

Announcements

- Homework 8
 - due Tuesday 4/12 at 11:59pm
- Project 5:
 - due Friday 4/15 at 5pm.

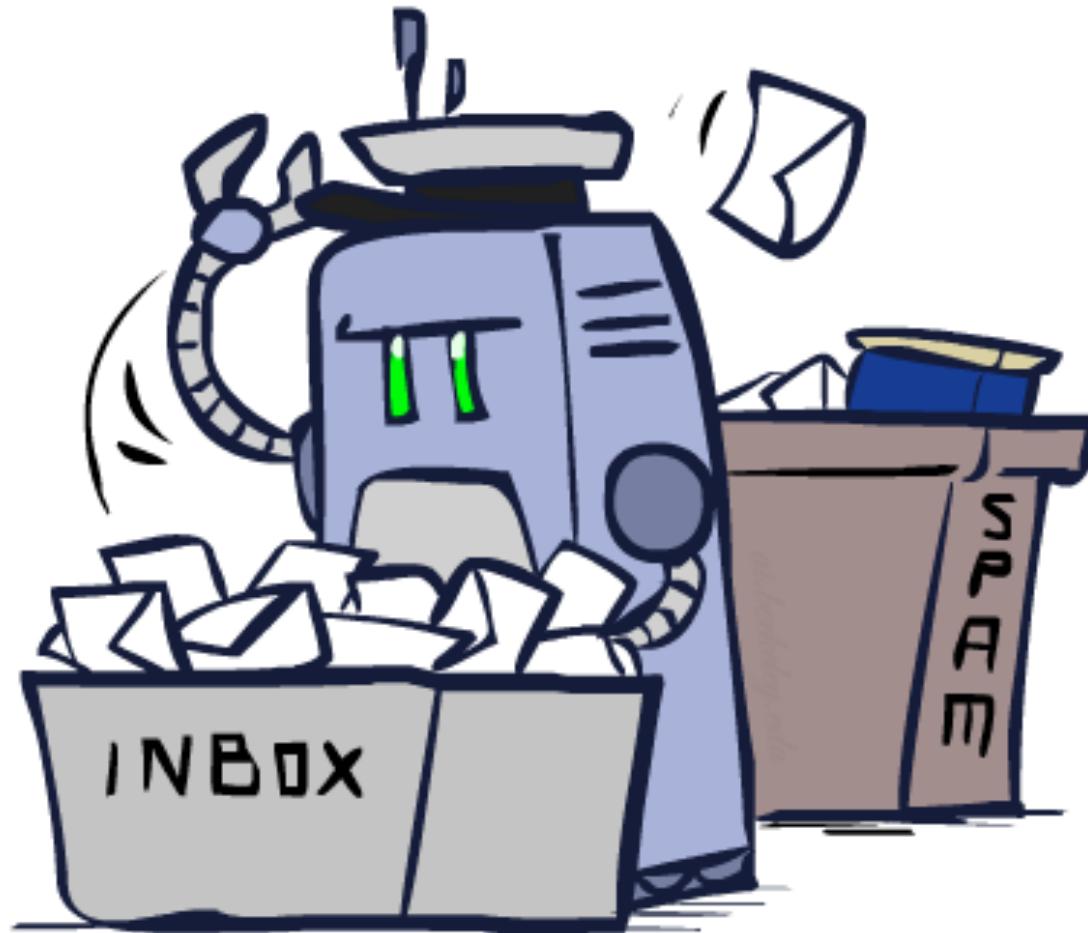
CS 188: Artificial Intelligence

Naïve Bayes Wrap-Up



Instructors: Pieter Abbeel & Anca Dragan --- University of California, Berkeley

Classification



Example: Spam Filter

- Input: an email
- Output: spam/ham
- Setup:
 - Get a large collection of example emails, each labeled "spam" or "ham"
 - Note: someone has to hand label all this data!
 - Want to learn to predict labels of new, future emails
- Features: The attributes used to make the ham / spam decision
 - Words: FREE!
 - Text Patterns: \$dd, CAPS
 - Non-text: SenderInContacts
 - ...



Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential 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, I know 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.

Example: 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: $(6,8)=\text{ON}$
 - Shape Patterns: NumComponents, AspectRatio, NumLoops
 - ...

0

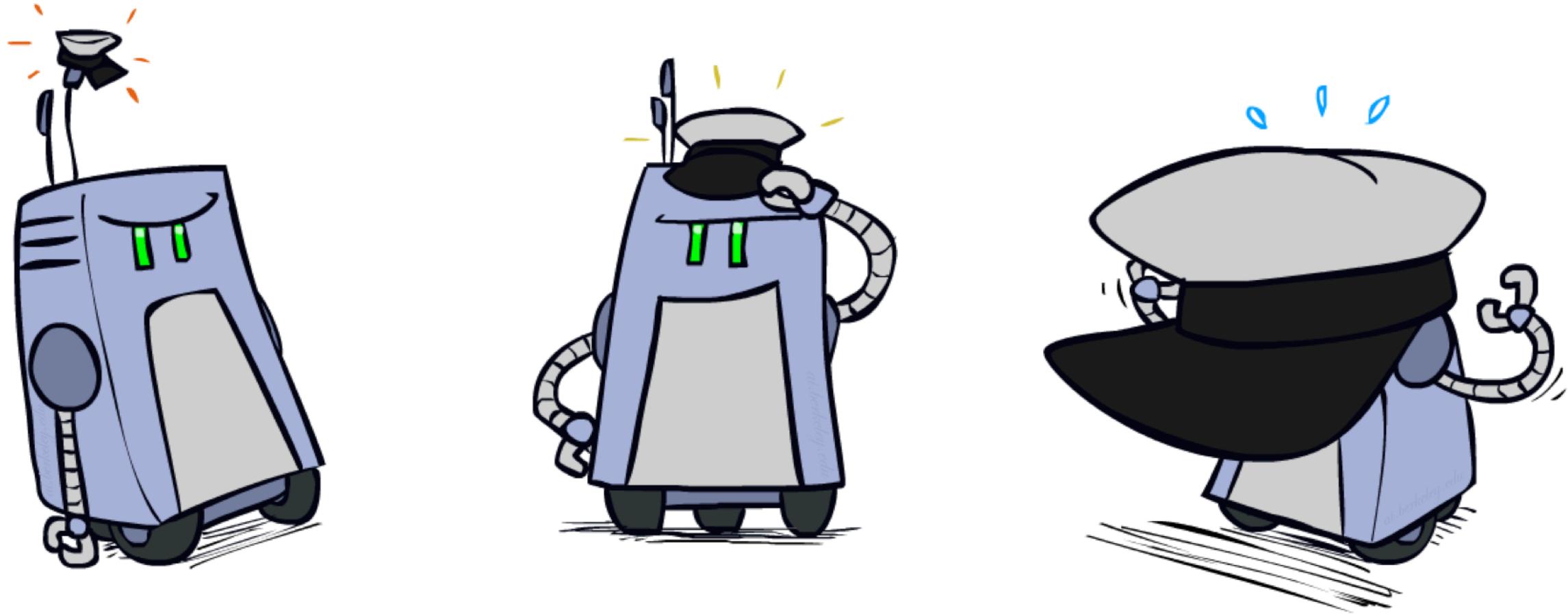
1

2

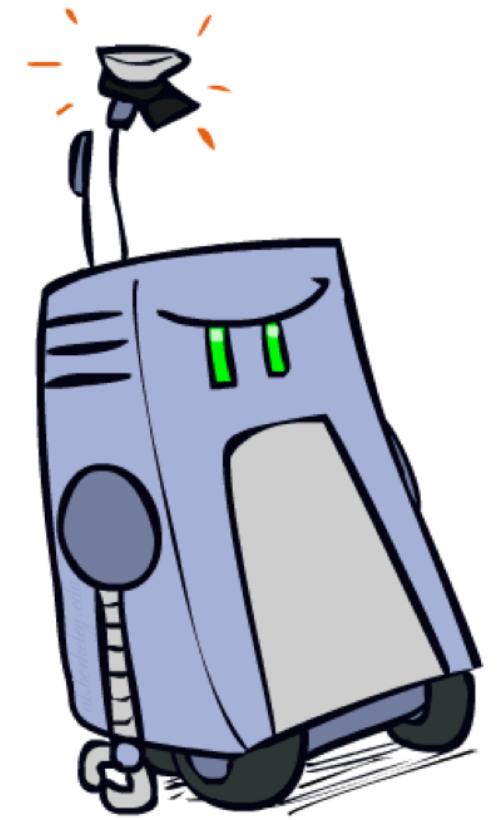
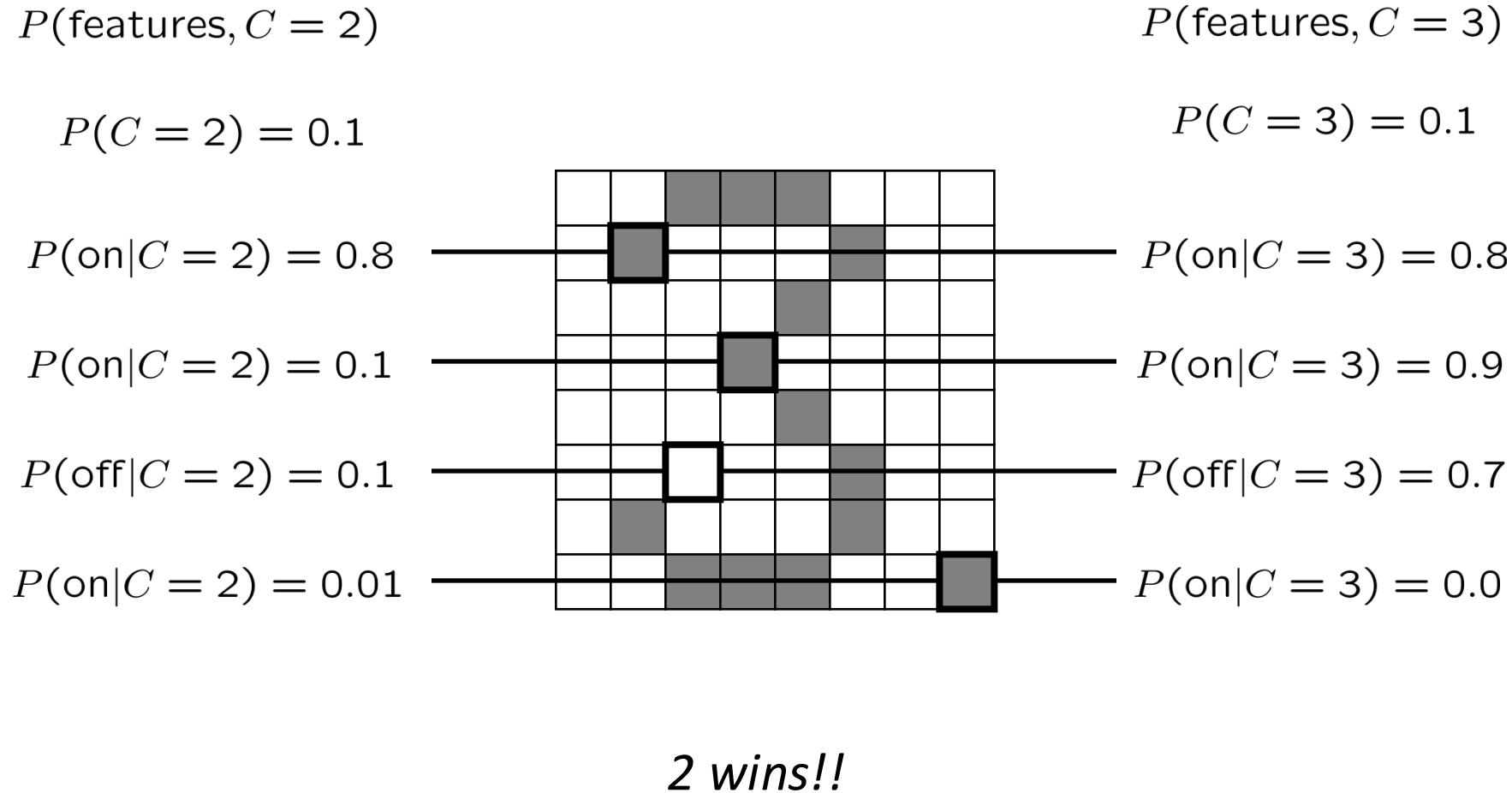
1

??

Generalization and Overfitting



Example: Overfitting



Example: Overfitting

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

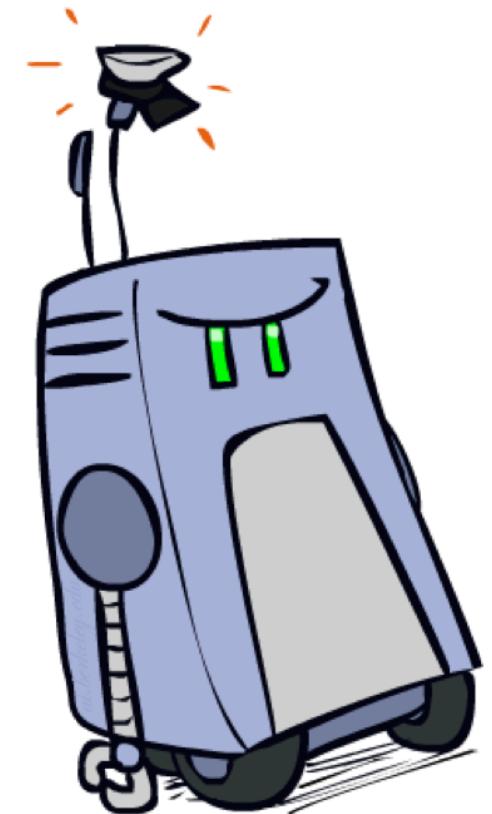
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

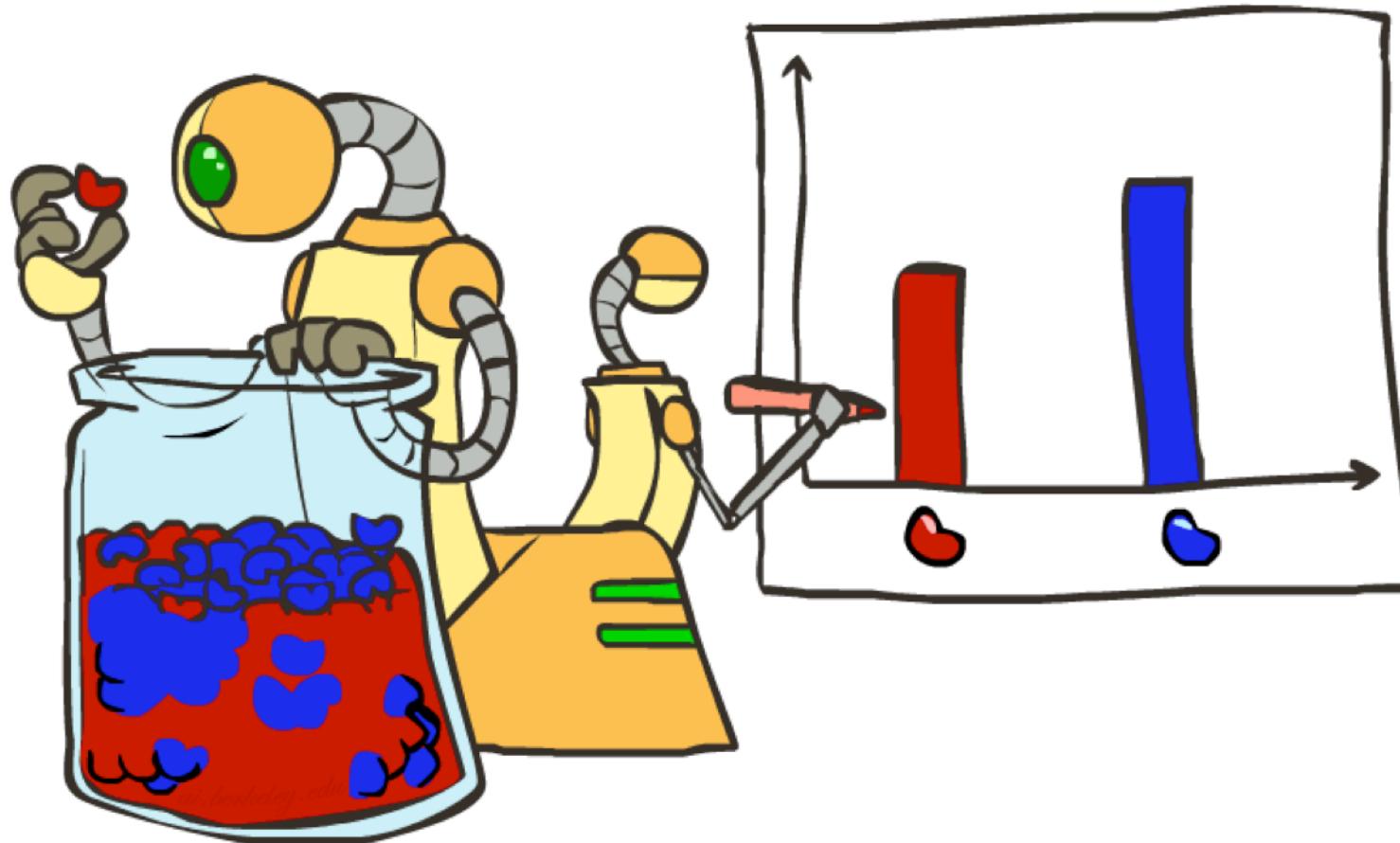
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

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

What went wrong here?



Parameter Estimation



Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation*: ask a human (why is this hard?)
- *Empirically*: use training data (learning!)
 - E.g.: for each outcome x , look at the *empirical rate* of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

r r b

$$P_{\text{ML}}(\text{r}) = 2/3$$

- This is the estimate that maximizes the *likelihood of the data*

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

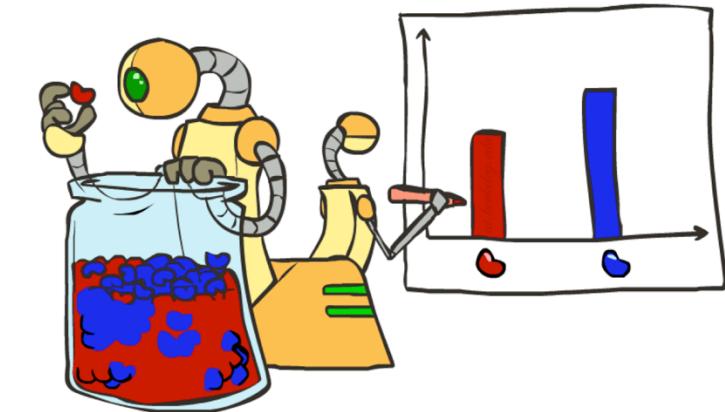
$$P(r) = \theta$$

$$P(b) = 1 - \theta$$

$$\max_\theta P(r)P(r)P(b)$$

$$\max_\theta \theta^2(1 - \theta)$$

$$2\theta - 3\theta^2$$

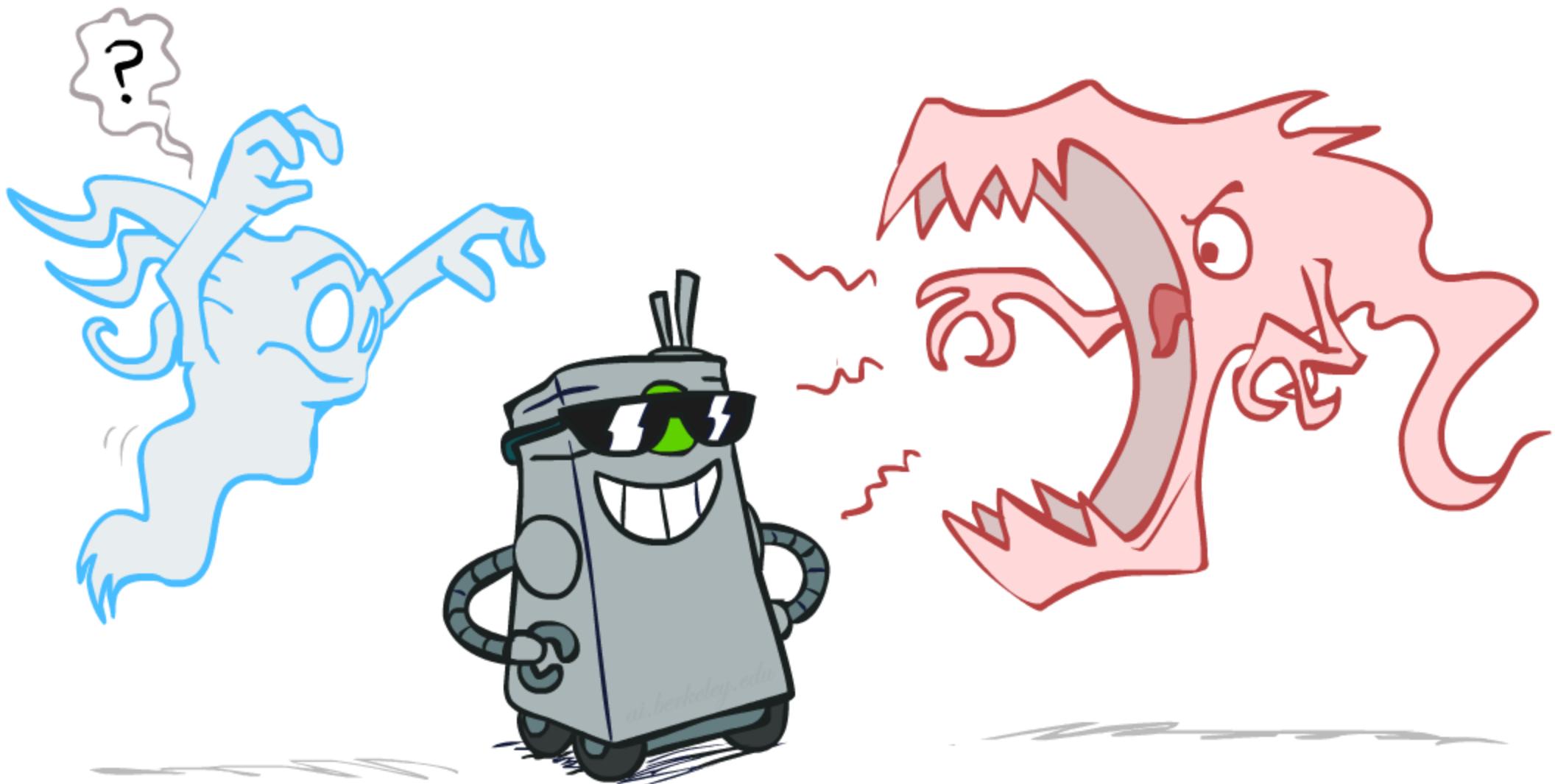


$$\theta = 0$$

$$2 - 3\theta = 0$$

$$\theta = 2/3$$

Smoothing



Maximum Likelihood?

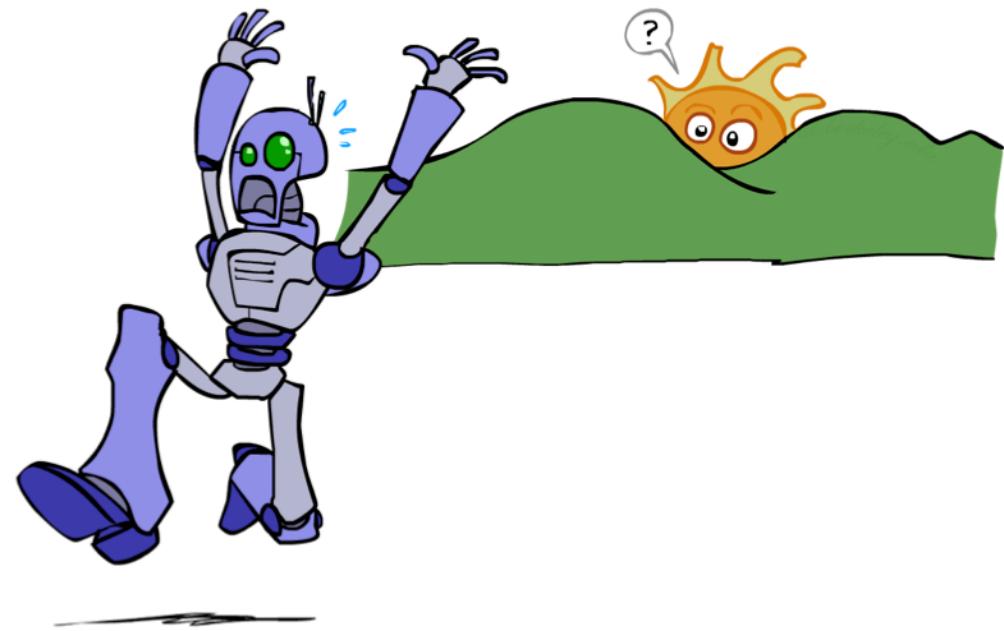
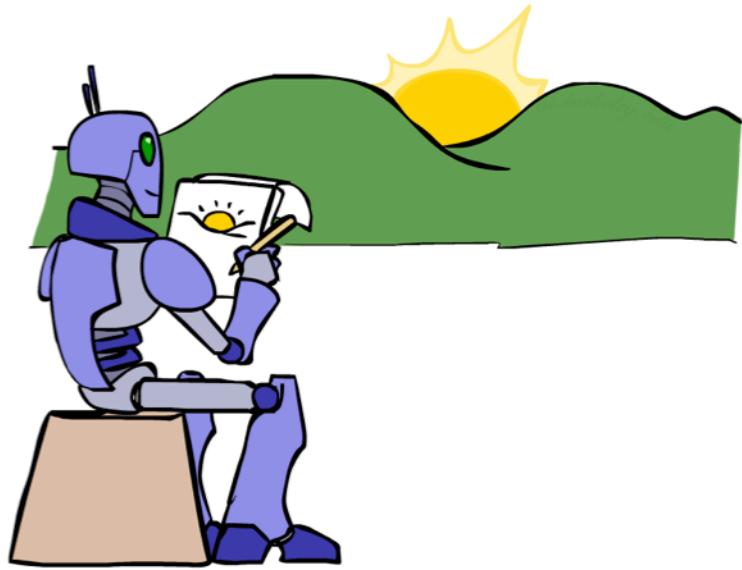
- Relative frequencies are the maximum likelihood estimates

$$\begin{aligned}\theta_{ML} &= \arg \max_{\theta} P(\mathbf{X}|\theta) \\ &= \arg \max_{\theta} \prod_i P_{\theta}(X_i)\end{aligned}\quad \Rightarrow \quad P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

- Another option is to consider the most likely parameter value given the data

$$\begin{aligned}\theta_{MAP} &= \arg \max_{\theta} P(\theta|\mathbf{X}) \\ &= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \\ &= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)\end{aligned}\quad \Rightarrow \quad \text{????}$$

Unseen Events



Laplace Smoothing

- Laplace's estimate:

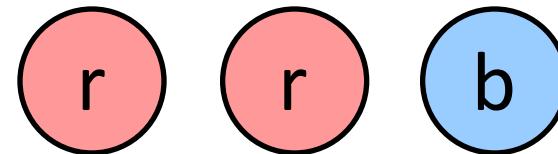
- Pretend you saw every outcome once more than you actually did

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

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

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

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



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

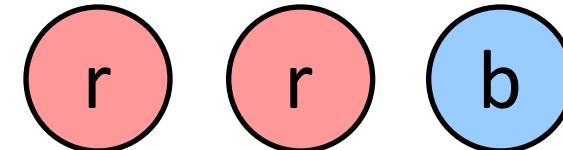
Laplace Smoothing

- 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



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

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

- Laplace for conditionals:

- Smooth each condition independently:

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

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

Estimation: Linear Interpolation*

- In practice, Laplace often performs poorly for $P(X|Y)$:
 - When $|X|$ is very large
 - When $|Y|$ is very large
- Another option: linear interpolation
 - Also get the empirical $P(X)$ from the data
 - Make sure the estimate of $P(X|Y)$ isn't too different from the empirical $P(X)$

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha) \hat{P}(x)$$

- What if α is 0? 1?
- For even better ways to estimate parameters, as well as details of the math, see cs281a, cs288

Real NB: Smoothing

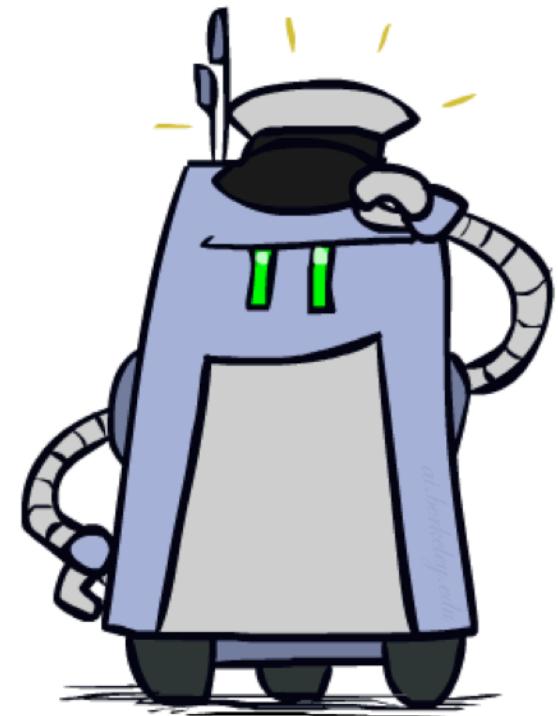
- For real classification problems, smoothing is critical
- New odds ratios:

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

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

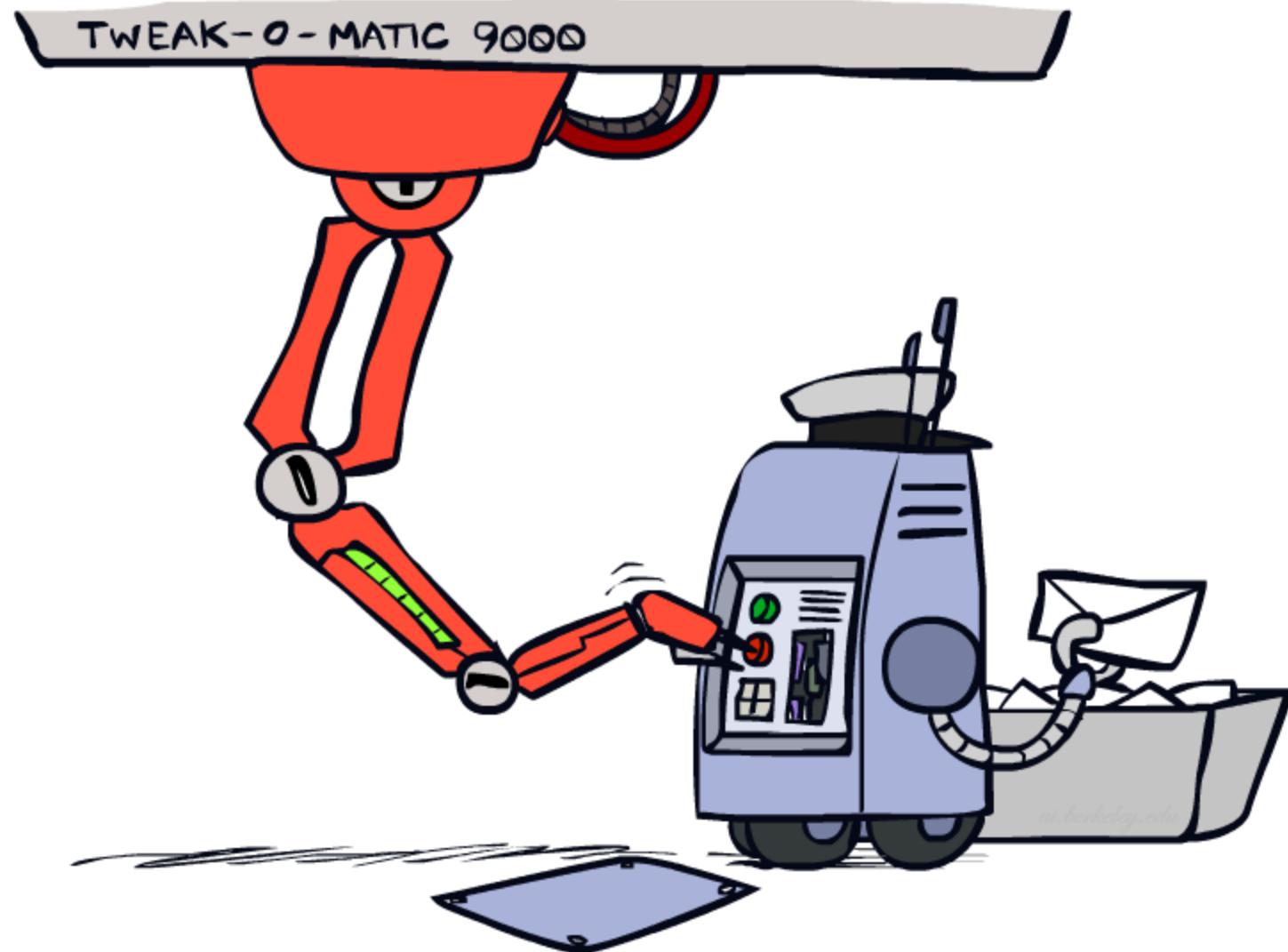
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
	:	26.9
money	:	26.5
...		



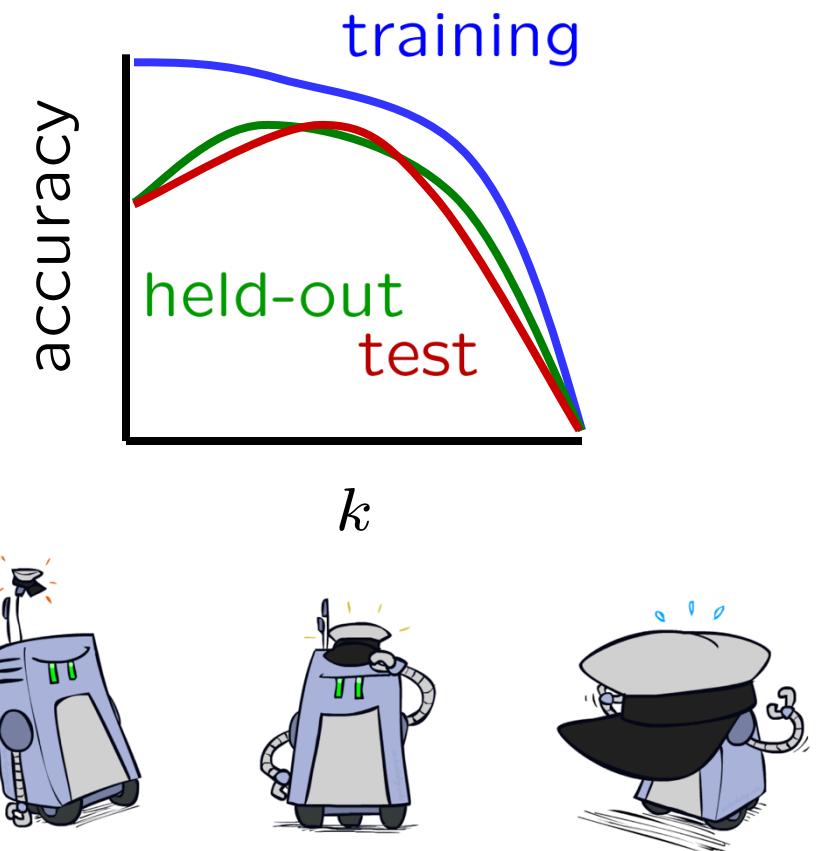
Do these make more sense?

Tuning



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 ,
- 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

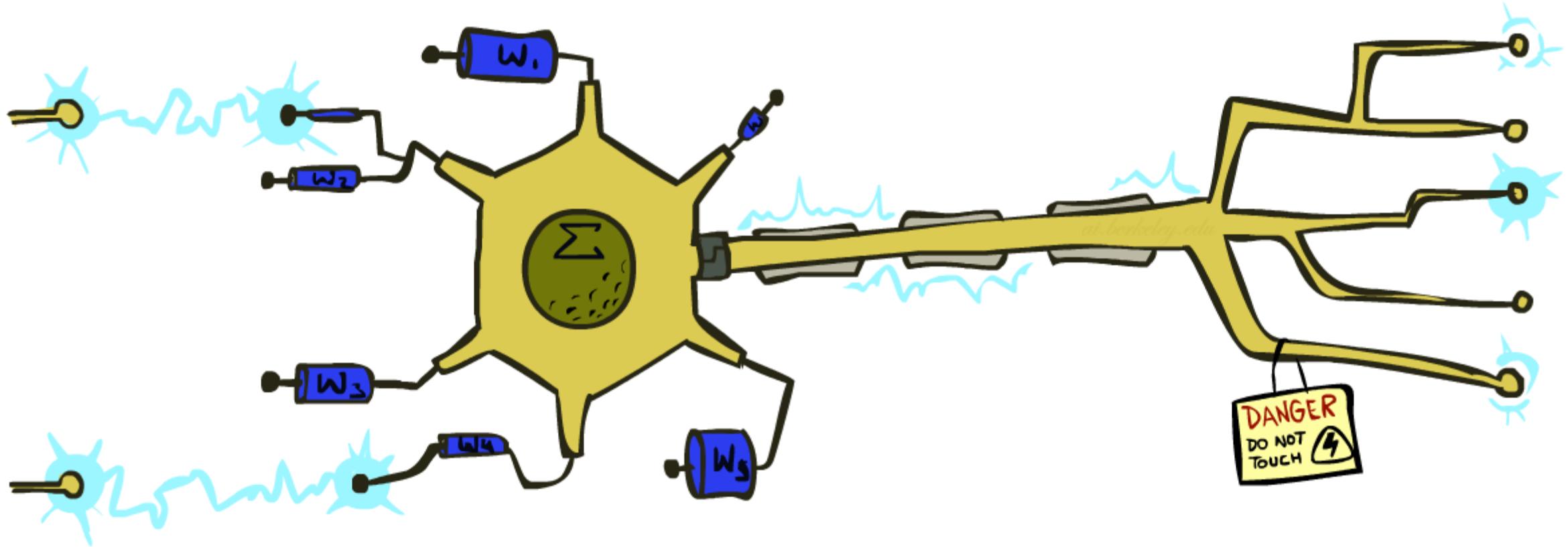


Baselines

- First step: get a **baseline**
 - Baselines are very simple “straw man” procedures
 - Help determine how hard the task is
 - Help know what a “good” accuracy is
- Weak baseline: most frequent label classifier
 - Gives all test instances whatever label was most common in the training set
 - E.g. for spam filtering, might label everything as ham
 - Accuracy might be very high if the problem is skewed
 - E.g. if calling everything “ham” gets 66%, then a classifier that gets 70% isn’t very good...
- For real research, usually use previous work as a (strong) baseline

CS 188: Artificial Intelligence

Perceptrons



Instructors: Pieter Abbeel & Anca Dragan --- University of California, Berkeley

Error-Driven Classification



Errors, and What to Do

■ Examples of errors

Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just \$99.99* - the regular list price is \$499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

. . . To receive your \$30 Amazon.com promotional certificate, click through to

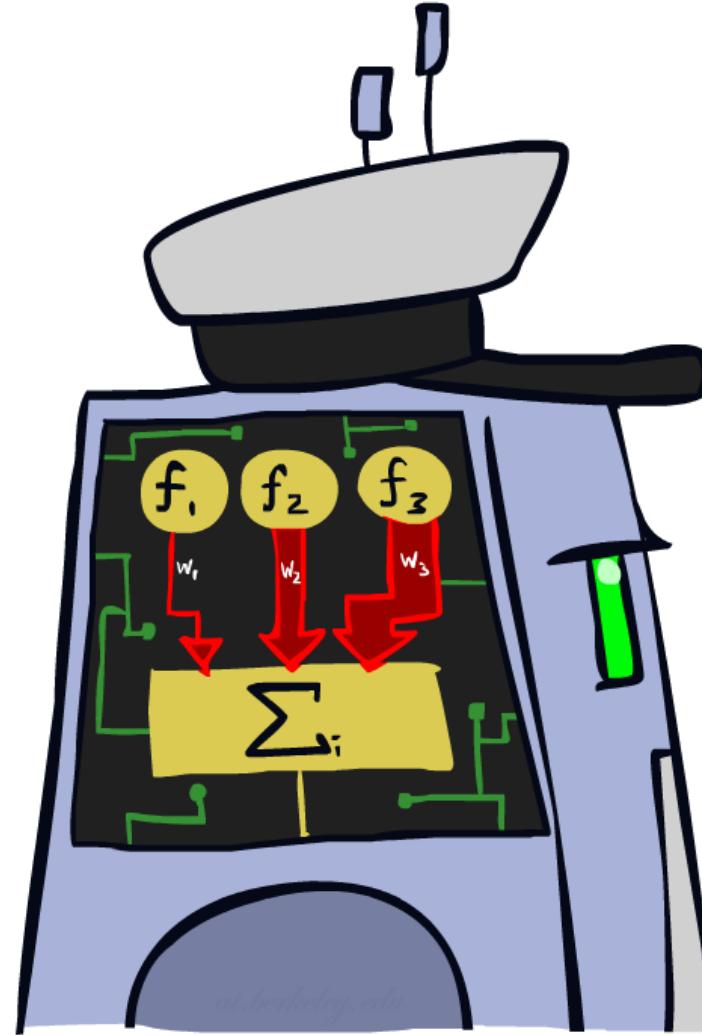
<http://www.amazon.com/apparel>

and see the prominent link for the \$30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .

What to Do About Errors

- Problem: there's still spam in your inbox
- Need more **features** – words aren't enough!
 - Have you emailed the sender before?
 - Have 1M other people just gotten the same email?
 - Is the sending information consistent?
 - Is the email in ALL CAPS?
 - Do inline URLs point where they say they point?
 - Does the email address you by (your) name?
- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)

Linear Classifiers



Feature Vectors

x

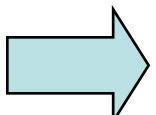
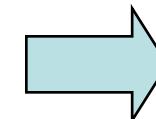
```
Hello,  
  
Do you want free printr  
cartridges? Why pay more  
when you can get them  
ABSOLUTELY FREE! Just
```

$f(x)$

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

y

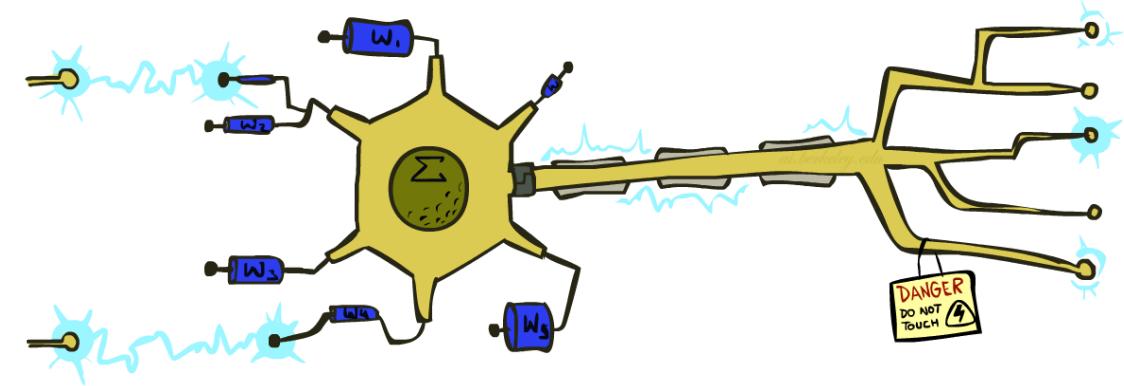
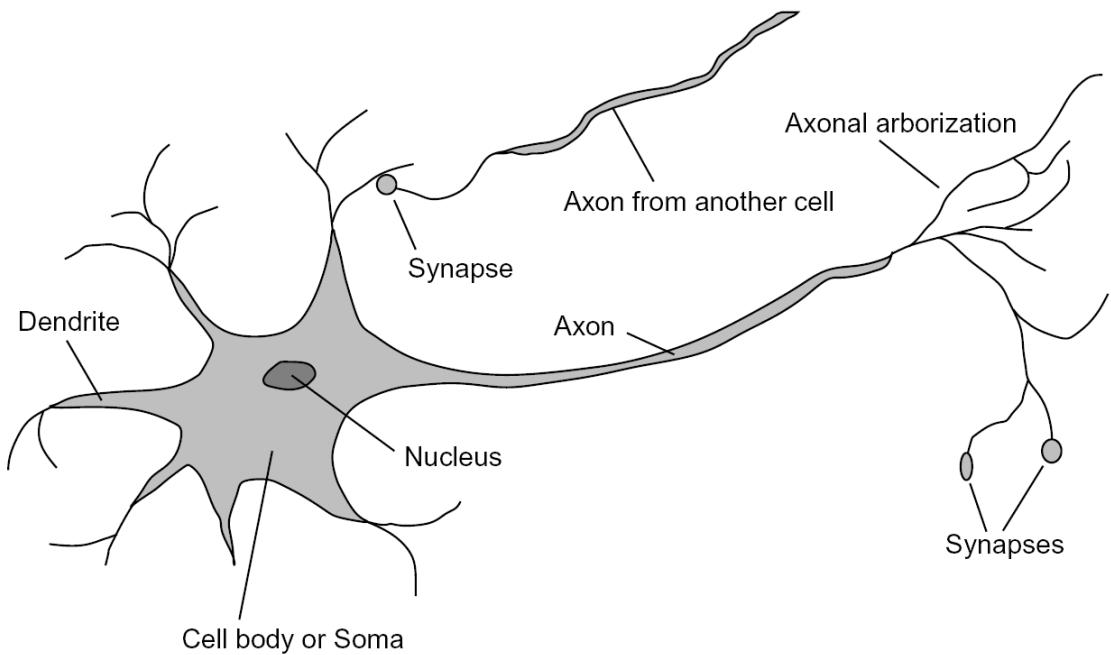
SPAM
or
+


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


“2”

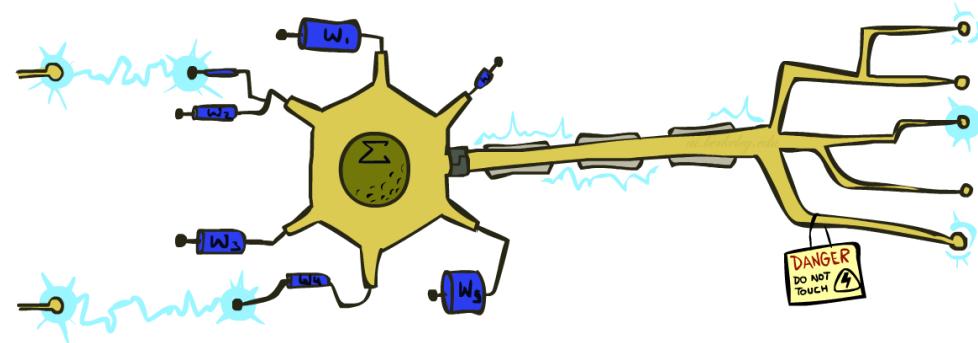
Some (Simplified) Biology

- Very loose inspiration: human neurons



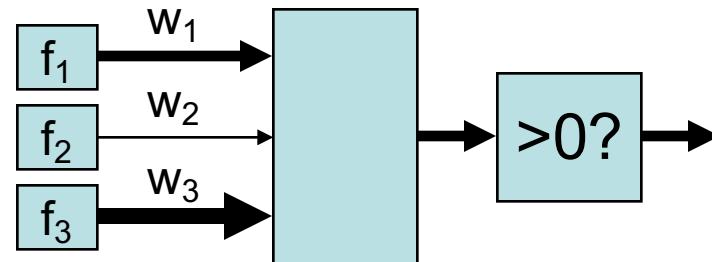
Linear Classifiers

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



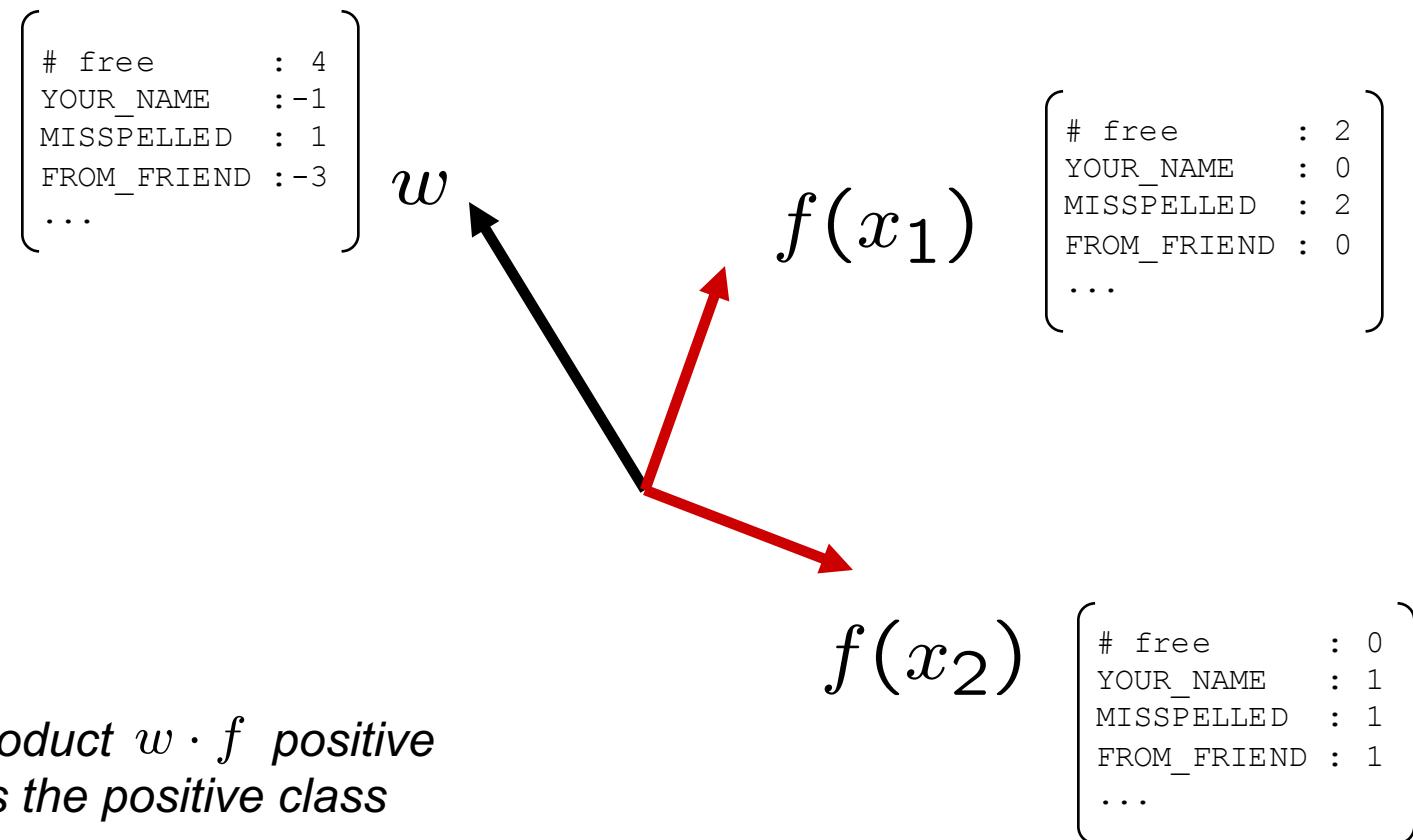
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

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

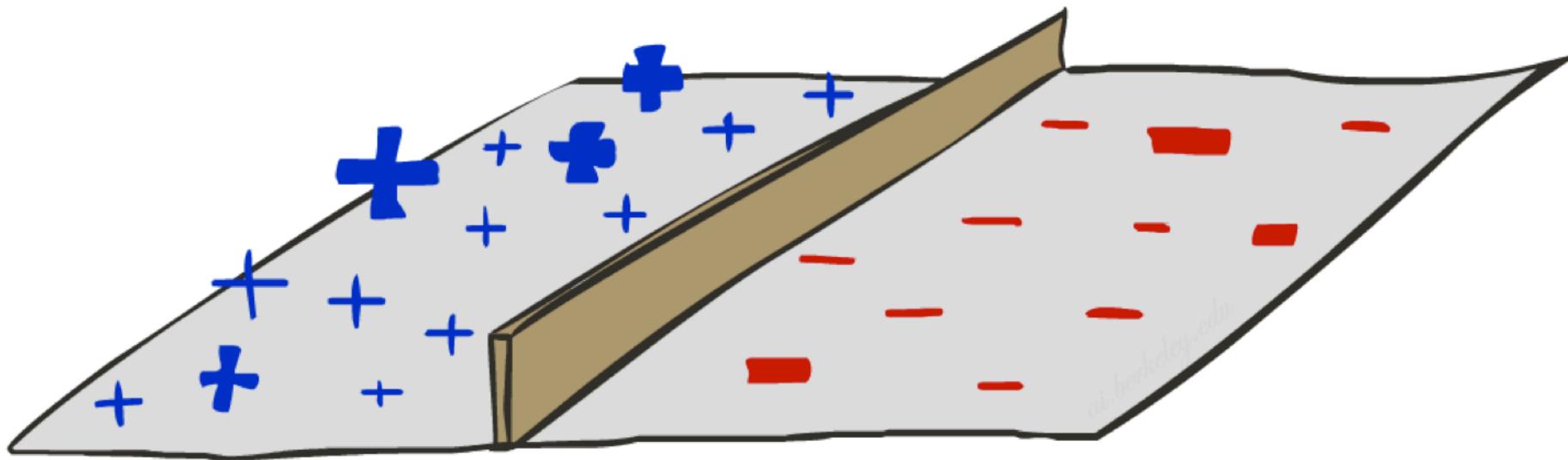


Weights

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



Decision Rules

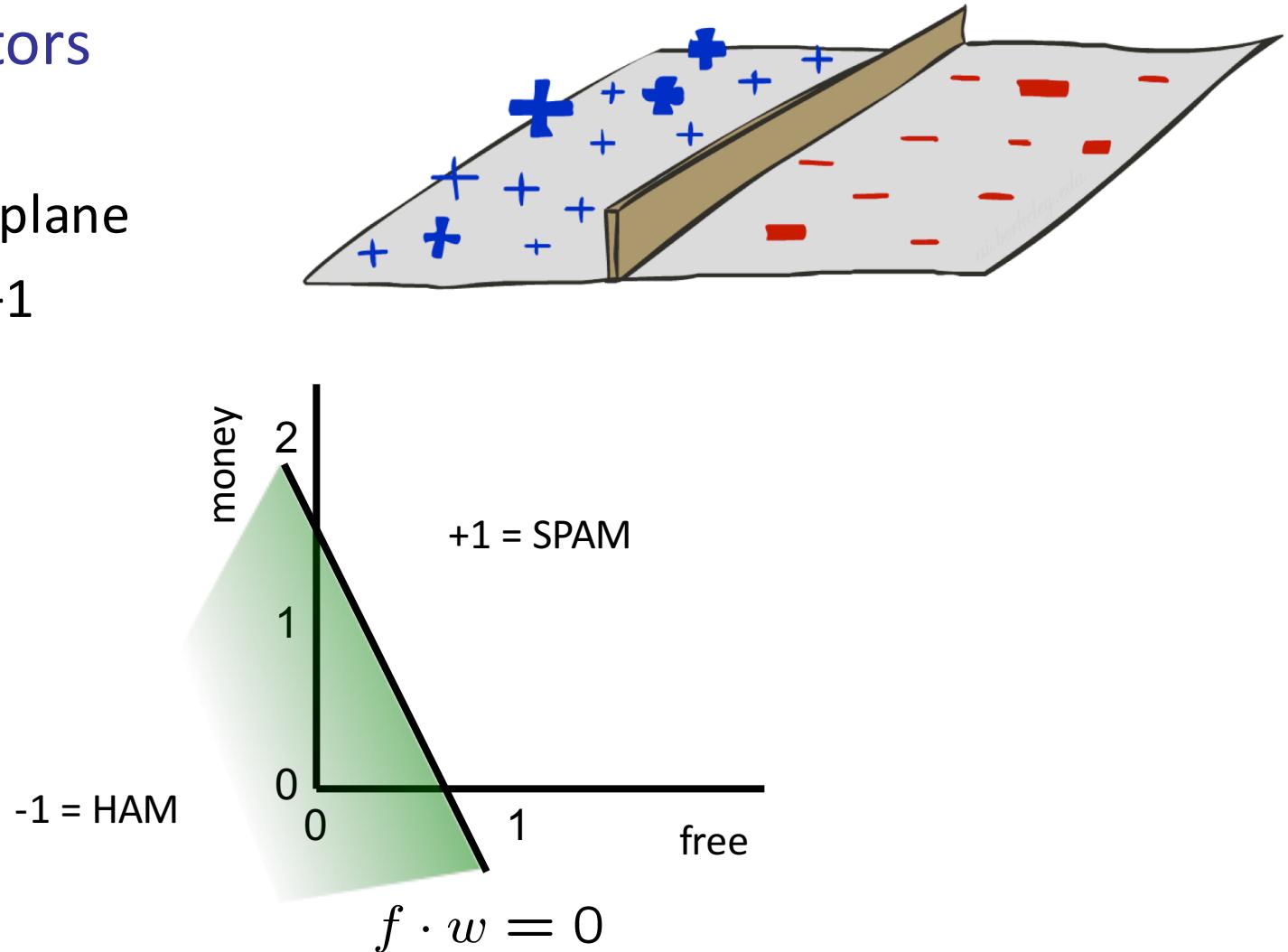


Binary Decision Rule

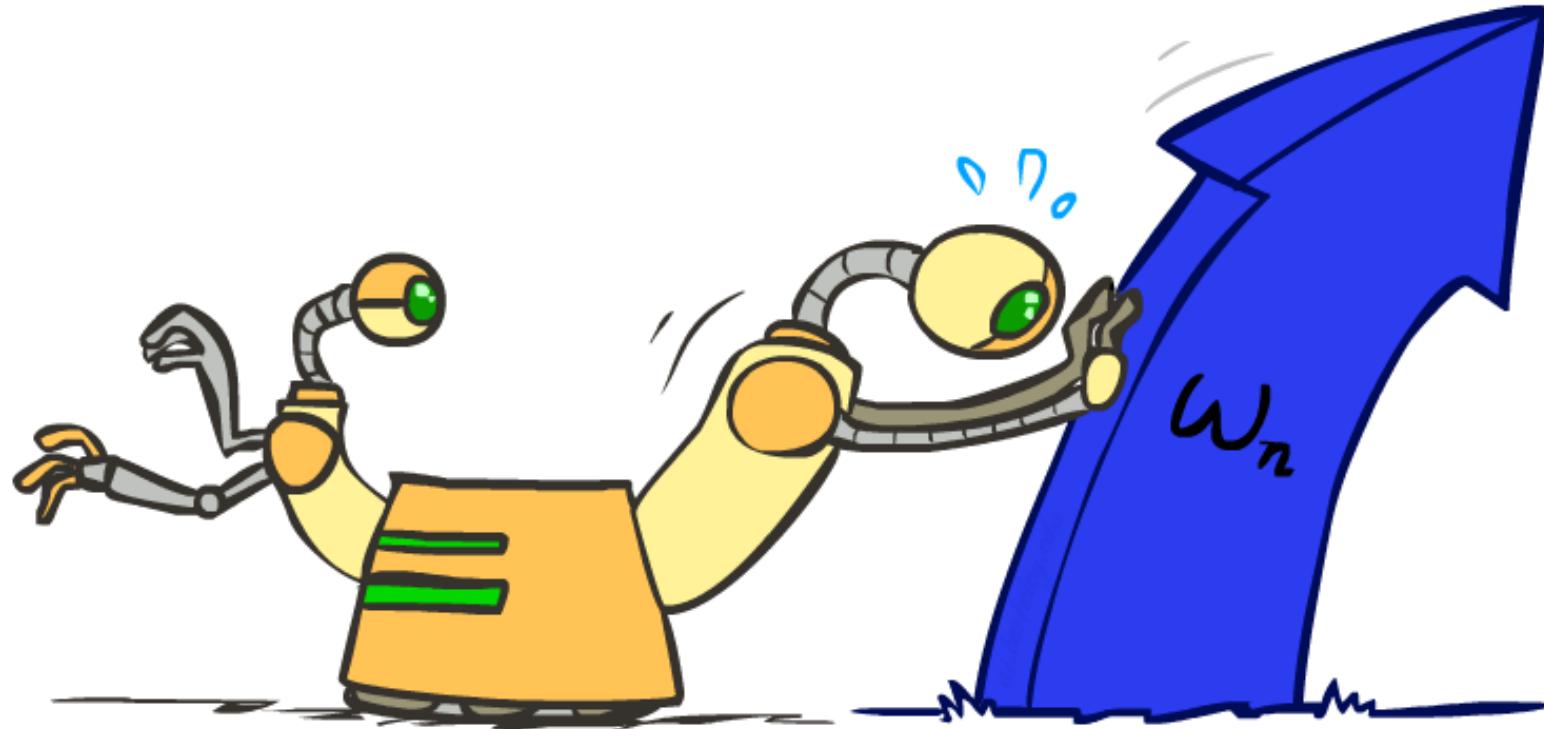
- In the space of feature vectors
 - Examples are points
 - Any weight vector is a hyperplane
 - One side corresponds to $Y=+1$
 - Other corresponds to $Y=-1$

w

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

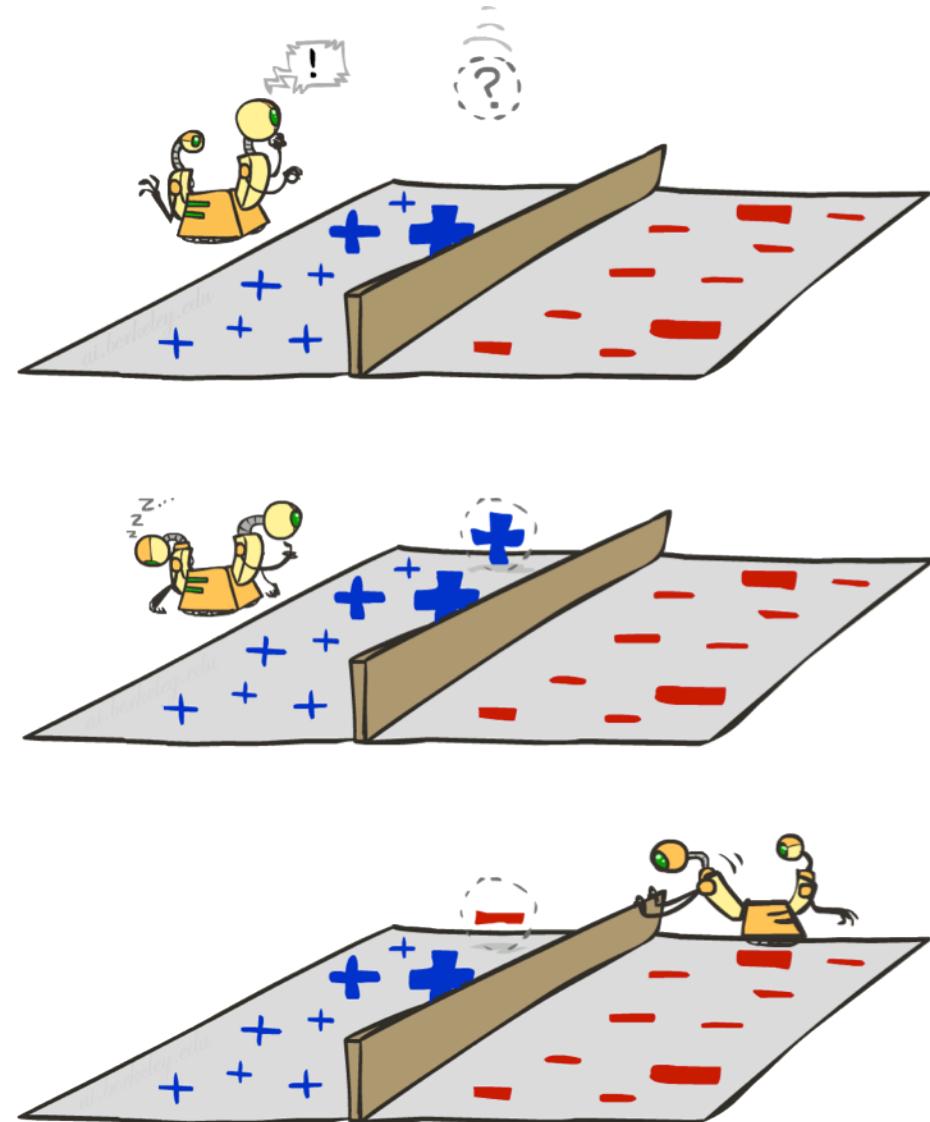


Weight Updates



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



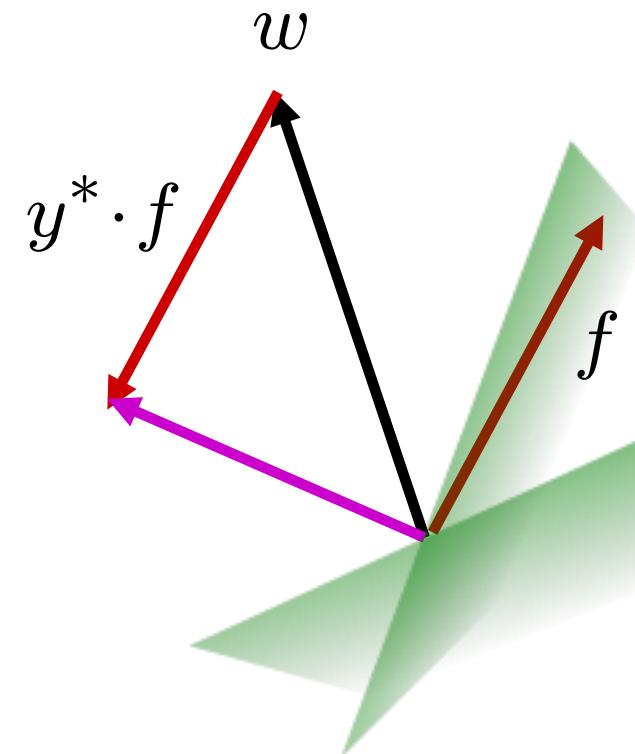
Learning: Binary Perceptron

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

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

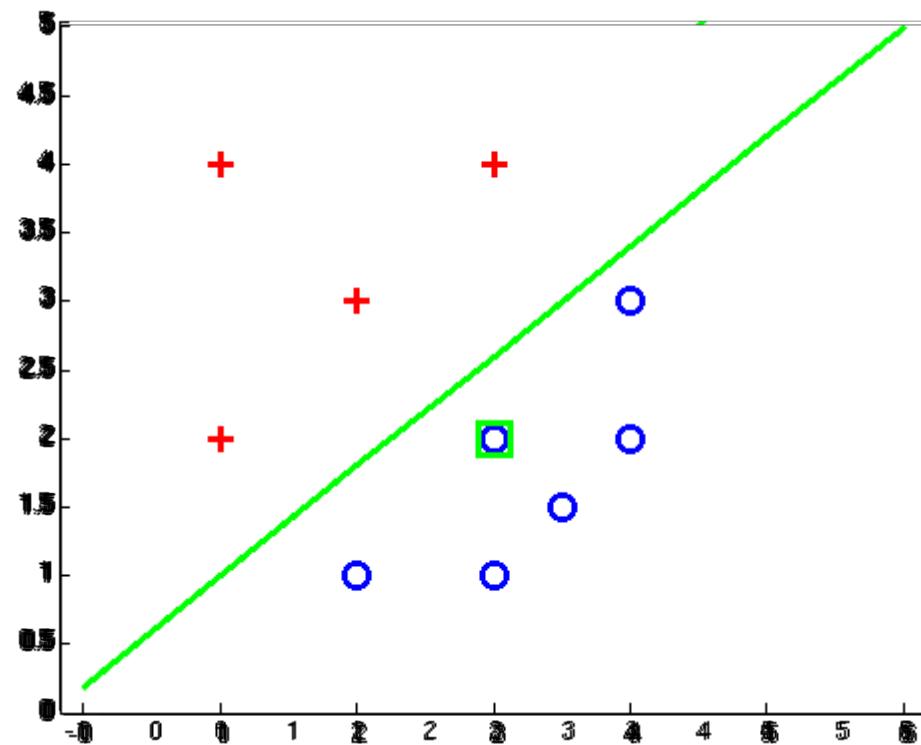
- If correct (i.e., $y=y^*$), no change!
- If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y^* is -1.

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



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

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

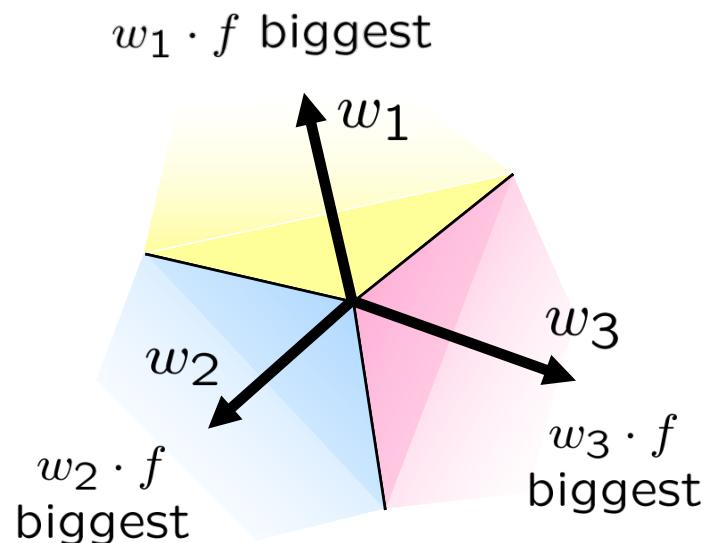
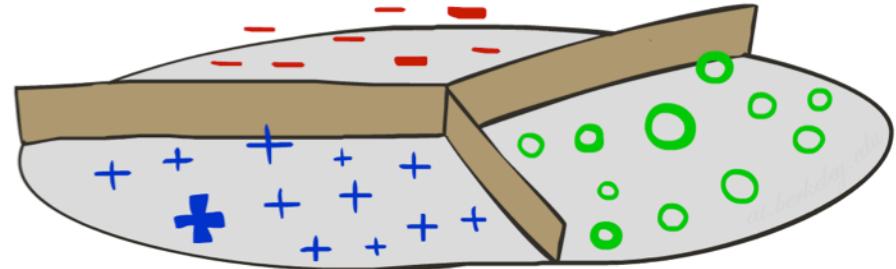
$$w_y$$

- 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

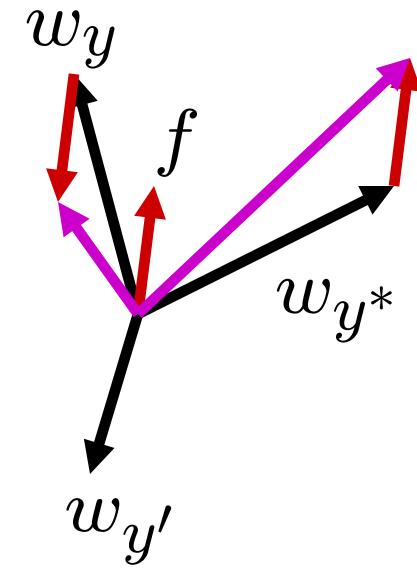
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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”

“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}

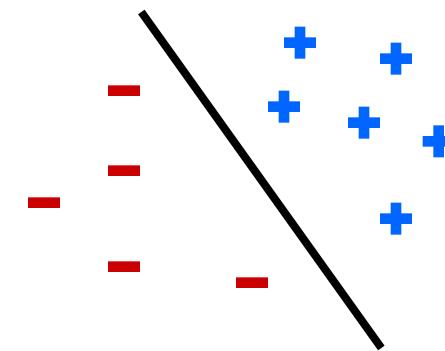
BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0
...		

Properties of Perceptrons

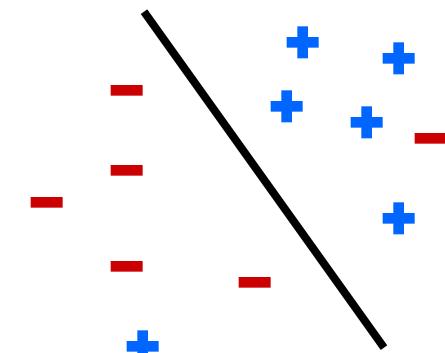
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

Separable

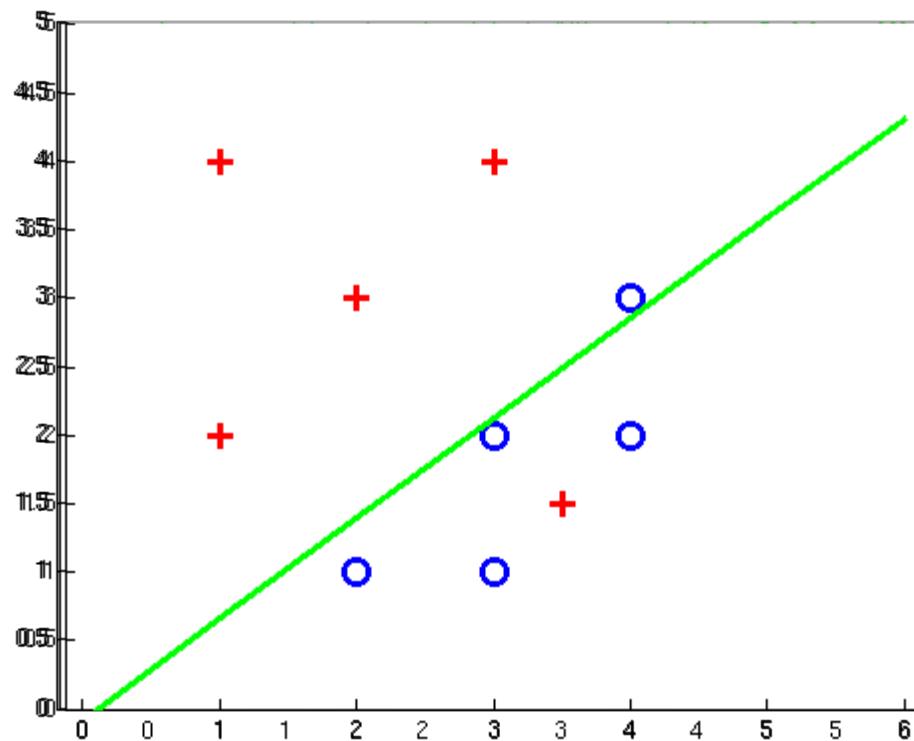


Non-Separable

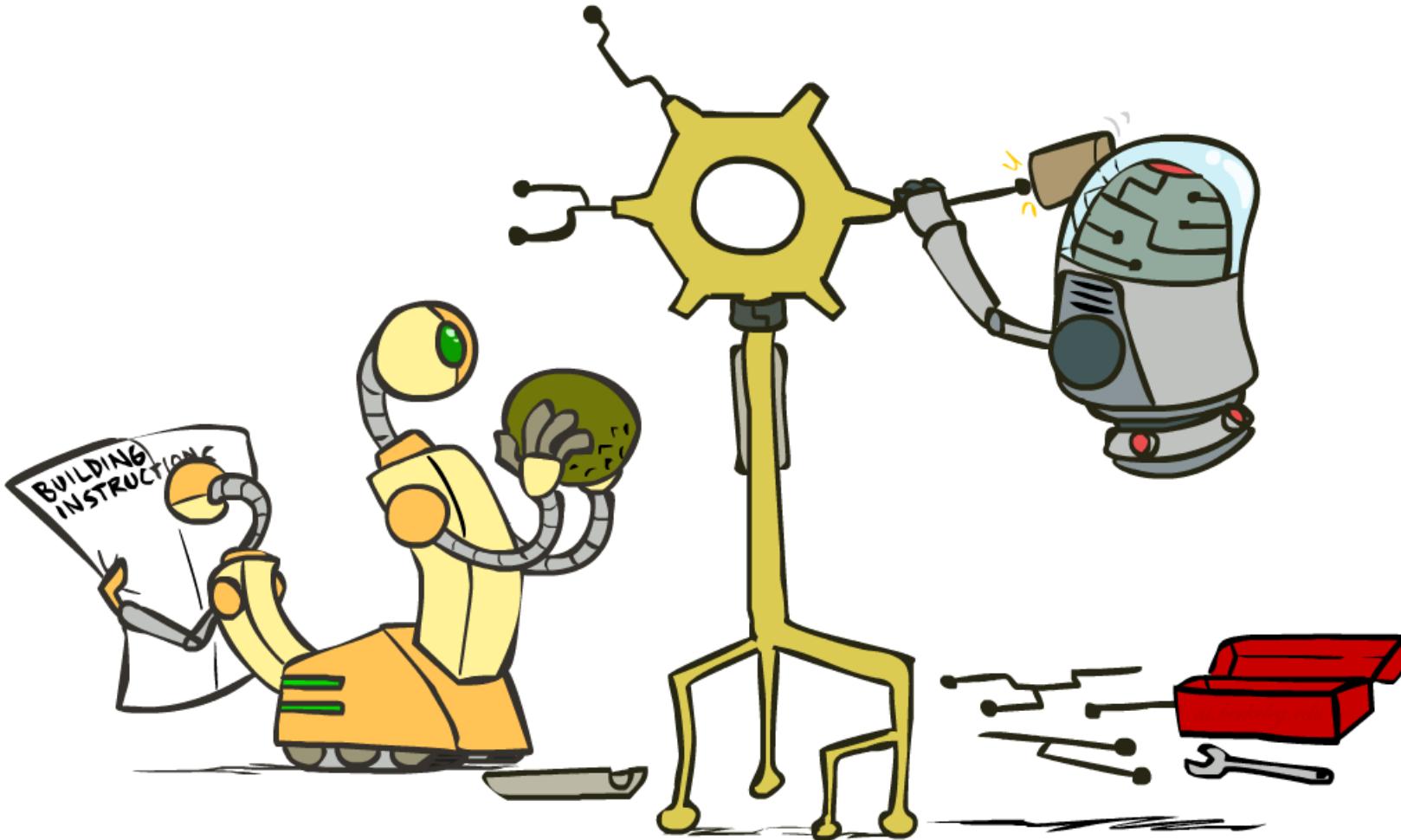


Examples: Perceptron

- Non-Separable Case

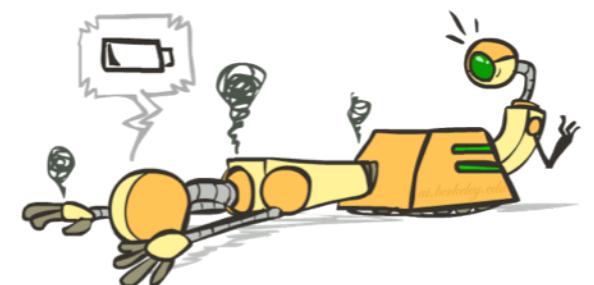
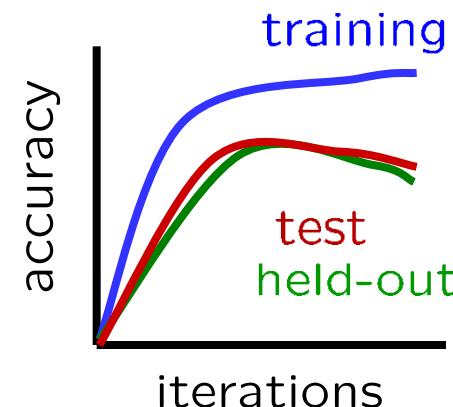
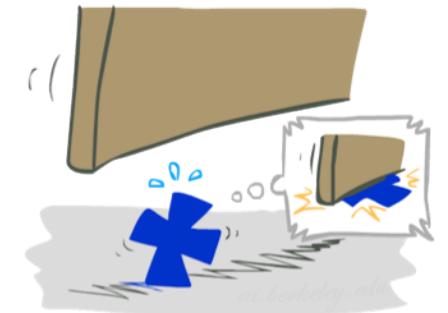
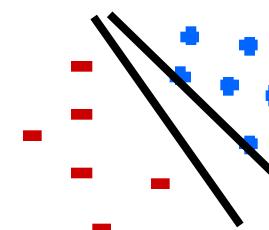
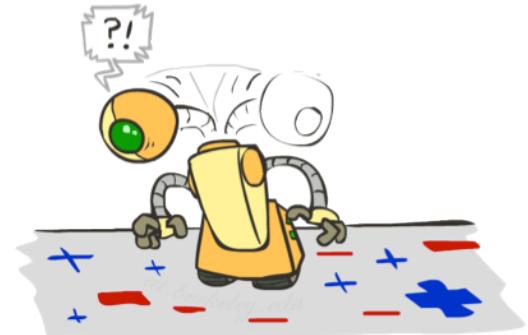
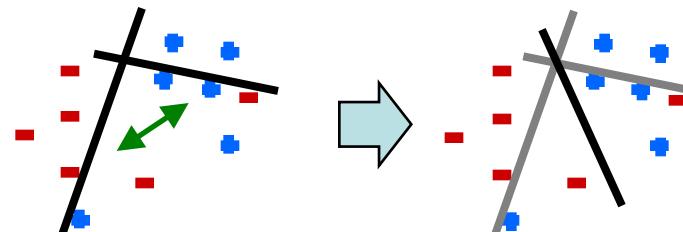


Improving the Perceptron



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



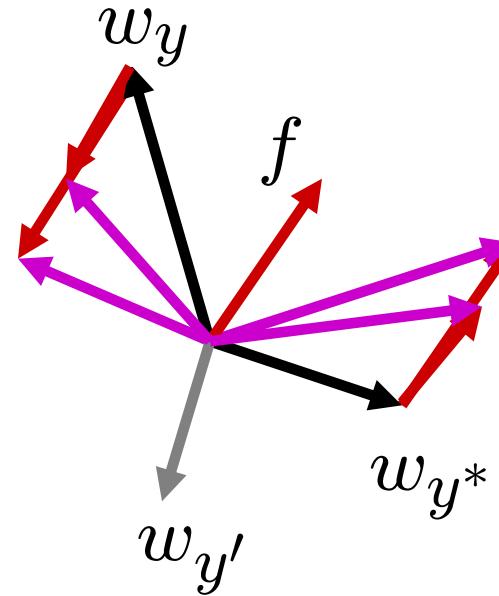
Fixing the Perceptron

- Idea: adjust the weight update to mitigate these effects
- MIRA*: choose an update size that fixes the current mistake...
- ... but, minimizes the change to w

$$\min_w \frac{1}{2} \sum_y \|w_y - w'_y\|^2$$

$$w_{y^*} \cdot f(x) \geq w_y \cdot f(x) + 1$$

- The +1 helps to generalize



Guessed y instead of y^* on example x with features $f(x)$

$$w_y = w'_y - \tau f(x)$$

$$w_{y^*} = w'_{y^*} + \tau f(x)$$

* Margin Infused Relaxed Algorithm

Minimum Correcting Update

$$\min_w \frac{1}{2} \sum_y \|w_y - w'_y\|^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$



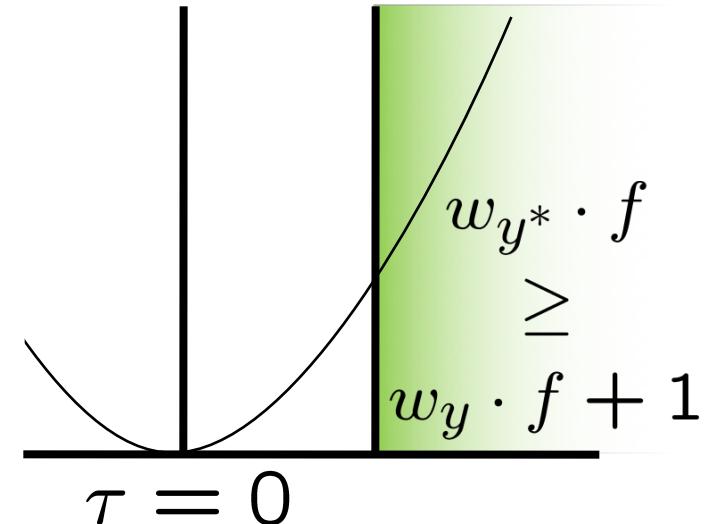
$$\min_{\tau} \|\tau f\|^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$



$$(w'_{y^*} + \tau f) \cdot f = (w'_y - \tau f) \cdot f + 1$$

$$\tau = \frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}$$

$$w_y = w'_y - \tau f(x)$$
$$w_{y^*} = w'_{y^*} + \tau f(x)$$



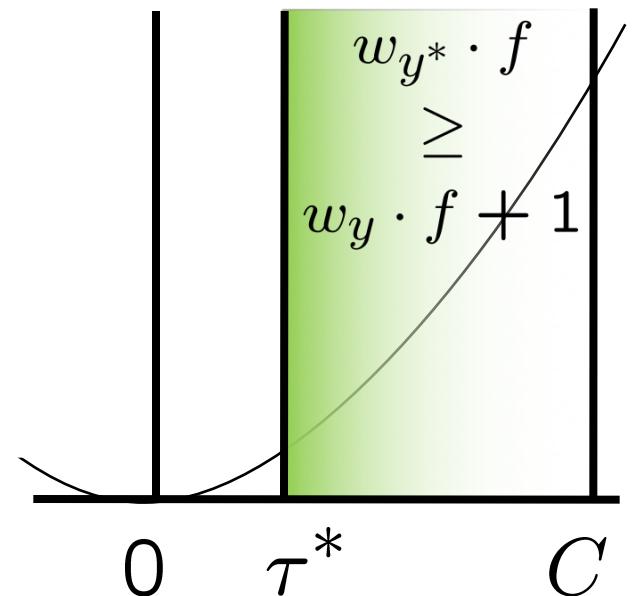
min not =0, or would not have made an error, so min will be where equality holds

Maximum Step Size

- In practice, it's also bad to make updates that are too large
 - Example may be labeled incorrectly
 - You may not have enough features
 - Solution: cap the maximum possible value of τ with some constant C

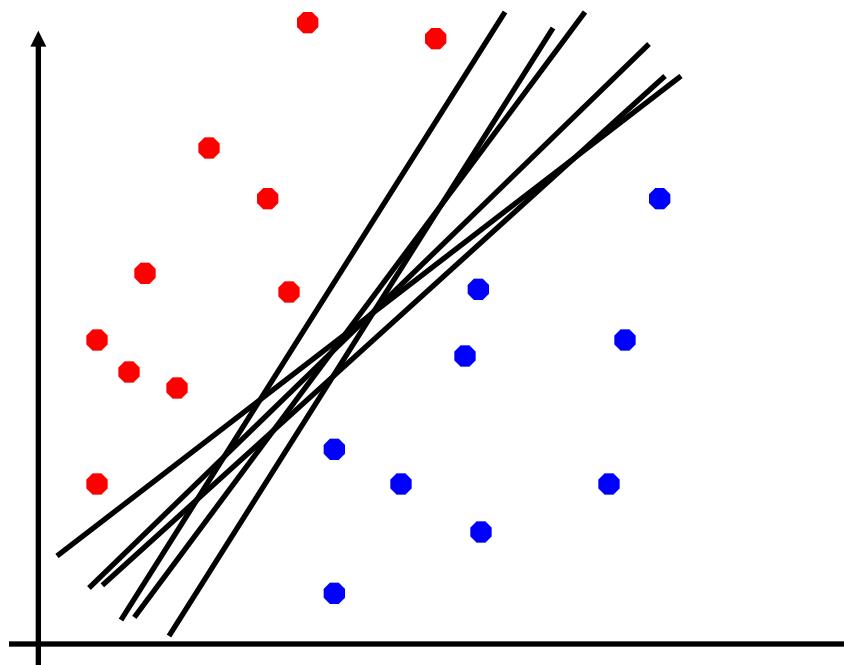
$$\tau^* = \min \left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C \right)$$

- Corresponds to an optimization that assumes non-separable data
- Usually converges faster than perceptron
- Usually better, especially on noisy data



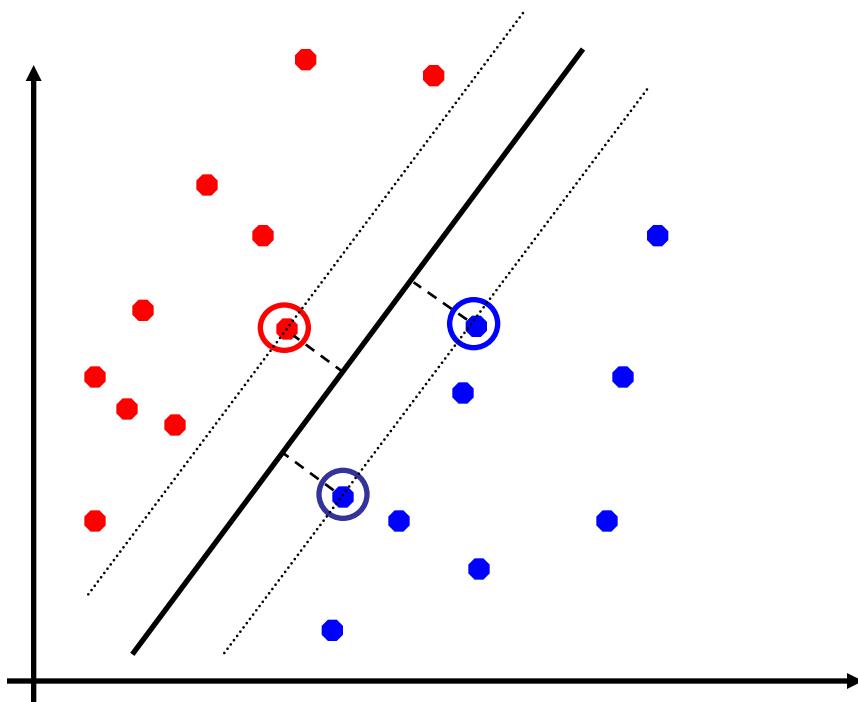
Linear Separators

- Which of these linear separators is optimal?



Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with max margin
- Basically, SVMs are MIRA where you optimize over all examples at once



MIRA

$$\min_w \frac{1}{2} \|w - w'\|^2$$

$$w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

SVM

$$\min_w \frac{1}{2} \|w\|^2$$

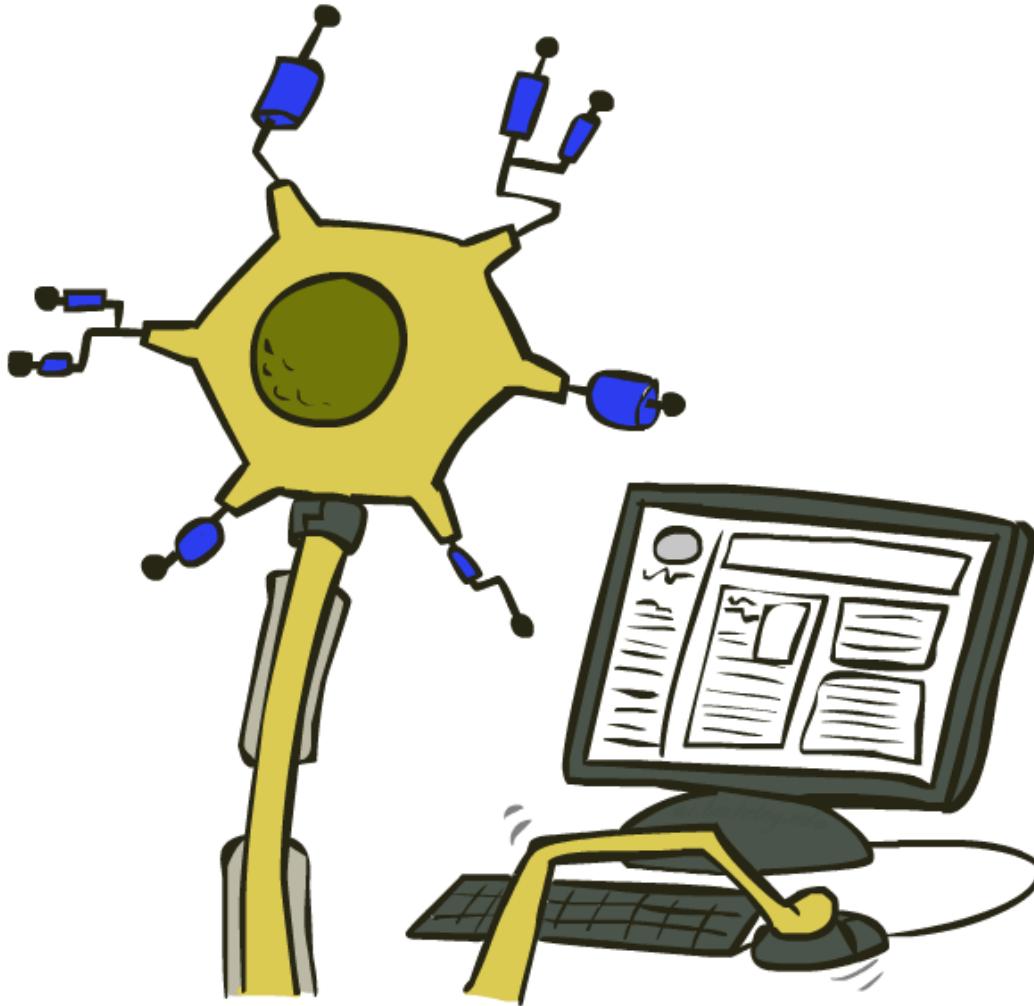
$$\forall i, y \quad w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

Classification: Comparison

- Naïve Bayes
 - Builds a model training data
 - Gives prediction probabilities
 - Strong assumptions about feature independence
 - One pass through data (counting)

- Perceptrons / MIRA:
 - Makes less assumptions about data
 - Mistake-driven learning
 - Multiple passes through data (prediction)
 - Often more accurate

Web Search



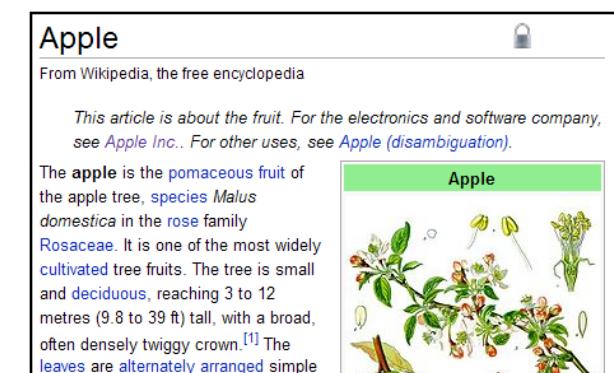
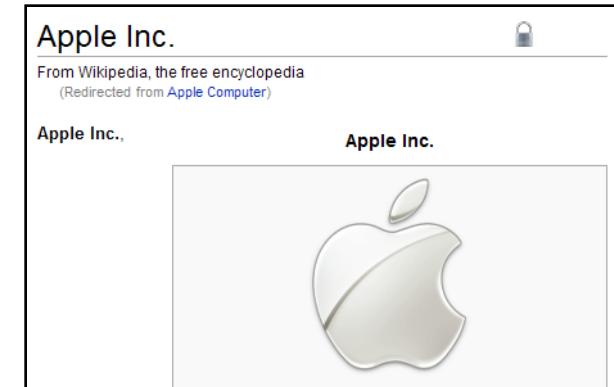
Extension: Web Search

- Information retrieval:

- Given information needs, produce information
- Includes, e.g. web search, question answering, and classic IR

- Web search: not exactly classification, but rather ranking

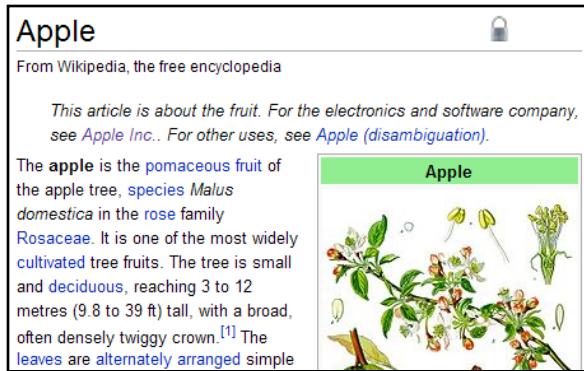
x = “Apple Computers”



Feature-Based Ranking

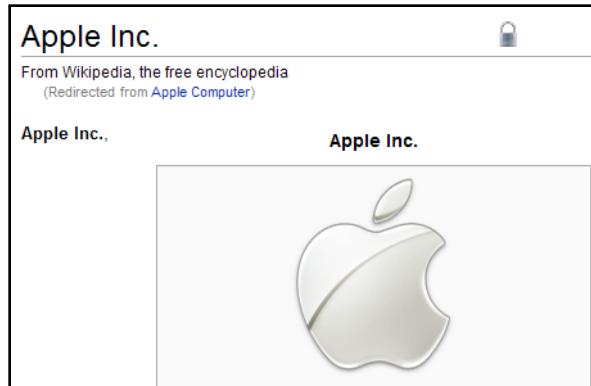
$x = \text{"Apple Computer"}$

$f(x,$



) = [0.3 5 0 0 ...]

$f(x,$



) = [0.8 4 2 1 ...]

Perceptron for Ranking

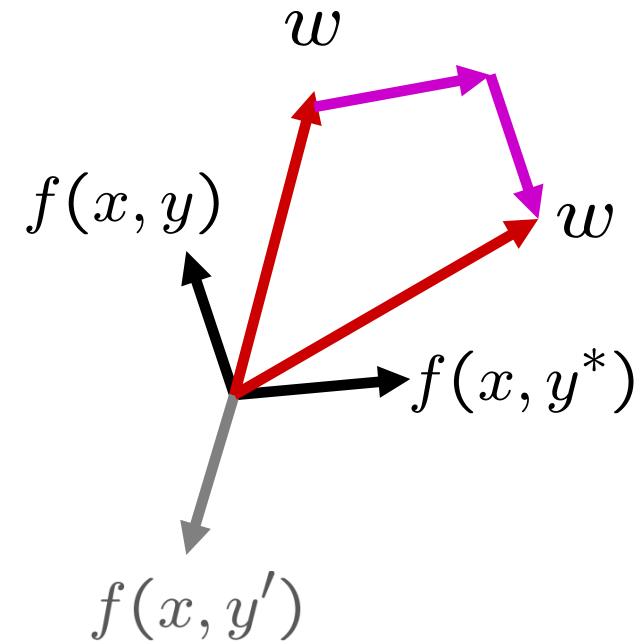
- Inputs x
- Candidates y
- Many feature vectors: $f(x, y)$
- One weight vector: w

- Prediction:

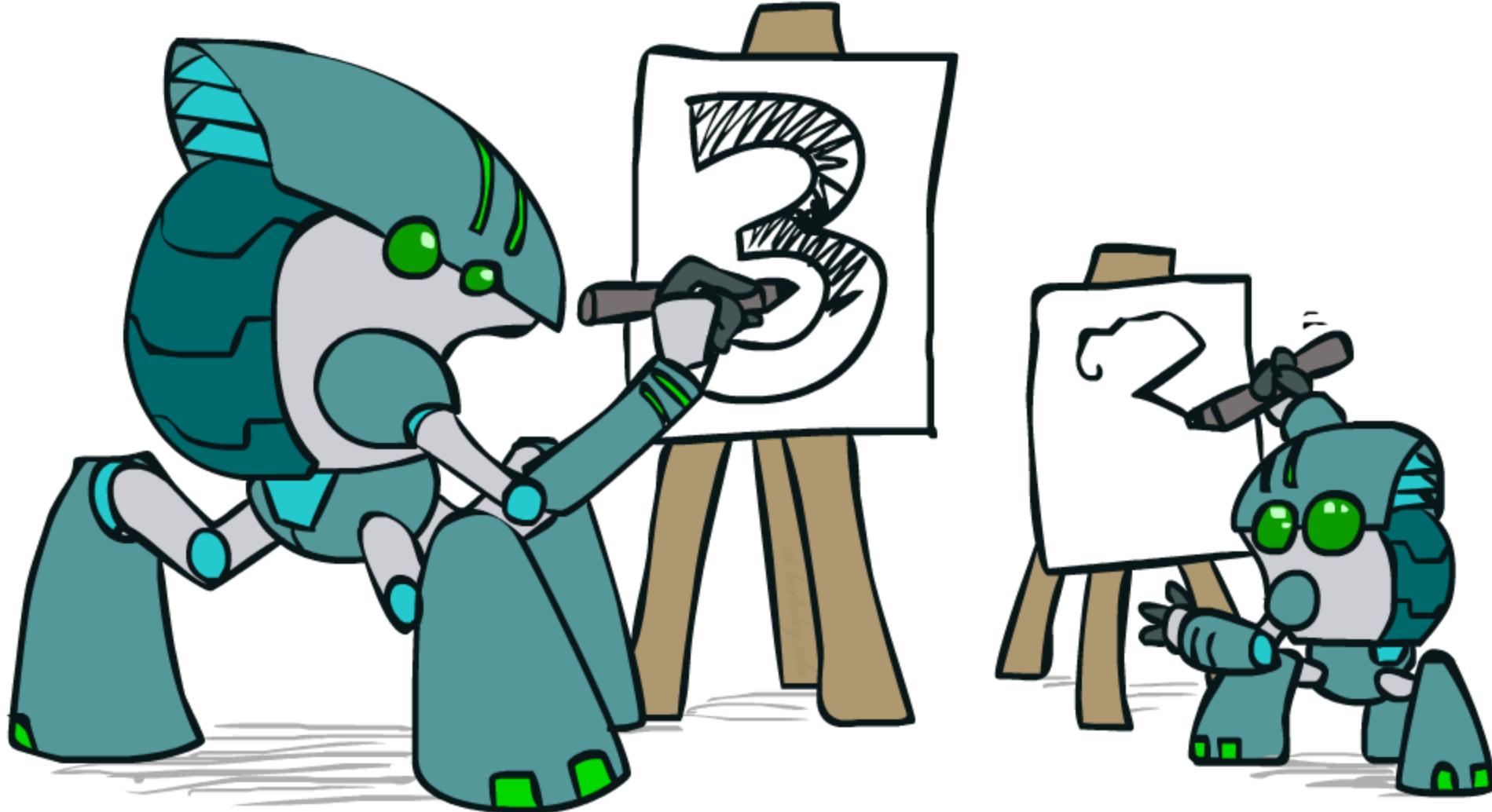
$$y = \arg \max_y w \cdot f(x, y)$$

- Update (if wrong):

$$w = w + f(x, y^*) - f(x, y)$$

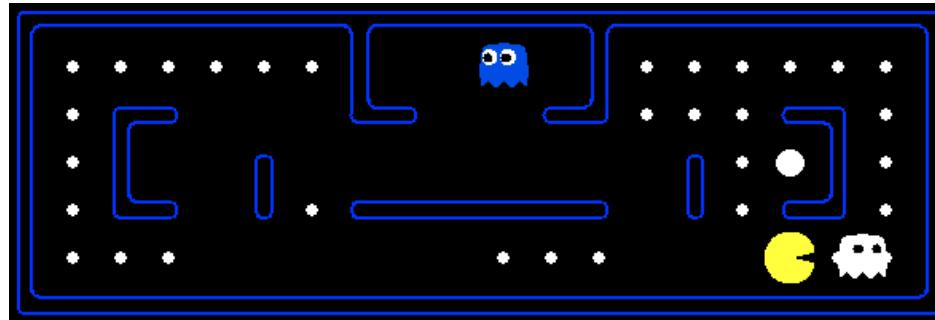


Apprenticeship



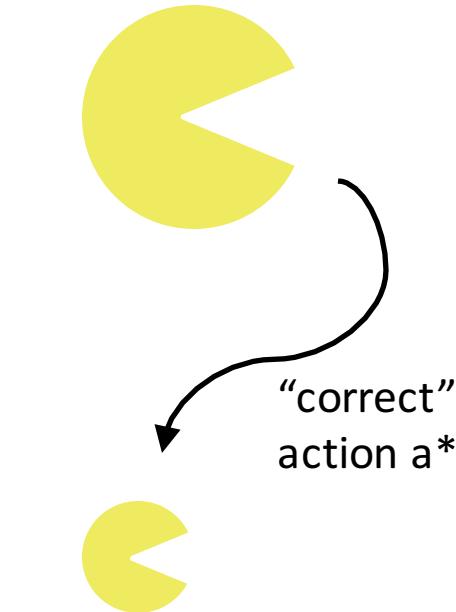
Pacman Apprenticeship!

- Examples are states s



- Candidates are pairs (s, a)
- “Correct” actions: those taken by expert
- Features defined over (s, a) pairs: $f(s, a)$
- Score of a q-state (s, a) given by:

$$w \cdot f(s, a)$$



$$\forall a \neq a^*,
w \cdot f(a^*) > w \cdot f(a)$$

- How is this VERY different from reinforcement learning?

Video of Demo Pacman Apprentice



Next: Kernels and Clustering
