

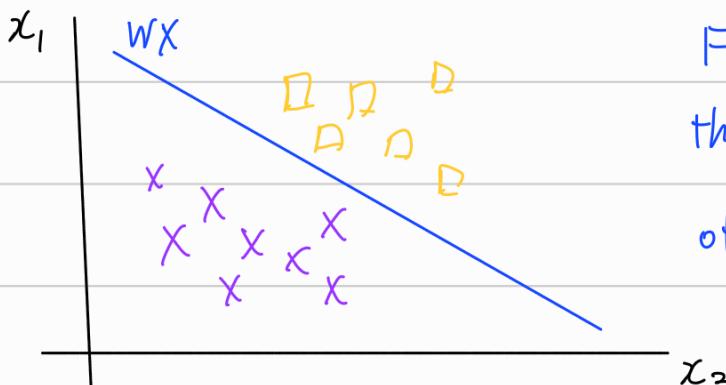
Recap: Logistic Regression

$$\cdot H_L(X) = W X$$

$$\cdot Z = H_L(X), g(Z) = \frac{1}{1 + e^{-Z}}$$

Linear Hypothesis is unfit for binary classification, as WX can produce any real value. \therefore we use the sigmoid function to compress the value of the hypothesis into a value btwn 0 and 1.

Ultimately, logistic classification training is about finding the line that separates two types of outcomes



Finding the W value of this blue line is the point of training.

Extending this to higher dimensions, training the model helps find the plane that separates two types of outcomes. This plane is called hyperplane

Multinomial Classification

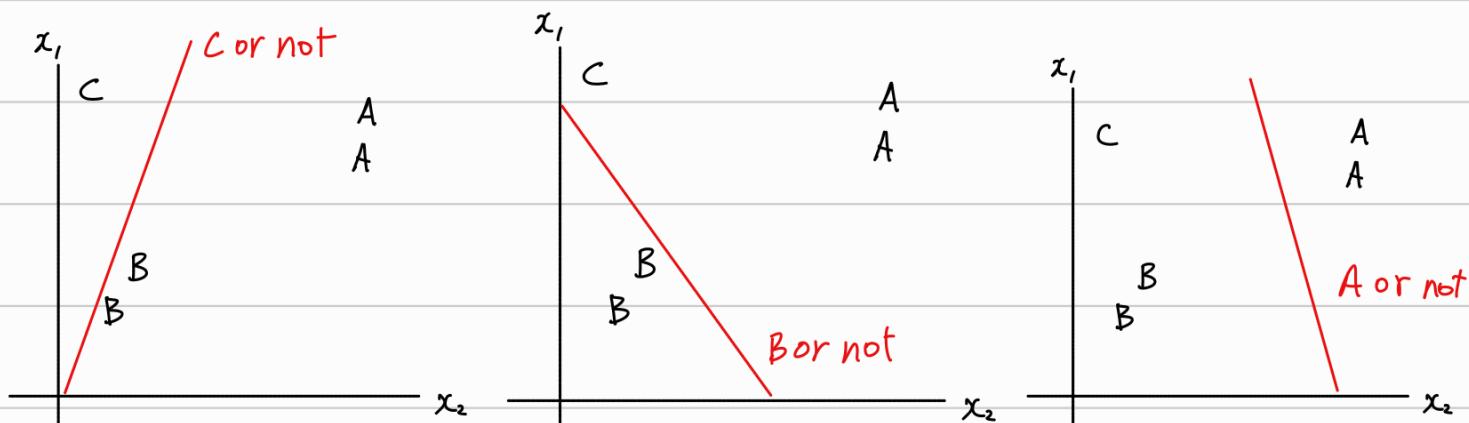
Classification where the outcome can be more than 2 outcomes is multinomial classification.

Consider the following example:

x_1 (hours)	x_2 (attendance)	y (grade)	x_1	x_2
10	5	A	C	A
9	5	A		A
3	2	B		B
2	4	B		B
11	1	C		

How do we achieve multinomial classification?

A: Multiple layers of binary classification



A binary classification for each letter grade is performed above. Each letter grade has a classifier that determines this, using a W matrix found w/ regression shown above.

Classifiers Ex:

$$\cdot W_A \rightarrow [w_{A1}, w_{A2}, w_{A3}]$$

$$\cdot W_B \rightarrow [w_{B1}, w_{B2}, w_{B3}]$$

$$\cdot W_C \rightarrow [w_{C1}, w_{C2}, w_{C3}]$$

Each classifier matrix is multiplied by the input vector matrix X .

But the classifier matrices can be merged into one matrix.

$$Wx = \begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix}$$

The resulting column matrix essentially contains $H_A(x)$, $H_B(x)$, and $H_C(x)$ respectively.

Normally, each element in the resulting matrix must be applied to independent sigmoid functions. However, there is a much more efficient method.

$$\text{Softmax function: } S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

This function has 2 important properties:

- ① The function always returns a value btwn 0 and 1
 - ② The sum of the outputs is 1.

∴ The process looks like:

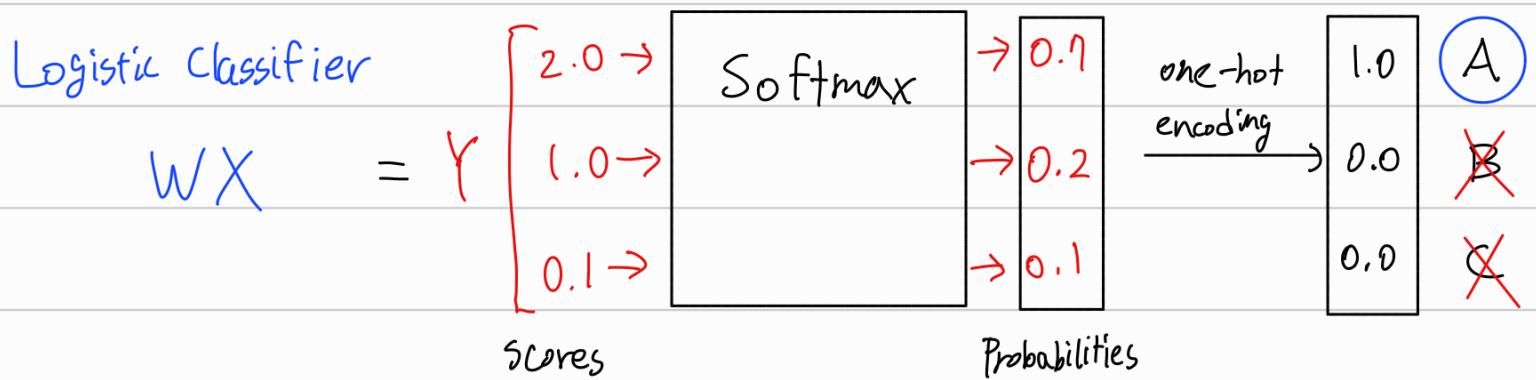
Logistic Classifier

The diagram illustrates the Softmax function mapping scores to probabilities. On the left, handwritten text reads "Classifier" above "WX". To its right is an equals sign followed by a red Y. Below the Y is the word "Scores". To the right of the Y is a large bracket containing three entries: "2.0 →", "1.0 →", and "0.1 →". Each entry maps to a box labeled "Softmax" above it. The first two boxes have arrows pointing to the right with values "0.1" and "0.2" respectively. The third box has an arrow pointing to the right with the value "0.1". To the far right, another set of boxes is labeled "Probabilitie" below them, with arrows pointing to the right showing values "0.7", "0.2", and "0.1" respectively.

Now we take the probabilities outputted by Softmax and return 1 for the highest probability class, and 0 for others.

This is called one-hot encoding. It's simple to find, as we just need the max of the probabilities. (Use argmax)

Now the process is:



Cost function : Cross-Entropy

Now that we finished the Hypothesis aspect, let us analyze the cost function. It uses a technique called Cross-Entropy.

$$D(S, L) = - \sum_i L_i \log(S_i)$$

Diagram illustrating the components of the cross-entropy loss function:

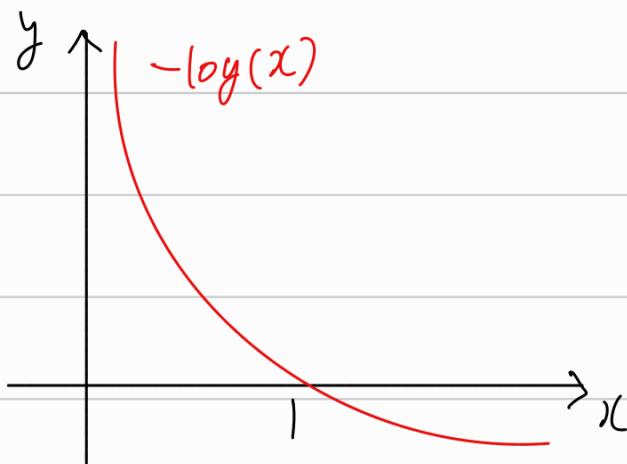
- Hypothesis value**: A red-bordered box containing the softmax probabilities $S(Y)$: 0.1 , 0.2 , 0.7 .
- Real value**: A blue-bordered box containing the target values L : 1.0 , 0.0 , 0.0 .
- Equation**: The formula for cross-entropy loss: $D(S, L) = - \sum_i L_i \log(S_i)$.
- Note**: A purple bracket below the equation points to the term $\log(S_i)$ with the text "use 'D' for cross entropy".

Why does this function work?

Recall: Purpose of cost function is to determine how close the hypothesis is to the actual data.

$$-\sum_i L_i \log(S_i) = \sum_i (\underbrace{L_i}_{\text{These are elements,}} \cdot \underbrace{(-\log(\bar{y}_i))}_{\text{not matrices!}})$$

Recall $-\log(x)$ looks like:



since \bar{y}_i can only have values $0 \approx 1$, we are only concerned w/ x values btwn 0 and 1 on the graph to the left.

As an example, let's say $L = [{}^{\circ}]_i$, where the top element tells us whether the grade is A and the bottom element tells us whether the grade is B.

$$L = {}^A_B [{}^{\circ} \ 1] \Rightarrow B$$

We have 2 cases for $\bar{F}(S \text{ hypothesis})$:

$$\bar{Y} = \begin{bmatrix} A \\ B \end{bmatrix} \Rightarrow A : \quad i=A$$

$$B : \quad i=B$$

$$\sum_i (L_i) \cdot (-\log(\bar{y}_i)) = 0 \cdot (-\log(1)) + 1 \cdot (-\log(0))$$

$$= 0 \cdot 0 + 1 \cdot \infty = \infty$$

$$\bar{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow B : \quad i=A$$

$$1 : \quad i=B$$

$$\sum_i (L_i) \cdot (-\log(\bar{y}_i)) = 0 \cdot (-\log(0)) + 1 \cdot (-\log(1))$$

$$= 0 \cdot \infty + 1 \cdot 0 = 0$$

\therefore We can see that when our hypothesis (\bar{Y}) matches our true data point (L), the cost is close to 0, and when it is not, the cost goes to infinity

Logistic cost vs cross-entropy:

Recall the logistic cost function:

$$C(H(x), y) = y \log(H(x)) - (1-y) \log(1-H(x))$$

and the cross-entropy function:

$$D(S, L) = - \sum_i L_i \log(S_i)$$

on close examination, we can see that these functions are actually the same.

The key insight is that there are 2 outcomes possible (0 or 1).

$$-\sum_i L_i \cdot \log(s_i) = -(L_1 \cdot \log(s_1)) - (L_2 \cdot \log(s_2))$$

The real data points L_1 and L_2 can only have values of 0 or 1 and have opposite values, so $L_2 = 1 - L_1$ is always true.

For our hypothesis values, thanks to Softmax, the values of s_1 and s_2 are between 0 and 1 and sum to 1 always.
(Ex. 0.7 and 0.3) $\therefore s_2 = 1 - s_1$

$$\therefore -(L_1 \cdot \log(s_1)) - (L_2 \cdot \log(s_2)) = -(L_1 \cdot \log(s_1)) - ((1-L_1) \cdot \log(1-s_1))$$

so if we change L_1 to y and s_1 to $H(x)$, we get

$$-y \cdot \log(H(x)) - (1-y) \cdot \log(1-H(x))$$

Cost function

Putting all this together, we can conclude our final cost/loss function:

$$J = \frac{1}{N} \sum_i D(S(wx_i + b), L_i)$$

↑
Training set
↑
Loss