

# Perceptron

Subhransu Maji

CMPSCI 689: Machine Learning

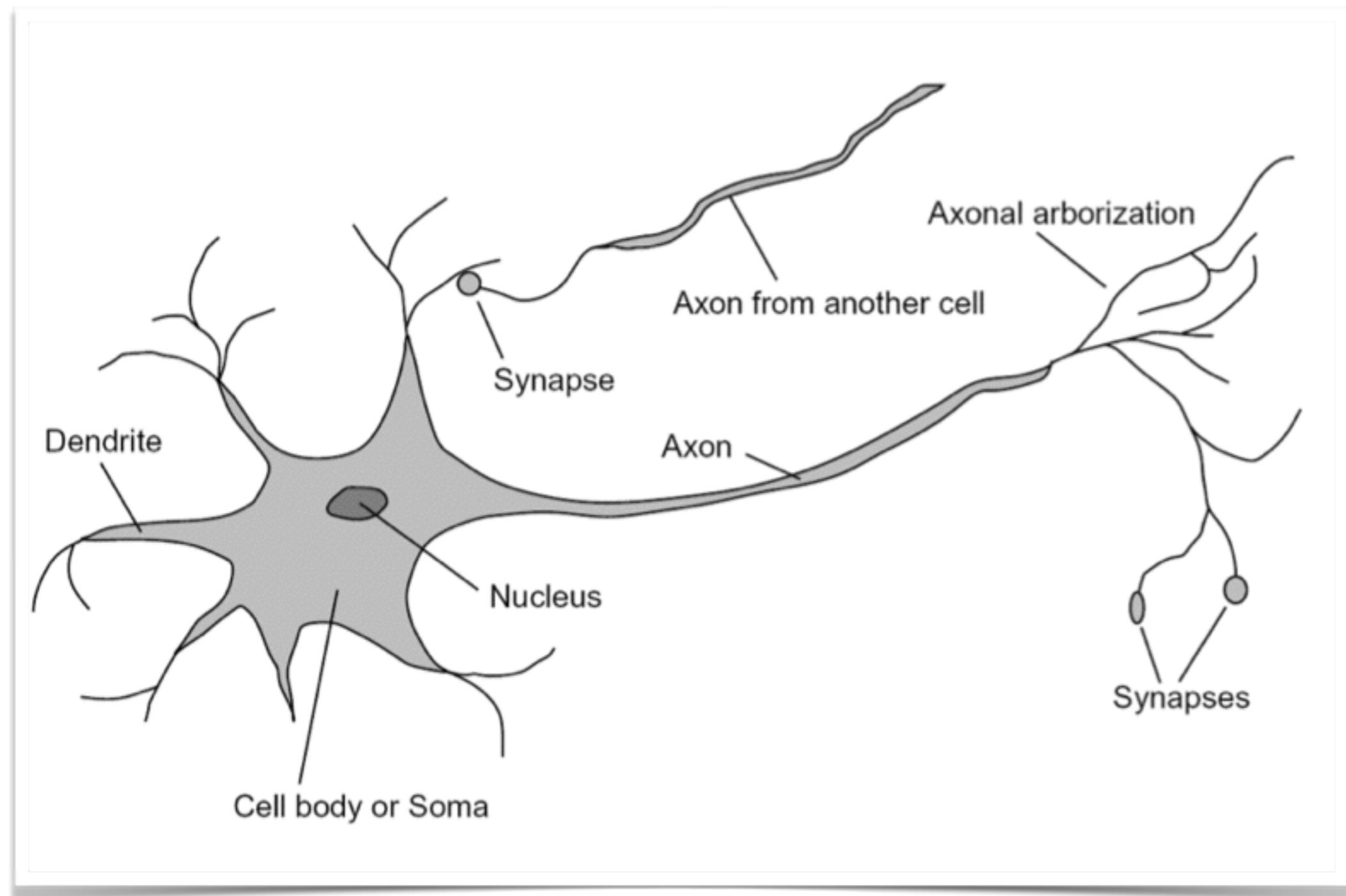
3 February 2015

5 February 2015

# So far in the class

- ◆ Decision trees
  - ▶ Inductive bias: use a combination of small number of features
- ◆ Nearest neighbor classifier
  - ▶ Inductive bias: all features are equally good
- ◆ Perceptrons    Today
  - ▶ Inductive bias: use all features, but some more than others

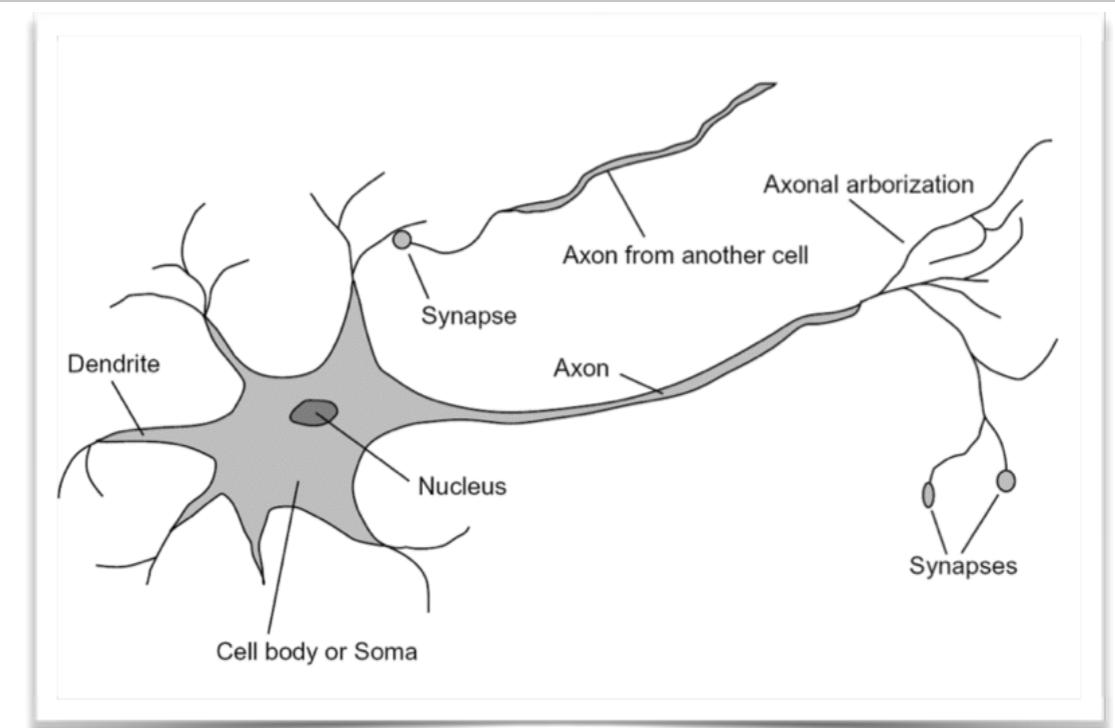
# A neuron (or how our brains work)



Neuroscience 101

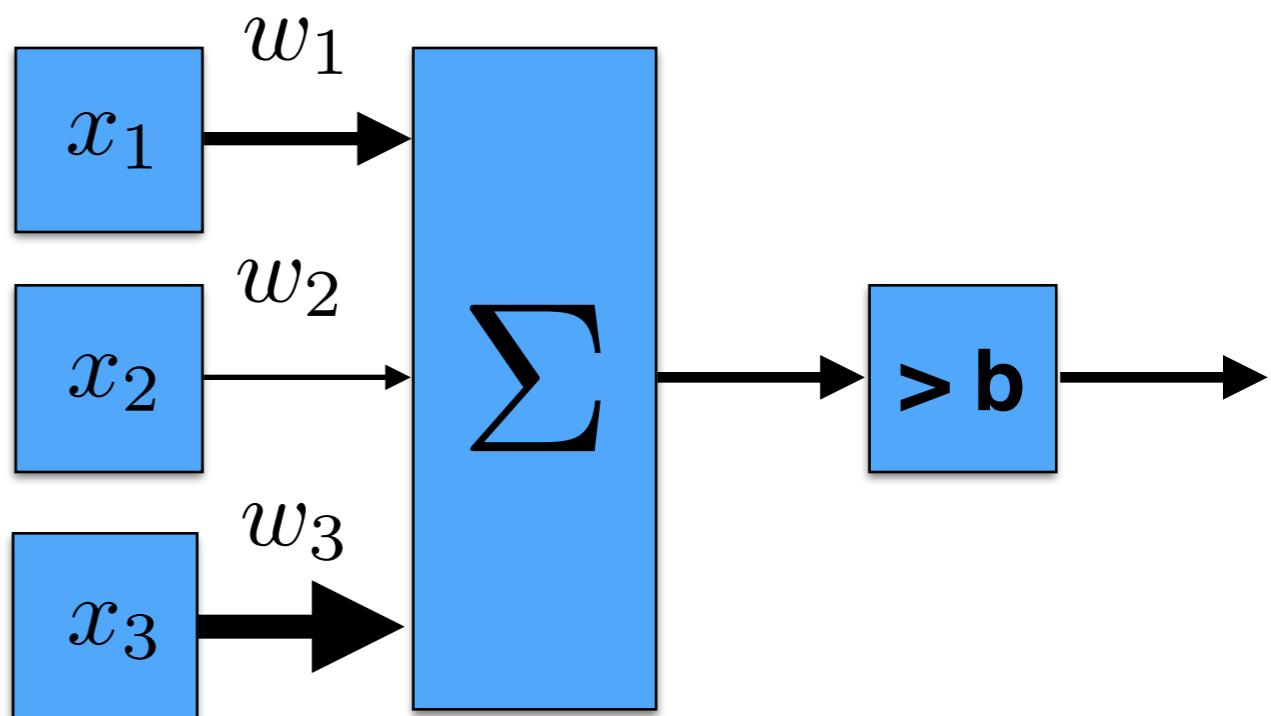
# Perceptron

- ◆ Input are **feature values**
- ◆ Each feature has a **weight**
- ◆ Sum in the **activation**



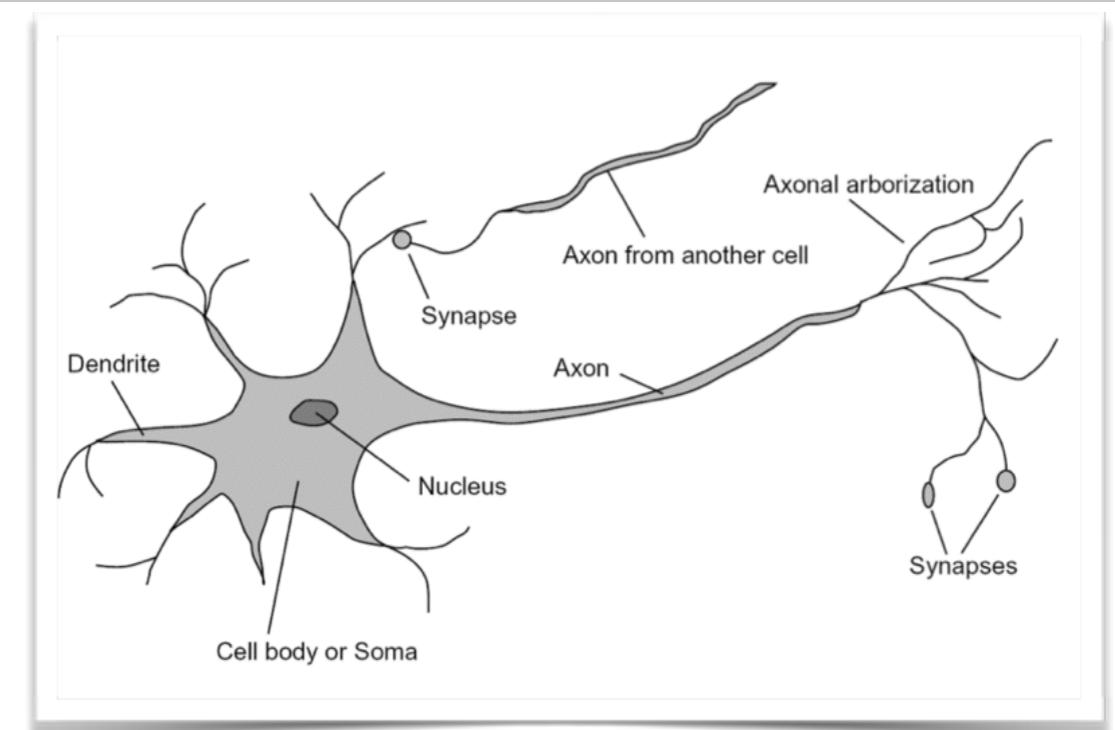
$$\text{activation}(\mathbf{w}, \mathbf{x}) = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$$

- ◆ If the activation is:
  - ▶  $> b$ , output *class 1*
  - ▶ otherwise, output *class 2*



# Perceptron

- ◆ Input are **feature values**
- ◆ Each feature has a **weight**
- ◆ Sum in the **activation**

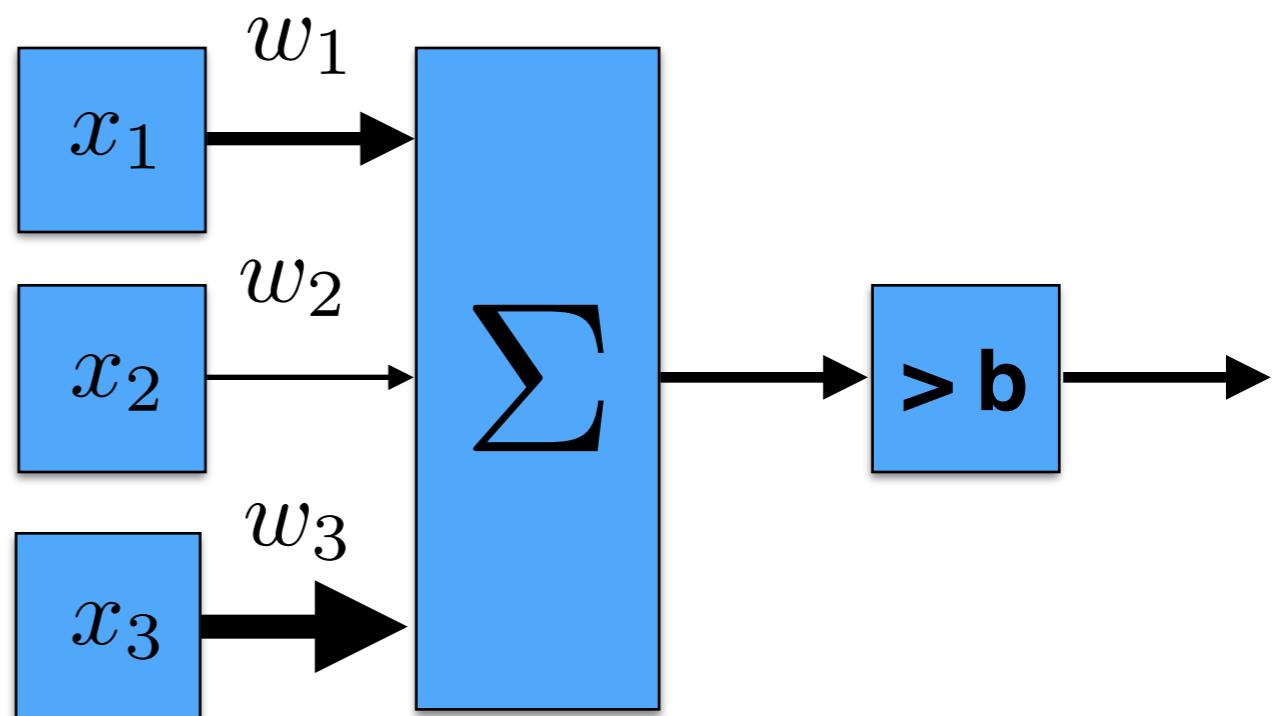


$$\text{activation}(\mathbf{w}, \mathbf{x}) = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$$

- ◆ If the activation is:
  - ▶  $> b$ , output *class 1*
  - ▶ otherwise, output *class 2*

$$\mathbf{x} \rightarrow (\mathbf{x}, 1)$$

$$\mathbf{w}^T \mathbf{x} + b \rightarrow (\mathbf{w}, b)^T (\mathbf{x}, 1)$$



# Example: Spam

◆ Imagine 3 features (spam is “positive” class):

- ▶ free (number of occurrences of “free”)
- ▶ money (number of occurrences of “money”)
- ▶ BIAS (intercept, always has value 1)

email	$\mathbf{x}$	$\mathbf{w}$	$\mathbf{w}^T \mathbf{x}$
“free money”	<code>BIAS : 1 free : 1 money : 1 ...</code>	<code>BIAS : -3 free : 4 money : 2 ...</code>	$\begin{aligned} &(1)(-3) + \\ &(1)(4) + \\ &(1)(2) + \\ &\dots \\ &= 3 \end{aligned}$

$\mathbf{w}^T \mathbf{x} > 0 \rightarrow \text{SPAM!!}$

# Geometry of the perceptron

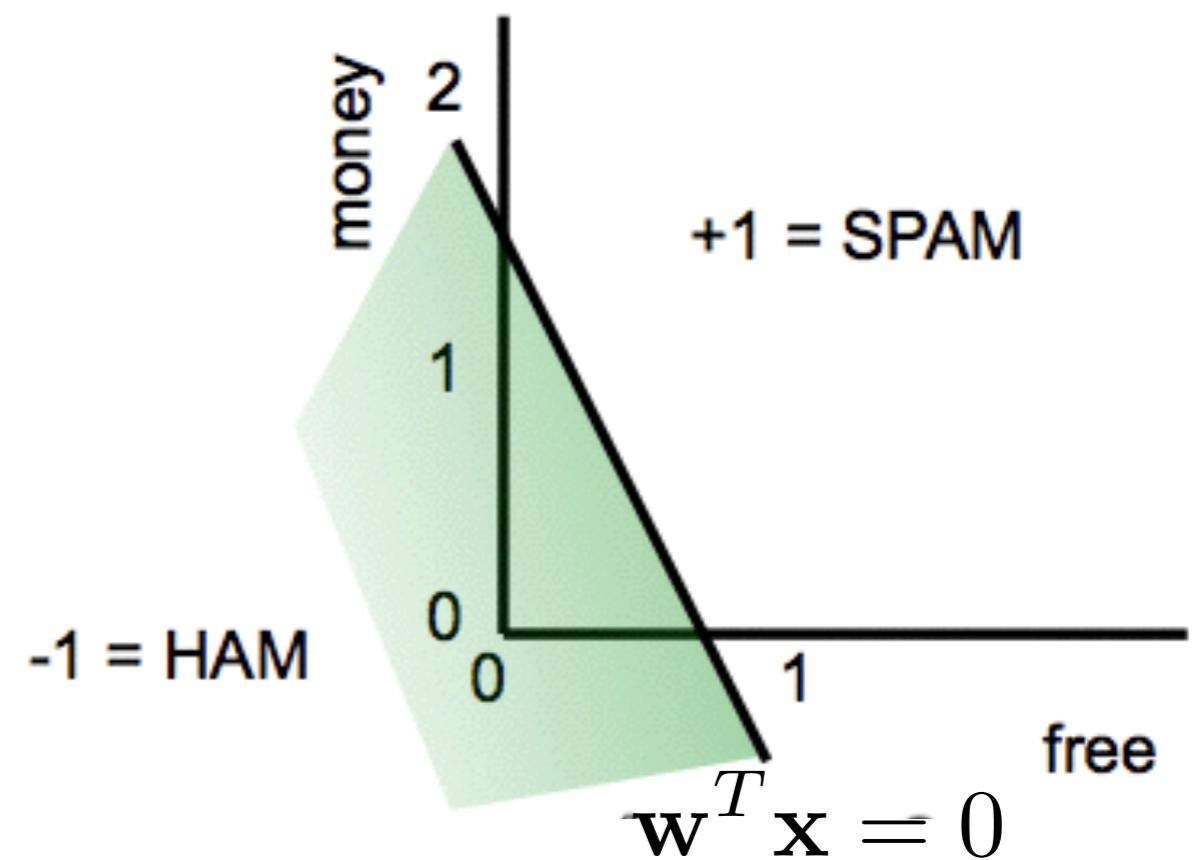
- ◆ In the space of feature vectors

- ▶ examples are points (in D dimensions)
- ▶ an weight vector is a hyperplane (a D-1 dimensional object)
- ▶ One side corresponds to  $y=+1$
- ▶ Other side corresponds to  $y=-1$

- ◆ Perceptrons are also called as linear classifiers

**w**

BIAS : -3
free : 4
money : 2
...



# Learning a perceptron

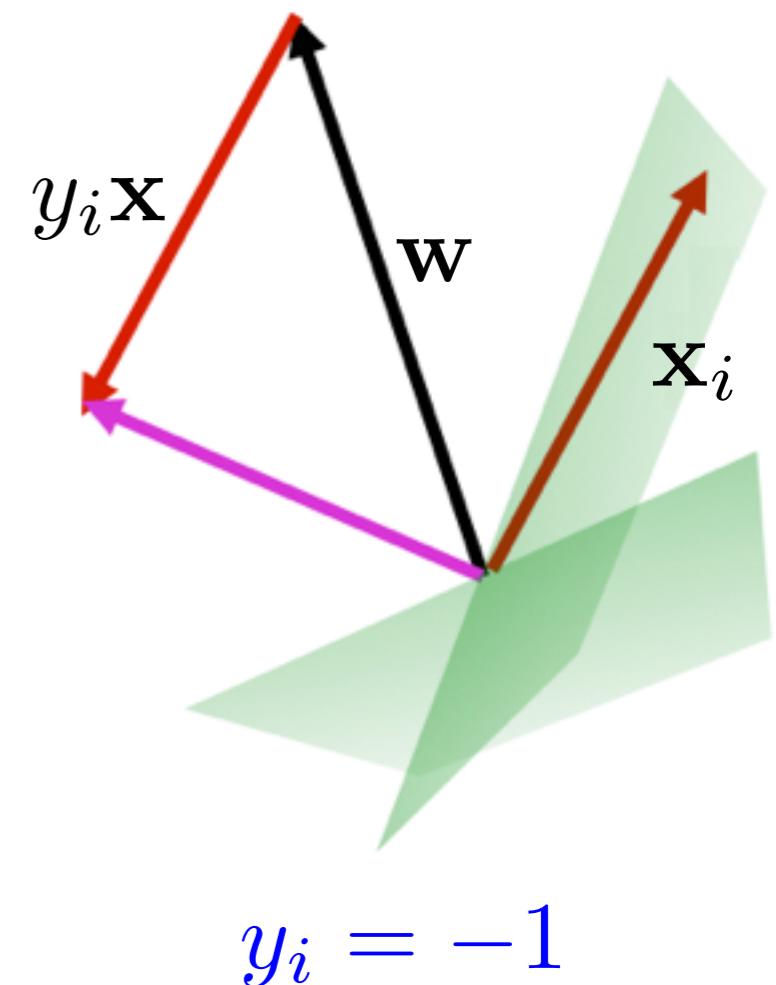
Input: training data  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$

## Perceptron training algorithm [Rosenblatt 57]

- ◆ Initialize  $\mathbf{w} \leftarrow [0, \dots, 0]$
- ◆ for iter = 1,...,T
  - ▶ for i = 1,..,n
    - predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x}_i > 0 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x}_i \leq 0 \end{cases}$$

- if  $y_i = \hat{y}_i$ , no change
- else,  $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$



# Learning a perceptron

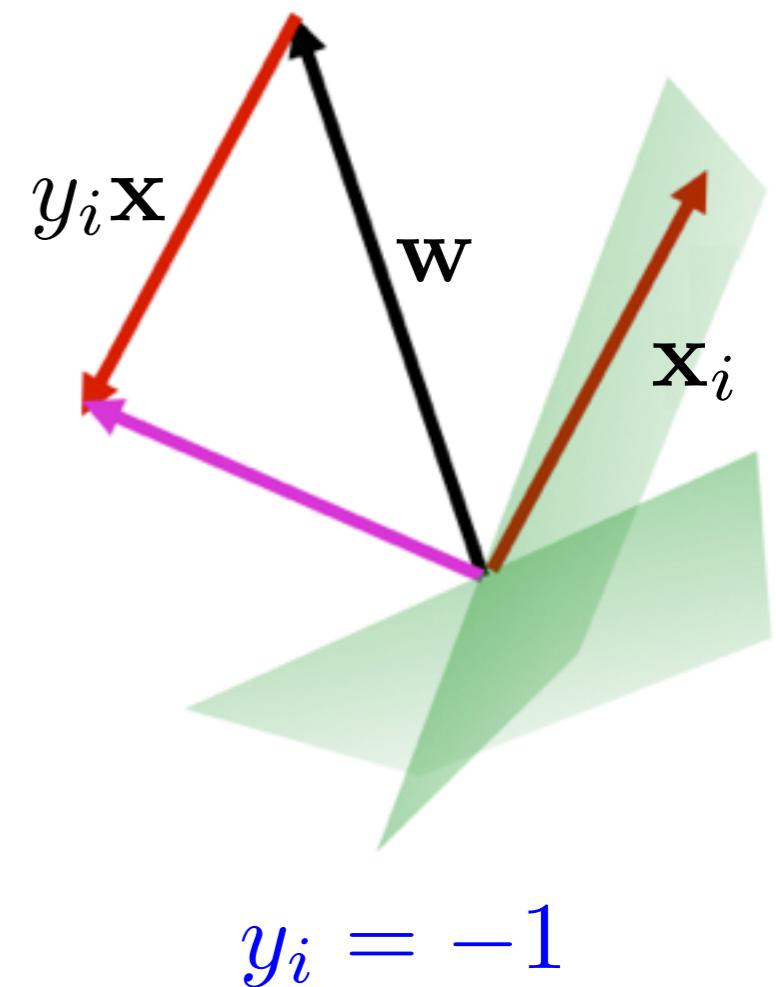
Input: training data  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$

## Perceptron training algorithm [Rosenblatt 57]

- ◆ Initialize  $\mathbf{w} \leftarrow [0, \dots, 0]$
- ◆ for iter = 1,...,T
  - ▶ for i = 1,..,n
    - predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x}_i > 0 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x}_i \leq 0 \end{cases}$$

- if  $y_i = \hat{y}_i$ , no change
- else,  $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$



error driven, online, activations increase for +, randomize

# Properties of perceptrons

- ◆ **Separability:** some parameters will classify the training data perfectly
- ◆ **Convergence:** if the training data is separable then the perceptron training will eventually converge [Block 62, Novikoff 62]

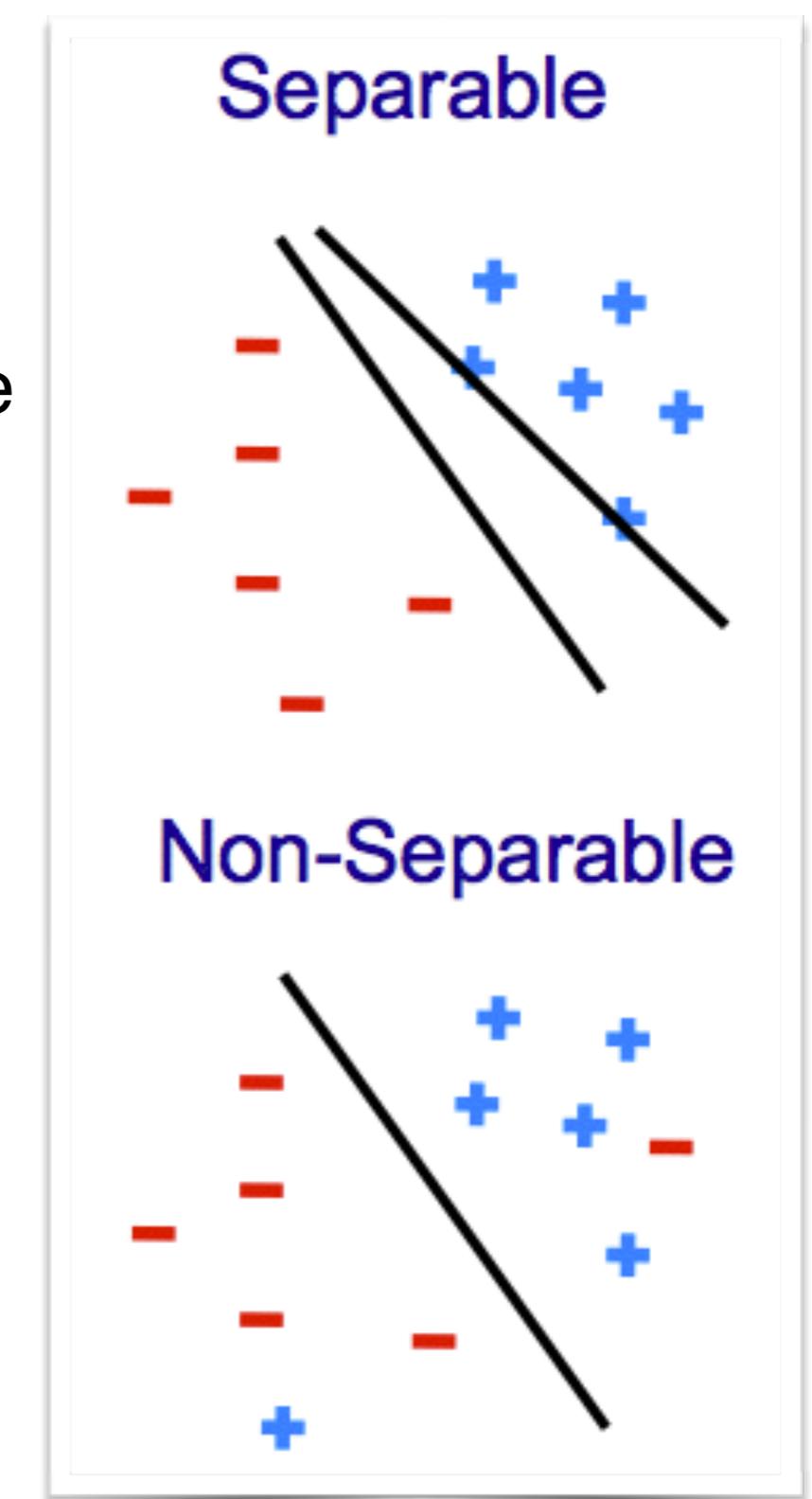
- ◆ **Mistake bound:** the maximum number of mistakes is related to the margin

assuming,  $\|\mathbf{x}_i\| \leq 1$

$$\#\text{mistakes} < \frac{1}{\delta^2}$$

$$\delta = \max_{\mathbf{w}} \min_{(\mathbf{x}_i, y_i)} [y_i \mathbf{w}^T \mathbf{x}_i]$$

$$\text{such that, } \|\mathbf{w}\| = 1$$



# Review geometry

# Proof of convergence

Let,  $\hat{w}$  be the separating hyperplane with margin  $\delta$

# Proof of convergence

Let,  $\hat{\mathbf{w}}$  be the separating hyperplane with margin  $\delta$

$$\begin{aligned}\hat{\mathbf{w}}^T \mathbf{w}^{(k)} &= \hat{\mathbf{w}}^T \left( \mathbf{w}^{(k-1)} + y_i \mathbf{x}_i \right) && \text{update rule} \\ &= \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \hat{\mathbf{w}}^T y_i \mathbf{x}_i && \text{algebra} \\ &\geq \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \delta && \text{definition of margin} \\ &\geq k\delta && \|\mathbf{w}^{(k)}\| \geq k\delta\end{aligned}$$

$\mathbf{w}$  is getting closer

# Proof of convergence

Let,  $\hat{\mathbf{w}}$  be the separating hyperplane with margin  $\delta$

$$\begin{aligned}\hat{\mathbf{w}}^T \mathbf{w}^{(k)} &= \hat{\mathbf{w}}^T \left( \mathbf{w}^{(k-1)} + y_i \mathbf{x}_i \right) && \text{update rule} \\ &= \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \hat{\mathbf{w}}^T y_i \mathbf{x}_i && \text{algebra} \\ &\geq \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \delta && \text{definition of margin} \\ &\geq k\delta && \|\mathbf{w}^{(k)}\| \geq k\delta\end{aligned}$$

$\mathbf{w}$  is getting closer

bound the norm

$$\begin{aligned}\|\mathbf{w}^{(k)}\|^2 &= \|\mathbf{w}^{(k-1)} + y_i \mathbf{x}_i\|^2 && \text{update rule} \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + \|y_i \mathbf{x}_i\|^2 && \text{triangle inequality} \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + 1 && \text{norm} \\ &\leq k && \|\mathbf{w}^{(k)}\| \leq \sqrt{k}\end{aligned}$$

# Proof of convergence

Let,  $\hat{\mathbf{w}}$  be the separating hyperplane with margin  $\delta$

$$\begin{aligned}\hat{\mathbf{w}}^T \mathbf{w}^{(k)} &= \hat{\mathbf{w}}^T \left( \mathbf{w}^{(k-1)} + y_i \mathbf{x}_i \right) && \text{update rule} \\ &= \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \hat{\mathbf{w}}^T y_i \mathbf{x}_i && \text{algebra} \\ &\geq \hat{\mathbf{w}}^T \mathbf{w}^{(k-1)} + \delta && \text{definition of margin} \\ &\geq k\delta && \|\mathbf{w}^{(k)}\| \geq k\delta\end{aligned}$$

$\mathbf{w}$  is getting closer

$$\begin{aligned}\|\mathbf{w}^{(k)}\|^2 &= \|\mathbf{w}^{(k-1)} + y_i \mathbf{x}_i\|^2 && \text{update rule} \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + \|y_i \mathbf{x}_i\|^2 && \text{triangle inequality} \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + 1 && \text{norm} \\ &\leq k && \|\mathbf{w}^{(k)}\| \leq \sqrt{k}\end{aligned}$$

$$k\delta \leq \|\mathbf{w}^{(k)}\| \leq \sqrt{k} \rightarrow k \leq \frac{1}{\delta^2}$$

# Limitations of perceptrons

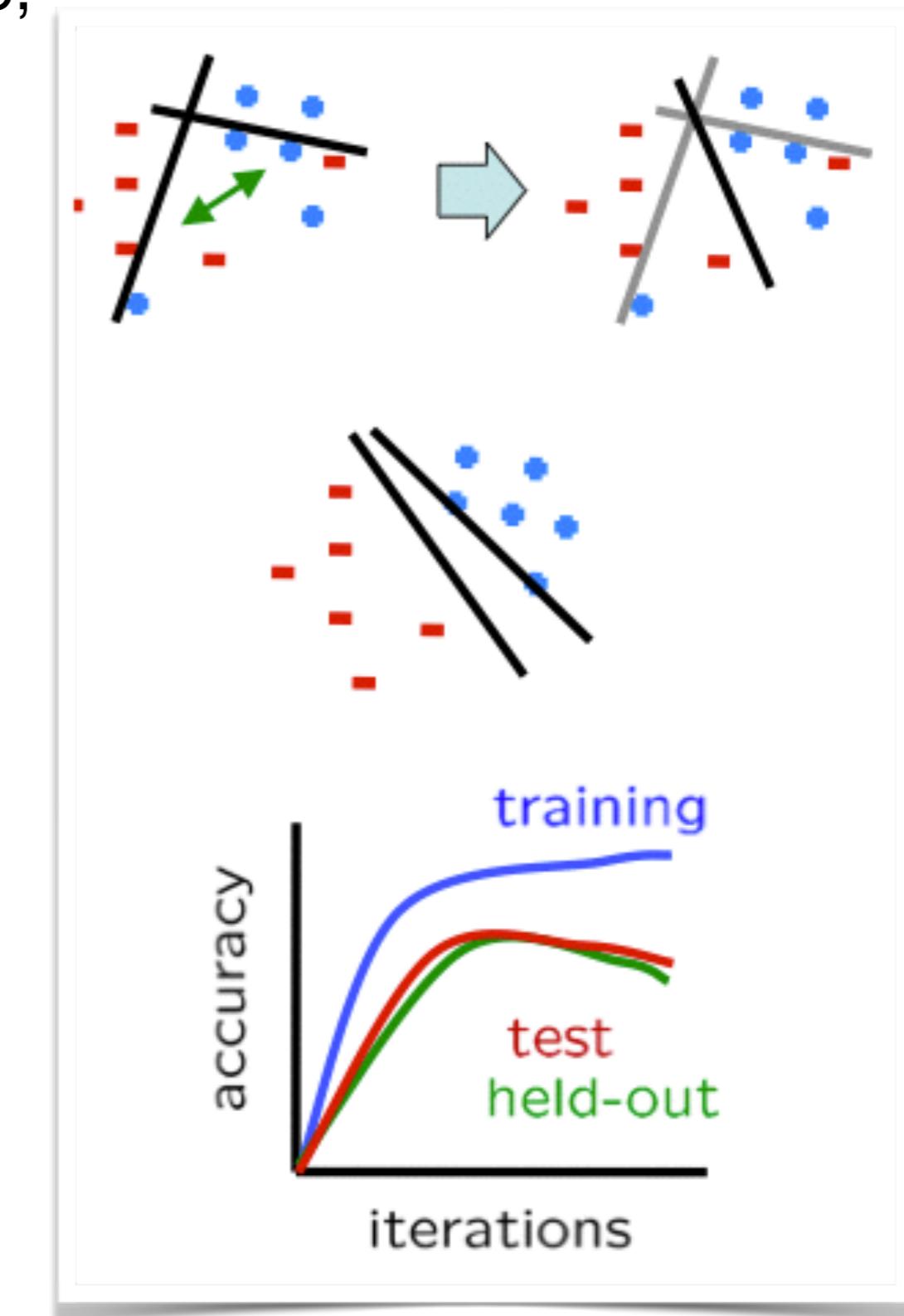
◆ **Convergence:** if the data isn't separable, the training algorithm may not terminate

- ▶ noise can cause this
- ▶ some simple functions are not separable (xor)

◆ **Mediocre generation:** the algorithm finds a solution that “barely” separates the data

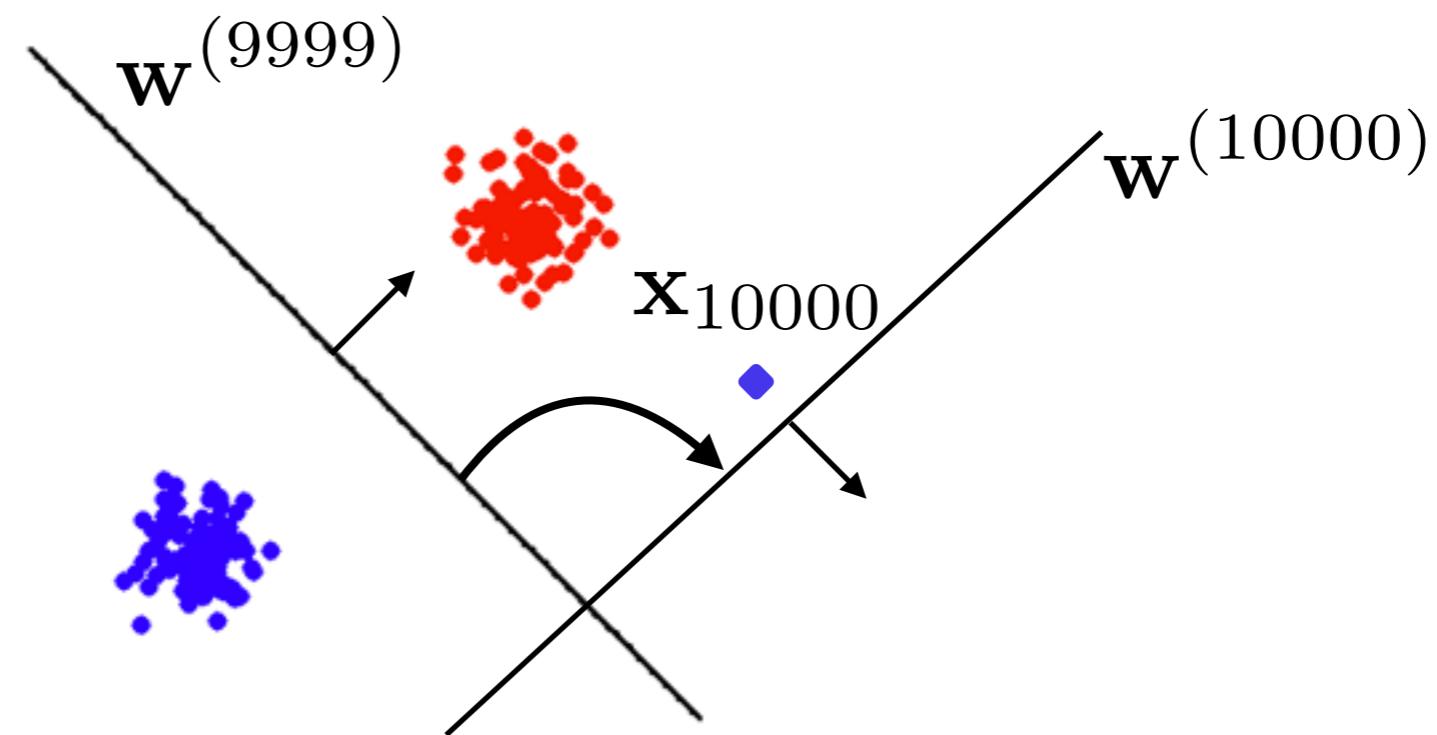
◆ **Overtraining:** test/validation accuracy rises and then falls

- ▶ Overtraining is a kind of overfitting



# A problem with perceptron training

- ◆ Problem: updates on later examples can take over
  - ▶ 10000 training examples
  - ▶ The algorithm learns weight vector on the first 100 examples
  - ▶ Gets the next 9899 points correct
  - ▶ Gets the 10000<sup>th</sup> point wrong, updates on the the weight vector
  - ▶ This completely ruins the weight vector (get 50% error)



- ◆ Voted and averaged perceptron (Freund and Schapire, 1999)

# Voted perceptron

Key idea: remember how long each weight vector survives

- ◆ Let,  $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(K)}$ , be the sequence of weights obtained by the perceptron learning algorithm.
- ◆ Let,  $c^{(1)}, c^{(2)}, \dots, c^{(K)}$ , be the survival times for each of these.
  - ▶ a weight that gets updated immediately gets  $c = 1$
  - ▶ a weight that survives another round gets  $c = 2$ , etc.
- ◆ Then,

$$\hat{y} = \text{sign} \left( \sum_{k=1}^K c^{(k)} \text{sign} \left( \mathbf{w}^{(k)T} \mathbf{x} \right) \right)$$

# Voted perceptron training algorithm

Input: training data  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$

Output: list of pairs  $(\mathbf{w}^{(1)}, c^{(1)}), (\mathbf{w}^{(2)}, c^{(2)}), \dots, (\mathbf{w}^{(K)}, c^{(K)})$

◆ Initialize:  $k = 0, c^{(1)} = 0, \mathbf{w}^{(1)} \leftarrow [0, \dots, 0]$

◆ for iter = 1,...,T

► for i = 1,...,n

- predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^{(k)T} \mathbf{x}_i > 0 \\ -1 & \text{if } \mathbf{w}^{(k)T} \mathbf{x}_i \leq 0 \end{cases}$$

- if  $y_i = \hat{y}_i$ ,  $c^{(k)} = c^{(k)} + 1$
- else,  $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + y_i \mathbf{x}_i$

$$c^{(k+1)} = 1$$

$$k = k + 1$$

# Voted perceptron training algorithm

Input: training data  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$

Output: list of pairs  $(\mathbf{w}^{(1)}, c^{(1)}), (\mathbf{w}^{(2)}, c^{(2)}), \dots, (\mathbf{w}^{(K)}, c^{(K)})$

◆ Initialize:  $k = 0, c^{(1)} = 0, \mathbf{w}^{(1)} \leftarrow [0, \dots, 0]$

◆ for iter = 1,...,T

► for i = 1,...,n

- predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^{(k)T} \mathbf{x}_i > 0 \\ -1 & \text{if } \mathbf{w}^{(k)T} \mathbf{x}_i \leq 0 \end{cases}$$

- if  $y_i = \hat{y}_i$ ,  $c^{(k)} = c^{(k)} + 1$
- else,  $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + y_i \mathbf{x}_i$

$$c^{(k+1)} = 1$$

$$k = k + 1$$

Better generalization,  
but not very practical

# Averaged perceptron

Key idea: remember how long each weight vector survives

- ◆ Let,  $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(K)}$ , be the sequence of weights obtained by the perceptron learning algorithm.
- ◆ Let,  $c^{(1)}, c^{(2)}, \dots, c^{(K)}$ , be the survival times for each of these.
  - ▶ a weight that gets updated immediately gets  $c = 1$
  - ▶ a weight that survives another round gets  $c = 2$ , etc.
- ◆ Then,

$$\hat{y} = \text{sign} \left( \sum_{k=1}^K c^{(k)} (\mathbf{w}^{(k)T} \mathbf{x}) \right) = \text{sign} (\bar{\mathbf{w}}^T \mathbf{x})$$

performs similarly, but much more practical

# Averaged perceptron training algorithm

Input: training data  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$

Output:  $\bar{\mathbf{w}}$

- ◆ Initialize:  $c = 0, \mathbf{w} = [0, \dots, 0], \bar{\mathbf{w}} = [0, \dots, 0]$
- ◆ for iter = 1,...,T
  - ▶ for i = 1,..,n
    - predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x}_i > 0 \\ -1 & \text{if } \mathbf{w}^T \mathbf{x}_i \leq 0 \end{cases}$$

- if  $y_i = \hat{y}_i, c = c + 1$
- else,  $\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} + c\mathbf{w}$

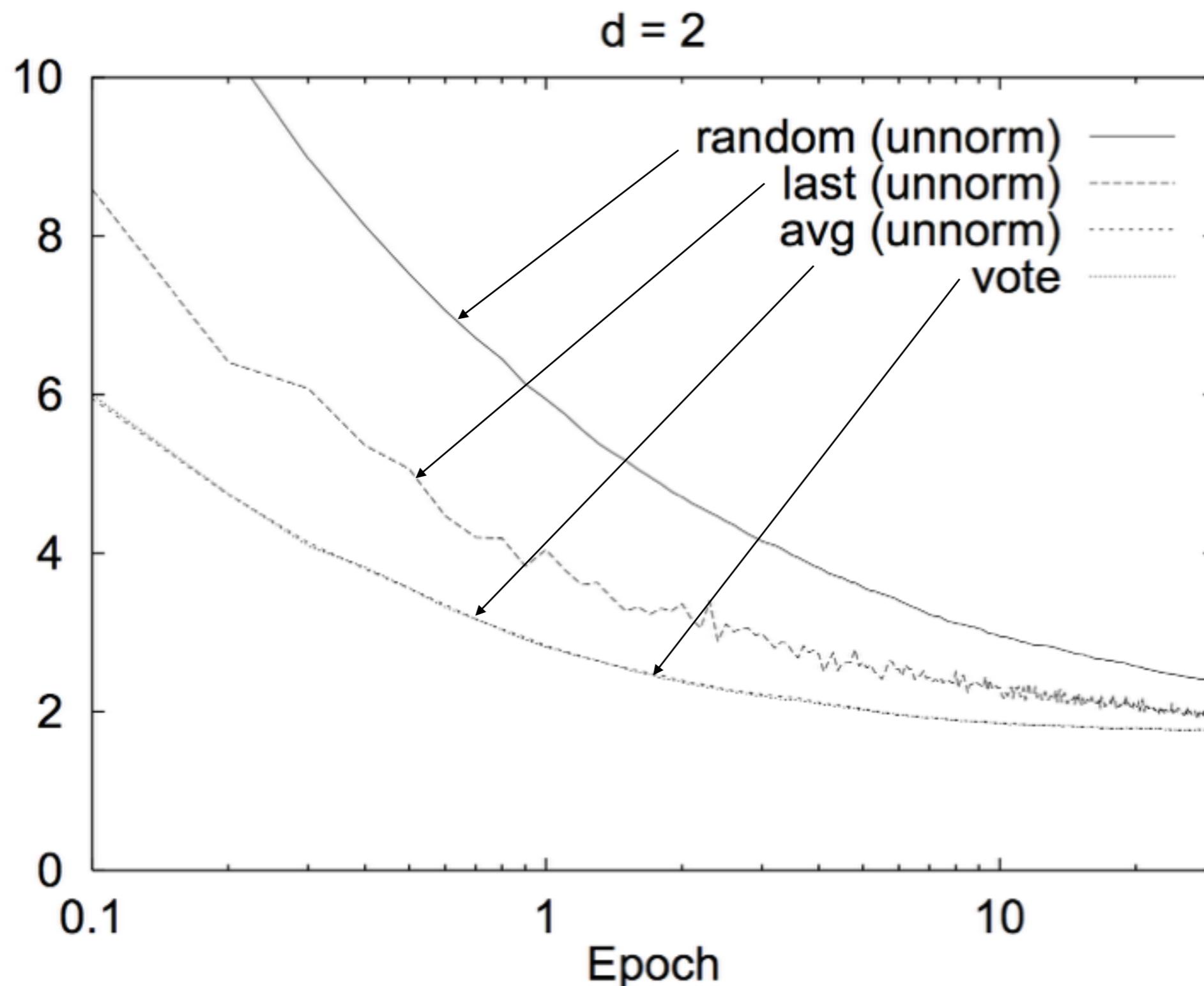
$$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$$
$$c = 1$$

update average

- ◆ Return  $\bar{\mathbf{w}} + c\mathbf{w}$

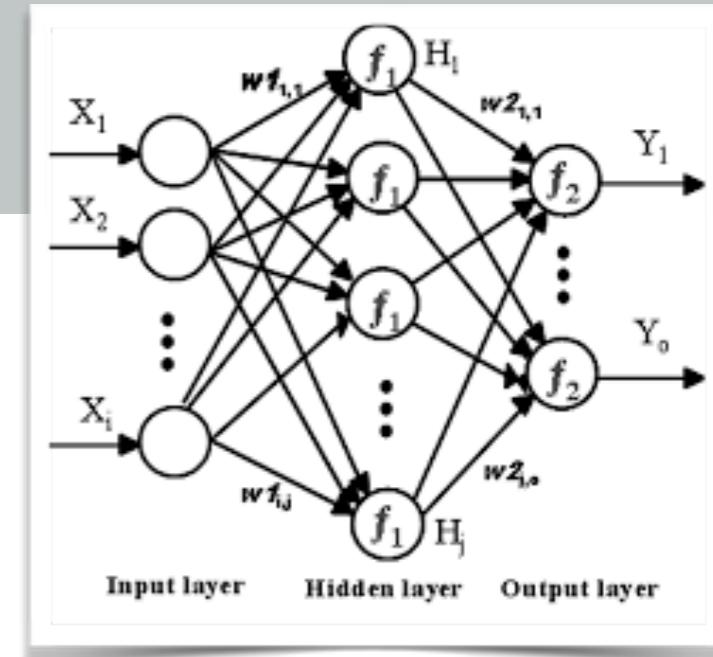
# Comparison of perceptron variants

MNIST dataset (Figure from Freund and Schapire 1999)

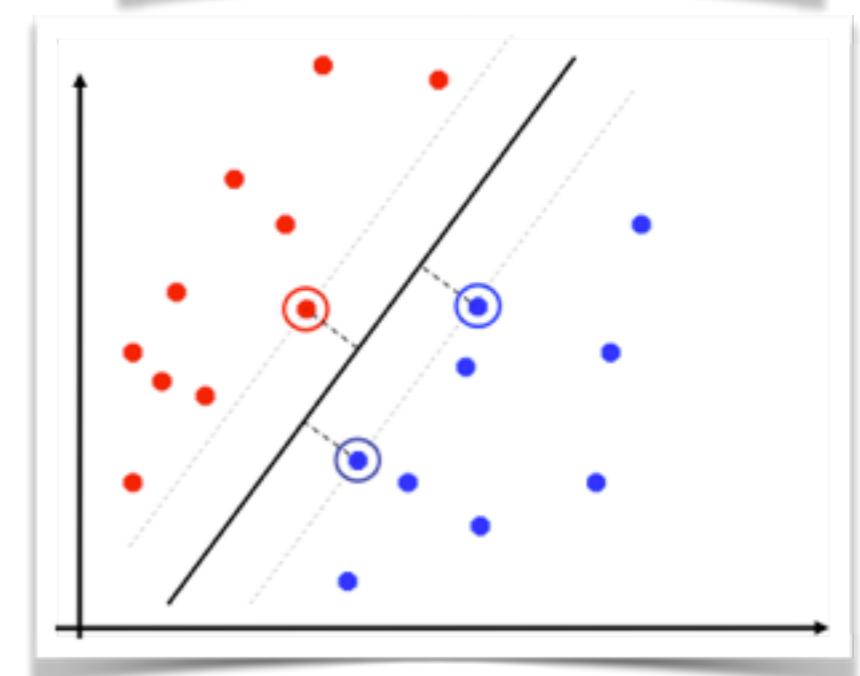


# Improving perceptrons

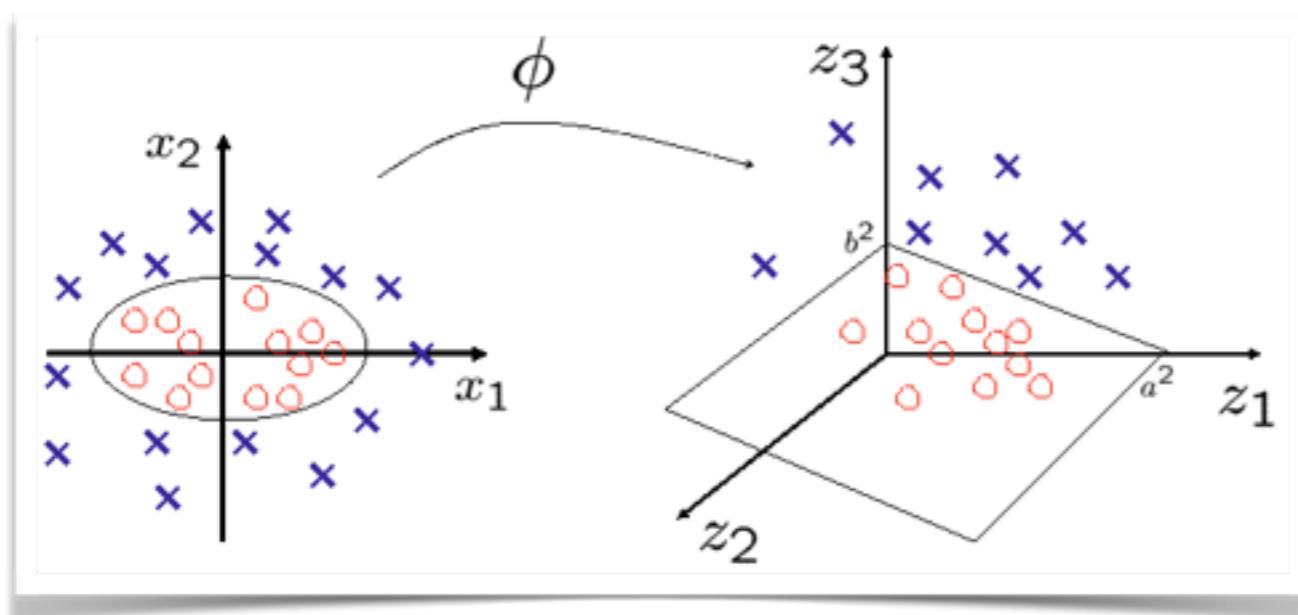
- ◆ **Multilayer perceptrons:** to learn non-linear functions of the input (neural networks)



- ◆ **Separators with good margins:** improves generalization ability of the classifier (support vector machines)



- ◆ **Feature-mapping:** to learn non-linear functions of the input using a perceptron
  - ▶ we will learn do this efficiently using **kernels**



$$\phi : (x_1, x_2) \rightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$
$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

# Slides credit

- ◆ Some slides adapted from Dan Klein at UC Berkeley and CML book by Hal Daume
- ◆ Figure comparing various perceptrons are from Freund and Schapire