CS 188: Artificial Intelligence
Spring 2010

Lecture 24: Perceptrons and More!

4/20/2010

Pieter Abbeel – UC Berkeley

Slides adapted from Dan Klein

Announcements

- W7 due Thursday [that's your last written for the semester!]
- Project 5 out Thursday
- Contest running

Classification: Feature Vectors

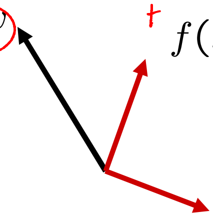
x	$f(x)$	y
Hello, Do you want free printr cartriges? Why pay more when you can get them ABSOLUTELY FREE! Just	# free : 2 YOUR_NAME : 0 MISSPELLED : 2 FROM_FRIEND : 0 ...	SPAM or +
	PIXEL-7,12 : 1 PIXEL-7,13 : 0 ... NUM_LOOPS : 1 ...	"2"

Classification: Weights

- Binary case: compare features to a weight vector
  $f(x)$ w $w^T f(x)$
- Learning: figure out the weight vector from examples

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

w



$+$ $f(x_1)$

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

$f(x_2)$ $-$

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

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

Learning: Binary Perceptron

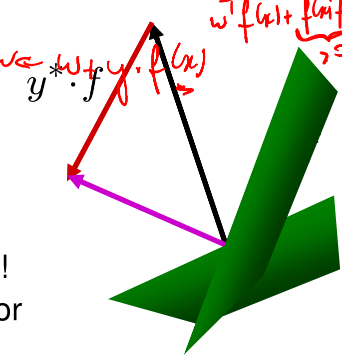
$w^T f(x)$ $(f(x),+)$

$w^T f(x) \to (w+f(x))^T f(x)$

$= w^T f(x) + \underbrace{f(x)^T f(x)}_{>0}$

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

$w \leftarrow w + y \cdot f(x)$



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

  - 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.

$$w = w + \underline{y^* \cdot f}$$

6

[Demo]

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

Multiclass Decision Rule

binary: w, $w^T f(x) > 0$

<0

$w_+^T f(x) \gtrless w_-^T f(x)$

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

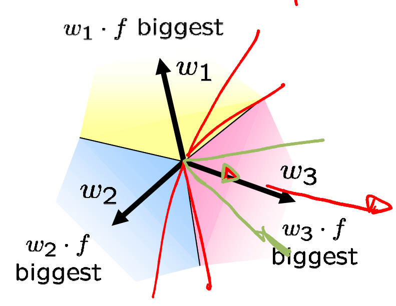
$$w_y$$

  - Score (activation) of a class y:

$$\underline{w_y} \cdot \underline{f(x)}$$

  - Prediction highest score wins

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



b

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

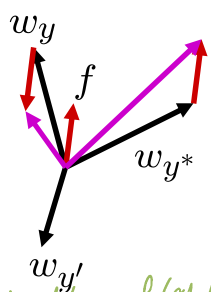
$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change! If correct by margin of 1
- If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x) \quad \rightarrow (w_y - f(x)) \cdot f(x) = w_y \cdot f(x) - f(x) \cdot f(x)$$

original; lower the score for label y on example x

$$w_{y^*} = w_{y^*} + f(x) \quad \rightarrow (w_{y^*} + f(x)) \cdot f(x) = w_{y^*} \cdot f(x) + f(x) \cdot f(x)$$



8

Example: Multiclass Perceptron

"win the vote"

"win the election"

"win the game"

w_{SPORTS}

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

$w_{POLITICS}$

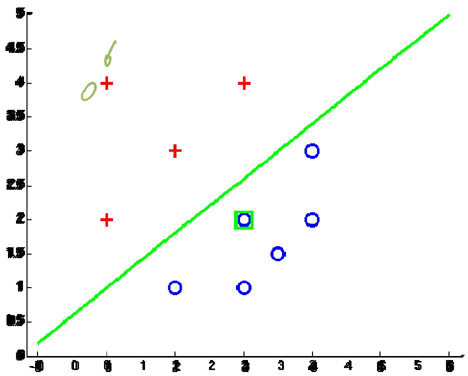
BIAS	0
win	0
game	0
vote	0
the	0
...	

w_{TECH}

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

Examples: Perceptron

- Separable Case



$w \mathrel{+}= f(x_i)$

Properties of Perceptrons

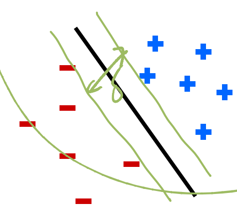
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\boxed{\text{mistakes} < \frac{R^2}{\delta^2}}$$

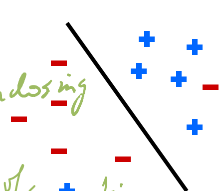
R = radius of a sphere enclosing all f(x)

δ = margin of separation

Separable

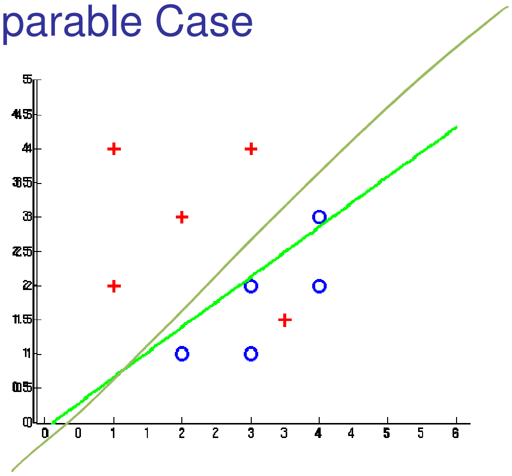


Non-Separable



Examples: Perceptron

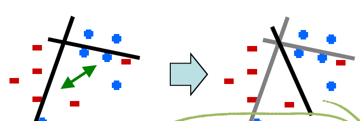
- Non-Separable Case



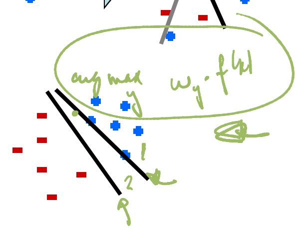
13

Problems with the Perceptron

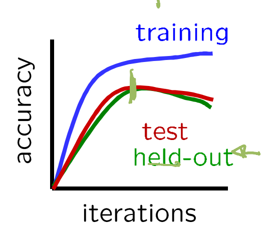
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution



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



$w = \begin{pmatrix} w_{y_1} \\ w_{y_2} \\ \vdots \\ w_{y_k} \end{pmatrix}$

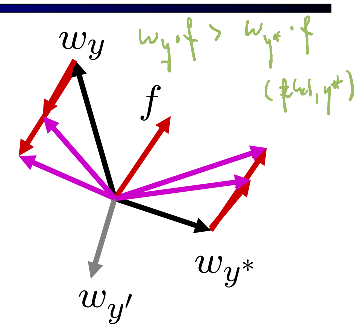
Fixing the Perceptron

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

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

$$\underbrace{w_{y^*} \cdot f(x)}_{\text{score for } y^*} \geq \underbrace{w_y \cdot f(x)}_{\text{score for } y} + \underbrace{1}_{+1}$$

- 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

constrained optimization cs 127a / cs 227a

Minimum Correcting Update

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

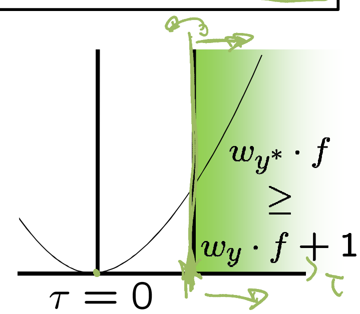
$$\min_\tau ||\tau f||^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$

(1)

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

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

$$\boxed{\begin{aligned} w_y &= w'_y - \tau f(x) \\ w_{y^*} &= w'_{y^*} + \tau f(x) \end{aligned}} \quad (1)$$

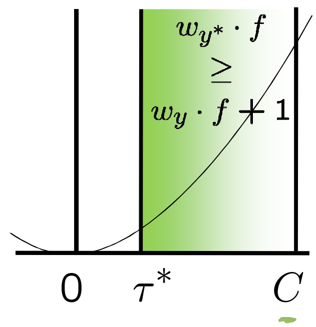


min not τ=0, or would not have made an error, so min will be where equality holds

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

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

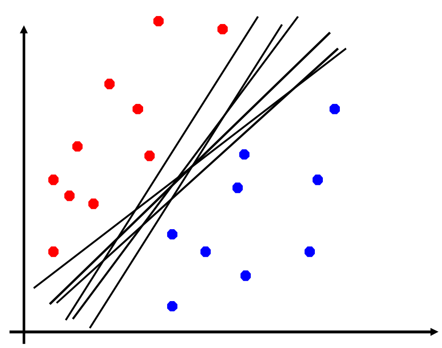


  - Corresponds to an optimization that assumes non-separable data
  - Usually converges faster than perceptron
  - Usually better, especially on noisy data

17

Linear Separators

- Which of these linear separators is optimal?



18

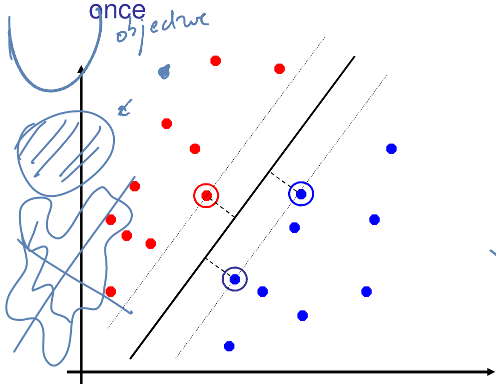
Max margin linear separator

Support Vector Machines

$w_{*}\cdot f(x) > w_y f(x)$

- 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

objective



MIRA

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

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

SVM

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

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

(*)

Separable case

hard margin SVM: $\min_w \frac{1}{2}||w||_2^2$

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

Inseparable case

soft margin SVM: $\min_{w,\xi} \frac{1}{2}||w||_2^2 + C\sum_i \xi_i$

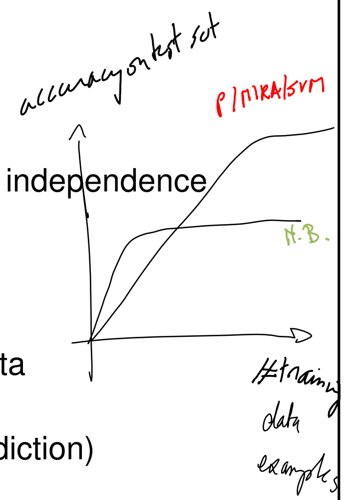
s.t. $\forall i, \forall y \neq y^* \quad w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1 - \xi_i$

$\forall i \quad \xi_i \geq 0$

slack variable

Classification: Comparison

- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)
- Perceptrons / MIRA / SVM:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate

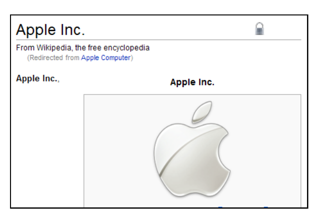


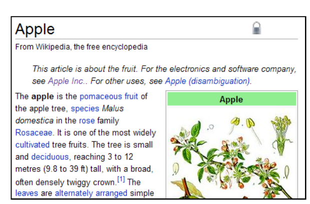
Extension: Web Search

(*)

- 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

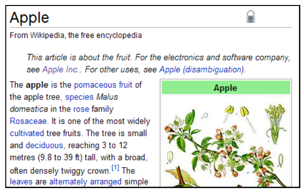
x = "Apple Computers"





Feature-Based Ranking

x = "Apple Computers"

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

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

---

$f($ *x*,  $) = [0.8\ 4\ 2\ 1\ \ldots]$

Apple Inc.

From Wikipedia, the free encyclopedia

(Redirected from Apple Computer)

Apple Inc.,

Perceptron for Ranking

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

posit

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

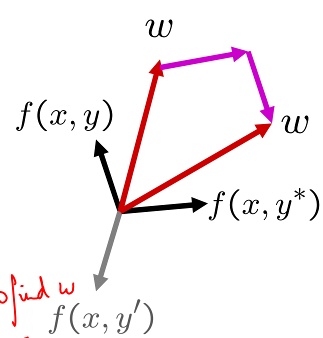
to find w

ML to ensure predictions are good

  - Update (if wrong):

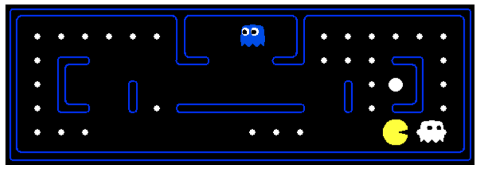
"perceptron"

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



Pacman Apprenticeship! (*)

- Examples are states s



human player

player w/ more computational cycles

- 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:

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

choose $\arg\max_a \; w \cdot f(s,a) \approx q(s,a)$

- How is this VERY different from reinforcement learning?