# Text as Data

Justin Grimmer

Associate Professor
Department of Political Science
Stanford University

November 6th, 2014

# Supervised Learning

1) Task
   - Classify documents to pre existing categories
   - Measure the proportion of documents in each category
2) Objective function
   - Suppose we have $K$ categories.
   - Select $N_{\text{train}}$ document to hand-label, $Y_i = k$, $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$

$$\boldsymbol{Y} = f(\boldsymbol{X}, \boldsymbol{\theta})$$

3) Optimization
   - Method specific: MLE, Bayesian, EM, ...
   - We learn $\widehat{\boldsymbol{\theta}}$
4) Validation
   - Obtain predicted fit for new data $f(\boldsymbol{X}_i, \widehat{\boldsymbol{\theta}})$
   - Examine prediction performance $\rightsquigarrow$ compare classification to gold standard

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:

- Models for categorizing texts

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:

- Models for categorizing texts
  - Know (develop) categories before hand

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:

- Models for categorizing texts
  - Know (develop) categories before hand
  - Hand coding: assign documents to categories
  - Infer: new document assignment to categories (distribution of documents to categories)

# Supervised Learning

Clustering and Topic Models:

- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:

- Models for categorizing texts
  - Know (develop) categories before hand
  - Hand coding: assign documents to categories
  - Infer: new document assignment to categories (distribution of documents to categories)
  - Pre-estimation: extensive work constructing categories, building classifiers
  - Post-estimation: relatively little work

# Supervised Learning

Today:

# Supervised Learning

Today:

- How to generate valid hand coding categories

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
  - Super learning: optimal ensemble methods

# Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
  - Super learning: optimal ensemble methods

Methods generalize beyond text

# Components to Supervised Learning Method

# Components to Supervised Learning Method

1) Set of categories

# Components to Supervised Learning Method

1) Set of categories
    - Credit Claiming, Position Taking, Advertising
    - Positive Tone, Negative Tone
    - Pro-war, Ambiguous, Anti-war

# Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war
2) Set of hand-coded documents

# Components to Supervised Learning Method

1) Set of categories
    - Credit Claiming, Position Taking, Advertising
    - Positive Tone, Negative Tone
    - Pro-war, Ambiguous, Anti-war
2) Set of hand-coded documents
    - Coding done by human coders
    - Training Set: documents we'll use to learn how to code
    - Validation Set: documents we'll use to learn how well we code

# Components to Supervised Learning Method

1) Set of categories
    - Credit Claiming, Position Taking, Advertising
    - Positive Tone, Negative Tone
    - Pro-war, Ambiguous, Anti-war
2) Set of hand-coded documents
    - Coding done by human coders
    - Training Set: documents we'll use to learn how to code
    - Validation Set: documents we'll use to learn how well we code
3) Set of unlabeled documents

# Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war
2) Set of hand-coded documents
   - Coding done by human coders
   - Training Set: documents we'll use to learn how to code
   - Validation Set: documents we'll use to learn how well we code
3) Set of unlabeled documents
4) Method to extrapolate from hand coding to unlabeled documents

# How Do We Generate Coding Rules and Categories?

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
    - Small working memories
    - Easily distracted
    - Insufficient motivation
2) Limits of Language:

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation
2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation
2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation
2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation
2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
   - Flow charts help simplify problems

# How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
    - Small working memories
    - Easily distracted
    - Insufficient motivation
2) Limits of Language:
    - Fundamental ambiguity in language [careful analysis of texts]
    - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
    - Flow charts help simplify problems
2) Train coders to remove ambiguity, misinterpretation

# How Do We Generate Coding Rules?

Iterative process for generating coding rules:

# How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules

# How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)

# How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)
3) Assess coder agreement

# How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)
3) Assess coder agreement
4) Identify sources of disagreement, repeat

# How Do We Identify Coding Disagreement?

Many measures of inter-coder agreement

Essentially attempt to summarize a confusion matrix

| | Cat 1 | Cat 2 | Cat 3 | Cat 4 | Sum, Coder 1 |
|---|---|---|---|---|---|
| Cat 1 | **30** | 0 | 1 | 0 | 31 |
| Cat 2 | 1 | **1** | 0 | 0 | 2 |
| Cat 3 | 0 | 0 | **1** | 0 | 1 |
| Cat 4 | 3 | 1 | 0 | **7** | 11 |
| Sum, Coder 2 | 34 | 2 | 2 | 7 | Total: **45** |

- **Diagonal**: coders agree on document
- Off-diagonal : coders disagree (confused) on document

Generalize across ($k$) coders:

- $\frac{k(k-1)}{2}$ pairwise comparisons
- $k$ comparisons: Coder A against All other coders

# How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

# How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

| | Coder A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **Tot** |
| Coder B | | | | | | | | | |
| **1** | **15** | 2 | 1 | 0 | 0 | 1 | 0 | 0 | |
| **3** | 1 | 0 | **0** | 1 | 0 | 0 | 0 | 0 | |
| **4** | 0 | 0 | 0 | **5** | 0 | 3 | 1 | 0 | |
| **5** | 0 | 0 | 0 | 1 | **13** | 7 | 0 | 2 | |
| **6** | 11 | 1 | 3 | 3 | 1 | **32** | 0 | 1 | |
| **7** | 1 | 0 | 0 | 0 | 0 | 13 | **26** | 36 | |
| **8** | 2 | 0 | 0 | 0 | 1 | 7 | 0 | **8** | |
| | | | | | | | | | |
| **Total** | 30 | 3 | 4 | 10 | 15 | 63 | 27 | 47 | |

# How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

| | Coder A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | Tota |
| Coder C | | | | | | | | | |
| **1** | **23** | 1 | 1 | 1 | 0 | 9 | 0 | 0 | |
| **2** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| **3** | 1 | 1 | **3** | 2 | 0 | 3 | 0 | 0 | |
| **4** | 0 | 0 | 0 | **4** | 0 | 8 | 1 | 0 | |
| **5** | 0 | 0 | 0 | 2 | **13** | 2 | 0 | 2 | |
| **6** | 4 | 1 | 0 | 1 | 1 | **32** | 1 | 2 | |
| **7** | 1 | 0 | 0 | 0 | 0 | 2 | **25** | 36 | |
| **8** | 1 | 0 | 0 | 0 | 1 | 6 | 0 | **7** | |
| | | | | | | | | | |
| Total | 30 | 3 | 4 | 10 | 15 | 63 | 27 | 47 | |

# How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

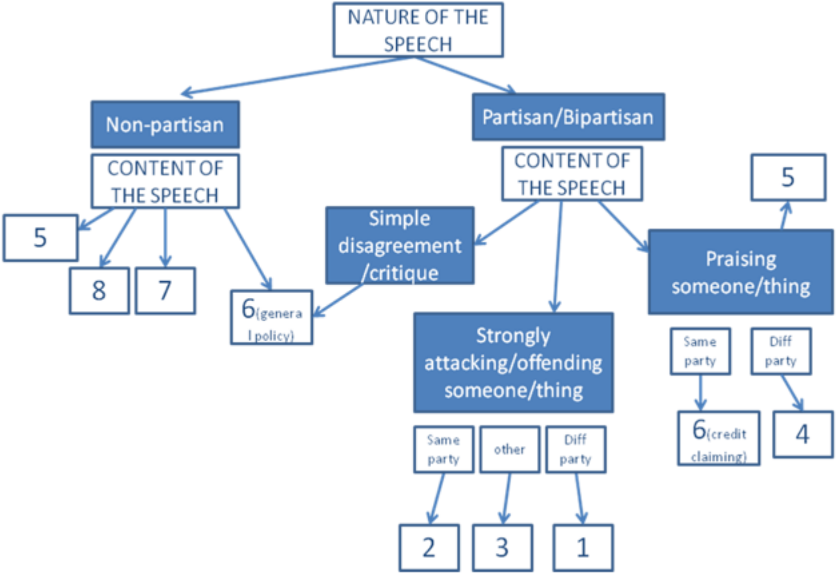
| | Coder C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | Tota |
| Coder B | | | | | | | | | |
| **1** | **18** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| **3** | 1 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | |
| **4** | 0 | 0 | 1 | **7** | 0 | 1 | 0 | 0 | |
| **5** | 0 | 0 | 0 | 2 | **18** | 3 | 0 | 0 | |
| **6** | 13 | 1 | 7 | 4 | 1 | **26** | 0 | 0 | |
| **7** | 3 | 0 | 0 | 0 | 0 | 8 | **63** | 2 | |
| **8** | 0 | 0 | 0 | 0 | 0 | 4 | 1 | **15** | |
| | | | | | | | | | |
| Total | 35 | 1 | 10 | 13 | 19 | 42 | 64 | 17 | |

# Example Coding Document

8 part coding scheme

- Across Party Taunting: explicit public and negative attacks on the other party or its members
- Within Party Taunting: explicit public and negative attacks on the same party or its members [for 1960's politics]
- Other taunting: explicit public and negative attacks not directed at a party
- Bipartisan support: praise for the other party
- Honorary Statements: qualitatively different kind of speech
- Policy speech: a speech without taunting or credit claiming
- Procedural
- No Content: (occasionally occurs in CR)

# Example Coding Document



# How Do We Summarize Confusion Matrix?

Lots of statistics to summarize confusion matrix:

- Most common: intercoder agreement

$$\text{Inter Coder}(A, B) = \frac{\text{No. (Coder A \& Coder B agree)}}{\text{No. Documents}}$$

Liberal measure of agreement:

Liberal measure of agreement:

- Some agreement by chance

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories $\{$ Class 1, Class 2$\}$.

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2}.
- Coder $A$ and Coder $B$ flip a (biased coin). ( Pr(Class 1) $=$ 0.75, Pr(Class 2) $=$ 0.25 )

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2}.
- Coder $A$ and Coder $B$ flip a (biased coin). ( Pr(Class 1) $=$ 0.75, Pr(Class 2) $=$ 0.25 )
- Inter Coder reliability: 0.625

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories $\{$ Class 1, Class 2$\}$.
- Coder $A$ and Coder $B$ flip a (biased coin). ( $\Pr(\text{Class 1}) = 0.75$, $\Pr(\text{Class 2}) = 0.25$ )
- Inter Coder reliability: 0.625

What to do?

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2}.
- Coder $A$ and Coder $B$ flip a (biased coin). ( $\Pr(\text{Class } 1) = 0.75$, $\Pr(\text{Class } 2) = 0.25$ )
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories $\{$ Class 1, Class 2$\}$.
- Coder $A$ and Coder $B$ flip a (biased coin). ( $\Pr(\text{Class } 1) = 0.75$, $\Pr(\text{Class } 2) = 0.25$ )
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

$$\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A \& Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}$$

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2}.
- Coder $A$ and Coder $B$ flip a (biased coin). ( Pr(Class 1) = 0.75, Pr(Class 2) = 0.25 )
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

$$\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A \& Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}$$

Question: what is amount expected by chance?

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories $\{$ Class 1, Class 2$\}$.
- Coder $A$ and Coder $B$ flip a (biased coin). ( Pr(Class 1) $= 0.75$, Pr(Class 2) $= 0.25$ )
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

$$\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A \& Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}$$

Question: what is amount expected by chance?

- $\frac{1}{\#\text{Categories}}$ ?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2}.
- Coder $A$ and Coder $B$ flip a (biased coin). ( Pr(Class 1) = 0.75, Pr(Class 2) = 0.25 )
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

$$\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A \& Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}$$

Question: what is amount expected by chance?

- $\frac{1}{\#\text{Categories}}$ ?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Best Practice: present confusion matrices.

# Krippendorf's Alpha

Define coder reliability as:

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

No. Pairwise Disagreements Observed = observe from data

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

No. Pairwise Disagreements Observed = observe from data

No Expected pairwise disagreements: coding by chance, with rate labels used available from data

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

No. Pairwise Disagreements Observed $=$ observe from data

No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

No. Pairwise Disagreements Observed = observe from data

No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:

- Pretend I know something I'm trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about "expectations" (from uniform, to data driven)

# Krippendorf's Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

No. Pairwise Disagreements Observed = observe from data

No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:

- Pretend I know something I'm trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about "expectations" (from uniform, to data driven)

Calculate in R with `concord` package and function `kripp.alpha`

# How Many To Code By Hand/How Many to Code By Machine

Next week: we'll discuss how to answer this question systematically for your data set.

Rules of thumb:

- Hopkins and King (2010): 500 documents likely sufficient
- Hopkins and King (2010): 100 documents may be enough
- BUT: depends on quantity of interest
- May REQUIRE many more documents

# Percent data coded, Error (From Dan Jurafsky)

## Training size

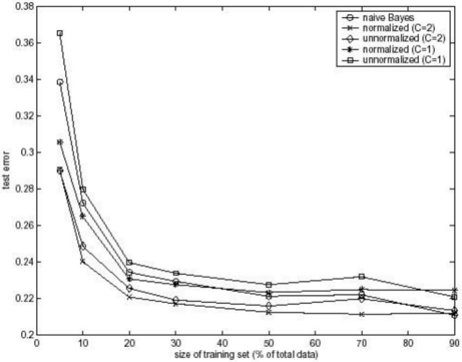



Figure 2: Test error vs training size on the newsgroups alt.atheism and talk.religion.misc

# Three categories of documents

Hand labeled

- Training set (what we'll use to estimate model)
- Validation set (what we'll use to assess model)

Unlabeled

- Test set (what we'll use the model to categorize)

Label more documents than necessary to train model

# Methods to Perform Supervised Classification

- Use the hand labels to train a statistical model.
- Naive Bayes
  - Shockingly simple application of Bayes' rule
  - Shockingly useful $\rightsquigarrow$ often default classifier

# Naive Bayes and General Problem Setup

Suppose we have document $i$, ($i = 1, \ldots, N$) with $J$ features

# Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features $\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$

# Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features
$\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$
Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$
$\{C_1, C_2, \ldots, C_K\}$

# Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features $\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$
Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$
$\{C_1, C_2, \ldots, C_K\}$
Subset of labeled documents $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where $Y_i \in \{C_1, C_2, \ldots, C_K\}$.

# Naive Bayes and General Problem Setup

Suppose we have document $i$, ($i = 1, \ldots, N$) with $J$ features
$\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$
Set of $K$ categories. Category $k$ ($k = 1, \ldots, K$)
$\{C_1, C_2, \ldots, C_K\}$
Subset of labeled documents $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where
$Y_i \in \{C_1, C_2, \ldots, C_K\}$.
Goal: classify every document into one category.

# Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features $\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$

Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$

$\{C_1, C_2, \ldots, C_K\}$

Subset of labeled documents $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where $Y_i \in \{C_1, C_2, \ldots, C_K\}$.

Goal: classify every document into one category.

Learn a function that maps from space of (possible) documents to categories

# Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features
$\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$
Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$
$\{C_1, C_2, \ldots, C_K\}$
Subset of labeled documents $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where
$Y_i \in \{C_1, C_2, \ldots, C_K\}$.
Goal: classify every document into one category.
Learn a function that maps from space of (possible) documents to categories
To do this: use hand coded observations to estimate (train) regression model

# Naive Bayes and General Problem Setup

Suppose we have document $i$, ($i = 1, \ldots, N$) with $J$ features
$\boldsymbol{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$
Set of $K$ categories. Category $k$ ($k = 1, \ldots, K$)
$\{C_1, C_2, \ldots, C_K\}$
Subset of labeled documents $\boldsymbol{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where
$Y_i \in \{C_1, C_2, \ldots, C_K\}$.
Goal: classify every document into one category.
Learn a function that maps from space of (possible) documents to categories
To do this: use hand coded observations to estimate (train) regression model
Apply model to test data, classify those observations

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$(1)$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg\max_k p(C_k|\boldsymbol{x}_i) \quad (1)$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg\max_k p(C_k|\boldsymbol{x}_i) \quad (1)$$

We're going to use Bayes' rule to estimate $p(C_k|\boldsymbol{x}_i)$.

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg\max_k p(C_k|\boldsymbol{x}_i)$$

We're going to use Bayes' rule to estimate $p(C_k|\boldsymbol{x}_i)$.

$$p(C_k|\boldsymbol{x}_i) = \frac{p(C_k, \boldsymbol{x}_i)}{p(\boldsymbol{x}_i)} \quad (1)$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg\max_k p(C_k|\boldsymbol{x}_i)$$

We're going to use Bayes' rule to estimate $p(C_k|\boldsymbol{x}_i)$.

$$\begin{aligned} p(C_k|\boldsymbol{x}_i) &= \frac{p(C_k, \boldsymbol{x}_i)}{p(\boldsymbol{x}_i)} \\ &= \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \end{aligned} \quad (1)$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $\boldsymbol{x}_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg\max_k p(C_k|\boldsymbol{x}_i)$$

We're going to use Bayes' rule to estimate $p(C_k|\boldsymbol{x}_i)$.

$$p(C_k|\boldsymbol{x}_i) = \frac{p(C_k, \boldsymbol{x}_i)}{p(\boldsymbol{x}_i)}$$

$$= \frac{\overbrace{p(C_k)}^{\text{Proportion in } C_k} \underbrace{p(\boldsymbol{x}_i|C_k)}_{\text{Language model}}}{p(\boldsymbol{x}_i)}$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$C_{\text{Max}} = \arg\max_k \ p(C_k|\boldsymbol{x}_i)$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$C_{\text{Max}} = \arg\max_k \; p(C_k|\boldsymbol{x}_i)$$
$$C_{\text{Max}} = \arg\max_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)}$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \arg\max_k \; p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \arg\max_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \arg\max_k \; p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \text{arg max}_k \; p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \text{arg max}_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \text{arg max}_k \; p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

Two probabilities to estimate:

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \text{arg max}_k \; p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \text{arg max}_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \text{arg max}_k \; p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \arg\max_k \; p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \arg\max_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \arg\max_k \; p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

$p(\boldsymbol{x}_i|C_k)$ complicated without assumptions

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \arg\max_k \; p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \arg\max_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \arg\max_k \; p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

$p(\boldsymbol{x}_i|C_k)$ complicated without assumptions

- Imagine each $x_{ij}$ just binary indicator. Then $2^J$ possible $\boldsymbol{x}_i$ documents

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$
\begin{aligned}
C_{\text{Max}} &= \arg\max_k \ p(C_k|\boldsymbol{x}_i) \\
C_{\text{Max}} &= \arg\max_k \ \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)} \\
C_{\text{Max}} &= \arg\max_k \ p(C_k)p(\boldsymbol{x}_i|C_k)
\end{aligned}
$$

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

$p(\boldsymbol{x}_i|C_k)$ complicated without assumptions

- Imagine each $x_{ij}$ just binary indicator. Then $2^J$ possible $\boldsymbol{x}_i$ documents
- Simplify: assume each feature is independent

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

$$C_{\text{Max}} = \text{arg max}_k \; p(C_k|\boldsymbol{x}_i)$$

$$C_{\text{Max}} = \text{arg max}_k \; \frac{p(C_k)p(\boldsymbol{x}_i|C_k)}{p(\boldsymbol{x}_i)}$$

$$C_{\text{Max}} = \text{arg max}_k \; p(C_k)p(\boldsymbol{x}_i|C_k)$$

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

$p(\boldsymbol{x}_i|C_k)$ complicated without assumptions

- Imagine each $x_{ij}$ just binary indicator. Then $2^J$ possible $\boldsymbol{x}_i$ documents
- Simplify: assume each feature is independent

$$p(\boldsymbol{x}_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)
- $p(\boldsymbol{x}_i | C_k) = \prod_{j=1}^{J} p(x_{ij} | C_k)$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)
- $p(\boldsymbol{x}_i | C_k) = \prod_{j=1}^{J} p(x_{ij} | C_k)$

Maximum likelihood estimation (training set):

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)
- $p(\boldsymbol{x}_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$

Maximum likelihood estimation (training set):

$$p(x_{im} = z|C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )}}{\text{No(C}= \text{C}_k)}$$

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)
- $p(\boldsymbol{x}_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$

Maximum likelihood estimation (training set):

$$p(x_{im} = z|C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )}}{\text{No(C} = \text{C}_k\text{)}}$$

Problem: What if No( Docs$_{ij}$ = z and C = C$_k$ ) = 0 ?

# Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)
- $p(\boldsymbol{x}_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$

Maximum likelihood estimation (training set):

$$p(x_{im} = z|C_k) = \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )}}{\text{No(C= C}_k\text{)}}$$

Problem: What if No( Docs$_{ij}$ = z and C = C$_k$ ) = 0 ?
$\prod_{j=1}^{J} p(x_{ij}|C_k) = 0$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C} = \text{C}_k) + k}$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z|C_k) = \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k\text{ )} + 1}{\text{No(C} = \text{C}_k) + k}$$

Algorithm steps:

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C} = \text{C}_k) + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i | C_k)$ on training data

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z | C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C} = \text{C}_k) + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i | C_k)$ on training data
2) Use this to identify most likely $C_k$ for each document $i$ in test set

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z|C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C= C}_k\text{)} + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i|C_k)$ on training data
2) Use this to identify most likely $C_k$ for each document $i$ in test set

$$C_i \quad = \quad \text{arg max }_k \hat{p}(C_k)\hat{p}(\boldsymbol{x}_i|C_k)$$

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z|C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C= C}_k) + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i|C_k)$ on training data
2) Use this to identify most likely $C_k$ for each document $i$ in test set

$$C_i \quad = \quad \text{arg max }_k \hat{p}(C_k)\hat{p}(\boldsymbol{x}_i|C_k)$$

Simple intuition about Naive Bayes:

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C= C}_k) + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i | C_k)$ on training data
2) Use this to identify most likely $C_k$ for each document $i$ in test set

$$C_i = \text{arg max }_k \hat{p}(C_k)\hat{p}(\boldsymbol{x}_i | C_k)$$

Simple intuition about Naive Bayes:

- Learn what documents in class $j$ look like

# Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

$$p(x_{ij} = z | C_k) \quad = \quad \frac{\text{No( Docs}_{ij} = \text{z and C} = \text{C}_k \text{ )} + 1}{\text{No(C} = \text{C}_k) + k}$$

Algorithm steps:

1) Learn $\hat{p}(C)$ and $\hat{p}(\boldsymbol{x}_i | C_k)$ on training data
2) Use this to identify most likely $C_k$ for each document $i$ in test set

$$C_i \quad = \quad \text{arg max }_k \hat{p}(C_k)\hat{p}(\boldsymbol{x}_i | C_k)$$

Simple intuition about Naive Bayes:

- Learn what documents in class $j$ look like
- Find class $k$ that document $i$ is most similar to

# Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

$$\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\
\boldsymbol{\theta} &\sim \text{Dirichlet}(\boldsymbol{\lambda}) \\
\boldsymbol{\tau}_i &\sim \text{Multinomial}(1, \boldsymbol{\pi}) \\
\boldsymbol{x}_i | \tau_{ik} = 1, \boldsymbol{\theta} &\sim \text{Multinomial}(n_i, \boldsymbol{\theta}_k)
\end{aligned}$$

# Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

$$\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\
\boldsymbol{\theta} &\sim \text{Dirichlet}(\boldsymbol{\lambda}) \\
\boldsymbol{\tau}_i &\sim \text{Multinomial}(1, \boldsymbol{\pi}) \\
\boldsymbol{x}_i | \tau_{ik} = 1, \boldsymbol{\theta} &\sim \text{Multinomial}(n_i, \boldsymbol{\theta}_k)
\end{aligned}$$

If we randomly sample documents $N_{\text{train}}$ and label them ($\boldsymbol{Y}$), then we can estimate

# Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

$$\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\
\boldsymbol{\theta} &\sim \text{Dirichlet}(\boldsymbol{\lambda}) \\
\boldsymbol{\tau}_i &\sim \text{Multinomial}(1, \boldsymbol{\pi}) \\
\boldsymbol{x}_i | \tau_{ik} = 1, \boldsymbol{\theta} &\sim \text{Multinomial}(n_i, \boldsymbol{\theta}_k)
\end{aligned}$$

If we randomly sample documents $N_{\text{train}}$ and label them ($\boldsymbol{Y}$), then we can estimate

$$\widehat{\pi}_k = \frac{\sum_{i=1}^{N} I(Y_i = k) + \alpha_k}{N_{\text{train}}}$$

# Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

$$
\begin{aligned}
\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\
\boldsymbol{\theta} &\sim \text{Dirichlet}(\boldsymbol{\lambda}) \\
\boldsymbol{\tau}_i &\sim \text{Multinomial}(1, \boldsymbol{\pi}) \\
\boldsymbol{x}_i | \tau_{ik} = 1, \boldsymbol{\theta} &\sim \text{Multinomial}(n_i, \boldsymbol{\theta}_k)
\end{aligned}
$$

If we randomly sample documents $N_{\text{train}}$ and label them $(\boldsymbol{Y})$, then we can estimate

$$
\begin{aligned}
\widehat{\pi}_k &= \frac{\sum_{i=1}^{N} I(Y_i = k) + \alpha_k}{N_{\text{train}}} \\
\widehat{\theta}_{jk} &= \frac{\sum_{i=1}^{N} I(Y_i = k) x_{ij} + \lambda_j}{\sum_{j=1}^{J} \sum_{i=1}^{N} I(Y_i = k) x_{ij}}
\end{aligned}
$$

# Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

# Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$p(\tau_{ik} = 1 | \boldsymbol{x}_i, \widehat{\boldsymbol{\pi}}, \widehat{\boldsymbol{\theta}}) \quad \propto \quad p(\tau_{ik} = 1) p(\boldsymbol{x}_i | \boldsymbol{\theta}, \tau_{ik} = 1)$$

# Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$
\begin{aligned}
p(\tau_{ik} = 1 | \boldsymbol{x}_i, \widehat{\boldsymbol{\pi}}, \widehat{\boldsymbol{\theta}}) &\propto p(\tau_{ik} = 1) p(\boldsymbol{x}_i | \boldsymbol{\theta}, \tau_{ik} = 1) \\
&\propto \widehat{\pi_k} \prod_{j=1}^{J} \left( \widehat{\theta}_{jk} \right)^{x_{ij}}
\end{aligned}
$$

# Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$
\begin{aligned}
p(\tau_{ik} = 1 | \boldsymbol{x}_i, \widehat{\boldsymbol{\pi}}, \widehat{\boldsymbol{\theta}}) &\propto p(\tau_{ik} = 1) p(\boldsymbol{x}_i | \boldsymbol{\theta}, \tau_{ik} = 1) \\
&\propto \widehat{\pi_k} \prod_{j=1}^{J} \left(\widehat{\theta}_{jk}\right)^{x_{ij}} \\
&\propto \overbrace{\widehat{\pi_k}}^{p(C_k)} \underbrace{\prod_{j=1}^{J} \left(\widehat{\theta}_{jk}\right)^{x_{ij}}}_{\text{Unigram model}}
\end{aligned}
$$

# Some R Code

```
library(e1071)
dep<- c(labels, rep(NA, no.testSet))
dep<- as.factor(dep)
out<- naiveBayes(dep~., as.data.frame(tdm))
predicts<- predict(out, as.data.frame(tdm[-training.set,]))
```

# Assessing Models (Elements of Statistical Learning)

- Model Selection: tuning parameters to select final model (next week's discussion)
- Model assessment : after selecting model, estimating error in classification

# Comparing Training and Validation Set

Text classification and model assessment

- Replicate classification exercise with validation set
- General principle of classification/prediction
- Compare supervised learning labels to hand labels

Confusion matrix

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

$$\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}$$

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

$$\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}$$

$$\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}$$

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

$$\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}$$

$$\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}$$

$$F_{\text{Liberal}} = \frac{2\text{Precision}_{\text{Liberal}}\text{Recall}_{\text{Liberal}}}{\text{Precision}_{\text{Liberal}} + \text{Recall}_{\text{Liberal}}}$$

# Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

| | Actual Label | |
|---|---|---|
| Classification (algorithm) | Liberal | Conservative |
| Liberal | True Liberal | False Liberal |
| Conservative | False Conservative | True Conservative |

$$\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}$$

$$\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}$$

$$\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}$$

$$F_{\text{Liberal}} = \frac{2\text{Precision}_{\text{Liberal}}\text{Recall}_{\text{Liberal}}}{\text{Precision}_{\text{Liberal}} + \text{Recall}_{\text{Liberal}}}$$

# ROC Curve

ROC as a measure of model performance

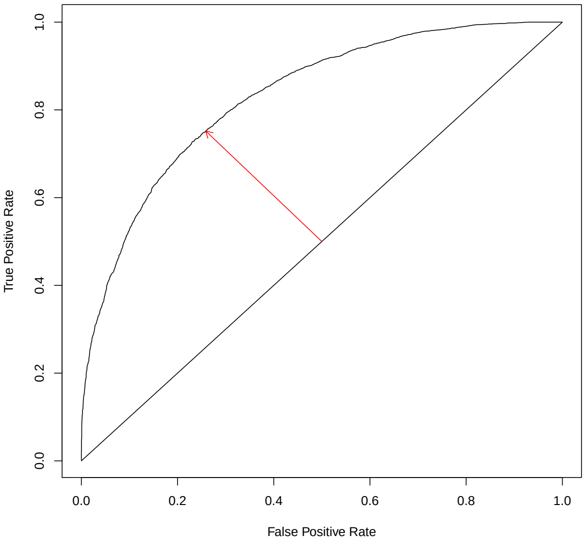
$$\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}$$
$$\text{Recall}_{\text{Conservative}} = \frac{\text{True Conservative}}{\text{True Conservative} + \text{False Liberal}}$$

Tension:

- Everything liberal: $\text{Recall}_{\text{Liberal}} = 1$ ; $\text{Recall}_{\text{Conservative}} = 0$
- Everything conservative: $\text{Recall}_{\text{Liberal}} = 0$ ; $\text{Recall}_{\text{Conservative}} = 1$

Characterize Tradeoff:
Plot True Positive Rate $\text{Recall}_{\text{Liberal}}$
False Positive Rate $(1 - \text{Recall}_{\text{Conservative}})$

# Precision/Recall Tradeoff



# Simple Classification Example

Analyzing house press releases
Hand Code: 1,000 press releases

- Advertising
- Credit Claiming
- Position Taking

Divide 1,000 press releases into two sets

- 500: Training set
- 500: Test set

Initial exploration: provides baseline measurement at classifier performances
Improve: through improving model fit

# Example from Ongoing Work

| | Actual Label | | |
|---|---|---|---|
| Classification (Naive Bayes) | Position Taking | Advertising | Credit Claim. |
| Position Taking | 10 | 0 | 0 |
| Advertising | 2 | 40 | 2 |
| Credit Claiming | 80 | 60 | 306 |

$$\begin{aligned}
\text{Accuracy} &= \frac{10+40+306}{500} = 0.71 \\
\text{Precision}_{PT} &= \frac{10}{10} = 1 \\
\text{Recall}_{PT} &= \frac{10}{10+2+80} = 0.11 \\
\text{Precision}_{AD} &= \frac{40}{40+2+2} = 0.91 \\
\text{Recall}_{AD} &= \frac{40}{40+60} = 0.4 \\
\text{Precision}_{Credit} &= \frac{306}{306+80+60} = 0.67 \\
\text{Recall}_{Credit} &= \frac{306}{306+2} = 0.99
\end{aligned}$$

# Fit Statistics in R

`RWeka` library provides **Amazing** functionality.

We'll have more to say on how to install, use this next week!