Introduction to Information Retrieval

CS276: Information Retrieval and Web Search
Christopher Manning and Pandu Nayak

Lecture 14: Support vector machines and machine learning on documents

[Introduces slides from Ray Mooney]

Text classification: Up until now and today

- Previously: 3 algorithms for text classification
  - Naive Bayes classifier
  - K Nearest Neighbor classification
    - Simple, expensive at test time, high variance, non-linear
  - Vector space classification using centroids and hyperplanes that split them
    - Simple, linear discriminant classifier; perhaps too simple

- Today
  - SVMs
  - Some empirical evaluation and comparison
  - Text-specific issues in classification

Linear classifiers: Which Hyperplane?

- Lots of possible solutions for a, b, c.
- Some methods find a separating hyperplane, but not the optimal one (according to some criterion of expected goodness)
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the “difficult points” close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

Support Vector Machine (SVM)

- SVMs maximize the margin around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, the support vectors.
- Solving SVMs is a quadratic programming problem
- Seen by many as the most successful current text classification method*

Maximum Margin: Formalization

- \( w \): decision hyperplane normal vector
- \( x_i \): data point \( i \)
- \( y_i \): class of data point \( i \) (+1 or -1) NB: Not 1/0
- Classifier is: \( f(x) = \text{sign}(w^T x + b) \)
- Functional margin of \( x_i \) is: \( y_i (w^T x_i + b) \)
  - But note that we can increase this margin simply by scaling \( w, b \)...
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

*but other discriminative methods often perform very similarly
Geometric Margin

- Distance from example to the separator is $r = \frac{w^T x + b}{||w||}$.
- Examples closest to the hyperplane are support vectors.
- Margin $\rho$ of the separator is the width of separation between support vectors of classes.

Derivation of finding $r$:

- Dotted line $x' - x$ is perpendicular to decision boundary so parallel to $w$.
- Unit vector is $w/||w||$, so line is $rw/||w||$.
- $x' = x - yrw/||w||$.
- $x'$ satisfies $w^T x' + b = 0$.
- So $w^T (x - yrw/||w||) + b = 0$.
- Recall that $||w|| = \sqrt{w^T w}$.
- So $w^T x + yrw = b = 0$.
- So, solving for $r$ gives: $r = yrw/ ||w||$.

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(x_i, y_i)\}$

$$w^T x_i + b \geq 1 \quad \text{if} \quad y_i = 1$$
$$w^T x_i + b \leq -1 \quad \text{if} \quad y_i = -1$$

- For support vectors, the inequality becomes an equality.
- Then, since each example's distance from the hyperplane is $\rho = \frac{w^T x_i + b}{||w||}$
- The margin is $\rho = \frac{2}{||w||}$

Linear Support Vector Machine (SVM)

- Hyperplane $w^T x + b = 0$.
- Extra scale constraint: $\min_{x_i} |w^T x_i + b| = 1$.
- This implies $w^T (x_i-x_j) = 2$.
- $\rho = |w^T x_i + b| = 2/||w||$.

Linear SVMs Mathematically (cont.)

- Then we can formulate the quadratic optimization problem:

Find $w$ and $b$ such that $\rho = \frac{2}{||w||}$ is maximized; and for all $\{(x_i, y_i)\}$

$$w^T x_i + b \geq 1 \quad \text{if} \quad y_i = 1, \quad w^T x_i + b \leq -1 \quad \text{if} \quad y_i = -1$$

- A better formulation (min $||w||$ = max $1/||w||$):

Find $w$ and $b$ such that $Q(w) = \frac{1}{2} ||w||^2$ is minimized; and for all $\{(x_i, y_i)\}$:

$$y_i (w^T x_i + b) \geq 1$$

Solving the Optimization Problem

Find $w$ and $b$ such that:

- This is now optimizing a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intractate) algorithms exist for solving them (with many special ones built for SVMs).
- The solution involves constructing a dual problem where a Lagrange multiplier $a_i$ is associated with every constraint in the primary problem:

Find $a_i \geq 0$, such that $Q(a) = \sum_i a_i - \frac{1}{2} \sum_i \sum_{j \neq i} y_i y_j a_i a_j x_i^T x_j$ is maximized and:

1. $\sum_i a_i y_i = 0$
2. $a_i \geq 0$ for all $i$

The Optimization Problem Solution

- The solution has the form:

$$w = \sum_i a_i y_i x_i$$
$$b = \frac{1}{||w||} \sum_i a_i y_i$$

- Each non-zero $a_i$ indicates that corresponding $x_i$ is a support vector.
- Then the classifying function will have the form:

$$f(x) = \sum_i a_i y_i x_i^T x + b$$

Notice that it relies on an inner product between the test point $x$ and the support vectors $x_i$.
- We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all pairs of training points.
Soft Margin Classification

- If the training data is not linearly separable, slack variables $\xi$ can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
  - Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)

Mathematically!

The dual problem for soft margin classification:

Find $a_1, \ldots, a_N$ such that $Q(a) = \sum_{i=1}^N \sum_{j \neq i} y_i y_j a_i a_j x_i x_j$ is maximized and

1. $\sum_i y_i a_i = 0$
2. $0 \leq a_i \leq C$ for all $i$

- Neither slack variables $\xi$ nor their Lagrange multipliers appear in the dual problem!
- Again, $x_i$ with non-zero $a_i$ will be support vectors.
- Solution to the dual problem is:

$$w = \sum_i a_i y_i x_i$$
$$b = y_k (1 - \sum_i a_i y_i x_i)$$

where $k = \arg \max_{i \neq k} a_i$.

Soft Margin Classification – Solution

$$w \text{ is not needed explicitly for classification!}$$

$$f(x) = \sum_i a_i y_i x_i + b$$

Classification with SVMs

- Given a new point $x$, we can score its projection onto the hyperplane normal:

$$w^T x + b = \sum_{i=1}^N a_i y_i x_i^T x + b$$

- Decide class based on whether $\langle x, w \rangle + b > 0$
- Can set confidence threshold $t$.

Score $> t$: yes
Score $< -t$: no
Else: don’t know

Linear SVMs: Summary

- The classifier is a separating hyperplane.
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $x_i$ are support vectors with non-zero Lagrangian multipliers $a_i$.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

$$f(x) = \sum_i a_i y_i x_i^T x + b$$

Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:
- But what are we going to do if the dataset is just too hard?
- How about … mapping data to a higher-dimensional space:
A large trade show, in conjunction with the congress, will feature the latest in technology in all control and eradication programs, the NPPC said.

The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) eradication program, the NPPC said.

Delegates to the three-day congress will be considering 26 resolutions concerning various issues, according to the National Pork Producers Council, NPPC.

The congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nation’s pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

### The “Kernel Trick”

- The linear classifier relies on an inner product between vectors $K(x_{ij}) = x_i^T x_j$.
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: x \mapsto \Phi(x)$, the inner product becomes:
  
  $$ K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) $$

- A kernel function is some function that corresponds to an inner product in some expanded feature space.

**Example:**

2-dimensional vectors $x=(x_1, x_2)$; let $K(x_{ij}) = (1 + \langle x_i | x_j \rangle)^d$.

Need to show that $K(x_{ij}) = \Phi(x_i)^T \Phi(x_j)$:

$$ K(x_{ij}) = (1 + \langle x_i | x_j \rangle)^d = 1 + x_i^T x_j + \binom{d}{2} x_i^T x_j^2 + \binom{d}{3} x_i^T x_j^3 + \cdots $$

$$ = \Phi(x_i)^T \Phi(x_j) $$

### Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3399 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 topics, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
  - Only about 10 out of 118 categories are large

#### Common categories ($#\text{train}$, $#\text{test}$)

- **Earn** (2307, 1087)
- **Acquisitions** (1630, 179)
- **MoxyRx** (538, 179)
- **Grain** (433, 149)
- **Crude** (389, 189)
- **Trade** (369, 119)
- **Interest** (350, 131)
- **Ship** (197, 99)
- **Whale** (212, 71)
- **Corn** (182, 56)

### Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

$$ \Phi: x \mapsto \phi(x) $$

$$ K(x_i, x_j) = e^{-||x_i - x_j||^2 / 2\sigma^2} $$

- Haven’t been very useful in text classification

### Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space

- Common kernels
  - Linear
  - Polynomial $K(x, z) = (1+\langle x | z \rangle)^d$
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)
    $$ K(x_i, x_j) = e^{-||x_i - x_j||^2 / 2\sigma^2} $$

### Reuters Text Categorization data set (Reuters-21578) document

- User: F. Lewis
- Date: 2 Mar 1987
- Topics: "YES" LEWISPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="11981" NEWSID="798"
- Title: AMERICAN PORK CONGRESS KICKS OFF TOMORROW
- DateLine: CHICAGO, March 2
- TimeLine: The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 100 of the nation’s pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three-day congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. 

### Per class evaluation measures

- Recall: Fraction of docs in class $i$ classified correctly:

$$ \frac{C_i}{ \sum_j C_j } $$

- Precision: Fraction of docs assigned class $i$ that are actually about class $i$:

$$ \frac{C_i / \sum_j C_j}{ \sum_i C_i / \sum_j C_j } $$

- Accuracy: (1 - error rate) Fraction of docs classified correctly:

$$ \sum_i C_i / \sum_{ij} C_{ij} $
### Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
- Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

#### Micro- vs. Macro-Averaging: Example

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 2</th>
<th>Micro Ave. Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth: yes</td>
<td>Truth: no</td>
<td>Truth: yes</td>
</tr>
<tr>
<td>Classifier: yes</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>10</td>
<td>970</td>
</tr>
</tbody>
</table>

- Macroaveraged precision: \((0.5 + 0.9)/2 = 0.7\)
- Microaveraged precision: \(100/120 = .83\)

- Microaveraged score is dominated by score on common classes

#### Precision-recall for category: Crude

<table>
<thead>
<tr>
<th>Evaluation measure: (F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) NB</td>
</tr>
<tr>
<td>micro-avg-L (90 classes)</td>
</tr>
<tr>
<td>macro-avg (90 classes)</td>
</tr>
<tr>
<td>(b) NB</td>
</tr>
<tr>
<td>earn</td>
</tr>
<tr>
<td>acq</td>
</tr>
<tr>
<td>money-fx</td>
</tr>
<tr>
<td>grain</td>
</tr>
<tr>
<td>crude</td>
</tr>
<tr>
<td>trade</td>
</tr>
<tr>
<td>interest</td>
</tr>
<tr>
<td>ship</td>
</tr>
<tr>
<td>wheat</td>
</tr>
<tr>
<td>corn</td>
</tr>
<tr>
<td>micro-avg (top 10)</td>
</tr>
<tr>
<td>micro-avg (113 classes)</td>
</tr>
</tbody>
</table>

#### Precision-recall for category: Ship

<table>
<thead>
<tr>
<th>Evaluation measure: (F_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dumais (1998)</td>
</tr>
</tbody>
</table>

#### Yang&Liu: SVM vs. Other Methods

<table>
<thead>
<tr>
<th>Table 1: Performance summary of classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
</tr>
<tr>
<td>SVM</td>
</tr>
<tr>
<td>KNN</td>
</tr>
<tr>
<td>LSF</td>
</tr>
<tr>
<td>NNet</td>
</tr>
<tr>
<td>NB</td>
</tr>
</tbody>
</table>

- \(miR = micro-avg\) recall, \(miP = micro-avg\) prec., \(miF1 = micro-avg\) F1, \(maF1 = macro-avg\) F1.
Good practice department: Make a confusion matrix

This \((i, j)\) entry means \(53\) of the docs actually in class \(i\) were put in class \(j\) by the classifier.

- In a perfect classification, only the diagonal has non-zero entries
- Look at common confusions and how they might be addressed

The Real World

- “There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate intranets, government departments, and Internet publishers”

- “Understanding the data is one of the keys to successful categorization, yet this is an area in which most categorization tool vendors are extremely weak. Many of the ‘one size fits all’ tools on the market have not been tested on a wide range of content types.”

The Real World

- Gee, I’m building a text classifier for real, now!
- What should I do?
- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Very little data?

- If you’re just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, ...
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- Perfect!
- We can use all our clever classifiers
- Roll out the SVM!

- But if you are using an SVM/NB etc., you should probably be prepared with the “hybrid” solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately
A huge amount of data?

- This is great in theory for doing accurate classification...
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical
- Naïve Bayes can come back into its own again!
  - Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)

Accuracy as a function of data size

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Data: Brill and Banko on context-sensitive spelling correction
- But the fact that you have to keep doubling your data to improve performance is a little unpleasant

How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
    - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
    - Mileage fairly unclear, but helps a bit (Tie-Yen Liu et al. 2005)
    - May need a hybrid automatic/manual solution

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn’t be.
  - E.g., part numbers, chemical formulas
- Does putting in “hacks” help?
  - You bet!
    - Feature design and non-linear weighting is very important in the performance of real-world systems

Upweighting

- You can get a lot of value by differentially weighting contributions from different document zones:
- That is, you count as two instances of a word when you see it in, say, the abstract
  - Upweighting title words helps (Cohen & Singer 1996)
  - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko et al, 2002)

Two techniques for zones

1. Have a completely separate set of features/parameters for different zones like the title
2. Use the same features (pooling/tying their parameters) across zones, but upweight the contribution of different zones

- Commonly the second method is more successful: it costs you nothing in terms of sparsifying the data, but can give a very useful performance boost
  - Which is best is a contingent fact about the data
Text Summarization techniques in text classification

- Text Summarization: Process of extracting key pieces from text, normally by features on sentences reflecting position and content
- Much of this work can be used to suggest weightings for terms in text categorization
  - See: Kolcz, Prabakarurthi, and Kalita, CKM 2001: Summarization as feature selection for text categorization
  - Categorizing purely with title,
  - Categorizing with first paragraph only
  - Categorizing with paragraph with most keywords
  - Categorizing with first and last paragraphs, etc.

Measuring Classification

Not just accuracy; in the real world, there are economic measures:

- Your choices are:
  - Do no classification
  - That has a cost (hard to compute)
  - Do it all manually
    - Has an easy-to-compute cost if doing it like that now
  - Do it all with an automatic classifier
    - Even more cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/new cases

A common problem: Concept Drift

- Categories change over time
- Example: "president of the united states"
  - 1999: clinton is great feature
  - 2010: clinton is bad feature

Does stemming/lowercasing/... help?

- As always, it’s hard to tell, and empirical evaluation is normally the gold standard
- But note that the role of tools like stemming is rather different for TextCat vs. IR:
  - For IR, you often want to collapse forms of the verb oxygenate and oxygenation, since all of those documents will be relevant to a query for oxygenation
  - For TextCat, with sufficient training data, stemming does no good. It only helps in compensating for data sparseness (which can be severe in TextCat applications). Overly aggressive stemming can easily degrade performance.

Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
  - Can regularize logistic regression (Zhang & Oles 2001)
  - SVM is powerful and elegant way to define similarity metric
  - Perhaps best performing text classifier

- Partly popular due to availability of good software
  - SVMlight is accurate and fast – and free (for research)
  - Now lots of good software: LibSVM, TinySVM, ...
  - Comparative evaluation of methods
  - Real world: exploit domain specific structure!

Resources for today’s lecture

- Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition