Text Classification and Naïve Bayes

Text Classification

- **Text Classification** (**Text Categorization**) is ***the task of assigning a label or categorization category to an entire text or document.***

- Some of common text categorization tasks are:
  - Sentiment analysis
    - extraction of **sentiment**, positive or negative orientation that writer expresses toward an object.
  - Spam detection
    - binary classification task of assigning an email to one of the two classes spam or not-spam.
  - Authorship identification
    - determining a text's author.
  - Age/gender identification
    - determining a text's author characteristics like gender and age.
  - Language Identification
    - finding the language of a text.

Text Classification: Definition

- Text classification can be defined as follows:

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$

Output:

- a predicted class $c \in C$

Classification Methods: Hand-Coded Rules

- The goal of classification is to take a single observation, extract some useful features, and thereby classify the observation into one of a set of discrete classes.
- One method for classifying text is to use **hand-written rules**.
- Rules based on combinations of words or other features
  - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high if rules carefully are refined by experts.
- But building and maintaining these **hand-written rules** can be expensive.
  - Rules can be fragile.
  - It may require domain knowledge.

Classification Methods:
Supervised Machine Learning

- Most cases of ***text classification*** in language processing are done via **supervised machine learning** methods.

- The goal of a **supervised machine learning algorithm** is to learn how to map from a new observation to a correct output.

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$
- a training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

Output:

- a learned classifier $model: d \rightarrow c$

Classification Methods: Supervised Machine Learning

- Our goal is to learn a classifier that is capable of mapping from a ***new document*** d to its ***correct class*** $c \in C$.
- A **probabilistic classifier** additionally will tell us the probability of the observation being in the class.
- **Generative classifiers** like **Naive Bayes** build a model of how a class could generate some input data.
  - Given an observation, they return the class most likely to have generated the observation.
- **Discriminative classifiers** like **logistic regression** instead learn what features from the input are most useful to discriminate between the different possible classes.
- Some classifiers:
  - Naïve Bayes
  - Logistic regression
  - Support-vector machines
  - k-Nearest Neighbors, …

Naïve Bayes Classifier

Naïve Bayes Classifier

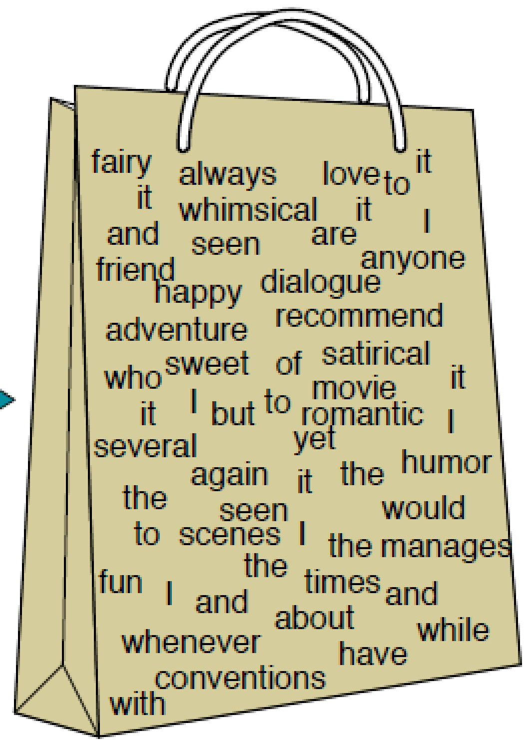
- A **Naïve Bayes Classifier** is a *Bayesian classifier* (based on *Bayes rule*) that makes a simplifying (*naive*) assumption about how the features interact.

- A text document is represented as a **bag of words**.
  - i.e. a text document is represented as an unordered set of words with their position ignored, keeping only their frequency in the document.

Bag of Words Representation

Bag of Words: ***The position of the words is ignored (the bag of words assumption) and we make use of the frequency of each word.***

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
...	...

Bag of Words Representation

- A Naive Bayes classifier may use **all words in the text as features**.

classifier (

```
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.
```

) **= class**

Bag of Words Representation

- Or, a Naive Bayes classifier may use **a subset of words in the text as features**.

classifier (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.) = class

Naïve Bayes Classifier

- Naive Bayes is a **probabilistic classifier**, meaning that for a document d, out of all classes $c \in C$ the classifier returns the class $\hat{c}$ which has the **maximum posterior probability** given the document.
  - the hat notation $\hat{c}$ to mean "our estimate of the correct class".

- For a document d and a class c

$$\mathbf{P(c|d) = \frac{P(d|c)\, P(c)}{P(d)}}$$

- Most likely class (MAP: maximum a posteriori):

$$\hat{\mathbf{c}} = \mathbf{c_{MAP}} = \underset{\mathbf{c \in C}}{\mathbf{argmax}}\ \mathbf{P(c|d)}$$

Naïve Bayes Classifier

$$\hat{c} = c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(c|d)$$

By Bayes Rule:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)\, P(c)}{P(d)}$$

Dropping the denominator (it is same for all classes):

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; P(d|c)\, P(c)$$

likelihood of the document

prior probability of the class

Naïve Bayes Classifier

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbf{C}}{\mathbf{argmax}}\, \mathbf{P}(\mathbf{d}|\mathbf{c})\, \mathbf{P}(\mathbf{c})$$

- we can represent a document **d** as a set of features $\mathbf{f_1,f_2, \ldots,f_n}$:

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbf{C}}{\mathbf{argmax}}\, \mathbf{P}(\mathbf{f_1,f_2,\ldots,f_n}|\mathbf{c})\, \mathbf{P}(\mathbf{c})$$

- Unfortunately, it is hard to compute $\mathbf{P}(\mathbf{f_1,f_2,\ldots,f_n}|\mathbf{c})$ directly:
  - estimating probability of every possible combination of features would require huge numbers of parameters and impossibly large training sets.

- Naive Bayes classifiers make **two simplifying assumptions**.
  - **Bag of Words Assumption**: Assume position doesn't matter.
  - **Naive Bayes Assumption:** ***conditional independence*** assumption that the probabilities $\mathbf{P(f_i|c)}$ are ***independent*** given the class **c** and hence they can be multiplied as follows:

$$\mathbf{P(f_1,\ldots,f_n|c) = P(f_1|c) * \ldots * P(f_n|c)}$$

Naïve Bayes Classifier

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(f_1, f_2, \ldots, f_n | c)\, P(c)$$

- With **simplifying assumptions (bag of words assumption** and **Naive bayes assumption)** the class chosen by a **naive Bayes classifier** is:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\ P(c) \prod_{f \in F} P(f|c)$$

Applying Naive Bayes Classifier to Text Classification

- To apply the naive Bayes classifier to text, we need to consider word positions, by simply walking an index through every word position in the document:

positions ← all word positions in test document

$$\mathbf{c_{NB} = \underset{c \in C}{argmax}\ P(c) \prod_{i \in positions} P(w_i|c)}$$

- Naive Bayes calculations can be done in log space, in order to avoid underflow and increase speed.

$$\mathbf{c_{NB} = \underset{c \in C}{argmax}\ \log\big(P(c)\big) + \sum_{i \in positions} \log(P(wi|c))}$$

Naive Bayes Classifier: Learning

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \, P(c) \prod_{i \in positions} P(w_i|c)$$

- How can we learn the probabilities **P(c)** and **$P(w_i|c)$** ?
- We learn these probabilities from our training set: $(d_1,c_1),....,(d_m,c_m)$

- **P(c)** can be computed (estimated) as follows:

$$P(c) = \frac{N_c}{N_{doc}}$$

- N_c is the **number of documents in our training data with class c** and

  N_{doc} is the **total number of documents**.

Naive Bayes Classifier: Learning

- In order to compute (estimate) $\mathbf{P(w_i|c)}$:
  - First, concatenate all documents with category **c** into one big "**category c**" text.
  - Then, use the frequency of $\mathbf{w_i}$ in this concatenated document to give a maximum likelihood estimate of the probability.

$$\mathbf{P(w_i|c) = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}}$$

- The vocabulary V consists of the union of all the word types in all classes, not just the words in one class c.

Naive Bayes Classifier: Learning

Example: We can compute **P(fantastic|positive)** as follows:

– The probability of **fantastic** word given category **positive**.

$$\mathbf{P}(\mathbf{fantastic}|\mathbf{positive}) = \frac{\mathbf{count}(\mathbf{fantastic},\mathbf{positive})}{\sum_{\mathbf{w} \in \mathbf{V}} \mathbf{count}(\mathbf{w},\mathbf{positive})}$$

Naive Bayes Classifier: Learning
Smoothing

- In order to get rid of zero probability values for $\mathbf{P(w_i|c)}$, we should use a **smoothing method**.

- The simplest solution is the **add-one (Laplace) smoothing:**

$$\mathbf{P(w_i|c)} = \frac{\mathbf{count(w_i,c)+1}}{\sum_{\mathbf{w \in V}}(\mathbf{count(w,c)+1})} = \frac{\mathbf{count(w_i,c)+1}}{(\sum_{\mathbf{w \in V}} \mathbf{count(w,c)})+|\mathbf{V}|}$$

Naive Bayes Classifier: Learning *unknown words*

- What do we do about words that occur in our test data but are not in our vocabulary at all because they did not occur in any training document in any class?
- The solution for such **unknown words** is to ***ignore them*** — ***remove them from the test document and not include any probability for them at all.***
- Some systems choose to completely ignore another class of words:
  - **stop words**, very frequent words like the and a.

Naive Bayes Algorithm: Train

using add-1 smoothing

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms

N_{doc} = number of documents in D

N_c = number of documents from D in class c

$logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$

$V \leftarrow$ vocabulary of D

$bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c

for each word w in V # Calculate $P(w|c)$ terms

$count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$

$loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$

return *logprior*, *loglikelihood*, V

Naive Bayes Algorithm: Test
using add-1 smoothing

function TEST NAIVE BAYES(*testdoc*, *logprior*, *loglikelihood*, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in *testdoc*
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word, c]$
return $\text{argmax}_c \; sum[c]$

Naive Bayes Classifier
sentiment analysis example

- A ***sentiment analysis domain*** with the **two classes positive (+) and negative (-),** and the following miniature **training and test documents** simplified from actual movie reviews.

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- The prior **P(c)** for the two classes: $\mathbf{P(c) = \frac{N_c}{N_{doc}}}$

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

Naive Bayes Classifier
sentiment analysis example

- The word **with** doesn't occur in the training set, so we drop it completely.
- The **likelihoods (conditional probabilities)** from the training set for the remaining three words "**predictable**", "**no**", and "**fun**" are:

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

$$\mathrm{P}(\mathrm{w_i}|\mathrm{c}) = \frac{\mathrm{count}(\mathrm{w_i},\mathrm{c})+1}{(\sum_{\mathrm{w}\in\mathrm{V}}\mathrm{count}(\mathrm{w},\mathrm{c}))+|\mathrm{V}|}$$

$|V| = 20$

$$\sum_{\mathrm{w}\in\mathrm{V}} \mathrm{count}(\mathrm{w},-) = 14$$

$$\sum_{\mathrm{w}\in\mathrm{V}} \mathrm{count}(\mathrm{w},+) = 9$$

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Naive Bayes Classifier
sentiment analysis example

- For the test sentence S = "**predictable with no fun**", after removing the word '**with**', the **chosen class** is **negative (-):**

$$\boxed{P(-)P(S|-) \ = \ \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) \ = \ \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Naïve Bayes and Language Modeling

- When we use **all** of the words in the text (not a subset), **Naïve Bayes** has an important similarity to **language modeling**.
- Each class = a **unigram language model**
- Assigning each sentence**: P(s|c) = Π P(word|c)**

<u>**P(w|pos)**</u>	<u>**w**</u>
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film
…	

<u>**I**</u>	<u>**love**</u>	<u>**this**</u>	<u>**fun**</u>	<u>**film**</u>
0.1	**0.1**	**0.01**	**0.05**	**0.1**

P(s|pos) = 0.0000005

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos

P(w|pos)	w
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg

P(w|neg)	w
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

P(s|pos) = 0.0000005

P(s|neg) = 0.000000001

P(s|pos) > **P(s|neg)**

Text Classification: Evaluation

Contingency Table, *Precision, Recall, F-measure*

- In order to evaluate ***how good is our classifier***, we can use different **evaluation metrics**.
- In evaluation, we compare the **test set results** of our classifier with **gold labels** (*the human labels for the test set documents*).
- As a result of this comparison, first we build a **contingency table** before calculate our **evaluation metrics**.

Contingency Table

		gold standard labels		
		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	**precision** $= \frac{tp}{tp+fp}$
	system negative	**false negative**	**true negative**	
		recall $= \frac{tp}{tp+fn}$		**accuracy** $= \frac{tp+tn}{tp+fp+tn+fn}$

Text Classification: Evaluation

Accuracy, Precision, Recall

- **Accuracy** is ***percentage of all the observations our system labeled correctly***.
  - Although accuracy might seem a natural metric, accuracy doesn't work well when the classes are unbalanced.

$$\textbf{Accuracy} = \frac{\textbf{tp+tn}}{\textbf{tp+fp+tn+fn}}$$

- **Precision** measures ***percentage of items that system detected that are in fact positive***.

$$\textbf{Precision} = \frac{\textbf{tp}}{\textbf{tp+fp}}$$

- **Recall** measures ***percentage of items actually present in the input that were correctly identified by the system***.

$$\textbf{Recall} = \frac{\textbf{tp}}{\textbf{tp+fn}}$$

Text Classification: Evaluation

A combined measure: F-measure

- **Precision** and **Recall**, unlike Accuracy, emphasize ***true positives***.
  - Looking only one of them can be misleading.
  - tp=1 fp=0 fn=99 ➔ Precision = %100 (while Recall=%1)
  - tp=1 fp=99 fn=0 ➔ Recall= %100 (while Precision=%1)

- **F-measure** is a single metric that incorporates aspects of both **precision** and **recall**.

$$\mathbf{F}_\beta = \frac{(\boldsymbol{\beta}^2+\mathbf{1}) * \mathbf{P} * \mathbf{R}}{\boldsymbol{\beta}^2 * \mathbf{P} + \mathbf{R}}$$

- The **β** parameter differentially weights the importance of recall and precision.
  - values of **β>1** favor recall, while values of **β<1** favor precision.

- The most frequently used metric, and is called $\mathbf{F}_{\boldsymbol{\beta}=\mathbf{1}}$ or just $\mathbf{F}_\mathbf{1}$:

$$\mathbf{F}_\mathbf{1} = \frac{\mathbf{2} * \mathbf{P} * \mathbf{R}}{\mathbf{P} + \mathbf{R}}$$

Text Classification ***with more than two classes***

- Many of classification tasks in language processing have more than two classes.
  - For sentiment analysis we generally have 3 classes (positive, negative, neutral).

- There are two kinds of multi-class classification tasks:
  - **any-of** (or **multi-label**) **classification**,
  - **one-of** (or **multinomial**) **classification**

any-of (or **multi-label**) **classification:**

- ***Each document can be assigned more than one label***.
- We can solve **any-of classification** by building separate binary classifiers for each class c, trained on positive examples labeled c and negative examples not labeled c.
- Given a test document d, then each classifier makes their decision independently, and we may assign multiple labels to d.

Text Classification *with more than two classes*

one-of (or **multinomial**) **classification:**

- ***classes are mutually exclusive and each document appears in exactly one class.***
- We again build a separate binary classifier trained on positive examples from c and negative examples from all other classes.
- Given a test document d, we run all the classifiers and ***choose the label from the classifier with the highest score.***

Text Classification: Confusion matrix

with more than two classes

- Confusion matrix for a three-class categorization task,
  - for each pair (c_1,c_2), how many documents from c_1 were (in)correctly assigned to c_2

		gold labels			
		urgent	normal	spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u = \frac{8}{8+5+3}$	**recall**$_n = \frac{60}{10+60+30}$	**recall**$_s = \frac{200}{1+50+200}$	

Text Classification: class evaluation measures

with more than two classes

- **Accuracy:** Fraction of documents classified correctly (= 1 - error rate).

$$\textbf{Accuracy} = \frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

- **Precision:** Fraction of documents assigned class **i** that are actually about class **i**:

$$\textbf{Precision} = \frac{\sum_i c_{ii}}{\sum_j c_{ji}}$$

- **Recall:** Fraction of documents in class **i** classified correctly:

$$\textbf{Recall} = \frac{\sum_i c_{ii}}{\sum_j c_{ij}}$$

Microaveraging and Macroaveraging

- In order to derive a single metric that tells us how well the system is doing, we can combine these values in two ways.
  - In **macroaveraging**, compute performance for each class, and then average over classes.
  - In **microaveraging**, collect decisions for all classes into a single contingency table, and then compute precision and recall from that table.

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

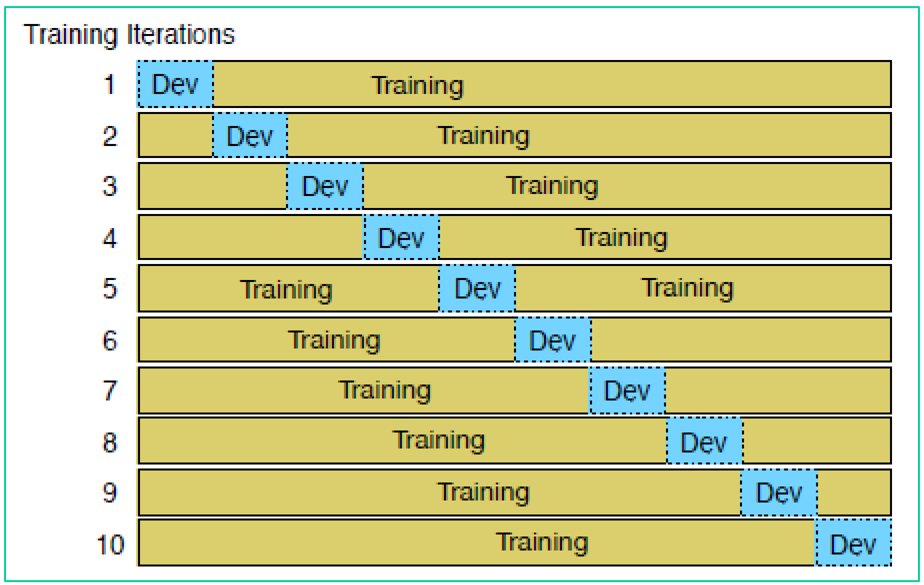
$$\text{microaverage precision} = \frac{268}{268+99} = .73$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$$

Cross-validation

cross-validation:

- we randomly choose a training and test set division of our data, train our classifier, and then compute the error rate on the test set.
- Then we repeat with a different randomly selected training set and test set.
- We do sampling process 10 times and average these 10 runs to get an average error rate.
- This is called **10-fold cross-validation**.



Text Classification: Summary

- Text categorization assigns an entire text to a class from a finite set,
- Some text categorization tasks are sentiment analysis, spam detection, language identification, and authorship attribution.
- Naive Bayes is a generative model that make the bag of words assumption (position doesn’t matter) and the conditional independence assumption (words are conditionally independent of each other given the class).
- Classifiers are evaluated based on precision and recall.
- Classifiers are trained using distinct training, dev, and test sets, including the use of cross-validation in the training set.