SC201 Lecture 10

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{nf} \end{bmatrix}_{nx1} \qquad X = \begin{bmatrix} X_{11} & X_{21} & & X_{m1} \\ X_{12} & X_{22} & \cdots & X_{m2} \\ \vdots & \vdots & & \vdots \\ X_{1n} & X_{2n} & & X_{m3} \end{bmatrix}_{nxm} \qquad Y = [y_1, y_2 \ldots y_m]_{1xm}$$

W.T =

Fowardprop

K = W.T.dot (X) + b

H = 1/ (1+np.exp(-k))

L = -(Y*np.log(H)+(1-Y)*np.log(1-H))

$J = \frac{1}{m} * np.sum(L)$

Backprop $W = W - \alpha\frac{dJ}{dW}$ $b = b - \alpha\frac{dJ}{db}$

$\frac{dJ}{dW} =$

$\frac{dJ}{dH} =$

$\frac{dH}{dK} =$

$\frac{dK}{dW} =$

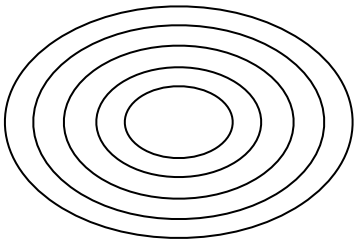
titanic_batch_gradient_descent.py

```
def main():
    # classifier = h.fit(X,Y)
    [                                    ]
    _______________ = W.T.dot(X)+b
    predictions = ____________________________________
    acc = ____________________________
    num_acc = ____________(acc)
    print('Acc:', num_acc/m)


def batch_gradient_descent():
    n, m = X.shape
    w = ____________________________________
    b = ____________________________________
    for epoch in range(NUM_epochs):
        K = W.T.dot(X)+b
        H =  1/ (1+np.exp(-k))
        L =  -(Y*np.log(H)+(1-Y)*np.log(1-H))
        J =  1/m * np.sum(L)                          } Fowardprop
        if epoch% 1000 == 0:
            print('Cost:', J)
        # W = W - alpha * dJ_dW
        W =  W - ALPHA * ( (1/m)* np.sum(X.dot((H-Y).T), axis=1, keepdims = True))
        # b = b - alpha * dJ_db
        b = b - ALPHA *  ( (1/m)* np.sum(H-Y))        } Backprop
    return W, b
```

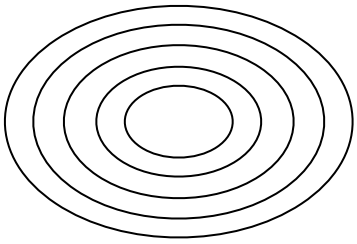
<SGD> Stochastic Gradient Descent

- Update weights on _____
- Easy to _____
- _____ updates
- W = _____



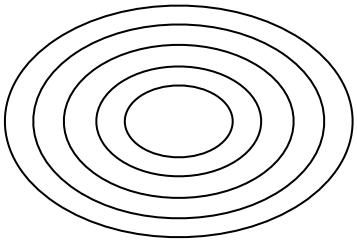
<MBGD> Mini-Batch Gradient Descent

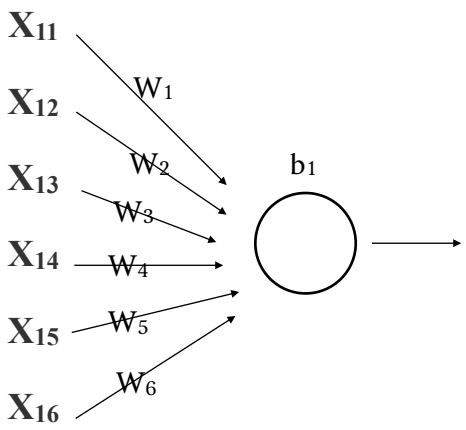
- Update weights on _____
- W = _____



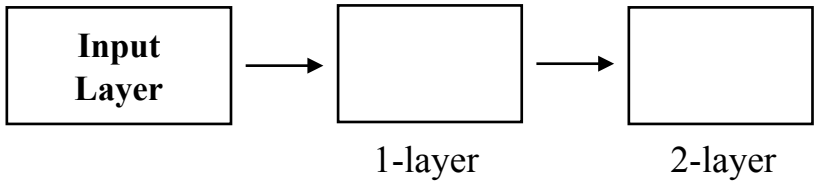
<BGD> Batch Gradient Descent

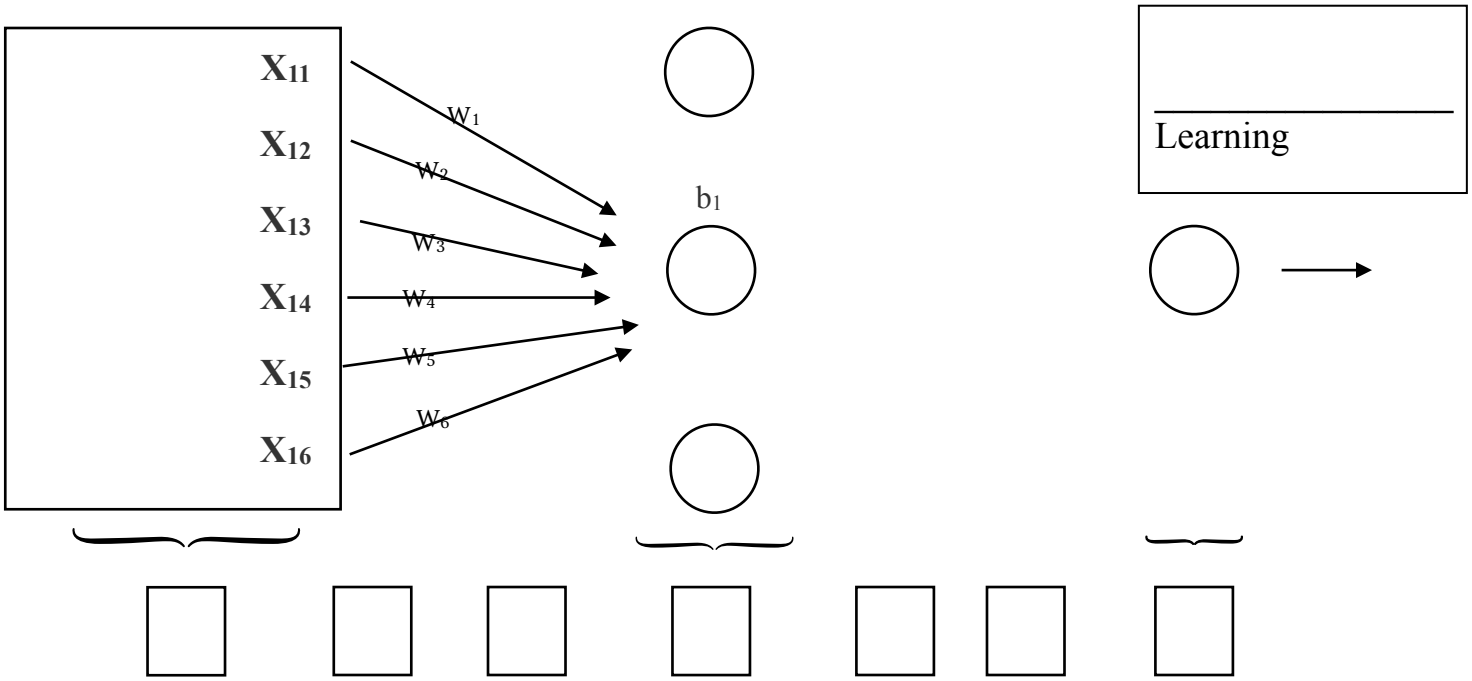
- Update weights on _____
- Never Overfit
- _____ updates
- W = _____





<2-layer Neural Network>





(,) (,)(,) (,) (,)(,) (,)

W1.shape ➠ (,)

B1.shape ➠ (,)

W2.shape ➠ (,)

B2.shape ➠ (,)

K1 = _____

K2 = _____

K2 = _____

K2 = _____

直接把 6 feature→1feature

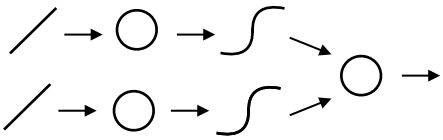
Still using _____

Activation Functions

- In order to create _____,

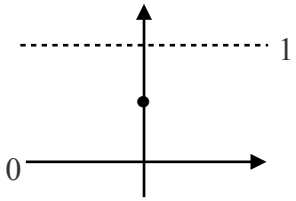
  we need _____ such that NN can learn

  _____ in data!



① Sigmoid

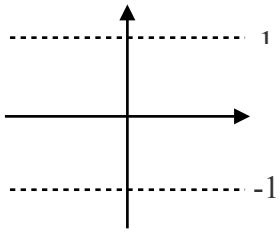
$$\frac{1}{1+e^{-k}}$$



- _____ gradient at high/low values
- _____
- _____

② tanh

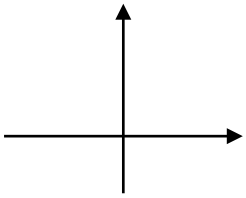
$$\frac{e^{k}-e^{-k}}{e^{k}+e^{-k}}$$



- _____ gradient
- _____

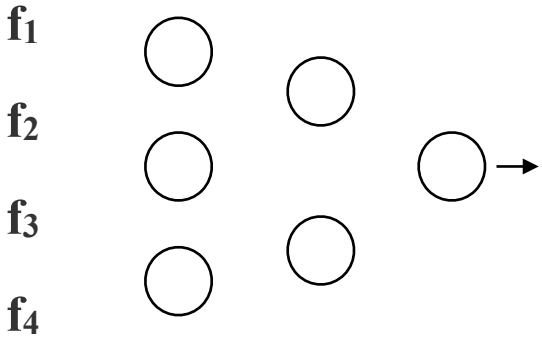
③ ReLU

$$max(0,X)$$



- No small gradient
- _____

<Fowardprop>



K1 = _____

A1 = _____

K2 = _____

A2 = _____

scores = _____

H = _____

L = _____

J = _____