DATE

week 2: Basics of NN programming.

→ Binary Classification

- when processing the training set we process the trainy set without necessarily using the for statement

- logistic regression is an algorithm for binary class.

converted to → [image] → feature vector (x) → 1 (Cat) v/s 0 (non Cat) → y.

- Notations:
  - Single training sample is denoted (x, y) where $x \in R^{n_x}$
    x is a feature vector in dimension n_x & y is 0 or 1
  - m → train examples.
  - $m = m_{train}$ m_{test} = # test examples
  - to put all of training examples in compact notation we define matrix:

$$X = \begin{bmatrix} | & | & | & | \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & | \end{bmatrix} \quad n_x \uparrow \quad \leftarrow m \rightarrow$$

This matrix takes $x^{(1)} \ldots x^{(m)}$ as stacks them to columns.

$$y = [y^{(1)} \quad \cdots \quad y^{(m)}]$$

$$y \in R^{1 \times m}$$

PAGE
DATE

→ logistic Regression

- This algorithm we use it when the output label y is 0 or 1.

- given X, want $\hat{y} = P(y=1|X)$ $0 \leq \hat{y} \leq 1$
  → means the prob. that y is 1 given the input feature X

ex: parameters $w \in R^{n_x}$, $b \in R$.

output $\hat{y} = \underset{\text{sigmoid}}{\sigma}(\underbrace{w^T x + b}_{z})$.

$\sigma(z) = \frac{1}{1+e^{-z}}$ → if z is large Then $\sigma(z)$ will be near 0 then it $\frac{1}{1+0} = 1$

if z is +ve $\simeq \frac{1}{1+\text{num}} = 0$

- to train the parameters w, b we need to define cost fn.

→ logistic regression Cost function

ex: MSE
- loss function: measures error for one training sample (x, y)
- Cost function: Avg. error for all training samples

- loss or error fun: $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ → measures how good our output $\hat{y}$ is while true output label y is

• in logistic regression we write loss fun:

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

& this loss fun. is called binary cross entropy.

∴ if $y=1$ $L(\hat{y}, y) = -\log \hat{y}$, so we wan $\hat{y}$ large

// $y=0$ $L(\hat{y}, y) = -\log(1-\hat{y})$, so $\log(1-\hat{y})$ large so $\hat{y}$ small

• Cost fun : $J(w,b) = \frac{1}{m} \sum^{m} L(\hat{y}^{(i)}, y^{(i)})$

→ gradien Descent

• recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ } logistic reg alg.

• Cost fun : $J(w,b) = \frac{1}{m} \sum^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}$

• loss fun mesures how well the alg. predicts y for each trainy sample compared to the true value

• So we always try to find w,b that minimize $J(w,b)$

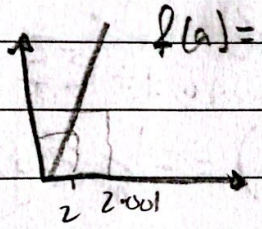
• gradient descent is optimization algorithm or tech. use to minimize the cost fun. by iteratively adjust the model parameter so we improve model performance

$$\Theta = \Theta - \alpha \cdot \nabla J(\Theta)$$

Θ → model parmetos, α learning rate, $\nabla J(\Theta)$ gradient of $J(w,b)$ cost func.

• when Calling a model we don't implement Cost of loss fun.
① we create model ② it has its own Built in cost fun.
ex: binary cross entropy for logistic reg. & its optimization algorithm ex: gradient descent)
③ when we call .fit(). the model Calculate loss & cost fun.
• update the model parameters
• iterate this process until
Convergence → when updates results no change in cost fun.
④ prediction

→ Derivatives.

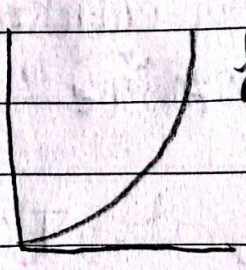


a = 2 f(a) = 6
a = 2.001 f(a) = 6.003

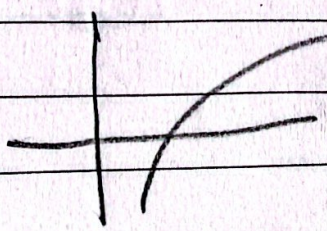
So slope (derivative) of f(a) at a = 2 is 3

• one propertie of derivative whatever you take the slope of the fun. 3 whether a = 2 or 5 the slope = 3 where the inc. is 0.001

→ more Derivatives example.



$\frac{d}{da} a^2 = 2a$: mean for every a is is pushed by 0.001 it will inc. $2a * 0.001$

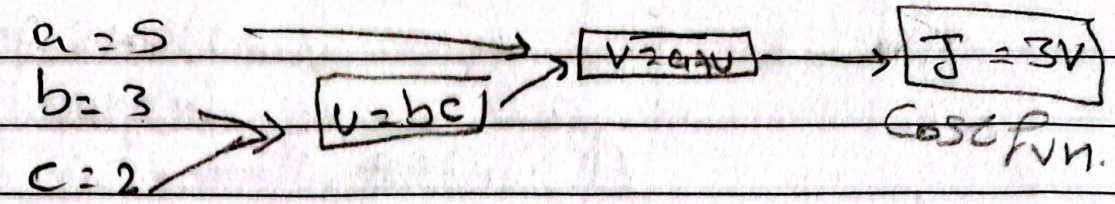


$f(a) = \log(a) = \frac{1}{a}$

→ Computation graph.

$J(a,b,c) = 3(a+bc) = 3(5+3\times2) = 33$

$u = bc$, $v = a+u$, $J = 3v$



→ Derivatives with Computation graphs ☆

backpropagation

→ logistic regression Gradient descent ☆

. we will talk about how to compute derivatives to implement gradient descent

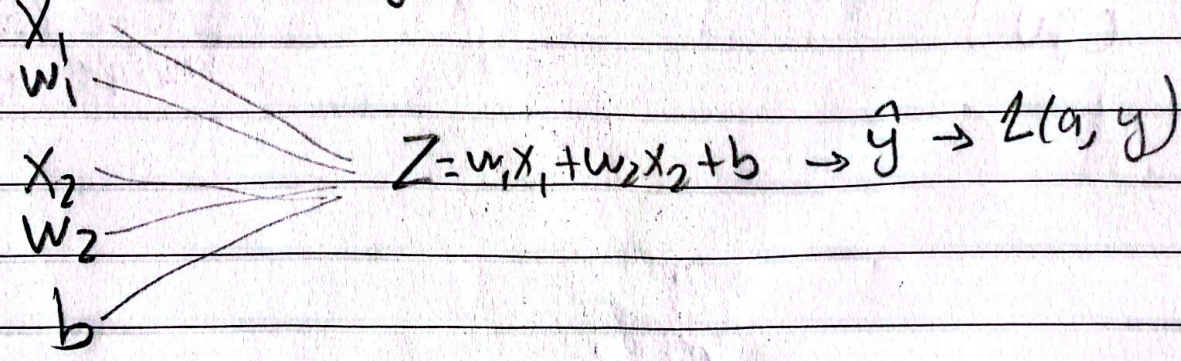
. logistic regression recap

$z = w^T x + b$

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

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

. lets say we have two features : to compute z we put w_1, w_2 , we modify w & b to reduce this loss



→ Vectorization.

- Vectorization is the art of getting rid of loops in your code & this causes speed up your code.
- What is vect. ex: $z = w^t x + b$ $w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$

to compute $w^t x$ if we had a non vecorized implem

is (nonvectorized)	(vectorized)
z = 0 for i in range (n-x) zt = w[i] * x[i] z += b	z = np.dot(w,x) + b

- GPU / CPU] both parallelization instruction SIMD
  - single instruction multiple data

PAGE
DATE

More vectorization examples.

$$V = \begin{bmatrix} v_1 \\ v_2 \\ v_n \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

```
u = np.zeros((n,1))
for i in range(n):
    u[i] = math.exp(v[i])
```

u = np.exp(v)

*vectorizing logistic regression. ✓

broad casting: is another tech. That make py code run faster

py & numpy vectors: a = np.random.randn(5)

a.shape = (5,) → rank 1 array

→ assert()