Lecture 18: Logistic Regression, Neural Networks

Instructor: Prof. Ganesh Ramakrishnan

MAP estimation and regularized LR

1. **FROM** MAP for LR: $\tilde{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\mathbf{w}) L(\mathcal{D}, \mathbf{w})$

   $= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{1}{\left(\frac{2\pi}{\lambda}\right)^{\frac{m}{2}}} e^{-\frac{\lambda}{2}\|\mathbf{w}\|_2^2} \prod_{i=1}^{m} \left(f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{y^{(i)}} \left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}$

   .....Taking $-\frac{1}{m}\log(.)$ transformation,
2. **TO** Min of the Regularized Logistic (Cross-Entropy) Loss function:

$$\tilde{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} - \left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \quad (1)$$

where we have ignored $-\frac{1}{m}\log\left(\left(\frac{2\pi}{\lambda}\right)^{\frac{m}{2}}\right)$ since this term is independent of $\mathbf{w}$.

.....Thus, MAP $\tilde{\mathbf{w}}$ can be found by minimizing the *Regularized Cross Entropy Error*

Gradient descent for Regularized LR

1. The final descent update

$$-\eta \nabla E(\mathbf{w}) = \eta \left[\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) - \underbrace{\lambda \mathbf{w}}_{\text{shrinkage.}} \right] \tag{2}$$

2. The iterative update rule:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{k} + \eta \left[\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}^k}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) - \lambda \mathbf{w}^{k} \right] \tag{3}$$

3. Stochastic version of the same:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{k} + \eta \left(y^{(i)} - f_{\mathbf{w}^k}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) - \eta \lambda \mathbf{w}^{k} \tag{4}$$

One Extension to multi-class logistic

1. Each class $c = 1, 2, \ldots, K-1$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2}, \ldots, \mathbf{w}_{c,k}, \ldots, \mathbf{w}_{c,p}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T\phi(\mathbf{x})}}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}} = \frac{e^{-w^T\phi(x)}}{1+e^{-w^T\phi(x)}} \quad (c=1)$$

for $c = 1, \ldots, K-1$ so that

For $k=2$

$$p(Y = K|\phi(\mathbf{x})) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}} = \frac{1}{1+e^{-w^T\phi(x)}} \quad (c=2)$$

special case: $K=2$ $\quad w_K = 0$

Alternative (equivalent) extension to multi-class logistic

(More intuitive)

1. Each class $c = 1, 2, \ldots, K$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2} \ldots \mathbf{w}_{c,p}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T\phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-(\mathbf{w}_k)^T\phi(\mathbf{x})}}$$

for $c = 1, \ldots, K$.

For weights w in second form, $\exists$ $\tilde{w}$ in first form with same distr & vice versa.

Tut 7: show equivalence of these two formulations!

Hint: In the second formulation divide each numerator & denominator by $e^{-w_k \phi(x)}$

Formulation ② to ①

$$\tilde{w}_c = w_c - w_K$$

Formulation ① to ②

$$w_c = \tilde{w}_c \text{ for } c = 1 \ldots K-1$$

$$w_K = 0$$

Assume w_c for ② & $\tilde{w}_c$ for ①

Tut 7: Are there other ways of generalizing Logistic Regression to multiple classes?

Logistic Regression Kernelized

1. We have already seen (a) Cross Entropy loss and (b) Bayesian interpretation for regularization
2. The Regularized (Logistic) Cross-Entropy Loss function (minimized wrt $\mathbf{w} \in \Re^p$):

$$E(w) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_w\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\|\mathbf{w}\|_2^2 \tag{5}$$

3. Equivalent dual kernelized objective[1] (minimized wrt $\alpha \in \Re^m$):

$$\equiv$$

$$E(w) = \frac{-1}{m}\left[\sum_{i=1}^{m} y^{(i)} w^T \phi(x^{(i)}) - \log\left(1 + e^{w^T \phi(x^{(i)})}\right)\right]$$

[1] http://perso.telecom-paristech.fr/~clemenco/Projets_ENPC_files/kernel-log-regression-svm-boosting.pdf

Logistic Regression Kernelized

1. We have already seen (a) Cross Entropy loss and (b) Bayesian interpretation for regularization
2. The Regularized (Logistic) Cross-Entropy Loss function (minimized wrt $\mathbf{w} \in \Re^p$):

$$E(w) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \log f_w\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\|\mathbf{w}\|_2^2 \tag{5}$$

3. Equivalent dual kernelized objective[1] (minimized wrt $\alpha \in \Re^m$):

$w^T\phi(x_i) = \sum_j \alpha_j K(x_i, x_j)$

$$E_D(\alpha) = \left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j + \frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1 + \sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\right)\right] \tag{6}$$

Decision function $f_{\mathbf{w}}(\mathbf{x}) = \dfrac{1}{1 + \sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}, \mathbf{x}^{(j)}\right)}$ $= P(y=1 \mid x)$

[1] http://perso.telecom-paristech.fr/~clemenco/Projets_ENPC_files/kernel-log-regression-svm-boosting.pdf

Some Tutorial 7 and 8 Questions

① $E(w) = -\sum_{i=1}^{m} y^{(i)} w^T \phi(x^{(i)}) + \log\left(1 + e^{w^T \phi(x^{(i)})}\right)$

② Matrix multiplication equality

- Prove that the Kernelized Logistic Regression form is equivalent to the original Logistic Regression minimum regularized cross entropy form: 2 Hints
- Contrast the Kernelized Logistic Regression with the dual of Support Vector Classifier

Logistic Regression Generalized to CRF

1. Multi-class LR: $c \in [1 \dots K]$ has weight vector $[w_{c,1} \dots w_{c,p}]$

$$\Pr(y = c \mid x) = \frac{e^{-w_c^T \phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-w_k^T \phi(\mathbf{x})}} = \frac{e^{-\mathbf{w}^T \vec{\phi}(x,c)}}{\sum_{k=1}^{K} e^{-\mathbf{w}^T \vec{\phi}(x,k)}}$$

$$\underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_c \\ w_k \end{bmatrix}}_{k \times p} \longrightarrow \mathbf{w}^T = [\underbrace{w_{1,1} \dots w_{1,p}}_{kp \text{ dim vector}} \quad w_{c,1} \dots w_{c,p} \quad \dots \quad w_{k,1} \dots w_{k,p}]$$

$$\vec{\phi}(x,c) = [\quad 0 \quad \phi(x) \quad 0 \quad]$$

[2] http://www.tzi.de/~edelkamp/lectures/ml/scripts/loglinearcrfs.pdf

Logistic Regression Generalized to CRF

1. Multi-class LR: $c \in [1 \dots K]$ has weight vector $[w_{c,1} \dots w_{c,p}]$

$$\Pr(y = c \mid x) = \frac{e^{-w_c^T \phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-w_k^T \phi(\mathbf{x})}} = \frac{e^{-\tilde{w}^T \phi(\mathbf{x}, y=c)}}{Z(\mathbf{x}, \tilde{w})}$$

   where $\tilde{w} = [w_{1,1} \dots w_{1,p}, \dots w_{c,1} \dots w_{c,p}, \dots w_{K,1} \dots w_{K,p}]$ and $\tilde{\phi}(\mathbf{x}, y) = [\underbrace{\delta(y,1)\phi(\mathbf{x})}_{\text{If } y \neq 1,\text{ will be } 0}, \dots, \underbrace{\delta(y,c)\phi(\mathbf{x})}_{\phi(x)} \dots \underbrace{\delta(y,K)\phi(\mathbf{x})}_{0}]$ and $\delta(a,b) = 1$ if $a = b$ and $= 0$ otherwise

2. Extended to non-iid inference in Conditional Random Fields[2] with $\mathbf{x} = [\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}]$ and $\mathbf{y} = [y^{(1)} \dots y^{(n)}]$:

[2] http://www.tzi.de/~edelkamp/lectures/ml/scripts/loglinearcrfs.pdf

Logistic Regression Generalized to CRF

1. Multi-class LR: $c \in [1 \dots K]$ has weight vector $[w_{c,1} \dots w_{c,p}]$

$$\Pr(y = c \mid x) = \frac{e^{-w_c^T \phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-w_k^T \phi(\mathbf{x})}} = \frac{e^{-\tilde{w}^T \phi(\mathbf{x}, y=c)}}{Z(\mathbf{x}, \tilde{w})}$$

① Structure over y can be used directly
② shared params

where $\tilde{w} = [w_{1,1} \dots w_{1,p}, \dots w_{c,1} \dots w_{c,p}, \dots w_{K,1} \dots w_{K,p}]$ and $\phi(\mathbf{x}, y) = [\delta(y, 1)\phi(\mathbf{x}), \dots, \delta(y, c)\phi(\mathbf{x}) \dots \delta(y, K)\phi(\mathbf{x})]$ and $\delta(a, b) = 1$ if $a = b$ and $= 0$ otherwise

2. Extended to non-iid inference in Conditional Random Fields[2] with $\mathbf{x} = [\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}]$ and $\mathbf{y} = [y^{(1)} \dots y^{(n)}]$:

[AD, VB, NN]   [Today, is, Thursday] $w_i \to -\infty$ [discouraging]

$$\Pr(\mathbf{y} \mid \mathbf{x}) = \frac{e^{-\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})}}{Z(\mathbf{x}, \mathbf{w})}$$

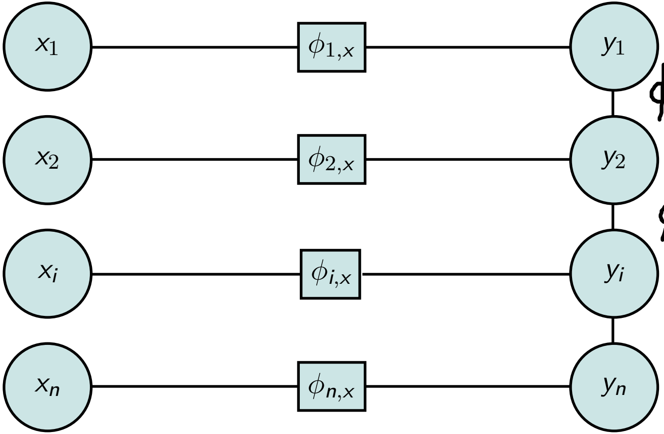
$\phi_i(-, [-\ AD, JJ] \dots) = 1$

[JJ, JJ, NN, VB, VB]   [Thursday, afternoon, lecture, is, boring]

[2] http://www.tzi.de/~edelkamp/lectures/ml/scripts/loglinearcrfs.pdf

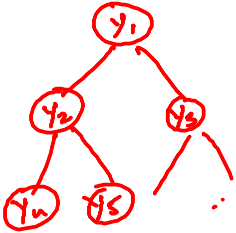
Conditional Random Fields (Linear)[3]

inputs $x-$potentials classes & $y-$potentials $\phi_{i,y}$

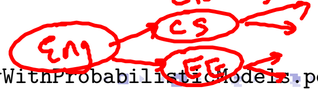


$\phi_{c_1,c_2}(y_1,y_2)$

$\phi_{c_1,c_2}(y_2,y_3)$



Eg: Hierarchical classification



[3]CRF is an instance of Graphical Models (detailed notes at https://www.cse.iitb.ac.in/~cs725/notes/classNotes/misc/CaseStudyWithProbabilisticModels.pdf

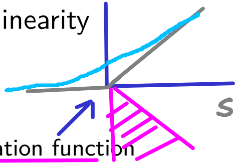
Non-linear perceptron?

- 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

- 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: $g(s) = \frac{1}{1+e^{-s}}$
    3. Rectified Linear Unit (ReLU): $g(s) = max(0, s)$: Most popular activation function
    4. Softplus: $g(s) = \ln\left(1 + e^s\right)$

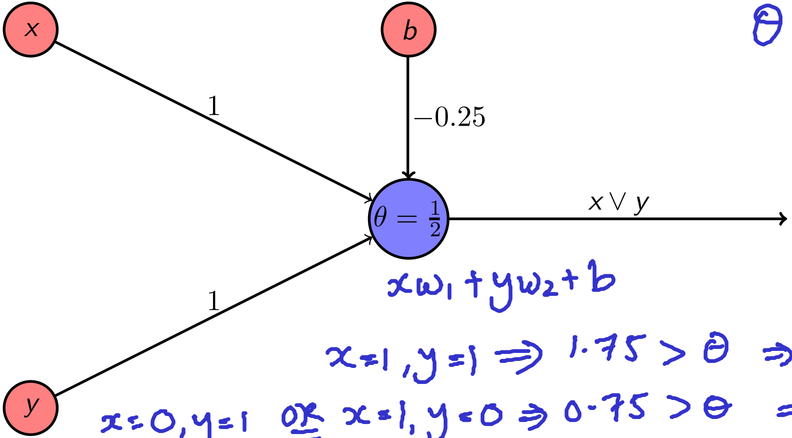
  Options 2, 3 and 4 have the thresholding step deferred. Threshold changes as bias is changed.

max(0,s)



OR using (step) perceptron $(x \vee y)$

$\theta \in (-0.25, 0.75)$



$xw_1 + yw_2 + b$

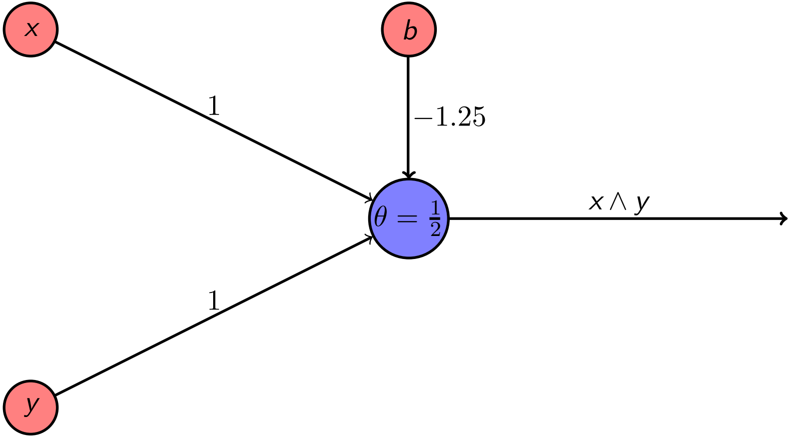
$x=1, y=1 \Rightarrow 1.75 > \theta \Rightarrow o/p = 1$

$x=0, y=1$ OR $x=1, y=0 \Rightarrow 0.75 > \theta \Rightarrow o/p = 1$

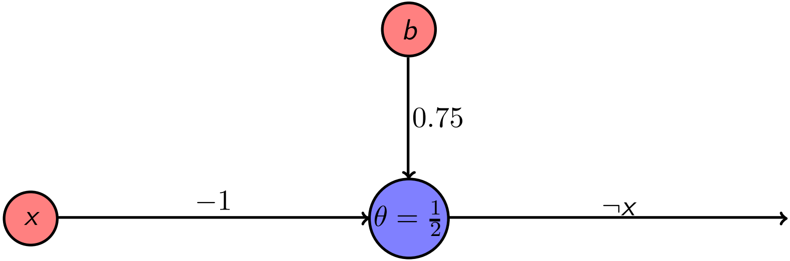
$x=0, y=0 \Rightarrow -0.25 < \theta \Rightarrow o/p = 0$

AND using (step) perceptron

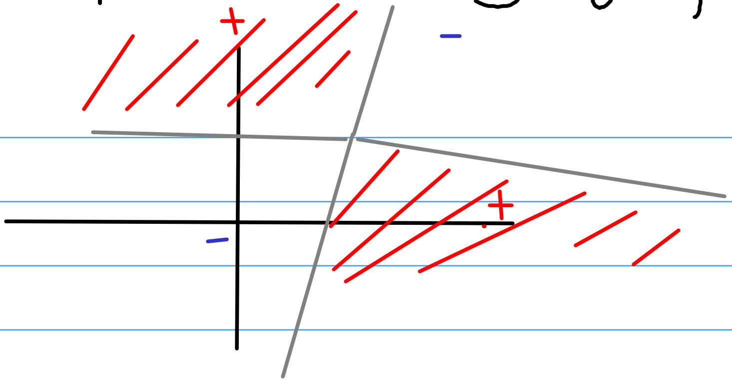
$(x \wedge y)$



NOT using perceptron

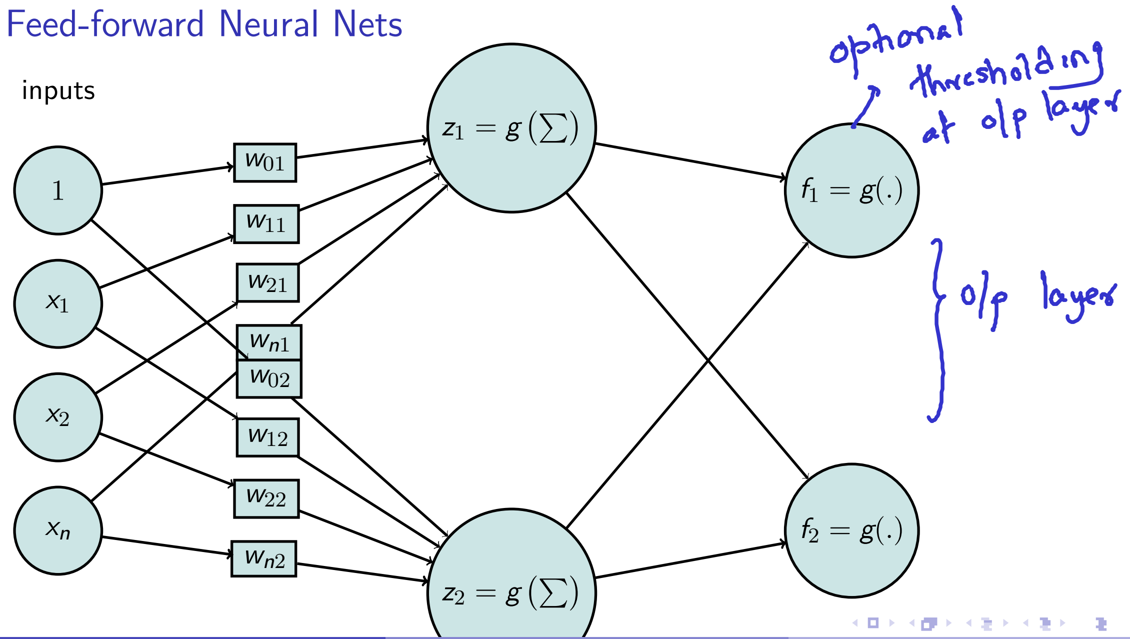


How about XOR using single perceptron



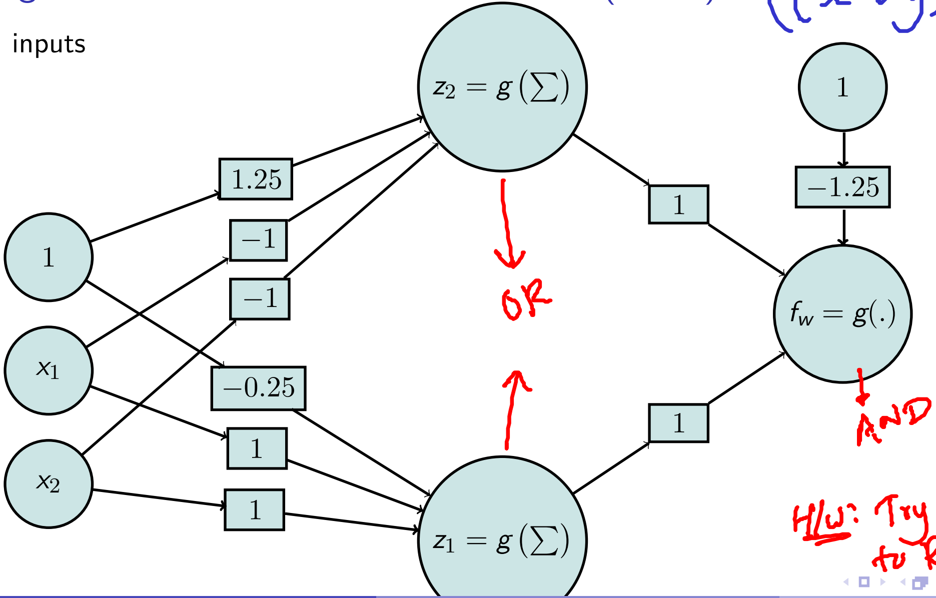
No linear seperator possible!

Feed-forward Neural Nets



Eg: Feed-forward Neural Net for XOR ($\theta = 0$)

$((x \vee y) \wedge (\neg x \vee \neg y))$



H/W: Try varying units to ReLU/Sigmoid.

safe1 education-1

cs224minoriit

user=passwd=anon to login to SAFE

cs725:419 is the quizid