Lecture 15: Kernel perceptron, Neural Networks, SVMs etc

Instructor: Prof. Ganesh Ramakrishnan

Non-linear perceptron?

Advantages: ① non-linear owing to K

- Kernelized perceptron: $f(x) = sign\left(\sum_i \alpha_i y_i K(x, x_i) + b\right)$
  - INITIALIZE: α=zeroes()
  - REPEAT: for $< x_i, y_i >$
    - ★ If $sign\left(\sum_j \alpha_j y_j K(x_j, x_i) + b\right) \neq y_i$
    - ★ then, $\alpha_i = \alpha_i + 1$
    - ★ endif

Typically run for fixed # iterations for non-sep data

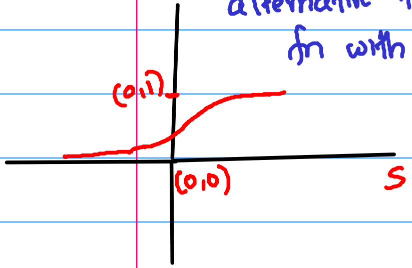
Disadvantages
① Struggles on non-seperable data (infinitely loops)
② How about more than 2 classes?

- Neural Networks: Cascade of layers of perceptrons giving you non-linearity
  - $sign\left((w^*)^T\phi(x)\right)$ replaced by $g\left((w^*)^T\phi(x)\right)$ where $g(s)$ is a
    1. step function: $g(s) = 1$ if $s \in [\theta, \infty)$ and $g(s) = 0$ otherwise OR
    2. sigmoid function: $g(s) = \frac{1}{1+e^{-s}}$ (smooth version of step fn)

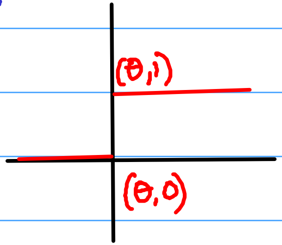
  Threshold changes as bias is changed.
  $y \in \{0,1\}$ (earlier: $y \in \{+1,-1\}$)

$g(s) = \frac{1}{1+e^{-s}}$

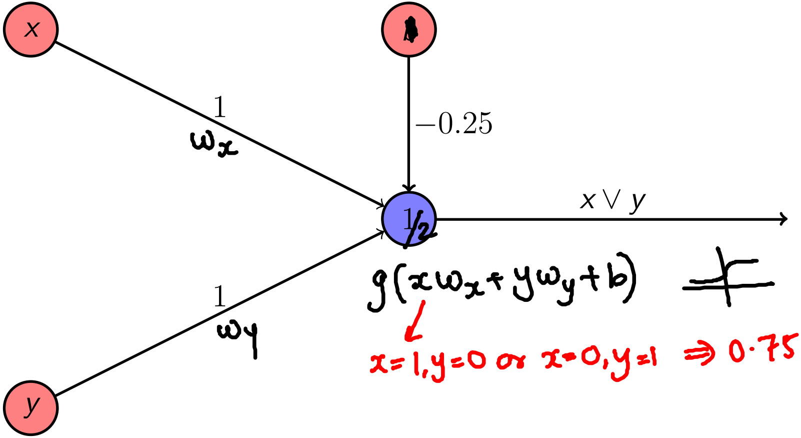
(sigmoid fn is a smooth alternative to step fn with $\theta = 0$)



step fn



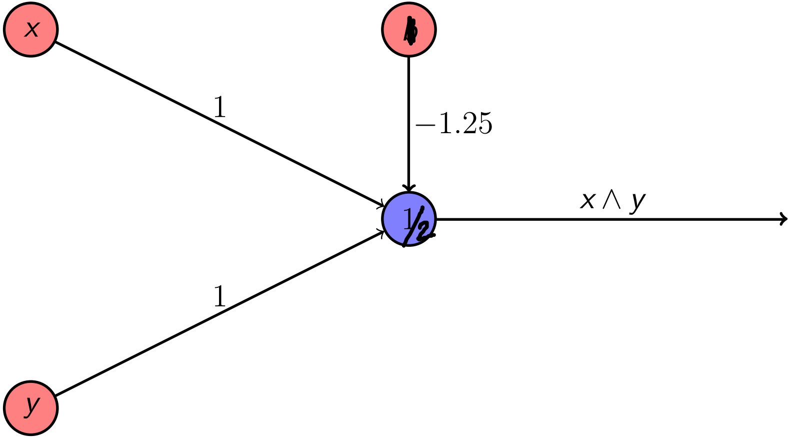
OR using perceptron : $x \vee y$



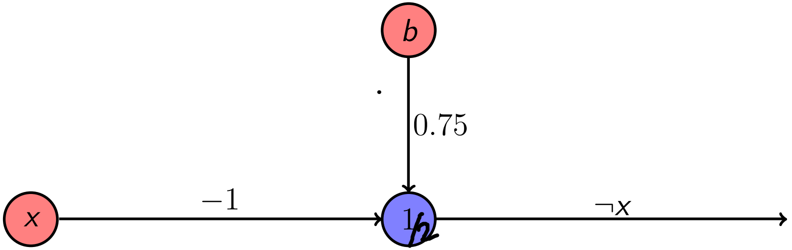
$g(x w_x + y w_y + b)$

$x=1, y=0$ or $x=0, y=1 \Rightarrow 0.75$

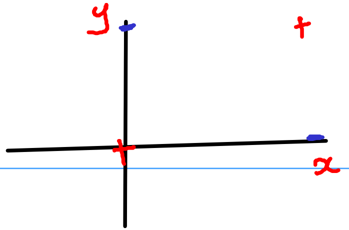
AND using perceptron



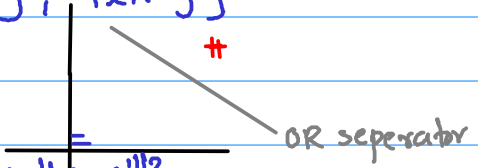
NOT using perceptron



How about XNOR: ?



Can you seperate the points in a diff space (of ϕ's) say: $\phi = [x \wedge y, \neg x \wedge \neg y]$



You can use kernel perceptron with this ϕ or even $K(x, x') = e^{-\frac{1}{2\sigma^2}\|x - x'\|_2^2}$ for some (large) value of σ

① We observe that the points are linearly seperable in space formed by $\phi = [x \wedge y, \neg x \wedge \neg y]$

② We also observe that we had already constructed perceptrons for $x \wedge y$, $\neg x$, $\neg y$ & therefore $\neg x \wedge \neg y$ & finally we have perceptron to seperate in the space $[x \wedge y, \neg x \wedge \neg y]$

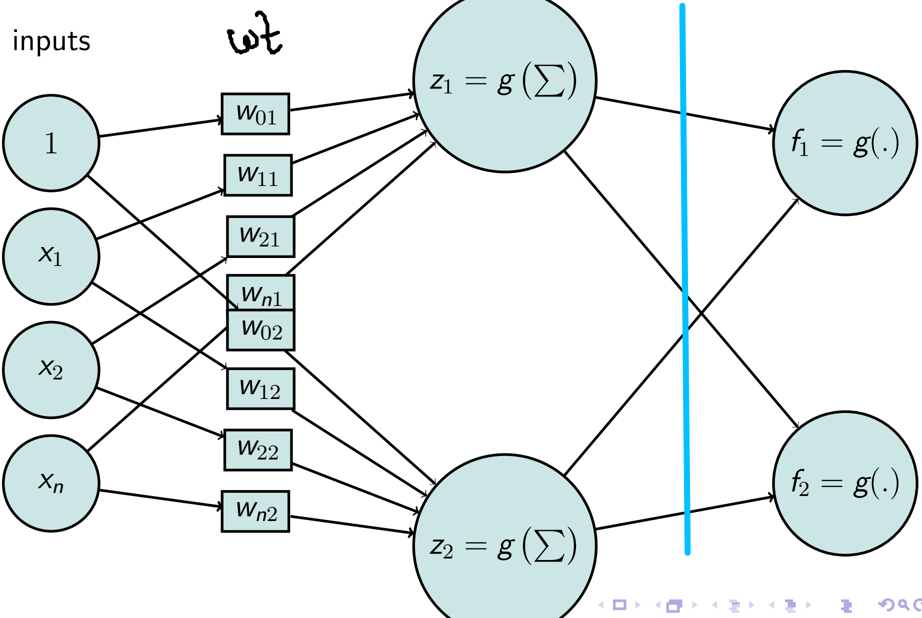
③ So why not employ a CASCADE (chain) of perceptrons, with output of some perceptrons becoming inputs to other perceptrons?

This is the central idea of neural networks!

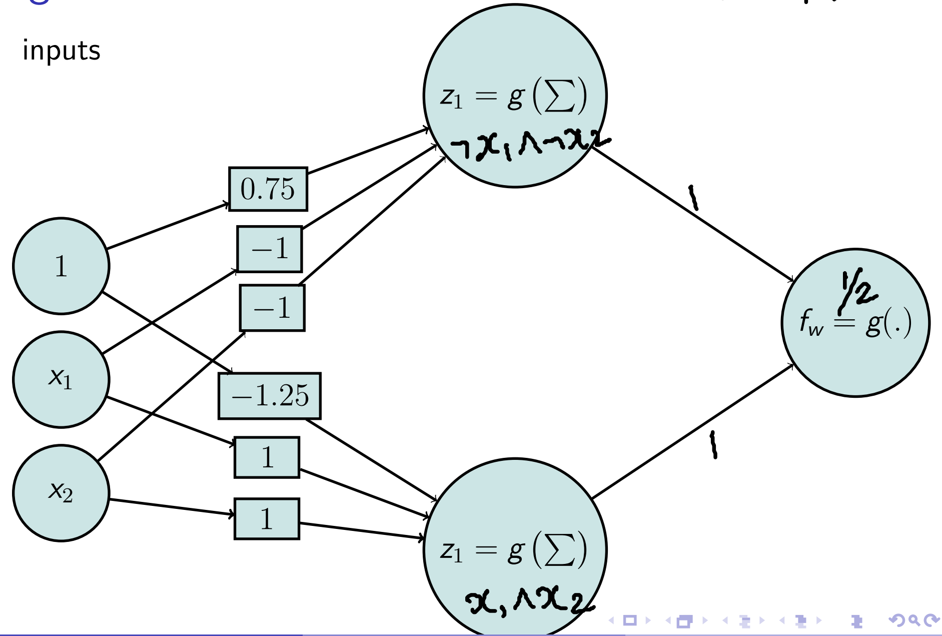
Feed-forward Neural Nets

hidden

more wts



Eg: Feed-forward Neural Net for ~~XOR~~ XNOR



1) While predicting class label for a new test/query point, I am fine with thresholding.

2) But thresholding creates problems while "training" owing to its non-smoothness

3) The non-smoothness has a compounding effect across layers of the neural net

Training a Neural Network

Good if you understand the domain & can customize the architecture o/w lots of layers, data & m/cs will serve the purpose

STEP 0: Pick a network architecture

- Number of input units: Dimension of features $x^{(i)}$.
- Number of output units: Number of classes.
- Reasonable default: 1 hidden layer, or if >1 hidden layer, have same number of hidden units in every layer.
- Number of hidden units in each layer a constant factor (3 or 4) of dimension of x.
- Logistic Loss function:

$$E(w) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_w\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_w\left(x^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2 \quad (5)$$