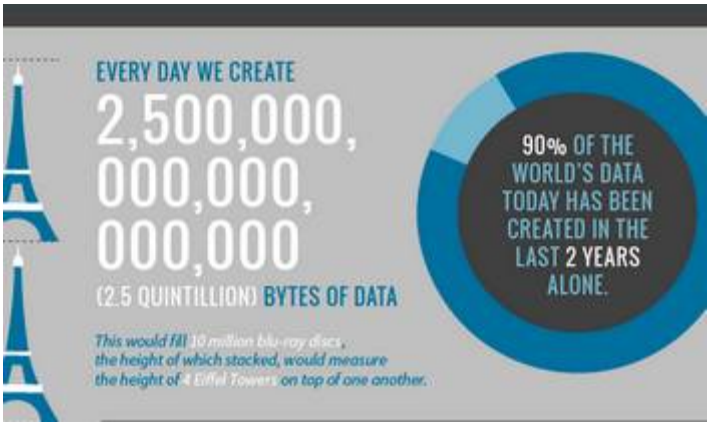
Introduction to Data Science

Machine Learning:
Classification (Supervised Learning)

(Created by HL Viktor: Based on subset of Chapters 8, 9 of Han et. al.)

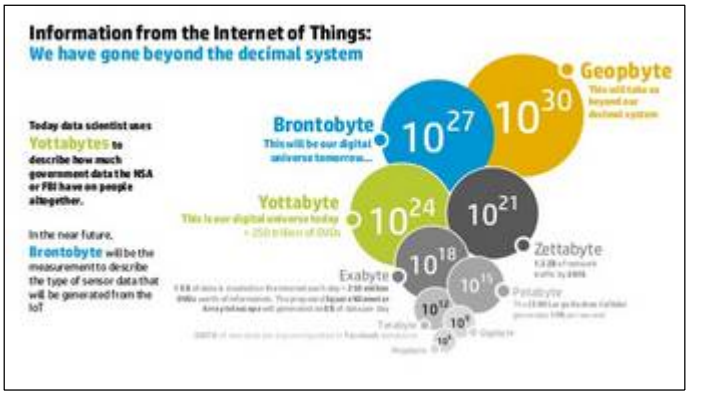
(Machine) Intelligence Revolution?



- http://dncapital.com/thoughts/beyond-big-data-to-data-driven-decisions/
- https://www2.deloitte.com/content/dam/Deloitte/ca/Documents/human-capital/ca-EN-HC-The-Intelligence-Revolution-FINAL-AODA.pdf
- https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=IML14576USEN

Machine Learning

Data driven discovery: making sense of the data deluge



Supervised learning

- Introduction and definitions
- A word about data preparation
- Popular techniques
  - Decision trees
  - Ensembles
  - Support Vector Machines
  - Lazy learners
  - Neural Networks
- Evaluating performance
- Improving performance: ensemble methods

Classification and Prediction

- An example of "Supervised learning"
- We have historic data and the outcome is known
  - Past home owners with a home loan (mortgage):
    - mortgage paid on time (class 0: good)
    - house repossessed by bank (class 1: bad)
  - Heart Surgery patients in a hospital:
    - Back at home (class 0: good)
    - in general ward (class 1: recovering)
    - in Intensive Care (class 2: seriously ill)
    - Deceased (class 3: bad)

5

The goal of classification

- We organize and categorize data in distinct classes
- We know the class labels and the number of classes
- E.g. Past Labor Negotiations (did they go no strike (or not))
- A model is created based on the data distribution
- The model is then used to classify new data
- Classification is used for the prediction of discrete and nominal values
  - Typically with classification, I aim to predict in which bucket to put the ball, not the exact weight of the ball.

6

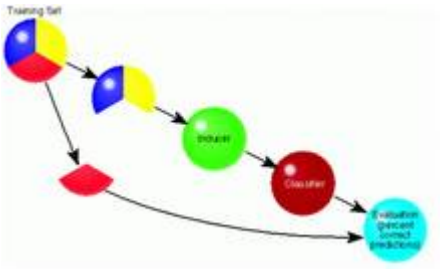
The goal of prediction

- We aim to forecast the value of an attribute based on values of other attributes.
- E.g. Exchange Rate of Canadian Dollar to Euro
- A model is first created based on the data distribution.
- The model is then used to predict future or unknown values.
- Prediction is used for the prediction of numeric
  - Typically with prediction, I aim to predict the exact weight of the ball.

7

The phases of building a classifier (for now)

1. Divide the data into training and test data
2. Induce a classifier (model construction)
3. Test (model evaluation)
4. Use to predict new values (use model)



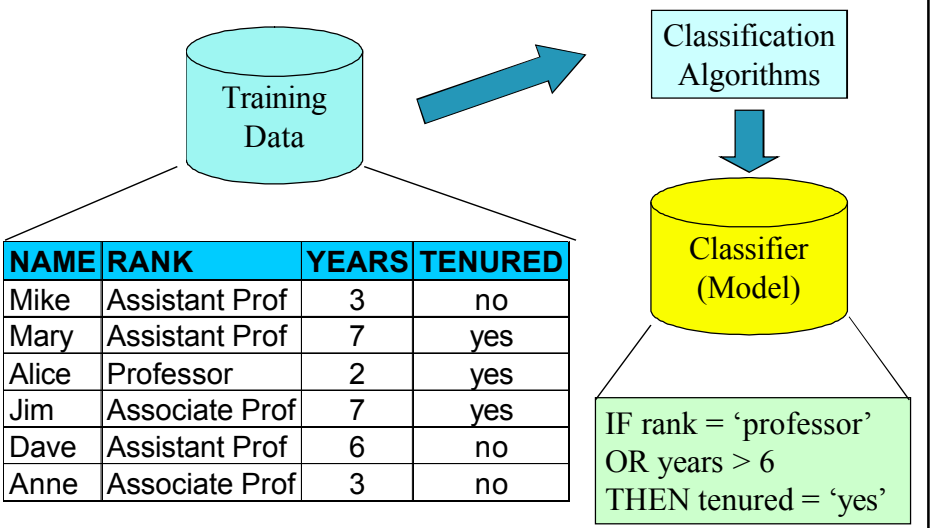
8

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

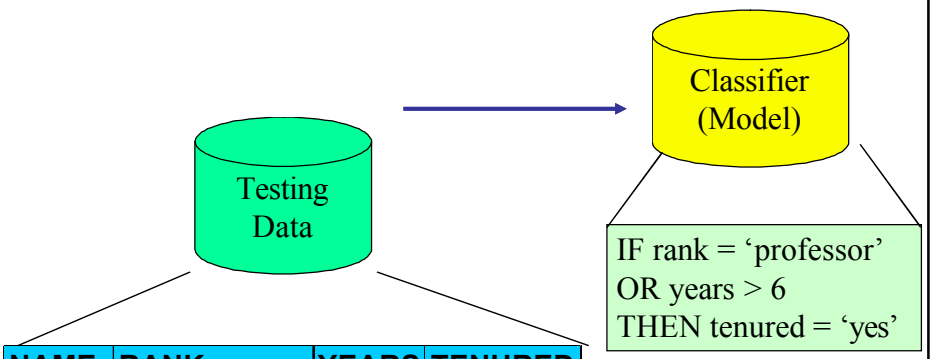
9

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Alice	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

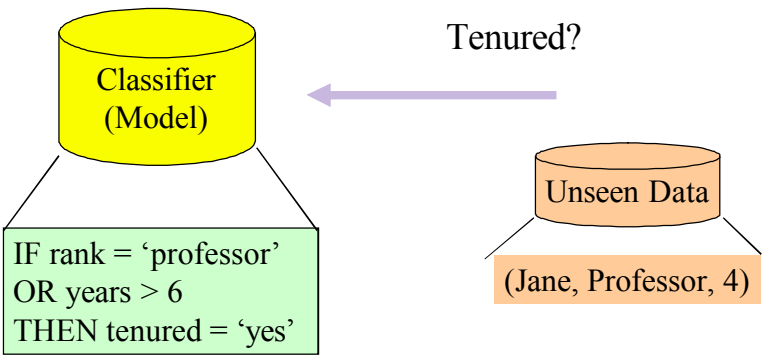
Process (2):
Testing the Model against other data



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
George	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

11

Process (3):
Using the Model in Prediction



12

Two important Issues

1. Data preparation
2. Evaluation

13

1. Preparing data for classification

Data transformation:

- Discretization of continuous data
- Normalization to [-1..1], [0..1], [0.1..0.9], z-score, etc

Data Cleaning:

- Smoothing to reduce noise

Relevance Analysis:

- Feature selection to eliminate irrelevant attributes

ETC. (recall from data staging slides)

14

2: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

15

Classifier 1: Decision trees

- A decision tree is a tree-like structure :-)
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - All tuples in branch have the same value for the tested attribute.
  - Leaf node represents class label or class label distribution.

16

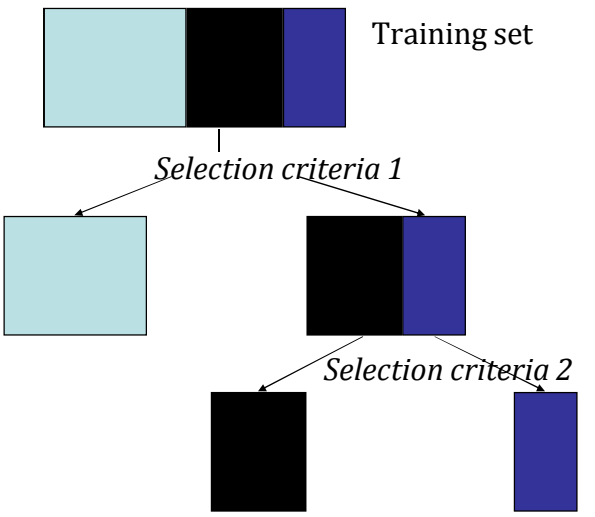
Decision tree definition

- A tree-like structure
  - Internal nodes denote attribute-value tests
  - Branches denotes the different outcomes of the test
  - Leafs denote the Class of that branch

```
wage increase first year <= 2.5 : bad (11.3/2.8)
wage increase first year > 2.5 :
|   statutory holidays > 10 : good (21.2/1.3)
|   statutory holidays <= 10 :
|   |  wage increase first year <= 4 : bad (4.5/1.7)
|   |  wage increase first year > 4 : good (3.0/1.1)
```

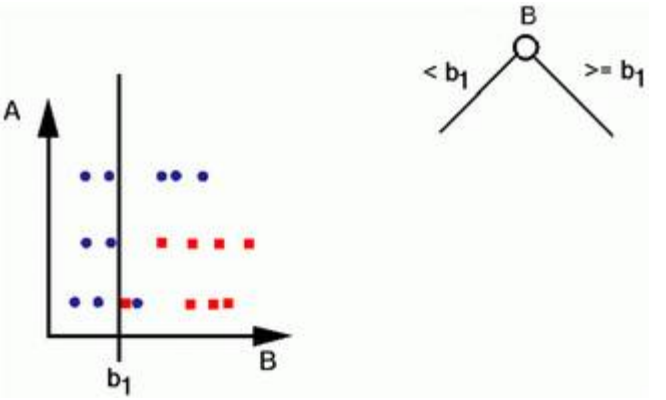
17

Decision trees: the basic idea



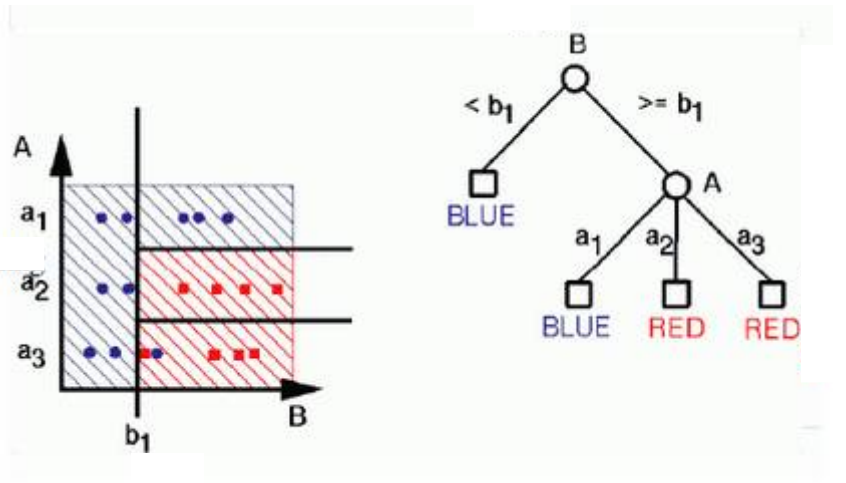
18

Tree induction illustration: First Split



19

Tree induction: Final result



20

Top-down decision tree induction

1. **Tree construction**
- At the start, select an attribute to partition.
- Recursively select attributes and do partition.
2. **Tree pruning**
- Tree may be over classified.
- Remove tree branches that may reflect noise in the training data and lead to errors when classifying test data
- Improve classification accuracy.

21

Recursive decision tree construction

- Starts at root representing all data.
- If all samples belong to same class, then node becomes a leaf labeled with the class label.
- Otherwise, *select attribute* that best separates sample into individual classes.
- Recursion stops when:
  - Sample in node belong to the same class ;
  - No remaining attributes to split;
  - No samples with attribute value.

22

Which attribute to split on? Goodness function

Information gain (ID3/C4.5)

- assume all attributes are categorical.
- can be modified for continuous attributes.

Gini index (CART)

- assume all attributes are continuous.
- assume there are several possible splits for each attribute.
- may need other tools, such as clustering, to get split
- the possible split values can be modified for categorical attributes

23

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

24

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

25

Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

26

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
  - gain_ratio(A = income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

27

Some issues in decision tree construction

Split criterion:
- which goodness functions: information gain, gini index, etc.

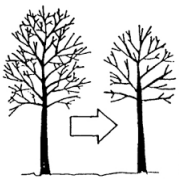
Branching scheme:
- Determining the tree branch to which a sample belongs.
- Binary splitting versus many splitting

Stopping decision:
- When to stop the further splitting of a node

Labeling rule:
- Which class to label a node as

28

What about overfitting?



- Two main approaches
- Pre-pruning
  - Prune during construction process
  - Make "best choice" during construction
- Post-pruning
  - Prune after, using a top-down approach

29

A word about evaluating the results of classification and prediction

30

Model evaluation and selection

- Scenario
  - We constructed one or more models from data
- Questions
  1. How do we evaluate the "quality" of the model?
  a) Predictive accuracy
  b) Significance of the results
  c) Performance (throughput and scalability)
  2. Which model (or combination) should we use

31

Measuring the accuracy

- Depends on what we want to find
  - Accuracy or recognition rate
  - Error to misclassification rate
- Issues
  - How important is it if we "miss" a classification
  - Domain dependent

32

Accuracy Measures

	C1	C2
C1	True positive	False negative
C2	False positive	True negative

Classes Actual Class\Predicted class	Buy_PC =y	Buy-PC =n	total	%
Buy_PC=y	6954	46	7000	99.34
Buy_PC=n	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy (acc(M)): % of tuples correctly classified by model M
- Error rate (err(M)) = 1 – acc(M)
- Confusion Matrix (CF) shown above
- Other accuracy measures
  - Sensitivity = t-pos/pos
  - Specificity = t-neg/neg

33

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positi

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall:

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

34

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity)*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13%
- *Recall* = 90/300 = 30.00%

$$precision = \frac{TP}{TP + FP}$$

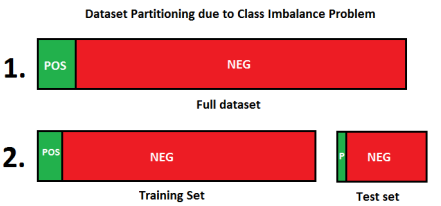
$$recall = \frac{TP}{TP + FN}$$

	C1	**C2**
C1	True positive	False negative
C2	False positive	True negative

35

Example of Class Imbalance Problem

- Consider a two-class scenario:



- In a multi-class setting, we have $n > 2$ classes
- Relevant in numerous domains such as medical diagnosis, network diagnosis, intrusion detection, fraudulent transaction prediction, etc.

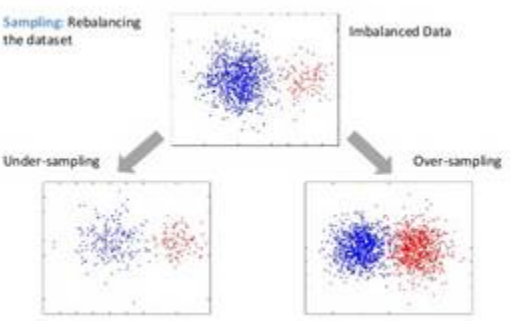
36

Sampling for class imbalance

- Used in to balance frequencies of instances across all classes in the data set.
- Oversampling:
  - Instances of the minority classes are introduced into the training set.
  - For example, synthetic instances are generated (using SMOTE): data may over-fit the model built.
  - Past examples may be accumulated to augment the minority examples in the training sets.
- Under-sampling:
  - Instances of the majority classes are removed.
  - Under-sampling causes information loss, thus cluster-based under-sampling ensures that almost all data regions of the classes are learned.
- Combination of under-sampling and oversampling possible

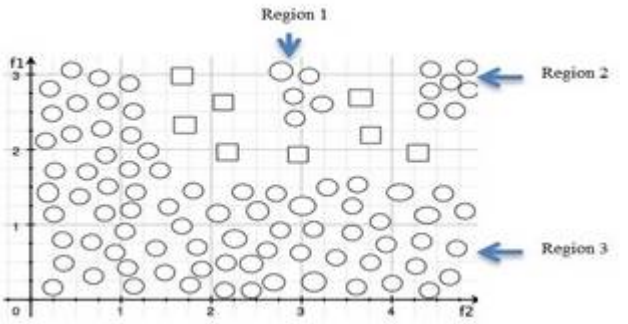
37

An Illustrative Example: 2 classes



38

Within class imbalance



A solution: Cluster and sample within the regions!

39

Evaluating the Accuracy of a Classifier or Predictor (cont.)

Holdout method

- Given data is randomly partitioned into two independent sets
  - Training set (e.g., 2/3) for model construction
  - Test set (e.g., 1/3) for accuracy estimation
- Random sampling: a variation of holdout
  - Repeat holdout k times, accuracy = avg. of the accuracies obtained

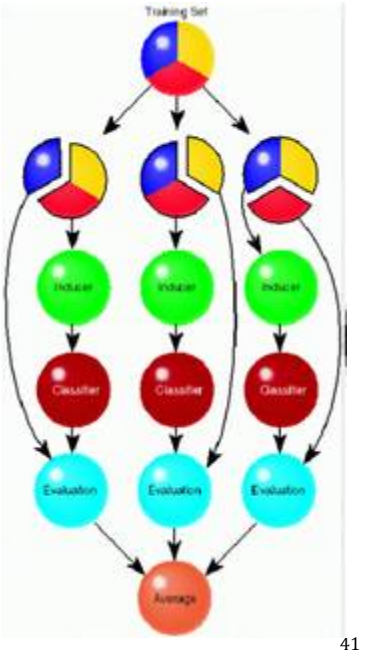
Cross-validation (*k*-fold, where k = 10 is most popular)

- Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
- At *i*-th iteration, use D_i as test set and others as training set
- Leave-one-out: k folds where k = # of tuples, for small sized data
- Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

40

Cross validation

- Repeat multiple times with different hold-out or test sets
- 10 fold cross validation very often used



41

Other ways to evaluate the quality

- Speed and scalability
- Time to construct/use the model
- Robustness
- Handling noise and missing values
- Scalability
- Efficiency in large databases
- Interpretability:
- Easy understanding and insight
- Size of results

42

Evaluating the significance of results

- Standard approach is the so-called t-test and also the Z-test; for pair-wise comparisons
- Have to be carefully used, since relevant for two normal distributions
- If sample size is to small, the tests may lead to misleading results

Alternatives

- Friedman Test: no assumptions about the distribution
- ANOVA test (normal distribution)

43

Evaluation of results and model selection

- Domain dependent
- Depends on our goal (e.g. highly accurate versus finding outliers)

Issues

- "Of high utility?"
- Significant
- Making sense of HUGE results
- Combination and multi-strategy learning

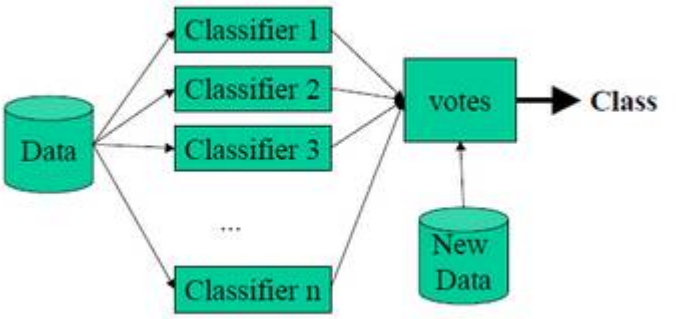
44

More classification algorithms

- Ensembles
- Support Vector Machines
- Lazy Learners

45

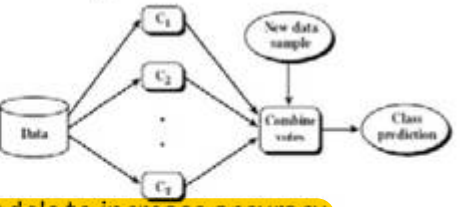
2: Ensembles or meta-learners to improve accuracy: general idea



Boosting, bagging and stacking

46

Improving accuracy: Ensembles



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M1, M2, …, Mk, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Stacking: combining a set of heterogeneous classifiers

7

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

48

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks over-fitting the model to misclassified data

49

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, $(\mathbf{X_1}, y_1), \ldots, (\mathbf{X_d}, y_d)$
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: $err(\mathbf{X_j})$ is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

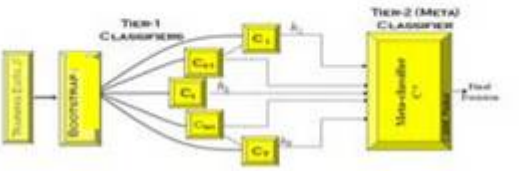
$$error\ (M_i) = \sum_{j}^{d} w_j \times err\ (\mathbf{X_j})$$

- The weight of classifier M_i's vote is $\log \frac{1 - error(M_i)}{error(M_i)}$

50

Stacking (also called Blending)

- Analogy: Consult several doctors, who each gives an option. Next, another doctor analyses this diagnosis, and makes a final recommendation (meta-learning)
- Useful to learn different parts of a problem (multi-view)
- How does it work?
  - Given a set D of *d* tuples, a classifier model M_i is learned for each training set D_i, and each classifier Mi returns its class prediction
  - These predictions are used to train another classifier (meta-learner) B, which ends up with its own model



51

Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

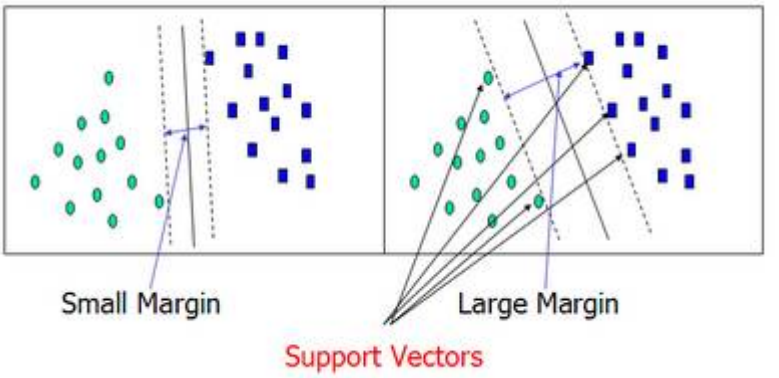
52

Classifier 3: Support Vector Machines (SVMs)

- A classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- **With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane**
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)

53

SVMs: The general idea

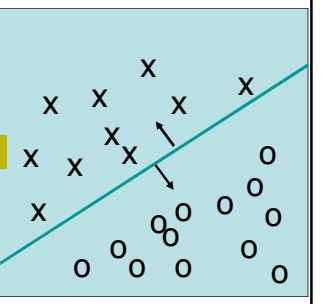


There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)
SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

54

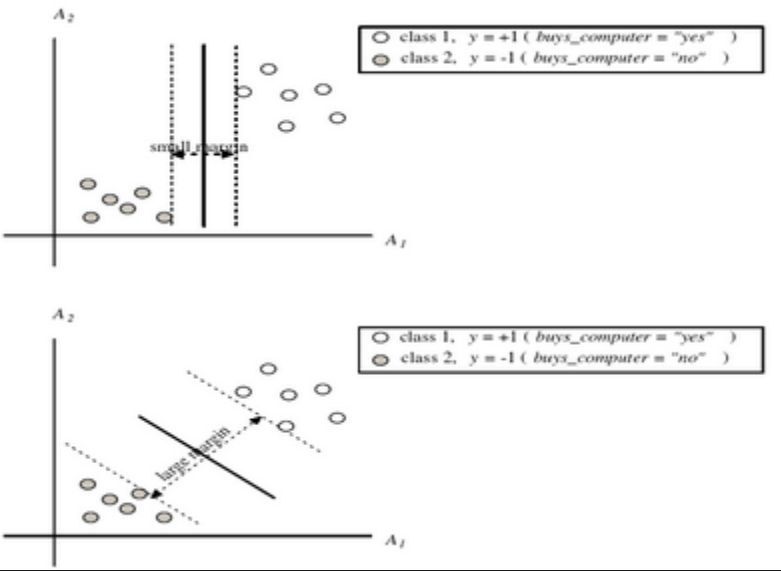
Classification: A Mathematical Mapping

- Classification: predicts categorical class labels
  - E.g., Personal homepage classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
    - x_1 : # of word "homepage"
    - x_2 : # of word "welcome"
- Mathematically, $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$
  - We want to derive a function f: X → Y
- Linear Classification
  - Binary Classification problem
  - Data above the blue line belongs to class 'x'
  - Data below blue line belongs to class 'o'
  - Examples: SVM, Perceptron, Probabilistic Classifiers



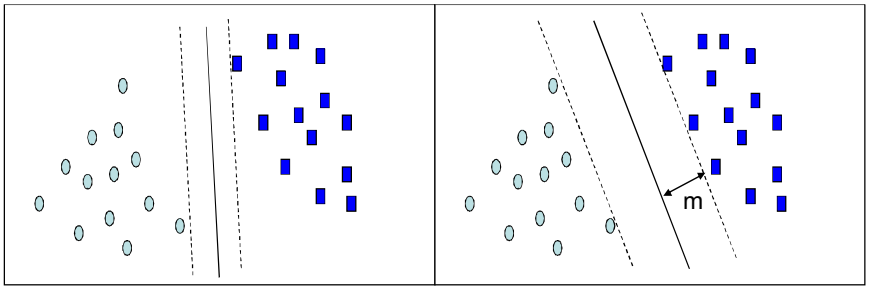
55

SVM—Margins and Support Vectors



56

SVM—When Data is Linearly Separable
(two classes)



Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

57

SVM—Linearly Separable

- A separating hyperplane may be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it may be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

58

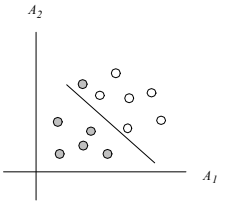
Why is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found!
- Thus, SVMs with a small number of support vectors may have good generalization, even when the dimensionality of the data is high

59

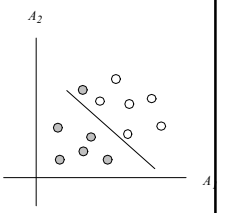
SVMs: What if the data is not linearly separable?

- Transform the original input data into a higher dimensional space
- Search for a linear separating hyperplane in the new space
- **Works well for highly dimensional data**



60

SVM—Linearly Inseparable



- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $X = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$

■

- Search for a linear separating hyperplane in the new space

61

SVM Related Links

- SVM "Historic" website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - Available in **WEKA (called SMO)** and also in **R (SVM package)**

62

Classifier 5: Lazy Learning…

63

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

64

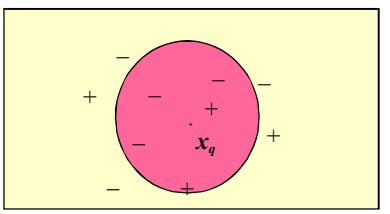
Lazy Learner: Instance-Based Methods

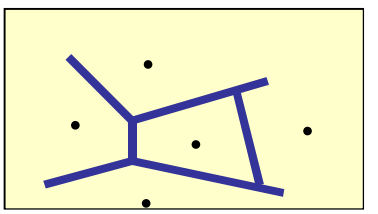
- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

65

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples





66

Discussion on the *k*-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

67

Classification: The Essence

- The algorithms use different statistical measures and procedures to build a model
- No "universal" algorithm that always outperforms (No Free Lunch Theorem)
- Choice highly domain-dependent

68

Other popular families of methods

- Bayesian belief networks and Naïve Bayes Learners
- Neural networks and Deep Learning
- Rough sets
- Rule induction (e.g. through set covering)

69

Next

Cluster analysis
(Unsupervised learning)

70