NEURAL NETWORK FOR MNIST

THE CLASSIFICATION IS THAT WE HAVE SOME GIVEN TRAINING SET T WITH

$$T = \{ (\vec{x}_1, y_1), \ldots (\vec{x}_n, y_n) \}$$

SUCH THAT $x_i \in \mathbb{R}^N$ FOR SOME LARGE N AND $y \in \{0, \ldots, 9\}$. IN THIS CASE, OUR TRAINING SET IS THE MNIST TRAINING SET SO $\vec{x}_i \in \{0, 255\}^{784}$. OUR GOAL IS TO FIND A CLASSIFIER $f : X \to Y$ THAT

(1) ACHIEVES A HIGH ACCURACY (AT LEAST 90%) ON THE TRAINING SET

(2) GENERALIZES TO NEW EXAMPLES

IN THESE NOTES, WE DESCRIBE HOW TO FIND A CLASSIFIER BY DEFINING AN ENERGY / COST FN AND MINIMIZING THIS FN BY USING GRADIENT DESCENT. MOREOVER, OUR CLASSIFIER WILL BE A NEURAL NETWORK.

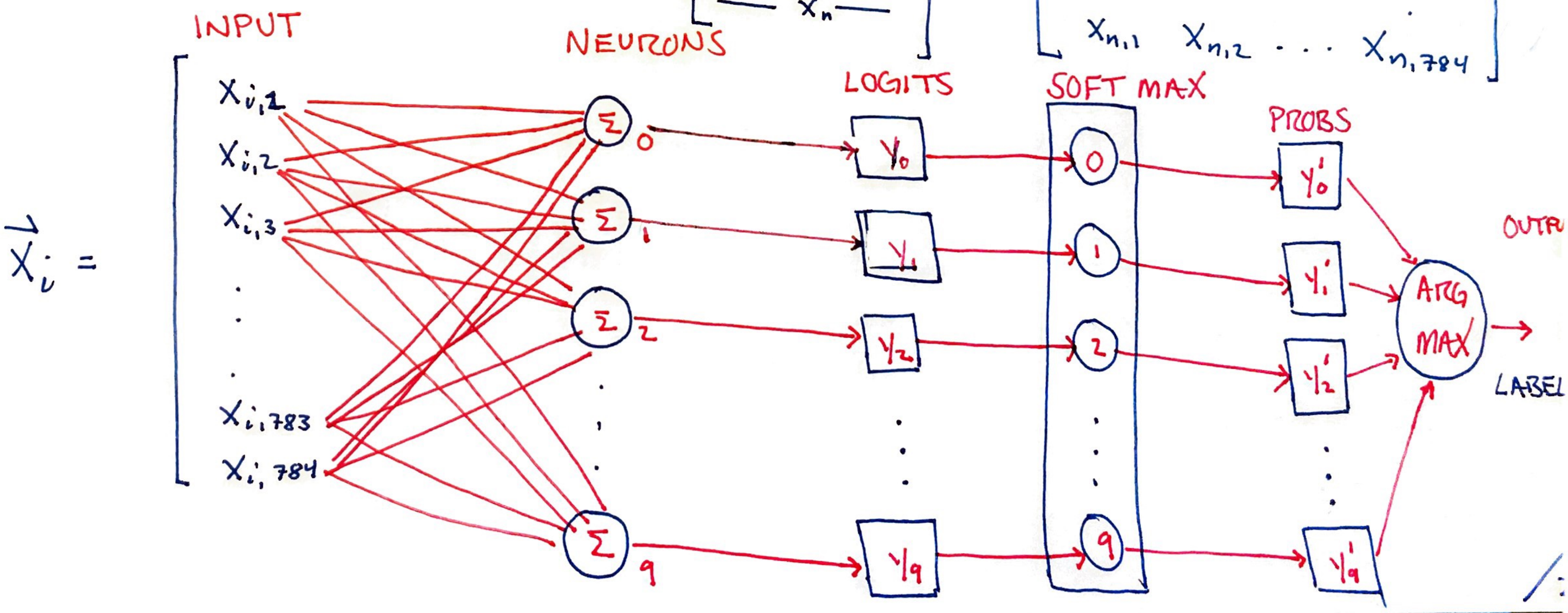
(1) NETWORK ARCHITECTURE

FOR $\forall$ EXAMPLE $\vec{x}_i$ IN OUR TRAINING SET, WE CAN INDEX EACH ELEMENT OF THIS VECTOR BY

$$\vec{x}_i = [\, x_{i,1} \;\; x_{i,2} \;\; x_{i,3} \; \ldots \; x_{i,784} \,]$$

<u>NOTE</u>: THE TRAINING EXAMPLES IN THE CODE ARE STORED LIKE THIS

$$\text{train\_im} = \begin{bmatrix} \text{---} \; \vec{x}_1 \; \text{---} \\ \text{---} \; \vec{x}_2 \; \text{---} \\ \vdots \\ \text{---} \; \vec{x}_n \; \text{---} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,784} \\ \vdots & & & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,784} \end{bmatrix}$$



(2) ENERGY FUNCTION

LET $E : \mathbb{R}^{10 \times 784} \to \mathbb{R}^{+}$ BE AN ENERGY FUNCTION DEFINED BY

$$E(W) = -\sum_{i=1}^{n} \sum_{j=1}^{10} \mathbb{1}_j(y_i) \ln p_j(\vec{x}_i) \qquad (1)$$

WITH $\mathbb{1}_j(y_i) = \begin{cases} 1, & \text{IF } y_i = j \\ 0, & \text{ELSE} \end{cases}$ AND $p_j(\vec{x}_i) = \dfrac{\exp(\langle \vec{x}_i, \vec{w}_j \rangle + b_j)}{\sum_{\ell=0}^{9} \exp(\langle \vec{x}_i, \vec{w}_\ell \rangle + b_\ell)}$

SUCH THAT THE WGT VECTOR $\vec{w}_j$ AND b_j CORRESPOND TO THE j^{th} NEURON IN THE NETWORK. WE CAN APPROXIMATE THE MINIMUM OF THIS FUNCTION BY USING GRADIENT DESCENT. THIS MEANS THAT WE NEED TO COMPUTE THE GRADIENT OF OUR ENERGY FN IN (1).

(3) COMPUTE PARTIAL DERIVATIVES

IN THIS CASE, OUR WEIGHT VECTOR IS A MATRIX, WHERE EACH ROW IS THE SET OF WGTS CORRESPONDING TO A NEURON IN OUR NETWORK.

$$
W = \begin{bmatrix} \text{---} & \vec{w}_0 & \text{---} \\ \text{---} & \vec{w}_1 & \text{---} \\ & \vdots & \\ \text{---} & \vec{w}_9 & \text{---} \end{bmatrix} = \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,784} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,784} \\ \vdots & \vdots & & \vdots \\ w_{9,1} & w_{9,2} & \cdots & w_{9,784} \end{bmatrix}
$$

THIS MEANS THAT ∇E IS A MATRIX OF PARTIAL DERVITIVES

$$
\nabla E = \begin{bmatrix} \frac{\partial E}{\partial w_{0,1}} & \frac{\partial E}{\partial w_{0,2}} & \cdots & \frac{\partial E}{\partial w_{0,784}} \\ \frac{\partial E}{\partial w_{1,1}} & \frac{\partial E}{\partial w_{1,2}} & \cdots & \frac{\partial E}{\partial w_{0,784}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{9,1}} & \frac{\partial E}{\partial w_{9,2}} & \cdots & \frac{\partial E}{\partial w_{9,784}} \end{bmatrix}
$$

FOR $\forall$ ENTRY $\omega_{j,k}$ IN THE MATRIX W, LET'S COMPUTE THE CORRESPONDING PARTIAL DERIVATIVE

$$\frac{\partial E}{\partial \omega_{j,k}}(\vec{W}) = \frac{\partial}{\partial \omega_{j,k}}\left(-\sum_{i=1}^{n}\sum_{j=0}^{9} \mathbb{1}_j(y_i)\ln p_j(\vec{x}_i)\right)$$

$$= \frac{\partial}{\partial \omega_{j,k}}\left(-\sum_{i=1}^{n}\sum_{j=0}^{9} \mathbb{1}_j(y_i)\ln \frac{\exp(\langle \vec{x}_i, \vec{w}_j\rangle + b_j)}{\sum_{\ell=0}^{9}\exp(\langle \vec{x}_i, \vec{w}_\ell\rangle + b_\ell)}\right)$$

$$= \frac{\partial}{\partial \omega_{j,k}}\left(-\sum_{i=1}^{n}\sum_{j=0}^{9} \mathbb{1}_j(y_i)\left(\ln \exp(\langle \vec{x}_i, \vec{w}_j\rangle + b_j) - \ln \sum_{\ell=0}^{9}\exp(\langle \vec{x}_i, \vec{w}_\ell\rangle + b_\ell)\right)\right)$$

$$= \frac{\partial}{\partial \omega_{j,k}}\left(-\sum_{i=1}^{n}\sum_{j=0}^{9} \mathbb{1}_j(y_i)\left(\langle \vec{x}_i, \vec{w}_j\rangle + b_j - \ln \sum_{\ell=0}^{9}\exp(\langle \vec{x}_i, \vec{w}_\ell\rangle + b_\ell)\right)\right)$$

$$= -\sum_{i=1}^{n} \mathbb{1}_j(y_i)\left(x_{i,k} - x_{i,k}\frac{\exp(\langle \vec{x}_i, \vec{w}_j\rangle + b_j)}{\sum_{\ell=0}^{9}\exp(\langle \vec{x}_i, \vec{w}_\ell\rangle + b_\ell)}\right)$$

$$= -\sum_{i=1}^{n} \mathbb{1}_j(y_i)\, x_{i,k} \left(1 - p_j(\vec{x}_i)\right)$$

$$\Rightarrow \quad \frac{\partial E}{\partial w_{j,k}}(\vec{w}) = \begin{cases} x_{i,k}\left(1 - p_j(\vec{x}_i)\right), & \text{IF} \quad y_i = j \\ x_{i,k}\, p_j(\vec{x}_i), & \text{IF} \quad y_i \neq j \end{cases}$$

NOW WE NEED TO TAKE PARTIAL DERIVATIVE WRT b_j, BUT ~~THIS~~ A I'LL JUST GIVE IT TO YOU WITHOUT GOING THROUGH ALL THE WORK.

$$\frac{\partial E}{\partial b_j}(\vec{w}) = \begin{cases} -\left(1 - p_j(\vec{x}_i)\right), & \text{IF} \quad y_i = j \\ -p_j(\vec{x}_i), & \text{ELSE} \end{cases}$$

NOW WE HAVE THE KEY INGREDIENT TO DO GRADIENT DESCENT. SO LET'S GO THROUGH THE PSEUDO-CODE.

(4) PSEUDO CODE

```
# INITIALIZATIONS
    - STEP_SIZE   }
    - WGT_MAT     }  FOR GRADIENT
    - BIAS        }    DESCENT
    - MAX_ITER    }
    - EPSILON     }  USED TO DETERMINE
    - ITER        }  WHEN TO TERMINATE
                      GRADIENT DESCENT

# TRAIN NEURAL NETWORK

While iter < max_iter:
    num_MISTAKES = 0
  (1) for i in range(len(train_lbs)):
        x_i ~ EXAMPLE IN TRAINING SET
      # FORWARD PASS
        LOGITS = WGT_MAT * x_i + BIAS

        NORM_FACTOR = sum(exp(LOGITS))
        PROBS = exp(LOGITS) / NORM_FACTOR
        nn_LABEL = argmax(PROBS)
```

```
IF    nn_LABEL ≠ y_i
        num_MISTAKES += 1
 COST[i] = COST[i] + -log( PROBS[y_i] )
# BACKWARD PASS

        for j in range(9)
            if j == y_i :
                WGT_MAT[j, 0:783] = WGT_MAT[j, 0:783]
                 + UPDATE BIAS        - STEP_SIZE · x_i (1-PROB[j])
            else :

                WGT_MAT[j, 0:783] = WGT_MAT[0,783] -
                                      STEP_SIZE · x_i PROB[x_i]
                 + UPDATE BIAS
```

* THEN WHEN YOU EXIT FOR LOOP (i)
CHECK IF YOU NEED TO DECREASE STEP
SIZE AND CHECK THE STOPPING CRITERIA.