Text classification

- Last week: 3 algorithms for text classification
  - Naive Bayes classifier
    - Simple, cheap, high bias, linear
  - K Nearest Neighbor classification
    - Simple, expensive at test time, high variance, non-linear
  - Vector space classification: Rocchio
    - Simple linear discriminant classifier; perhaps too simple
- Today
  - Support Vector Machines (SVMs)
  - Soft margin SVMs and kernels for non-linear classifiers
  - Some empirical evaluation and comparison
  - Text-specific issues in classification

Linear classifiers

A hyperplane decision boundary

- Lots of possible choices 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
  - A 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
  - The decision boundary has a “margin”

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: data point i
- y: 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 $x_i$: $y_i (w^T x + b)$
- The functional margin of a dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin
- Problem: we can increase this margin simply by scaling $w$, $b$...
**Geometric Margin**

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

![Geometric Margin Diagram](image)

**Linear SVM Mathematically**

The linearly separable case

- Assume that the functional margin of each data item is at least 1, then the following two constraints follow for a training set $(x_i, y_i)$:
  
  $w^Tx_i + b \geq 1$ if $y_i = 1$
  
  $w^Tx_i + b \leq -1$ if $y_i = -1$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is $r = \frac{w^Tx + b}{||w||}$
- The margin is: $\rho = \frac{2}{||w||}$

**Worked example: Geometric margin**

- Maximum margin weight vector is parallel to line from $(1, 1)$ to $(2, 3)$. So weight vector is $(1, 2)$.
- Decision boundary is normal (“perpendicular”) to it halfway between.
- It passes through $(1.5, 2)$
- So $y = x_1 + 2x_2 = 5.5$
- Geometric margin is $\sqrt{5}$

**Linear SVMs Mathematically (cont.)**

- Then we can formulate the quadratic optimization problem:

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

  $w^Tx_i + b \geq 1$ if $y_i = 1$
  
  $w^Tx_i + b \leq -1$ if $y_i = -1$

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

  Find $w$ and $b$ such that $\Phi(w) = \frac{1}{2}w^Tw$ is minimized;

  and for all $(x_i, y_i)$: $y_i(w^Tx_i + b) \geq 1$
Introducing Information Retrieval

Solving the Optimization Problem

Find \( w \) and \( b \) such that:
\[ \Phi(w) = \frac{1}{2} w^T w \text{ is minimized; and for all } \{(x_i, y_i)\} : y_i (w^T x_i + b) \geq 1 \]

- 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 (instructive) algorithms exist for solving them (with many special ones built for SVMs: SMO, Pegasos, …)
- The solution usually involves constructing a dual problem where a Lagrange multiplier \( a_i \) is associated with every constraint in the primary problem:

\[ \begin{align*}
\Omega(a_i) &= \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j x_i^T x_j \\
\text{subject to } &\sum_i a_i y_i = 0 \\
&0 \leq a_i \leq C \text{ for all } i \\
\end{align*} \]

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.
- 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)

Soft Margin Classification

- The old formulation:

\[ \begin{align*}
\Phi(w) &= \frac{1}{2} w^T w \\
\text{subject to } &y_i (w^T x_i + b) \geq 1 \\
\end{align*} \]

- The new formulation incorporating slack variables:

\[ \begin{align*}
\Phi(w) &= \frac{1}{2} w^T w + \sum_i \xi_i \\
\text{subject to } &y_i (w^T x_i + b) \geq 1 - \xi_i \\
&\xi_i \geq 0 \text{ for all } i \\
\end{align*} \]

- Parameter \( C \) can be viewed as a way to control overfitting
  - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

\[ \begin{align*}
\Phi(a) &= \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j x_i^T x_j \\
\text{subject to } &\sum_i a_i y_i = 0 \\
&0 \leq a_i \leq C \text{ for all } a_i \\
\end{align*} \]

Neither slack variables \( \xi_i \) 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:

\[ \begin{align*}
w &= \sum_i a_i y_i x_i \\
b &= y_j (1 - \xi_j) \cdot w^T x_i \text{ where } k = \arg \max a_i \\
w \text{ is not needed explicitly for classification!} \\
\end{align*} \]
**Classification with SVMs**

- Given a new point \( \mathbf{x} \), we can score its projection onto the hyperplane normal:
  - i.e., compute score: \( \mathbf{w}^T \mathbf{x} + b = \sum a_i y_i (\mathbf{x}^T \mathbf{x}) + b \)
  - Decide class based on whether \( < 0 \) or \( > 0 \)

- 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 \( \mathbf{x} \) are support vectors with non-zero Lagrange multipliers \( a_i \).
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products.

\[
\begin{align*}
\mathbf{w} &= \sum a_i y_i \mathbf{\Phi}(\mathbf{x}_i) \\
\mathbf{w}^T \Phi(\mathbf{x}) + b &= 0
\end{align*}
\]

**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:
  
**The “Kernel Trick”**

- The linear classifier relies on an inner product between vectors \( K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \)
- If every datapoint is mapped into high-dimensional space via some transformation \( \Phi: \mathbf{x} \rightarrow \Phi(\mathbf{x}) \), the inner product becomes:
  
\[
K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)
\]

- A kernel function is some function that corresponds to an inner product in some expanded feature space.
- Example: 2-dimensional vectors \( \mathbf{w}[x_1, x_2] \): let \( K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^2 \)
  
Need to show that \( K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \):
  
\[
K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^2 = 1 + x_1 x_2 + 2 x_1 x_2 x_3 x_4 + x_1^2 x_2 + \ldots + 2 x_1 x_4 x_2^2 + 2 x_2 x_4 x_2^2 \\
= (1 + x_1^2 + x_2 x_2 + x_3^2 + x_4^2) (1 + x_2^2 + x_1 x_2 x_2 + x_3 x_4 + x_4 x_4) \\
= \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)
\]

**Kernels**

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial \( K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d \)
  - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

\[
K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}
\]

- Haven't been very useful in text classification
Most (over)used data set
- 21578 documents
- 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
- #train, #test

Text Classification Evaluation:
Classic Reuters-21578 Data Set
- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- Learn 118 binary category distinctions
- An article can be in more than one category
- Average document: about 90 types, 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

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</th>
<th>Truth:</th>
<th>Classifi er:</th>
<th>Micro Ave. Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yes</td>
<td>yes</td>
<td>Truth: yes</td>
</tr>
<tr>
<td>Class 1</td>
<td>10</td>
<td>Classifier: yes</td>
<td>90</td>
</tr>
<tr>
<td>Class 2</td>
<td>10</td>
<td>Classifier: no</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>yes</td>
<td>Truth: no</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Classifier: yes</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Classifier: no</td>
<td>20</td>
</tr>
</tbody>
</table>

- Macroaveraged precision: \(0.5 + 0.9)/2 = 0.7\)
- Microaveraged precision: \(100/120 = 0.83\)
- Microaveraged score is dominated by score on common classes

Micro- vs. Macro-Averaging:

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-avg: (90 classes)</td>
<td>80</td>
<td>88</td>
<td>68</td>
<td>80</td>
</tr>
<tr>
<td>macro-avg: (90 classes)</td>
<td>47</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>trees</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-avg: (top 10)</td>
<td>82</td>
<td>82</td>
<td>88</td>
<td></td>
<td>82</td>
</tr>
<tr>
<td>micro-avg: (118 classes)</td>
<td>75</td>
<td>62</td>
<td>n/a</td>
<td>n/a</td>
<td>87</td>
</tr>
</tbody>
</table>

Evaluation measure: \(F_1\)
**Introduc)on to Informa)on Retrieval**

**Sec. 15.2.4**

**Figure 1:** Precision-recall for category: Crude

- **Dumais** (1998)

**Figure 2:** Precision-recall for category: Ship

- **Dumais** (1998)

**Sec. 15.2.4**

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

- **Table 1:** Performance summary of classifiers

<table>
<thead>
<tr>
<th>method</th>
<th>miR</th>
<th>mIP</th>
<th>miF1</th>
<th>maF1</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>.8120</td>
<td>.9137</td>
<td>.8599</td>
<td>.5251</td>
<td>.00965</td>
</tr>
<tr>
<td>KNN</td>
<td>.8339</td>
<td>.8807</td>
<td>.8567</td>
<td>.5242</td>
<td>.00985</td>
</tr>
<tr>
<td>LSF</td>
<td>.8507</td>
<td>.8480</td>
<td>.8408</td>
<td>.5008</td>
<td>.00414</td>
</tr>
<tr>
<td>NNet</td>
<td>.7842</td>
<td>.8785</td>
<td>.8287</td>
<td>.3765</td>
<td>.00447</td>
</tr>
<tr>
<td>NB</td>
<td>.7688</td>
<td>.8245</td>
<td>.7956</td>
<td>.3886</td>
<td>.00544</td>
</tr>
</tbody>
</table>

- miR = micro-avg recall; mIP = micro-avg prec.; miF1 = micro-avg F1; maF1 = macro-avg F1.

**Sec. 15.3**

**The Real World**

- **P. Jackson and I. Moulinier** 2002. *Natural Language Processing for Online Applications*

  - “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.”

**Sec. 15.3.1**

**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
**Introduc)on to Information Retrieval**

**Sec. 15.3.1**

**Manually written rules**

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - Construe: 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class ... plus maintenance

**Very little data?**

- If you’re just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naive 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
  - Naive 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-Yan Liu et al. 2005)
      - Definitely helps for scalability, even if not in accuracy
    - 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

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

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)

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, Prabakaramurthi, and Kalita, CIKM 2001: Summarization as feature selection for text categorization
  - Categorizing with title,
  - Categorizing with first paragraph only
  - Categorizing with paragraph with most keywords
  - Categorizing with first and last paragraphs, etc.

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.

Measuring Classification

Figures of Merit

- 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
      - Mistakes have a cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/"new" cases
  - Commonly the last method is most cost efficient and is adopted
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
- One measure of a text classification system is how well it protects against concept drift.
  - Favors simpler models like Naïve Bayes
- Feature selection: can be bad in protecting against concept drift

Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
  - Support vector = “critical” point close to decision boundary
  - (Degree-1) SVMs are linear classifiers.
  - Kernels: powerful and elegant way to define similarity metric
  - Perhaps best performing text classifier
  - But there are other methods that perform about as well as SVM, such as regularised logistic regression [Zhang & Oles 2001]
- Partly popular due to availability of good software
  - SVMlight is accurate and fast – and free (for research)
- Perhaps best performing text classifier
  - But there are other methods that perform about as well as SVM, such as regularized logistic regression [Zhang & Oles 2001]
- 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
- Yiming Yang, Xin Liu. 1999. A re-examination of text categorization methods. 22nd Annual International SIGIR
- Yiming Yang, Xin Liu. 1999. A re-examination of text categorization methods. 22nd Annual International SIGIR