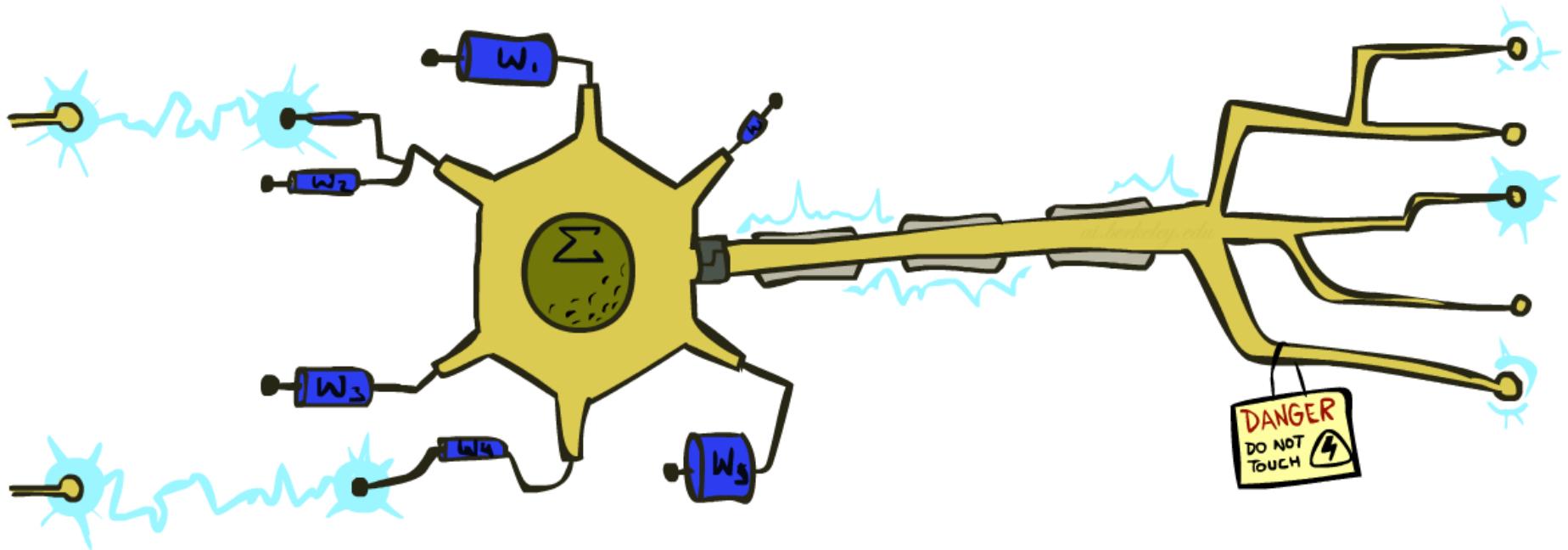


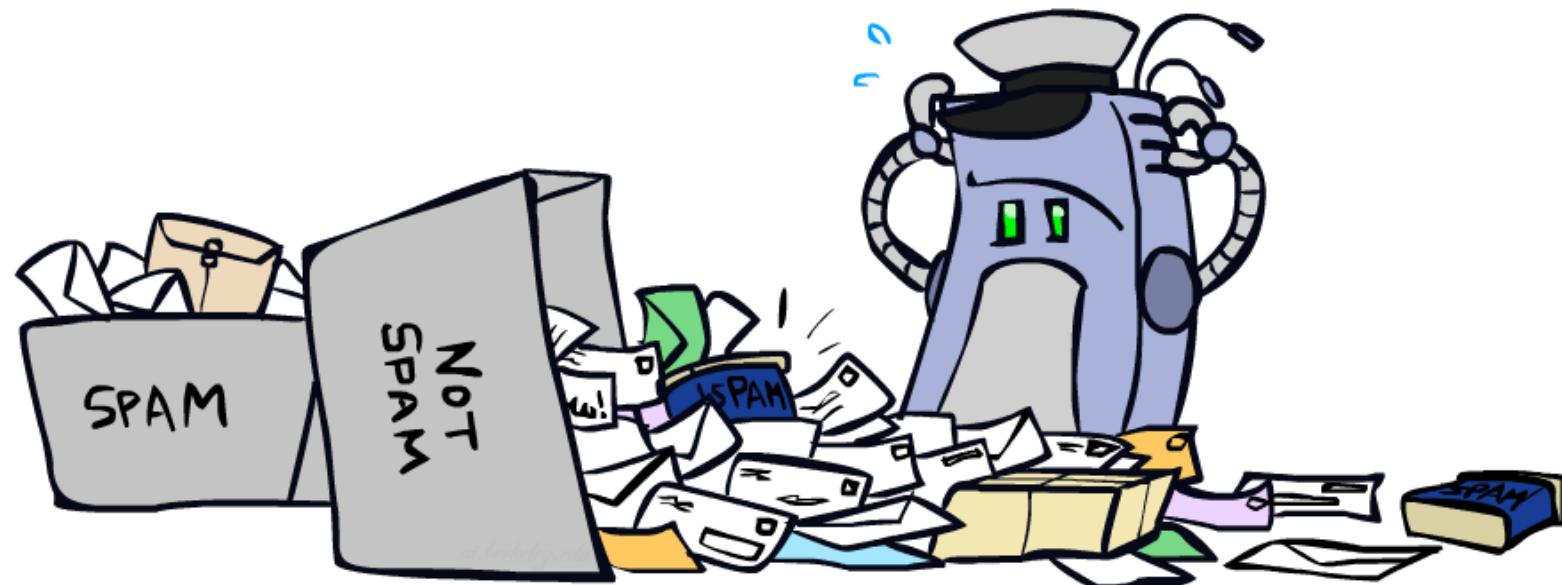
Perceptrons



Slides Courtesy of Dan Klein and Pieter Abbeel --- University of California, Berkeley

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at <http://ai.berkeley.edu>.]

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

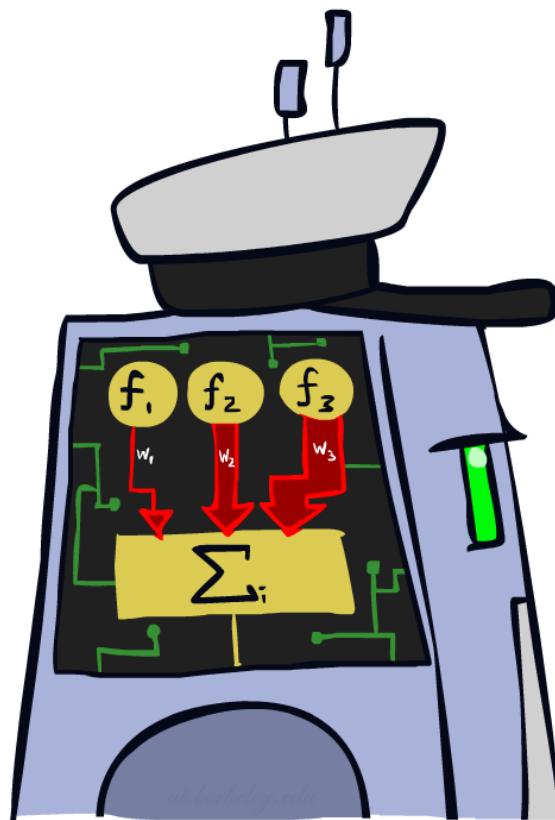
<http://www.amazon.com/apparel>

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



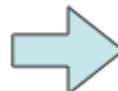
Feature Vectors

x

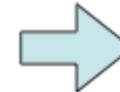
$f(x)$

y

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

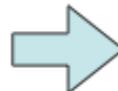


```
# free      : 2  
YOUR_NAME  : 0  
MISSPELLED : 2  
FROM_FRIEND : 0  
...
```

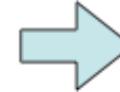


SPAM
or
+

2



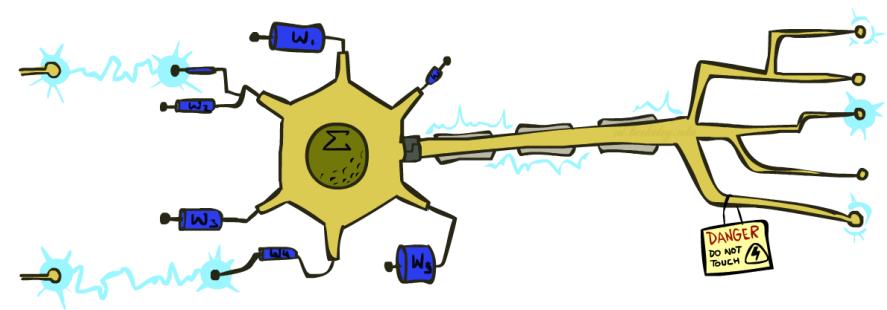
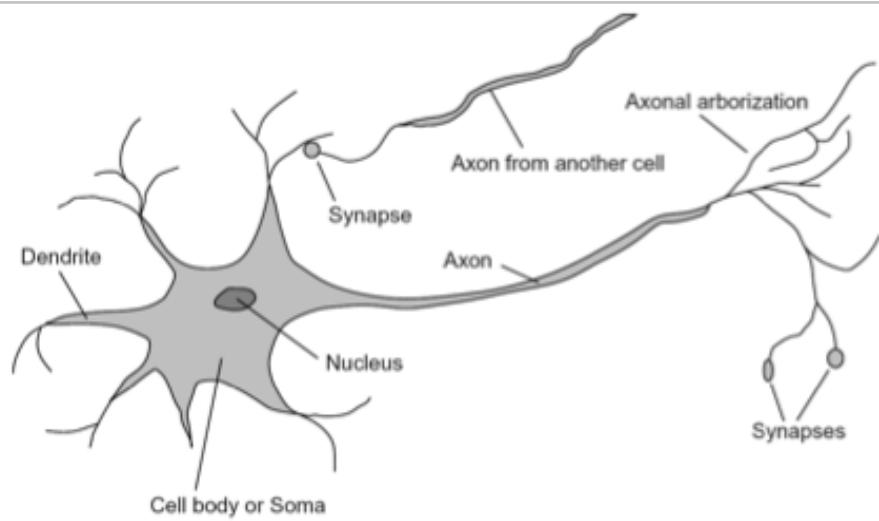
```
PIXEL-7,12  : 1  
PIXEL-7,13  : 0  
...  
NUM_LOOPS   : 1  
...
```



“2”

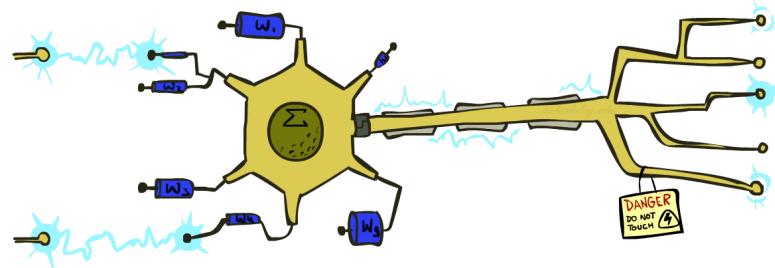
Some (Simplified) Biology

- Very loose inspiration: human neurons



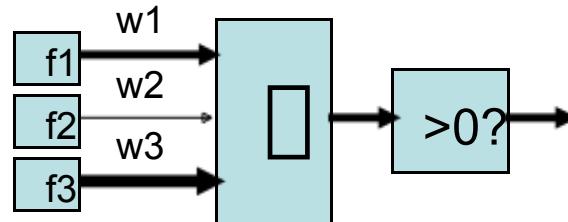
Linear Classifiers

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



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

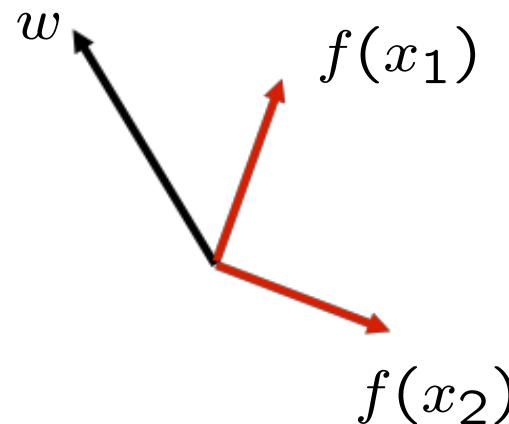
- If the activation is:
 - Positive, output +1
 - Negative, output -1



Weights

- 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  
...
```

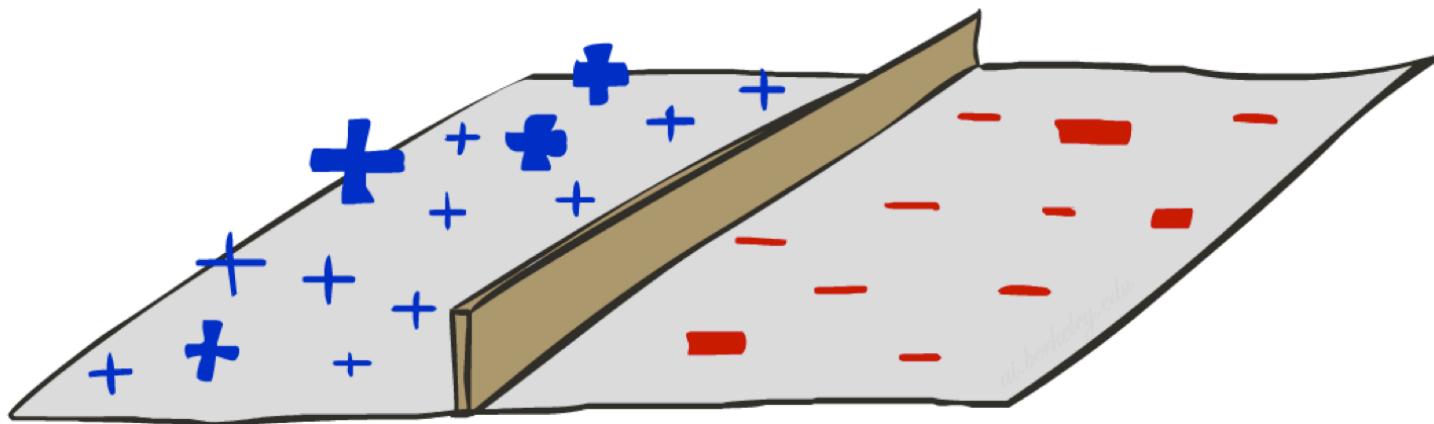


```
# free      : 2  
YOUR_NAME  : 0  
MISSPELLED : 2  
FROM_FRIEND : 0  
...
```

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

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

Decision Rules

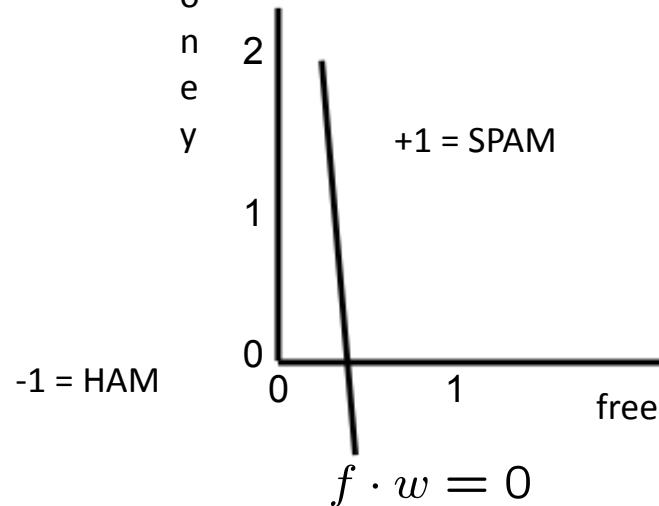
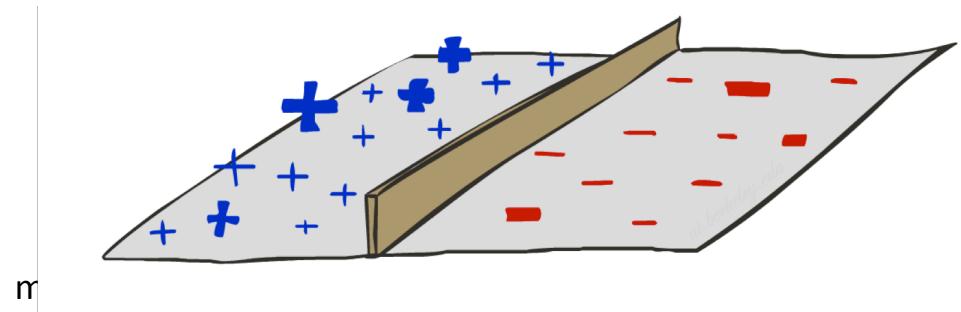


Binary Decision Rule

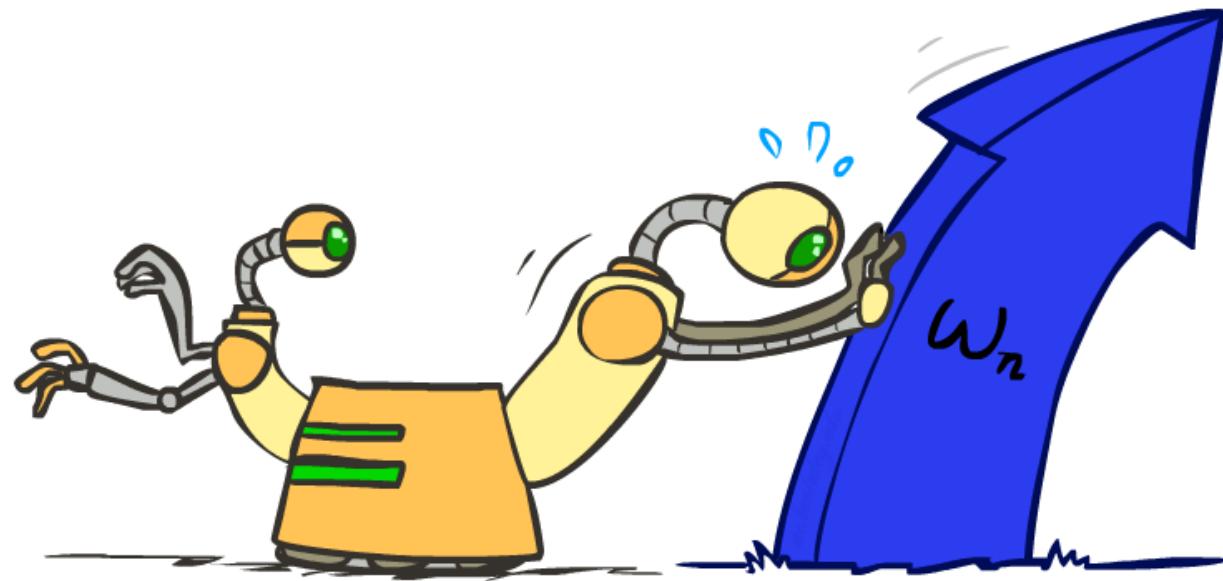
- In the space of feature vectors
 - Examples are points
 - Any weight vector is a hyperplane
 - One side corresponds to $Y=+1$
 - Other corresponds to $Y=-1$

w

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

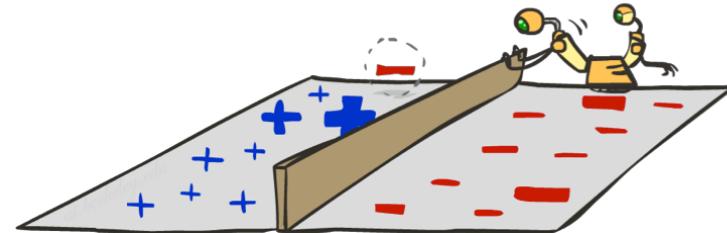
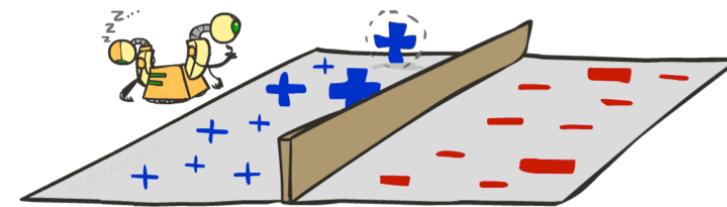
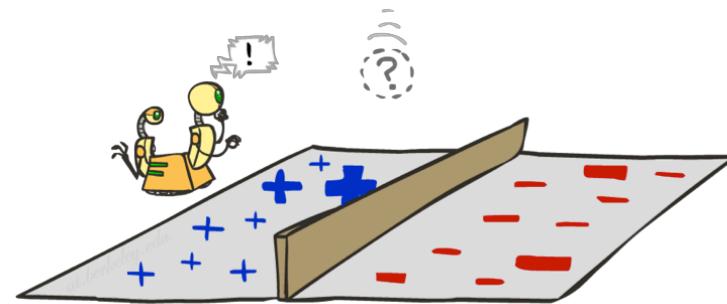


Weight Updates



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



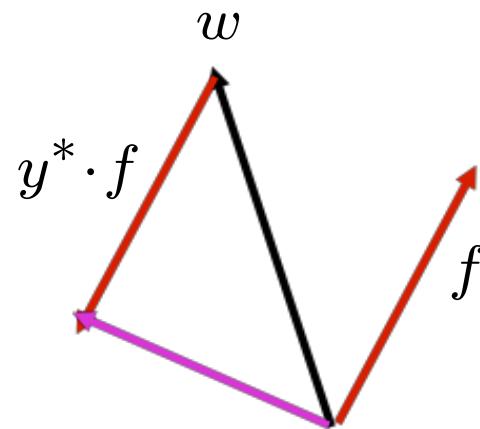
Learning: Binary Perceptron

- 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.

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

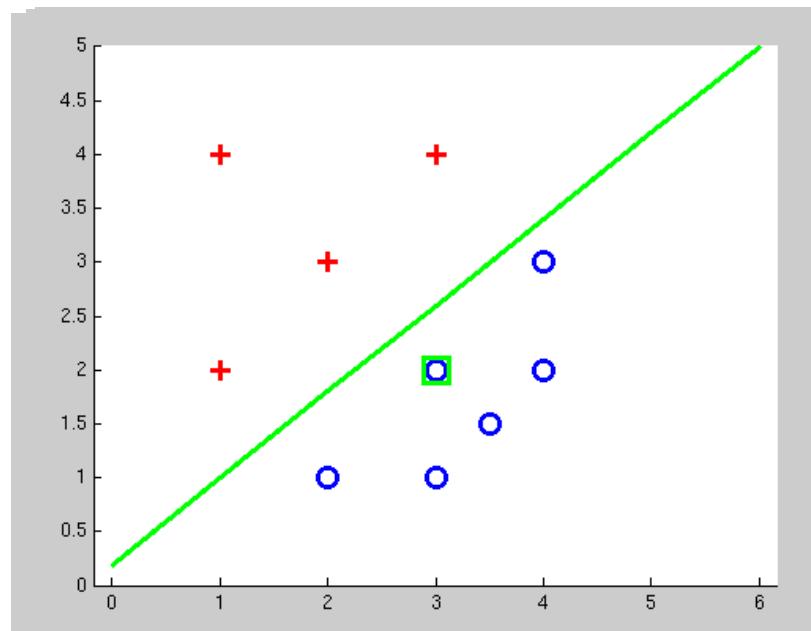


3Blue1Brown

- Need a refresher on linear algebra topics like vector addition and matrix multiplication?
- I recommend the wonderful YouTube series by Grant Sanderson. His channel is called 3Blue1Brown.
- Grant gives intuitive visual tutorials to a ton of math concepts
- Here is his “Essence of Linear Algebra” series:
 - [https://www.youtube.com/playlist?
list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab](https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab)

Examples: Perceptron

- Separable Case



Multiclass Decision Rule

- If we have multiple classes:

- A weight vector for each class:

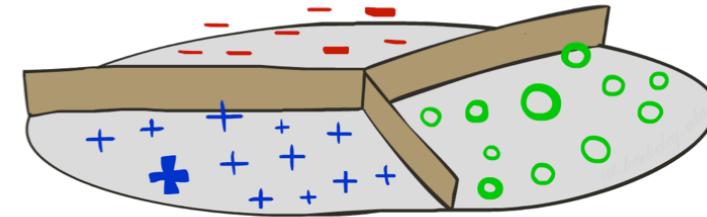
$$w_y$$

- Score (activation) of a class y :

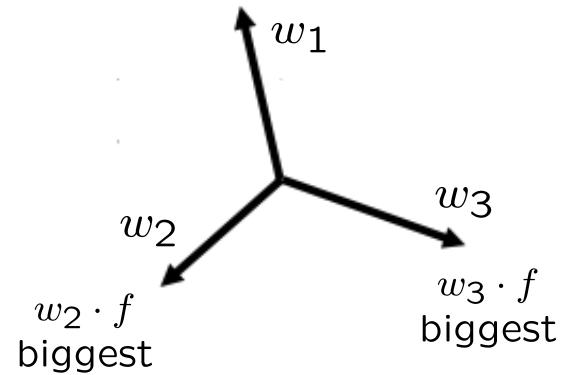
$$w_y \cdot f(x)$$

- Prediction highest score wins

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



$w_1 \cdot f$ biggest



Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

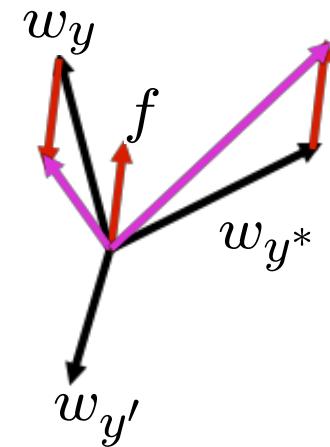
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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}

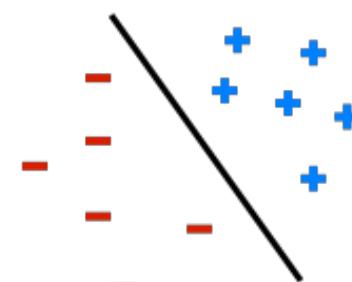
BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0
...		

Properties of Perceptrons

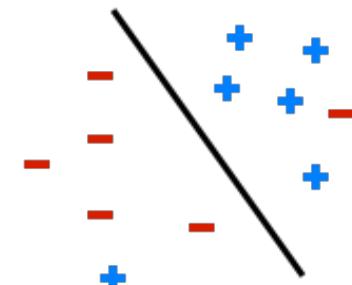
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

Separable

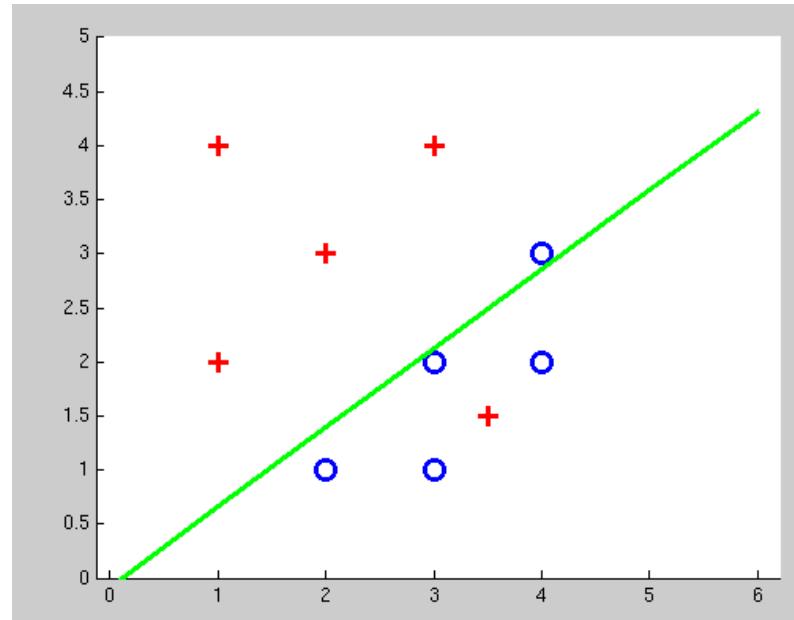


Non-Separable

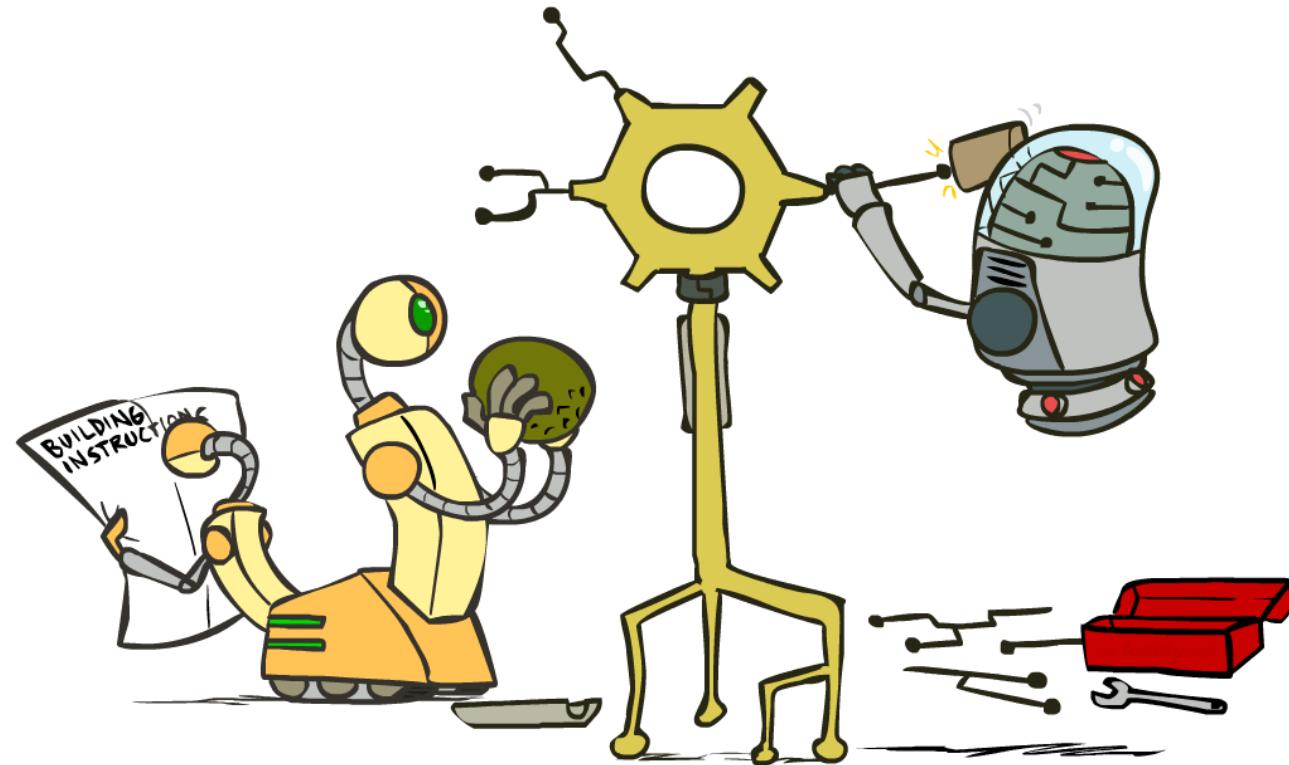


Examples: Perceptron

- Non-Separable Case

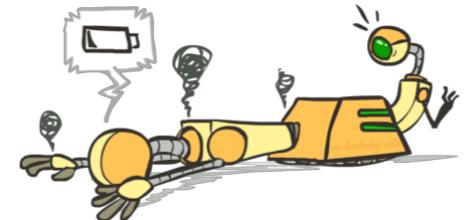
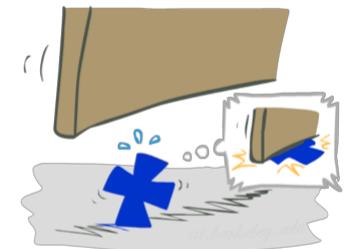
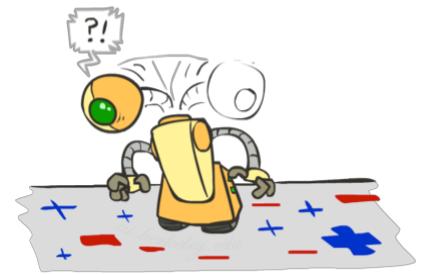
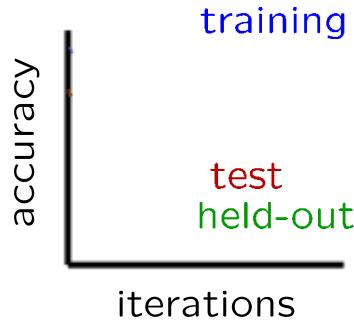
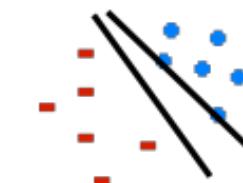
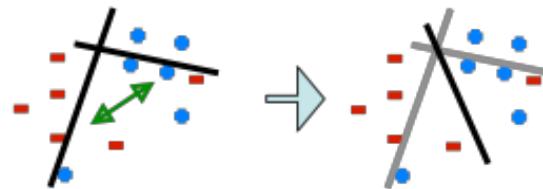


Improving the Perceptron



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

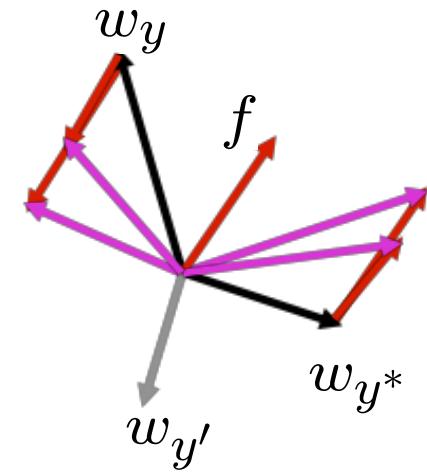
- Idea: adjust the weight update to mitigate these effects
- MIRA*: choose an update size that fixes the current mistake...
- ... but, minimizes the change to w

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

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

- The +1 helps to generalize

* Margin Infused Relaxed Algorithm



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

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

Minimum Correcting Update

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

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



$$\min_\tau \|\tau f\|^2$$

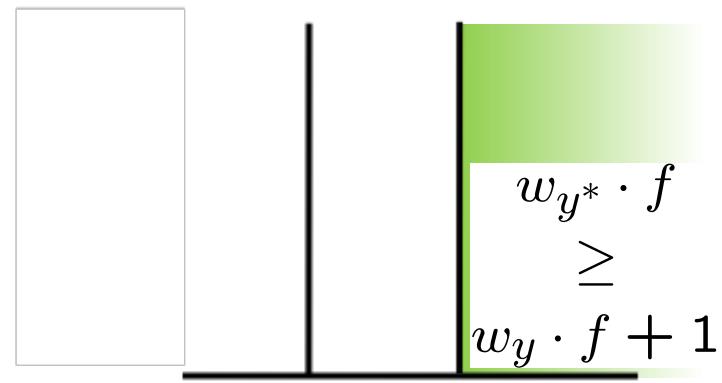
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$



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

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

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



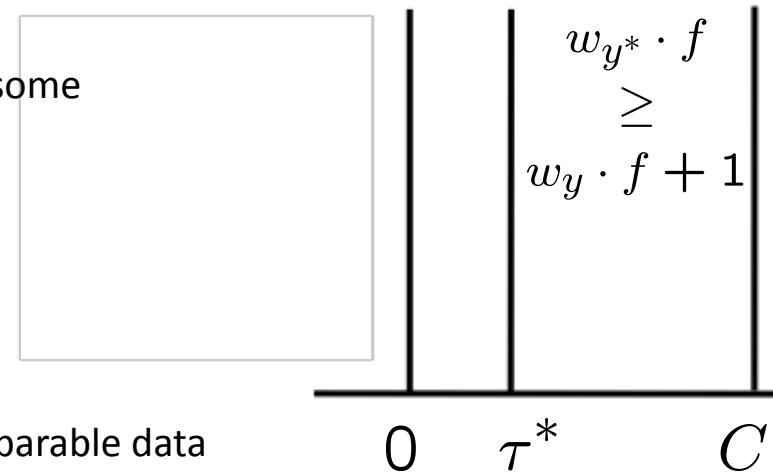
$$\tau = 0$$

min not $\square=0$, 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 \square with some constant C

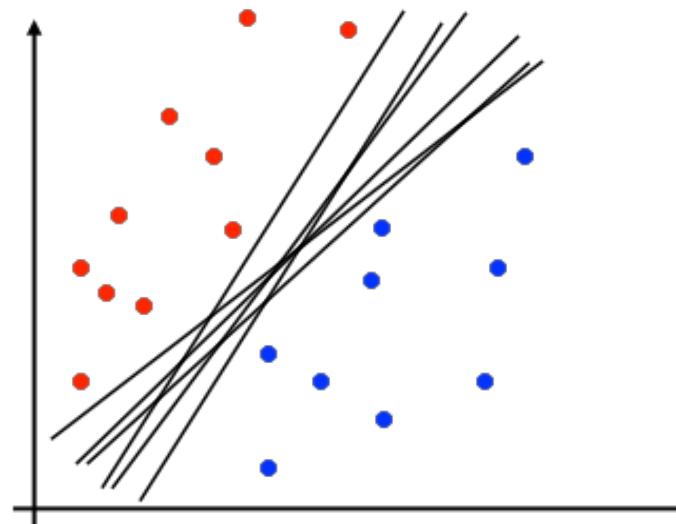
$$\tau^* = \min \left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C \right)$$



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

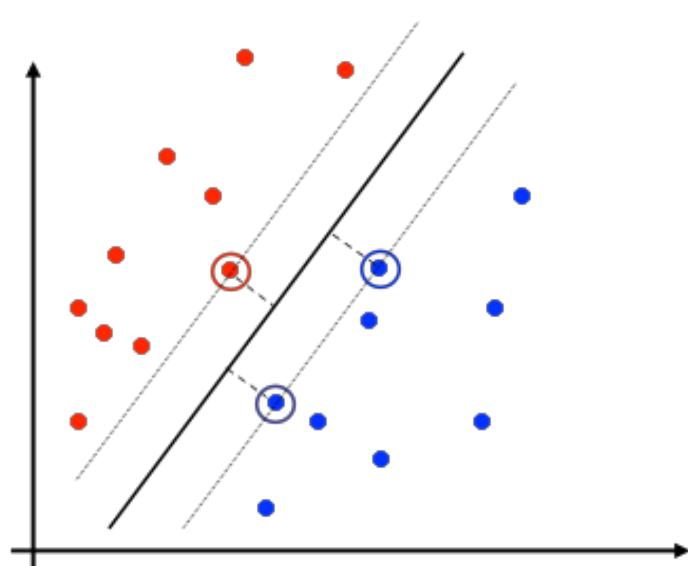
Linear Separators

- 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

$$\begin{aligned} \min_w \quad & \frac{1}{2} \|w - w'\|^2 \\ \text{s.t. } & w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1 \end{aligned}$$

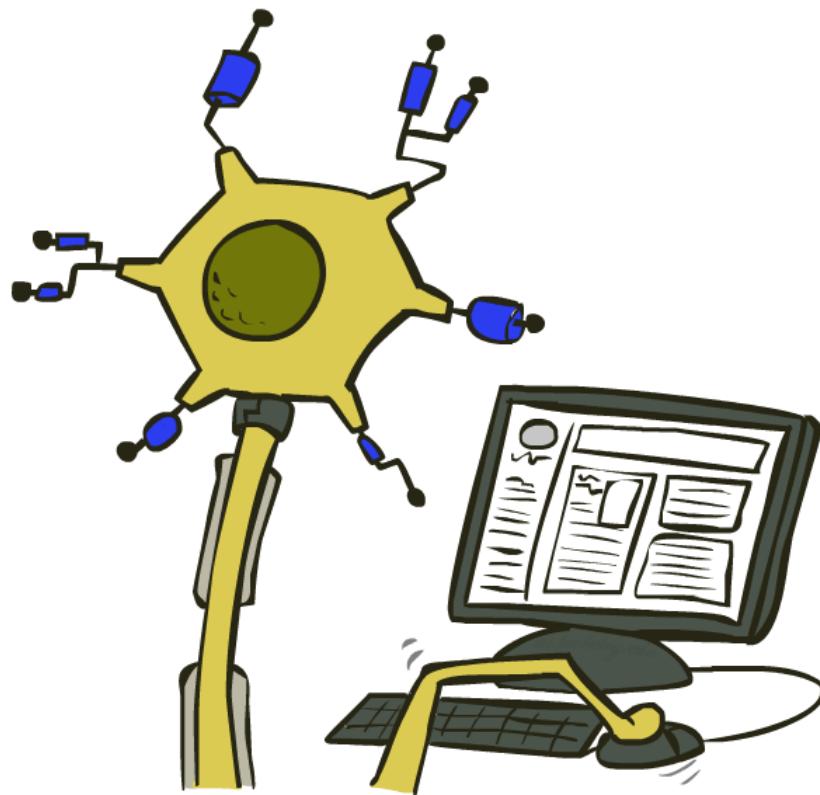
SVM

$$\begin{aligned} \min_w \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t. } & \forall i, y \quad w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1 \end{aligned}$$

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

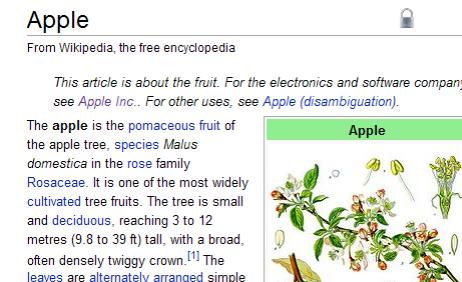
Web Search



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

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



Feature-Based Ranking

x = “Apple Computer”

$$f(x,) = [0.3 \ 5 \ 0 \ 0 \ \dots]$$

Apple

From Wikipedia, the free encyclopedia

This article is about the fruit. For the electronics and software company, see Apple Inc.. For other uses, see Apple (disambiguation).

The apple is the pomaceous fruit of the apple tree, species *Malus domestica* in the rose family Rosaceae. It is one of the most widely cultivated tree fruits. The tree is small and deciduous, reaching 3 to 12 metres (9.8 to 39 ft) tall, with a broad, often densely twiggy crown.^[1] The leaves are alternately arranged simple



Apple Inc.

From Wikipedia, the free encyclopedia
(Redirected from Apple Computer)

Apple Inc.,

Apple Inc.



Perceptron for Ranking

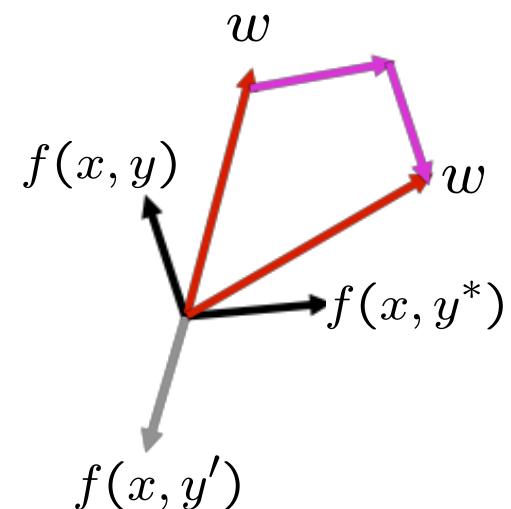
- Inputs x
- Candidates y
- Many feature vectors: $f(x, y)$
- One weight vector: w

- Prediction:

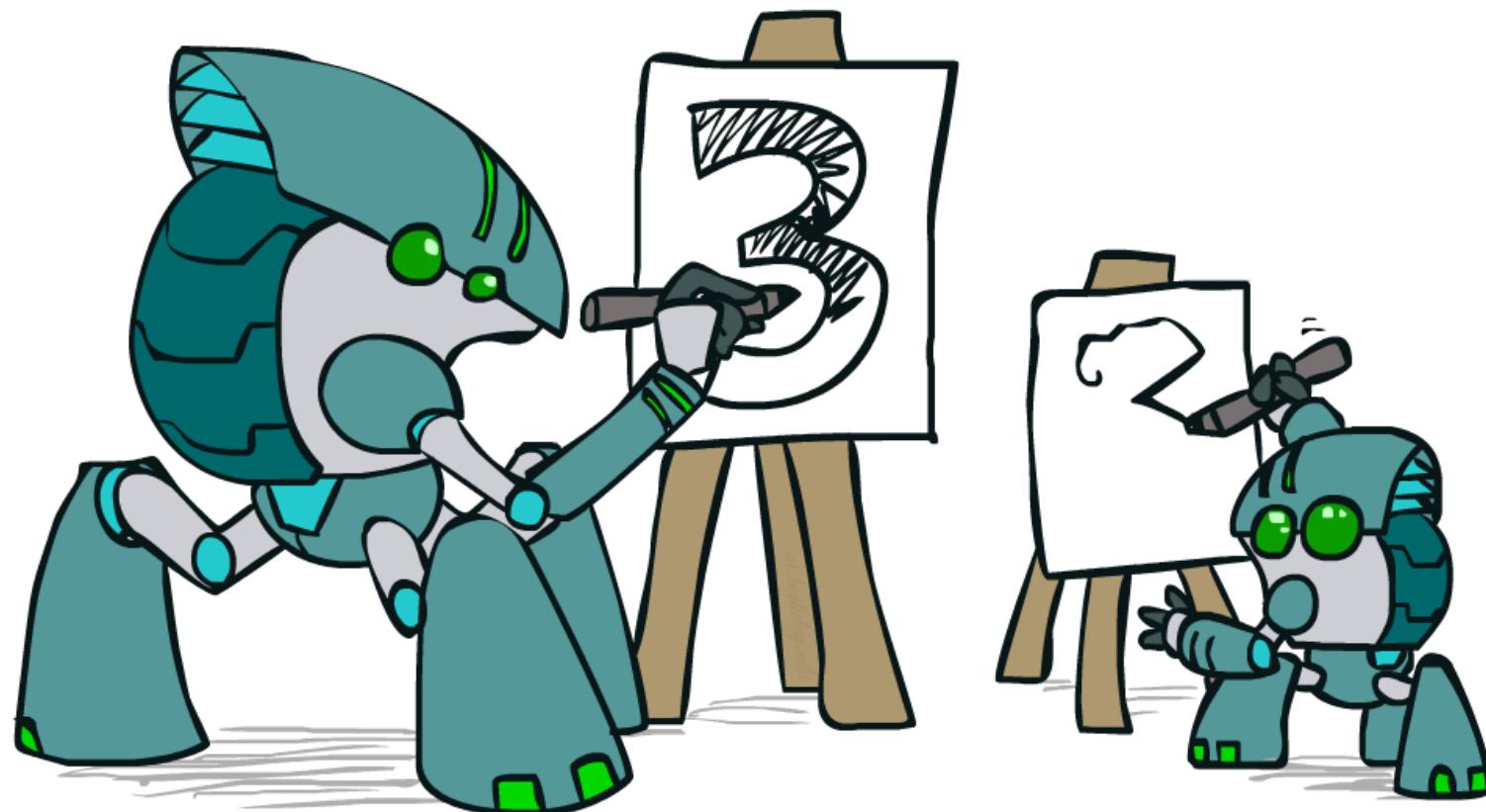
$$y = \arg \max_y w \cdot f(x, y)$$

- Update (if wrong):

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

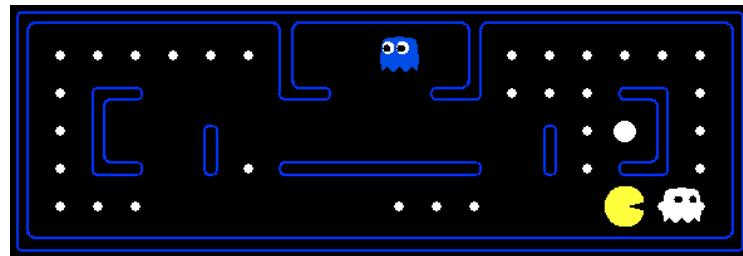


Apprenticeship



Pacman Apprenticeship!

- Examples are states s

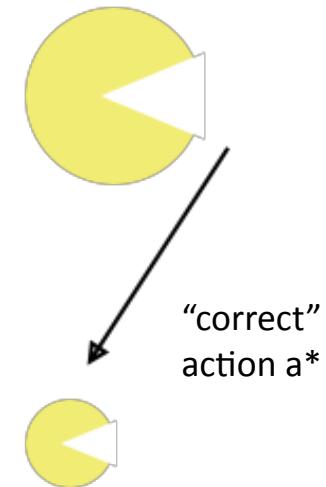


- Candidates are pairs (s, a)
- “Correct” actions: those taken by expert
- Features defined over (s, a) pairs: $f(s, a)$
- Score of a q-state (s, a) given by:

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

- How is this VERY different from reinforcement learning?

[Demo: Pacman Apprentice (L22D1,2,3)]



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

Video of Demo Pacman Apprentice



Next: Kernels and Clustering
