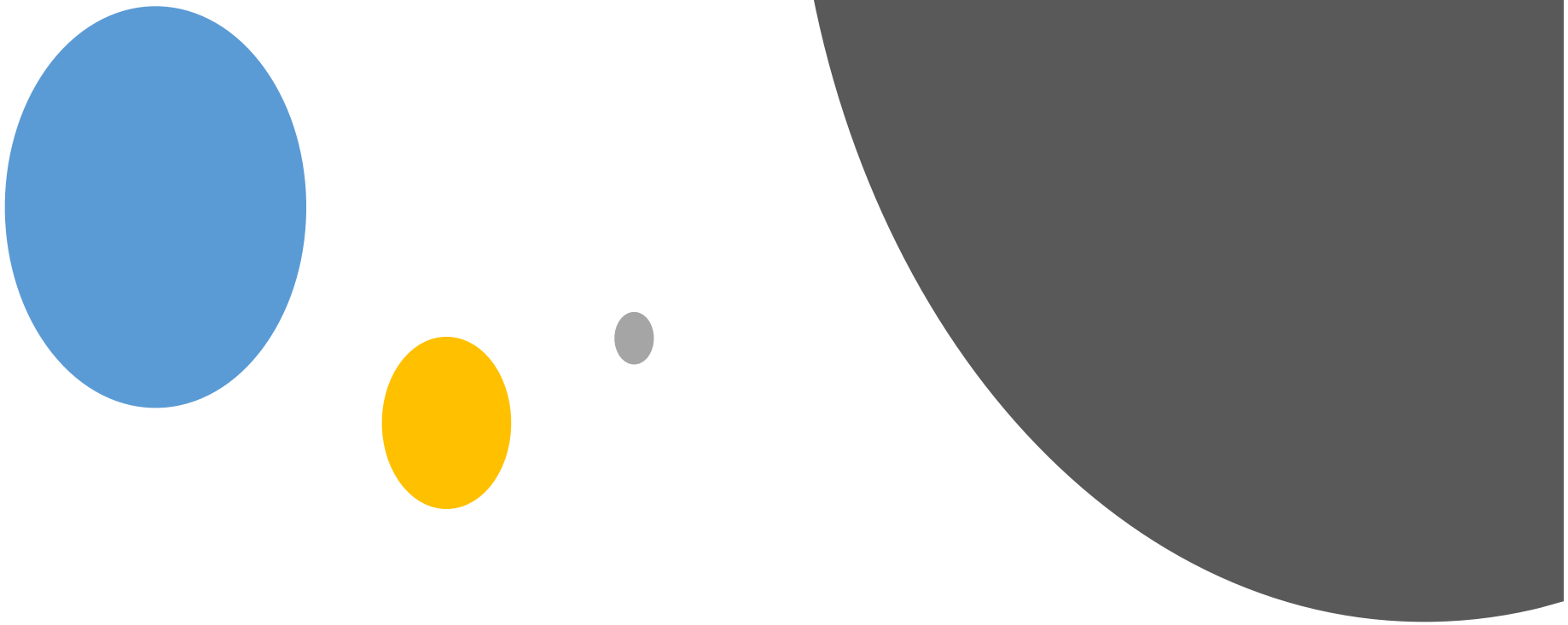


ECS763U/P Natural Language Processing

Julian Hough

Week 3: Text Classification

(with material from Massimo Poesio, Matthew Purver)

CONTENTS

CONTENTS

TEXT CLASSIFICATION

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
==========

Spam or not Spam?

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:

    $C = \{c_1, c_2, \ldots, c_n\}$
- 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 $C = \{c_1, c_2, \ldots, c_J\}$
- *Output*: a predicted class $c \in C$
- *Classifier:* a function $\gamma: d \rightarrow c$
- *Example*: spam filtering
  - $C = \{spam, not\_spam\}$

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

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

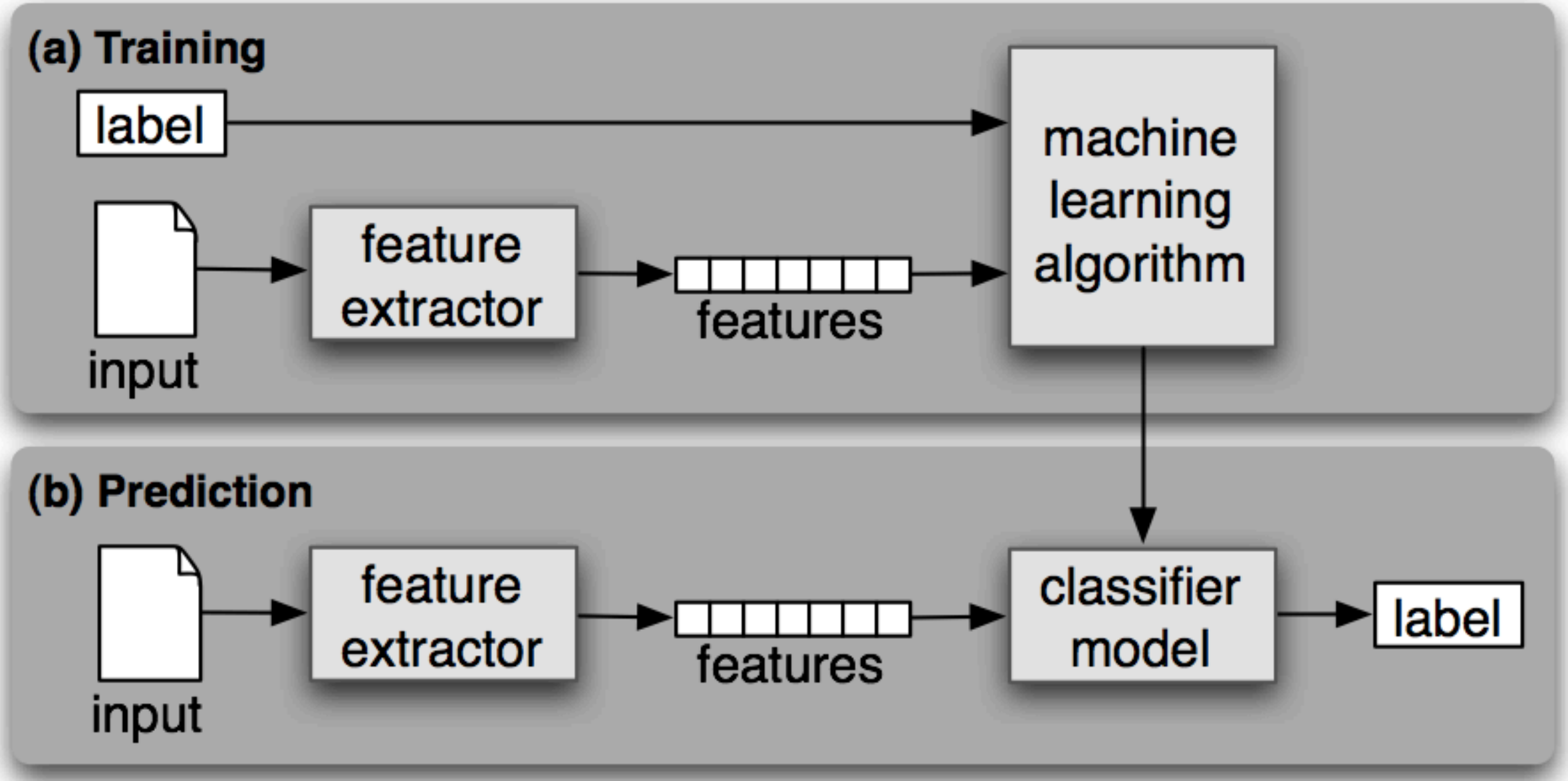
- *Output*: a learned classifier $\gamma:d \rightarrow c$

Supervised text classification

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

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

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


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

CONTENTS

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.

Product Rule and Bayes Rule

$$P(C, X) = P(C \mid X)P(X) = P(X \mid C)P(C)$$

$$\boxed{P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}}$$

Maximum a posteriori Hypothesis for classification

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid X)$$

Pick the value for c_j which gives the following equation the highest (Maximum) value

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \frac{P(X \mid c_j) P(c_j)}{P(X)}$$

Apply Bayes Rule

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(X \mid c_j) P(c_j)$$

Denominator not needed if only concerned with the arg max

Maximum likelihood Hypothesis

If all hypotheses are a priori equally likely, we only need to consider the $P(\mathrm{X}|c_j)$ term:

$$c_{ML} = \underset{c_j \in C}{\operatorname{argmax}} P(X \mid c_j)$$

Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values $\langle x_1, x_2, \ldots, x_n \rangle$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$c_{NB} = c_{ML} = \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

Naïve Bayes

- But how can we calculate $P(x_1,x_2 \ldots x_n|c)$?
  - (how often will we see it?)
- Instead, assume, **naively**, that:
  - features are unordered words ("bag of words")
  - features are conditionally independent

$$P(x_1,\ldots,x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

- So:

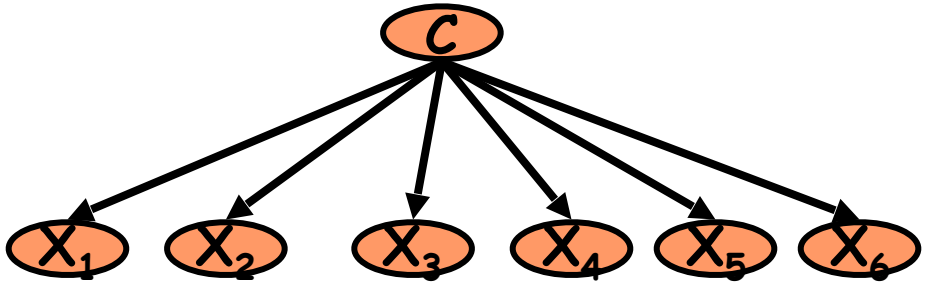
$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \log(P(c_j)) + \sum_{i \in positions} \log(P(x_i \mid c_j))$$

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

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

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

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	0.66	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?`

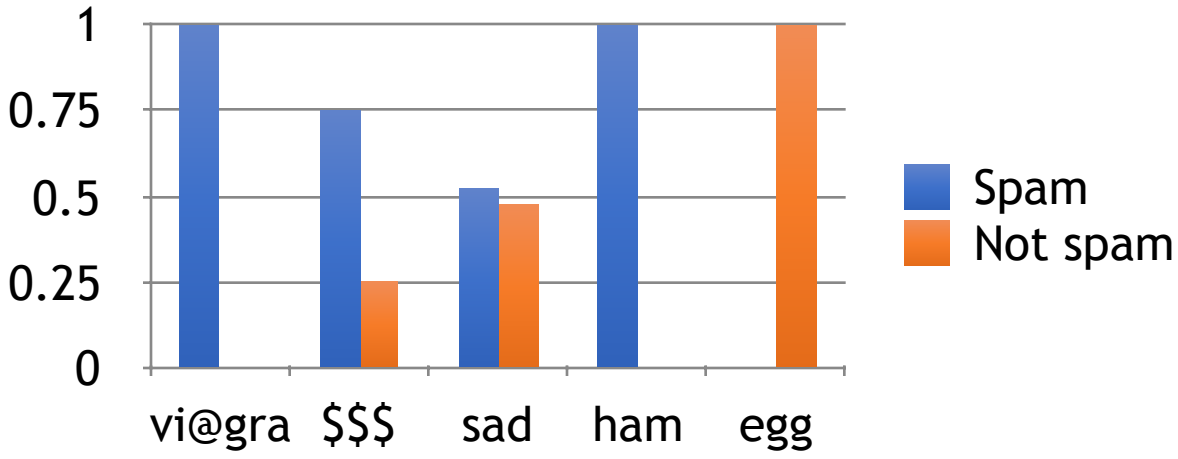
Problem with the first approximation

- But what if we haven't seen "viagra" with c = not_spam?

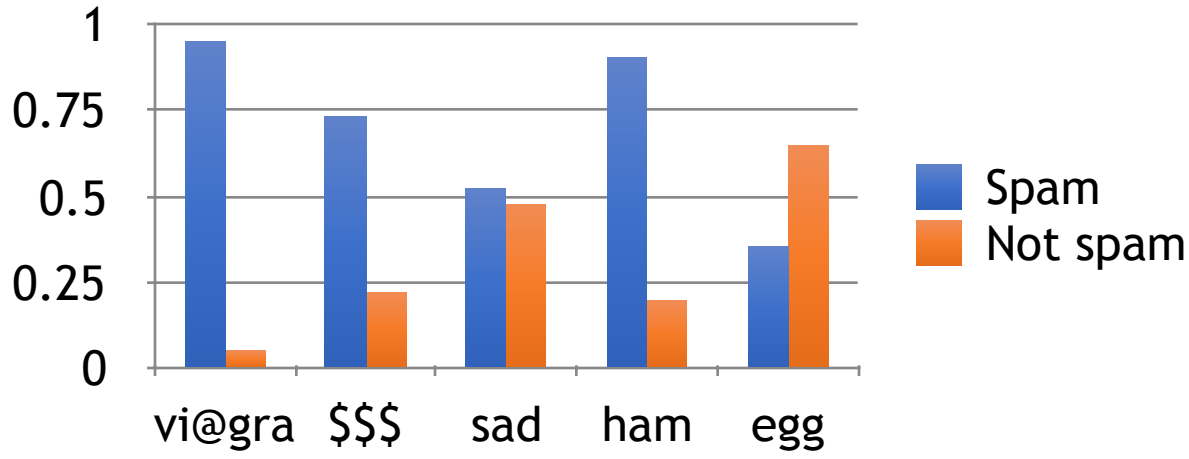
$$\hat{P}(w_i \mid c_j) = 0!$$

Solution: smoothing

- Need to reserve some probability mass for unobserved events



Smoothed:



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

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

Text Classification Algorithms: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ subset of documents for which the target class is c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

$Text_j \leftarrow$ single document containing all $docs_j$
For each word x_k in *Vocabulary* do:

$n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k \mid c_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|}$$

Text Classification Algorithms: Classifying

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

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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.

CONTENTS

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?**

Precision and recall

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

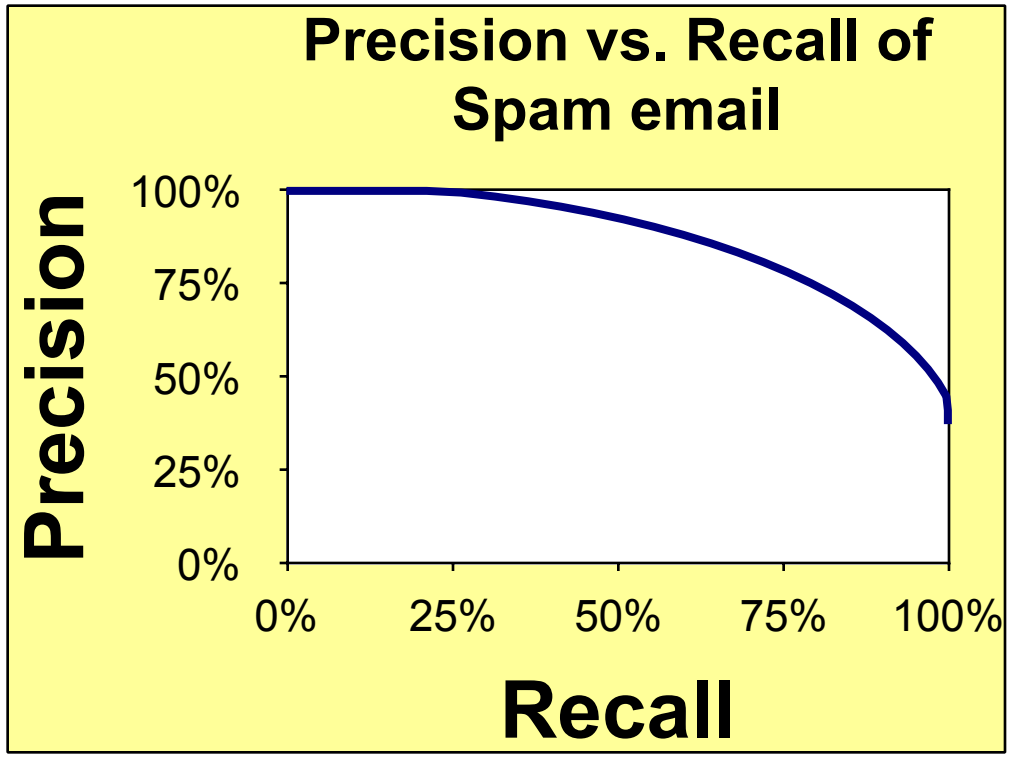
	correct	not correct
Predicted	tp	fp
Not predicted	fn	tn

P = tp/(tp+fp)

R = tp/(tp+fn)

tp: true positive
fp: false positive
fn: false negative

Precision and recall



- Precision = $\frac{\text{\#spam correctly predicted}}{\text{\#spam predicted}}$
- Recall = $\frac{\text{\#spam correctly predicted}}{\text{\#actual spam messages}}$

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

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, $\alpha = ½$):

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

- General formula:

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

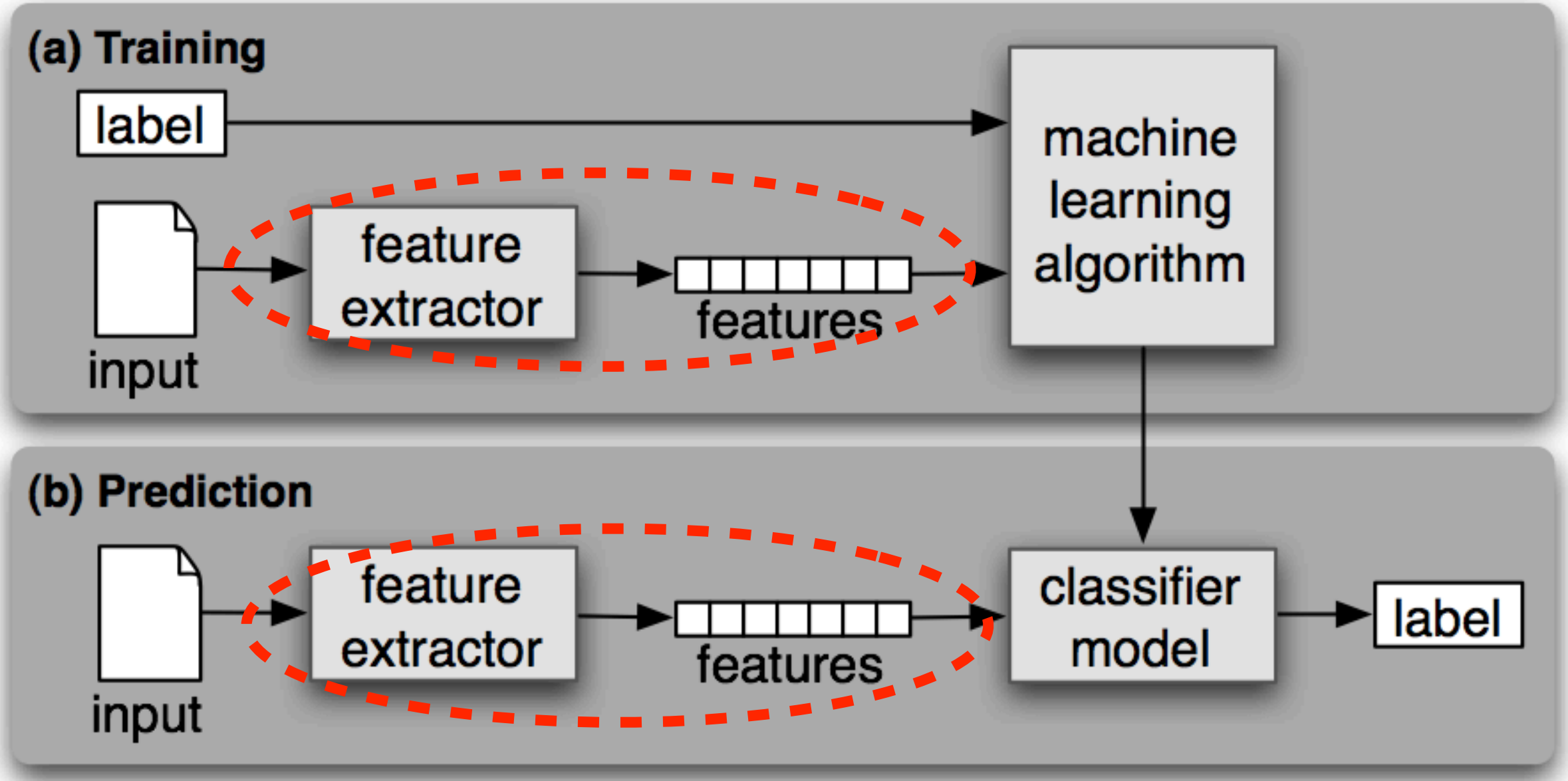
Why not simple (arithmetic) mean?

- 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)

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
  - **Notebook 'text_categorization_sklearn' in QMPlus 'Text Classification' section**
- 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/

CONTENTS


(a) Training
label
machine learning algorithm
feature extractor
features
input
(b) Prediction
feature extractor
features
classifier model
label
input

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。

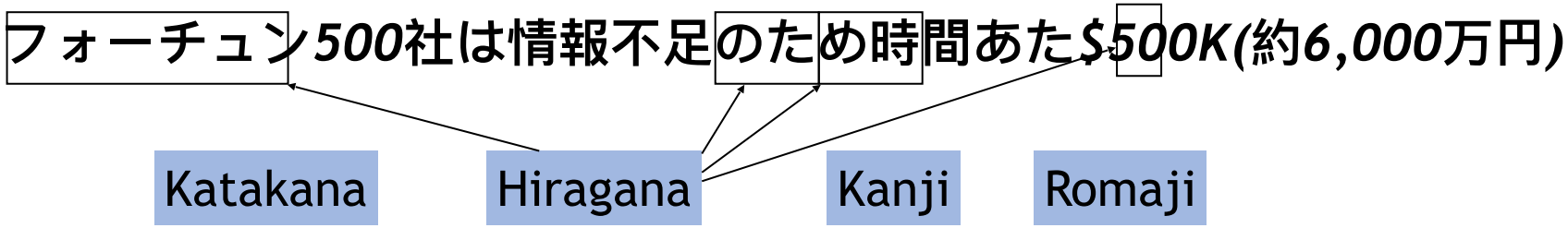
  莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

  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

Tokenisation

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



End-user can express query entirely in hiragana!

- How about German?
  - ‘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
    - Tranposing 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 → 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 → ø	cats	→	cat

Step 1b

(*v*)ing → ø	walking	→	walk
	sing	→	sing
(*v*)ed → ø	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 → ø	revival	→	reviv
able → ø	adjustable	→	adjust
ate → ø	activate	→	activ
…			

Why check for vowels in Step 1b?

(*v*)ing → ø	walking	→	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 Information Theoretic measures

- We need notions from **Information Theory** (Shannon 1948) for effective feature selection:

  - **Mutual Information**
  - **Information Gain**
  - **Entropy**

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 ((P)MI)**

(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

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.

Game on locked-in syndrome sufferer communicating: (Paul Curzon)

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

p*N belong to class Spam (positive)

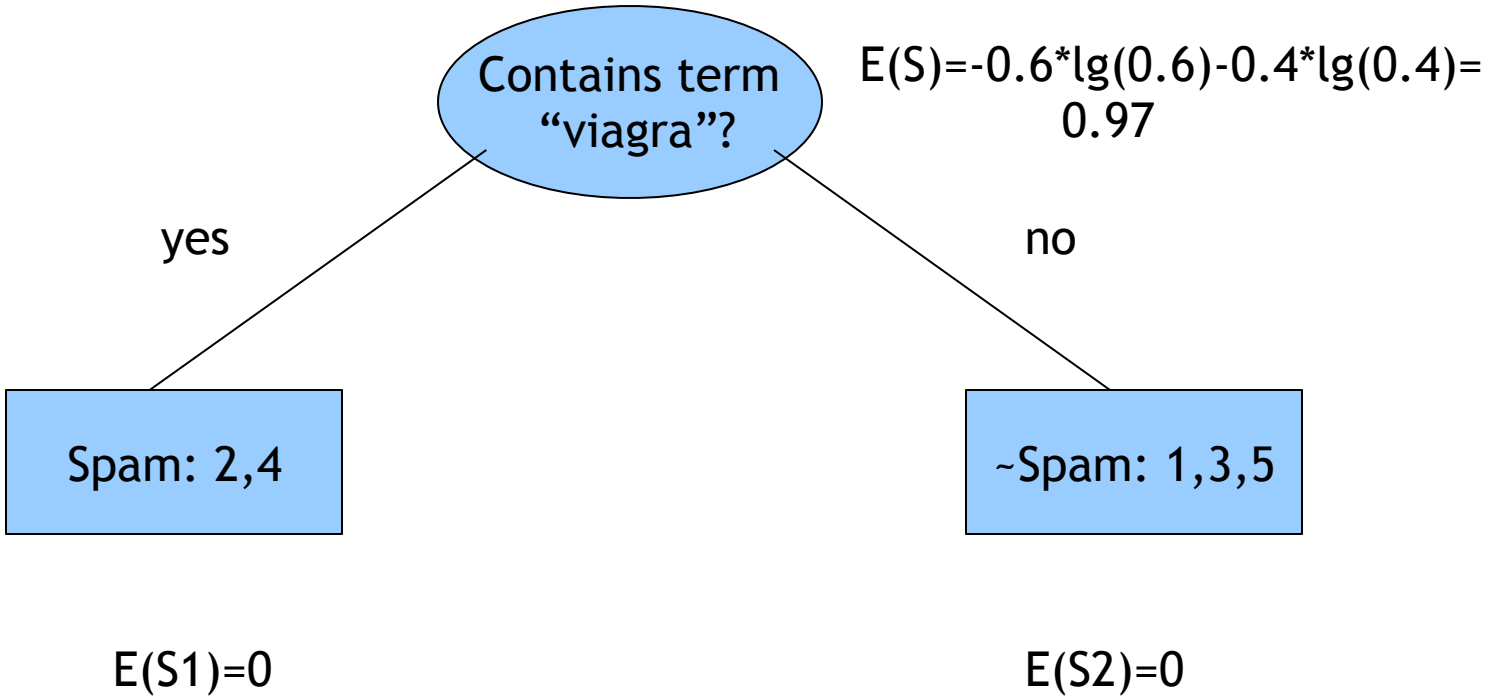
q*N negative (do NOT belong to class Spam)

Entropy(S)= $-p*log(p) – q*log(q)$

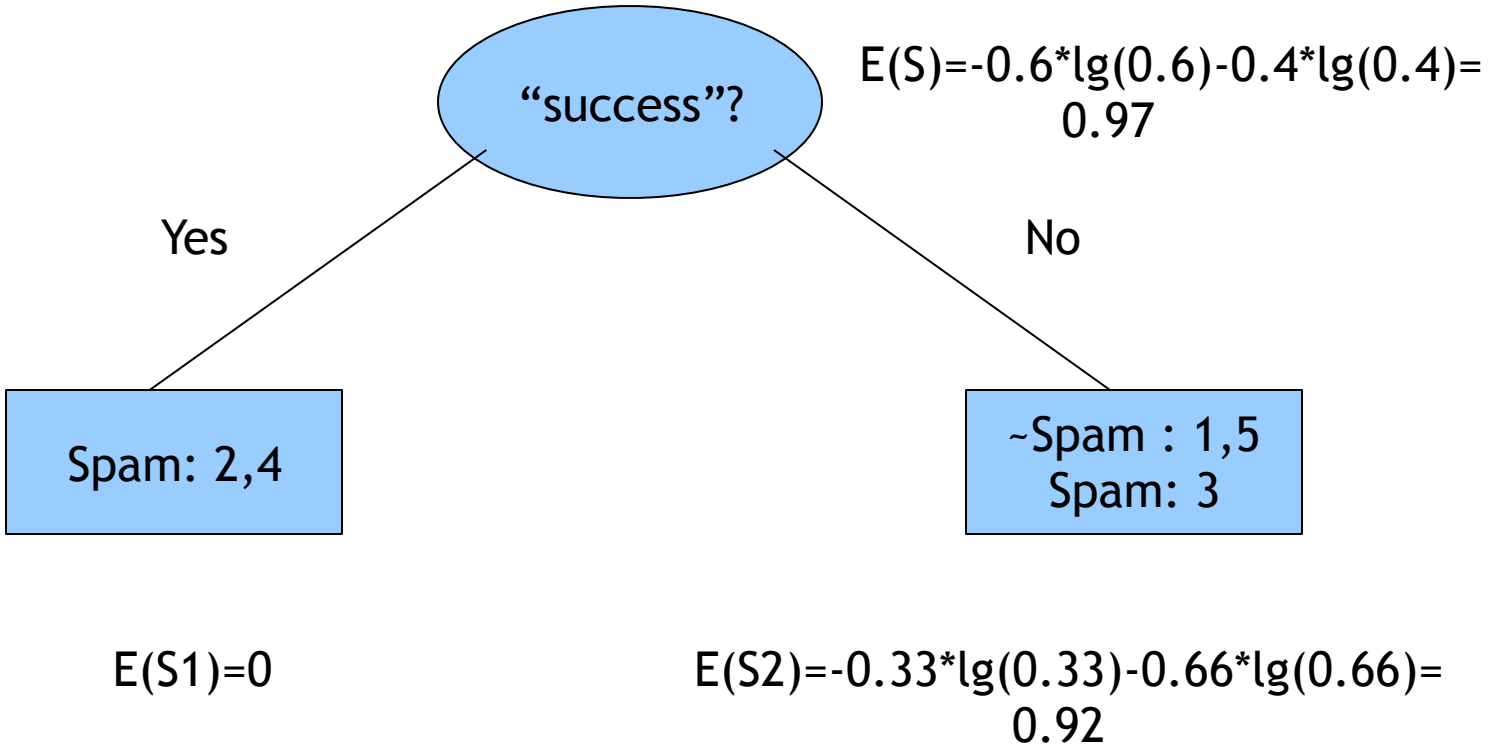
p=1, q=0 => Entropy(S)=0

p=1/2, q=1/2 => 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:

$$Gain(S,t)=E(S)-\sum(Entropy(Si)*|Si|/|S|)$$

t=viagra? : Gain(S,f)=0.97

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

t=clothing?: 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

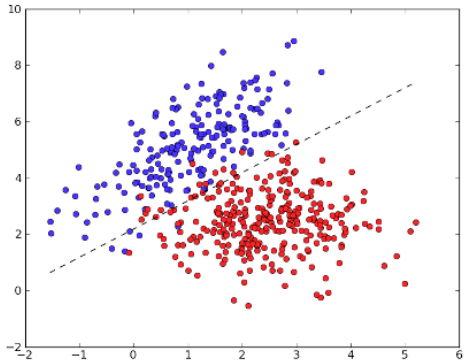
- **Back to the Notebook**

CONTENTS

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

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



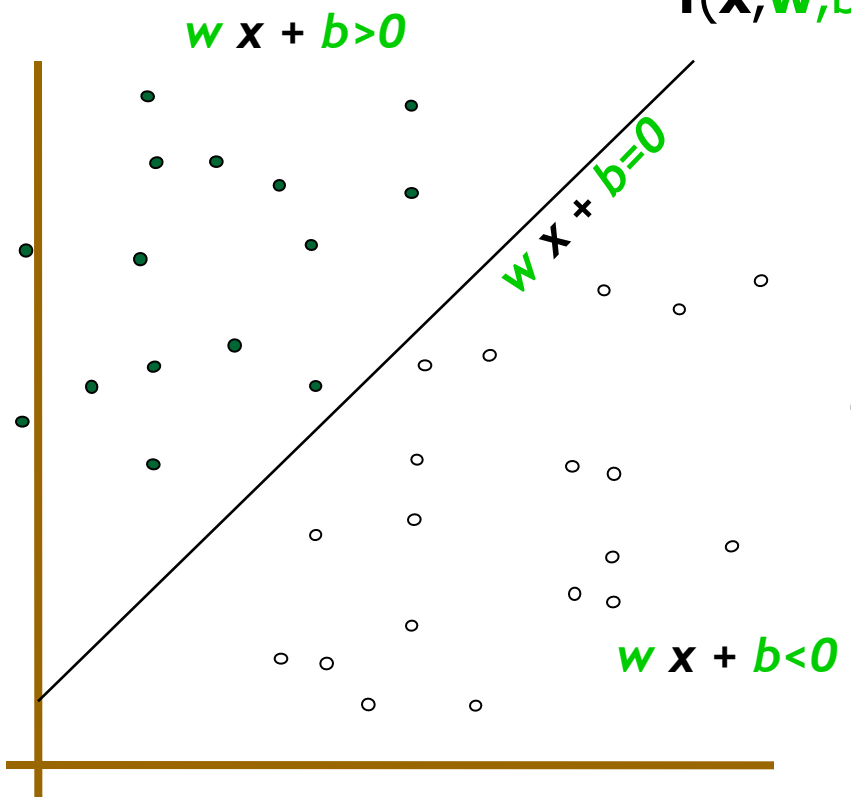
- Why not:
  - estimate the boundary directly?
  - use a discriminative model?

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

Linear Classifiers

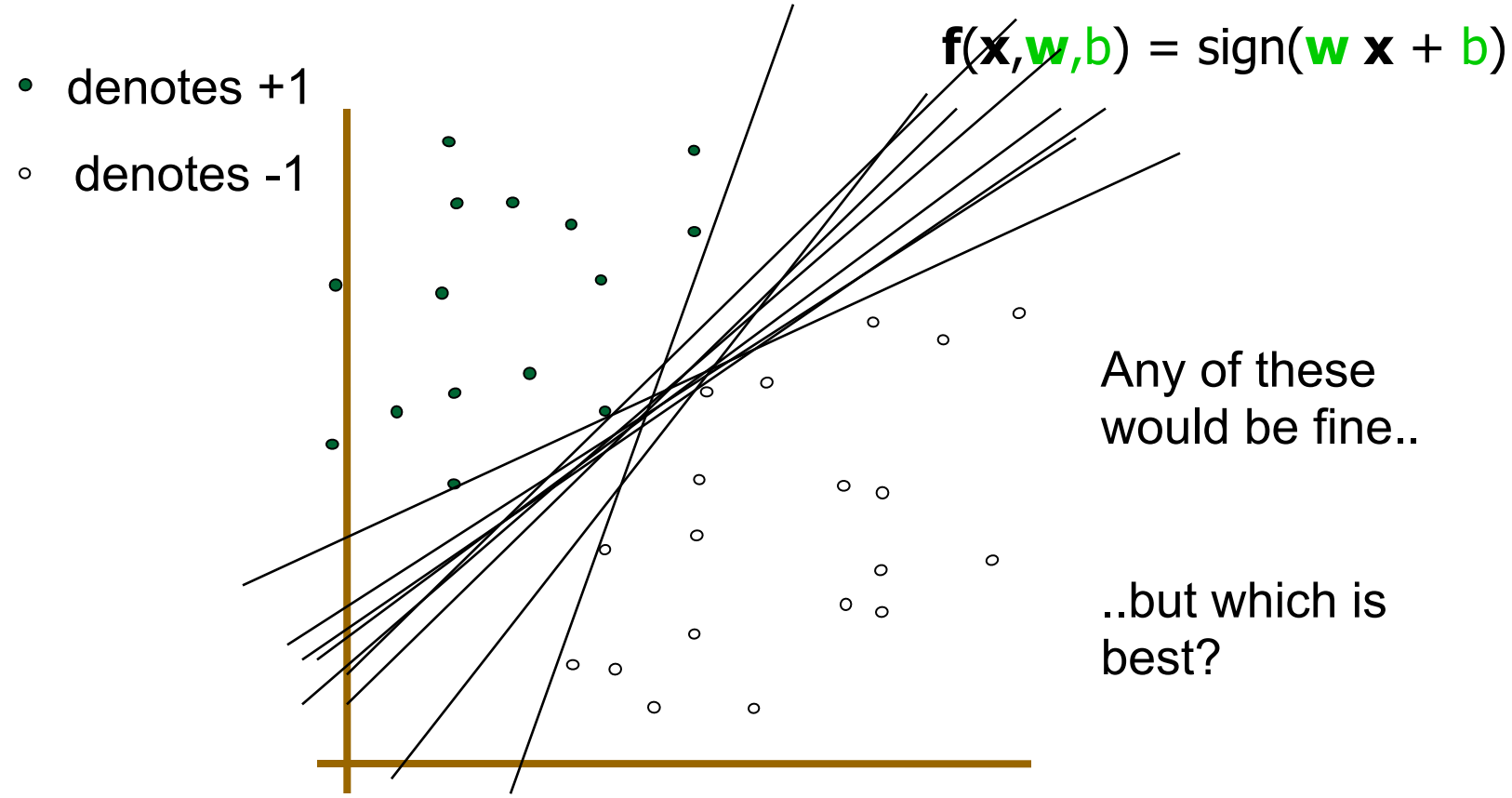
- denotes +1
- denotes -1

$\mathbf{f}(\mathbf{x},\mathbf{w},b) = \text{sign}(\mathbf{w}\,\mathbf{x} + b)$



How would you classify this data?

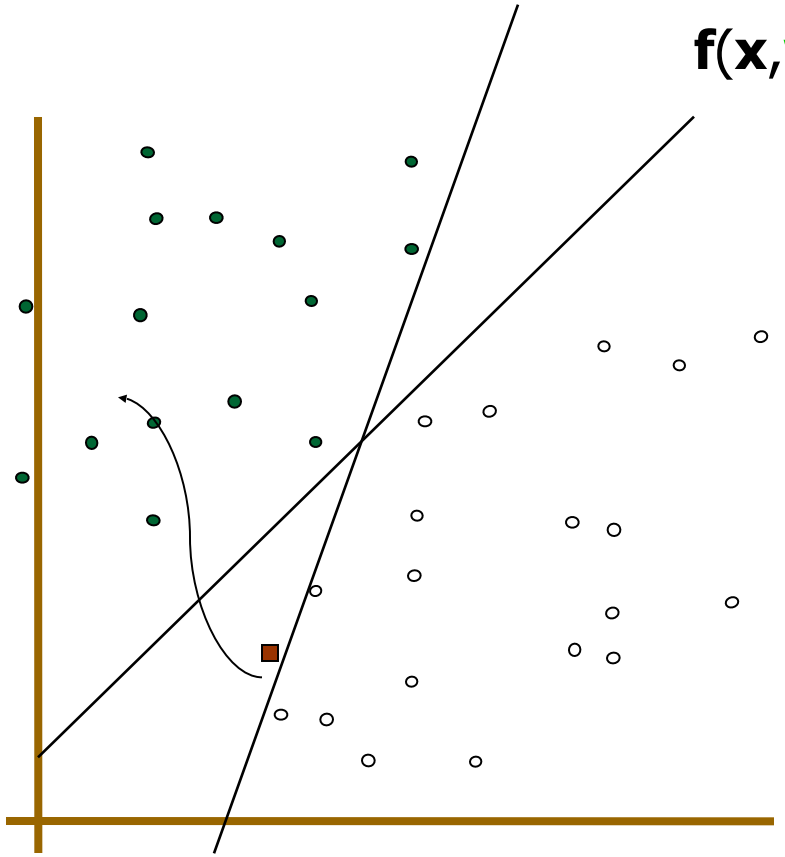
Linear Classifiers



Linear Classifiers

- denotes +1
- denotes -1

$\mathbf{f}(\mathbf{x},\mathbf{w},b) = \sin(\mathbf{w}\,\mathbf{x} + b)$



How would you classify this data?

Misclassified to +1 class

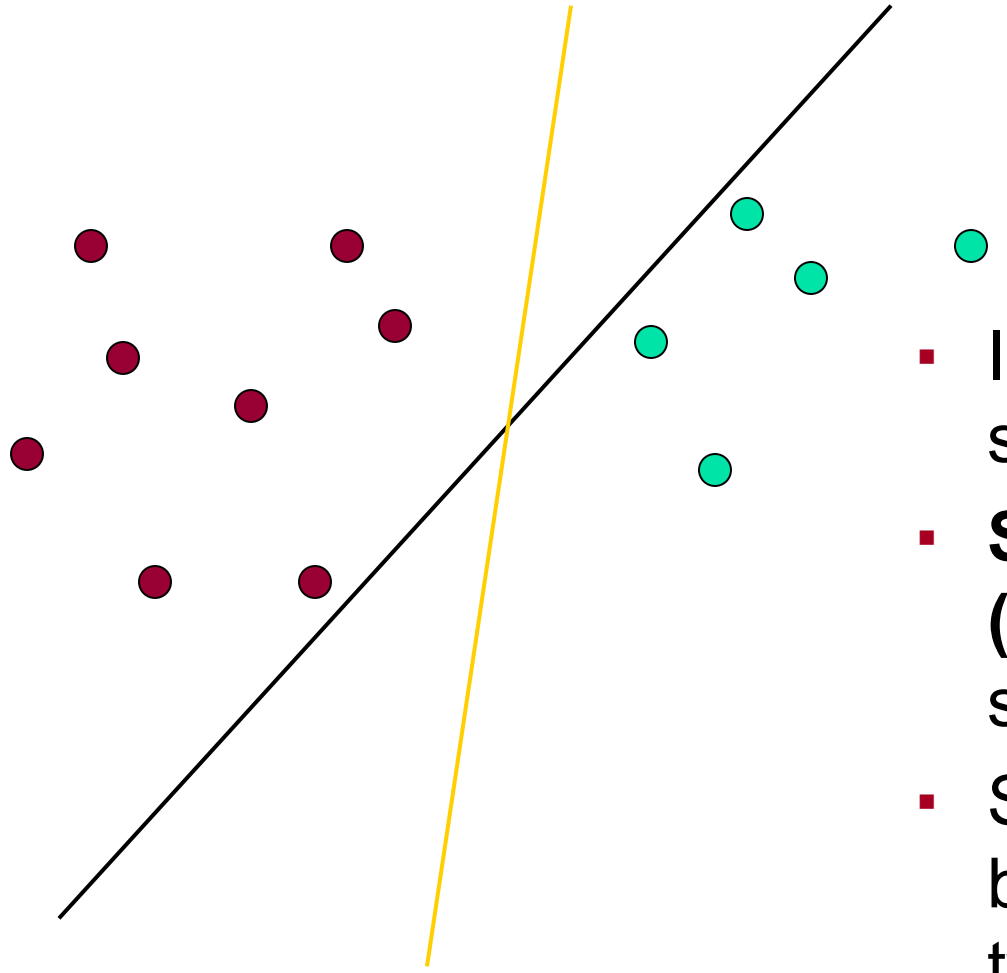
Linear Classifiers: summary

- 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 $ax + by = c$

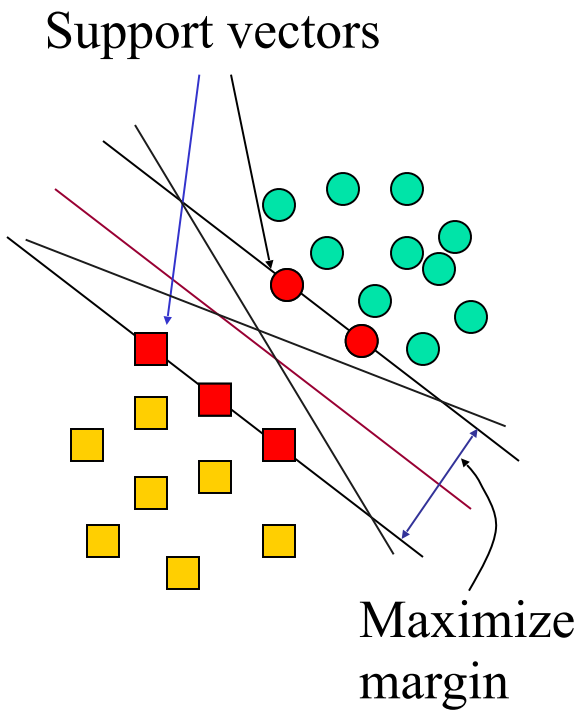
Which Hyperplane?



- In general, lots of possible solutions for *a,b,c*.
- **Support Vector Machine (SVM)** finds an optimal solution.
- SVMs answer: which line best splits the data into the two groups?

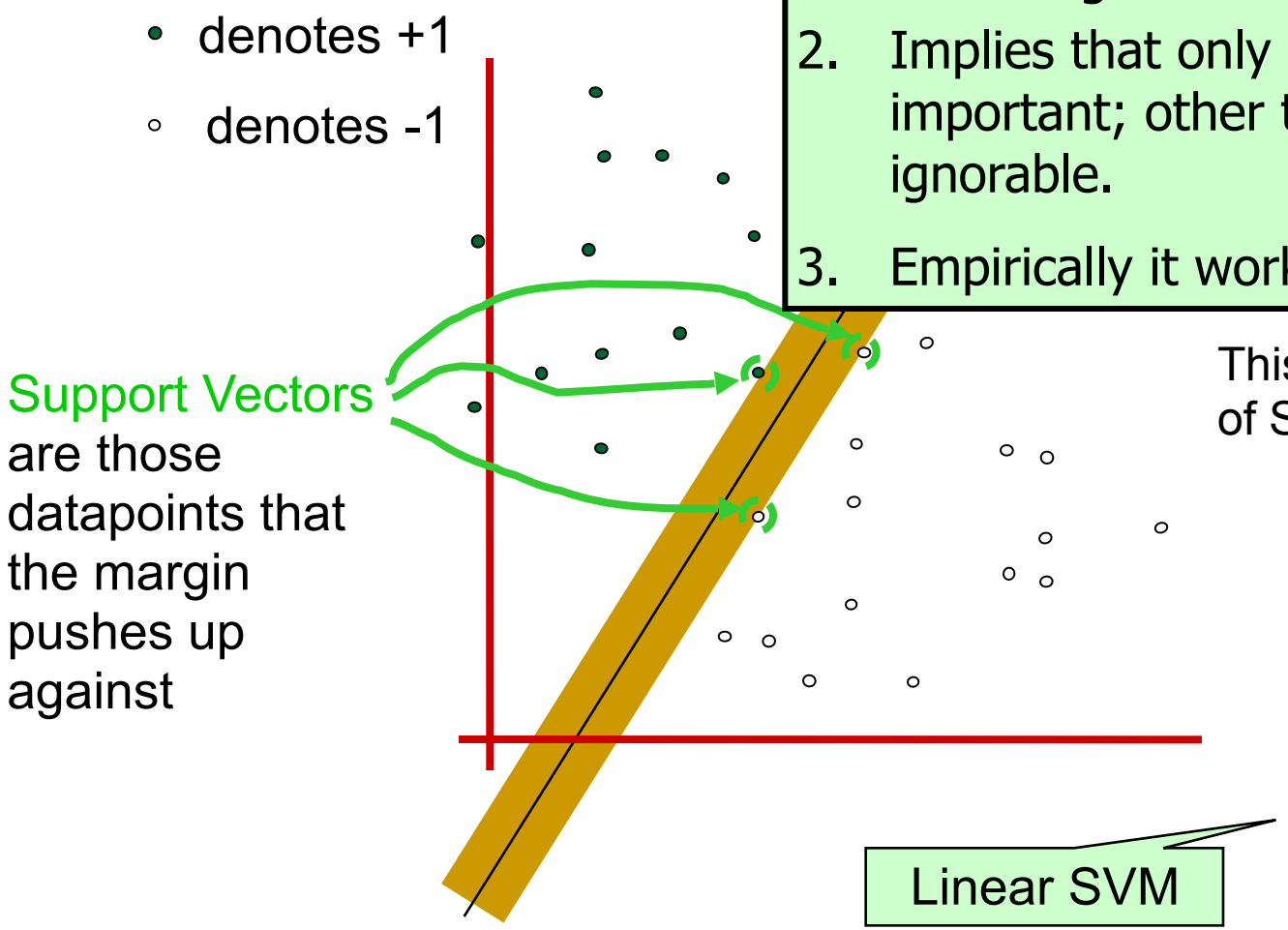
Support Vector Machine (SVM)

- SVMs **maximize the margin** around the separating hyperplane.
  - The distance between the points and the line are is as far as possible.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- **Constraint optimisation problem:**
  - Optimisation is maximise the margin
  - Constraint is that the data points can’t be on the margins.



Maximum Margin

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.



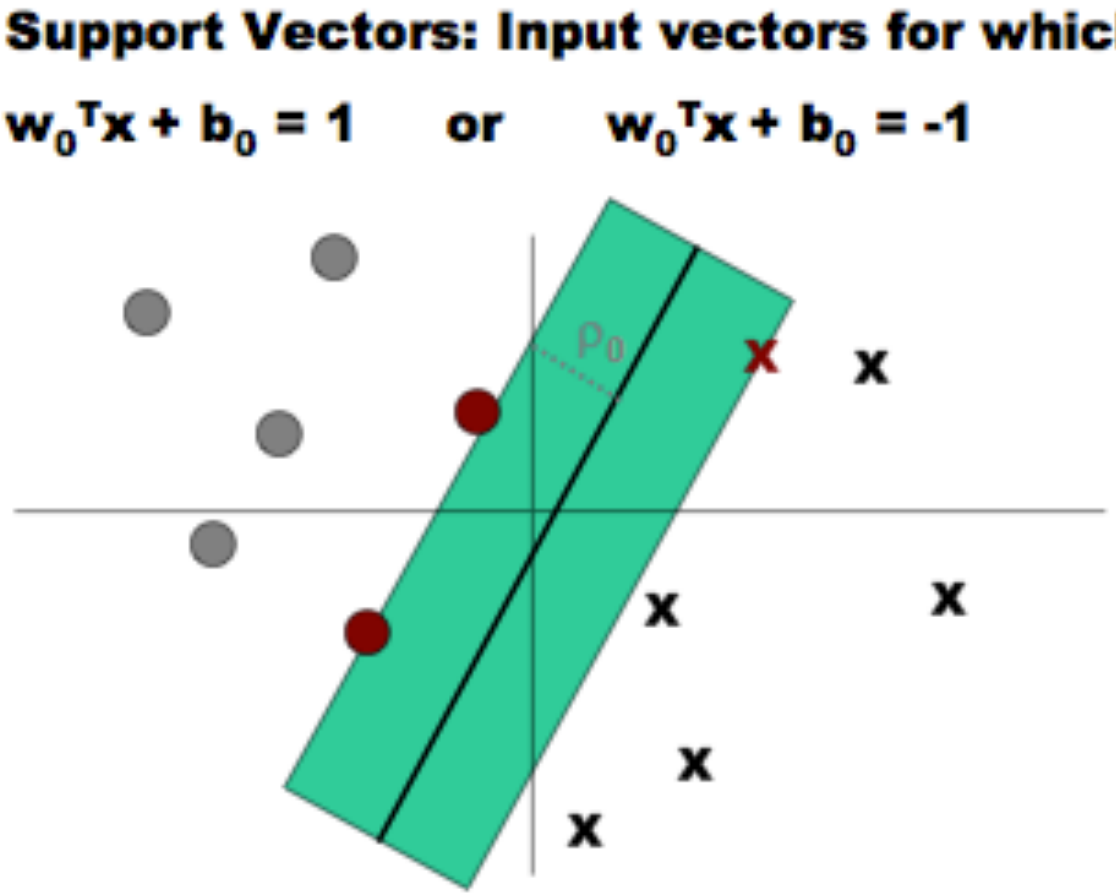
Maximum Margin: Formulation

- w: hyperplane normal vector
- x_i: data point i
- y_i: class of data point i (+1 or -1)

- Constraint optimization formalization:

- (1) $$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1$$
  $$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1$$
- (2) maximize margin: 2/||w|| (i.e. distance of point from hyperplane)

Maximum Margin in geometrical terms



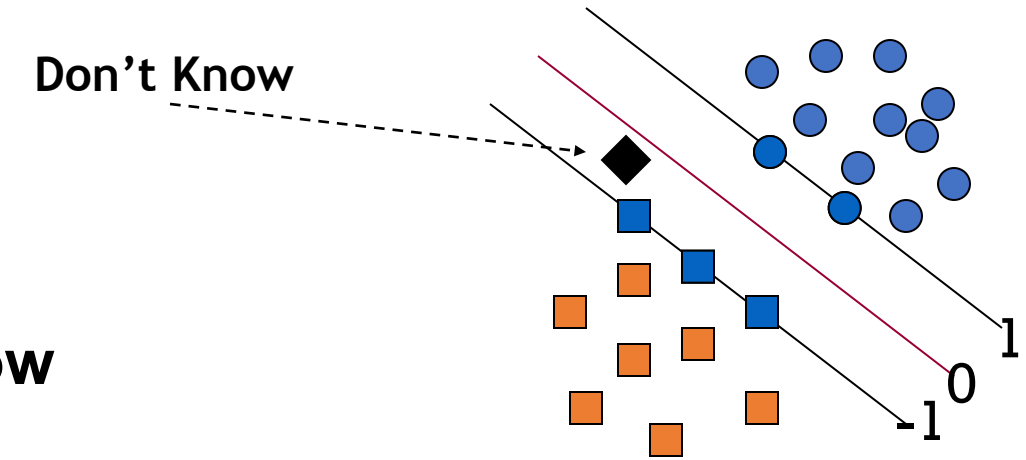
Classification with SVMs

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

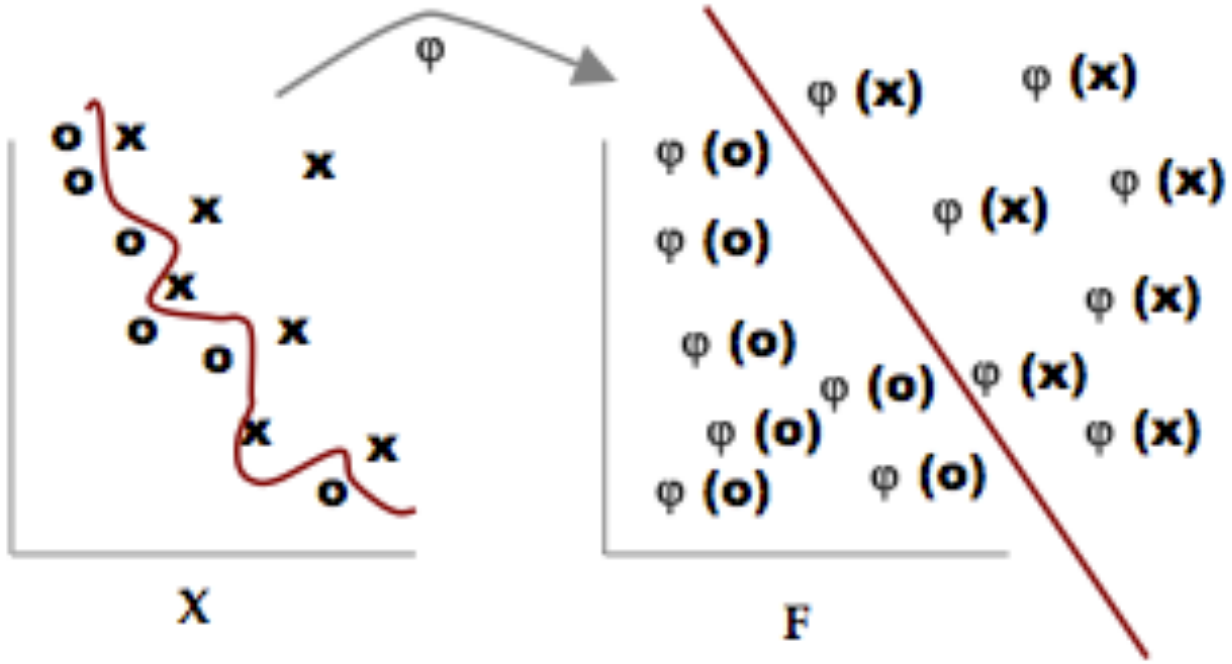
Score > t: **yes**

Score < -t: **no**

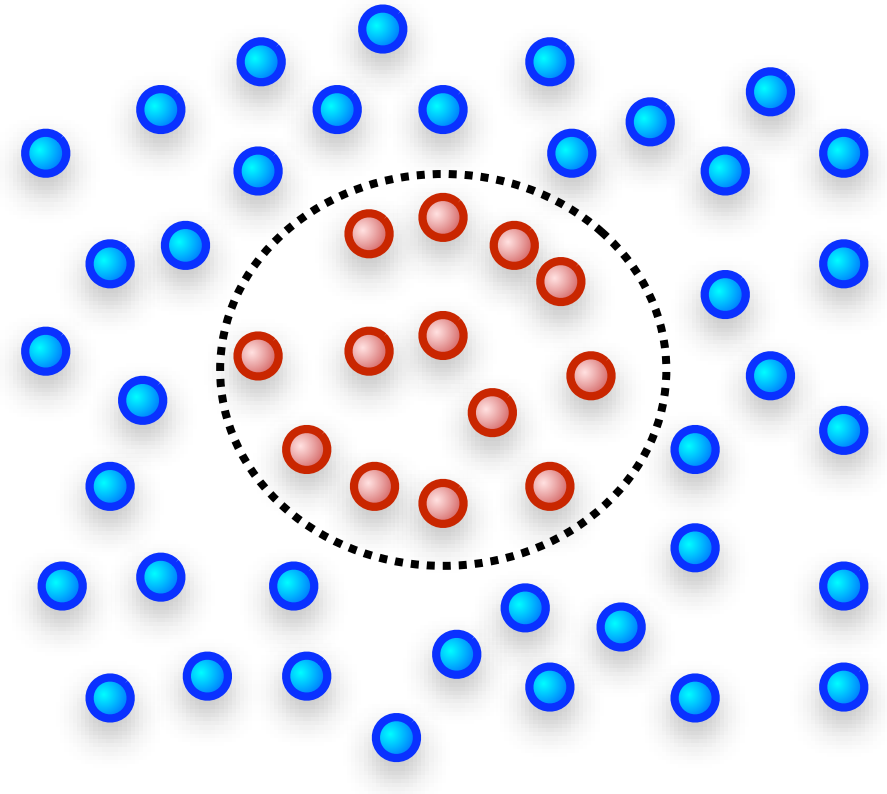
Else: **don't know**



Non-linear mapping can make a non-linear problem linearly separable



Non-linear mapping can make a non-linear problem linearly separable



Kernals

Kernals

- Map each point into a new space:

$$x_i \longmapsto \Phi(x_i)$$

- Making the constraint optimization now:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

$$\min_{\alpha} \quad \sum_{i,j} \alpha_i \alpha_j \, y_i y_j \, K(x_i, x_j)$$

$$s.t. \qquad \alpha_i \geq 0 \qquad \forall i$$

$$\sum_{I_+} \alpha_i = 1$$

$$\sum_{I_-} \alpha_i = 1$$

SVMs vs other approaches: Classic Reuters-21578 Data Set

- Most (over)used data set.
- 21578 documents:
  - 9603 training, 3299 test articles
- 118 categories.
  - An article can be in more than one category.
  - Learn 118 binary category distinctions.
- Average document: about 90 types, 200 tokens.
- Quite sparse problem: only about 10 out of 118 categories are large.

Common categories
(#train, #test)

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

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>

(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

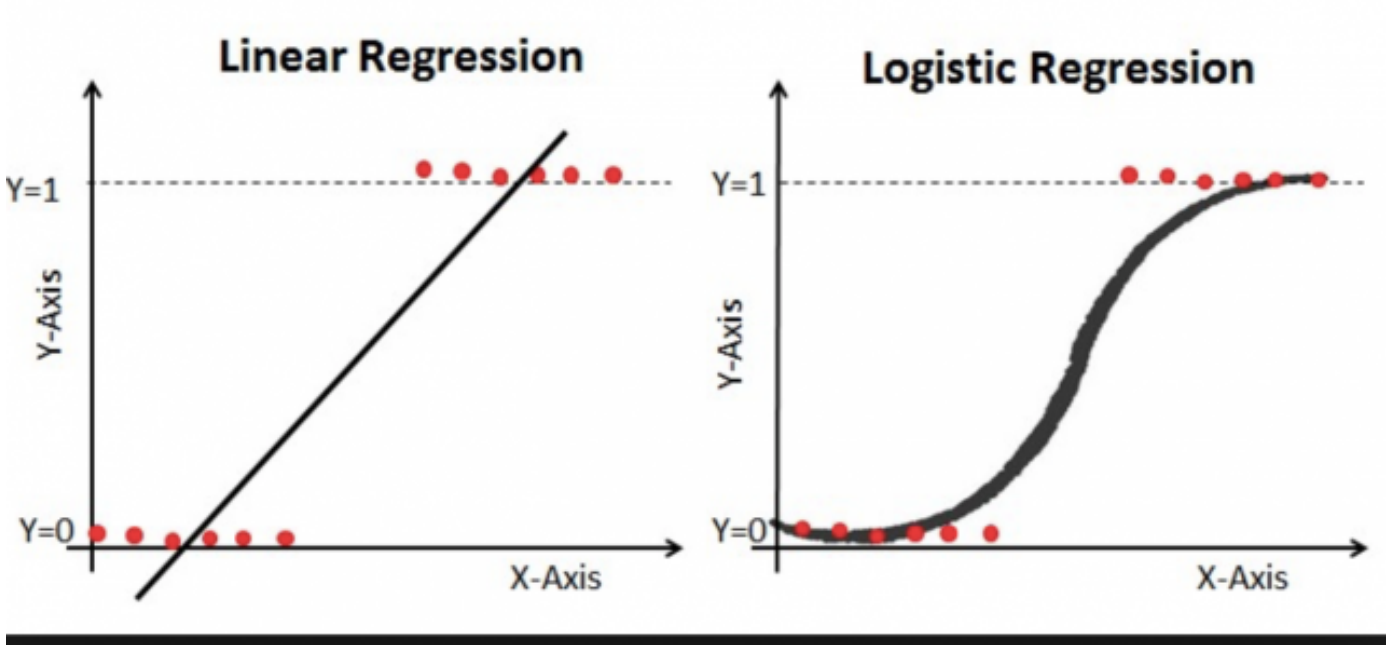
Evaluation measure: F_1

Text classification in Python

- **Back to the Notebook: SVMs in scikit-learn**

Logistic Regression

- While most problems are linearly separable after transforming the space, you can also use **nonlinear classifier** like **Logistic Regression**.



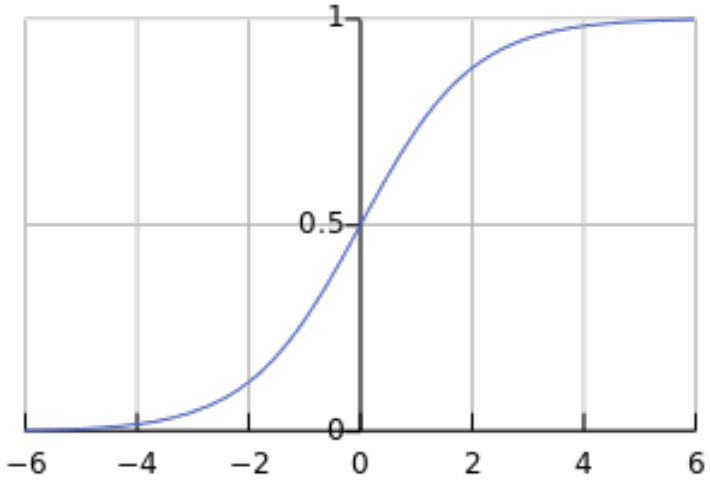
Logistic Regression

- Output is a linear combination of features
  - i.e. weight matrix *w*, feature vector *x*
- … with exponential to make it positive ...
- … and normalised to [0-1]

$$p(y \mid x) = \sum_i w_i . x_i$$

$$p(y \mid x) = \vec{w} . \vec{x}$$

$$p(y \mid x) = \frac{1}{1 + e^{-\vec{w} . \vec{x}}}$$



- Training task is to learn the optimal *w*
- *More next term in Deep Learning + NLP.*

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.

Linguistic features

- Basic Measurements:
  - Average syllable/word/sentence count, letter distribution, punctuation.
- Lexical Density
  - *Unique_Words / Total_Words*
- Gunning-Fog Readability Index:
  - *0.4 * (Average_Sentence_Length + 100 * 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

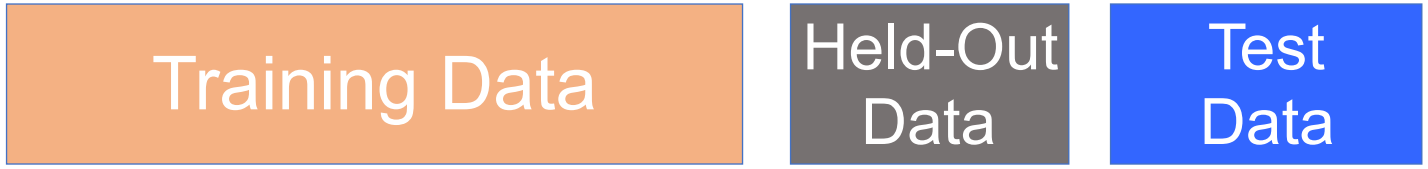
- **This is what your task is for Lab 2.**

CONTENTS

Practical Text Classification

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

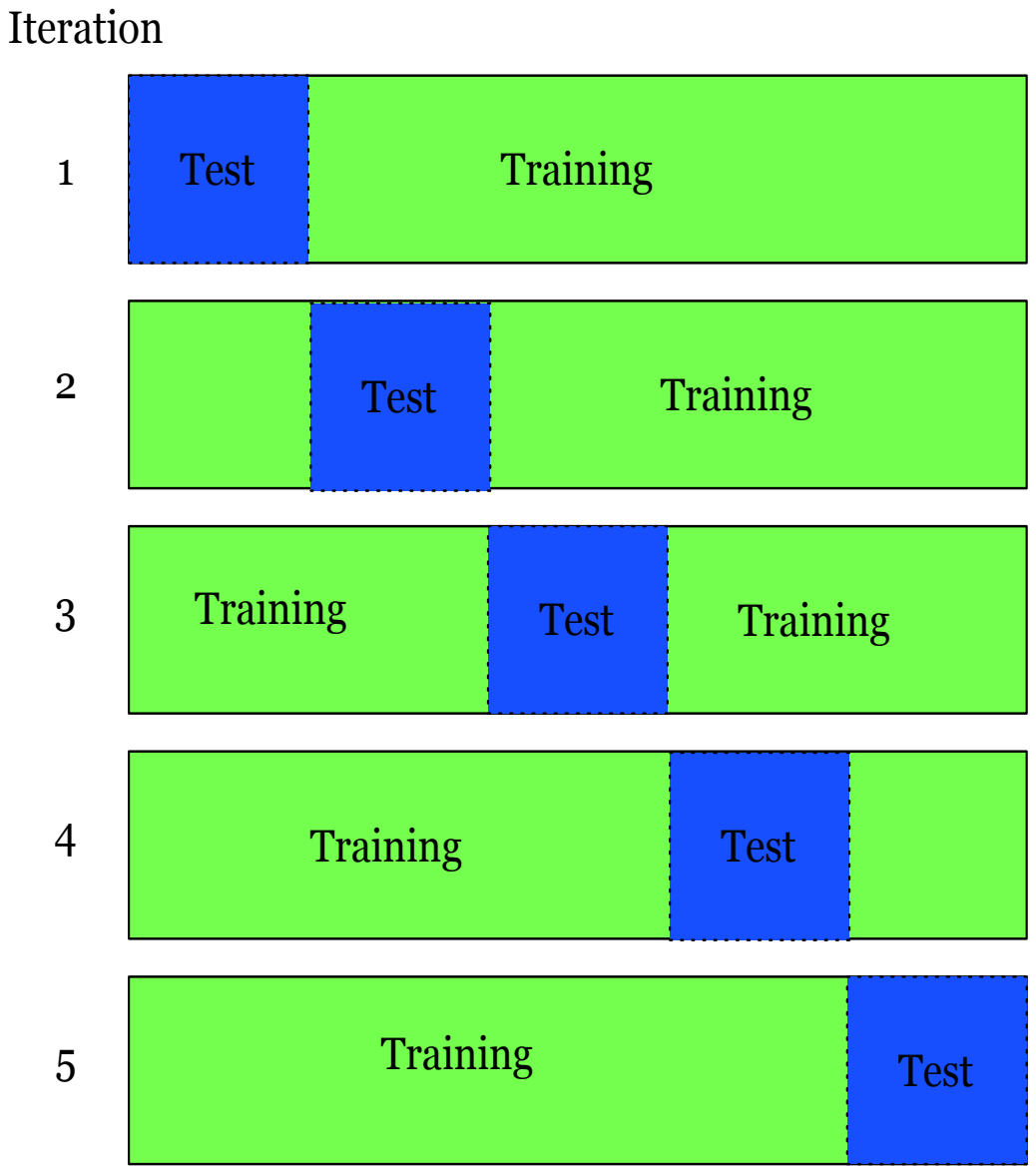
Heldout/Development Test Sets and Cross-validation



- Training set
  - (labelled however you want).
- Unseen test set
  - (independently labelled).
  - avoid overfitting (‘tuning to the test set’).
  - more conservative estimate of performance.
- Cross-validation over multiple splits
  - Handle sampling errors from different datasets.
  - Pool results over each split.
  - Compute pooled test set performance.

Cross-Validation

- For **k-fold cross-validation**, break up data into k folds.
  - (Equal positive and negative inside each fold?)
- For each fold
  - Choose the fold as a temporary test set.
  - Train on k-1 folds, compute performance on the test fold.
- Report **average performance** of the k runs.



Avoiding over-fitting

- Always use separate test set in development (don’t test on training!)
- Use smoothing/regularisation.
- 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 > d
  - Remember: techniques for feature reduction in include minimum document frequency, using information gain and other measures to select most useful features.

Summary

- Classification is the assignment of categories (or classes) to instances of data, and in ML classification this uses features of the instance to achieve this. There are generative and discriminative models.
- Text classification uses lexical features (those derived from the words), but not necessarily just plain word values.
- Feature reduction can be achieved pre-processing steps like stemming and spelling correction and also through various information-theoretic tests of the utility of features.
- **In the lab exercise you will fill in some missing methods for some off-the-shelf classifiers to do fake review detection.**
- **Hand-in Friday Week 5 (with Lab exercise 1)**

READING

- Jurafsky and Martin (3rd Ed)
  - Chapter 4 (Naïve Bayes and Classification Evaluation measures)
  - Chapter 5 (Logistic Regression)
- 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

READING (EXTRA)

- 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.