Probabilistic Models

Probabilistic Model

- Let's start with a simple biased coin example.
  - You flip a coin $N = 100$ times and get outcomes $\{x_1, \ldots, x_N\}$ where $x_i \in \{0, 1\}$ and $x_i = 1$ is interpreted as heads H.
  - Suppose you had $N_H = 55$ heads and $N_T = 45$ tails.
  - What is the probability it will come up heads if we flip again? Let's design a model for this scenario, fit the model. We can use the fit model to predict the next outcome.

Model?

- The coin is possibly loaded. So, we can assume that one coin flip outcome x is a Bernoulli random variable for *some currently unknown parameter* $\theta \in [0, 1]$

$$P(x = 1; \theta) = \theta \text{ and } P(x = 0; \theta) = 1 - \theta$$

$$\text{or compactly } P(x; \theta) = \theta^x (1 - \theta)^{1-x}$$

- It's sensible to assume that $\{x_1, \ldots, x_N\}$ are independent and identically distributed (i.i.d.) Bernoullis.
- Thus the joint probability of the outcome $\{x_1, \ldots, x_N\}$ is

$$P(x_1, \ldots, x_N; \theta) = \prod_{i=1}^{N} \theta^{x_i} (1 - \theta)^{1-x_i}$$

Likelihood Function

- We call the probability mass (or density for continuous) of the observed data the likelihood function (as a function of the parameters θ):

$$L(\theta) = \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i}$$

- We usually work with log-likelihoods:

$$\ell(\theta) = \sum_{i=1}^{N} x_i \log\theta + (1-x_i)\log(1-\theta)$$

- How can we choose θ? Good values of θ should assign high probability to the observed data. This motivates the maximum likelihood criterion, that we should pick the parameters that maximize the likelihood:

$$\hat{\theta}_{ML} = \max_{\theta\in[0,1]} \ell(\theta)$$

Maximum Likelihood Estimation for the Coin Example

- We find the maximum by setting derivatives to zero

$$\begin{aligned}\frac{d\ell}{d\theta} &= \frac{d}{d\theta}\left(\sum_{i=1}^{N} x_i \log\theta + (1-x_i)\log(1-\theta)\right) \\ &= \frac{d}{d\theta}\left(N_H \log\theta + N_T \log(1-\theta)\right) \\ &= \frac{N_H}{\theta} - \frac{N_T}{1-\theta}\end{aligned}$$

  where $N_H = \sum_i x_i$ and $N_T = N - \sum_i x_i$

- Setting this to zero gives the maximum likelihood estimate:

$$\hat{\theta}_{ML} = \frac{N_H}{N_H + N_T}$$

Maximum Likelihood Estimation

- Notice, in the coin example we are actually minimizing cross-entropies!

$$\begin{aligned}\hat{\theta}_{ML} &= \max_{\theta \in [0,1]} \ell(\theta) \\ &= \min_{\theta \in [0,1]} -\ell(\theta) \\ &= \min_{\theta \in [0,1]} \sum_{i=1}^{N} -x_i \log \theta - (1 - x_i) \log(1 - \theta)\end{aligned}$$

- This is an example of maximum likelihood estimation.
  - define a model that assigns a probability (or has a probability density at) to a dataset
  - maximize the likelihood (or minimize the neg. log-likelihood).
- Many examples we've considered fall in this framework! Let's consider classification again.

Generative vs Discriminative

Two approaches to classification:

- Discriminative approach: estimate parameters of decision boundary/class separator directly from labeled examples.
  - Model $P(t|\mathbf{x})$ directly (logistic regression models)
  - Learn mappings from inputs to classes (linear/logistic regression, decision trees etc)
  - Tries to solve: How do I separate the classes?
- Generative approach: model the distribution of inputs characteristic of the class (Bayes classifier).
  - Model $P(\mathbf{x}|t)$
  - Apply Bayes Rule to derive $P(t|\mathbf{x})$.
  - Tries to solve: What does each class “look" like?
- Key difference: is there a distributional assumption over inputs?

A Generative Model: Bayes Classiffier

- Aim to classify text into spam/not-spam (yes $c = 1$; no $c = 0$)
- Example: “You are one of the very few who have been selected as a winners for the free $1000 Gift Card."
- Use bag-of-words features, get binary vector $\mathbf{x}$ for each email
- Vocabulary:
  - “a”: 1
  - . . .
  - “car”: 0
  - “card”: 1
  - . . .
  - “win”: 0
  - “winner”: 1
  - “winter”: 0
  - . . .
  - “you”: 1

Bayes Classifier

- Given features $\mathbf{x} = [x_1, \ldots, x_D]^\top$ we want to compute class probabilities using Bayes Rule:

$$\underbrace{P(c|\mathbf{x})}_{\text{Pr. class given words}} = \frac{P(\mathbf{x}, c)}{P(\mathbf{x})} = \frac{\overbrace{P(\mathbf{x}|c)}^{\text{Pr words given class}} \quad P(c)}{P(\mathbf{x})}$$

- Informally

$$\text{posterior} = \frac{\text{Class likelihood} \times \text{Prior}}{\text{Evidence}}$$

- How can we compute $P(\mathbf{x})$ for the two class case? (Do we need to?)

$$P(\mathbf{x}) = P(\mathbf{x}|c=0)P(c=0) + P(\mathbf{x}|c=1)P(c=1)$$

- To compute $P(c|\mathbf{x})$ we need: $P(\mathbf{x}|c)$ and $P(c)$

Naive Bayes

- Assume we have two classes: spam and non-spam. We have a dictionary of D words, and binary features $\mathbf{x} = [x_1, \ldots, x_D]$ saying whether each word appears in the e-mail.
- If we define a joint distribution $P(c, x_1, \ldots., x_D)$, this gives enough information to determine $P(c)$ and $P(\mathbf{x}|c)$.
- Problem: specifying a joint distribution over $D + 1$ binary variables requires $2^{D+1} - 1$ entries. This is computationally prohibitive and would require an absurd amount of data to fit.
- We'd like to impose structure on the distribution such that:
  - it can be compactly represented
  - learning and inference are both tractable

Naive Bayes

- Naive assumption: Naive Bayes assumes that the word features x_i are conditionally independent given the class c.
  - This means x_i and x_j are independent under the conditional distribution $P(\mathbf{x}|c)$.
  - Note: this doesn't mean they're independent.
  - Mathematically,

$$P(c, x_1, \ldots, x_D) = P(c)P(x_1, \ldots, x_D|c)$$

reduces to

$$P(c, x_1, \ldots, x_D) = P(c)P(x_1|c)\ldots, P(x_D|c)$$

- Compact representation of the joint distribution
  - Prior probability of class: $P(c = 1) = \pi$ (e.g. spam email)
  - Conditional probability of word feature given class: $P(x_j = 1|c) = \theta_{jc}$ (e.g. word "price" appearing in spam)
  - $2D + 1$ parameters total (before $2^{D+1} - 1$)

Naive Bayes: Learning

- The parameters can be learned efficiently because the log-likelihood decomposes into independent terms for each feature.

$$
\begin{aligned}
\ell &= \sum_{i=1}^{N} \log P(c^{(i)}, \mathbf{x}^{(i)}) \\
&= \sum_{i=1}^{N} \log \left(P(\mathbf{x}^{(i)} | c^{(i)}) P(c^{(i)}) \right) \\
&= \sum_{i=1}^{N} \log \left(P(c^{(i)}) \prod_{j=1}^{D} P(x_j^{(i)} | c^{(i)}) \right) \\
&= \sum_{i=1}^{N} \left(\log P(c^{(i)}) + \sum_{j=1}^{D} \log P(x_j^{(i)} | c^{(i)}) \right) \\
&= \sum_{i=1}^{N} \log P(c^{(i)}) + \sum_{j=1}^{D} \sum_{i=1}^{N} \log P(x_j^{(i)} | c^{(i)})
\end{aligned}
$$

- Each of these log-likelihood terms depends on different sets of parameters, so they can be optimized independently.

Naive Bayes: Learning

- We can handle these terms separately. For the prior we maximize: $\sum_{i=1}^{N} \log P(c^{(i)})$
- This is a minor variant of our coin flip example. Let $P(c^{(i)} = 1) = \pi$. Note $P(c^{(i)}) = \pi^{c^{(i)}}(1-\pi)^{(1-c^{(i)})}$
- Log-likelihood:

$$\sum_{i=1}^{N} \log P(c^{(i)}) = \sum_{i=1}^{N} c^{(i)} \log \pi + \sum_{i=1}^{N} (1 - c^{(i)}) \log(1-\pi)$$

- Obtain MLEs by setting derivatives to zero:

$$\hat{\pi} = \frac{\sum_i \mathbb{1}[c^{(i)} = 1]}{N} = \frac{\text{\# spam in dataset}}{\text{total \# samples}}$$

Naive Bayes: Learning

- Each θ_{jc} can be treated separately: maximize $\sum_{i=1}^{N} \log P(x_j^{(i)}|c^{(i)})$
- This is (again) a minor variant of our coin flip example. Let $\theta_{jc} = P(x_j^{(i)}|c)$. Note $P(x_j^{(i)}|c) = \theta_{jc}^{x_j^{(i)}} (1-\theta_{jc})^{1-x_j^{(i)}}$
- Log-likelihood:

$$\sum_{i=1}^{N} \log P(x_j^{(i)}|c^{(i)}) = \sum_{i=1}^{N} c^{(i)} \left(x_j^{(i)} \log \theta_{j1} + (1-x_j^{(i)}) \log(1-\theta_{j1}) \right) + \sum_{i=1}^{N} (1-c^{(i)}) \left(x_j^{(i)} \log \theta_{j0} + (1-x_j^{(i)}) \log(1-\theta_{j0}) \right)$$

- Obtain MLEs by setting derivatives to zero:

$$\hat{\theta}_{jc} = \frac{\sum_i \mathbb{1}[x_j^{(i)} = 1 \;\&\; c^{(i)} = c]}{\sum_i \mathbb{1}[c^{(i)} = c]} \overset{\text{for } c=1}{=} \frac{\#\text{ word } j \text{ appears in spam}}{\#\text{ spams in dataset}}$$

Naive Bayes: Inference

- We predict the category by performing inference in the model.
- Apply Bayes' Rule:

$$P(c|\mathbf{x}) = \frac{P(c)P(\mathbf{x}|c)}{\sum_{c'} P(c')P(\mathbf{x}|c')} = \frac{P(c)\prod_{j=1}^{D} P(x_j|c)}{\sum_{c'} P(c')\prod_{j=1}^{D} P(x_j|c')}$$

- We need not compute the denominator if we're simply trying to determine the most likely c.
- Shorthand notation

$$P(c|\mathbf{x}) \propto P(c)\prod_{j=1}^{D} P(x_j|c)$$

- For input $\mathbf{x}$, predict the class c by comparing the values of $P(c)\prod_{j=1}^{D} P(x_j|c)$ for different c, e.g., choose the largest.

$$h(\mathbf{x}) = \arg\max P(c)\prod_{j=1}^{D} P(x_j|c)$$

Naive Bayes: Inference

General case with k classes ($c \in \{c_1, \ldots, c_k\}$)

- For input $\mathbf{x}$, predict the class c by comparing the values of $P(c) \prod_{j=1}^{D} P(x_j|c)$ for different c, e.g., choose the largest, that is

$$h(\mathbf{x}) = \arg\max_{c_k} P(c = c_k) \prod_{j=1}^{D} P(x_j|c = c_k)$$

- In practice, we use log-probabilities to prevent underflow
  - As D grows, $\prod_{j=1}^{D} P(x_j|c)$ get smaller and smaller

$$h(\mathbf{x}) = \arg\max_{c_k} \log \left(P(c = c_k) \prod_{j=1}^{D} P(x_j|c = c_k) \right)$$

$$= \arg\max_{c_k} \log P(c = c_k) + \sum_{j=1}^{D} \log P(x_j|c = c_k)$$

Naive Bayes

- Naive Bayes is an amazingly cheap learning algorithm!
- Training time: estimate parameters using maximum likelihood
  - Compute co-occurrence counts of each feature with the labels.
  - Requires only one pass through the data!
- Test time: apply Bayes' Rule
  - Cheap because of the model structure. (For more general models, Bayesian inference can be very expensive and/or complicated.)
- We covered the Bernoulli case for simplicity. But our analysis easily extends to other probability distributions.
- Unfortunately, it's usually less accurate in practice compared to discriminative models due to its "naive" independence assumption.

MLE issue: Data Sparsity

- Maximum likelihood has a pitfall: if you have too little data, it can overfit.
- e.g., what if you flip the coin twice and get H both times?

$$\theta_{ML} = \frac{N_H}{N_H + N_T} = \frac{2}{2+0} = 1$$

- Because it never observed T, it assigns this outcome probability 0. This is known as data sparsity.
- Easy fix: Laplace smoothing : Add 1 to each count
  - we should study this in more detail, but this is ok for now...

$$\theta_{ML} = \frac{(N_H + 1)}{(N_H + 1) + (N_T + 1)} = \frac{N_H + 1}{N_H + N_T + 2} = \frac{3}{4} = 0.75$$

Document Classification (with Naive Bayes)

Document Classification

PROBLEM SETTING

Given:

- Representation of a document
- Set of classes 1,...,K

Classes:

(AI) (Programming) (HCI)

ML Planning Semantics Garb.Coll. Multimedia GUI

Training Data:						
	learning	planning	programming	garbage	...	...
	intelligence	temporal	semantics	collection		
	algorithm	reasoning	language	memory		
	reinforcement	plan	proof...	optimization		
	network...	language...		region...		

Document Classification

Test Data:

"planning language proof intelligence"

PROBLEM SETTING

Given:

- Representation of a document
- Set of classes 1,...,K

Determine:

- Class to which document d belongs

Classes:

(AI) (Programming) (HCI)

ML Planning Semantics Garb.Coll. Multimedia GUI

Training Data:

learning	planning	programming	garbage	...	...
intelligence	temporal	semantics	collection		
algorithm	reasoning	language	memory		
reinforcement	plan	proof...	optimization		
network...	language...		region...		

Text Classification: Examples

- Best-studied benchmark: Reuters-21578 newswire stories
  - 9603 train, 3299 test documents, 80-100 words each, 93 classes

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

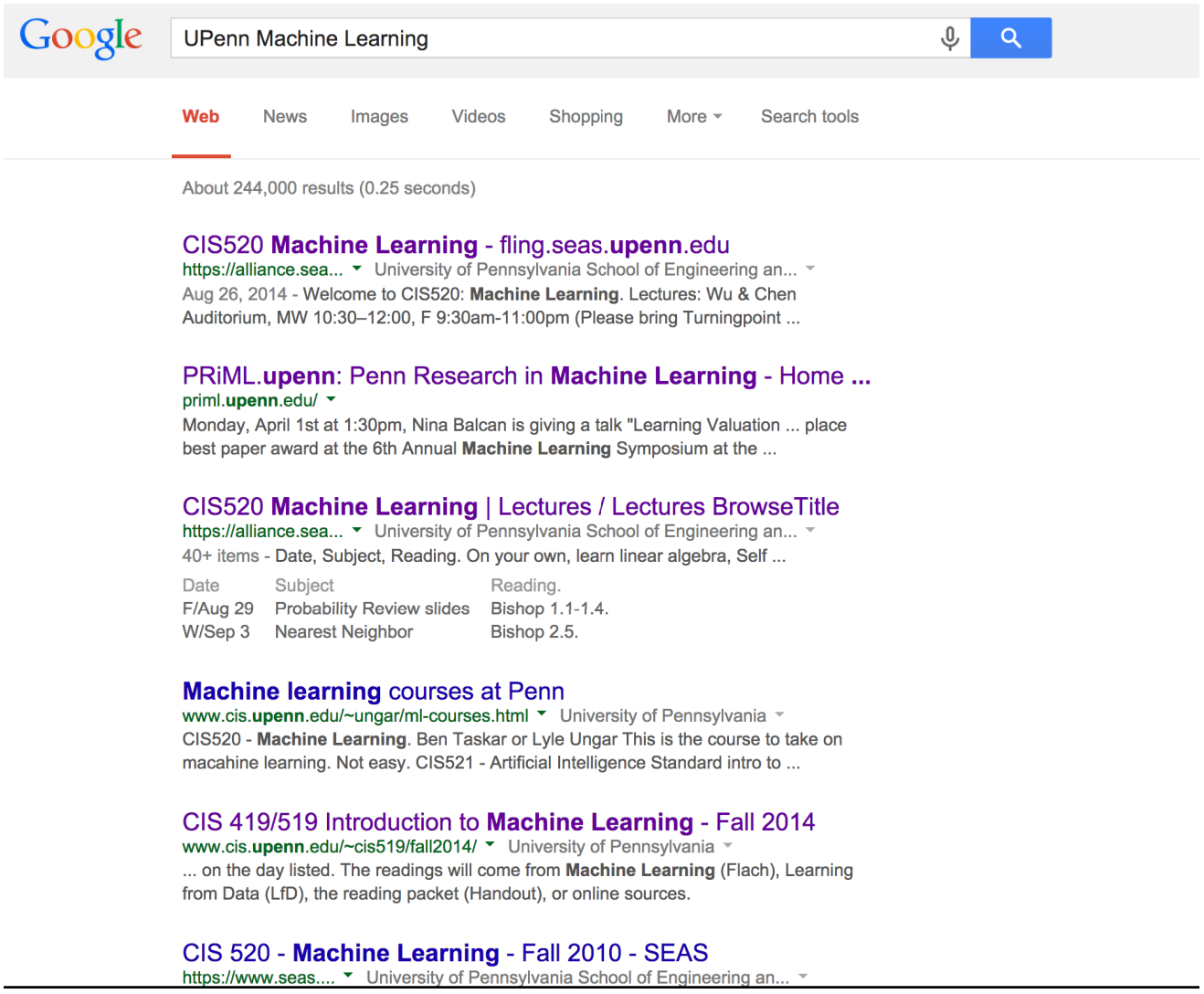
Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

- Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
- Maize Mar 48.0, total 48.0 (nil).
- Sorghum nil (nil)
- Oilseed export registrations were:
- Sunflowerseed total 15.0 (7.9)
- Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

Categories: grain, wheat (out of 93)

Document Retrieval



Spam Filtering

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

==========
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
==========

Bag of Words Representation

- What is the best way to represent a document?

Bag of Words Representation

- What is the best way to represent a document?
- Actually, What is the simplest, yet useful way to represent a document?

Idea: treat each documents as a sequence of words

- Assume that word positions are generated independently

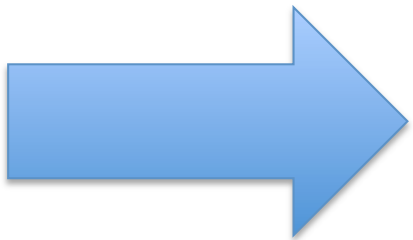
Dictionary: set of all possible words

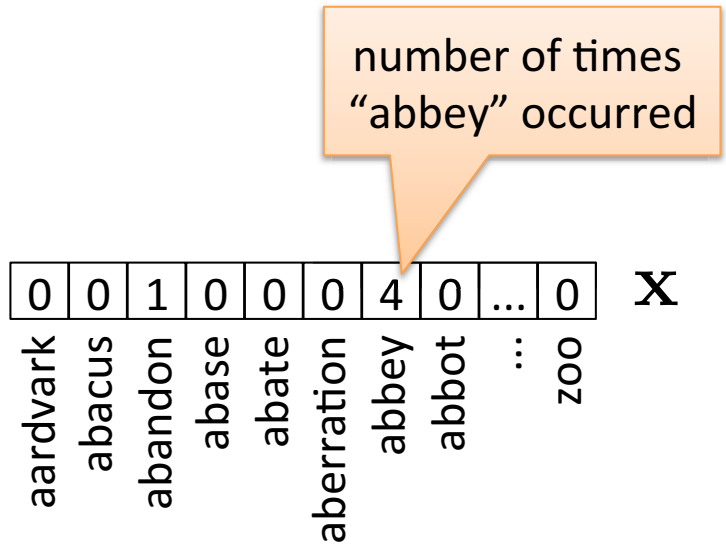
- Compute over set of documents
- Use a dictionary, etc.

Bags of Words Representation

Represent document *d* as a vector of words counts **x**

- x_j represent the count of word *j* in the document
  - **x** is sparse (few non-zero entries)





Naive Bayes for Document Classification

- Let the model parameters for class c be given by:

$$\boldsymbol{\theta}_c = \{\theta_{c1}, \ldots, \theta_{c|D|}\}$$

  - D dictionary
  - $\theta_{cj} = P(\text{word } j \text{ occurs in a document of class } c)$
  - $\sum_j \theta_{cj} = 1, \forall c$
- The likelihood of a document d characterized by $\mathbf{x}$ is

$$P(d|c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

Naive Bayes for Document Classification

- Let the model parameters for class c be given by:

$$\boldsymbol{\theta}_c = \{\theta_{c1}, \ldots, \theta_{c|D|}\}$$

  - D dictionary
  - $\theta_{cj} = P(\text{word } j \text{ occurs in a document of class } c)$
  - $\sum_j \theta_{cj} = 1, \forall c$
- The likelihood of a document d characterized by $\mathbf{x}$ is

$$P(d|c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

- which is just the multinomial distribution, a generalization of the binomial distribution $\binom{n}{k} p^k (1-p)^{n-k}$

Naive Bayes for Document Classification

- The likelihood of a document d characterized by $\mathbf{x}$ is

$$P(d|c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

- Using Bayes rule:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \propto P(d|c)P(c) \propto \prod_j (\theta_{cj})^{x_j} P(c)$$

- Thefore (after taking the log)

$$h(d) = \arg\max_c \left(\log P(c) + \sum_{j=1}^{|D|} x_j \log \theta_{jc} \right)$$

which is just a linear function of $\mathbf{x}$.

Document Classification with Naive Bayes

1. Compute dictionary D over training set
2. Estimate class priors via counting, e.g.,

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

   - N is the number of documents
   - N_c is the number of class c documents
3. Estimate conditional probabilities (with Laplace smoothing), e.g.,

$$\hat{\theta}_{cj} = \frac{W_{cj} + 1}{W_c + |D|}$$

   - W_c is the total number of words in all documents from class c
   - W_{cj} is the number of times word j occurs in documents from class c

- The Naive Bayes classification for a new document d is:

$$h(d) = \arg\max_c \left(\log \hat{P}(c) + \sum_j x_j \log \hat{\theta}_{jc} \right)$$

Issues with Bag of Words Representation

- Documents have different lenghts
- Some words aren't meaningful to represent the content of a document
  - e.g., "the", "a", etc.
- Rare words may be more meaningful than common words

Need a better representation for the documents...

Eliminate Stop Words

Common, “less meaningful” words are called stop words

- Delete stop words before doing any document processing

Example stop words:

a	because	does	haven't	i	more	our	some	they'll	we'll	why
about	been	doesn't	having	i'd	most	ours	such	they're	we're	why's
above	before	doing	he	i'll	mustn't	ourselves	than	they've	we've	with
after	being	don't	he'd	i'm	my	out	that	this	were	won't
again	below	down	he'll	i've	myself	over	that's	those	weren't	would
against	between	during	he's	if	no	own	the	through	what	wouldn't
all	both	each	her	in	nor	same	their	to	what's	you
am	but	few	here	into	not	shan't	theirs	too	when	you'd
an	by	for	here's	is	of	she	them	under	when's	you'll
and	can't	from	hers	isn't	off	she'd	themselves	until	where	you're
any	cannot	further	herself	it	on	she'll	then	up	where's	you've
are	could	had	him	it's	once	she's	there	very	which	your
aren't	couldn't	hadn't	himself	its	only	should	there's	was	while	yours
as	did	has	his	itself	or	shouldn't	these	wasn't	who	yourself
at	didn't	hasn't	how	let's	other	so	they	we	who's	yourselves
be	do	have	how's	me	ought		they'd	we'd	whom	

Term Frequency

Term frequency $f_{t,d}$ is some measure of importance of term t to document d. Let $c_{t,d}$ be the number of times t occurs in d

- **Boolean**: $f_{t,d} = 1$ if $c_{t,d} > 0$, 0 otherwise
- **Raw Counts**: $f_{t,d} = c_{t,d}$
- **Log-Scaled Counts**: $f_{t,d} = \begin{cases} 1 + \log c_{t,d} & \text{if } c_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$
  - Reduce relative impact of frequent term
- **Normalized Counts**: $f_{t,d} = c_{t,d}/|d|$
  - Normalize raw counts by the length of document d

Inverse Document Frequency

- **Idea**: rare terms are more important than common terms
- *Example*: if all training documents for a class contain
  - the (relatively) common word “water”, and
  - the (relatively) rare word “hippopotamus”,
  - the term “hippopotamus” is likely more important
- Inverse Document Frequency

$$if_{t,N} = \log \frac{|N|}{|N_t| + 1}$$

- N is the total set of documents
- N_t is the subset of documents containing term t

TF-IDF Transform

- To compensate for issues with raw word counts, use TF-IDF transform on the features with naive Bayes

$$fif_{t,d,X} = f_{t,d} \times if_{t,X}$$

  - Represent documents as a vector x of TF-IDF features
  - x_j represents the TF-IDF of word j in the document

- **Recommendation** (from [Rennie, et al, ICML'03])
  - Use raw counts of log-scaled counts for $f_{t,d}$
  - Normalize each TF-IDF vector x to have unit length, i.e., $\tilde{x} = x / \parallel x \parallel_2$ and use these unit vectors in naive Bayes