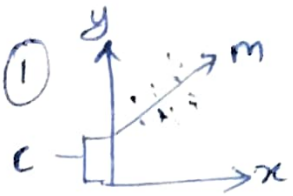
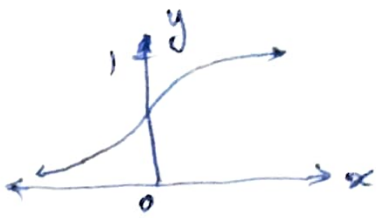
(1)



$y = mx + c$



$y = \sigma'(mx + c)$

(2) How to Evaluate? Cost = $J(\theta)$

MSE: $\frac{1}{n}\sum_{i=1}^{n}(y_i - y_{i\,Pred})^2$

$\div 2 \Rightarrow \frac{1}{2n}\sum_{i=0}^{n}(y_i - y_{i\,Pred})^2$

Categorical Crossentropy

$= -\sum_{i=1}^{\text{output size}} y_i \cdot \log(\hat{y}_i)$

(3) How to Optimize? Gradient Descent

$$\Delta_m = \frac{\partial(J(\theta))}{\partial m} = \frac{\partial\left(\frac{1}{2n}\left(\sum_{i=1}^{n}(y_i - y_{i\,Pred})^2\right)\right)}{\partial m}$$

$$= \frac{1}{2n}\frac{\partial}{\partial m}\left[\sum_{i=1}^{n}(y_i - y_{i\,Pred})^2\right]$$

$$= \frac{1}{2n}\left[2\sum_{i=1}^{n}(y_i - y_{i\,Pred}) \cdot \frac{\partial(y_{i\,Pred})}{\partial m}\right] = \frac{1}{n}\left[\sum_{i=1}^{n}(y_i - y_{i\,Pred}) \cdot \frac{\partial}{\partial m}(mx_i + c)\right]$$

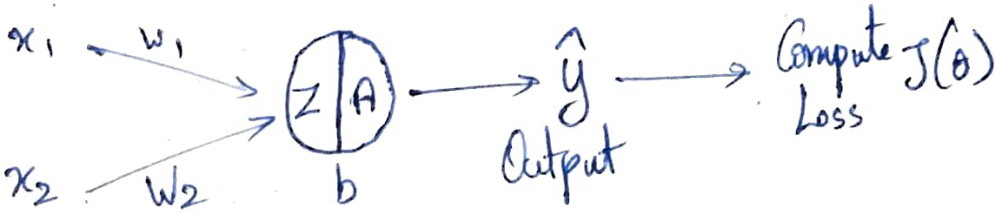
$$\Delta_m = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,Pred}) \cdot x_i$$

Similarly, $\Delta_c = \frac{1}{n}\left[\sum_{i=1}^{n}(y_i - y_{i\,Pred}) \cdot \frac{\partial}{\partial c}(mx_i + c)\right]$

$$= \frac{1}{n}\left[\sum_{i=1}^{n}(y_i - y_{i\,Pred}) \cdot 1\right]$$

[if $y = f(x)$, $\frac{\partial y^2}{\partial x} = 2y\frac{dy}{dx}$]

(4) Neural Networks:



$Z = w_1x_1 + w_2x_2 + b$, A → Activation function

Let $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})^2$ (Don't use MSE for classification. This is just for example)

Output $(\hat{y}) = \sigma(z) = \sigma(w_1x_1 + w_2x_2 + b)$

σ: Sigmoid Activation Function; $z = w_1x_1 + w_2x_2 + b$

Parameters are, w_1, w_2, b.

* When you have multiple paths, sum up the gradients w.r.t all the paths to get the final value which is used to update the parameter.

* $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial output} \cdot \frac{\partial output}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

* $\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial output} \cdot \frac{\partial output}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

* $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial output} \cdot \frac{\partial output}{\partial z} \cdot \frac{\partial z}{\partial b}$

$$\frac{\partial L}{\partial output} = \frac{\partial\left(\frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})^2\right)}{\partial \hat{y}} = \frac{-2}{n}\sum_{i=1}^{n}(y-\hat{y})$$

$$\frac{\partial output}{\partial z} = \frac{\partial(\sigma(z))}{\partial z} = \sigma(z)\cdot(1-\sigma(z))$$

$$\frac{\partial z}{\partial w_1} = \frac{\partial(w_1x_1 + w_2x_2 + b)}{\partial w_1} = x_1$$

$$\frac{\partial z}{\partial w_2} = x_2 \;,\; \frac{\partial z}{\partial b} = 1$$

This must be minus and not plus. Because, where we take slope at a point, it points upward. But we need to move downward to reach the minimum point.

$$\Rightarrow w_1 = w_1 - \alpha\left(\frac{\partial L}{\partial w_1}\right),\; w_2 = w_2 - \alpha\left(\frac{\partial L}{\partial w_2}\right),\; b = b - \alpha\left(\frac{\partial L}{\partial b}\right)$$

$\alpha \rightarrow$ learning rate.

(5) Activation Functions:

* **Binary Step Function** : $\{0, 1\}$

$$f(x) = \begin{cases} 1, & x >= 0 \\ 0, & x < 0 \end{cases}$$

$$f'(x) = 0$$

* **Linear** : $[-\infty, \infty]$ ← Output range of the activation function

$$f(x) = ax$$

$$f'(x) = a$$

* **Sigmoid** : $[0, 1]$

$$f(x) = \frac{1}{1+e^{-x}} = \sigma(x)$$

$$f'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

Causes: Vanishing Gradient

* **tanh** : $[-1, 1]$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$\tanh(x) = 2\sigma(2x) - 1$$

$$f'(x) = 1 - f(x)^2$$

Causes: Vanishing Gradient

May Cause these Problems

* **Softmax** :

$$S(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \; ; j = 1 \ldots k.$$

$$\sum_{j=1}^{k} \sigma(z)_j = 1 \; ; k \Rightarrow \text{No. of classes}$$

* **ReLU**: $[0, \infty]$

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

Causes: Dead Neuron

* **Parametric ReLU**: $[-\infty, \infty]$

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} a, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

a is found using Optimization Algorithm (Eg: gradient descent)

* **Leaky ReLU**:

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} a, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

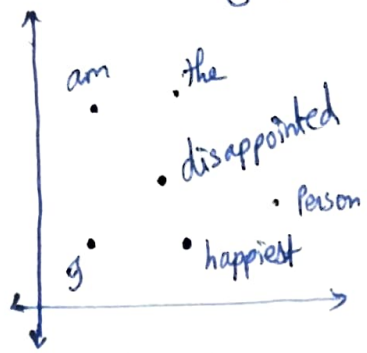
a is set manually.

⑥ Embeddings:

		Class
1.	I am the happiest person.	→ Joy
2.	I am disappointed.	→ Sad

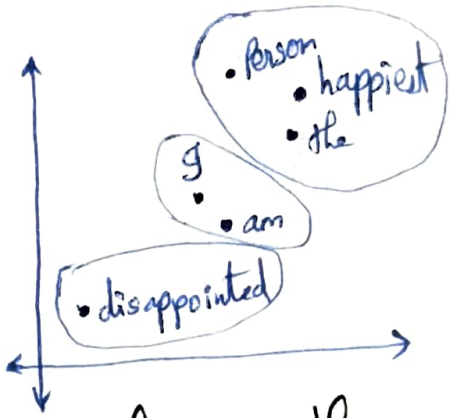
let Emb. dimension = 2

Initial Embedding Space



After Optimization →

Final Embedding Space



first, the tokens (individual words) are projected Randomly in Embedding Space.

After learning, the words are clustered together according to the impact they make on the outcome (Outputs)

* TOPICS:

* Linear Regression, MSE, Gradient descent.
* Logistic Regression, Log Loss, Sigmoid Activation Function.
* Multiclass classification, Softmax Activation, Categorical Crossentropy.
* Neural Networks, chain rule.
* Other Activations: tanh, ReLU, Parametric ReLU, Leaky ReLU
* Vanishing gradient: gradient (slope) value approaches 0.
  Exploding gradient: gradient value becomes too large.
* Embeddings - for projecting some value to a higher dimensional Space called Embedding Space.