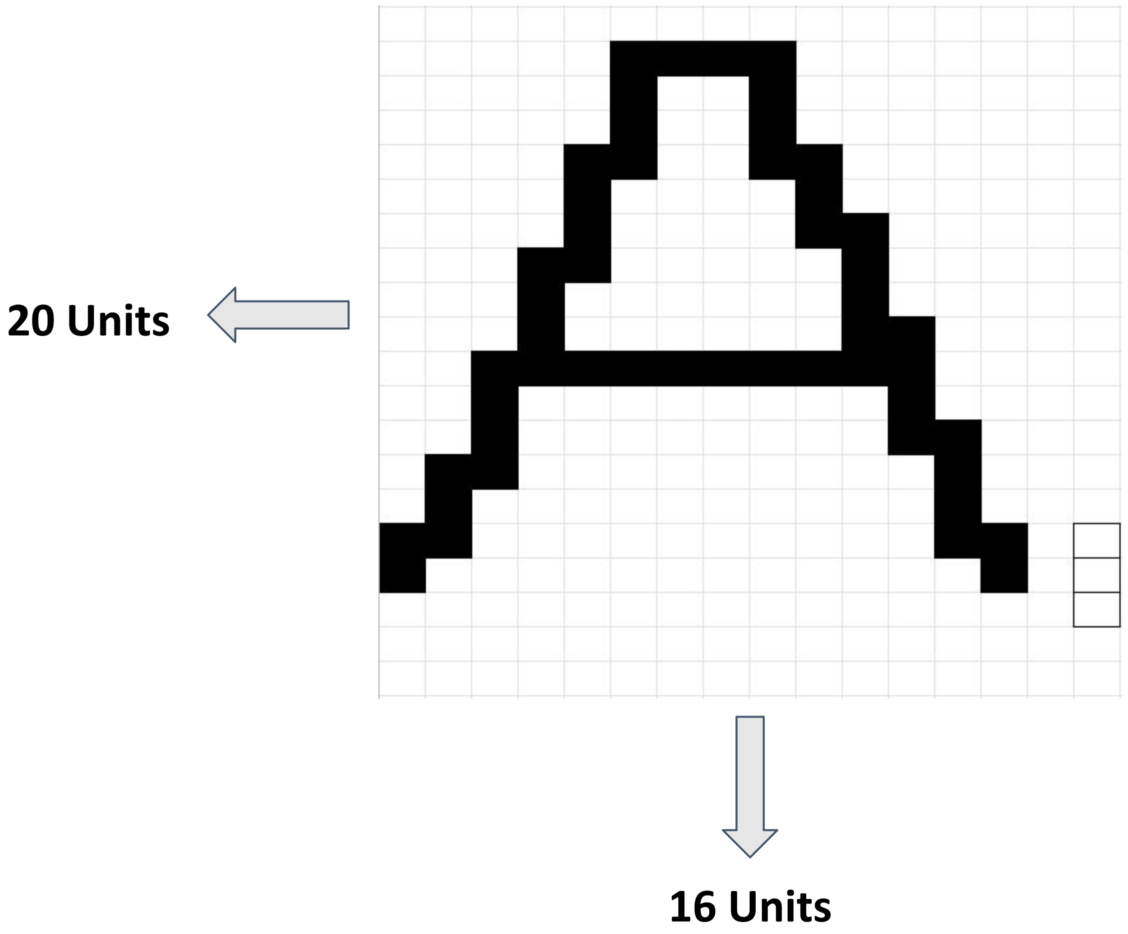
QMBU 450 FINAL PRESENTATION

Kaan Yıldırım
Eren Çokyaşar

Description of Problem



- Letters can be A,B,C,D or E.
- We have got 39 samples for each letter as csv files.
- Pixel datas are given as 320 length vectors.
- If value is 1, pixel is black. If value is 0, pixel is white.

Before the Solution..

- 25 sample for training, 14 for testing (for each letter)

- Contribution Matrixes will demonstrate accuracy.

	A	B	C	D	E
A	25				
B		25			
C			24	1	1
D			1	24	1
E					23

Solution 1: Naive Bayes Method

- Probability (j,i) Values → If the image is letter "i", the probability of j'th pixel is equal to "1".

- For each image

  For each pixel

  ScoreA += pixel_value x log(Probability (pixel number,1)) + (1-pixel_value) x log(1-Probability (pixel number,1))

  ScoreB += pixel_value x log(Probability (pixel number,2)) + (1-pixel_value) x log(1-Probability (pixel number,2))

  .

  .

- The letter with the maximum score is the prediction

Solution 2: Discrimination by Regression

- y = w1*x1 + w2*x2 + w3*x3 + ... + w320*x320 + w0 (For each class)
- Use sigmoid activation function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
$$= \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

How can we find W-values?

$$\text{output} = W \cdot X$$

?

Derivation of Error With Respecting to W.

Error = (Y_Truth - Y_Predicted)

$$\frac{\partial Error}{\partial w} = \frac{\partial\left[\frac{1}{2}\left(y_i^2 - 2y_i w^T.x_i + w^T.x_i x_i^T.w\right)\right]}{\partial w}$$

$$= \cancel{\frac{\partial \frac{1}{2}y_i^2}{\partial w}} \to 0 - \underbrace{\frac{\partial\, y_i w^T.x_i}{\partial w}}_{-\; y_i.x_i} + \underbrace{\frac{\partial \frac{1}{2}\overbrace{w^T.x_i}^{f(w)}\overbrace{x_i^T.w}^{g(w)}}{\partial w}}_{x_i.\hat{y}_i}$$

$$\frac{\partial \frac{1}{2} w^T.x_i.x_i^T.w}{\partial w} = \frac{1}{2}\frac{\partial w^T.x_i}{\partial w}.x_i^T.w + \frac{1}{2}\frac{\partial x_i^T.w}{\partial w}.w^T.x_i$$

$$= \frac{1}{2}.\,x_i\underbrace{x_i^T.w}_{\hat{y}_i} + \frac{1}{2}.\,x_i\underbrace{w^T.x_i}_{\hat{y}_i}$$

$$= \frac{1}{2}.x_i.\hat{y}_i + \frac{1}{2}.x_i.\hat{y}_i = x_i.\hat{y}_i$$

$$\frac{\partial Error}{\partial w} = (\hat{y}_i - y_i).x_i$$

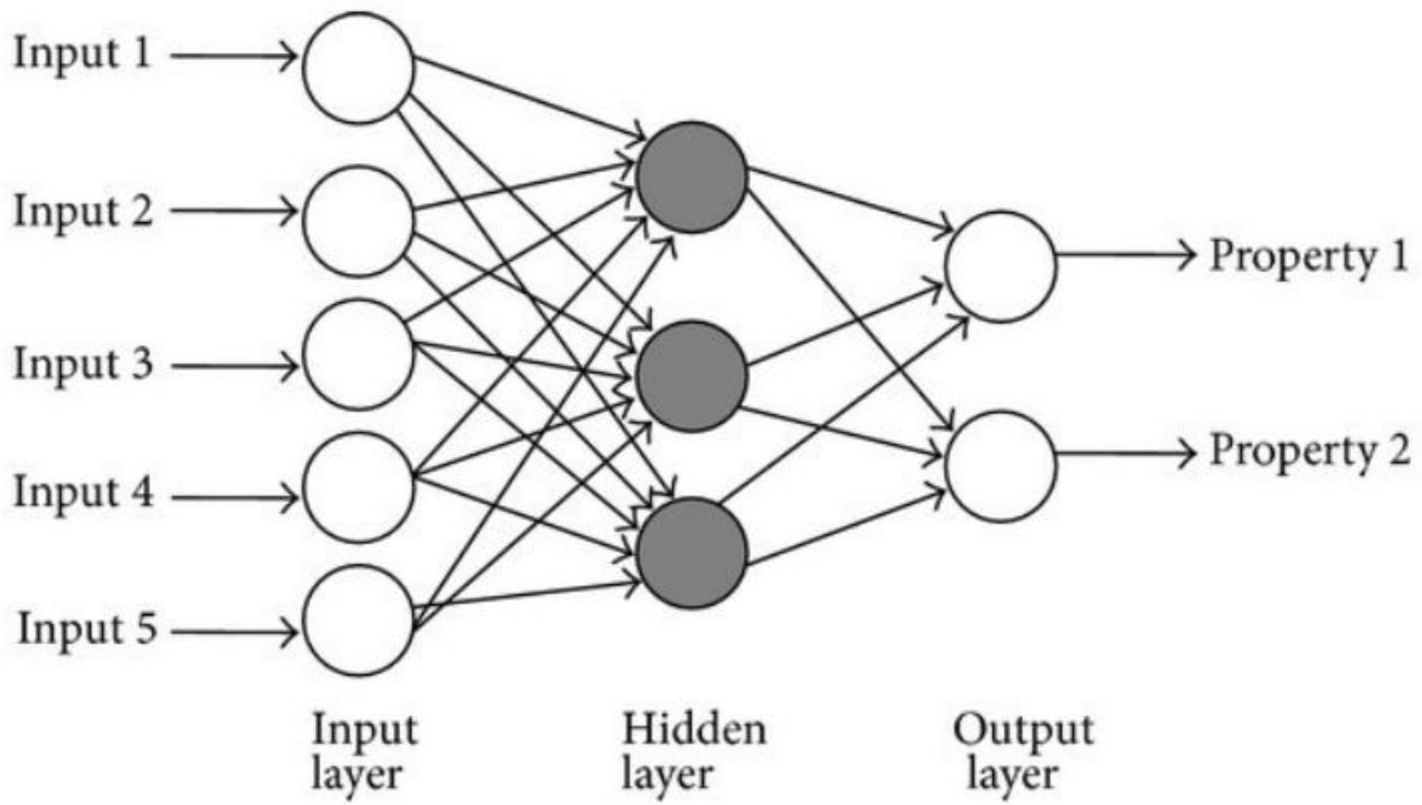
$$\Delta w = -\eta.\frac{\partial Error}{\partial w}$$

$$\Delta w = \boxed{\eta\,(y_i - \hat{y}_i).x_i}$$

The Algorithm:

- Step 1 : Initialize W-values randomly. (between -0.001 and +0.001)
- Step 2 : Calculate the gradients.
- Step 3 : Update the W-values.
- Step 4 : If the error is greater than epsilon, go to Step 2.

Solution 3: Multilayer Perceptron (Neural Network)



y = v*z

z = w*x

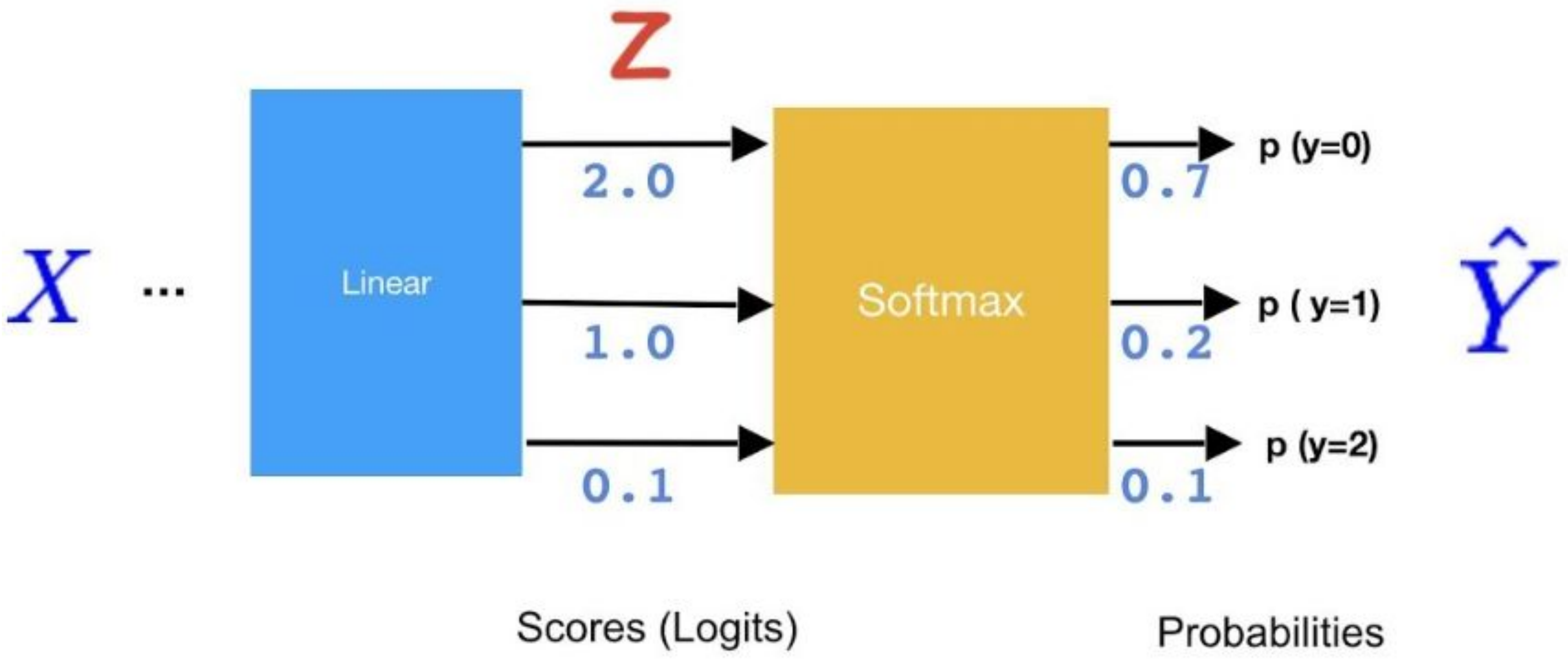
Sigmoid vs Softmax

Sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$= \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Z
X
...
Linear
2.0
1.0
0.1
Softmax
0.7
0.2
0.1
p (y=0)
p (y=1)
p (y=2)
Ŷ
Scores (Logits)
Probabilities

Gradient of V-values

$$\text{Error}(W, v \mid \mathcal{X}) = \frac{1}{2} \sum_{i=1} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \left[\sum_{k=1}^{H} v_k . z_{ik} + v_0 \right] \right)^2$$

$$\frac{\partial \text{Error}}{\partial v_h} = \frac{1}{\cancel{2}} . \cancel{2} . \sum_{i=1}^{N} \left(y_i - \left[\sum_{k=1}^{H} v_k . z_{ik} + v_0 \right] \right) . -z_{ih}$$

$$\boxed{= - \sum_{i=1}^{N} (y_i - \hat{y}_i) . z_{ih}}$$

$$\frac{\partial \left(\sum_{i=1}^{N} x_i . a_i \right)}{\partial x_3} = a_3$$

$$\Delta v_h = \eta \sum_{i=1}^{N} (y_i - \hat{y}_i) . z_{ih}$$

Gradient of W-Values

$$\frac{\partial Error}{\partial W_{hd}} = \boxed{\sum_{i=1}^{N}} \underbrace{\frac{\partial Error_i}{\partial \hat{y}_i}}_{=-(y_i-\hat{y}_i)} \cdot \underbrace{\frac{\partial \hat{y}_i}{\partial z_{ih}}}_{v_h} \cdot \underbrace{\frac{\partial z_{ih}}{\partial W_{hd}}}_{z_{ih}.(1-z_{ih})\, x_{id}}$$

$$Error_i = \frac{1}{2}(y_i-\hat{y}_i)^2$$

$$\hat{y}_i = \sum_{k=1}^{H} v_k . z_{ik} + v_0$$

$$z_{ih} = \text{sigmoid}(\underbrace{w_h^T . x_i}_{\sum_{d=1}^{D} w_{hd} . x_{id}})$$

$$\boxed{\frac{\partial Error}{\partial W_{hd}} = \sum_{i=1}^{N} (y_i-\hat{y}_i) . v_h . z_{ih}(1-z_{ih}) . x_{id}}$$

$$\Delta W_{hd} = \eta \sum_{i=1}^{N} (y_i-\hat{y}_i) . v_h . z_{ih}(1-z_{ih}) . x_{id}$$

The Algorithm:

- Step 1 : Initialize V-values and W-values randomly. (between -0.001 and +0.001)
- Step 2 : Calculate the gradients.
- Step 3 : Update the V-values, W-values and Z-values.
- Step 4 : If the error is greater than epsilon, go to Step 2.

RESOURCES

David Carlson’s Lectures

https://github.com/carlson9/KocPython2019/blob/master/13.NeuralNets/NN1.pdf

Mehmet Gönen’s Lectures

http://home.ku.edu.tr/~mehmetgonen/contact.html