CS 188: Artificial Intelligence

Lecture 21: Perceptrons

Pieter Abbeel – UC Berkeley
Many slides adapted from Dan Klein.

Outline

- Generative vs. Discriminative
- Binary Linear Classifiers
- Perceptron
- Multi-class Linear Classifiers
- Multi-class Perceptron
- Fixing the Perceptron: MIRA
- Support Vector Machines*

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"

Generative vs. Discriminative

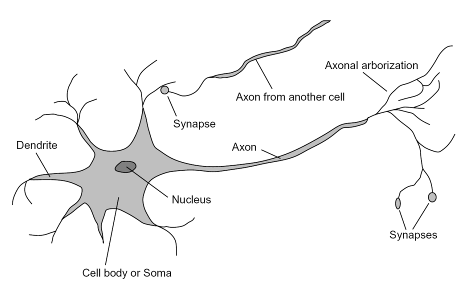
- Generative classifiers:
  - E.g. naïve Bayes
  - A causal model with evidence variables
  - Query model for causes given evidence
- Discriminative classifiers:
  - No causal model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Robust, accurate with varied features
  - Loosely: mistake driven rather than model driven

Outline

- Generative vs. Discriminative
- ***Binary Linear Classifiers***
- Perceptron
- Multi-class Linear Classifiers
- Multi-class Perceptron
- Fixing the Perceptron: MIRA
- Support Vector Machines*

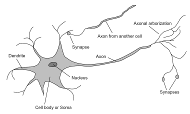
Some (Simplified) Biology

- Very loose inspiration: human neurons



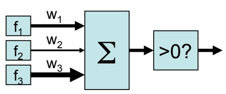
Linear Classifiers

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



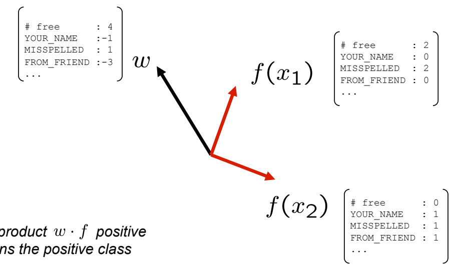
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

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



Classification: Weights

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



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

Linear Classifiers Mini Exercise

$f(x_1) = \begin{bmatrix} \texttt{\# free} : 2 \\ \texttt{YOUR\_NAME} : 0 \end{bmatrix}$ $f(x_2) = \begin{bmatrix} \texttt{\# free} : 4 \\ \texttt{YOUR\_NAME} : 1 \end{bmatrix}$ $f(x_3) = \begin{bmatrix} \texttt{\# free} : 1 \\ \texttt{YOUR\_NAME} : 1 \end{bmatrix}$

$w = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

- 1. Draw the 4 feature vectors and the weight vector w
- 2. Which feature vectors are classified as +? As - ?
- 3. Draw the line separating feature vectors being classified + and -.

Linear Classifiers Mini Exercise 2 --- Bias Term

$f(x_1) = \begin{bmatrix} \texttt{Bias} : 1 \\ \texttt{\# free} : 2 \\ \texttt{YOUR\_NAME} : 0 \end{bmatrix}$ $f(x_2) = \begin{bmatrix} \texttt{Bias} : 1 \\ \texttt{\# free} : 4 \\ \texttt{YOUR\_NAME} : 1 \end{bmatrix}$ $f(x_3) = \begin{bmatrix} \texttt{Bias} : 1 \\ \texttt{\# free} : 1 \\ \texttt{YOUR\_NAME} : 1 \end{bmatrix}$

$w = \begin{bmatrix} -3 \\ -1 \\ 2 \end{bmatrix}$

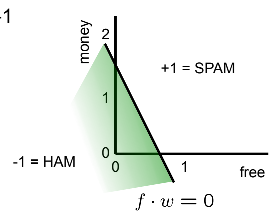
- 1. Draw the 4 feature vectors and the weight vector w
- 2. Which feature vectors are classified as +? As - ?
- 3. Draw the line separating feature vectors being classified + and -.

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
...
```



Outline

- Generative vs. Discriminative
- Binary Linear Classifiers
- ***Perceptron: how to find the weight vector w from data.***
- Multi-class Linear Classifiers
- Multi-class Perceptron
- Fixing the Perceptron: MIRA
- Support Vector Machines*

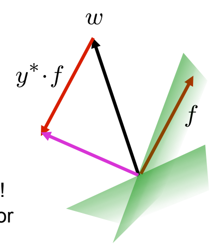
Binary Perceptron Update

- 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.

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



[demo]

18

Outline

- Generative vs. Discriminative
- Binary Linear Classifiers
- Perceptron
- ***Multi-class Linear Classifiers***
- Multi-class Perceptron
- Fixing the Perceptron: MIRA
- Support Vector Machines*

Multiclass Decision Rule

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

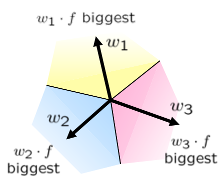
$$w_y$$

  - 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

Example Exercise --- Which Category is Chosen?

"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
...		

Exercise: Multiclass linear classifier for 2 classes and binary linear classifier

- Consider the multiclass linear classifier for two classes with $w_1 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ $w_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
- Is there an equivalent binary linear classifier, i.e., one that classifies all points x = (x_1, x_2) the same way?

Outline

- Generative vs. Discriminative
- Binary Linear Classifiers
- Perceptron
- Multi-class Linear Classifiers
- ***Multi-class Perceptron: learning the weight vectors*** w_i ***from data***
- Fixing the Perceptron: MIRA
- Support Vector Machines*

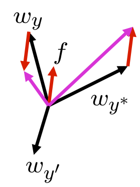
Learning Multiclass Perceptron

- Start with zero weights
- Pick up training instances one by one
- Classify with current weights

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

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

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



24

Example

"win the vote"

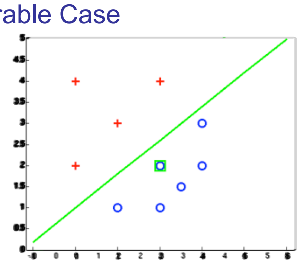
"win the election"

"win the game"

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

Examples: Perceptron

- Separable Case



26

Outline

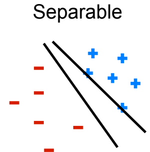
- Generative vs. Discriminative
- Binary Linear Classifiers
- Perceptron
- Multi-class Linear Classifiers
- Multi-class Perceptron: learning the weight vectors w_i from data
- ***Fixing the Perceptron: MIRA***
- Support Vector Machines*

Properties of Perceptrons

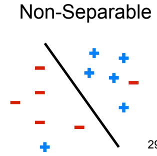
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

Separable



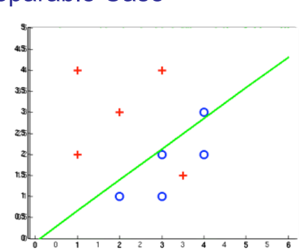
Non-Separable



29

Examples: Perceptron

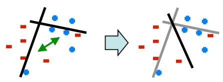
- Non-Separable Case



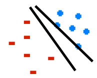
30

Problems with the Perceptron

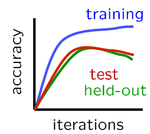
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution



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



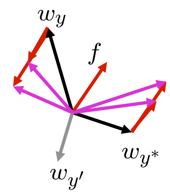
Fixing the Perceptron

- 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



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

Minimum Correcting Update

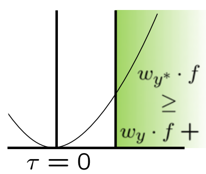
$$\min_w \frac{1}{2}\sum_y ||w_y - w'_y||^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$

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

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

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

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



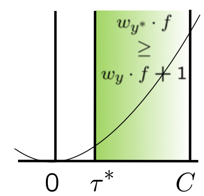
min not τ=0, or would not have made an error, so min will be where equality holds

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

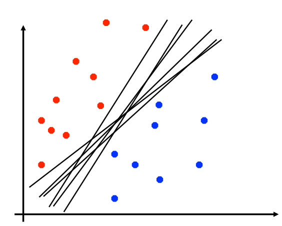


Outline

- Generative vs. Discriminative
- Binary Linear Classifiers
- Perceptron
- Multi-class Linear Classifiers
- Multi-class Perceptron: learning the weight vectors w_i from data
- Fixing the Perceptron: MIRA
- ***Support Vector Machines****

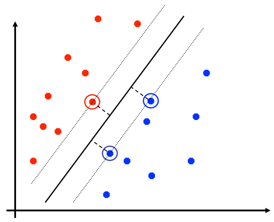
Linear Separators

- 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

$$\min_w \frac{1}{2}||w||^2$$
$$\forall i, y \; w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

Mini-Exercise: Give Example Dataset that Would be Overfit by SVM, MIRA and running perceptron till convergence

- Could running perceptron less steps lead to better generalization?

Classification: Comparison

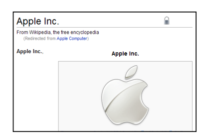
- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)
- Perceptrons / MIRA:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)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

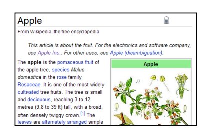
40

Extension: Web Search

x = "Apple Computers"

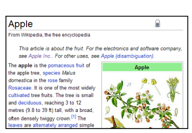
- Information retrieval:
  - Given information needs, produce information
  - Includes, e.g. web search, question answering, and classic IR
- Web search: not exactly classification, but rather ranking

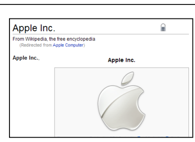




Feature-Based Ranking

x = "Apple Computers"

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

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

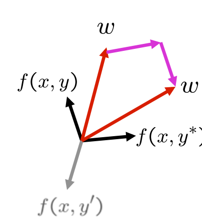
Perceptron for Ranking

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

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

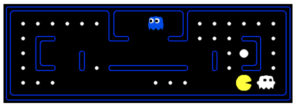
  - Update (if wrong):

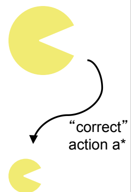
$$w = w + f(x, y^*) - f(x, y)$$



Pacman Apprenticeship!

- Examples are states s





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

- 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?