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 exmplify constrasting 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.

Type	**Test Design**	**Causes of Disagreements**	Strength
Stability	test-retest	intraobserver inconsistencies	weakest
Reproducibility	test-test	intraobserver inconsistencies + interobserver disagreements	medium
Accuracy	test-standard	intraobserver inconsistencies + interobserver disagreements + deviations from a standard	strongest

Measures of agreement

- Percent agreement Very simple: (number of agreeing ratings) / (total ratings) * 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 occured by chance*
  - Cohen's κ is most common
  - Krippendorf's α is a generalization of Cohen's κ
  - Both range from [0,1]

Reliability data matrixes

Example here used binary data (from Krippendorff)

Article:	1	2	3	4	5	6	7	8	9	10
Coder A	1	1	0	0	0	0	0	0	0	0
Coder B	0	1	1	0	0	1	0	1	0	0

- A and B agree on 60% of the articles: 60% agreement
- Correlation is (approximately) 0.10
- Observed *dis*agreement: 4
- Expected *dis*agreement (by chance): 4.4211
- Krippendorff's $\alpha = 1 - \frac{D_o}{D_e} = 1 - \frac{4}{4.4211} = 0.095$
- Cohen's κ (nearly) identical

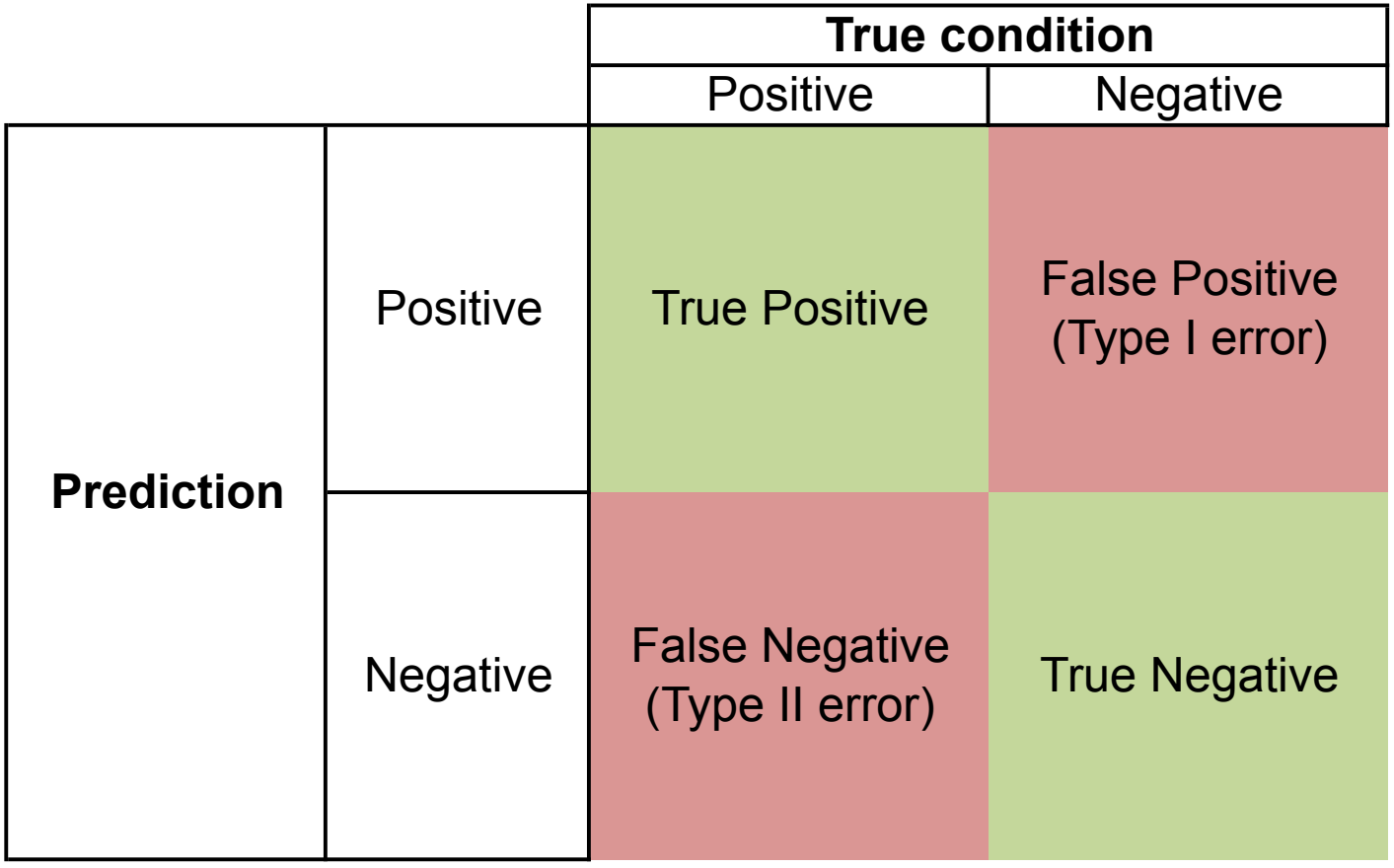
EVALUATING CLASSIFIER PERFORMANCE

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 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



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

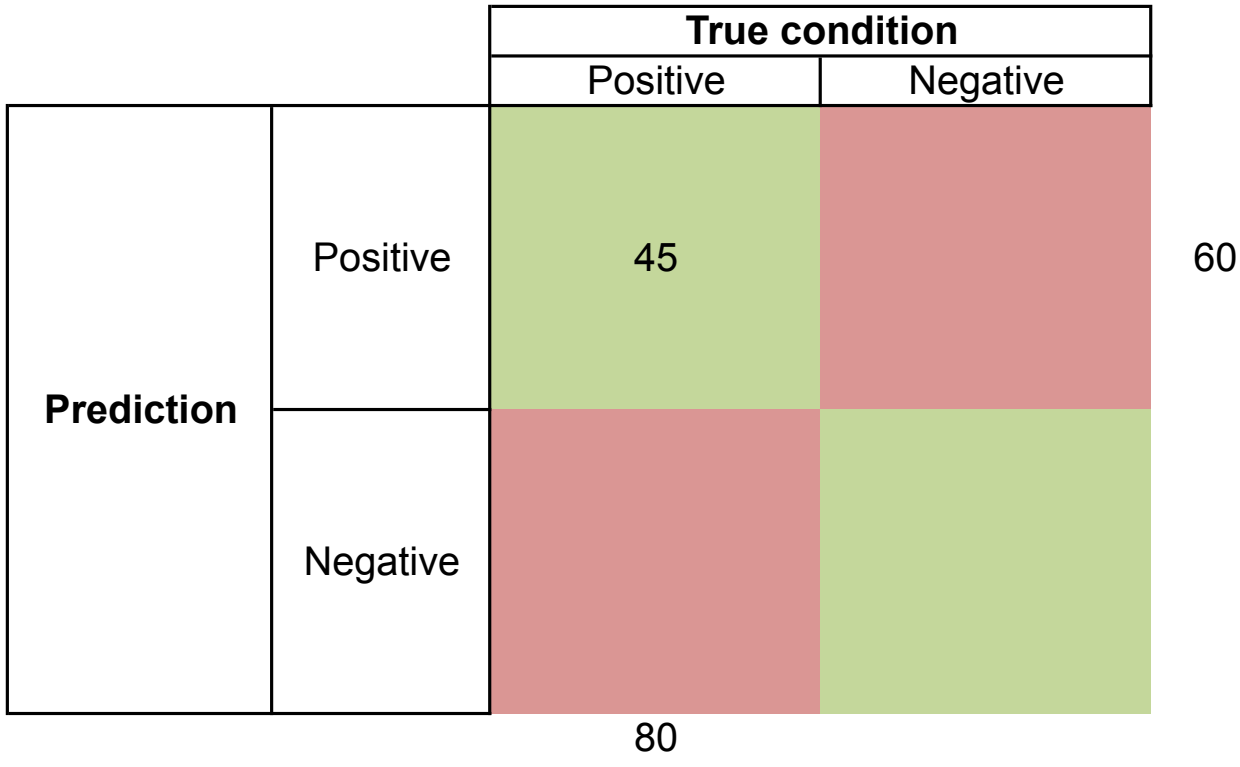
Assume:

- 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

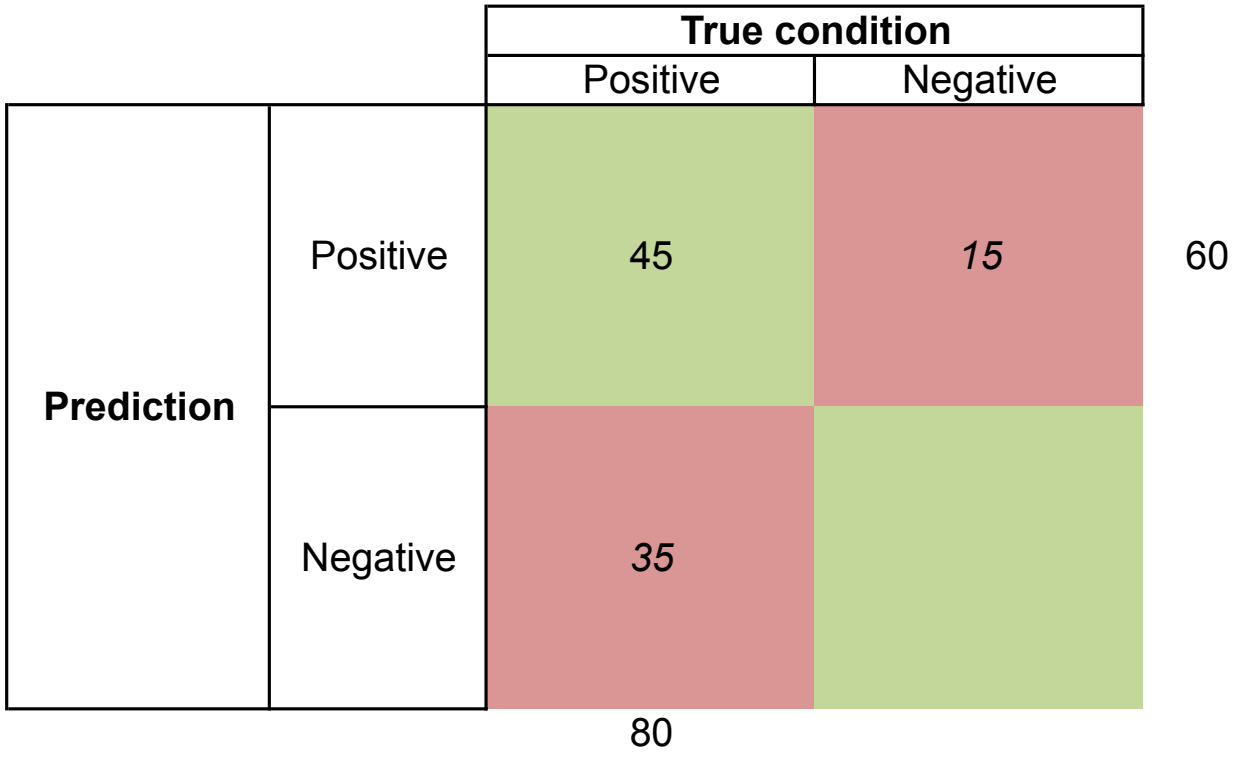
Solution:

$$\text{Precision} = (45/(45+15)) = 45/60 = 0.75$$
$$\text{Recall} = (45/(45+35)) = 45/80 = 0.56$$

Accuracy?


True condition
Positive
Negative
Prediction
Positive
Negative
45
60
80

add in the cells we can compute



but need True Negatives and N to compute accuracy

		True condition		
		Positive	Negative	
Prediction	Positive	45	*15*	60
	Negative	*35*	**???**	
		80		

assume 10 True Negatives:

		True condition		
		Positive	Negative	
Prediction	Positive	45	*15*	60
	Negative	*35*	***10***	***45***
		80	***25***	***105***

$$\text{Accuracy} = (45 + 10)/105 = 0.52$$

$$\text{F1} = 2 * (0.75 * 0.56)/(0.75 + 0.56) = 0.64$$

now assume 100 True Negatives:

		True condition		
		Positive	Negative	
Prediction	Positive	45	*15*	60
	Negative	*35*	***100***	***135***
		80	***115***	***195***

$$\text{Accuracy} = (45 + 100)/195 = 0.74$$

$$\text{F1} = 2 * (0.75 * 0.56)/(0.75 + 0.56) = 0.64$$

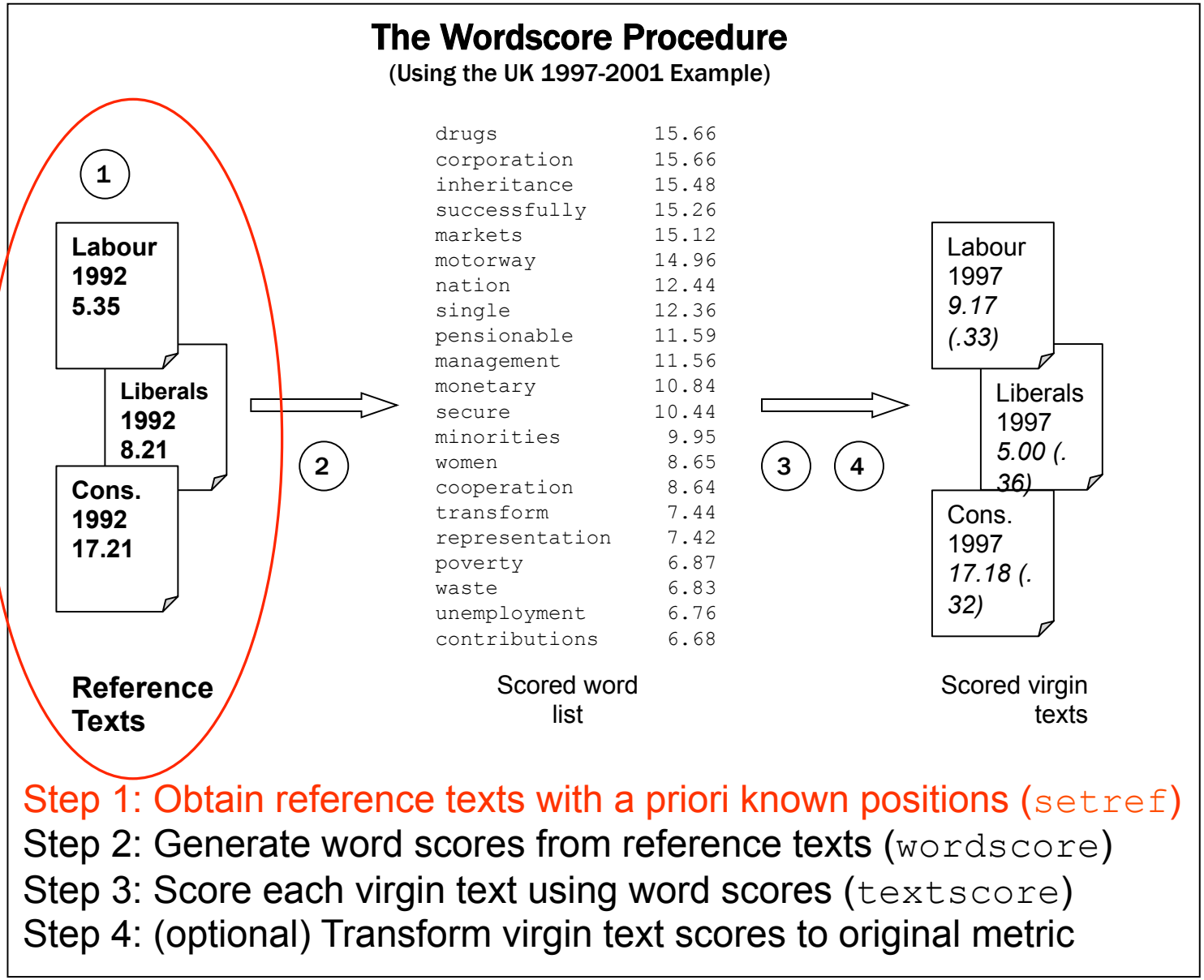
From Classification to Scaling

- 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 wed 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

Wordscores Procedure



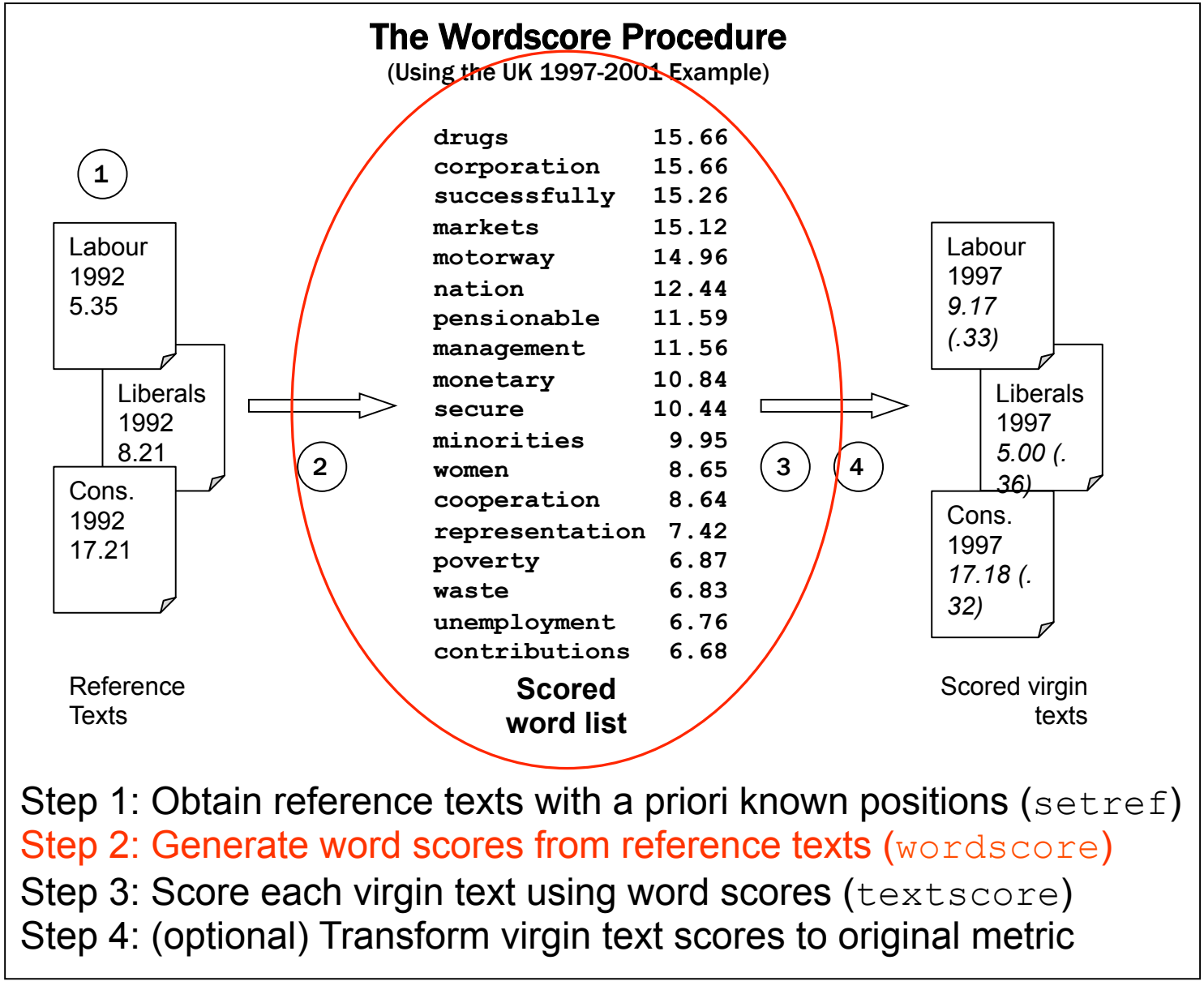
Step 1: Obtain reference texts with a priori known positions (`setref`)

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 Procedure



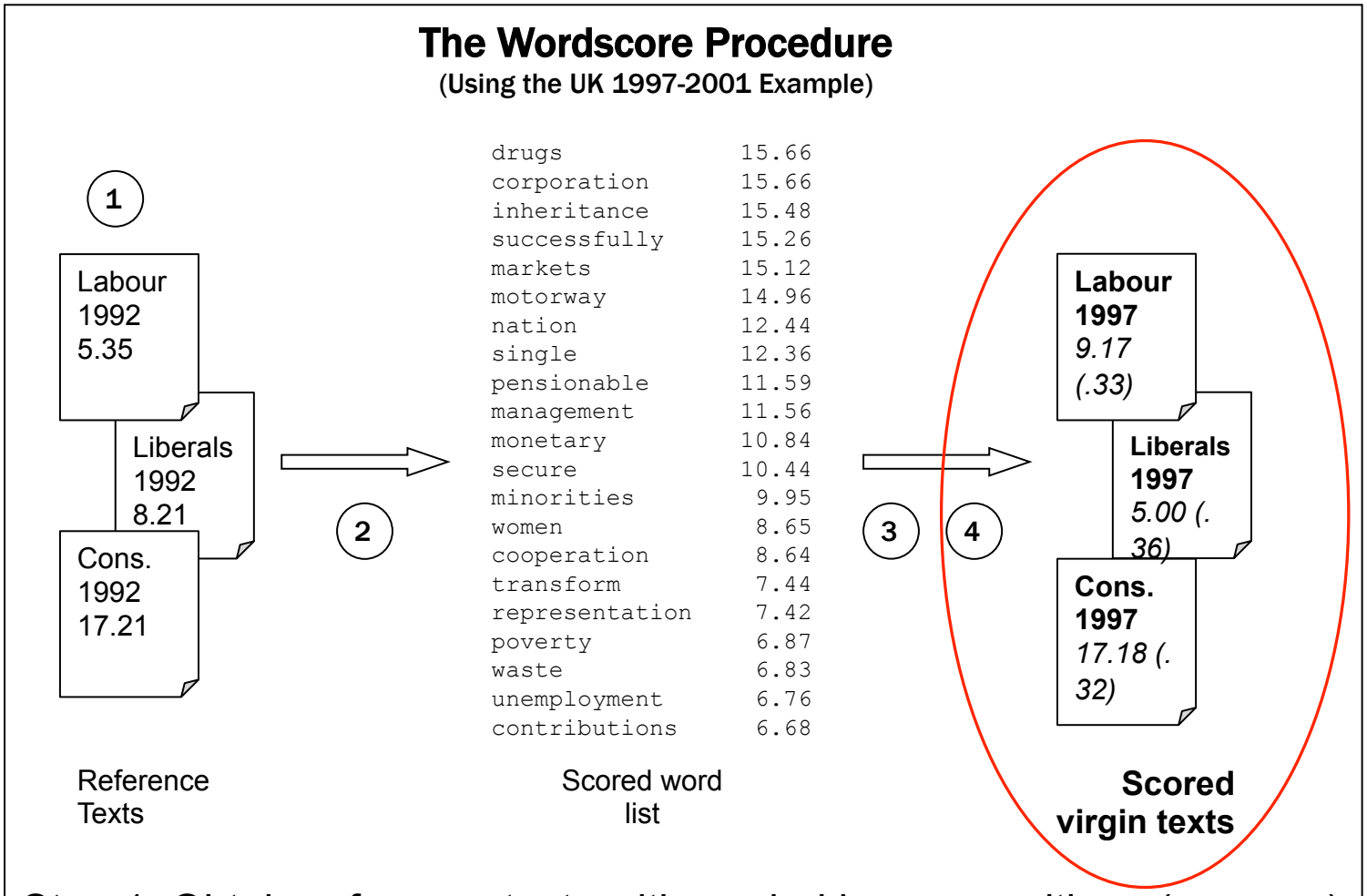
Step 1: Obtain reference texts with a priori known positions (`setref`)

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

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

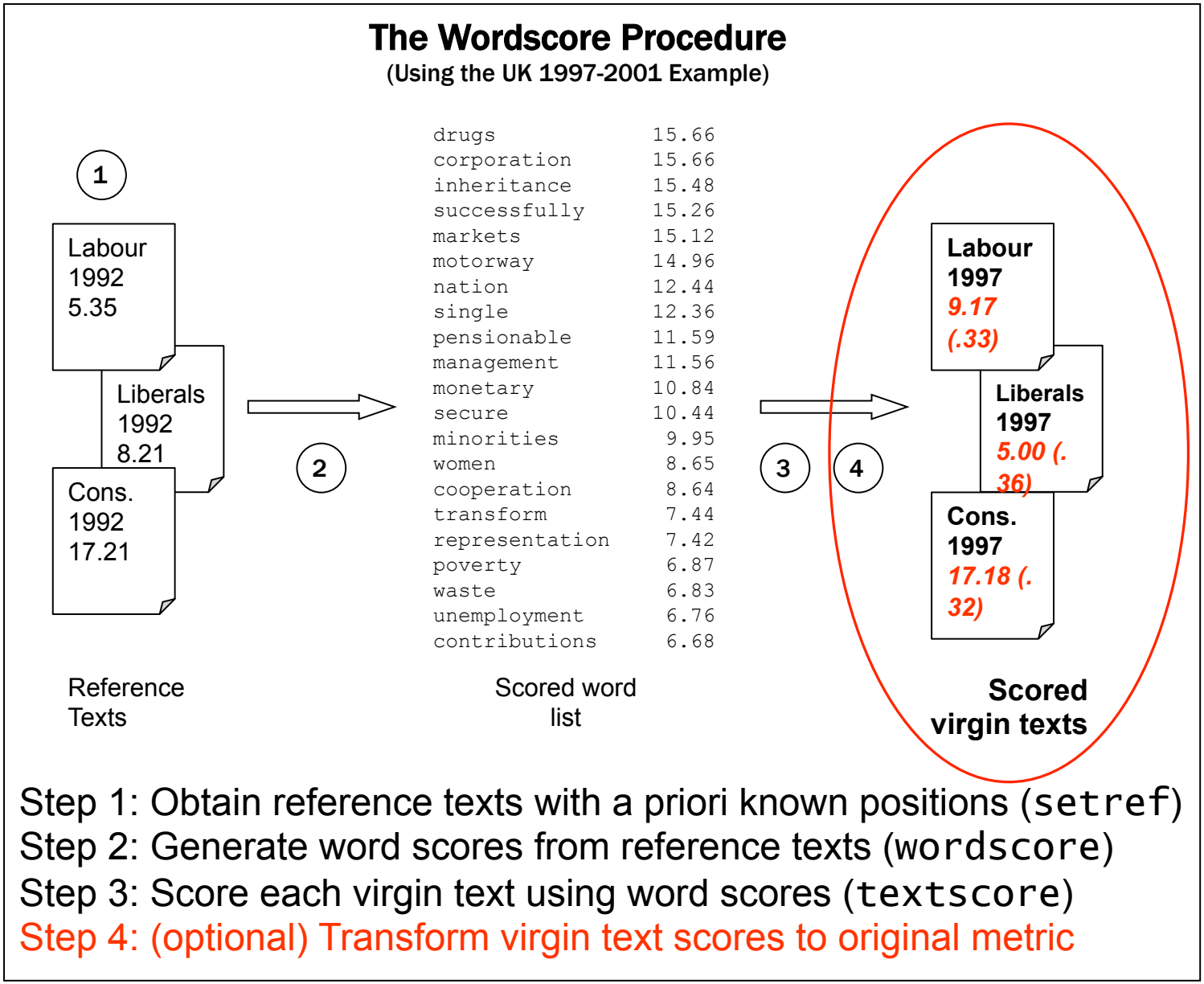
Step 4: (optional) Transform virgin text scores to original metric

Wordscores Procedure



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 Procedure



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:

$$F_{ij} = \frac{C_{ij}}{C_{i\cdot}} \tag{1}$$

where $C_{i\cdot} = \sum_{j=1}^{J} C_{ij}$

Wordscores mathematically: Word scores

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

$$P_{ij} = \frac{F_{ij}}{\sum_{i=1}^{I} F_{ij}} \tag{2}$$

- 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 $F_{i\ \text{"choice"}} = \{.010, .030\}$
- 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

$$P_{A\ \text{"choice"}} = \frac{.010}{(.010 + .030)} = 0.25 \tag{3}$$

$$P_{B\ \text{"choice"}} = \frac{.030}{(.010 + .030)} = 0.75 \tag{4}$$

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}:

$$S_j = \sum_{i=1}^{I} a_i P_{ij} \tag{5}$$

- In matrix algebra, $\underset{1\times J}{S} = \underset{1\times I}{a} \cdot \underset{I\times J}{P}$
- 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

Wordscores mathematically: Word scores

- 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:

$$v_k = \sum_j (F_{kj} \cdot s_j) \tag{6}$$

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)

Multinomial Bayes model of Class given a Word

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

$$= \frac{P(w_j|c_k)P(c_k)}{P(w_j|c_k)P(c_k) + P(w_j|c_{\neg k})P(c_{\neg k})} \tag{7}$$

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 oberved 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

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

- 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 $P(c_{\neg k}|w_j)$

"Certain conditions"

- 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

$$\begin{aligned} s_j^* &= s_1 P_1 + s_2 P_2 \\ &= s_1 P_1 + s_2(1 - P_1) \\ &= s_1 P_1 + s_2 - s_2 P_1) \\ &= P_1(s_1 - s_2) + s_2 \end{aligned}$$

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	?

Naive Bayes Classification Example

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 $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\mathsf{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\mathsf{Tokyo}|c) = \hat{P}(\mathsf{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\mathsf{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\mathsf{Tokyo}|\overline{c}) = \hat{P}(\mathsf{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

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

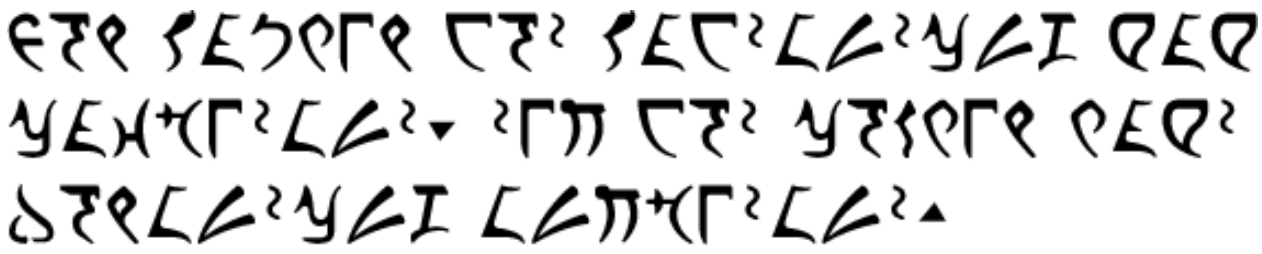
We then get:

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Thus, the classifier assigns the test document to $c = \mathit{China}$. 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

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

- 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 $P(c_{\neg k}|w_j)$

"Certain conditions"

- 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

$$\begin{aligned} s_j^* &= s_1 P_1 + s_2 P_2 \\ &= s_1 P_1 + s_2(1 - P_1) \\ &= s_1 P_1 + s_2 - s_2 P_1) \\ &= P_1(s_1 - s_2) + s_2 \end{aligned}$$

"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

$$
\begin{aligned}
s_j^* &= s_1 P_1 + s_2 P_2 + \cdots + s_K P_K \\
&= s_1 P_1 + s_2 P_2 + \cdots + s_K (1 - \sum_{k=1}^{K-1} P_k) \\
&= \sum_{k=1}^{K-1} P_i (s_k - s_K) + s_I
\end{aligned}
$$

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 $s_1 = -1.0$, $s_2 = 1.0$,

$$\begin{aligned} s_j^B &= -\log P_1 + \log P_2 \\ &= \log \frac{1 - P_1}{P_1} \end{aligned}$$

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:

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

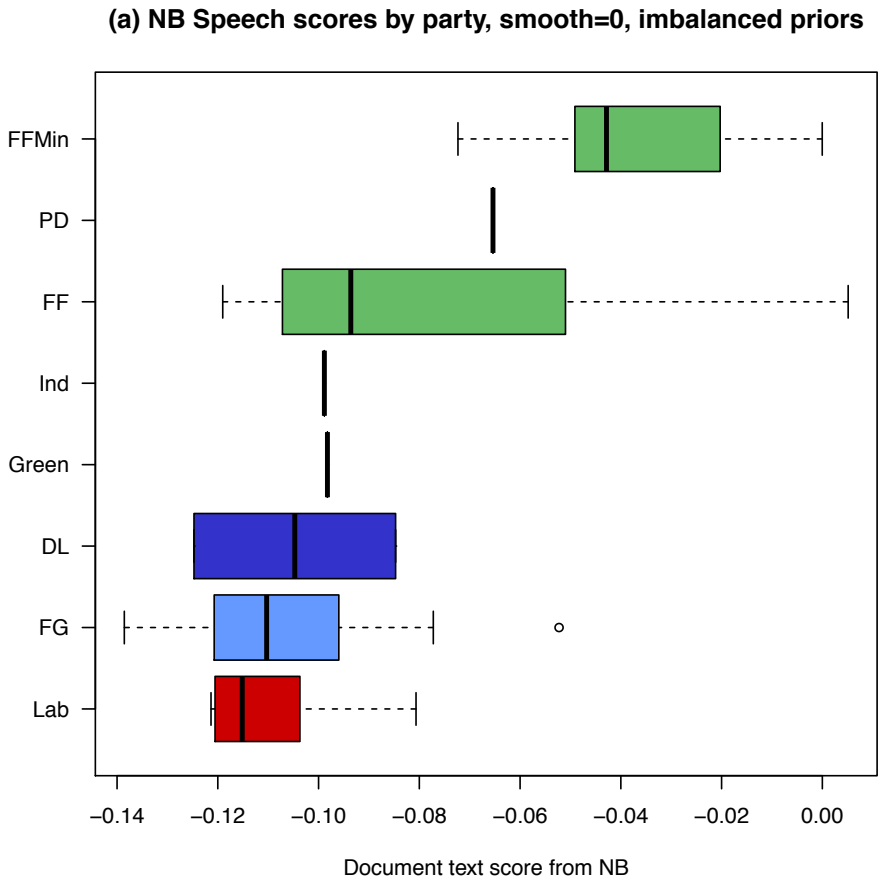
- So we *could* consider a document-level relative score, e.g. $1 - 2P(c_1|d)$ (for a two-class problem)
- But this turns out to be *useless*, since the predictions of class are highly separated

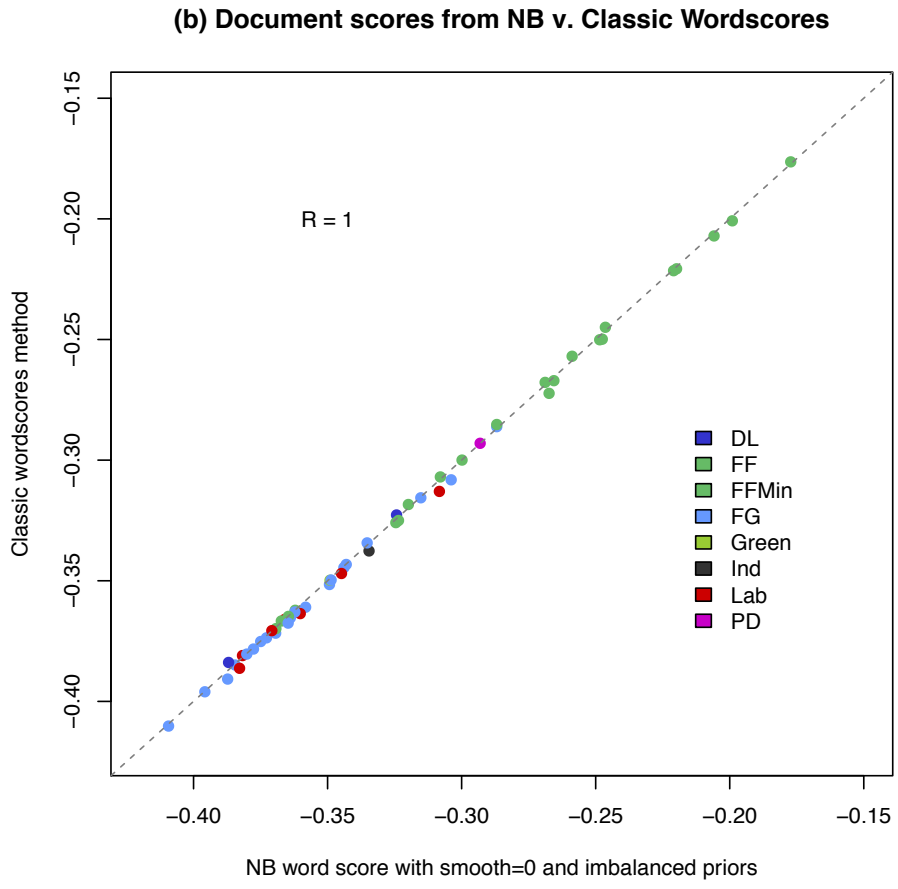
Moving to the document level

- 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...

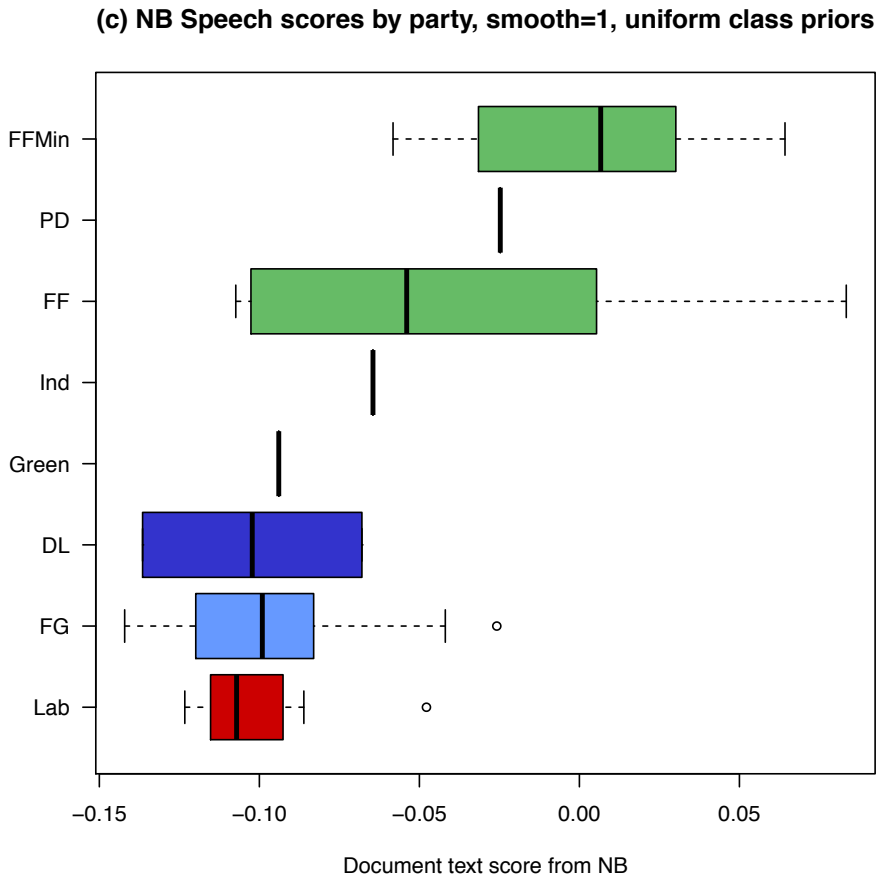
Application 1: Dail speeches from LBG (2003)

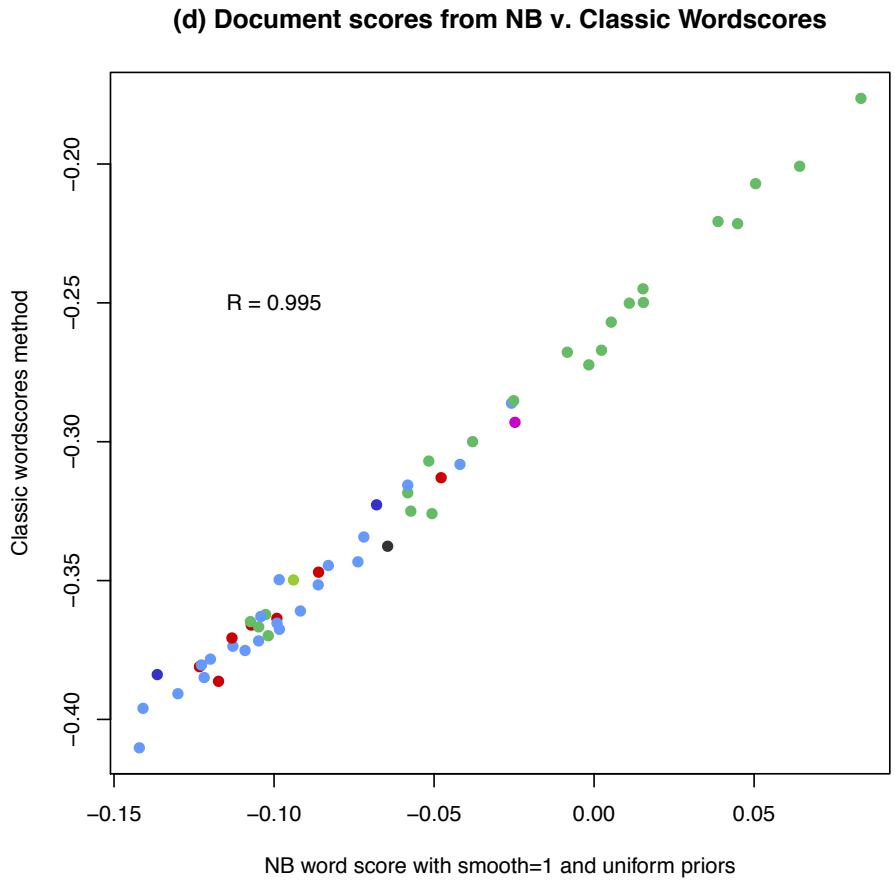




- three reference classes (Opposition, Opposition, Government) at {-1, -1, 1}
- no smoothing

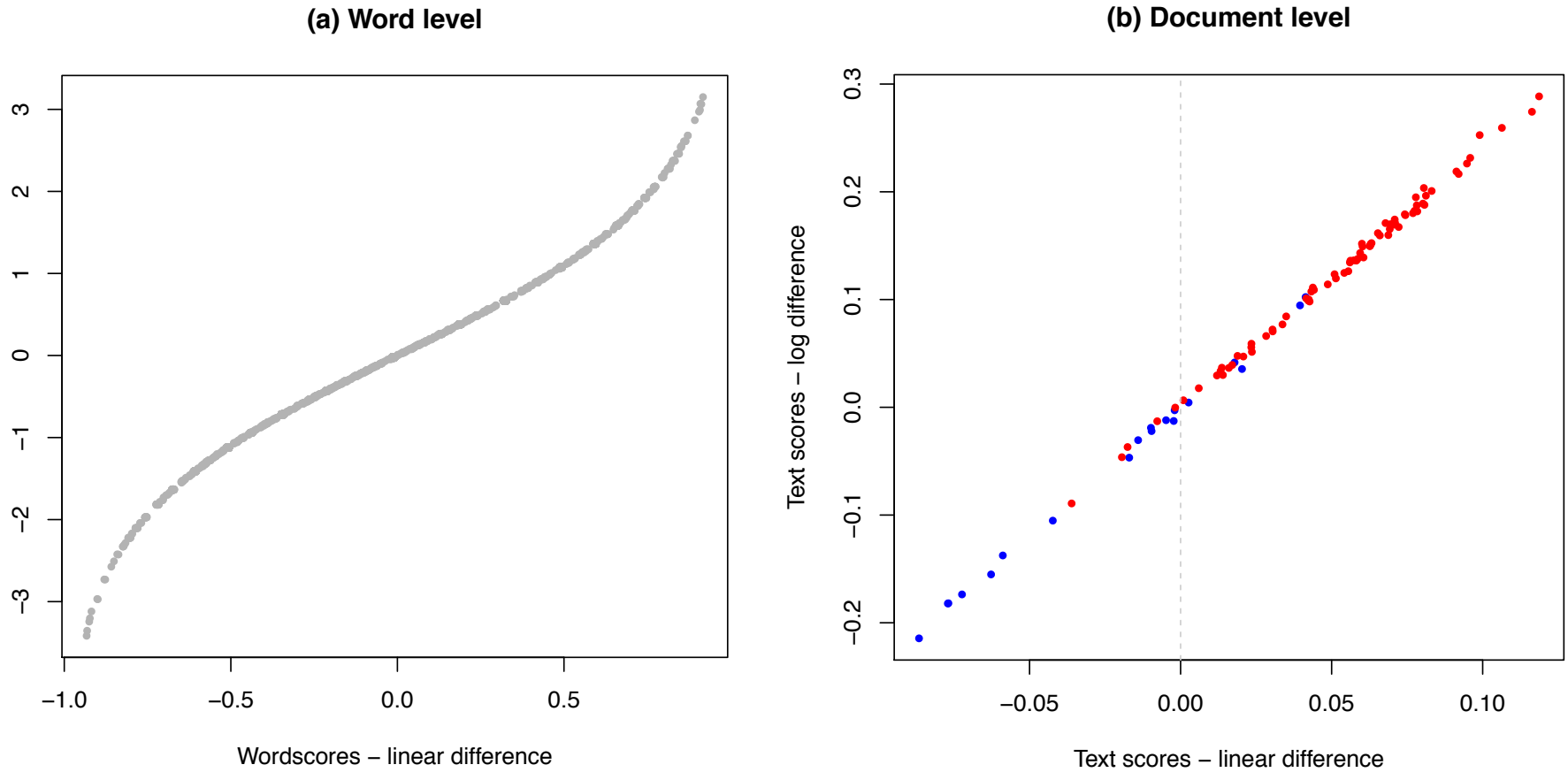
Application 1: Dail speeches from LBG (2003)





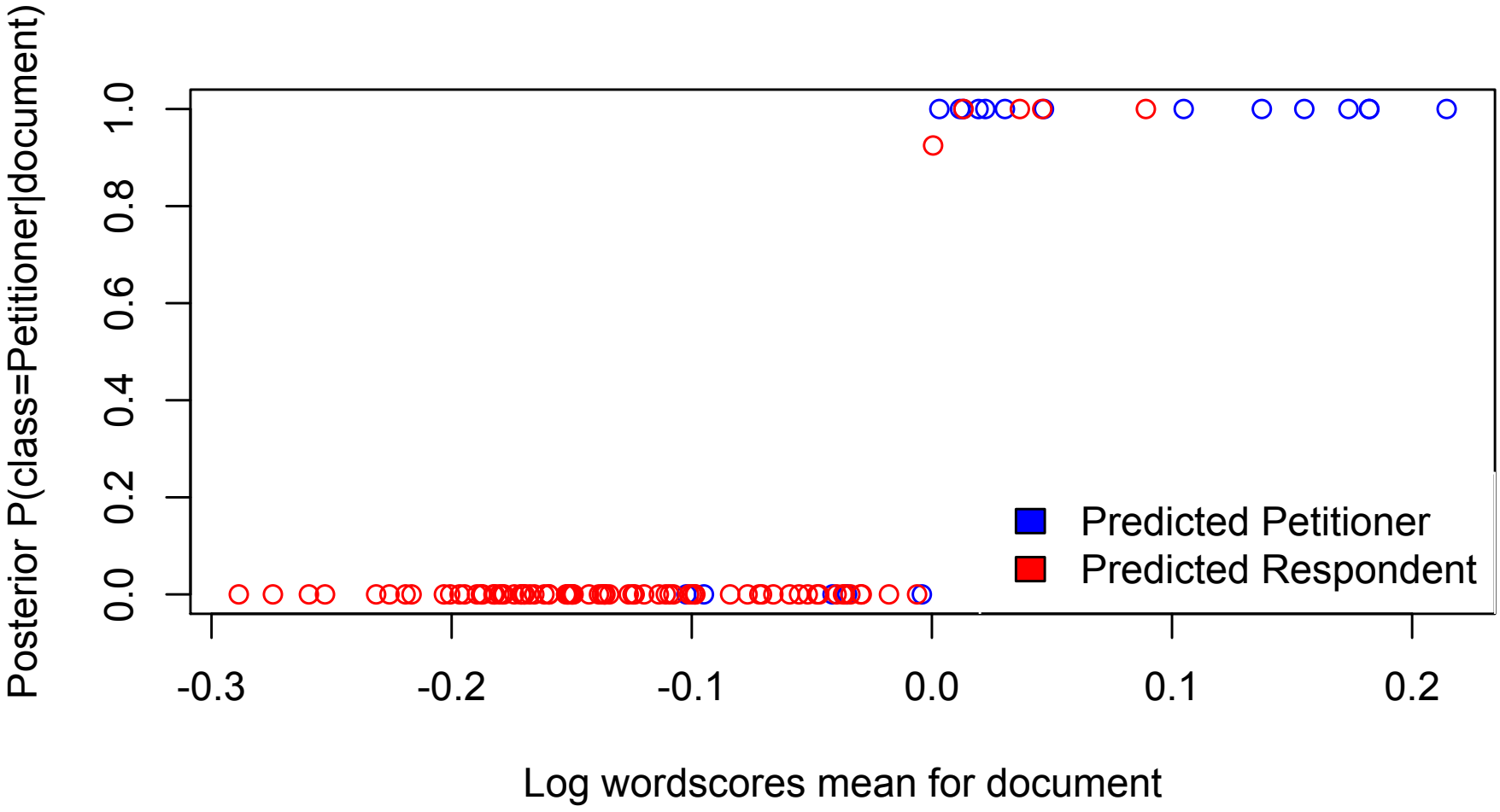
- two reference classes (Opposition+Opposition, Government) at {-1, 1}
- Laplace smoothing

Application 2: Classifying legal briefs (Evans et al 2007) Wordscores v. Bayesscore

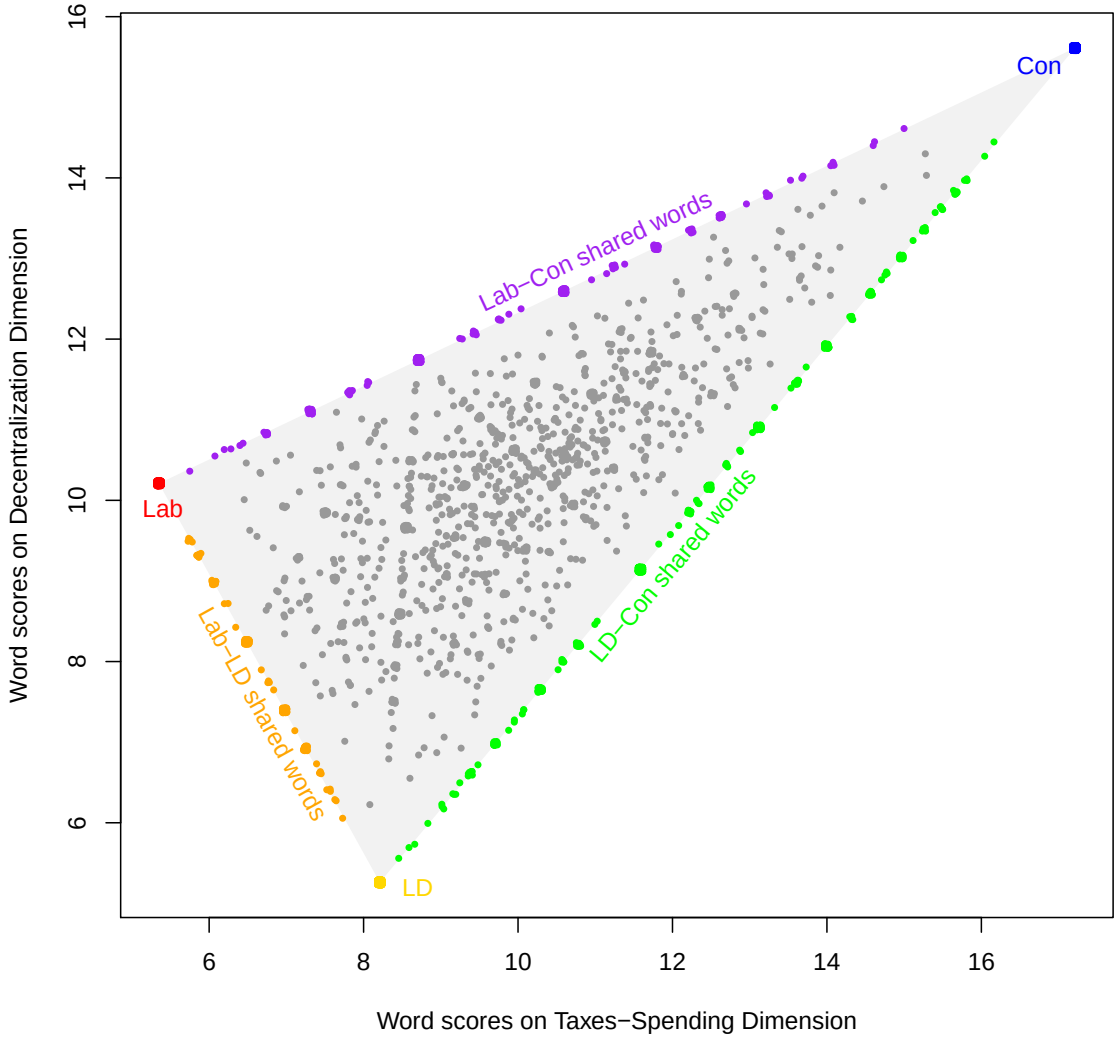


- Training set: Petitioner and Respondent litigant briefs from *Grutter/Gratz v. Bollinger* (a U.S. Supreme Court case)
- Test set: 98 amicus curiae briefs (whose P or R class is known)

Application 2: Classifying legal briefs (Evans et al 2007)
Posterior class prediction from NB versus log wordscores



Application 3: LBG's British manifestos
More than two reference classes



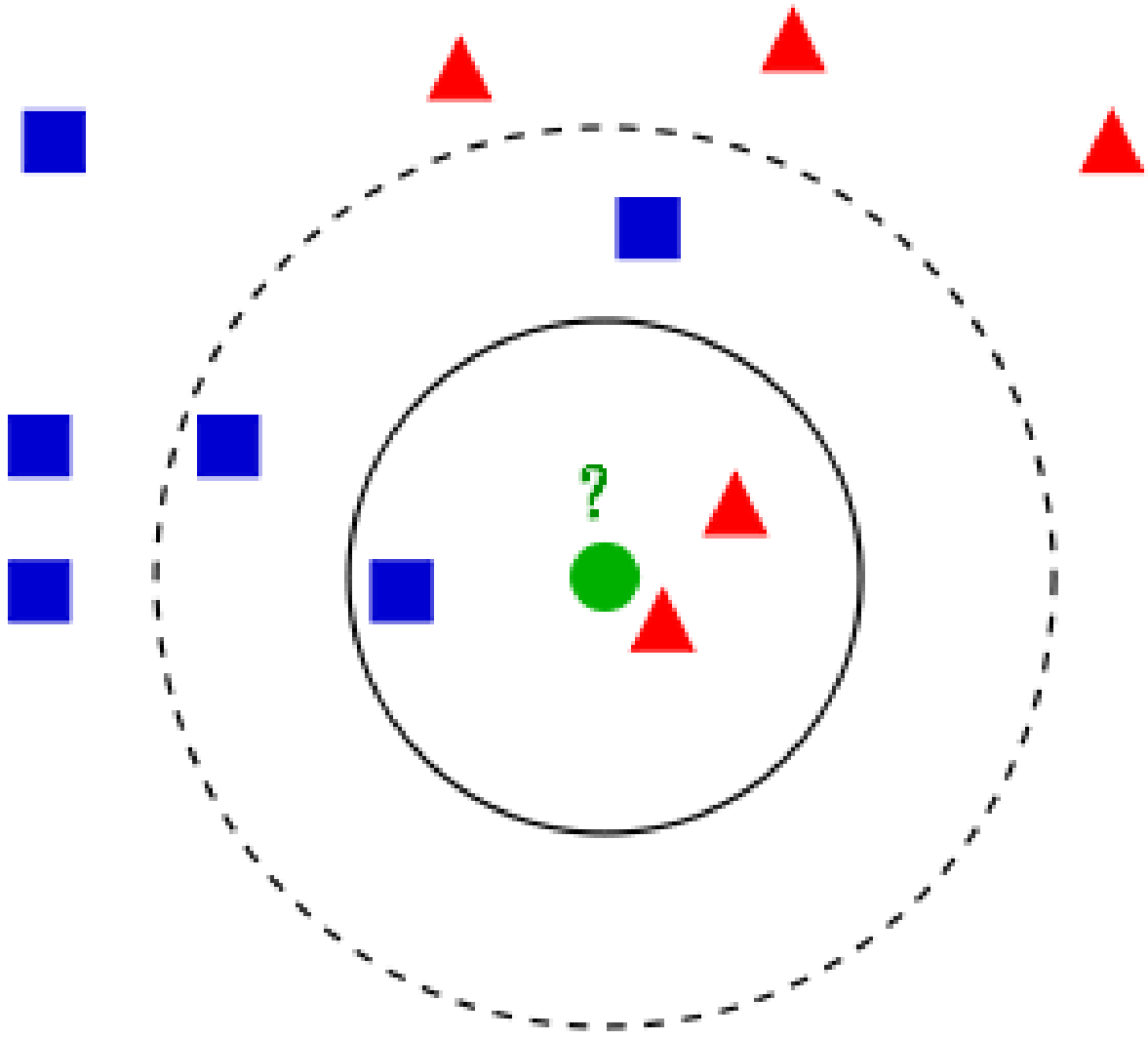
- x-axis: Reference scores of $\{5.35, 8.21, 17.21\}$ for Lab, LD, Conservatives
- y-axis: Reference scores of $\{10.21, 5.26, 15.61\}$

THE k-NN CLASSIFIER

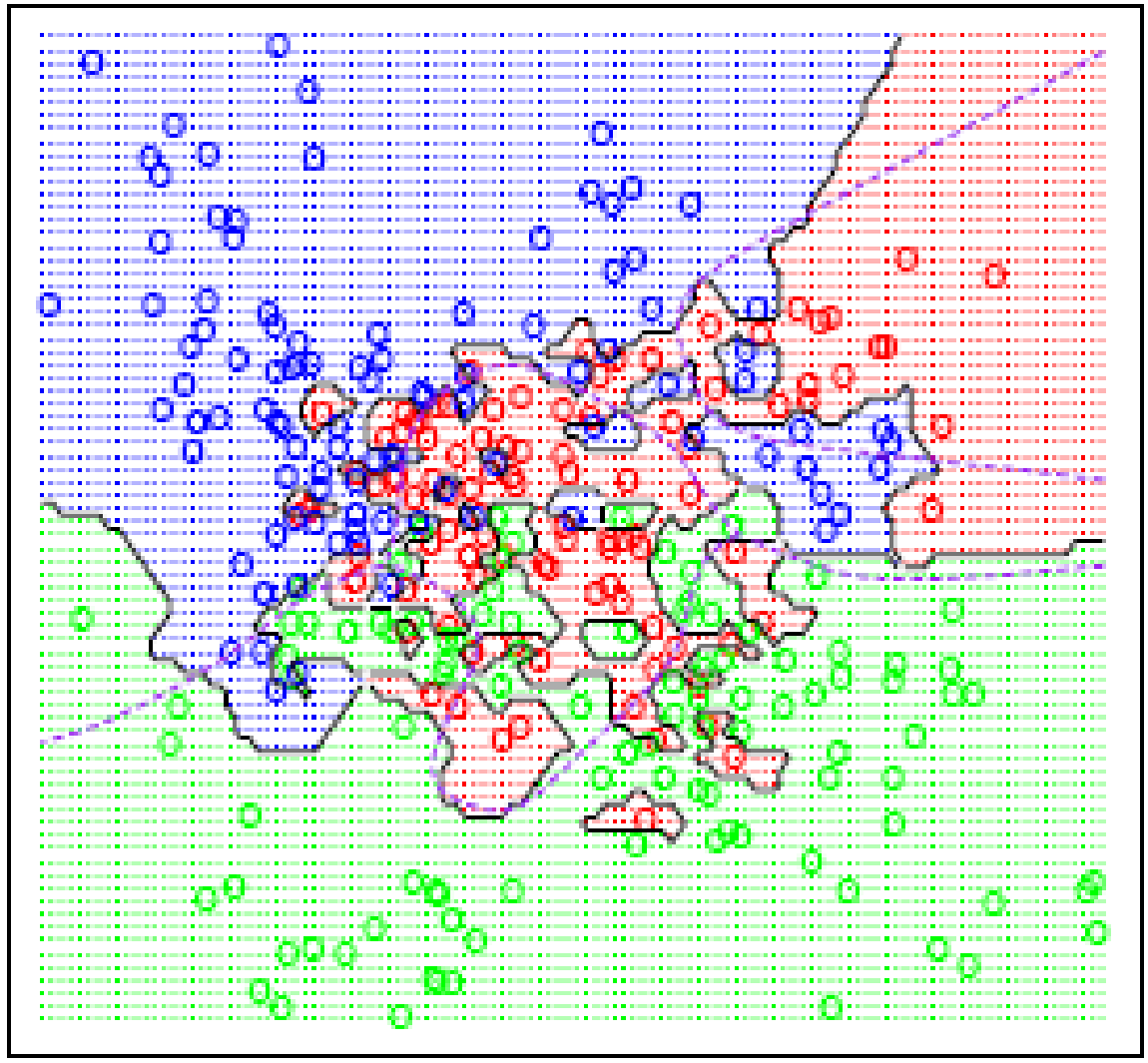
A brief introduction to other classification methods: k-nearest neighbour

- A non-parametric method for classifying objects based on the training examples taht are *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 k nearest neighbors (where k is a positive integer, usually small)
- Extremely *simple*: the only parameter that adjusts is k (number of neighbors to be used) - increasing k *smooths* the decision boundary

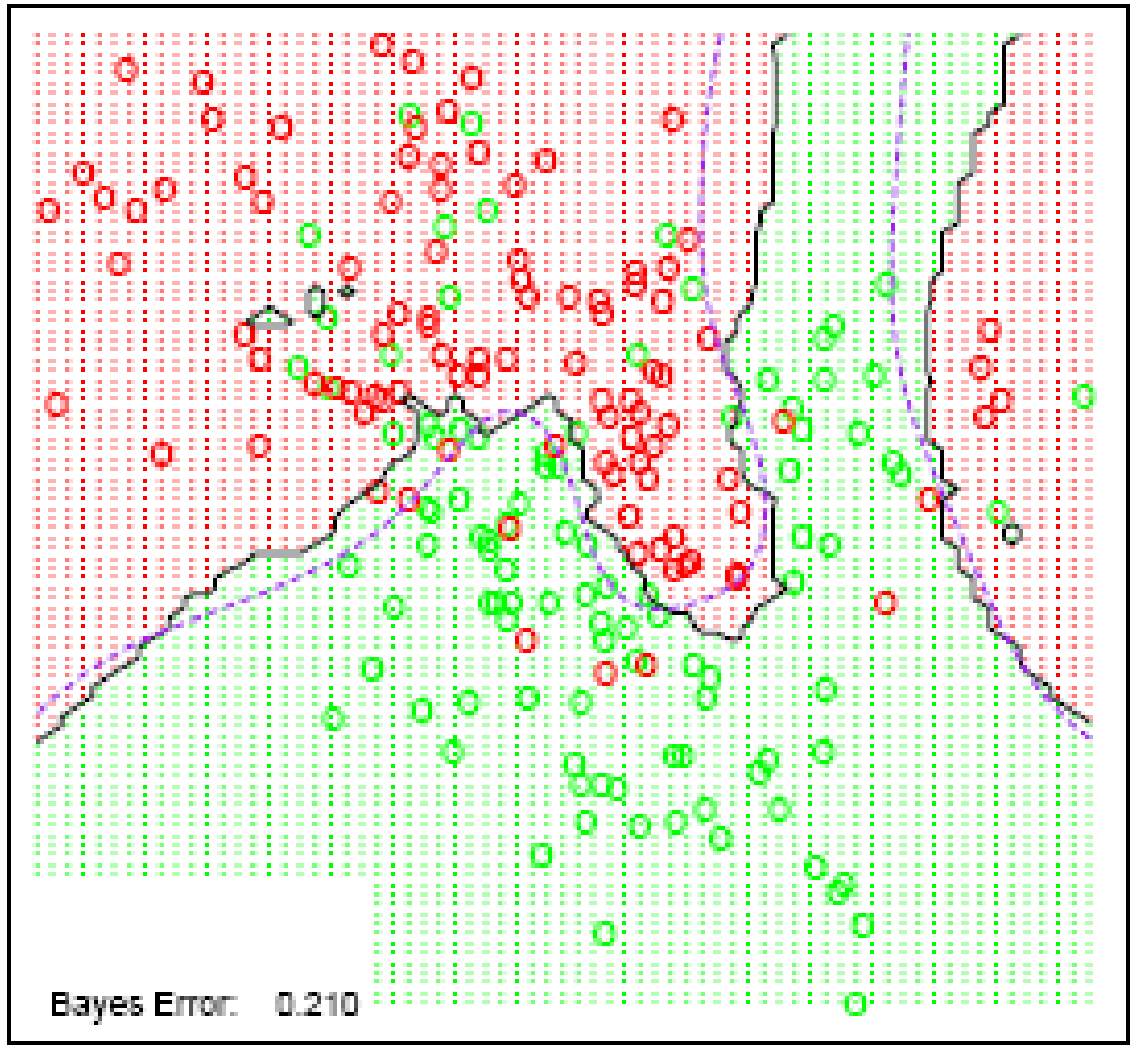
k-NN Example: Red or Blue?



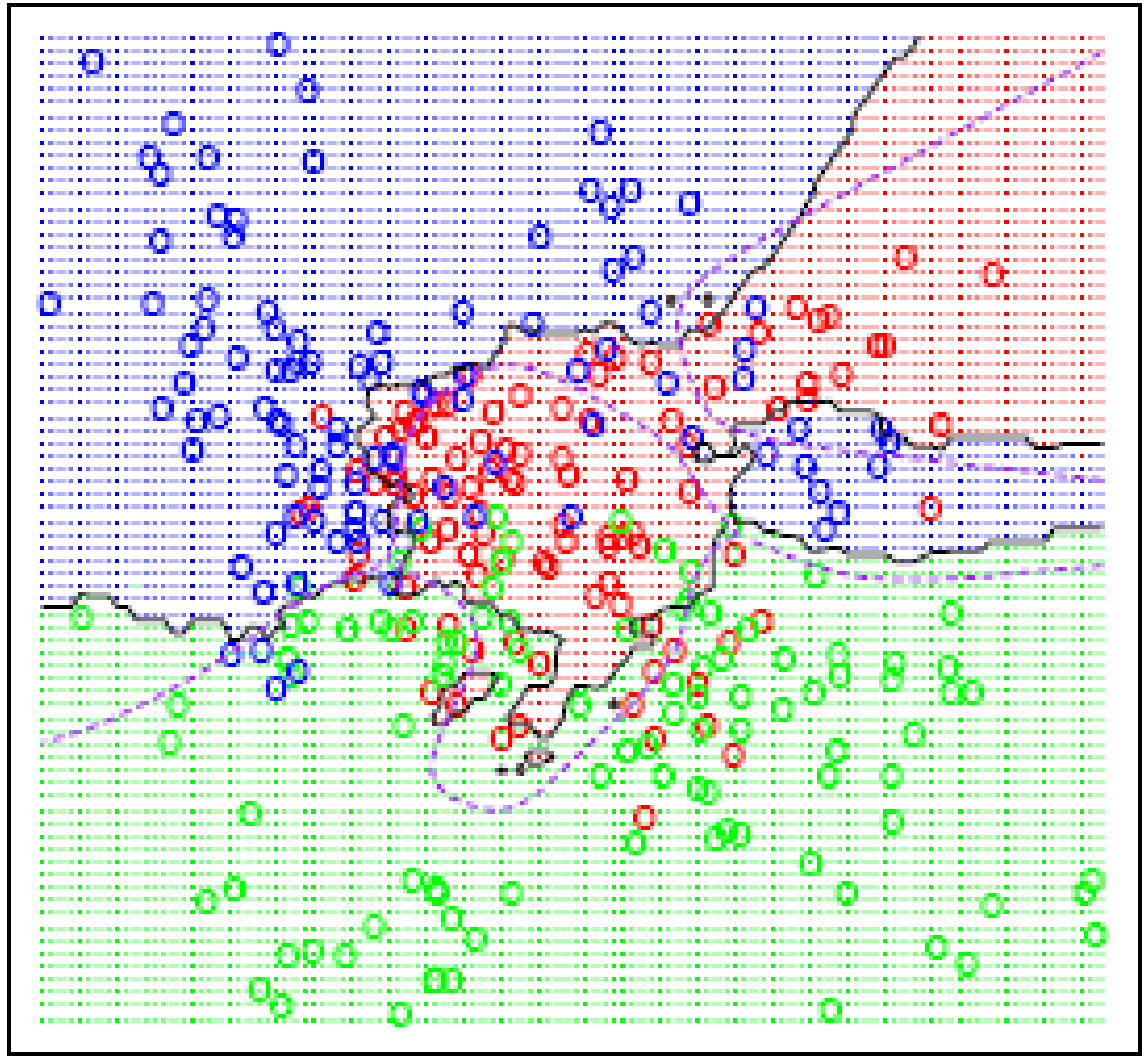
$k = 1$



$k = 7$



$k = 15$



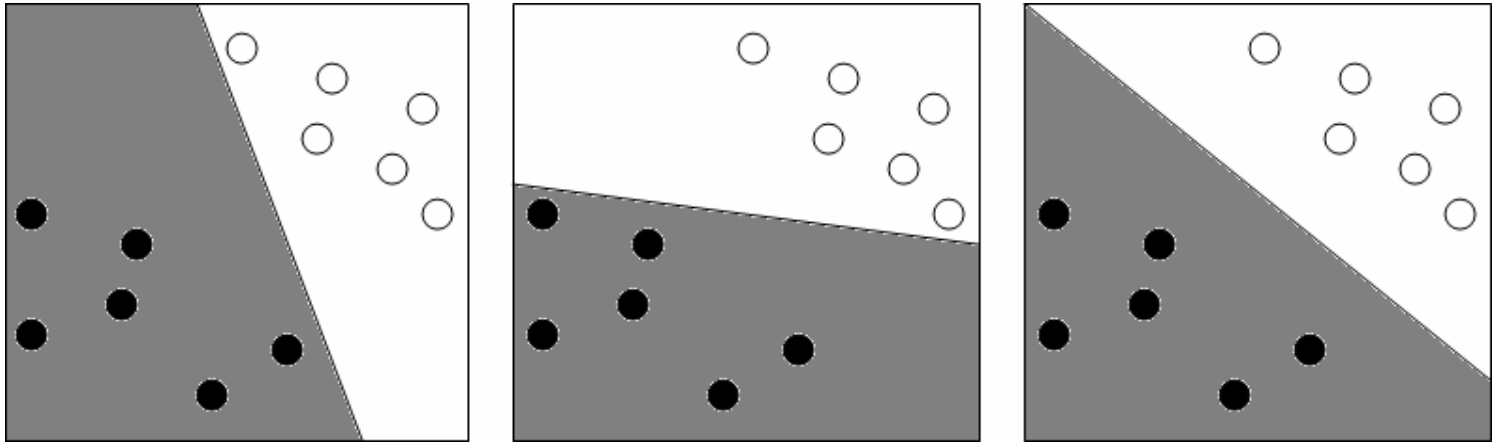
k-nearest neighbour issues: Dimensionality

- Distance usually relates to all the attributes and assumes all of them have the same effects on distance
- Misclassification may results 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

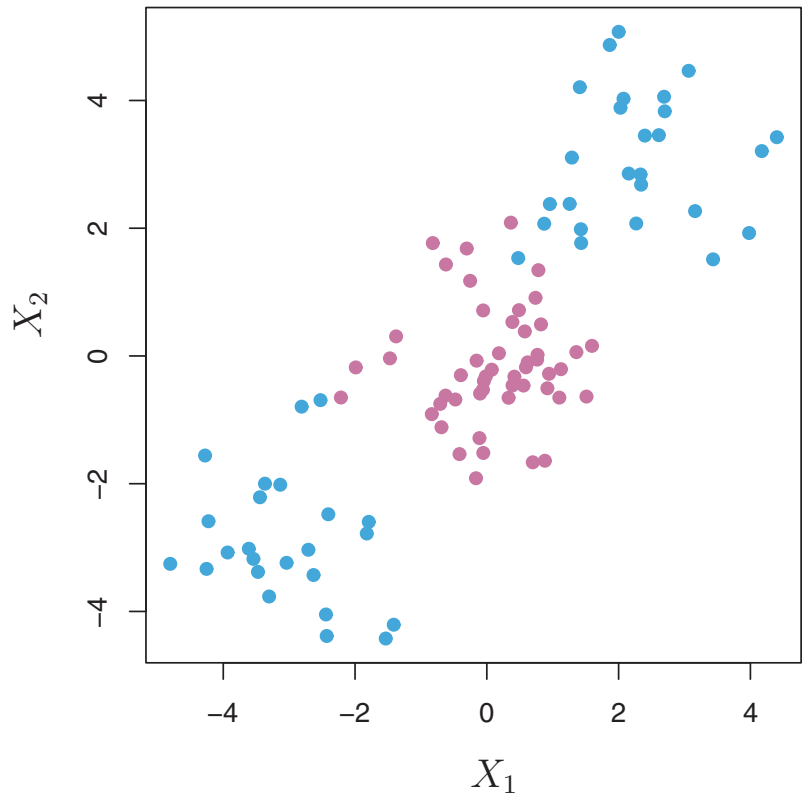
(Very) General overview to SVMs

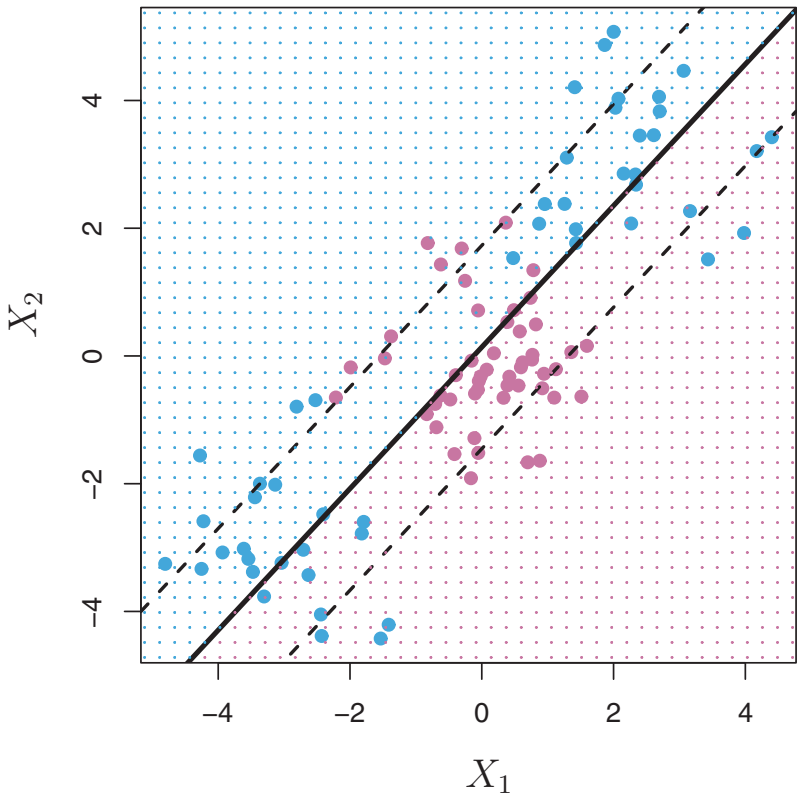
- Generalization of maximal margin classifier
- The idea is to find the classification boundary that maximizes the distance to the marginal points



- Unfortunately MMC does not apply to cases with non-linear decision boundaries

No solution to this using support vector classifier

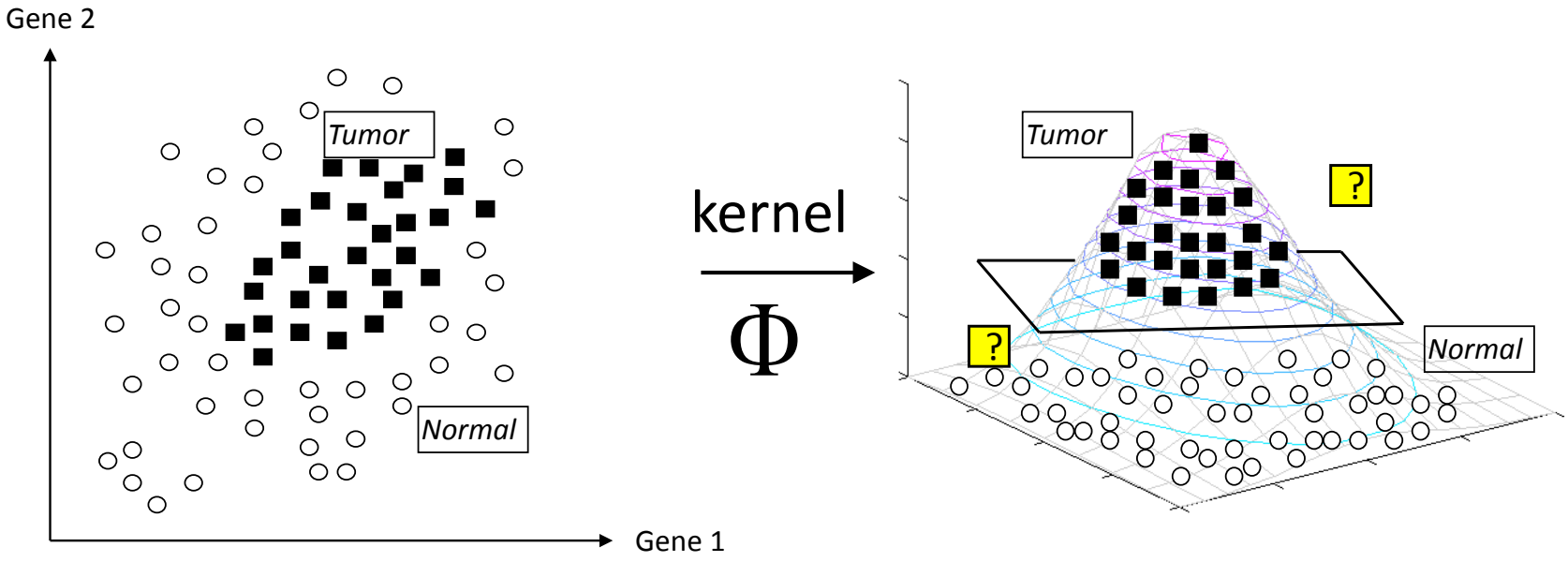




One way to solve this problem

- 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

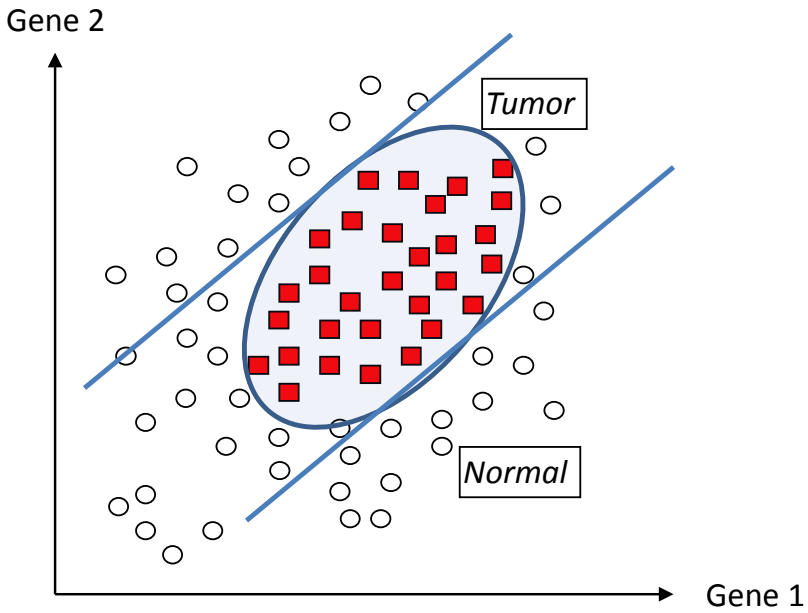
SVMs represent the data in a *higher* dimensional projection using a kernel, and bisect this using a hyperplane

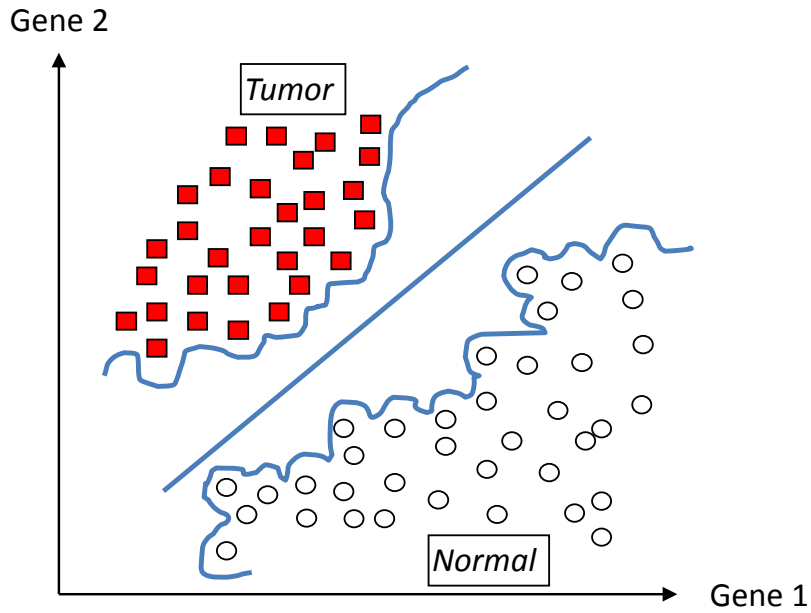


Data is not linearly separable in the input space

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





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