CSCI 5417
Information Retrieval Systems

Jim Martin

Lecture 16
10/18/2011

Today

- Review clustering
  - K-means
- Review naïve Bayes
- Unsupervised classification
  - EM
  - Naïve Bayes/EM for text classification
- Topic models model intuition

K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\mu(\mathrm{c}) = \frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x}$$

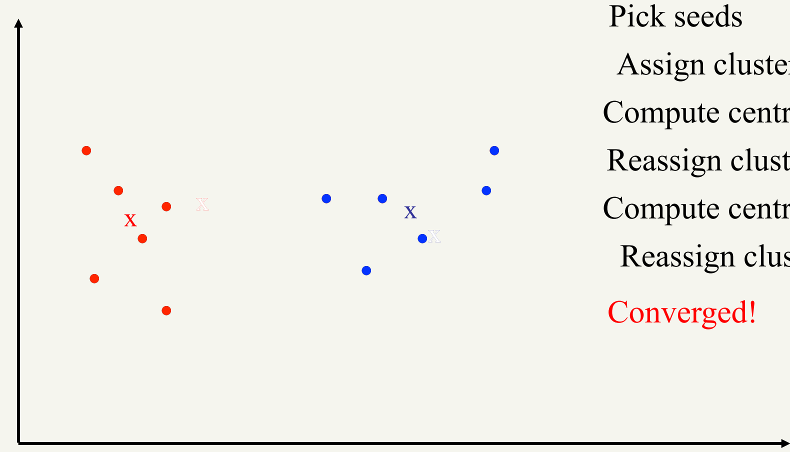
- Iterative reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select K random docs $\{s_1, s_2, \ldots s_K\}$ as seeds.
Until stopping criterion:
  For each doc d_i:
    Assign d_i to the cluster c_j
    such that $dist(d_i, s_j)$ is minimal.

  For each cluster c_j
      s_j = m(c_j)

K Means Example
(*K*=2)



Pick seeds

Assign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

Termination conditions

- Several possibilities
  - A fixed number of iterations
  - Doc partition unchanged
  - Centroid positions don't change

Convergence

- Why should the K-means algorithm ever reach a *fixed point*?
  - A state in which clusters don't change.
- K-means is a special case of a general procedure known as the *Expectation Maximization (EM) algorithm*.
  - EM is known to converge.
  - Number of iterations could be large.
    - But in practice usually isn't

Naïve Bayes: Learning

- From training corpus, extract $Vocabulary$
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$
    - $Text_j \leftarrow$ single document containing all $docs_j$
    - for each word x_k in $Vocabulary$
      - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
      - $$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha\,|Vocabulary|}$$

Multinomial Model

```
TRAINMULTINOMIALNB(ℂ, 𝔻)
V ← EXTRACTVOCABULARY(𝔻)
N ← COUNTDOCS(𝔻)
for each c ∈ ℂ
do N_c ← COUNTDOCSINCLASS(𝔻, c)
    prior[c] ← N_c / N
    text_c ← CONCATENATETEXTOFALLDOCSINCLASS(𝔻, c)
    for each t ∈ V
    do T_ct ← COUNTTOKENSOFTERM(text_c, t)
    for each t ∈ V
    do condprob[t][c] ← (T_ct + 1) / Σ_t'(T_ct' + 1)
return V, prior, condprob
```


Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Apply Multinomial

```
ApplyMultinomialNB(ℂ, V, prior, condprob, d)
W ← ExtractTokensFromDoc(V, d)
for each c ∈ ℂ
do score[c] ← log prior[c]
    for each t ∈ W
    do score[c] += log condprob[t][c]
return arg max_{c∈ℂ} score[c]
```


Naïve Bayes Exampl

Doc	Category
{China, soccer}	Sports
{Japan, baseball}	Sports
{baseball, trade}	Sports
{China, trade}	Politics
{Japan, Japan, exports}	Politics

Doc	Category
D1	Sports
D2	Sports
D3	Sports
D4	Politics
D5	Politi

Using +1; |V| =6; |Sports| = 6; |Politics| = 5

Sports (.6)	
baseball	3/12
China	2/12
exports	1/12
Japan	2/12
soccer	2/12
trade	2/12

Politics (.4)	
baseball	1/11
China	2/11
exports	2/11
Japan	3/11
soccer	1/11
trade	2/11

Naïve Bayes Example

- Classifying
  - Soccer (as a doc)
    - Soccer | sports = .167
    - Soccer | politics = .09

    Sports > Politics

Example 2

- Howa about?
  - *Japan soccer*
    - Sports
      - P(japan|sports)P(soccer|sports)P(sports)
      - .166 * .166* .6 = .0166
    - Politics
      - P(japan|politics)P(soccer|politics)P(politics)
      - .27 * .09 *. 4 = .00972
    - Sports > Politics

Break

- No class Thursday; work on the HW
  - No office hours either.
- HW questions?
  - The format of the test docs will be same as the current docs minus the .M field which will be removed.
  - How should you organize your development efforts?

Example 3

- What about?
  - China trade

Sports (.6)	
baseball	3/12
China	2/12
exports	1/12
Japan	2/12
soccer	2/12
trade	2/12

Politics (.4)	
baseball	1/11
China	2/11
exports	2/11
Japan	3/11
soccer	1/11
trade	2/11

Sports

$.166 * .166 * .6 = .0166$

Politics

$.1818 * .1818 *. 4 = .0132$

Again Sports > Politics

Problem?

Doc	C
{China, soccer}	S
{Japan, baseball}	S
{baseball, trade}	Sports
{China, trade}	Politics
{Japan, Japan, exports}	Politics

Naïve Bayes doesn't remember the training data. It just extracts statistics from it. There's no guarantee that the numbers will generate correct answers for all members of the training set.

What if?

- What if we just have the documents but no class assignments?
  - But assume we do have knowledge about the number of classes involved
- Can we still use probabilistic models? In particular, can we use naïve Bayes?
  - Yes, via EM
    - Expectation Maximization

EM

1. Given some model, like NB, make up some class assignments randomly.
2. Use those assignments to generate model parameters P(class) and P(word|class)
3. Use those model parameters to re-classify the training data.
4. Go to 2

Naïve Bayes Example (EM)

Doc	Category
D1	?
D2	?
D3	?
D4	?
D5	?

Naïve Bayes Example (EM)

Doc	Category
D1	Sports
D2	Politics
D3	Sports
D4	Politics
D5	Sports

Doc	Category
{China, soccer}	Sports
{Japan, baseball}	Politics
{baseball, trade}	Sports
{China, trade}	Politics
{Japan, Japan, exports}	Sports

Sports (.6)	
baseball	2/13
China	2/13
exports	2/13
Japan	3/13
soccer	2/13
trade	2/13

Politics (.4)	
baseball	2/10
China	2/10
exports	1/10
Japan	2/10
soccer	1/10
trade	2/10

Naïve Bayes Example (EM)

- Use these counts to reassess the class membership for D1 to D5. Reassign them to new classes. Recompute the tables and priors.
- Repeat until happy

Doc	Category
D1	Sports
D2	Politics
D3	Sports
D4	Politics
D5	Sports

Doc	Category
{China, soccer}	Sports
{Japan, baseball}	Politics
{baseball, trade}	Sports
{China, trade}	Politics
{Japan, Japan, exports}	Sports

Sports (.6)	
baseball	2/13
China	2/13
exports	2/13
Japan	3/13
soccer	2/13
trade	2/13

Politics (.4)	
baseball	2/10
China	2/10
exports	1/10
Japan	2/10
soccer	1/10
trade	2/10

Topics

Doc	Category
{China, soccer}	Sports
{Japan, baseball}	Sports
{baseball, trade}	Sports
{China, trade}	Politics
{Japan, Japan, exports}	Politics

What's the deal with trade?

Topics

Doc	Category
{China$_1$, soccer$_2$}	Sports
{Japan$_1$, baseball$_2$}	Sports
{baseball$_2$, trade$_2$}	Sports
{China$_1$, trade$_1$}	Politics
{Japan$_1$, Japan$_1$, exports$_1$}	Politics

{basketball$_2$, strike$_3$}

Topics

- So let's propose that instead of assigning documents to classes, we assign each word token in each document to a class (topic).
- Then we can some new probabilities to associate with words, topics and documents
  - Distribution of topics in a doc
  - Distribution of topics overall
  - Association of words with topics

Topics

- Example. A document like
  - {basketball_2, strike_3}

  Can be said to be .5 about topic 2 and .5 about topic 3 and 0 about the rest of the possible topics (may want to worry about smoothing later.
- For a collection as a whole we can get a topic distribution (prior) by summing the words tagged with a particular topic, and dividing by the number of tagged tokens.

Problem

- With “normal” text classification the training data associates a document with one or more topics.
- Now we need to associate topics with the (content) words in each document
- This is a semantic tagging task, not unlike part-of-speech tagging and word-sense tagging
  - It’s hard, slow and expensive to do right

Topic modeling

- Do it without the human tagging
  - Given a set of documents
  - And a fixed number of topics (given)
  - Find the statistics that we need