Classification: Introduction

Application Examples

Example

- You are a bank loan officer and need to know which loan applicants are "safe" and which are "risky" for the bank
  - in other words, you want to assign labels "safe" or "risky" to a loan applicant

Example

- You are a marketing manager at an electronics consumer shop and want to guess whether a customer will buy a new computer
- "buys" or "doesn't buy" to a customer

Example

- You are a medical researcher and want to predict which of (say three) specific treatments a patient should receive
- "treat_A" or "treat_B" or "treat_C" to a patient

How To Do That?

1. **Gather a set of previous (e.g., archived) data**
   - such as past loan applicant/customer/patient data
   - this dataset is called training set

RID	*age*	*income*	*student*	*credit_rating*	*class: buys_computer*
1	youth	high	no	fair	**no**
2	youth	high	no	excellent	**no**
3	middle_aged	high	no	fair	**yes**
4	senior	medium	no	fair	**yes**
5	senior	low	yes	fair	**yes**
6	senior	low	yes	excellent	**no**
7	middle_aged	low	yes	excellent	**yes**
8	youth	medium	no	fair	**no**
9	youth	low	yes	fair	**yes**
10	senior	medium	yes	fair	**yes**
11	youth	medium	yes	excellent	**yes**
12	middle_aged	medium	no	excellent	**yes**
13	middle_aged	high	yes	fair	**yes**
14	senior	medium	no	excellent	**no**

   - each object in the training set may have a set of attributes (of any type)
   - each object must have a categorical attribute that takes as values the desired labels.
     - this attribute is called class attribute.
     - the class attribute values must be available in the training set

How To Do That?...

2. **Analyze the training set**
   - create a model (or classifier) that takes as input the non-class attribute values and returns a class value

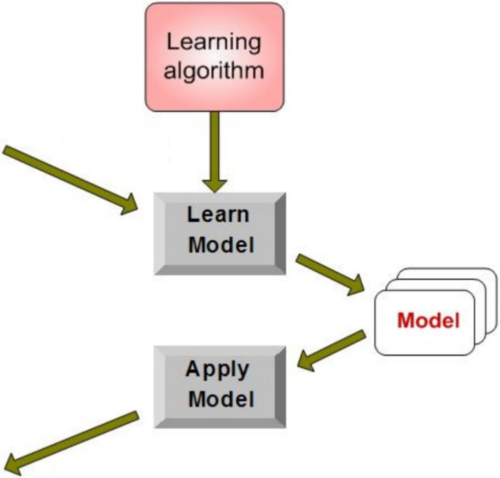
Example

$$f(age, income)$$
$$f(age = \text{“young”}, income = \text{“low”}) = \text{“risky”}$$

- the classifier can be implemented in different ways
  - decision trees
  - rule-based classification
  - Bayesian classification
  - neural network-based classification
  - support vector machines
  - k-nearest neighbor classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



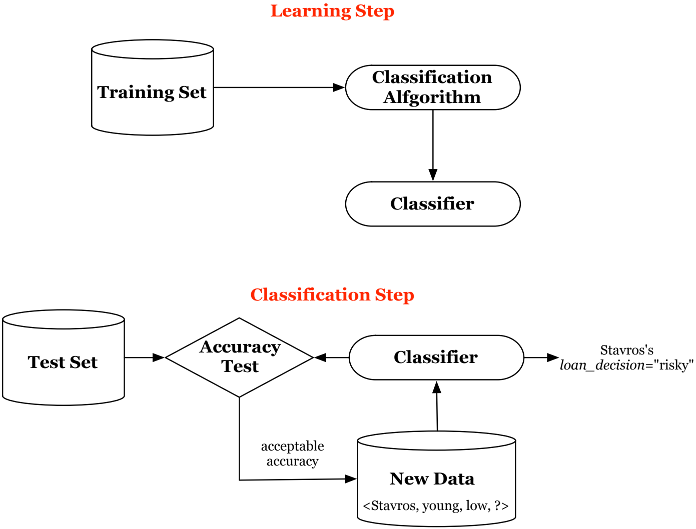
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

How To Do That?...

3. **Evaluate the accuracy of the classifier**
   - find another set of previous (archived) data, with the same attributes as the training set
     - the new data tuples should be disjoint from those in the training set.
     - the new dataset is called test set
   - the accuracy of the classifier will be evaluated based on the test set

How To Do That?...



Classification Accuracy

How to measure the accuracy of a classifier?

- suppose we have already selected the training and test sets, and we have created a classifier based on the training set
- predict the class value of every tuple in the set, and compare it against their actual ones (which are already stored in the set)

Accuracy Measures: Classification Accuracy

- the percentage of tuples that are correctly classified by the classifier
- a useful tool for analyzing how well the classifier can recognize tuples of different classes is the confusion matrix

tuples from class "yes" classified by the classifier as "no"

accuracy of classifying "yes" tuples

Predicted class

Actual class

Classes	*buys_computer="yes"*	*buys_computer="no"*	**Total**	**Accuracy #1 (%)**
buys_computer="yes"	6,954	46	7,000	99.34
buys_computer="no"	412	2,588	3,000	86.27
Total	7,366	2,634	10,000	95.42

total accuracy

$Acc_1 = (6{,}954 + 2{,}588) / 10{,}000$

- rows and columns are the different classes
- $c_{i,j}$: value of cell at row i and column j
- $c_{i,j}$: number of tuples from class i that are classified as those of class j by the classifier

Can be Misleading

- Let us focus on a two-class problem (e.g., "non_cancer" / "cancer" patients) where
  - number of class C_1 tuples: 9,990
  - number of class C_2 tuples: 10
- If the classifier predicts everything to be class C_1, then $Acc_1 = 9,990/10,000 = 99.9\%$
- However, this is misleading because the classifier does not correctly predict any tuple form C_2

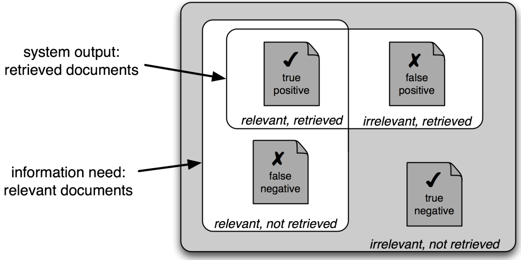
Alternative Accuracy Measures

Consider a two-class problem and the confusion matrix below

C_1 is the main class of interest

	Predicted class			
	Classes	C_1	C_2	**Total**
Actual class	C_1	true positives (t_pos)	false negatives (f_neg)	positives (pos)
	C_2	false positives (f_pos)	true negatives (t_neg)	negatives (neg)

- the positives refer to the tuples of the main class of interest



Document Collection

Alternative Accuracy Measures...

C_1 is the main class of interest

		Predicted class		
	Classes	C_1	C_2	**Total**
Actual class	C_1	true positives (t_pos)	false negatives (f_neg)	positives (pos)
	C_2	false positives (f_pos)	true negatives (t_neg)	negatives (neg)

- precision $= \frac{TP}{TP+FP}$
  - (information retrieval) the fraction of retrieved documents that are relevant to the search (i.e., probability that a (randomly selected) retrieved document is relevant)
- recall $= \frac{TP}{TP+FN}$
  - (IR) the fraction of the documents that are relevant to the query that are successfully retrieved (i.e., probability that a (randomly selected) relevant document is retrieved in a search)
- F-measure $= \frac{2 \cdot \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$
  - combines precision and recall (harmonic mean)

Performance Evaluation

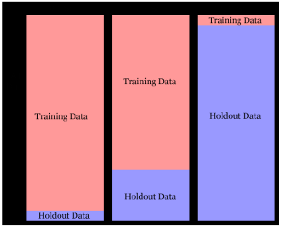
How do we select the training and test sets?

Holdout method

- suppose we possess a dataset D with $|D|$ tuples
- randomly partition D into two disjoint datasets D_1 and D_2
  - typically $|D_1| = \frac{2}{3}|D|$ and $|D_2| = \frac{1}{3}|D|$
- use D_1 as the training set and D_2 as the test set
- use the accuracy measures to derive the classifier's accuracy

Weakness

- fewer records are available for training because some are used in the test set
  - the classifier may not be as good as when all the records of D are used
- the induced classifier may be highly dependent on the composition of the training and test sets
- a class over-represented in the training set will be under-represented in the test set, and vice versa



Better Approaches

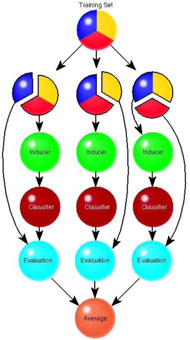
Random subsampling, cross-validation

- all these use several different training sets and test sets
- for every new training set we select, we create a new classifier that is tested against the respective new test set
- the final accuracy estimate is the average over these accuracies

Repeated Subsampling

- repeat the holdout method k times (creating a new classifier each time)
- the overall accuracy estimate is the average of the classifier accuracies obtained from each iteration

k-Fold Cross-Validation



- suppose we possess a dataset D with $|D|$ tuples
- partition D into k disjoint datasets $D_1, D_2, \ldots, D_k$ (called folds) of approximately equal size
- training and testing is performed k times
- in iteration i, D_i is selected as the test set (validation set), and the rest collectively serve as the training set

k-Fold Cross-Validation

Each example is used

- in the validation set once, and
- in the training set for the other $k - 1$ folds

A high proportion of the available data $(1 - \frac{1}{k})$ is used for training, while making use of all the data in computing the error

How many times do we need to perform training?

- typically, $k \sim 10$ is considered reasonable

If $k = |D|$

- train on $|D| - 1$ examples and validate on 1 example
- called leave-one-out cross-validation
- useful for small data sets

k-fold Stratified Cross-validation

Another problem:

- examples in the training set of each fold may not be representative
- e.g., all the examples of a certain class are missing $\rightarrow$ the classifier cannot learn to predict this class

How to ensure that each class is represented with approximately equal proportions in both the training and validation sets?

- partition the $|D|$ examples into k folds such that each class is uniformly distributed among the k folds
- the class distribution in each fold is similar to that in the original data set
- typically, use 10-fold (stratified) cross-validation

Cross-validation...

Which classifier should we eventually create in order to make predictions for the future (i.e., unseen) tuples?

- create the classifier using the entire dataset (D)
- it contains the maximum possible data we can find and thus may lead to better accuracy