Day 1: Machine learning

Supervised and
Un-supervised Learning

Decision Trees, Random Forest, SVM,
One class SVM, Anomaly detection with LOF

Sutharshan Rajasegarar
School of Information Technology
Deakin University

Outline

Supervised Learning

- Induction of Decision Tree (DT)
  - Basic Algorithm
  - Heuristics: Entropy & Information Gain
- Convert DT to Decision rules
- Evaluation the Quality of DT
- Decision Tree Pruning
- Ensemble learning: Bagging, Boosting, Random forest
- Support vector machine

Unsupervicew

- Anomaly detection with One class SVM, LOF

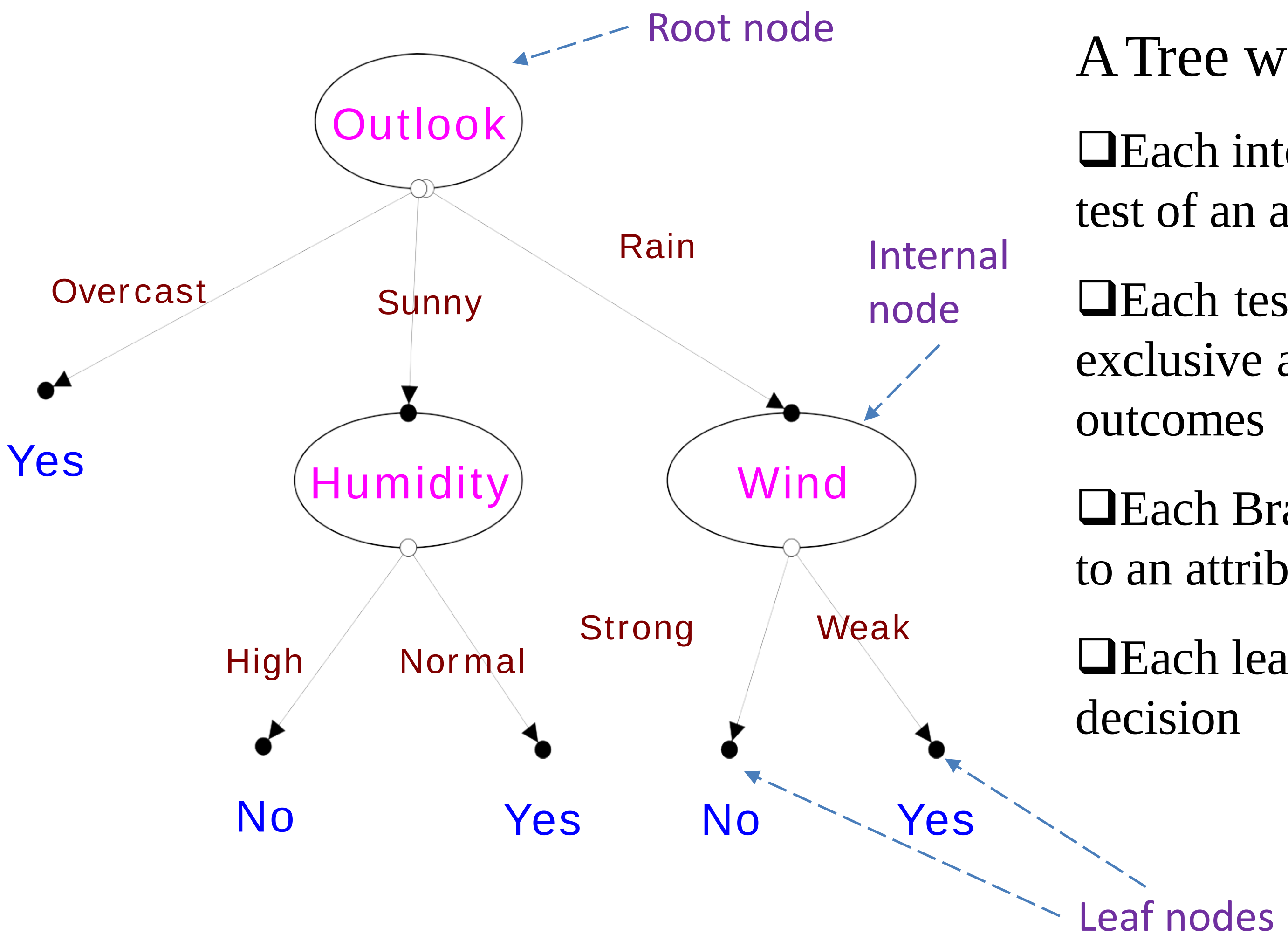
Decision Tree

Weather data: Play Football

Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Is Saturday morning suitable for playing football?
Given that: Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong

Decision Tree Representation



A Tree whose

- Each internal node is a test of an attribute
- Each test has mutually exclusive and exhaustive outcomes
- Each Branch corresponds to an attribute value
- Each leaf node assigns a decision

Other Knowledge Representation

- Logic Expression for PLAY_FOOTBALL=YES
  - (Outlook=Sunny $\wedge$ Humidity=Normal) $\vee$
    (Outlook=Overcast) $\vee$
    (Outlook=Rain $\wedge$ Wind=Weak)
- Production Rules
  - IF (Outlook=Sunny $\wedge$ Humidity=Normal) THEN Yes
  - IF (Outlook=Overcast) THEN No
  - …

Advantages of Decision Trees

- Natural and Succinct
- Suitable for Classification Problems
  - Classify an example into one of a discrete set of possible values

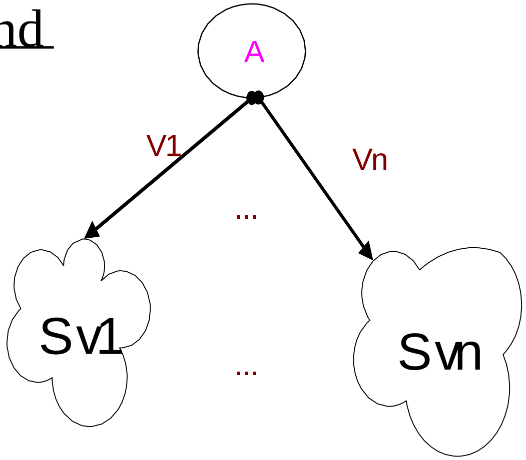
How to Build a Decision Tree from Training Data Set AUTOMATICALLY?

Brief history of Decision Tree Construction

- The first decision tree algorithm is CLS (Concept Learning System)
  - E.B.Hunt, J.Martin, and P.T.Stone's book published by Academic Press in 1966
- The algorithm raising the interests in Decision Tree is ID3
  - J.R. Quinlan's paper in a book edited by D. Michie, published by Gordon and Breach in 1979
    - Uses information ratio to select attribute
- The current decision tree algorithms include C4.5 (and C5)
  - J.R.Quinlan's book published by Morgan Kaufmann in 1993
  - C4.5 is known as J48 in WEKA.
  - Uses gain ratio to select attribute
- The most popular decision tree algorithm that can be used in regression is CART (Classification and Regression Tree)
  - L.Breiman, J.H.Friedman, R.A.Olshen, and C.J.Stone's book published by Wadsworth in 1984
  - uses Gini index to select attribute.

ID3: Learning of Decision Trees

- ID3, Concept Learning System(CLS) algorithm
  - Create a root node for the tree
  - IF all examples from S belong to the same class C_j , THEN label the root with C_j and return
  - ELSE
    - Select an attribute A with values $v_1,\ldots,v_n$, and let the root be an Internal node about A
    - Partition the data set S into subset $S_1,\ldots,S_n$ according to the values of attribute A
    - Apply the algorithm recursively to each subset $S_1,\ldots,S_n$



How to Select the best attribute?

Search Heuristics

- Which is the attribute that is most useful for classifying examples?
- Information Gain
  - How many information contained by Test of an attribute

Entropy

- Given a data set S about C1,…,Cj classes
- Entropy E(S)
  - Measures the uncertainty of the data set S

$$E(S) = -\sum_{C=1}^{j} P_C \times \log_2 P_C$$

  - P_C is the probability of class C, i.e., proportion of the data that belong to class C.
  - The Range of E(S) is 0 to 1
    - It is 0 if all members of S belong to the same class
    - It is 1 if S is completely random
  - The less the Entropy, the more predictable for S

Information Gain

- Information Gain
  - Expected reduction in Entropy of S due to the result of attribute A

$$Gain(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} \times E(S_v)$$

- The larger the Information Gain, the more informative the attribute A

When to Stop?

- **Stopping Criterion**
  - If all examples are classified perfectly, OR
  - All attributes are used
    - Label the leaf with the most possible class value in the sub training data set.

Example:
From Data to Decision Tree

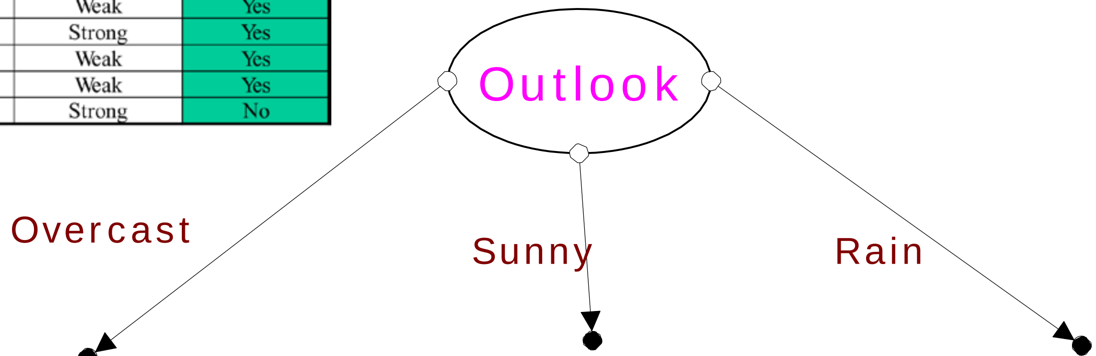
Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

- S={D1,D2,…,D14}, written as [9+,5-]
- Class: {Yes, No} for Play_Football
  - Entropy of S:
    $E(S)=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)=0.940$
- Which Attribute as Root of the Tree?
  - Compare the Information gain of each attribute

Example

Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



{D3,D7,D12, D13} [4+,0-]

E=0

$$E = -\frac{4}{4}\log_2\left(\frac{4}{4}\right) - 0\log_2(0)$$
$$= \log_2(1) - 0 = 0$$

{D1,D2,D8,D9, D11} [2+,3-]

E=0.970

$$E = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$
$$= 0.970$$

{D4,D5,D10,D14} [3+,2-]

E=0.970

$$E = -\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)$$
$$= 0.970$$

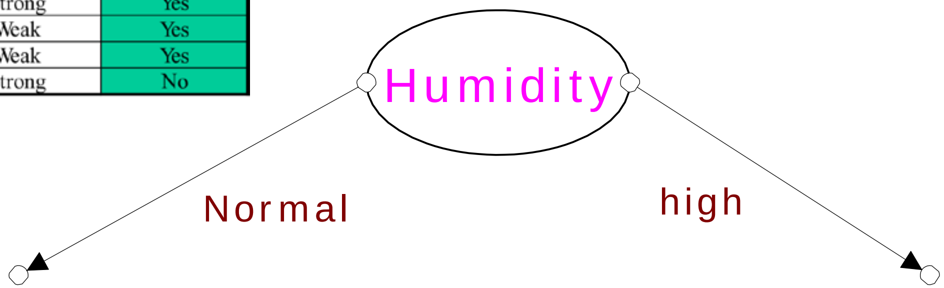
Gain(S,Outlook) = $E(S) - \left(\frac{4}{14}\right) * 0 - \left(\frac{5}{14}\right) * 0.970 - \left(\frac{5}{14}\right) * 0.970$

$= 0.940 - \left(\frac{4}{14}\right) * 0 - \left(\frac{5}{14}\right) * 0.970 - \left(\frac{5}{14}\right) * 0.970$

$= 0.246$

Example

Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



{D5,D6,D7,D9,D10, D11,D13} [6+,1-]

E=0.592

$$E = -\frac{6}{7}\log_2\left(\frac{6}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)$$
$$= 0.592$$

{D1,D2,D3,D4,D8, D12,D14} [3+,4-]

E=0.985

$$E = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$
$$= 0.985$$

$$\text{Gain(S,Humidity)} = 0.940 - \left(\frac{7}{14}\right) * 0.592 - \left(\frac{7}{14}\right) * 0.985$$
$$= 0.151$$

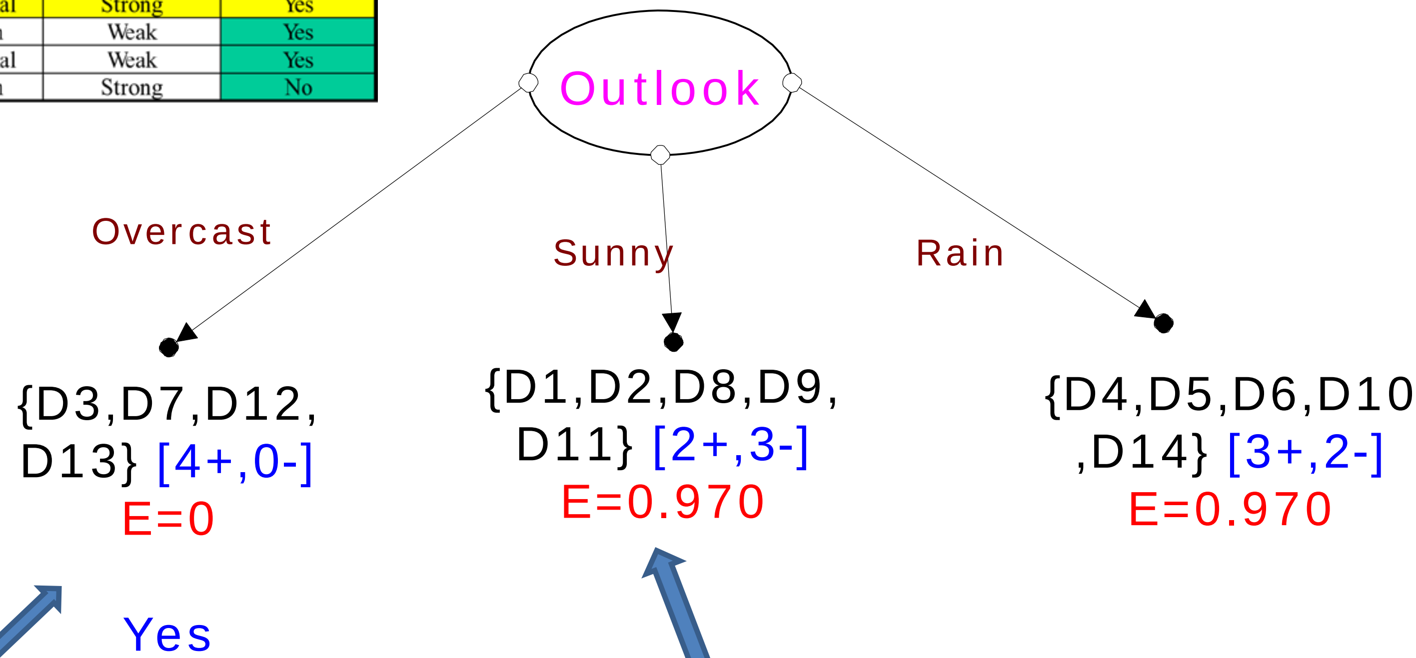
Example

- Gain(S,Wind)=0.180
- Gain(S,Temperature)=0.029
- ✓ Gain(S,Outlook)=0.246
- Gain(S,Humidity)=0.151

Select the attribute with the largest gain as the (root) node

Example

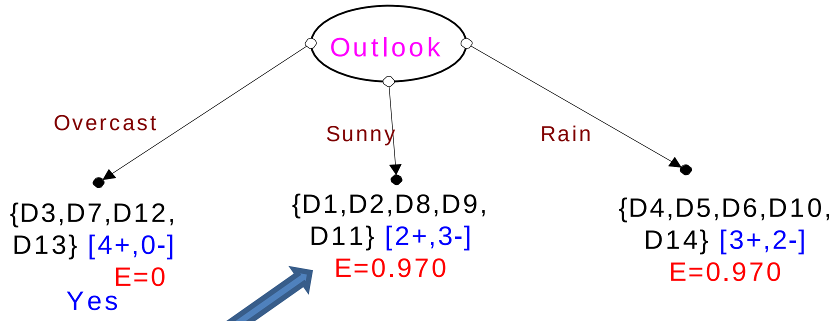
Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Which Attribute to test here?

All the values belong to one of the class ("Yes" here), i.e, E=0.
Hence this becomes a leaf node "Yes"

Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Which Attribute to test here?

Data in "Sunny" are: $S_{sunny} = \{D1,D2,D8,D9,D11\}$ **[2+,3-]**

$$E(S_{sunny}) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.970$$

For Humidity:

Normal [2+,0-] => $E = -\frac{2}{2}\log_2\left(\frac{2}{2}\right) - 0 = 0$

High [0+,3-] => $E = -\frac{3}{3}\log_2\left(\frac{3}{3}\right) - 0 = 0$

$$\text{Gain}(S_{sunny}, \text{Humidity}) = 0.970 - \frac{2}{5} * 0 - \frac{3}{5} * 0 = 0.970$$

For Temperature:

Hot [0+,2-] => $E = 0$

Mild [1+,1-] => $E = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$

Cool [1+,0-] => E = 0

$$\text{Gain}(S_{sunny}, \text{Temperature}) = 0.970 - \frac{2}{5} * 1 = 0.570$$

For Wind:

Weak [1+,2-] => $E = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - -\frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.918$

Strong [1+,1-] => $E = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$

$$\text{Gain}(S_{sunny}, \text{Wind}) = 0.970 - \frac{3}{5} * 0.918 - \frac{2}{5} * 1 = 0.0192$$

✓Gain(S_{sunny}, Humidity)=0.970

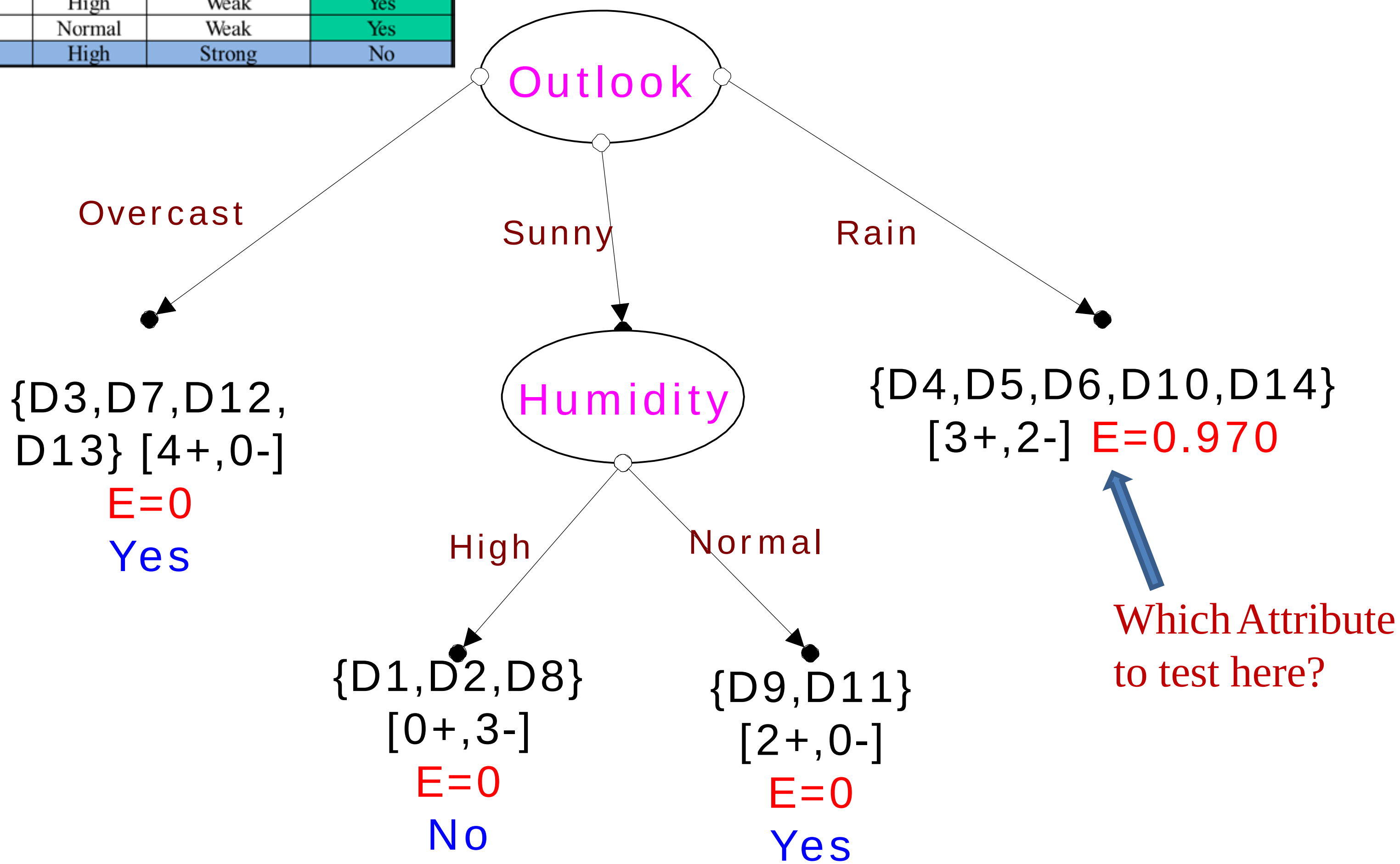
➢Gain(S_{sunny}, Temperature)=0.570

➢Gain(S_{sunny}, Wind)=0.019

Since **Humidity** attribute has the largest gain, select it as the internal node.

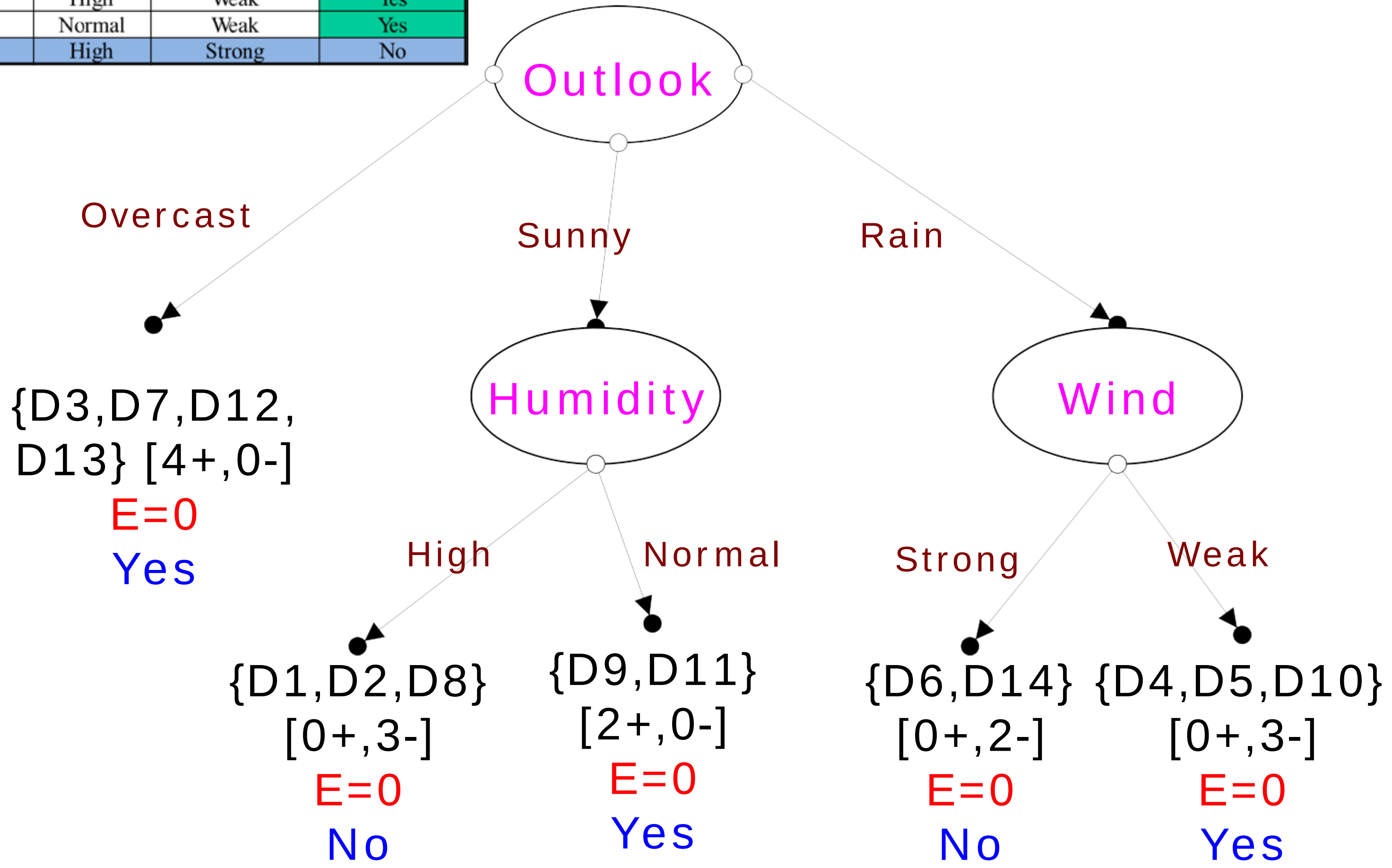
Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

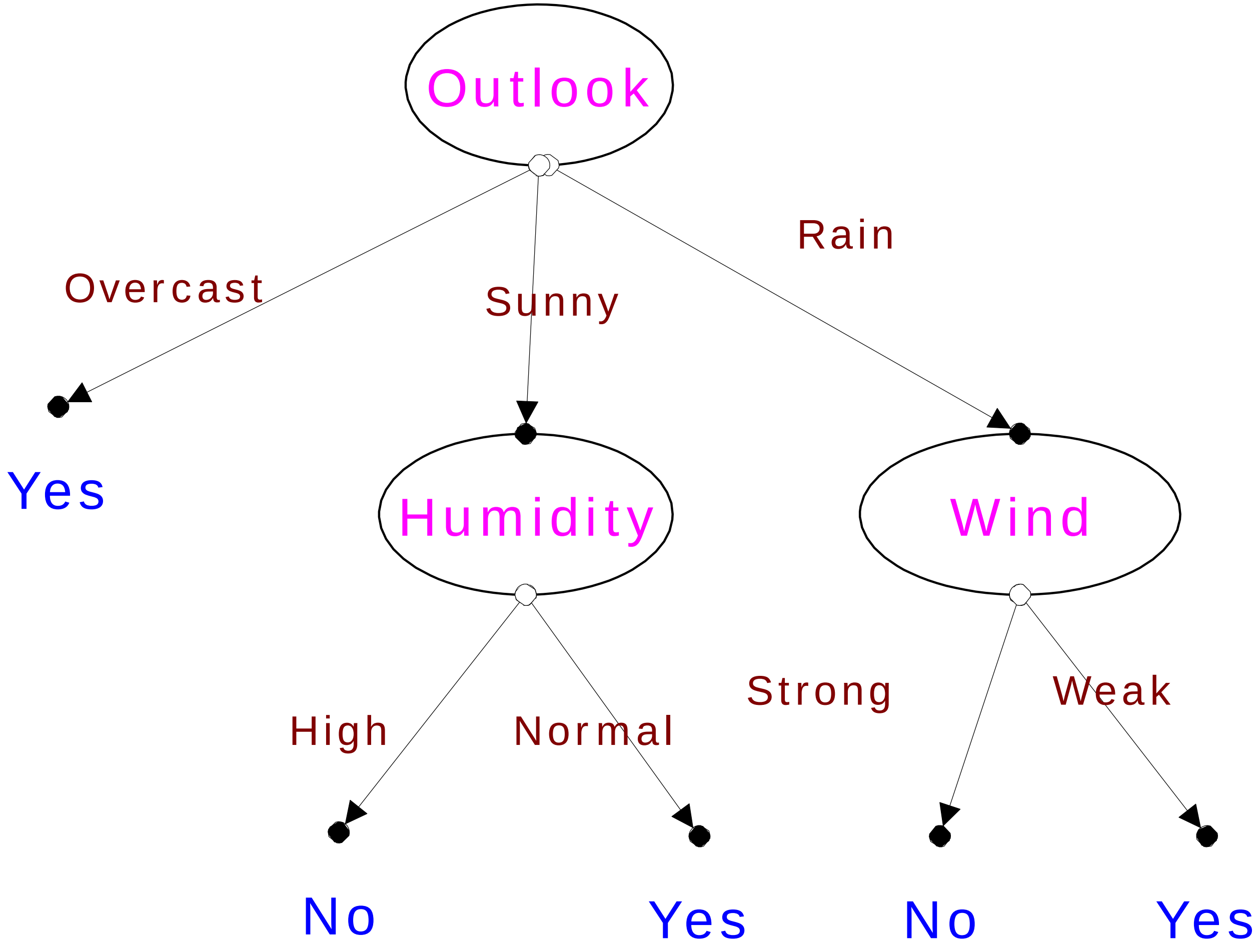


Day	Outlook	Temperature	Humidity	Wind	Play Football
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Weak	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example



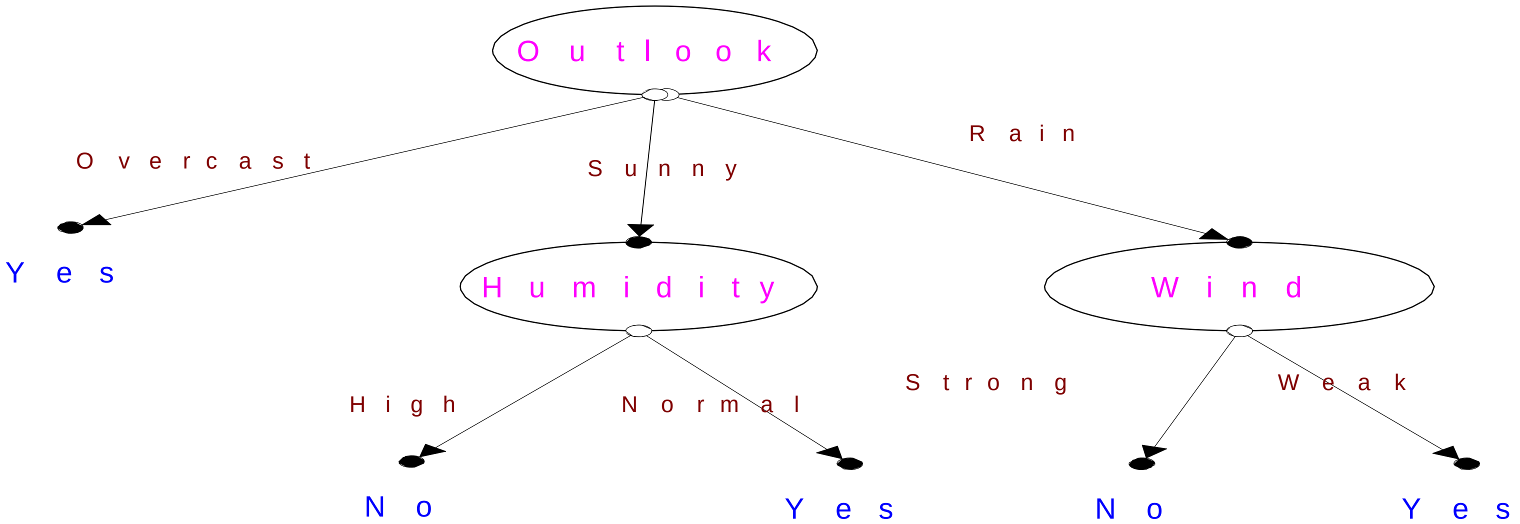
Example



Outlook
Overcast
Sunny
Rain
Yes
Humidity
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Map Decision Tree to rules

- A rule is created for each path from the root to a leaf.
- Each attribute-value pair along a given path forms a conjunction in the rule antecedent.
- The class label held by the leaf forms the rule consequent



- IF outlook=overcast THEN play=yes
- IF outlook=sunny AND humidity=High THEN play =no
- IF outlook=sunny AND humidity=normal THEN play =yes
- IF outlook= rain AND wind=strong THEN play =no
- IF outlook= rain AND wind=weak THEN play =yes

Evaluate the quality of a decision tree (I)

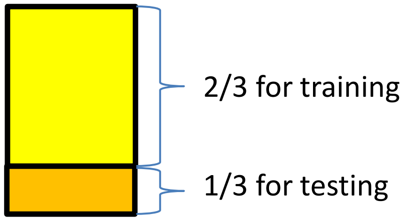
- Interpretability:
  - The level of comprehensibility of the model
  - The simpler, the better
- Classification Accuracy
  - The ability of the model to correctly predict unseen instances

Evaluate the quality of a decision tree (II)

- Two methods to evaluate predictive accuracy

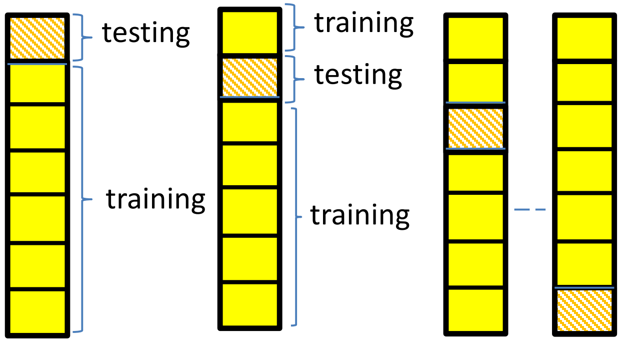
– **Hold-out:**

  - Partition the data set into two independent subsets, i.e. a training data set and a test data set (say, 2/3 for data training, 1/3 for testing)



– **K-fold cross-validation**

  - Partition the data set into k mutually exclusive subsets with approximately equal size.
  - Perform training and test for k times.
  - In i-th time, the i-th subset is used for test while the rest of the subsets are collectively used for training
  - 10-fold cross-validation is often used.



Yellow portions are used for training and the red portion is used for testing.

Prune Decision Tree (I)

- **Why need pruning?**
  - Overfitting: the trained model approximates the training data set so much that it deviates the real distribution of the instance space
  - The main reason is that the training data set may be polluted by noise
  - When a decision tree is built, many branches may reflect anomalies in the training set due to noise or outliers
  - Pruning is used to address the problem of overfitting

Prune Decision Tree (II)

- **How to prune?**
  - Goal: balance the tree complexity and the performance
- **Pre-pruning:**
  - Terminate tree construction early: do not split a node if it would result in the goodness measure falling below a threshold
- **Post-pruning:**
  - Remove branches from a fully grown tree:
    - Get a sequence of pruned trees
    - Use a set of data different from the training data set to decide which is the best pruned tree

Decision Trees: Advantages

- Decision trees are very easy to understand, as they represent rules.
- Decision trees are capable of modelling nonlinear functions.
- Decision tree can handle categorical variable (i.e. where weather being "sunny" vs "cloudy" where we cannot compute Euclidean distance between two vectors having weather as variable.)

Decision Trees - Disadvantages

- Sensitive to small changes in the data
- May overfit easily
- Trees may not be as competitive in terms of accuracy as some of the other regression and classification techniques such as SVM or neural networks.

Readings

- **Readings:**
  - **Tom Mitchell, Machine Learning, Ch 3.**
  - **I. Witten & E. Frank, Data Mining, 6.1**
  - **J. Han & M. Kamber, Data Mining, 7.1-3**

Decision tree in R

Run the “DecisionTreeID3.R” file.

- This uses ‘weatherData.csv’ file to perform ID3

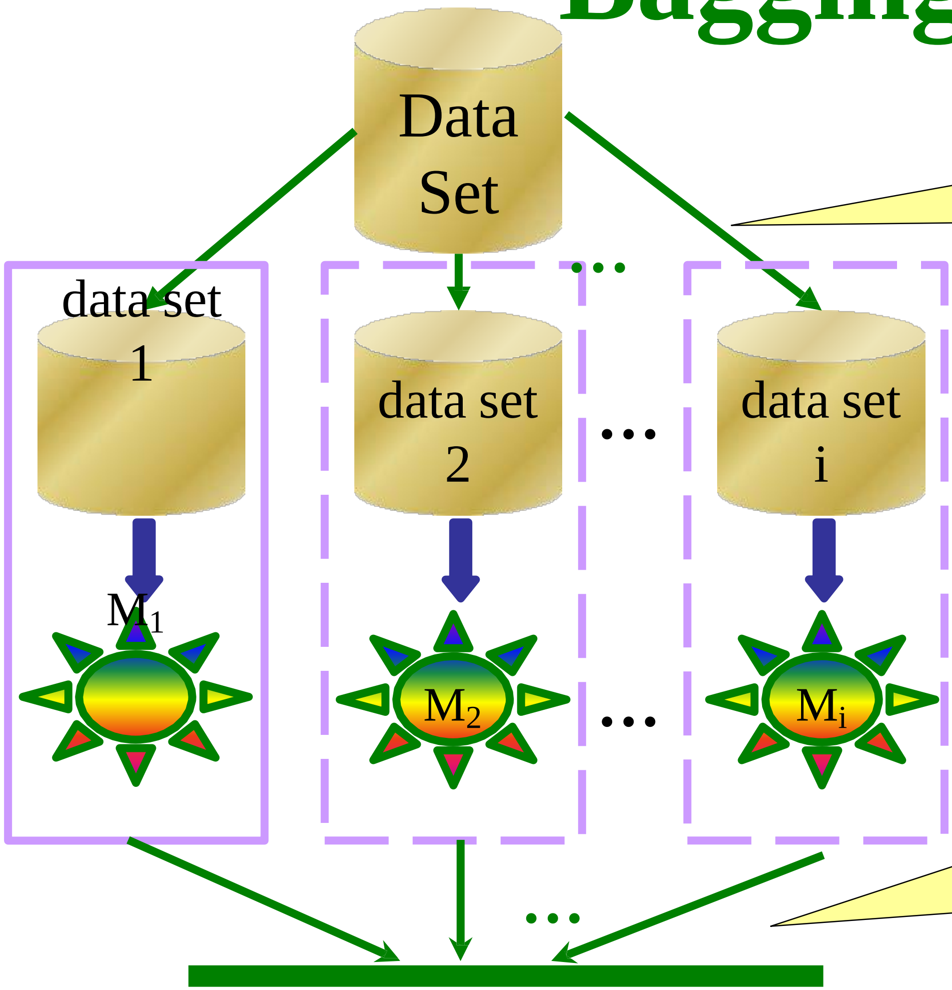
Ensemble Approaches

- Bagging
  - **B**ootStrap **Agg**regat**ing**

- Boosting
  - Adaboost

- Random Forests
  - Bagging reborn

Bagging



Bootstrap a set of data sets:

generate many data sets from the original data set through bootstrap sampling (sampling with replacement)

Train each component model:

from each bootstrap sampling data, learn one model from it.

Model Integration:

1. For classification, the output is the class label receiving the most number of votes
2. For regression, the output is the average output of the component models.

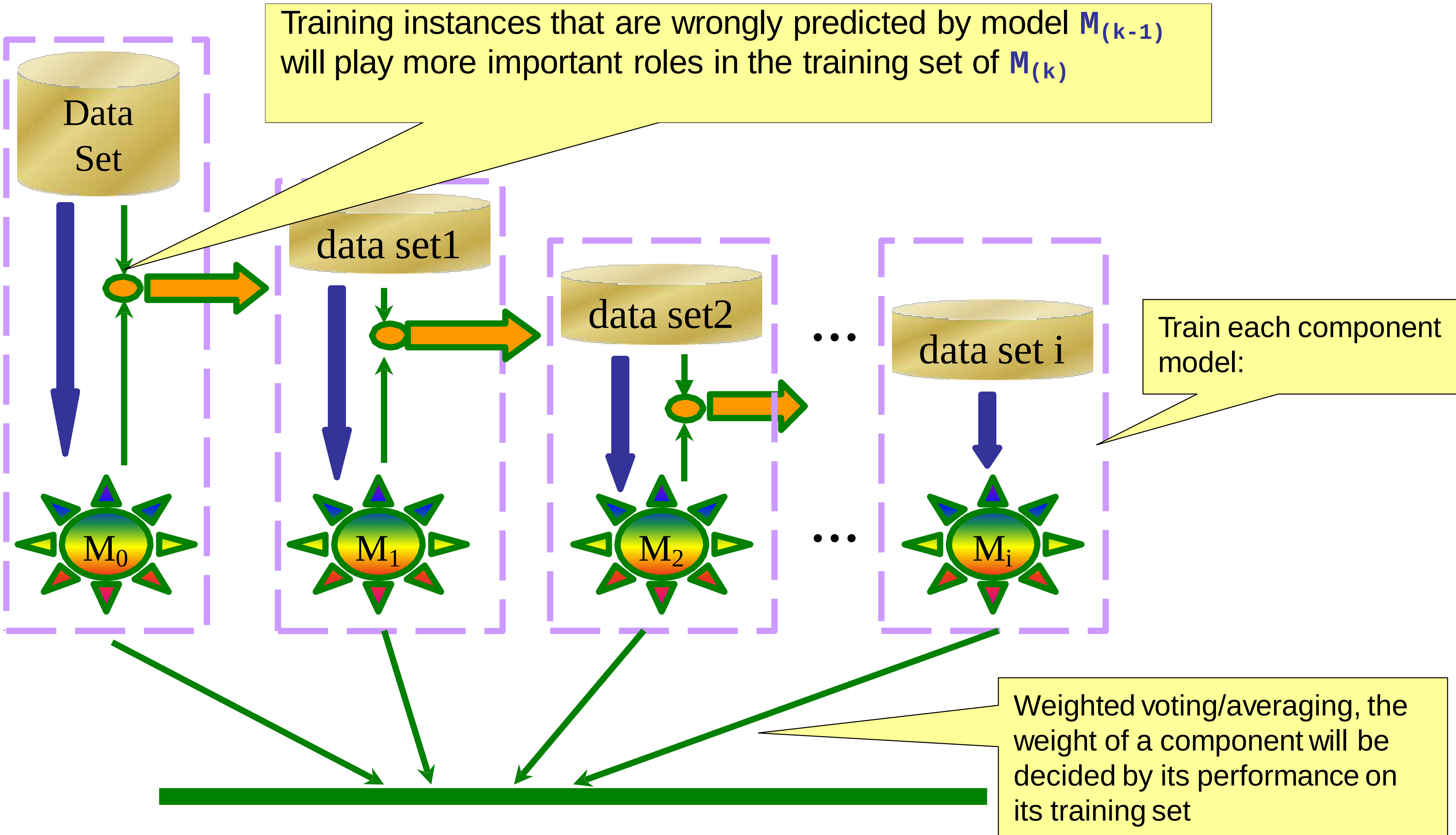
Bagging Details

- On average each Bootstrap sample has **63**% instances
  - Encourages models to have uncorrelated errors
- Usually set the **$i \approx 30$**
  - Or we use cross-validation to pick the value
- The base learner needs to be unstable
  - Usually full length (or slightly pruned) decision trees or ANN

Boosting

- The Boosting Assumption
- The Adaboost Algorithm

Boosting


Training instances that are wrongly predicted by model $M_{(k-1)}$ will play more important roles in the training set of $M_{(k)}$
Data Set
data set1
data set2
data set i
M_0
M_1
M_2
M_i
Train each component model:
Weighted voting/averaging, the weight of a component will be decided by its performance on its training set

Boosting

- Boost a sequence of models
  - Generate a sequence of component models, where ***the training sets of the successors*** are determined by ***the performance of the predecessors***.

- Each predictor is created by using a biased sample of the training data
  - ***instances with high error*** are ***weighted higher*** than those with lower error.
  - ***Difficult instances get more attention***
    - this is the motivation behind boosting

Adaboost Algorithm

Input:

- N instances $S_N=\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$
- a base learner $h = h(\underline{w}, \underline{x})$

Initialize: equal instance weights $w_i=1/N$ for all $i=1..N$

Iterate for $t=1..T$:

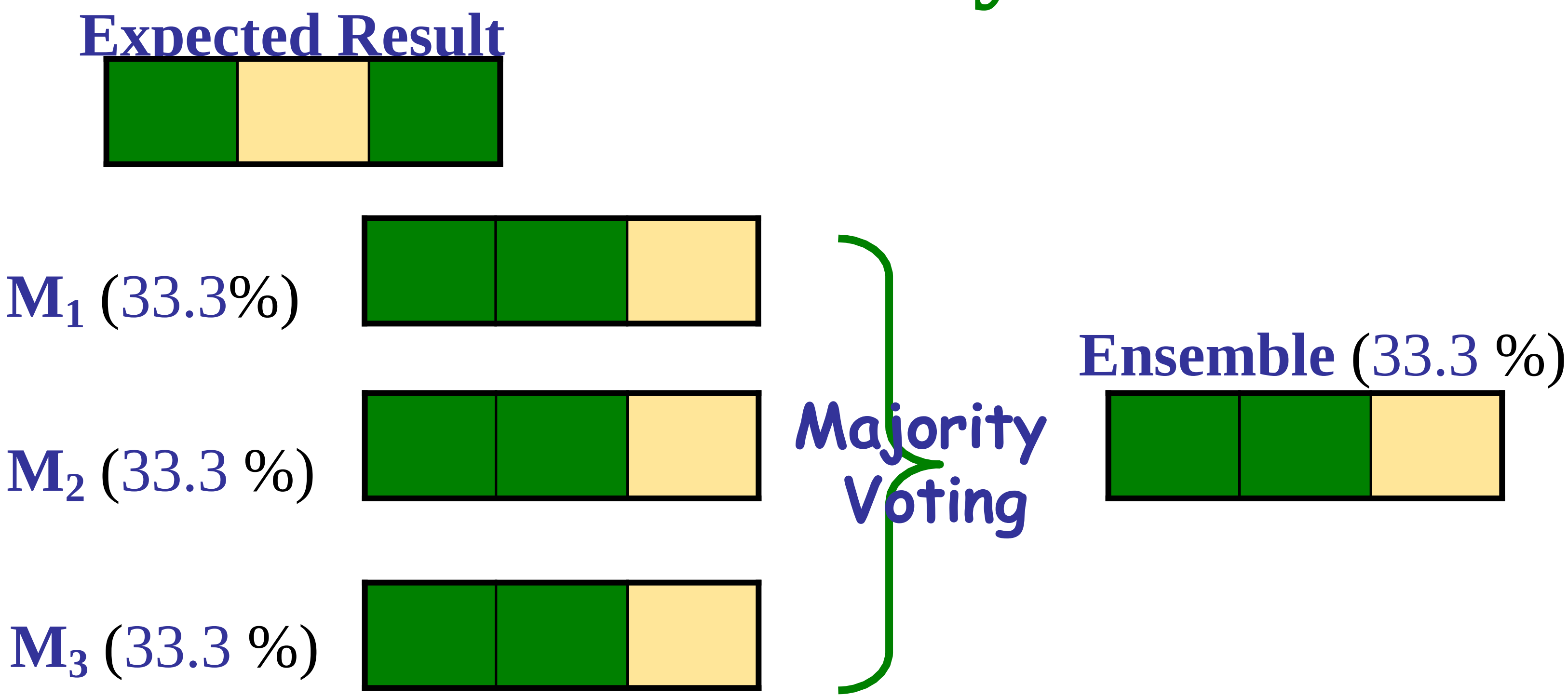
1. train base learner according to weighted data set $(\underline{w}^{(t)}, \underline{x})$ and obtain model $h_t = h(\underline{w}^{(t)}, \underline{x})$
2. compute model error ε_t
3. compute model weight α_t
4. update instance weights for next iteration $w^{(t+1)}$

Output: final model as a ***linear combination*** of h_t

More on Ensemble Learning

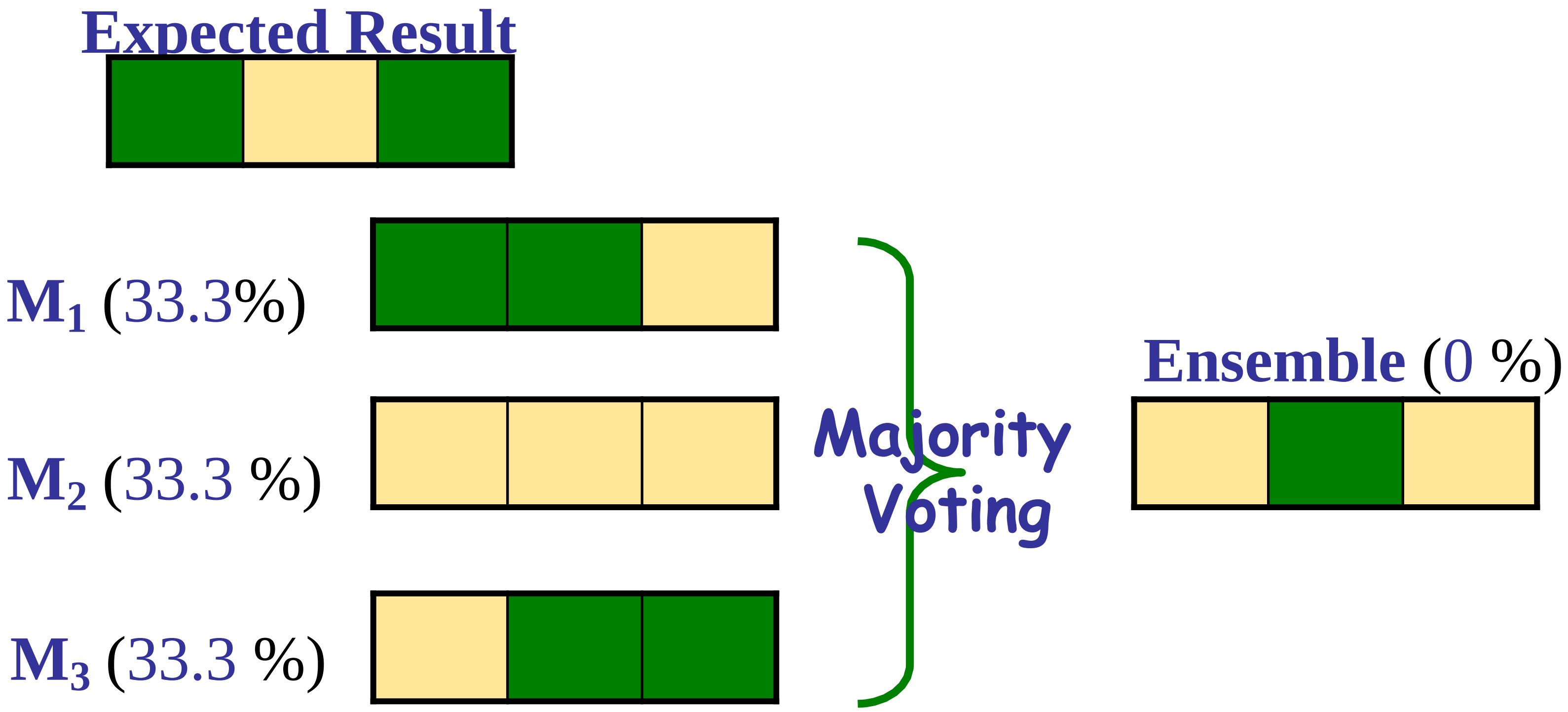
- Is ensemble easy?
- How to choose base learner?

Is Ensemble Easy?



- ***<u>So, the component models can't be identical</u>***

Is Ensemble Easy?



- ***<u>the component models must not be very bad</u>***

Is Ensemble Easy?

- ***The more accurate*** and ***the more diverse, the better***
  - most ensemble methods ***work if and only if*** the base learner are ***better than random guessing***

- How to build an ensemble?
  - ***which algorithm will be used as the base learner ?***
  - ***how many to ensemble together?***

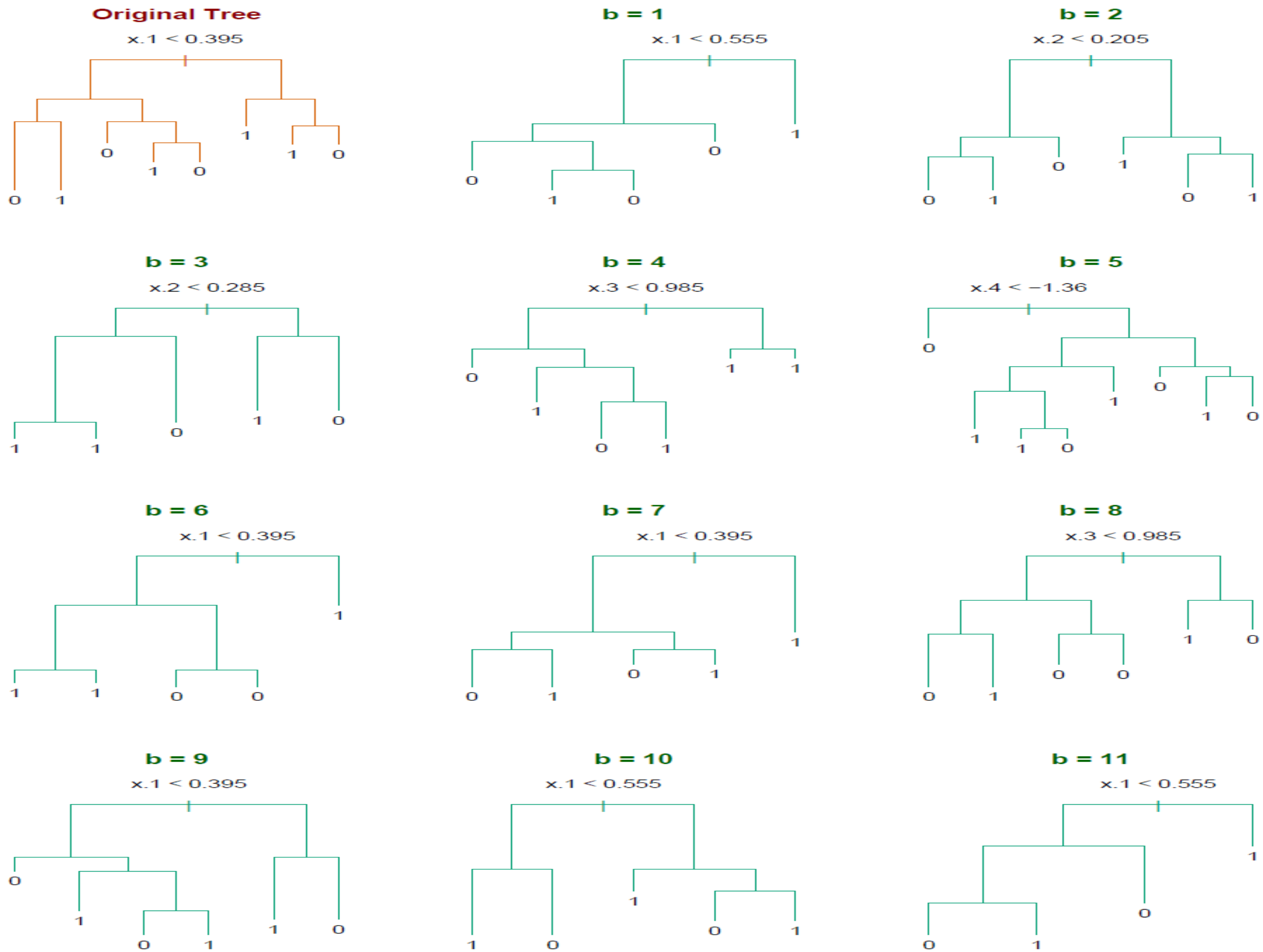
Base Learner

- Generally it is believed that
  - ***Bagging*** works best for those ***unstable classifier***, in which small changes in training data produce large changes in the model
    - e.g. Decision Tree, Neural Network, etc.
  - ***Boosting*** works best for those ***stable classifier***
    - e.g., KNN, SVM, etc.

The More, the better?

- ***The more component models, the better the performance of the ensemble?***
  - More component models means:
    - much more ***computational cost*** in prediction, because more component predictions must be computed
    - much more ***storage cost*** for component models
  - Moreover, generating diverse component models becomes a big challenge, because our training data is not infinite
    - If "the more, the better", then much energy must be spent in designing methods that could exploit the data more efficiently.
  - So, the answer should be "**No**"

Random Forest: Bagging Decision Trees



Random Forest

- Builds upon the idea of bagging
- Each tree is built from a bootstrap sample of data
- Node splits are calculated from random feature subsets.
- A popular choice for the number of features is:
- $m_{try} = \sqrt{Number\ of\ features}.$

Random Forest

All trees are fully grown
No pruning

Two parameters:

Number of trees (T)
Number of features (m_{try})

Random Forest Algorithm

Let T be the number of trees to build.

Training: for each of T iterations:

1. Select a new bootstrap sample from the training set
2. Build an un-pruned tree on this bootstrap sample.
3. At each internal node of the tree, randomly select m_{try} features and determine the best split using only these features.

Testing: Output overall prediction as a mean (or majority vote) from all individually trained trees.

Error Rate in Random Forest

Error rate depends on:

Correlation between trees (lower is better)
Strength of single trees (higher is better)

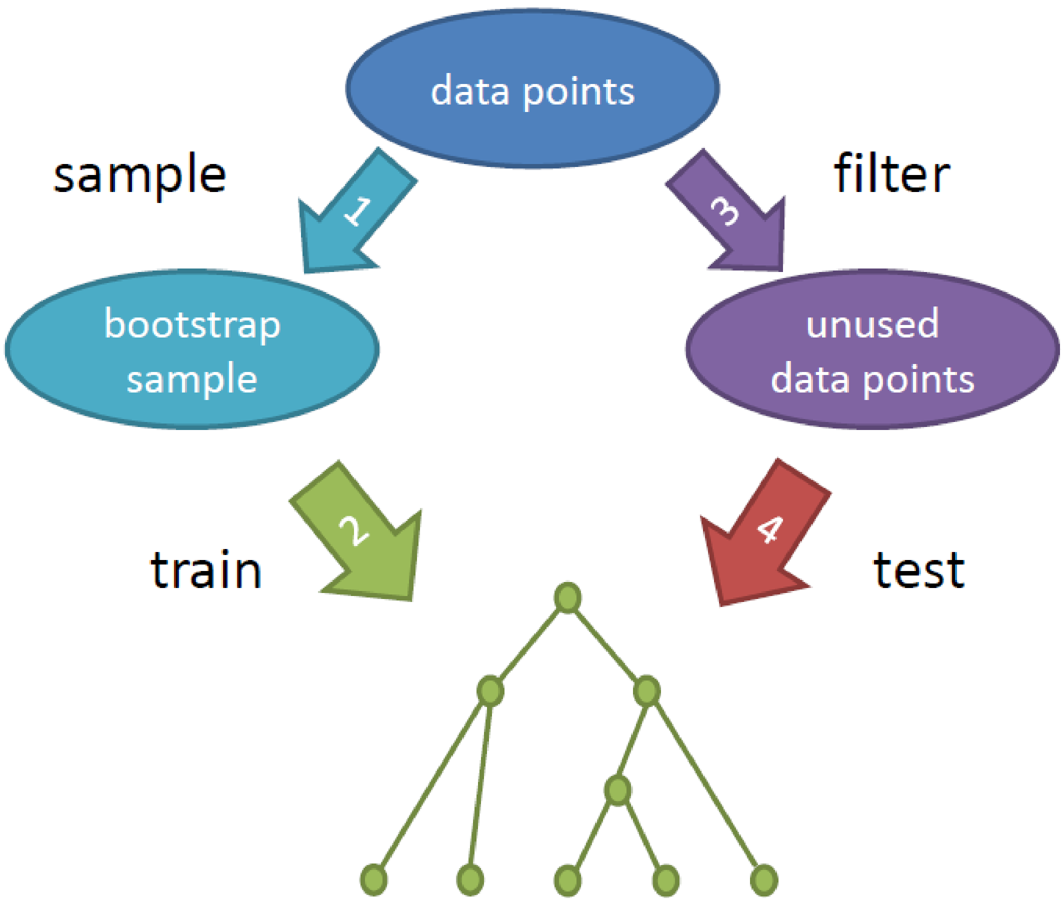
Increasing number of features for each split:

Increases correlation
Increases strength of single trees

Out of Bag Error

- It is possible to estimate the goodness of a bagged model.
- Each tree is trained on a bootstrapped sample. It can be shown that on average, each bagged tree makes use of 2/3 of the training instances.
- The remaining 1/3 of the instances are referred to as the out-of-bag (OOB) instances.
- We can predict the response for the *i*-th observation using each of the trees in which that observation was OOB. This will yield around B/3 predictions for the *i*-th observation, which we average.

Out of Bag Error



Out of Bag Error

Very similar to cross-validation

Measured during training

Advantages/Disadvantages of Random Forest

RF is fast to build. Even faster to predict!

Decision Tree complexity is $O(dn \log n)$. A random forest with T trees would have $O(Tdn \log n)$.

Fully parallelizable … to go even faster!

Ability to handle data without pre-processing

Data does not need to be rescaled, transformed, or modified!

Resistant to outliers

Automatic handling of missing values (a property of decision trees)

Less interpretable results than a single decision tree

Random forest with R

Run “randomForestEx.R”

Good references:

- https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

- "Classification and Regression by randomForest" by Andy Liaw and Matthew Wiener in R News (ISSN 1609-3631).

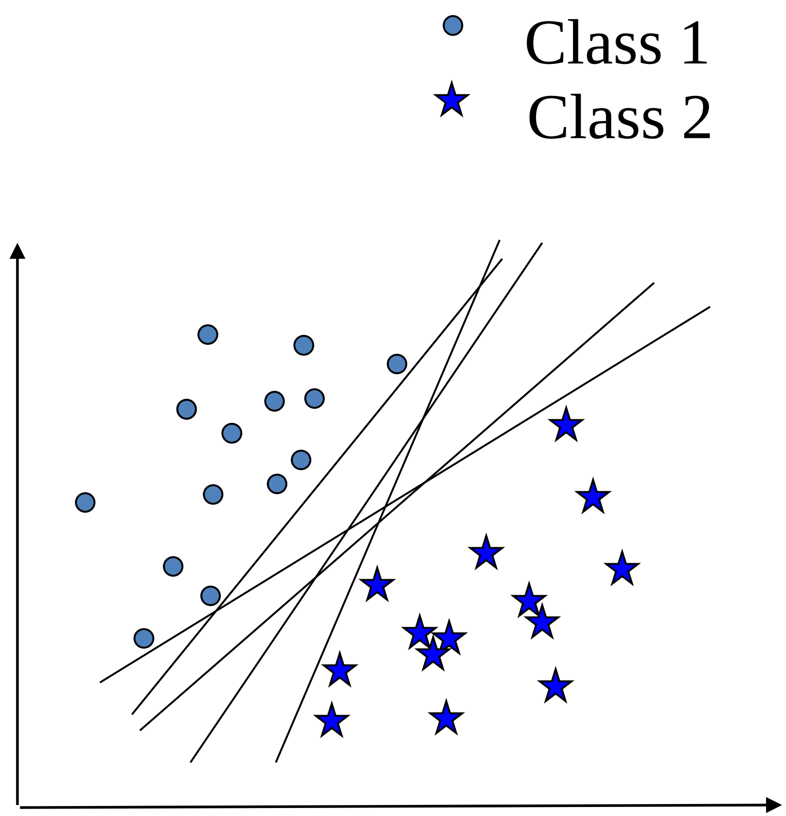
Support vector machine

Support Vector Machine (SVM)

Given training data in different classes (labels known) Predict test data (labels unknown)

Applications

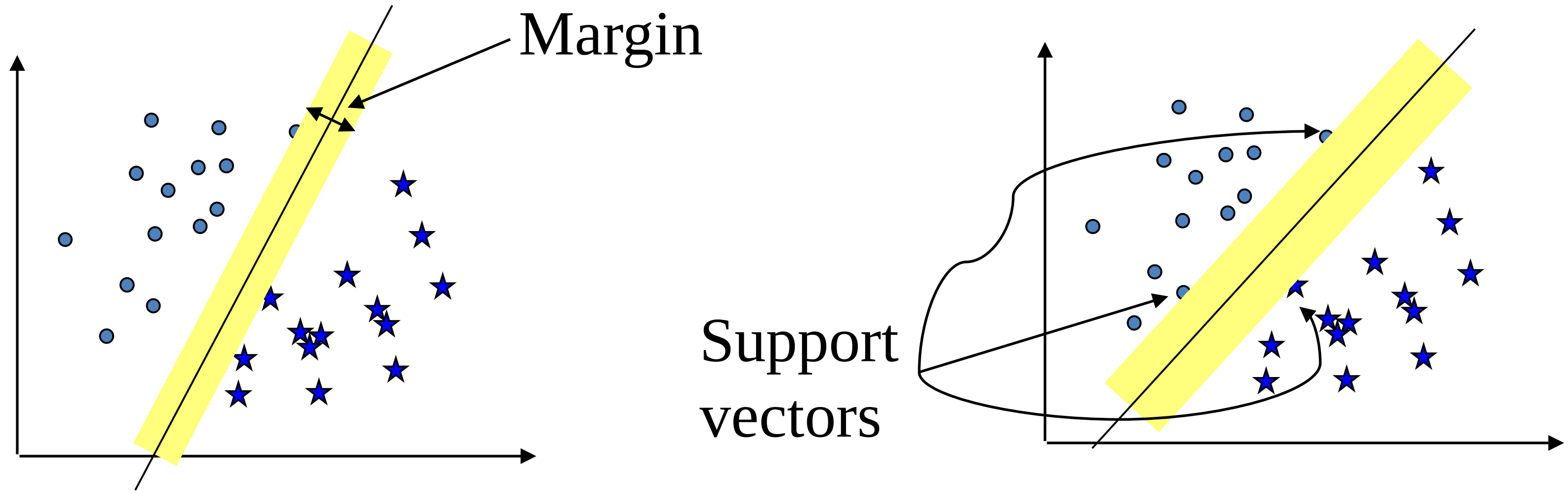
- Handwritten digits recognition
- Text classification
- proteins
- Training and testing



Example:

- Classifying data into two classes

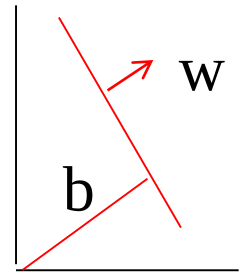
Best separating plane



- Best separating plane is the one that maximizes the margin
- The data points that lie on the margin are called support vectors
- This is the simplest Linear SVM

Support Vector Machine (SVM)

Note: a line can be represented by a normal vector and a distance from the origin



- **Let xi be the data vectors and yi be the labels**

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ belongs to class } 1 \\ -1 & \text{if } x_i \text{ belongs to class } 2 \end{cases}$$

- **Separating hyperplane**

$$w.x + b = 0$$

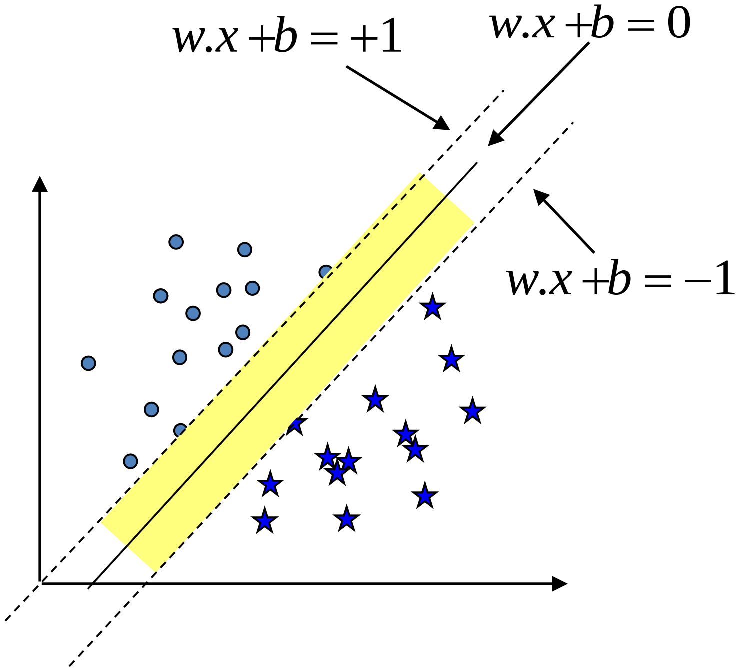
- **select w and b with maximal margin**

$$w.x + b >= 1 \quad \text{if} \quad y_i = 1$$

$$w.x + b <= -1 \quad \text{if} \quad y_i = -1$$

- **Combining the above two**

$$y_i(w.x + b) >= 1 \qquad i = 1, 2, \ldots, n$$



SVM formulation

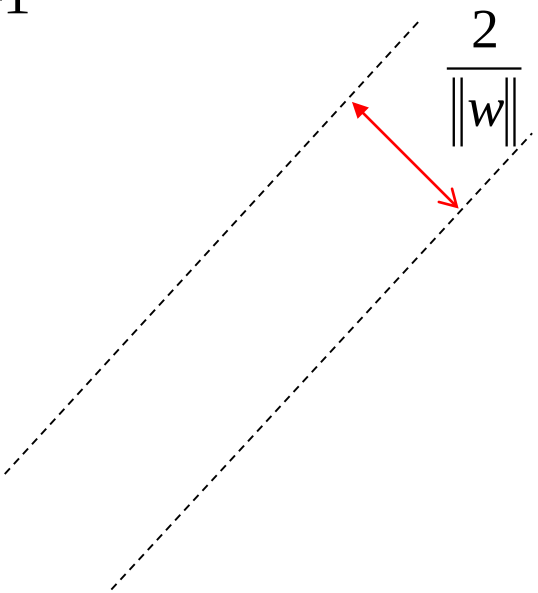
Distance between $w.x + b = 1 \quad and \quad w.x + b = -1$

Is given by $\dfrac{2}{\|w\|}$

Hence to maximise margin: Max $\dfrac{2}{\|w\|}$

Or equivalently minimise $\dfrac{\|w\|}{2}$

Hence the optimum hyperplane is



$$\boxed{\begin{aligned} &\min_{w,b} \frac{\|w\|}{2} \\ &s.t \quad y_i(w.x + b) >= 1 \qquad i = 1,2,\dots,n \end{aligned}}$$

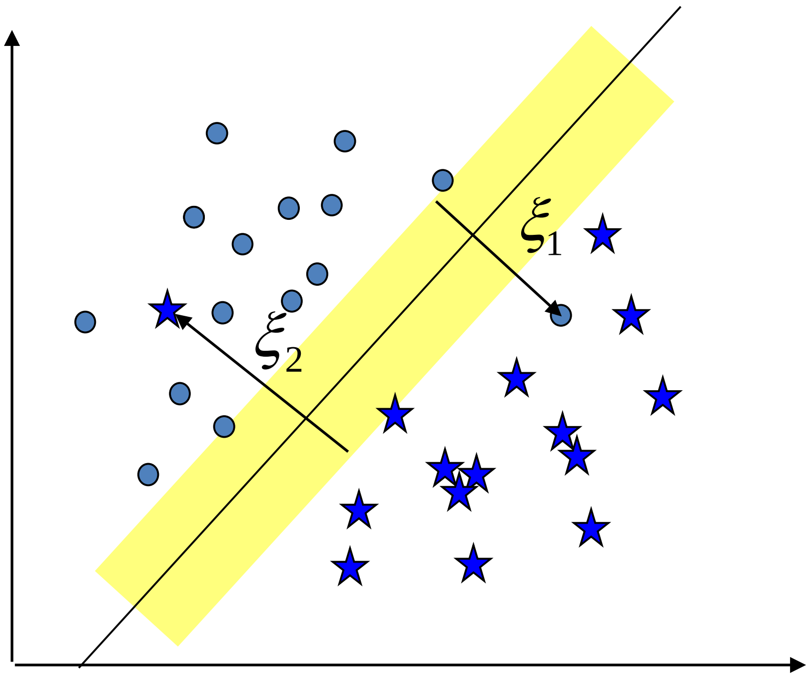
Non linearly separable

Allowing some training errors (soft margin)

$$\min_{w,b,\xi} \frac{\|w\|}{2} + C\sum_{i=1}^{n} \xi_i$$

$$s.t \quad y_i(w.x+b) >= 1-\xi_i$$

$$\xi_i \geq 0 \quad i = 1,2,\ldots,n$$



If $\xi_i > 1$ - point not on the correct side of the separating plane

C –parameter

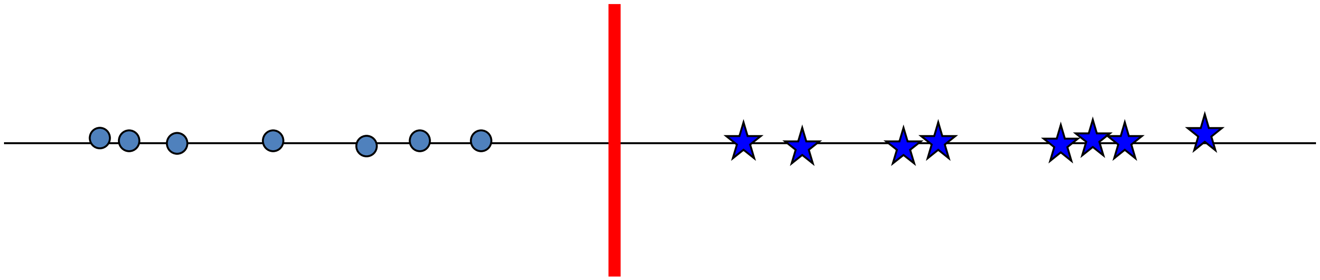
Non linearly separable

Non linearly seperable in the input space – but lineaerly seperable in another space.

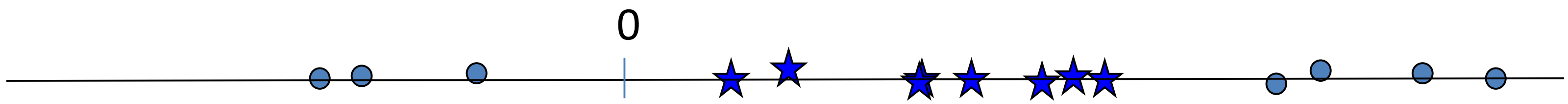
- Transform the data to a higer dimensional space
- Linealy separate in that higher dimensional space

One – dimensional example

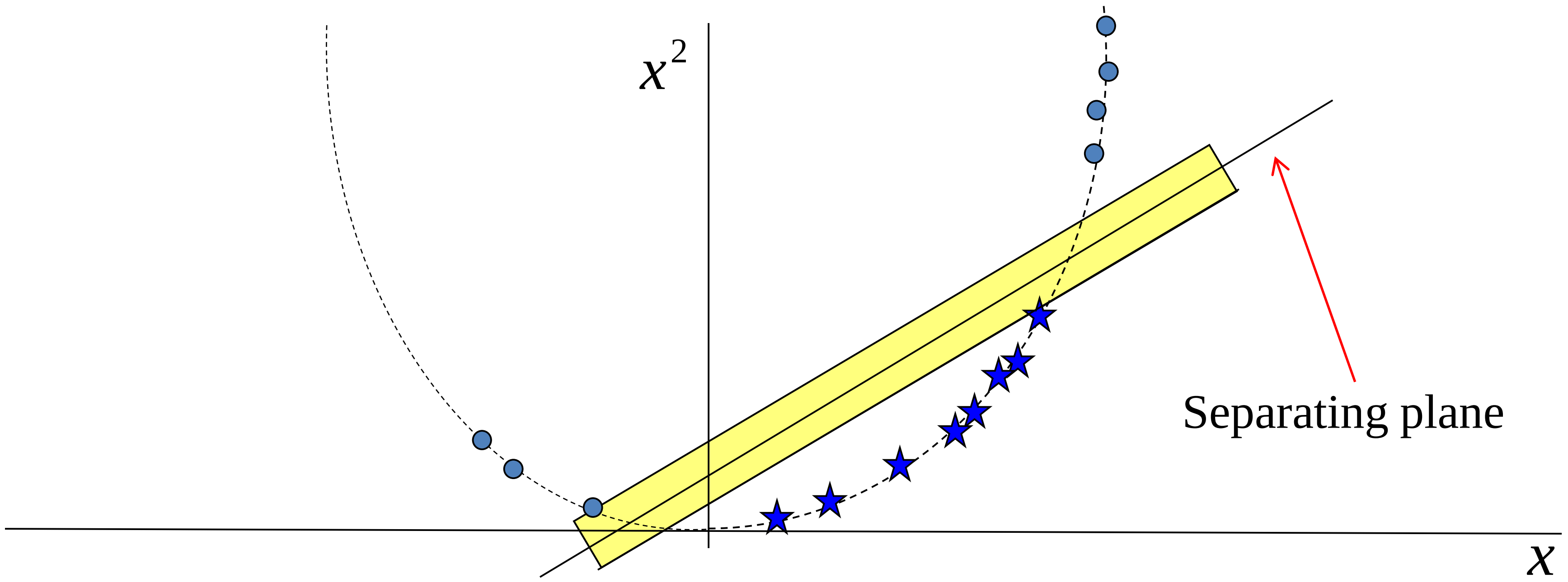
- Linearly seperable:



Linearly non separable

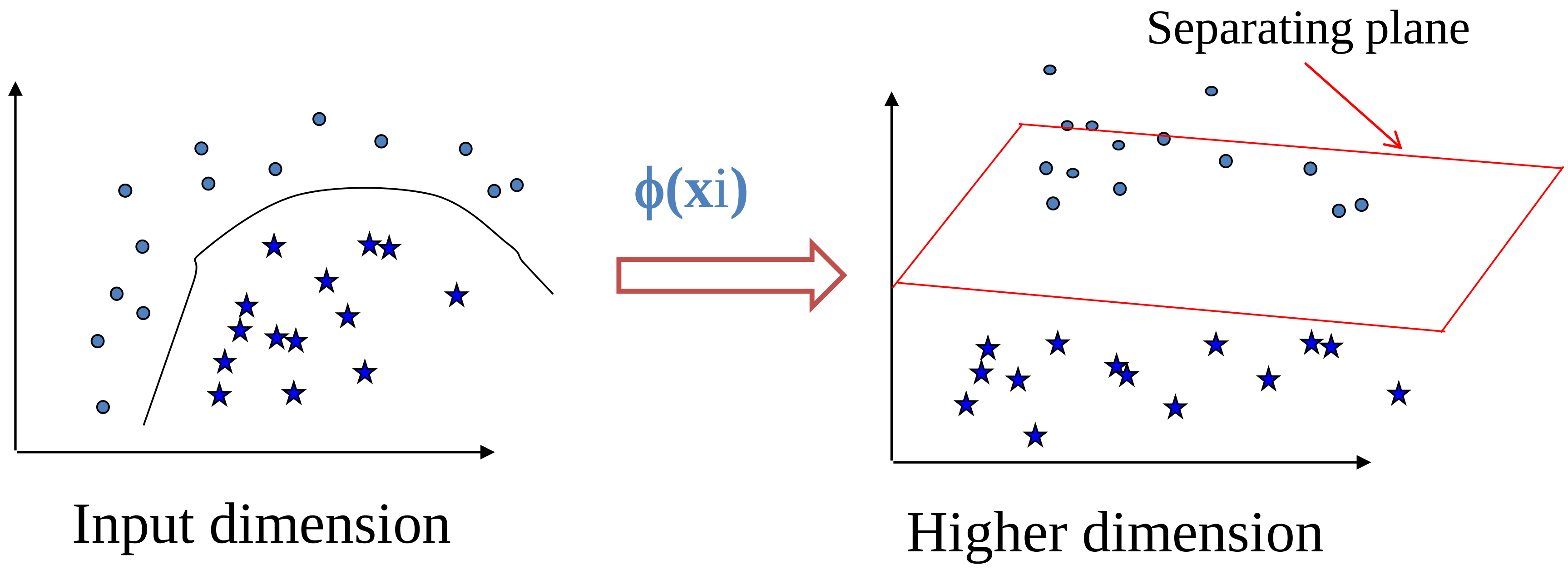


Transform the data to a two dimensional space as follows: (x, x^2)



Kernel functions

Map the data vectors **xi** from the *input space* into a higher dimensional *feature space* by some non-linear mapping **ϕ(xi)** (using kernel function k)



Primal and Dual forms of the optimization

Primal

$$\min_{w,b,\xi} \frac{\|w\|}{2} + C\sum_{i=1}^{n} \xi_i$$

$$s.t \quad y_i(w.\phi(x)+b) >= 1-\xi_i$$

$$\xi_i \geq 0 \quad i = 1,2,\ldots,n$$

Dual – Using Lagrangian techniques (alpha are the Lagrangian multipliers)

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$s.t. \quad 0 \leq \alpha_i \leq C$$

$$y_i^T \alpha = 0$$

$$where \quad Q_{ij} = y_i y_j \phi(x_i).\phi(x_j)$$

$$e = [1,\ldots..,1]^T, \quad i = 1,2,\ldots,n$$

This is a convex quadratic optimisation problem – Unique solution with global minimum

Data vectors with non-zero alpha are called support vectors

At optimum

Decision function

$$w = \sum_i \alpha_i y_i \phi(x_i)$$

$$sign(w^T \phi(x) + b)$$

$$= sign\left(\sum_{i=1} \alpha_i y_i \phi(x_i).\phi(x)^T + b \right)$$

Note: w is not required

$$= sign\left(\sum_i \alpha_i y_i k(x_i.x) + b \right)$$

Kernel trick

Kernel trick

Kernal trick

Replace the dot product with Kernel functions.

$$k = \phi(x_i).\phi(x_j)$$

Hence: only required to compute the dot products in the input space.

The transformed space dimension is non needed – may be infinity

Kernel functions

Radial basis function: sigma is called kernel width parameter

$$k = \exp\left(-\frac{\left\| x_i - x_j \right\|}{\sigma^2} \right)$$

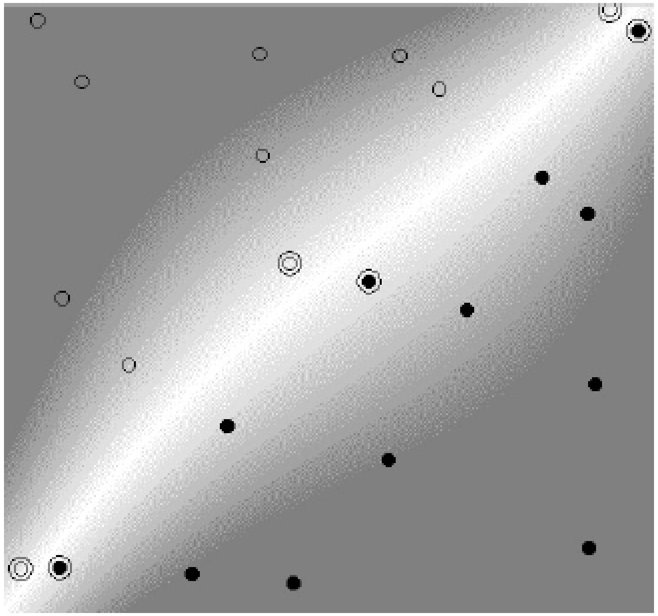
Polynimial function: d is called order (degree) of the polynomial and a and b are some constants.

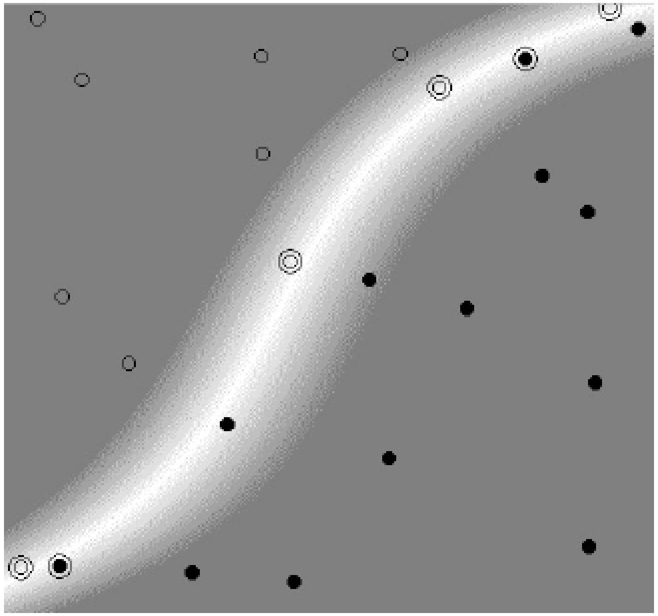
$$k = \left(x.x / a + b \right)^d$$

k(.) – needs to satisfy a condition called ‘Mercer Condition’ – see reference for more details

Example

Degree 3 polynomial

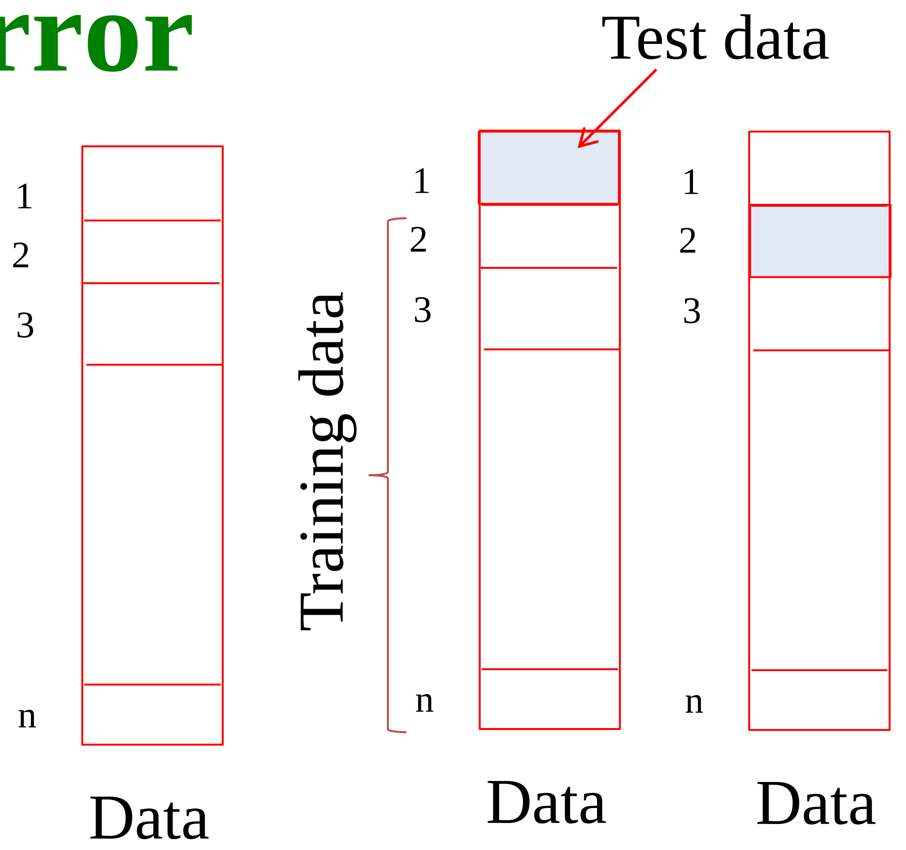




Classification error

n-fold Cross validation

- Divide the data into n chunks
- Take one of the chunk as test data and the remaining as training data
- Train the test the SVM. Compute the error comparing with the true labels
- Repeat the above now by selecting the second chunk as the test data and the remaining as the training data
- Finally get the average



Parameter selection

Select proper C and kernel parameters

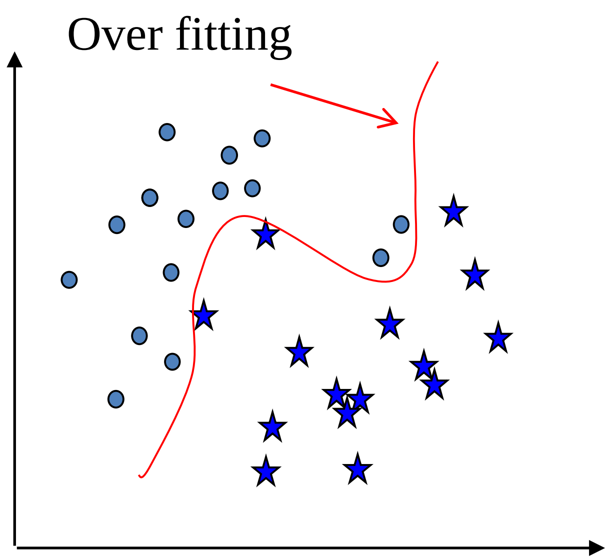
For large values of C, the penalty for misclassifying points is very high, so the decision boundary will perfectly separate the data if possible.

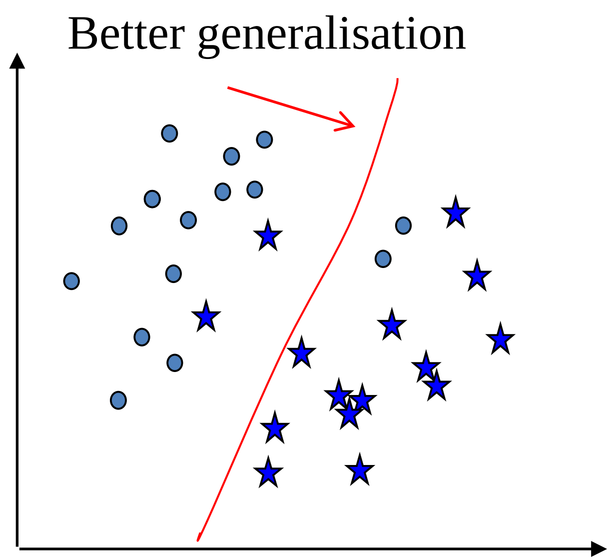
Lower C value - can maximize the margin between most of the points, while misclassifying a few points, because the penalty is so low.

Grid search

Avoid over-fitting :

Aim is to get a better classification accuracy on test data. Not to get a very strong fit to the training data - Generalisation





SVMs

Pros:

ONLY support vectors (only a subset of the data vectors) are required to completely define the trained SVM. – Sparseness of the SVM – Less memory space

Training is time consuming but testing is fast.

Also good for data sets that have larger dimensions but fewer number of samples

microarray data – dna, etc…

Cons:

Appropriately tuning the parameters

C, kernel parameters

Computational complexity is mainly for Quadratic optimization

SVM with R

Run SVMEx.R

https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf
https://www.rdocumentation.org/packages/e1071/versions/1.6-8/topics/svm

Unsupervised learning

Unsupervised Learning

Learn patterns from (unlabeled) data $(x_1, \ldots, x_n)$

Popular Approaches

- **clustering (similarity-based)**
- **Dimensionality reduction**
- **Anomaly detection**

What Are Anomalies/Outliers?

Outlier: A data object that deviates significantly from the normal objects as if it were generated by a different mechanism

Ex.: Unusual credit card purchase, sports: Michael Jordon, Wayne Gretzky, ...

Outliers are different from the noise data

Noise is random error or variance in a measured variable

Noise should be removed before outlier detection

Outliers are interesting: It violates the mechanism that generates the normal data

Outlier detection vs. *novelty detection*: early stage, outlier; but later merged into the model

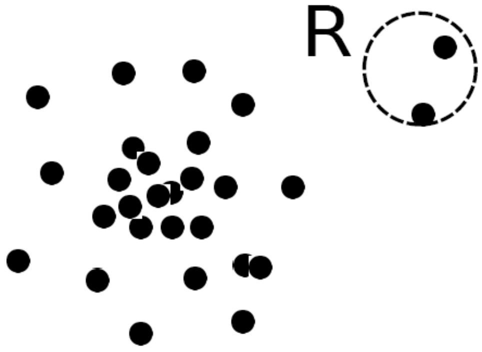
Applications:

Credit card fraud detection

Telecom fraud detection

Customer segmentation

Medical analysis



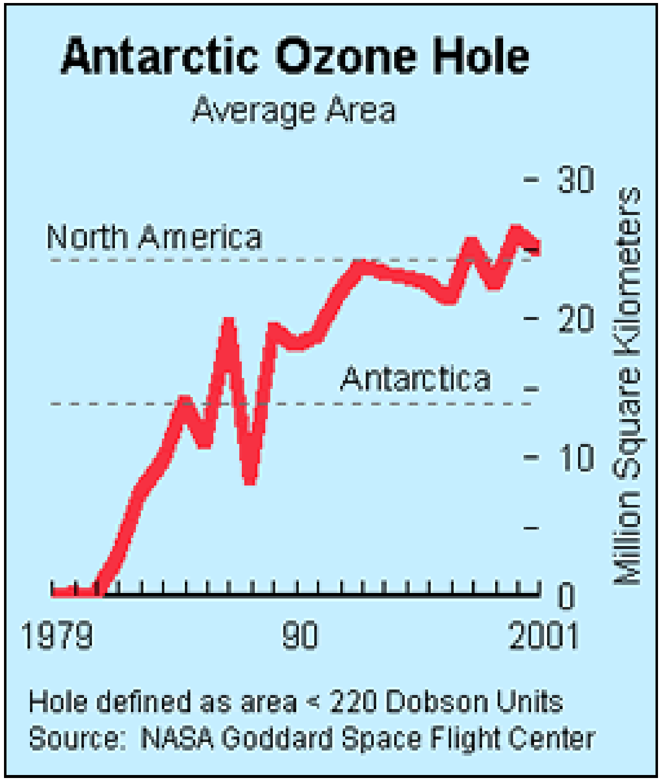
Importance of Anomaly Detection

Ozone Depletion History

- In 1985 three researchers (Farman, Gardinar and Shanklin) were puzzled by data gathered by the British Antarctic Survey showing that ozone levels for Antarctica had dropped 10% below normal levels

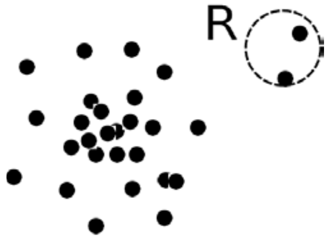
- Why did the Nimbus 7 satellite, which had instruments aboard for recording ozone levels, not record similarly low ozone concentrations?

- The ozone concentrations recorded by the satellite were so low they were being treated as outliers by a computer program and discarded!



Sources:
http://exploringdata.cqu.edu.au/ozone.html
http://www.epa.gov/ozone/science/hole/size.html

Types of Outliers (I)



Global Outlier

Three kinds: *global, contextual* and *collective* outliers
(definitions based on : Jiawei Han et. al's book…)

Global outlier (or point anomaly): O_g

- Object is O_g if it significantly deviates from the rest of the data set
- Ex. Intrusion detection in computer networks
- Issue: Find an appropriate measurement of deviation

Contextual outlier (or *conditional outlier*): O_c

- Object is O_c if it deviates significantly based on a selected context
- Ex. 80° F in Urban a outlier? (depending on summer or winter?)
- Attributes of data objects should be divided into two groups
  - Contextual attributes: defines the context, e.g., time & location
  - Behavioral attributes: characteristics of the object, used in outlier evaluation, e.g., temperature
- Can be viewed as a generalization of *local outliers*—whose density significantly deviates from its local area
- Issue: How to define or formulate meaningful context?

Slides: Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann Publishers, July 2011. ISBN 978-0123814791
http://hanj.cs.illinois.edu/bk3/bk3_slidesindex.htm

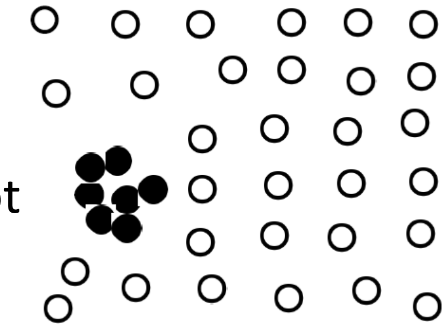
Types of Outliers (II)

Collective Outliers

A subset of data objects *collectively* deviate significantly from the whole data set, even if the individual data objects may not be outliers

Applications: E.g., *intrusion detection*:

When a number of computers keep sending denial-of-service packages to each other



Collective Outlier

- Detection of collective outliers
  - Consider not only behavior of individual objects, but also that of groups of objects
  - Need to have the background knowledge on the relationship among data objects, such as a distance or similarity measure on objects.
- A data set may have multiple types of outlier
- One object may belong to more than one type of outlier

Challenges of Outlier Detection

- Modeling normal objects and outliers properly
  - Hard to enumerate all possible normal behaviors in an application
  - The border between normal and outlier objects is often a gray area
- Application-specific outlier detection
  - Choice of distance measure among objects and the model of relationship among objects are often application-dependent
  - E.g., clinic data: a small deviation could be an outlier; while in marketing analysis, larger fluctuations
- Handling noise in outlier detection
  - Noise may distort the normal objects and blur the distinction between normal objects and outliers. It may help hide outliers and reduce the effectiveness of outlier detection
- Understandability
  - Understand why these are outliers: Justification of the detection
  - Specify the degree of an outlier: the unlikelihood of the object being generated by a normal mechanism

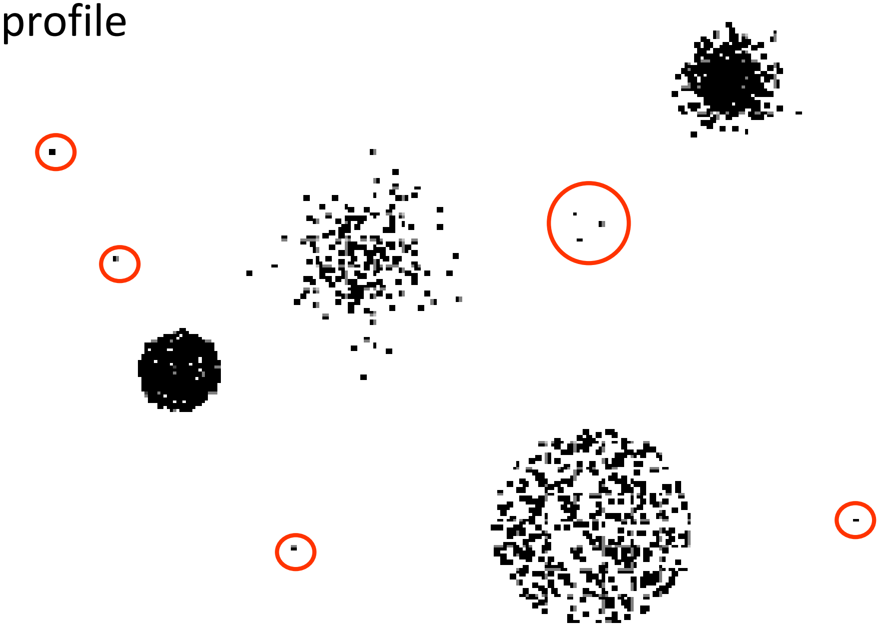
Anomaly Detection Schemes

General Steps

- Build a profile of the “normal” behavior
  - Profile can be patterns or summary statistics for the overall population
- Use the “normal” profile to detect anomalies
  - Anomalies are observations whose characteristics differ significantly from the normal profile

Types of anomaly detection schemes

- Graphical & Statistical-based
- Distance-based
- Model-based



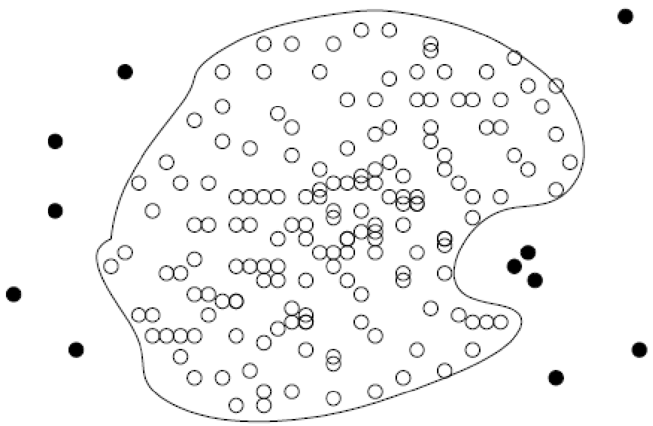
Classification-Based Method I: One-Class Model

Idea: Train a classification model that can distinguish "normal" data from outliers

A brute-force approach: Consider a training set that contains samples labeled as "normal" and others labeled as "outlier"

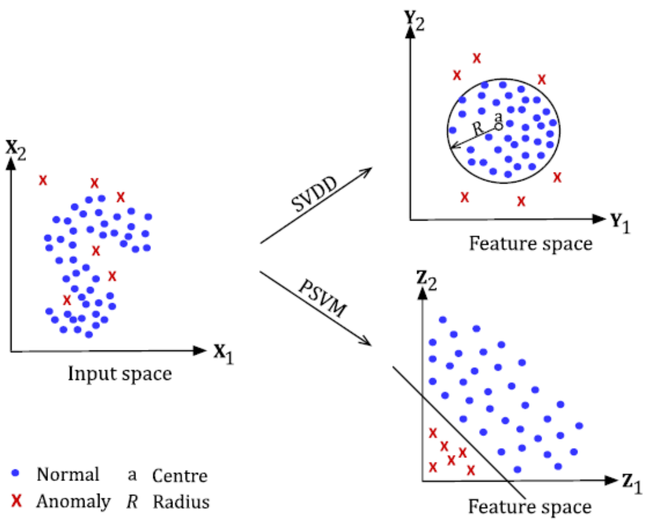
But, the training set is typically heavily biased: # of "normal" samples likely far exceeds # of outlier samples

Cannot detect unseen anomaly



- One-class model: A classifier is built to describe only the normal class.
  - Learn the decision boundary of the normal class using classification methods such as SVM
  - Any samples that do not belong to the normal class (not within the decision boundary) are declared as outliers
  - Adv: can detect new outliers that may not appear close to any outlier objects in the training set
  - Extension: Normal objects may belong to multiple classes

One class SVM



$$\min_{\mathbf{a},\, R,\, \boldsymbol{\xi}} \quad R^2 + \frac{1}{m\nu}\sum_{l}^{m} \xi_l$$

$$\text{s.t.} \quad \|\phi(\mathbf{x}_l) - \mathbf{a}\|^2 \le R^2 + \xi_l,$$

$$\forall l = 1, \ldots, m,\ \xi_l \ge 0.$$

$$\max_{\boldsymbol{\alpha}} \quad \sum_{l=1}^{m} \alpha_l (\mathbf{x}_l \cdot \mathbf{x}_l) - \sum_{l,t} \alpha_l \alpha_t (\mathbf{x}_l \cdot \mathbf{x}_t),$$

$$\forall 1 \le l, t \le m,$$

$$\text{s.t.} \quad 0 \le \alpha_l \le \frac{1}{m\nu}.$$

$$\min_{\mathbf{s}, \xi, \rho} \quad \frac{1}{2}\|\mathbf{s}\|^2 + \frac{1}{m\nu}\sum_{l=1}^{m} \xi_l - \rho$$

$$\text{s.t.} \quad (\mathbf{s} \cdot \mathbf{x}_l) \ge \rho - \xi_l,$$

$$\forall l = 1, \ldots, m, \xi_l \ge 0.$$

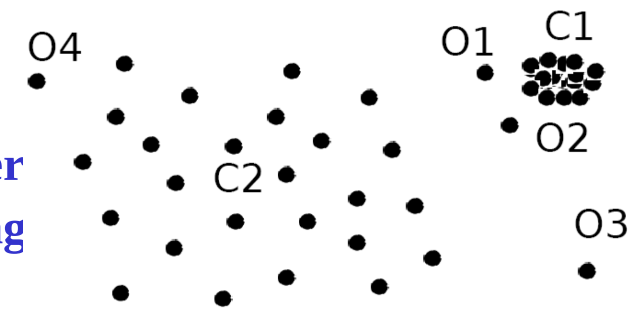
$$\min_{\alpha} \quad \frac{1}{2}\sum_{lt} k(\mathbf{x}_l, \mathbf{x}_t)$$

$$\text{s.t.} \quad 0 \le \alpha_l \le \frac{1}{m\nu}, \sum_{l} \alpha_l = 1.$$

- D.M. Tax,R.P.Duin,Supportvectordatadescription,Mach.Learn.54(2004) 45–66.
- B. Schölkopf,J.C.Platt,J.Shawe-Taylor,A.J.Smola,R.C.Williamson,Estimating the supportofahigh-dimensionaldistribution,NeuralComput.13(7)(2001) 1443–1471.

Density-Based Outlier Detection

Local outliers: Outliers comparing to their local neighborhoods, instead of the global data distribution
In Fig., o_1 and o2 are local outliers to C_1, o_3 is a global outlier but o_4 is not an outlier. However, proximity-based clustering cannot find o_1 and o_2 are outlier (e.g., comparing with O_4).



- Intuition (density-based outlier detection): The density around an outlier object is significantly different from the density around its neighbors
- Method: Use the relative density of an object against its neighbors as the indicator of the degree of the object being outliers
- *k-distance* of an object o, $dist_k(o)$: distance between o and its k-th NN
- *k-distance neighborhood* of o, $N_k(o) = \{o' | o' \text{ in } D, dist(o, o') \leq dist_k(o)\}$
  - $N_k(o)$ could be bigger than k since multiple objects may have identical distance to o

M.M. Breunig,H.P.Kriegel,R.T.Ng,J.Sander, LOF: identifyingdensity-based local outliers,SIGMODRec.29(2000)93–104.

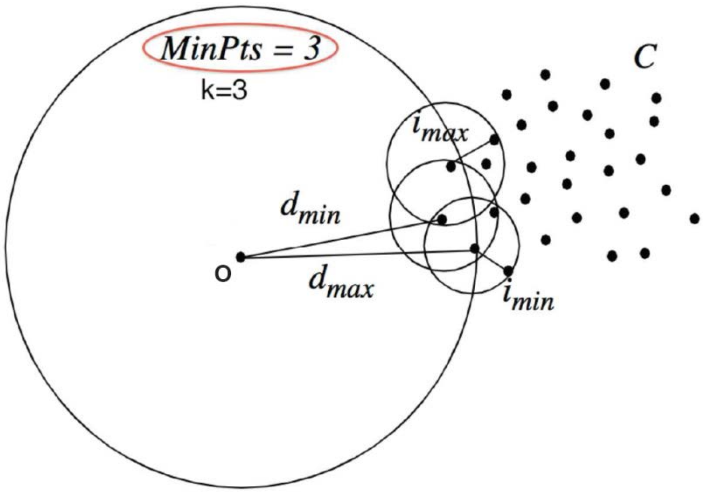
Local Outlier Factor: LOF

Reachability distance from *o'* to *o*:

$$reachdist_k(o \leftarrow o') = \max\{dist_k(o), dist(o, o')\}$$

where k is a user-specified parameter

Local reachability density of *o*:



$$lrd_k(o) = \frac{\|N_k(o)\|}{\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)}$$

- LOF (Local outlier factor) of an object o is the average of the ratio of local reachability of *o* and those of *o*'s k-nearest neighbors

$$LOF_k(o) = \frac{\sum_{o' \in N_k(o)} \frac{lrd_k(o')}{lrd_k(o)}}{\|N_k(o)\|} = \sum_{o' \in N_k(o)} lrd_k(o') \cdot \sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)$$

- The lower the local reachability density of o, and the higher the local reachability density of the kNN of o, the higher LOF
- This captures a local outlier whose local density is relatively low comparing to the local densities of its kNN

Slides: Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann Publishers, July 2011. ISBN 978-0123814791
http://hanj.cs.illinois.edu/bk3/bk3_slidesindex.htm

R examples

References/Reading

- I. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 2000
  - Chapter 8
- Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, Pearson Higher Ed, 2013.
  - Chapter 5
- Hastie et al. "The elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009).
- Some of the slides are developed after discussion with Dr Santu Rana.
- Some of the material are adapted from the slides of Statistical Learning MOOC by Hastie and Tibshirani available through http://www-bcf.usc.edu/~gareth/ISL/.
- Mehmed Kantardzic, Datamining: Concepts, Models, Methods and Algorithms, 2003 edition, IEEE Press
- CHRISTOPHER J.C. BURGES, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2, 121-167, 1998
- Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann Publishers, July 2011. ISBN 978-0123814791 http://hanj.cs.illinois.edu/bk3/bk3_slidesindex.htm