CSE 473: Artificial Intelligence

Machine Learning: Perceptron

Hanna Hajishirzi

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

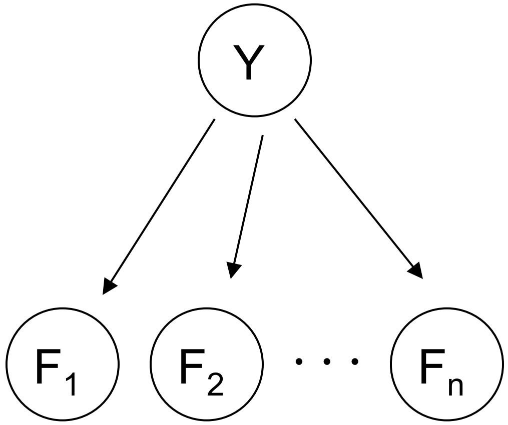
Exam Topics

- 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, likelihood)
- Machine Learning:
  - Naïve Bayes,
  - Perceptron (high level)

General Naïve Bayes

- A general naive Bayes model:

$$P(\mathsf{Y}, \mathsf{F}_1 \dots \mathsf{F}_n) = P(\mathsf{Y}) \prod_i P(\mathsf{F}_i | \mathsf{Y})$$



- We only specify how each feature depends on the class
- Total number of parameters is linear in n

Parameter Estimation

- Estimating distribution of random variables like X or X | Y
- Elicitation: ask a human!
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating
- Empirically: use training data
  - For each outcome x, look at the empirical rate of that value:

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

$$P_{\text{ML}}(\text{r}) = 1/3$$

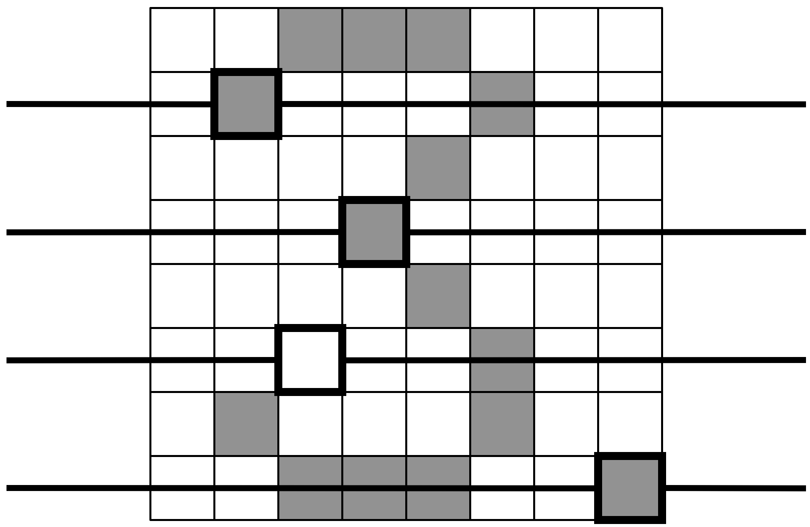
  - This is the estimate that maximizes the likelihood of the data

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

Example: Overfitting

$P(\text{features}, C = 2)$ $\qquad$ $P(\text{features}, C = 3)$

$P(C = 2) = 0.1$ $\qquad$ $P(C = 3) = 0.1$



$P(\text{on}|C = 2) = 0.8$ $\qquad$ $P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 2) = 0.1$ $\qquad$ $P(\text{on}|C = 3) = 0.9$

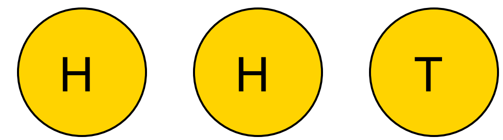
$P(\text{off}|C = 2) = 0.1$ $\qquad$ $P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 2) = 0.01$ $\qquad$ $P(\text{on}|C = 3) = 0.0$

2 wins!!

Estimation: Laplace Smoothing

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



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

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

$$P_{ML}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

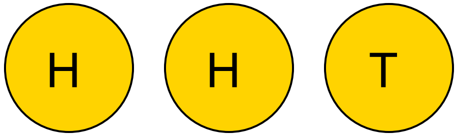
$$P_{LAP}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

Estimation: 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



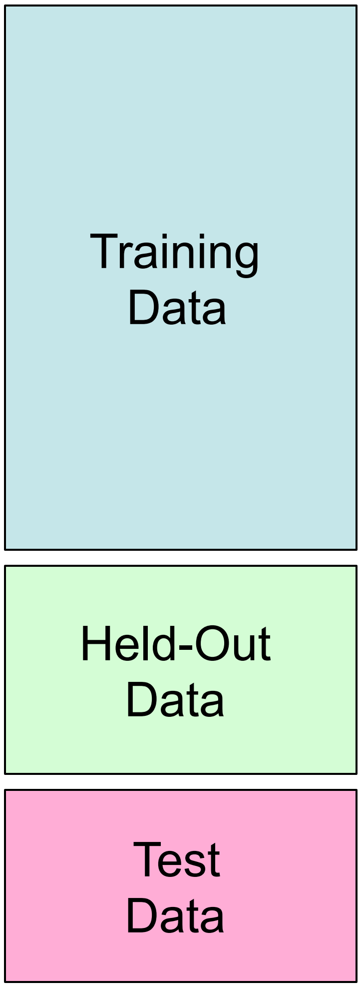
$$P_{LAP,0}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

$$P_{LAP,1}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

$$P_{LAP,100}(X) = \left\langle \frac{102}{203}, \frac{101}{203} \right\rangle$$

Important Concepts

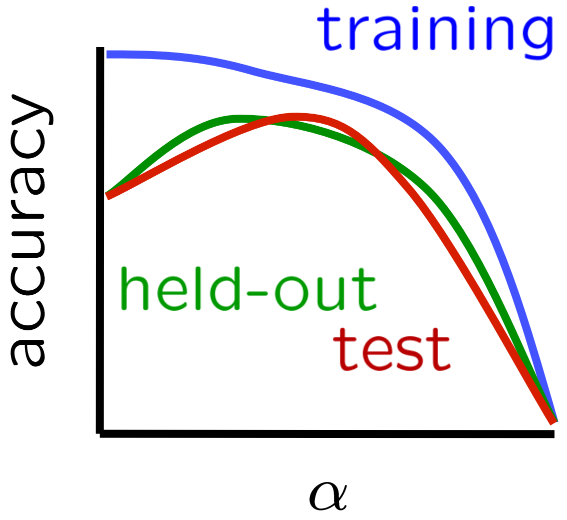
- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Very important: never "peek" at the test set!
- Evaluation
  - Compute accuracy of test set
  - 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



Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(Y|X), P(Y)
  - Hyperparameters, like the amount of smoothing to do: k, α

- Where to learn?
  - Learn parameters from training data
  - Must 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. calling everything "ham" gets 66%, so a classifier that gets 70% isn't very good…
- For real research, usually use previous work as a (strong) baseline

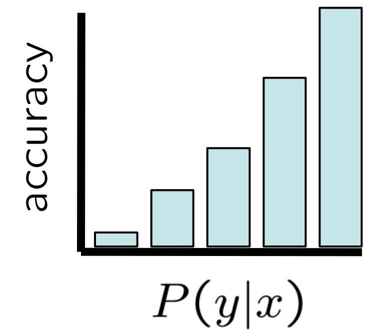
Confidences from a Classifier

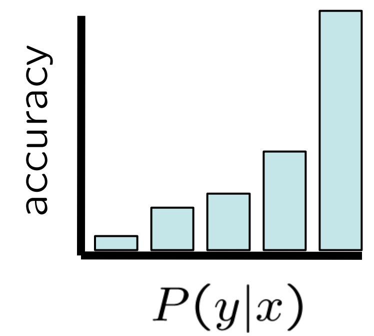
- The confidence of a probabilistic classifier:
  - Posterior over the top label

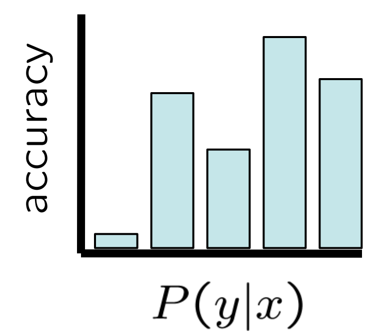
$$\text{confidence}(x) = \max_y P(y|x)$$

  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

- Calibration
  - Weak calibration: higher confidences mean higher accuracy
  - Strong calibration: confidence predicts accuracy rate
  - What's the value of calibration?

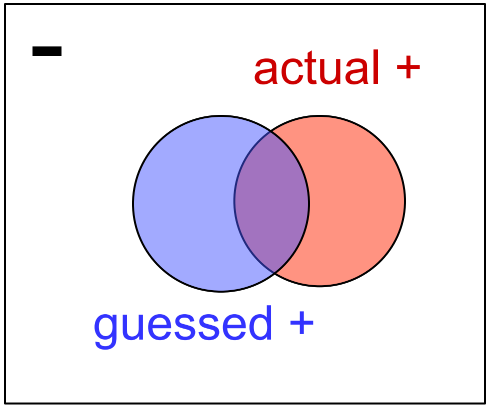






Precision vs. Recall

- Let's say we want to classify web pages as homepages or not
  - In a test set of 1K pages, there are 3 homepages
  - Our classifier says they are all non-homepages
  - 99.7 accuracy!
  - Need new measures for rare positive events



- Precision: fraction of guessed positives which were actually positive

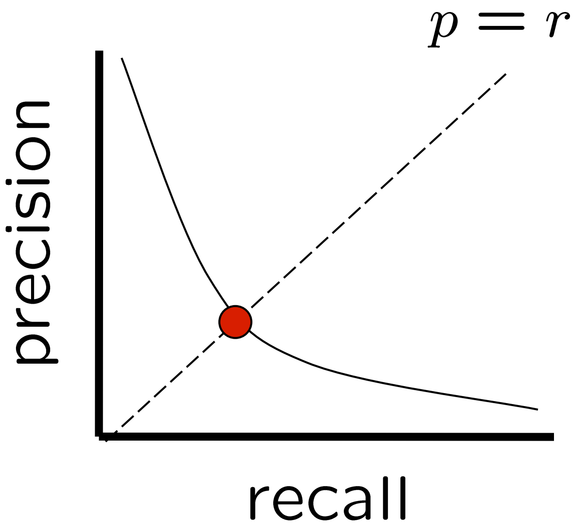
- Recall: fraction of actual positives which were guessed as positive

- Say we detect 5 spam emails, of which 2 were actually spam, and we missed one
  - Precision: 2 correct / 5 guessed = 0.4
  - Recall: 2 correct / 3 true = 0.67

- Which is more important in customer support email automation?
- Which is more important in airport face recognition?

Precision vs. Recall

- Precision/recall tradeoff
  - Often, you can trade off precision and recall



- To summarize the tradeoff:
  - Break-even point: precision value when p = r
  - F-measure: harmonic mean of p and r:

$$F_1 = \frac{2}{1/p + 1/r}$$

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?

- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K 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?

- Can add these information sources as new variables in the NB model

- Classifiers which let you easily add arbitrary features more easily

Summary

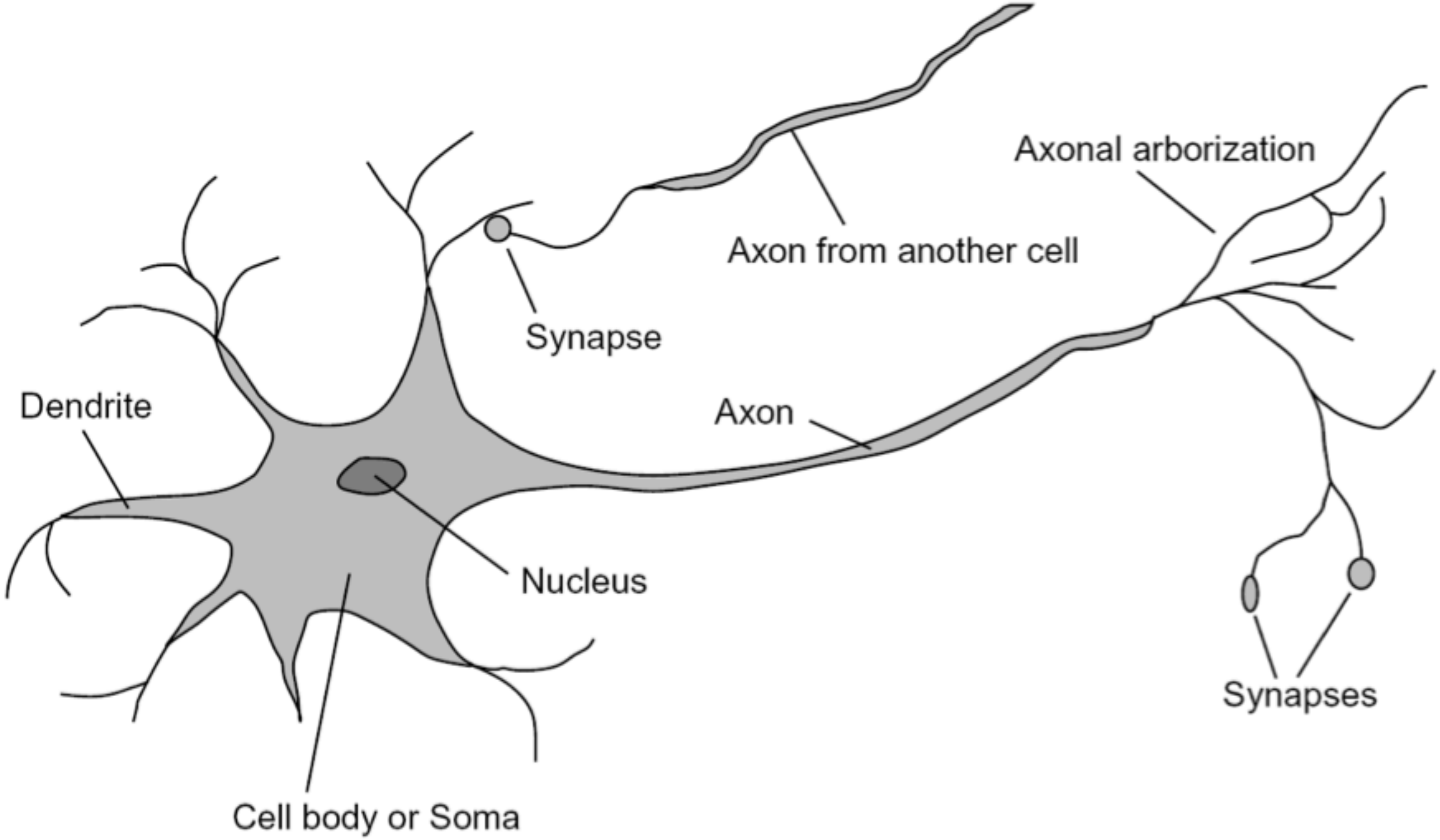
- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems

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

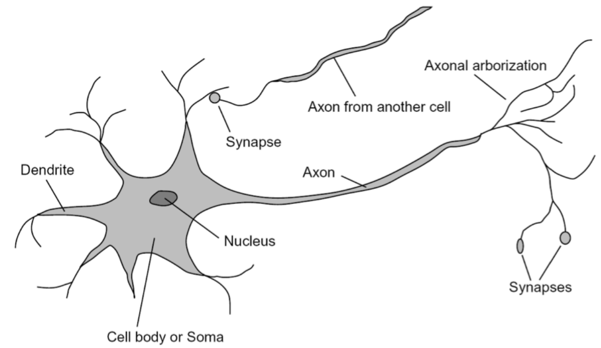
Some (Simplified) Biology

- Very loose inspiration: human neurons



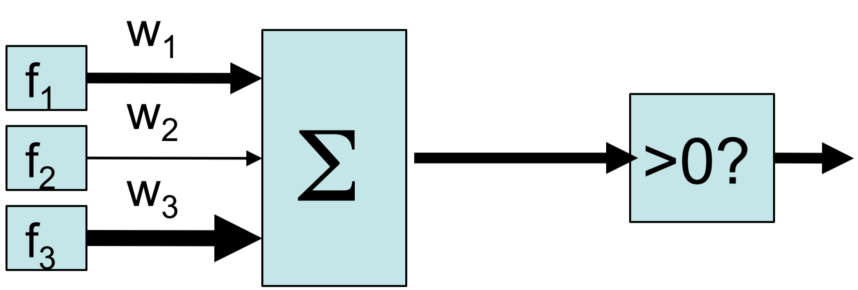
Linear Classifiers

- 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	$f(x)$	w
"free money"	BIAS : 1 free : 1 money : 1 ...	BIAS : -3 free : 4 money : 2 ...

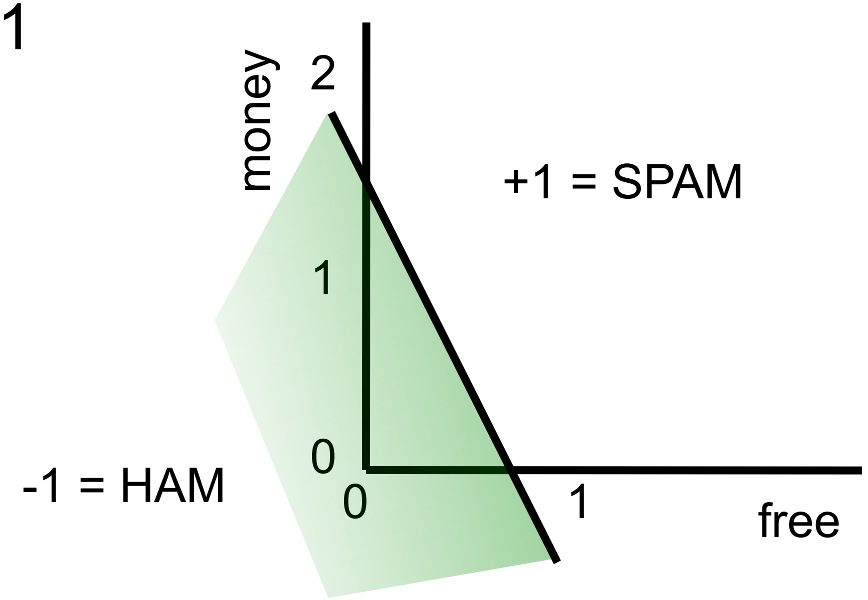
$$\begin{aligned} &(1)(-3) \quad + \\ &(1)(4) \quad + \\ &(1)(2) \quad + \\ &\dots \\ &= 3 \end{aligned}$$

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
...
```



$f \cdot w = 0$

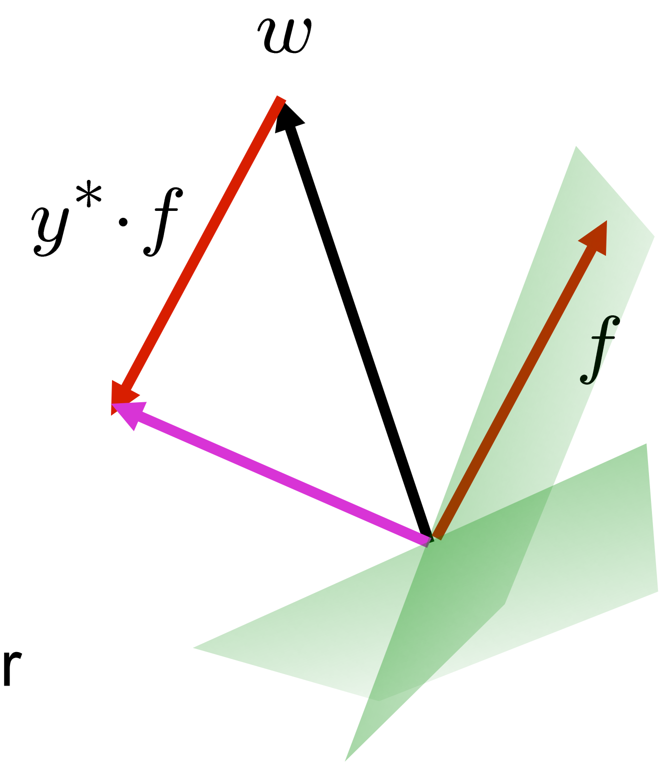
Binary Perceptron Algorithm

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

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

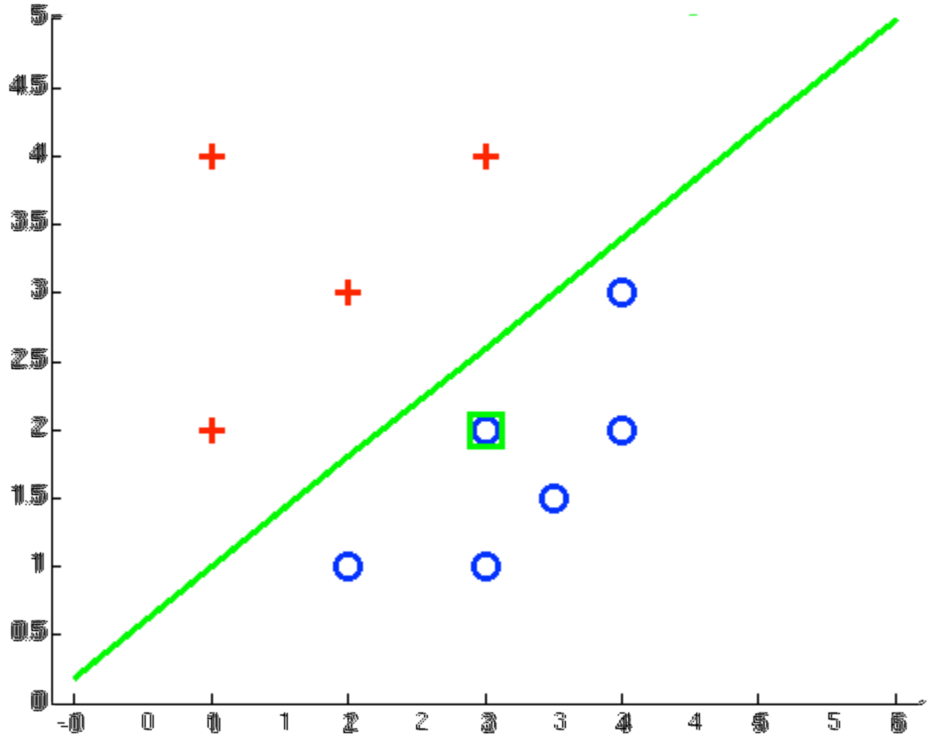
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

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



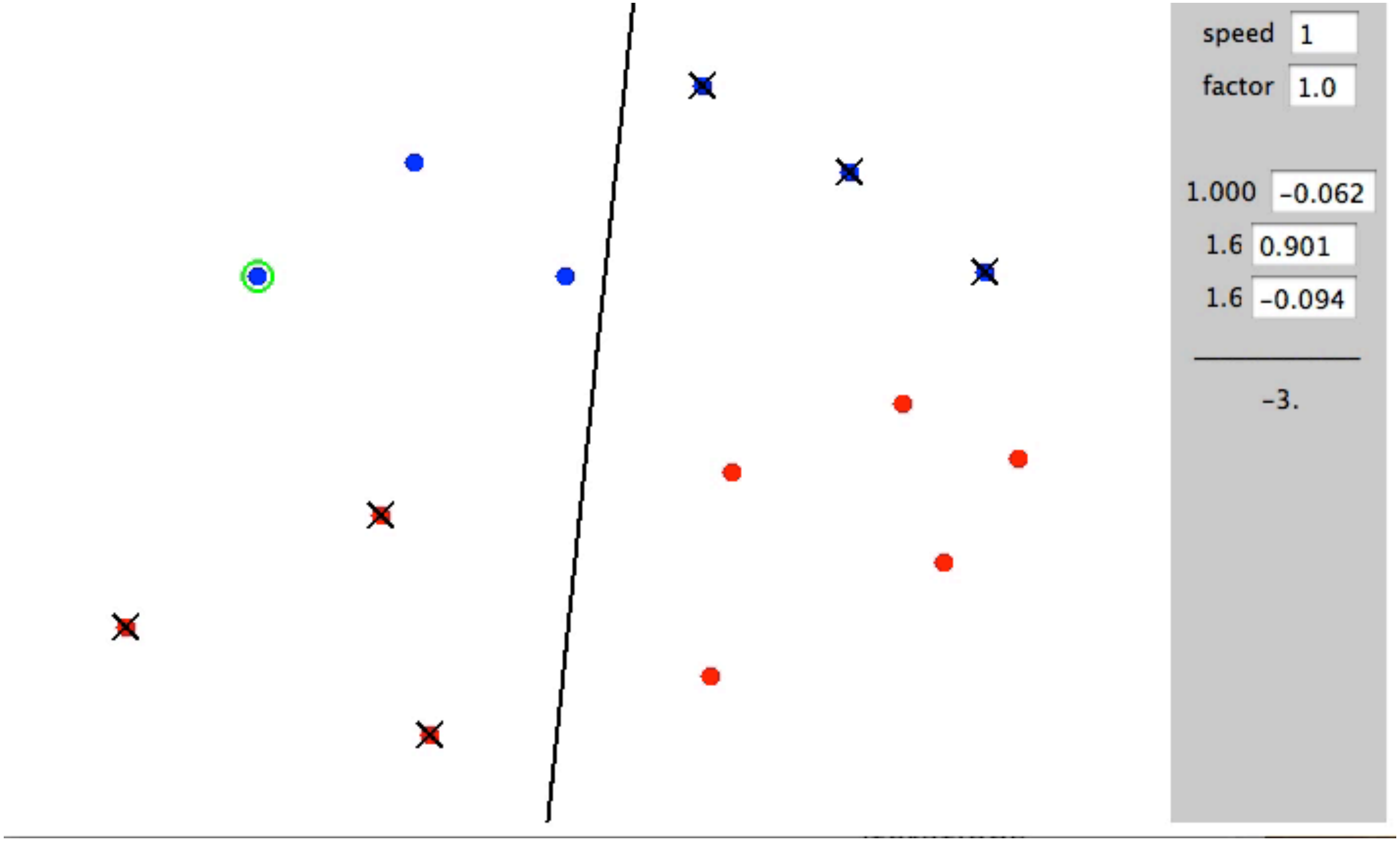
Examples: Perceptron

- Separable Case



Examples: Perceptron

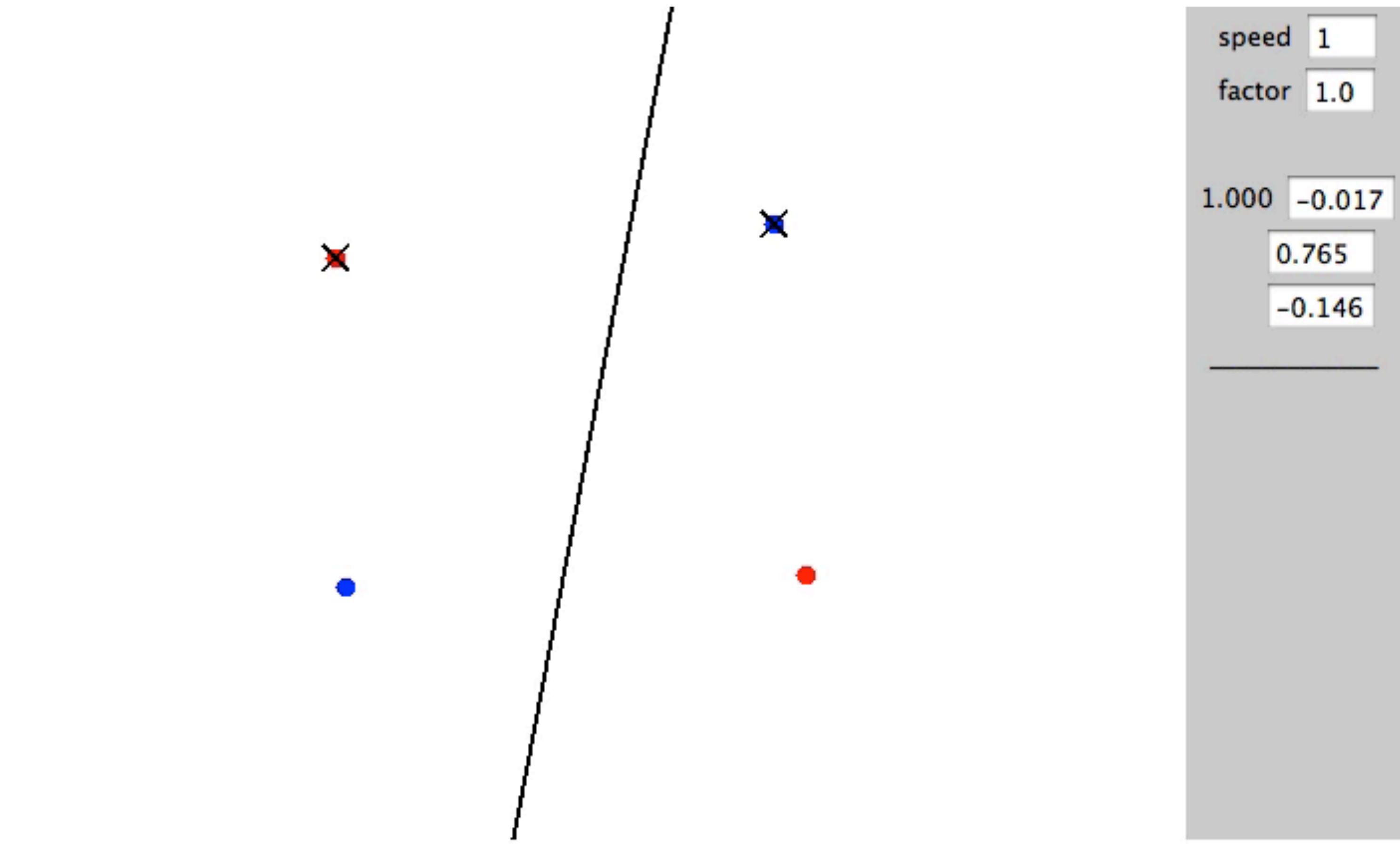
- Separable Case



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

Examples: Perceptron

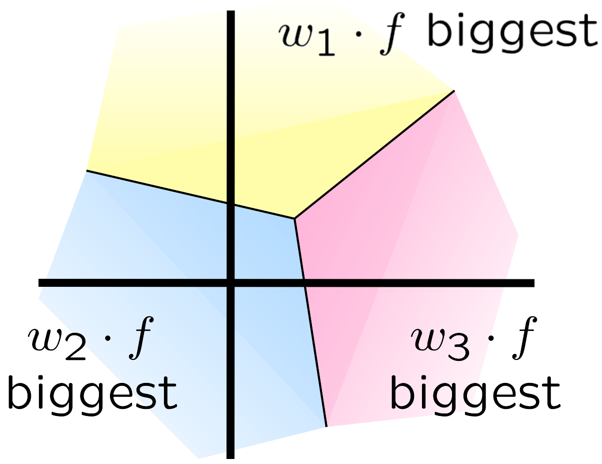
- Inseparable Case



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

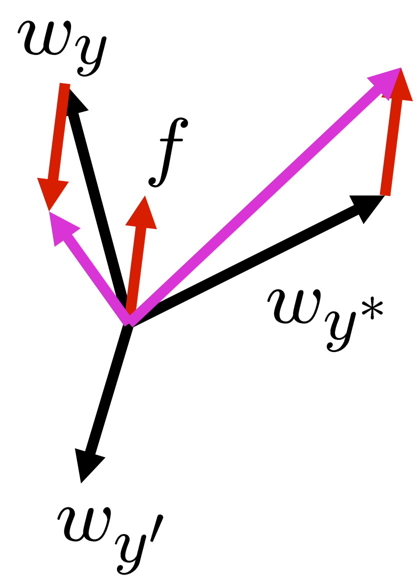
- Start with zero weights
- Iterate training examples
  - Classify with current weights

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

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

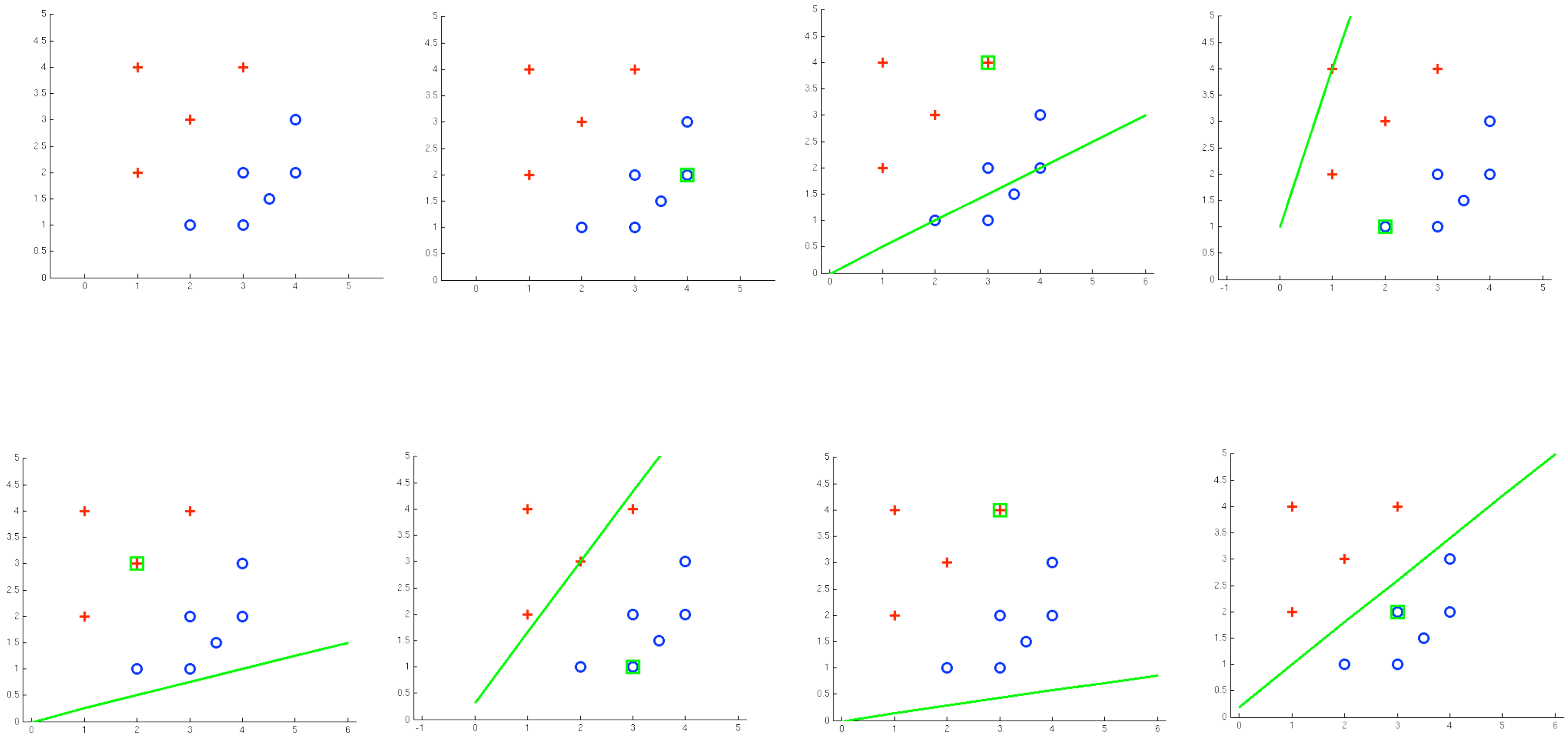
$$w_y = w_y - f(x)$$

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



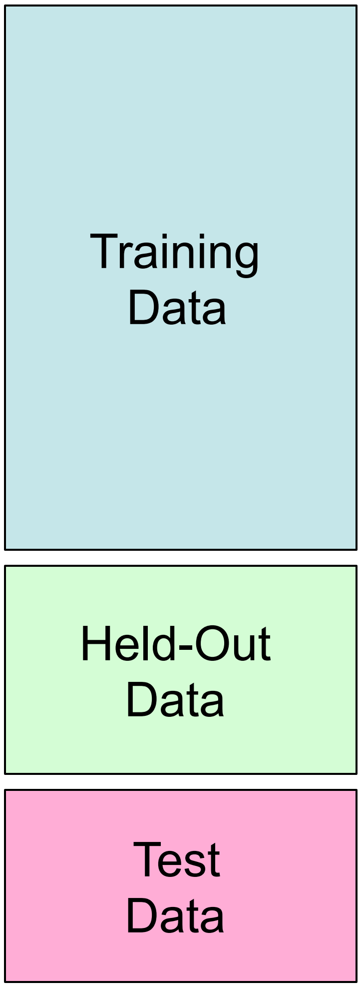
Examples: Perceptron

- Separable Case



Mistake-Driven Classification

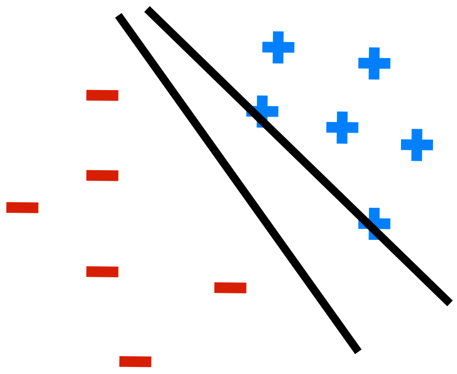
- For Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data
- For the perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until held-out accuracy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



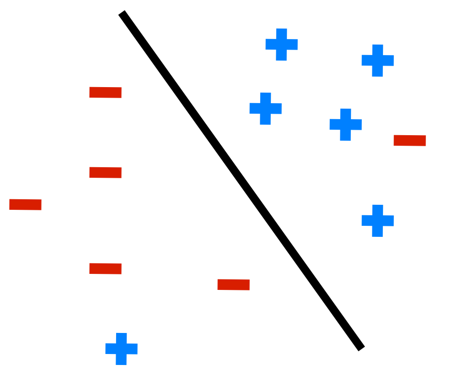
Properties of Perceptrons

- 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

Separable

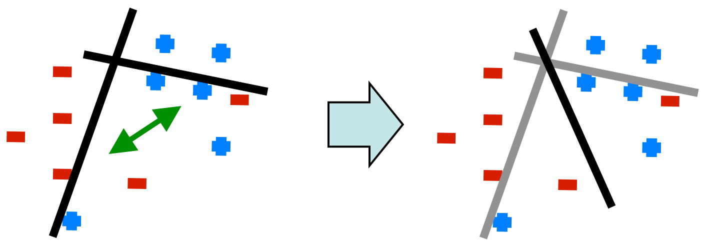


Non-Separable

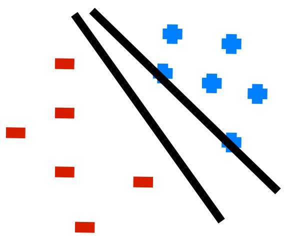


Problems with the Perceptron

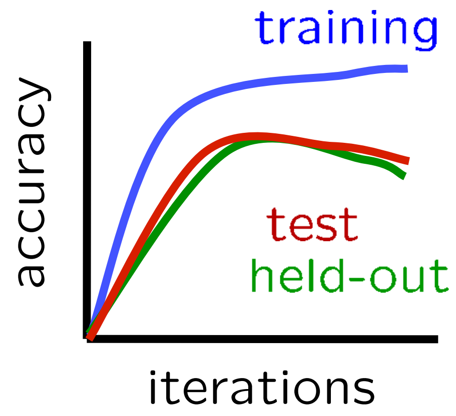
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- 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

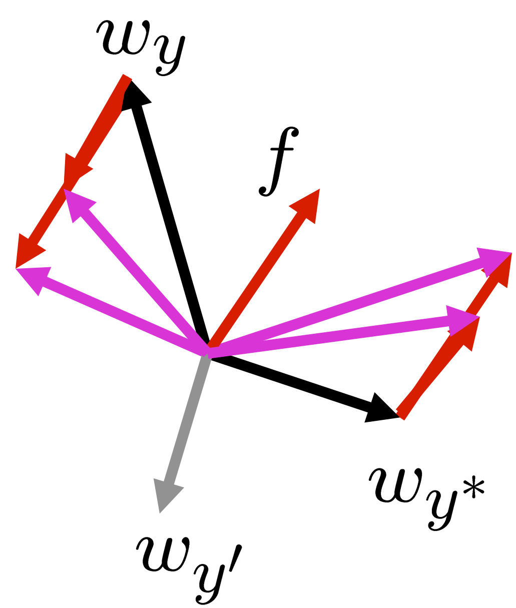
- Idea: adjust the weight update to mitigate these effects
- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to w

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

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

- 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

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

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

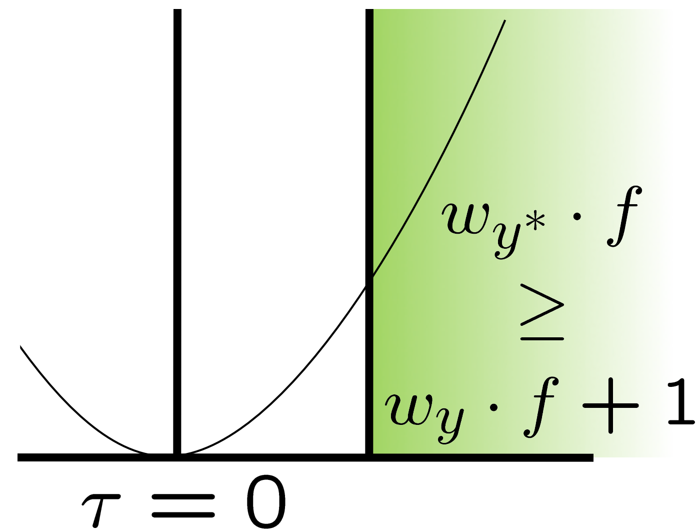
$$\min_\tau \ ||\tau f||^2$$

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

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

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

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

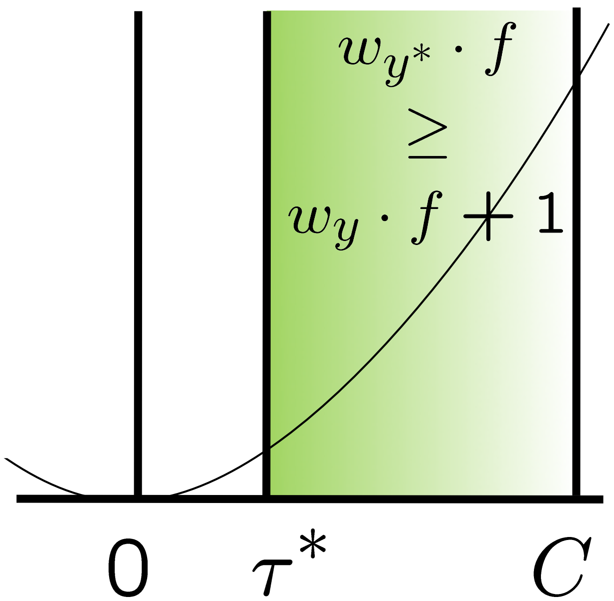


min not τ=0, or would not have made an error, so min will be where equality holds

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

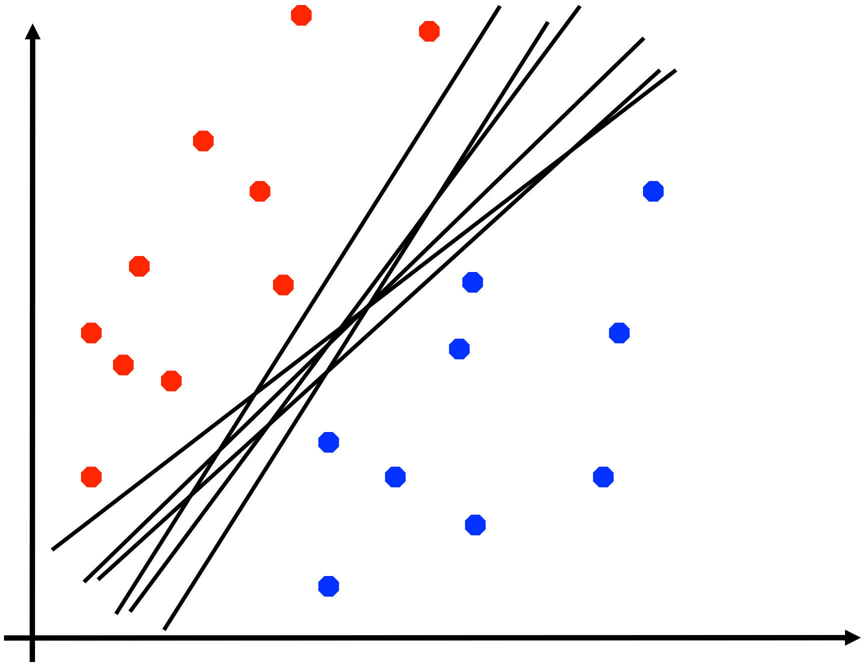
$$\tau^* = \min\left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2 f \cdot f}, C\right)$$



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

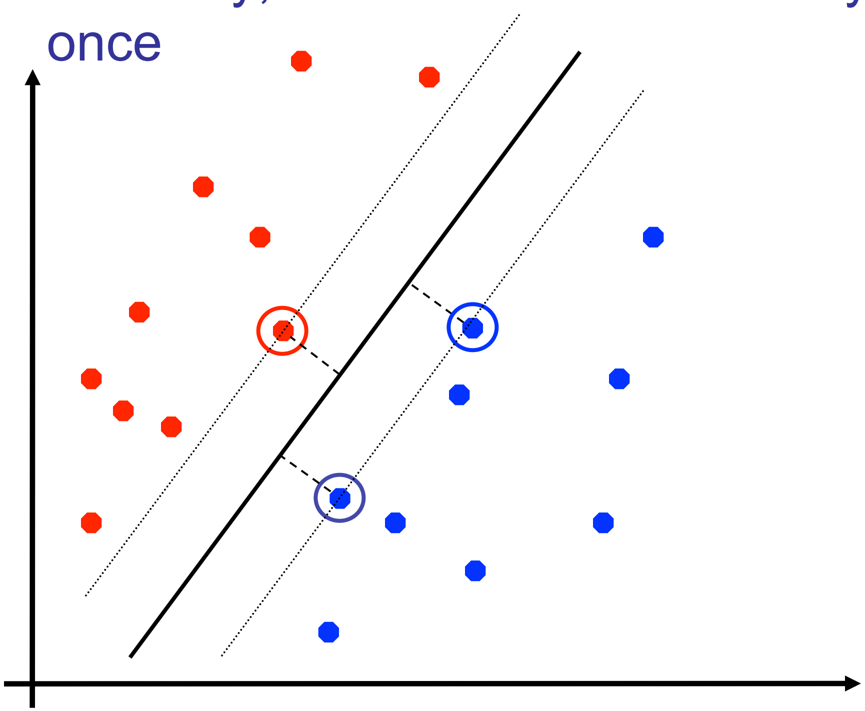
Linear Separators

- 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)

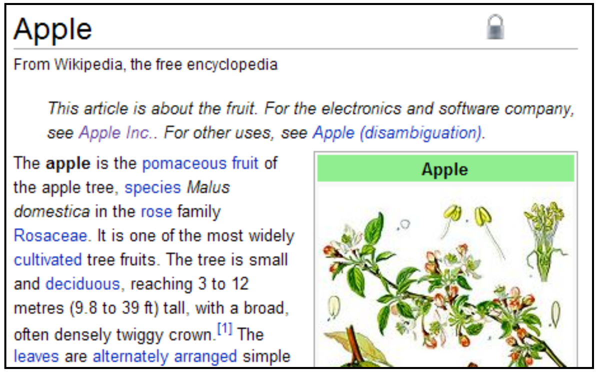
- Perceptrons / MIRA:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate

Extension: Web Search

x = "Apple Computers"

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

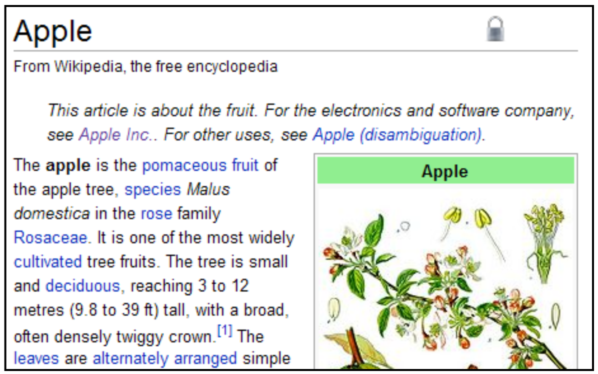
- Web search: not exactly classification, but rather ranking



Feature-Based Ranking

x = "Apple Computers"

$f($x,


Apple

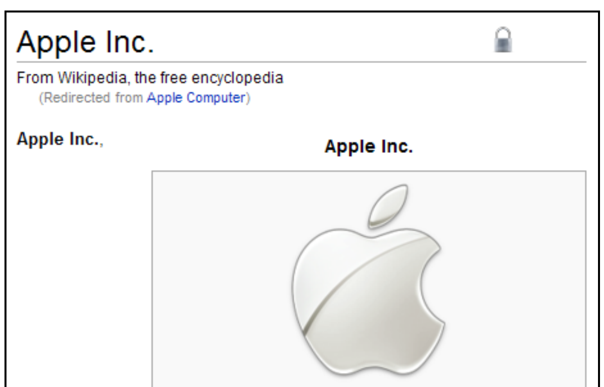
From Wikipedia, the free encyclopedia

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

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

$f($x,


Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

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

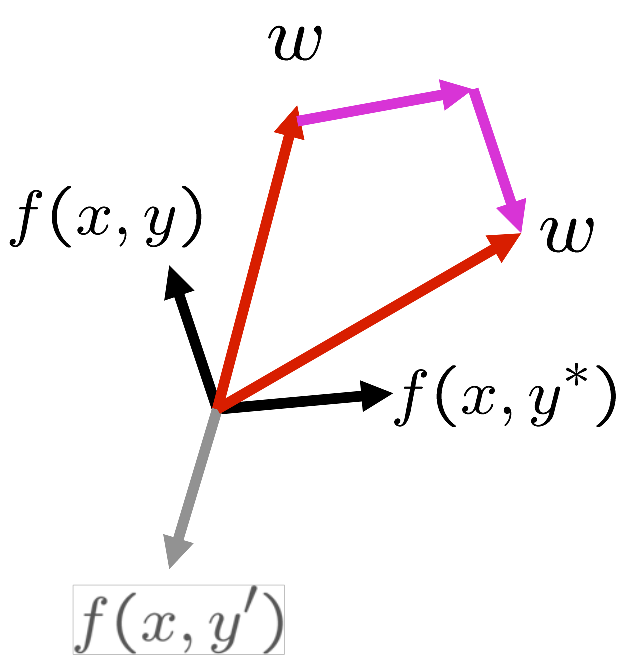
Perceptron for Ranking

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

$$y = \text{arg max}_y \; w \cdot f(x, y)$$

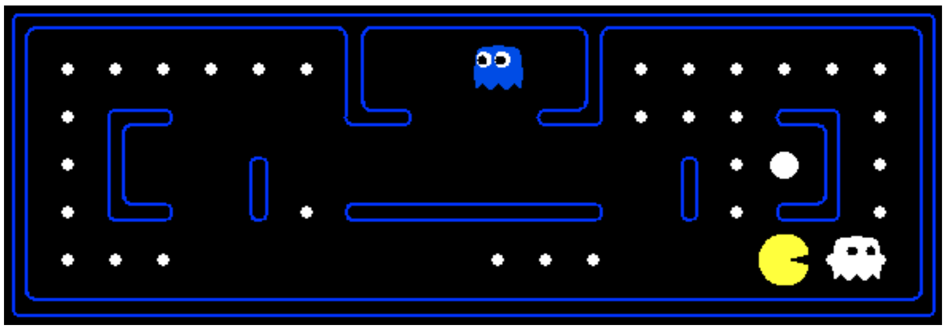
  - Update (if wrong):

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



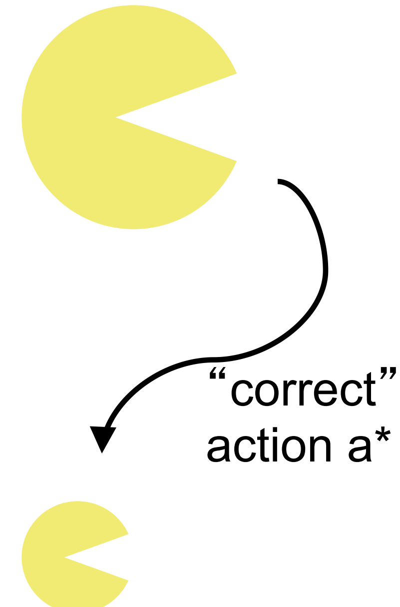
Pacman Apprenticeship!

- 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?