Topic Models

topic_models.R
Bayesian Methods

Simple

Naive Bayes, a “set the baseline” method

Introduces common independence assumption used in other models

Complex

Topic models, a hierarchical modeling approach

Example of a probabilistic generative model

Unsupervised, like LSA

Supervised version also available

Linked to vector space models
Naive Bayes

Classification problem

Assign class label to $Y$ given collection of categorical indicators (e.g., word present/absent)

Assign to category $\hat{Y}$ that maximizes conditional probability

$$\max_y P(Y=y|X_1, X_2, \ldots X_k)$$

Complication

Suppose $k$ is very large, possibly larger than number of obs

Lack enough examples to build conditional probability from frequencies

Example: Federalist papers

75 documents, but 10,000 word vocabulary

Naive Bayes is competitive in cases with few training examples

Provided its assumptions hold
Naive Bayes Solution

Employ Bayes rule

\[ P(Y|X) \cdot P(X) = P(X|Y) \cdot P(Y) \rightarrow P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)} \]

\[ \max_y P(Y=y|X_1, X_2, \ldots X_k) = \max_y P(X_1, X_2, \ldots X_k|Y) \cdot P(Y) \]

Assumptions

Know prior probabilities (such as equal!)

\[ \max_y P(Y=y|X_1, X_2, \ldots X_k) = \max_y P(X_1, X_2, \ldots X_k|Y) \]

\( X_j \) are conditionally independent given \( Y \)

\[ \max_y P(Y=y|X_1, X_2, \ldots X_k) = \max_y P(X_1|Y) \cdot P(X_2|Y) \cdot \ldots \cdot P(X_k|Y) \]

Rationale in language

Reduces problem to product of frequencies from 2x2 contingency tables in case of words/text
Example: Federalist Papers

Federalist papers

85 essays advocating US Constitution in 1787-1788

Revisit text by Mosteller and Wallace (1964)
Who wrote the 12 disputed Federalist papers?

Supervised classification

Hamilton 51
Madison 14
Jay 5

Hamilton & Madison 3
Federalist Papers

Data

Nothing fancy: a CSV file
  Elaborate data processing needed for web-scale applications

Three “variables” for each of 85 documents
  author, number, text

Sample

  To the People of the State of New York:  AFTER an unequivocal experience
  of the inefficacy of the subsisting federal government, you are called upon
  to deliberate on a new Constitution for the United States of America…

Preprocessing

  Downcase

  Want a document-term matrix for identifying useful words
Results of Naive Bayes

Simple analysis

Identify whether a word appears or not (0/1) rather than count

Component probabilities $P(X_w|Y)$ reduce to relative frequency of a word appearing in the papers written by each author

Which words to use

Words that are reasonably common

Avoid words that appear in every document.

Avoid words that don’t get used by an author.

What about the prior probability?

Compare to other classifiers
Topic Models

Conceptual model for the generation of text

Text expresses an idea or “topic”
   Presidential address might move from domestic economics to foreign policy to health care.

Current topic determines the chances for various word choices
   The words “inflation” or “interest rate” are more likely to appear when discussing economic policies rather than foreign policy

Hierarchical model

Identify the number of topics

Define a probability distribution for each

Each document mixes words drawn from topics

Conditional independence, given topic (naive Bayes)
Heuristic Motivation

Each document mixes words from collection of topics

topic = probability distribution over words

Original details: Blei, Ng, and Jordan 2003
Probability Model

Latent Dirichlet allocation (LDA)

Define K topics

Discrete dist over vocabulary \( P_k \sim \text{Dirichlet}(\alpha), k = 1, \ldots, K \)

Parameter \( \alpha \) controls sparsity of the distribution

Each document mixes topics

Distribution over topics in doc \( i \) \( \theta_i \sim \text{Dirichlet}, i = 1, \ldots, n \)

\( \theta_i \) are probabilities

Word probability \( \text{P}(w \text{ in doc } i) = P_k(w) \quad k \sim \text{Multi}(\theta_i) \)

Number of words within doc allowed to be random/fixed

Beta:Binomial as Dirichlet:Multinomial
Expected Word Counts

Matrix product determines counts

Let $K \times m$ matrix $P$ denote the matrix with probability distribution $P_k$ in the $k^{th}$ row.

Let the $n \times K$ matrix $T$ denote the mix of topics in the documents, with the mix for document $i$ in row $i$.

Then the expected number of word tokens of type $j$ in document $i$ is $(T P)_{ij}$.

Factorization

Topics models imply a factorization of the expected count matrix, the document term matrix $C$

$$E(C) = n_i T P$$

and the SVD is one way of factoring $C$!
Example

Simulate data from a topic model

Pick the number $K$ of topics

Pick size $m$ of the vocabulary and the number of documents $n$

Choose $\alpha_P$ that controls “sparsity” of topic distributions

Small $\alpha_P$ produces nearly singular distributions with little overlap.

\[ \alpha_P = 0.025 \]

\[ \alpha_P = 0.100 \]
Simulate the Documents

Generate documents

Choose average length of documents (poisson distribution)

Pick \( \alpha_T \) that controls the mix of topics within documents

Small \( \alpha_T \) produces documents predominantly of one topic.

\[ \alpha_T = 0.1 \]

\[ \alpha_T = 0.4 \]

\( n = 5000 \)

\( m = 1000 \)

\( n_i \approx 100 \)

\( K = 10 \) topics

\( \alpha_T = 0.4 \) in following
Word Frequencies

Typically not very close to Zipf as we find in real text
LSA Analysis

Compute the SVD of the counts

Raw counts and using CCA weights

Number of topics stands out clearly, particularly in CCA
LSA Analysis

Loadings have the “ray-like” behavior

Similar to those in LSA analysis of wine tasting notes

More clearly defined
Topic Model Analysis

Same simulated data

Pick number of topics (e.g., know there are 10)
Input the associated DTM

Results

Indicates which topics most prevalent in documents
Associates word types with the discovered topics

Goodness-of-fit

Obtain overall log-likelihood of fitted model
Vary the number of topics to see how fit changes
Topic Models: Wine

Fit topic models to the data set of wine tasting notes

Use all 20508 documents, with 2659 word types
after removing/merging the OOV types

Fit with K=10 topics

Topics in documents

Lists topics comprising the tasting notes

Word types in topics

Not real exciting…

Documents too short?
Unsupervised Modeling

Pretend we don’t have a response.

Do frequencies of words reveal clusters?

Unsupervised model

No response variable

Which documents are similar

Document similarity

Data is very sparse:
2659 types (OOV) but only $\approx 37$ tokens in doc

Random projection preserves distances
Word Embedding
n-grams
Bigrams, n-Grams

Document term matrix

- Associates words that appear in same “context”
- A document defines the context
- Natural association for modeling a property of a document

n-Gram matrix

- Bigram: The adjacent word defines the context
- Trigram: The adjacent words to either side define the context
- n-gram: Use varying numbers of adjacent words
- Designed to study the relationship of words
Return to Token Space

Bigram matrix origins

Consider two matrices with elements 0 and 1

Total number of rows in each = total number of word tokens - 1

<table>
<thead>
<tr>
<th>Prior word type</th>
<th>W_1</th>
<th>W_2</th>
<th>W_3</th>
<th>...</th>
<th>W_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word types</th>
<th>W_1</th>
<th>W_2</th>
<th>W_3</th>
<th>...</th>
<th>W_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t_5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_{N-1}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>t_N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

N = total # tokens
m = # word types
Bigram Matrix

Matrix calculation

Matrix product times its “lag”

\[ B = W_{-1}^T W \]

so that

\[ B_{ij} = \#\{\text{token of word type } w_i \text{ precedes } w_j\} \]

\( B \) is an \( m \times m \) matrix, where \( m = \) size of vocabulary

Interpretation as covariance

Consider the rows of the \( N \times m \) matrix \( W \) as flowing over time

stochastic process that picks the words

\[ B_{ij} = N \ \text{cov}(w_i, w_j) \]

again, ignoring the mean values that will be very close to 0

Word order matters!
Bigram Matrix

Standardization

Word types that are more common will tend to co-occur more often than word types that are more rare.

Weighting, such as CCA or td-idf, are common.
CCA divides by square root of the product of the type frequencies.

CCA weights convert the covariance into a correlation approximately, because \( \sqrt{m_j} \approx \text{sd}(j^{th \text{ column of } W}) \).

Tokenization

Key choices remain highly relevant.

Stemming, removing punctuation, handling OOVs.
Bigrams and Models

Hidden Markov model

Imagine underlying language communicates sequence of ideas or concepts, say $H_k$, for $k = 1,\ldots, K$

Each concept is associated with a certain vocabulary of words, say $V_k$.

We can learn about the concepts by discovering words that tend to occur near each other, or be used in the same way.
Word Embedding

Theory

SVD of the bigram matrix $B$ reveals aspects of hidden states

Conversion using “thin” SVD

Retain some of the components of the SVD of bigram matrix (after standardizing)

$$B \rightarrow \text{UDV}^T_{mxm}$$

Suppose we retain $d$ components, then the rows of $U$ (an $m \times d$ matrix) provide an embedding of words in a $d$-dimensional, real-valued space.

Random projection ideas are typically necessary for handling a large corpus with a diverse vocabulary ($m \approx 100,000$ or more)
Examples of Embeddings

Parts of speech

Obtained from analysis of much larger corpus

Regular text rather than domain specific text like wine reviews

noun
verb
adj
unk

OOV in black
Examples of Embeddings

Plot of two singular vectors
Examples of Embeddings

Zoomed in view of same singular vectors
Examples of Embedding

Numbers as words and digits

PC 3

PC 2

1 2 3 4 5 6 7 8 9 10

2005
Bigrams in R

Typically weighted, but worked better here with small corpus to leave raw counts.
Word2vec

Alternative approach to word embedding
   Derived from “deep” neural network

Motivating probability model
   Build a model for $P(W_t|W_{t-1}, W_{t-2}, \ldots)$
   Output a probability distribution over next word
   Bigram case has one preceding word for the context

Popularity
   Algorithm for solving large neural network
   Fast implementation, very effective demonstration
**Word2vec Structure**

**Deep network**

**Network structure**

Input $x$ is dummy word indicator, $(x^TW) = h^T$ hidden state

Output “softmax” $y_j = \exp u_j / \sum \exp u_j$, $u_j = (h^TW')_j$

---

Rong, “word2vec parameter encoding explained”

---

[Diagram showing the structure of a Word2vec network with input, hidden, and output layers, one-hot encoding, linear and logit functions, and probability distribution with a large number of parameters to estimate.]
Idea of Embedding

Text

The quick brown fox **jumped** over the fence.

context    target

Choose vector of coordinates $V_c$ to represent context word and to represent target word $V_T$ so that score $V_c^T V_T$ is high.

Wrong text: The quick brown fox **ate** over the fence.

Choose vector of coordinates $V_c$ to represent context word and to represent WRONG target word $V_w$ so that score $V_c^T V_w$ is small.
Example

Code widely available on Internet

Train during class

  Compute intensive, so I will run on a server back at Penn
  Build N = 200 dimensional hidden state vector
  Loads a corpus to build
  Trains in about 5 minutes

Word analogies

  paris:france :: london: ???
  king:man :: queen: ???

Lots better with much larger corpus
More Examples

See papers of Mikolov et al (Google)

from TensorFlow site

<table>
<thead>
<tr>
<th>Czech + currency</th>
<th>Vietnam + capital</th>
<th>German + airlines</th>
<th>Russian + river</th>
<th>French + actress</th>
</tr>
</thead>
<tbody>
<tr>
<td>koruna</td>
<td>Hanoi</td>
<td>airline Lufthansa</td>
<td>Moscow</td>
<td>Juliette Binoche</td>
</tr>
<tr>
<td>Check crown</td>
<td>Ho Chi Minh City</td>
<td>carrier Lufthansa</td>
<td>Volga River</td>
<td>Vanessa Paradis</td>
</tr>
<tr>
<td>Polish zolty</td>
<td>Viet Nam</td>
<td>flag carrier Lufthansa</td>
<td>upriver</td>
<td>Charlotte Gainsbourg</td>
</tr>
<tr>
<td>CTK</td>
<td>Vietnamese</td>
<td>Lufthansa</td>
<td>Russia</td>
<td>Cecile De</td>
</tr>
</tbody>
</table>

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.
Deep Learning
Deep Learning

Continuing development of large neural networks in models of language

Recursive neural network

Sequence to sequence learning

Used for grammatical error correction, language translation

Long-short term memory (LSTM) network nodes

Very large networks require substantial computing resources to run in reasonable time

Commonly built using graphics processors for faster matrix calculations
TensorFlow Animation

Online example of large neural networks

Not for text

Useful to explore flexibility

www.tensorflow.org
Text Examples

Language generator

Show it lots of examples of language
Builds a probability model for the language
Can use to classify language source

Example

Version of the code from Zaremba “Learning to Execute”
Build model (takes a while to train on a laptop!)
Character level generator (not words, it works at char level)
Need a lot more text for training than the “few” wine reviews
Example: Generating Text

Generate new reviews

Can you tell the type of wine being described?

Pale golden straw color. Lemon oil, lemon zest, and cinnenfruit flavors. Finishes with a slighty spring, and fruit dry finish. A very nice depth of fruit, this has mouch tarmen and floral acidity in the finish.
Brilliant yellow hue. Yeast, dried citrus and dried apple and merballoon aromas. Medium-bodied, this has aple and crisp tropical fruit and a touch of spice and an-partint valb finish.
A somp, short finish with elegantly langer.
Pale golden silver color. Rubber bash aromas follow through on a broem, melon fone food win e.
Golden color. Floral toasty, lemon seaut, and roable cake rind intession and apple skin Ri nese.

Crunchy leafy cherry, blackberry, nutmeg and dried hell fade. A very nice effort has distinctive noce-depty medium-full body and a long, zesty, and quaffer.
Creamy berry, mydill-bodied palate with soft tannins, nicely integrated, tangy f inish with lively thavines, tasty Shyrom marinagek that tasty!
Black cherry, plum, saged oak and singer accented finish. Ample oak and pleasn's lacked, well maunteriled rubbing's steak.
Limerbinoaro, tomato, cherry and black fruit kis aromas follow through on a mediu m-bodied palate with chewy tannins and lively acidity. A ripe, remonion foods.
Example: Scoring Text

Scoring existing text

DNN builds a probability model, so it can assign a likelihood to a review as being a review of red or of white wine.

Feed notes on tasting red wines into both models

Ability to compress = log-likelihood

High compression = good match
Closing Comments
Parting Comments

Text analytics

Continues to move into mainstream

Objectives

Build features for “familiar” models
Understanding the structure of language

Issues of statistical modeling for large data sets remain
Overfitting, missing data, outliers, …

Computing

Methods related to deep learning have become more widely accessible, and hence more common

What’s the role for the social scientist?