Day 5: Document classifiers and supervised scaling models

Kenneth Benoit

Quantitative Analysis of Textual Data

October 28, 2014
Today’s Road Map

Supervised learning overview

Assessing the reliability of a training set

Evaluating classification: Precision and recall

Wordscores

The Naive Bayes Classifier

The k-Nearest Neighbour Classifier

Support Vector Machines (SVMs)
Classification as a goal

- Machine learning focuses on identifying classes (classification), while social science is typically interested in locating things on latent traits (scaling).
- But the two methods overlap and can be adapted – will demonstrate later using the Naive Bayes classifier.
- Applying lessons from machine to learning to supervised scaling, we can
  - Apply classification methods to scaling
  - Improve it using lessons from machine learning
Supervised v. unsupervised methods compared

The goal (in text analysis) is to differentiate documents from one another, treating them as “bags of words”

Different approaches:

- *Supervised methods* require a training set that exemplify contrasting classes, identified by the researcher
- *Unsupervised methods* scale documents based on patterns of similarity from the term-document matrix, without requiring a training step

Relative advantage of supervised methods: You already know the dimension being scaled, because you set it in the training stage

Relative disadvantage of supervised methods: You *must* already know the dimension being scaled, because you have to feed it good sample documents in the training stage
 Supervised v. unsupervised methods: Examples

General examples:

- Supervised: Naive Bayes, k-Nearest Neighbor, Support Vector Machines (SVM)
- Unsupervised: correspondence analysis, IRT models, factor analytic approaches

Political science applications

- Supervised: Wordscores (LBG 2003); SVMs (Yu, Kaufman and Diermeier 2008); Naive Bayes (Evans et al 2007)
- Unsupervised “Wordfish” (Slapin and Proksch 2008); Correspondence analysis (Schonhardt-Bailey 2008); two-dimensional IRT (Monroe and Maeda 2004)
RELIABILITY TESTING FOR THE TRAINING SET
How do we get "true" condition?

- In some domains: through more expensive or extensive tests
- In social sciences: typically by expert annotation or coding
- A scheme should be tested and reported for its reliability
Inter-rater reliability

Different types are distinguished by the way the reliability data is obtained.

<table>
<thead>
<tr>
<th>Type</th>
<th>Test Design</th>
<th>Causes of Disagreements</th>
<th>Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stability</strong></td>
<td>test-retest</td>
<td>intraobserver inconsistencies</td>
<td>weakest</td>
</tr>
<tr>
<td><strong>Reproducibility</strong></td>
<td>test-test</td>
<td>intraobserver inconsistencies + interobserver disagreements</td>
<td>medium</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>test-standard</td>
<td>intraobserver inconsistencies + deviations from a standard</td>
<td>strongest</td>
</tr>
</tbody>
</table>
Measures of agreement

- **Percent agreement** Very simple: \( \frac{\text{number of agreeing ratings}}{\text{total ratings}} \times 100\% \)

- **Correlation**
  - (usually) Pearson’s \( r \), aka product-moment correlation
  - Formula: \( r_{AB} = \frac{1}{n-1} \sum_{i=1}^{n} \left( \frac{A_i - \bar{A}}{s_A} \right) \left( \frac{B_i - \bar{B}}{s_B} \right) \)
  - May also be ordinal, such as Spearman’s rho or Kendall’s tau-b
  - Range is [0,1]

- **Agreement measures**
  - Take into account not only observed agreement, but also agreement that would have occurred by chance
  - Cohen’s \( \kappa \) is most common
  - Krippendorff’s \( \alpha \) is a generalization of Cohen’s \( \kappa \)
  - Both range from [0,1]
Reliability data matrixes

Example here used binary data (from Krippendorff)

<table>
<thead>
<tr>
<th>Article:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Coder B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- A and B agree on 60% of the articles: 60% agreement
- Correlation is (approximately) 0.10
- Observed disagreement: 4
- Expected disagreement (by chance): 4.4211
- Krippendorff’s $\alpha = 1 - \frac{D_o}{D_e} = 1 - \frac{4}{4.4211} = 0.095$
- Cohen’s $\kappa$ (nearly) identical
EVALUATING CLASSIFIER PERFORMANCE
Basic principles of machine learning: Generalization and overfitting

- **Generalization**: A classifier or a regression algorithm learns to correctly predict output from given inputs not only in previously seen samples but also in previously unseen samples.
- **Overfitting**: A classifier or a regression algorithm learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples. This causes poor prediction/generalization.
- **Goal**: The goal is to maximize the frontier of precise identification of true condition with accurate recall.
Precision and recall

- Same intuition as specificity and sensitivity earlier in course

<table>
<thead>
<tr>
<th>Prediction</th>
<th>True condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>True Positive</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative (Type II error)</td>
</tr>
</tbody>
</table>
Precision and recall and related statistics

- **Precision:** \( \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \)

- **Recall:** \( \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \)

- **Accuracy:** \( \frac{\text{Correctly classified}}{\text{Total number of cases}} \)

- **F1:** \( 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \)  
  (the harmonic mean of precision and recall)
Example: Computing precision/recall

Assume:

▷ We have a corpus where 80 documents are really positive (as opposed to negative, as in sentiment)
▷ Our method declares that 60 are positive
▷ Of the 60 declared positive, 45 are actually positive

Solution:

Precision = \( \frac{45}{45 + 15} \) = 45/60 = 0.75

Recall = \( \frac{45}{45 + 35} \) = 45/80 = 0.56
<table>
<thead>
<tr>
<th>True condition</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>80</td>
<td>60</td>
</tr>
</tbody>
</table>

Accuracy?
add in the cells we can compute

<table>
<thead>
<tr>
<th>Prediction</th>
<th>True condition</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>15</td>
</tr>
<tr>
<td>Positive</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>Negative</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
but need True Negatives and $N$ to compute accuracy

<table>
<thead>
<tr>
<th>True condition</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>Negative</td>
<td>35</td>
<td>???</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
</tr>
<tr>
<td>Negative</td>
</tr>
</tbody>
</table>

80
assume 10 True Negatives:

<table>
<thead>
<tr>
<th>True condition</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>Negative</td>
<td>35</td>
<td>10</td>
</tr>
<tr>
<td>80</td>
<td>25</td>
<td>105</td>
</tr>
</tbody>
</table>

Accuracy = (45 + 10) / 105 = 0.52

F1 = 2 * (0.75 * 0.56) / (0.75 + 0.56) = 0.64
now assume 100 True Negatives:

<table>
<thead>
<tr>
<th>True condition</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>Negative</td>
<td>35</td>
<td>100</td>
</tr>
</tbody>
</table>

Prediction:
- Positive
- Negative

Accuracy = \( \frac{45 + 100}{195} \) = 0.74

\[ F1 = 2 \times \frac{0.75 \times 0.56}{0.75 + 0.56} \] = 0.64
From Classification to Scaling

- The class predictions for a collection of words from NB can be adapted to scaling
- The intermediate steps from NB turn out to be excellent for scaling purposes, and identical to Laver, Benoit and Garry’s “Wordscores”
- There are certain things from machine learning that ought to be adopted when classification methods are used for scaling
  - Feature selection
  - Stemming/pre-processing
Wordscores conceptually

- Two sets of texts
  - Reference texts: texts about which we know something (a scalar dimensional score)
  - Virgin texts: texts about which we know nothing (but whose dimensional score we'd like to know)

- These are analogous to a “training set” and a “test set” in classification

- Basic procedure:
  1. Analyze reference texts to obtain word scores
  2. Use word scores to score virgin texts
The Wordscore Procedure
(Using the UK 1997-2001 Example)

Reference Texts

Step 1: Obtain reference texts with a priori known positions (setref)

Step 2: Generate word scores from reference texts (wordscore)

Step 3: Score each virgin text using word scores (textscore)

Step 4: (optional) Transform virgin text scores to original metric

Scored word list

Scored virgin texts

Labour
1992
5.35

Liberals
1992
8.21

Cons.
1992
17.21

Labour
1997
9.17
(.33)

Liberals
1997
5.00 (.36)

Cons.
1997
17.18 (.32)

words
15.66
corporation
15.66
inheritance
15.48
successfully
15.26
markets
15.12
motorway
14.96
nation
12.44
single
12.36
pensionable
11.59
management
11.56
monetary
10.84
secure
10.44
minorities
9.95
women
8.65
cooperation
8.64
transform
7.44
representation
7.42
poverty
6.87
waste
6.83
unemployment
6.76
contributions
6.68
The Wordscore Procedure
(Using the UK 1997-2001 Example)

Step 1: Obtain reference texts with a priori known positions (setref)
Step 2: Generate word scores from reference texts (wordscore)
Step 3: Score each virgin text using word scores (textscore)
Step 4: (optional) Transform virgin text scores to original metric
The Wordscore Procedure
(Using the UK 1997-2001 Example)

Step 1: Obtain reference texts with a priori known positions (setref)
Step 2: Generate word scores from reference texts (wordscore)
Step 3: Score each virgin text using word scores (textscore)
Step 4: (optional) Transform virgin text scores to original metric
The Wordscore Procedure
(Using the UK 1997-2001 Example)

Step 1: Obtain reference texts with a priori known positions (setref)
Step 2: Generate word scores from reference texts (wordscore)
Step 3: Score each virgin text using word scores (textscore)
Step 4: (optional) Transform virgin text scores to original metric
Wordscores mathematically: Reference texts

- Start with a set of \( I \) reference texts, represented by an \( I \times J \) document-term frequency matrix \( C_{ij} \), where \( i \) indexes the document and \( j \) indexes the \( J \) total word types.

- Each text will have an associated “score” \( a_i \), which is a single number locating this text on a single dimension of difference.
  - This can be on a scale metric, such as \( 1-20 \).
  - Can use arbitrary endpoints, such as \(-1, 1\).

- We normalize the document-term frequency matrix within each document by converting \( C_{ij} \) into a relative document-term frequency matrix (within document), by dividing \( C_{ij} \) by its word total marginals:

\[
F_{ij} = \frac{C_{ij}}{C_{i.}} \tag{1}
\]

where \( C_{i.} = \sum_{j=1}^{J} C_{ij} \).
Wordscores mathematically: Word scores

- Compute an $I \times J$ matrix of relative document probabilities $P_{ij}$ for each word in each reference text, as

$$P_{ij} = \frac{F_{ij}}{\sum_{i=1}^{I} F_{ij}}$$  \hspace{1cm} (2)

- This tells us the probability that given the observation of a specific word $j$, that we are reading a text of a certain reference document $i$
Wordscores mathematically: Word scores (example)

- Assume we have two reference texts, A and B
- The word “choice” is used 10 times per 1,000 words in Text A and 30 times per 1,000 words in Text B
- So \( F_i \) “choice” = \{.010, .030\}
- If we know only that we are reading the word choice in one of the two reference texts, then probability is 0.25 that we are reading Text A, and 0.75 that we are reading Text B

\[
P_A \text{ “choice”} = \frac{.010}{(.010 + .030)} = 0.25 \quad (3)
\]

\[
P_B \text{ “choice”} = \frac{.030}{(.010 + .030)} = 0.75 \quad (4)
\]
Wordscores mathematically: Word scores

- Compute a $J$-length “score” vector $S$ for each word $j$ as the average of each document $i$’s scores $a_i$, weighted by each word’s $P_{ij}$:

  $$S_j = \sum_{i=1}^{I} a_i P_{ij}$$  \hspace{1cm} (5)

- In matrix algebra, \( S = a \cdot P \)

- This procedure will yield a single “score” for every word that reflects the balance of the scores of the reference documents, weighted by the relative document frequency of its normalized term frequency
Continuing with our example:

- We “know” (from independent sources) that Reference Text A has a position of $-1.0$, and Reference Text B has a position of $+1.0$
- The score of the word choice is then:
  
  $0.25(-1.0) + 0.75(1.0) = -0.25 + 0.75 = +0.50$
Wordscores mathematically: Scoring “virgin” texts

- Here the objective is to obtain a single score for any new text, relative to the reference texts.
- We do this by taking the mean of the scores of its words, weighted by their term frequency.
- So the score $v_k$ of a virgin document $k$ consisting of the $j$ word types is:
  \[ v_k = \sum_j (F_{kj} \cdot s_j) \]  
  \[ (6) \]
  where $F_{kj} = \frac{C_{kj}}{C_k}$ as in the reference document relative word frequencies.
- Note that new words outside of the set $J$ may appear in the $K$ virgin documents — these are simply ignored (because we have no information on their scores).
- Note also that nothing prohibits reference documents from also being scored as virgin documents.
Wordscores mathematically: Rescaling raw text scores

- Because of overlapping or non-discriminating words, the raw text scores will be dragged to the interior of the reference scores (we will see this shortly in the results).
- Some procedures can be applied to rescale them, either to a unit normal metric or to a more “natural” metric.
- Martin and Vanberg (2008) have proposed alternatives to the LBG (2003) rescaling.
Computing confidence intervals

- The score $v_k$ of any text represents a weighted mean.
- LBG (2003) used this logic to develop a standard error of this mean using a weighted variance of the scores in the virgin text.
- Given some assumptions about the scores being fixed (and the words being conditionally independent), this yields approximately normally distributed errors for each $v_k$.
- An alternative would be to bootstrap the textual data prior to constructing $C_{ij}$ and $C_{kj}$ — see Lowe and Benoit (2012).
Consider $J$ word types distributed across $I$ documents, each assigned one of $K$ classes.

At the word level, Bayes Theorem tells us that:

$$P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}$$

For two classes, this can be expressed as

$$= \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}$$  \hspace{1cm} (7)$$
Multinomial Bayes model of Class given a Word
Class-conditional word likelihoods

\[
P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})}
\]

- The word likelihood within class
- The maximum likelihood estimate is simply the proportion of times that word \( j \) occurs in class \( k \), but it is more common to use Laplace smoothing by adding 1 to each observed count within class
Multinomial Bayes model of Class given a Word

Word probabilities

\[
P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j)}
\]

- This represents the word probability from the training corpus
- Usually uninteresting, since it is constant for the training data, but needed to compute posteriors on a probability scale
Multinomial Bayes model of Class given a Word

Class prior probabilities

\[ P(c_k|w_j) = \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{-k})P(c_{-k})} \]

- This represents the class prior probability
- Machine learning typically takes this as the document frequency in the training set
- This approach is flawed for scaling, however, since we are scaling the latent class-ness of an unknown document, not predicting class – uniform priors are more appropriate
Multinomial Bayes model of Class given a Word

Class posterior probabilities

\[
P(c_k | w_j) = \frac{P(w_j | c_k)P(c_k)}{P(w_j | c_k)P(c_k) + P(w_j | c_{\neg k})P(c_{\neg k})}
\]

- This represents the posterior probability of membership in class \( k \) for word \( j \)
- Under *certain conditions*, this is identical to what LBG (2003) called \( P_{wr} \)
- Under those conditions, the LBG “wordscore” is the linear difference between \( P(c_k | w_j) \) and \( P(c_{\neg k} | w_j) \)
The LBG approach required the identification not only of texts for each training class, but also “reference” scores attached to each training class.

Consider two “reference” scores $s_1$ and $s_2$ attached to two classes $k = 1$ and $k = 2$. Taking $P_1$ as the posterior $P(k = 1|w = j)$ and $P_2$ as $P(k = 2|w = j)$, a generalised score $s_j^*$ for the word $j$ is then

$$s_j^* = s_1 P_1 + s_2 P_2$$
$$= s_1 P_1 + s_2 (1 - P_1)$$
$$= s_1 P_1 + s_2 - s_2 P_1$$
$$= P_1 (s_1 - s_2) + s_2$$
Moving to the document level

The “Naive” Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a “test” document, to produce:

\[
P(c|d) = P(c) \prod_j \frac{P(w_j|c)}{P(w_j)}
\]

This is why we call it “naive”: because it (wrongly) assumes:

- conditional independence of word counts
- positional independence of word counts
Naive Bayes Classification Example

(From Manning, Raghavan and Schütze, *Introduction to Information Retrieval*)

```
> Table 13.1 Data for parameter estimation examples.

docID   words in document      in c = China?
    ---                       ----------
training set  1    Chinese Beijing Chinese  yes
                2    Chinese Chinese Shanghai yes
                3    Chinese Macao    yes
                4    Tokyo Japan Chinese  no
      test set  5    Chinese Chinese Chinese Tokyo Japan  ?
```
Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors \( \hat{P}(c) = 3/4 \) and \( \hat{P}(\neg c) = 1/4 \) and the following conditional probabilities:

\[
\begin{align*}
\hat{P}(\text{Chinese}|c) &= (5 + 1)/(8 + 6) = 6/14 = 3/7 \\
\hat{P}(\text{Tokyo}|c) &= \hat{P}(\text{Japan}|c) = (0 + 1)/(8 + 6) = 1/14 \\
\hat{P}(\text{Chinese}|\neg c) &= (1 + 1)/(3 + 6) = 2/9 \\
\hat{P}(\text{Tokyo}|\neg c) &= \hat{P}(\text{Japan}|\neg c) = (1 + 1)/(3 + 6) = 2/9
\end{align*}
\]

The denominators are \((8 + 6)\) and \((3 + 6)\) because the lengths of \(text_c\) and \(text_{\neg c}\) are 8 and 3, respectively, and because the constant \(B\) in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

\[
\begin{align*}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\neg c|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{align*}
\]

Thus, the classifier assigns the test document to \(c = \text{China}\). The reason for this classification decision is that the three occurrences of the positive indicator Chinese in \(d_5\) outweigh the occurrences of the two negative indicators Japan and Tokyo.
Pros and Cons of the Wordscores approach

- Fully automated technique with minimal human intervention or judgment calls – only with regard to reference text selection
- Language-blind: all we need to know are reference scores
- Could potentially work on texts like this:

(See http://www.kli.org)
Pros and Cons of the Wordscores approach

- Estimates unknown positions on a priori scales – hence no inductive scaling with a posteriori interpretation of unknown policy space
- Very dependent on correct identification of:
  - appropriate reference texts
  - appropriate reference scores
Suggestions for choosing reference texts

- Texts need to contain information representing a clearly dimensional position
- Dimension must be known a priori. Sources might include:
  - Survey scores or manifesto scores
  - Arbitrarily defined scales (e.g. -1.0 and 1.0)
- Should be as discriminating as possible: extreme texts on the dimension of interest, to provide reference anchors
- Need to be from the same lexical universe as virgin texts
- Should contain lots of words
Suggestions for choosing reference values

- Must be “known” through some trusted external source
- For any pair of reference values, all scores are simply linear rescalings, so might as well use (-1, 1)
- The “middle point” will not be the midpoint, however, since this will depend on the relative word frequency of the reference documents
- Reference texts if scored as virgin texts will have document scores more extreme than other virgin texts
- With three or more reference values, the mid-point is mapped onto a multi-dimensional simplex. The values now matter but only in relative terms (we are still investigating this fully)
Multinomial Bayes model of Class given a Word
Class posterior probabilities

\[ P(c_k \mid w_j) = \frac{P(w_j \mid c_k)P(c_k)}{P(w_j \mid c_k)P(c_k) + P(w_j \mid c_{\neg k})P(c_{\neg k})} \]

- This represents the posterior probability of membership in class \( k \) for word \( j \)
- Under *certain conditions*, this is identical to what LBG (2003) called \( P_{wr} \)
- Under those conditions, the LBG “wordscore” is the linear difference between \( P(c_k \mid w_j) \) and \( P(c_{\neg k} \mid w_j) \)
The LBG approach required the identification not only of texts for each training class, but also “reference” scores attached to each training class.

Consider two “reference” scores $s_1$ and $s_2$ attached to two classes $k = 1$ and $k = 2$. Taking $P_1$ as the posterior $P(k = 1|w = j)$ and $P_2$ as $P(k = 2|w = j)$, a generalised score $s_j^*$ for the word $j$ is then

$$s_j^* = s_1 P_1 + s_2 P_2$$
$$= s_1 P_1 + s_2 (1 - P_1)$$
$$= s_1 P_1 + s_2 - s_2 P_1$$
$$= P_1 (s_1 - s_2) + s_2$$
"Certain conditions": More than two reference classes

For more than two reference classes, if the reference scores are ordered such that $s_1 < s_2 < \cdots < s_K$, then

$$s_j^* = s_1 P_1 + s_2 P_2 + \cdots + s_K P_K$$

$$= s_1 P_1 + s_2 P_2 + \cdots + s_K \left(1 - \sum_{k=1}^{K-1} P_k\right)$$

$$= \sum_{k=1}^{K-1} P_i (s_k - s_K) + s_l$$
A simpler formulation:
Use reference scores such that $s_1 = -1.0$, $s_K = 1.0$

- From above equations, it should be clear that any set of reference scores can be linearly rescaled to endpoints of $-1.0, 1.0$
- This simplifies the “simple word score”

$$s_j^* = (1 - 2P_1) + \sum_{k=2}^{K-1} P_k (s_k - 1)$$

- which simplifies with just two reference classes to:

$$s_j^* = 1 - 2P_1$$
Implications

- LBG’s “word scores” come from a linear combination of class posterior probabilities from a Bayesian model of class conditional on words.
- We might as well always anchor reference scores at $-1.0, 1.0$.
- There is a special role for reference classes in between $-1.0, 1.0$, as they balance between “pure” classes — more in a moment.
- There are alternative scaling models, such that used in Beauchamp’s (2012) “Bayesscore”, which is simply the difference in logged class posteriors at the word level. For $s_1 = -1.0, s_2 = 1.0$,

\[
s_j^B = -\log P_1 + \log P_2 = \log \frac{1 - P_1}{P_1}
\]
The “Naive” Bayes model of a joint document-level class posterior assumes conditional independence, to multiply the word likelihoods from a “test” document, to produce:

\[ P(c|d) = P(c) \frac{\prod_j P(w_j|c)}{P(w_j)} \]

So we could consider a document-level relative score, e.g. \(1 - 2P(c_1|d)\) (for a two-class problem)

But this turns out to be useless, since the predictions of class are highly separated
Moving to the document level

- A better solution is to score a test document as the arithmetic mean of the scores of its words.
- This is exactly the solution proposed by LBG (2003).
- Beauchamp (2012) proposes a “Bayesscore” which is the arithmetic mean of the log difference word scores in a document – which yields extremely similar results.

And now for some demonstrations with data...
Application 1: Dail speeches from LBG (2003)

(a) NB Speech scores by party, smooth=0, imbalanced priors

(b) Document scores from NB v. Classic Wordscores

- three reference classes (Opposition, Opposition, Government) at \{-1, -1, 1\}
- no smoothing
Application 1: Dail speeches from LBG (2003)

- two reference classes (Opposition+Opposition, Government) at \{-1, 1\}
- Laplace smoothing
Application 2: Classifying legal briefs (Evans et al 2007)

Wordscores v. Bayesscore

(a) Word level

(b) Document level

- Training set: Petitioner and Respondent litigant briefs from Grutter/Gratz v. Bollinger (a U.S. Supreme Court case)
- Test set: 98 amicus curiae briefs (whose P or R class is known)
Application 2: Classifying legal briefs (Evans et al 2007)
Posterior class prediction from NB versus log wordscores

![Graph showing posterior class prediction for Petitioner and Respondent classes based on log wordscores.](chart)

- Predicted Petitioner
- Predicted Respondent
Application 3: LBG’s British manifestos
More than two reference classes

- **x-axis**: Reference scores of \{5.35, 8.21, 17.21\} for Lab, LD, Conservatives
- **y-axis**: Reference scores of \{10.21, 5.26, 15.61\}
THE \( k \)-NN CLASSIFIER
A brief introduction to other classification methods: 
\textit{k}-nearest neighbour

- A non-parametric method for classifying objects based on the training examples that are \textit{closest} in the feature space.
- A type of instance-based learning, or “lazy learning” where the function is only approximated locally and all computation is deferred until classification.
- An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its \textit{k} nearest neighbors (where \textit{k} is a positive integer, usually small).
- Extremely \textit{simple}: the only parameter that adjusts is \textit{k} (number of neighbors to be used) - increasing \textit{k} \textit{smooths} the decision boundary.
$k$-NN Example: Red or Blue?
$k = 1$
$k = 7$
$k = 15$
Distance usually relates to all the attributes and assumes all of them have the same effects on distance.

Misclassification may result from attributes not confirming to this assumption (sometimes called the “curse of dimensionality”) – solution is to reduce the dimensions.

There are (many!) different metrics of distance.
SUPPORT VECTOR MACHINES
(Very) General overview to SVMs

- Generalization of maximal margin classifier
- The idea is to find the classification boundary that maximizes the distance to the marginal points

Unfortunately MMC does not apply to cases with non-linear decision boundaries
No solution to this using support vector classifier

FIGURE 9.8. Left: The observations fall into two classes, with a non-linear boundary between them. Right: The support vector classifier seeks a linear boundary, and consequently performs very poorly.

depends on the mean of all of the observations within each class, as well as the within-class covariance matrix computed using all of the observations. In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary. In fact we will see in Section 9.5 that the support vector classifier and logistic regression are closely related.

9.3 Support Vector Machines
We first discuss a general mechanism for converting a linear classifier into one that produces non-linear decision boundaries. We then introduce the support vector machine, which does this in an automatic way.

9.3.1 Classification with Non-linear Decision Boundaries
The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear. However, in practice we are sometimes faced with non-linear class boundaries. For instance, consider the data in the left-hand panel of Figure 9.8. It is clear that a support vector classifier will perform poorly here. Indeed, the support vector classifier shown in the right-hand panel of Figure 9.8 is useless here.

In Chapter 7, we are faced with an analogous situation. We see there that the performance of linear regression can suffer when there is a non-linear relationship between the predictors and the outcome. In that case, we consider enlarging the feature space using functions of the predictors,
One way to solve this problem

- Basic idea: If a problem is non-linear, don’t fit a linear model
- Instead, map the problem from the *input space* to a new (higher-dimensional) *feature space*
- Mapping is done through a non-linear transformation using suitably chosen basis functions
  - the “kernel trick”: using kernel functions to enable operations in the high-dimensional feature space without computing coordinates of that space, through computing inner products of all pairs of data in the feature space
  - different kernel choices will produce different results (polynomial, linear, radial basis, etc.)
- Makes it possible to then use a linear model in the feature space
SVMs represent the data in a *higher* dimensional projection using a kernel, and bisect this using a hyperplane.

Data is not linearly separable in the **input space**

Data is linearly separable in the **feature space** obtained by a kernel.
This is only needed when no linear separation plane exists - so not needed in second of these

• It is impossible to find a linear SVM classifier that separates tumors from normals!
• Need a non-linear SVM classifier, e.g. SVM with polynomial kernel of degree 2 solves this problem without errors.
• We should not apply a non-linear SVM classifier while we can perfectly solve this problem using a linear SVM classifier!
Different “kernels” can represent different decision boundaries

- This has to do with different projections of the data into higher-dimensional space
- The mathematics of this are complicated but solveable as forms of optimization problems - but the kernel choice is a user decision

![Diagram showing decision boundaries for SVM with different kernels.](image-url)