

# ECS763

## Natural Language Processing

### Week 2

### Text Classification

Massimo Poesio

(with material from Matthew Purver, Jurafsky &  
Martin book, Ng machine learning course)

Classification and text classification

## TEXT CLASSIFICATION

From: "" <akworld@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>

=====

## CLASSIFICATION AND CATEGORIZATION

- Given:

- A description of an instance,  $x \in X$ , where  $X$  is the *instance language* or *instance space*.

- A fixed set of categories:

$$C = \{c_1, c_2, \dots, c_n\}$$

- Determine:

- The category of  $x$ :  $c(x) \in C$ , where  $c(x)$  is a *categorization function* whose domain is  $X$  and whose range is  $C$ .

- We want to know how to build categorization functions ("classifiers").

## Text Classification: definition

- *Input:*
  - a document  $d$ 
    - Issue: how to represent text documents.
  - a fixed set of classes  $C = \{c_1, c_2, \dots, c_J\}$
- *Output:* a predicted class  $c \in C$
- *Classifier:* a function  $\gamma: d \rightarrow c$
- *Example:* spam filtering
  - $C = \{\text{spam}, \text{not\_spam}\}$

## Applications of Text Classification

- Classification is the core method in applications including
  - Spam filtering
  - Document categorisation (e.g., divide news articles in articles about sport, finance, etc)
  - Sentiment analysis
  - Language identification
  - Deception detection
  - Stylometry
- Also at the sub-document level

(Supervised) text classification with bags of words

## Supervised text classification

- An application of supervised machine learning
- *Input*:
  - a document  $d$
  - a fixed set of classes  $C = \{c_1, c_2, \dots, c_J\}$
  - a training set of  $m$  hand-labeled documents  $(d_1, c_1), \dots, (d_m, c_m)$
- *Output*: a learned classifier  $\gamma: d \rightarrow c$

## Supervised text classification

- An application of supervised machine learning
- *Input*:
  - a document  $d$
  - a fixed set of classes  $C = \{c_1, c_2, \dots, c_j\}$
  - a training set of  $m$  hand-labeled documents  $(d_1, c_1), \dots, (d_m, c_m)$

Features of documents used for  
classification: WORDS in the first instance

- *Output*: a learned classifier  $\gamma: d \rightarrow c$

## Text Classification using Naïve Bayes

## Bayesian Methods

- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role
- Learn a *generative model* that approximates how data is produced
- Use *prior* probability of each category
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

## Bayes' Rule

$$P(C, X) = P(C | X)P(X) = P(X | C)P(C)$$

$$P(C | X) = \frac{P(X | C)P(C)}{P(X)}$$

## *Maximum a posteriori* Hypothesis

$$h_{MAP} \equiv \operatorname{argmax}_{h \in H} P(h \mid D)$$

$$h_{MAP} = \operatorname{argmax}_{h \in H} \frac{P(D \mid h)P(h)}{P(D)}$$

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D \mid h)P(h)$$

## *Maximum likelihood* Hypothesis

If all hypotheses are a priori equally likely, we only need to consider the  $P(D|h)$  term:

$$h_{ML} \equiv \operatorname{argmax}_{h \in H} P(D \mid h)$$

## Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values

$$\langle x_1, x_2, \dots, x_n \rangle$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_n)$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} \frac{P(x_1, x_2, \dots, x_n | c_j) P(c_j)}{P(c_1, c_2, \dots, c_n)}$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

## Naïve Bayes

- But how can we calculate  $P(x_1, x_2, \dots, x_n | c)$ 
  - (how often will we see it?)

- Instead, assume:
  - features are unordered words (“bag of words”)
  - features are conditionally independent

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \cdot P(x_3 | c) \cdot \dots \cdot P(x_n | c)$$

- So:
 
$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

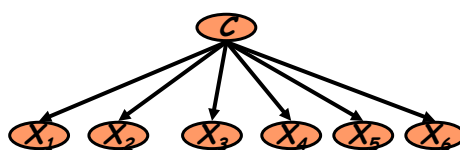
$$= \operatorname{argmax}_{c_j \in C} \log(P(c_j)) + \sum_{i \in \text{positions}} \log(P(x_i | c_j))$$



## Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since  $\log(xy) = \log(x) + \log(y)$ , it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

## Learning a Naïve Bayes model: first approximation



- Common practice: maximum likelihood
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

## Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp
SPAM	50	24	30	14	7	33	50	23	0
NOT	200	1	0	21	150	129	200	51	32
	P(c)	P(w c)							
SPAM	0.2								
NOT	0.8								

## Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp
SPAM	50	24	30	14	7	33	50	23	0
NOT	200	1	0	21	150	129	200	51	32
	P(c)	P(w c)							
SPAM	0.2	0.48	0.60	0.28	0.14	0.60	1.0	0.46	0.00
NOT	0.8	0.01	0.00	0.11	0.75	0.65	1.0	0.26	0.16

## Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp
SPAM	50	24	30	14	7	33	50	23	0
NOT	200	1	0	21	150	129	200	51	32

	P(c)	P(w c)							
SPAM	0.2	0.48	0.60	0.28	0.14	0.60	1.0	0.46	0.00
NOT	0.8	0.01	0.00	0.11	0.75	0.65	1.0	0.26	0.16

	log	log							
SPAM	-1.61	-0.74	-0.51	-1.27	-1.97	-0.51	0.00	-0.78	-Inf
NOT	-0.22	-5.30	-Inf	-2.25	-0.29	-0.44	0.00	-1.37	-1.35

- Can we classify:

hi matt

hi matt want some viagra?

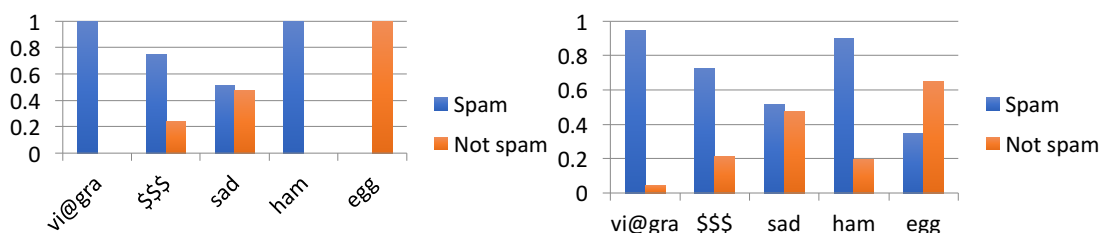
## Problem with the first approximation

- But what if we haven't seen "viagra" with c = not\_spam?

$$\hat{P}(w_i | c_j) = 0!$$

## Solution: smoothing

- Need to reserve some probability mass for unobserved events



- E.g. Laplace (add-1, add-N) smoothing for Naïve Bayes

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

## Text Classification Algorithms: Learning

- From training corpus, extract *Vocabulary*
- Calculate required  $P(c_j)$  and  $P(x_k | c_j)$  terms
  - For each  $c_j$  in  $C$  do
    - $\text{docs}_j \leftarrow$  subset of documents for which the target class is  $c_j$
    - $P(c_j) \leftarrow \frac{|\text{docs}_j|}{|\text{total \# documents}|}$
  - $\text{Text}_j \leftarrow$  single document containing all  $\text{docs}_j$
  - for each word  $x_k$  in *Vocabulary*
    - $n_k \leftarrow$  number of occurrences of  $x_k$  in  $\text{Text}_j$
    - $P(x_k | c_j) \leftarrow \frac{n_k + 1}{n + |\text{Vocabulary}|}$

## Text Classification Algorithms: Classifying

- positions  $\leftarrow$  all word positions in current document which contain tokens found in *Vocabulary*
- Return  $c_{NB}$ , where

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

## A note about Naïve Bayes posterior probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are generally very close to 0 or 1.

## Evaluating text classifiers

## Measuring Performance

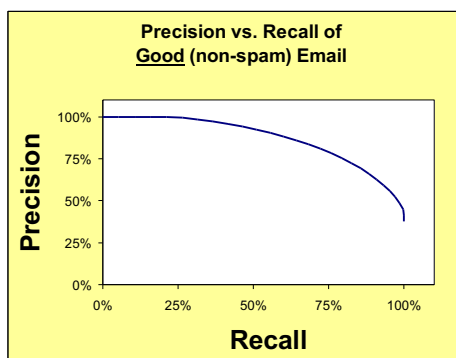
- **Classification accuracy:** What % of messages were classified correctly?
- **Is this what we care about?**

	Overall accuracy	Accuracy on spam	Accuracy on ham
System 1	95%	99.99%	90%
System 2	95%	90%	99.99%

- Which system do you prefer?

33

## Measuring Performance



• **Precision** =  $\frac{\text{good messages kept}}{\text{all messages kept}}$

• **Recall** =  $\frac{\text{good messages kept}}{\text{all good messages}}$

Trade off precision vs. recall by setting threshold  
 Measure the curve on annotated dev data (or test data)  
 Choose a threshold where user is comfortable

34

## Precision and recall

- **Precision**: % of selected items that are correct
- **Recall**: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn

$$P = \frac{tp}{(tp+fp)}$$

$$R = \frac{tp}{(tp+fn)}$$

## A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted HARMONIC MEAN)
- Most used version: F1 measure
  - i.e., with  $\beta = 1$  (that is,  $\alpha = \frac{1}{2}$ ):

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

- General formula:

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

## Why not simple (arithmetic) mean?

- Consider the case of a (binary) classification problem, and a system that classifies every item as 1
- That system would have  $R = 1$ , but  $P = 0$
- So the mean of  $R$  and  $P$  would be .5 even though this is the worst possible system
- But that system would have an  $F$  of 0 (check)



## Text classification in Python

### Good tutorials on text classification in Python

- The Text Categorization tutorial in scikit-learn:
  - [http://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)
  - **Notebook 'text\_categorization\_sklearn' in Lecture 2 page**
- Spam detection with scikit-learn:
  - <http://zacstewart.com/2015/04/28/document-classification-with-scikit-learn.html>
- Sentiment analysis:
  - Twitter sentiment analysis with Python and NLTK:  
<http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>

## Features

## Features for text classification

- Bag of words assumption
  - Good baseline!
- N-gram features
  - Subsequences of N consecutive words
  - Often with START, END symbols

## The need for feature selection

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words – and more
- Feature selection can make using a particular classifier feasible
  - Some classifiers can't deal with 100,000s of feat's
- It can also reduce training time
  - Training time for some methods is quadratic or worse in the number of features (e.g., logistic regression)
- Improve generalization
  - Eliminate noise features
  - Avoid overfitting

## Feature reduction using linguistic knowledge

- Ways of reducing feature space for text using info about words
  - Stemming
    - Laugh, laughs, laughing, laughed -> laugh
  - Stop word removal
    - E.g., eliminate all prepositions
  - Conversion to lower case
  - Tokenization
    - Break on all special characters: fire-fighter -> fire, fighter

## Tokenisation & Normalisation

- We want to split a string into words
  - Tokenisation
    - `I thought "I like it!"`  
`→ [I, thought, I, like, it]`
    - First try: split on non-alphanumeric?
- And have predictable, regular forms
  - Normalisation
    - `WINDOWS = Windows = windows`
    - First try: convert to lower-case?

## Regular Expressions

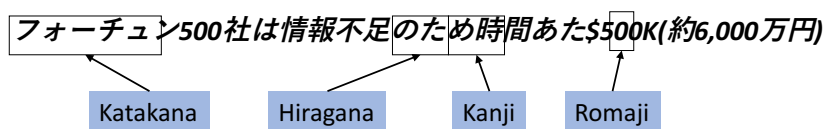
- Disjunctions `[wW] (windows|Windows)`
- Ranges `[a-z] [A-Z] [0-9] [a-zA-Z]`
- Negations `[^a]`
- Wildcards `? . * +`  
`windows?` `window.?` `window*`
- Anchors `^ $`
- Classes `\w \s \W \S`
- Find all instances of the word "the":
  - `the` Misses capitalized
  - examples
  - `[tT]he` Incorrectly returns other or
  - theology
  - `[^a-zA-Z][tT]he[^a-zA-Z]`
  - `\b[tT]he\b`
- Surprisingly powerful and commonly used!

## Tokenisation

- For English:
  - `split(r'\W')` or `split(r'\b')`
  - (or many similar options)
- How about Chinese?
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long. No spaces!
- Standard baseline segmentation algorithm:
  1. Start pointer at beginning of string
  2. Find longest word in dictionary matching string starting at pointer
  3. Move pointer over word, go to 2

## Tokenization

- How about Japanese?
  - multiple alphabets intermingled
  - dates/amounts in multiple formats



- How about German?
  - End-user can express query entirely in hiragana!
  - Lebensversicherungsgesellschaftsangestellter
  - 'life insurance company employee'
  - (wait for sequence modelling lecture ...)

## Spelling Correction

- Spelling errors are common (26% of web queries)
  - Detection
  - Correction
- Hard to estimate error probability directly
  - (have we seen x before?)
- Bayes' Rule again!

$$P(w | x)$$

$$P(w | x) = \frac{P(x | w)P(w)}{P(x)}$$

## Spelling Correction

- Need probability of original word
  - estimate from corpus
  - “unigram language model” (see next week)

$$P(w)$$

- Need probability of error introduction
  - “noisy channel model”
  - e.g. probability of:
    - Leaving a character out
    - Transposing two characters
    - Adding a space character
    - ... etc
  - estimate from corpus of errors

$$P(x | w)$$

- E.g. Kernighan et al (1990) 
$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

## Stemming / Lemmatisation

- We often want base form (stem, lemma)

windows → window

estimating → estimate (→ estimat)

running → run

are, is  $\rightarrow$  be

- **Morphemes:**

- The small meaningful units that make up words
- **Stems:** The core meaning-bearing units
- **Affixes:** Bits and pieces that adhere to stems
  - Often with grammatical functions

## Porter's algorithm

## The most common English stemmer

### Step 1a

sses → ss      caresses → caress

ies → i      ponies → poni

SS → SS    caress → caress

$s \rightarrow \emptyset$        $\text{cats} \rightarrow \text{cat}$

### Step 1b

(*\*v\**)ing  $\rightarrow \emptyset$  walking  $\rightarrow$  walk

sing → sing

(*\*v\**)ed → ∅   plastered → plaster

...

### Step 2 (for long stems)

ational→ ate relational→ relate

izer → **ize**      digitizer → **digitize**

ator → ate      operator → operate

...

### Step 3 (for longer stems)

al → ∅    revival → reviv

able → ∅    adjustable → adjust

ate → ∅    activate → activ

...

## Why check for vowels in Step 1b?

(*\*v\**)ing  $\rightarrow \emptyset$    walking  $\rightarrow$  walk

sing → sing

## Stemming in other languages

- What if we're working with Turkish?
  - **Uygarlaştıramadıklarımızdanmışsınızcasına**
  - '(behaving) as if you are among those whom we could not civilize'
  - **Uygar** 'civilized' + **laş** 'become'
    - + **tır** 'cause' + **ama** 'not able'
    - + **dık** 'past' + **lar** 'plural'
    - + **ımız** 'p1pl' + **dan** 'abl'
    - + **mış** 'past' + **sınız** '2pl' + **casına** 'as if'
- More advanced grammar-based methods:
  - e.g. finite state transducers (see later)

## Feature Selection using statistical 'weighing' methods

- Yang and Pedersen 1997: a comparison of different selection criteria
  - DF – document frequency
  - IG – information gain
  - MI – mutual information
  - CHI – chi square
- Other widely used methods
  - TF/IDF
- Common strategy
  - Compute statistic for each term
  - Keep n terms with highest value of this statistic



## Document Frequency

- Number of documents a term occurs in
- Is sometimes used for eliminating both very frequent and very infrequent terms
- How is document frequency measure different from the other 3 measures?

## Feature selection via Mutual Information

- We might not want to use all words, but just reliable, good discriminators
- In training set, choose  $k$  words which best discriminate the categories.
- One way is in terms of (pointwise) Mutual Information

## (Pointwise) Mutual Information

$$I(t, c) = \log \frac{P_r(t \wedge c)}{P_r(t) \times P_r(c)}$$

$$I_{avg}(t) = \sum_{i=1}^m P_r(c_i) I(t, c_i)$$

$$I_{max}(t) = \max_{i=1}^m \{I(t, c_i)\}$$

## Feature selection via MI (contd.)

- For each category we build a list of  $k$  most discriminating terms.
- For example (on 20 Newsgroups):
  - **sci.electronics**: circuit, voltage, amp, ground, copy, battery, electronics, cooling, ...
  - **rec.autos**: car, cars, engine, ford, dealer, mustang, oil, collision, autos, tires, toyota, ...
- Greedy: does not account for correlations between terms
- In general feature selection is *necessary* for binomial NB, but not for multinomial NB

## TERM WEIGHTING IN VECTOR SPACE MODELS: THE TF.IDF MEASURE

$$tfidf_{i,k} = f_{i,k} * \log\left(\frac{N}{df_i}\right)$$

FREQUENCY of term i  
in document k

Number of documents  
with term i

## Information Gain

- A metric much used in ML to decide on the importance of features e.g., when learning decision trees
- It measures how much a given feature (in this case, term) t REDUCES THE ENTROPY

# Entropy

Information theory: entropy – number of bits needed to encode some information.

Let us suppose we have a set  $S$  of  $N$  documents, belonging to two classes (e.g., 'spam', 'not spam')

$p \cdot N$  belong to class Spam (positive)

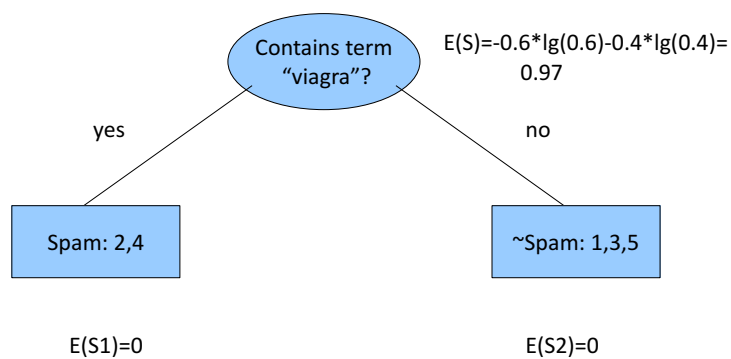
$q \cdot N$  negative (do NOT belong to class Spam)

$$\text{Entropy}(S) = -p \cdot \lg(p) - q \cdot \lg(q)$$

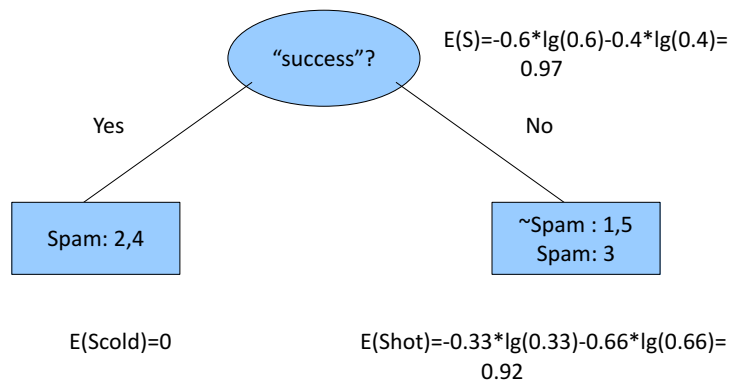
$$p=1, q=0 \Rightarrow \text{Entropy}(S)=0$$

$$p=1/2, q=1/2 \Rightarrow \text{Entropy}(S)=1$$

## Entropy and Decision Trees



## Entropy and Decision Trees



## Information gain

- For each term  $t$ , compute the reduction in entropy on the split:

$$\text{Gain}(S,t) = E(S) - \sum (Entropy(S_i) * |S_i| / |S|)$$

$$t=\text{viagra?} : \text{Gain}(S,f)=0.97$$

$$t=\text{success?} : \text{Gain}(S,f)=0.97 - 0*2/5 - 0.92*3/5=0.42$$

$$t=\text{clothing?} : \text{Gain}(S,f)= ?$$

## Information Gain

$$\begin{aligned} G(t) = & -\sum_{i=1}^m P_r(c_i) \log P_r(c_i) \\ & + P_r(t) \sum_{i=1}^m P_r(c_i|t) \log P_r(c_i|t) \\ & + P_r(\bar{t}) \sum_{i=1}^m P_r(c_i|\bar{t}) \log P_r(c_i|\bar{t}) \end{aligned}$$

## Text classification in Python, 2

## Text classification in Python

- [Back to the Notebook](#)

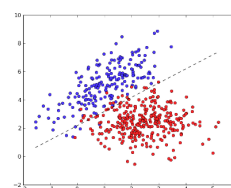
## Text classification with discriminative methods

## Discriminative approaches

- Naïve Bayes is a linear classifier
  - Decision is a linear function of weighted feature values
- But:
  - we calculate feature weights & boundary independently
    - (i.e. we're assuming statistical independence)
  - we're using a generative model

$$y = \operatorname{argmax}_y P(y|x) = \operatorname{argmax}_y P(x|y) \cdot P(y)$$

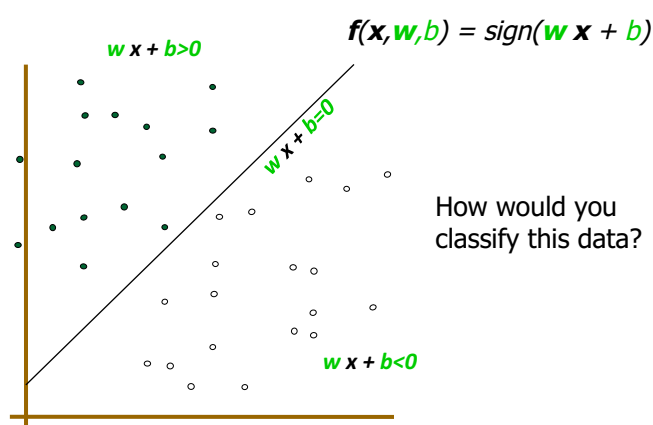
- Why not:
  - estimate the boundary directly?
  - use a discriminative model?



$$y = \operatorname{argmax}_y P(y|x)$$

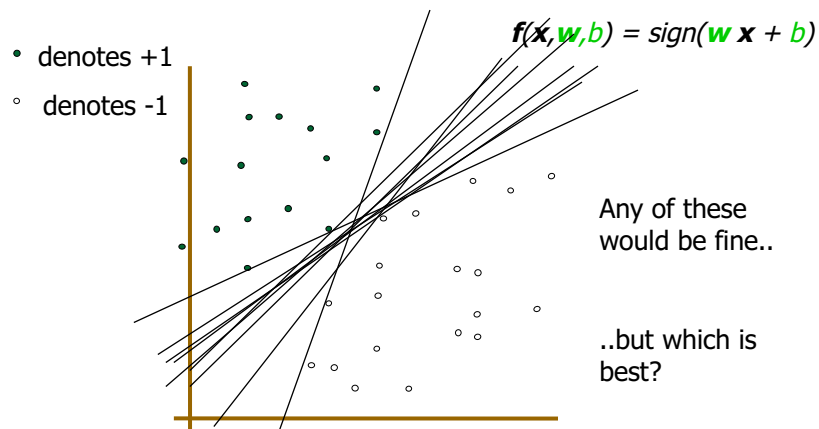
## Linear Classifiers

- denotes +1
- denotes -1



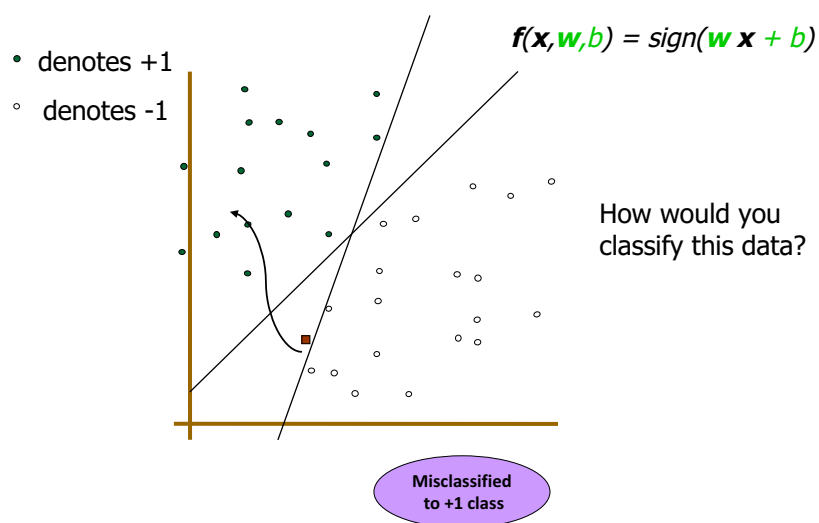


## Linear Classifiers



03/19/12

## Linear Classifiers



03/19/12

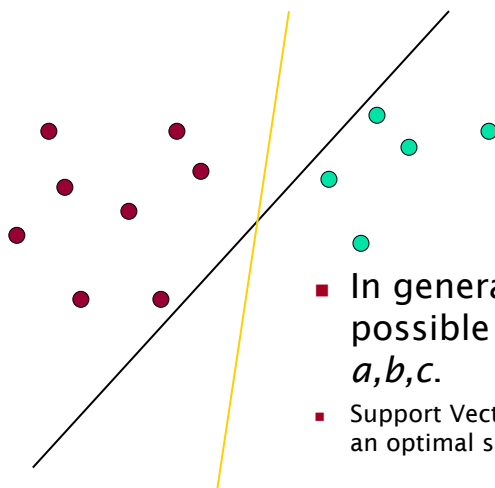
## Linear Classifiers: summary

- Many common text classifiers are linear classifiers
- Despite this similarity, large performance differences
  - For separable problems, there is an infinite number of separating hyperplanes. Which one do you choose?
  - What to do for non-separable problems?

## Separation by Hyperplanes

- Assume *linear separability* for now:
  - in 2 dimensions, can separate by a line
  - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming* (e.g. perceptron):
  - separator can be expressed as  $ax + by = c$

## Which Hyperplane?

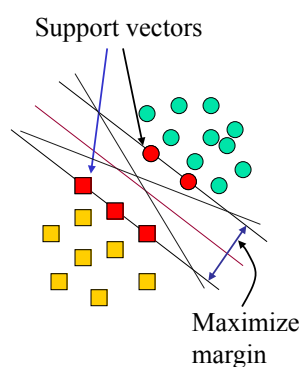


- In general, lots of possible solutions for  $a, b, c$ .
- Support Vector Machine (SVM) finds an optimal solution.

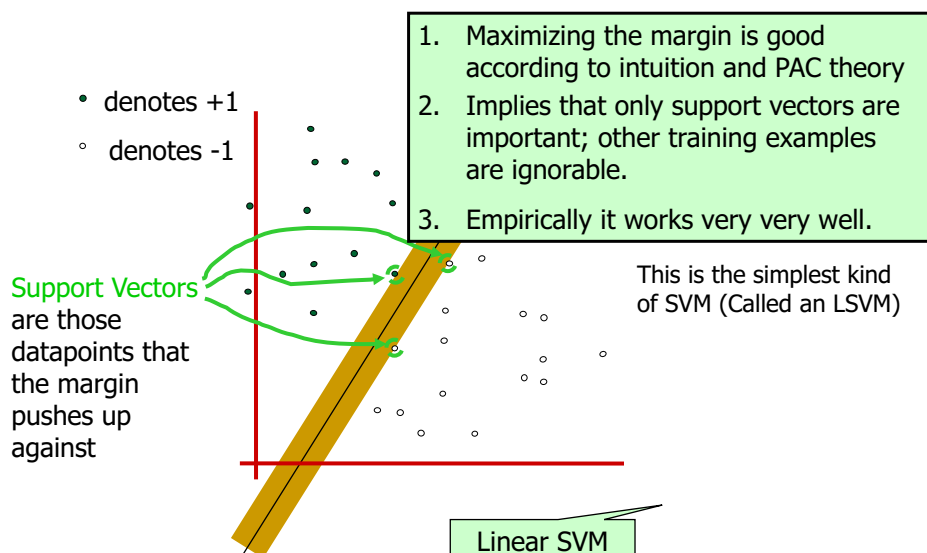
03/19/12

## Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- *Quadratic programming* problem



## Maximum Margin



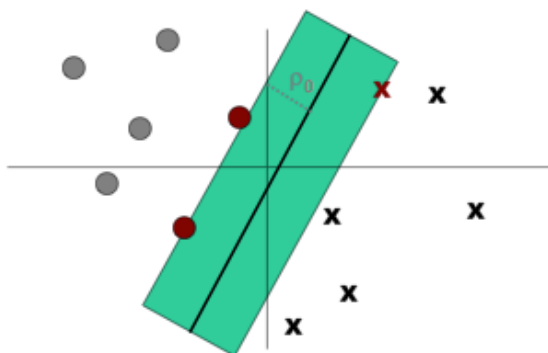
## Maximum Margin: Formalization

- $w$ : hyperplane normal
- $x_i$ : data point  $i$
- $y_i$ : class of data point  $i$  (+1 or -1)
- Constraint optimization formalization:

- (1)  $\mathbf{x}_i \cdot \mathbf{w} + b \geq +1$  for  $y_i = +1$   
 $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$  for  $y_i = -1$
- (2) maximize margin:  $2/||w||$

## Maximum margin in geometrical terms

**Support Vectors: Input vectors for which**  
 $\mathbf{w}_0^T \mathbf{x} + b_0 = 1$  or  $\mathbf{w}_0^T \mathbf{x} + b_0 = -1$

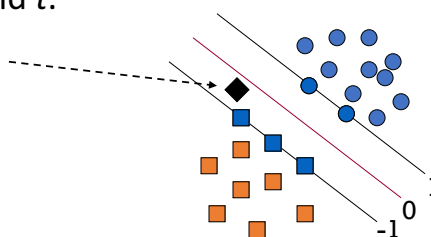


03/19/12

## Classification with SVMs

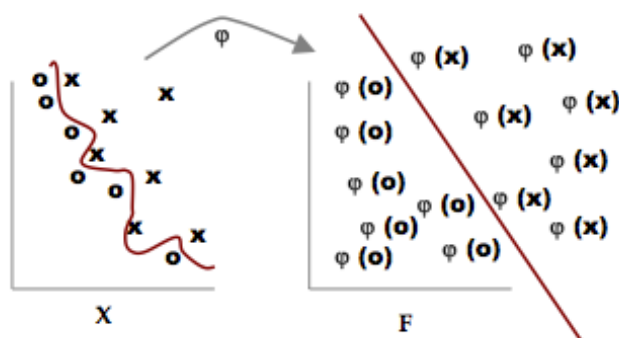
- Given a new point  $\mathbf{x}$ , we can score its projection onto the hyperplane normal:
  - I.e., compute score:  $\mathbf{w}^T \mathbf{x} + b = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$ 
    - Decide class based on whether  $<$  or  $>$  0
  - Can set confidence threshold  $t$ .

Score  $> t$ : yes  
 Score  $< -t$ : no  
 Else: don't know



80

## Non-linearly separable categories: transforming the space



03/19/12

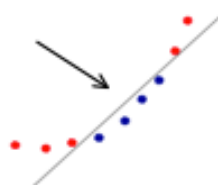
## How non-linear mapping can make a non-linear problem linearly separable

The idea is to gain linearly separation by mapping the data to a higher dimensional space

- The following set can't be separated by a linear function, but can be separated by a quadratic one

$$(x-a)(x-b) = x^2 - (a+b)x + ab$$

- So if we map  $x \mapsto \{x^2, x\}$  we gain linear separation



03/19/12

## Kernels

- The linear classifier relies on an inner product between vectors  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation  $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$ , the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors  $\mathbf{x} = [x_1 \ x_2]$ ; let  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$ ,

Need to show that  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ :

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} = \\ &= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad \text{where } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2] \end{aligned}$$

83

## The Kernel Trick

- We don't have to compute  $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$  explicitly,  $K(\mathbf{x}_i, \mathbf{x}_j)$  is enough for SVM learning

## Some popular kernels

- Polynomials of degree  $d$

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

*Kernel fn.*

- Polynomials of degree up to  $d$

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

- Gaussian/Radial kernels (polynomials of all orders – projected space has infinite dimension)

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$$

## SVMs vs other approaches: Classic Reuters-21578 Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories  
(#train, #test)

- |                            |                       |
|----------------------------|-----------------------|
| • Earn (2877, 1087)        | • Trade (369,119)     |
| • Acquisitions (1650, 179) | • Interest (347, 131) |
| • Money-fx (538, 179)      | • Ship (197, 89)      |
| • Grain (433, 149)         | • Wheat (212, 71)     |
| • Crude (389, 189)         | • Corn (182, 56)      |

86



## Reuters Text Categorization data set (Reuters-21578) document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>

87

(a)	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

Evaluation measure:  $F_1$

88

## Text classification in Python, 2

## Text classification in Python, 3

- [Back to the Notebook: SVMs in scikit-learn](#)

## Applications of Text Classification

- Spam detection
- Sentiment analysis
- Deception detection
- Stylometry

Example: deception detection

## Detecting lies

- Lies are much more common in communication than one would expect (Vrij, 2008)
- There are different types of lies. Many of these are harmless or even beneficial, but some cases of lying are harmful and even criminal, so the ability to detect them could be useful
  - Lying in court (Fornaciari and Poesio, 2013)
  - Deceptive online reviews (Fornaciari and Poesio, 2014)
- People however are not very good at detecting lies
  - Their performance is not better than chance (Bond and DePaulo, 2006)
  - .. And does not improve after specific training (Levine et al, 2005).
- Well-known tools like the polygraph ('lie detector') are far less successful than we would expect

## Detecting deception using verbal clues

- The surest way to tell that somebody is telling a lie is knowing for sure that what the person is saying is not true
  - E.g., Jeffrey Archer telling journalists he was on the phone with the Prime Minister when one of the journalists knew that wasn't possible because the Prime Minister was delivering a speech at the time
  - In most successful trials for lying in court, the police knew for sure that a certain statement is false
- However, in most cases we have no such certain knowledge. It may still be possible however to tell whether somebody is lying purely on the basis of the style they are using Vrij (2008)
  - On the assumption that liars feel guilty, and such guilt may 'leak through' their speech
  - Or that telling a lie requires effort, so the liar may use a simplified form of language, more generic terms, etc.

## Deception detection using NLP methods

- Create a corpus of deceptive and non-deceptive text
- Train a classifier using the corpus
- Typically, classifiers use **STYLOMETRIC** features

## Stylometry

- Assumes that the essence of the **individual style** of an author can be captured with reference to a number of quantitative criteria, called **discriminators**
- Obviously, some (many) aspects of style are conscious and deliberate
  - as such they can be easily imitated and indeed often are
  - many famous pastiches, either humorous or as a sort of homage
- Computational stylometry is focused on **subconscious** elements of style less easy to imitate or falsify

## Basic methodologies

- Word or sentence length too obvious and easy to manipulate
- Frequencies of letter pairs strangely successful, though limited
- Distribution of words of a given length (in syllables), especially **relative frequencies**, ie length of gaps between words of same syllable length.

97/22

## How does it work? Linguistic Features

- ▶ Basic Measurements:
  - ▶ Average syllable/word/sentence count, letter distribution, punctuation.
- ▶ Lexical Density
  - ▶  $\text{Unique\_Words} / \text{Total\_Words}$
- ▶ Gunning-Fog Readability Index:
  - ▶  $0.4 * (\text{Average\_Sentence\_Length} + 100 * \text{Complex\_Word\_Ratio})$
  - ▶ Result: years of formal education required to read the text.

## Detecting deceptive Amazon Reviews

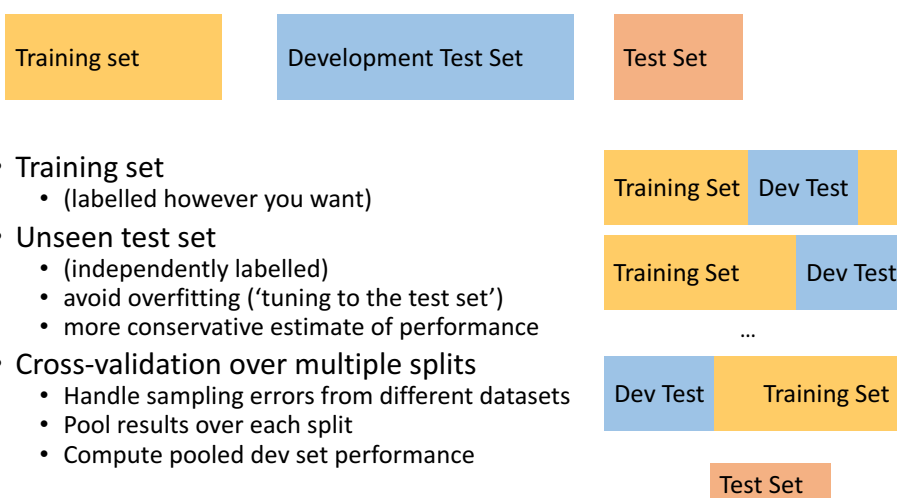
- Fornaciari and Poesio (2014)
  - Created dataset of Amazon reviews by collecting all reviews that were officially announced as being deceptive
  - Used unigrams, bigrams and trigrams as features
  - Selected using Information Gain
  - Using a SVM classifier
  - Performance: around  $F=.77$

## Practical tips

## Practical Text Classification

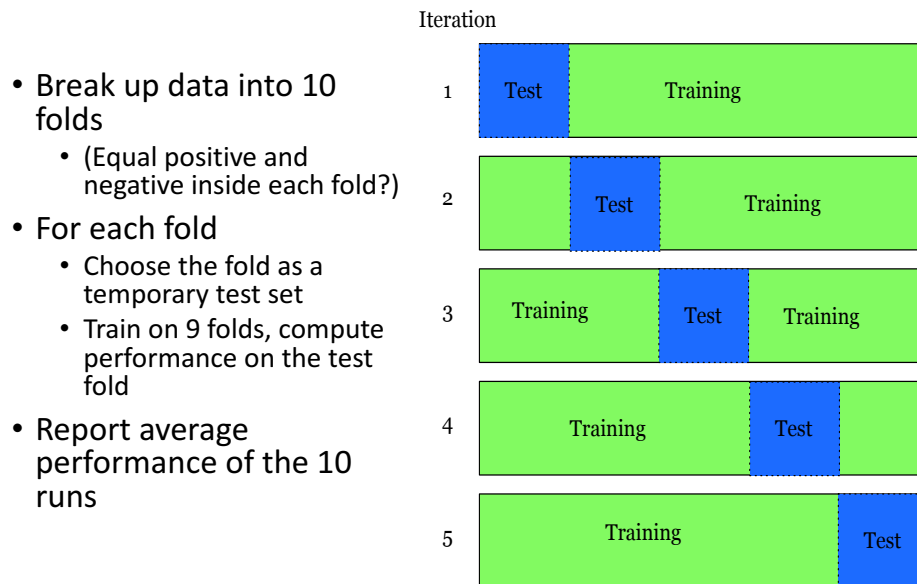
- For any text classification task, we must decide on (at least):
  - Classifier
  - Features
    - including pre-processing
  - Dataset
    - training & test
    - including labels
  - Evaluation

## Development Test Sets and Cross-validation





## Cross-Validation



## Avoiding over-fitting

- Always use a held-out test set
- Use regularisation / smoothing
- Don't use test set for parameter selection
  - (cost parameters, smoothing etc.)
  - Use cross-validation
  - Or a separate development set
- Avoid high dimensionality of feature set
  - $N \gg d$
  - You know about feature selection from Data Mining ...
    - Common: remove features with  $\text{freq} < 5$  (or 10 or  $N$  or ...)
    - Feature selection: don't use the test set!
      - (or the whole set, if cross-validating)

## REFERENCES

- Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002
- Yang and Pedersen. A comparative study of feature selection in text categorization. *ICML*, 1997.
- Manning, Raghavan, and Schuetze, *Introduction to Information Retrieval*. Cambridge, 2008.
  - Chapter 13, Text Classification and Naïve Bayes.
- Tom Mitchell, *Machine Learning*. McGraw-Hill, 1997.
  - Chapter 6, Bayesian Learning - 6.1, 6.2, 6.9, 6.10