Lecture 4
Text Classification with Naive Bayes

Ivan Titov
(with slides from Sharon Goldwater)

School of informatics

Today's lecture

- What are some examples of text classification tasks?
- What is a Naive Bayes classifier and how do we apply it to text classification (in general, or for specific tasks)?
- What are some pros and cons of Naive Bayes?
- How do we evaluate classification accuracy?

Text classification

We might want to classify the *content* of the text:

- Spam detection (binary classification: spam/not spam)
- Sentiment analysis (binary or multiway)
  - movie, restaurant, product reviews (pos/neg, or 1-5 stars)
  - political argument (pro/con, or pro/con/neutral)
- Topic classification (multiway: sport/finance/travel/etc)

Text classification

Or we might want to classify the *author* of the text (**authorship attribution**):

- Native language identification (e.g., to tailor language tutoring)
- Diagnosis of disease (psychiatric or cognitive impairments)
- Identification of gender, dialect, educational background (e.g., in forensics [legal matters], advertising/marketing).

Formalizing the task

- Given document x and set of categories Y, we want to assign x to the most probable category $\hat{y}$:

$$
\begin{aligned}
\hat{y} &= \operatorname*{argmax}_{y \in Y} p(y|x) \\
&= \operatorname*{argmax}_{y \in Y} \frac{p(x|y)p(y)}{p(x)} \\
&= \operatorname*{argmax}_{y \in Y} p(x|y)p(y)
\end{aligned}
$$

Document model

Each document x is represented by **features** $f_1, f_2, \dots f_n$, e.g.:

- For topic classification: 2000 most frequent words, excluding **stopwords** like *the, a, do, in*.
- For sentiment classification: words from a **sentiment lexicon**

In fact, we only care about the feature *counts*, so this is a **bag-of-words** (unigram) model.

Task-specific features

Example words from a **sentiment lexicon**:

Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
...		...

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		...

From `http://www.enchantedlearning.com/wordlist/`

Example documents

- Possible feature counts from training documents in a spam-detection task (where we did not exclude stopwords):

	the	your	model	cash	Bitcoin	class	account	orderz
doc 1	12	3	1	0	0	2	0	0
doc 2	10	4	0	4	0	0	2	0
doc 3	25	4	0	0	0	1	1	0
doc 4	14	2	0	1	3	0	1	1
doc 5	17	5	0	2	0	0	1	1

Document model, cont.

- Representing x using its features gives us:

$$p(x|y) = P(f_1, f_2, \ldots f_n|y)$$

- But we can't estimate this joint probability well (too sparse).
- So, make a **Naive Bayes** assumption: features are conditionally independent given class.

$$p(x|y) \approx P(f_1|y)p(f_2|y) \ldots p(f_n|y)$$

Full model

- Given document with features $f_1, f_2, \ldots f_n$ and set of categories Y, choose

$$\hat{y} = \operatorname*{argmax}_{y \in Y} p(y) \prod_{i=1}^{n} p(f_i|y)$$

- This is called a **Naive Bayes classifier**

Generative process

- Naive Bayes classifier is a generative model.
- Assumes the data (features in each doc) were generated as
  - For each document, choose its class y with prob $p(y)$.
  - For each feature in each doc, choose the value of that feature with prob $p(f|y)$

Learning the class priors

- $p(y)$ normally estimated with MLE:

$$\hat{p}(y) = \frac{|D_y|}{|D|}$$

  - $|D_y|$ = the number of training documents in class y
  - $|D|$ = the total number of training documents

Learning the class priors: example

- Given training documents with correct labels:

	the	your	model	cash	Bitcoin	class	account	orderz	spam?
doc 1	12	3	1	0	0	2	0	0	-
doc 2	10	4	0	4	0	0	2	0	+
doc 3	25	4	0	0	0	1	1	0	-
doc 4	14	2	0	1	3	0	1	1	+
doc 5	17	5	0	2	0	0	1	1	+

- $\hat{p}(\text{spam}) = 3/5$

Problem: zero counts

As a first attempt, maybe estimate the likelihood for feature f_i as:

$$\hat{p}(f_i|y) = \frac{\text{count}(f_i, y)}{\sum_{f \in F}(\text{count}(f, y))}$$

- $\text{count}(f_i, y)$ = the number of times f_i occurs in class y
- Some features may receive 0 counts $\rightarrow \hat{p}(f_i|y) = 0 \rightarrow p(x|y) = 0$
- Aggravated by the sparsity of linguistic data (cf Zipf's law)

Learning the feature probabilities

- $p(f_i|y)$ normally estimated with simple α-smoothing:

$$\hat{p}(f_i|y) = \frac{\text{count}(f_i, y) + \alpha}{\sum_{f \in F}(\text{count}(f, y) + \alpha)}$$

  - $\text{count}(f_i, y)$ = the number of times f_i occurs in class y
  - F = the set of possible features
  - α: the smoothing parameter, optimised on held-out data

Learning the feature probabilities: example

	the	your	model	cash	Bitcoin	class	account	orderz	spam?
doc 1	12	3	1	0	0	2	0	0	-
doc 2	10	4	0	4	0	0	2	0	+
doc 3	25	4	0	0	0	1	1	0	-
doc 4	14	2	0	1	3	0	1	1	+
doc 5	17	5	0	2	0	0	1	1	+

Learning the feature probabilities: example

	the	your	model	cash	Bitcoin	class	account	orderz	spam?
doc 1	12	3	1	0	0	2	0	0	-
doc 2	10	**4**	0	4	0	0	2	0	+
doc 3	25	4	0	0	0	1	1	0	-
doc 4	14	**2**	0	1	3	0	1	1	+
doc 5	17	**5**	0	2	0	0	1	1	+

$$\hat{p}(\text{your}|+) = \frac{(4+2+5+\alpha)}{(\text{tokens in } + \text{ class})+\alpha|F|} = (11+\alpha)/(68+\alpha|F|)$$

Learning the feature probabilities: example

	the	your	model	cash	Bitcoin	class	account	orderz	spam?
doc 1	12	3	1	0	0	2	0	0	-
doc 2	10	4	0	4	0	0	2	0	+
doc 3	25	4	0	0	0	1	1	0	-
doc 4	14	2	0	1	3	0	1	1	+
doc 5	17	5	0	2	0	0	1	1	+

$$\hat{p}(\text{your}|+) = \frac{(4+2+5+\alpha)}{(\text{tokens in } + \text{ class})+\alpha|F|} = (11+\alpha)/(68+\alpha|F|)$$

$$\hat{p}(\text{your}|-) = \frac{(3+4+\alpha)}{(\text{tokens in } - \text{ class})+\alpha|F|} = (7+\alpha)/(49+\alpha|F|)$$

$$\hat{p}(\text{orderz}|+) = \frac{(2+\alpha)}{(\text{tokens in } + \text{ class})+\alpha|F|} = (2+\alpha)/(68+\alpha|F|)$$

Classifying a test document: example

- Test document x:

  get your cash and your orderz

- Suppose there are no other features besides those in previous table (so get and and are not counted). Then

$$
\begin{aligned}
p(+|d) &\propto p(+)\prod_{i=1}^{n} p(f_i|+) \\
&= \frac{3}{5}\cdot\frac{11+\alpha}{(68+\alpha|F|)}\cdot\frac{7+\alpha}{(68+\alpha|F|)} \\
&\quad \cdot\frac{11+\alpha}{(68+\alpha|F|)}\cdot\frac{2+\alpha}{(68+\alpha|F|)}
\end{aligned}
$$

Classifying a test document: example

- Test document x:

$$\text{get your cash and your orderz}$$

- Do the same for $p(-|x)$

- Choose the one with the larger value

Alternative feature values and feature sets

- Use only **binary** values for f_i: did this word occur in x or not?

- Use only a subset of the vocabulary for F
  - Ignore **stopwords** (function words and others with little content)
  - Choose a small task-relevant set (e.g., using a sentiment lexicon)

- Use more complex features (bigrams, syntactic features, morphological features, ...)

Task-specific features

Example words from a **sentiment lexicon**:

Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
...		...

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		...

From `http://www.enchantedlearning.com/wordlist/`

Task-specific features

- But: other words might be relevant for specific sentiment analysis tasks.
  - E.g., quiet, memory for product reviews.
- And for other tasks, stopwords might be very useful features
  - E.g., People with schizophrenia use more 2nd-person pronouns (**?**), those with depression use more 1st-person (**?**).
- Probably better to use too many irrelevant features than not enough relevant ones.

Advantages of Naive Bayes

- Very easy to implement
- Very fast to train and test
- Doesn't require as much training data as some other methods
- Usually works reasonably well

Use as a simple baseline for any classification task.

Problems with Naive Bayes

- Naive Bayes assumption is naive!

- Consider categories TRAVEL, FINANCE, SPORT.

- Are the following features independent given the category?

  beach, sun, ski, snow, pitch, palm, football, relax, ocean

Problems with Naive Bayes

- Naive Bayes assumption is naive!
- Consider categories TRAVEL, FINANCE, SPORT.
- Are the following features independent given the category?

  beach, sun, ski, snow, pitch, palm, football, relax, ocean

- No! They might be closer if we defined finer-grained categories (beach vacations vs. ski vacations), but we don't usually want to.

Non-independent features

- Features are not usually independent given the class
- Adding multiple feature types (e.g., words and morphemes) often leads to even stronger correlations between features
- Accuracy of classifier can sometimes still be ok, but it will be highly **overconfident** in its decisions.
  - Ex: NB sees 5 features that all point to class 1, treats them as five independent sources of evidence.
  - Like asking 5 friends for an opinion when some got theirs from each other.

How to evaluate performance?

- Important question for any NLP task
- **Intrinsic** evaluation: design a measure inherent to the task
  - Language modelling: perplexity
  - classification: F-score (coming up next)

How to evaluate performance?

- Important question for any NLP task
- **Intrinsic** evaluation: design a measure inherent to the task
  - Language modelling: perplexity
  - classification: F-score (coming up next)
- **Extrinsic** evaluation: measure effects on a downstream task
  - Language modelling: does it improve my QA/MT system?
  - classification: does it reduce user search time in an IR setting?

Intrinsic evaluation for classification

- Classes may be very unbalanced.

- Example: classification as detection. Is the document about sport or not?

- We can get 95% accuracy by always choosing "not"; but this isn't useful.

- We need a better measure.

Two measures

- Assume we have a **gold standard**: correct labels for test set
- We also have a system for detecting the items of interest (docs about sport)

$$\text{Precision} = \frac{\#\text{ items detected and was right}}{\#\text{ items system detected}}$$

$$\text{Recall} = \frac{\#\text{ items detected and was right}}{\#\text{ items system should have detected}}$$

Example of precision and recall

	Doc about sports?										
Gold standard	**Y**	**Y**	N	N	**Y**	N	N	N	**Y**	N	N
System output	N	**Y**	N	**Y**	N	N	N	N	**Y**	N	N

- # 'Y' we got right = 2
- # 'Y' we guessed = 3
- # 'Y' in GS = 4

- Precision = 2/3
- Recall = 2/4

Why use both measures?

Systems often have (implicit or explicit) tuning thresholds on how many answers to return.

- e.g., Return as **Y** all docs where system thinks p(C=sport) is greater than t.
- Raise t: higher precision, lower recall.
- Lower t: lower precision, higher recall.

Doc	Sys prob	Gold
23	0.99	**Y**
12	0.98	**Y**
45	0.93	**Y**
01	0.93	**Y**
37	0.89	N
24	0.84	**Y**
16	0.78	**Y**
18	0.75	N
20	0.72	**Y**
...	...	...
38	0.03	N
19	0.03	N

Precision-Recall curves

- If system has a tunable parameter to vary the precision/recall:

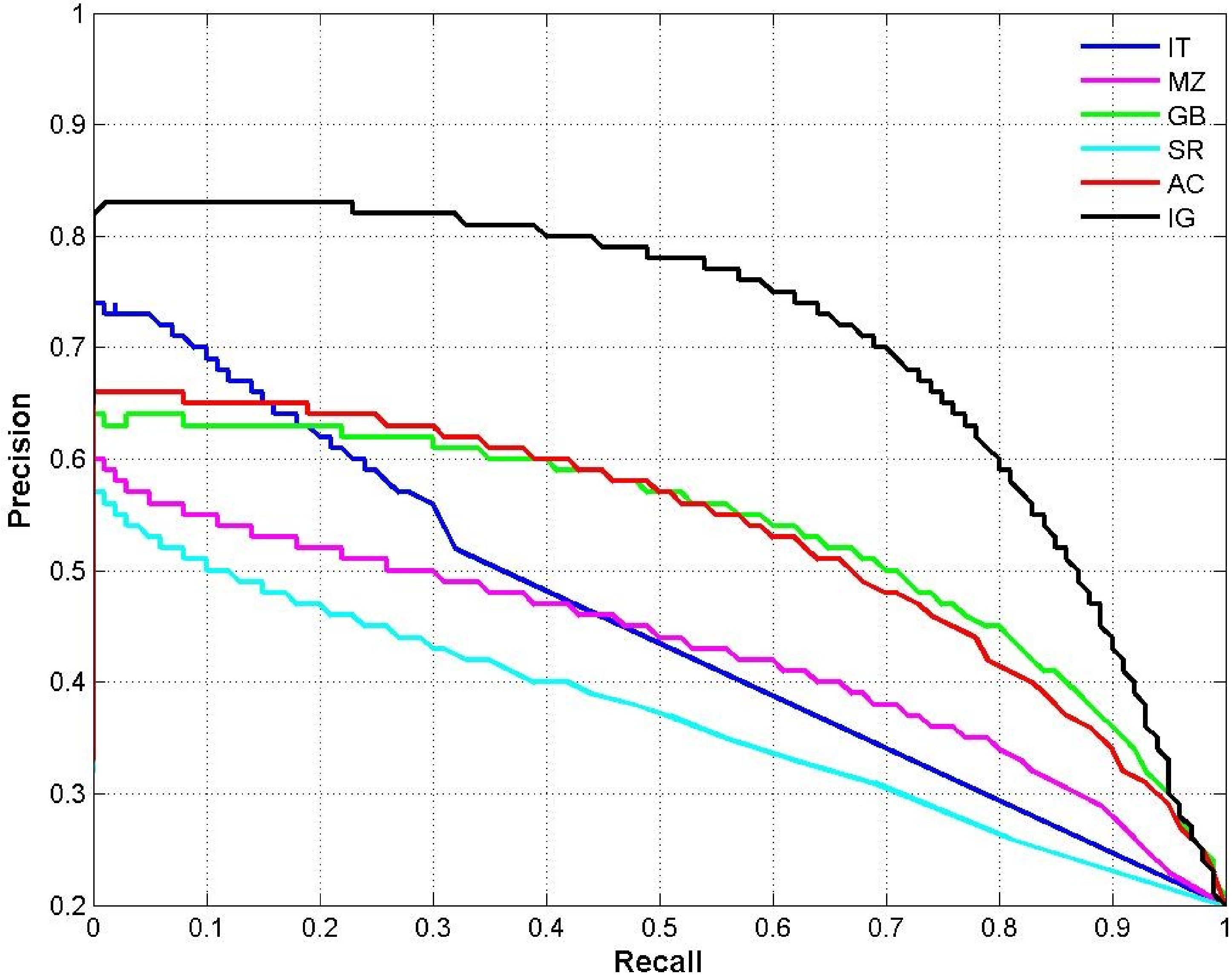

Figure from: http://ivrgwww.epfl.ch/supplementary_material/RK_CVPR09/

F-measure

- Can also combine precision and recall into single **F-measure**:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- Normally we just set $\beta = 1$ to get F_1:

$$F_1 = \frac{2PR}{P + R}$$

- F_1 is the harmonic mean of P and R: similar to arithmetic mean when P and R are close, but penalizes large differences between P and R.

Key Takeaways

- Text classification is common in NLP.
- Naive Bayes is a simple, general model.
- Its strong assumptions affect accuracy and confidence.
- Next lecture: building a stronger model, logistic regression.
- Evaluation: precision vs. recall trade-off.