Lecture Notes for

Machine Learning in Python

Professor Eric Larson

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
  - **Flipped**: Programming Multi-layer training

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.

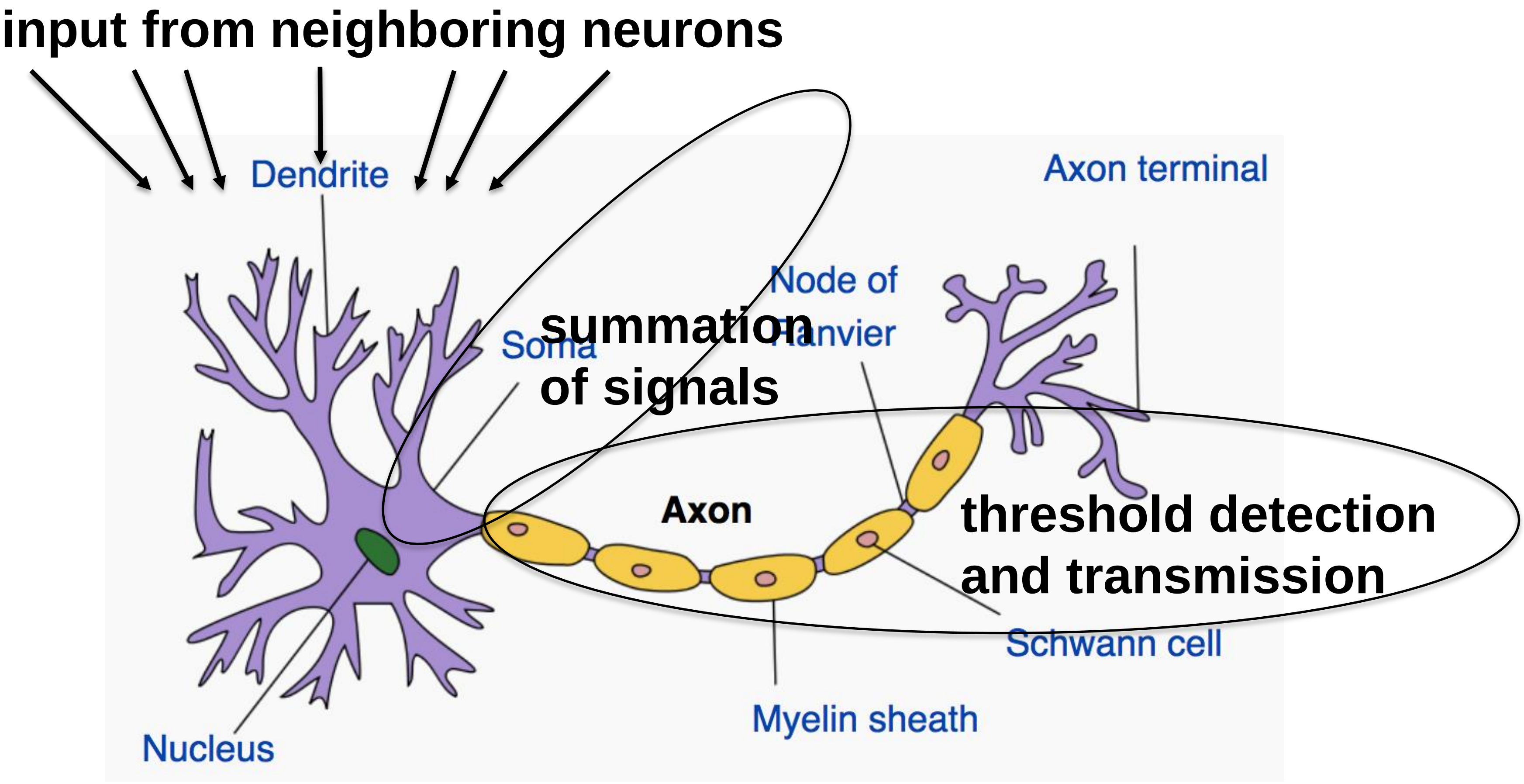
Ethics in Language Models

ConceptNet
Case studies

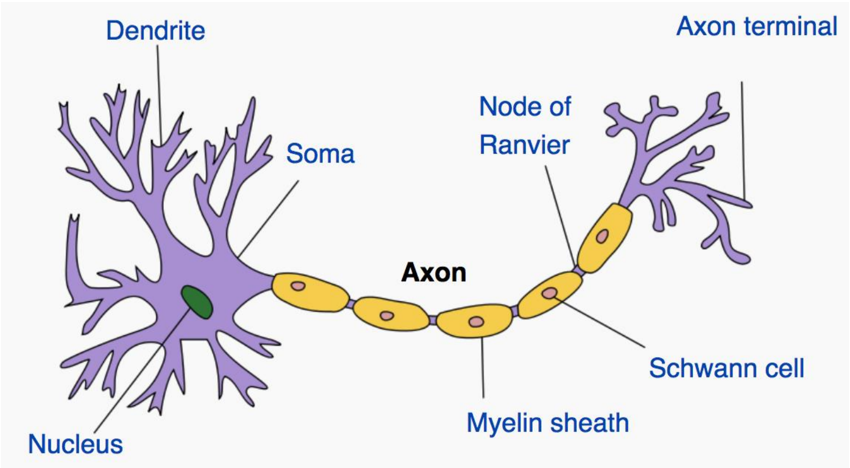
A History of Neural Networks

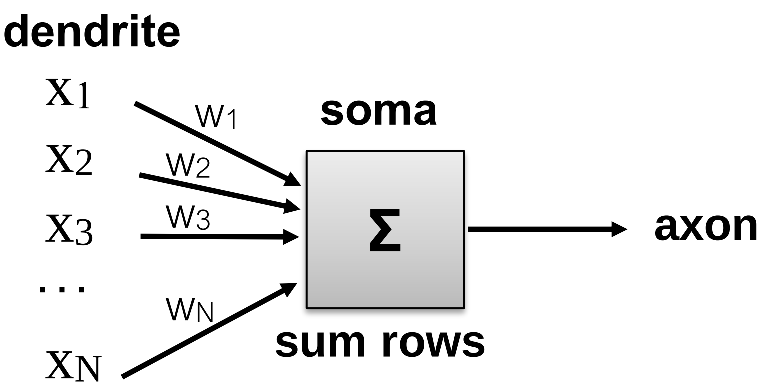
Neurons

- From biology to modeling:



McCulloch and Pitts, 1943



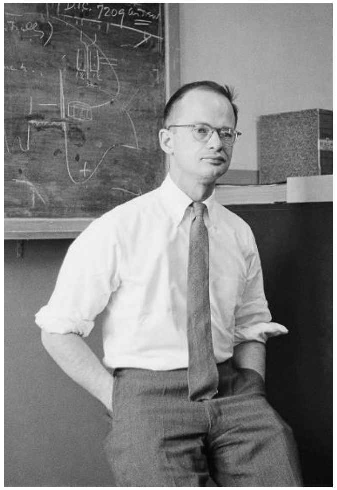


input

$$X \cdot W = a$$

Logic gates of the mind

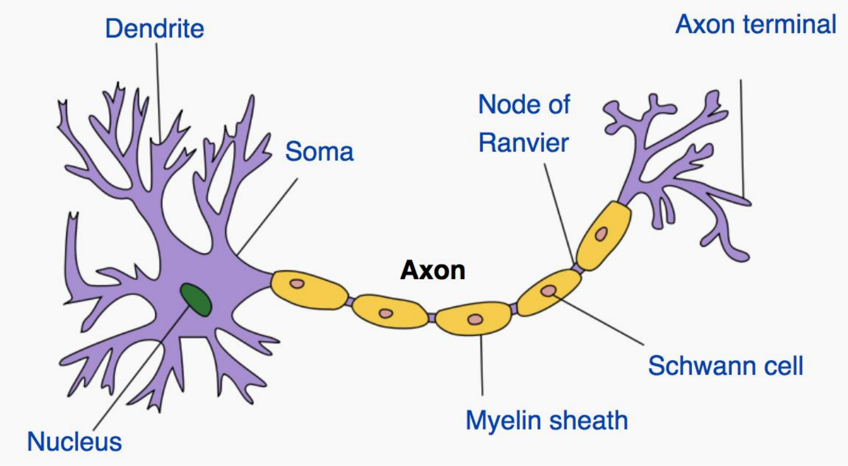
Warren McCulloch



Walter Pitts

Neurons

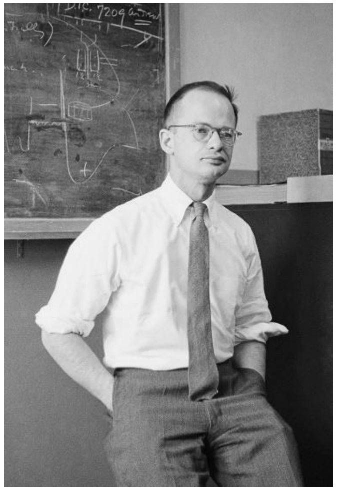
- McCulloch and Pitts 1943
- Donald Hebb, 1949
- ➢ Hebb's law: close neurons fire together
- ➢ Neurons learn to couple
- ➢ Easier synaptic transmission
- ➢ Basis of neural pathways



Donald O. Hebb

Logic gates of the mind

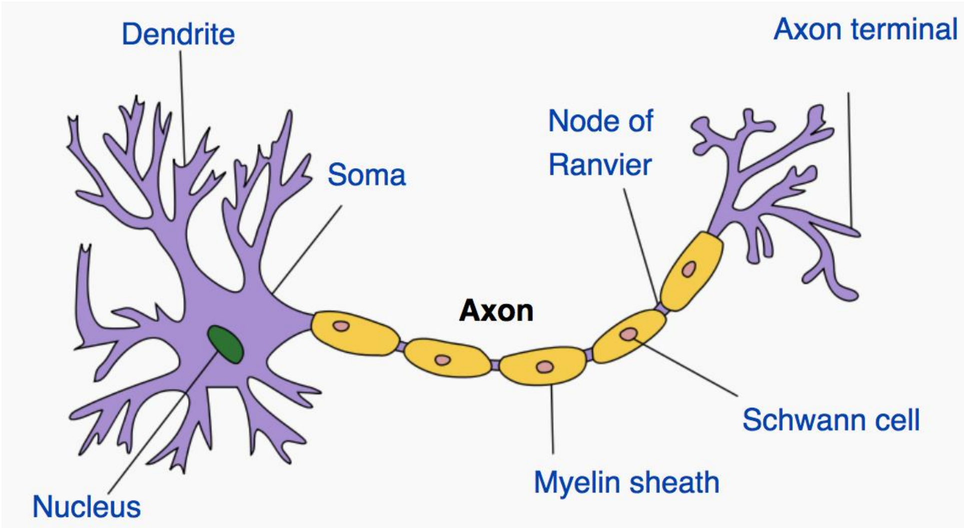
Warren McCulloch

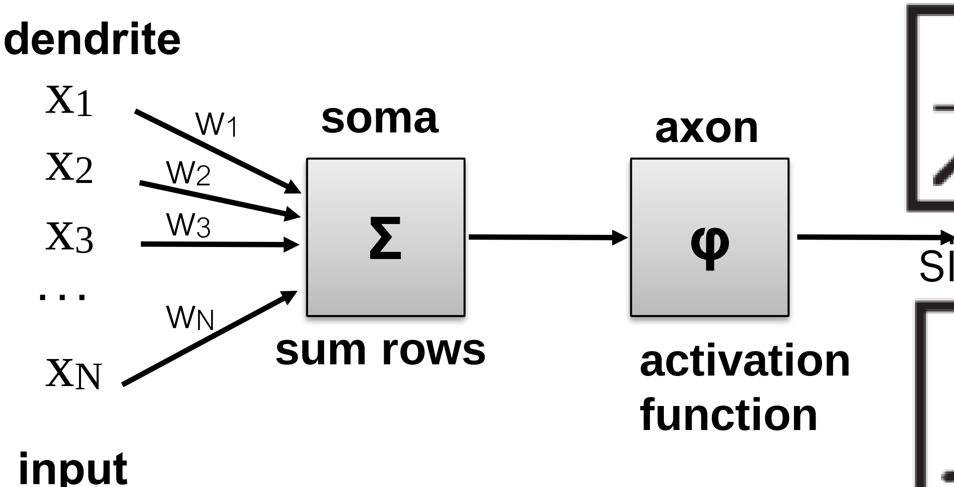


Walter Pitts

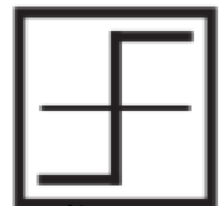
Rosenblatt’s perceptron, 1957

Frank Rosenblatt





hard limit



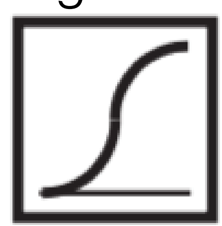
$a=-1 \quad z<0$

$a=1 \quad z>=0$

linear

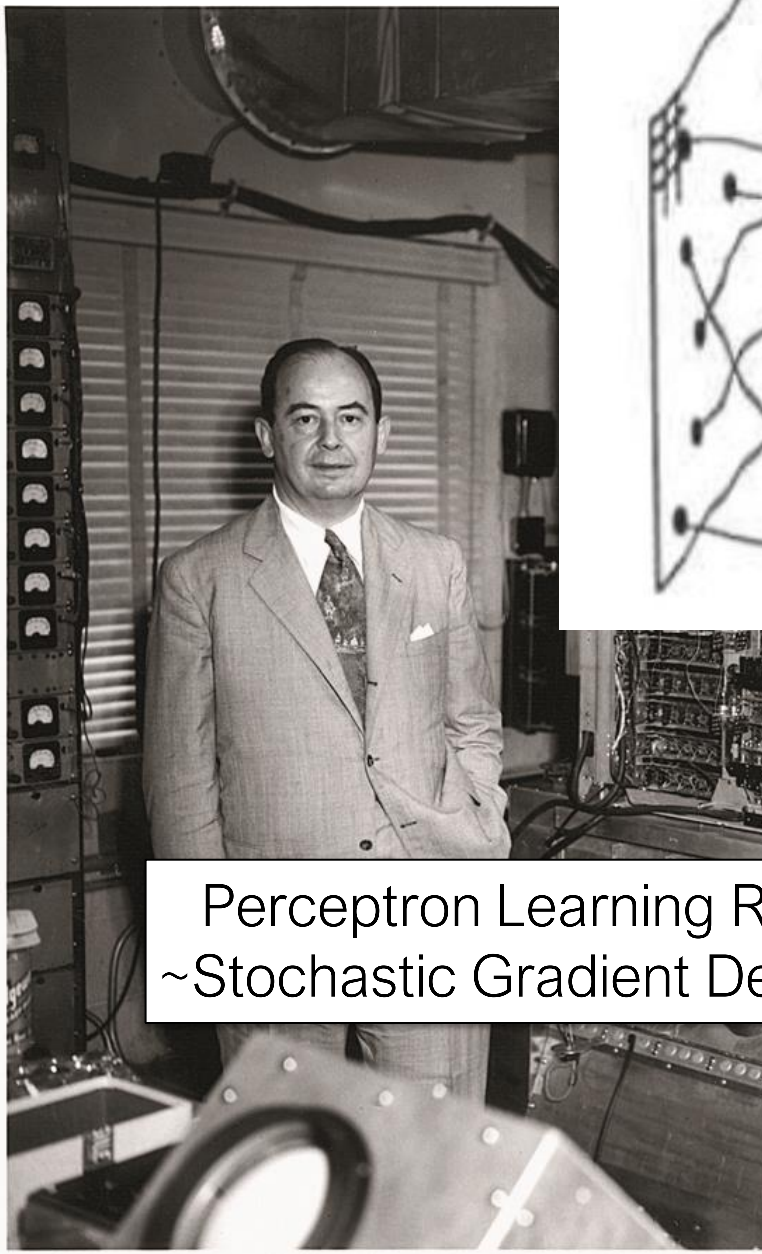
$a=z$

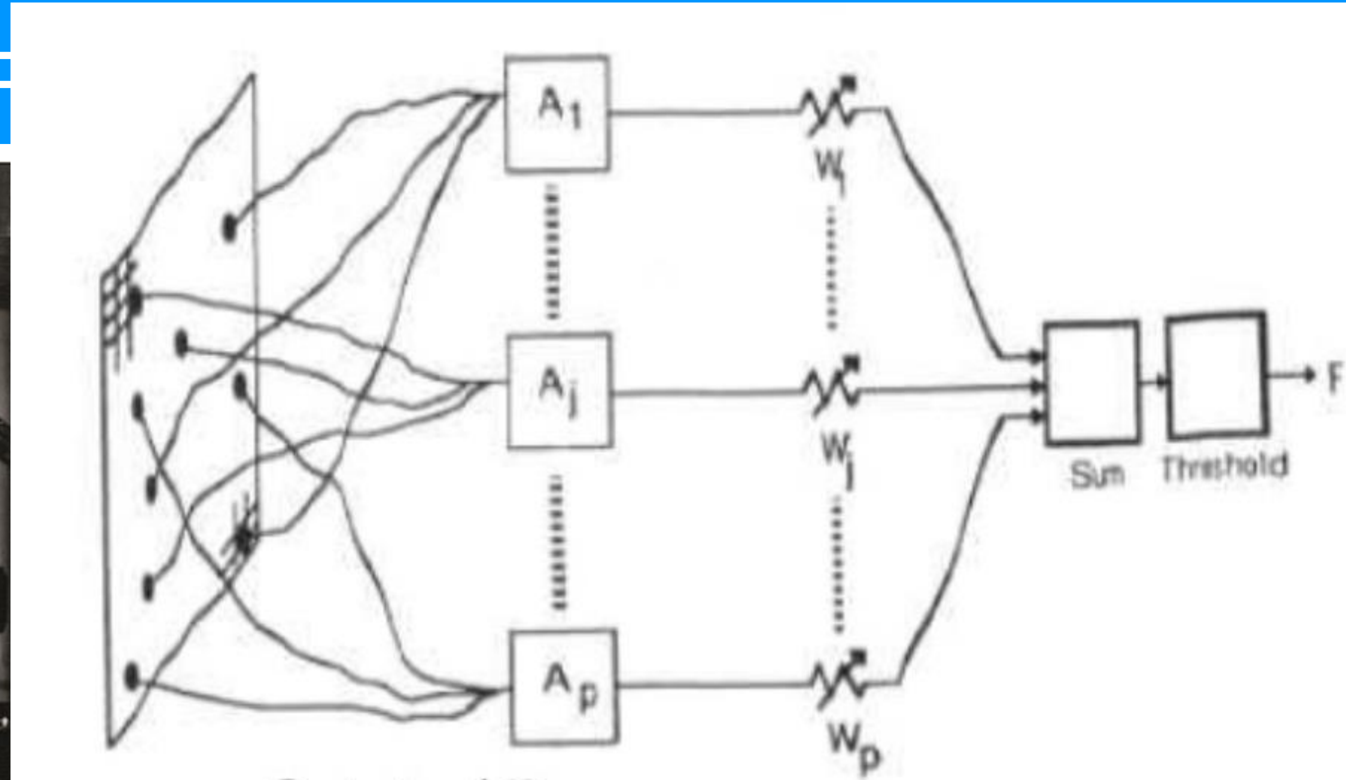
sigmoid

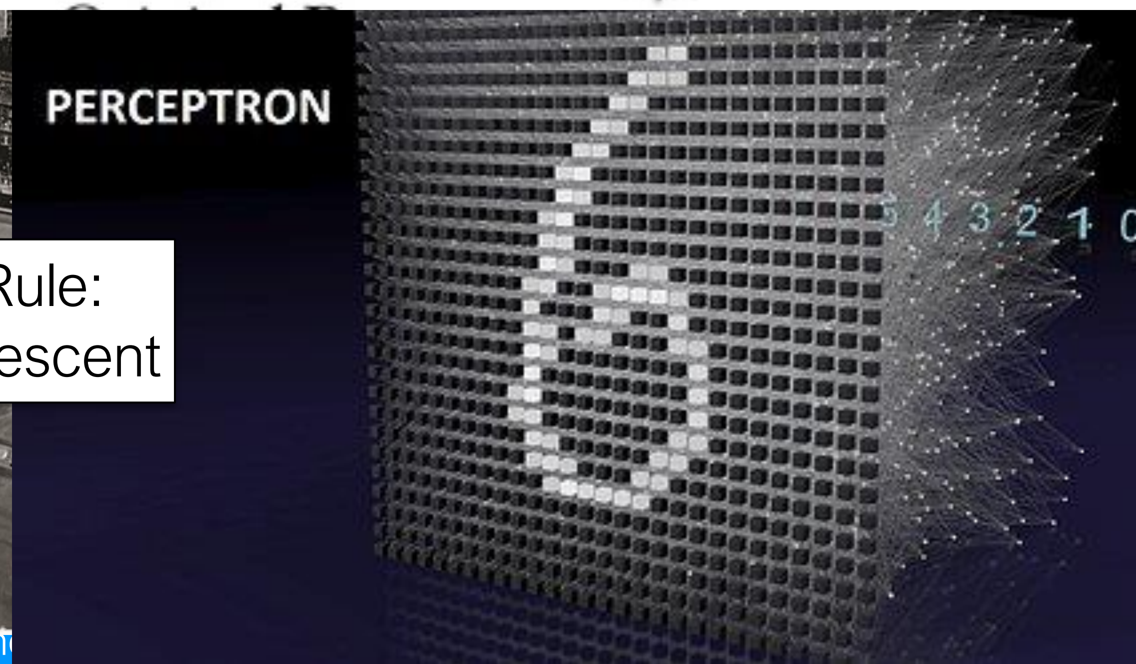


$$a=\frac{1}{1+\exp(-z)}$$

The Mark 1 P

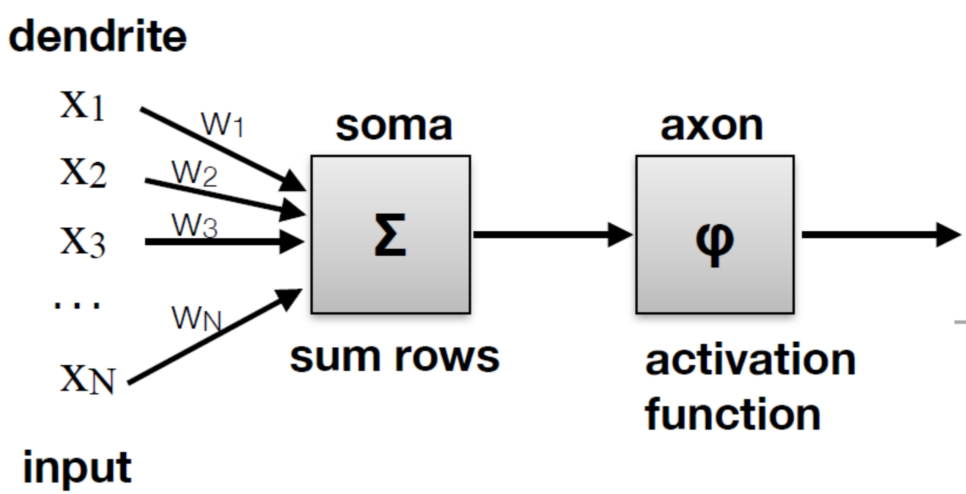






Perceptron Learning Rule:
~Stochastic Gradient Descent

Layers Notation

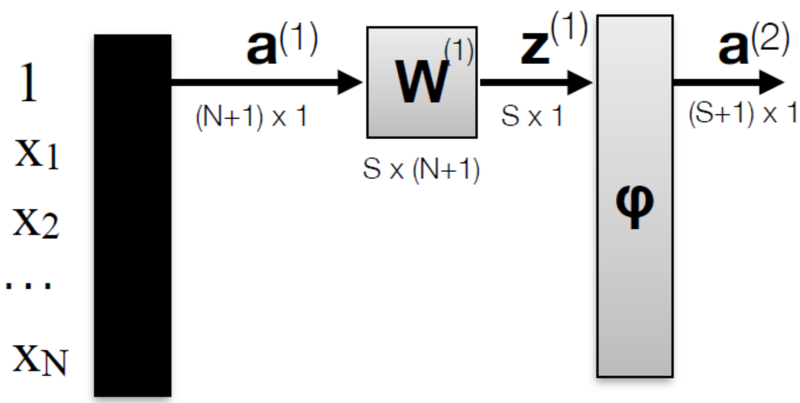


a=**x** with concat bias term

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

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

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



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

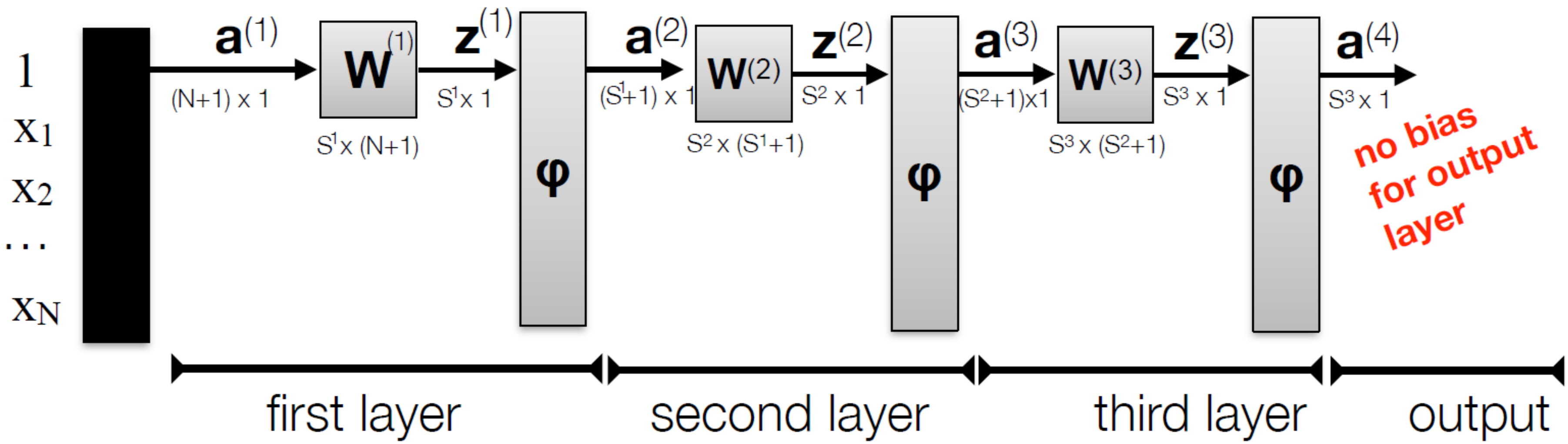
$$\mathbf{a}^{next} = \phi(\mathbf{z}^{\mathbf{current}}),$$ concat bias term

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

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

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

Generic Multiple Layers Notation



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

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

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

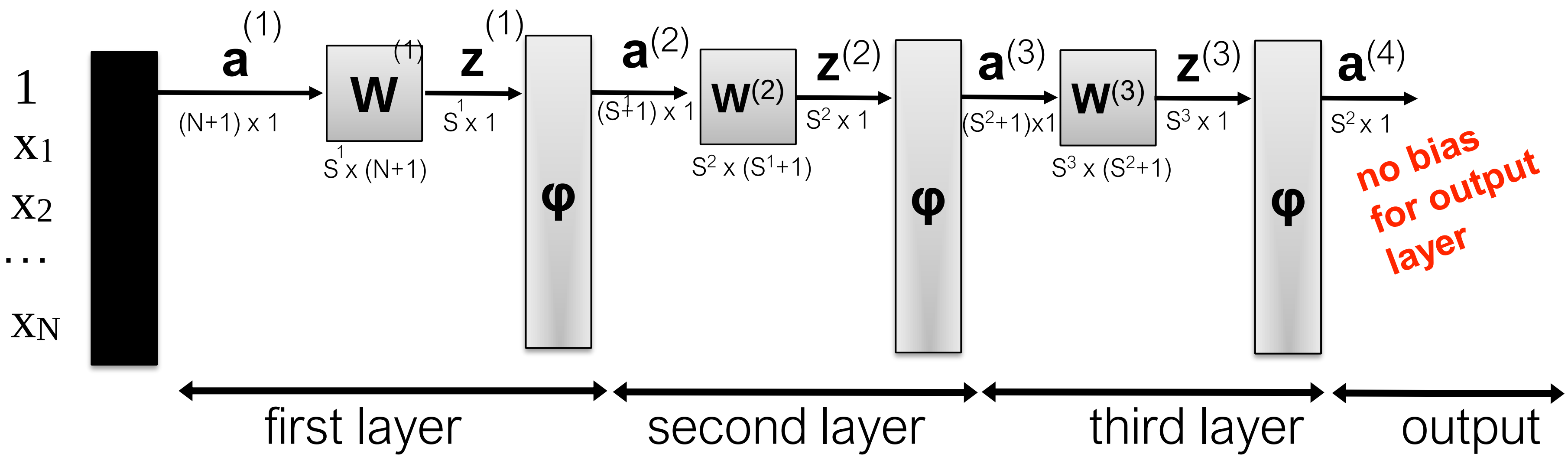
$$\mathbf{W} = \begin{bmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,N} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,N} \\ \vdots & & & \\ w_{S,0} & w_{S,1} & \cdots & w_{S,N} \end{bmatrix}$$

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

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

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

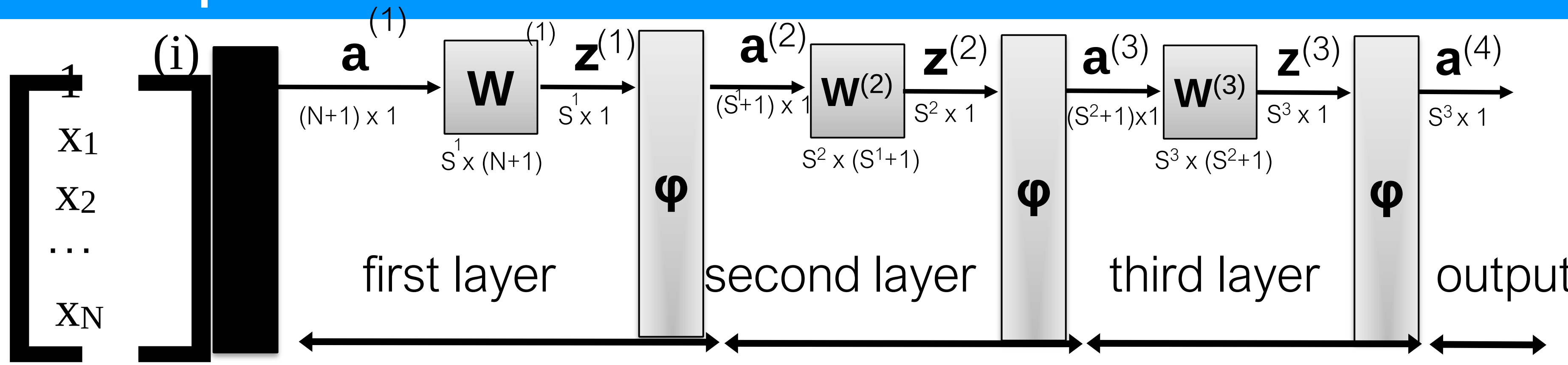
Multiple layers notation



- **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



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

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

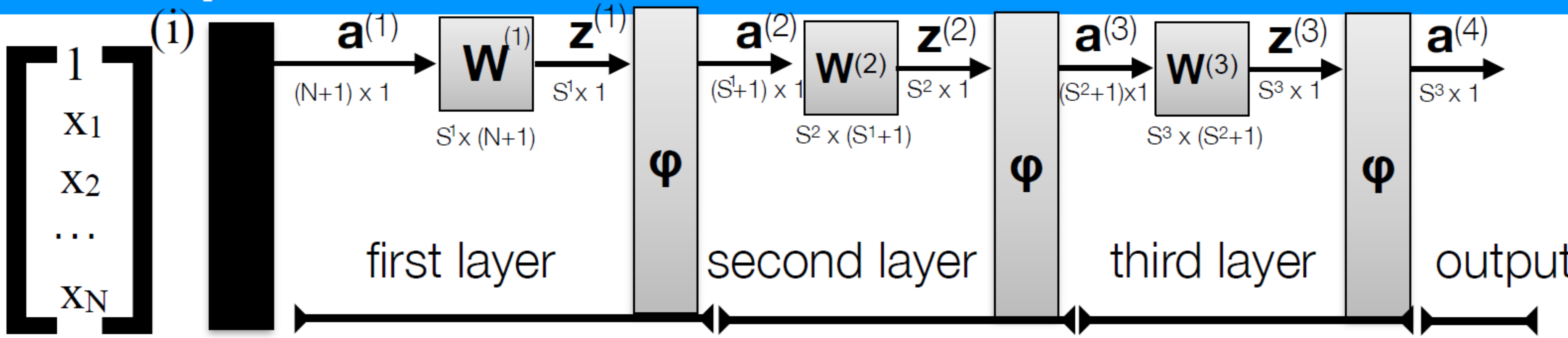
$$\begin{bmatrix} z^{(L)}_1 \\ \cdots \\ z^{(L)}_{S^L} \end{bmatrix}^{(i)} = \left[\mathbf{W}^{(L)}\right] \begin{bmatrix} a^{(L)}_1 \\ \cdots \\ a^{(L)}_{S^{L-1}+1} \end{bmatrix}^{(i)}$$

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

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

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

Compact feedforward notation



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

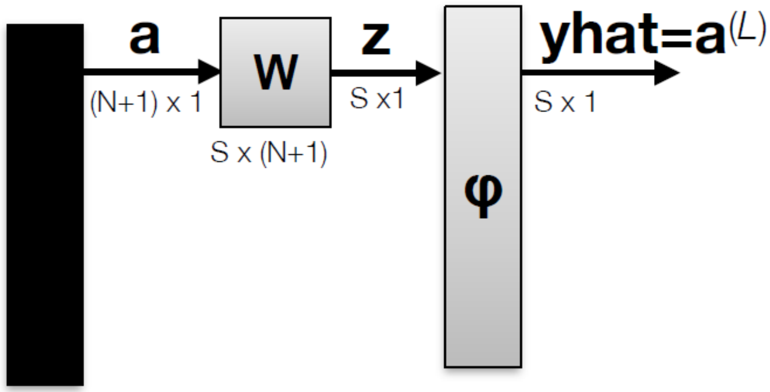
$$\left[\mathbf{z}^{(L)}\right]^{(i)} = \mathbf{W}^{(L)} \cdot \left[\mathbf{a}^{(L)}\right]^{(i)}$$

$$\begin{bmatrix} z_1^{(L)} \\ z_2^{(L)} \\ \vdots \\ z_{S^L}^{(L)} \end{bmatrix}^{(i)} = \mathbf{W}^{(L)} \cdot \begin{bmatrix} a_0^{(L)} \\ a_1^{(L)} \\ \vdots \\ a_{S^{L-1}}^{(L)} \end{bmatrix}^{(i)}$$

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

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

Start Simple: Simplifying to One Layer



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

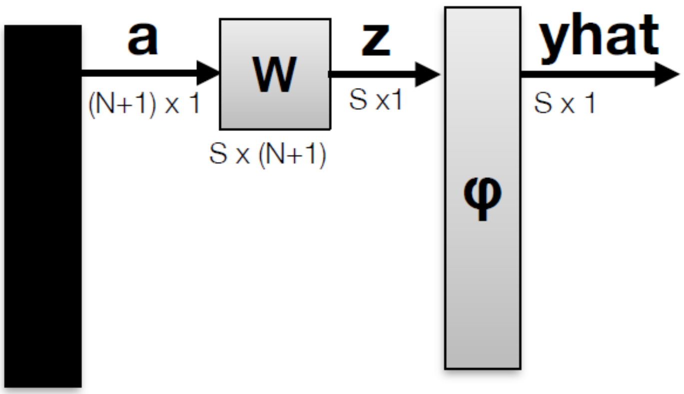
Need objective Function, minimize MSE

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

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

Simple Architectures

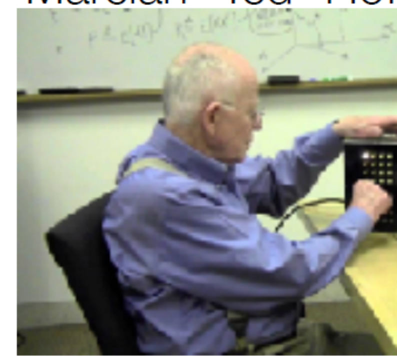
- Adaline network, Widrow and Hoff, 1960



linear

phi

Marcian “Ted” Hoff



Bernard Widrow

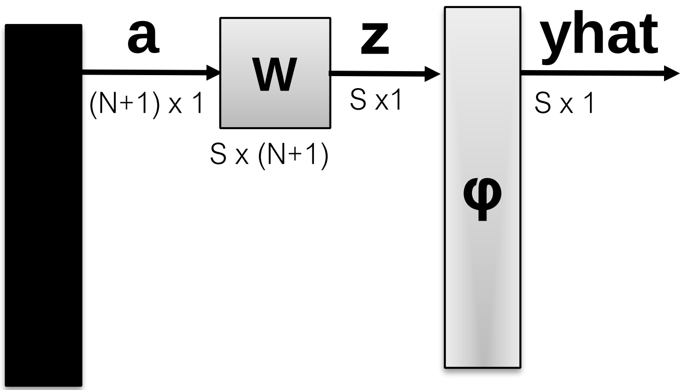
Simplify Objective Function:

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

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**

- Modern Perceptron network



sigmoid

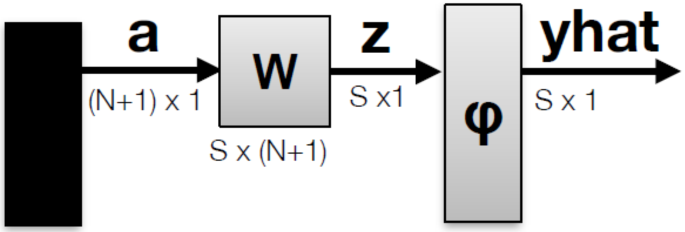
$$\text{phi} = \frac{1}{1+\exp(-z)}$$

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**!

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(\mathbf{y} - g(\mathbf{X} \cdot \mathbf{w})) \odot \mathbf{X}$$

What if we have more than S=1?



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

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

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

$$\ldots$$

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

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

Each target class in **Y** can be independently optimized

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

which is one versus-all!

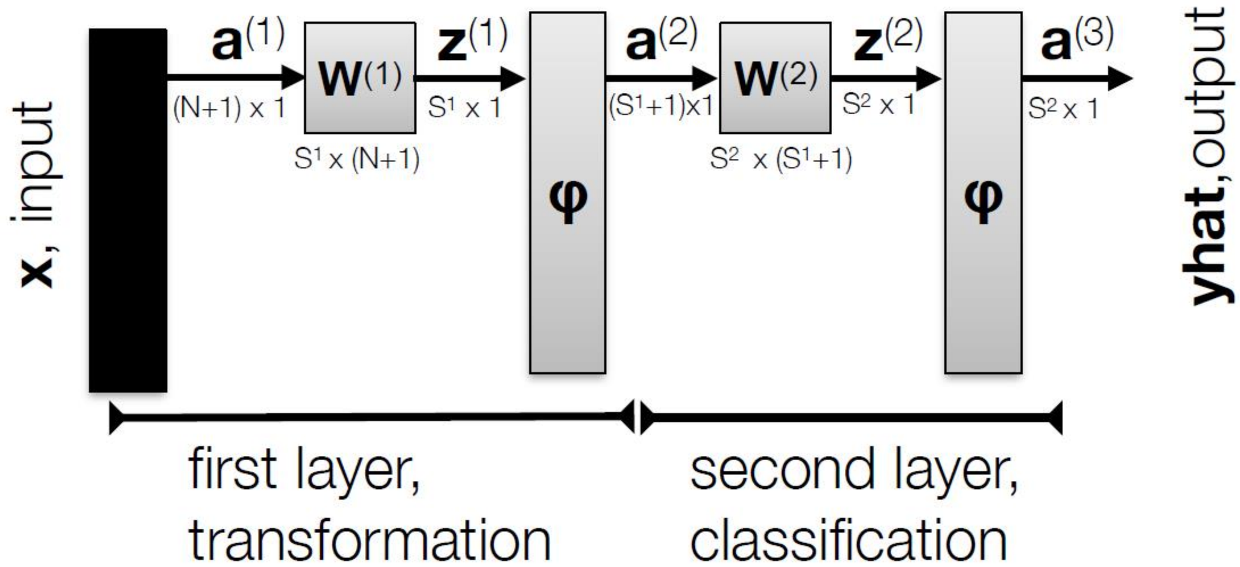
Simple Architectures: Summary

- 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, $\mathbf{W}$

what happens when we have multiple layers?

Moving to multiple layers…

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



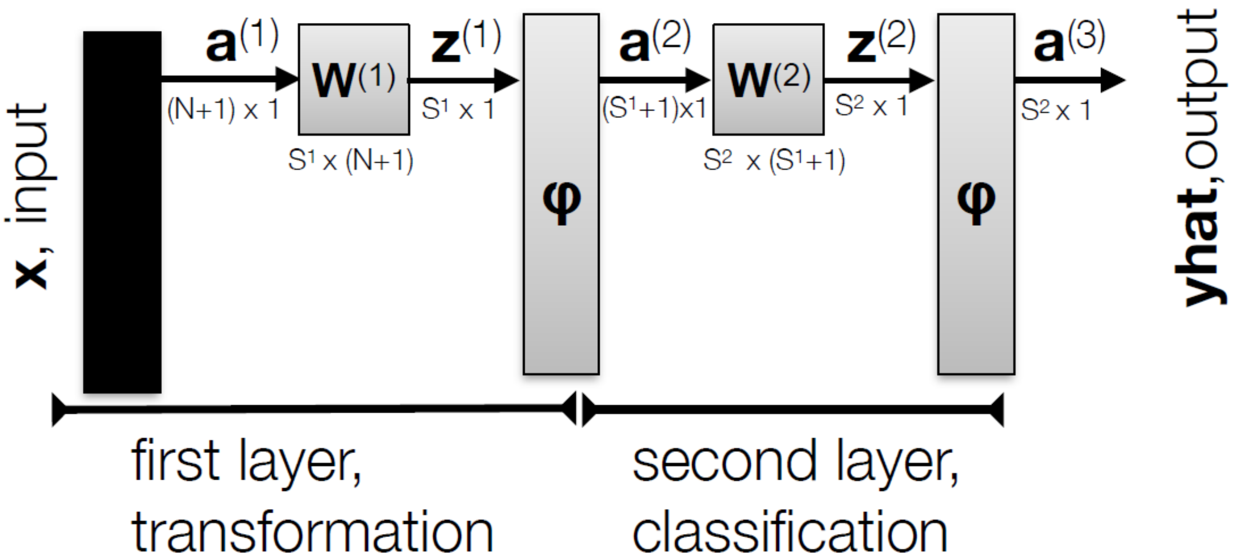
each row of **yhat** is no longer independent of the rows in early **W** so we cannot optimize using one versus all 😢

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_C \end{bmatrix} = \begin{bmatrix} \phi(\phi(\boxed{\mathbf{z}^{(1)}}) \cdot \mathbf{w}^{(2)}_{row=1}) \\ \phi(\phi(\mathbf{z}^{(1)}) \cdot \mathbf{w}^{(2)}_{row=2}) \\ \vdots \\ \phi(\phi(\mathbf{z}^{(1)}) \cdot \mathbf{w}^{(2)}_{row=C}) \end{bmatrix}$$

$$\boxed{\mathbf{z}^{(1)} = \mathbf{W}^{(1)} \cdot \mathbf{a}^{(1)}}$$

Back propagation

- Optimize all weights of network at once
- Steps:
    - propagate weights forward
    - calculate gradient at final layer
    - back propagate gradient for each layer
        - via recurrence relation



Back-propagation is solved in flipped assignment!!