

CS 188: Artificial Intelligence Spring 2010

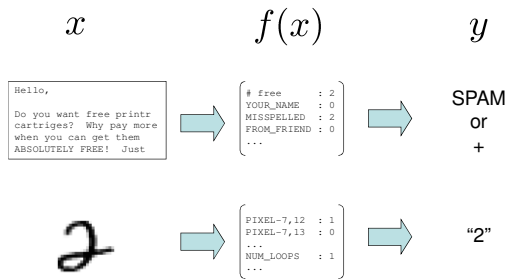
Lecture 24: Perceptrons and More! 4/20/2010

Pieter Abbeel – UC Berkeley
Slides adapted from Dan Klein

Announcements

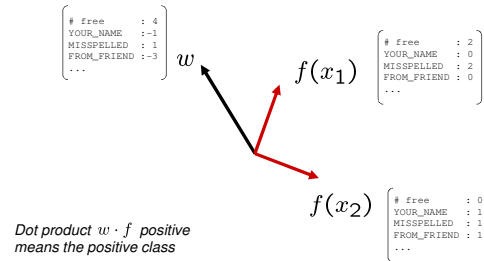
- W7 due Thursday [that's your last written for the semester!]
- Project 5 out Thursday
- Contest running

Classification: Feature Vectors



Classification: Weights

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



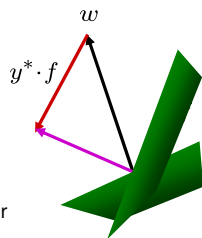
Learning: Binary Perceptron

- 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.

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



[demo] http://ml.ira.uka.de/neuralNetCourse2004/VL_11_5/Perceptron.html

Multiclass Decision Rule

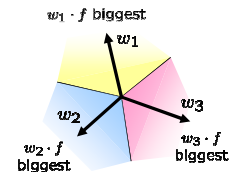
- If we have multiple classes:
 - A weight vector for each class: w_y

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

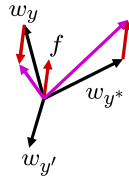
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

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

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

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



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
...
```

$w_{POLITICS}$

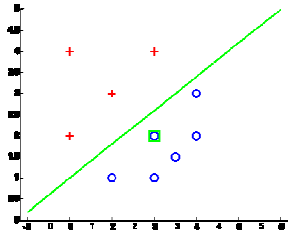
```
BIAS : 0
win  : 0
game : 0
vote : 0
the  : 0
...
```

w_{TECH}

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

Examples: Perceptron

- Separable Case



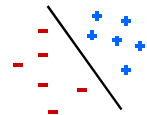
10

Properties of Perceptrons

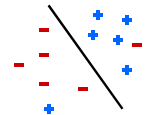
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the margin or degree of separability

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

Separable



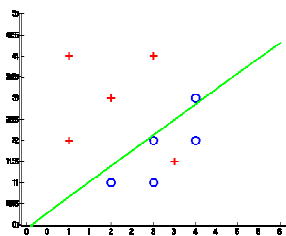
Non-Separable



12

Examples: Perceptron

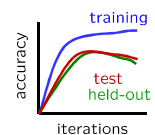
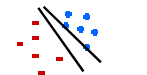
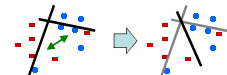
- Non-Separable Case



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



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

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

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

$$\min_w \|\tau f\|^2$$

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

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

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

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

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

$\tau = 0$

min not $\tau=0$, 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

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

20

Extension: Web Search

- Information retrieval:

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

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



- Web search: not exactly classification, but rather ranking

Feature-Based Ranking

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

$$f(x, \text{Screenshot of Apple Inc. result}) = [0.3 \ 5 \ 0 \ 0 \ \dots]$$

$$f(x, \text{Screenshot of Apple result}) = [0.8 \ 4 \ 2 \ 1 \ \dots]$$

Perceptron for Ranking

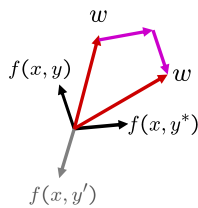
- Inputs x
- Candidates y
- Many feature vectors: $f(x, y)$
- One weight vector: w

- 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:

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

- How is this VERY different from reinforcement learning?

