Tackling the Poor Assumptions of Naive Bayes Text Classifiers

Jason Rennie
MIT Computer Science and Artificial Intelligence Laboratory
jrennie@ai.mit.edu

Joint work with Lawrence Shih, Jaime Teevan and David Karger.

Outline

- Naive Bayes Intro
- Why you should stay awake
- Improving Naive Bayes
  - Imbalanced Training Data
  - Weight Normalization
  - Log Transform
  - Inverse Document Frequency
  - Length Normalization
- Experimental Results

Naive Bayes

- Multinomial Naive Bayes assumes that words are drawn independently from a multinomial

$$p(\vec{x};\vec{\theta}) = \frac{(\|\vec{x}\|_1)!}{\prod_{k=1}^{d} x_k!} \prod_{k=1}^{d} (\theta_k)^{x_k} \tag{1}$$

- θ_k - chance of seeing word k
- x_k - number of times word k appears in document

Naive Bayes

- Estimate parameters separately for each label: $\hat{\theta}_{yk} = \frac{n_{yk}+1}{N_y+d}$
- Classify new document:

$$l(\vec{x}) = \arg\max_y p(y|\vec{x}) = \arg\max_y p(\vec{x}|y)p(y) \tag{2}$$

$$= \arg\max_y \sum_{k=1}^{d} \hat{w}_{yk} x_k + b_y \tag{3}$$

where $\hat{w}_{yk} = \log \hat{\theta}_{yk}$ and $b_y = \log p(y)$.

- Linear classifier form. Same as SVM, RLR, RLS, etc.
- NB is poor classifier because of choice of $\hat{w}_{yk}$ and b_y.

Why you should stay awake

- Naive Bayes is usually the "punching bag of classifiers"
- It's bad, but there are things you can do to fix it
- We give fixes for
  1. Learning better classification weights
  2. Modeling text better (transforming the data)
- End result is a fast classifier ($O(nd)$ time) that performs almost as well as the SVM (on text)

Better Weights: Imbalanced Training Data

- Problem: NB heavily favors classes with more training examples

- Real Problem:

$$E[\hat{w}_{yk}] = E[\log \hat{\theta}_{yk}] < \log E[\hat{\theta}_{yk}] = \log \theta_{yk} \tag{4}$$

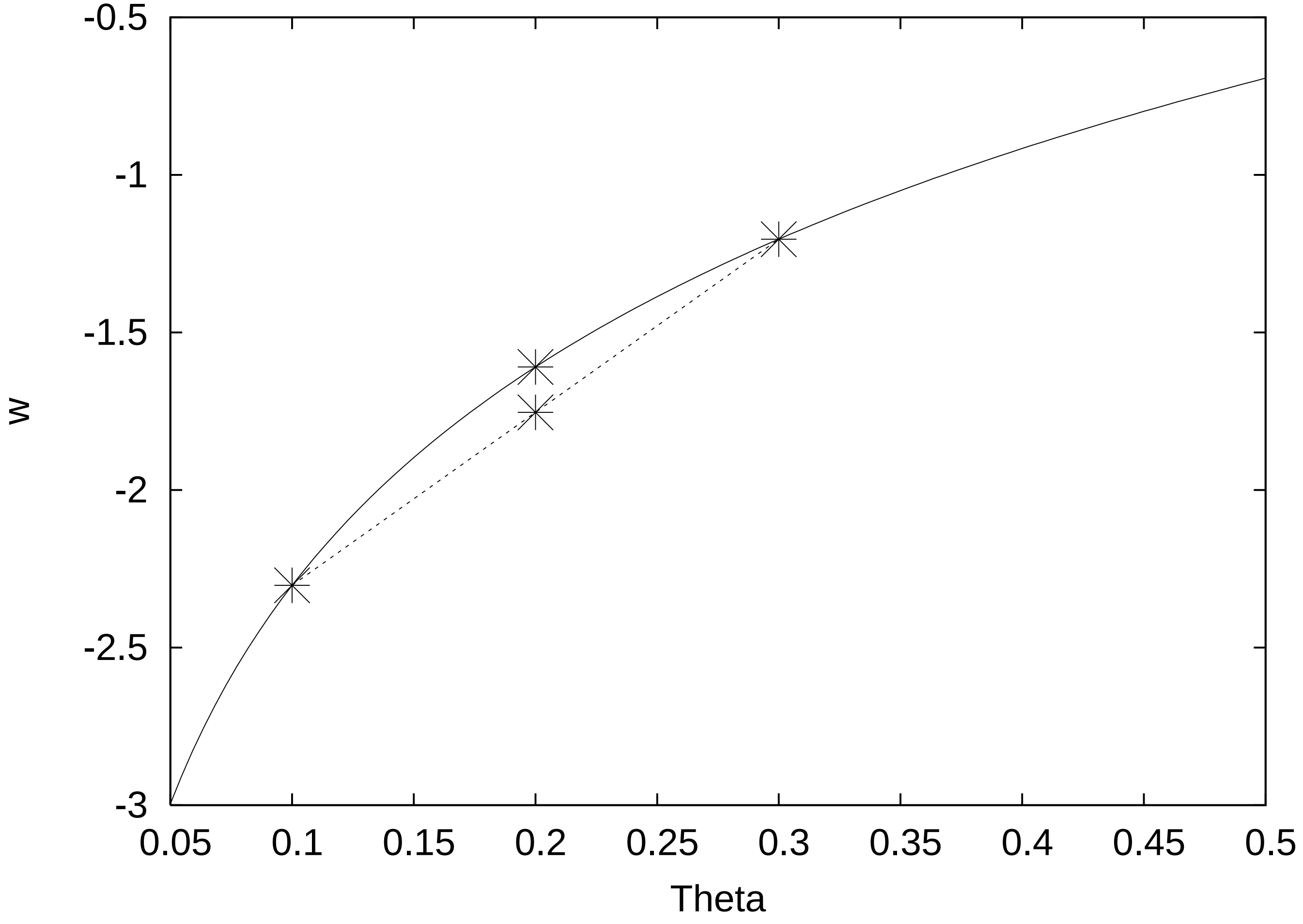
  Fewer samples $\Rightarrow$ smaller weights

- Solution (multiclass): Calculate score for class using statistics from all other classes; pick class with minimum score

A Closer Look: Imbalanced Training Data

- Let $\theta_{y1}, \ldots, \theta_{yd}$ be correct word probabilities for classes.
- Optimial classifier has weights $w_{yk} = \log \theta_{yk}$.
- $\hat{\theta}_{yk}$ is (nearly) unbiased estimate of θ_{yk}
- $\log \hat{\theta}_{yk}$ is *biased* estimate of w_{yk}

Jensen's Inequality



A Closer Look: Imbalanced Training Data

- Bias is $E[\log \hat{\theta}_{yk}] - \log \theta_{yk}$
- Bias is negative, so weights are (on average) smaller than they should be
- Magnitude of bias depends on number of samples
- Classes with fewer examples will have smaller weights

Complement Class Naive Bayes

- NB asks: Statistics of which class best fit this document?
- We ask: Statistics of which complement class least fit this document?
- *Complement class* is agglomeration of all other classes
  - One-vs-all compares class vs. complement class
- Why do this? More examples $\Rightarrow$ smaller bias.

class #	1	2	3	$\cdots$
regular	10	50	20	$\cdots$
complement	440	400	430	$\cdots$

Complement Class Naive Bayes

- Normally, (ignoring prior) we label according to

$$l(\vec{x}) = \arg\max_y p(\vec{x}|y) \tag{5}$$

- Complement class labels according to

$$l_{\mathrm{CC}}(\vec{x}) = \arg\min_y p(\vec{x}|\neg y) \tag{6}$$

- CC parameters are estimated using all documents not labeled y

One-vs-all

- [Zhang and Oles, 2001], [Berger, 1999] both have emprical evidence showing that OVA NB does better than NB.
- Here's why: OVA uses sum of normal NB and CC NB:

$$l_{\text{OVA}}(\vec{x}) = \arg\max_y p(\vec{x}|y) - p(\vec{x}|\neg y) \tag{7}$$

- How to do better:
  - One-label-per-document: just use complement part
  - Multi-label: compare against the all-document class:

$$l_{\text{OVACC}}(\vec{x}) = \arg\max_y p(\vec{x}) - p(\vec{x}|\neg y) \tag{8}$$

  $p(\vec{x})$ parameters learned using all documents

□

Better Weights: Normalization

- Problem: Word duplication can create larger weight vector in some classes
- For Example: Class 1 is "Boston," Class 2 is "San Francisco" "San" and "Francisco" are counted independently
- Solution: Normalize weight vector

$$w'_{yk} = \frac{w_{yk}}{\|\vec{w}_y\|_1} \tag{9}$$

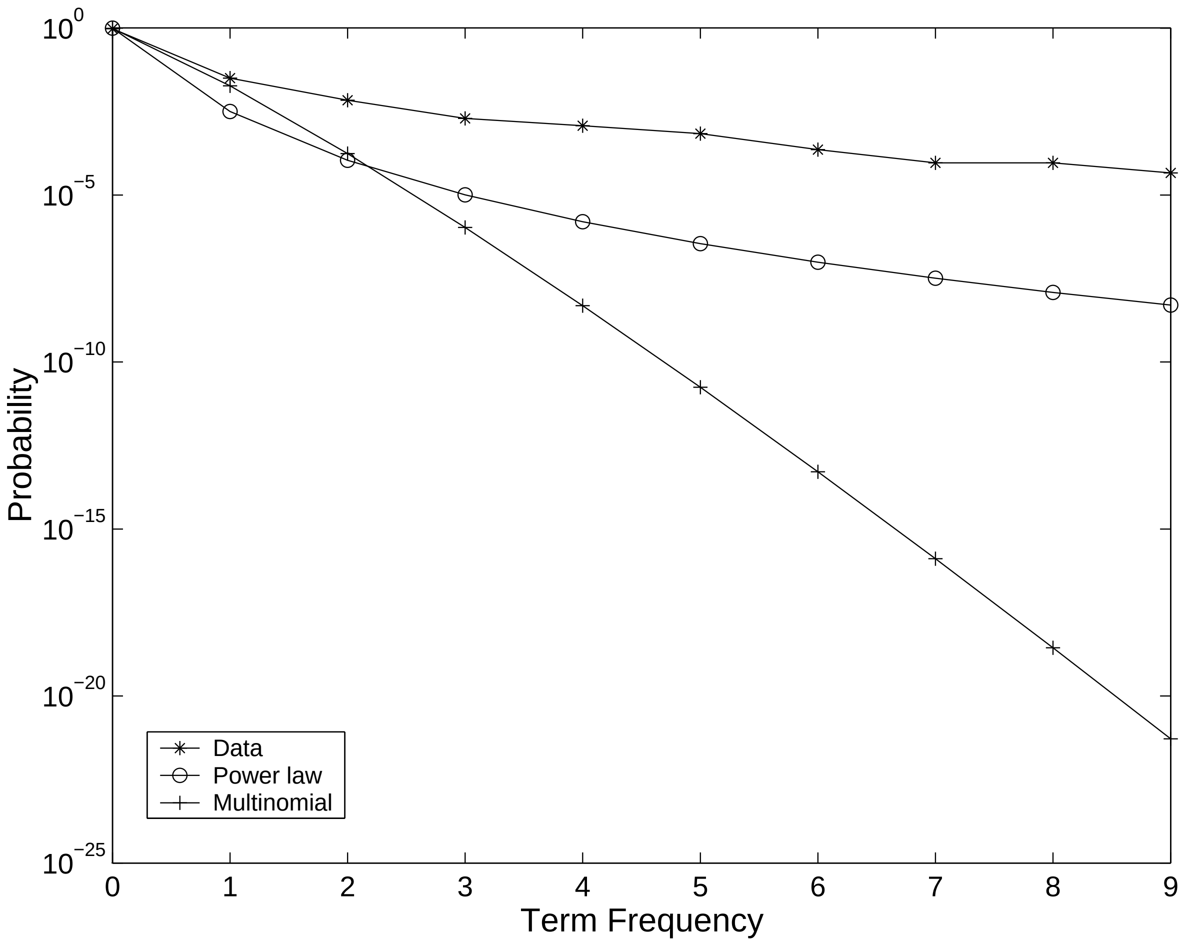
Better Data: Term Frequency Distribution

- Problem: Distribution of word frequency is heavy-tailed (a word is more likely once it has first appeared)
- Solution: Transform word counts by

$$x'_k = \log_2(1 + x_k) \tag{10}$$

  Maps $0 \rightarrow 0$, $1 \rightarrow 1$, downweights larger values
- $\log(x_k) + 1$ popular in IR community; $\log(1 + x_k)$ better motivated

A Closer Look: Frequency Distribution



A Closer Look: Frequency Distribution

- Multinomial estimate of word occuring many times are horribly wrong.
- Log transform: $p(x_k) \propto \theta_k^{\log(x_k+1)} = (x_k+1)^{\log \theta_k}$
- Heavy-tailed power law distribution: $p(x) \propto (x+a)^b$
- We get $a = 1$, $b = \log \theta$ via transform
- Bad: Parameters not optimized
- Good: BIG improvement over multinomial; learning is still linear time

□

Better Data: Inverse Document Frequency

- Problem: Words that appear in lots of documents are not useful
- Solution: Use inverse document frequency:

$$x'_k = x_k \log \frac{\sum_i 1}{\sum_i \delta_{ik}} \tag{11}$$

Better Data: Document Normalization

- Problem: Longer documents have more influence in training set
- Solution: Normalize document vectors (after other transforms)

$$x'_k = \frac{x_k}{\|\vec{x}\|_2} \tag{12}$$

Experiments

- Applying these solutions improves NB greatly

	MNB	TWCNB
Industry Sector	0.582	0.923
20 Newsgroups	0.848	0.861
Reuters (micro)	0.739	0.844
Reuters (macro)	0.270	0.647

- # training examples vary greatly by class for Industry Sector (10-105) & Reuters (1-3000)
- SVM results include IR transforms (improves performance)

Experiments

- Applying these solutions improves NB greatly

	MNB	TWCNB	SVM
Industry Sector	0.582	0.923	0.934
20 Newsgroups	0.848	0.861	0.862
Reuters (micro)	0.739	0.844	0.887
Reuters (macro)	0.270	0.647	0.694

- # training examples vary greatly by class for Industry Sector (10-105) & Reuters (1-3000)
- SVM results include IR transforms (improves performance)

The End

- Naive Bayes is a bad classifier
- We can fix most glaring problems with alterations and transforms
- Result is classifier that approaches state-of-the-art (on text), but runs in $O(nd)$ time

b_y (bonus slide)

- $\log p(y)$ would be good choice if $\sum_k w_{yk} x_k \approx \log p(y|x)$
- Independence assumption makes weights too large
- Solution (binary): choose $(b_+ - b_-)$ to minimize training error
- Approximate algorithms can be used for multiclass [Webb and Pazzani, 1998] (exact is exponential in # classes)

References

[Berger, 1999] Berger, A. (1999). Error-correcting output coding for text classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, Stockholm, Sweeden.

[Webb and Pazzani, 1998] Webb, G. I. and Pazzani, M. J. (1998). Adjusted probability naive bayesian induction. In *Australian Joint Conference on Artificial Intelligence*, pages 285–295.

[Zhang and Oles, 2001] Zhang, T. and Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31.