Natural Language Processing
Applications to Sentiment Analysis and Mental Health Monitoring

Avi Schwarzschild & Abhi Chatterjee

Department of Applied Mathematics, Columbia University

October 5, 2016

Overview

Importance

- Natural Language Processing (NLP) is field that merges computer science, applied mathematics, and computational linguistics to interpret interactions between computers and human language
- NLP is a field with fascinating applications for machine learning, many modern learning algorithms were developed for NLP
- We believe that detecting depression would be a highly nontrivial application of the field

References

- Balani, S., & De Choudhury, M. (2015). Detecting and Characterizing Mental Health Related Self-Disclosure in Social Media. *In Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 13731378). New York, NY, USA: ACM.
- Blei, D. M., Ng, A. Y., Jordan, M. I., & Lafferty, J. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*.
- McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification.
- Warner HR, Toronto AF, Veasey L, Stephenson R. A Mathematical Approach to Medical Diagnosis: Application to Congenital Heart Disease. *JAMA*. 1961; 177(3): 177-183.

NLP Timeline

- 1950: Alan Turing publishes "Computing Machinery and Intelligence"
- 1964: MIT AI Lab develops ELIZA, using pattern matching and scripts to process user inputs and engage in conversation
- Late 1980s: First statistical machine translation systems developed, introduction of machine learning revolutionizes the field
- 2006: IBM Watson developed, and by 2013 is used in MSKCC to manage decisions in lung cancer treatment
- Very popular modern application: NLP is used by brands to follow relevant sentiment trends on the internet
- Current research: diagnosing depression, detecting lies and much more

Naive Bayes Classifier

Text Classification

- Input:
  1. Set of documents $\mathcal{D} = \{d_1, d_2, ..., d_k\}$
  2. Set of classes $\mathcal{C} = \{c_1, c_2, ..., c_n\}$
- Output:
  1. Predicted class $c \in \mathcal{C}$ for each document $d \in \mathcal{D}$

Naive Bayes Classifier

- Feature - a description of some aspect of out data, ex: color, size, presence
- Label - a classification of a data point, ex: apple, dog, evidence of depression
- Training Data - a set of data that we know everything about
- Test Data - more classified data to use for testing the accuracy of the algorithm
- The Real Data - unclassified data for which we want classifications

Naive Bayes Classifier

- Feature vector: $\vec{x} = (x_1, x_2, ...x_d)$
- Label set: $y \in \{1, 2, 3..k\}$
- Training data is a set: $(\vec{x}^{(i)}, y^{(i)})$, where each data point is properly labeled

The classifier: $f(\vec{x}) : \{0, 1\}^d \rightarrow y$

Naive Bayes Classifier

Given an unclassified data point, we express it as a feature vector:

$$\vec{x} = (x_1, x_2, ...x_d)$$

We look for:

$$\max_y P(y|\vec{x}) = \max_y \frac{P(\vec{x}|y)P(y)}{P(\vec{x})} \tag{1}$$

$$\max_y P(y|\vec{x})P(\vec{x}) = \max_y P(\vec{x}|y)P(y) \tag{2}$$

$$\max_y P(y, \vec{x}) = \max_y P(\vec{x}|y)P(y) \tag{3}$$

Derivation of Naive Bayes

From the previous slide:

$$\max_y P(y, \vec{x}) = \max_y P(\vec{x}|y)P(y)$$

Let's examine the following term: $P(\vec{x}|y)$
Because of our assumption that features are independent, we can say:

$$P(\vec{x}|y) = P(x_1, x_2, ...x_d|y) \tag{4}$$

$$P(\vec{x}|y) = \prod_{j=1}^{d} P(x_j|y) \tag{5}$$

$$\boxed{P(y, \vec{x}) = P(y) \prod_{j=1}^{d} P(x_j|y)}$$

Bernoulli and Multinomial Naive Bayes

Simple (a.k.a Multi-variate Bernoulli) Model:

Features are binary: $\vec{x} \in \{0, 1\}^d$

Multinomial Model:

Features can take integer values: $\vec{x} \in \mathbb{Z}^d$

How does the math change?

Bernoulli and Multinomial Naive Bayes

Given a document d_i and a label y, what is the probability of the document having the given label y?
We find the document's feature vector $\vec{x_i}$, and we use our model's vocabulary V:

$$P(y, \vec{x_i}) = P(y)P(\vec{x_i}|y)$$

Bernoulli Model:

$$P(\vec{x_i}|y) = \prod_{j=1}^{|V|}[x_{ij}P(w_j|y) + (1 - x_{ij})(1 - P(w_j|y))]$$

Multinomial Model:

$$P(\vec{x_i}|y) = (\sum_{j=1}^{|V|} x_{ij})! \prod_{j=1}^{|V|} \frac{P(w_j|y)^{x_{ij}}}{x_{ij}!}$$

Smoothing

- What happens when the classifier encounters a feature in the test set that it hasn't seen before?
- The probability that it is assigned to its appropriate class is zero, since the estimator is based on observed counts
- Laplace smoothing compensates for this by applying a correction to the probability estimator:

$$\theta_i = \frac{X_i + \alpha}{N + d\alpha}$$

- Laplace smoothing is built in to the nltk module and assigns $\alpha = 1$ (d is the range of the counts)

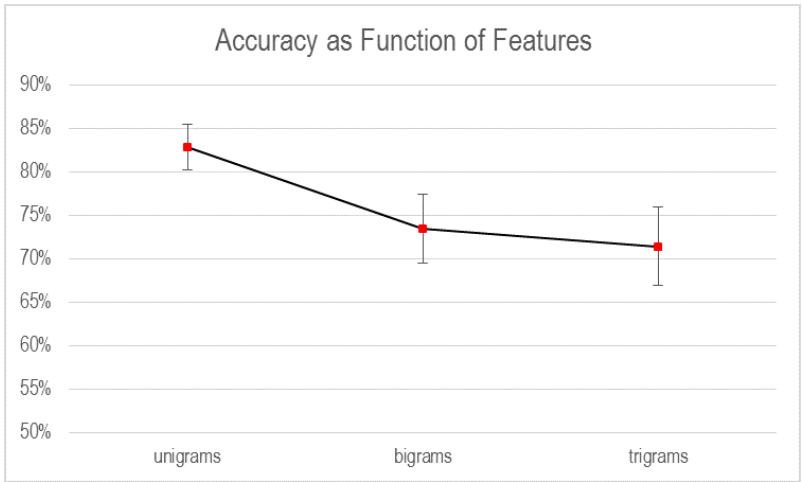
Pseudo Code

Example (Pseudocode for Training Bernoulli NB Classifier)

```
1   Input(C, D)
2   V <- extract_vocabulary(D)
3   N <- count_docs(D)
4   Prior[length(C)]
5   likelihood[V,length(C)]
6   for c in C:
7       N_c <- count_all_docs_in_class(D, c)
8       prior[c] <- N_c / N
9       text_c <- all_text_in_class(D, c)
10      for word in V
11          likelihood(word, c) <- count_word_c/length(text_c)
12
13  Output(V, Prior, likelihood)
```

Detecting Depression on Social Media

- We used the Natural Language Toolkit (nltk) Python module to apply the Bernoulli Naive Bayes Algorithm to detect depression self disclosure on Twitter
- We streamed 1,000 tweets from the Twitter API and assigned them to a binary class: 1 = contains depression self-disclosure, 0 = no self-disclosure
- We did 10 fold cross validation using either unigrams, bigrams, or trigrams as relevant features



Detecting Depression on Social Media

- The independence assumption starts to influence the accuracy of the classifier as the words attached to each feature increases
- Accuracy decreases and the variability of the estimator increases as more words are attached to each feature
- The number of bigrams and trigrams per sentence decreases, and they are not independent from each other

Table 1.

Unigrams	Bigrams	Trigrams
"Depressed"	"I'm depressed"	"Feel like im"
"Sad"	"so depressed"	"Was so depressed"
"Mood"	"feel like"	"my life is"
"Feel"	"life is"	"so depressed at"
"Making	"the same"	"I feel bad"

- Table 1 displays n-grams that were the most relevant features when predicting whether a tweet had depression self-disclosure content

Latent Dirichlet Allocation

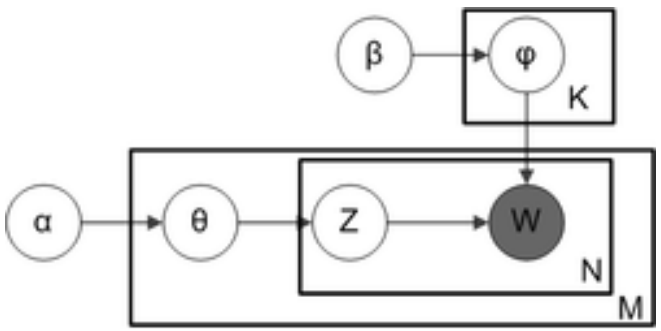


Figure 1: Graphical model for LDA (Blei, 2003)

- Latent Dirichlet Allocation (LDA) broadly is the process of generating topic models that describe a set of documents
- LDA is an improvement over other topic models in that it employs a *three* level hierarchical bayesian approach

LDA Notation

- **N** - The length of the document is drawn from *Poisson*(δ)
- α - Parameter for the per document topic distribution
- β - Parameter for the per topic word distribution
- θ_i - The topic distribution for document i which is drawn from *Dirichlet*(α)
- z_{ij} - Topic for the jth word in document i which is drawn from Multinomial(θ_i)
- w_{ij} - The specific word which is drawn from p($w_{ij}|z_{ij}, \beta_j$)

LDA Assumptions

- Dimensionality k of the Dirchlet distribution, and hence the number of topics, is fixed
- The word probabilities are parameterized by the matrix β which has dimension $k * V$ and is to be estimated
- The k-dimensional Dirichlet random variable θ is constrained to take values in the (k-1) simplex: $(\theta_i \geq 0, \sum_{i=1}^{k} \theta_i = 1)$
- These properties dictate LDA inference and parameter estimation approaches

LDA Assumptions

- When given the per document topic distribution parameter α, and the per topic word distribution parameter β, the joint distribution of θ, $\mathbf{z}$, and $\mathbf{w}$ is:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

- The marginal word distribution of a document is:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) d\theta \right)$$

Inference and Estimation

- We can gain useful information from LDA by computing the posterior distribution of the hidden topics given a document:

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}$$

- We know that θ is drawn from $Dir(\alpha)$ so it has density:

$$p(\theta|\alpha) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1}$$

- If we plug this into the expression from the previous slide we get an intractable function

Variational Inference

- Convexity-based variational inference uses Jensen's inequality to provide a lower bound for the marginal log likelihood which can then be maximized
- We can use this to approximate the posterior which we just showed to be intractable
- We first replace α and β with free parameters γ and ϕ

$$q(\theta, z|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n)$$

Variational Inference

- Expectation Maximization Algorithm: E step - For each document, find the optimizing values $(\gamma*, \phi*)$
- We will show that γ and ϕ satisfy: $(\gamma*, \phi*) = \text{argmin } \mathcal{D}(q(\theta, z|\gamma, \phi)||p(\theta, z, |w, \alpha, \beta))$
- Therefore these values minimize the KL divergence between the variational distribution and the true posterior
- The Dirichlet is an exponential distribution with sufficient statistic $T(\theta_i) = log(\theta_i)$ and natural param $\eta_i = \alpha_i - 1$
- Therefore $E[T(\theta_i)|\alpha] = \Psi(\alpha_i) - \Psi(\sum_{i=1}^{k} \alpha_i)$ where $\Psi = \frac{d}{d\alpha} log(\Gamma)$ is the digamma function

Variational Inference

- M step - Maximize the resulting lower bound on log likelihood
- Hence we must maximize the marginal log likelihood: $l(\alpha, \beta) = \sum_{d=1}^{M} log(p(w_d|\alpha, \beta))$
- Apply Jensen Inequality:

$$log(p(\mathbf{w}|\alpha, \beta)) = log\left(\int \sum \frac{p(\theta, z, w|\alpha, \beta) q(\theta, z|\gamma, \phi)}{q(\theta, z|\gamma, \phi)} d\theta\right)$$

$$\geq \int \sum q(\theta, z) logp(\theta, z, w|\alpha, \beta) d\theta - \int \sum q(\theta, z) logq(\theta, z) d\theta$$

$$= E_q[logp(\theta, z, w|\alpha, \beta)] - E_q[logq(\theta, z|\gamma, \phi)]$$

- Applying Lagrange multipliers and setting derivatives of the KL divergence = 0 yields $\phi_{ni} = \beta_{iw_n} e^{E_q[log(\theta_i)|\gamma]}$ and $\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}$

Variational Inference

- $E_q[log(p(\theta, z, w|\alpha, \beta))] - E_q[log(q(\theta, z|\gamma, \phi))] =$
- $E_q[log(p(\theta|\alpha))] + E_q[log(p(z|\theta))] + E_q[log(p(w|z, \beta))] - E_q[log(q(z|\gamma, \phi))] - E_q[log(q(z|\gamma, \phi))]$
- $E_q[log(p(\theta|\alpha))] = log(\Gamma(\sum_j^k \alpha_j)) - \sum_i^k log(\Gamma(\alpha_i)) + \sum_i^k (\alpha_i - 1)(\psi(\gamma_i) - \psi(\sum_j^k \gamma_j))$
- $E_q[log(p(z|\theta))] = \sum_n^N \sum_i^k \phi_{ni}(\psi(\gamma_i) - \psi(\sum_j^k \gamma_j))$
- $E_q[log(p(w|z, \beta))] = \sum_n^N \sum_i^k \sum_j^V \phi_{ni} w_n^{(j)} log(\beta_{ij})$
- $E_q[log(q(z|\gamma, \phi))] = log(\Gamma(\sum_j^k \gamma_j)) - \sum_i^k log(\Gamma(\gamma_i)) + \sum_i^k (\gamma_i - 1)(\psi(\gamma_i) - \psi(\sum_j^k \gamma_j))$
- $E_q[log(q(z|\gamma, \phi))] = \sum_n^N \sum_i^k \phi_{ni} log(\phi_{ni})$

LDA Inference Algorithm

Example (Pseudocode for Variational Inference)

1 for i in k
2 init $\phi_{ni}[0] = 1/k$ for n in N
3 init $\gamma_i = \alpha_i + N/k$
4 while convergence != 1
5 $\phi_{ni}[t+1] = \beta_{iw_n} exp(\psi(\gamma_i[t]))$
6 normalize $\phi_n[t+1]$
7 $\gamma[t+1] = \alpha + \sum_{n=1}^{N} \phi_n[t+1]$

Future Research

For the field:

- Use LDA to improve classification
- Follow patients longitudinally to extract more informative features
- Use other supervised classifiers (Perceptron, SVM, etc.) that might improve accuracy

For us:

- Med School, Grad School