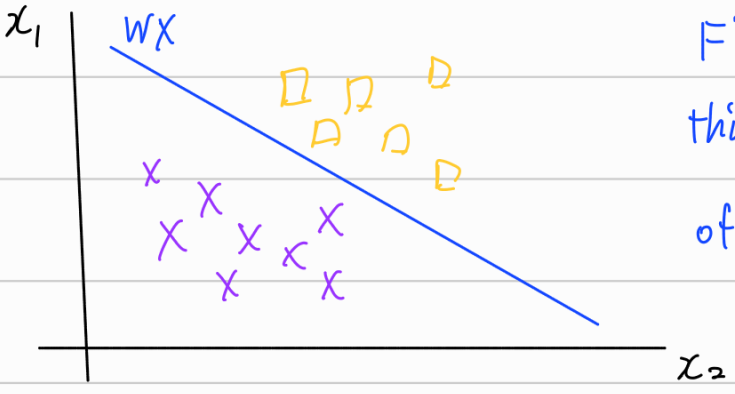
Recap: Logistic Regression

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

Linear Hypothesis is unfit for binary classification, as WX can produce any real value. ∴ 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.

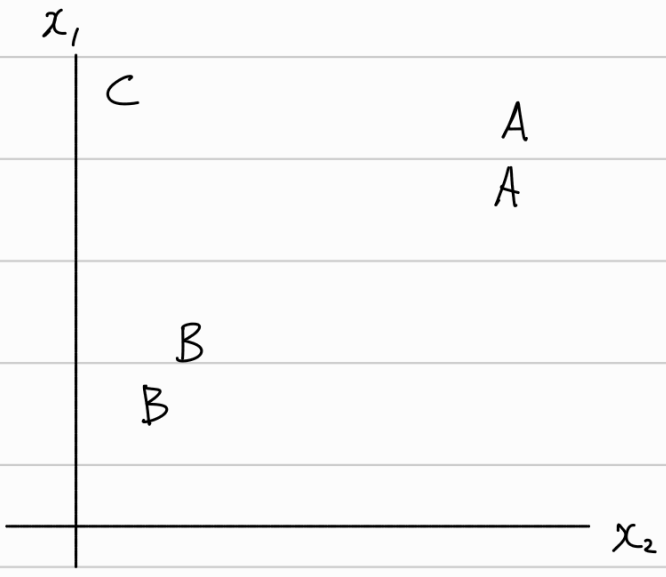
Extending this to higher dimensions, training the model helps find the plane that separates two types of outcomes. This plane is called hyperplane

Multinomial Classification

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

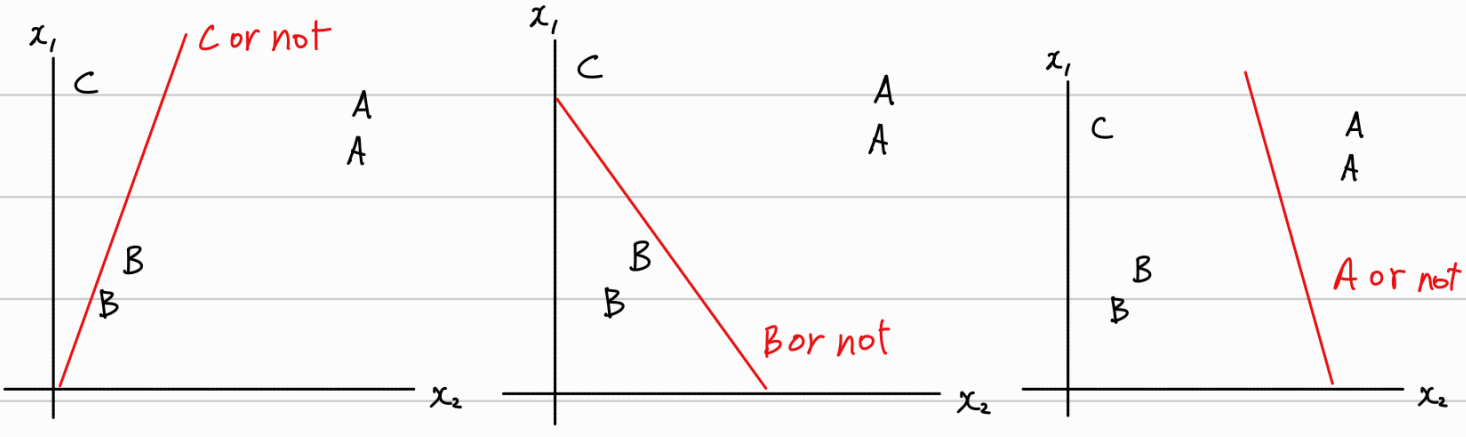
Consider the following example:

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



How do we achieve multinomial classification?

A: Multiple layers of binary classification



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

Classifiers Ex:

- $W_A \rightarrow [W_{A1}, W_{A2}, W_{A3}]$
- $W_B \rightarrow [W_{B1}, W_{B2}, W_{B3}]$
- $W_C \rightarrow [W_{C1}, W_{C2}, W_{C3}]$

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.

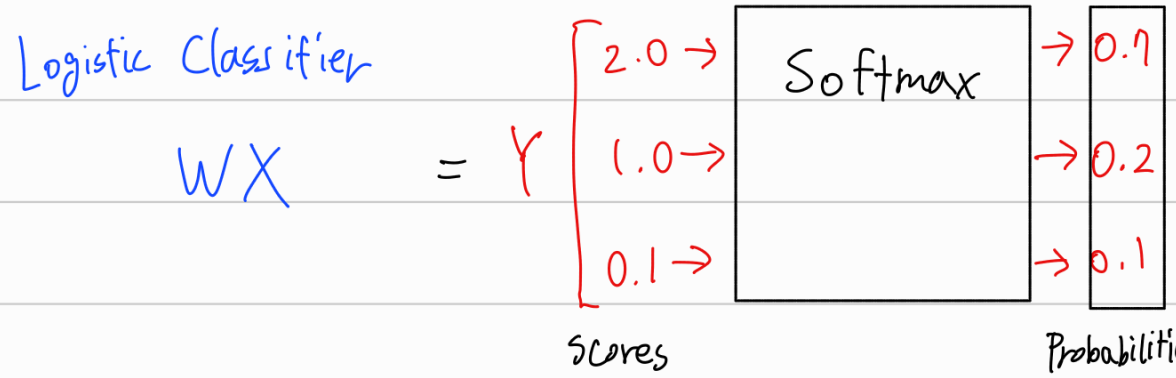
Softmax function: $$S(y_i) = \frac{e^{y_j}}{\sum_j e^{y_j}}$$

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.

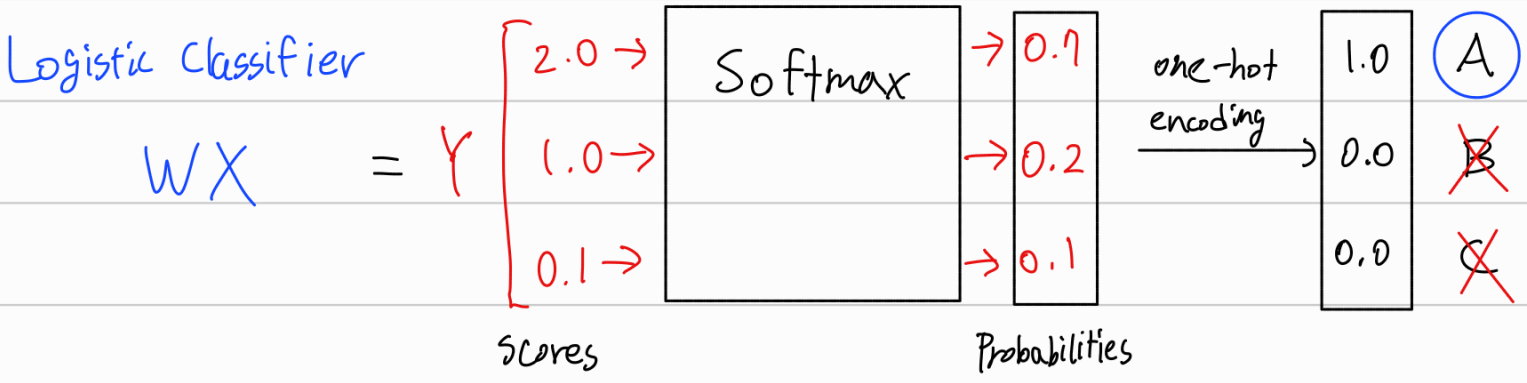
∴ The process looks like:



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

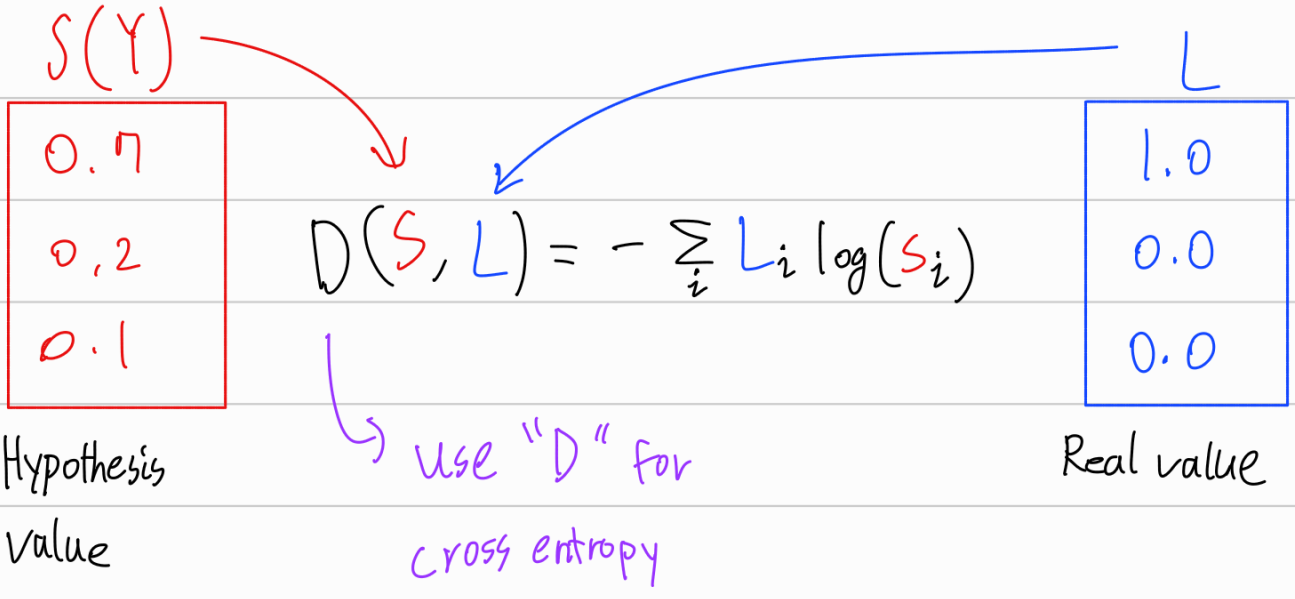
This is called one-hot encoding. It's simple to find, as we just need the max of the probabilities. (Use argmax)

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.



Why does this function work?

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

$$-\sum_i L_i \log(S_i) = \sum_i (L_i) \cdot (-\log(\bar{y}_i))$$

→ These are elements, not matrices!

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

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

As an example, lets say $L = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, where the top element tells us whether the grade is A and the bottom element tells us whether the grade is B.

$$L = \begin{matrix} A \\ B \end{matrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow B$$

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

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

$$\sum_i (L_i) \cdot (-\log(\bar{y}_i)) = \overbrace{0 \cdot (-\log(1))}^{i=A} + \overbrace{1 \cdot (-\log(0))}^{i=B}$$

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

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

$$\sum_i (L_i) \cdot (-\log(\bar{y}_i)) = \overbrace{0 \cdot (-\log(0))}^{i=A} + \overbrace{1 \cdot (-\log(1))}^{i=B}$$

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

$\therefore$ 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:

$$\mathcal{L} = \frac{1}{N} \sum_i D(S(Wx_i + b), L_i)$$

loss ↑ (pointing to $\mathcal{L}$); Training set (pointing to $\sum_i$, x_i and L_i)