (1, m)$

# Logistic Regression

- Given x , output $\hat{y} = P(y=1|x)$  $0 \leq \hat{y} \leq 1$

  $x \in \mathbb{R}^{n_x}$      parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

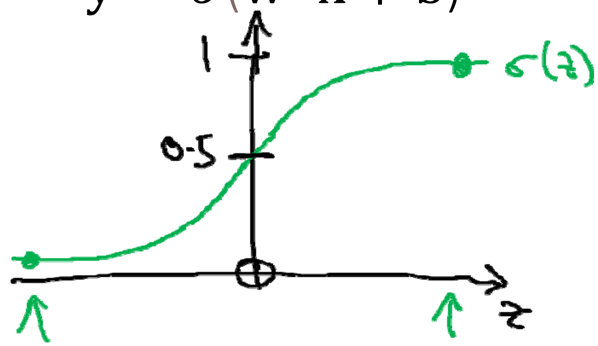
$$\hat{y} = w^T x + b$$

$$\hat{y} = \sigma(\overbrace{w^T x + b}^{z})$$

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

if z large $\sigma(z) \approx 1$

if z large negative $\sigma(z) \approx 0$



# Logistic Regression

- $\hat{y} = \sigma(\underbrace{w^T x + b}_{z})$

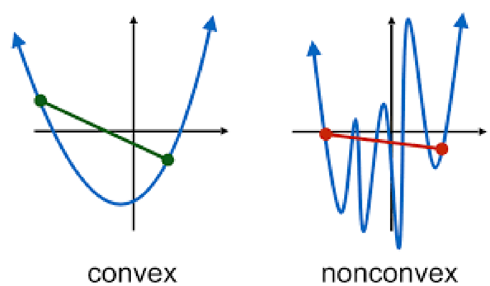
$x_0 = 1, x \in \mathbb{R}^{n_x+1}$

$$\hat{y} = w^T x$$

$$W = \begin{bmatrix} b = w_0 \\ w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_{nx} \end{bmatrix} \begin{array}{l} \}\ b \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\}\ w \end{array}$$

# Logistic Regression cost function

- Loss (error) function: $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}\left(\sigma\left(w^T x + b\right) - y\right)^2$ SE: Square Error
- Non-convex graph:




https://mlstory.org/optimization.html

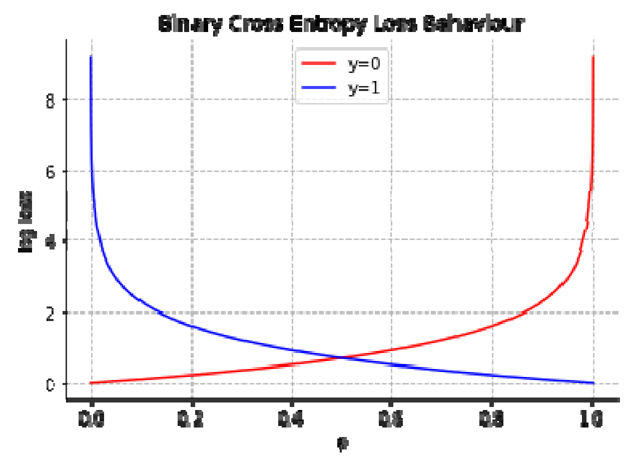
- In mathematics, a real-valued function is called **convex** if the line segment between any two distinct points on the graph of the function lies above the graph between the two points.
- Equivalently, a function is convex if its epigraph (the set of points on or above the graph of the function) is a convex set.

# Cross Entropy

- $L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$

if $y=1$ : $L(\hat{y}, y) = -\log \hat{y}$

if $y=0$ : $L(\hat{y}, y) = -\log(1-\hat{y})$




https://datamonje.com/classification-loss-functions/

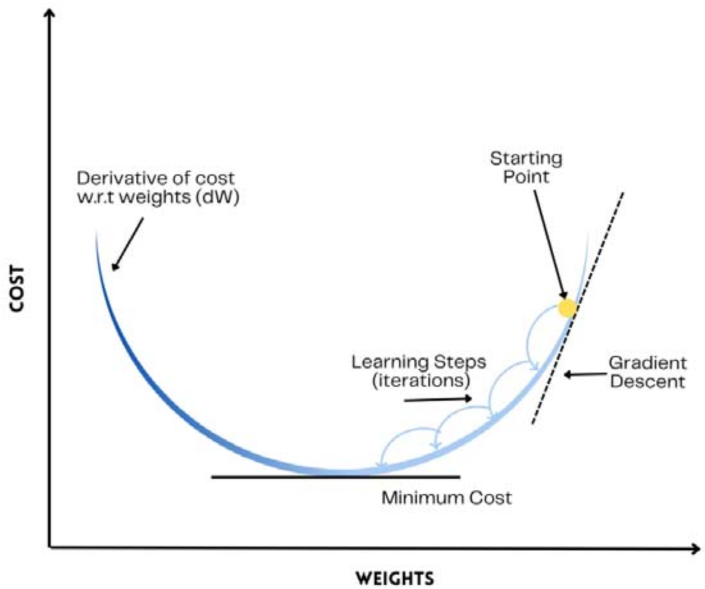
- Cost function: $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, \hat{y}^{(i)}\right)$

$$= -\frac{1}{m}\sum_{i=1}^{m} [y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

# Basics of Neural Network Programming

## Gradient Descent

# Gradient Descent




- 1) $\alpha > 0$

$$\text{Repeat}\{$$
$$w = w - \alpha \underbrace{\frac{dJ(w)}{d(w)}}_{dw}$$
$$\}\text{until convergence}$$
$$w = w - \alpha dw$$
$$w^* = (\underbrace{x^T x}_{n^3})^{-1} x^T y$$
$$y = (x - 1)^2$$
$$\frac{dy}{dx} = 2(x - 1) = 0$$
$$x = 1$$

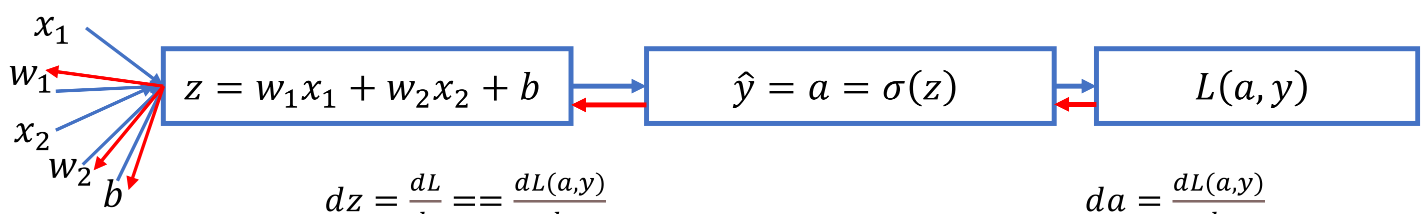
- $z = w^T x + b$
- $\hat{y} = a = \sigma(z)$
- $L(a, y) = -(y \log a + (1 - y) \log(1 - a))$

https://www.askpython.com/

# Gradient Descent

## Computational Graph

$$da = \frac{dL}{da} = \frac{-y}{a} + \frac{1-y}{1-a} = \frac{a-y}{a(1-a)}$$




$$dz = \frac{dL}{dz} == \frac{dL(a,y)}{dz}$$

$$da = \frac{dL(a,y)}{da}$$

$$dz = a - y_{a(1-a)}$$

$$dz = \frac{dL}{da} \times \frac{\overbrace{da}}{dz} = \frac{dL}{dz}$$

$$a = \sigma(z)$$

$$\frac{da}{dz} = \acute{\sigma}(z) = \underbrace{\sigma(z)}_{a}\,\underbrace{(1 - \sigma(z))}_{1-a}$$

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

$$\frac{\mathrm{d}L}{\mathrm{d}a} = \mathrm{d}a = \frac{a-y}{a(1-a)} \times a(1-a) = a - y$$

$$dw_1 = \frac{dL}{dw_1} = \underbrace{\frac{dL}{dz}}_{dz} \times \underbrace{\frac{dz}{dw_1}}_{x_1} = x_1\, dz$$

$$dw_2 = x_2\, dz$$

$$db = \frac{dL}{db} = \frac{dL}{dz} \times \frac{\overbrace{dz}^{1}}{db} = dz$$

# Gradient Descent

- $\left.\begin{cases} w_1 = w_1 - \alpha\, dw_1 \\ w_2 = w_2 - \alpha\, dw_2 \\ b = b - \alpha\, db \end{cases}\right]$ بروزرسانی همزمان

- $\begin{cases} w_1 temp = w_1 - \alpha\, dw_1 \\ w_2 temp = w_2 - \alpha\, dw_2 \\ btemp = b - \alpha\, db \end{cases}$

- $\begin{cases} w_1 = w_1 temp \\ w_2 = w_2 temp \\ b = btemp \end{cases}$

- $J(w,b) = \frac{1}{\mathrm{m}} \sum_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{L}\left(a^{(\mathrm{i})}, \mathrm{y}^{(\mathrm{i})}\right)$

- $a^{(i)} = \hat{y}^{(i)} = \sigma\left(z^{(i)}\right) = \sigma\left(wx^{(i)} + b\right)$

- $dw_1^{(i)} \qquad dw_2^{(i)} \qquad db^{(i)}$

- $\underbrace{\mathrm{d}J(w,b)}_{dw_1} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i=1}}^{\mathrm{m}} \underbrace{\frac{d\mathrm{L}\left(a^{(\mathrm{i})},\mathrm{y}^{(\mathrm{i})}\right)}{dw_1}}_{dw_1^{(i)}}$

# Logistic regression on $m$ examples

$J = 0$; $dw_1 = 0$; $dw_2 = 0$; $db = 0$;

$w_1 \leftarrow \text{random}$ $w_2 \leftarrow \text{random}$ $b \leftarrow \text{random}$

***Repeat{***

*For i=1 to m*

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= [y^{(i)} Log\, a^{(i)} + (1 - y^{(i)}) Log(1 - a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)}\, dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)}\, dz^{(i)}$

$db \mathrel{+}= dz^{(i)}$

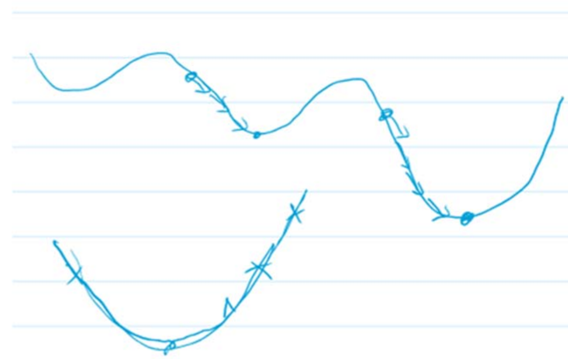
$J /= m$; $dw_1 /= m$; $dw_2 /= m$;

$w_1 = w_1 - \alpha\, dw_1$

$w_2 = w_2 - \alpha\, dw_2$ $d\theta = \begin{bmatrix} dw_1 \\ dw_2 \\ db \end{bmatrix}$ $\theta^t = \begin{bmatrix} w_1^t \\ w_2^t \\ b^t \end{bmatrix}$

$b = b - \alpha\, db$

***} until convergence*** $\|d\theta\| \le \varepsilon = 10^{-4}$ $\theta^{t+1} = \begin{bmatrix} w_1^{t+1} \\ w_2^{t+1} \\ b^{t+1} \end{bmatrix}$ $\|\theta^{t+1} - \theta^t\|_2 \le \varepsilon$



# Basics of Neural Network Programming

## Vectorizing Logistic Regression's Gradient Computation

# Vectorizing Logistic Regression

- *Z=0;*

  *For i in range(n_x)*

  $$z \mathrel{+}= w[i] * x[i]$$

  $$z \mathrel{+}= b$$

  روش اول

- *Z=0;*

  $z = np.dot(w, x) + b$

  روش دوم

  **SIMD**

  **GPU**

# Vectorizing Logistic Regression

- $z^{(1)} = w^T x^{(1)} + b$ $\quad$ $z^{(2)} = w^T x^{(2)} + b$ $\quad$ $z^{(3)} = w^T x^{(3)} + b$
  $a^{(1)} = \sigma\left(z^{(1)}\right)$ $\quad$ $a^{(2)} = \sigma\left(z^{(2)}\right)$ $\quad$ $a^{(3)} = \sigma\left(z^{(3)}\right)$

- $\mathrm{X} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots\dots\dots & x^{(\mathrm{m})} \\ | & | & & | \end{bmatrix} \in R^{n_x \times m}$ $\quad$ $\underbrace{[w_1 \dots w_{n_x}]}_{W^T} \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots\dots\dots & x^{(\mathrm{m})} \\ | & | & & | \end{bmatrix}$

- $Z = \left[ \underbrace{z^{(1)}}_{w^T x^{(1)}+b} \quad \underbrace{z^{(2)}}_{w^T x^{(2)}+b} \quad \underbrace{z^{(3)}}_{w^T x^{(3)}+b} \quad \dots \quad \underbrace{z^{(m)}}_{w^T x^{(m)}+b} \right] = W^T X + [b\ b \cdots b]_{1\times m}$

- $Z = \underbrace{np \cdot dot(W \cdot T, X)}_{1_{\times} m} + \underbrace{b}_{(1,1)}$ *"broadcasting"* $[b, b \dots, b]_{1\ \times\ m}$

# Vectorizing Logistic Regression

- $A = [a^{(1)}, a^{(2)}, \dots, a^{(m)}] = \sigma(\underbrace{Z}_{1\times m})$
- $dz^{(1)} = a^{(1)} - y^{(1)}$ $\quad dz^{(2)} = a^{(2)} - y^{(2)}$
- $dZ = [dz^{(1)}\ dz^{(2)} \dots dz^{(m)}]_{1 \times m}$
- $A = [a^{(1)}\ a^{(2)} \dots a^{(m)}]$ $\quad Y = [y^{(1)}\ y^{(2)} \dots y^{(m)}]$
- $dZ = A - Y = [a^{(1)} - y^{(1)}\quad a^{(2)} - y^{(2)} \ \dots\ a^{(m)} - y^{(m)}]$
  - $dw=0$ $\quad db=0$
- $\begin{bmatrix} dw_1 \\ dw_2 \\ dw_3 \\ \vdots \\ dw_{nx} \end{bmatrix}_{nx \times 1}$

$$\begin{cases} dw += \underbrace{x^1}_{\text{بردار}} \overbrace{dz^1}^{\overset{\text{عدد}}{a^1-y^1}} \\ dw += x^2 dz^2 \\ dw += xmdzm \end{cases}$$

$dw/=m$

$$\begin{cases} db += \overbrace{dz1}^{\overset{\text{عدد}}{a1-y1}} \\ db += dz2 \\ db += dzm \end{cases}$$

$db/=m$

$$db = \frac{1}{m}\sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} np.sum(dZ$$

$$dz = [dz^1\ \ dz^2 \dots dz^m]_{1 \times m}$$

$$\boxed{dW = \frac{1}{m} X\, dZ^T} = \frac{1}{m}\begin{bmatrix} x^{(1)} x^{(2)} & \dots\dots\dots & x^{(\mathrm{m})} \end{bmatrix}_{nx \times m} \begin{bmatrix} dz^1 \\ dz^2 \\ dz^3 \\ \vdots \\ \vdots \\ \vdots \\ dz^m \end{bmatrix}_{m \times 1}$$

$$dw = \frac{1}{m}\left[x^{(1)} dz^1 + x^{(2)} dz^2 + \dots + x^{(m)} dz^m\right]$$

# Vectorizing Logistic Regression

- $w_1, w_2, b \leftarrow random$

*For iter in range(1000)*

$$\text{1 epoch}\left\{\begin{array}{l} Z = W^T x + b = np \cdot dot(W.T, X) + b \\ A = \sigma(Z) \\ dZ = A - Y \\ dW = \frac{1}{m} X\, dZ^T \\ db = \frac{1}{m} np.sum(dZ) \\ w = w - \alpha\, dw \\ b = b - \alpha\, db \end{array}\right.$$

$a = np \cdot random \cdot \mathrm{rand}n(5,1)$ $\quad a.\mathrm{sh}ape = (5,1)$

# Logistic Regression Cost function

- $\hat{y} = \sigma(w^T x + b)$ $\qquad 0 < \sigma(z) = \frac{1}{1+e^{-z}} < 1$
- $\hat{y} = p(y = 1|x)$

$$\left.\begin{array}{l} \bullet\ if \quad y = 1 \quad : \quad p(y|x) = \hat{y} \\ \bullet\ if \quad y = 0 \quad : \quad p(y|x) = 1 - \hat{y} \end{array}\right\} \boldsymbol{p(y|x)}$$

- $\boxed{p(y|x) = \hat{y}^y + (1 - \hat{y})^{1-y}}$ *distribution? Bernoulli*
- $if \quad y = 1 \quad : \quad p(y|x) = \hat{y}$
- $if \quad y = 0 \quad : \quad p(y|x) = 1 - \hat{y}$
- $\mathrm{logp(y|x)} = \log[\hat{y}^y + (1 - \hat{y})^{1-y}] = y \log \hat{y} + (1 - y) \log(1 - \hat{y})$
  $= -L(\hat{y}, y)$ *Max Likelihood*

# Logistic Regression Cost function

- $\log P(\textit{ labels in trainingset}) = \log \prod_{i=1}^{m} P\left(y^{i} \mid x^{i}\right)$
- $\log P(\cdots) = \sum_{i=1}^{m} \log P\left(y^{(i)} \mid x^{(i)}\right) = -\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$

- Cost function $\underbrace{J(w,b)}_{\boldsymbol{minimize}} = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$

# Neural Networks