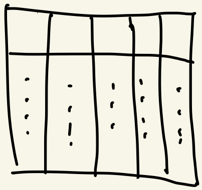
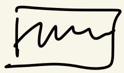
Week 2 Notes

stanard NN   Image CNN   Audio, text RNN

struture data vs unstruture

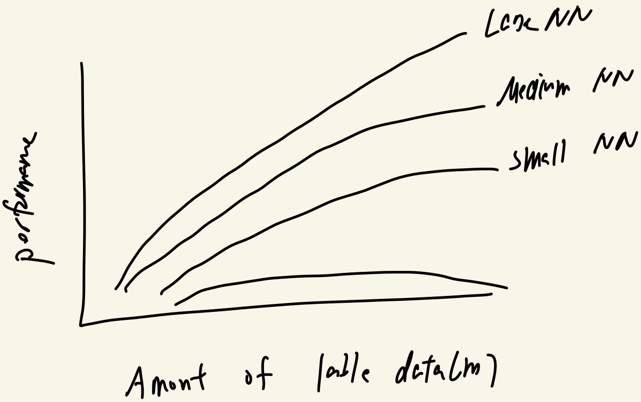




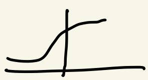
Audio

image

text

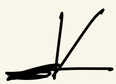


sismal

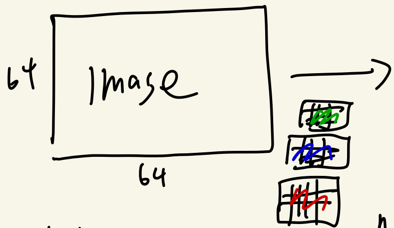


→

ReLu



Binary Classification



1 cat
or
0 no cat

$n = n_x = 64 \times 64 \times 3$

Notation

(x, y) $\quad x \in \mathbb{R}^{n_x}$ $\quad y \in \{0, 1\}$

$x = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$

m training sample $\{(x^1, y^1)(x^2, y^2) \cdots (x^m, y^m)\}$

m_{train} = # of training sample

m_{test} = # of test sample

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^m \\ | & | & & | \end{bmatrix}}_{m} \Big\updownarrow n_x$$

$$Y = [y^1 \; y^2 \cdots y^m]$$

$Y.shape = (1, m)$

$X.shape = (n_x, m)$

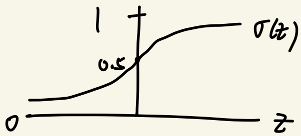
Logistic Regression

- given X, want $\hat{y} = P(y=1 | x)$

  $x \in \mathbb{R}^{n_x}$ $\quad 0 \le \hat{y} \le 1$ $\quad$ W = weight matrix

  parameter $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$ $\quad$ b = bias vector

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



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

If z large $\quad \sigma \approx 1$

z small $\quad \sigma \approx 0$

Loss (Eror) function

$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ $\quad$ how good $\hat{y}$ predict y

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

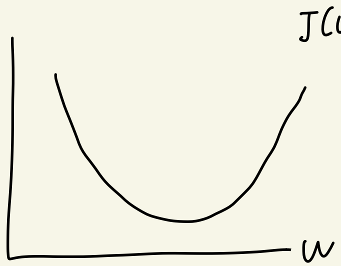
Cost function vs Loss function

entire sample $\quad$ one sample

Cost function

$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Gradient Descent



Repeat {

$$w := w - \overbrace{\alpha}^{\text{learning rate}} \underbrace{\frac{dJ}{dw}}_{\text{"dw"}}$$

code: $w := w - \alpha\, dw$

$$w := w - \alpha \frac{\partial J(w,b)}{\partial w} \quad dw$$

$$b := b - \alpha \frac{\partial J(w,b)}{\partial b} \quad db$$

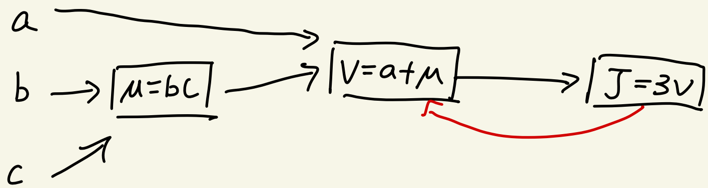
Computation Graph

$J(a,b,c) = 3(a+bc)$

$u = bc$

$V = a + u$

$J = 3V$



deritive = 1 step backward

$$\frac{dJ}{da} = \frac{dJ}{dV}\frac{dV}{da} = 3 \qquad a \to V \to J$$

$$\boxed{\frac{d\ \text{Final output variable}}{d\ \text{var}}} \to \text{code "dvar"}$$

Logistic Regression

$$z = w^T x + b$$

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

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

$$x_1, w_1, x_2, w_2, b \rightarrow \boxed{z = w_1 x_1 + w_2 x_2 + b} \rightarrow \boxed{\hat{y} = a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$$

$$dz = \frac{d\mathcal{L}}{dz} = a - y = \frac{d\mathcal{L}}{da}\frac{da}{dz}$$

$$\frac{da}{dz} \rightarrow a(1-a)$$

$$\text{"}da\text{"} = \frac{d\mathcal{L}}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\frac{d\mathcal{L}}{dw_1} = \text{"}dw_1\text{"} = x_1 dz$$

$$dw_2 = x_2 dz$$

$$db = dz$$

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a^i, y^i) \qquad (x^i, y^i)$$

$$a^i = y^i = \sigma(z^i) = \sigma(w^T x^i + b) \qquad dw_1^i \quad dw_2^i \quad db^i$$

$$\frac{\partial J}{\partial w_1} J(w, b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1}\mathcal{L}(a^i, y^i)}_{dw_1^i - (x^i, y^i)}$$

Logistic Regression Example

$J=0 \quad dw_1=0 \quad dw_2=0 \quad db=0$

$dw = np.zeros((n_x, 1))$

For $i=1$ to m

$z^i = w^T x^i + b$

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

$J += -[y^i \log a^i + (1-y^i)\log(1-a^i)]$

$dz^i = a^i - y^i$

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

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

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

$n=2$

$dw \mathrel{+}= x^i dz^i$

J/m

$\underbrace{dw_1/m \quad dw_2/m}_{dw/m} \quad db/m$

$dw_1 = \frac{\partial J}{\partial w_1}$

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$

Vectorization

↳ get rid of for loop

Vectorization

$z = w^T x + b$

$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix} \quad x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$

non vectorized

```
z=0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

vectorized

$z = \underbrace{np.dot(w, x)}_{w^T x} + b$

Avoid explicit for-loop

Vectorized logistic Regression

$$z^i = w^T x^i + b$$
$$a^i = \sigma(z^i)$$

$$X = \begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^m \\ | & | & & | \end{bmatrix} \qquad \frac{(n_x, m)}{\mathbb{R}^{n_x \times m}}$$

$$Z = [z^1 \ z^2 \cdots z^m] = w^T X + \underbrace{[b \ b \ b \cdots b]}_{1 \times m}$$

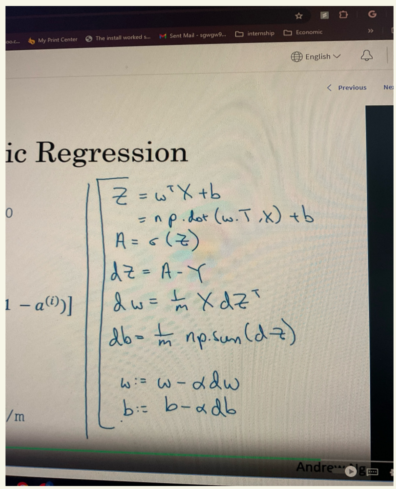
$$z = np.dot(w.T, X) + b \longrightarrow [1,1]$$

"Broadcasting"

$$A = [a^1 \ a^2 \cdots a^m] = \sigma(z)$$

$$db = \frac{1}{m} np.sum(dz)$$
$$dw = \frac{1}{m} X dz^T$$

Vectorized



a = np. random. randn(5,1) a.shape = (5,1)

or

" " " (1,5) a.shape = (1,5)

assert(a.shape == (5,1))

or

a = a. reshape((5,1))