Lecture Notes for

Machine Learning in Python

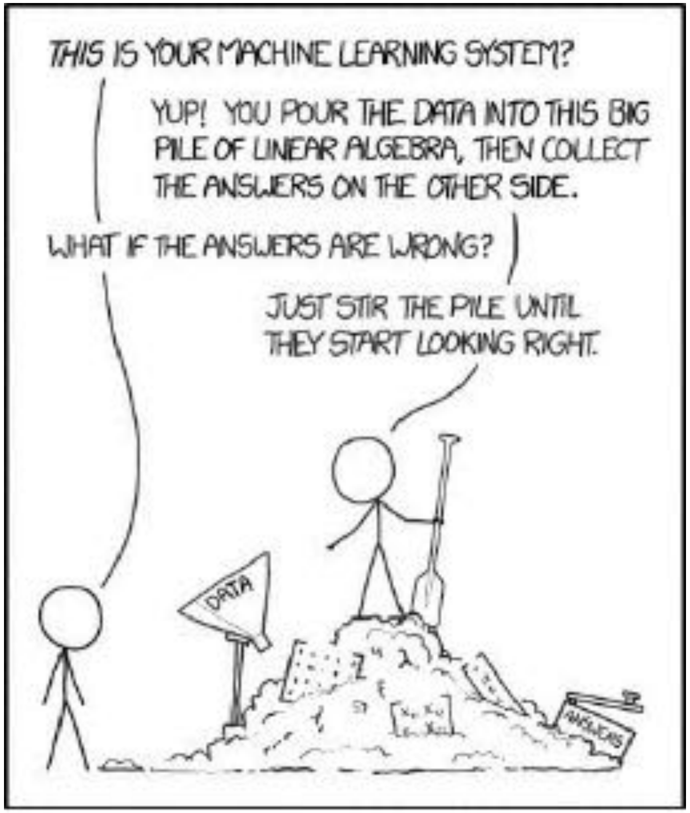
Professor Eric Larson

Town Hall + MLP History

Class Logistics and Agenda

- Logistics:
  - Next time: Flipped Module on back propagation
- Multi Week Agenda:
  - Today: Neural Networks History, up to 1980
  - Today: Multi-layer Architectures
  - Town Hall, Lab 3 (if time)
  - Flipped: Programming Multi-layer training

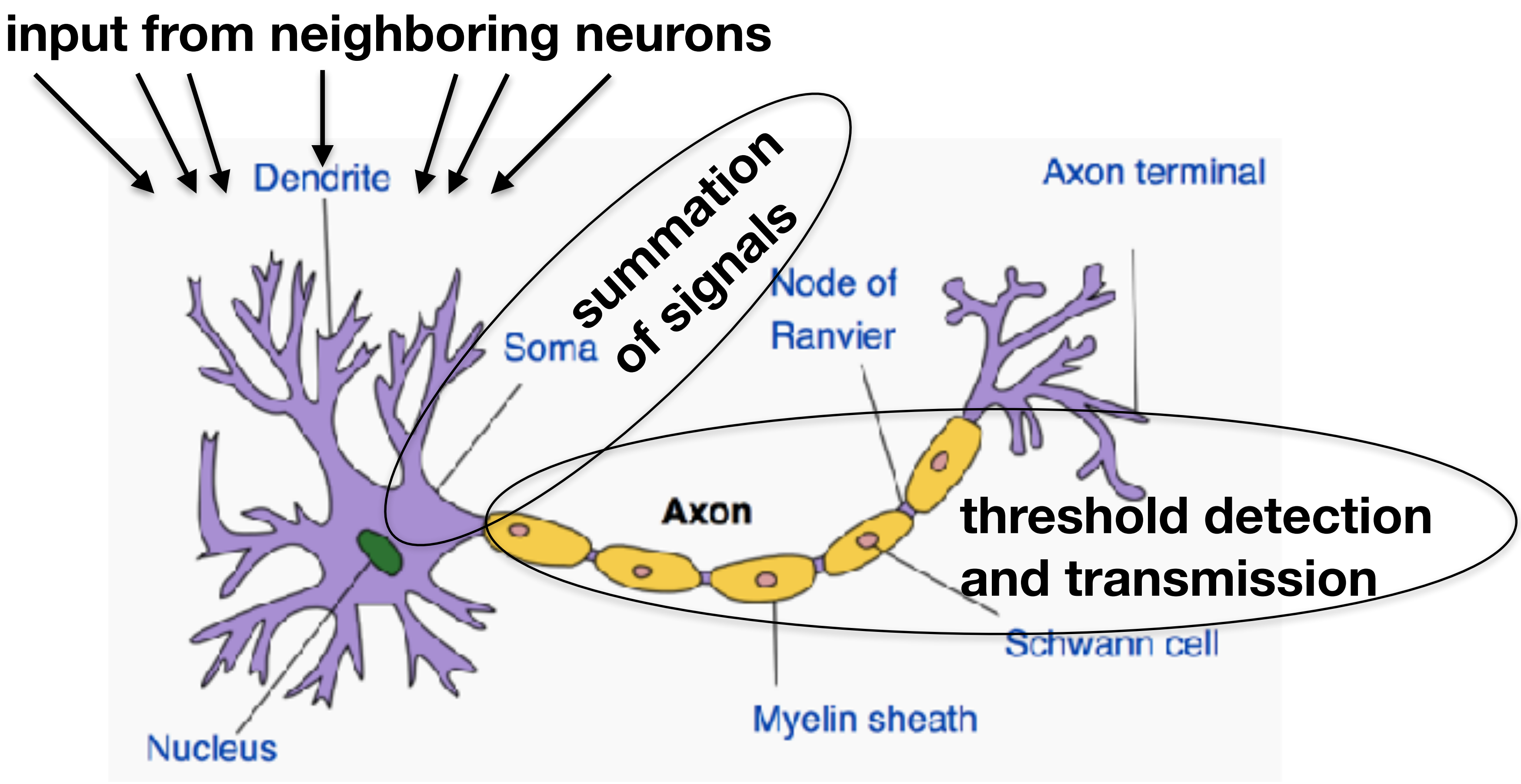
A History of Neural Networks



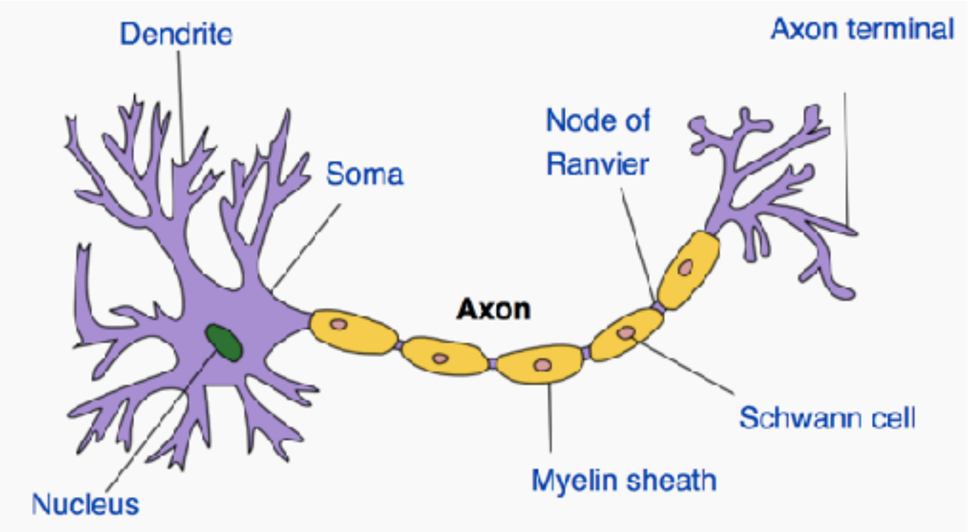
Machine Learning 101

Neurons

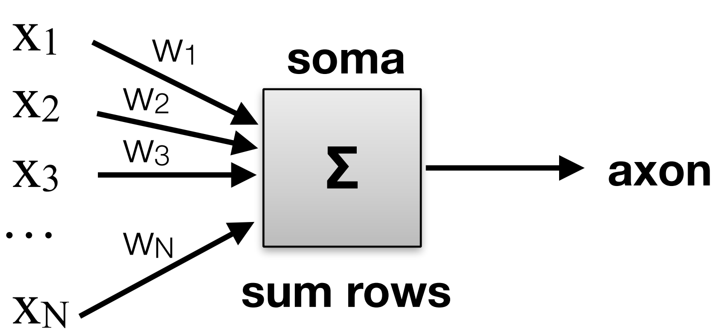
- From biology to modeling:



McCulloch and Pitts, 1943



dendrite



input

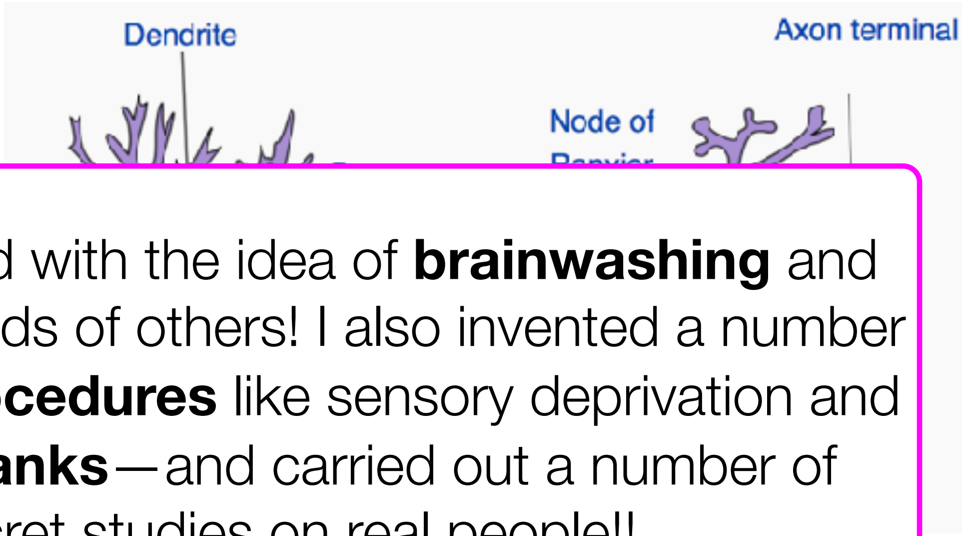
logic gates of the mind

Warren McCulloch

Walter Pitts

Neurons

- McCulloch and Pitts, 1943
- Donald Hebb, 1949
    - Hebb's Law: close neurons fire together
    - neurons "learn
    - easier synaptic
    - basis of neura



I as infatuated with the idea of **brainwashing** and controlling minds of others! I also invented a number of **torture procedures** like sensory deprivation and **isolation tanks**—and carried out a number of secret studies on real people!!

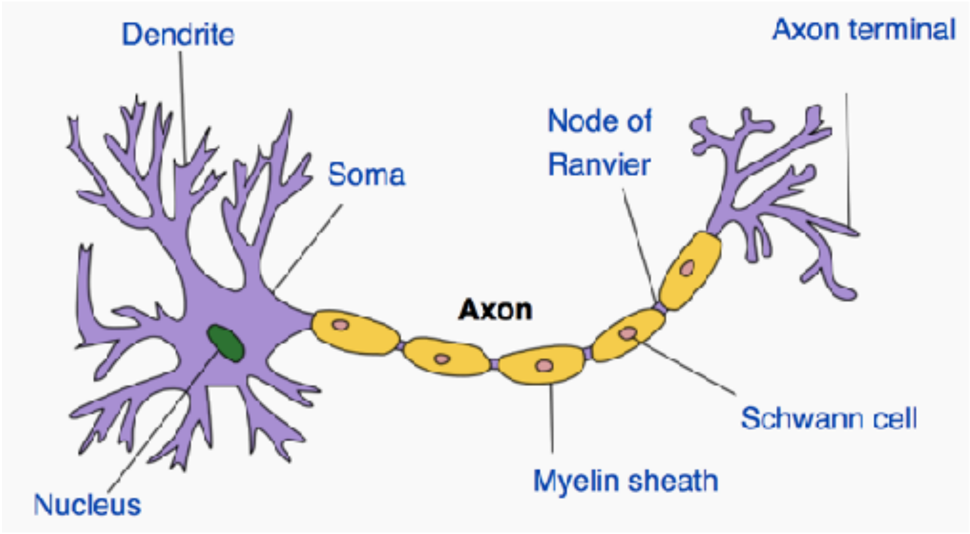
Donald O. Hebb

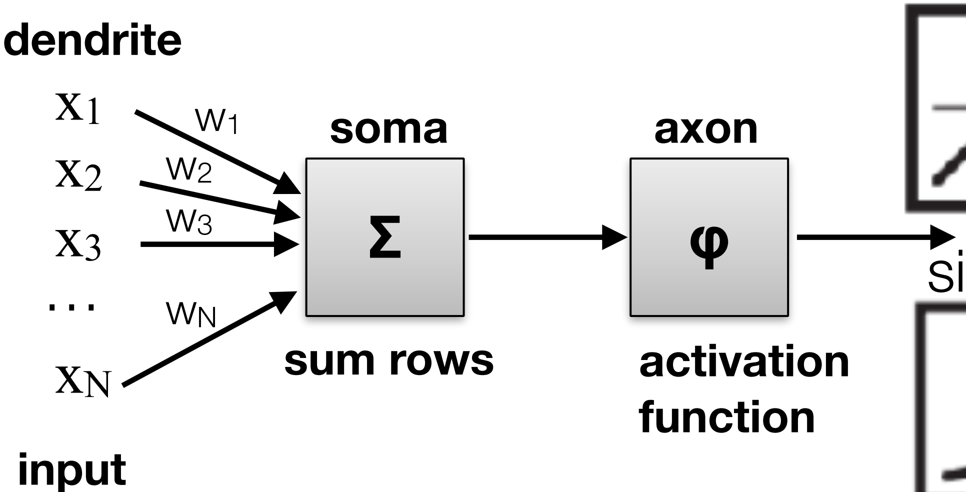
Warren McCulloch

Walter Pitts

Rosenblatt’s perceptron, 1957

Frank Rosenblatt





hard limit

$$a = -1 \quad z < 0$$
$$a = 1 \quad z >= 0$$

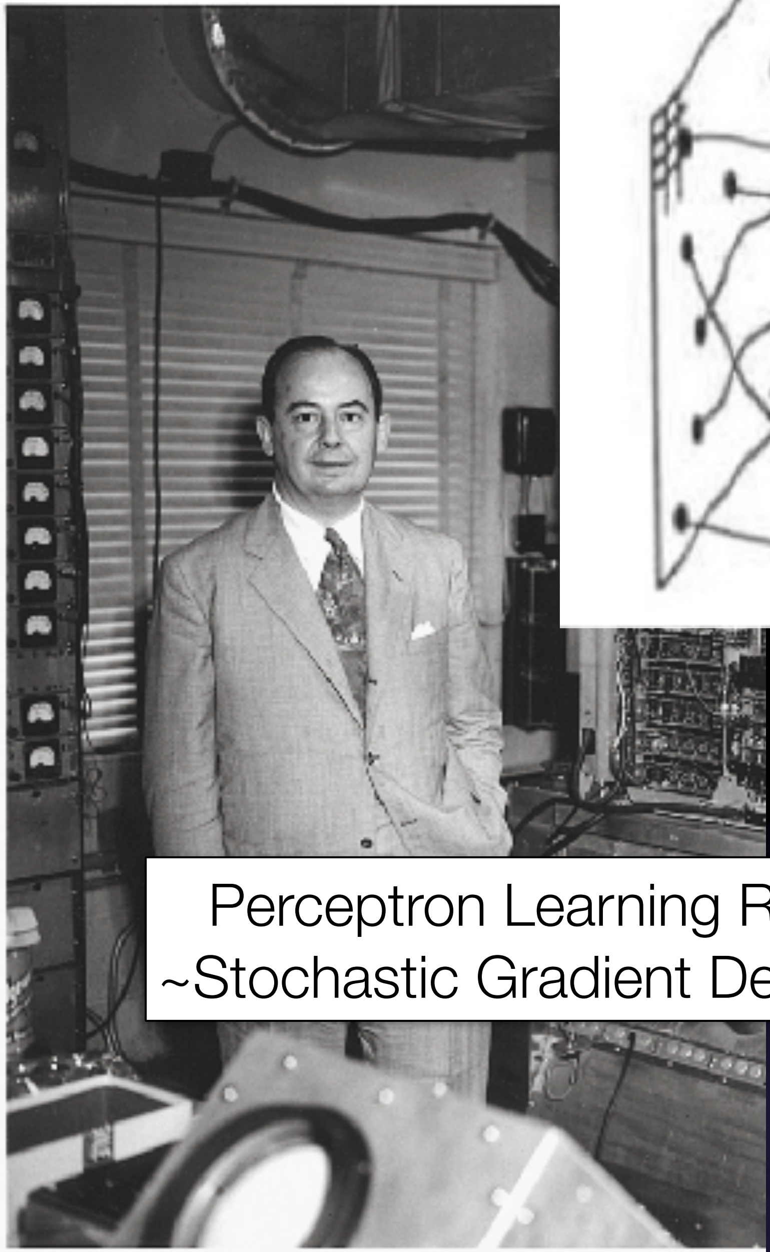
linear

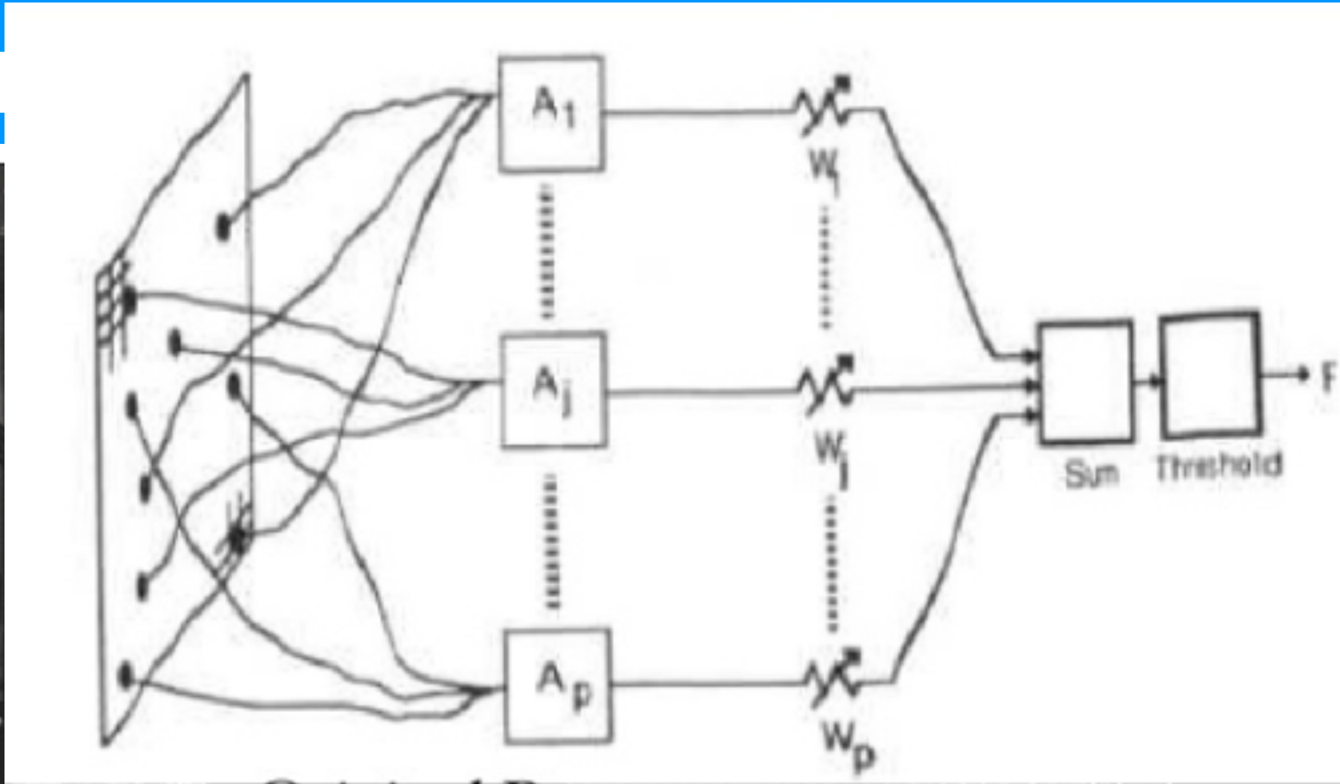
$$a = z$$

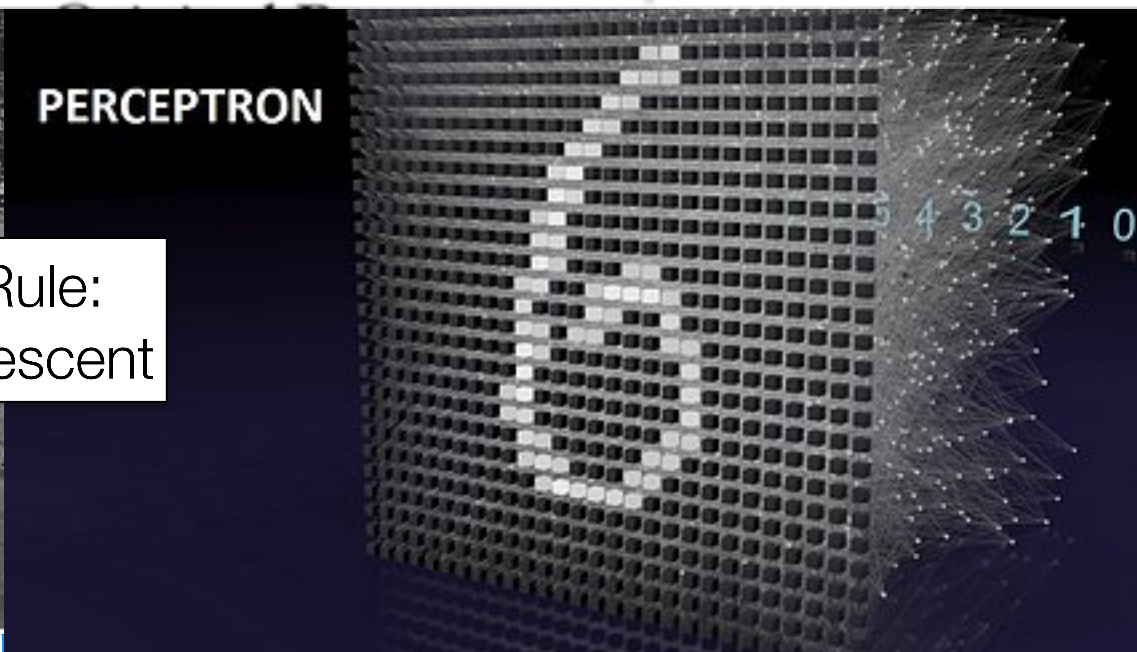
sigmoid

$$a = \frac{1}{1+\exp(-z)}$$

The Mark 1

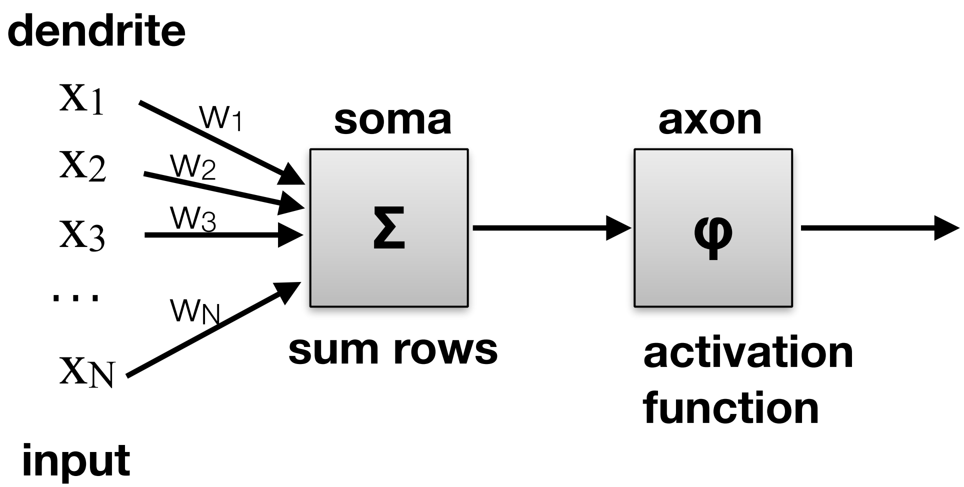


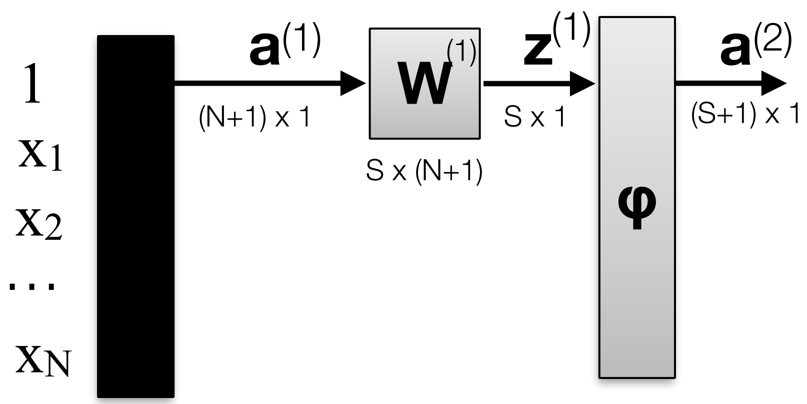




Perceptron Learning Rule:
~Stochastic Gradient Descent

Layers Notation





a=**x** + concat bias term

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_N \end{bmatrix}^{(i)} \qquad \mathbf{a} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_j \\ \vdots \\ x_N \end{bmatrix}$$

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{a} = \mathbf{W}_{1:N} \cdot \mathbf{x}^{(i)} + \mathbf{b}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & \dots & w_{1,N} \\ & & \dots & & \\ w_{S,0} & w_{S,1} & w_{S,2} & \dots & w_{S,N} \end{bmatrix}$$

$$[\mathbf{z}^{(1)}]^{(i)} = \mathbf{W}^{(1)} \cdot [\mathbf{a}^{(1)}]^{(i)} = \mathbf{W}^{(1)}_{1:N} \cdot \mathbf{x}^{(i)} + \mathbf{b}^{(1)}$$

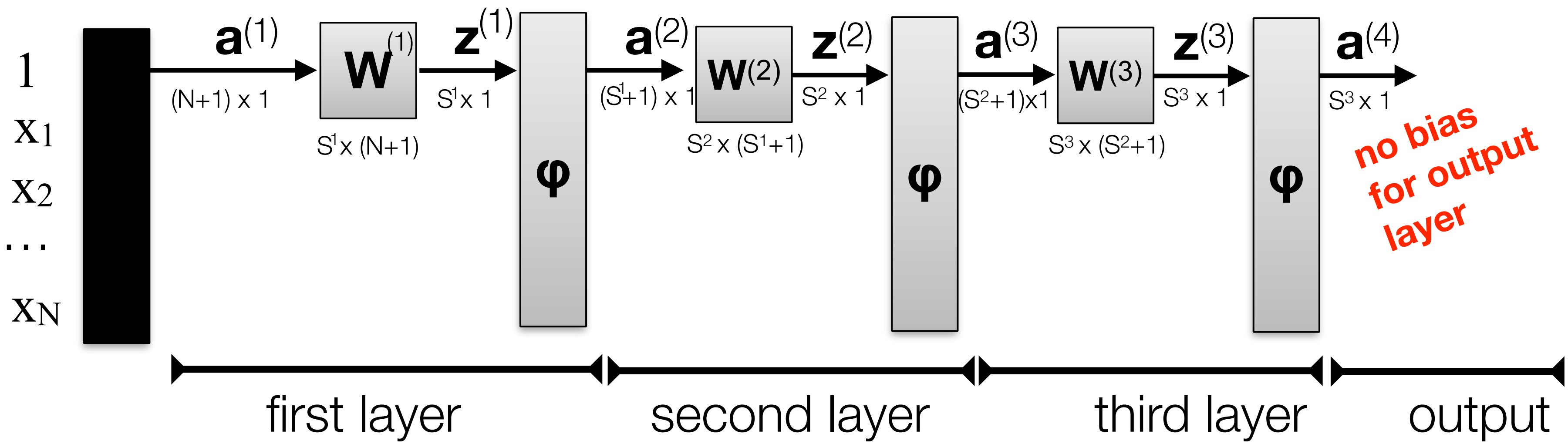
$\mathbf{a}^{(next)} = \boldsymbol{\phi}(\mathbf{z}^{(current)})$ + concat bias term

$$\mathbf{a}^{(next)} = \begin{bmatrix} 1 \\ \phi(z_1^{curr}) \\ \vdots \\ \phi(z_N^{curr}) \end{bmatrix} \rightarrow \mathbf{a}^{(L)} = \begin{bmatrix} 1 \\ \phi(z_1^{L-1}) \\ \vdots \\ \phi(z_N^{L-1}) \end{bmatrix}$$

$\mathbf{x}^{(i)}$ One row from Table data becomes input column to model

notation adapted from *Neural Network Design*, Hagan, Demuth, Beale, and De Jesus

Multiple Layers Notation



$\mathbf{a}^{(L+1)} = \boldsymbol{\phi}(\mathbf{z}^{(L)})$ + concat bias term

$\mathbf{a}^{(4)}$ rows=unique classes

$\mathbf{z}^{(L)} = \mathbf{W}^{(L)}\mathbf{a}^{(L)}$

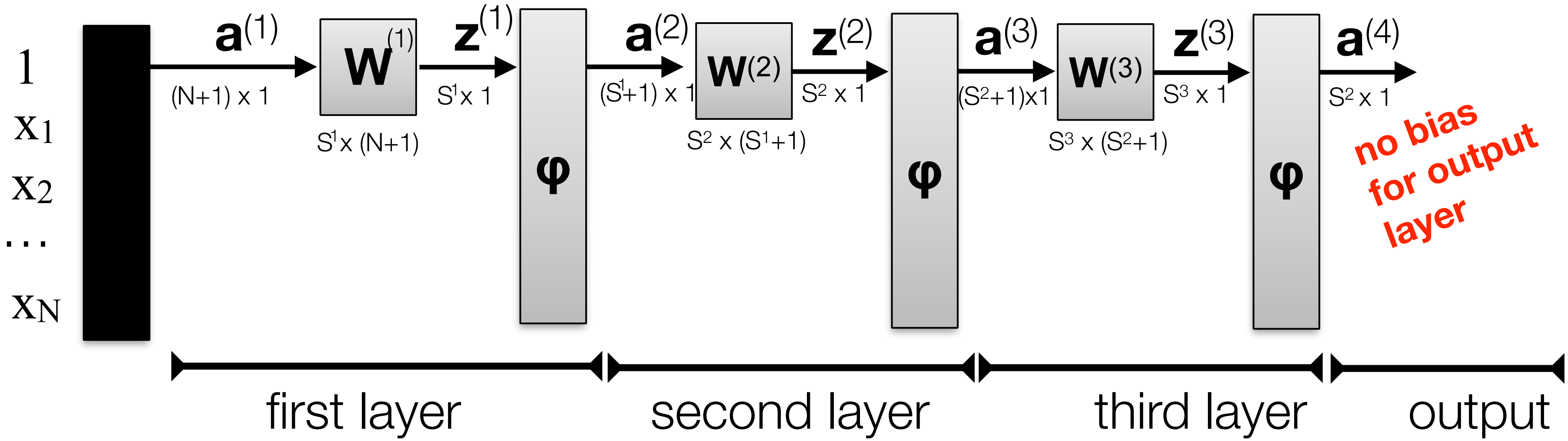
$\mathbf{z}^{(L)} = \mathbf{W}^{(L)}\boldsymbol{\phi}(\mathbf{z}^{(L-1)})$

$\mathbf{z}^{(L)} = \mathbf{W}^{(L)}\boldsymbol{\phi}(\mathbf{W}^{(L-1)}\mathbf{a}^{(L-1)})$

$$\underset{S^L \times (S^{L-1}+1)}{\mathbf{W}^{(L)}} = \begin{bmatrix} w^{(L)}_{1,0} & w^{(L)}_{1,1} & w^{(L)}_{1,2} & \ldots & w^{(L)}_{1,S^{L-1}} \\ & & \ldots & & \\ w^{(L)}_{S^L,0} & w^{(L)}_{S^L,1} & \ldots & & w^{(L)}_{S^L,S^{L-1}} \end{bmatrix}$$

notation adapted from *Neural Network Design*, Hagan, Demuth, Beale, and De Jesus

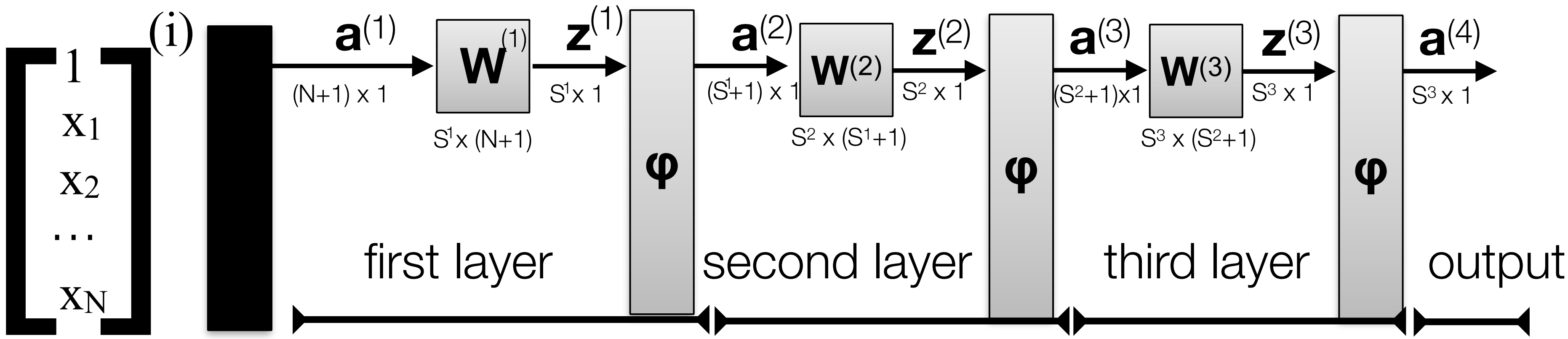
Multiple layers notation



- **Self test**: How many parameters need to be trained in the above network?
  - A. $[(N+1) \times S^1] + [(S^1+1) \times S^2] + [(S^2+1) \times S^3]$
  - B. $|\mathbf{W}^{(1)}| + |\mathbf{W}^{(2)}| + |\mathbf{W}^{(3)}|$
  - C. can't determine from diagram
  - D. it depends on the sizes of intermediate variables, $\mathbf{z}^{(i)}$

notation adapted from *Neural Network Design*, Hagan, Demuth, Beale, and De Jesus

Compact feedforward notation



$$\mathbf{z}^{(L)}=\mathbf{W}^{(L)}\mathbf{a}^{(L)}$$

$$[\mathbf{z}^{(L)}]^{(i)}=\mathbf{W}^{(L)}[\mathbf{a}^{(L)}]^{(i)}$$

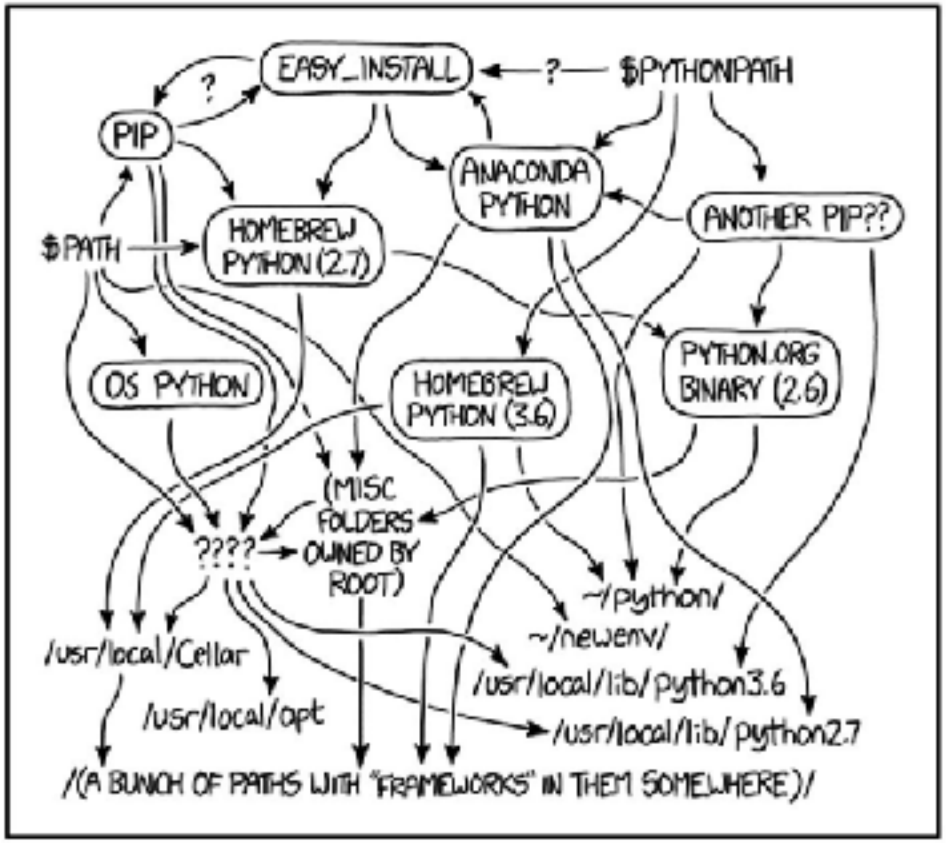
$$\begin{bmatrix} z^{(L)}_1 \\ \vdots \\ z^{(L)}_{S^L}\end{bmatrix}^{(i)} = \begin{bmatrix} w^{(L)}_{1,1} & w^{(L)}_{1,2} & w^{(L)}_{1,3} & \dots & w^{(L)}_{1,S^{L-1}+1} \\ & & \vdots & & \\ w^{(L)}_{S^L,1} & w^{(L)}_{S^L,2} & \dots & & w^{(L)}_{S^L,S^{L-1}+1}\end{bmatrix}\begin{bmatrix} a^{(L)}_1 \\ \vdots \\ a^{(L)}_{S^{L-1}+1}\end{bmatrix}^{(i)}$$

$$\left[\begin{bmatrix} z^{(L)}_1 \\ \vdots \\ z^{(L)}_{S^L}\end{bmatrix}^{(1)} \dots \begin{bmatrix} z^{(L)}_1 \\ \vdots \\ z^{(L)}_{S^L}\end{bmatrix}^{(M)}\right] = [\mathbf{W}^{(L)}]\left[\begin{bmatrix} a^{(L)}_1 \\ \vdots \\ a^{(L)}_{S^{L-1}+1}\end{bmatrix}^{(1)} \dots \begin{bmatrix} a^{(L)}_1 \\ \vdots \\ a^{(L)}_{S^{L-1}+1}\end{bmatrix}^{(M)}\right]$$

$$\mathbf{Z}^{(L)}=\mathbf{W}^{(L)}\mathbf{A}^{(L)}$$

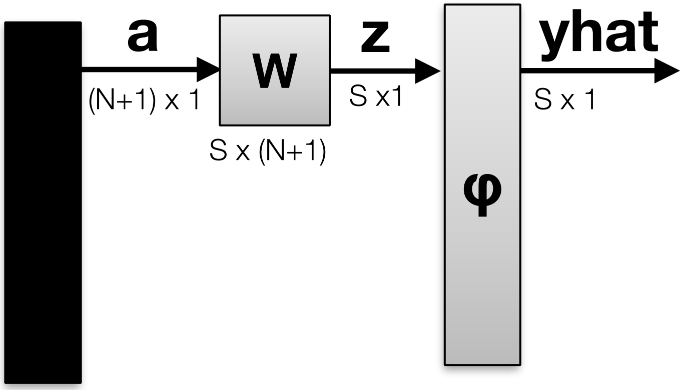
$$\mathbf{Z}^{(L)}=\mathbf{W}^{(L)}\boldsymbol{\phi}(\mathbf{W}^{(L-1)}\mathbf{A}^{(L-1)})$$

Training Neural Network Architectures



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Rosenblatt’s Perceptron, 1957



$$\sum_{i}^{M}(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)})^2$$

Need objective Function, minimize MSE $J(\mathbf{W}) = ||\mathbf{Y} - \hat{\mathbf{Y}}||^2$

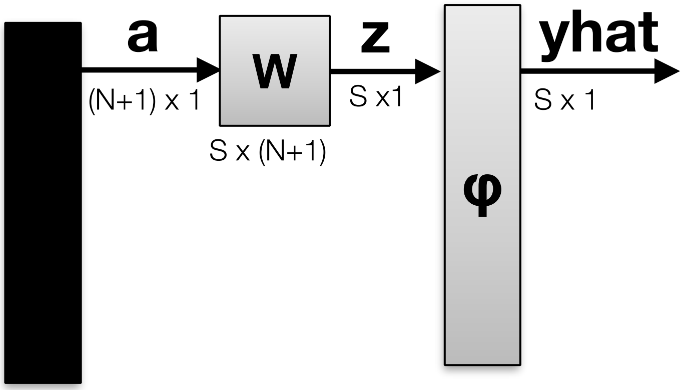
where ground truth $\mathbf{y}^{(i)}$ is one-hot encoded!

$$\begin{bmatrix} y_1 \\ \dots \\ y_C \end{bmatrix}^{(i)}$$

$$\left[\begin{bmatrix} y_1 \\ \dots \\ y_C \end{bmatrix}^{(1)} \cdots \begin{bmatrix} y_1 \\ \dots \\ y_C \end{bmatrix}^{(M)} \right] = \mathbf{Y}$$

Simple Architectures

- Rosenblatt's perceptron, 1957

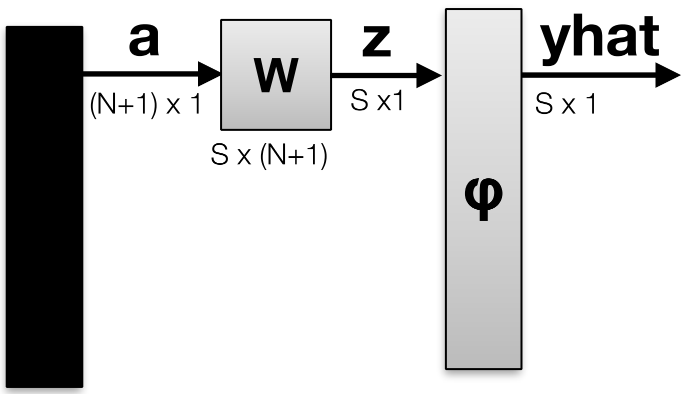


Self Test - If this is a binary classification problem, how large is *S*, the length of yhat?

A. Can't determine
B. 2
C. 1
D. N

Simple Architectures

- Adaline network, Widrow and Hoff, 1960



linear

`phi`

Marcian “Ted” Hoff

Bernard Widrow

Objective Function, minimize MSE $J(\mathbf{W}) = ||\mathbf{Y} - \hat{\mathbf{Y}}||^2$

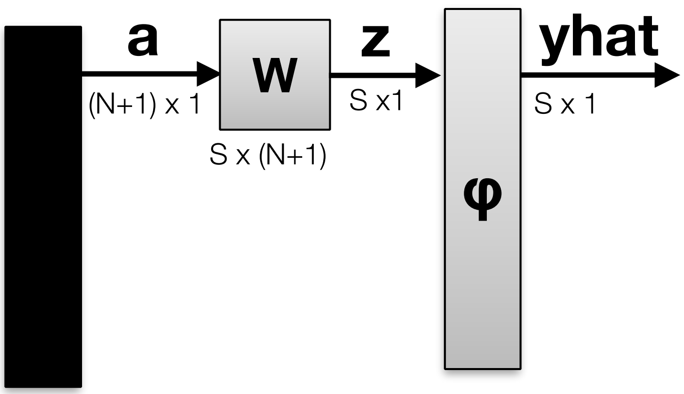
New objective function becomes: $J(\mathbf{W}) = ||\mathbf{Y} - \mathbf{W} \cdot \mathbf{A}||^2$

Need gradient $\nabla J(\mathbf{W})$ for update equation $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla J(\mathbf{W})$

We have been using the **Widrow-Hoff Learning Rule**

Simple Architectures

- Adaline network, Widrow and Hoff, 1960



linear

phi

Marcian “Ted” Hoff

Bernard Widrow

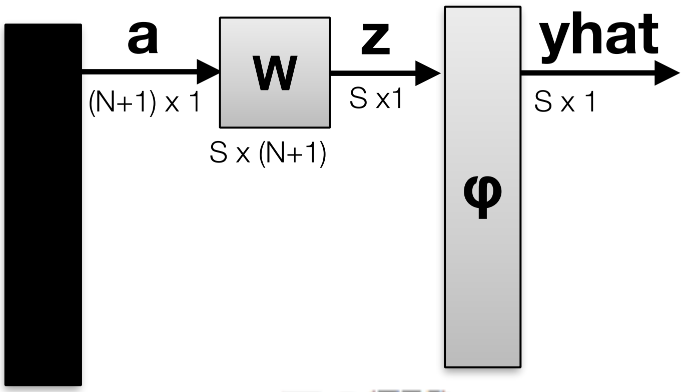
need gradient $\nabla J(\mathbf{W})$ for update equation $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla J(\mathbf{W})$

For case S=1, $\mathbf{W}$ has only one row, $\mathbf{w}$
this is just **linear regression…**

$$\mathbf{w} \leftarrow \mathbf{w} + \eta[\mathbf{X} * (\mathbf{y} - \hat{\mathbf{y}})]$$

Simple Architectures

- Modern Perceptron network



sigmoid

$$\text{phi} = \frac{1}{1+\exp(-z)}$$

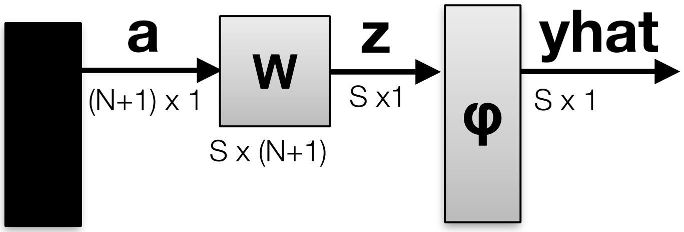
need gradient $\nabla J(\mathbf{W})$ for update equation $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla J(\mathbf{W})$

For case S=1, this is just **logistic regression…**
and **we have already solved this**!

$$\mathbf{w} \leftarrow \mathbf{w} + \eta[\mathbf{X} * (\mathbf{y} - \mathbf{g}(\mathbf{x}))]$$

What happens when S > 1 ?

What if we have more than S=1?



$$\left[\begin{bmatrix}\phi(z_1)\\ \dots\\ \phi(z_S)\end{bmatrix}^{(1)} \cdots \begin{bmatrix}\phi(z_1)\\ \dots\\ \phi(z_S)\end{bmatrix}^{(M)}\right] = \hat{\mathbf{Y}}$$

$$\left[\begin{bmatrix}y_1\\ \dots\\ y_S\end{bmatrix}^{(1)} \cdots \begin{bmatrix}y_1\\ \dots\\ y_S\end{bmatrix}^{(M)}\right] = \mathbf{Y}$$

$$J(\mathbf{W}) = ||\mathbf{Y} - \hat{\mathbf{Y}}||^2$$

Each target class in $\mathbf{Y}$ can be independently optimized

$$\mathbf{yhat}^{(i)} = \begin{bmatrix}\varphi({}_{row=1}\mathbf{w}\cdot\mathbf{x}^{(i)})\\ \varphi({}_{row=2}\mathbf{w}\cdot\mathbf{x}^{(i)})\\ \dots\\ \varphi({}_{row=S}\mathbf{w}\cdot\mathbf{x}^{(i)})\end{bmatrix}$$

one hot

which is one-versus-all!

$$J({}_1\mathbf{w}) = \sum_{i=1}[y_1{}^{(i)} - \varphi({}_1\mathbf{w}\cdot\mathbf{x}^{(i)})]^2$$

$$J({}_2\mathbf{w}) = \sum_{i=1}[y_2{}^{(i)} - \varphi({}_2\mathbf{w}\cdot\mathbf{x}^{(i)})]^2$$

...

$$J({}_S\mathbf{w}) = \sum_{i=1}[y_S{}^{(i)} - \varphi({}_S\mathbf{w}\cdot\mathbf{x}^{(i)})]^2$$

Simple Architectures: Summary

- Adaline network, Widrow and Hoff, 1960
  - linear regression
- Perceptron
  - *with sigmoid*: logistic regression
- One-versus-all implementation is the same as having $\mathbf{w}_{class}$ be rows of weight matrix, $\mathbf{W}$
  - works in adaline
  - works in logistic regression

these networks were created in the 50's and 60's
but were abandoned

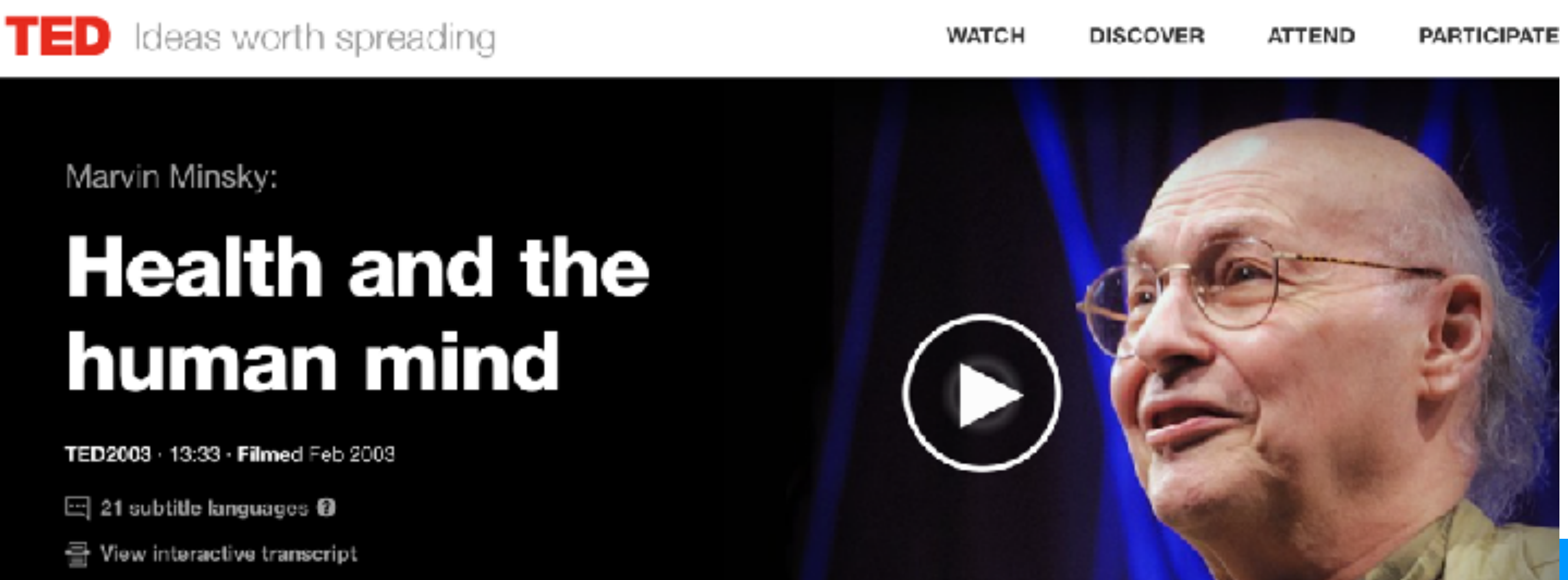
why were they not used?

The Rosenblatt-Widrow-Hoff Dilemma

- 1960's: Rosenblatt got into a public academic argument with Marvin Minsky and Seymour Papert

> "Given an elementary α-perceptron, a stimulus world W, and any classification C(W) for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to C(W) in finite time…"

- Minsky and Papert publish limitations paper, 1969:



More Advanced Architectures: history

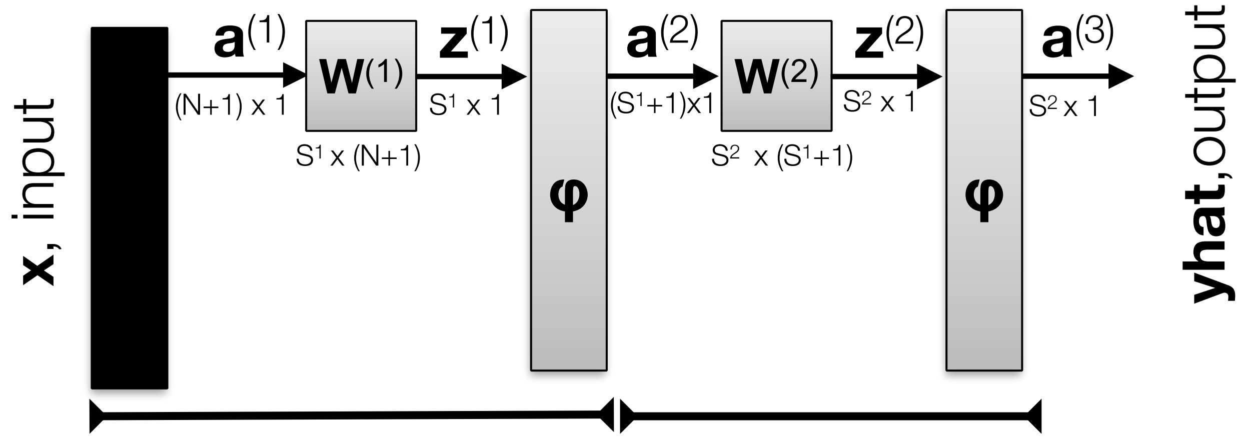
- 1986: *Rumelhart*, *Hinton*, and *Williams* popularize gradient calculation for multi-layer network
  - *technically* introduced by Werbos in 1982
- **difference**: Rumelhart *et al.* validated ideas with a computer
- until this point no one could train a multiple layer network consistently
- algorithm is popularly called **Back-Propagation**
- wins pattern recognition prize in 1993, becomes de-facto machine learning algorithm until: SVMs and Random Forests in ~2004
- would eventually see a resurgence for its ability to train algorithms for Deep Learning applications: **Hinton is widely considered the founder of deep learning**

David Rumelhart

Geoffrey Hinton

More Advanced Architectures: MLP

- The multi-layer perceptron (MLP):
  - two layers shown, but could be arbitrarily many layers
  - algorithm is agnostic to number of layers (*kinda*)



first layer

second layer
output layer

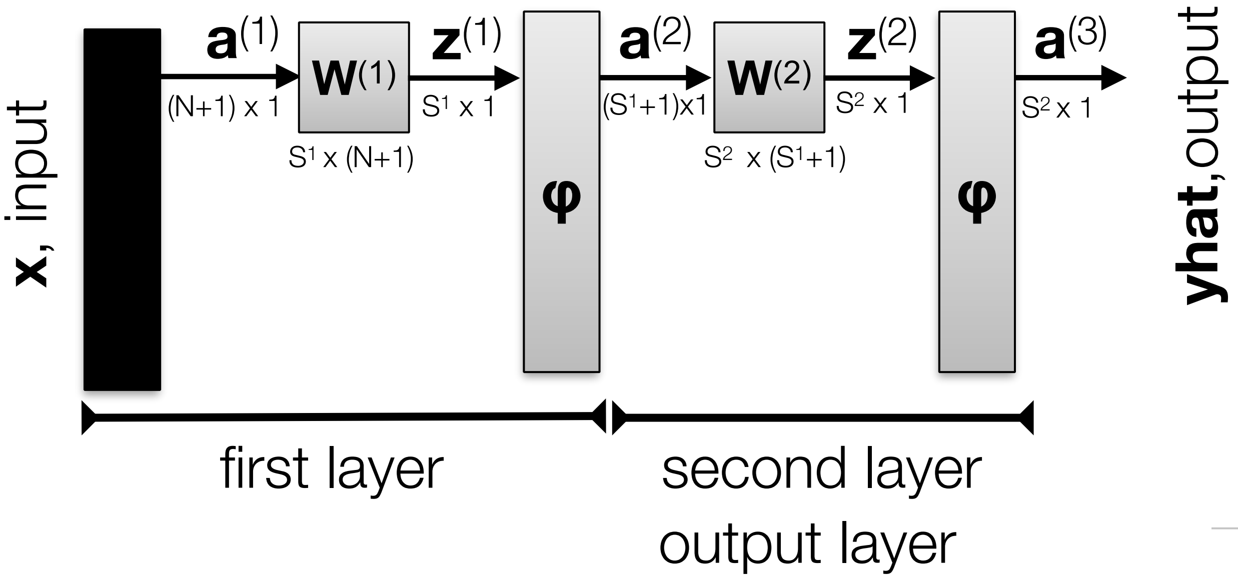
each row of **yhat** is no longer independent of the rows in **W** so we cannot optimize using one versus all!!!

$$\mathbf{yhat}^{(i)} = \begin{bmatrix} \varphi\left({}_{\text{row}=1}\mathbf{w}^{(2)} \cdot \varphi(\mathbf{W}^{(1)}\mathbf{a}^{(1)})\right) \\ \cdots \\ \varphi\left({}_{\text{row}=S}\mathbf{w}^{(2)} \cdot \varphi(\mathbf{W}^{(1)}\mathbf{a}^{(1)})\right) \end{bmatrix}$$

one hot

Back propagation

- Steps:
  - propagate weights forward
  - calculate gradient at final layer
  - back propagate gradient for each layer
    - via recurrence relation

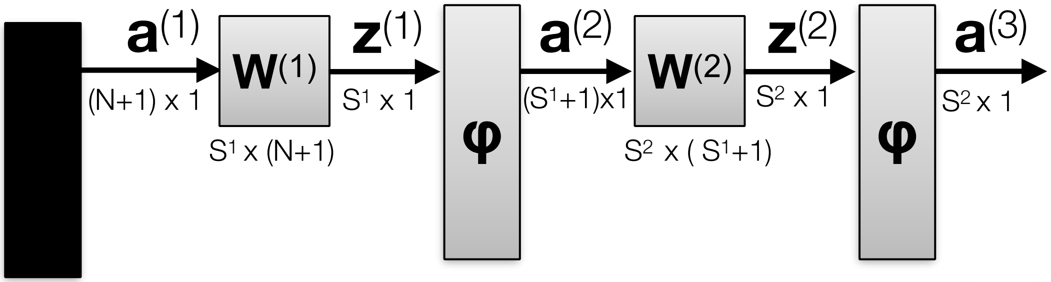


$$J(\mathbf{W}) = \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2$$

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \eta \frac{\partial J(\mathbf{W})}{\partial w_{i,j}^{(l)}}$$

Back propagation

$$J(\mathbf{W}) = \sum_{k}^{M} (\mathbf{y}^{(k)} - \mathbf{a}^{(L)})^2$$



$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \eta \boxed{\frac{\partial J(\mathbf{W})}{\partial w_{i,j}^{(l)}}}$$

use chain rule: $\dfrac{\partial J(\mathbf{W})}{\partial w_{i,j}^{(l)}} = \dfrac{\partial J(\mathbf{W})}{\partial \mathbf{z}^{(l)}} \boxed{\dfrac{\partial \mathbf{z}^{(l)}}{\partial w_{i,j}^{(l)}}}$

Solve this in explainer video for next in class assignment!

Lab 3, Town Hall (if time)

End of Session

- thanks! **Next time is Flipped Assignment!!!**

More help on neural networks to prepare for next time:

Sebastian Raschka
https://github.com/rasbt/python-machine-learning-book/blob/master/code/ch12/ch12.ipynb

Martin Hagan
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwioprvn27fPAhWMx4MKHYbwDlwQFggeMAA&url=http%3A%2F%2Fhagan.okstate.edu%2FNNDesign.pdf&usg=AFQjCNG5YbM4xSMm6K5HNsG-4Q8TvOu_Lw&sig2=bgT3k-5ZDDTPZ07Qu8Oreg

Michael Nielsen
http://neuralnetworksanddeeplearning.com