Note: This is the summary note from Udacity Introduction to Deep Learning with PyTorch

Neural Network

- Find the boundary that separate two / more data set
- Complicated data -> need complicate algorithm
- Classify problem

Classification

- Classify problem
- How do we find the line / boundary between two data set

Linear Boundary

- Linear boundary -- Linear equation
  Equation: $w_1x_1 + w_2x_2 + b = 0$
  Vector notation: $WX + b = 0$
  Weight: $W(w_1, w_2)$
  Input: $X(x_1, x_2)$
  Label: $y = 0\ or\ 1$

$$prediction: \hat{y} = \begin{cases} 1\ if\ WX + b \geq 0 \\ 0\ if\ WX + b < 0 \end{cases}$$

- **Mission**: have Y-hat resemble to Y as close as possible in order to create linear boundary

Higher Dimension

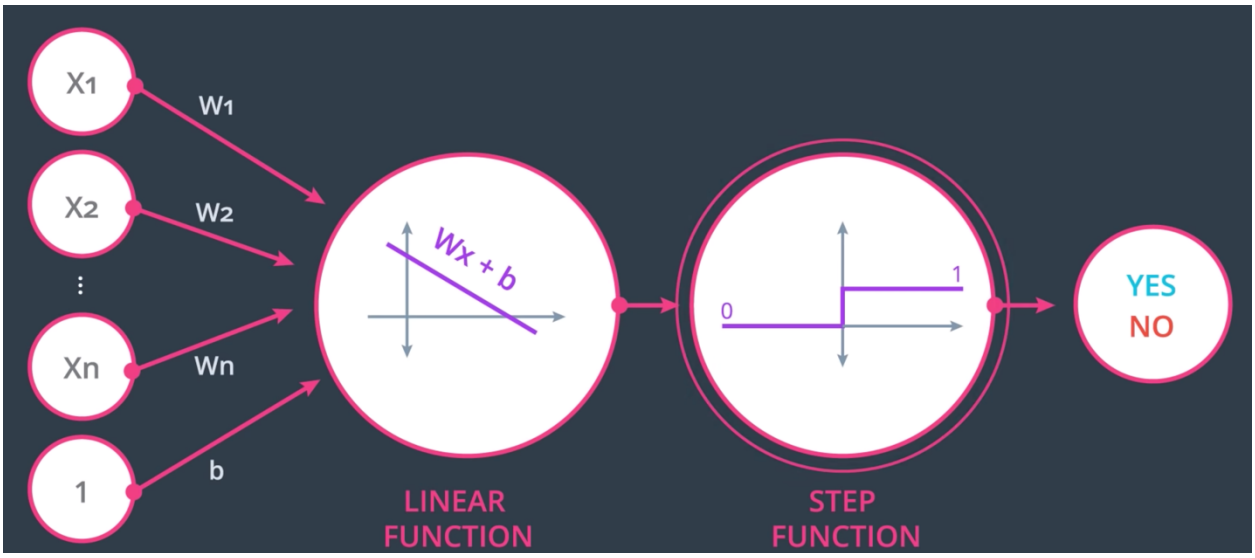
- Focus on what if we have more features / inputs

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$$

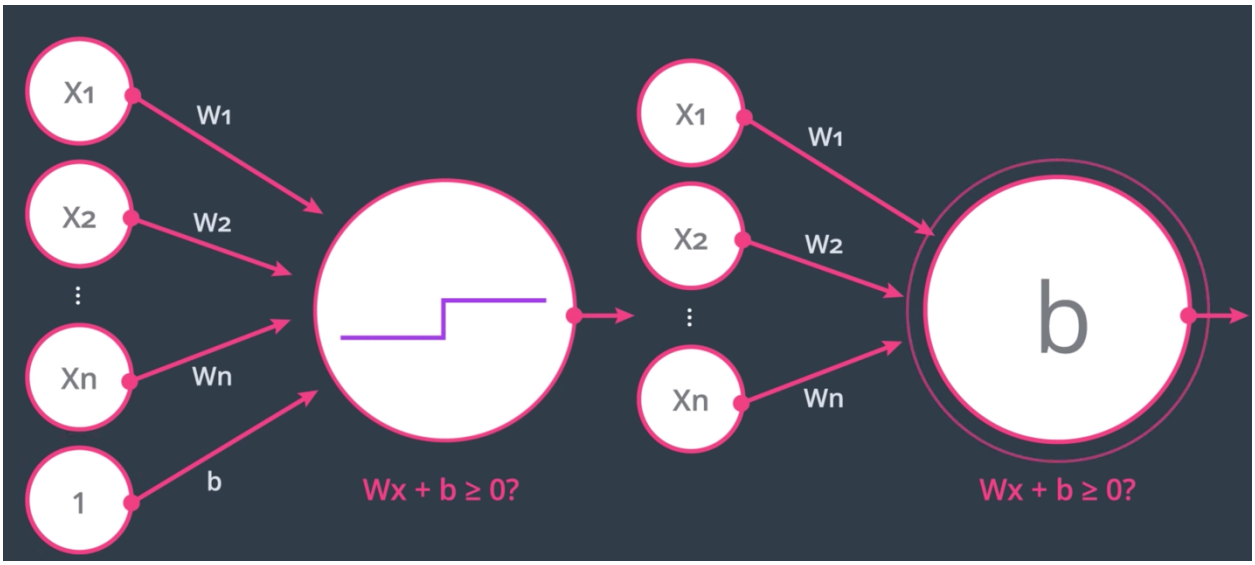
- Conclude: still same vector notation and prediction as in linear boundary

Perceptron

- Has features similar to human brain
- Building block of neural network ---> encoding the linear boundary equation into small graph
- Step function (can be discrete or continuous or some other functions) that's why it should live in a separate node



- Two ways represent precentron



Perceptron as Logical Operator

- Show application of perceptron as Logical operator
- https://medium.com/@stanleydukor/neural-representation-of-and-or-not-xor-and-xnor-logic-gates-perceptron-algorithm-b0275375fea1

Perceptron trick

- How we split the data?
  - Misclassified points wants the line close to them
  - So move the line to the misclassified points
- Point $A(x,\ y)$, Line: $w_1x_1 + w_2x_2 + b = 0$, Learning Rate = 0.1: this is used so that the line doesn't move drastically over already correct classified points, so move little by little
  - If A in positive area want Line to come closer, we do subtraction
    - $d = w_1 - (\mathrm{x} * 0.1)$
    - $e = w_2 - (\mathrm{y} * 0.1)$
    - $\mathrm{f} = \mathrm{b} - (1 * 0.1)$ (1 is added for bias unit)
    - New Line: $dx_1 + \mathrm{e}x_2 + f = 0$
  - If A in negative area want Line to come closer, we do addition
    - $d = w_1 + (\mathrm{x} * 0.1)$
    - $e = w_2 + (\mathrm{y} * 0.1)$
    - $\mathrm{f} = \mathrm{b} + (1 * 0.1)$ (1 is added for bias unit)
    - New Line: $\mathrm{d}x_1 + \mathrm{e}x_2 + f = 0$

Perceptron algorithm

- Algorithm

  For a point with coordinates $(\mathrm{p},\ \mathrm{q})$, label y, learning rate α and prediction given by the equation $\hat{y} = step(w_1 x_1 + w_2 x_2 + b)$:

  - If the point is correctly classified, do nothing.
  - If the point is classified positive, but it has a negative label:
    - $\alpha p - w_1$
    - $\alpha q - w_2$
    - $(\alpha * 1) - b$
  - If the point is classified negative, but it has a positive label:
    - $\alpha p + w_1$
    - $\alpha q + w_2$
    - $(\alpha * 1) + b$

  Note:

  - Step is the step function in this case the discrete step function that return either 1 or 0
  - we do this until we have minimum classified points or until a given threshold given

==========

Mini summary:

- We know what is the problem that we try to solve: classification (Linear data)
- We know what the perceptron is = single layer neural network
- Why we use perceptron?
- Solution (The algorithm that create linear boundary between points)
- We can split two points with Perceptron algorithm

==========