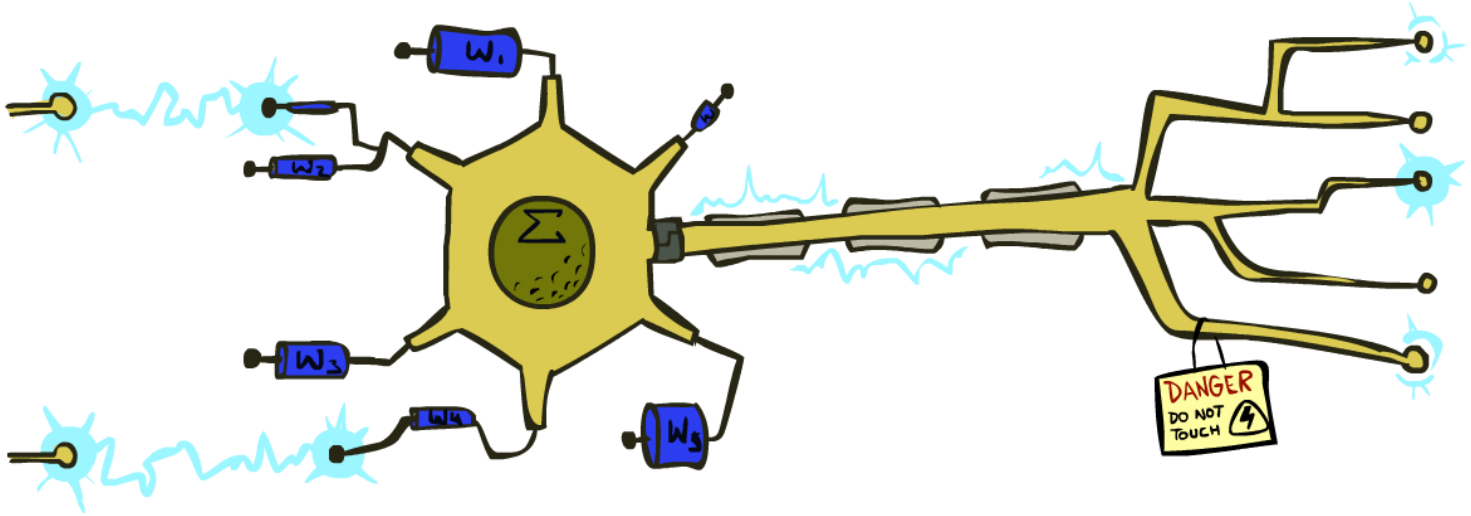
Artificial Intelligence - INFOF311

Perceptrons and regression

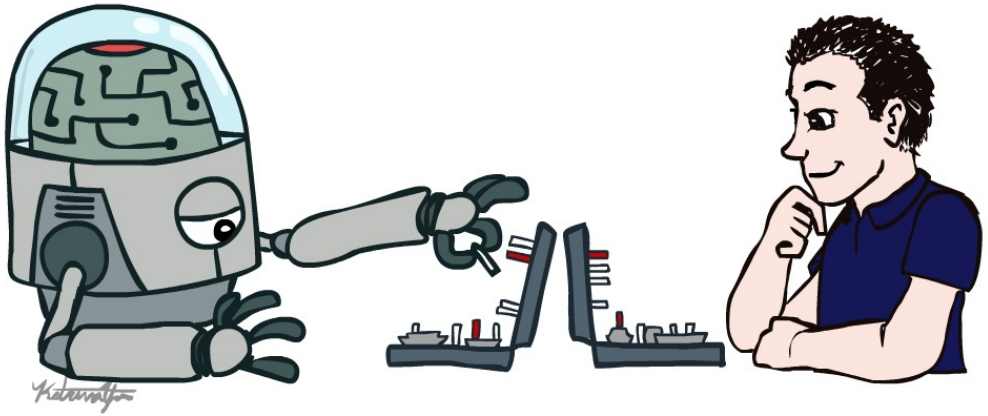


Instructor : Tom Lenaerts

Acknowledgement

We thank Stuart Russell for his generosity in allowing us to use the slide set of the UC Berkeley Course CS188, Introduction to Artificial Intelligence. These slides were created by Dan Klein, Pieter Abbeel and Anca Dragan for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.

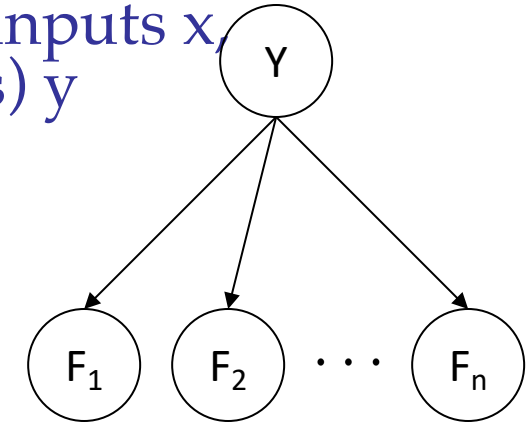
Center for Human-Compatible Artificial Intelligence



The slides for INFOF311 are slightly modified versions of the slides of the spring and summer CS188 sessions in 2021 and 2022

Last Time

- Classification: given inputs x, predict labels (classes) y



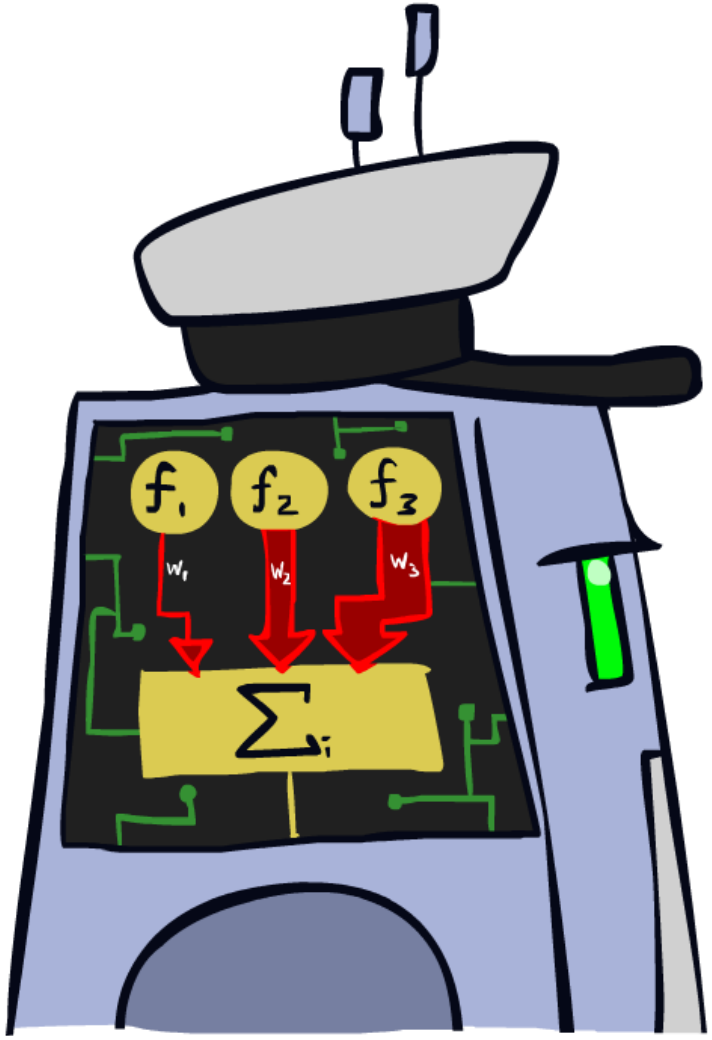
- Naïve Bayes

$$P(Y|F_{0,0} \dots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$$

- Parameter estimation:
  - MLE

- Training set, held-out set, test set

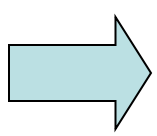
Linear Classifiers



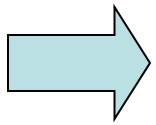
Feature Vectors

x $f(x)$ y

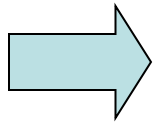
```
Hello,

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



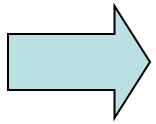
```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```



SPAM
or
+



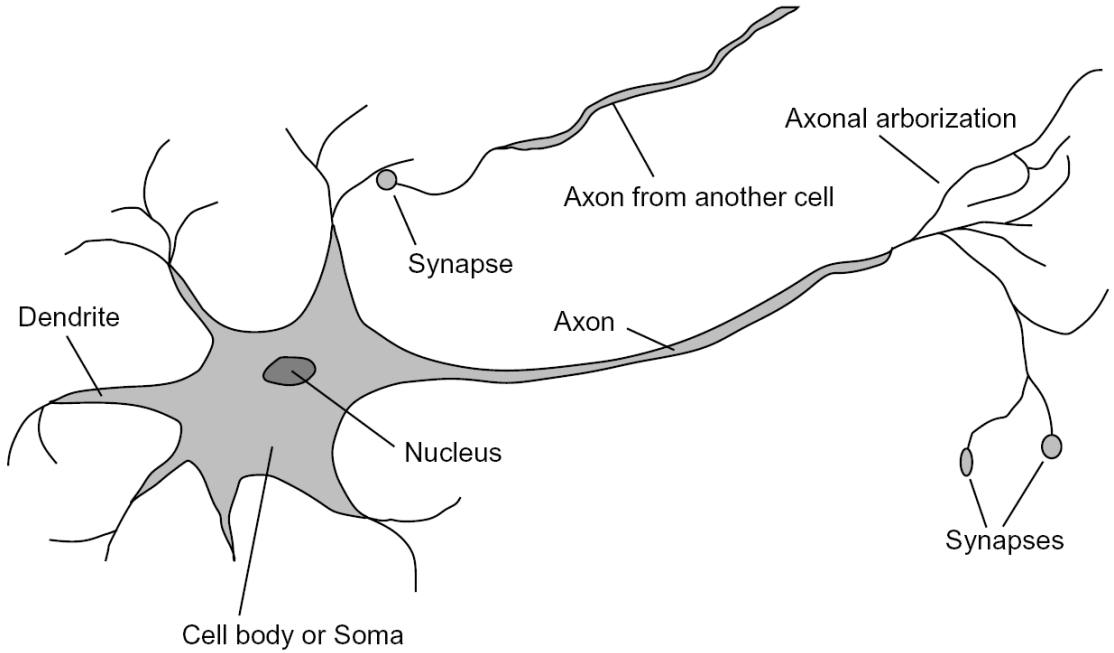
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

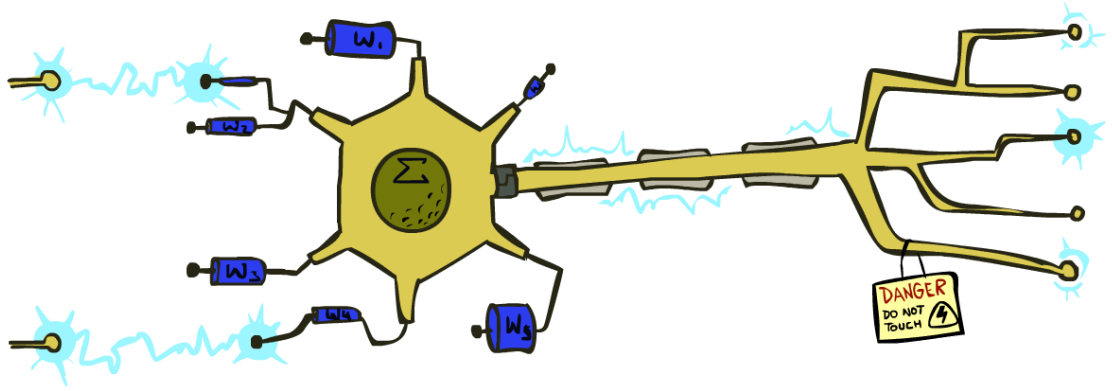


“2”

Some (Simplified) Biology

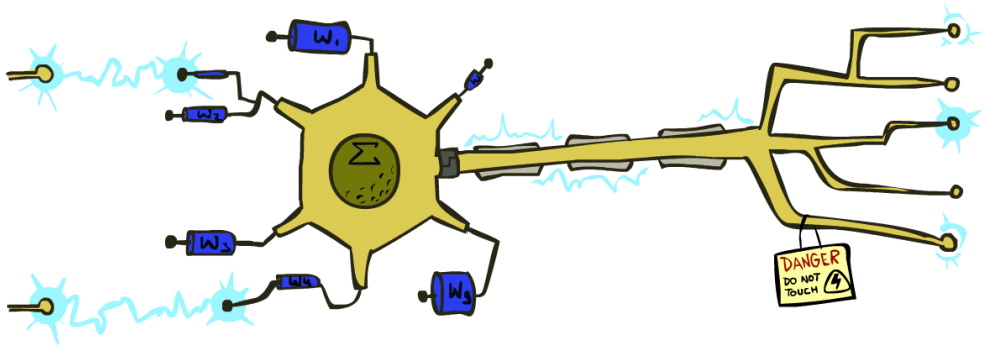
- Very loose inspiration: human neurons





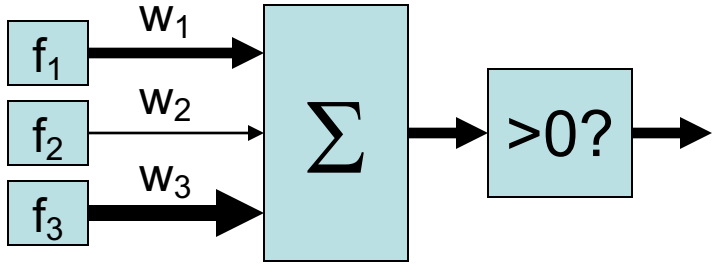
Linear Classifiers

- 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

Dot product $w \cdot f$ positive means the positive class (spam)

w · $f(x_1)$

w		$f(x_1)$
# free : 4		# free : 2
YOUR_NAME :-1		YOUR_NAME : 0
MISSPELLED : 1		MISSPELLED : 2
FROM_FRIEND :-3		FROM_FRIEND : 0
...		...

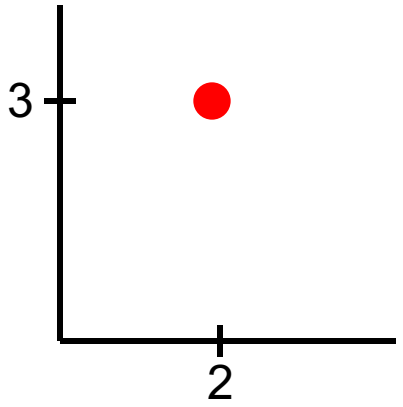
w · $f(x_2)$

w		$f(x_2)$
# free : 4		# free : 0
YOUR_NAME :-1		YOUR_NAME : 1
MISSPELLED : 1		MISSPELLED : 1
FROM_FRIEND :-3		FROM_FRIEND : 1
...		...

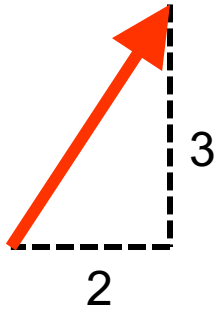
Do these weights make sense for spam classification?

Review: Vectors

- A tuple like (2,3) can be interpreted two different ways:



A **point** on a coordinate grid

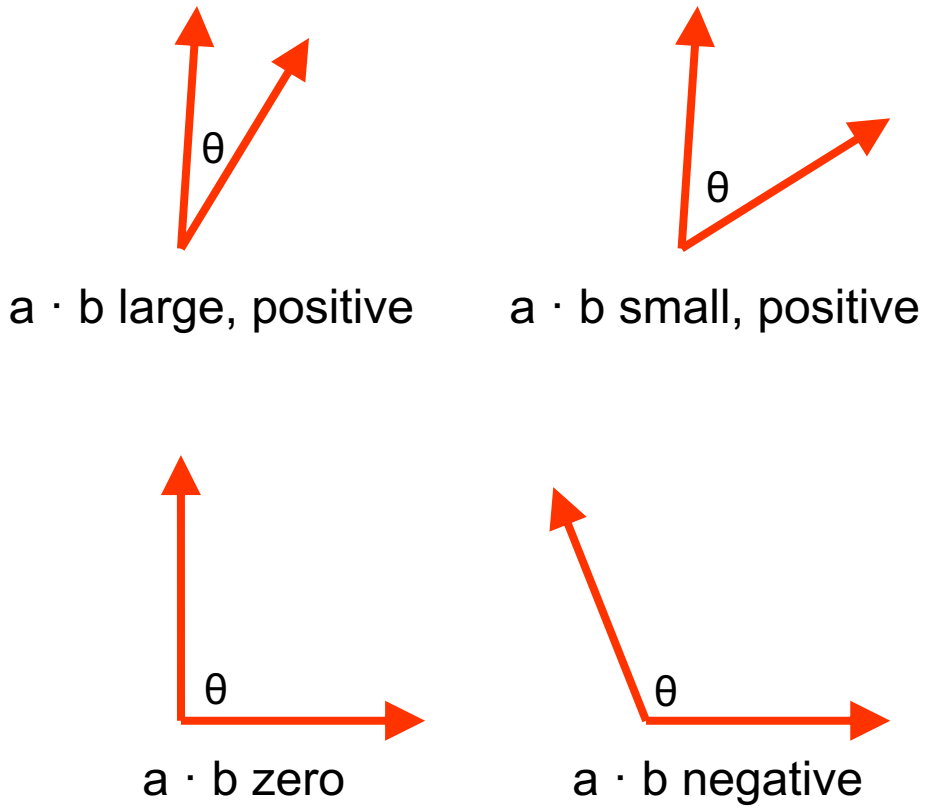


A **vector** in space. Notice we are not on a coordinate grid.

- A tuple with more elements like (2, 7, -3, 6) is a point or vector in higher-dimensional space (hard to visualize)

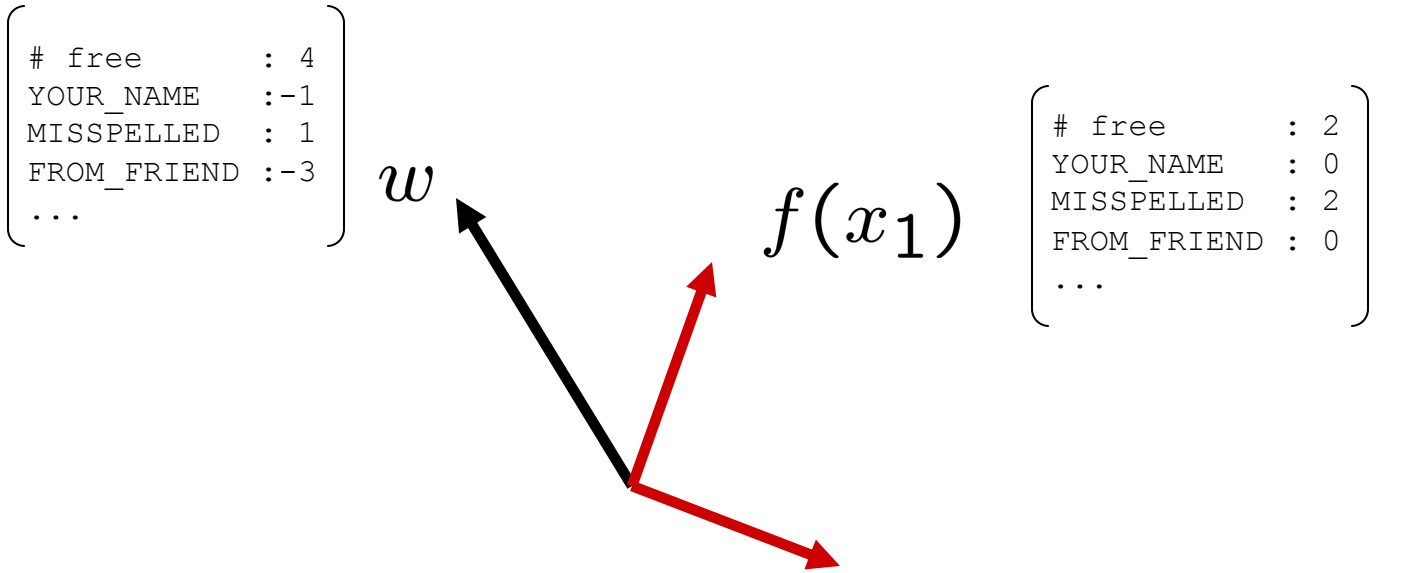
Review: Vectors

- Definition of dot product:
  - $a \cdot b = |a|\ |b| \cos(\theta)$
  - θ is the angle between the vectors a and b
- Consequences of this definition:
  - Vectors closer together
    = "similar" vectors
    = smaller angle θ between vectors
    = larger (more positive) dot product
  - If $\theta < 90°$, then dot product is positive
  - If $\theta = 90°$, then dot product is zero
  - If $\theta > 90°$, then dot product is negative



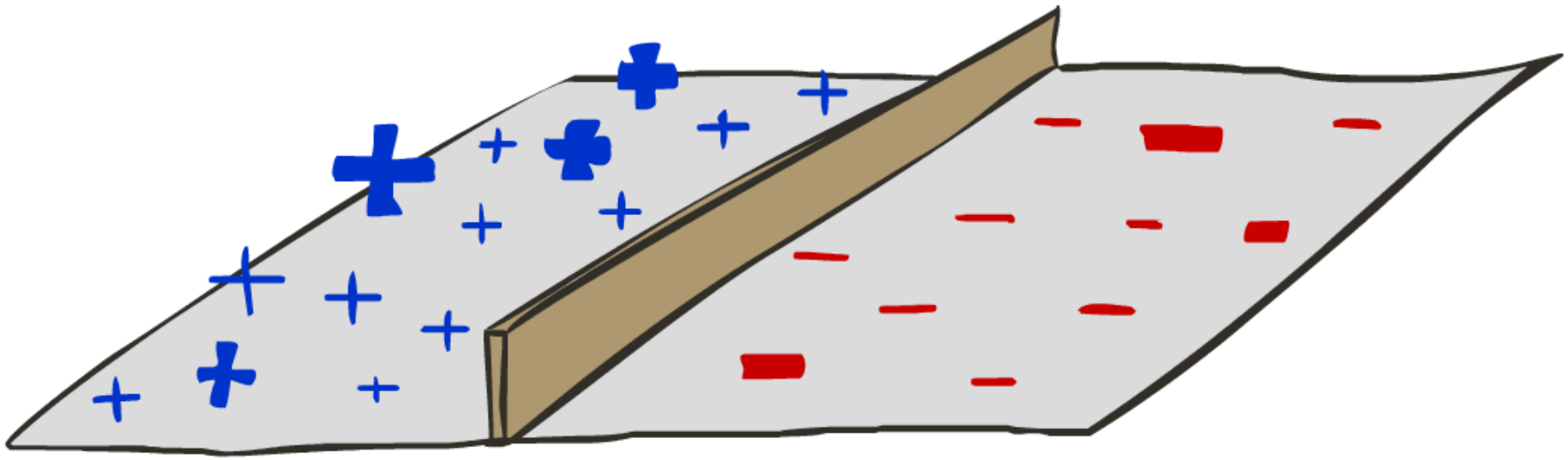
Weights

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

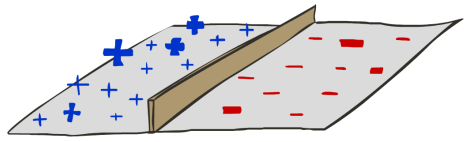


Dot product $w \cdot f$ positive means the positive class

Decision Rules



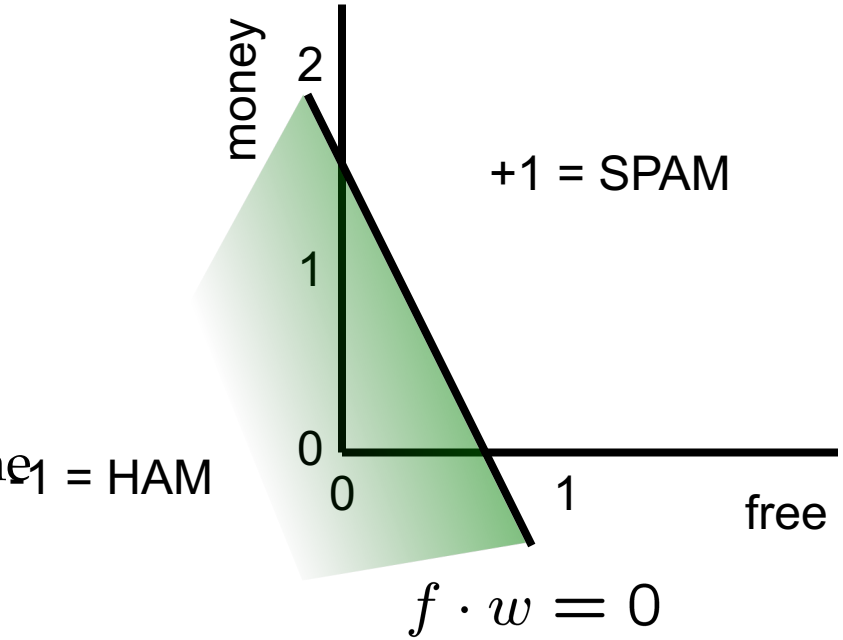
Binary Decision Rule



- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane (divides space into two sides)
  - One side corresponds to Y=+1, the other corresponds to Y=-1
- In the example:
  - $f \cdot w > 0$ when 4*free + 2*money > 0
    $f \cdot w < 0$ when 4*free + 2*money < 0
    These equations correspond to two halves of the feature space
  - $f \cdot w = 0$ when 4*free + 2*money = 0
    This equation corresponds to the decision boundary (a line in 2D, a hyperplane in higher dimensions)

w

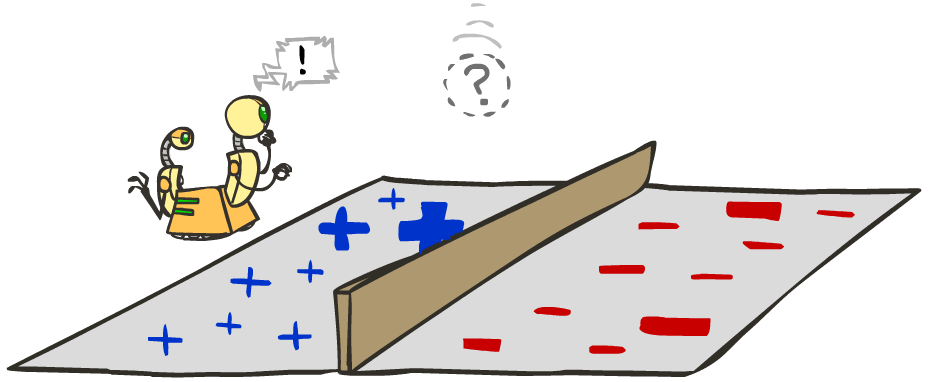
free	:	4
money	:	2



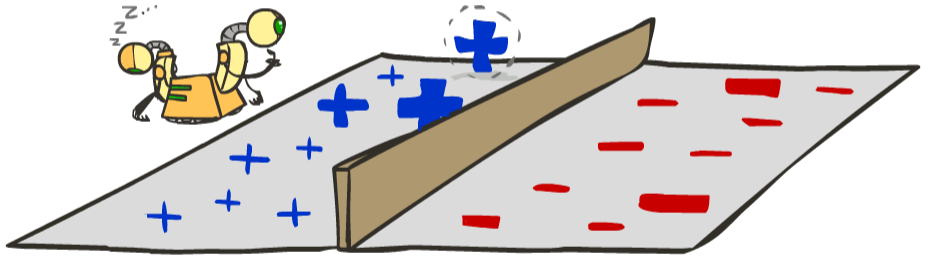
Weight Updates

Learning: Binary Perceptron

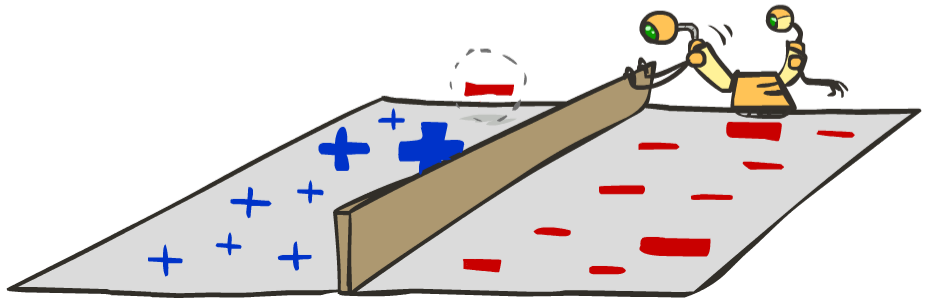
- Start with weights = 0
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



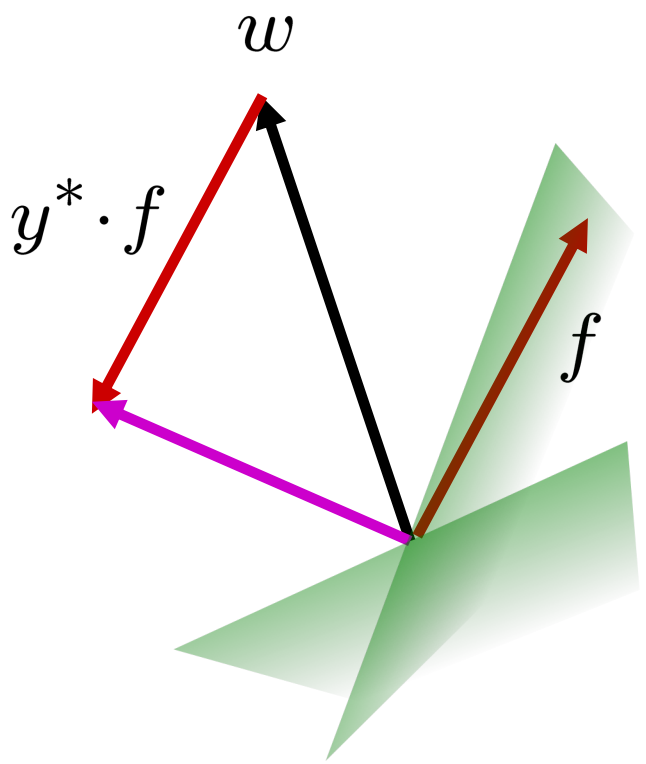
Learning: Binary Perceptron

- 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.

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



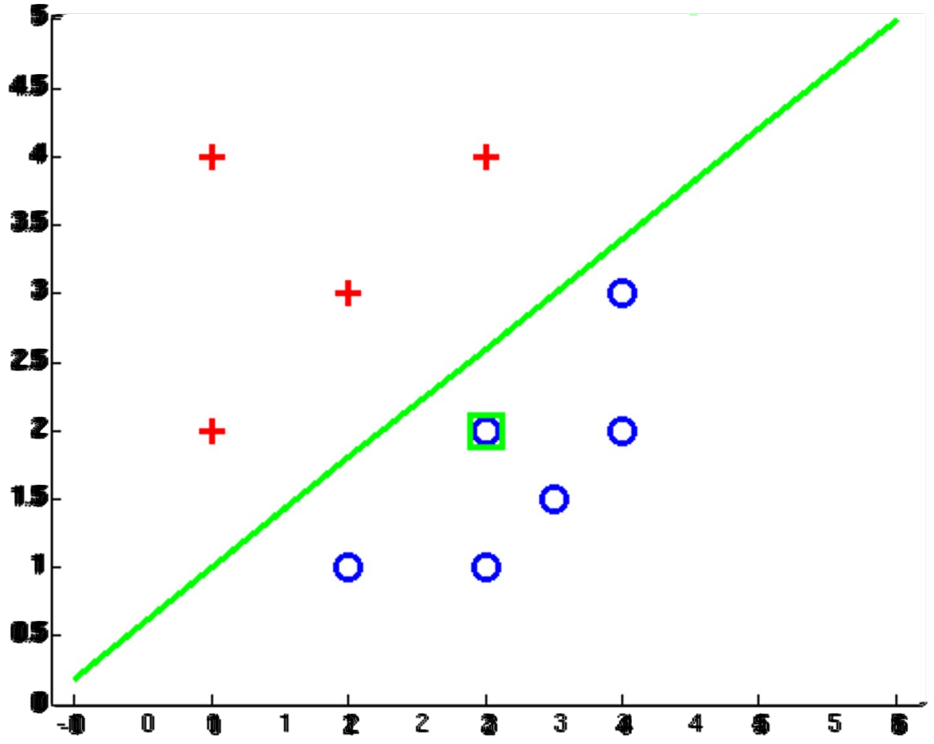
Before: $w f$
After: $wf + y^* f f$
$f f >= 0$

Learning: Binary Perceptron

- Misclassification, Case I:
  - $w \cdot f > 0$, so we predict +1
  - True class is -1
  - We want to modify w to w' such that dot product $w' \cdot f$ is *lower*
  - **Update if we misclassify a true class -1 sample: $w' = w - f$**
  - Proof: $w' \cdot f = (w - f) \cdot f = (w \cdot f) - (f \cdot f) = (w \cdot f) - |f|^2$
    Note that $|f|^2$ is always positive
- Misclassification, Case II:
  - $w \cdot f < 0$, so we predict -1
  - True class is +1
  - We want to modify w to w' such that dot product $w' \cdot f$ is *higher*
  - **Update if we misclassify a true class +1 sample: $w' = w + f$**
  - Proof: $w' \cdot f = (w + f) \cdot f = (w \cdot f) + (f \cdot f) = (w \cdot f) + |f|^2$
    Note that $|f|^2$ is always positive
- Write update compactly as $w' = w + y^* \cdot f$, where y^* = true class

Examples: Perceptron

- Separable Case



Multiclass Decision Rule

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

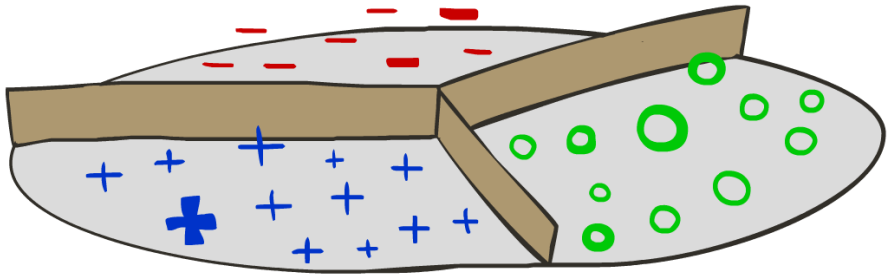
$$w_y$$

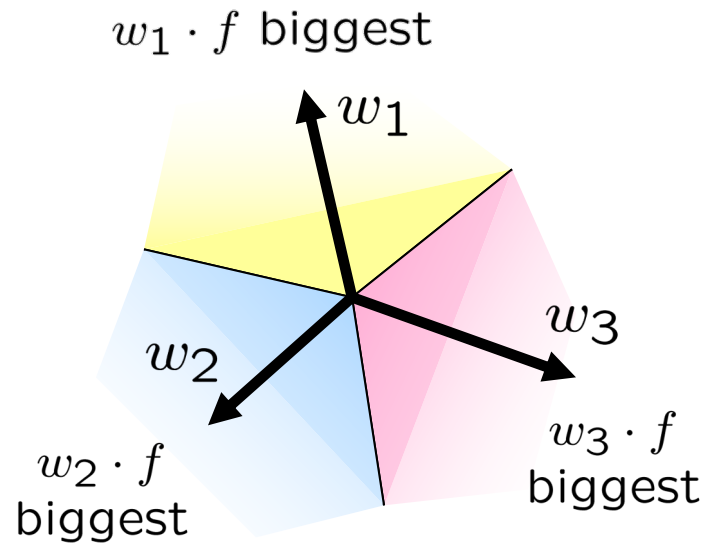
  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - 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

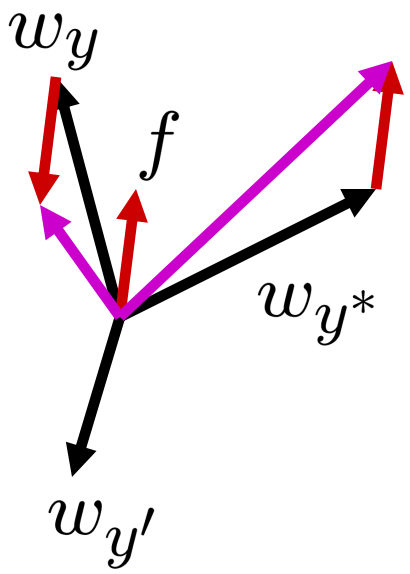
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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
...			

$w_{POLITICS}$

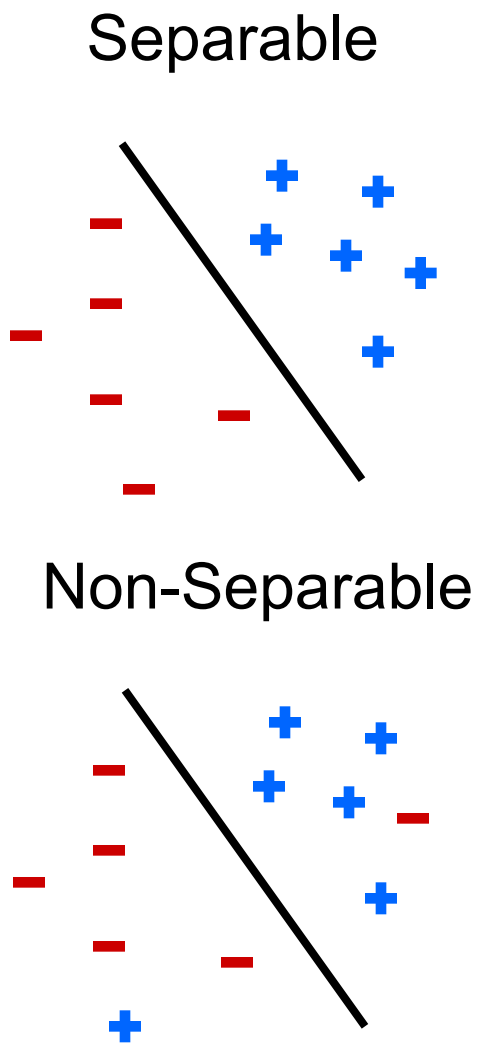
	0	3	3
BIAS : 0	1		0
win : 0	1		0
game : 0	0		-1
vote : 0	1		1
the : 0	1		0
...			

w_{TECH}

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

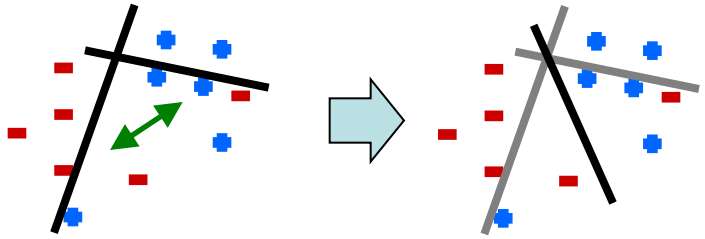
Properties of Perceptrons

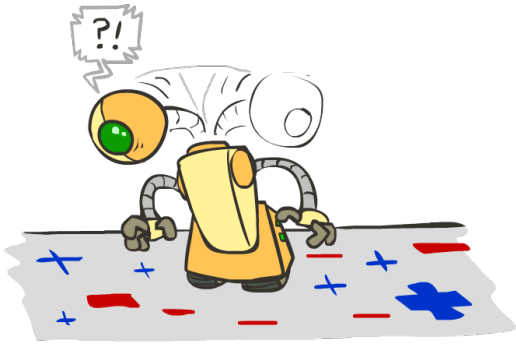
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake bound: the maximum number of mistakes (binary case) related to the *margin* or degree separability



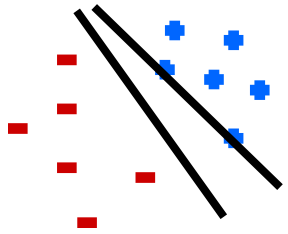
Problems with the Perceptron

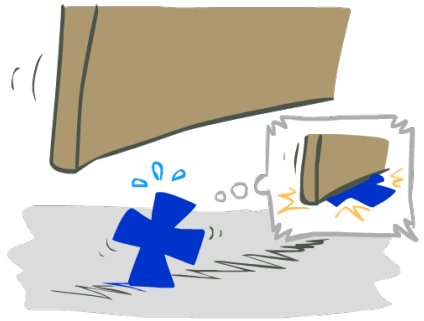
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



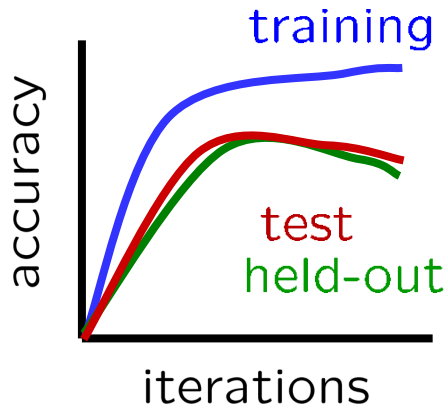


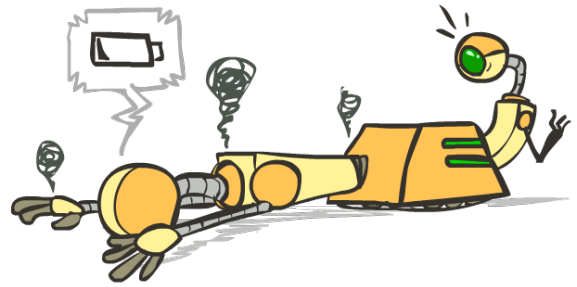
- Mediocre generalization: finds a "barely" separating solution



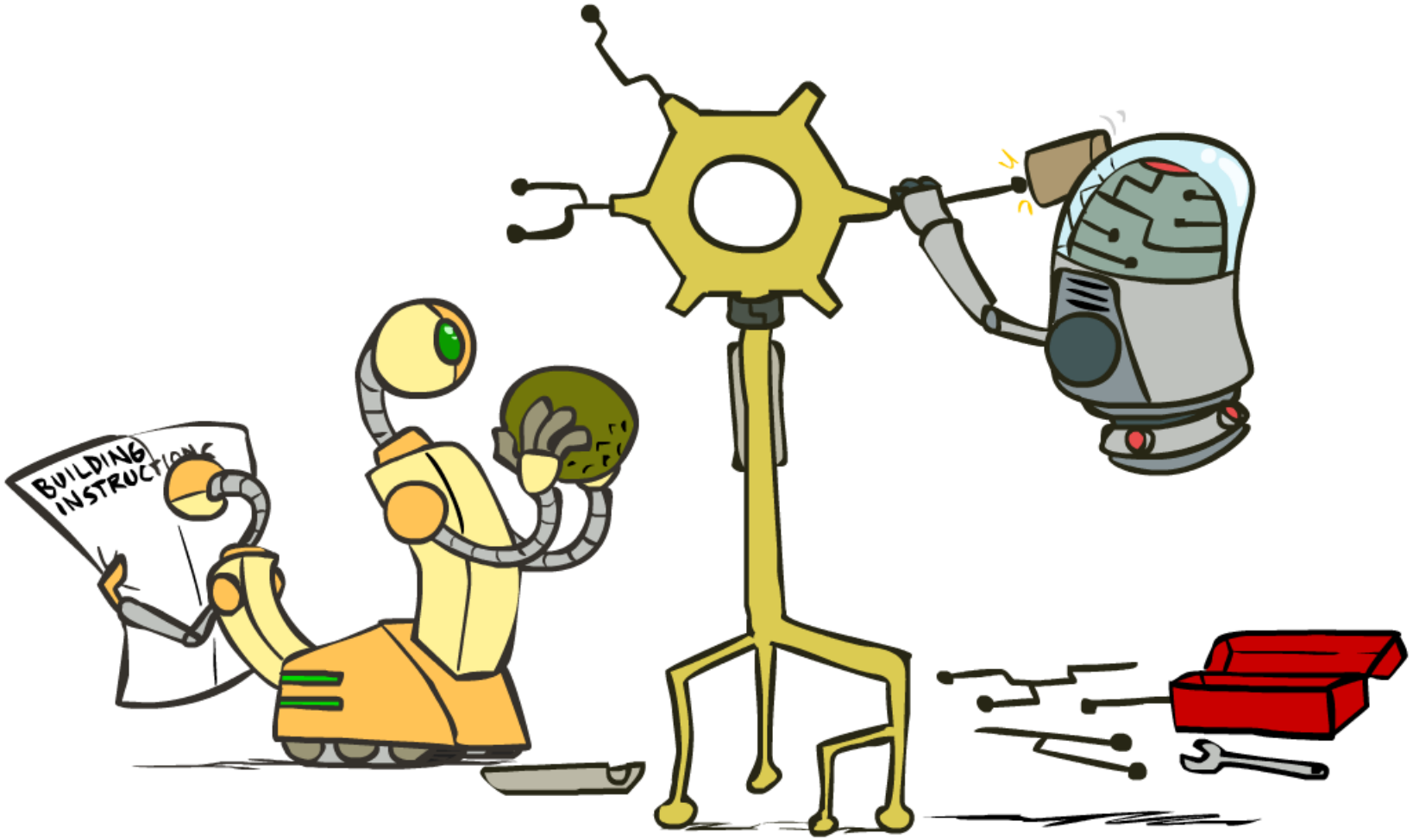


- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



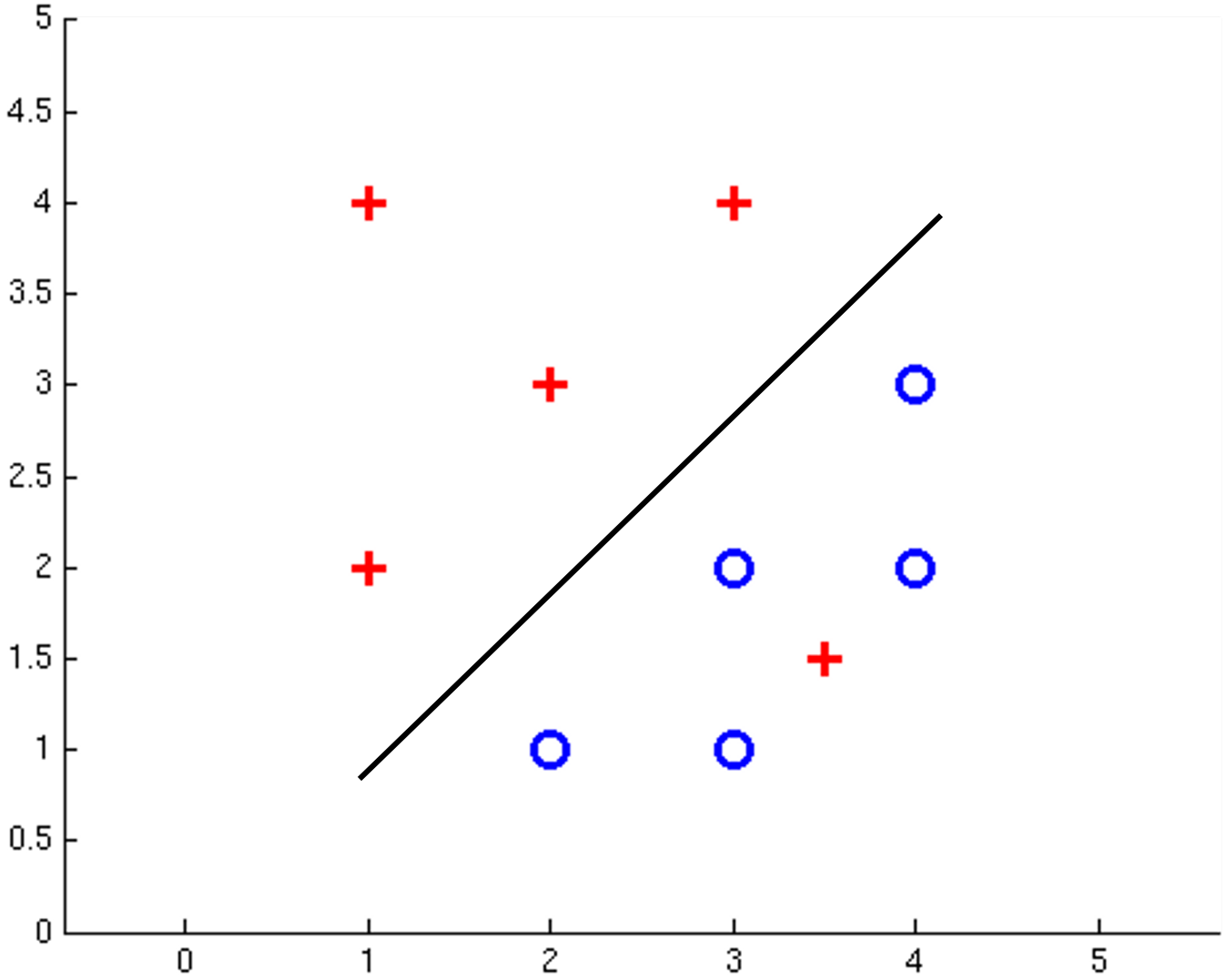


Improving the Perceptron

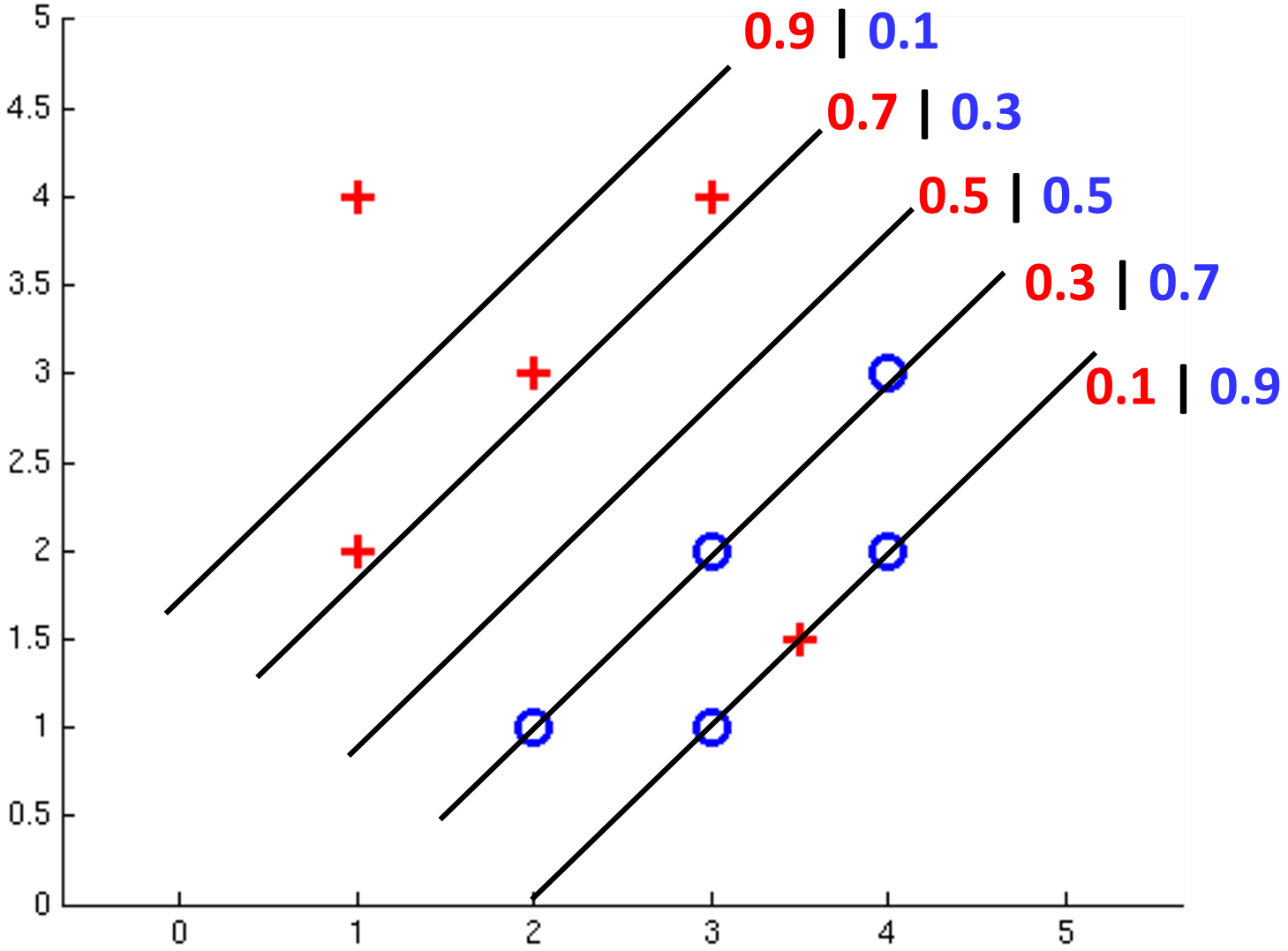


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



Non-Separable Case: Probabilistic Decision

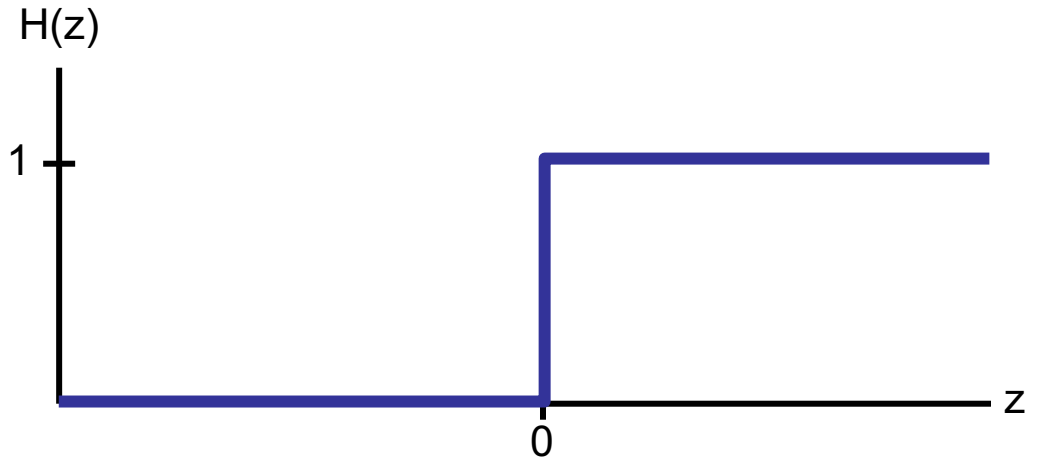


How to get deterministic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ positive → classifier says: 1.0 probability this is class +1
- If $z = w \cdot f(x)$ negative → classifier says: 0.0 probability this is class +1

- Step function

$$H(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$$



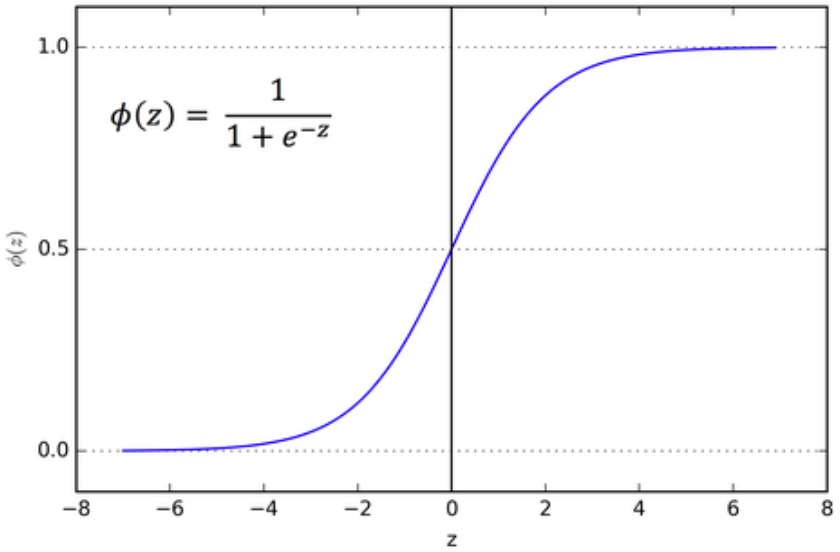
- z = output of perceptron
  H(z) = probability the class is +1, according to the classifier

How to get probabilistic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ very positive → probability of class +1 should approach 1.0
- If $z = w \cdot f(x)$ very negative → probability of class +1 should approach 0.0

- Sigmoid function

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



- z = output of perceptron
  $\phi(z)$ = probability the class is +1, accordin

Probabilistic Decisions: Example

$$\frac{1}{1 + e^{-wx}}$$

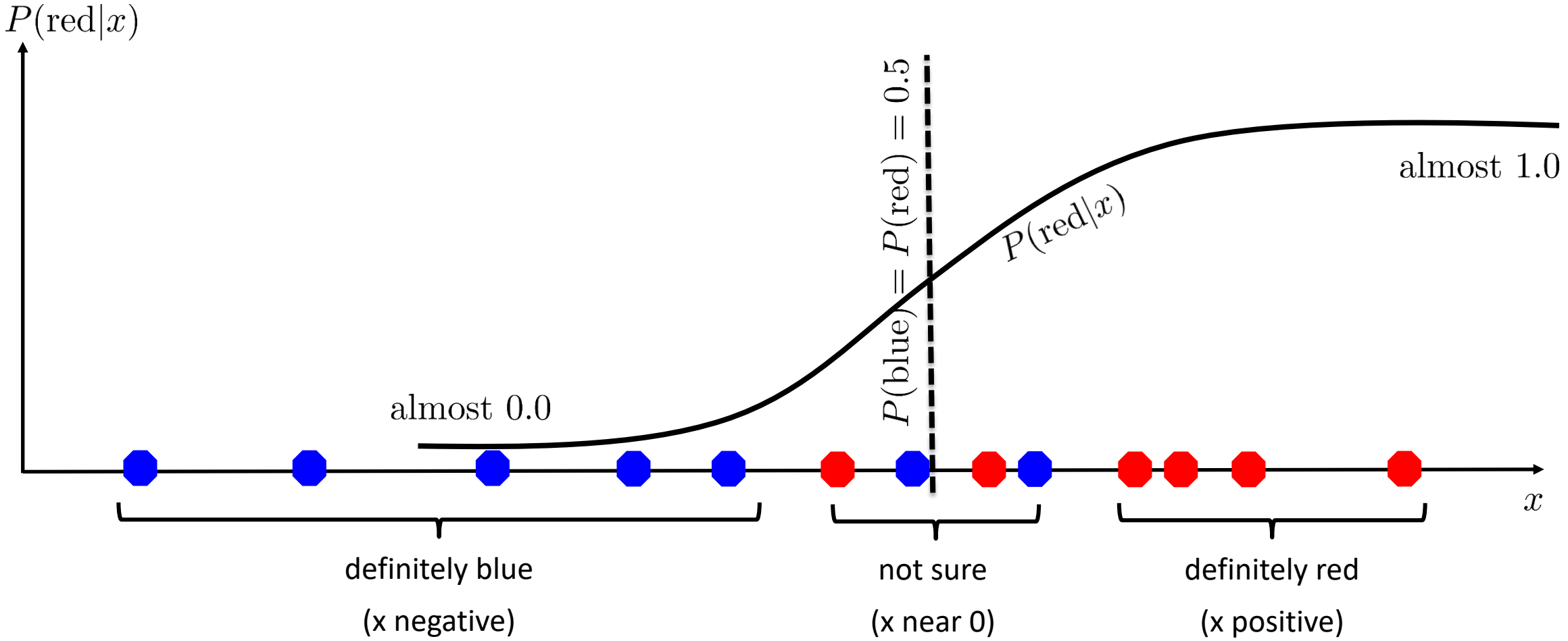
where w is some weight constant (vector) we have to learn, and wx is the dot product of w and x

- Suppose w = [–3, 4, 2] and x = [1, 2, 0]
- What label will be selected if we classify deterministically?
  - wx = –3+8+0 = 5
  - 5 is positive, so the classifier guesses the positive label
- What are the probabilities of each label if we classify probabilistically?
  - $1 / (1 + e^{-5}) = 0.9933$ probability of positive label
  - 1 – 0.9933 = 0.0067 probability of negative label

A 1D Example

$$P(\text{red}|x) = \frac{1}{1 + e^{-wx}}$$

where w is some weight constant (1D vector) we have to learn
(assume w is positive in this example)



Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)}; w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)}; w)
\end{aligned}$$

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = +1 \mid w) \\
= \; & P(y^{(i)} = +1 \mid x^{(i)}; w) \\
= \; & \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = -1 \mid w) \\
= \; & P(y^{(i)} = -1 \mid x^{(i)}; w) \\
= \; & 1 - \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

Best w?

- Maximum likelihood estimation:

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

with:

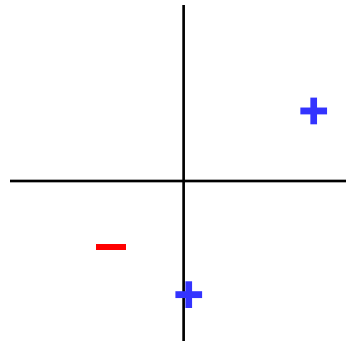
$$P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

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

= Logistic Regression

Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class +1
  - Data point [0, –2] is class +1
  - Data point [–1, –1] is class –1



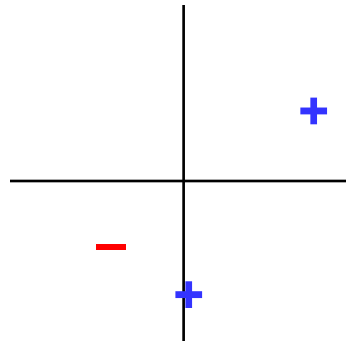
$$\max_w \; ll(w) = \max_w \; \sum_i \log P(y^{(i)}|x^{(i)};w)$$

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

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

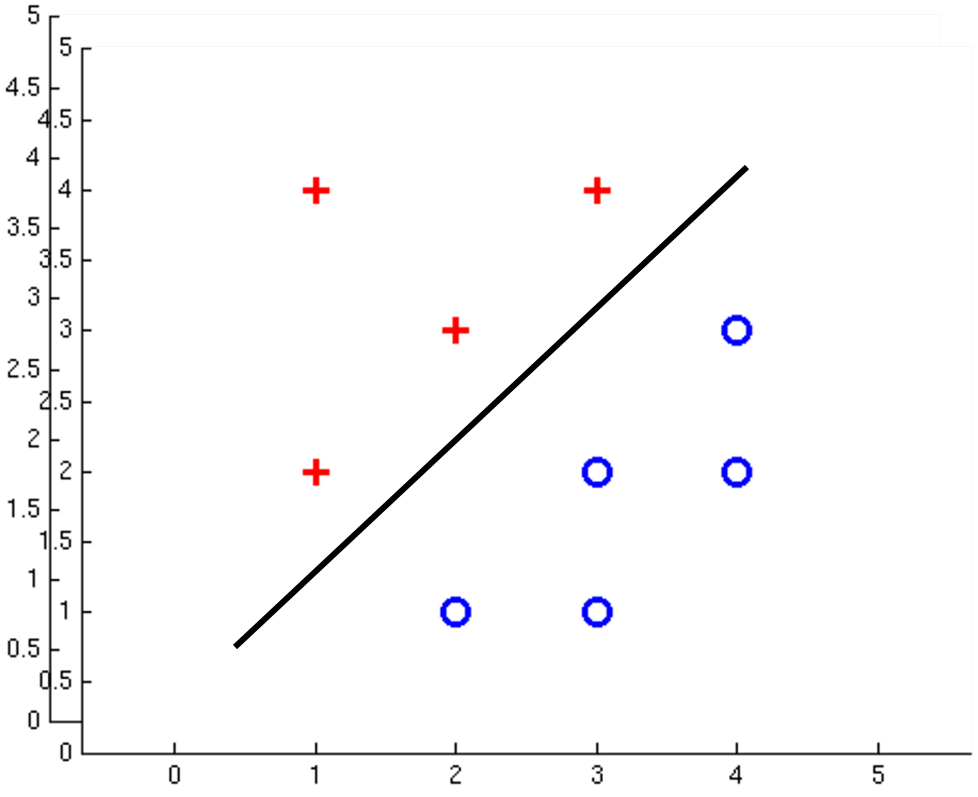
Logistic Regression Example

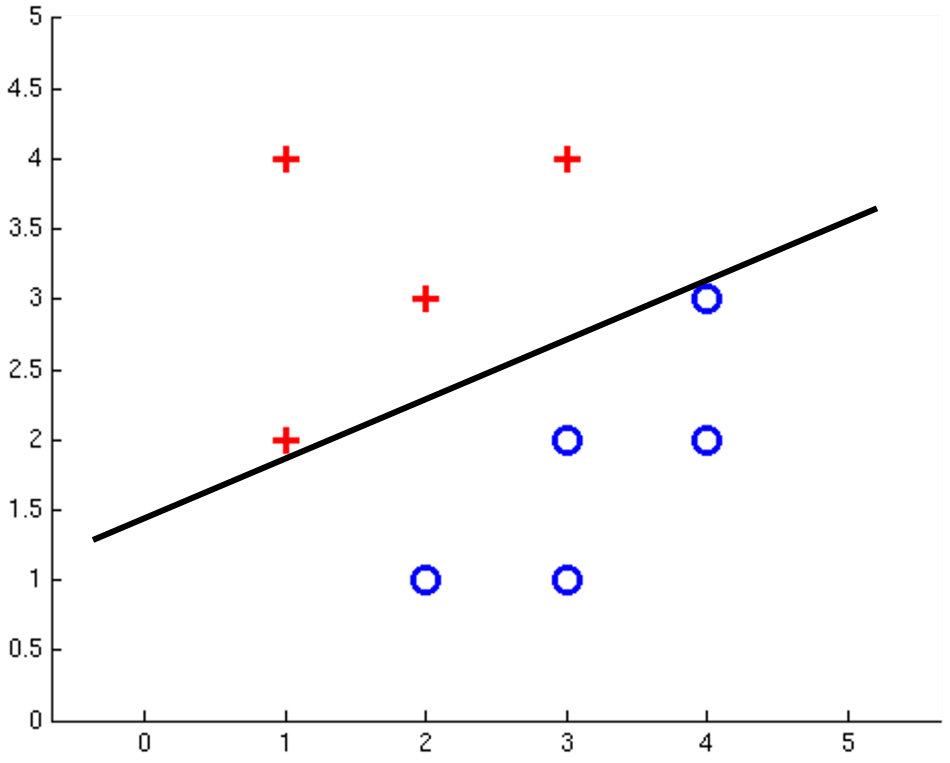
- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class +1
  - Data point [0, –2] is class +1
  - Data point [–1, –1] is class –1



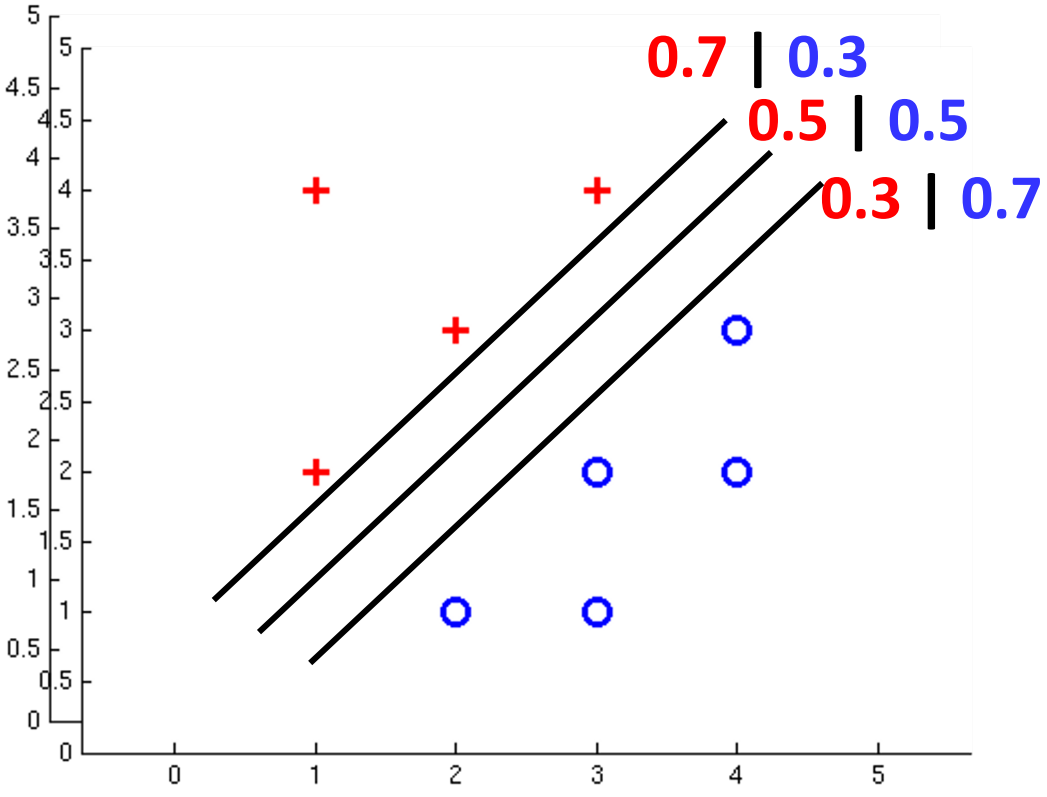
$$\underset{w}{\operatorname{argmax}} \left[\log \left(\frac{1}{1+e^{-(2w_1+w_2)}} \right) + \log \left(\frac{1}{1+e^{-(-2w_2)}} \right) + \log \left(1 - \frac{1}{1+e^{-(-w_1-w_2)}} \right) \right]$$

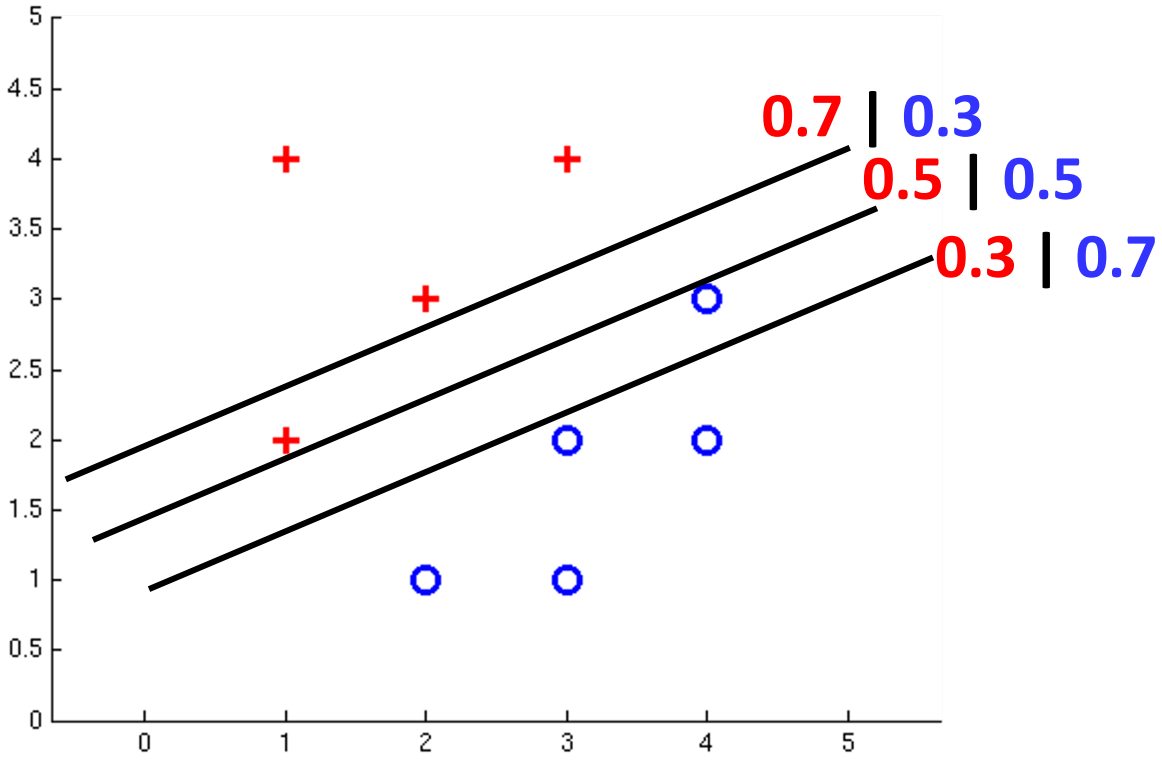
Separable Case: Deterministic Decision – Many Options





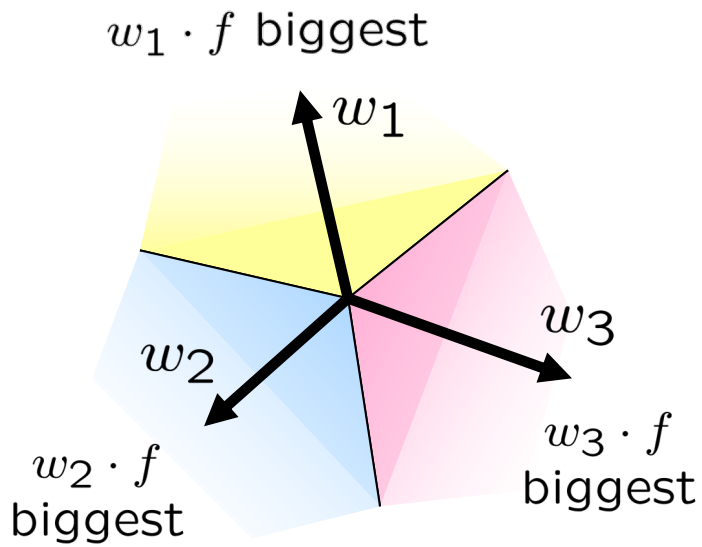
Separable Case: Probabilistic Decision – Clear Preference





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?

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

Multi-Class Probabilistic Decisions: Example

$$z_1, z_2, z_3 \rightarrow \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

- Suppose $w_1 = [-3, 4, 2]$, $w_2 = [2, 2, 7]$, $w_3 = [0, -1, 0]$, and $x = [1, 2, 0]$
- What label will be selected if we classify deterministically?
  - $w_1 \cdot x = 5$, and $w_2 \cdot x = 6$, and $w_3 \cdot x = -2$
  - $w_2 \cdot x$ has the highest score, so the classifier guesses class 2
- What are the probabilities of each label if we classify probabilistically?
  - Probability of class 1: $e^5 / (e^5 + e^6 + e^{-2}) = 0.2689$
  - Probability of class 2: $e^6 / (e^5 + e^6 + e^{-2}) = 0.7310$
  - Probability of class 3: $e^{-2} / (e^5 + e^6 + e^{-2}) = 0.0002$

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)}; w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)}; w)
\end{aligned}$$

Best w?

- Maximum likelihood estimation:

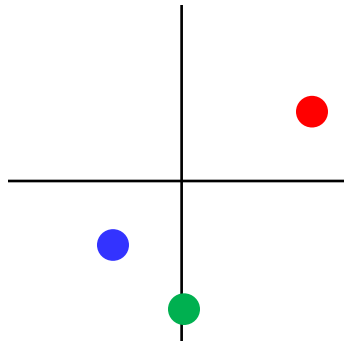
$$\max_w \; ll(w) = \max_w \; \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

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

= Multi-Class Logistic Regression

Multi-Class Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class Red
  - Data point [0, –2] is class Green
  - Data point [–1, –1] is class Blue

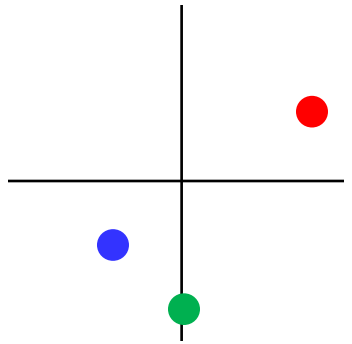


$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

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

Multi-Class Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class Red
  - Data point [0, –2] is class Green
  - Data point [–1, –1] is class Blue



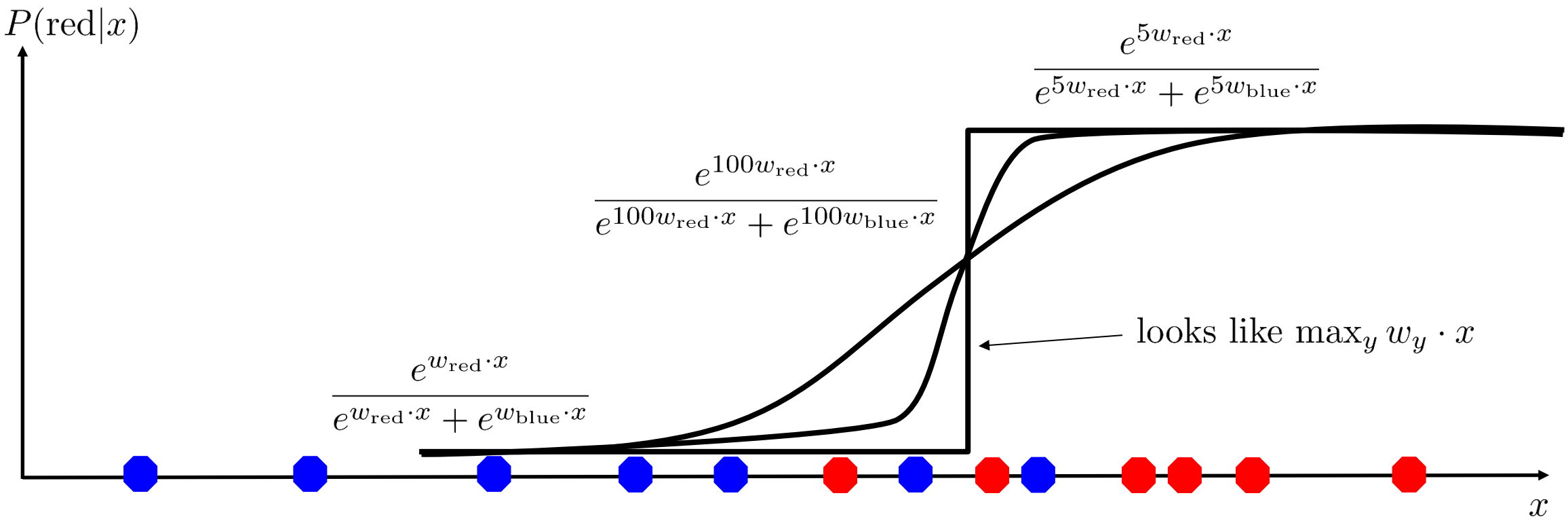
$$\underset{w}{\operatorname{argmax}} \left[\begin{array}{l} \log\left(\frac{e^{2w_1+w_2}}{e^{2w_1+w_2}+e^{2w_1+w_2}+e^{2w_1+w_2}}\right) \\ +\log\left(\frac{e^{-2w_2}}{e^{-2w_2}+e^{-2w_2}+e^{-2w_2}}\right) \\ +\log\left(\frac{e^{-w_1-w_2}}{e^{-w_1-w_2}+e^{-w_1-w_2}+e^{-w_1-w_2}}\right) \end{array} \right]$$

Log probability of [2, 1] being red

Log probability of [0,–2] being green

Log probability of [–1, –1] being blue

Softmax with Different Bases



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

Softmax and Sigmoid

- Recall: Binary perceptron is a special case of multi-class perceptron
  - Multi-class: Comput$w_y \cdot f(x)$ for each class y, pick class with the highest activation
  - Binary case:
    Let the weight vector of +1 be w (which we learn).
    Let the weight vector of -1 always be 0 (constant).
  - Binary classification as a multi-class problem:
    Activation of negative class is always 0.
    If w · f is positive, then activation of +1 (w · f) is higher than -1 (0).
    If w · f is negative, then activation of -1 (0) is higher than +1 (w · f).

Softmax

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

with $w_{\text{red}} = 0$ becomes:

Sigmoid

$$P(\text{red}|x) = \frac{1}{1 + e^{-wx}}$$

Naïve Bayes vs Logistic Regression

	Naïve Bayes	Logistic Regression
Model	Joint over all features and label: $P(Y, F_1, F_2, \dots)$	Conditional: $P(y \mid f_1, f_2, \dots; w)$
Predicted class probabilities	Inference in a Bayes Net: $P(Y \mid f) \propto P(Y)\, P(f_1 \mid Y) \dots$	Directly output label: $P(y = +1 \mid f; w) = 1/(1 + e^{-w \cdot f})$
Features	Discrete	Discrete or Continuous
Parameters	Entries of probability tables $P(Y)$ and $P(F_k \mid Y)$	Weight vector w
Learning	Counting occurrences of events	Iterative numerical optimization

Next lecture

Optimization

i.e., how do we solve:

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