CS 188: Artificial Intelligence
Fall 2008

Lecture 25: Perceptron

4/21/2008

John DeNero – UC Berkeley

Slides from Dan Klein

Announcements

- Project 4 due tomorrow
  - Use up to two slip days
  - Issues with the autograder are resolved
  - What tracking multiple ghosts should look like
- Written assignment 4 posted
  - Shortest written assignment ever!
  - Due next Thursday at the beginning of lecture
  - Turn it in on time

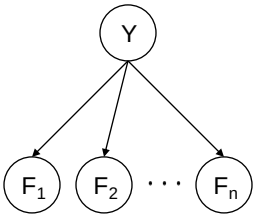
General Naïve Bayes

- A general *naïve Bayes* model:
  - Y: label to be predicted
  - $F_1, \ldots, F_n$: features of each instance

$$P(Y, F_1 \ldots F_n) = P(Y) \prod_i P(F_i|Y)$$

|Y| parameters

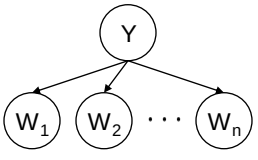
n x |F| x |Y| parameters



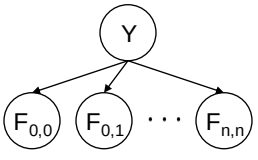
- We only specify how each feature depends on the class
- Total number of parameters is *linear* in n

Example Naïve Bayes Models

- Bag of words for text
  - One feature for every word position in the document
  - All features share the same conditional distributions
  - Maximum likelihood estimates: word frequencies, by label

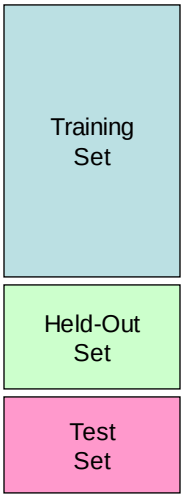


- Pixels for digit recognition
  - One feature for every pixel, indicating whether it is on (black)
  - Each pixel has a different conditional distribution
  - Maximum likelihood estimates: how often a pixel is on, by label



Naïve Bayes Classification

- Data: labeled instances, e.g. emails marked as spam/ham by a person
  - Divide into training, held-out and test
- Features are known for every training, held-out and test instance
- Estimation: count feature values in the training set and normalize to get maximum likelihood estimates of probabilities
- Smoothing (aka regularization): adjust estimates to account for unseen data



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
- Laplace for conditionals:
  - Smooth each condition:
  - Can be derived by dividing

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

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

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

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

Linear Interpolation Smoothing

- Linear interpolation for conditional likelihoods
  - **Idea**: the conditional probability of a feature x given a label y should be close to the marginal probability of x
  - **Example**: A rare word like "interpolation" should be similarly rare in both ham and spam
  - **Procedure**: Collect relative frequency estimates of both conditional and marginal, then average

$$P_{ML}(x|y) = \frac{\text{count}(x,y)}{\text{count}(y)} \qquad P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{LIN}(x|y) = \alpha P_{ML}(x|y) + (1-\alpha) P_{ML}(x)$$

  - **Effect**: Features have odds ratios closer to 1

7

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})} \qquad \frac{P(W|\text{spam})}{P(W|\text{ham})}$$

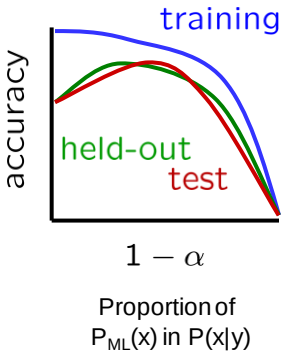
helvetica	: 11.4
seems	: 10.8
group	: 10.2
ago	: 8.4
areas	: 8.3
...	

verdana	: 28.8
Credit	: 28.4
ORDER	: 27.2
<FONT>	: 26.9
money	: 26.5
...	

Do these make more sense?

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: $P(F_i|Y)$ and $P(Y)$
  - Hyperparameters, like the amount of smoothing to do: k, α
- Where to learn which unknowns
  - Learn parameters from training set
  - Must tune hyperparameters on different data (why?)
  - For each possible 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

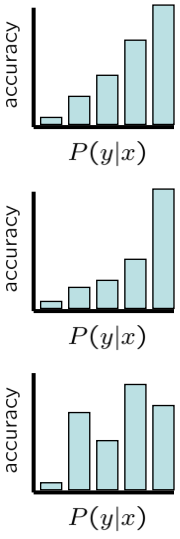
- First task when classifying: get a baseline
  - Baselines are very simple "straw man" procedures
  - Help determine how hard the task is
  - Help know what a "good" accuracy is
- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as spam
  - Accuracy might be very high if the problem is skewed
- When conducting real research, we usually use previous work as a (strong) baseline

Confidences from a Classifier

- The confidence of a classifier:
  - Posterior over the most likely 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
  - Strong calibration: confidence predicts accuracy rate
  - Weak calibration: higher confidences mean higher accuracy
  - What's the value of calibration?



Naïve Bayes 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
- Confidences are useful when the classifier is calibrated

What to Do About Errors

- Problem: there's still spam in your inbox
- 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?
- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)

14

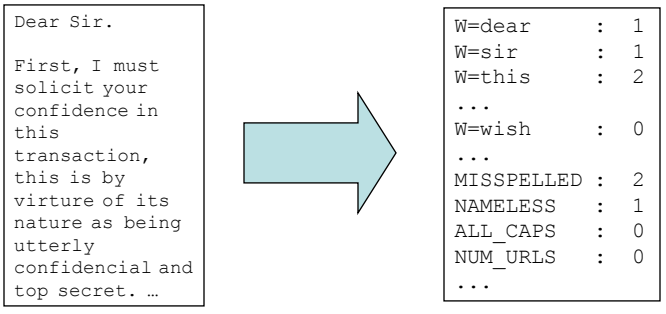
Features

- A feature is a function that signals a property of the input
  - **Naïve Bayes**: features are random variables & each value has conditional probabilities given the label.
  - **Most classifiers**: features are real-valued functions
  - **Common special cases**:
    - Indicator features take values 0 and 1 or -1 and 1
    - Count features return non-negative integers
- Features are anything you can think of for which you can write code to evaluate on an input
  - Many are cheap, but some are expensive to compute
  - Can even be the output of another classifier or model
  - Domain knowledge goes here!

15

Feature Extractors

- A feature extractor maps inputs to feature vectors



- Many classifiers take feature vectors as inputs
- Feature vectors usually very sparse, use sparse encodings (i.e. only represent non-zero keys)

16

Generative vs. Discriminative

- Generative classifiers:
  - E.g. naïve Bayes
  - We build a causal model of all the variables, including observed X
  - We then query that model for causes, given evidence

$$P(X,Y)$$

Who cares about P(X) and P(X|Y)?

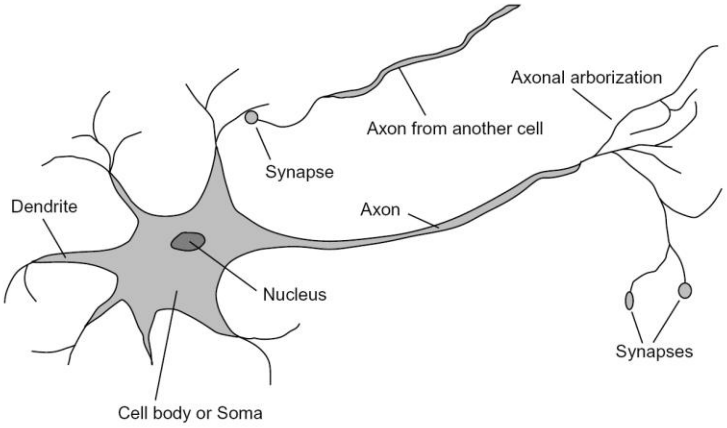
- Discriminative classifiers:
  - No causal model, no Bayes rule, often no probabilities at all!
  - Try to predict the label Y directly from X
  - Loosely: mistake driven rather than model driven

$$P(Y|X)$$

17

Some (Vague) Biology

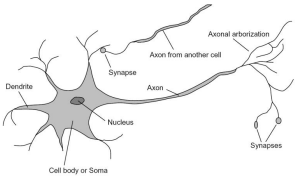
- Very loose inspiration: human neurons



18

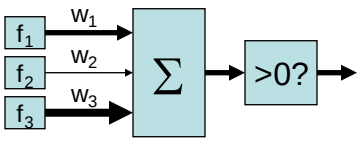
Linear Classifiers

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



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

- If the activation is:
  - Positive, output 1
  - Negative, output 0



19

Example: Spam

- Imagine 4 features:
  - Free (number of occurrences of "free")
  - Money (occurrences of "money")
  - BIAS (always has value 1)
  - the (occurrences of "the")

$$w \cdot f(x)$$

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

x

"free money"

$f(x)$

```
BIAS  :  1
free  :  1
money :  1
the   :  0
...
```

w

```
BIAS  : -3
free  :  4
money :  2
the   :  0
...
```

$$(1)(-3) + (1)(4) + (1)(2) + (0)(0) + \ldots = 3$$

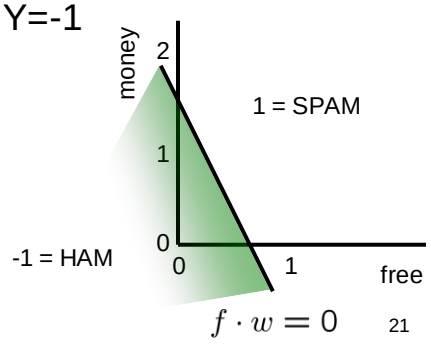
20

Binary Decision Rule

- In the space of feature vectors
  - Any weight vector is a hyperplane
  - One side corresponds to Y=1
  - Other corresponds to Y=-1

w

```
BIAS  : -3
free  :  4
money :  2
the   :  0
...
```



21

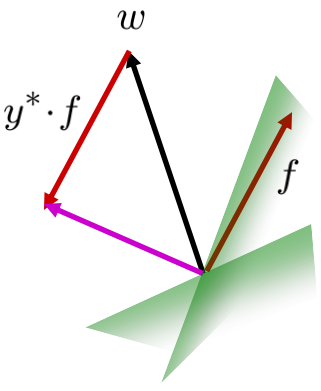
Binary Perceptron Update

- 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.

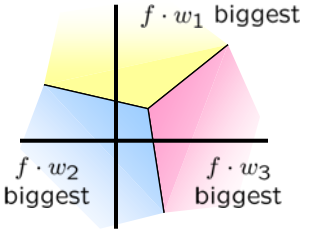
$$w = w + y^* \cdot f$$



[Demo]

Multiclass Decision Rule

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



$$\text{activation}_w(x, y) = \sum_i w_{y,i} \cdot f_i(x)$$

  - Highest activation wins

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

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
...
```

24

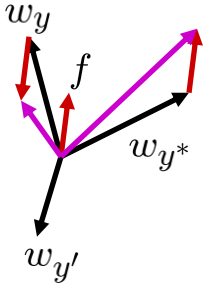
The Perceptron Update Rule

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

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

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

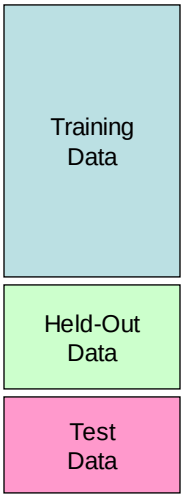
$$w_y = w_y - f(x)$$
$$w_{y^*} = w_{y^*} + f(x)$$



25

Mistake-Driven Classification

- In naïve Bayes, parameters:
  - From data statistics
  - Have a causal interpretation
  - One pass through the data
- For the perceptron parameters:
  - From reactions to mistakes
  - Have a discriminative interpretation
  - Go through the data until held-out accuracy maxes out



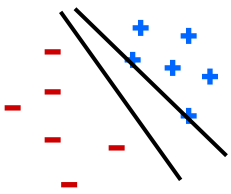
29

Properties of Perceptrons

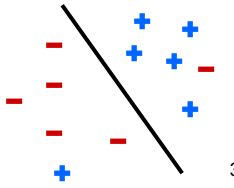
- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{1}{\delta^2}$$

Separable



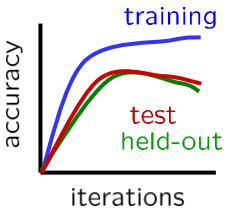
Non-Separable

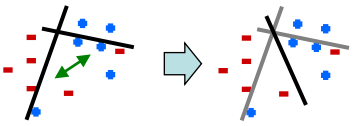


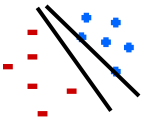
30

Issues with Perceptrons

- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining isn't quite as bad as overfitting, but is similar
- Regularization: if the data isn't separable, weights might thrash around
  - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a "barely" separating solution







33