Text Classification

CISC689/489, Lecture #16

Wednesday, April 15th

Ben Carterette

Relevance Feedback Review

- User submits a query
- System gets some feedback
  - True feedback: user sees ranked results, marks some relevant and some not relevant
  - Blind feedback: system assumes top *k* are relevant
- System uses the feedback to re-compute document scores
  - Query expansion: system adds "relevant terms" to the query to form a new query

Feedback to Classification

- An alternative way to think about relevance feedback:
  - The feedback provides "class" information about the documents
  - The system uses the class information to learn a classifier: relevant class versus nonrelevant class
  - It uses the learned classifier to predict the relevance of unjudged documents
- Look at it as a classification problem

Feedback in the BIM

For term i:

	Relevant	Non-relevant	Total
$d_i = 1$	r_i	$n_i - r_i$	n_i
$d_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - r_i$
Total	R	$N - R$	N

Number of relevant documents that contain term i

Number of relevant documents

Number of documents

Number of documents that contain term i

$$p_i = P(d_i|R) = (r_i + 0.5)/(R + 1)$$
$$s_i = P(d_i|NR) = (n_i - r_i + 0.5)/(N - R + 1)$$

Gives BIM feedback scoring function:

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

Other Classification Problems in IR

- Filtering:
  - Spam filtering: spam class versus non-spam class
  - Standing queries:
    - You give the system a query
    - As the system obtains new documents, it scores them against the query
    - If they are scored high enough, you will see them
    - Low-scoring documents are filtered out

Other Classification Problems in IR

- Categorization:
  - Classify documents by subject (news/sports/…)
  - Or by language (English/French/Arabic/…)
  - Or by "sentiment" (positive/negative/neutral)
  - Or by reading level (1st grade/8th grade/12th grade)
  - …

Classification Problem Definition

- General statement:
  - Given:
    - a description of an instance x in X, where X is the *instance space*
    - a fixed set of classes C = $\{c_1, c_2, \ldots, c_k\}$
  - Determine:
    - the class x belongs to: f(x) in C, where f(x) is a *classification function* with domain X and range C
- Binary classification:
  - Two classes: C = {positive, negative} or C = {+, -} or C = {1, 0}
- IR classification problems can often be treated as binary
  - Instances are documents
  - Classes are relevant/nonrelevant (or spam/nonspam, …)

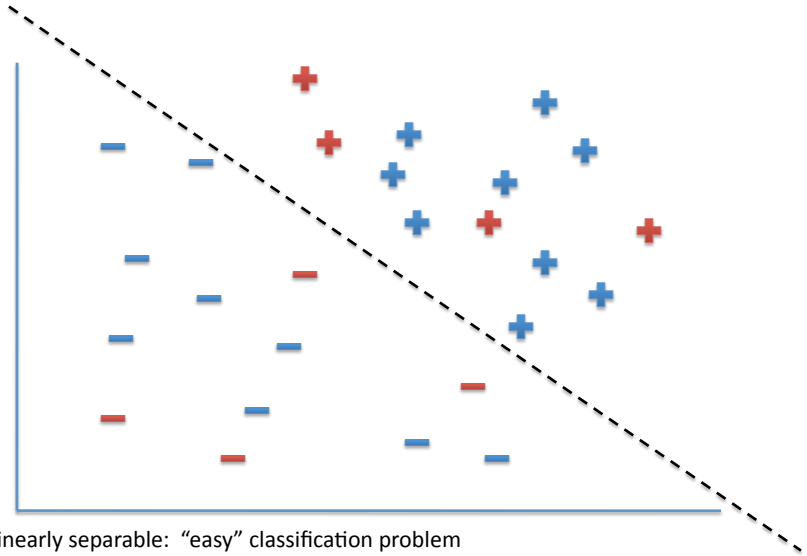
Classification Methods

- Manual classification:
  - An expert classifies each document
  - Yahoo hierarchy, Open Directory Project, etc.
  - Accurate but very expensive

Classification Methods

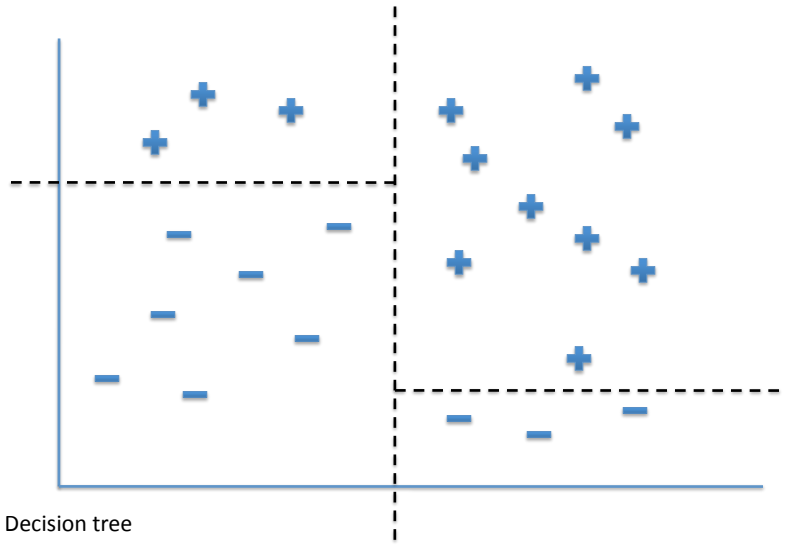
- Automatic:
  - Rule-based classification:
    - Check a Boolean sentence of rules
    - If sentence is true, classify as positive, otherwise as negative
    - Example: "IF (subject IS NULL AND (sender IS NOT ON whitelist OR sender IS ON blacklist)) THEN message = spam"
  - Machine learned classifier:
    - Use a set of "training instances" that have been manually classified to "learn" a classification function f(x)
    - Apply learned function to new instances

Classification Illustration



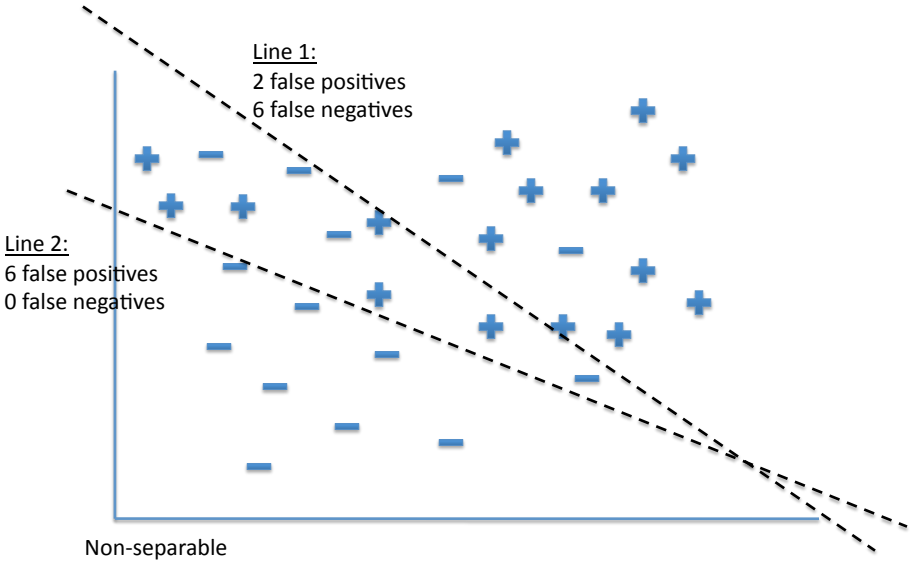
Linearly separable: "easy" classification problem

Classification Illustration



Decision tree

Classification Illustration



Non-separable

Probabilistic Binary Classification

- Given instance x, calculate P(C | x)
  - P(not C | x) = 1 – P(C | x)
  - If x is a document and C and (not C) are relevant and nonrelevant, then
    - P(C | x) = P(R | D)
    - P(not C | x) = P(NR | D)
- If P(C | x)/P(not C | x) > 1, classify as C
- Requires an estimate of P(C | x)

Classification

- Estimating P(C | x) can be hard
  - Think of x as a vector of features that represent the instance
  - Estimating P(C | x) involves estimating a relationship between every feature and the class, e.g.
    - Feature 1 present → more likely to be in C
    - Feature 2 present → less likely to be in C
    - …
  - More features → harder to estimate each relationship
  - Documents usually have a feature for each term

Bayesian Classification

- Instead of estimating P(C | x), estimate P(x | C), e.g.
  - x in C → feature 1 more likely to be present
  - x in C → feature 2 more likely to be absent
  - …
- But how do we use P(x | C) for classification?
  - Our rule is defined by P(C | x), not P(x | C)

Bayesian Classification

- Apply Bayes' rule:

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)}$$

- P(C) is the *prior* of C: the probability of something being in class C when you have no information about it
- P(x | C) is the *model*: if we sampled from a bucket containing things in C, what is the probability we would get x?
- P(x) is the *prior* of x: the probability of something being x. Also called the *normalizing constant*.
- P(C | x) is the *posterior* and what we want to estimate

Example

- C = spam, x = does the word "sale" appear?
- Estimate prior P(C):

$$P(C) = \frac{\#\text{ of emails marked spam}}{\text{total }\#\text{ of emails received}}$$

- Estimate model P(x | C):

$$P(x|C) = \frac{P(x,C)}{P(C)} = \frac{\frac{\#\text{ of emails marked spam that contain "sale"}}{\text{total }\#\text{ of emails received}}}{\frac{\#\text{ of emails marked spam}}{\text{total }\#\text{ of emails received}}} = \frac{\#\text{ of emails marked spam that contain "sale"}}{\#\text{ of emails marked spam}}$$

Example

- What is instance prior P(x)?
  - Normalizing constant: it just needs to be set to ensure that P(C | x) + P(not C | x) = 1

$$P(x) = P(x|C)P(C) + P(x|\bar{C})P(\bar{C})$$

  - where

$$P(\bar{C}) = 1 - P(C)$$

$$P(x|\bar{C}) = \frac{\#\text{ of emails marked not spam that contain "sale"}}{\#\text{ of emails marked not spam}}$$

Example

	Spam	Not spam	total
"sale" appears	r	n – r	n
"sale" does not appear	|C| – r	N – |C| + r – n	N – n
total	|C|	N – |C|	N

$$P(C) = \frac{|C|}{N} \qquad P(\bar{C}) = \frac{N-|C|}{N}$$
$$P(x|C) = \frac{r}{|C|} \qquad P(x|\bar{C}) = \frac{n-r}{N-|C|}$$

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} = \frac{P(x|C)P(C)}{P(x|C)P(C)+P(x|\bar{C})P(\bar{C})}$$
$$= \frac{r/N}{r/N+(n-r)/N} = \frac{r}{n}$$

Example

- I got an email that contains the word "sale"
- Is it spam?

$$P(C|x) = \frac{r}{n}$$
$$P(\bar{C}|x) = 1 - \frac{r}{n} = \frac{n-r}{n}$$
$$\frac{P(C|x)}{P(\bar{C}|x)} = \frac{r}{n-r}$$

If r/(n-r) > 1, then yes.

Naïve Bayes Model

- Usually x will have more than one feature
- More features → harder to estimate model

$$P(C|x_1, x_2) = \frac{P(x_1, x_2|C)P(C)}{P(x_1, x_2)}$$

  - Four possibilities to estimate P(x_1, x_2 | C)
  - n features → 2^n probabilities to estimate
- Naïve Bayes assumption:
  - P(x_1, x_2 | C) = P(x_1 | C)P(x_2 | C)

Naïve Bayes Model

- Consider a document D of terms
- We want to estimate P(C | D)
  - First apply Bayes rule:

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

  - Then the naïve Bayes assumption:

$$\frac{P(D|C)P(C)}{P(D)} = \frac{\prod_{i=1}^{n} P(t_i|C)P(C)}{P(D)}$$

  - Estimate P(t_i | C) using term presence statistics
  - Estimate P(C) as before

Naïve Bayes Classification

- Calculate odds ratio

$$\frac{P(C|D)}{P(\bar{C}|D)} = \frac{P(C)/P(D)\prod_{i=1}^{n} P(t_i|C)}{P(\bar{C})/P(D)\prod_{i=1}^{n} P(t_i|\bar{C})}$$

- Where

$$P(t_i|C) = \frac{\#\text{ of documents in } C \text{ that contain } t_i}{\#\text{ of documents in } C}$$

$$P(t_i|\bar{C}) = \frac{\#\text{ of documents in } \bar{C} \text{ that contain } t_i}{\#\text{ of documents in } \bar{C}}$$

Zero-Probability Problem

- If $P(t_i \mid C) = 0$, then $P(C \mid D) = 0$
  - E.g. if an email contains just one word that we haven't seen in a spam before,
  - But all the other words in it are spam words,
  - It would not be considered spam.
- Solution: smoothing
  - Adding a constant to numerator and denominator is a simple smoothing method

$$P(t_i|C) = \frac{0.5 + \#\text{ of documents in } C \text{ that contain } t_i}{1 + \#\text{ of documents in } C}$$

An Alternate Model

- Estimating P(t | C) using the number of documents t appears in is a *multivariate Bernoulli* model
  - Similar to the binary independence model
- A *multinomial* model uses term frequencies within documents
  - Like a language model for the class C

$$P(t_i|C) = (1-\alpha_C)\frac{\sum_{D_j \in C} tf_{i,D_j}}{\sum_{D_j \in C} |D_j|} + \alpha_C \frac{ctf_i}{\sum_j |D_j|}$$

- Multinomial is usually better

k-ary Classification

- Naïve Bayes can be used for 3 classes, 4 classes, …
  - Example: classify by topic (news, sports, finance, technology, …)
- Estimate P(t |c_j) and P(c_j) for every class c_j
- Predicted class is the one with greatest probability:

$$C^* = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{n} P(t_i \ c_j)$$

Feature Selection

- Vocabulary sizes run to hundreds of thousands or millions of words
- Many of these are not useful for distinguishing between classes
  - A term that appears equally often in both classes is not useful
  - A term that only appears once is probably not useful
  - The most useful terms are those that appear often in one class and less often in the other
- Feature selection lets us ignore some terms in our probabilistic classifier

Information Theoretic Feature Selection

- Only model the terms that are *most informative* about a class
- A term can provide information about the class in several ways:
  - The presence of the term indicates the class
  - The absence of the term indicates the class
  - The presence of the term does not indicate the class
  - The absence of the term does not indicate the class

Mutual Information

- Presence of term indicates class:
  - Compare probability that term appears in documents in the class to probability of the term and probability of the class

$$\frac{P(t_i = 1, C = 1)}{P(t_i = 1)P(C = 1)} = \frac{\frac{\#\text{ of documents in } C \text{ that contain } t_i}{\#\text{ of documents}}}{\frac{\#\text{ of documents that contain } t_i}{\#\text{ of documents}} \frac{\#\text{ of documents in } C}{\#\text{ of documents}}}$$

  - If this ratio is closer to 1, term is less informative
- Calculate the other three types similarly:

$$\frac{P(t_i = 1, C = 0)}{P(t_i = 1)P(C = 0)}, \frac{P(t_i = 0, C = 1)}{P(t_i = 0)P(C = 1)}, \frac{P(t_i = 0, C = 0)}{P(t_i = 0)P(C = 0)}$$

Mutual Information

$$\begin{aligned} MI(t_i, C) = & \quad P(t_i = 1, C = 1) \log_2 \frac{P(t_i = 1, C = 1)}{P(t_i = 1)P(C = 1)} \\ & + \quad P(t_i = 1, C = 0) \log_2 \frac{P(t_i = 1, C = 0)}{P(t_i = 1)P(C = 0)} \\ & + \quad P(t_i = 0, C = 1) \log_2 \frac{P(t_i = 0, C = 1)}{P(t_i = 0)P(C = 1)} \\ & + \quad P(t_i = 0, C = 0) \log_2 \frac{P(t_i = 0, C = 0)}{P(t_i = 0)P(C = 0)} \end{aligned}$$

Train our classifier using only the top *k* highest mutual information terms

Recap

- Assume we have training data
  - Documents labeled spam vs not spam, or labeled by category, etc
- Use training data to estimate mutual information for all terms and all classes
- Throw out lowest mutual information terms
- For remaining terms, use training data to estimate P(C), P(t | C), etc
- Now we have a Naïve Bayes classifier

Evaluating a Classifier

- Classifiers can make errors
  - (recall nonseparable example)
- We need to evaluate our classifier's performance to determine whether it is good enough
- How do we do it?
  - Testing data: data that has been labeled but that we did not use for training
  - Use trained classifier to predict which class each testing instance belongs to
  - Compare predictions to actual labels

Evaluating a Classifier

	In class	Not in class	
Predicted in class	A	B	A+B
Predicted not in class	C	D	C+D
	A+C	B+D	N=A+B+C+D

$$precision = \frac{A}{A+B} \qquad \text{false positive rate} = \frac{B}{B+D}$$

$$recall = \frac{A}{A+C} \qquad \text{false negative rate} = \frac{C}{A+C}$$

$$accuracy = \frac{A+D}{A+B+C+D}$$