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

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

Text Categorization: Evaluation

Contingency Table Per Category (for all docs)

\[ n: \text{total number of docs} \]

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

Idea: find your language by what language your neighbors speak

Use \( K \) nearest neighbors to vote

1-NN:Red; 5-NN:Brown; 10-NN:?; 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  \( D = \{(x_i, y_i)\}, \ x_i \in \mathbb{R}^M, \text{docs}, \ y_i \in \{0,1\} \)

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

The neighborhood is  \( D_k \in D \)

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

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

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

- **Euclidean distance**: $d(x_1, x_2) = \sqrt{\sum_v (x_{1v} - x_{2v})^2}$
- **Kullback-Leibler distance**: $d(x_1, x_2) = \sum_v x_{1v} \log \frac{x_{1v}}{x_{2v}}$
- **Dot product**: $\vec{x}_1 \cdot \vec{x}_2 = \sum_v x_{1v} * x_{2v}$
- **Cosine Similarity**: $\cos(x_1, x_2) = \frac{\sum_v x_{1v} * x_{2v}}{\sqrt{\sum_v x_{1v}^2} \sqrt{\sum_v x_{2v}^2}}$
- **Kernel functions**: $k(x_1, x_2) = e^{-d(x_1, x_2)^2/2\sigma^2}$ (Gaussian Kernel)

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: \( \tilde{d}_{1,1}, \tilde{d}_{1,2}, \ldots, \tilde{d}_{1,n_1} \) \( \rightarrow \) Language model \( \theta_1 \)
    Category 2: \( \tilde{d}_{2,1}, \tilde{d}_{2,2}, \ldots, \tilde{d}_{2,n_2} \) \( \rightarrow \) Language model \( \theta_2 \)
    .......
    Category C: \( \tilde{d}_{C,1}, \tilde{d}_{C,2}, \ldots, \tilde{d}_{C,n_k} \) \( \rightarrow \) Language model \( \theta_C \)
  – What is the language model? (Multinomial distribution)
  – How to estimate the language model for all the documents in one category?

• 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
    \[
p(\tilde{d}_{c,1}, \ldots, \tilde{d}_{c,n_k} | \theta_c)
    \]
  – 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:

$$\theta^*_c = \arg\max_{\theta_c} p(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta_c)$$

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

$$\theta^*_c = \arg\max_{\theta_c} p(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta_c)$$

---

Maximum Likelihood Estimation (MLE)

$$p(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta) = \prod_{i=1}^{n_c} \left( \frac{|\vec{d}_c|}{c_i(w_1) \ldots c_i(w_k)} \right) \prod_{k=1}^{K} p_{ik}^{c_i(w_k)} \propto \prod_{i=1}^{n_c} \prod_{k=1}^{K} p_{ik}^{c_i(w_k)}$$

$$l(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta) = \log p(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta) = \sum_{i=1}^{n_c} \sum_{k=1}^{K} c_i(w_k) \log p_k$$

$$l(\vec{d}_{c1}, \ldots, \vec{d}_{cn} | \theta) = \sum_{i=1}^{n_c} \sum_{k=1}^{K} c_i(w_k) \log \theta_k + \lambda(\sum_k p_k - 1)$$

$$\frac{\partial l}{\partial p_k} = \frac{\sum_{i=1}^{n_c} c_i(w_k)}{p_k} + \lambda = 0 \Rightarrow p_k = -\frac{\sum_{i=1}^{n_c} c_i(w_k)}{\lambda}$$

Use Lagrange multiplier approach
Set partial derivatives to zero
Get maximum likelihood estimate

Since $\sum_k p_k = 1$, $\lambda = -\sum_k \sum_i c_i(w_k) = -\sum_i |\vec{d}_c|$ So, $p_k = p(w_k) = \frac{\sum_{i=1}^{n_c} c_i(w_k)}{\sum_i |\vec{d}_c|}$
Naïve Bayes Classification

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

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

---

Naïve Bayes Classification

- **Smoothed Estimator**:
  - Laplace Smoothing
  \[
p(w \mid \theta^*_c) = \frac{1 + \sum_{i=1}^{n_c} c_{ci}(w)}{K + \sum_{i=1}^{n_c} \left| d_{ci} \right|}
\]
  - Hierarchical Smoothing
  \[
p(w \mid \theta^*_c) = \lambda_1 P(w \mid \theta^*_c) + \lambda_2 P(w \mid \theta^*_{c_{ap1}}) + \ldots + \lambda_m P(w \mid \theta^*_{c_{non}})
\]
  - Dirichlet Smoothing
Naïve Bayes Classification

Prediction:
\[ c^* = \arg\max_c p(c \mid d_i) \]
\[ = \arg\max_c \left \{ \frac{p(c)p(d_i \mid c)}{p(d_i)} \right \} \]
\[ = \arg\max_c \left \{ p(c)p(d_i \mid c) \right \} \quad \text{(Bayes Rule)} \]
\[ = \arg\max_c \left \{ p(c) \prod_k p(w_k \mid c)^{c(w_k)} \right \} \quad \text{(Multinomial Dist)} \]
\[ = \arg\max_c \left \{ \log(p(c)) + \sum_k c_i(w_k) \log(p(w_k \mid c)) \right \} \]

Plug in the estimator

---

Naïve Bayes Classification

Example of Binary Classification

Two classes
\[ c^* = \arg\max_{l \in \{-,+\}} p(c_l \mid d_i) \rightarrow \frac{p(c_+ \mid d_i)}{p(c_- \mid d_i)} \]
\[ p(c_+ \mid d_i) \propto \prod_k [p(w_k \mid c_+)]^{c(w_k)} \frac{n_+}{n_+ + n_-} \]
\[ p(c_- \mid d_i) \propto \prod_k [p(w_k \mid c_-)]^{c(w_k)} \frac{n_-}{n_+ + n_-} \]
Naïve Bayes Classification

- Example of Binary Classification

\[ c^* = \arg \max_{l \in \{-1, +1\}} p(c_l \mid \vec{d}_i) \Rightarrow \frac{p(c_+ \mid \vec{d}_i)}{p(c_- \mid \vec{d}_i)} \]

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

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

\[ \log \frac{p(c_+ \mid \vec{d})}{p(c_- \mid \vec{d})} \propto b_0 + \sum_k c_i(w_k) \times \text{weight}(w_k) \]

Naïve Bayes = Linear Classifier

- \( \ast \) denotes +1
- \( \ast \) 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)
Text Categorization (III)

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:

\[ m = \frac{2}{\|w\|} \]

Closest positive data point to boundary:

\[ W^T X_i + b = 1 \]

Closest negative data point to boundary:

\[ W^T X_j + b = -1 \]
Linear SVM

- Let \( \{x_1, \ldots, x_n\} \) 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:

\[
\text{Minimize } \frac{1}{2} ||w||^2 \\
\text{subject to } y_i (w^T x_i + b) \geq 1 \quad \forall i
\]

The Karush-Kuhn-Tucker Condition

- The optimal solution of model parameter satisfies:

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

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

- 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

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

The optimization problem becomes

Minimize \( \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i \)

subject to \( y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \)

---

Soft Margin Linear SVM

• The dual of the problem for soft margin linear SVM is:

\[
\text{max. } W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

subject to \( C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \)

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

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

Once again, a QP solver can be used to find \( \alpha_i \)
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
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)