CSEP 573: Artificial Intelligence

Machine Learning: Perceptron

Ali Farhadi

Many slides over the course adapted from Luke Zettlemoyer and Dan Klein.

Generative vs. Discriminative

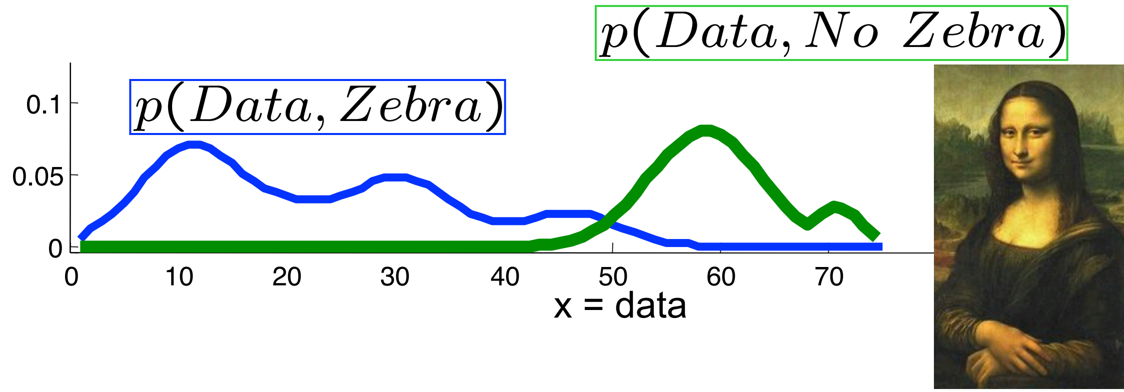
- Generative classifiers:
  - E.g. naïve Bayes
  - A joint probability model with evidence variables
  - Query model for causes given evidence

- Discriminative classifiers:
  - No generative model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Robust, accurate with varied features
  - Loosely: mistake driven rather than model driven

Discriminative vs. generative

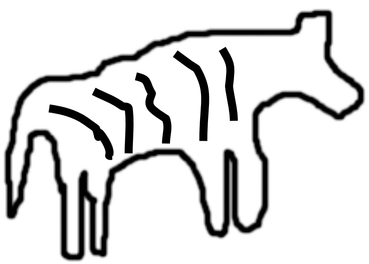
- Generative model

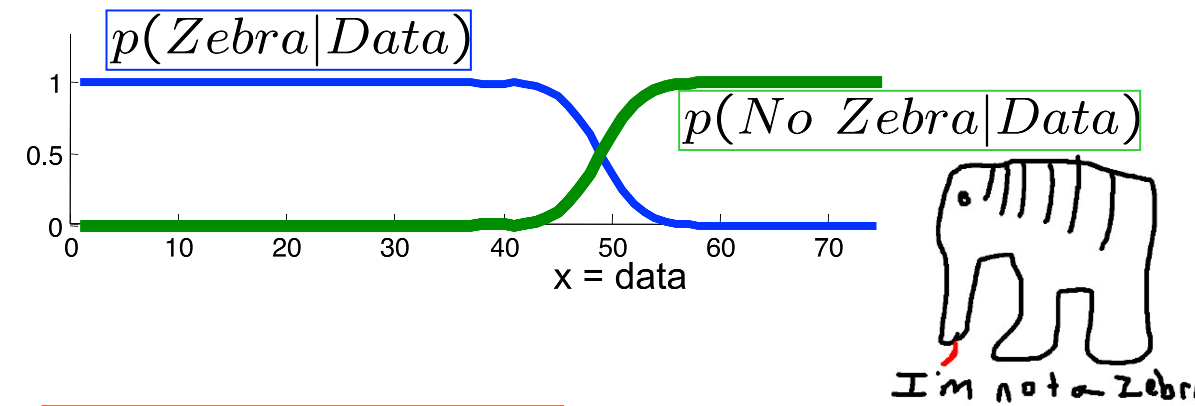
  (*The artist*)



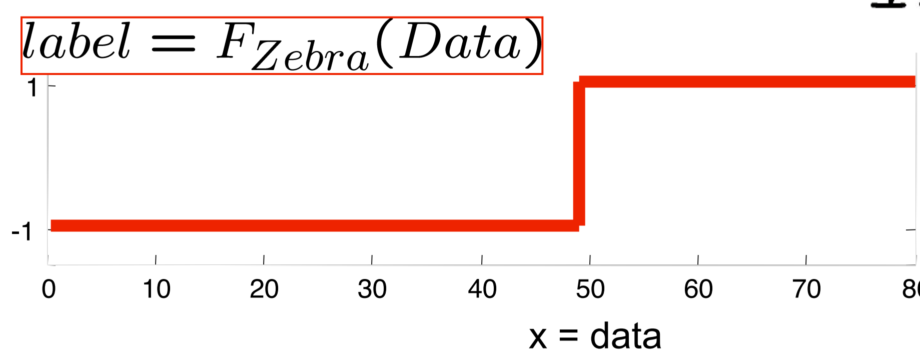
- Discriminative model

  (The lousy painter)



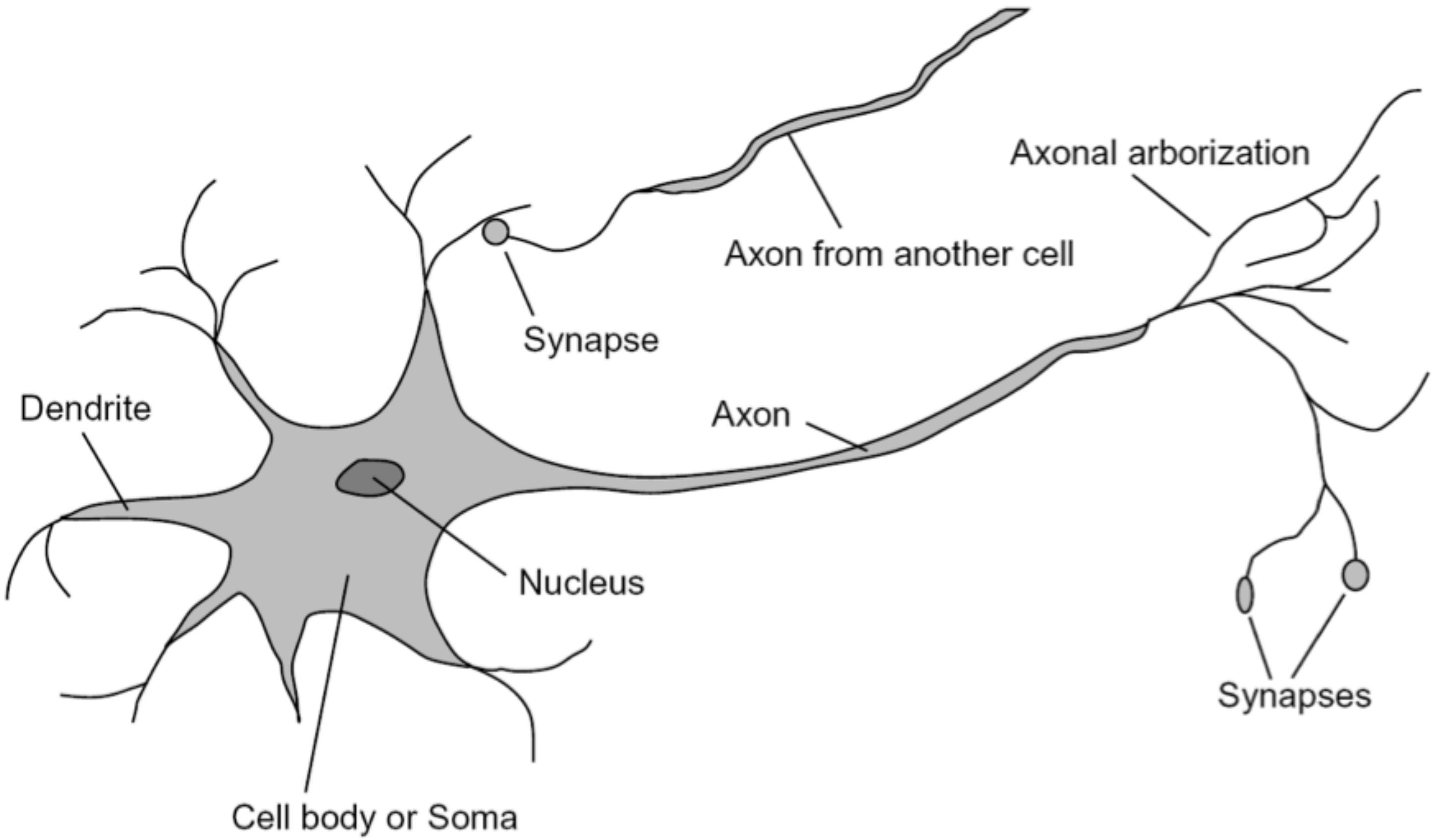


- Classification function



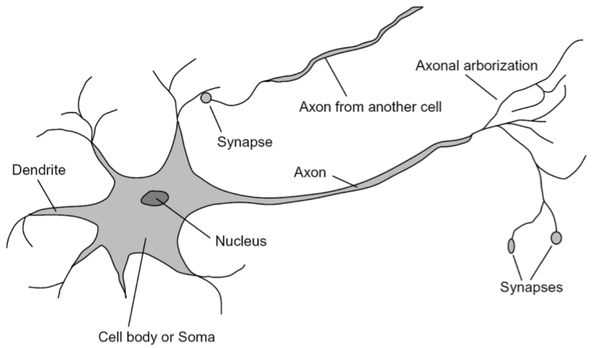
Some (Simplified) Biology

- Very loose inspiration: human neurons



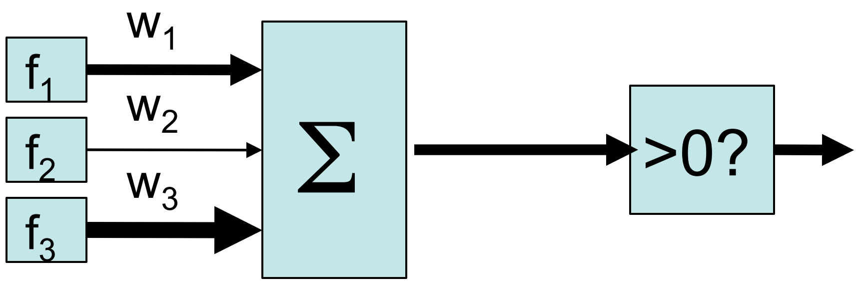
Linear Classifiers

- 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



Example: Spam

- Imagine 4 features (spam is "positive" class):
  - free (number of occurrences of "free")
  - money (occurrences of "money")
  - BIAS (intercept, always has value 1)

$$w \cdot f(x)$$

$$\sum_i w_i \cdot f_i(x)$$

x: "free money"

$f(x)$:

```
BIAS  :  1
free  :  1
money :  1
...
```

w:

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

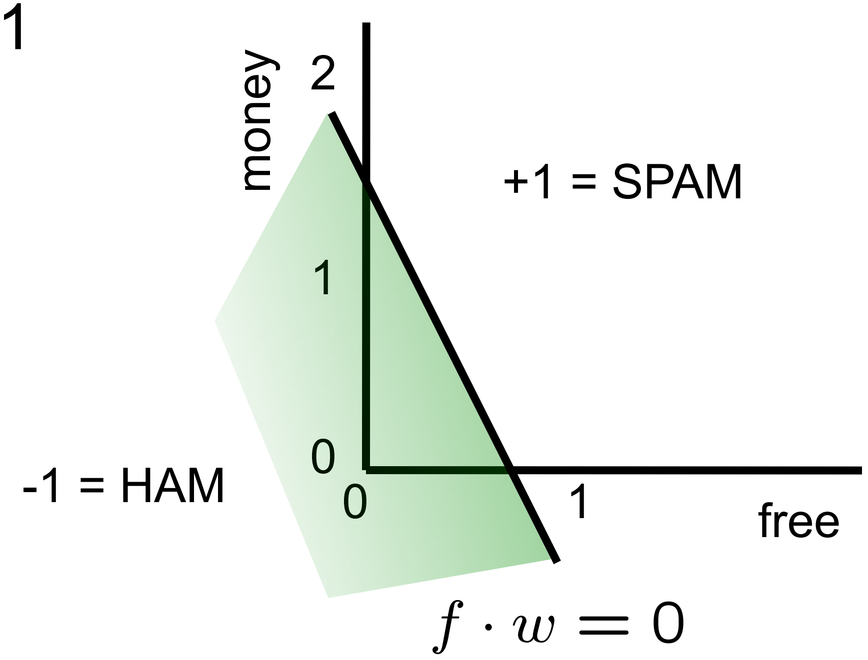
$$(1)(-3) \; + \\ (1)(4) \; + \\ (1)(2) \; + \\ \ldots \\ = 3$$

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
...
```



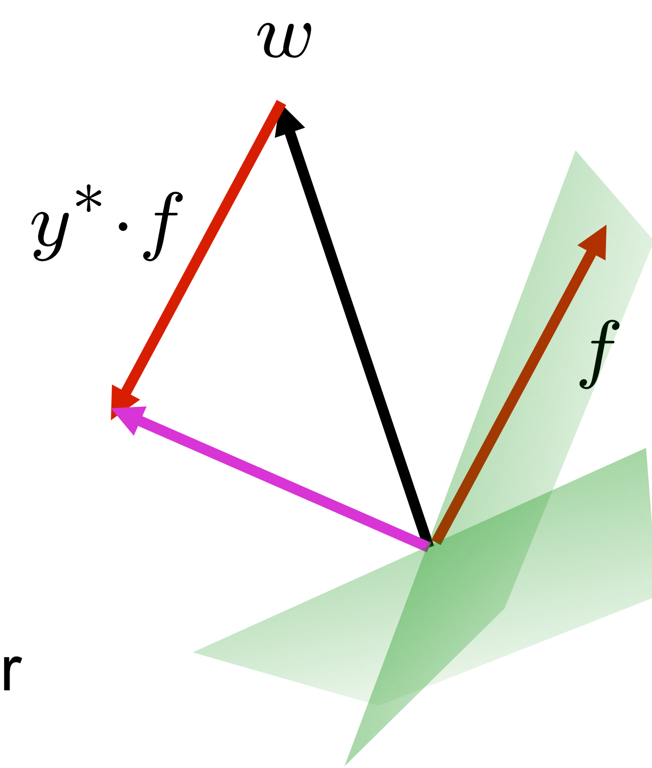
Binary Perceptron Algorithm

- Start with zero weights
- For each training instance:
  - Classify with current weights

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

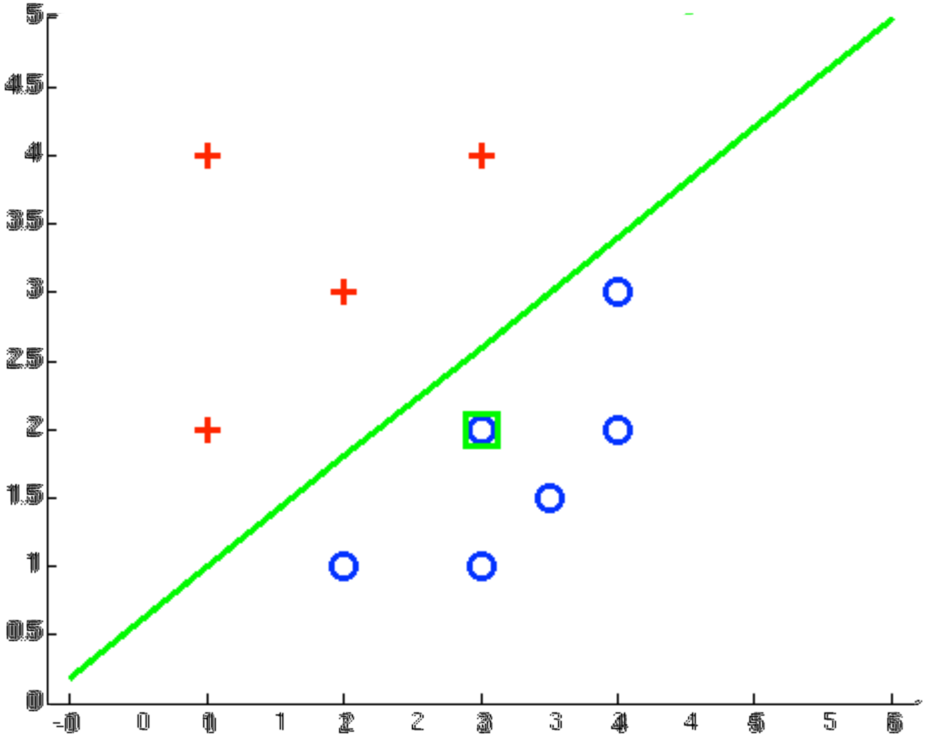
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

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



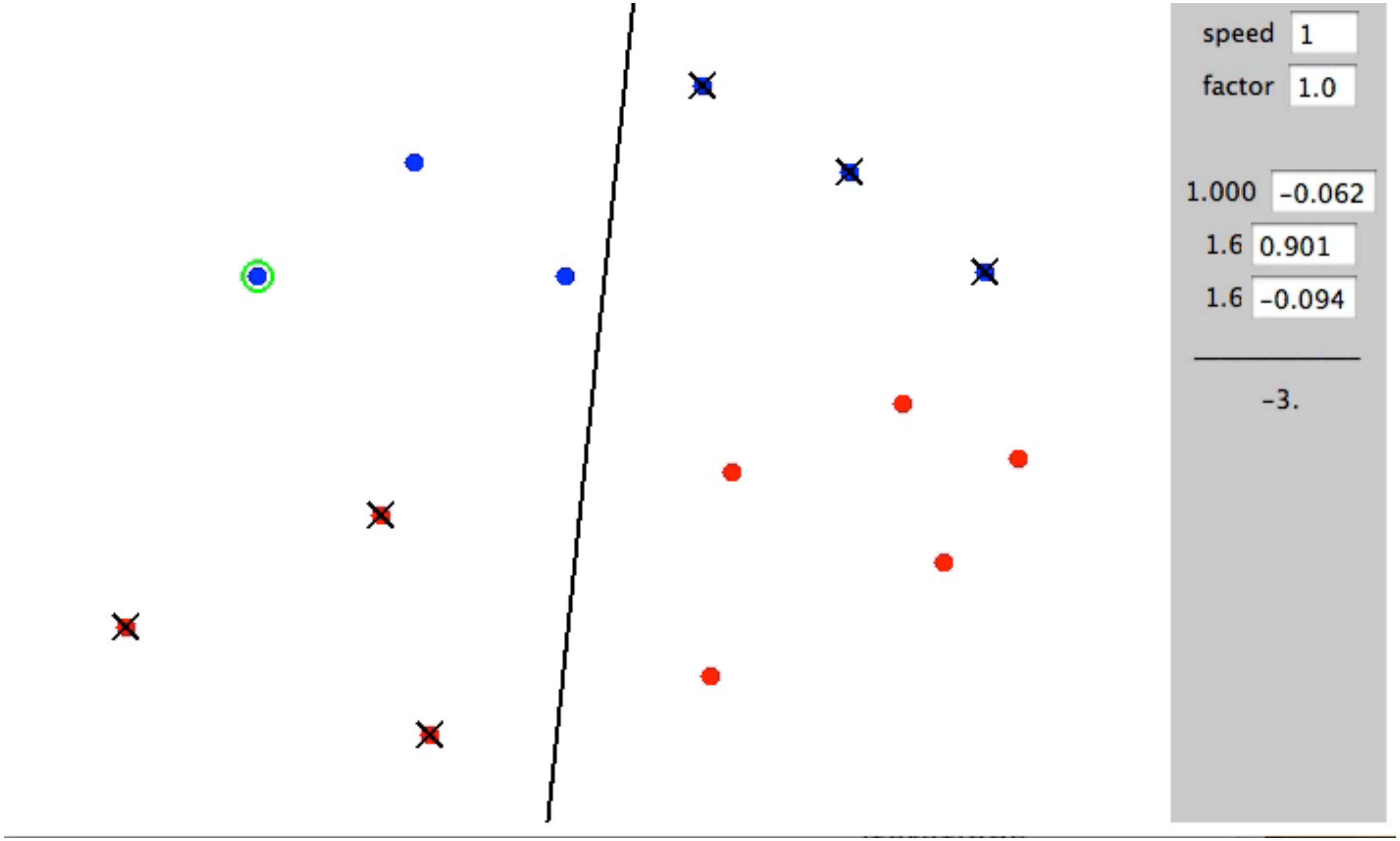
Examples: Perceptron

- Separable Case



Examples: Perceptron

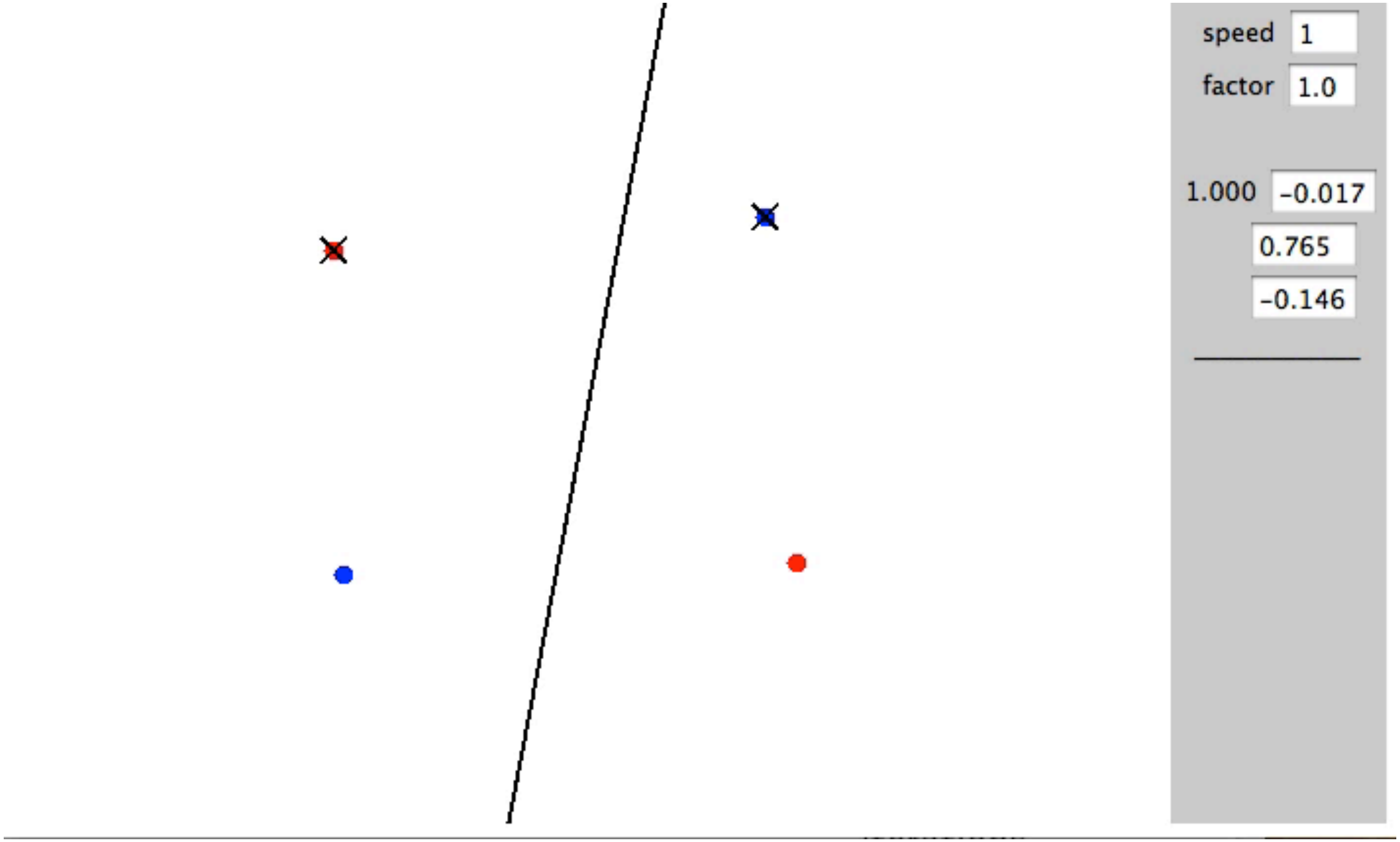
- Separable Case



http://isl.ira.uka.de/neuralNetCourse/2004/VL_11_5/Perceptron.html

Examples: Perceptron

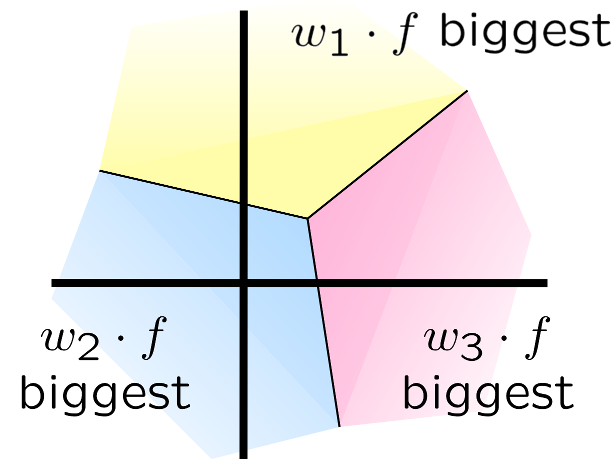
- Inseparable Case



http://isl.ira.uka.de/neuralNetCourse/2004/VL_11_5/Perceptron.html

Multiclass Decision Rule

- If we have more than two classes:
  - Have a weight vector for each class: w_y
  - Calculate an activation for each class



$$\text{activation}_w(x, y) = w_y \cdot f(x)$$

  - Highest activation wins

$$y = \arg\max_y \left(\text{activation}_w(x, y)\right)$$

Example

"win the vote"

"win the election"

"win the game"

w_{SPORTS}	$w_{POLITICS}$	w_{TECH}
BIAS :	BIAS :	BIAS :
win :	win :	win :
game :	game :	game :
vote :	vote :	vote :
the :	the :	the :
...	...	...

Example

"win the vote"

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

w_{SPORTS}

BIAS	:	-2
win	:	4
game	:	4
vote	:	0
the	:	0
...		

$w_{POLITICS}$

BIAS	:	1
win	:	2
game	:	0
vote	:	4
the	:	0
...		

w_{TECH}

BIAS	:	2
win	:	0
game	:	2
vote	:	0
the	:	0
...		

The Multi-class Perceptron Alg.

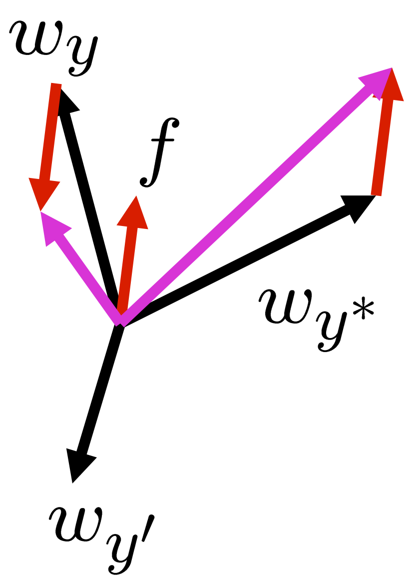
- Start with zero weights
- Iterate training examples
  - Classify with current weights

$$\begin{aligned} y &= \arg\max_y \; w_y \cdot f(x) \\ &= \arg\max_y \; \textstyle\sum_i w_{y,i} \cdot f_i(x) \end{aligned}$$

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

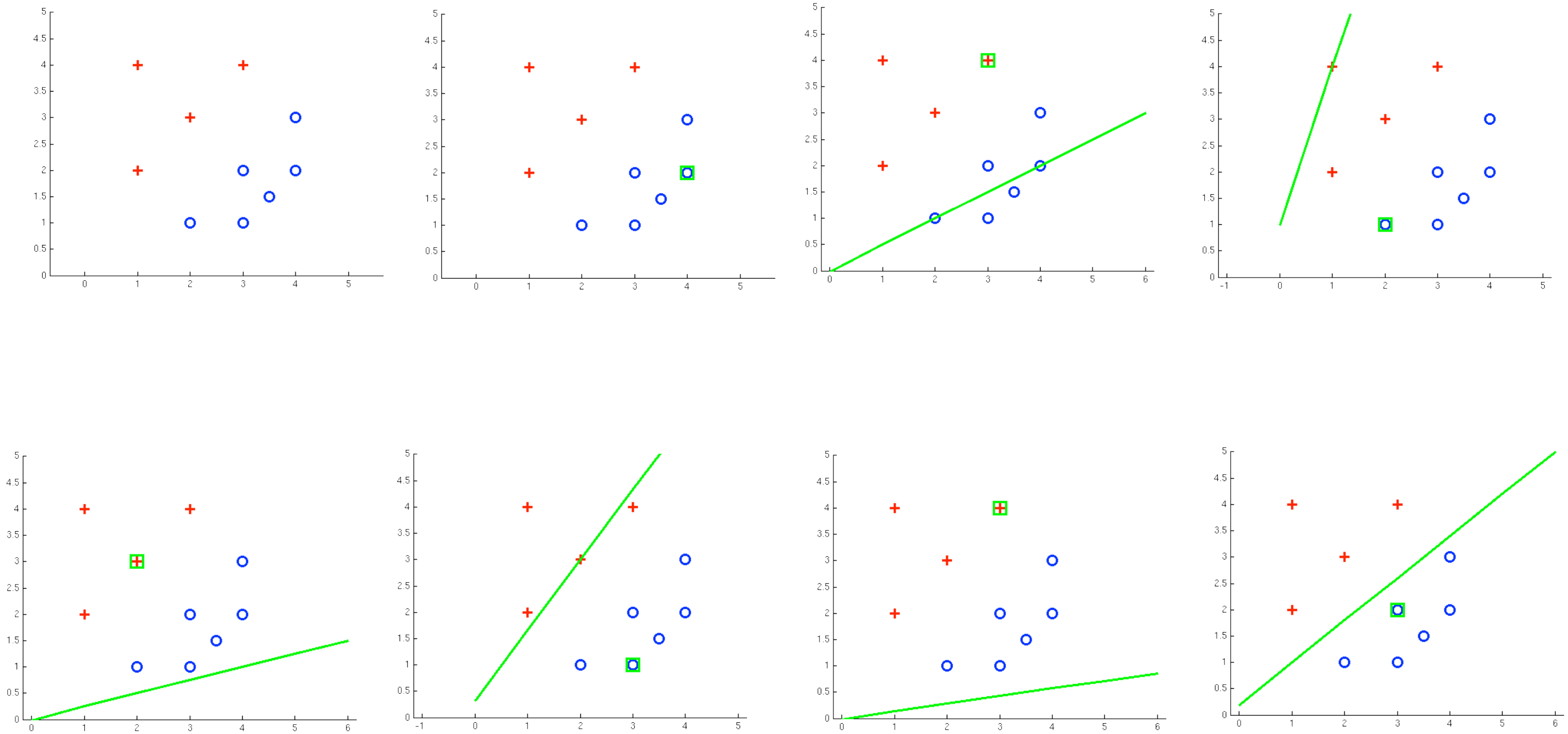
$$w_y = w_y - f(x)$$

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



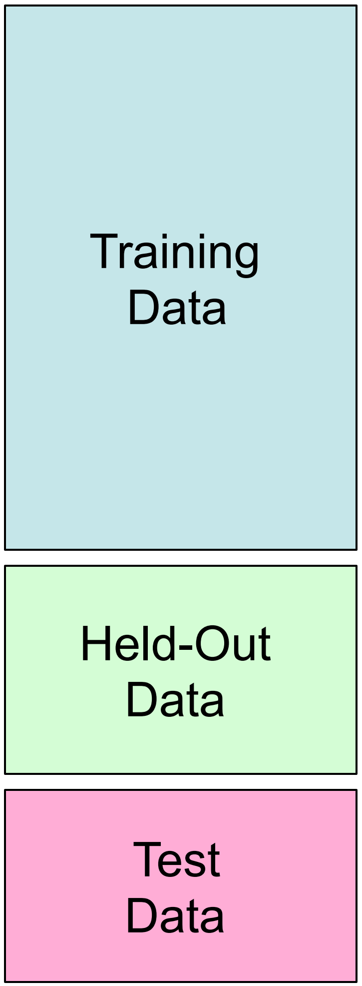
Examples: Perceptron

- Separable Case



Mistake-Driven Classification

- For Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data
- For the perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until held-out accuracy maxes out

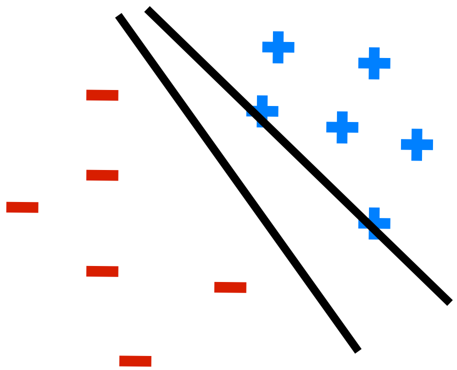


Properties of Perceptrons

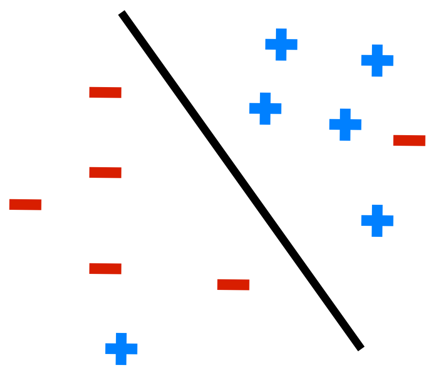
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability

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

Separable

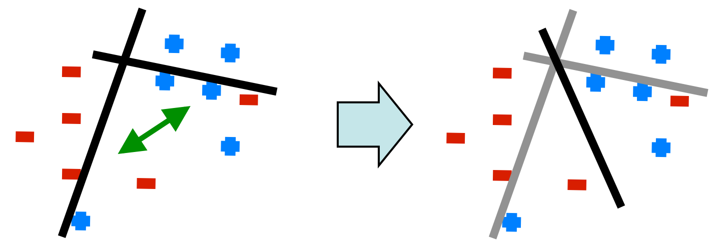


Non-Separable

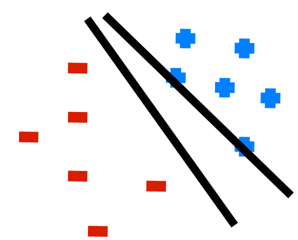


Problems with the Perceptron

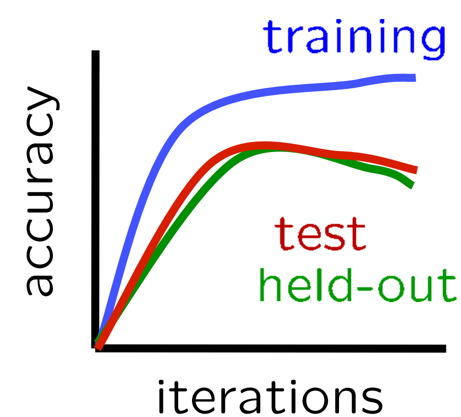
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution



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



Fixing the Perceptron

- Idea: adjust the weight update to mitigate these effects

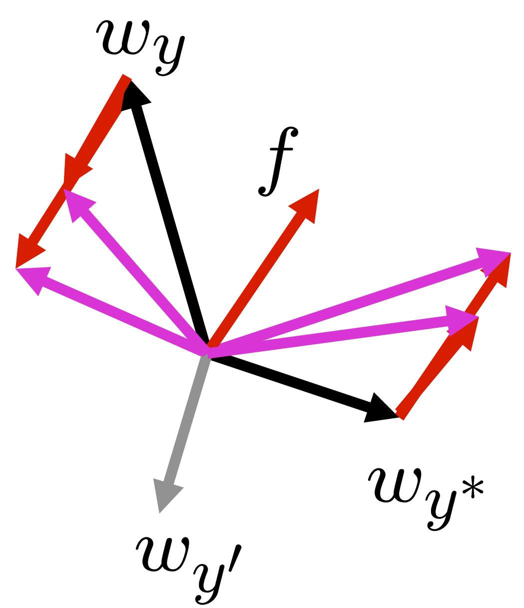
- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to w

$$\min_{w} \frac{1}{2} \sum_{y} ||w_y - w'_y||^2$$

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

- The +1 helps to generalize

* Margin Infused Relaxed Algorithm



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

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

Minimum Correcting Update

$$\min_{w} \frac{1}{2}\sum_{y} ||w_y - w'_y||^2$$

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

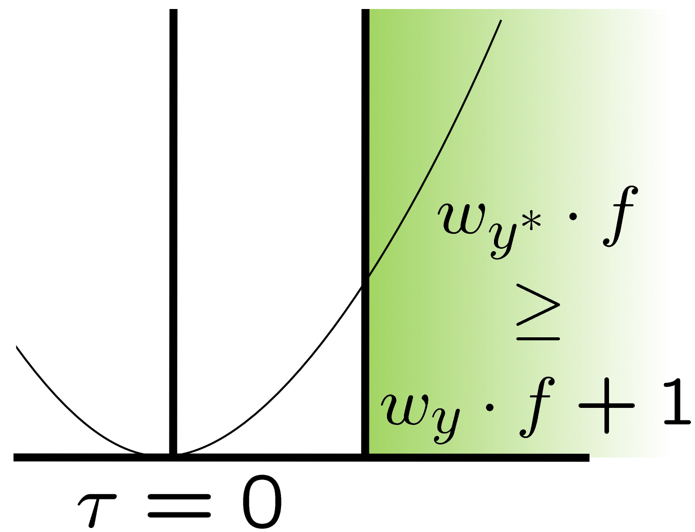
$$\min_{\tau} ||\tau f||^2$$

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

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

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

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

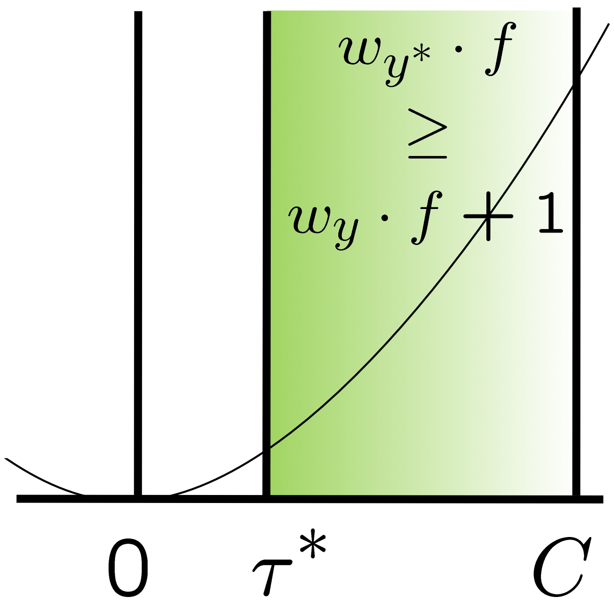


min not τ=0, or would not have made an error, so min will be where equality holds

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

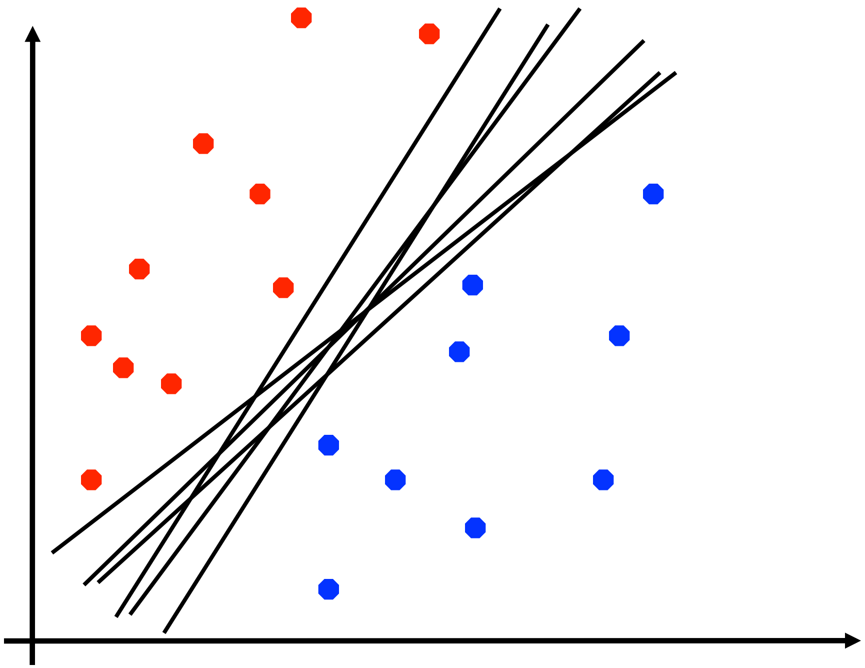
$$\tau^* = \min\left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C\right)$$



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

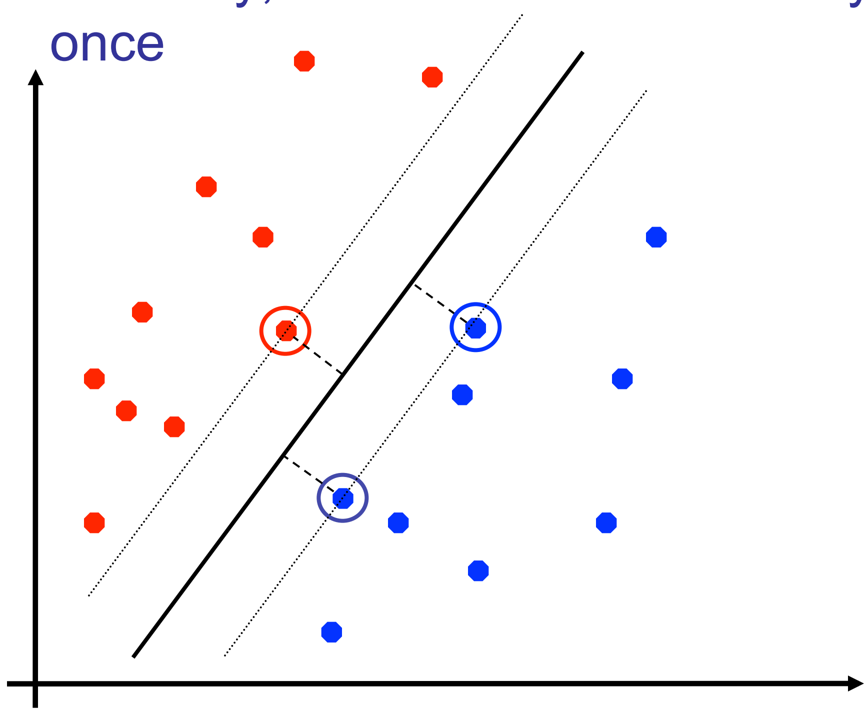
Linear Separators

- 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

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

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

Classification: Comparison

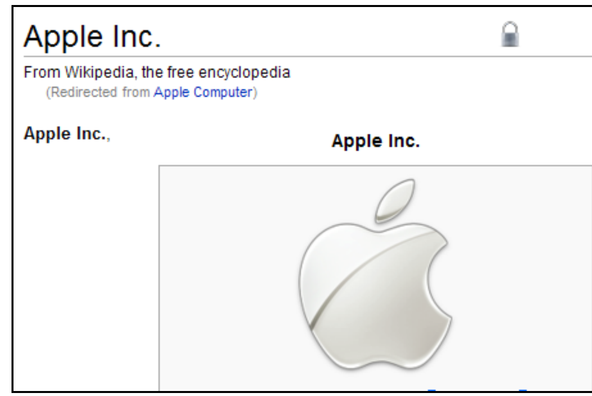
- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)

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

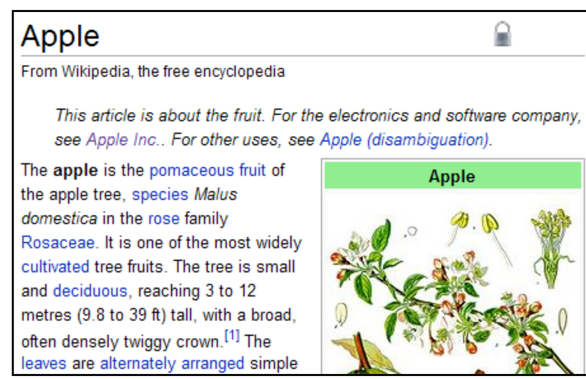
Extension: Web Search

x = “Apple Computers”

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



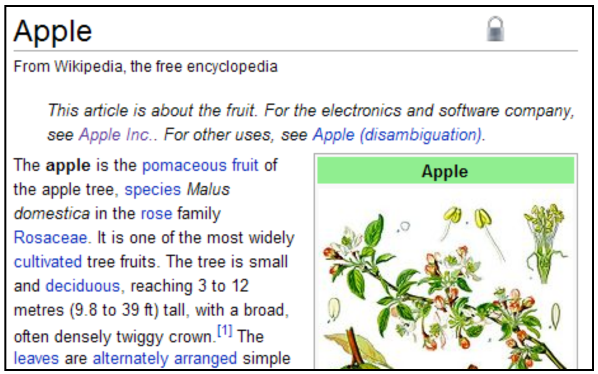
- Web search: not exactly classification, but rather ranking



Feature-Based Ranking

x = “Apple Computers”

$f($x,


Apple

From Wikipedia, the free encyclopedia

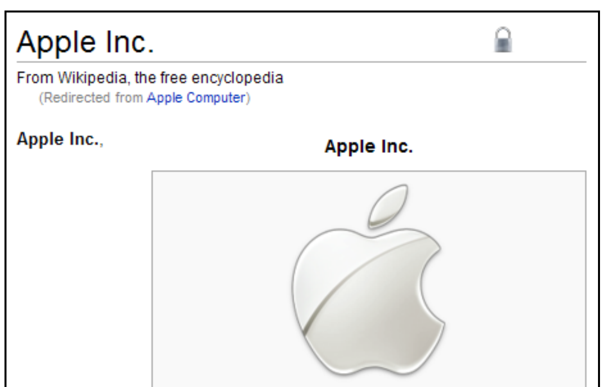
This article is about the fruit. For the electronics and software company, see Apple Inc.. For other uses, see Apple (disambiguation).

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

$) = [0.3\ 5\ 0\ 0\ \ldots]$

$f($x,


Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

Apple Inc.

$) = [0.8\ 4\ 2\ 1\ \ldots]$

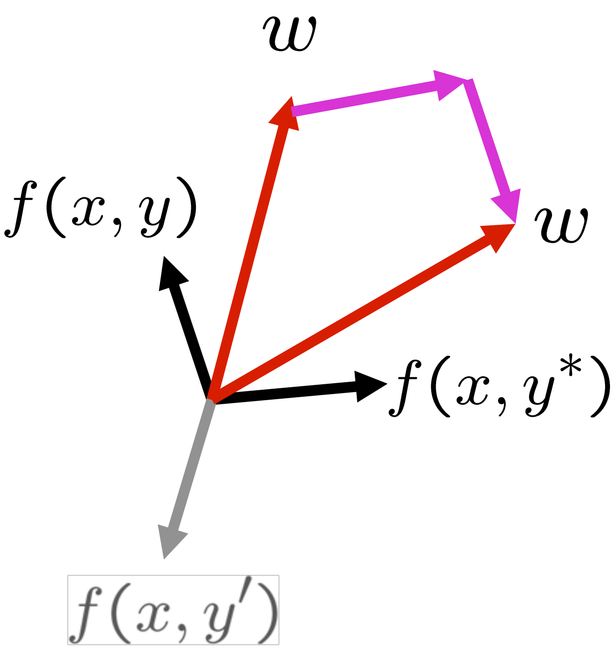
Perceptron for Ranking

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

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

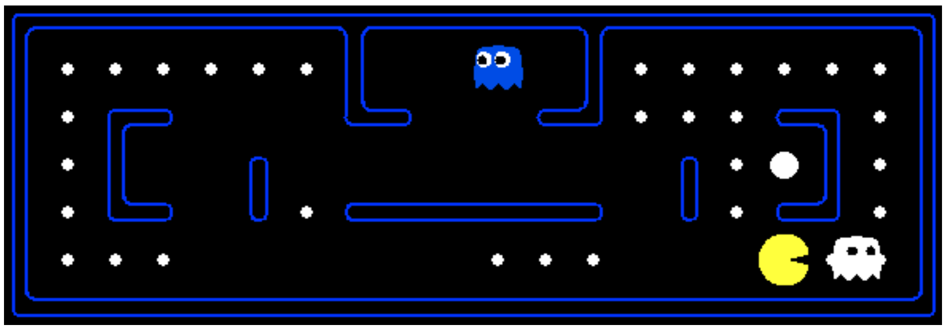
  - Update (if wrong):

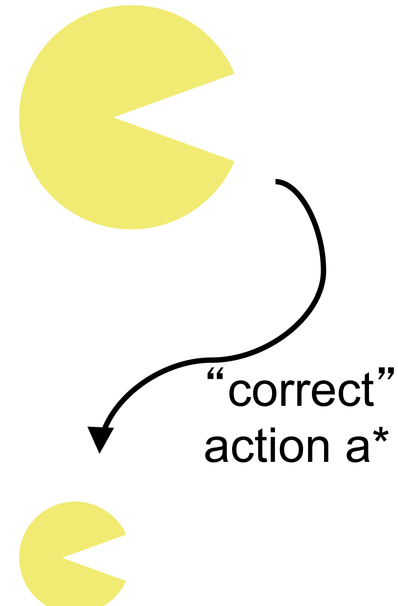
$$w = w + f(x, y^*) - f(x, y)$$



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?

Exam Topics

- Search
  - BFS, DFS, UCS, A* (tree and graph)
  - Completeness and Optimality
  - Heuristics: admissibility and consistency
- Games
  - Minimax, Alpha-beta pruning, Expectimax, Evaluation Functions
- MDPs
  - Bellman equations
  - Value and policy iteration
- Reinforcement Learning
  - Exploration vs Exploitation
  - Model-based vs. model-free
  - TD learning and Q-learning
  - Linear value function approx.
- Hidden Markov Models
  - Markov chains
  - Forward algorithm
  - Particle Filter
- Bayesian Networks
  - Basic definition, independence
  - Variable elimination
  - Sampling (prior, rejection, importance)
- Machine Learning:
  - Naïve Bayes
  - Perceptron