SFU NatLangLab

Natural Language Processing

Angel Xuan Chang
angelxuanchang.github.io/nlp-class
adapted from lecture slides from Anoop Sarkar

Simon Fraser University

2020-01-23

Natural Language Processing

Angel Xuan Chang
angelxuanchang.github.io/nlp-class
adapted from lecture slides from Anoop Sarkar

Simon Fraser University

January 23, 2020

Part 1: Classification tasks in NLP

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Sentiment classification: Movie reviews

- neg unbelievably disappointing
- pos Full of zany characters and richly applied satire, and some great plot twists
- pos this is the greatest screwball comedy ever filmed
- neg It was pathetic. The worst part about it was the boxing scenes.

Intent Detection

- ADDR_CHANGE I just moved and want to change my address.
- ADDR_CHANGE Please help me update my address.
- FILE_CLAIM I just got into a terrible accident and I want to file a claim.
- CLOSE_ACCOUNT I'm moving and I want to disconnect my service.

Prepositional Phrases

- noun attach: *I bought* **the shirt with pockets**
- verb attach: *I* **bought** *the shirt* **with my credit card**
- noun attach: *I washed* **the shirt with mud**
- verb attach: *I* **washed** *the shirt* **with soap**
- Attachment depends on the meaning of the entire sentence – needs world knowledge, etc.
- Maybe there is a simpler solution: we can attempt to solve it using heuristics or associations between words

Ambiguity Resolution: Prepositional Phrases in English

- Learning Prepositional Phrase Attachment: Annotated Data

v	n_1	p	n_2	Attachment
join	board	as	director	V
is	chairman	of	N.V.	N
using	crocidolite	in	filters	V
bring	attention	to	problem	V
is	asbestos	in	products	N
making	paper	for	filters	N
including	three	with	cancer	N
⋮	⋮	⋮	⋮	⋮

Prepositional Phrase Attachment

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2

Back-off Smoothing

- Random variable a represents attachment.
- $a = n_1$ or $a = v$ (two-class classification)
- We want to compute probability of noun attachment: $p(a = n_1 \mid v, n_1, p, n_2)$.
- Probability of verb attachment is $1 - p(a = n_1 \mid v, n_1, p, n_2)$.

Back-off Smoothing

1. If $f(v, n_1, p, n_2) > 0$ and $\hat{p} \neq 0.5$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, n_1, p, n_2)}{f(v, n_1, p, n_2)}$$

2. Else if $f(v, n_1, p) + f(v, p, n_2) + f(n_1, p, n_2) > 0$ and $\hat{p} \neq 0.5$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, n_1, p) + f(a_{n_1}, v, p, n_2) + f(a_{n_1}, n_1, p, n_2)}{f(v, n_1, p) + f(v, p, n_2) + f(n_1, p, n_2)}$$

3. Else if $f(v, p) + f(n_1, p) + f(p, n_2) > 0$

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, v, p) + f(a_{n_1}, n_1, p) + f(a_{n_1}, p, n_2)}{f(v, p) + f(n_1, p) + f(p, n_2)}$$

4. Else if $f(p) > 0$ (try choosing attachment based on preposition alone)

$$\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = \frac{f(a_{n_1}, p)}{f(p)}$$

5. Else $\hat{p}(a_{n_1} \mid v, n_1, p, n_2) = 1.0$

Prepositional Phrase Attachment: Results

- **Results (Collins and Brooks 1995)**: 84.5% accuracy with the use of some limited word classes for dates, numbers, etc.
- **Toutanova, Manning, and Ng, 2004**:
  use sophisticated smoothing model for PP attachment
  86.18% with words & stems; with word classes: 87.54%
- **Merlo, Crocker and Berthouzoz, 1997**:
  test on multiple PPs, generalize disambiguation of 1 PP to 2-3 PPs
  1PP: 84.3%   2PP: 69.6%   3PP: 43.6%

Natural Language Processing

Angel Xuan Chang
angelxuanchang.github.io/nlp-class
adapted from lecture slides from
Anoop Sarkar, Danqi Chen and Karthik Narasimhan

Simon Fraser University

January 23, 2020

Part 2: Probabilistic Classifiers

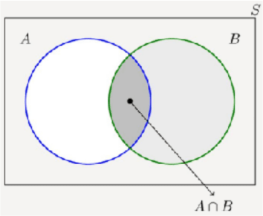
Classification Task

- Input:
  - A document d
  - a set of classes $C = \{c_1, c_2, \ldots, c_m\}$
- Output: Predicted class c for document d
- Example:
  - neg unbelievably disappointing
  - pos this is the greatest screwball comedy ever filmed

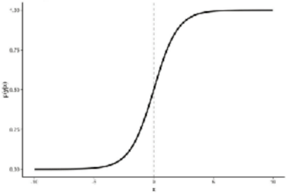
Supervised learning: Let's use statistics!

- Inputs:
  - Set of m classes $C = \{c_1, c_2, \ldots, c_m\}$
  - Set of n *labeled* documents: $\{(d_1, c_1), (d_2, c_2), \ldots, (d_n, c_n)\}$
- Output: Trained classifier $F : d \rightarrow c$
  - What form should F take?
  - How to learn F?

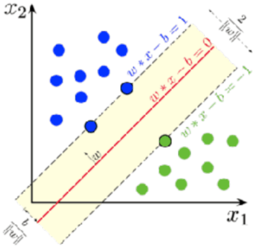
Types of supervised classifiers



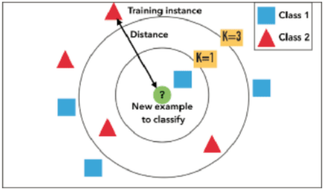
Naive Bayes



Logistic regression



Support vector machines



k-nearest neighbors

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Naive Bayes Classifier

- $\mathbf{x}$ is the input that can be represented as d independent features f_j, $1 \leq j \leq d$
- y is the output classification
- $P(y \mid \mathbf{x}) = \frac{P(y) \cdot P(\mathbf{x}|y)}{P(\mathbf{x})}$ (Bayes Rule)
- $P(\mathbf{x} \mid y) = \prod_{j=1}^{d} P(f_j \mid y)$
- $P(y \mid \mathbf{x}) \propto P(y) \cdot \prod_{j=1}^{d} P(f_j \mid y)$
- We can ignore $P(\mathbf{x})$ in the above equation because it is a constant scaling factor for each y.

Naive Bayes Classifier for text classification

- For text classificaiton: input $\mathbf{x} = document\mathbf{d} = (w_1, \ldots, w_k)$,
- Use as our features the words w_j, $1 \leq j \leq |V|$ where V is our vocabulary
- c is the output classification
- Assume that position of each word is irrelevant and that the words are conditionally independent given class c

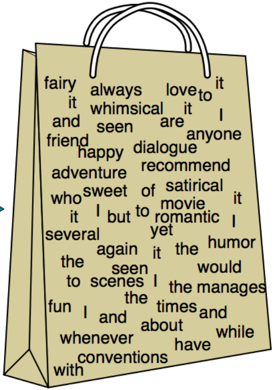
$$P(w_1, w_2, \ldots, w_k|c) = P(w_1|c)P(w_2|c)\ldots P(w_k|c)$$

- Maximum a posteriori estimate

$$c_{\text{MAP}} = \arg\max_c P(c)P(d|c) = \arg\max_c \hat{P}(c)\prod_{i=1}^{k}\hat{P}(w_i|c)$$

Bag of words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Estimating probabilities

Maximum likelihood estimate

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

$$\hat{P}(w_i|c_j) = \frac{\text{Count}(w_i, c_j)}{\sum_{w \in V}[\text{Count}(w, c_j)]}$$

Smoothing

$$\hat{P}(w_i|c) = \frac{\text{Count}(w_i, c) + \alpha}{\sum_{w \in V}[\text{Count}(w, c_j) + \alpha]}$$

Overall process

Input: Set of labeled documents: $\{(d_i, c_i)\}_{i=1}^{n}$

- Compute vocabulary V of all words
- Calculate

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

- Calculate

$$\hat{P}(w_i|c_j) = \frac{\text{Count}(w_i, c_j) + \alpha}{\sum_{w \in V}[\text{Count}(w, c_j) + \alpha]}$$

- Prediction: Given document $d = (w_1, \ldots, w_k)$

$$c_{\text{MAP}} = \arg\max_c \hat{P}(c) \prod_{i=1}^{k} \hat{P}(w_i|c)$$

Naive Bayes Example

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

Choosing a class:

P(c|d5) $\propto$ 3/4 * $(3/7)^3$ * 1/14 * 1/14 $\approx$ 0.0003

P(j|d5) $\propto$ 1/4 * $(2/9)^3$ * 2/9 * 2/9 $\approx$ 0.0001

Tokenization

Tokenization matters - it can affect your vocabulary

- *aren't* $\boxed{\text{aren't}}$
  $\boxed{\text{arent}}$
  $\boxed{\text{are}}$ $\boxed{\text{n't}}$
  $\boxed{\text{aren}}$ $\boxed{\text{t}}$
- Emails, URLs, phone numbers, dates, emoticons

Features

- Remember: Naive Bayes can use any set of features
- Captitalization, subword features (end with -ing), etc
- Domain knowledge crucial for performance

Top features for spam detection

Rank	Category	Feature	Rank	Category	Feature
1	Subject	Number of capitalized words	1	Subject	Min of the compression ratio for the bz2 compressor
2	Subject	Sum of all the character lengths of words	2	Subject	Min of the compression ratio for the zlib compressor
3	Subject	Number of words containing letters and numbers	3	Subject	Min of character diversity of each word
4	Subject	Max of ratio of digit characters to all characters of each word	4	Subject	Min of the compression ratio for the lzw compressor
5	Header	Hour of day when email was sent	5	Subject	Max of the character lengths of words
		(a)			(b)
		Spam URLs Features			
1	URL	The number of all URLs in an email	1	Header	Day of week when email was sent
2	URL	The number of unique URLs in an email	2	Payload	Number of characters
3	Payload	Number of words containing letters and numbers	3	Payload	Sum of all the character lengths of words
4	Payload	Min of the compression ratio for the bz2 compressor	4	Header	Minute of hour when email was sent
5	Payload	Number of words containing only letters	5	Header	Hour of day when email was sent
		(c)			(d)

[Alqatawna et al, IJCNSS 2015]

Evaluation

- Table of prediction (binary classification)

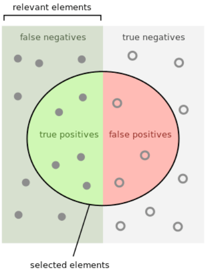
		Truth: Positive	Truth: Negative
Predicted	Positive	100	5
	Negative	45	100

- Ideally we want to get

	Positive	Negative
Positive	145	0
Negative	0	105

Evaluation Metrics

		Truth	
		Positive	**Negative**
Predicted	**Positive**	100	5
	Negative	45	100

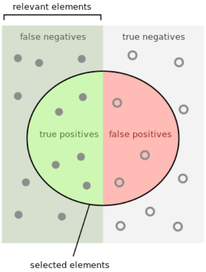


$$\textbf{Accuracy} = \frac{TP + TN}{Total} = \frac{200}{250} = 80\%$$

Evaluation Metrics

		Truth	
		Positive	**Negative**
Predicted	**Positive**	100	5
	Negative	45	100

	Positive	**Negative**
Positive	100	25
Negative	25	100



$$\textbf{Accuracy} = \frac{TP + TN}{Total} = \frac{200}{250} = 80\%$$

Precision and Recall

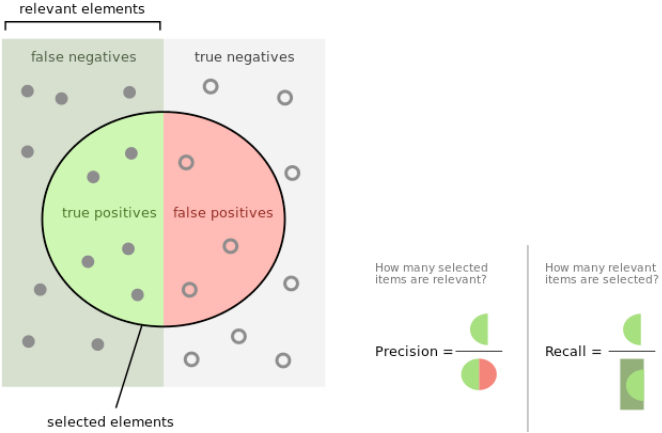
- Precision: % of selected classes that are correct

$$\textbf{Precision}(+) = \frac{TP}{TP + FP} \qquad \textbf{Precision}(-) = \frac{TN}{TN + FN}$$

- Recall: % of correct items selected

$$\textbf{Recall}(+) = \frac{TP}{TP + FN} \qquad \textbf{Recall}(-) = \frac{TN}{TN + FP}$$

Precision and Recall



from Wikipedia

F-Score

- Combined measure

- Harmonic mean of Precision and Recall

$$F_1 = \frac{2 \cdot \textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

- Or more generally,

$$F_\beta = \frac{(1+\beta^2) \cdot \textbf{Precision} \cdot \textbf{Recall}}{\beta^2 \cdot \textbf{Precision} + \textbf{Recall}}$$

Choosing Beta

		Truth	
		Positive	**Negative**
Predicted	**Positive**	200	100
	Negative	50	100

$$F_\beta = \frac{(1+\beta^2)\cdot \textbf{Precision} \cdot \textbf{Recall}}{\beta^2 \cdot \textbf{Precision} + \textbf{Recall}}$$

- Which value of Beta maximizes F_β for positive class?

A. $\beta = 0.5$

B. $\beta = 1$

C. $\beta = 2$

Aggregating scores

- We have Precision, Recall, F1 for each class
- How to combine them for an overall score?
  - Macro-average: Compute for each class, then average
  - Micro-average: Collect predictions for all classes and jointly evaluate

Macro vs Micro average

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

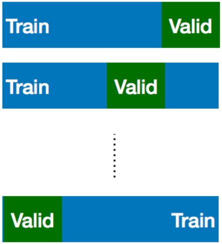
Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: $(0.5 + 0.9)/2 = 0.7$
- Microaveraged precision: $100/120 = .83$
- Microaveraged score is dominated by score on common classes

Validation

- Choose a metric: Precision/Recall/F1
- Optimize for metric on Validation (aka Development) set
- Finally evaluate on ‘unseen’ Test set
- Cross-validation
  - Repeatedly sample several train-val splits
  - Reduces bias due to sampling errors



Advantanges of Naive Bayes

- Very fast, low storage requirements
- Robust to irrelevant features
- Very good in domains with many equally important features
- Optimal if the independence assumptions hold
- Good dependable baseline for text classification

When to use Naive Bayes

- Small data sizes: Naive Bayes is great! .
  Rule-based classifiers can work well too
- Medium size datasets: More advanced classifiers might perform better (SVM, logistic regression)
- Large datasets: Naive Bayes becomes competive again (most learned classifiers will work well)

Failings of Naive Bayes (1)

Independence assumptions are too strong

- XOR problem: Naive Bayes cannot learn a decision boundary

x1	x2	Class: x_1 XOR x_2
1	1	0
0	1	1
1	0	1
0	0	0

- Both variables are jointly required to predict class. Independence assumption broken!

Failings of Naive Bayes (2)

Class Imbalance

- One or more classes have more instances than others
- Data skew causes NB to prefer one class over the other

Failings of Naive Bayes (3)

Weight magnitude errors

- Classes with larger weights are preferred
- 10 documents with class=MA and "Boston" occurring once each
- 10 documents with class=CA and "San Francisco" occurring once each
- New document d: "Boston Boston Boston San Francisco San Francisco"

$$P(\text{class} = \text{CA}|d) > P(\text{class} = \text{MA}|d)$$

Naive Bayes Summary

- Domain knowledge is crucial to selecting good features
- Handle class imbalance by re-weighting classes
- Use log scale operations instead of multiplying probabilities

$$P(c_{NB}) = \arg\max_{c_j \in C} \log P(c_j) + \sum_i \log P(x_i | c_j)$$

- Model is now just max of sum of weights

Classification tasks in NLP

Naive Bayes Classifier

Log linear models

Log linear model

- The model classifies input into output labels $y \in \mathcal{Y}$
- Let there be m features, $f_k(\mathbf{x}, y)$ for $k = 1, \ldots, m$
- Define a parameter vector $\mathbf{v} \in \mathbb{R}^m$
- Each $(\mathbf{x}, y)$ pair is mapped to score:

$$s(\mathbf{x}, y) = \sum_k v_k \cdot f_k(\mathbf{x}, y)$$

- Using inner product notation:

$$\begin{aligned} \mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y) &= \sum_k v_k \cdot f_k(\mathbf{x}, y) \\ s(\mathbf{x}, y) &= \mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y) \end{aligned}$$

- To get a probability from the score: Renormalize!

$$\Pr(y \mid \mathbf{x}; \mathbf{v}) = \frac{exp\left(s(\mathbf{x}, y)\right)}{\sum_{y' \in \mathcal{Y}} exp\left(s(\mathbf{x}, y')\right)}$$

Log linear model

- The name 'log-linear model' comes from:

$$\log \Pr(y \mid \mathbf{x}; \mathbf{v}) = \underbrace{\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y)}_{\text{linear term}} - \underbrace{\log \sum_{y'} exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}, y')\right)}_{\text{normalization term}}$$

- Once the weights $\mathbf{v}$ are learned, we can perform predictions using these features.
- The goal: to find $\mathbf{v}$ that maximizes the log likelihood $L(\mathbf{v})$ of the labeled training set containing $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots n$

$$\begin{aligned} L(\mathbf{v}) &= \sum_i \log \Pr(y_i \mid \mathbf{x}_i; \mathbf{v}) \\ &= \sum_i \mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right) \end{aligned}$$

Log linear model

- Maximize:

$$L(\mathbf{v}) = \sum_i \mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \log \sum_{y'} exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

- Calculate gradient:

$$\left.\frac{dL(\mathbf{v})}{d\mathbf{v}}\right|_{\mathbf{v}}$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \frac{1}{\sum_{y''} exp\left(\mathbf{v} \cdot \mathbf{f}(x_i, y'')\right)} \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \cdot exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)$$

$$= \sum_i \mathbf{f}(\mathbf{x}_i, y_i) - \sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \frac{exp\left(\mathbf{v} \cdot \mathbf{f}(\mathbf{x}_i, y')\right)}{\sum_{y''} exp\left(\mathbf{v} \cdot \mathbf{f}(x_i, y'')\right)}$$

$$= \underbrace{\sum_i \mathbf{f}(\mathbf{x}_i, y_i)}_{\text{Observed counts}} - \underbrace{\sum_i \sum_{y'} \mathbf{f}(\mathbf{x}_i, y') \Pr(y' \mid \mathbf{x}_i; \mathbf{v})}_{\text{Expected counts}}$$

Gradient ascent

- Init: $\mathbf{v}^{(0)} = \mathbf{0}$
- $t \leftarrow 0$
- Iterate until convergence:
  - Calculate: $\Delta = \left.\frac{dL(\mathbf{v})}{d\mathbf{v}}\right|_{\mathbf{v}=\mathbf{v}^{(t)}}$
  - Find $\beta^* = \arg\max_\beta L(\mathbf{v}^{(t)} + \beta\Delta)$
  - Set $\mathbf{v}^{(t+1)} \leftarrow \mathbf{v}^{(t)} + \beta^*\Delta$

Learning the weights: $\mathbf{v}$: Generalized Iterative Scaling

$f^{\#} = max_{x,y} \sum_j f_j(x,y)$
(the maximum possible feature value; needed for scaling)
Initialize $\mathbf{v}^{(0)}$
For each iteration t

- `expected[j]` $\leftarrow$ 0 for j = 1 .. # of features
- For i = 1 to $|$ training data $|$
  - For each feature f_j
    - `expected[j]` $\mathrel{+}= f_j(x_i, y_i) \cdot P(y_i \mid x_i; \mathbf{v}^{(t)})$
- For each feature $f_j(x,y)$
  - `observed[j]` $= f_j(x,y) \cdot \frac{c(x,y)}{|\text{training data}|}$
- For each feature $f_j(x,y)$
  - $v_j^{(t+1)} \leftarrow v_j^{(t)} \cdot \sqrt[f^{\#}]{\frac{\texttt{observed[j]}}{\texttt{expected[j]}}}$

cf. Goodman, NIPS '01

Acknowledgements

Many slides borrowed or inspired from lecture notes by Anoop Sarkar, Danqi Chen, Karthik Narasimhan, Dan Jurafsky, Michael Collins, Chris Dyer, Kevin Knight, Chris Manning, Philipp Koehn, Adam Lopez, Graham Neubig, Richard Socher and Luke Zettlemoyer from their NLP course materials.

All mistakes are my own.

A big thank you to all the students who read through these notes and helped me improve them.