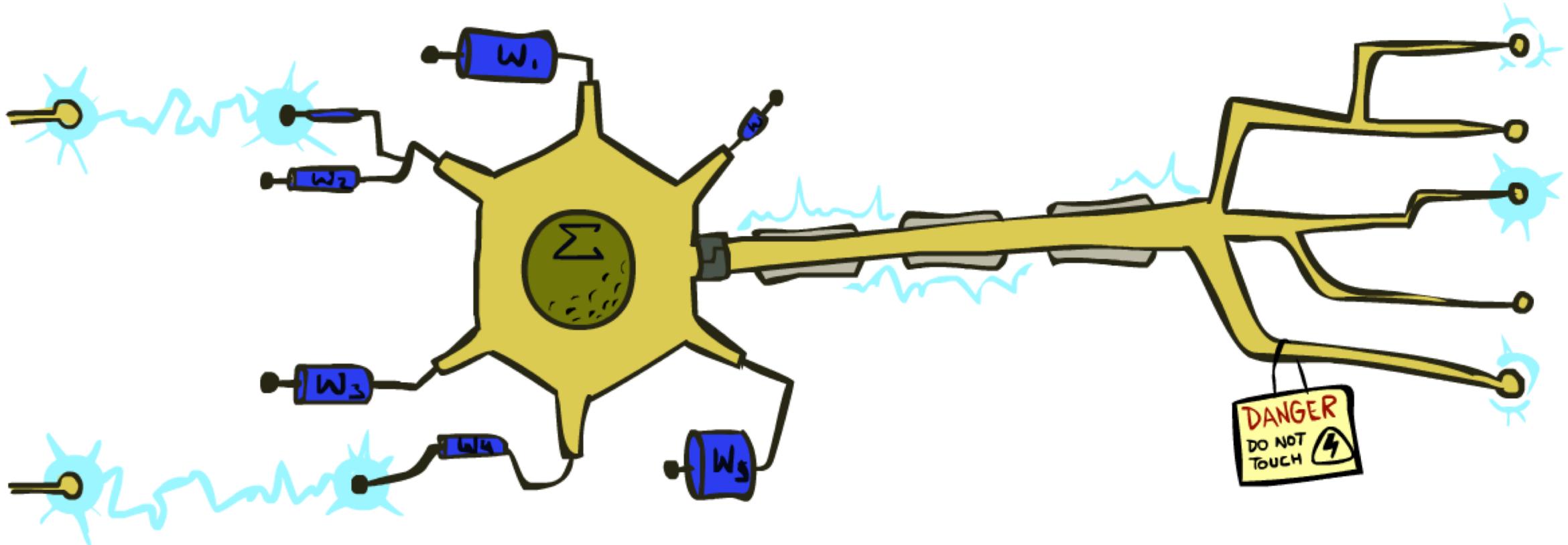


CS 188: Artificial Intelligence

Perceptrons and Logistic Regression



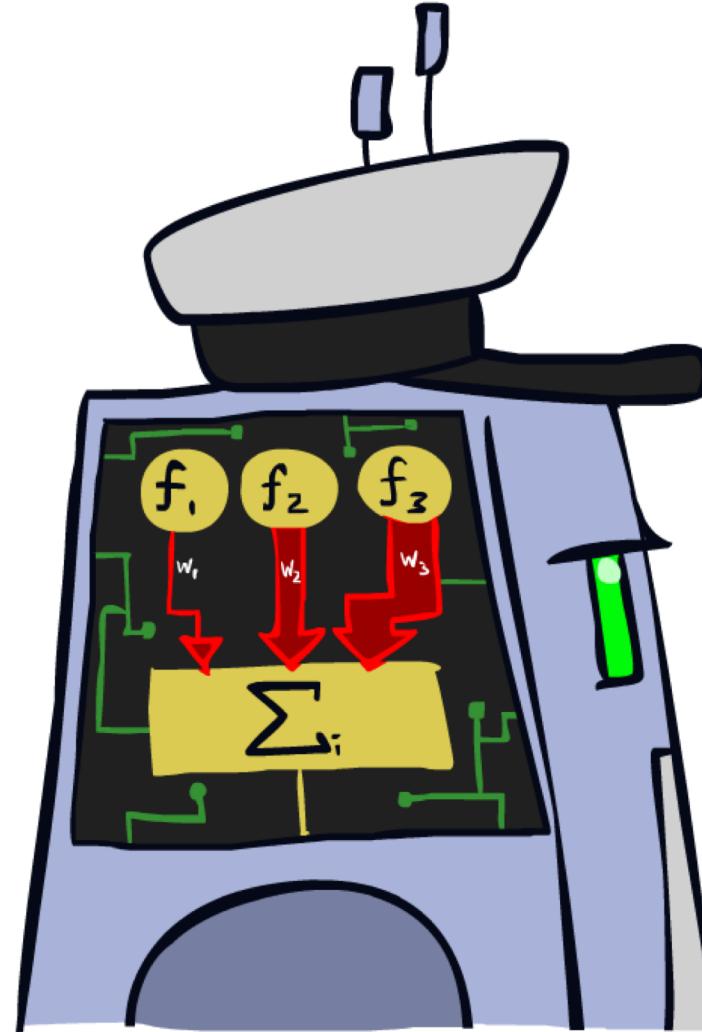
Instructor: Mesut XiaoCheng Yang, Carl Qi

University of California, Berkeley

Recap: Naïve Bayes

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them

Linear Classifiers



Feature Vectors

x

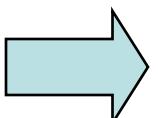
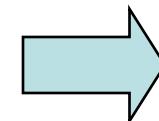
```
Hello,  
  
Do you want free printr  
cartridges? Why pay more  
when you can get them  
ABSOLUTELY FREE! Just
```

$f(x)$

$$\begin{Bmatrix} \# \text{ free} & : 2 \\ \text{YOUR_NAME} & : 0 \\ \text{MISSPELLED} & : 2 \\ \text{FROM_FRIEND} & : 0 \\ \dots \end{Bmatrix}$$

y

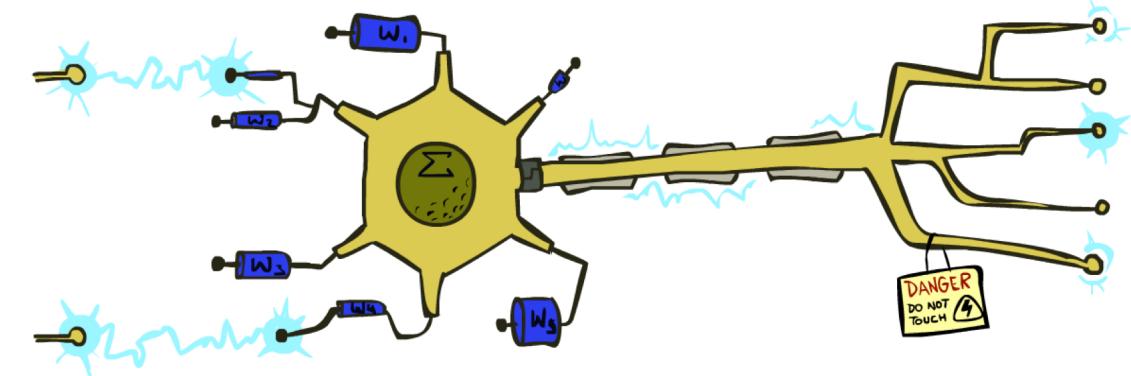
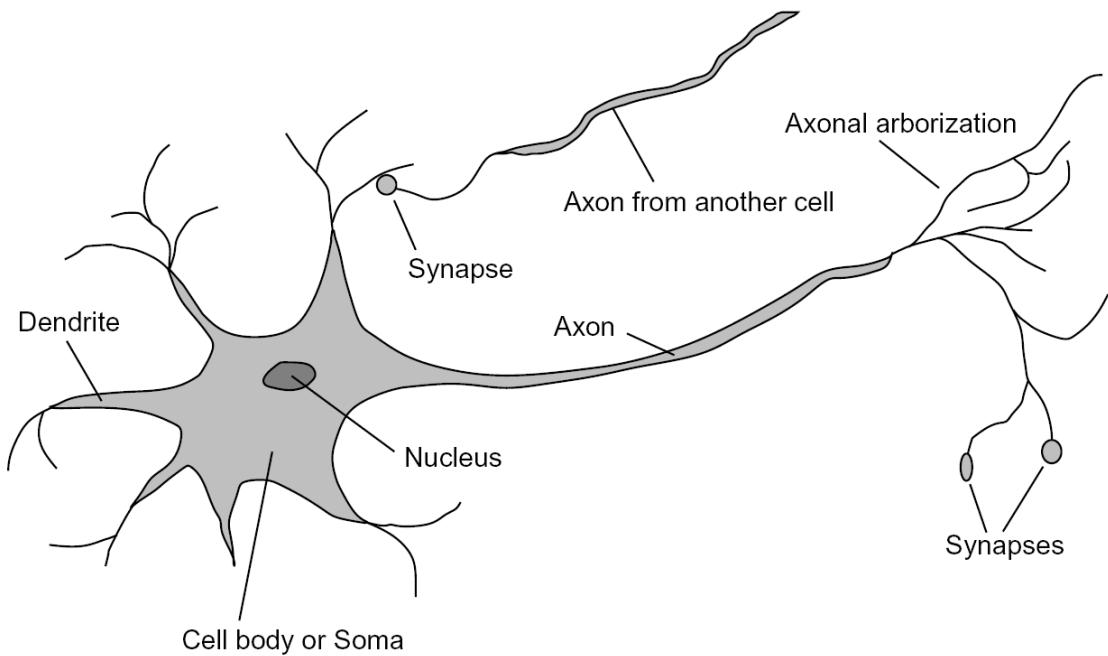
SPAM
or
+


$$\begin{Bmatrix} \text{PIXEL-7,12} & : 1 \\ \text{PIXEL-7,13} & : 0 \\ \dots \\ \text{NUM_LOOPS} & : 1 \\ \dots \end{Bmatrix}$$


“2”

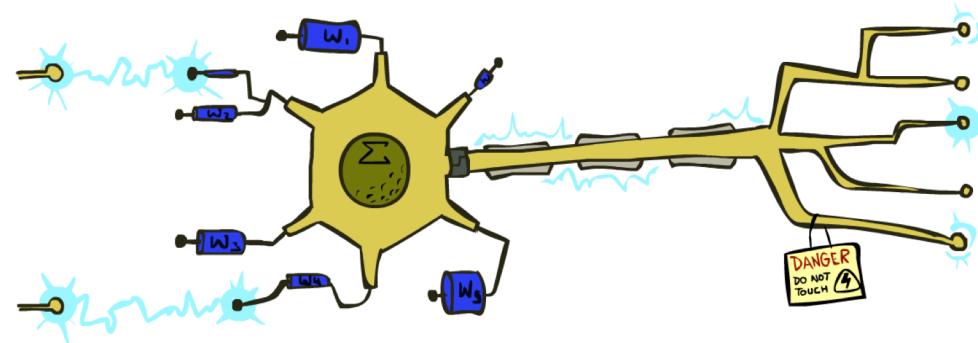
Some (Simplified) Biology

- Very loose inspiration: human neurons



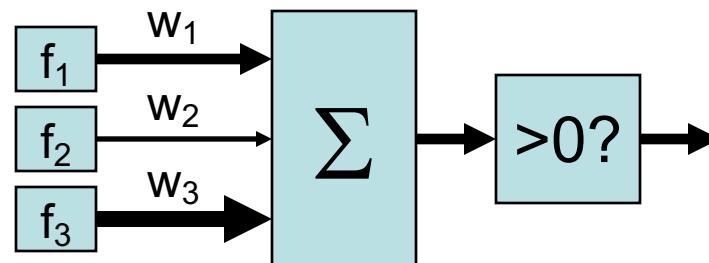
Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**



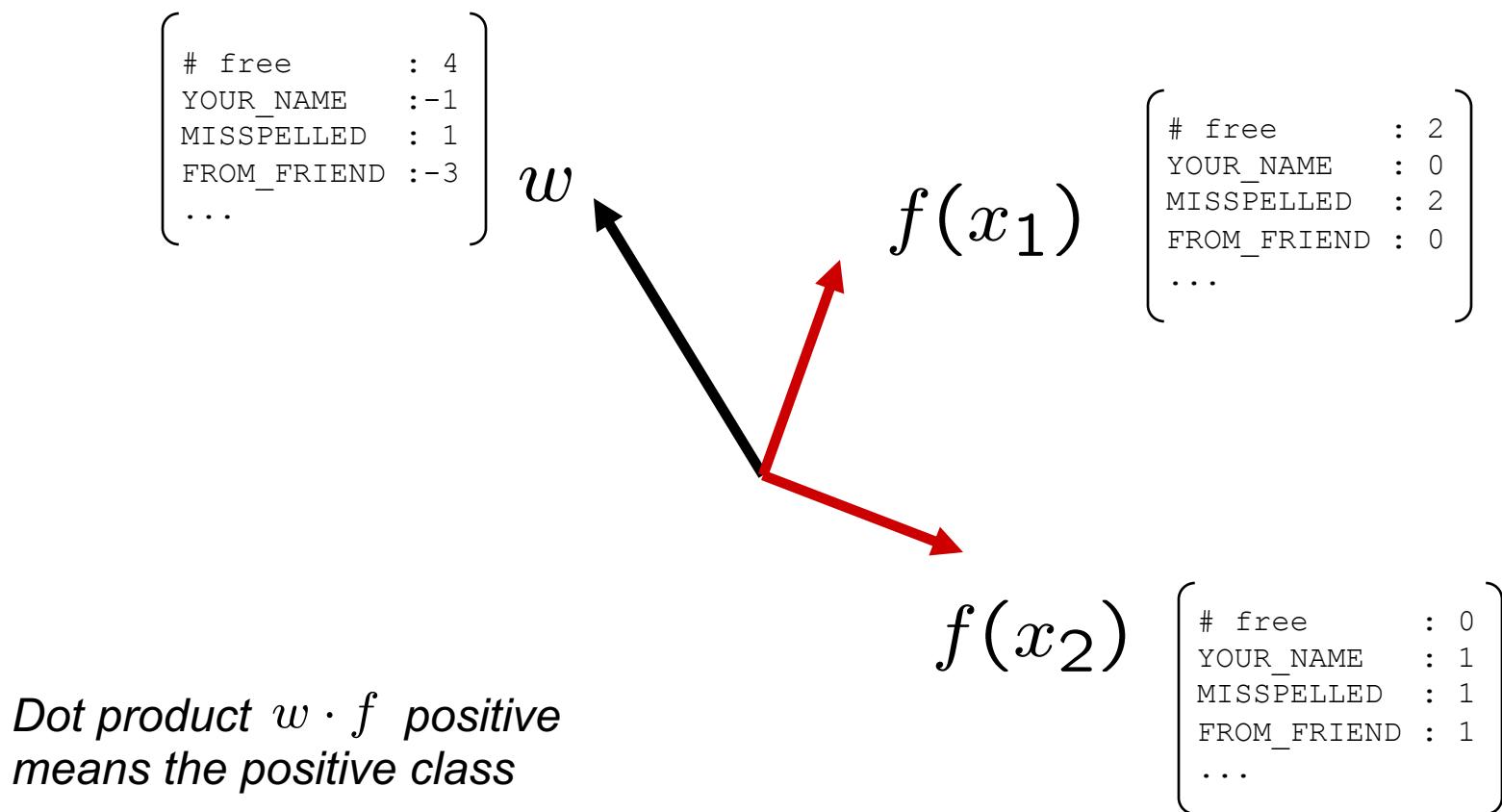
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
 - Positive, output +1
 - Negative, output -1

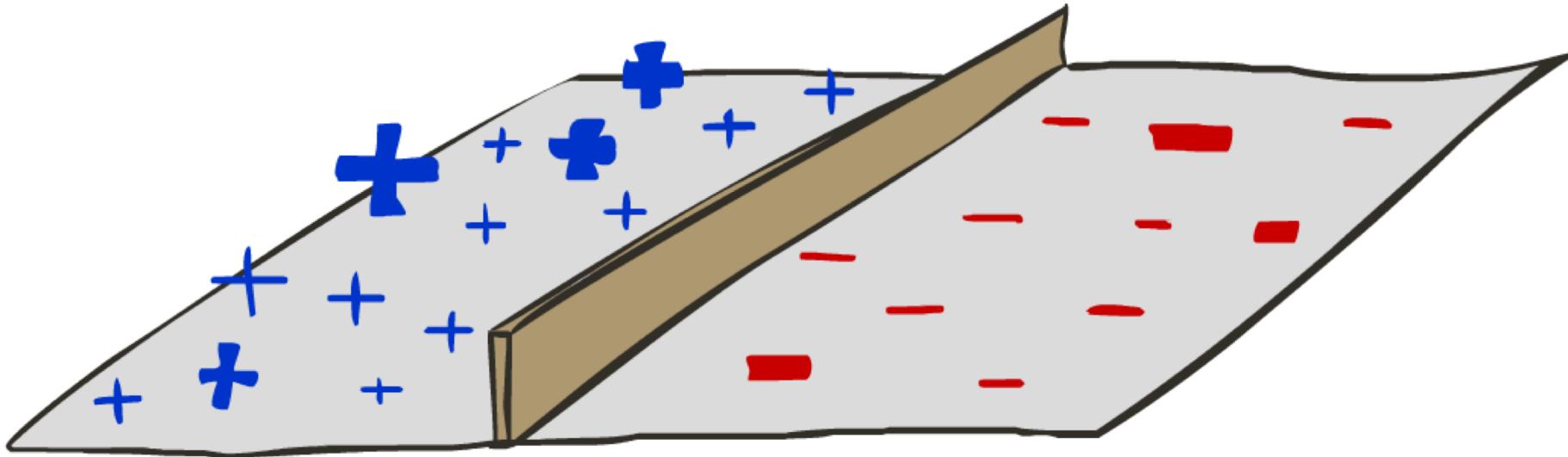


Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples



Decision Rules



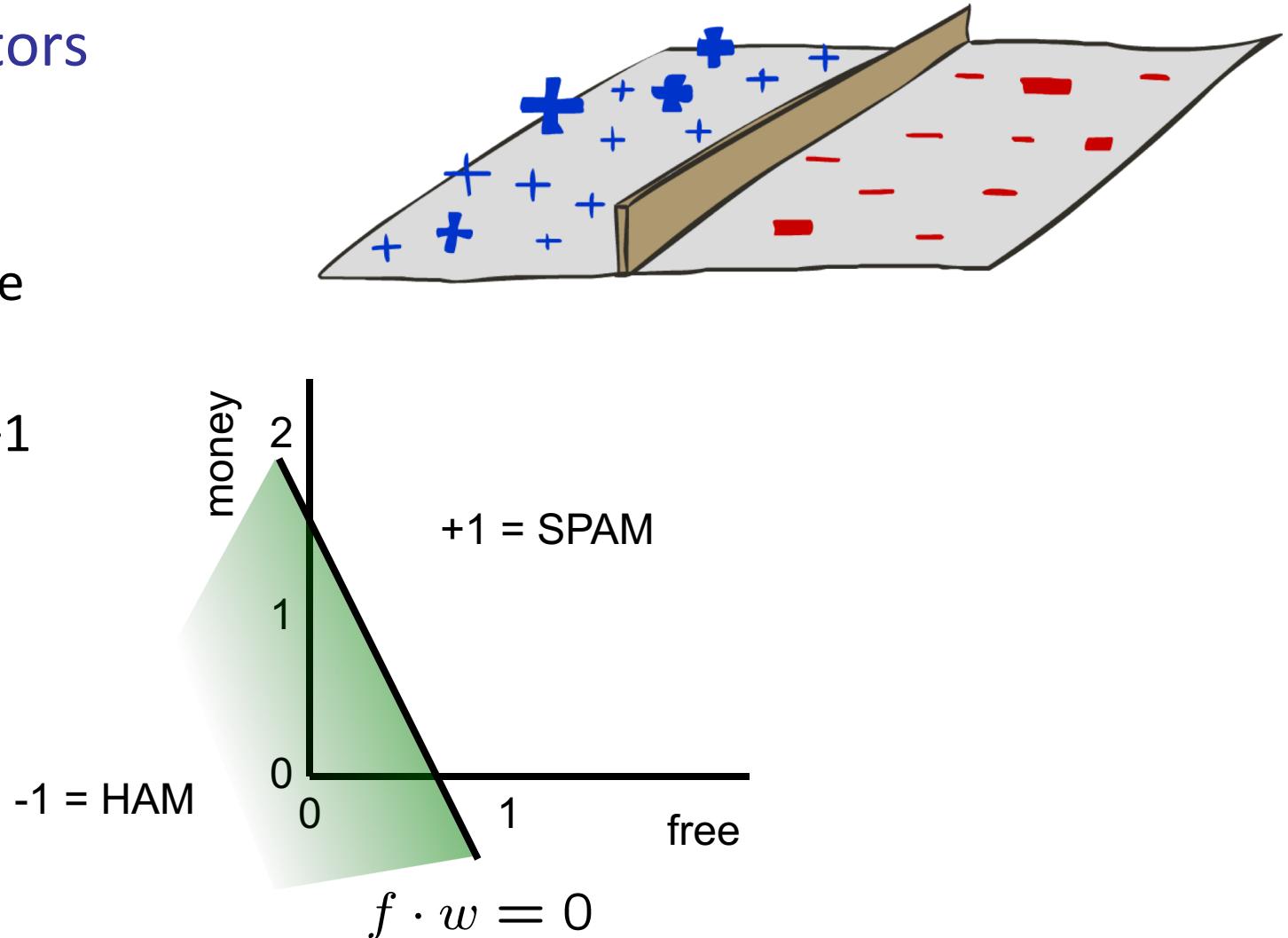
Binary Decision Rule

- In the space of feature vectors

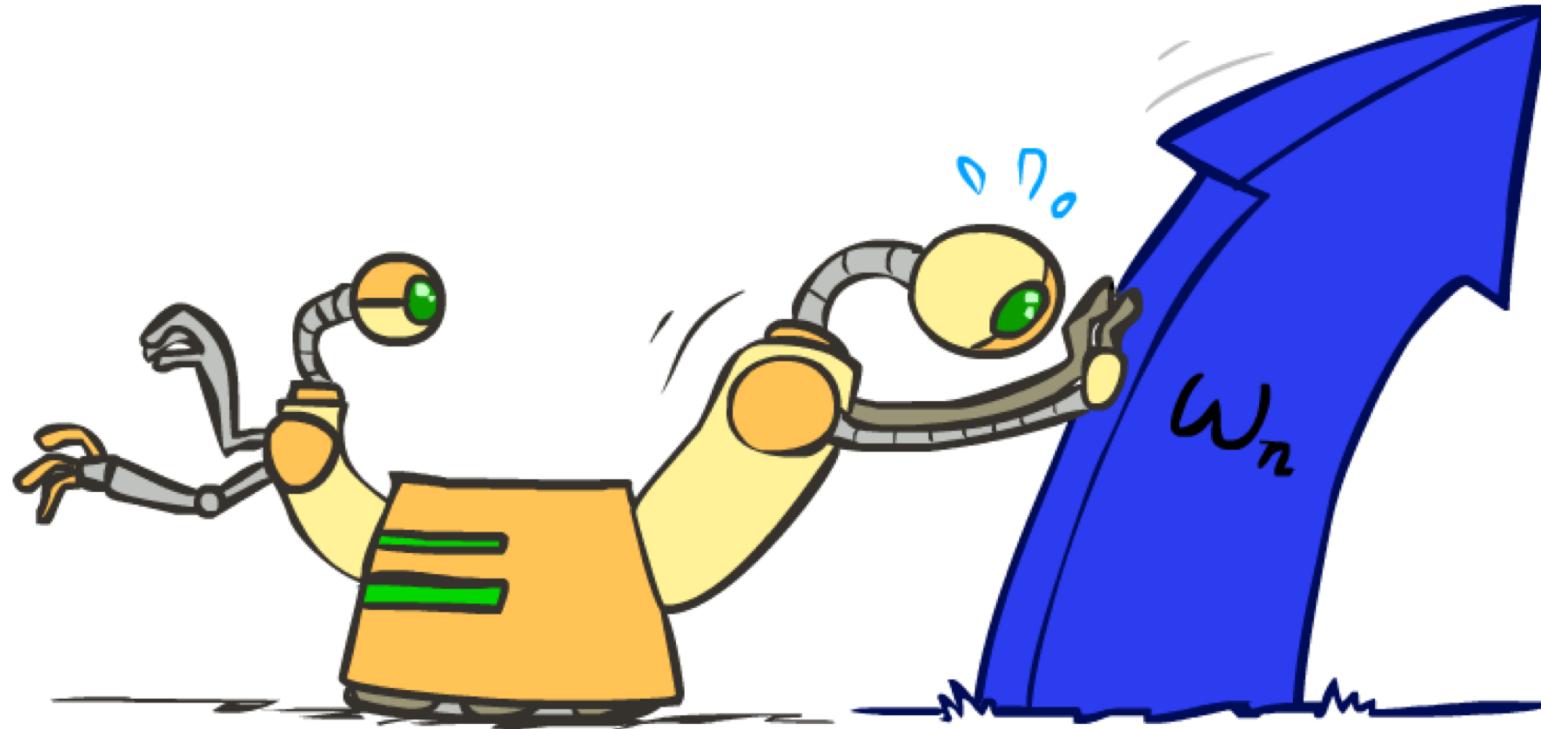
- Examples are points
- Any decision boundary is a hyperplane orthogonal to the weight vector
- One side corresponds to $Y=+1$
- Other corresponds to $Y=-1$

w

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

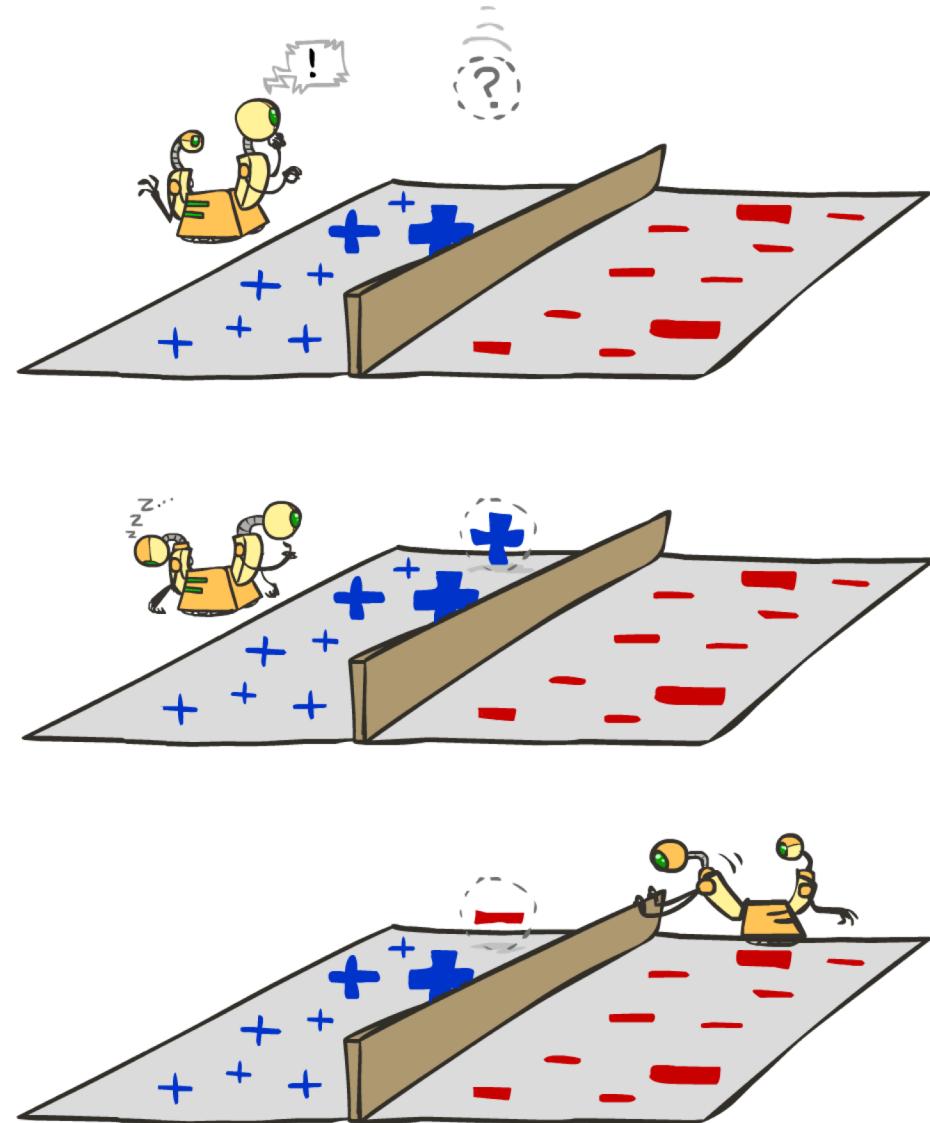


Weight Updates



Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
 - Classify with current weights
 - If correct (i.e., $y=y^*$), no change!
 - If wrong: adjust the weight vector



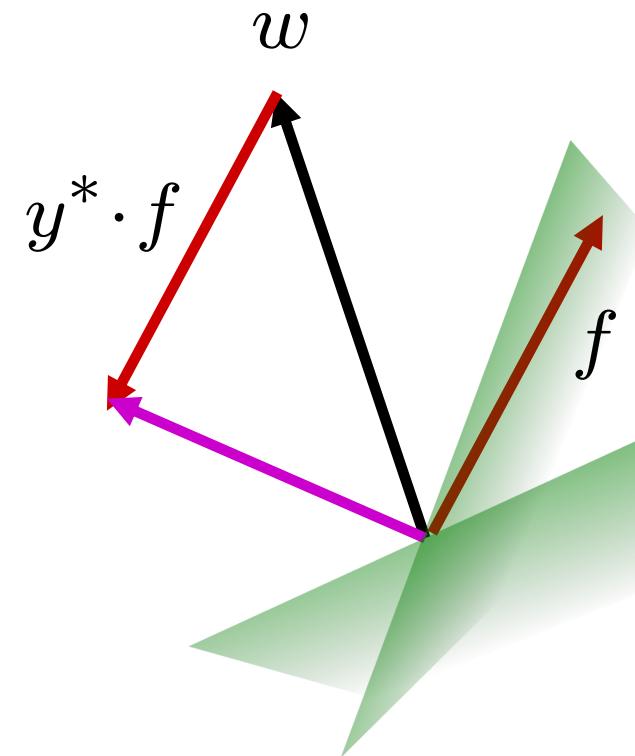
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
 - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

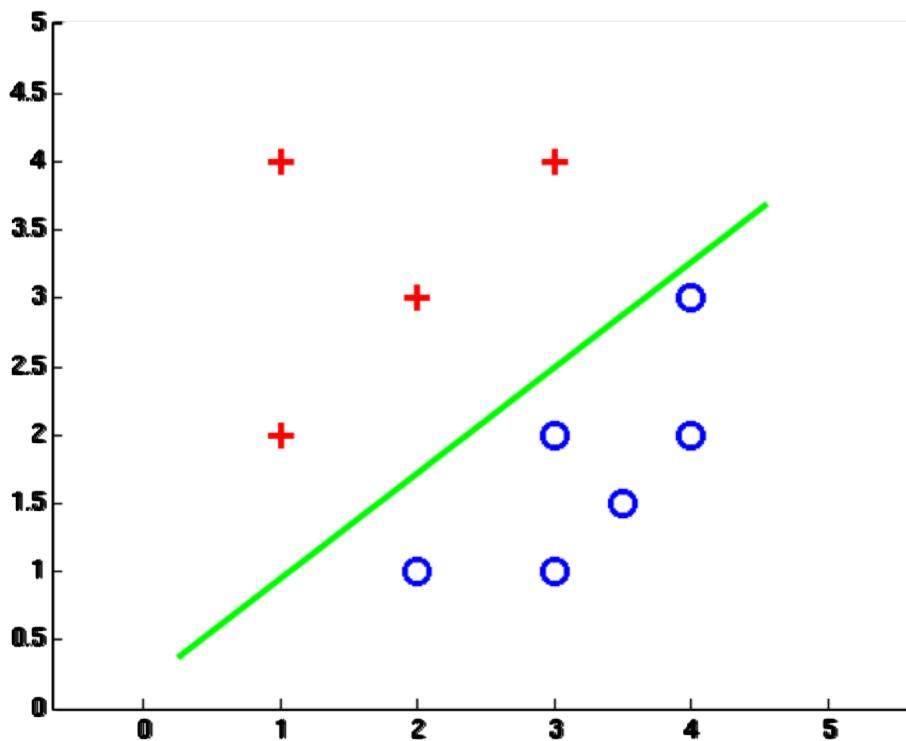
- If correct (i.e., $y=y^*$), no change!
- If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y^* is -1.

$$w = w + y^* \cdot f$$



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

- If we have multiple classes:
 - A weight vector for each class:

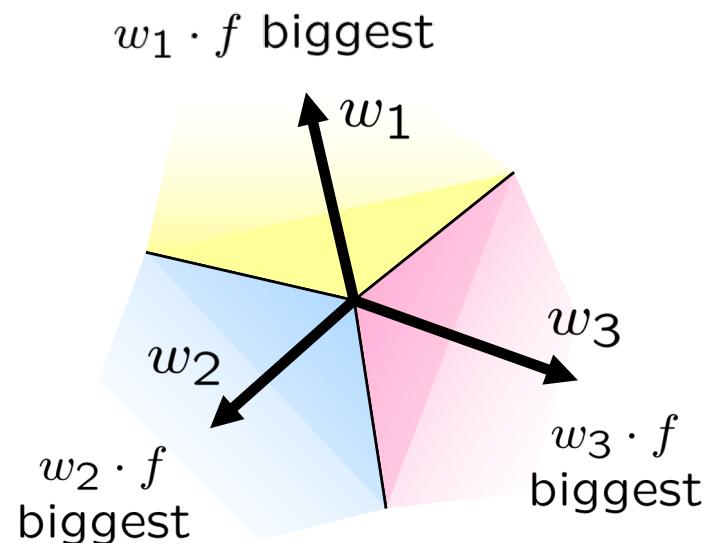
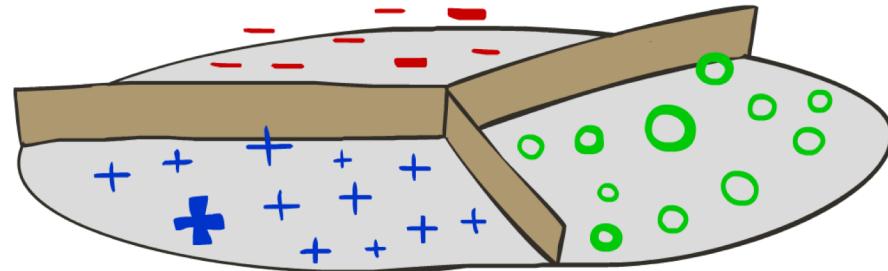
$$w_y$$

- Score (activation) of a class y :

$$w_y \cdot f(x)$$

- Prediction highest score wins

$$y = \arg \max_y w_y \cdot f(x)$$



Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

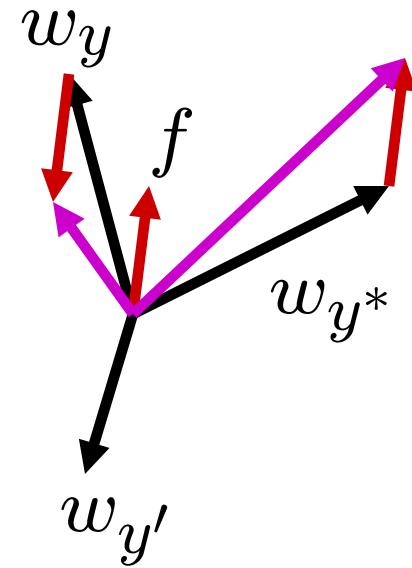
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg \max_y w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$



Example: Multiclass Perceptron

“win the vote” [1 1 0 1 1]

“win the election” [1 1 0 0 1]

“win the game” [1 1 1 0 1]

w_{SPORTS}

| | | |
|----------|----|----|
| 1 | -2 | -2 |
| BIAS : 1 | 0 | 1 |
| win : 0 | -1 | 0 |
| game : 0 | 0 | 1 |
| vote : 0 | -1 | -1 |
| the : 0 | -1 | 0 |
| ... | | |

$w_{POLITICS}$

| | | |
|----------|---|----|
| 0 | 3 | 3 |
| BIAS : 0 | 1 | 0 |
| win : 0 | 1 | 0 |
| game : 0 | 0 | -1 |
| vote : 0 | 1 | 1 |
| the : 0 | 1 | 0 |
| ... | | |

w_{TECH}

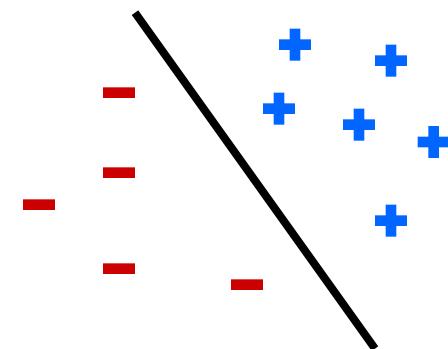
| | |
|----------|---|
| 0 | 0 |
| BIAS : 0 | |
| win : 0 | |
| game : 0 | |
| vote : 0 | |
| the : 0 | |
| ... | |

Properties of Perceptrons

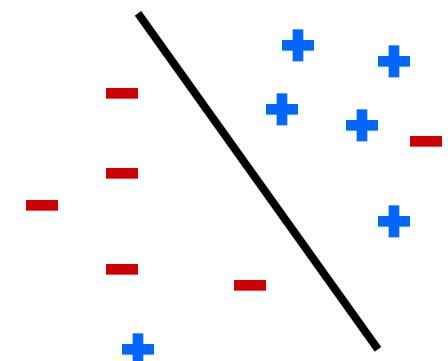
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2} \quad \begin{matrix} \text{number of features} \\ \text{margin} \end{matrix}$$

Separable

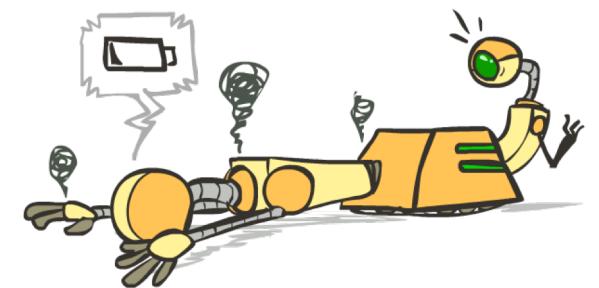
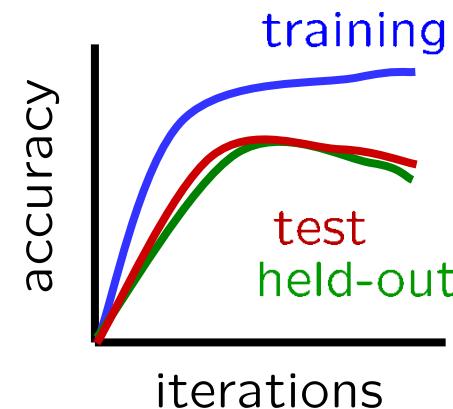
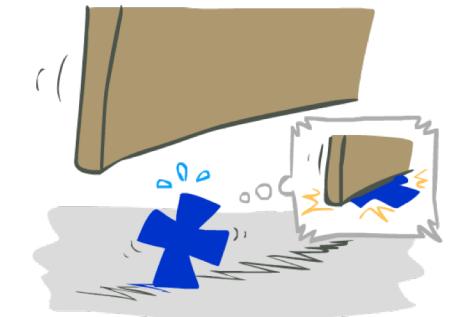
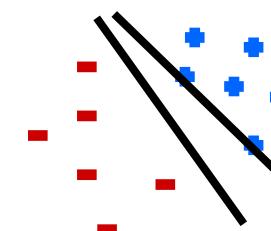
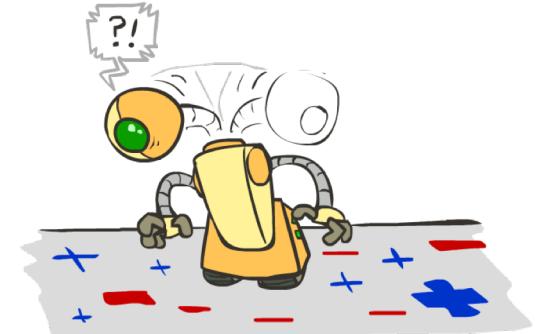
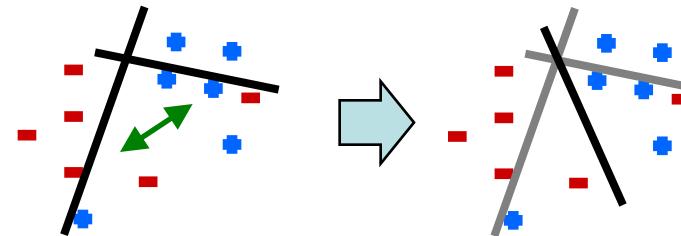


Non-Separable

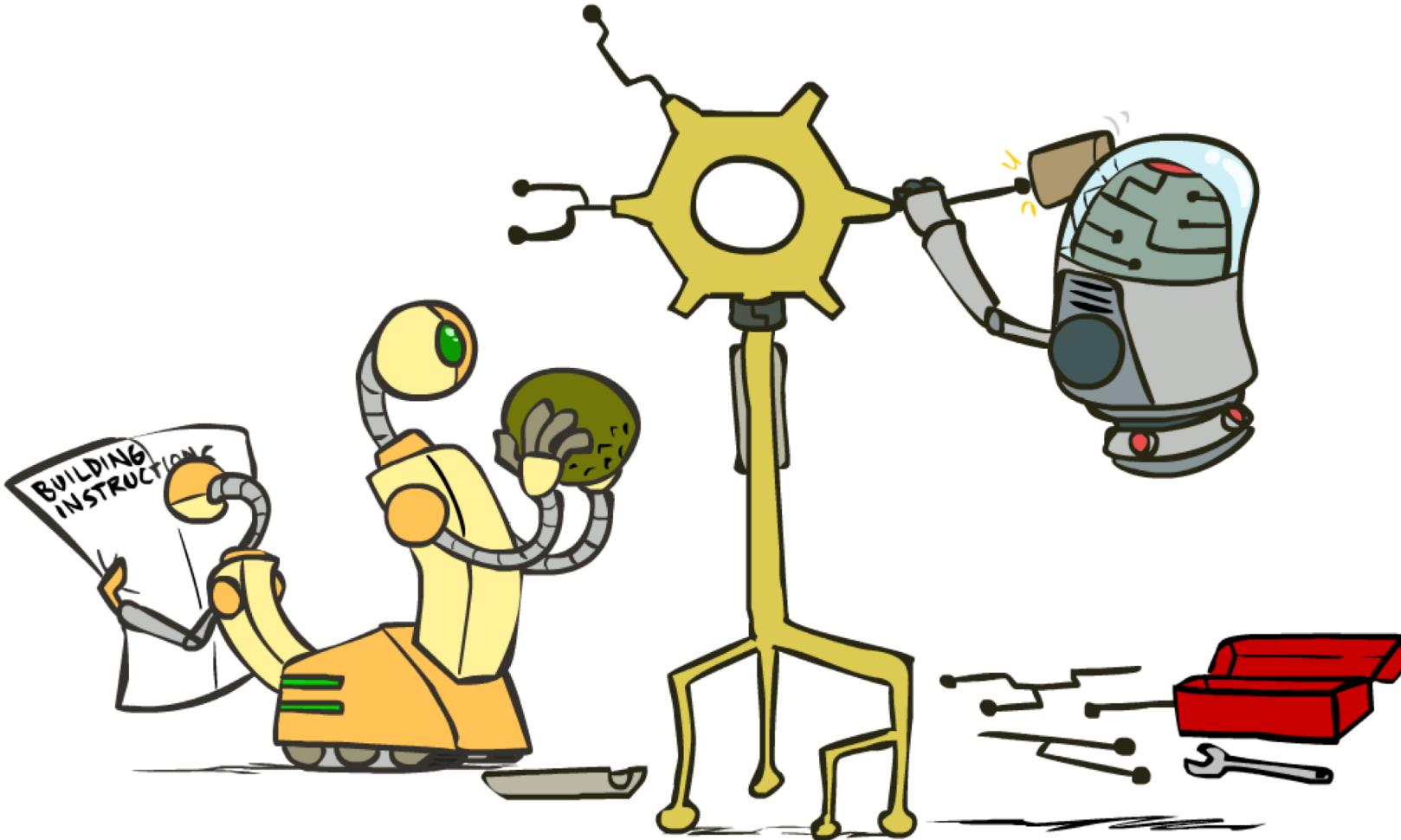


Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
 - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a “barely” separating solution
- Overtraining: test / held-out accuracy usually rises, then falls
 - Overtraining is a kind of overfitting

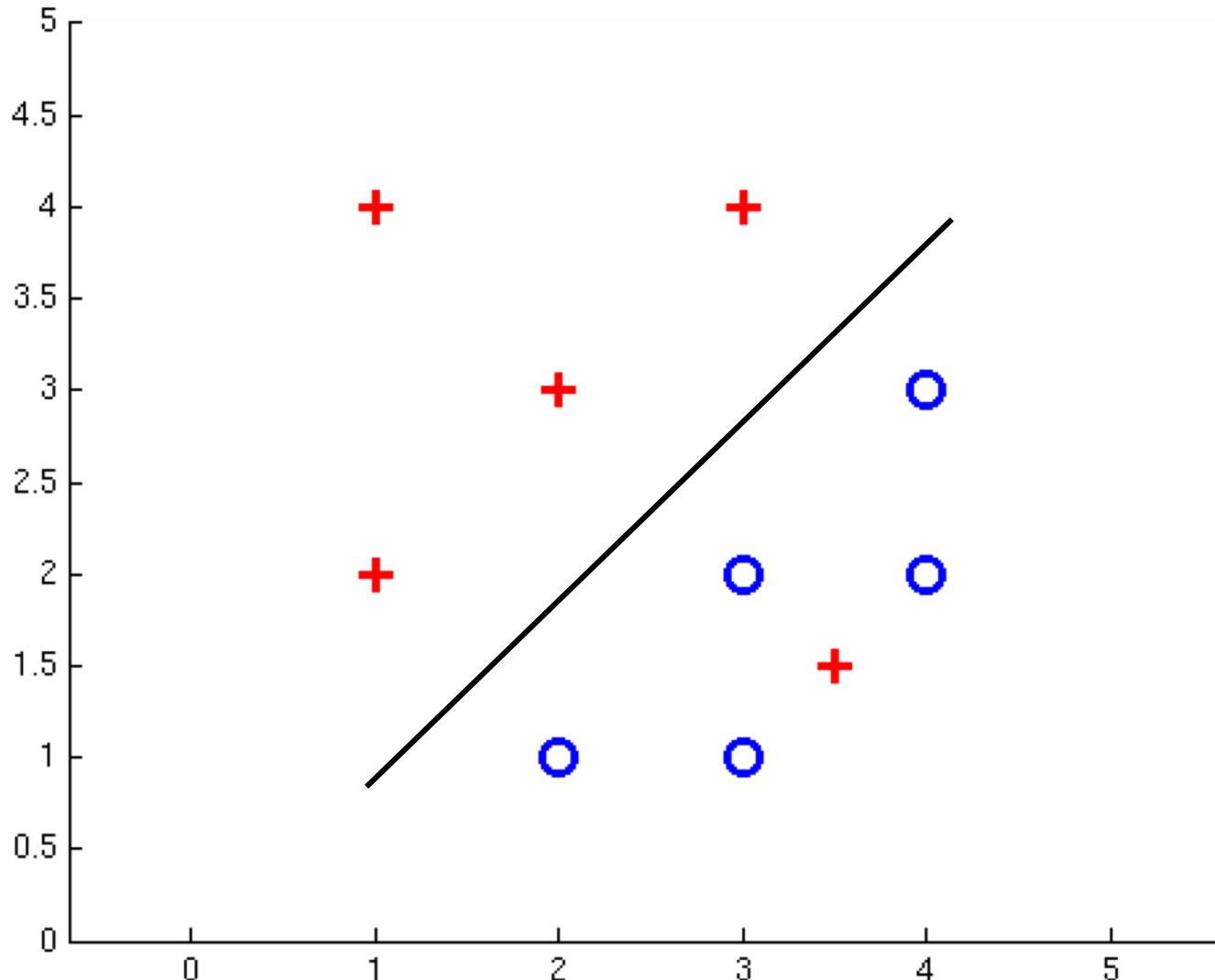


Improving the Perceptron

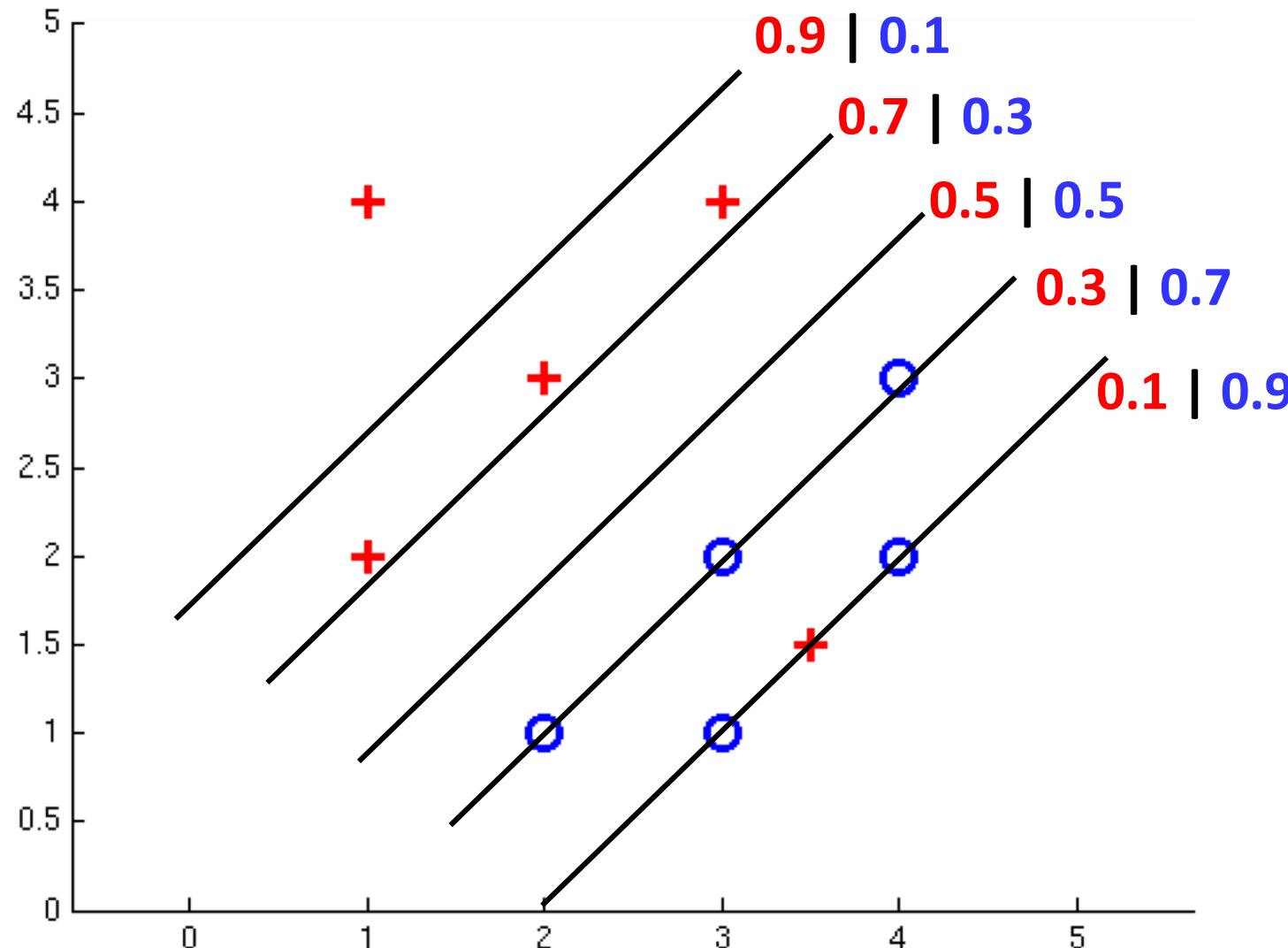


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



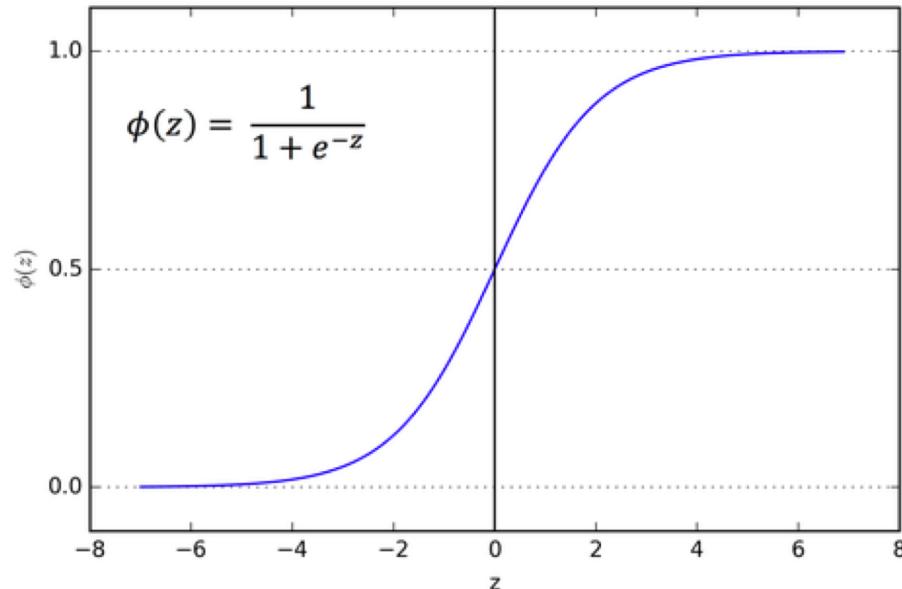
Non-Separable Case: Probabilistic Decision



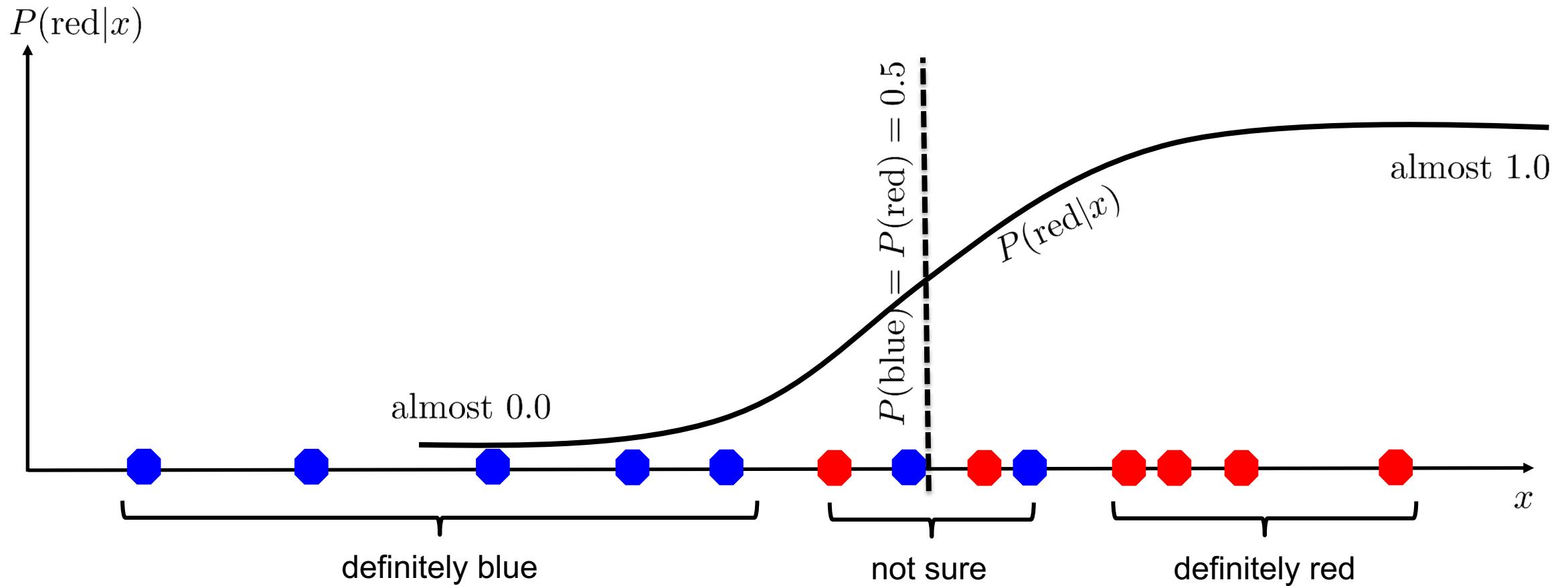
How to get probabilistic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ very positive \rightarrow want probability going to 1
- If $z = w \cdot f(x)$ very negative \rightarrow want probability going to 0
- Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



A 1D Example



$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

probability increases exponentially as we move away from boundary

normalizer

Best w?

- Maximum likelihood estimation:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

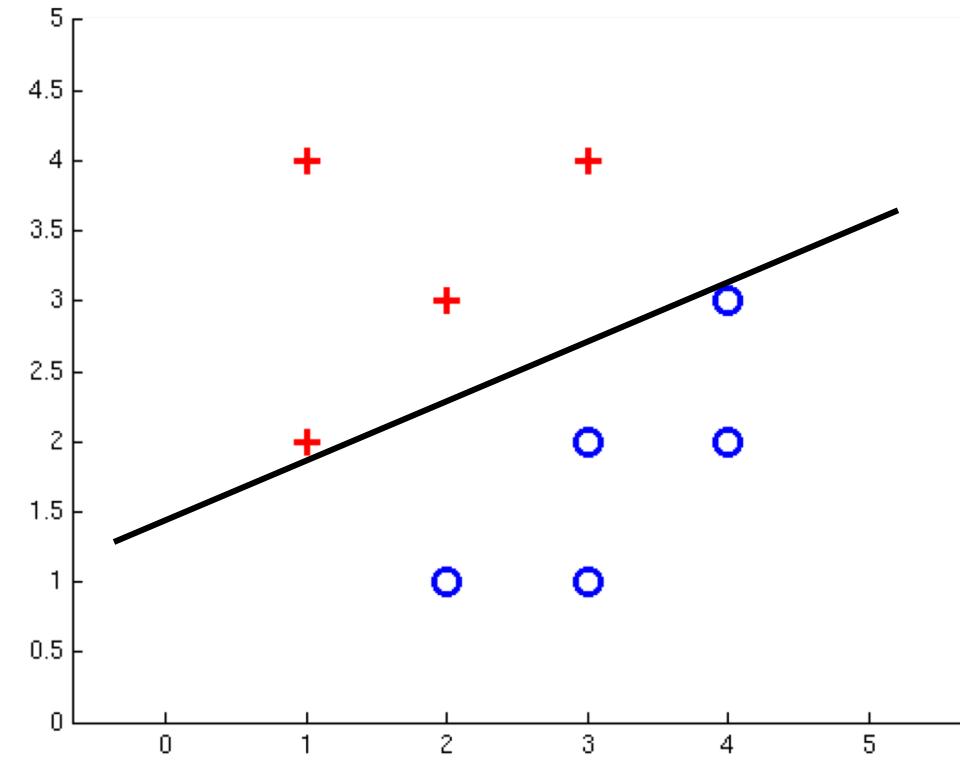
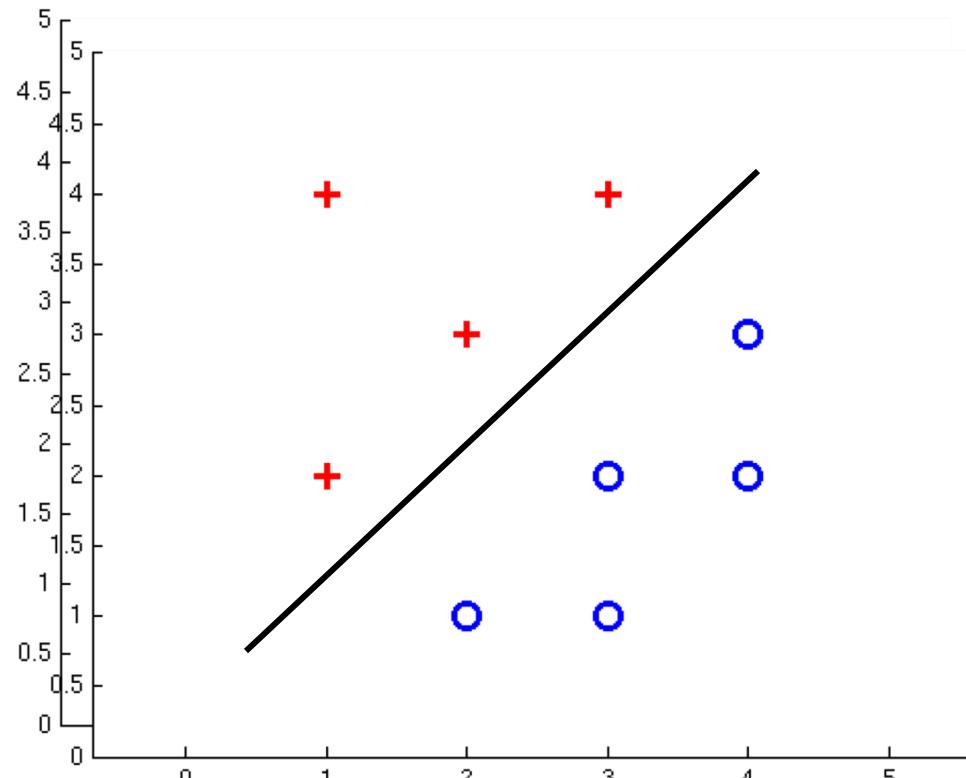
with:

$$P(y^{(i)} = +1 | x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

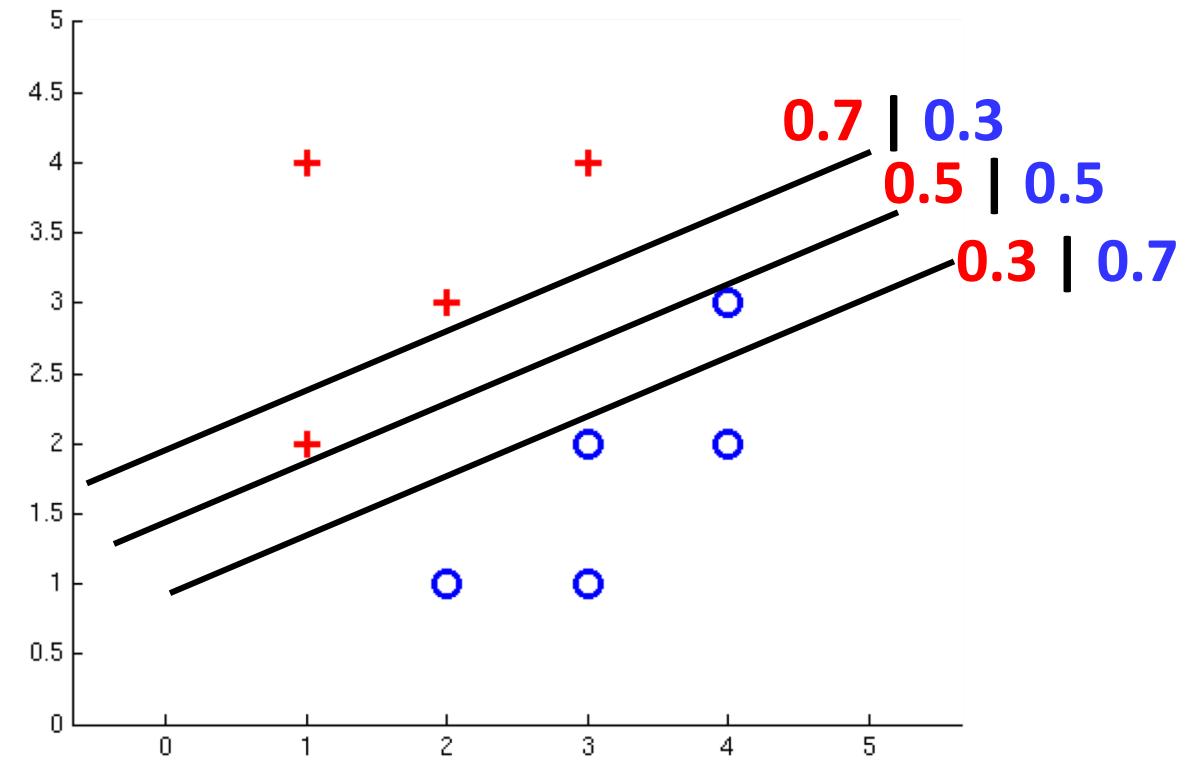
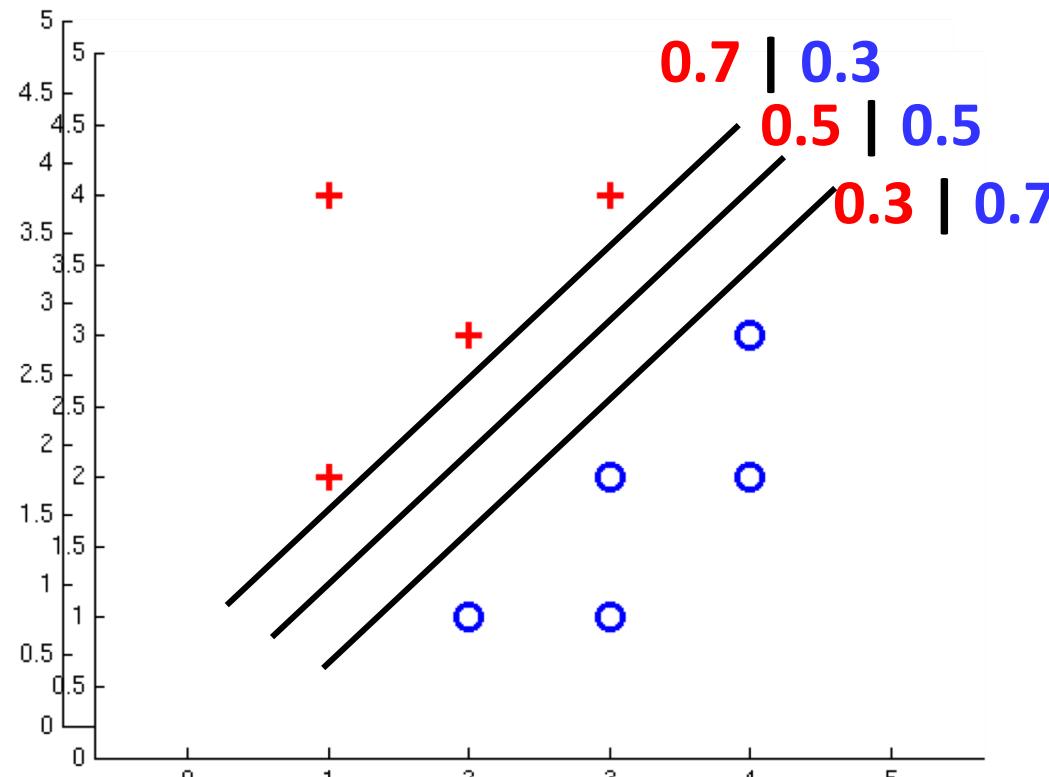
$$P(y^{(i)} = -1 | x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

= Logistic Regression

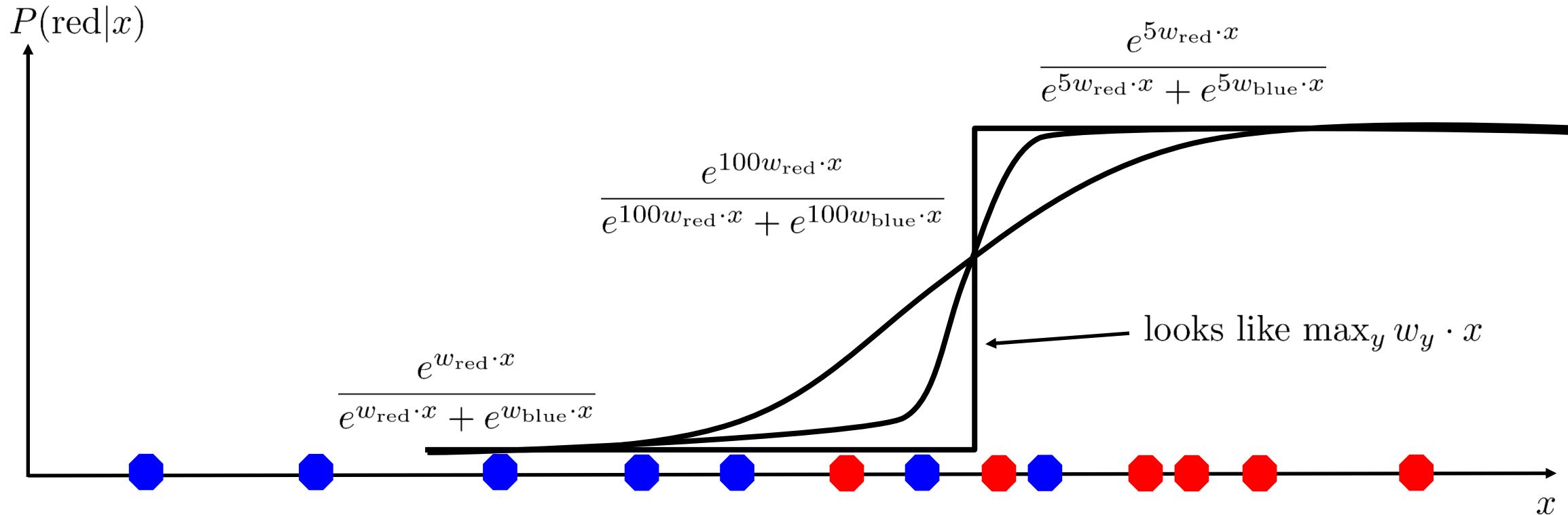
Separable Case: Deterministic Decision – Many Options



Separable Case: Probabilistic Decision – Clear Preference



The Soft Max



$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

Multiclass Logistic Regression

- Recall Perceptron:

- A weight vector for each class:

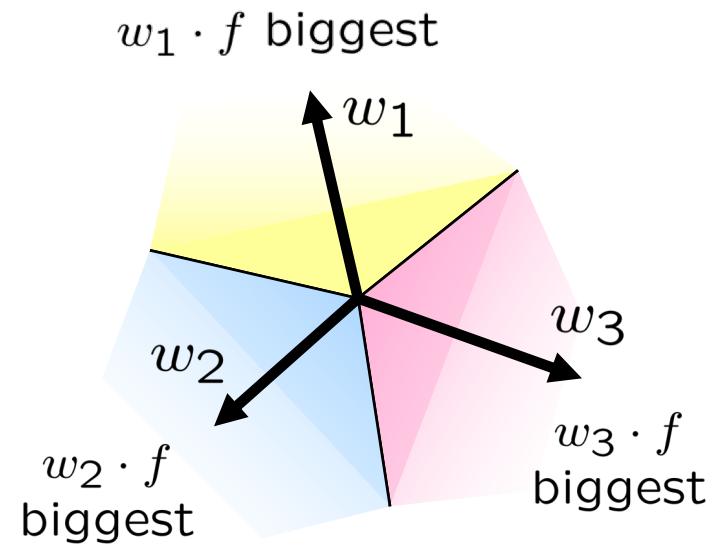
$$w_y$$

- Score (activation) of a class y :

$$w_y \cdot f(x)$$

- Prediction highest score wins

$$y = \arg \max_y w_y \cdot f(x)$$



- How to make the scores into probabilities?

$$z_1, z_2, z_3 \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}}_{\text{softmax activations}}$$

original activations

Best w?

- Maximum likelihood estimation:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

with:

$$P(y^{(i)} | x^{(i)}; w) = \frac{e^{w_y \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

= Multi-Class Logistic Regression

Next Lecture

- Optimization
 - i.e., how do we solve:

$$\max_w \text{ll}(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$$