Text Categorization
Prof. Chris Clifton
1 October 2019
Material adapted from course created by Dr. Luo Si, now leading Alibaba research group

Text Categorization

• Introduction to the task of text categorization
  – Manual vs. automatic text categorization
• Text categorization applications
• Evaluation of text categorization
• K nearest neighbor text categorization method
Text Categorization

• Tasks
  – Assign predefined categories to text documents / objects

• Motivation
  – Provide an organizational view of the data

• Large cost of manual text categorization
  – Millions of dollars spent for manual categorization in companies, governments, public libraries, hospitals
  – Manual categorization is almost impossible for some large scale application (Classification or Web pages)

Text Categorization

• Automatic text categorization
  – Learn algorithm to automatically assign predefined categories to text documents / objects
  – automatic or semi-automatic

• Procedures
  – Training: Given a set of categories and labeled document examples; learn a method to map a document to correct category (categories)
  – Testing: Predict the category (categories) of a new document

• Automatic or semi-automatic categorization can significantly reduce manual effort
Text Categorization: Examples

Categories

- Computer Supported Cooperative Work (CSCW)
- Conferences (7)
- Courses (6)
- Ergonomics@
- Information Architecture and Design@
- Institutes (19)
- Journals (1)
- Organizations (3)
- Projects (6)
- Web Directories (2)

© 2019 Christopher W. Clifton
Text Categorization: Examples

Example: 1990 US Census

- Included 22 million responses
- Needed to be classified into industry categories (200+) and occupation categories (500+)
- Estimated $15 million if done by hand
- Two alternative automatic text categorization methods evaluated
  - Knowledge-Engineering (Expert System)
  - Machine Learning (k-nearest neighbor method)
Example: 1990 US Census

- Knowledge-Engineering Approach
  - Expert System (Designed by domain expert)
  - Hand-Coded rule
    (e.g., “Professor” and “Lecturer” → “Education”)
  - Development cost: 2 experts, 8 years (192 Person-months)
  - Accuracy = 47%

- Machine Learning Approach
  - k-Nearest Neighbor (KNN) classification
    - “You are like people like you”, details later
  - Fully automatic
  - Development cost: 4 Person-months
  - Accuracy = 60%

Many Applications!

- Web page classification (Yahoo-like category taxonomies)
- News article classification (more formal than most Web pages)
- Automatic email sorting (spam detection; into different folders)
- Word sense disambiguation (Java programming vs. Java in Indonesia)
- Gene function classification (find the functions of a gene from the articles talking about the gene)
- What is your favorite application?...
Techniques Explored in Text Categorization

- Rule-based Expert system (Hayes, 1990)
- Nearest Neighbor methods (Creecy’92; Yang’94)
- Decision symbolic rule induction (Apte’94)
- Naïve Bayes (Language Model) (Lewis’94; McCallum’98)
- Regression method (Furh’92; Yang’92)
- Support Vector Machines (Joachims’98)
- Boosting or Bagging (Schapier’98)
- Neural networks (Wiener’95)
- ……

Text Categorization: Evaluation

Performance of different algorithms on Reuters-21578 corpus: 90 categories, 7769 Training docs, 3019 test docs, (Yang, JIR 1999)
### Text Categorization: Evaluation

**Contingency Table Per Category (for all docs)**

<table>
<thead>
<tr>
<th></th>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td>a+c</td>
<td>b+d</td>
</tr>
</tbody>
</table>

- a: number of truly positive docs
- b: number of false-positive docs
- c: number of false negative docs
- d: number of truly-negative docs
- n: total number of test documents

**Sensitivity:** \( \frac{a}{a+c} \)  truly-positive rate, the larger the better  
**Specificity:** \( \frac{d}{b+d} \)  truly-negative rate, the larger the better  
Depends on decision threshold, trade off between the values.
Text Categorization: Evaluation

• Micro F1-Measure
  – Calculate a single contingency table for all categories and calculate F1 measure
  – Treat each prediction with equal weight; better for algorithms that work well on large categories

• Macro F1-Measure
  – Calculate a single contingency table for every category; calculate F1 measure separately and average the values
  – Treat each category with equal weight; better for algorithms that work well on many small categories

K-Nearest Neighbor Classifier

• Also called “Instance-based learning” or “lazy learning”
  – low/no cost in “training”, high cost in online prediction
• Commonly used in pattern recognition (5 decades)
• Theoretical error bound analyzed by Duda & Hart (1957)
• Applied to text categorization in 1990’s
• Among top-performing text categorization methods
K-Nearest Neighbor Classifier

From all training examples:
• Find \( k \) examples that are most similar to the new document
  – “neighbor” documents
• Assign the category that is most common in these neighbor documents
  – neighbors “vote” for the category
• Can also consider the distance of a neighbor
  – a closer neighbor has more weight/influence

K-Nearest Neighbor Classifier

- Idea: find your language by what language your neighbors speak
  (k=1) Red; (k=5) Brown; (k=10) ?; Weighted 10-NN: Brown
K Nearest Neighbor: Technical Elements

- Document representation
- Document distance measure: closer documents should have similar labels; neighbors speak the same language
- Number of nearest neighbors (value of K)
- Decision threshold

K Nearest Neighbor: Framework

Training data \( D = \{(x_i, y_i)\}, \quad x_i \in \mathbb{R}^M, \text{docs,} \quad y_i \in \{0,1\} \)

Test data \( x \in \mathbb{R}^M \)

Scoring Function \( \hat{y}(x) = \frac{1}{k} \sum_{x_i \in D_k(x)} \text{sim}(x, x_i)y_i \)

Classification: 
\[
\begin{cases} 
1 \quad \text{if} \quad \hat{y}(x) - t > 0 \\
0 \quad \text{otherwise}
\end{cases}
\]

Document Representation: \( X_i \) uses tf.idf weighting for each dimension
# Choices of Similarity Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean distance</td>
<td>( d(x_1, x_2) = \sqrt{\sum_v (x_{1v} - x_{2v})^2} )</td>
</tr>
<tr>
<td>Kullback Leibler distance</td>
<td>( d(x_1, x_2) = \sum_v x_{1v} \log \frac{x_{1v}}{x_{2v}} )</td>
</tr>
<tr>
<td>Dot product</td>
<td>( x_1 * x_2 = \sum_v x_{1v} * x_{2v} )</td>
</tr>
<tr>
<td>Cosine Similarity</td>
<td>( \cos(x_1, x_2) = \frac{\sum_v x_{1v} * x_{2v}}{\sqrt{\sum_v x_{1v}^2} \sqrt{\sum_v x_{2v}^2}} )</td>
</tr>
<tr>
<td>Kernel functions</td>
<td>( k(x_1, x_2) = e^{-\frac{d(x_1, x_2)^2}{2\sigma^2}} ) (Gaussian Kernel)</td>
</tr>
</tbody>
</table>

Automatic learning of the metrics

# Choices of Number of Neighbors (K)

Trade off between small number of neighbors and large number of neighbors
Choices of Number of Neighbors (K)

- Find desired number of neighbors by cross validation
  - Choose a subset of available data as training data, the rest as validation data
  - Find the desired number of neighbors on the validation data
  - The procedure can be repeated for different splits; find the consistent good number for the splits

Characteristics of KNN

Pros
- Simple and intuitive, based on local-continuity assumption
- Widely used and provide strong baseline in TC Evaluation
- No training needed, low training cost
- Easy to implement; can use standard IR techniques (e.g., tf.idf)

Cons
- Heuristic approach, no explicit objective function
- Difficult to determine the number of neighbors
- High online cost in testing; find nearest neighbors has high time complexity
Problem: Weighting of Terms

- K-NN treats all terms equally
  - Frequent but unimportant terms may dominate
- Which terms are more important?
  - TF.IDF?
  - ...
- Solution – machine learning
  - We have training data

Naïve Bayes Classification

- Naïve Bayes (NB) Classification
  - Generative Model: Model both the input data (i.e., document contents) and output data (i.e., class labels)
  - Make strong assumption of the probabilistic modeling approach
- Methodology
  - Similar with the idea of language modeling approaches for information retrieval
  - Train a language model for all the documents in one category
Naïve Bayes Classification

• Methodology
  – Train a language model for all the documents in one category
    
    Category 1: \( \tilde{d}_{1,1}, \tilde{d}_{1,2}, \ldots, \tilde{d}_{1,n_1} \) → Language model \( \theta_1 \)
    
    Category 2: \( \tilde{d}_{2,1}, \tilde{d}_{2,2}, \ldots, \tilde{d}_{2,n_2} \) → Language model \( \theta_2 \)
    
    .......
    
    Category C: \( \tilde{d}_{C,1}, \tilde{d}_{C,2}, \ldots, \tilde{d}_{C,n_C} \) → Language model \( \theta_C \)

  – What is the language model? (Multinomial distribution)
  – How to estimate the language model for all the documents in one category?

Naïve Bayes Classification

• Representation
  – Each document is a “bag of words” with weights (e.g., TF.IDF)
  – Each category is a super “bag of words”, which is composed of all words in all the documents associated with the category
  – For all the words in a specific category \( c \), it is modeled by a multinomial distribution as
    
    \[ p(\tilde{d}_{c1}, \ldots, \tilde{d}_{cn_c} | \theta_c) \]

  – Each category (\( c \)) has a prior distribution \( P(c) \), which is the probably of choosing category \( c \) BEFORE observing the content of a document
Naïve Bayes Classification

Maximum Likelihood Estimation:

- Find model parameters for a category that maximizes generation likelihood:
  \[ \theta_c^* = \arg \max_{\theta} p(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) \]

There are \( K \) words in vocabulary, \( w_1...w_K \)
Data: documents \( \vec{d}_{c1}, ..., \vec{d}_{cn_c} \)
For \( \vec{d}_c \) with counts \( c_i(w_1), ..., c_i(w_k) \), and length \( |\vec{d}_c| \)
Model: multinomial M with parameters \( \{p(w_k)\} \)
Likelihood: \( \Pr(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) \)

\[ \theta_c^* = \arg \max_{\theta} p(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) \]

---

Maximum Likelihood Estimation (MLE)

\[ p(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) = \prod_{i=1}^{n_c} \left( \frac{|\vec{d}_{ci}|}{\sum_{i=1}^{n_c} |\vec{d}_{ci}|} \right) \prod_{k=1}^{K} p^{c_i}(w_k) = \prod_{i=1}^{n_c} \prod_{k=1}^{K} p^{c_i}(w_k) \]

\[ l(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) = \log p(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) = \sum_{i=1}^{n_c} \sum_{k=1}^{K} c_i(w_k) \log p_k \]

\[ l'(\vec{d}_{c1}, ..., \vec{d}_{cn_c} | \theta) = \sum_{i=1}^{n_c} \sum_{k=1}^{K} c_i(w_k) \log \theta_k + \lambda \left( \sum_{k=1}^{K} p_k - 1 \right) \]

\[ \frac{\partial l'}{\partial p_k} = \sum_{i=1}^{n_c} c_i(w_k) \] 

Use Lagrange multiplier approach

Set partial derivatives to zero

Get maximum likelihood estimate

Since \( \sum_k p_k = 1 \), \( \lambda = - \sum_k \sum_{i=1}^{n_c} c_i(w_k) = - \sum_{i=1}^{n_c} |\vec{d}_{ci}| \)

So, \( p_k = p(w_k) = \frac{\sum_{i=1}^{n_c} c_i(w_k)}{\sum_{i=1}^{n_c} |\vec{d}_{ci}|} \)
Naïve Bayes Classification

- **MLE Estimator:** Normalization by simple counting
  - Train a language model for all the documents in one category
    \[
    p(w | \theta^*_c) = \frac{\sum_{i=1}^{n_c} c^*_c(w)}{\sum_{i=1}^{n_c} |d_{ci}|}
    \]
    \[
    p(c) = \frac{n_c}{\sum_{c'} n_{c'}}
    \]

- **Category Prior:**
  - Number of documents in the category divided by the total number of documents

---

**Naïve Bayes Classification**

- **Smoothed Estimator:**
  - Laplace Smoothing
    \[
    p(w | \theta^*_c) = \frac{1 + \sum_{i=1}^{n_c} c^*_c(w)}{K + \sum_{i=1}^{n_c} |d_{ci}|}
    \]
  - Hierarchical Smoothing
    \[
    p(w | \theta^*_c) = \lambda_1 P(w | \theta^*_c) + \lambda_2 P(w | \theta^*_{c_{apl}}) + \ldots + \lambda_m P(w | \theta^*_{c_{null}})
    \]
  - Dirichlet Smoothing
Naïve Bayes Classification

- **Prediction:**

\[ c^* = \arg \max_c p(c \mid d_i) \]

\[ = \arg \max_c \left\{ \frac{p(c) p(d_i \mid c)}{p(d_i)} \right\} \]

\[ = \arg \max_c \left\{ p(c) p(d_i \mid c) \right\} \quad \text{(Bayes Rule)} \]

\[ = \arg \max_c \left\{ \prod_k p(w_k \mid c)^{c(w_k)} \right\} \quad \text{(Multinomial Dist)} \]

\[ = \arg \max_c \left\{ \log(p(c)) + \sum_k c(w_k) \log(p(w_k \mid c)) \right\} \]

- **Example of Binary Classification**

Two classes

\[ c^* = \arg \max_{c \in \{-, +\}} p(c \mid d_i) \rightarrow \frac{c^* + d_i}{p(c_+ \mid d_i)} + \frac{c^* - d_i}{p(c_- \mid d_i)} \]

\[ p(c_+ \mid d_i) \propto \prod_k p(w_k \mid c_+)^{c(w_k)} \frac{n_+}{n_+ + n_-} \]

\[ p(c_- \mid d_i) \propto \prod_k p(w_k \mid c_-)^{c(w_k)} \frac{n_-}{n_+ + n_-} \]
Naïve Bayes Classification

- Example of Binary Classification

\[ c^* = \arg \max_{l \in \{-1, +1 \}} p(c_l \mid \vec{d}_j) \rightarrow \ast + \mid \vec{d}_j \]

\[
\log \frac{p(c_+ \mid \vec{d}_j)}{p(c_- \mid \vec{d}_j)} = \log \left( \frac{\prod_l \left[ p(w_k \mid c_{+}) \right]^{c_i(w_k)} \frac{n_+}{n_+ + n_-}}{\prod_l \left[ p(w_k \mid c_{-}) \right]^{c_i(w_k)} \frac{n_-}{n_+ + n_-}} \right)
\]

\[ = \log \left( \frac{n_+}{n_-} \right) + \sum_k c_i(w_k) \log \left( \frac{p(w_k \mid c_{+})}{p(w_k \mid c_{-})} \right) \]

\[ \log \frac{p(c_+ \mid \vec{d})}{p(c_- \mid \vec{d})} + \sum_k c_i(w_k) \times \text{weight}(w_k) \]

Naïve Bayes =
Linear Classifier

- denotes +1
- denotes -1
Naïve Bayes Classification

• Summary
  – Utilize multinomial distribution for modeling categories and documents
  – Use posterior distribution (posterior of category given document) to predict optimal category

• Pros
  – Solid probabilistic foundation
  – Fast online response, linear classifier for binary classification

• Cons
  – Empirical performance not very strong
  – Probabilistic model for each category is estimated to maximize the data likelihood for documents in the category (generative), not for purpose of distinguishing documents in different categories (discriminative)
Outline

• Support Vector Machine (SVM)
  A Large-Margin Classifier
  – Introduction to SVM
  – Linear, hard margin
  – Linear, Soft margin
  – Non-Linear SVM (kernel functions)
  – Discussion

History of SVM

• A brief history of SVM
• SVM is inspired from statistical learning theory by Vapnik (1979) [3]
• Put into practical application as “Large Margin Classifiers” in (1992) [1]
• SVM became famous for it success in handwritten digit recognition [2]
• SVM has been successfully utilized in
  – Image detection
  – Speaker identification
  – Text categorization
  – Many other problems…

Support Vector Machine

- Consider a two-class (binary classification problem like text categorization)
  - Find a line to separate data points in two classes
- There are many possible solutions!
  - Are those decision boundaries equally good?

A slight variation of the data makes some decision boundaries incorrect
Large-Margin Decision Criterion

- The decision boundary should be far away from the data points of two classes as much as possible
- Indicates the margin between data points and the decision boundary should be large

The margin is:

$$m = \frac{2}{||W||}$$
Linear SVM

- Let \( \{x_1, ..., x_n\} \) denote input data. For example, vector representation of all documents.
- Let \( y_i \) be the binary indicator 1 or -1 that indicates whether \( x_i \) belongs to a particular category c or not.

The decision boundary should classify all points correctly

\[
y_i (w^T x_i + b) \geq 1, \quad \forall i
\]

The decision boundary can be found by solving the following constrained optimization problem

\[
\text{Minimize } \frac{1}{2} ||w||^2
\]

subject to \( y_i (w^T x_i + b) \geq 1 \quad \forall i \)

The Karush-Kuhn-Tucker Condition

- The optimal solution of model parameter satisfies

\[
\alpha_i (1 - y_i (W^T X_i + b)) = 0 \quad \forall i
\]

\[
\downarrow
\]

Support Vectors

\[
\begin{cases} 
\alpha_i = 0 \\
(\alpha_i > 0) \land (1 - y_i (W^T X_i + b) = 0)
\end{cases}
\]

- Each support vector \( x_i \) has positive weight
- Non-support vectors have a zero weight
The Karush-Kuhn-Tucker Condition

- The optimal solution of model parameter satisfies
  - Each support vector $x_i$ has positive weight
  - Non-support vectors have a zero weight

Prediction only needs to consider support vectors; save storage and computation

Hard Margin Linear SVM Solution

- The optimal parameters are
  $$w^* = \sum_{i \in SV} \alpha_i y_i x_i$$

  $$y_i (W^* x_i - b) = 1 \quad \forall i \in SV$$

Prediction is made by:

$$\text{sign}(WX - b) = \text{sign}(\sum_{i \in SV} \alpha_i y_i (x_i \cdot X) - b)$$
The Karush-Kuhn-Tucker Condition

• What about data that isn’t linearly separable?

The Karush-Kuhn-Tucker Condition

• We tolerate some error for specific data points as
Soft Margin Linear SVM

Introduction “slack variables”, slack variables are always positive

\[
\begin{cases}
    w^T x_i + b \geq 1 - \xi_i & y_i = 1 \\
    w^T x_i + b \leq -1 + \xi_i & y_i = -1 \\
    \xi_i \geq 0 & \forall i
\end{cases}
\]

Introduce const C to balance error for linear boundary and the margin

\[
\frac{1}{2} ||w||^2 + C \sum \xi_i
\]

The optimization problem becomes

Minimize \( \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \xi_i \)
subject to \( y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \)

\begin{itemize}
  \item The dual of the problem for soft margin linear SVM is:
  \[
  \max. \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j
  \]
  subject to \( C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \)

  \[w\] is calculated as \( w^* = \sum_{i=SV} \alpha_i y_i X_i \)

  This is very similar to the optimization problem in the linear separable case, except that there is an upper bound \( C \) on \( a_i \) now

  Once again, a QP solver can be used to find \( a_i \)
\end{itemize}
Non-linear SVM

- Linear SVM only uses a line to separate data points, how to generalize it to non-linear case?
- Key idea: transform $X_i$ to a higher dimension space
  - Input space: the space the point $x_i$ are located
  - Feature space: the space of $f(x_i)$ after transformation

Key idea: transform $X_i$ to a higher dimension space
The Kernel Trick

• Recall the SVM optimization problem

\[
\begin{align*}
\text{max. } W(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i &= 0
\end{align*}
\]

The data points only appear as inner product
As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
Many common geometric operations (angles, distances) can be expressed by inner products
Define the kernel function \( K \) by \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \)

Example Kernels

• Suppose \( f(.) \) is given as follows

\[ \phi\left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2) \]

• An inner product in the feature space is

\[ \langle \phi\left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right), \phi\left( \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right) \rangle = (1 + x_1y_1 + x_2y_2)^2 \]

• So, if we define the kernel function as follows, there is no need to carry out \( f(.) \) explicitly

\[ K(x, y) = (1 + x_1y_1 + x_2y_2)^2 \]
More Kernel Functions

- Polynomial kernel with degree d
  \[ K(x, y) = (x^T y + 1)^d \]
- Gaussian Radial basis function kernel with width \( \sigma \)
  \[ K(x, y) = \exp(-||x - y||^2/(2\sigma^2)) \]
- Two-layer sigmoid neural network
  \[ K(x, y) = \tanh(\kappa x^T y + \theta) \]

Kernel SVM Solution

- The optimal parameters are
  \[ w^* = \sum_{i \in SV} \alpha_i y_i \phi(X_i) \]
  \[ y_i(W^* X_i - b) = 1 \quad \forall i \in SV \]

Prediction is made by:

\[ \text{sign}(WX - b) = \text{sign}(\sum_{i \in SV} \alpha_i y_i (\phi(X_i) \cdot \phi(X)) - b) \]
\[ = \text{sign}(\sum_{i \in SV} \alpha_i y_i (K(X_i, X) - b)) \]
Text Categorization: Evaluation

Performance of different algorithms on Reuters-21578 corpus: 90 categories, 7769 Training docs, 3019 test docs, (Yang, JIR 1999)

Clustering

- Document clustering
  - Motivations
  - Document representations
  - Success criteria
- Clustering algorithms
  - K-means
  - Model-based clustering (EM clustering)
What is clustering?

- **Clustering** is the process of grouping a set of physical or abstract objects into classes of similar objects
  - It is the commonest form of unsupervised learning
    - Unsupervised learning = learning from raw data, as opposed to supervised data where the correct classification of examples is given
  - It is a common and important task that finds many applications in IR and other places

Why cluster documents?

- Whole corpus analysis/navigation
  - Better user interface
- For improving recall in search applications
  - Better search results
- For better navigation of search results
- For speeding up vector space retrieval
  - Faster search
Navigating document collections

- Standard IR is like a book index
- Document clusters are like a table of contents
- People find having a table of contents useful

<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aardvark, 15</td>
</tr>
<tr>
<td>Blueberry, 200</td>
</tr>
<tr>
<td>Capricorn, 1, 45-55</td>
</tr>
<tr>
<td>Dog, 79-99</td>
</tr>
<tr>
<td>Egypt, 65</td>
</tr>
<tr>
<td>Falafel, 78-90</td>
</tr>
<tr>
<td>Giraffes, 45-59</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Table of Contents
1. Science of Cognition  
   1.a. Motivations  
   1.a.i. Intellectual Curiosity  
   1.a.ii. Practical Applications  
   1.b. History of Cognitive Psychology  
2. The Neural Basis of Cognition  
   2.a. The Nervous System  
   2.b. Organization of the Brain  
   2.c. The Visual System  
3. Perception and Attention  
   3.a. Sensory Memory  
   3.b. Attention and Sensory Information Processing

Corpus analysis/navigation

- Given a corpus, partition it into groups of related docs  
  - Recursively, can induce a tree of topics  
  - Allows user to browse through corpus to find information  
  - Crucial need: meaningful labels for topic nodes.
- Yahoo!: manual hierarchy  
  - Often not available for new document collection
For improving search recall

- *Cluster hypothesis* - Documents with similar text are related
- Therefore, to improve search recall:
  - Cluster docs in corpus a priori
  - When a query matches a doc $D$, also return other docs in the cluster containing $D$
- Hope if we do this: The query “car” will also return docs containing *automobile*
  - Because clustering grouped together docs containing *car* with those containing *automobile*.

Why might this happen?
For better navigation of search results

• For grouping search results thematically
  – clusty.com / Vivisimo

For better navigation of search results

• And more visually: Kartoo.com
Navigating search results (2)

• One can also view grouping documents with the same sense of a word as clustering
• Given the results of a search (e.g., jaguar, NLP), partition into groups of related docs
• Can be viewed as a form of word sense disambiguation
• E.g., jaguar may have senses:
  – The car company
  – The animal
  – The football team
  – The video game
• Recall query reformulation/expansion discussion
For speeding up vector space retrieval

- In vector space retrieval, we must find nearest doc vectors to query vector
- This entails finding the similarity of the query to every doc – slow (for some applications)
- By clustering docs in corpus a priori
  - find nearest docs in cluster(s) close to query
  - inexact but avoids exhaustive similarity computation

What Is A Good Clustering?

- Internal criterion: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used
- External criterion: The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns or latent classes
  - Assessable with gold standard data
External Evaluation of Cluster Quality

- Assesses clustering with respect to ground truth
- Assume that there are $C$ gold standard classes, while our clustering algorithms produce $k$ clusters, $\pi_1$, $\pi_2$, ..., $\pi_k$ with $n_i$ members.
- Simple measure: purity, the ratio between the dominant class in the cluster $\pi_i$ and the size of cluster $\pi_i$

$$Purity(\pi_i) = \frac{1}{n_i} \max_{j} (n_{ij}) \quad j \in C$$

- Others are entropy of classes in clusters (or mutual information between classes and clusters)

Purity

Cluster I: Purity = $1/6$ (max(5, 1, 0)) = 5/6

Cluster II: Purity = $1/6$ (max(1, 4, 1)) = 4/6

Cluster III: Purity = $1/5$ (max(2, 0, 3)) = 3/5
Issues for clustering

- Representation for clustering
  - Document representation
    - Vector space? Normalization?
  - Need a notion of similarity/distance

- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid “trivial” clusters - too large or small
      - In an application, if a cluster’s too large, then for navigation purposes you’ve wasted an extra user click without whittling down the set of documents much.
What makes docs “related”?  

- Ideal: semantic similarity.  

- Practical: statistical similarity  
  - We will use cosine similarity.  
  - Docs as vectors.  
  - For many algorithms, easier to think in terms of a distance (rather than similarity) between docs.  
  - We will describe algorithms in terms of cosine similarity.  

Cosine similarity of normalized $D_j, D_k$:  

$$sim(D_j, D_k) = \sum_{i=1}^{m} w_{ij} \times w_{ik}$$  
Aka normalized inner product.

Recall doc as vector  

- Each doc $j$ is a vector of $tf \times idf$ values, one component for each term.  

- Can normalize to unit length.  

- So we have a vector space  
  - terms are axis - aka features  
  - $n$ docs live in this space  
  - even with stemming, may have 20,000+ dimensions  
  - do we really want to use all terms?  
    - Different from using vector space for search. Why?
**Intuition**

Postulate: Documents that are “close together” in vector space talk about the same things.

**Clustering Algorithms**

- **Partitioning “flat” algorithms**
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - $k$ means/medoids clustering
    - Model based clustering

- **Hierarchical algorithms**
  - Bottom-up, agglomerative
  - Top-down, divisive
Partitioning Algorithms

- Partitioning method: Construct a partition of $n$ documents into a set of $k$ clusters
- Given: a set of documents and the number $k$
- Find: a partition of $k$ clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic methods: k-means and k-medoids algorithms

How hard is clustering?

- One idea is to consider all possible clusterings, and pick the one that has best inter and intra cluster distance properties
- Suppose we are given $n$ points, and would like to cluster them into $k$-clusters
  - How many possible clusterings?
    \[
    \frac{k^n}{k!}
    \]
- Too hard to do it brute force or optimally
- Solution: Iterative optimization algorithms
  - Start with a clustering, iteratively improve it (e.g., K-means)
K-Means

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

\[
\mu(c) = \frac{1}{|c|} \sum_{x \in c} \bar{x}
\]

• 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

Let \( d \) be the distance measure between instances.
Select \( k \) random instances \( \{s_1, s_2, \ldots, s_k\} \) as seeds.
Until clustering converges or other stopping criterion:
  
  For each instance \( x_i \):
  
  Assign \( x_i \) to the cluster \( c_j \) such that \( d(x_i, s_j) \) is minimal.

  *Update the seeds to the centroid of each cluster*

  For each cluster \( c_j \)

  \( s_j = \mu(c_j) \)
K Means Example
(K=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!

Termination conditions

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

Does this mean that the docs in a cluster are unchanged?
Time Complexity

• Assume computing distance between two instances is $O(m)$ where $m$ is the dimensionality of the vectors.
• Reassigning clusters: $O(kn)$ distance computations, or $O(knm)$.
• Computing centroids: Each instance vector gets added once to some centroid: $O(nm)$.
• Assume these two steps are each done once for $i$ iterations: $O(iknm)$.
• Linear in all relevant factors, assuming a fixed number of iterations, more efficient than hierarchical agglomerative methods.

Seed Choice

• Results can vary based on random seed selection.
• Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  – Select good seeds using a heuristic (e.g., doc least similar to any existing mean)
  – Try out multiple starting points
  – Initialize with the results of another method.

Example showing sensitivity to seeds

Exercise: find good approach for finding good starting points
Recap

• Why cluster documents?
  – For improving recall in search applications
  – For speeding up vector space retrieval
  – Navigation
  – Presentation of search results
• $k$-means basic iteration
  – At the start of the iteration, we have $k$ centroids.
  – Each doc assigned to the nearest centroid.
  – All docs assigned to the same centroid are averaged to compute a new centroid;
    • thus have $k$ new centroids.

How Many Clusters?

• Number of clusters $k$ is given
  – Partition $n$ docs into predetermined number of clusters
• Finding the “right” number of clusters is part of the problem
  – Given docs, partition into an “appropriate” number of subsets.
  – E.g., for query results - ideal value of $k$ not known up front - though UI may impose limits.
• Can usually take an algorithm for one flavor and convert to the other.
$k$ not specified in advance

- Say, the results of a query.
- Solve an optimization problem: penalize having lots of clusters
  - application dependent, e.g., compressed summary of search results list.
- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters

---

$K$ not specified in advance

- Given a clustering, define the **Benefit** for a doc to be the cosine similarity to its centroid
- Define the **Total Benefit** to be the sum of the individual doc Benefits.

Why is there always a clustering of Total Benefit $n$?
Penalize lots of clusters

- For each cluster, we have a Cost $C$.
- Thus for a clustering with $k$ clusters, the Total Cost is $kC$.
- Define the Value of a clustering to be $\text{Value} = \text{Total Benefit} - \text{Total Cost}$.
- Find the clustering of highest value, over all choices of $k$.
  - Total benefit increases with increasing $K$. But can stop when it doesn’t increase by “much”. The Cost term enforces this.

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.
Convergence of K-Means

- Define goodness measure of cluster k as sum of squared distances from cluster centroid:
  \[ G_k = \sum_i (v_i - c_k)^2 \] (sum all \( v_i \) in cluster k)
- \( G = \sum_k G_k \)
- Reassignment monotonically reduces \( G \) since each vector is assigned to the closest centroid.
- Recomputation monotonically decreases each \( G_k \) since: (\( m_k \) is number of members in cluster)
  \[ \sum (v_{in} - a)^2 \text{ reaches minimum for:} \]
  \[ \sum -2(v_{in} - a) = 0 \]

K-means issues, variations, etc.

- Recomputing the centroid after every assignment (rather than after all points are re-assigned) can improve speed of convergence of K-means
- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.
- Disjoint and exhaustive
  - Doesn’t have a notion of “outliers”
Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- **Soft clustering** gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (**dendrogram**) from a set of unlabeled examples.

![Dendrogram example with categories and subcategories]

- One option to produce a hierarchical clustering is recursive application of a partitional clustering algorithm to produce a hierarchical clustering.
“The Curse of Dimensionality”

• Why document clustering is difficult
  – While clustering looks intuitive in 2 dimensions, many of our applications involve 10,000 or more dimensions…
  – High-dimensional spaces look different: the probability of random points being close drops quickly as the dimensionality grows.
  – One way to look at it: in large-dimension spaces, random vectors are almost all almost perpendicular. Why?
• Solution: Dimensionality reduction … important for text

Related Tasks

• TDT
  – Topic Detection: “Dynamic” Clustering
  – Topic Tracking: on-line categorization
  – Story Segmentation
  – First Story Detection
  – New Information Detection
  – Story Link Detection
• TIDES
  – All of the above in multilingual and multimedia
• Word cloud
• And others…