Lecture Notes for

Machine Learning in Python

[👨‍🏫, 👨‍💻, 🐍, 👨‍🔬]

Professor Eric Larson

MLP History

Class Logistics and Agenda

- Logistics:
  - Grading Update (Lab 3 due this weekend)
  - **Next time: Flipped Module on back propagation (work on your own time)**
- Multi Week Agenda:
  - Today: Neural Networks History, up to 1980 and Multi-layer Architectures
  - **Flipped**: Programming Multi-layer training
  - More Neural Networks,
  - Town Hall, Lab 4 (after flipped)
  - **Flipped**: Cross Validation

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

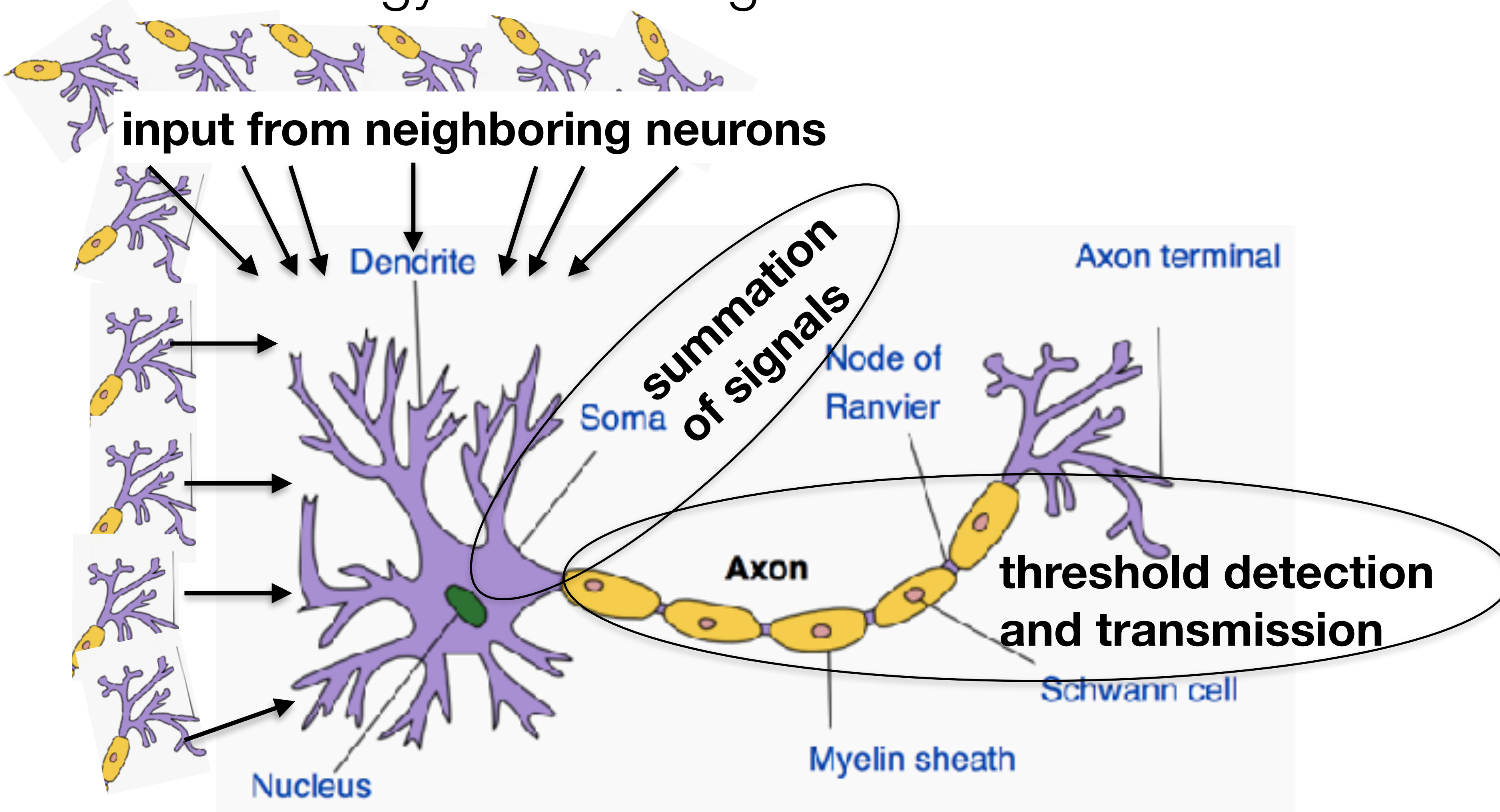
Ethics in Language Models
ConceptNet
Case studies

Lab 3, Town Hall (if needed)

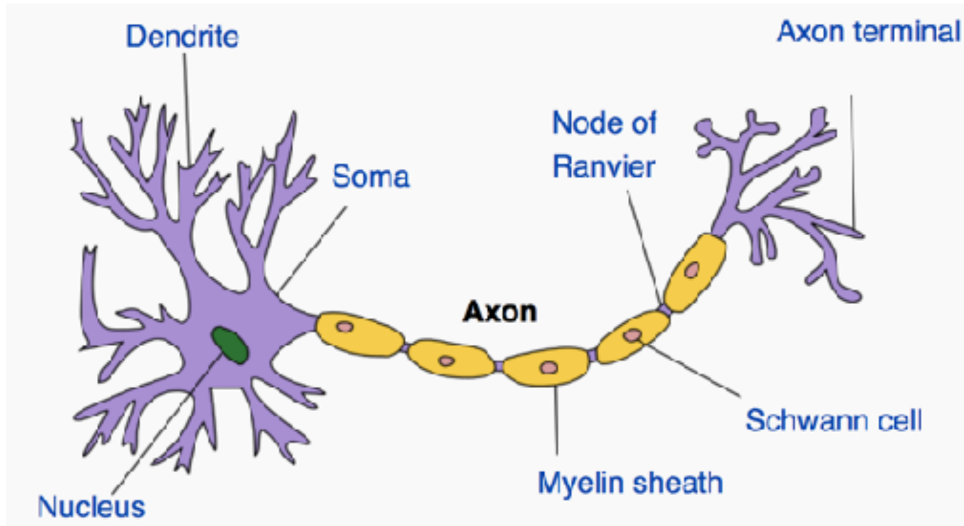
A History of Neural Networks

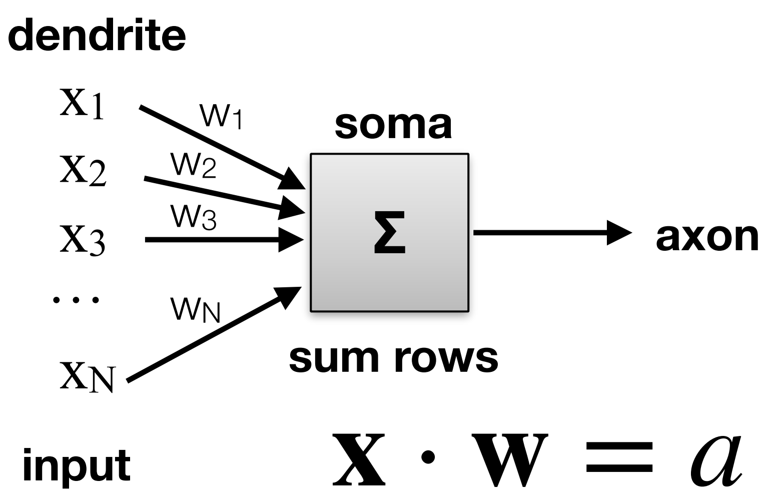
Neurons

- From biology to modeling:



McCulloch and Pitts, 1943





$$\mathbf{x} \cdot \mathbf{w} = a$$

for each neuron

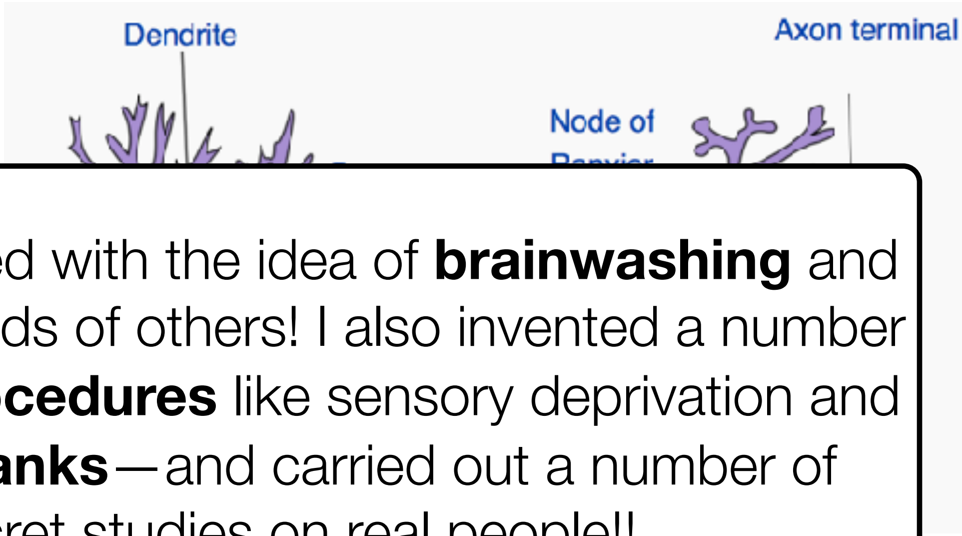
logic gates of the mind

Warren McCulloch

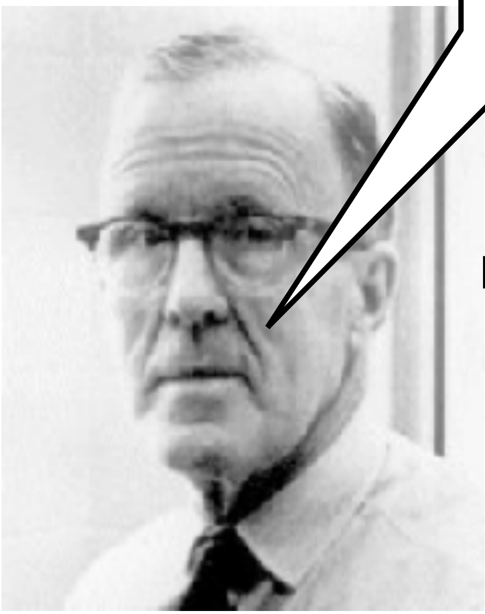
Walter Pitts

Neurons

- McCulloch and Pitts, 1943
- Donald Hebb, 1949
    - Hebb’s Law: close neurons fire together
    - neurons “lear
    - easier synapti
    - basis of neura



I was 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!!



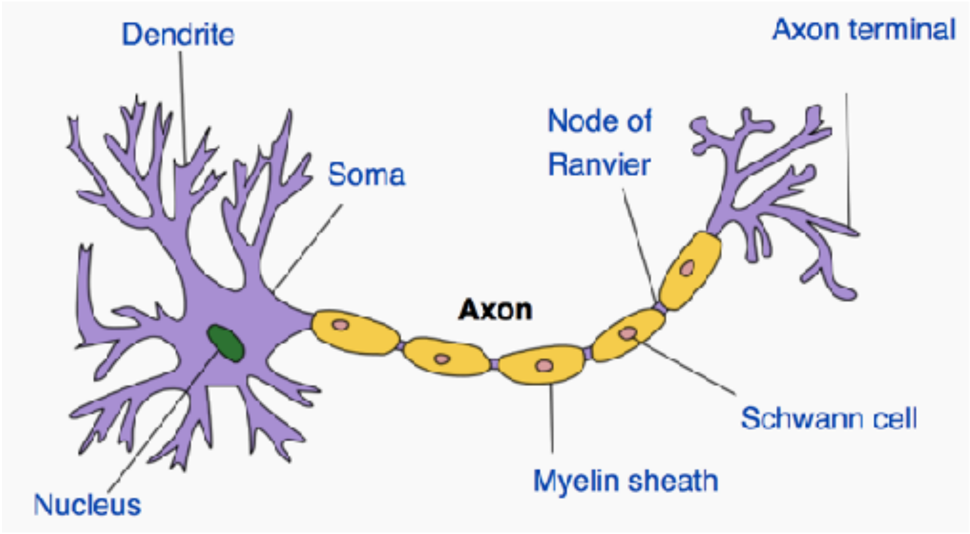
Donald O. Hebb

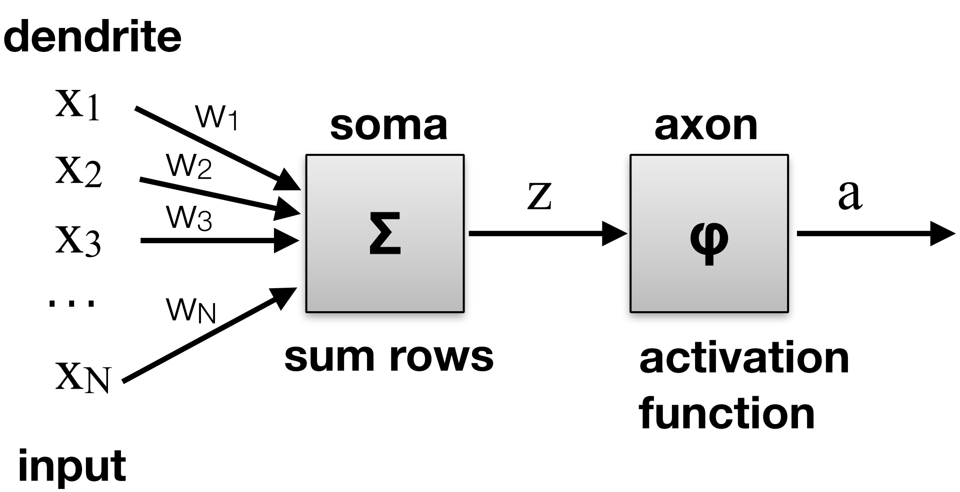
Warren McCulloch

Walter Pitts

Rosenblatt's perceptron, 1957

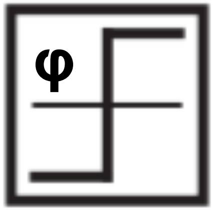
Frank Rosenblatt





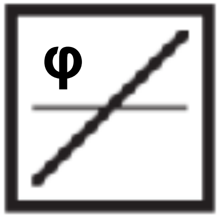
Axon Functions

hard limit



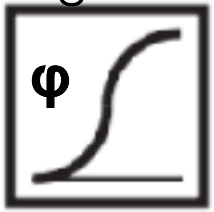
a=−1 z< 0
a= 1 z>=0

linear



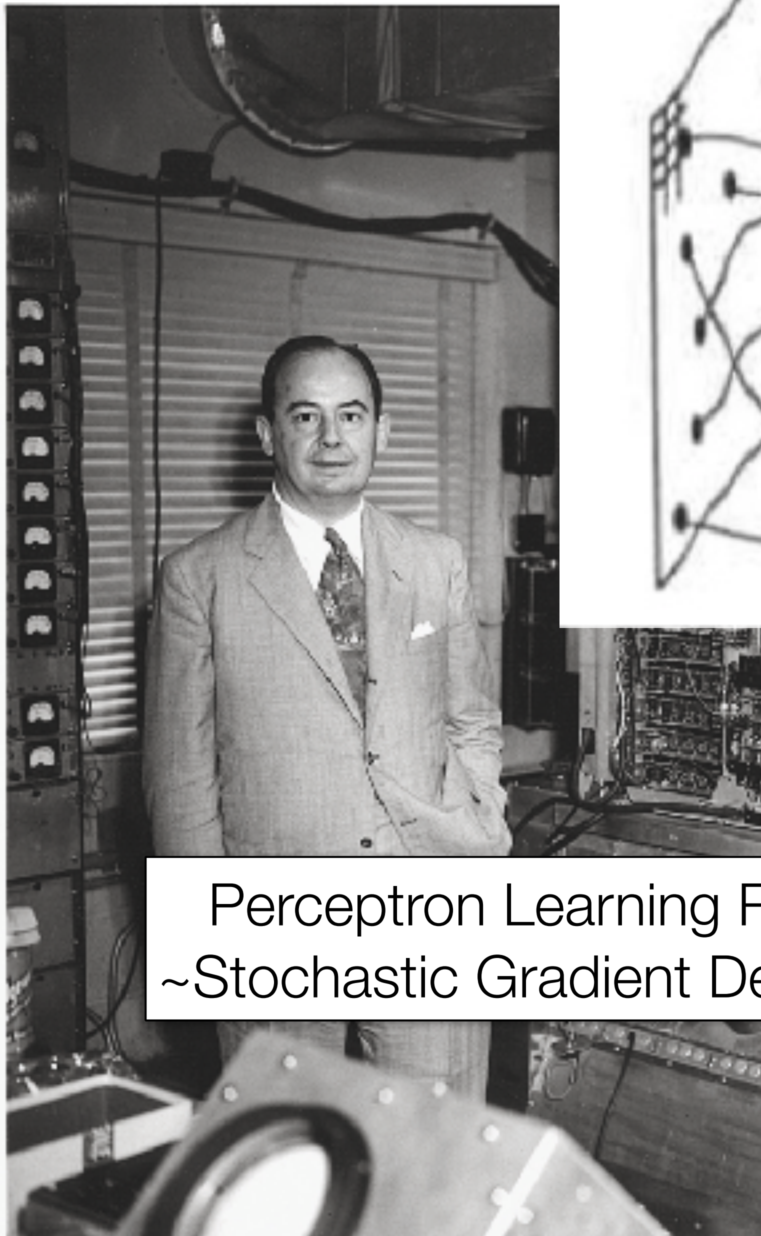
a=z

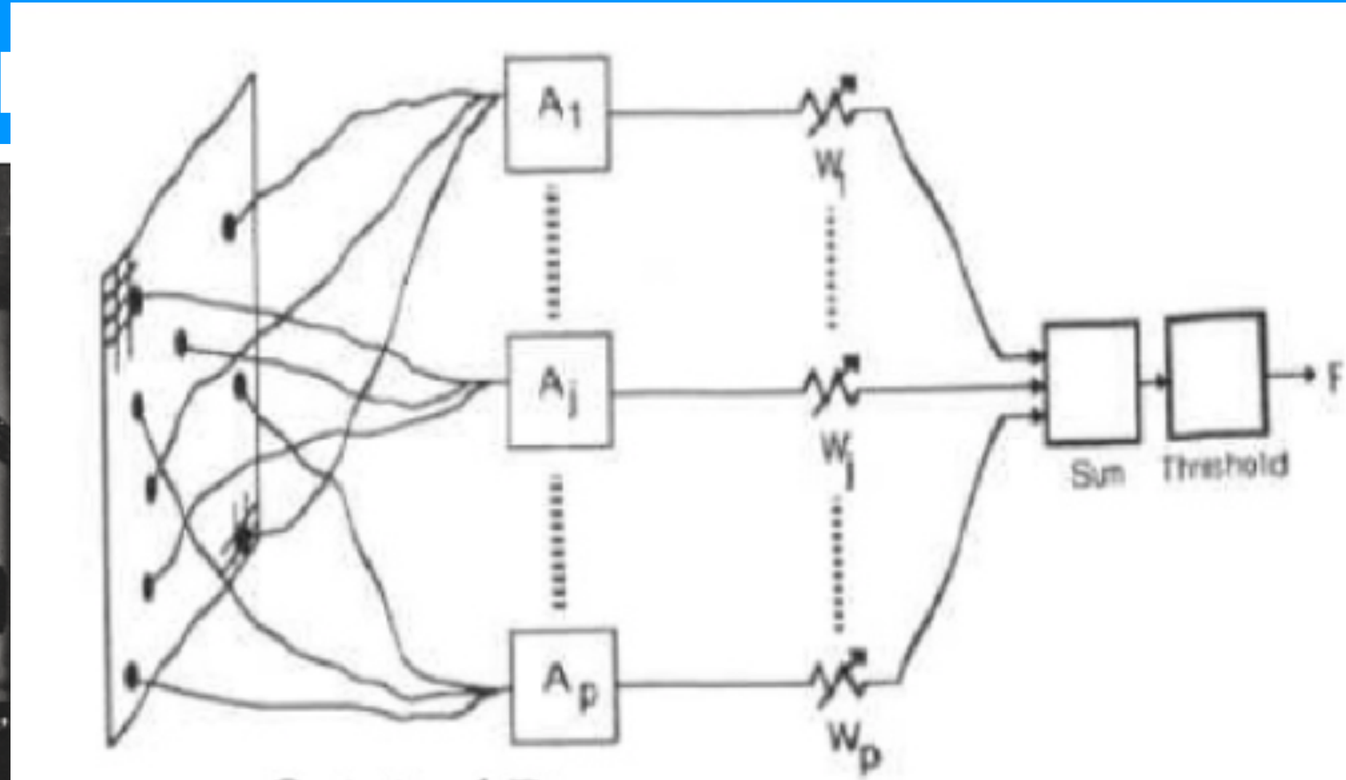
sigmoid

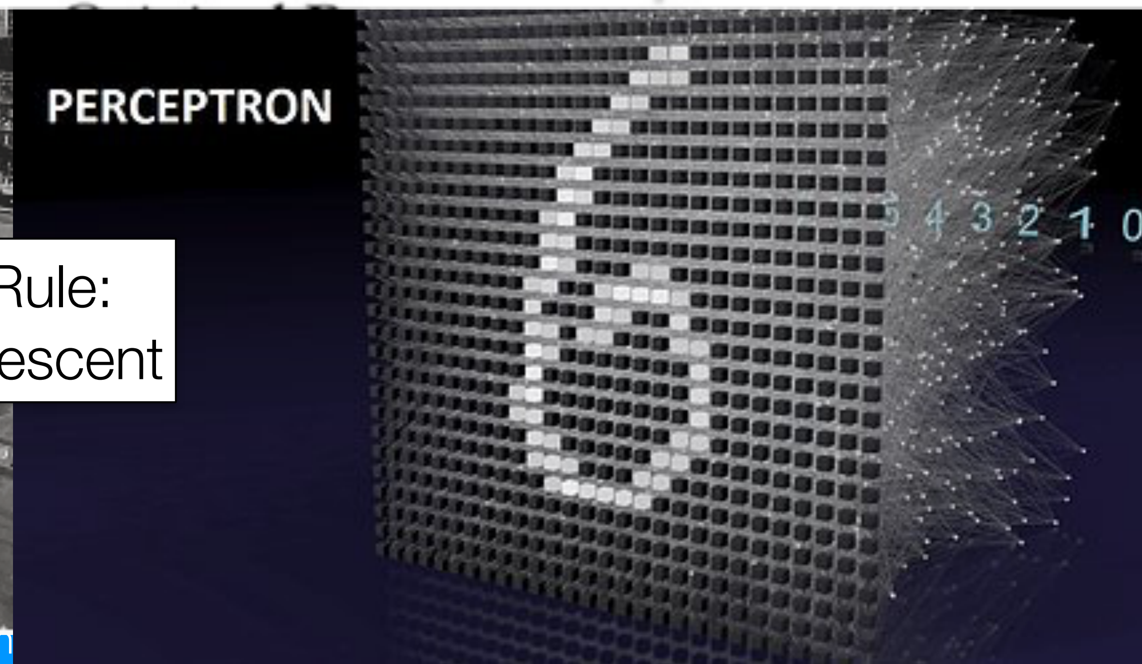


$$a = \frac{1}{1+\exp(-z)}$$

The Mark 1



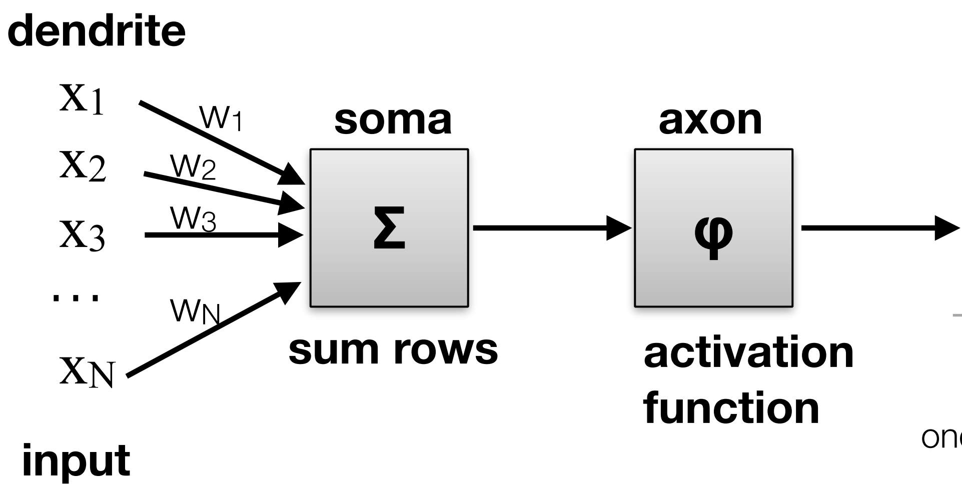


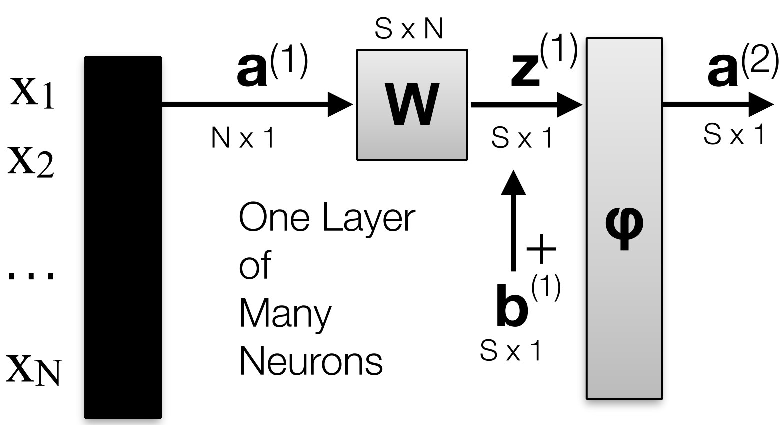


Perceptron Learning Rule:
~Stochastic Gradient Descent

Layers Notation for Table Data

One Neuron





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

a=x

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

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x}^{(i)} + \mathbf{b}$$

one neuron →

$$\mathbf{W} = \begin{bmatrix} \boxed{w_{1,1} \quad \dots \quad w_{1,N}} \\ w_{2,1} \quad \dots \quad w_{2,N} \\ \vdots \\ w_{S,1} \quad \dots \quad w_{S,N} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \boxed{b_1} \\ b_2 \\ \vdots \\ b_S \end{bmatrix}$$

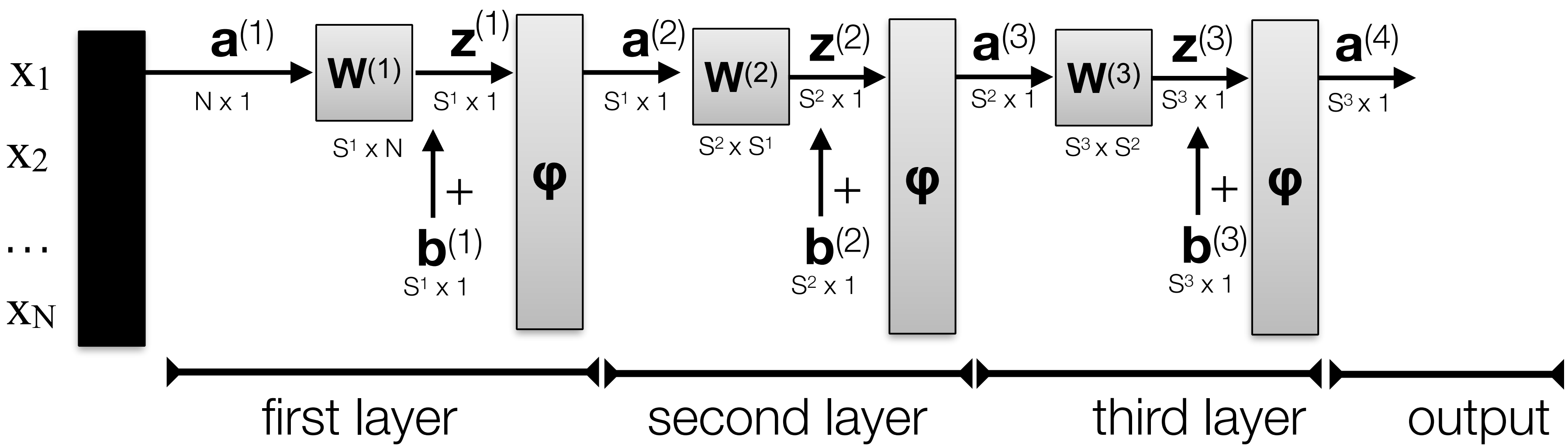
$$[\mathbf{z}^{(1)}]^{(i)} = \mathbf{W}^{(1)} \cdot [\mathbf{a}^{(1)}]^{(i)} + \mathbf{b}^{(1)}$$

$$\mathbf{a}^{next} = \phi(\mathbf{z}^{\mathbf{current}})$$

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

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

Generic Multiple Layers Notation



$$\mathbf{a}^{(L+1)} = \phi(\mathbf{z}^{(L)})$$

$\mathbf{a}^{(final)}$ size=unique classes, C

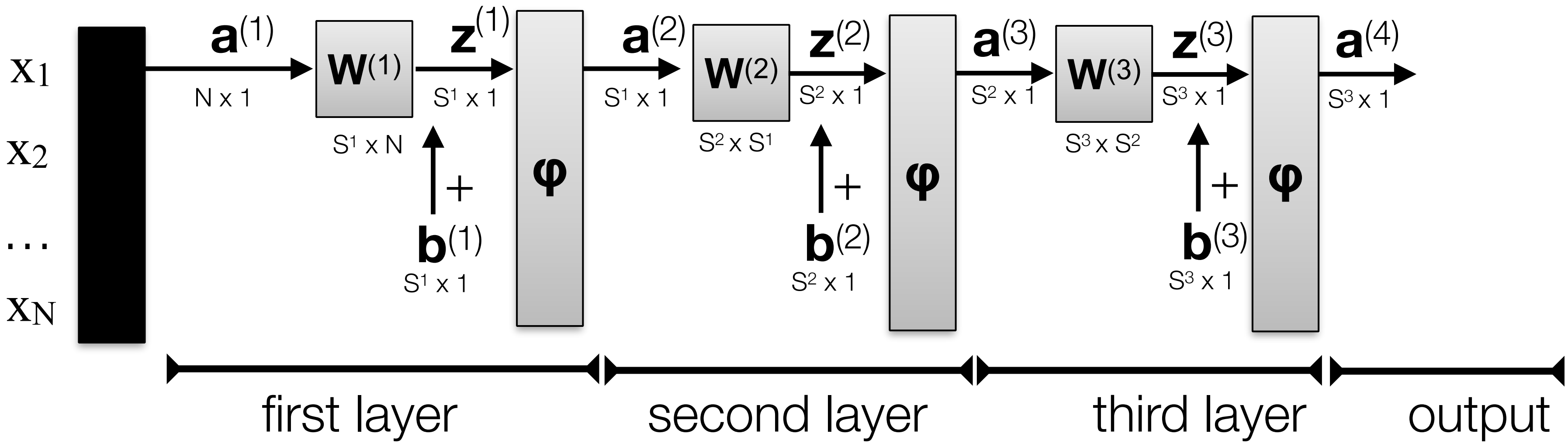
$$\mathbf{z}^{(L)} = \mathbf{W}^{(L)} \cdot \mathbf{a}^{(L)} + \mathbf{b}^{(L)}$$

$$\mathbf{z}^{(L)} = \mathbf{W}^{(L)} \cdot \phi(\mathbf{z}^{(L-1)}) + \mathbf{b}^{(L)}$$

$$\mathbf{z}^{(L)} = \mathbf{W}^{(L)} \cdot \phi\left(\mathbf{W}^{(L-1)} \cdot \phi(\mathbf{z}^{(L-2)}) + \mathbf{b}^{(L-1)}\right) + \mathbf{b}^{(L)}$$

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

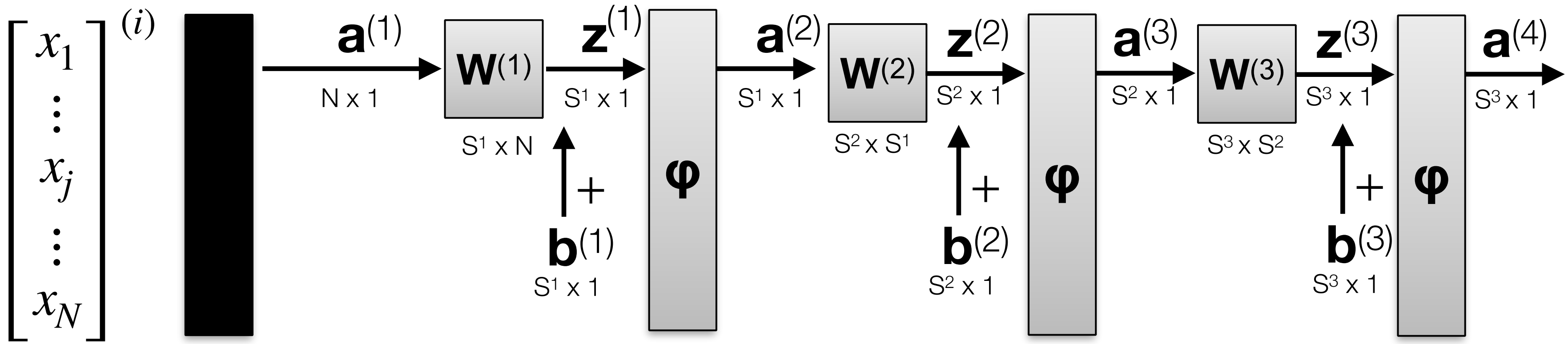
Multiple layers notation



- **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. $|\mathbf{W}^{(1)}| + |\mathbf{W}^{(2)}| + |\mathbf{W}^{(3)}| + |\mathbf{b}^{(1)}| + |\mathbf{b}^{(2)}| + |\mathbf{b}^{(3)}|$
  - 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

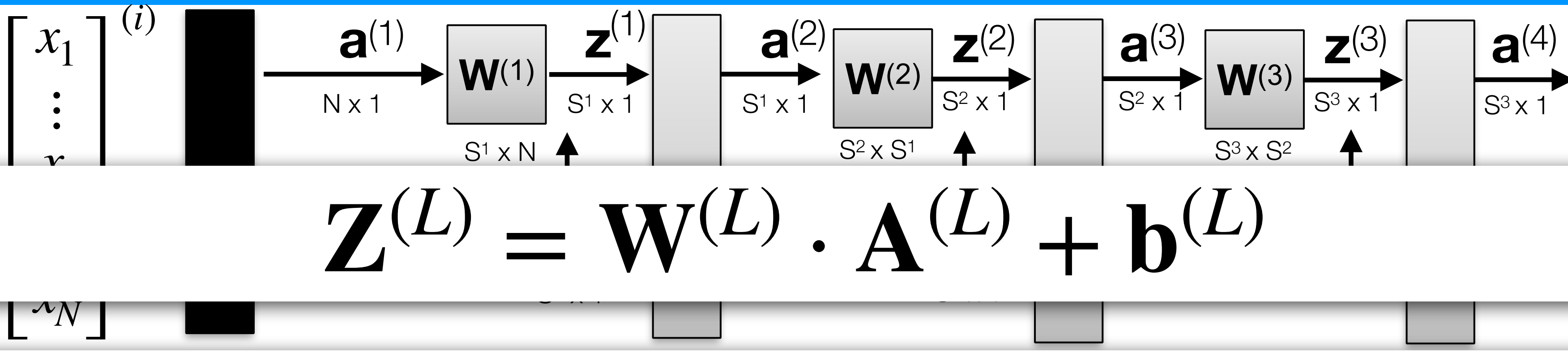
Compact feedforward notation



$$\mathbf{X}^T = \left[\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{(1)}, \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{(2)} \cdots \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{(M)} \right] = \left[\begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_N^{(1)} \end{bmatrix}^{(1)}, \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_N^{(1)} \end{bmatrix}^{(2)} \cdots \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_N^{(1)} \end{bmatrix}^{(M)} \right] = \mathbf{A}^{(1)}$$

Table Data | Table Data, in Neural Net Notation

Compact feedforward notation



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

$$\mathbf{Z}^{(L)} = \mathbf{W}^{(L)} \cdot \phi(\mathbf{Z}^{(L-1)}) + \mathbf{b}^{(L)}$$

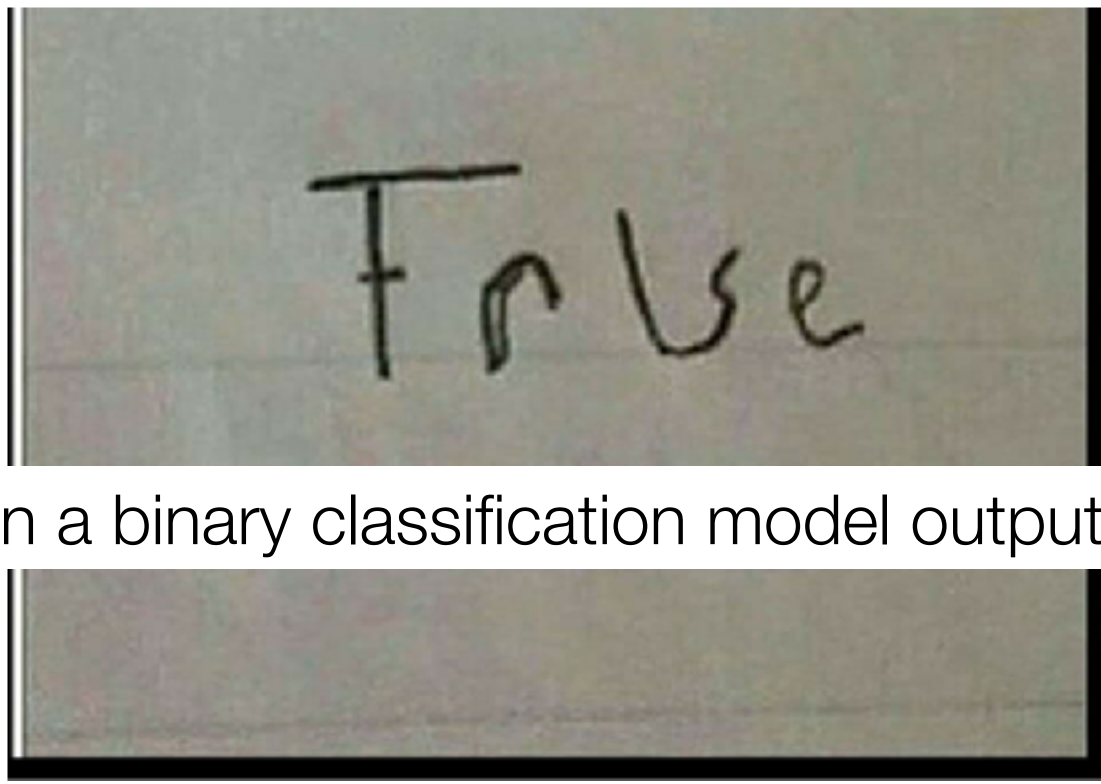
$$\left[\mathbf{z}^{(L)}\right]^{(i)} = \mathbf{W}^{(L)} \cdot \left[\mathbf{a}^{(L)}\right]^{(i)} + \mathbf{b}^{(L)}$$

$$\begin{bmatrix} z_2^{(L)} \\ \vdots \\ z_{S^L}^{(L)} \end{bmatrix} = \mathbf{W}^{(L)} \cdot \begin{bmatrix} a_1^{(L)} \\ \vdots \\ a_{S^{L-1}}^{(L)} \end{bmatrix} + \mathbf{b}^{(L)}$$

$$\left[\begin{bmatrix} z_1^{(L)} \\ z_2^{(L)} \\ \vdots \\ z_{S^L}^{(L)} \end{bmatrix}^{(1)}, \begin{bmatrix} z_1^{(L)} \\ z_2^{(L)} \\ \vdots \\ z_{S^L}^{(L)} \end{bmatrix}^{(2)} \cdots \begin{bmatrix} z_1^{(L)} \\ z_2^{(L)} \\ \vdots \\ z_{S^L}^{(L)} \end{bmatrix}^{(M)} \right] = \mathbf{W}^{(L)} \cdot \left[\begin{bmatrix} a_0^{(L)} \\ a_1^{(L)} \\ \vdots \\ a_{S^{L-1}}^{(L)} \end{bmatrix}^{(1)}, \begin{bmatrix} a_0^{(L)} \\ a_1^{(L)} \\ \vdots \\ a_{S^{L-1}}^{(L)} \end{bmatrix}^{(2)} \cdots \begin{bmatrix} a_0^{(L)} \\ a_1^{(L)} \\ \vdots \\ a_{S^{L-1}}^{(L)} \end{bmatrix}^{(M)} \right] + \mathbf{b}^{(L)}$$

$\mathbf{b}$ is broadcast added

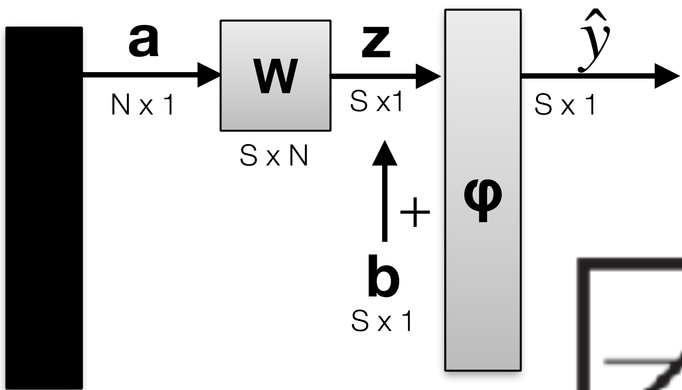
Historical Training of Neural Network Architectures



When a binary classification model outputs 0.5

One Layer Linear Architectures

- Adaline network, Widrow and Hoff, 1960



==scalar, continuous output

$$\hat{\mathbf{Y}} = \phi(\mathbf{A} \cdot \mathbf{w}) = \mathbf{A} \cdot \mathbf{w}$$

linear

Marcian “Ted” Hoff

Bernard Widrow

Simplify Objective Function:

$$J(\mathbf{W}) = \left\| \mathbf{Y} - \hat{\mathbf{Y}} \right\|^2 \longrightarrow J(\mathbf{w}) = \left\| \mathbf{Y} - \mathbf{A} \cdot \mathbf{w} \right\|^2$$

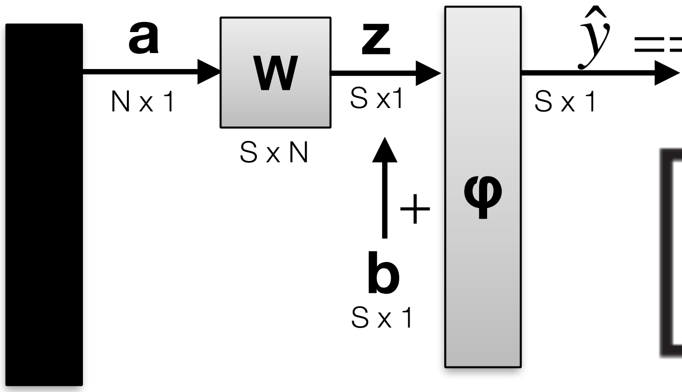
Need gradient $\nabla J(\mathbf{w})$ for update equation $\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla J(\mathbf{w})$

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

One Layer Linear Architectures

- Adaline network, Widrow and Hoff, 1960

Marcian “Ted” Hoff



$\hat{y}$ == scalar, continuous output

$$\hat{\mathbf{Y}} = \mathbf{A} \cdot \mathbf{w}$$

linear

Bernard Widrow

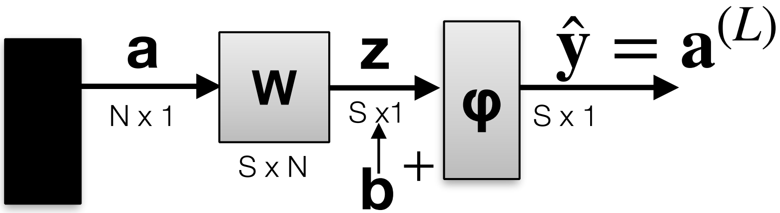
Need gradient $\nabla J(\mathbf{w})$ for update equation $\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla J(\mathbf{w})$

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

$$J(\mathbf{w}) = \sum_{i=1}^{M} (y^{(i)} - \mathbf{x}^{(i)} \cdot \mathbf{w})^2$$

$$\mathbf{w} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot y$$

From Regression to Classification



No longer regression, $\mathbf{Y}$ is a category

ground truth $\mathbf{Y}$ is one-hot encoded!

Ground Truth Labels

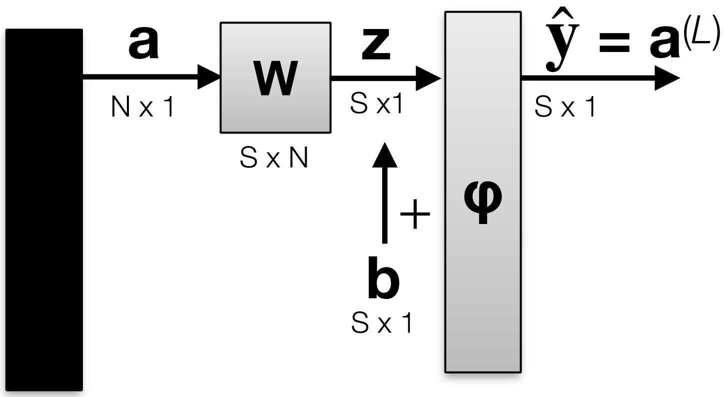
$$\mathbf{Y} = \left[\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(1)} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(2)} \cdots \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(M)} \right] \rightarrow \left[\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^{(1)} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}^{(2)} \cdots \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}^{(M)} \right]$$

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

$$J(\mathbf{W}) = \left| \underbrace{\left[\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(1)} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(2)} \cdots \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(M)} \right]}_{\mathbf{Y}} - \underbrace{\left[\begin{bmatrix} a_1^{(L)} \\ a_2^{(L)} \\ \vdots \\ a_C^{(L)} \end{bmatrix}^{(1)} \begin{bmatrix} a_1^{(L)} \\ a_2^{(L)} \\ \vdots \\ a_C^{(L)} \end{bmatrix}^{(2)} \cdots \begin{bmatrix} a_1^{(L)} \\ a_2^{(L)} \\ \vdots \\ a_C^{(L)} \end{bmatrix}^{(M)} \right]}_{\hat{\mathbf{Y}}} \right|^2$$

One Layer Classification

- Rosenblatt's perceptron, 1957

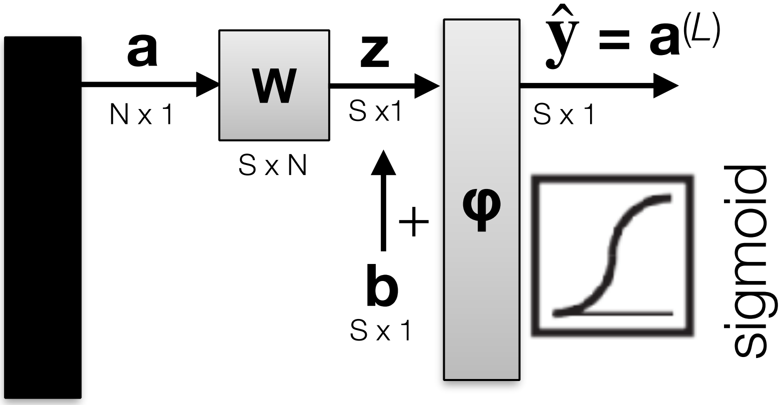


Self Test - If this is a binary classification problem, how large is *S*, the length of $\hat{\mathbf{y}}$ and number of rows in **W**?

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

One Layer Classification

- Modern Perceptron network



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

$$g(\mathbf{w} \cdot \mathbf{x}) = \phi(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})}$$

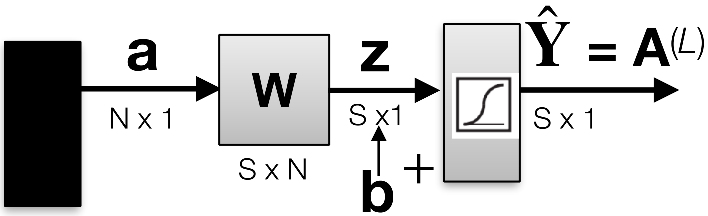
Need gradient $\nabla J(\mathbf{w})$ for update equation $\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla J(\mathbf{w})$

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

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \text{mean}\left(\underbrace{(\mathbf{y} - g(\mathbf{X} \cdot \mathbf{w}))}_{\mathbf{y}_{diff}} \odot \mathbf{X} \right)_{cols}$$

What happens when S > 1 ?

One Layer Architectures of Many Classes



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

$$J(\mathbf{W}) = \left\| \mathbf{Y} - \phi\left(\mathbf{W} \cdot \mathbf{X}^T\right) \right\|^2$$

$$J(\mathbf{w}_{row=1}) = \sum_i \left(y_1^{(i)} - \phi\left(\mathbf{w}_{row=1} \cdot \mathbf{x}^{(i)}\right) \right)^2$$

… for each class/row …

$$J(\mathbf{w}_{row=C}) = \sum_i \left(y_C^{(i)} - \phi\left(\mathbf{w}_{row=C} \cdot \mathbf{x}^{(i)}\right) \right)^2$$

$$\mathbf{Y} = \left[\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(1)} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(2)} \cdots \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_C \end{bmatrix}^{(M)} \right] \rightarrow \left[\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^{(1)} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}^{(2)} \cdots \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}^{(M)} \right]$$

Each target class and row of $\mathbf{W}$ can be independently optimized

$$\hat{\mathbf{Y}} = \left[\begin{bmatrix} \phi(\mathbf{w}_{row=1} \cdot \mathbf{x}^{(1)}) \\ \phi(\mathbf{w}_{row=2} \cdot \mathbf{x}^{(1)} \cdot) \\ \vdots \\ \phi(\mathbf{w}_{row=C} \cdot \mathbf{x}^{(1)}) \end{bmatrix}^{(1)}, \begin{bmatrix} \phi(\mathbf{w}_{row=1} \cdot \mathbf{x}^{(2)}) \\ \phi(\mathbf{w}_{row=2} \cdot \mathbf{x}^{(2)} \cdot) \\ \vdots \\ \phi(\mathbf{w}_{row=C} \cdot \mathbf{x}^{(2)}) \end{bmatrix}^{(2)}, \cdots, \begin{bmatrix} \phi(\mathbf{w}_{row=1} \cdot \mathbf{x}^{(M)}) \\ \phi(\mathbf{w}_{row=2} \cdot \mathbf{x}^{(M)} \cdot) \\ \vdots \\ \phi(\mathbf{w}_{row=C} \cdot \mathbf{x}^{(M)}) \end{bmatrix} \right]^{(M)}$$

which is one versus-all!

Singel Layer Early Architectures: Summary

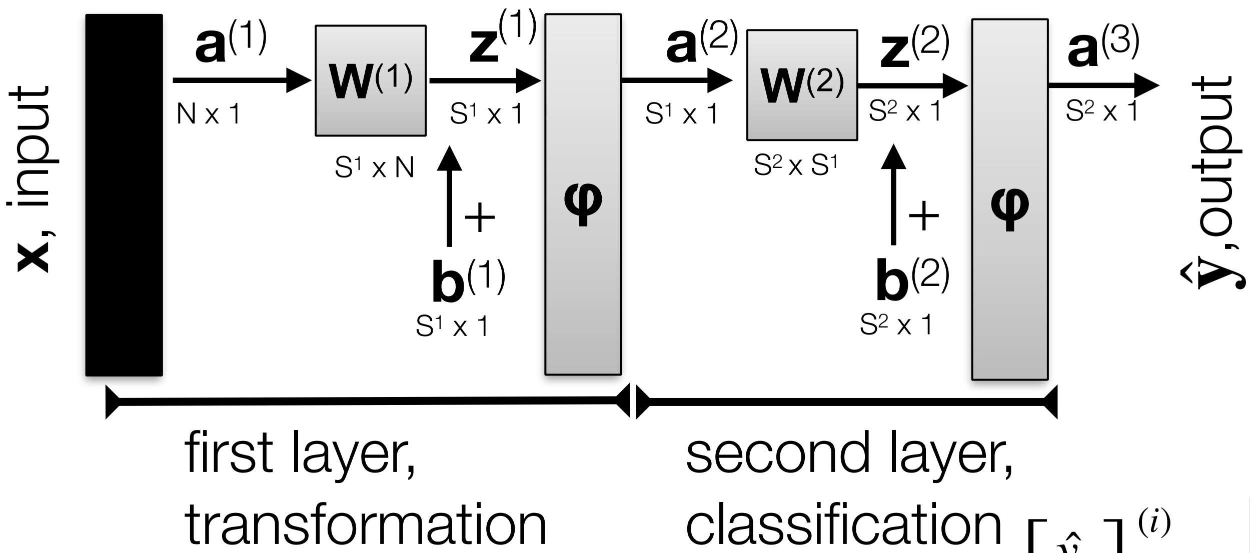
- Adaline network, Widrow and Hoff, 1960
  - linear regression, iterative updates
- Perceptron
  - NN *with sigmoid*: logistic regression
- Multi-class perceptron: one versus all

- **But what about when we have more than one layer?**

Beyond Single Layer Networks

Moving to multiple layers…

- The multi-layer perceptron (MLP):
  - two layers shown, but could be arbitrarily many layers



each element of $\hat{\mathbf{y}}$ is no longer independent.

$\mathbf{W}^{(1)}$ used for all classes so we cannot optimize using one versus all 😢

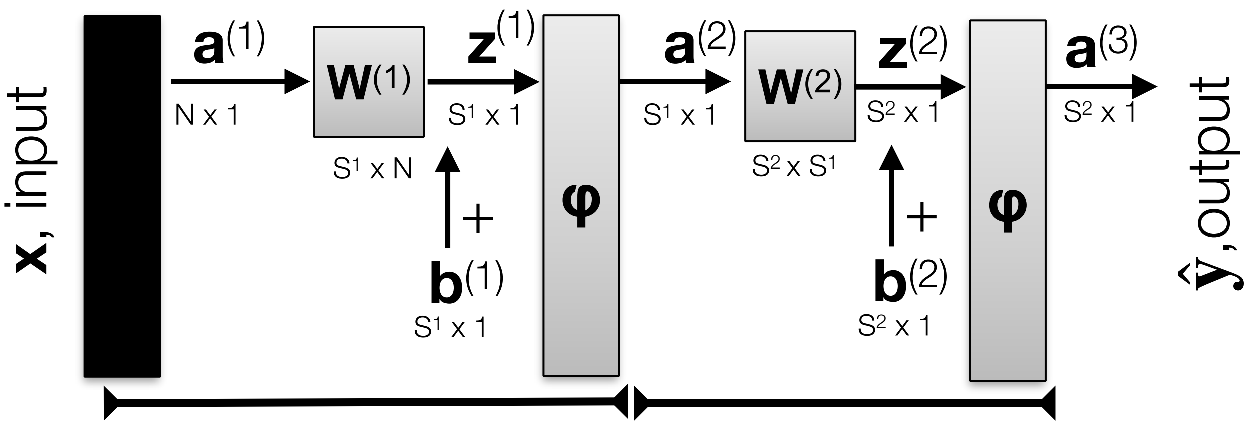
$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_C \end{bmatrix}^{(i)} = \begin{bmatrix} \phi\left(\phi(\boxed{[\mathbf{z}^{(1)}]^{(i)}}) \cdot \mathbf{w}^{(2)}_{row=1} + b^{(2)}_1\right) \\ \phi\left(\phi([\mathbf{z}^{(1)}]^{(i)}) \cdot \mathbf{w}^{(2)}_{row=2} + b^{(2)}_2\right) \\ \vdots \\ \phi\left(\phi([\mathbf{z}^{(1)}]^{(i)}) \cdot \mathbf{w}^{(2)}_{row=C} + b^{(2)}_c\right) \end{bmatrix}$$

$$\boxed{[\mathbf{z}^{(1)}]^{(i)} = \mathbf{W}^{(1)} \cdot [\mathbf{a}^{(1)}]^{(i)} + \mathbf{b}^{(1)}}$$

Back propagation

- Optimize all weights of network at once, using chain rule many times…
- Steps:
    1. Forward propagate to get all $\mathbf{Z}^{(l)}$, $\mathbf{A}^{(l)}$
    2. Get final layer gradient
    3. Back propagate sensitivities (chain rule)
    4. Update each $\mathbf{W}^{(l)}$

Back-propagation is solved in flipped assignment!!



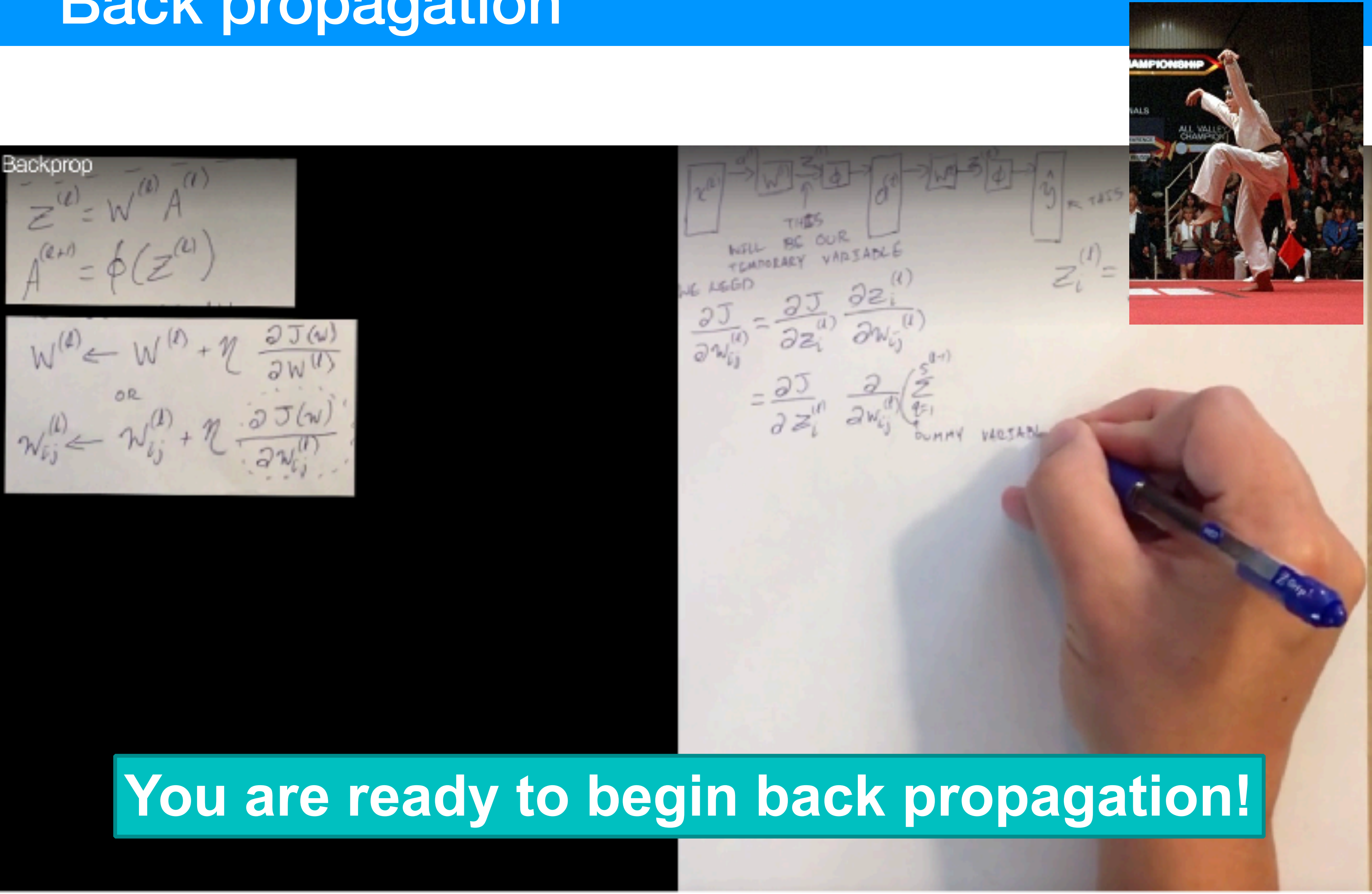
first layer, transformation

second layer, classification

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

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

Back propagation



You are ready to begin back propagation!