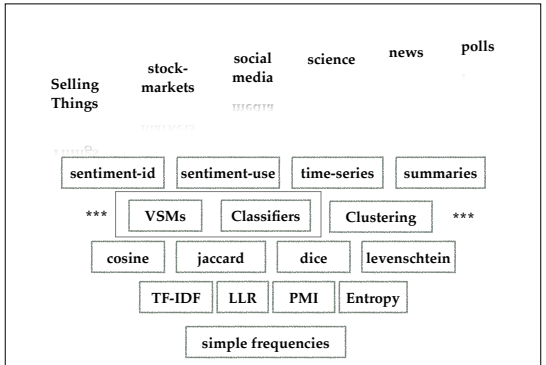
Classification

Lecture 6: Text Analytics for Big Data
Mark Keane, Insight/CSI, UCD



Outline

- Why Do This?
- Classification (using Supervised Techniques)
- A Tangent on Cross Validation
- Next week Clustering…

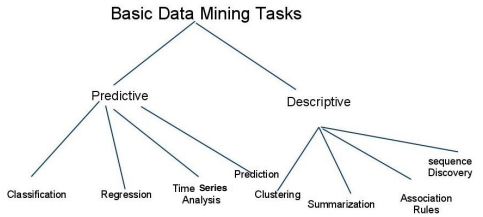
Classification

Why We Do This?

To Classify…

- Classification is used in text analytics to identify an item as a member of a class and/or to make predictions about that item
- Is article about sports or business; is this email spam or non-spam; is this buyer profile a binge-shopper
- If X is a member of class, then I can predict other features; if Mark is a binge-shopper he will need periodic, expanded credit-limits and targeted adverts

One View of Tasks…

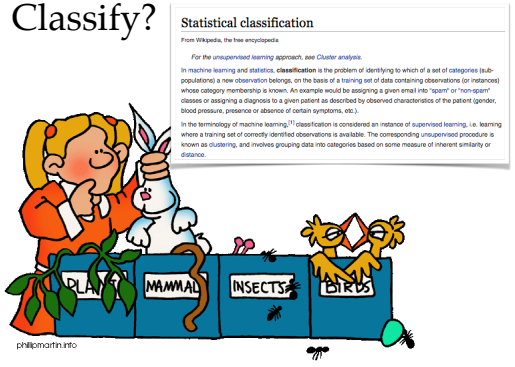


http://www.slideshare.net/KapilRode/data-mining-32694954

Classification is Ubiquitous

- We have seen similarity, as a way to compare individual items to one another (shopper1 is like shopper 2)
- Sometimes you have a class/category; so you can say item-X is a category-Y; shopper1 is a BingeShopper or shopper2 is a YoungFamilyShopper
- Y is a general class of things; BingeShopper is a class
- Many ML techniques create **classifiers**; programs that can accurately classify items as being in one class or another

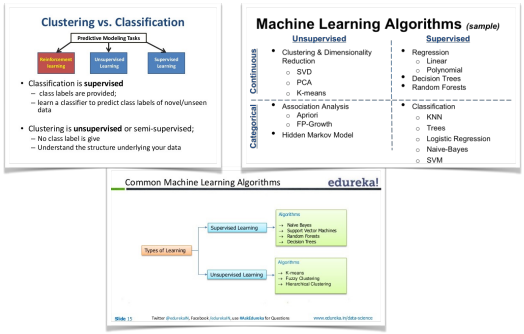
Classify?



Classification Examples

- Based on activity and profiles Tweets have three classes: lurkers, active-tweeters and celebrity-tweeters
- Company reports can be classified as negative or positive and this predicts share prices
- Crime-readers buy books by Chandler with "blood" in the title; use this to recommend others
- Tweeters can be classified as right-wing or left-wing by hashtags, this allows us predict voting preferences

Classification is Supervised…



Selecting Some…

- **Supervised Learning: Classification**
  - *k* Nearest Neighbours
  - Naive Bayes
  - Logistic Regression
  - Support Vector Machine
- **Unsupervised Learning: Clustering** (next lecture)
  - K-Means
  - Hierarchical Clustering
  - Graph-based Clustering

http://www.slideshare.net/mirjalil/clustering-on-database-systems-rkm-42588852

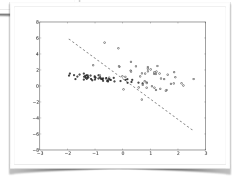
Classification

Supervised Techniques for Classification

Supervised Techniques

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).

Typically, we have a training phase on a sample items, then a test phase in which to test the classifier on unseen items; as in neural networks



Supervised Techniques

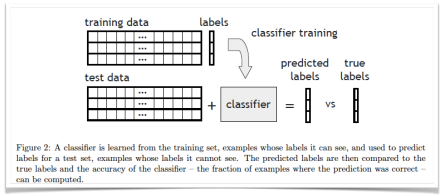


Figure 2: A classifier is learned from the training set, examples whose labels it can see, and used to predict labels for a test set, examples whose labels it cannot see. The predicted labels are then compared to the true labels and the accuracy of the classifier – the fraction of examples where the prediction was correct – can be computed.

Typically, we have a training phase on a sample of items, then a test phase in which to test the classifier on unseen items

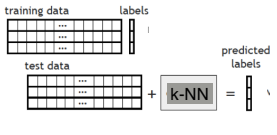
Supervised: Classification

- *k* - Nearest Neighbours (*k*-NN)
- Naive Bayes
- Support Vector Machines (SVM)
- Logistic Regression

Classification

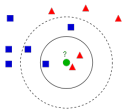
Supervised Techniques: *k*-NN

- *Training Data:* a set of animals each labelled as a panda or a wookie (to find *k*)
- *Test Data:* a set of unseen animals, with k-NN used to determine if panda/wookie



k-NN: The Idea

- If you pick 3 most-similar guys to me, and two are pandas and one a wookie, I am probably a panda
- If you pick 5 most-similar guys to me and three of them are wookies and two are pandas, then I am probably a wookie
- Using k-nearest neighbours; where k is 3 or 5



http://en.wikipedia.org/wiki/K-NN

k-NN: Formulae

Input: D, the set of k training objects, and test object $z = (\mathbf{x}', y')$

Process:

Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every object, $(\mathbf{x}, y) \in D$.

Select $D_z \subseteq D$, the set of k closest training objects to z.

Output: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

Figure 6 provides a high-level summary of the nearest-neighbor classification method. Given a training set D and a test object $z = (\mathbf{x}', y')$ the algorithm computes the distance (or similarity) between z and all the training objects $(\mathbf{x}, y) \in D$ to determine its nearest-neighbor list, D_z. ($\mathbf{x}$ is the data of a training object, while y is its class. Likewise, $\mathbf{x}'$ is the data of the test object and y' is its class.)

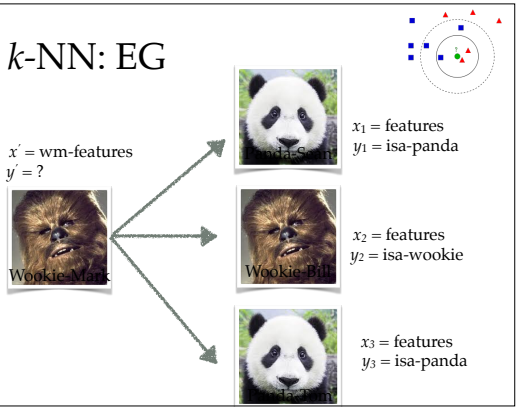
Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

$$\text{Majority Voting: } y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i), \qquad (18)$$

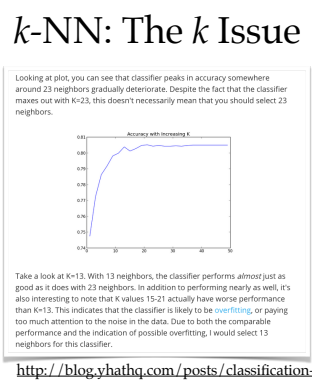
where v is a class label, y_i is the class label for the ith nearest neighbors, and $I(\cdot)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

Wu, X. et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14(1)*, 1-37.

k-NN: EG



k-NN: The *k* Issue



Main issue in *k*-NN is knowing *k;* different *k*s may give different answers , you want most accurate; often done empirically

http://blog.yhathq.com/posts/classification-using-knn-and-python.html

k-NN: Including *Distance*

- Rather than simple voting of neighbours for one class you can measure the distance of items from one another (e.g., cosine similarity/euclidean distance over some tf-idf vector)
- So, if 10 neighbours have 5 from class-A that are all closer than the 5 from class-B; then weighting the voting on distance will give better classification
- So, you can make the classification on the basis of the mean distance score for class-A-NNs versus class-B-NNs (nb, this gets more complicated…)

Cunningham, P., & Delany, S. J. (2007). k-Nearest neighbour classifiers. Mult Classif Syst, 1-17.

k-NN #1: Text Classification

- Classic use is in text processing; to take docs and classify them into news categories (sport, current affairs) or wrt medical categories (vascular, psychiatric)
- Aggressive pre-processing: stop-word removal, stemming, word weighting to achieve an 85% vocabulary reduction
- Docs then cast as tf-idf vectors, k-NN used
- Large scale test across several corpora and techniques shows *k*-NN to do surprisingly well; often used a baseline test

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information retrieval, 1(1-2)*, 69-90.

k-NN #2: Intruder Detection

- 1998 DARPA BSM Competition; idea is to capture "normal" behaviour of users, systems, networks, match new behaviour to these and classify as normal/abnormal
- Uses program profiles broken down in the freqs of system calls (unix commands: *su chmod open close*)
- Corpus is a set of programs, which contain some known "attacks" on various US Navy Systems

Liao, Y., & Vemuri, V. R. (2002). Use of K-nearest neighbor classifier for intrusion detection. *Computers & Security, 21(5)*, 439-448.

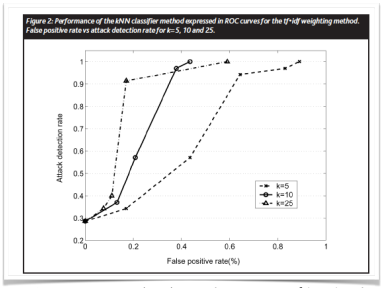
k-NN #2: Intruder Detection

Table 1: Analogy between text categorization and intrusion detection when applying the kNN classifier.

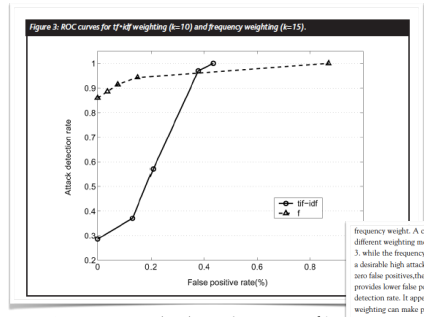
Terms	Text categorization	Intrusion Detection
N	total number of documents	total number of processes
M	total number of distinct words	total number of distinct system calls
n_i	number of times *i*th word occurs	number of times *i*th system call was issued
f_{ij}	frequency of *i*th word in document *j*	frequency of *i*th system call in process *j*
D_j	*j*th training document	*j*th training process
$\overrightarrow{X}$	test document	test process

Liao, Y., & Vemuri, V. R. (2002). Use of K-nearest neighbor classifier for intrusion detection. *Computers & Security, 21(5)*, 439-448.

k-NN #2: Intruder Detection



Liao, Y., & Vemuri, V. R. (2002). Use of K-nearest neighbor classifier for intrusion detection. *Computers & Security, 21(5)*, 439-448.



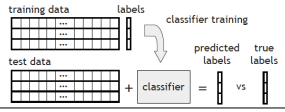
frequency weight. A comparison of two different weighting methods is shown in Figure 3. while the frequency weighting method offers a desirable high attack detection rate (86%) at zero false positives,the *tf-idf* weighting method provides lower false positive rate at 100% attack detection rate. It appears that the *tf-idf* weighting can make process vectors of two classes more distinguishable than the frequency weighting. Therefore, a lower threshold value is needed, and better false positive rate can be achieved with the *tf-idf* weighting method.

Liao, Y., & Vemuri, V. R. (2002). Use of K-nearest neighbor classifier for intrusion detection. *Computers & Security, 21(5)*, 439-448.

Classification

Supervised Techniques: *Naive Bayes*

- *Training Data:* some set of examples with different features:
  animal-a {black-eyes, white-fur, 4-paws,ears} -> panda
  animal-b {brown-eyes, brown-fur, 2-paws, no-ears} -> wookie
  animal-c {black-eyes, cream-fur, 3-paws,ears} -> panda
  animal-d {brown-eyes, brown-fur, 2-paws, no-ears} -> wookie
- *Test Data:* a set of unseen animals, with Naive Bayes generating label for it based on the presented features:
  animal-x {brown-eyes, brown-fur, 1-paw, no-ears} -> ?



Naive Bayes: The Idea

- "Most" of the wookies I have seen have brown eyes and pandas have black; one or two wookies have black eyes, and I saw one panda with brown eyes (but pandas can also have blue, green and red eyes !).
- Use the probability of features (e.g., eye-colour) occurring in some sample of instances (wookies and pandas I have seen), wrt a category (wookie or panda), to predict the probability than an unseen Beast-X is in one category or the other
- Clever Bayes bit tells you how to combine the probabilities of different features, to compute an overall probability for X

http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Naive Bayes: Formula

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some n features (dependent variables), it assigns to this instance probabilities

$p(C_k|x_1, \ldots, x_n)$

for each of k possible outcomes or *classes*.[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Naive Bayes: Bayes Theorem

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j,
  This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j,
  We can imagine that being in class c_j causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j,
  This is just how frequent the class c_j is in our database
- $p(d)$ = probability of instance d occurring
  This can actually be ignored, since it is the same for al

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naive Bayes: EG (single feature)

Eyes	Beast
brown	panda
black	wookie
brown	wookie
brown	wookie
red	panda
brown	wookie
blue	wookie
green	panda

Beast-X

Naive Bayes: EG (single feature)

Eyes	Beast
brown	panda
black	wookie
brown	wookie
brown	wookie
red	panda
brown	wookie
blue	wookie
green	panda

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

p(brown/panda) = 1/3
p(panda) = 3/8
p(brown) = 4/8

$$p(\text{panda/brown}) = \frac{1/3 * 3/8}{4/8} = \frac{0.125}{4/8}$$

$$p(\text{wookie/brown}) = \frac{3/5 * 5/8}{4/8} = \frac{0.375}{4/8}$$

Naive Bayes: EG (many features)

Officer Drew and Nina McGrew….

Name	Over 170CM	Eye	Hair length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naive Bayes: Turtles !

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

The probability of class c_j generating the observed value for feature 2, multiplied by..

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naive Bayes: Drew Eg…

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

$$p(\text{officer drew}|c_j) = p(\text{over\_170}_{cm} = \text{yes}|c_j) * p(\text{eye} = blue|c_j) * \ldots$$

Officer Drew is blue-eyed, over 170_{cm} tall, and has long hair

$p(\text{officer drew}|\text{Female}) = 2/5 * 3/5 * \ldots$

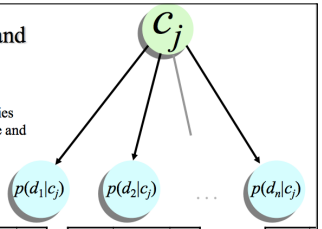
$p(\text{officer drew}|\text{Male}) = 2/3 * 2/3 * \ldots$

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naive Bayes: Drew Eg…

Naïve Bayes is fast and space efficient

We can look up all the probabilities with a single scan of the database and store them in a (small) table…



Sex	Over190$_{cm}$	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

Sex	Long Hair	
Male	Yes	0.05
	No	0.95
Female	Yes	0.70
	No	0.30

Sex	
Male	
Female	

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naïve Bayes is NOT sensitive to irrelevant features...

Suppose we are trying to classify a persons sex based on several features, including eye color. (Of course, eye color is completely irrelevant to a persons gender)

$$p(\text{Jessica}|c_j) = p(\text{eye} = \text{brown}|c_j) * p(\text{wears\_dress} = \text{yes}|c_j) * \ldots$$

$p(\text{Jessica} \mid \text{Female}) = 9{,}000/10{,}000 * 9{,}975/10{,}000 * \ldots$

$p(\text{Jessica} \mid \text{Male}) = 9{,}001/10{,}000 * 2/10{,}000 * \ldots$

Almost the same!

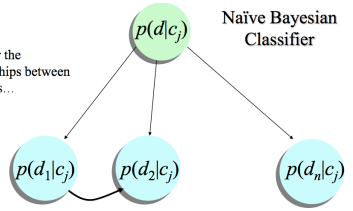
However, this assumes that we have good enough estimates of the probabilities, so the more data the better.

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Independence Problem?

Solution

Consider the relationships between attributes…



Finding arcs is external to Bayes

Sex	Over 6 foot	
Male	Yes	0.15
	No	0.85
Female	Yes	0.01
	No	0.99

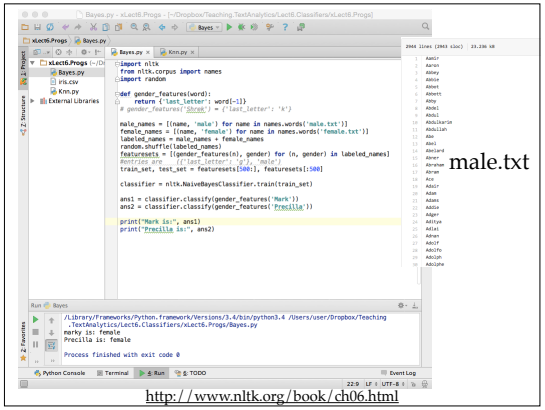
Sex	Over 200 pounds	
Male	Yes and Over 6 foot	0.11
	No and Over 6 foot	0.59
	Yes and NOT Over 6 foot	0.05
	No and NOT Over 6 foot	0.35

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Naive Bayes: Issues

- Advantages:
  - Fast to train (single scan)
  - Robust for irrelevant features
  - Handles streaming data well
- Disadvantages
  - Assumes independence of features

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf



male.txt

http://www.nltk.org/book/ch06.html

Naive Bayes: Li & Jain (1998)

- Bayes heavily used in classification of texts
- What topic is a news article about? What words probabilistically occur in business news versus entertainment news…
- Item is doc, words are features (stemmed?, less stops); category is "business", "entertainment", "sport"
- Classifier uses test set to learn which word features best predict which news category

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *Computer Journal, 41(8)*, 537-546.

Naive Bayes: Li & Jain (1998)

We adopt the commonly used 'bag-of-words' [12] document representation scheme (vector space model), in which we ignore the structure of a document and the order of words in the document. The feature vectors represent the words observed in the documents. The *word-list* $W = (w_1, \ldots, w_d)$ in the training set consists of all the distinct words (also called *terms*) that appear in the training samples after removing the *stopwords* [13] (those words which are not helpful for retrieval, such as 'the', 'some' and 'of') and the low-frequency words (which only occur once in the training examples). Typically, there can be thousands of features in document classification (the number of commonly used English words is about 50,000). Given a document D, its feature (term) vector is represented by $\mathcal{T} = (t_1, \ldots, t_d)$ constructed from W. The value of each component of $\mathcal{T}$ could be either binary (a value of 1 indicates whether the corresponding word appeared in the document) or an integer representing the number of times the corresponding word was observed. In this paper we use binary representation. Frequency representation is also used

6. EXPERIMENTAL RESULTS

The data used in our experiments are the news items downloaded from the Yahoo news group. We preprocess the HTML news items by (i) document parsing (remove headers and tags in the HTML files) and (ii) removing *stopwords* and low-frequency words as mentioned earlier. We have used a total of 814 documents belonging to seven different classes (business (B), entertainment (E), health (H), international (I), politics (P), sports (S) and technology (T)) for training and two test data sets (news items at different time intervals, see Table 1). There are 265 distinct terms (only observed in at most one of the classes) in the training data and 75 terms are common among all the categories. On average, a document contains about 120 words.

Data & Corpus

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *Computer Journal, 41(8)*, 537-546.

Naive Bayes: Li & Jain (1998)

Minimum Wage Rises ~ Nearly 7 million Americans are getting a raise on this Labor Day. The federal minimum wage is rising to $5.15 an hour. Fast food workers, retail clerks, gas station attendants and others will be earning 40 cents an hour more when they report to work as the second phase of the hike goes into effect. It was first raised to $4.75 last Oct. 1. According to a report to be issued tomorrow by the Economic Policy Institute, most of the 6.8 million workers affected by the minimum wage hike are women who work in the service sector. The Washington, D.C.-based think tank's study found that in 18 states, more than 10 percent of the work force will be affected by the minimum wage increase.	minimum wage rises _ nearly _ million americans ___ getting _ raise __ ____ labor day_ ___ federal minimum wage __ rising __ _____ __ hour_ fast food workers_ retail clerks_ gas station attendants ___ _____ __ __ earning __ cents __ hour __ ____ ____ report __ _____ __ __ _____ phase __ __ hike goes ____ effect_ __ ___ _____ raised __ _____ ____ oct_ __ according __ _ report __ __ issued tomorrow __ __ ___ economic policy institute _____ __ ____ million workers affected __ ___ minimum wage hike ___ women ___ ____ __ ___ service sector_ ___ washington_ _____based _____ tank__ study found _____ __ __ _____ ____ __ percent ___ __ ____ force ____ __ affected __ ___ minimum wage increase
(a)	(b)

FIGURE 3. An example of the *business* news group: (a) a training sample; (b) extracted word list.

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *Computer Journal, 41(8)*, 537-546.

Naive Bayes: Li & Jain (1998)

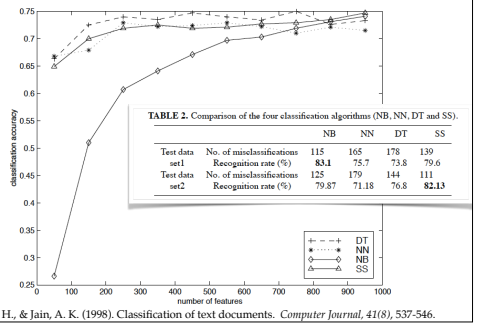


TABLE 2. Comparison of the four classification algorithms (NB, NN, DT and SS).

		NB	NN	DT	SS
Test data set1	No. of misclassifications	115	165	178	139
	Recognition rate (%)	**83.1**	75.7	73.8	79.6
Test data set2	No. of misclassifications	125	179	144	111
	Recognition rate (%)	79.87	71.18	76.8	**82.13**

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *Computer Journal, 41(8)*, 537-546.

Classification

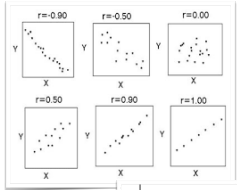
Supervised Techniques: *Logistic Regression*

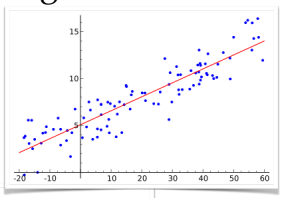
Correlations & Regression

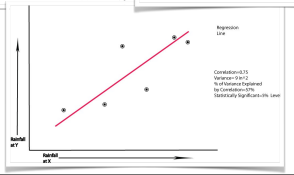
- **Correlation:** tells you how two sets of figures (two variables) co-vary with one another; using co-variance and variance (i.e., degree of linear dependence between 2 vars)
- **Linear Regression**: tells you about the relationship between some dependent variable and one or more explanatory variables (>1 being Multiple Regression)

https://en.wikipedia.org/wiki/Linear_regression

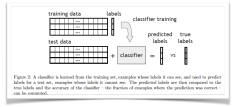
Correlations & Regression: Pics







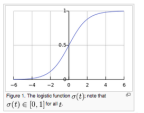
Regression: Linear V Logistic



- **Linear & Multiple Regression** deals with predicting some ***continuous*** dependent variable from one or more independent variables (normally distributed)
- **Logistic Regression** deals with predicting some ***binary*** dependent variable (1/0; cannot be normally distributed) from one or more independent variables

https://en.wikipedia.org/wiki/Logistic_regression

Regression: The Idea



- **Logistic Regression** projects "normal" regression into a logistic function, so that the output is between 0 and 1 (not cont.)
- It gives the probability that something will/will not (1/0) occur (or be the case) given value(s) of some independent variable(s)

https://en.wikipedia.org/wiki/Logistic_regression

Logistic Regression: Formula

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\frac{P}{1-P} = e^{a+bX}$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

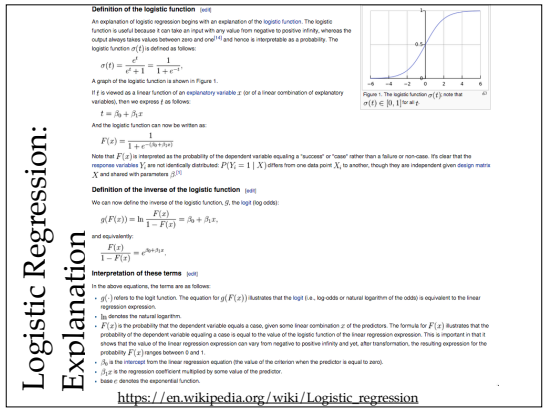
formula relates the log of the odds (P/1-P) to the linear function for the explanatory variable

a is intercept, ***b*** the slope of the regression line and ***X*** the value of the independent variable

when there are more variables you just add more terms to the linear fn

http://en.wikipedia.org/wiki/Jaccard_index

Logistic Regression: Explanation



Definition of the logistic function [edit]

An explanation of logistic regression begins with an explanation of the logistic function. The logistic function is useful because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one[1] and hence is interpretable as a probability. The logistic function $\sigma(t)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t+1} = \frac{1}{1+e^{-t}}.$$

A graph of the logistic function is shown in Figure 1.

If t is viewed as a linear function of an explanatory variable x (or of a linear combination of explanatory variables), then we express t as follows:

$$t = \beta_0 + \beta_1 x$$

And the logistic function can now be written as:

$$F(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

Note that $F(x)$ is interpreted as the probability of the dependent variable equalling a "success" or "case" rather than a failure or non-case. It's clear that the responses variables Y_i are not identically distributed: $P(Y_i = 1 \mid X)$ differs from one data point X_i to another, though they are independent given design matrix X and shared with parameters β.[3]

Definition of the inverse of the logistic function [edit]

We can now define the inverse of the logistic function, g, the logit (log odds):

$$g(F(x)) = \ln\frac{F(x)}{1-F(x)} = \beta_0 + \beta_1 x,$$

and equivalently:

$$\frac{F(x)}{1-F(x)} = e^{\beta_0+\beta_1 x}.$$

Interpretation of these terms [edit]

In the above equations, the terms are as follows:

- $g(\cdot)$ refers to the logit function. The equation for $g(F(x))$ illustrates that the logit (i.e., log-odds or natural logarithm of the odds) is equivalent to the linear regression expression.
- $\ln$ denotes the natural logarithm.
- $F(x)$ is the probability that the dependent variable equals a case, given some linear combination x of the predictors. The formula for $F(x)$ illustrates that the probability of the dependent variable equaling a case is equal to the value of the logistic function of the linear regression expression. This is important in that it shows that the value of the linear regression expression can vary from negative to positive infinity and yet, after transformation, the resulting expression for the probability $F(x)$ ranges between 0 and 1.
- β_0 is the intercept from the linear regression equation (the value of the criterion when the predictor is equal to zero).
- $\beta_1 x$ is the regression coefficient multiplied by some value of the predictor.
- base e denotes the exponential function.

https://en.wikipedia.org/wiki/Logistic_regression

Logistic Regression: EG1

- **Data:** Imagine you have some data on the number of servers that are hacked (1) or not hacked (0) after different time periods in weeks (28 to 33 weeks)
- **Logistic Regression**: will give you a probability for whether a new server is likely to be hacked, based on this data, after a given time period (say 31 weeks)

http://vassarstats.net/logreg1.html

Logistic Regression: EG1

Data: The number of servers that are hacked (1) or not hacked (0) after different time periods measured in weeks (28 to 33 weeks)

i	ii	iii	iv	v	vi	vii
	Instances of Y Coded as			Y as		
X	0	1	Total ii+iii	Observed Probability	Y as Odds	Y as Log Odds
28	4	2	6	.3333	.5000	-.6931
29	3	2	5	.4000	.6667	-.4055
30	2	7	9	.7778	3.5000	1.2528
31	2	7	9	.7778	3.5000	1.2528
32	4	16	20	.8000	4.0000	1.3863
33	1	14	15	.9333	14.0000	2.6391

$$\text{Odds} = \frac{\text{probability of the event}}{1 - \text{probability of the event}} = \frac{P}{1-P}$$

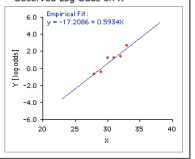
OR

$$P = \frac{\text{Odds}}{1+\text{Odds}}$$

http://vassarstats.net/logreg1.html

X	Observed Probability	Log Odds	Weight
28	.3333	-.6931	6
29	.4000	-.4055	5
30	.7778	1.2528	9
31	.7778	1.2528	9
32	.8000	1.3863	20
33	.9333	2.6391	15

C. Weighted Linear Regression of Observed Log Odds on X



For each level of X, the weighting factor is the number of observations for that level.

Intercept=-17.2086 is the point on the Y-axis (log odds) crossed by the regression line when X=0.

Slope=.5934 is the rate at which the predicted log odds increases (or, in some cases, decreases) with each successive unit of X. Within the context of logistic regression, you will usually find the slope of the log odds regression line referred to as the "constant."

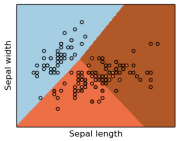
The exponent of the slope
exp(.5934) = 1.81
describes the proportionate rate at which the predicted odds changes with each successive unit of X. In the present example, the predicted odds for X=29 is 1.81 times as large as the one for X=28; the one for X=30 is 1.81 times as large as the one for X=29; and so on.

http://vassarstats.net/logreg1.html

Logistic Regression: Prog

Logistic Regression 3-class Classifier

Show below is a logistic-regression classifiers decision boundaries on the iris dataset. The datapoints are colored according to their labels.

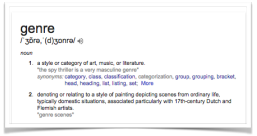


Python source code: plot_iris_logistic.py

```
print(__doc__)


# Code source: Gaël Varoquaux
# Modified for documentation by Jaques Grobler
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model, datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target

h = .02  # step size in the mesh

logreg = linear_model.LogisticRegression(C=1e5)
```

http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

LR: Genres


genre
/ ʒänrə, (d)ʒänrə/
noun
1. a style or category of art, music, or literature.
"the spy thriller is a very masculine genre"
synonyms: category, class, classification, categorization, group, grouping, bracket, head, heading, list, listing, set; More
2. denoting or relating to a style of painting depicting scenes from ordinary life, typically domestic situations, associated particularly with 17th-century Dutch and Flemish artists.
"genre scenes"

- Genre: "reflects way text created, how it was distributed, register of language, audience addressed"
- So, may be about words used, expressions, syntactic constructions, …
- **Generic Facets**: features of text identified as important in determining genre: *broadcast* V *directed, narrative / suasive / descriptive / directive*

Kessler, B., Numberg, G., & Schütze, H. (1997, July). Automatic detection of text genre. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (pp. 32-38). ACL

LR: Genre Facets

- Facets are quite high-level based on lower-level "visible" features:
  - *structural*: passives, nominalisation, POS counts
  - *lexical:* terms of address Mr…Mrs (register)
  - *character-level*: punctation, !s, ?s
  - *derivative*: ratios of words, sentence lengths…

Kessler, B., Numberg, G., & Schütze, H. (1997, July). Automatic detection of text genre. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (pp. 32-38). ACL

LR: Genre Facets

- 499 texts Brown Corpus: training (402), test (97)
- Facets: Brow (popular, middle…high), Narrative (yes or no), Genre (reportage, editorial, scitech, legal, fiction, nonfiction)
- Use LR for each facet on each training text; then tested against unseen 97 (NN compare)

Kessler, B., Numberg, G., & Schütze, H. (1997, July). Automatic detection of text genre. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (pp. 32-38). ACL

LR: Results

Table 2: Classification Results for Each Facet Level.

Levels	Baseline	LR (Surf.) All	LR (Surf.) Sel.	2LP All	3LP All	LR (Struct.) All	LR (Struct.) Sel.
Genre							
Rep	81	89*	88	94*	94*	90*	90*
Edit	81	75	75	74	80	79	77
Legal	95	96	96	95	95	93	93
Scitech	94	100*	96	99*	94	93	96
Nonfict	67	67	68	78*	67	73	74
Fict	81	93*	96*	99*	81	96*	96*
Brow							
Popular	74	74	75	74	74	72	73
Middle	68	66	67	64	64	58	64
Uppermiddle	88	74	78	86	88	79	82
High	70	84*	88*	89*	90*	85*	86*

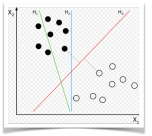
Note. Numbers are the percentage of the evaluation subcorpus ($N = 97$) which was correctly classified on a binary discrimination task. The Baseline column tells what percentage would be got correct by guessing No for each level. Readers have the same meaning as in Table 1.
* means significantly better than Baseline at $p < .05$, using a binomial distribution (N=97, p as per first column).

Kessler, B., Numberg, G., & Schütze, H. (1997, July). Automatic detection of text genre. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (pp. 32-38). ACL

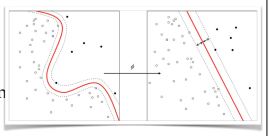
Classification

Supervised Techniques: *Support Vector Machines*

SVM: The Idea

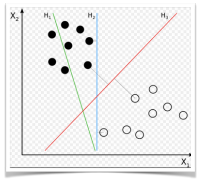


- Explores hyperplanes for an multi-dimensional space; finding the "best" one; geometrically, the one with the largest separation (or margin) between classes
- Basically, linear classification but can use **kernel** idea to handle non-linear classifications (projecting into linear)



In geometry a **hyperplane** is a subspace of one dimension less than its ambient space. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are the 1-dimensional lines. This notion can be used in any general space in which the concept of the dimension of a subspace is defined.

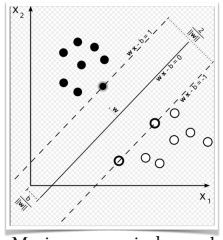
SVM: The Idea



H1 does not separate classes

H2 does, but with small margin

H3 separates with maximal margin



Maximum-margin hyperplane and margins for an SVM trained on samples from two classes. Samples on the margin are called the **support vectors.**

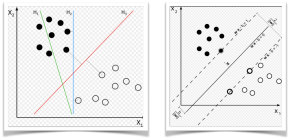
http://en.wikipedia.org/wiki/Support_vector_machine

SVM: Formula

The reason why SVM insists on finding the maximum margin hyperplanes is that it offers the best generalization ability. It allows not only the best classification performance (e.g., accuracy) on the training data, but also leaves much room for the correct classification of the future data. To ensure that the maximum margin hyperplanes are actually found, an SVM classifier attempts to maximize the following function with respect to $\vec{\mathbf{w}}$ and b:

$$L_P = \frac{1}{2} \| \vec{\mathbf{w}} \| - \sum_{i=1}^{t} \alpha_i y_i (\vec{\mathbf{w}} \cdot \vec{\mathbf{x}}_i + b) + \sum_{i=1}^{t} \alpha_i \qquad (2)$$

where t is the number of training examples, and α_i, $i = 1, \ldots, t$, are non-negative numbers such that the derivatives of L_P with respect to α_i are zero. α_i are the Lagrange multipliers and L_P is called the Lagrangian. In this equation, the vectors $\vec{\mathbf{w}}$ and constant b define the hyperplane.



SVM: Formula

Linear SVM [edit]

Given some training data $\mathcal{D}$, a set of n points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the y_i is either 1 or −1, indicating the class to which the point $\mathbf{X}_i$ belongs. Each $\mathbf{X}_i$ is a p-dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where · denotes the dot product and $\mathbf{w}$ the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

By using geometry, we find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. As we also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \text{ of the first class}$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \text{ of the second.}$$

This can be rewritten as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \qquad (1)$$

We can put this together to get the optimization problem:

Minimize (in $\mathbf{w}, b$)

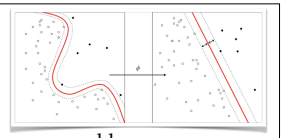
$$\|\mathbf{w}\|$$

subject to (for any $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$$

http://en.wikipedia.org/wiki/Jaccard_index

SVM: More…



- Linear SVM assumes linear separable
- But, there are extensions using kernel function to non-linear cases; many different including quadratic and exponential kernel functions

In Euclidean geometry **linear separability** is a geometric property of a pair of sets of points. This is most easily visualized in two dimensions (the Euclidean plane) by thinking of one set of points as being colored blue and the other set of points as being colored red. These two sets are *linearly separable* if there exists at least one line in the plane with all of the blue points on one side of the line and all the red points on the other side. This idea immediately generalizes to higher-dimensional Euclidean spaces if line is replaced by hyperplane.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem.[2]

SVM Eg: SPAM:Context

- 2014, 4.1B email accounts; 196.3B e-mails per day (108.7B business); 12% of business mails are spam (after spam filtering); 13B spam mails…(www.radicati.com "email statistics')
- Traditional methods: white-lists, black-lists, handcrafted rules for text ("viagra"), patterns like missing headers, body only pic, too many non-words (v.i.a.g.r.a)
- Idea: to have tractable learning mechanism for spam/non-spam
- Nearly every classifier has been applied to spam

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM:The Problem

- Take a corpus of emails, labelled as spam/non-spam; create a classifier to distinguish two classes, apply to incoming mails
- Spam-detection becomes a text-classification problem; tailored to specific mails of a given users
- Also, a cost-sensitive problem; blocking a non-spam mail has higher cost than allowing a spam mail (weighted classification)
- Also, look at encoding issues; size of training set, size of attribute set, use of ngrams as features and Sequential Minimal Optimisation (SMO), time and memory efficient version of SVM

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM: Data

- Training works on smallish sets; 100s and 1000s of emails; concentrated on non"regular correspondents"
- Headers, html removed, but stop-words left, no lemmatisation and "$" and "£" treated as tokens (nb stops in twitter)
- Each message is a feature-vector; sometimes binary, but they used frequency and length of doc ("$", "money")
- Also ngrams as they capture typical phrases: "get rich", "rich quick"; tho' with size of n, this grows exponentially, no n > 3

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM: Subplot

- ngrams will produce many features and sparsity; so how do you cut them down
- Took all items; 1gram, 2grams, 3grams
- Remove all items with count < 4;
- Get Information Gain of every feature wrt its category (legitimate v spam); reduces entropy/redundancy of the feature set; optimises difference in features wrt category

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM: Feature Selection

Table II. The n-grams with the highest information gain scores in PU1, for $n \in \{1, 2, 3\}$, ranked by decreasing score. Also shown are the a priori and conditional probabilities of the n-grams, assuming Boolean attributes. Shaded and non-shaded rows indicate n-grams that are more frequent in the spam and legitimate category, respectively.

i	n-gram	$P(X_i = 1)$	$P(X_i = 1 \mid c_L)$	$P(X_i = 1 \mid c_S)$
1	! !	0.231607	0.001612	0.527950
2	!	0.484105	0.216129	0.829157
3	$	0.257947	0.040322	0.538302
4	! ! !	0.181653	0.001612	0.414078
5	language	0.247956	0.440322	0.002070
6	money	0.163487	0.001612	0.372670
7	remove	0.146230	0.001612	0.333333
8	free	0.309718	0.104838	0.573498
9	00	0.141689	0.001612	0.322981
10	fax ;	0.174386	0.309677	0.002070
11	university	0.219800	0.374193	0.022774
12	click	0.118982	0.001612	0.271221
13	mailing	0.169845	0.029032	0.351966
14	natural	0.156221	0.277419	0.002070
15	the internet	0.110808	0.001612	0.252587
16	removed	0.110808	0.001612	0.252587
17	research	0.281562	0.445161	0.072463
18	university of	0.154405	0.274193	0.002070

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM: Complexity

Table III. Computational complexity of the four learning algorithms. N is the number of training messages, $m \leq M$ the number of selected attributes, and m' the number of iterations in LogitBoost.

algorithm	training	classification
Naive Bayes	$O(mN)$	$O(m)$
Flexible Bayes	$O(mN)$	$O(mN)$
SVM with SMO	$O(mN^2)$	$O(mN)$
linear SVM with SMO	$O(mN^2)$	$O(m)$
LogitBoost with regression stumps	$O(m'mN^2)$	$O(m')$

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM Eg: SPAM: Results

Table VIII. Real-life evaluation results of Filtron, using the SVM with 520 1-gram attributes, for $\lambda = 1$. The SVM was trained on PU3. Bracketed precision, recall, and $WAcc$ scores are the corresponding scores we had obtained with 10-fold cross validation on PU3.

days used	212	
messages received	6732	(avg. 31.75 per day)
spam messages received	1623	(avg. 7.66 per day)
legitimate messages received	5109	(avg. 24.10 per day)
legitimate-to-spam ratio	3.15	
correctly classified legitimate messages ($L \rightarrow L$)	5057	
incorrectly classified legitimate messages ($L \rightarrow S$)	52	(avg. 1.72 per week)
correctly classified spam messages ($S \rightarrow S$)	1450	
incorrectly classified spam messages ($S \rightarrow L$)	173	(avg. 5.71 per week)
precision	96.54%	(PU3: 96.43%)
recall	89.34%	(PU3: 95.05%)
$WAcc$	96.66%	(PU3: 96.22%)

Of the 52 fails, 52% were auto-mails (newsletters), 22% 2-3 word mails with attachments, 26% were causal 1-2 line mails in spam style with no attachments/hyperlinks (lambda relates to cost fn)

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

SVM: Issues

- Theoretically, well-founded and can work with a few dozen examples, also insensitive to number of dimensions; but nb complexity

Classification

A Tangent on Cross-Validation

A Problem ?

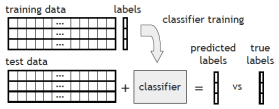


Figure 2: A classifier is learned from the training set, examples whose labels it can see, and used to predict labels for a test set, examples whose labels it cannot see. The predicted labels are then compared to the true labels and the accuracy of the classifier – the fraction of examples where the prediction was correct – can be computed.

- Note, we have built this classifier on a sample of training data (randomly selected?)
- It might not be a good/representative sample making classifier crap/overfitted

A Solution?

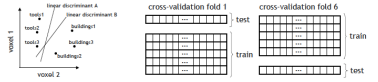


Figure 3: **left:** Learning a linear classifier is equivalent to learning a line that separates examples in the two classes (vectors in a 2-voxel brain) as well as possible. **right:** During cross-validation each of 6 groups of examples takes a turn as the test set while the rest serve as the training set.

- Cross-validation deals with this problem
- Use several different (systematic) samples of training sets and report the mean solution
- Also indicates the model's performance; very diff answers for diff training sets is bad

Types of Cross Validation

- **Exhaustive:** (nb complexity)
  - *leave-p-out*: use p items as test subset; the rest as training subset; explore all subsets of p-items
  - *leave-one-out*: same, but where p = 1
- **Non-Exhaustive:** (nb better on complexity)
  - ***k**-fold*: randomly partition items into k-subsamples; and use one for test, rest for training
  - *2-fold*: same where k is 2
  - *repeated random sub-sampling*: as it suggests

k-fold cross-validation

1. Partition original sample into *k* sub-samples, by random selection of items
2. One sub-sample is used as test-subset, and remaining *k*-1 sub-samples as training subset
3. Cross-validation is repeated *k* times for each; with results combined or averaged

- *Stratified, k-fold*; mean response value is kept approx. equal; e.g. 50/50 split in binary label set
- *Benefit*: all training set is used and better complexity than exhaustive methods

k-fold cross-validation: EG

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). Learning to filter unsolicited commercial e-mail. " DEMOKRITOS", National Center for Scientific Research.

All of the experiments were performed using stratified ten-fold *cross-validation*. That is, each message collection was divided into ten equally large parts, maintaining the original distribution of the two classes in each part. Each experiment was repeated ten times, reserving a different part for testing at each iteration, and using the remaining parts for training. The results were then averaged over the ten iterations, producing more reliable results, and allowing the entire corpus to be exploited for both training and testing. The attribute selection process was repeated anew at each iteration of the cross-validation. When we refer to the training corpus of cross-validation experiments, we mean the nine parts that were used for training at each iteration.

2-fold cross-validation

1. Partition original sample into 2 equally-sized sub-samples, by random selection of items (usually shuffle the data a split it in two)
2. So, you have two subsets; one the test-set, the other the training subset
3. Cross-validation is repeated *twice*, swapping these subsets as test/training

- Benefit: all items used for either training or testing; subsets are large

Repeated, random sub-sampling

- Randomly split the sample into training and testing subsets
- Repeat this a number of times; taking average of results
- NB: may mean a given item never appears in training set

Classifications

Conclusions

Ahem…

- Classification and the use of classifiers is a standard solution to lots of problems
- Given some data-set; can I just develop one of these programs and then re-use it
- There are, however, many issues about their use that need to be kept in mind

Issues…

- Classifiers are only as good as your data-set; if it is unrepresentative or biased then it may be learning weird stuff (tanks+skies)
- When something is learned, we may not really know what it has learned (e.g., what are the important features; could be critical)
- There are dark arts and key choices required to make these work (e.g., pre-processing)