Announcements

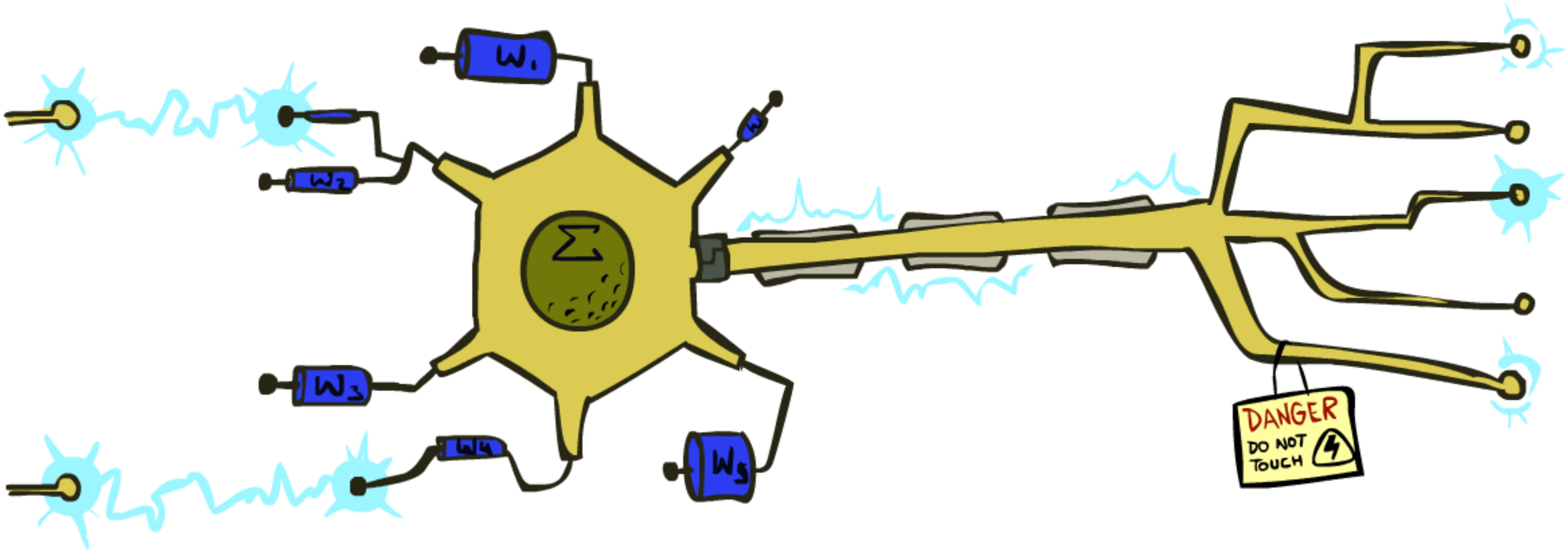
- HW7: due **Today, Apr 2, 11:59 PM PT**
- Releasing later this week:
  - HW8: due **Tuesday, April 9, 11:59 PM PT**
  - Project 5: due **Tuesday, April 16, 11:59 PM PT**
- Releasing next week:
  - HW9: due **Tuesday, April 16, 11:59 PM PT**
- TA 1-1s: see announcement on Ed

Same day!

Pre-scan attendance QR code now!

CS 188: Artificial Intelligence

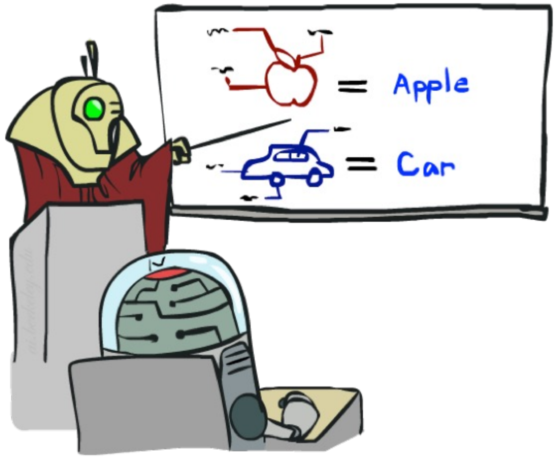
Perceptrons

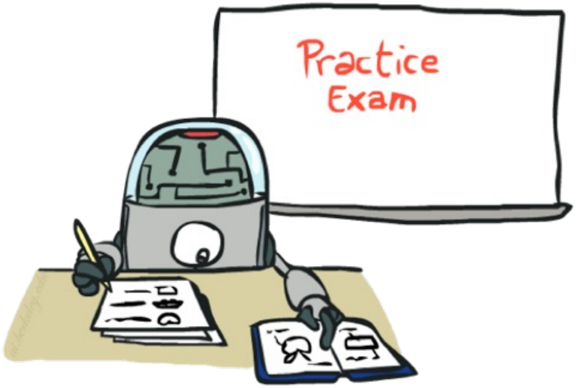


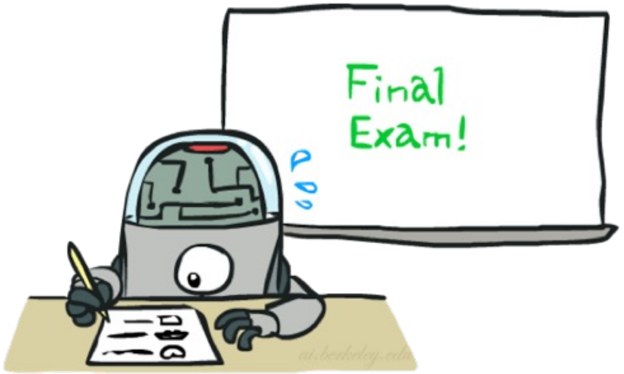
Spring 2024

Recap: Training and Machine Learning

- Big idea: ML algorithms learn patterns between features and labels from *data*
  - You don't have to reason about the data yourself
  - You're given **training data**: lots of example datapoints and their actual labels



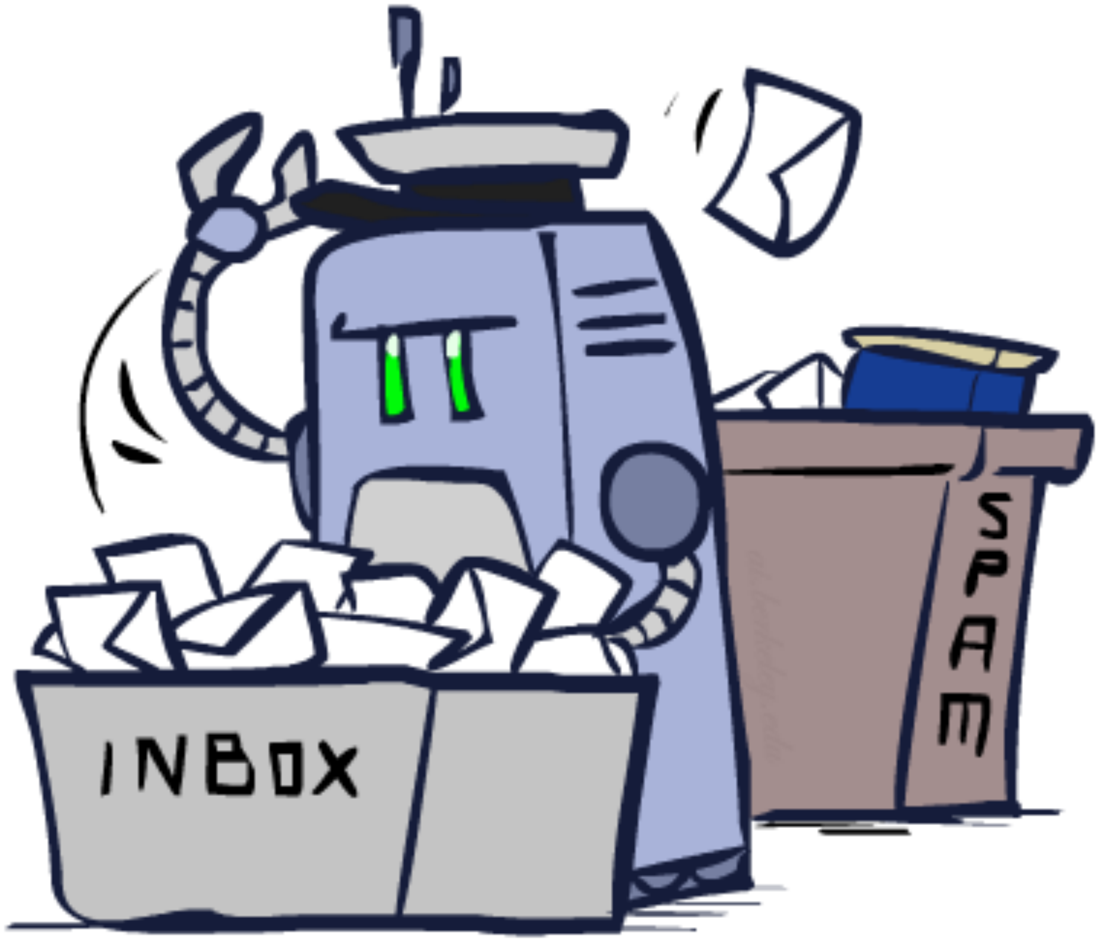




Training: Learn patterns from labeled data, and periodically test how well you're doing

Eventually, use your algorithm to predict labels for unlabeled data

Classification: Ham vs. Spam Emails



✗

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

✗

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES
FOR ONLY $99

✓

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Classification: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …
  - Features are increasingly induced rather than crafted

0

1

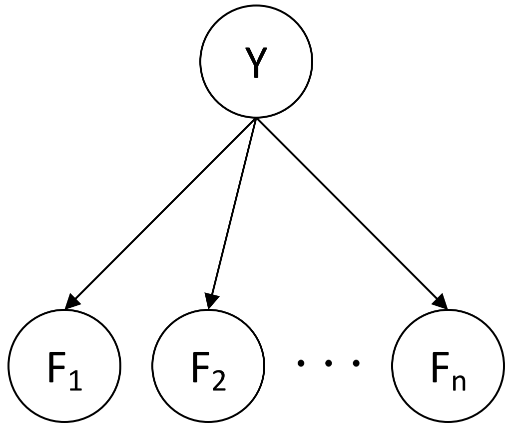
2

1

??

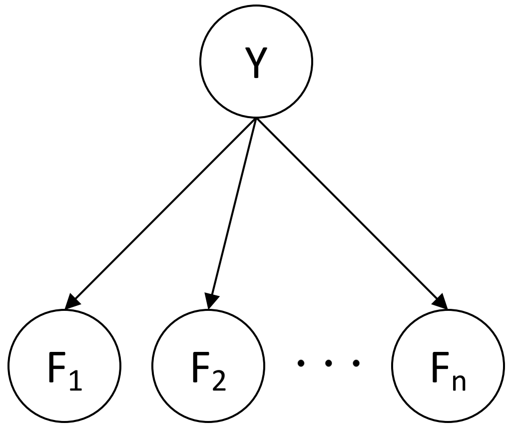
Recap: Naïve Bayes Model

- Random variables in this Bayes net:
  - Y = The label
  - $F_1, F_2, \ldots, F_n$ = The n features
- Probability tables in this Bayes net:
  - P(Y) = Probability of each label occurring, given no information about the features. Sometimes called the *prior*.
  - $P(F_i|Y)$ = One table per feature. Probability distribution over a feature, given the label.



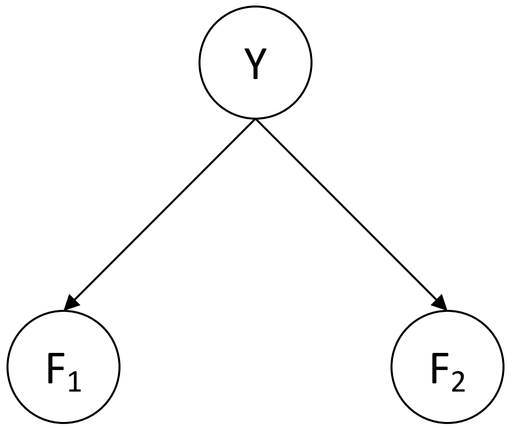
Recap: Naïve Bayes Model

- To perform training:
  - Use the training dataset to estimate the probability tables.
  - Estimate P(Y) = how often does each label occur?
  - Estimate $P(F_i|Y)$ = how does the label affect the feature?
- To perform classification:
  - Instantiate all features. You know the input features, so they're your evidence.
  - Query for $P(Y|f_1, f_2, \ldots, f_n)$. Probability of label, given all the input features. Use an inference algorithm (e.g. variable elimination) to compute this.



Recap: Naïve Bayes for Spam Filter

- Step 1: Select a ML algorithm. We choose to model the problem with Naïve Bayes.
- Step 2: Choose features to use.



Y: The label (spam or ham)	
Y	P(Y)
ham	?
spam	?

F_1: A feature (do I know the sender?)		
F_1	Y	$P(F_1 \mid Y)$
yes	ham	?
no	ham	?
yes	spam	?
no	spam	?

F_2: Another feature (# of occurrences of FREE)		
F_2	Y	$P(F_2 \mid Y)$
0	ham	?
1	ham	?
2	ham	?
0	spam	?
1	spam	?
2	spam	?

Example: Overfitting

- Posteriors determined by *relative* probabilities (odds ratios):

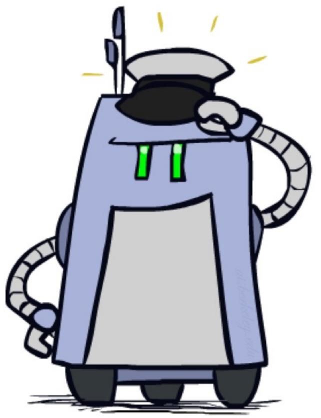
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

south-west	:	inf
nation	:	inf
morally	:	inf
nicely	:	inf
extent	:	inf
seriously	:	inf
...		

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

screens	:	inf
minute	:	inf
guaranteed	:	inf
$205.00	:	inf
delivery	:	inf
signature	:	inf
...		

What went wrong here?



Example: Overfitting

$P(\text{features}, C = 2)$

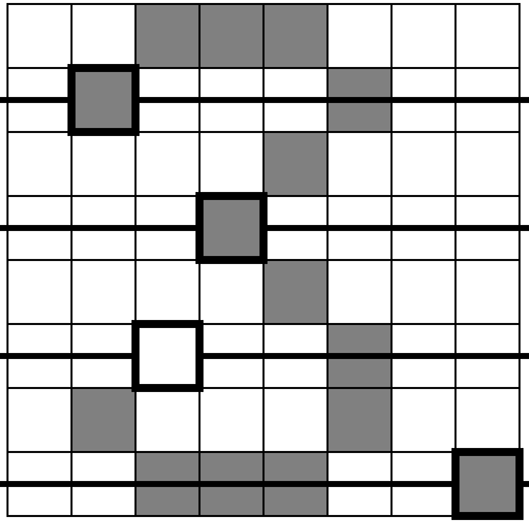
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

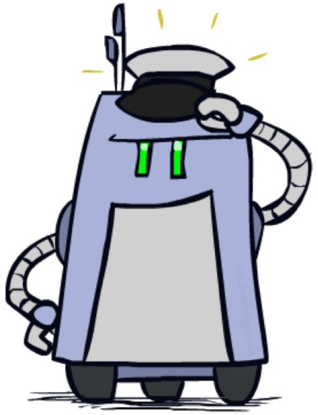
$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$



2 wins!!

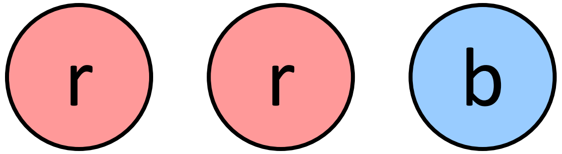
Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

$$= \frac{c(x) + 1}{N + |X|}$$

  - Can derive this estimate with *Dirichlet priors* (see cs281a)



$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

Laplace Smoothing

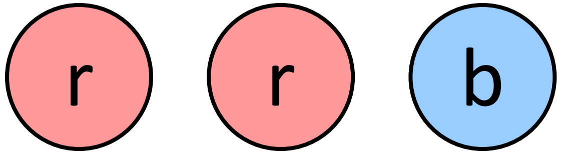
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$



$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Real Naïve Bayes: Smoothing

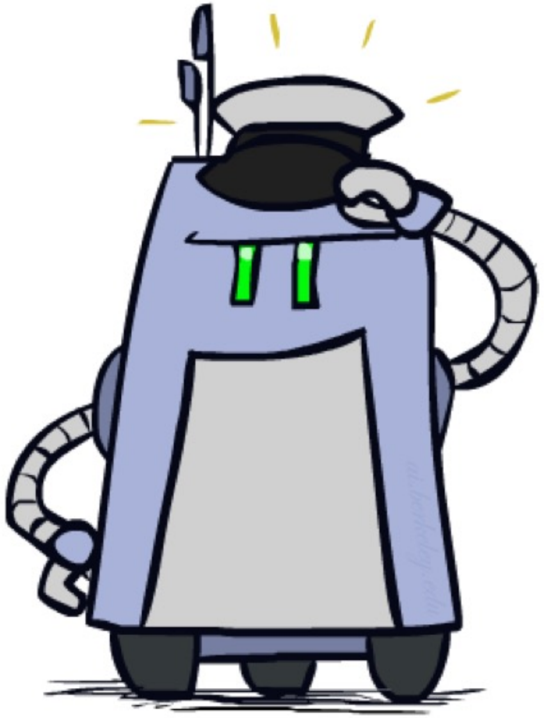
- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

helvetica	:	11.4
seems	:	10.8
group	:	10.2
ago	:	8.4
areas	:	8.3
...		

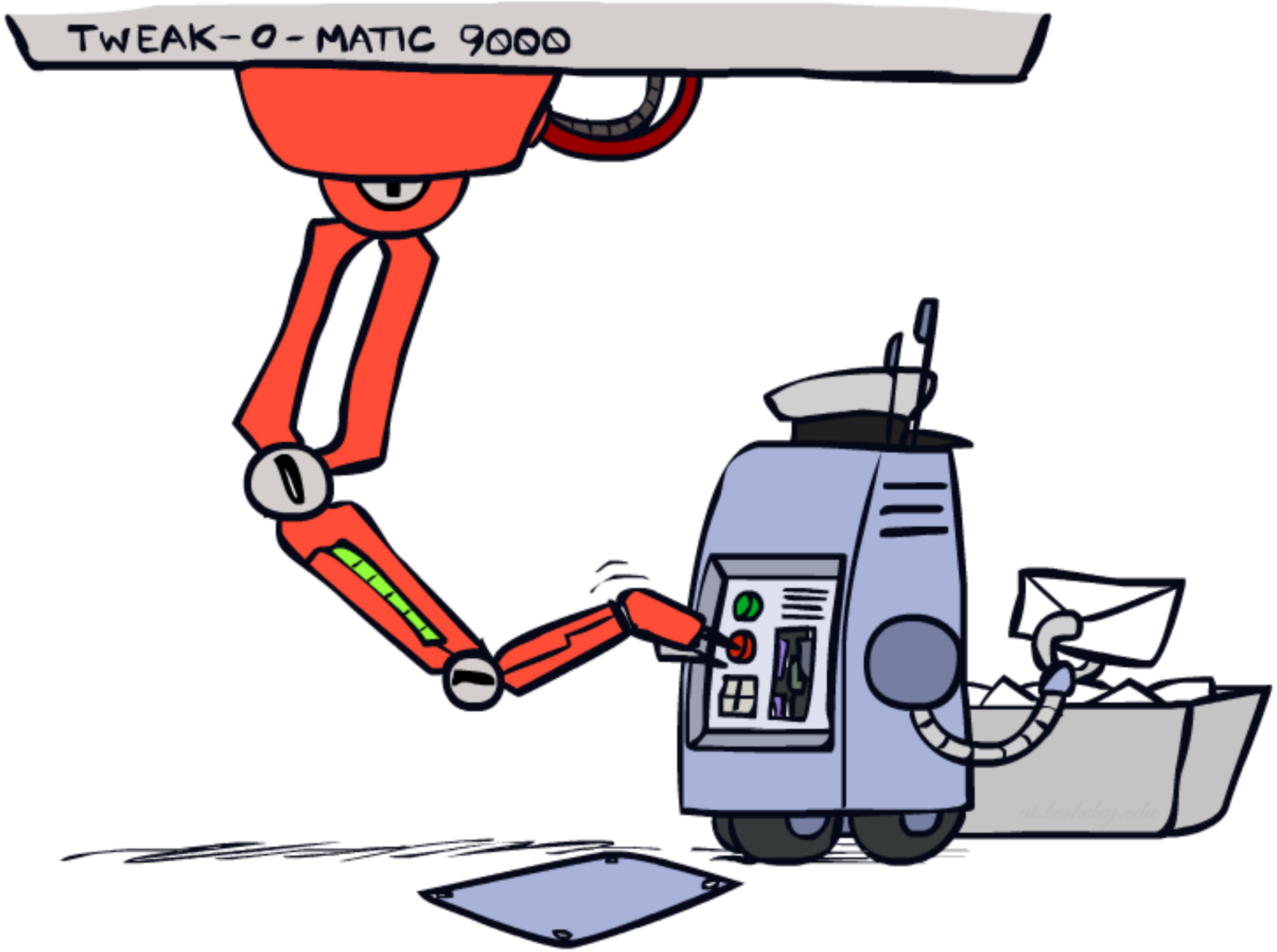
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
<FONT>	:	26.9
money	:	26.5
...		

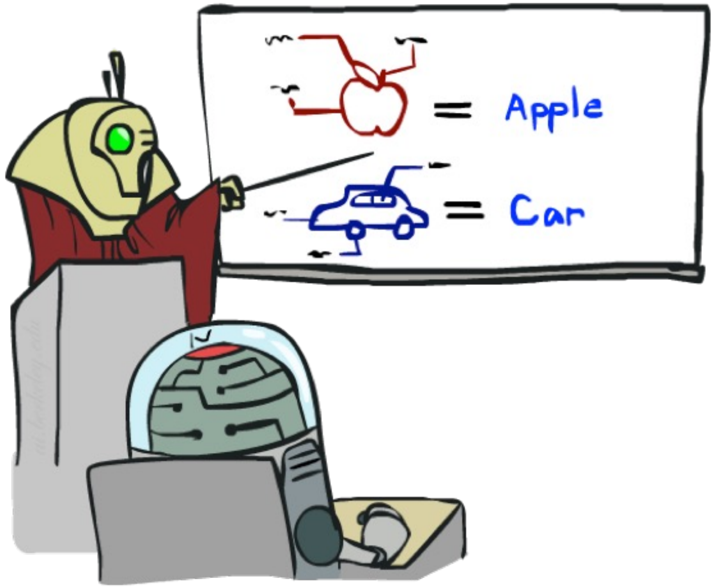


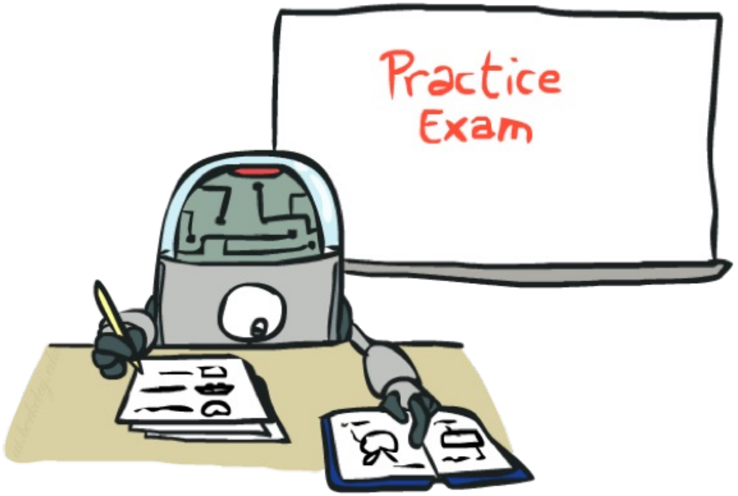
Do these make more sense?

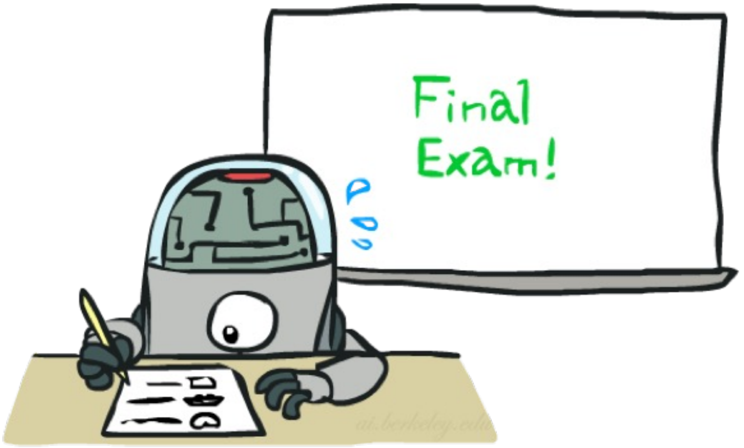
Tuning



Training and Testing

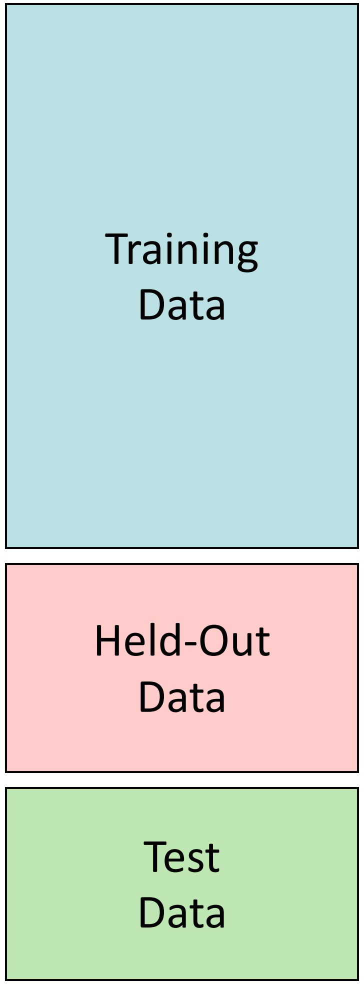


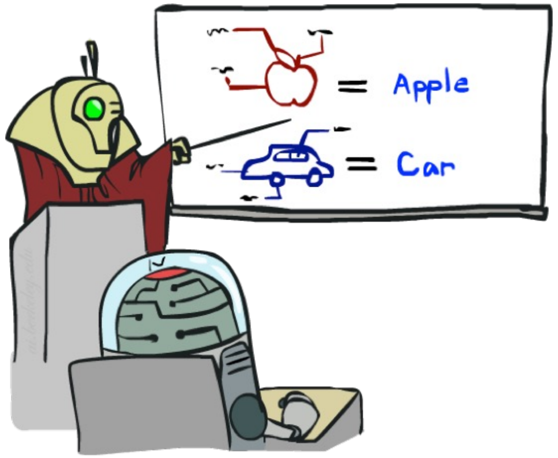


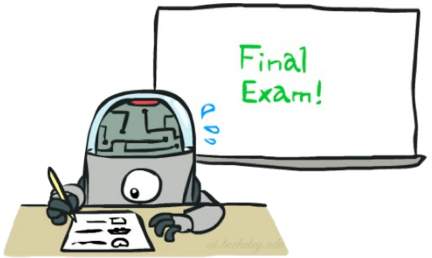


Important Concepts

- How do we check that we're not overfitting during training?
- Split training data into 3 different sets:
  - Training set
  - Held out set (more on this later)
  - Test set
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation (many metrics possible, e.g. accuracy)
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - We'll investigate overfitting and generalization formally in a few lectures

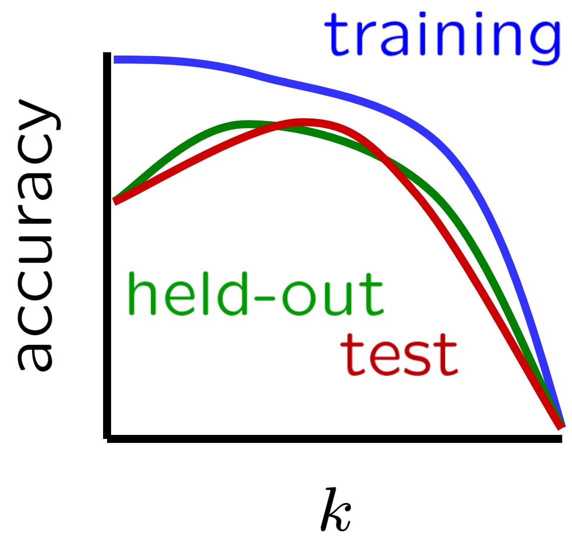


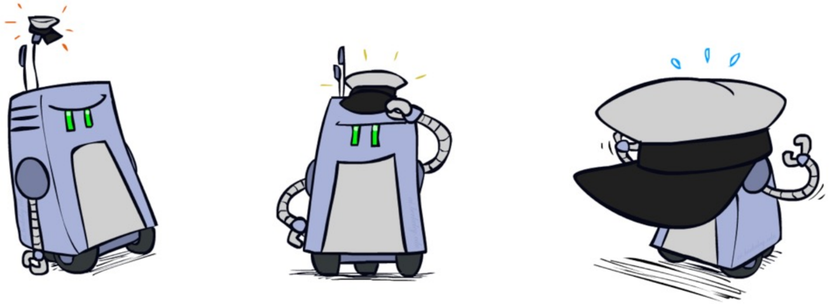




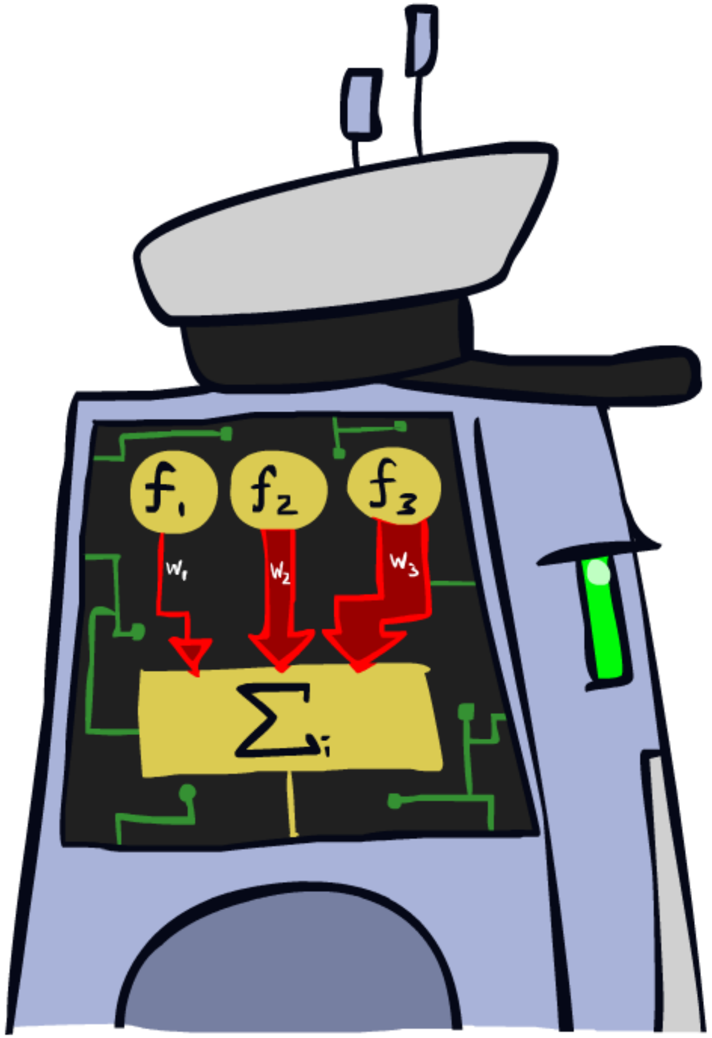
Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data





Linear Classifiers

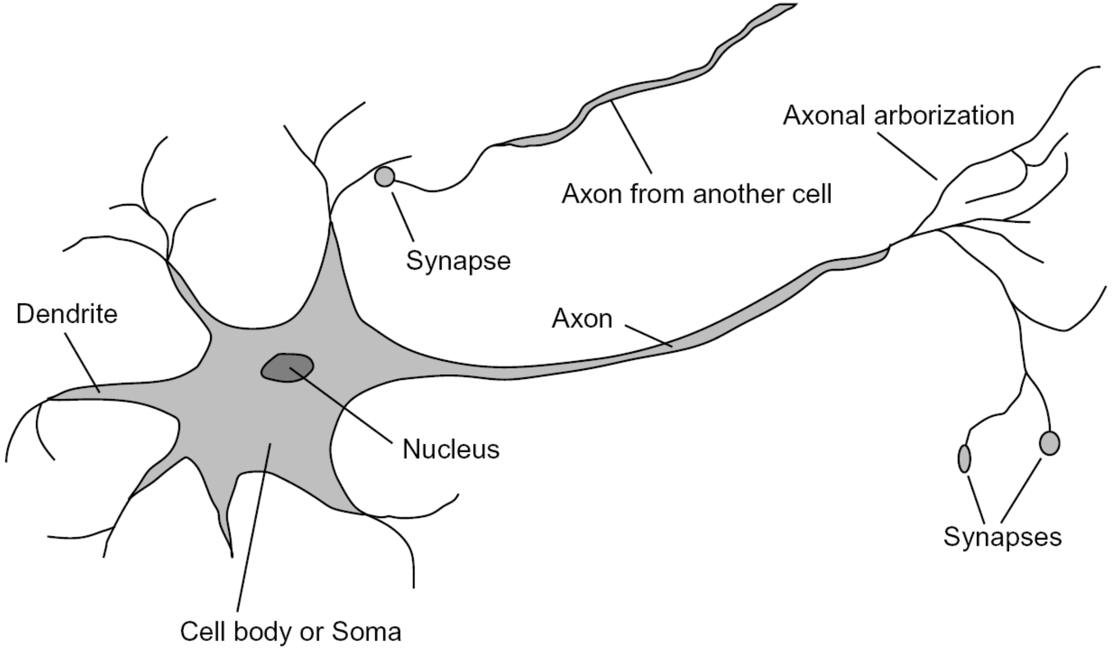


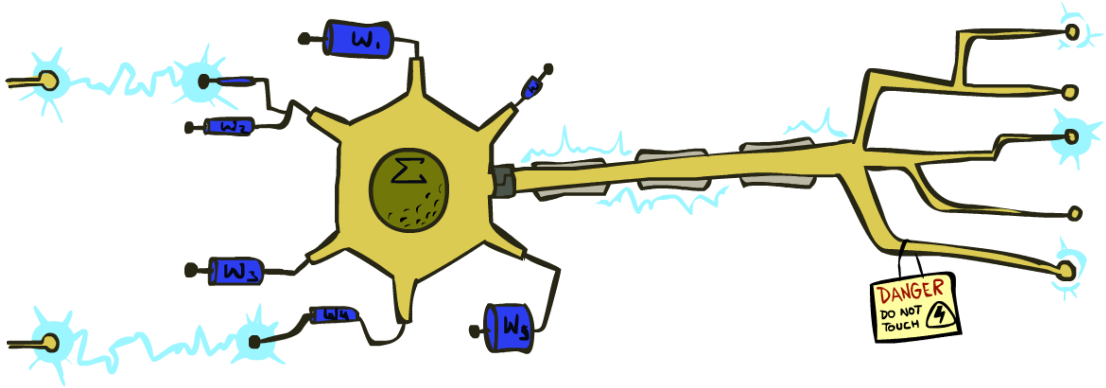
Feature Vectors

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

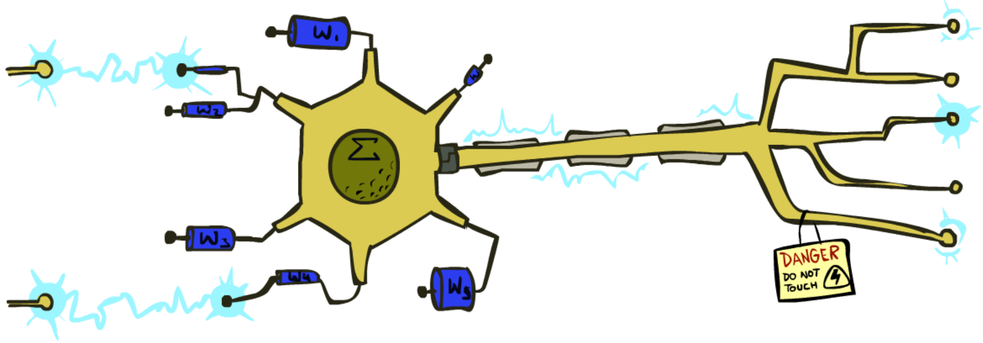
- Very loose inspiration: human neurons





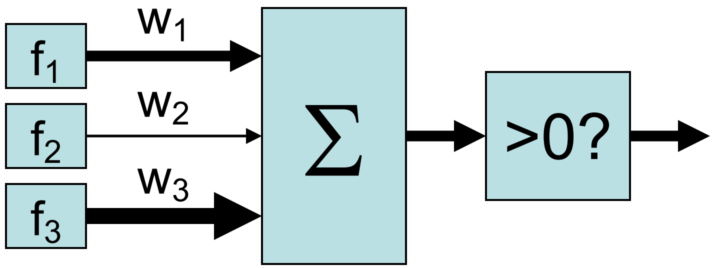
Linear Classifiers

- 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)*

$$w \cdot f(x_1)$$

```
# free      : 4
YOUR_NAME   :-1
MISSPELLED  : 1
FROM_FRIEND :-3
...
```

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

$$w \cdot f(x_2)$$

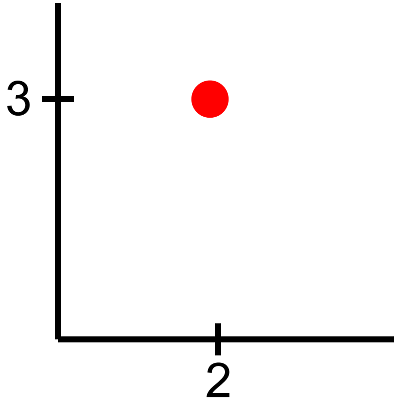
```
# free      : 4
YOUR_NAME   :-1
MISSPELLED  : 1
FROM_FRIEND :-3
...
```

```
# free      : 0
YOUR_NAME   : 1
MISSPELLED  : 1
FROM_FRIEND : 1
...
```

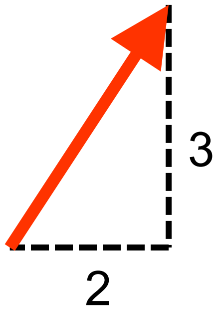
Do these weights make sense for spam classification?

Review: Vectors

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



A **point** on a coordinate grid

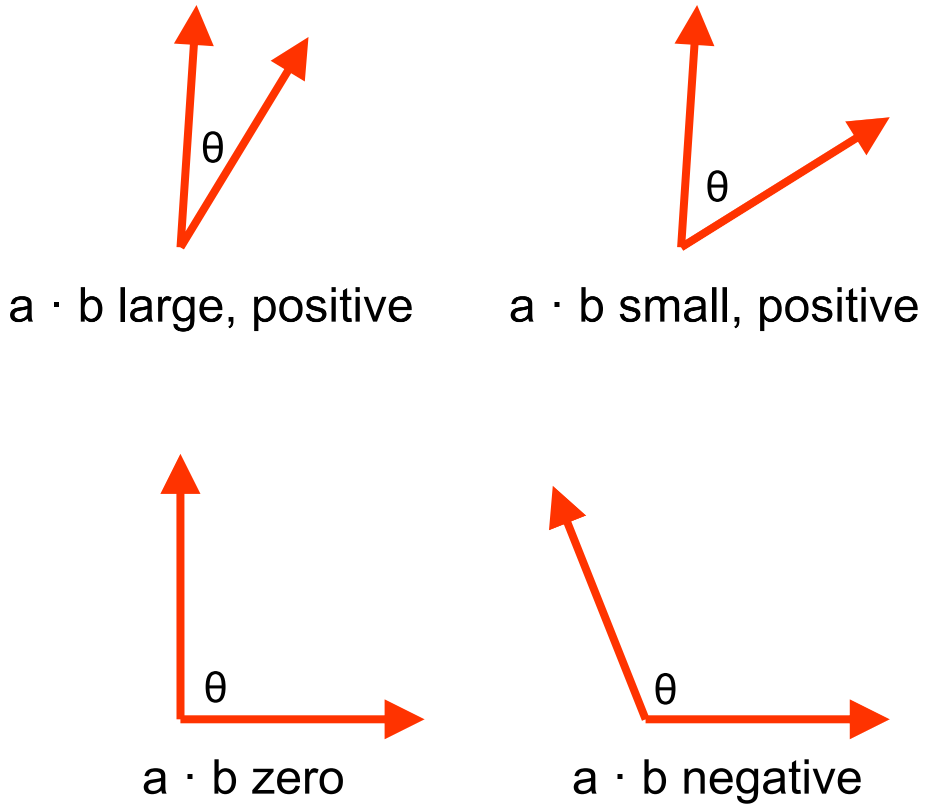


A **vector** in space. Notice we are not on a coordinate grid.

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

Review: Vectors

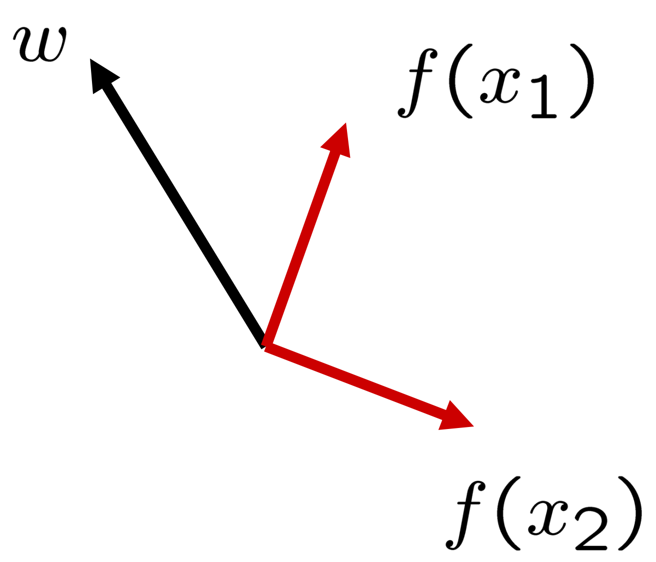
- Definition of dot product:
  - a · b = $\sum_i a_i b_i$ = |a| |b| cos(θ)
  - θ is the angle between the vectors a and b
- Consequences of this definition:
  - Vectors closer together
    = "similar" vectors
    = smaller angle θ between vectors
    = larger (more positive) dot product
  - If θ < 90°,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



Weights

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

```
# free        :  4
YOUR_NAME     : -1
MISSPELLED    :  1
FROM_FRIEND   : -3
...
```

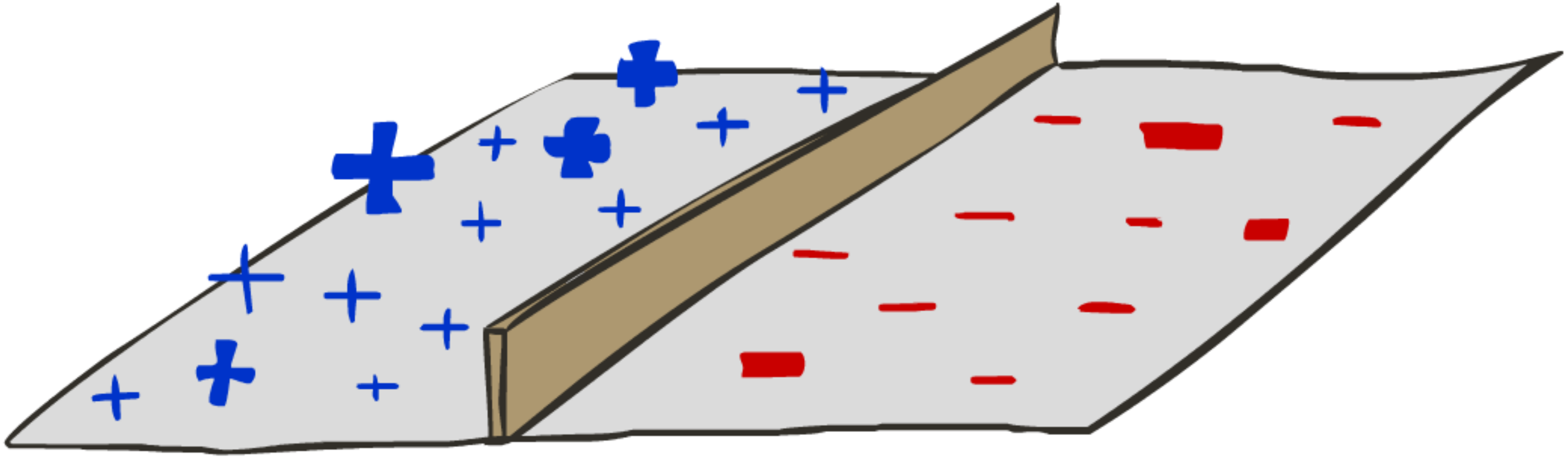



```
# free        : 2
YOUR_NAME     : 0
MISSPELLED    : 2
FROM_FRIEND   : 0
...
```

```
# free        : 0
YOUR_NAME     : 1
MISSPELLED    : 1
FROM_FRIEND   : 1
...
```

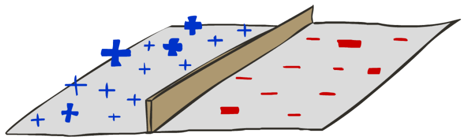
Dot product $w \cdot f$ *positive means the positive class*

Decision Rules



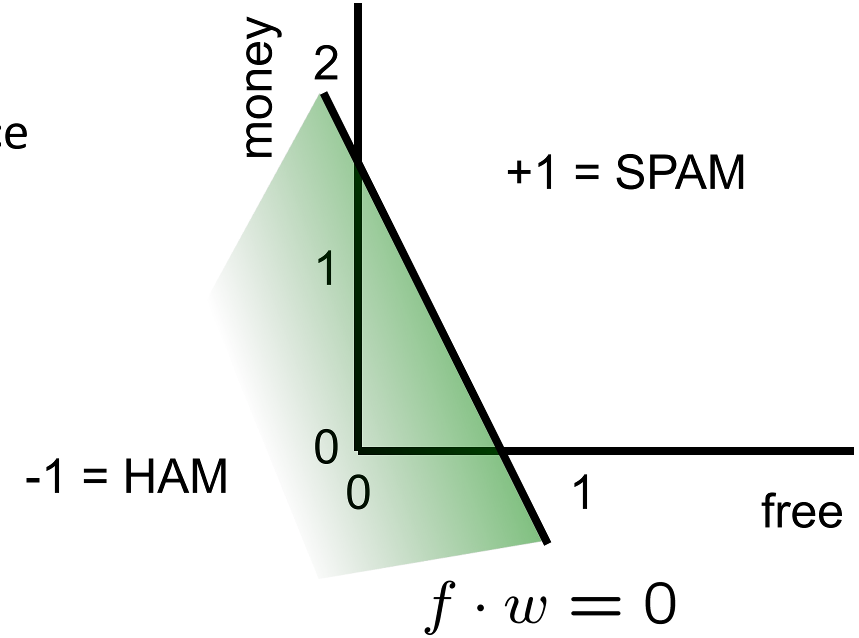
Binary Decision Rule

- 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 · w > 0 when 4*free + 2*money > 0
    f · w < 0 when 4*free + 2*money < 0
    These equations correspond to two halves of the feature space
  - f · 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

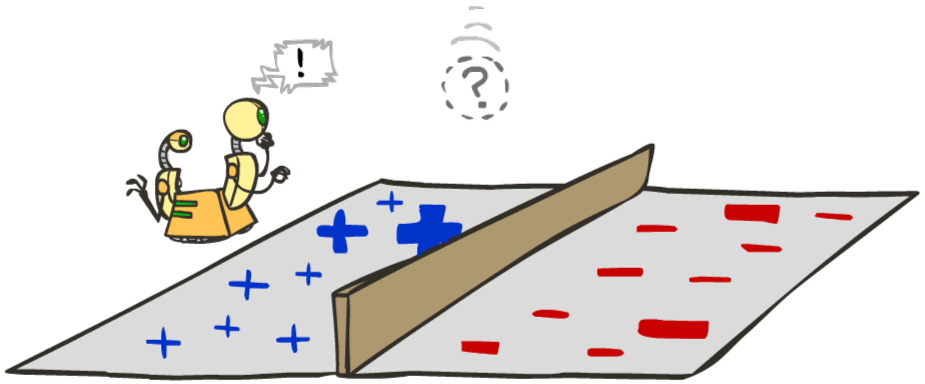
free	:	4
money	:	2



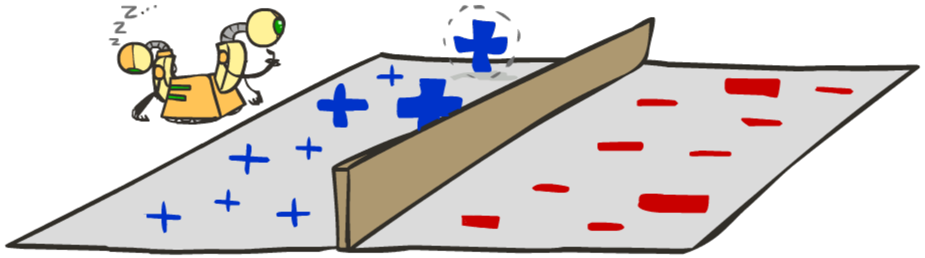
Weight Updates

Learning: Binary Perceptron

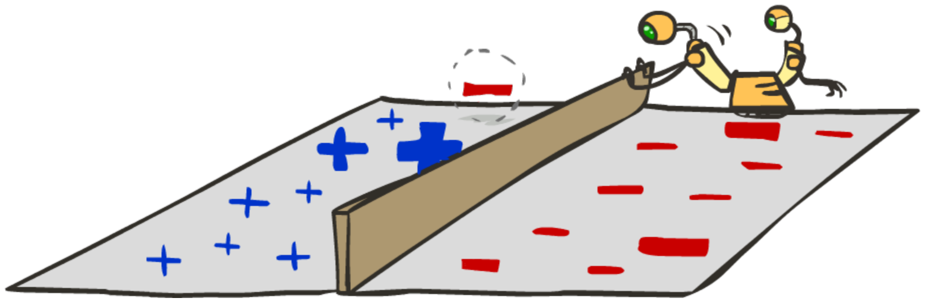
- Start with weights = 0
- For each training instance:
  - Classify with current weights



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



  - If wrong: adjust the weight vector



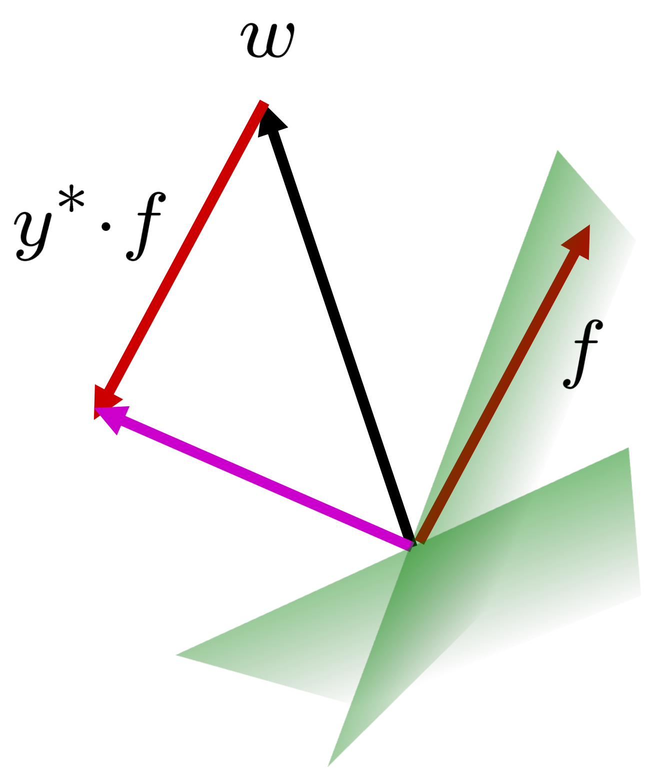
Learning: Binary Perceptron

- 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.

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

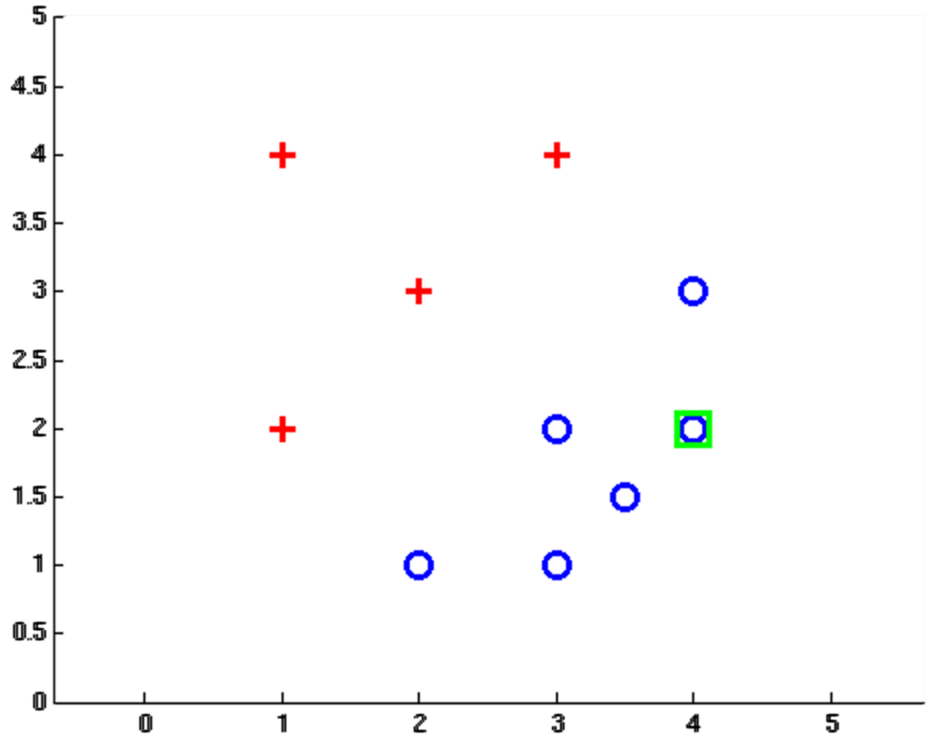


Learning: Binary Perceptron

- 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

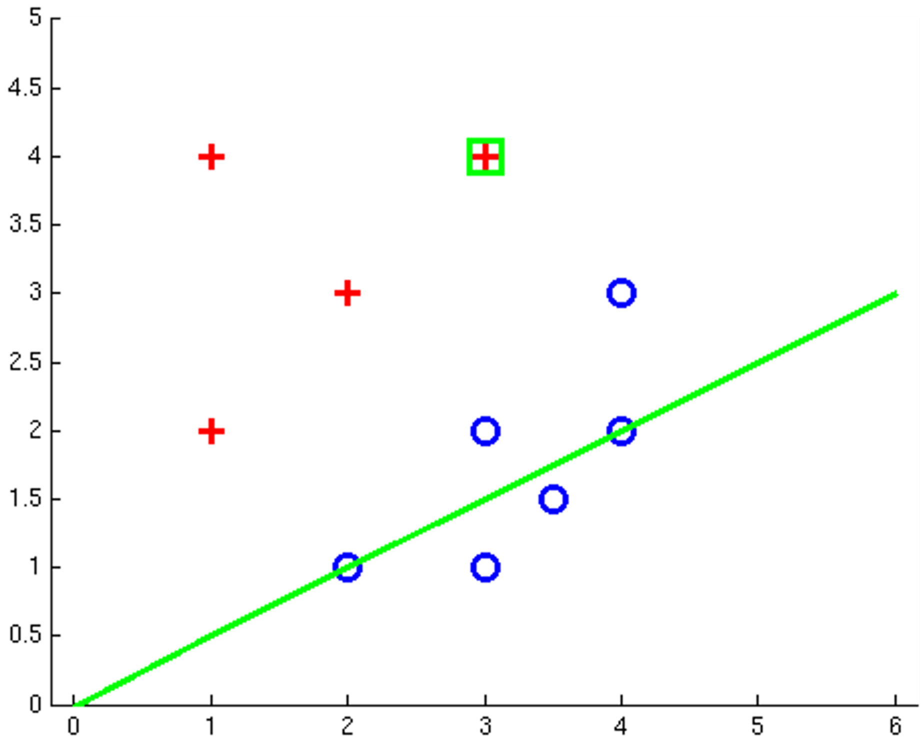
Examples: Perceptron

- Separable Case



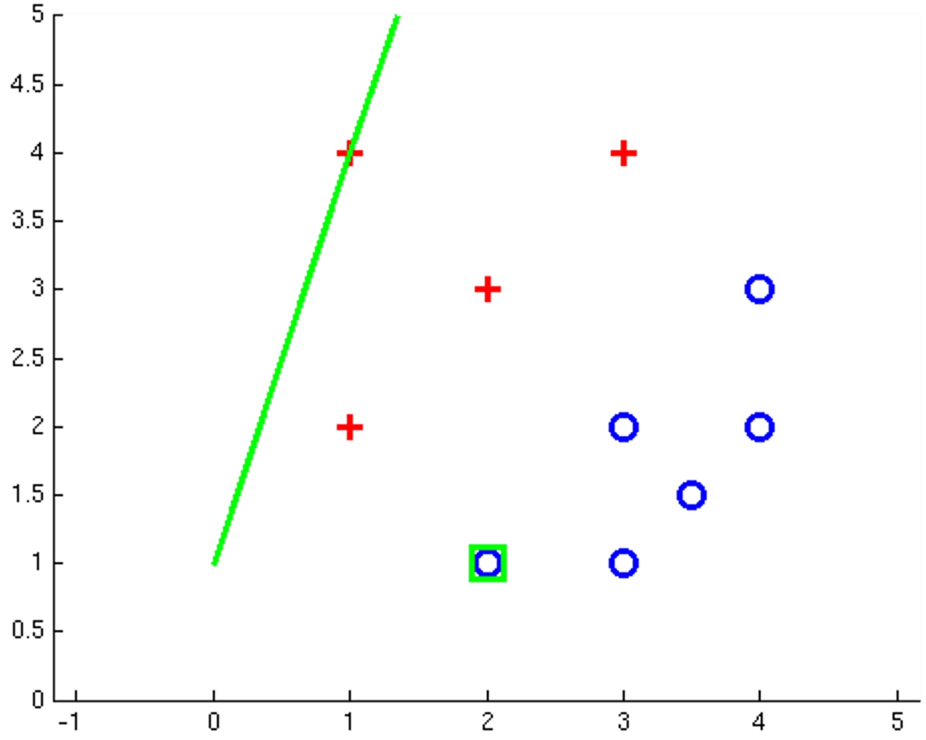
Examples: Perceptron

- Separable Case



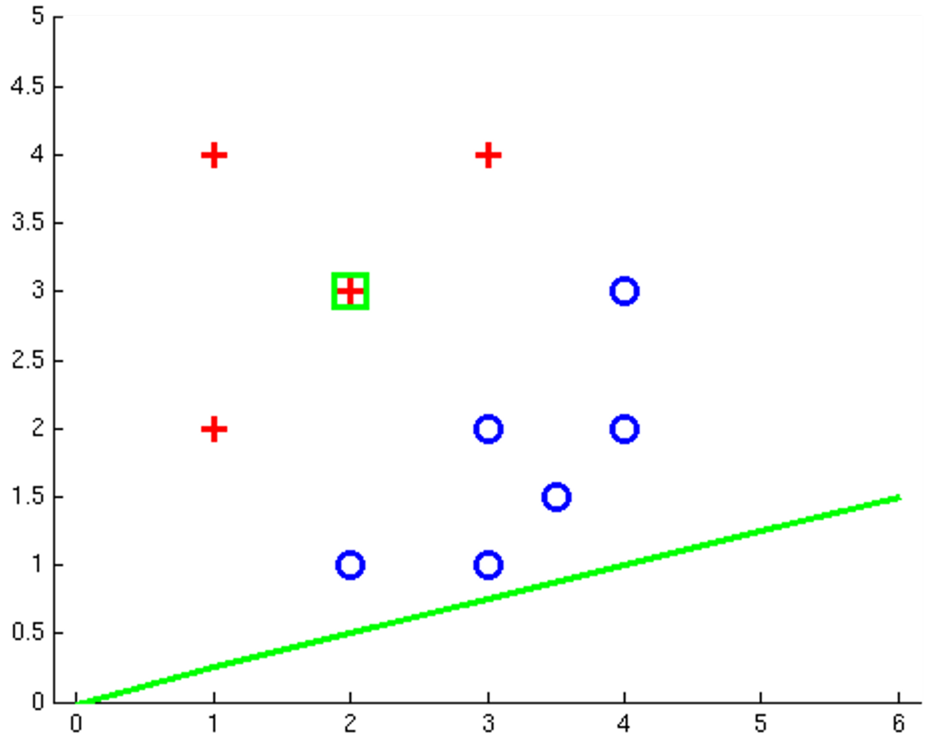
Examples: Perceptron

- Separable Case



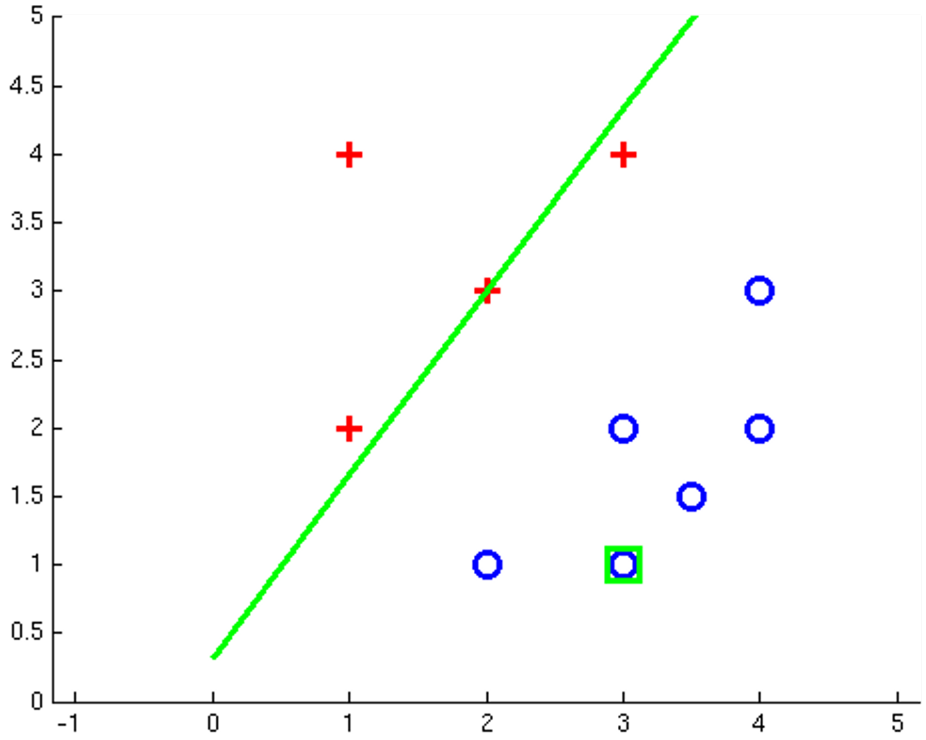
Examples: Perceptron

- Separable Case



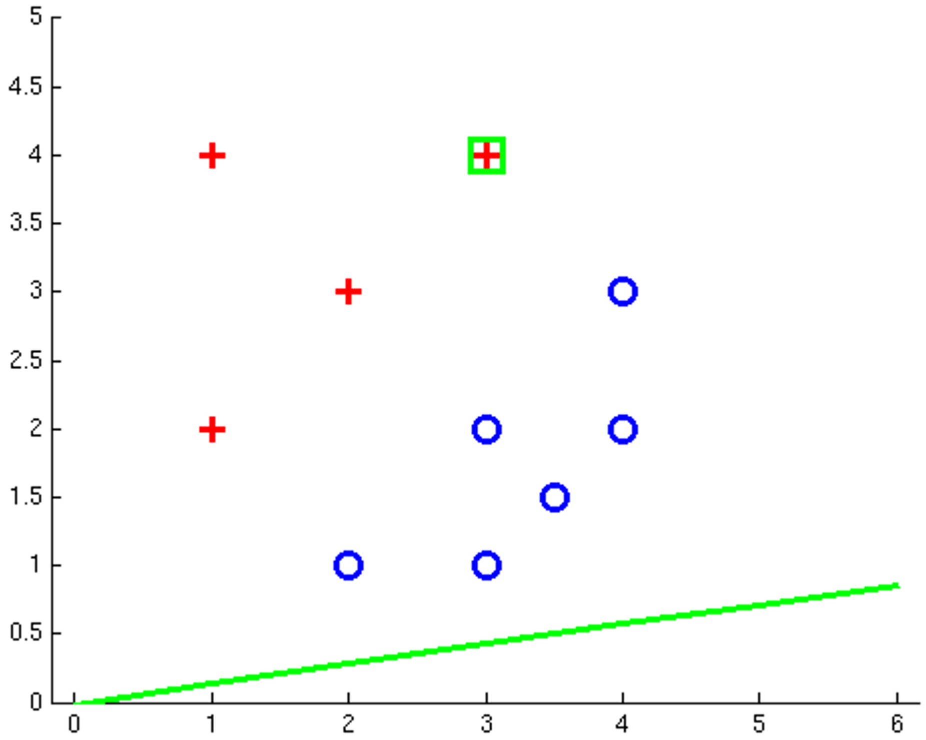
Examples: Perceptron

- Separable Case



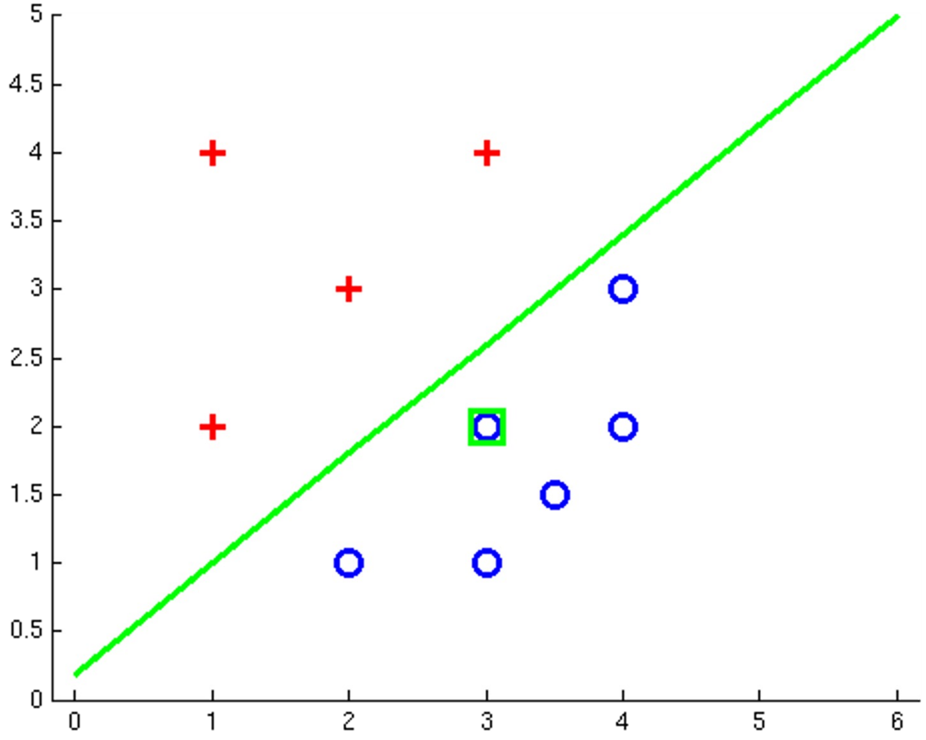
Examples: Perceptron

- Separable Case



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

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

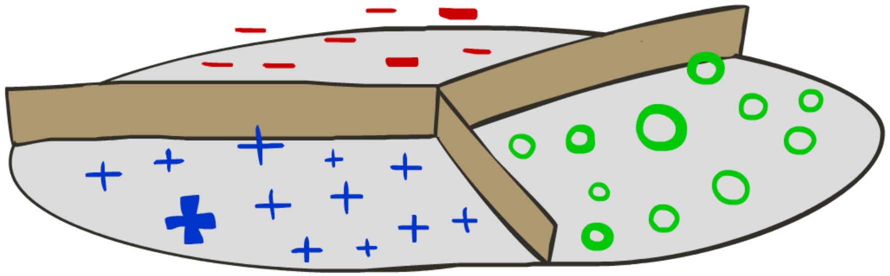
$$w_y$$

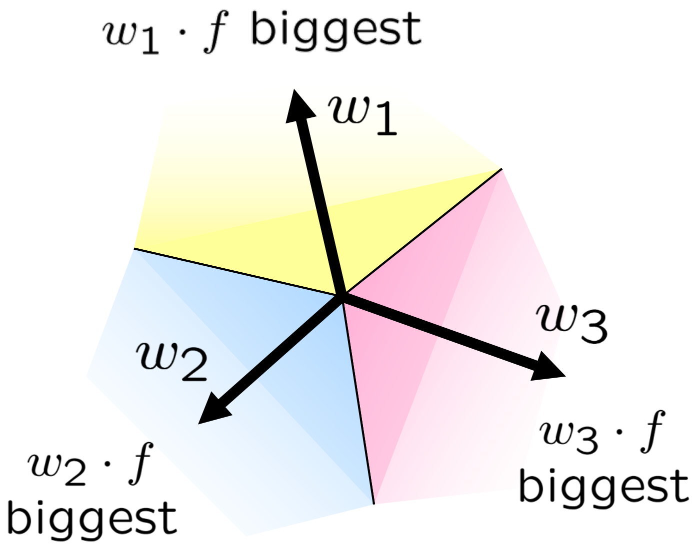
  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - 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

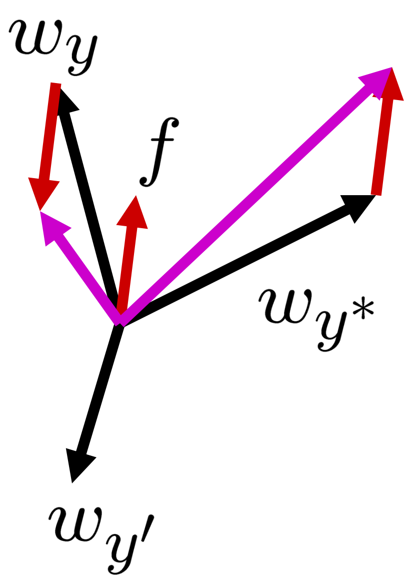
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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"

"win the election"

"win the game"

w_{SPORTS}

```
BIAS  : 1
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

$w_{POLITICS}$

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

w_{TECH}

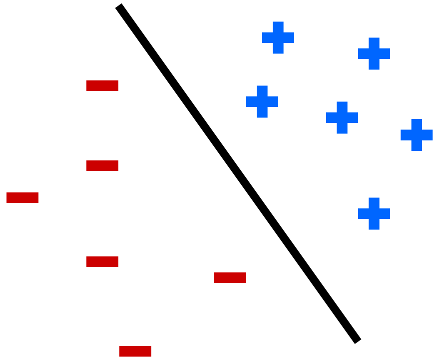
```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

Properties of Perceptrons

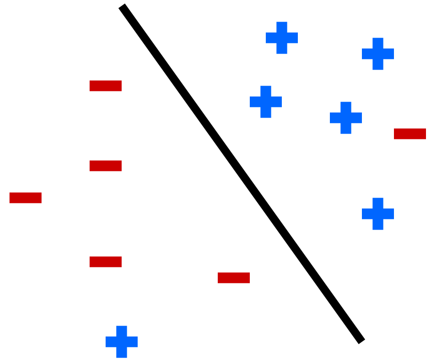
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

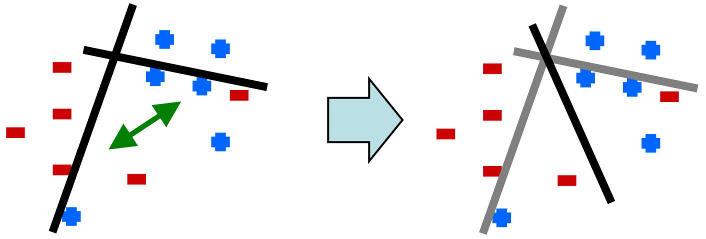


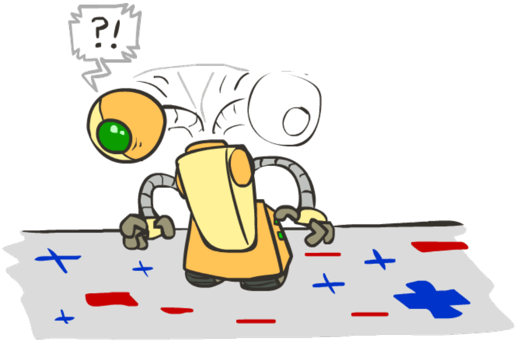
Non-Separable



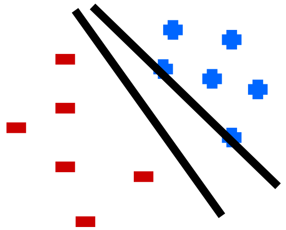
Problems with the Perceptron

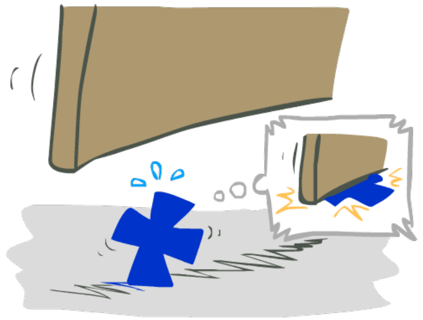
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



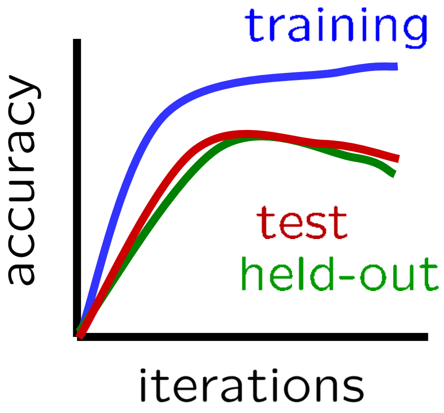


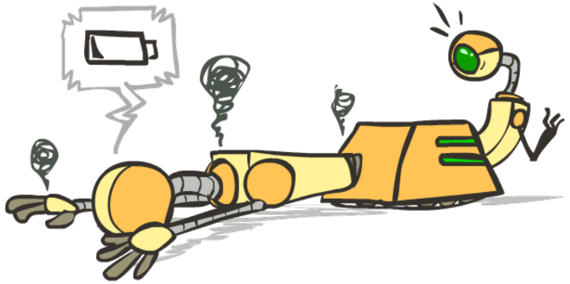
- Mediocre generalization: finds a "barely" separating solution



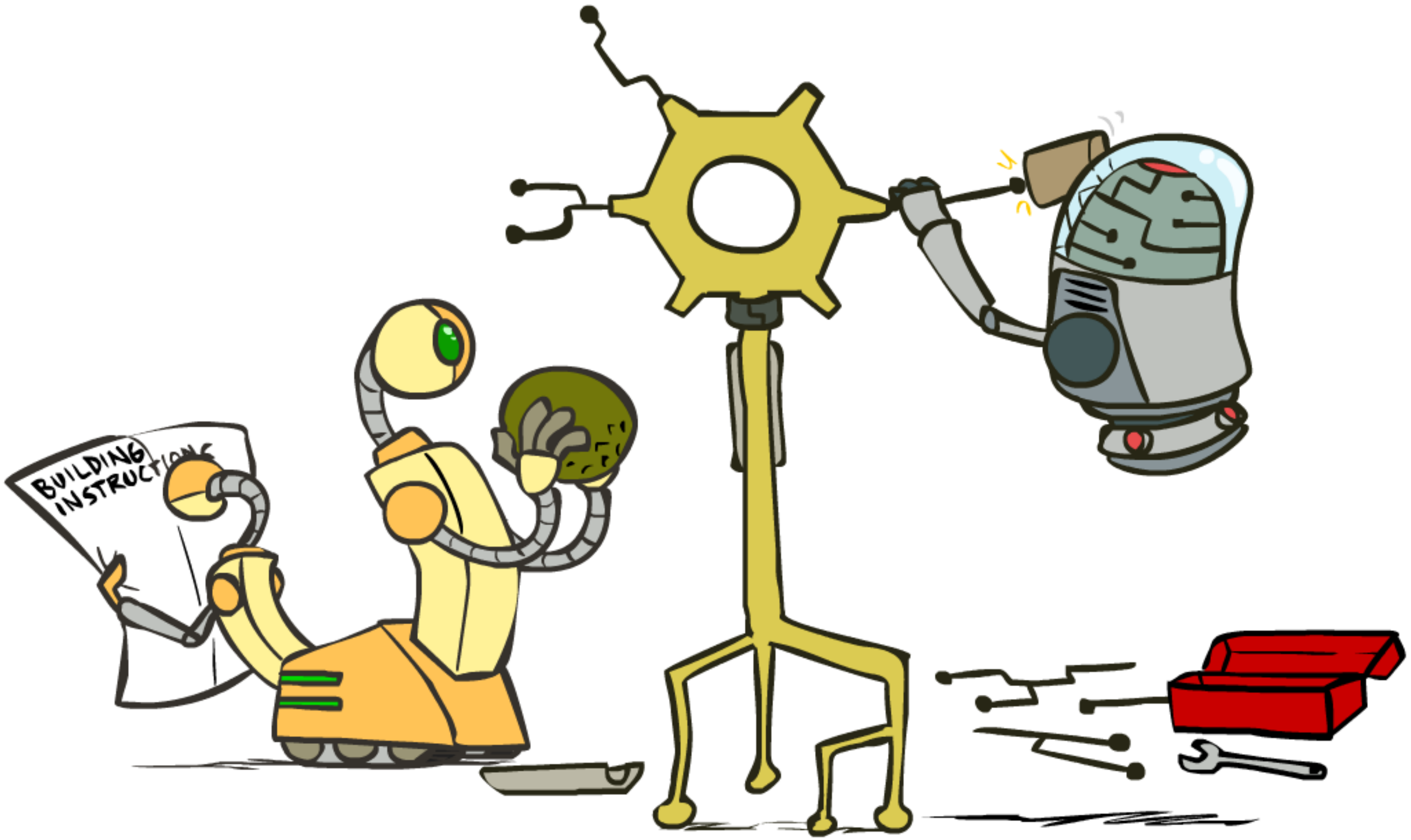


- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



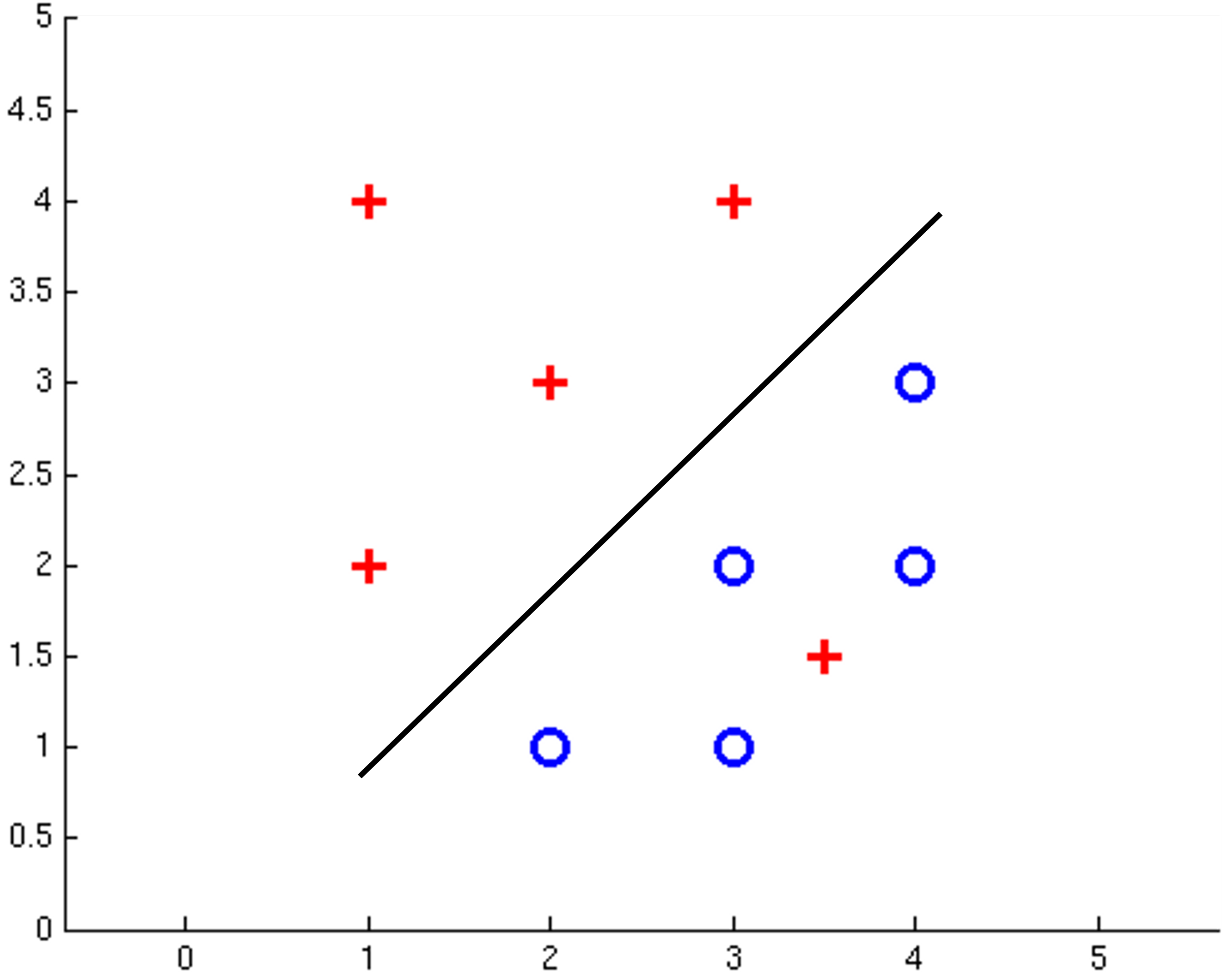


Improving the Perceptron

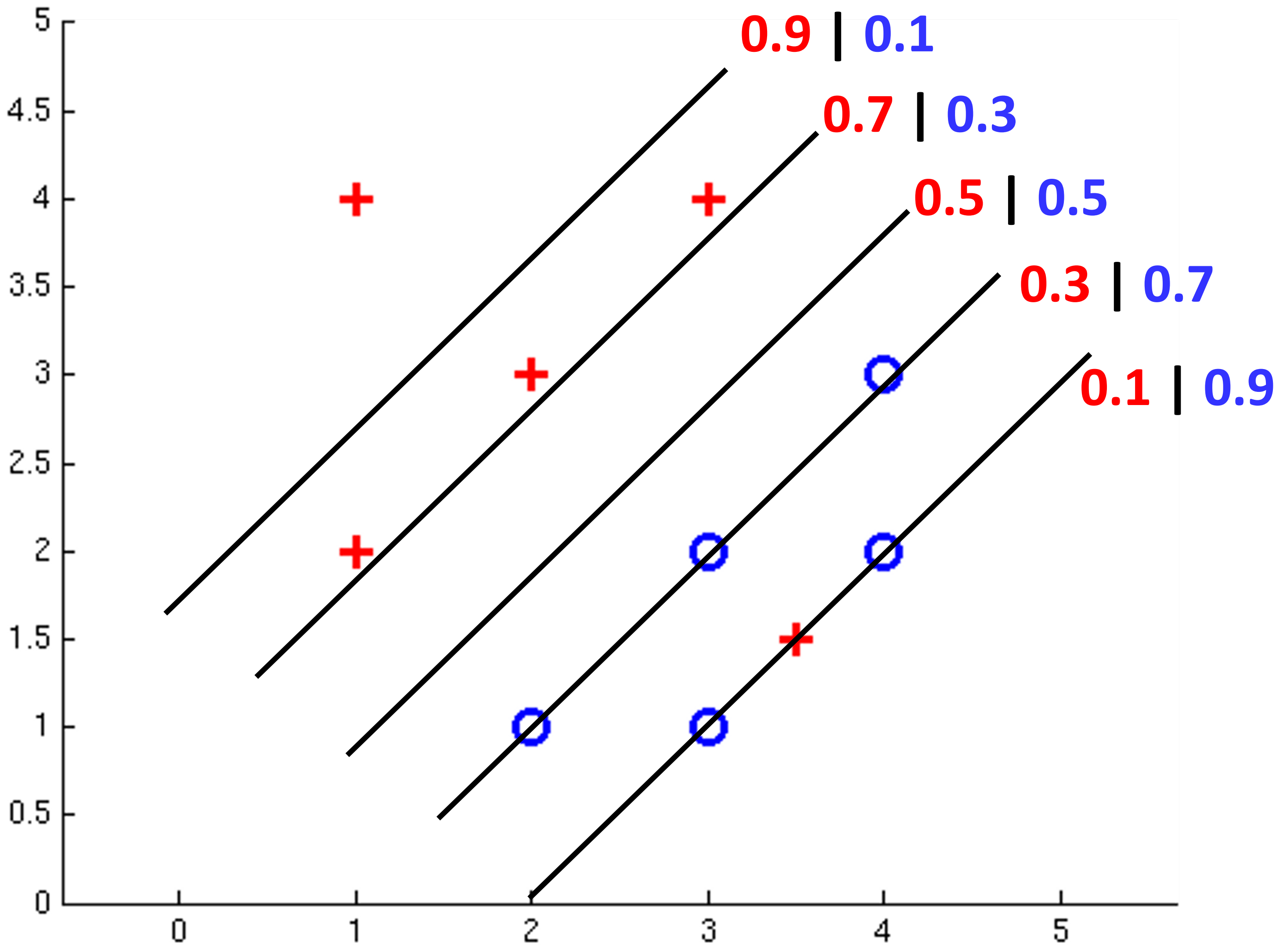


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



Non-Separable Case: Probabilistic Decision

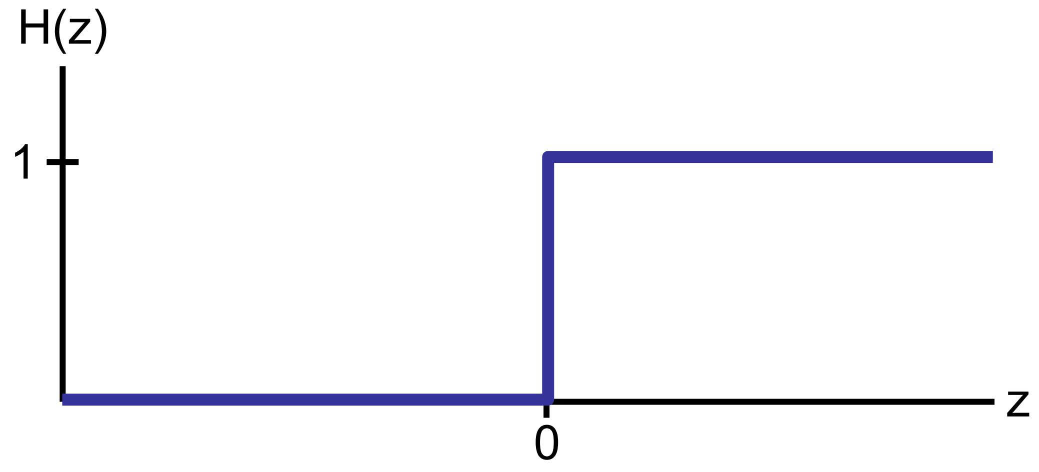


How to get deterministic decisions?

- 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

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



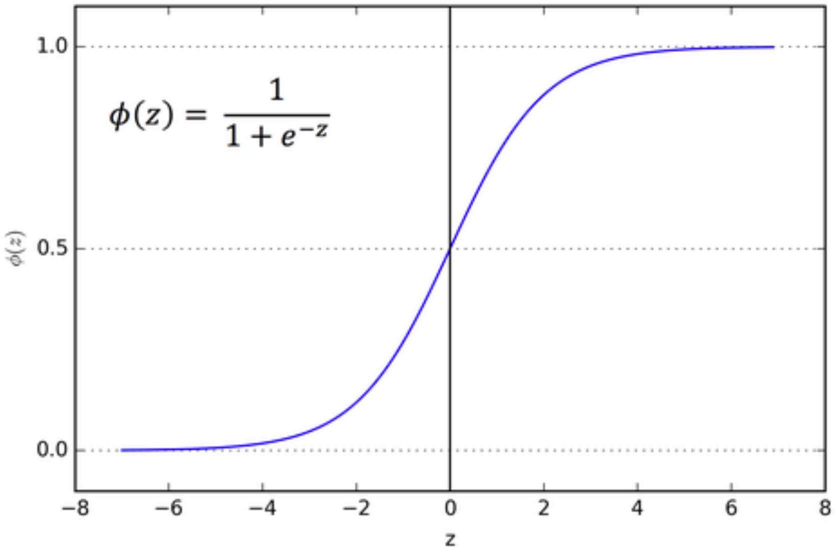
- z = output of perceptron
  H(z) = probability the class is +1, according to the classifier

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

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

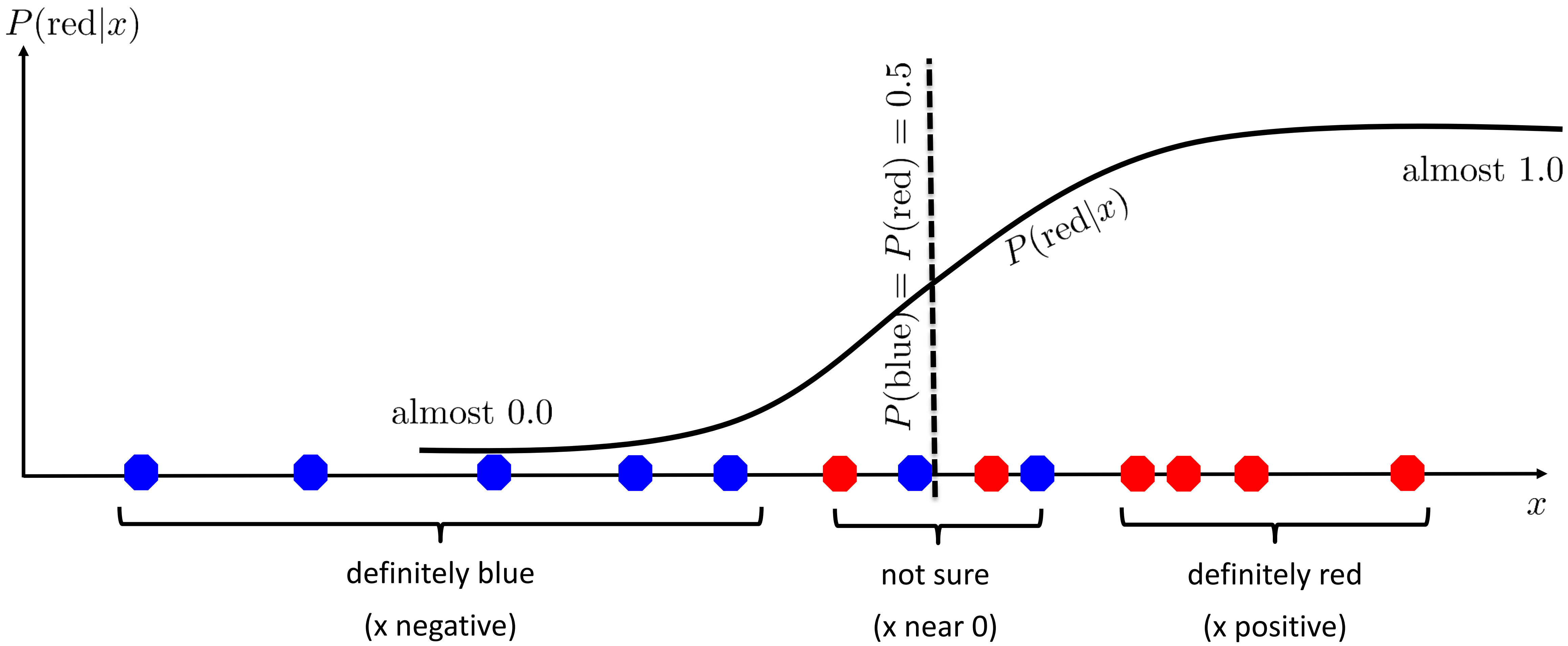


- z = output of perceptron
  $\phi(z)$ = probability the class is +1, according to the classifier

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:

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

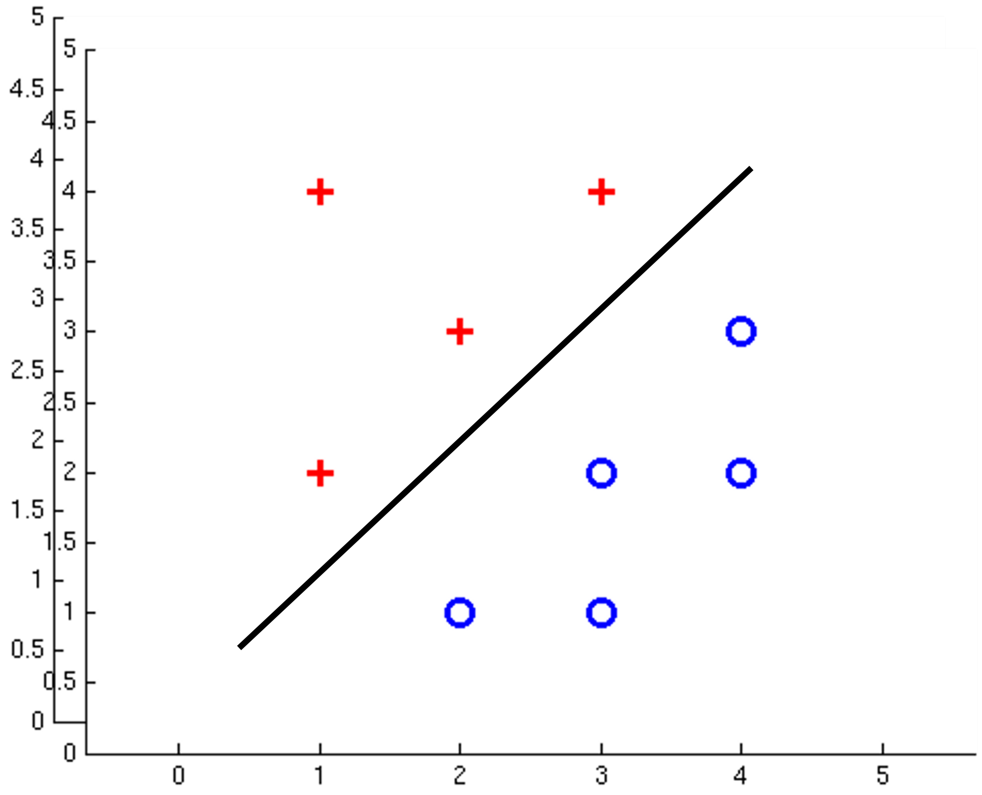
with:

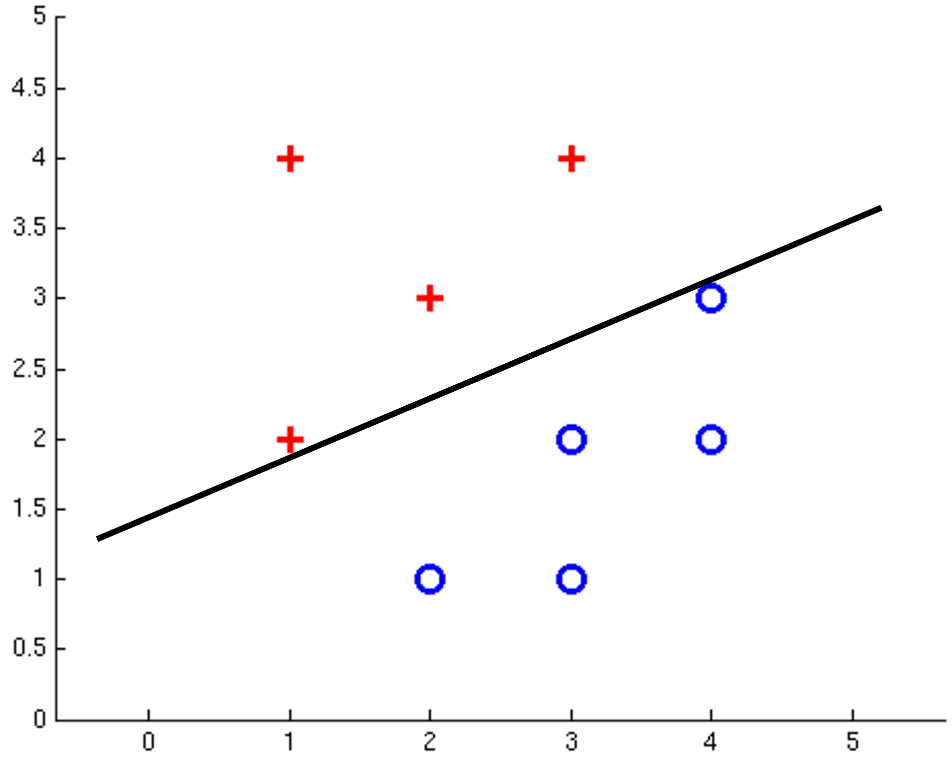
$$P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

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

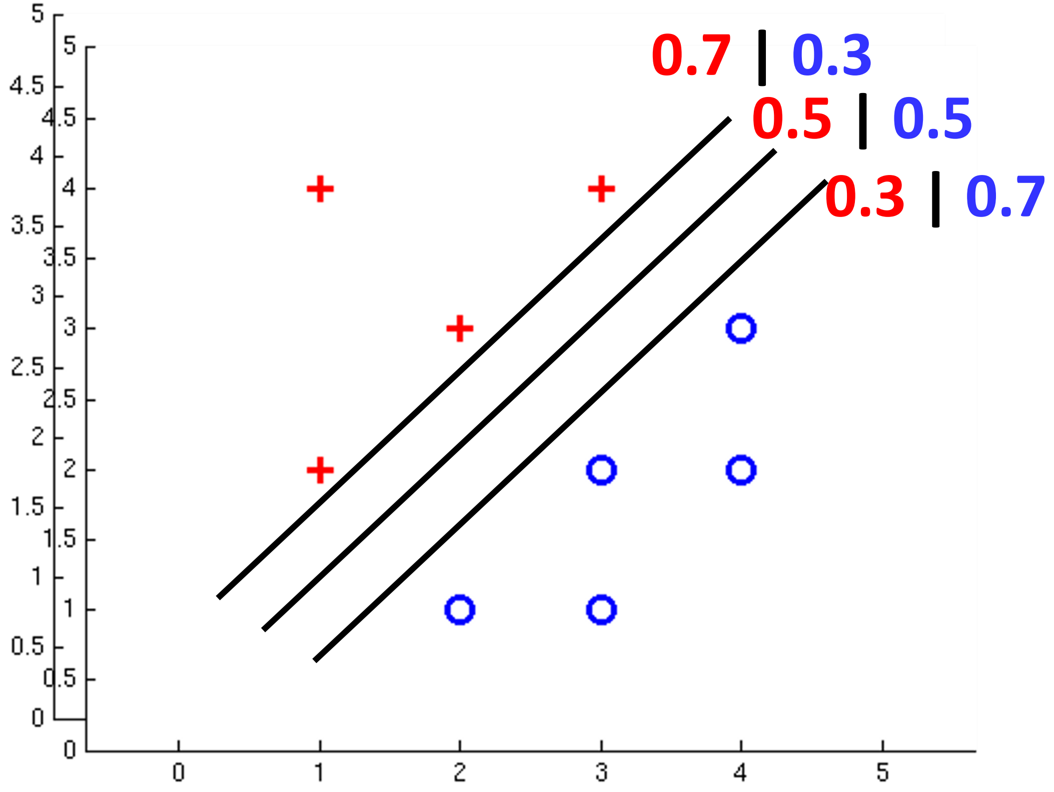
= Logistic Regression

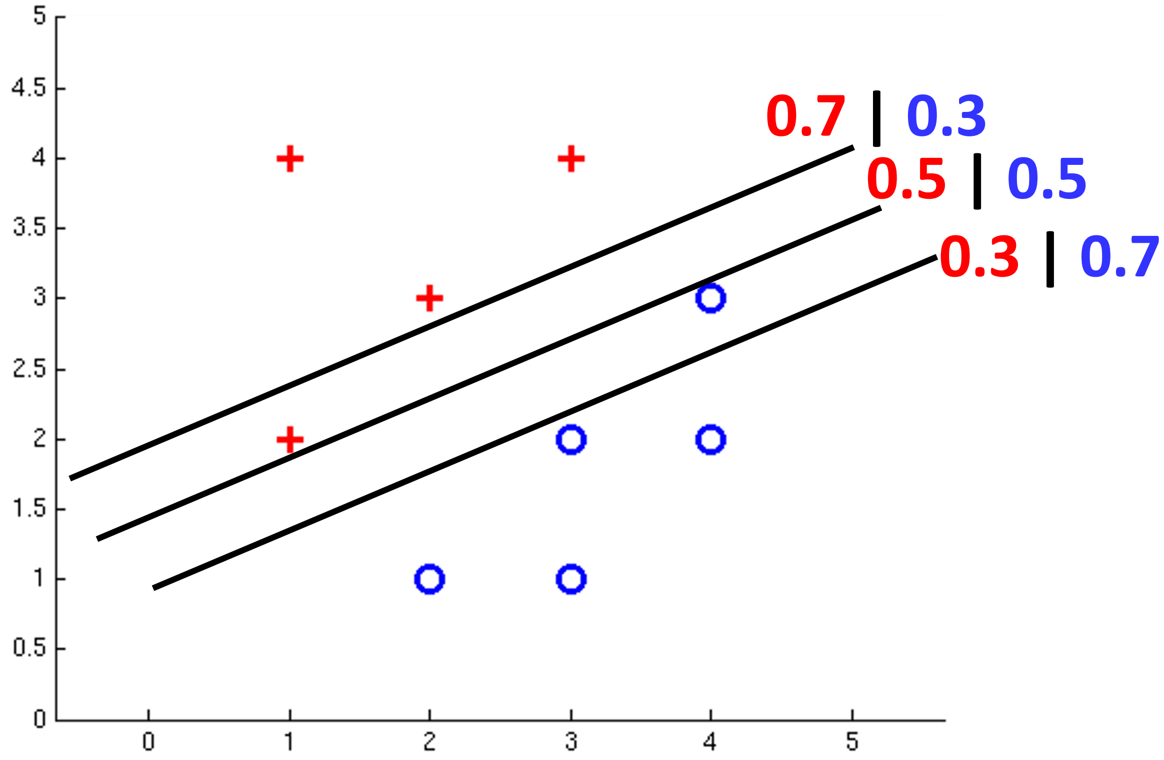
Separable Case: Deterministic Decision – Many Options





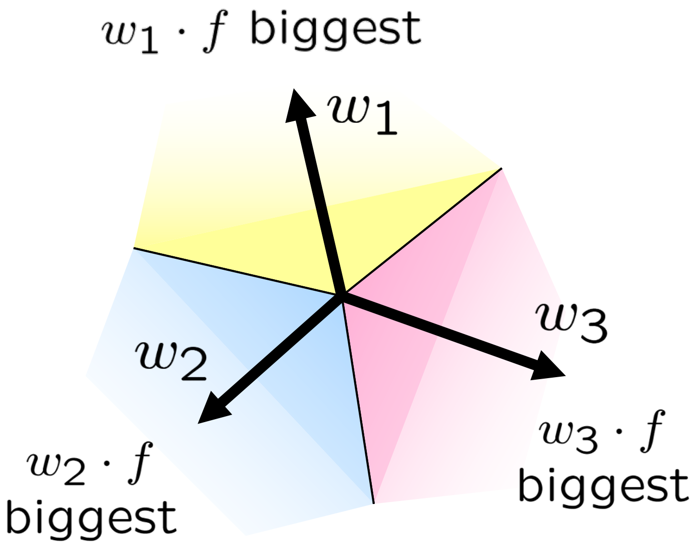
Separable Case: Probabilistic Decision – Clear Preference





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?

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

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?

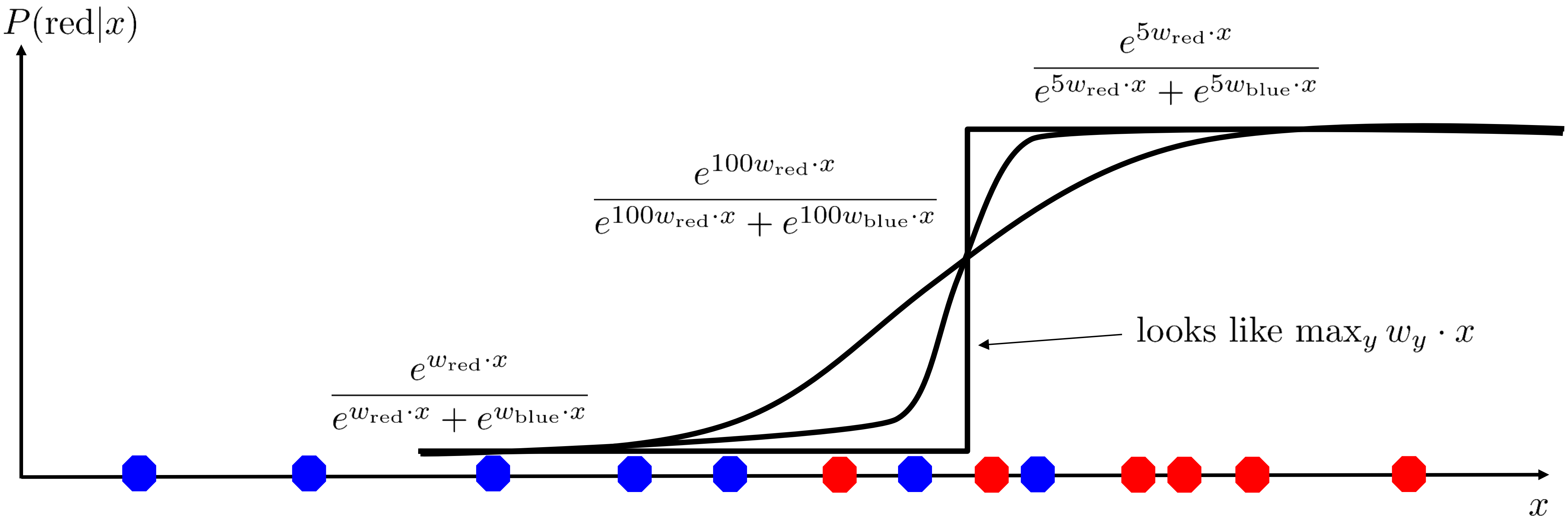
- Maximum likelihood estimation:

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

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

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: Compute $w_y \cdot f(x)$ 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_{red} = 0 becomes:

Sigmoid

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

Next Lecture

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

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