CS256 – Midterm Exam Study Guide

By: Zayd Hammoudeh

Chapter #04 – Classification: Basic Concepts, Decision Trees, and Model Evaluation

Classification	Training Set	Model	Test Set	Example Classification Techniques
Task of assigning objects to one of several predefined categories.	A collection of records. Each **record** contains a set of attributes one of which is the **class**.	A function from the value of record attributes to the class attribute.	A collection of records used to determine the accuracy of the classification model.	1. Neural Networks 2. Decision Tree 3. Rule Based Classifier 4. Memory Based Reasoning 5. Support Vector Machines 6. Naïve Bayes and Bayesian Belief Networks

Induction
Using a training set to generate a model.

Deduction
Process of applying a model to a training set.

Decision Tree Induction

- **Greedy Strategy**
- **Key Decision #1:** Attribute to expand next
- **Key Decision #2:** When to stop expanding

Hunt's Decision Tree Induction Algorithm:

- Let D_t be the set of training records that reach a node t.

1. If D_t contains records that **all belong to the same class y_t**, then t is a leaf node with class value y_t.
2. If D_t is an **empty set**, then t is a leaf node with default value y_d.
3. If D_t contains **records that belong to more than one class and there are no attributes left**, then t is a leaf node with default value is a leaf node with default value y_d.
4. If D_t contains **records that belong to more than one class**, then use an attribute test to split the data into smaller subsets. Recursively apply the same procedure above.

Attribute Types

- **Binary** – Attribute with exactly two possible values.
- **Nominal** – Two or more class values with no intrinsic order
- **Ordinal** – Two or more class values that can be ordered or ranked
- **Continuous** – Quantitative attribute that can be measured along a continuum.

Splitting Nominal and Ordinal Attributes

- **Binary** – Divides attribute values into two subsets. **This requires the additional step of finding optimal partitioning.**
- **Multi-way** – Use as many partitions as distinct values.

Splitting Based on Continuous Attributes

- **Discretization** – Form an ordinal categorical attribute.
  - **Static** – Discretize once at the beginning
  - **Dynamic** – Ranges can be found by equal interval bucketing, equal frequency bucketing, or clustering.
- **Binary Decision** (A < v or A >v) – Consider all possible splits and find the best cut.
  - **Binary Decision Procedure:** Go between each training set record value and calculate the GINI index if the splitting point was at that value. **Select the splitting point with the lowest $\text{GINI}_{\text{SPLIT}}$ value.**
    - **Computationally inefficient** $O(n)$ – where n is the number of records.

Homogeneity/Low Impurity – Extent to which nodes in the decision tree have the same class value/distribution.

Nodes with high levels of homogeneity (i.e. low levels of impurity) are preferred.

Impurity Measures

For all of these metrics, a lower score is generally preferable.

GINI Index

$$GINI(t) = 1 - \sum_{j=1}^{n_c} \left(p(j|t)\right)^2$$

- t – Node in the decision tree
- j – Class value
- n_c – Number of class values
- $p(j|t)$ – Probability (i.e. relative frequency) of class value j in node t

Minimum Value: 0 when:

$$\exists j(p(j|t) = 1)$$

Maximum Value: $1 - \frac{1}{n_c}$ when:

$$\forall j\left(p(j|t) = \frac{1}{n_c}\right)$$

$\text{GINI}_{\text{SPLIT}}$ (Weighted GINI Index)

$$GINI_{SPLIT} = \sum_{i=1}^{k} \frac{n_i}{n} \cdot GINI(i)$$

- i – Child node
- n – Number of records in parent node. Note:

$$n = \sum_{i=1}^{k} n_i$$

- n_i – Number of child nodes (i.e. attribute partitions)
- $GINI(i)$ – GINI index value of node i.

Minimum Value: 0 when:

$$\forall i(GINI(i) = 0)$$

Maximum Value: $1 - \frac{1}{n_c}$ when:

$$\forall i\left(GINI(i) = 1 - \frac{1}{n_c}\right)$$

Entropy

$$Entropy(t) = -\sum_{j=1}^{n_c} p(j|t) \cdot log_2\left(p(j|t)\right)$$

- t – Node in the decision tree
- j – Class value
- n_c – Number of class values
- $p(j|t)$ – Probability (i.e. relative frequency) of class value j in node t

Minimum Value: 0 when:

$$\exists j(p(j|t) = 1)$$

Maximum Value: $\log_2(n_c)$ when:

$$\forall j\left(p(j|t) = \frac{1}{n_c}\right)$$

Information Gain

$$GAIN_{SPLIT}(t) = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} \cdot Entropy(i)\right)$$

- p – Parent node in the decision tree
- i – Child node in the decision tree
- k – Number of child nodes
- n_i – Number of records in child node i
- n – Number of records in parent node p

$$n = \sum_{i=1}^{k} n_i$$

Key Note: A higher $GAIN_{SPLIT}$ is preferable unlike with the other metrics where a lower value was better.

Disadvantage of Information Gain: Tends to prefer splits that result in a large number of partitions, each being small but pure (i.e. overfitting)

Normalizing for Split Size

$$GainRATIO_{Split} = \frac{Gain_{SPLIT}(t)}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \cdot log_2\left(\frac{n_i}{n}\right)$$

$Split_{INFO}$ penalizes a large split by reducing the gain.

Classification Error

$$Error(t) = 1 - max_j(p(j|t))$$

- t – Node in the decision tree
- j – Class value
- $p(j|t)$ – Probability (i.e. relative frequency) of class value j in node t

Minimum Value: 0 when:

$$\exists j(p(j|t) = 1)$$

Maximum Value: $1 - \frac{1}{n_c}$ when:

$$\forall j\left(p(j|t) = \frac{1}{n_c}\right)$$

Stopping Criteria for Decision Tree Induction

Three Stopping Criteria for Decision Tree Induction

- **When all records in a node have the same class value**
- **When all records in a node have similar attribute values.**
- **Early Termination**

Underfitting – When a model is too simple, both training and test errors are large.

Overfitting – When a model becomes too complex (e.g. too large a tree), the test error begins to increase even though the training error decreases.

- **Result:** Training error is **NOT** representative for generalization error.

Causes of Overfitting

- **Noise**
- **Insufficient training records** (i.e. lack of representative samples)

Resubstitution Error

Error on the **training** set.

Single Leaf Node Error: $e(t)$
Total Resubstitution Error: $e(T)$

$$e(T) = \sum e(t)$$

Generalization Error

Error on the **testing** data.

Single Leaf Node Error: $e'(t)$
Total Generalization Error: $e'(T)$

$$e'(T) = \sum e'(t)$$

Generalization Error Estimation

Optimistic Estimation

Training error is equal to the testing error.

$$\sum e(t) = \sum e'(t)$$

Pessimistic Estimation

Assign a penalty term to ea.

$$e'(t) = e(t) + 0.5$$

Total Pessimistic Error

$$e'(T) = \sum(e(t)) + N \cdot 0.5$$

N – Number of leaf nodes.

Reduced Error Pruning

Use a validation set to estimate the generalization error.

Occam's Razor

Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.

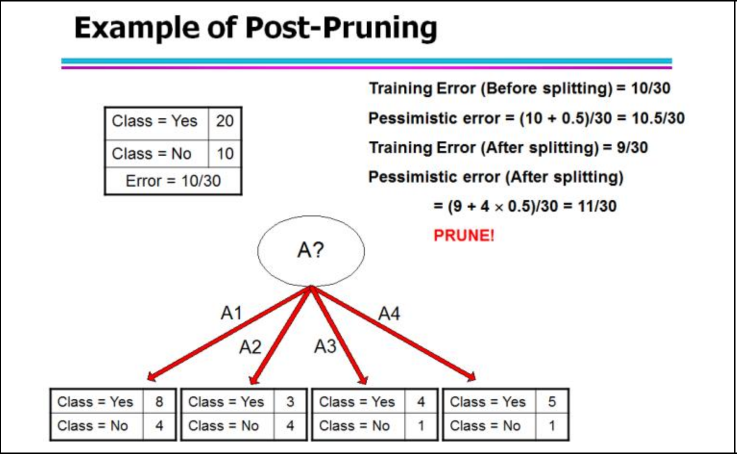
This is because more complex model has a greater chance of fitting accidentally by errors in the data.

Pre-pruning (Early Stopping Rule)

- **Stop the induction algorithm before it becomes a full tree.**
- **Typical Stopping Rules:**
  - All remaining records have the same class value
  - All attribute values are the same.
- **More restrictive conditions:**
  - Number of instances is below a user-specified threshold.
  - Expanding the current node does not improve impurity measures (e.g. GINI Index, Information Gain)
  - Class distribution of instances are independent of available features (using χ^2 test).

Post-pruning (Early Stopping Rule)

- **Grow the decision tree to its entirety.**
- Trim nodes in the tree in a **bottom-up fashion.**
- Only **trim nodes if by trimming the estimate of the generalization error improves.**
- New leaf node's **class label is determined from the majority class of instances in the merged node.**

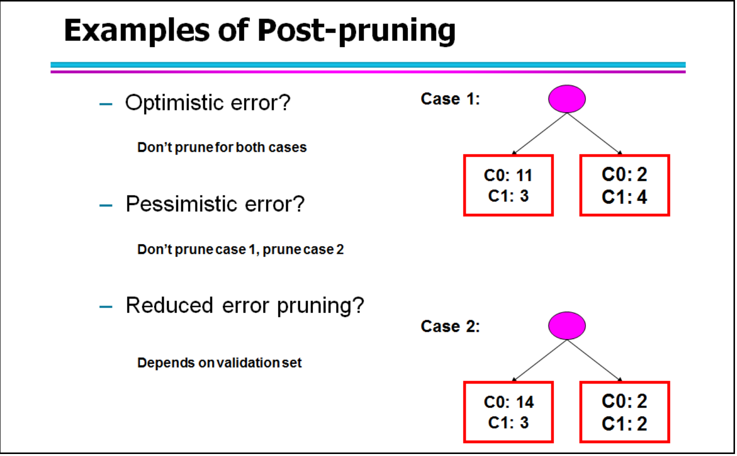


Example of Post-Pruning

Training Error (Before splitting) = 10/30
Pessimistic error = (10 + 0.5)/30 = 10.5/30
Training Error (After splitting) = 9/30
Pessimistic error (After splitting)
= (9 + 4 × 0.5)/30 = 11/30
PRUNE!

Class = Yes	20
Class = No	10
Error = 10/30	

Class = Yes	8	Class = Yes	3	Class = Yes	4	Class = Yes	5
Class = No	4	Class = No	4	Class = No	1	Class = No	1



Examples of Post-pruning

- Optimistic error?
  Don't prune for both cases
- Pessimistic error?
  Don't prune case 1, prune case 2
- Reduced error pruning?
  Depends on validation set

Case 1:
C0: 11
C1: 3
C0: 2
C1: 4
Case 2:
C0: 14
C1: 3
C0: 2
C1: 2

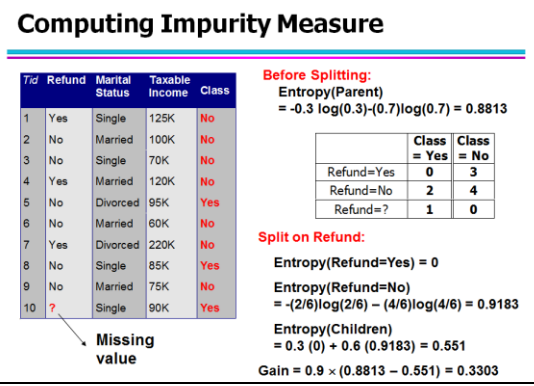
Handling Missing Attribute Values

Issues Associated with Missing Attribute Values

- **Affects how impurity measures are computed**
- **Affects how to distribute instances with missing value to child nodes.**
- **Affects how to test instance with missing value is classified.**

Computing Impurity Measure

- **Calculate entropies (i.e. information gain) with element with missing value EXCLUDED.**
- **Multiply by scalar of elements included over total number of elements** (in below example 9 elements included over 10 total elements hence 0.9):



Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	?	Single	90K	Yes

Missing value

Before Splitting:
Entropy(Parent)
= -0.3 log(0.3)-(0.7)log(0.7) = 0.8813

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

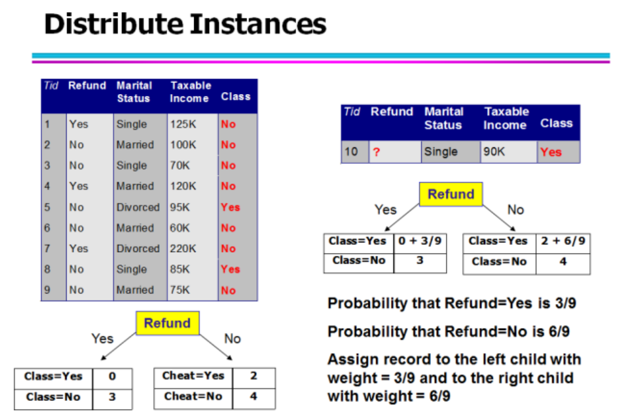
Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

Entropy(Children)
= 0.3 (0) + 0.6 (0.9183) = 0.551

Gain = 0.9 × (0.8813 – 0.551) = 0.3303

Distribute Instances

- **Split the missing record between the two child nodes**
- **Percentage of child node that goes to each child is portion to the relative frequency of that attribute value.**



Distribute Instances

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No

Refund
Yes
No

Class=Yes	0
Class=No	3

Cheat=Yes	2
Cheat=No	4

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes

Refund
Yes
No

Class=Yes	0 + 3/9
Class=No	3

Class=Yes	2 + 6/9
Class=No	4

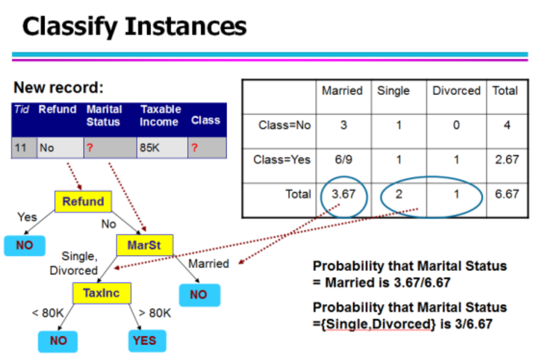
Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Classifying New/Unseen Records with Missing Data

- **Pick the most likely of child nodes and use continue down that portion of the tree.**



Classify Instances

New record:

Tid	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?

	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67

Probability that Marital Status = Married is 3.67/6.67

Probability that Marital Status ={Single,Divorced} is 3/6.67

Data Fragmentation – At each level of the tree, the number of instances gets smaller. At leaf nodes, the number of instances could be too small to be statistically significant.

Oblique Decision Tree – Test condition in a node may involve multiple attributes.

- **Advantage** – Most expressive decision tree
- **Disadvantage** – Finding optimal test condition is computationally expensive.

Optimal Tree Induction: NP Hard

Alternate Strategies

- **Bottom Up Tree Generation**
- **Bidirectional Tree Generation**
  - Inside-out Bidirectional
  - Outside-in Bidirectional

Expressiveness – Decision trees do not generalize well to certain types of functions. **Example:** a parity function which would require a complete tree.

Decision Boundary – Borderline between two neighboring regions of different classes. In non-oblique decision trees, this is parallel to access since it involves a single attribute at a time.

Tree Replication – In a decision tree, a subtree may appear in multiple branches. This leads to unnecessary memory usage.

Performance Evaluation

- **Focus on the predictive capability of a model.**

Confusion Matrix

		Predicted Class	
		Class = Yes	Class=No
Actual Class	Class = Yes	a	b
	Class=No	c	d

a – True Positive (TP)
b – False Negative (FN)
c – False Positive (FP)
d – True Negative (TN)

Accuracy

$$Accuracy = \frac{A + D}{A + B + C + D}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- **Accuracy only tells part of the story.**
  - **Example:** Two Class Problem
    - Number of Class 0 Examples: 9990
    - Number of Class 1 Examples: 10
    - If the model predicts everything as class 0, its accuracy is 99.9% but it cannot detect any class 1.

Cost Matrix

		Predicted Class	
		Class = Yes	Class=No
Actual Class	Class = Yes	C(Y|Y)	C(N|Y)
	Class=No	C(Y|N)	C(N|N)

- $C(j\,|\,k)$– Cost of predicting class "j" given the actual class is "k".

$$TotalCost = a \cdot C(Y|Y) + b \cdot C(N|Y) + c \cdot C(Y|N) + d \cdot C(N|N)$$

- Cost matrix can be a better performance evaluation as it accounts for different costs of depending on the type of error.

$$Precision\ (p) = \frac{a}{a + c}$$

- **Precision** – Accuracy of positive predictions. Biased towards C(Y|Y) & C(Y|N).
  - a – True positive.
  - c – False positive.

$$Recall\ (r) = \frac{a}{a + b}$$

- **Precision** – Accuracy of records with positive class value. Biased towards C(Y|Y) & C(N|Y).
  - a – True positive.
  - b – False negative.

$$F\_Measure\ (F) = \frac{2 \cdot r \cdot p}{r + p} = \frac{a + c}{2 \cdot a + b + c}$$

- **F-Measure** – Biased two all except C(N|N) (i.e. true negative)
  - r – Recall
  - p – Precision
  - c – False Positive

Recall and precision are two widely used metrics employed in applications where the successful detection of one of the classes is considered more significant than detection of other classes.

$$WA = \frac{w_1 \cdot a + w_4 \cdot d}{w_1 \cdot a + w_2 \cdot b + w_3 \cdot c + w_4 \cdot d}$$

n – Number of instances covered by rule
n_c – Number of positive instances covered by rule.

Proportionality of Cost and Accuracy

- Cost and accuracy are proportional if:

$C(Y|Y) = C(N|N)$ and $C(Y|N) = C(N|Y)$

Sample Size and Model Performance

- **Learning Curve** – Shows how model accuracy changes (and varies) with sample size.
- **Effects of Small Sample Size:**
  - **Bias in the estimate**
  - **Variance in the estimate.**

Methods for Model Comparison

Holdout – Reserve 2/3 of labelled examples for training and 1/3 for testing.

Disadvantages
- **Uses only a subset of the labelled examples** when training the model.
- Model **dependent on the composition of the training and test sets**.
- **Training and test sets are not independent** since they both come from the same original set. If one class value is over-represented or under-represented in either set, the results will be skewed.

Random Subsampling – Repeats the holdout method multiple times with replacement.

Disadvantages:
- Still **uses only a subset of the labelled examples** to build the model.
- **No control over how many times each record appears in the training and test sets**. If a particular record is always in the training set, it may skew the model.

Aggregated Accuracy of k Random Subsamplings

$$acc_{sub} = \frac{1}{k} \cdot \sum_{i=1}^{k} acc_i$$

- k – Number of iterations
- acc_i – Accuracy of the i^{th} iteration.

Cross Validation – Partition the labelled dataset into k disjoint subsets.
- **k-Fold** – Train on k-1 partitions and test on the remaining one.
- **Leave-One-Out** – The number of partitions equals the number of training samples.

Aggregated Accuracy of k-Fold Cross Validation

$$acc_{sub} = \frac{1}{k} \cdot \sum_{i=1}^{k} acc_i$$

- k – Number of iterations
- acc_i – Accuracy of the i^{th} iteration.

Disadvantages:
- **Computationally expensive** as process is repeated k times.
- Depending on size of partition (e.g. 1 for Leave-One-Out), **accuracy from iteration to iteration** can vary significantly.

Bootstrap
- Build a training data set. After selecting an element for training data set, place it back in the pool of possible selections so it may possibly be selected again.
- Any elements not selected for the training data set are placed in the verification data set.

Minimum Description Length

$$Cost(Model, Data) = Cost(Data|Model) + Cost(Model)$$

Chapter #05 – Additional Classification Techniques

Rule-Based Classifiers

Classifies records using a collection of "if…then…" rules.
Form of Rule:

$$(Condition) \rightarrow y$$

- **Condition** (**Antecedent**, **LHS**) – Conjunction of attributes.
- y (**Consequent**, **RHS**) – Class value.

Cover – A rule r covers an instance x if the attributes of x satisfy the condition (LHS) of the rule.

Coverage of a Rule – Fraction of records that satisfy the antecedent of a rule.

$$Coverage = \frac{n_R}{N}$$

Accuracy of a Rule – For records covered by a rule, it is the fraction of records that have the matching class value.

Mutually Exclusive Rule Set – Rules in the set are independent of each other such that **each record is covered by at most one rule**.

Exhaustive Rule Set – A set of rules that covers every possible combination of attribute values. Hence, **each record is covered by at least one rule**.

Decision Tree – Can be used to formed a mutually exclusive, exhaustive rule set.

Rules in a decision tree can be simplified.

Effects of rule simplification:

- **Problem #1:** Rules become non-mutually exclusive.
- **Solution:**
  - **Ordered Rule Set** – Rules ordered from highest to lowest priority. Records classified according to highest priority rule they satisfy.
  - **Unordered Rule Set** – Voting scheme
- **Problem #2:** Rules become non-exhaustive.
- **Solution:** Use a default class.

Rule Ordering Schemes

Rules Based Ordering – Individual rules are ranked based off their quality.

- **Advantage:** Ensures each record is classified by the "best rule" covering it.
- **Disadvantage:** Interpreting lower priority rules becomes more difficult as they are negations of higher priority rules.

Class-Based Ordering – Rules that belong to the same class appear together.

- **Advantage:** Simplifies rule ordering.
- **Disadvantage:** May allow a lower quality rule to have higher priority than a higher quality one.

Direct Method for Rule Building – Extract rules directly from the data.

- **Examples:** RIPPER, CN2, Holte's 1R

Indirect Method for Rule Building – Extract rules from other classification models (e.g. decision tree, neural network, etc.)

- **Examples:** C4.5rules

Sequential Covering Algorithm

1. **Start with an empty rule set.**
2. **Grow a rule using the "Learn-One-Rule" function.**
3. **Remove training records covered by the rule.**
4. **Repeat steps #2 and #3 until stopping criterion is met.**

Aspects of Sequential Covering

1. **Rule Growing**
2. **Instance Elimination**
3. **Rule Evaluation**
4. **Stopping Criterion**
5. **Rule Pruning**

Rule Growing Strategies

1. **General to Specific**
   a. Example: Ripper
2. **Specific to General**

CN2 Algorithm

1. Start from an empty rule
2. Add conjuncts that minimize the entropy measure.
3. Determine the rule consequent by taking majority class of covered instances.

Instance Elimination

- **Reason for Eliminating Instances** – Otherwise next rule is identical to previous rule.
- **Reason for Removing Positive Instances** – To ensure future rules are different.
- **Reason for Removing Negative Instances** – Prevent underestimating accuracy of the rule.

Stopping Criterion

Compute the information gain with the rule. If the gain is insignificant, discard the rule.

Rule Pruning

- **Similar to post-pruning of decision trees.**
- Uses reduced error pruning.
  - **Remove one of the conjuncts of the rule.**
  - **Compare error rate on validation set before and after pruning.**
  - **If error improves, remove the conjunct.**

Rule Simplification – Used to reduce the likelihood of overfitting.

Rule Evaluation Metrics

$$Accuracy = \frac{n_c}{n}$$

n – Number of instances covered by rule
n_c – Number of positive instances covered by rule.

$$Laplace = \frac{n_c + 1}{n + k}$$

n – Number of instances covered by rule
n_c – Number of positive instances covered by rule.
k – Number of classes
Used to ensure greater coverage for a rule.

$$m\_estimate = \frac{n_c + p \cdot k}{n + k}$$

n – Number of instances covered by rule
n_c – Number of positive instances covered by rule.
k – Number of classes
p – Prior probability of positive class.

FOIL Information Gain

$$Gain(R0, R1) = p_1 \cdot \left(\log_2\left(\frac{p_1}{p_1 + n_1}\right) + \log_2\left(\frac{p_0}{p_0 + n_0}\right)\right)$$

$R0$ – Initial Rule
$R1$ – Modified version of $R0$ with added conjunct
t – Number of positive instances covered by both $R0$ and $R1$
p_0 – Positive instances covered by $R0$
n_0 – Negative instances covered by $R0$
p_1 – Positive instances covered by $R1$
n_1 – Negative instances covered by $R1$

RIPPER Algorithm

1. For two classes, define one class as positive class and other as negative class.
   a. In two class problem, **negative class is the default class**.
2. **In multi-class problem, create list of classes ordered by increasing prevalence**.
   a. Select smallest as first as positive class and rest are negative class.
   b. **Learn rules for the smallest class first.**
   c. **Repeat with next smallest class**.

RIPPER Algorithm – Growing a Rule

1. Start from an empty rule set.
2. Add conjuncts as long as they improve **FOIL Information Gain** (i.e. **General-to-Specific**).
3. Stop adding conjuncts when the rule starts covering negative examples.
4. Begin pruning the rule immediately (i.e. before generating new rules) using Reduced Error Pruning.
5. Delete conjuncts to maximize v as defined by:

$$v = \frac{p - n}{p + n}$$

p – Number of positive instances covered by the rule.
n – Number of negative instances covered by the rule.

RIPPER Algorithm – Building the Rule Set

1. **Use Sequential Covering**
   a. Find the rule that best covers the current set of positive examples.
   b. Eliminate both positive and negative examples covered by the rule.
   c. **Uses ordered rule set with class based ordering.**
2. Each time a rule is added to the rule set, compute the new description length. **Example Stopping Conditions:**
   a. Stopping growing the rule set if the new **rule increases the description length of the rule set by more than d** (e.g. 64) **bits**.
   b. Stop if the error rate of the rule **on the validation set** is more than 50%.

C4.5rules – Indirect Method

1. Start from an **unpruned** decision tree.
2. For each rule $r: A \rightarrow y$,
   a. Consider an alternative rule $r': A' \rightarrow y$ where A' is obtained by **removing one of the conjuncts** of A
   b. **Keep the rule with the lowest pessimistic error rate** (assuming it is less than the original).
   c. Repeat until it is no longer possible to improve the generalization error.
3. Use class-based ordering for the rule set (i.e. group by the rule consequent).
4. Compute the description length of each class and order the rules by increasing description length.

$$DescriptionLength = L(error) + g \cdot L(model)$$

- $L(error)$ – Number of bits required to encode misclassified examples.
- $L(model)$ – Number of bits required to encode the model.
- g – Tuning parameter whose default is 0.5 and takes into account the presence of redundant attributes in the rule set.

Nearest Neighbor Classifiers

Instance-Based Classifier – Stores all training records and uses the training records directly to predict the class label of unseen records.

Rote-Learner – Memorizes the entire training set and performs classification only if attributes of a record match one the training examples exactly.

Nearest Neighbor – Uses k "closest" training records (i.e. nearest neighbors) for performing classification.

Nearest Neighbor Classifier Requirements

1. **Set of stored labelled records.**
2. **Distance metric to compute distance between records.**
3. **Value of k, the number of nearest neighbors to retrieve.**

Classifying an Unseen Record

1. Compute the distance to all other training records.
2. Identify the k nearest neighbors.
3. Use class labels of nearest neighbors to determine the class label of unknown records

Voronoi Diagram – Used to depict the decision boundaries for a Nearest Neighbor classifier.

Euclidean Distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

Manhattan Distance

$$d(p,q) = \sum_i |p_i - q_i|$$

Determining the Class from the Nearest Neighbor List

Option #1 – Take the majority vote among the k-Nearest Neighbors.
Option #2 – Weight the vote according the distance using the weight factor:

$$WeightFactor = \frac{1}{d^2}$$

Effect of the Value of k

- **k is too Small** – Underfitting and the classifier becomes sensitive to noise points.
- **k is too Large** – Overfitting and the neighborhood make include points from other classes.

Attribute Value Scaling Issues

- Attributes may have to be scaled to normalize for different attribute ranges and values.
- This is done to prevent one of the attributes dominating the distance measure.

PEBLS

PEBLS

- Nearest neighbor algorithm that works with both continuous and nominal features.
- Each record is assigned a weight factor.
- Number of nearest neighbors, $k = 1$

Weighted Euclidean Distance

$$d(p,q) = \sqrt{\sum_i w_i \cdot (p_i - q_i)^2}$$

w_i – Weight of parameter i

Distance Between Nominal Attributes

Value Difference Metric

$$d(v_1, v_2) = \sum_i \left| \frac{n_{1,i}}{n_1} - \frac{n_{2,i}}{n_2} \right|$$

v_1 – Attribute value 1
v_2 – Attribute value 2
i – The i^{th} class value.
$n_{1,i}$ – Number of records with attribute value 1 and class value i
$n_{2,i}$ – Number of records with attribute value 2 and class value i
n_1 – Total number of records with attribute value 1
n_2 – Total number of records with attribute value 2

Similarity Function Used in PEBLS

$$d(X,Y) = w_X \cdot w_Y \sum_{i=1}^{k} d(X_i, Y_i)^2$$

X & Y – Two records
w_X – Distance weighting factor for record X
w_Y – Distance weighting factor for record Y. If Y is an unseen record, then $w_Y = 1$
$d(X_i, Y_i)$ – Distance between records X and Y in the i^{th} dimension.

$$w_X = \frac{Number\ of\ Times\ X\ is\ Used\ in\ Prediction}{Number\ of\ Times\ X\ Predicts\ Correctly}$$

If $w_X \cong 1$, then X makes an accurate prediction most of the time. If $w_X > 1$, then X does not make reliable predictions.

Bayesian Classifiers

Condition Probability Review

$P(C|A)$ – Probability of C given A.

$$P(C|A) = \frac{P(C \cap A)}{P(A)}$$

$$P(A|C) = \frac{P(C \cap A)}{P(C)}$$

Bayes Theorem

$$P(A|C) = \frac{P(C|A) \cdot P(A)}{P(C)}$$

Bayes Classifier – A probabilistic framework for solving classification problems.

Bayes Theorem Example

$$P(SN|M) = 0.5$$
$$P(M) = \frac{1}{50000}$$
$$P(SN) = \frac{1}{20}$$

Hence:

$$P(M|SN) = \frac{P(SN|M) \cdot P(M)}{P(SN)} = \frac{0.5 \cdot \frac{1}{50000}}{\frac{1}{20}}$$
$$P(M|SN) = 0.0002$$

Requirement of Naïve Bayesian Classifiers

- Consider **each attribute** $(A_1, A_2, \ldots, A_n)$ **as independent random variables**.
- Consider the **class** (C) **label as a random variable**.

Goal is to find:
$$P(C|A_1, A_2, \ldots, A_n)$$

Multi-Attribute Bayes Theorem

$$P(C|A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n|C) \cdot P(C)}{P(A_1, A_2, \ldots, A_n)}$$

Classification Approach: Select the value of C that maximums the above equation.

Naïve Bayesian Simplification

$$P(A_1, A_2, \ldots, A_n|C) = P(A_1|C) \cdot P(A_2|C) \cdot \ldots \cdot P(A_n|C)$$

This equations assumes independence among each A_i attribute when the class is given.

Estimating the Class Probability

$$P(C) = \frac{N_C}{N}$$

C – Class value
N_C – Number of training records with class value C
N – Total number of training records.

Estimating the Attribute-Class Probability

$$P(A_i|C) = \frac{|A_{i,C}|}{N_C}$$

- C – Class value
- N_C – Number of training records with class value C
- $|A_{i,C}|$ – Number of training records with class value C and attribute value A_i.

Handling Continuous Variables

Option #1: **Discretize the continuous range into bins**. This makes a series of ordinal attribute values. Conditional probability is estimated by the number of records that fall in each bin.

Simplest Approach: Use a two-way (i.e. binary) split.

Standard Deviation (σ)

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}}$$

Option #2: Assume attribute follows a normal distribution and use that mean (μ) and standard deviation (σ) to estimate the conditional probability.

$$P(A_i = a_i | C = c_j) = \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} \cdot e^{\frac{-1*(a_i - \mu_{i,j})^2}{2\sigma_{i,j}^2}}$$

a_i – Continuous value for attribute
c_j – Class value
$\mu_{i,j}$ – Mean for value for records with attribute A_i and class value c_j
$\sigma_{i,j}$ – Standard deviation for value for records with attribute A_i and class value c_j

Handling Zero Value Conditional Probabilities – Further Conditional Probability Estimation

Original/Standard

$$P(A_i|C) = \frac{|A_{i,C}|}{N_C}$$

$|A_{i,C}|$ – Number of records with attribute value A_i and class value C
N_C – Number of records with class value C

Laplace

$$P(A_i|C) = \frac{|A_{i,C}| + 1}{N_C + k}$$

$|A_{i,C}|$ – Number of records with attribute value A_i and class value C
N_C – Number of records with class value C
k – Number of classes

m-Estimate

$$P(A_i|C) = \frac{|A_{i,C}| + mp}{N_C + m}$$

$\lvert A_{i,C}\rvert$	Number of records with attribute value A_i and class value C
N_C	Number of records with class value C
p	User specified "prior probability." Most important when $\lvert A_{i,C}\rvert = 0$. Between 0 and 1.
m	Equivalent sample size. Used balance between p and $\frac{\lvert A_{i,C}\rvert}{N_C}$

Support Vector Machine (SVM)

Goal: Find a linear hyperplane (i.e. **decision boundary**) that has maximum separation (i.e. **margin**) between the two classes.

Reason for Maximum Margin: Decision boundaries with large margins tend to have better (i.e. lower) generalization error.

$$Margin = \frac{2}{||\vec{w}||^2}$$

$||\vec{w}||$ – Distance between the decision boundary and a plane running parallel to the decision boundary that intersects the nearest point to the boundary This will be maxim

Non-Linearly Separable Datasets

Slack Variables

- Determining decision boundary in SVM is an optimization problem.
- **Slack Variable** (ξ_i) – Used in the constraint equation to allow for nonlinearly separable decision boundaries.

Higher Order Remapping

Problem: Not all classification problems will be linearly separable.

Solution: Remap the data into a higher dimensional space (e.g. combine multiple variables at higher order).

Ensemble Method

Ensemble Method: Construct a set of classifiers from the training data. Predict class label of unseen records by aggregating predictions made by multiple classifiers.

Example: For a two class problem, assume there are n **independent** classifiers each with an error rate ϵ. If our ensemble classifiers uses the voting method for determining the class, then the error rate is:

$$\epsilon' = \sum_{i=\frac{n}{2}+1}^{n} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i}$$

If $n = 25$ and $\epsilon = 0.35$, then $\epsilon' = 0.06$

Bagging

- Create a dataset by repeatedly picking samples from the training dataset via a uniform distribution.
    - Since this technique uses **replacement**, **some training records may appear multiple times in the same dataset**.
- Probability a record appears at least once in the dataset is:

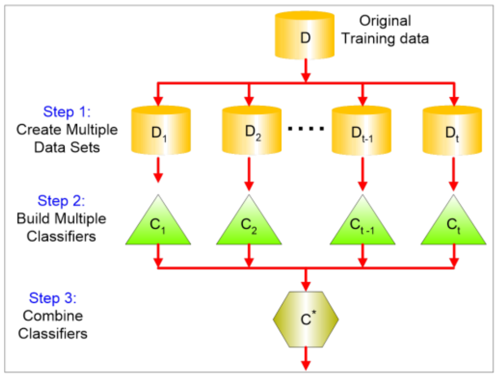
$$\left(1-\frac{1}{n}\right)^k$$

n – Number of training records.
k – Number of elements in the training data set.

Often $n = k$

This process is repeated a series of times generate a new model for each data set.

Illustration of the Ensemble Method



D
Original Training data
Step 1: Create Multiple Data Sets
D1
D2
Dt-1
Dt
Step 2: Build Multiple Classifiers
C1
C2
Ct-1
Ct
Step 3: Combine Classifiers
C*

AdaBoost

Boosting

- Variant of the Bagging Algorithm whereby **records that are wrongly classified have a higher probability of being selected in the next round** of training dataset selection.
- Records that are correctly classified are less likely to be selected in the next round of training dataset selection.

Review the AdaBoost Example from the Lecture Slides to Check Understanding of the Equations and their Usage

- Consists of a set of base classifiers: $C_1, C_2, \ldots, C_T$
- The error rate (ϵ_i) of a classifier (C_i) is defined as:

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j \cdot I\left(C_i(x_j) \neq y_j\right)\right]$$

Where:

$$I(p) = \begin{cases} 1, & p \text{ is true} \\ 0, & p \text{ is false} \end{cases}$$

N – Number of training examples
w_j – Weight assigned to record j in the current boosting round.

This equation is the essentially the sum of the weights of all incorrectly classified records divided by the number of records.

- **If any round has an error rate higher than 50%, all record weights are reset to $\frac{1}{N}$**

Importance of Classifier C_i

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$$

Weight Update

$$w_{j+1} = \begin{cases} \frac{w_j}{Z} \cdot e^{-\alpha_j}, & C_i(x_j) \neq y_j \\ \frac{w_j}{Z} \cdot e^{\alpha_j}, & C_i(x_j) = y_j \end{cases}$$

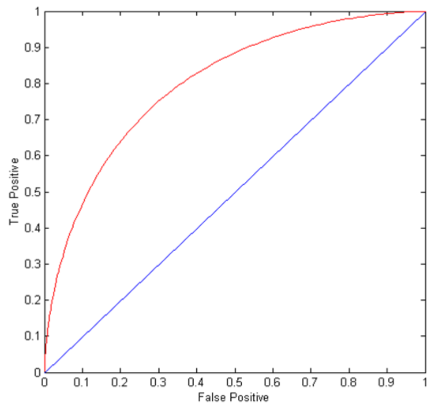
w_j – Record weight in round j
w_{j+1} – Record weight in subsequent round (i.e. $j+1$)
Z – Weight scaling factor.
α_j – Importance weight of classifier j

Classification: Simplifies to picking the class value with the highest weighted score.

$$C^* = argmax_y \sum_{j=1}^{T} \alpha_j \cdot \delta\left(C_j(x) = y\right)$$

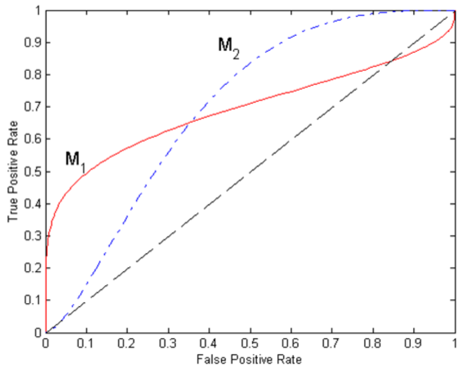
Receiver Operating Characteristics

ROC Curve



- Used to **illustrate the performance of a binary classifier**.
- **ROC Curve is two dimensional**
    - **X-Axis** – False Positive Rate (FPR)
    - **Y-Axis** – True Positive Rate (TPR)
- **Critical Points Along an ROC Curve**
    - **(TPR=0, FPR=0)** – Always predict a **negative** class.
    - **(TPR=1, FPR=1)** – Always predict a **positive** class.
    - **(TPR=1, FPR=0)** – Ideal case. Totally accurate model.
- **Area Under the Curve**
    - **1** – Ideal case. Model is always correct.
    - **0.5** – Random guessing.
    - **0** – Worst case. Model is always wrong.

- In the ROC curve, the **blue diagonal line shows the performance of random guessing**.
    - If the ROC curve is **above** the blue line, the performance is **better** than random guessing.
    - If the ROC curve is **below** the blue line, the performance is **worse** than random guessing.
- Figure below shows the ROC curve of two models (e.g. M_1 and M_2). Neither out performs the other. However, you **can combine the two models to get the best of both models about some pivot point**. (**M_1 better for small false positive rates and M_2 better for high false positive rates**).



Key Terms:

$$True\ Positive\ Rate = TPR = \frac{TP}{TP+FN}$$

$$False\ Positive\ Rate = FPR = \frac{FP}{FP+TN}$$

Other Terms:

$$True\ Negative\ Rate = TNR = \frac{TN}{FP+TN}$$

$$False\ Negative\ Rate = FNR = \frac{FN}{TP+FN}$$

Chapter #08 – Cluster Analysis: Basic Concepts and Algorithms

Intercluster distance – Distance between objects in two different clusters.

Intracluster distance – Distance between objects within the same cluster.

Cluster Analysis – Process of finding objects such that objects in a group will be similar to one another and different from objects in other groups.

Partitional Clustering – A division of data objects into non-overlapping subsets (i.e. clusters) such that each data object is in **exactly one subset** (i.e. cluster). (**Unnested**)

Hierarchical Clustering – A set of **nested** clusters organized as a hierarchical tree.

Distinctions between Sets of Clusters

Exclusive versus Non-exclusive Clustering – In non-exclusive clustering, objects may belong to multiple clusters. This can be used to represent multiple classes or border points.

Fuzzy versus Non-fuzzy – In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1. It resembles probabilistic clustering.

Partial versus Complete – Only a portion of the data is clustered.

Heterogeneous versus Homogeneous – Clusters of widely different sizes, shapes, and densities. If the data is non-homogeneous, then the clusters will not be homogeneous.

Different Types of Clusters

Well-Separated Clusters – A set of points such that any point in a cluster is closer to all other points in the cluster than it is to any point not in the cluster.

Center-Based Clusters – A set of objects such that an object in a cluster is **closer to the center of its cluster that to the center of any other cluster**.

Centroid – Average of all the points in the cluster.
Medoid – Most representative point in the cluster.

Contiguous Cluster – A cluster is set of points such that each point in the cluster is closer to one of the other points in the cluster than to any point not in the cluster.

Density-Based Cluster – A cluster is a dense region of points which is separated from other high density regions (i.e. clusters) by regions of low density.

Used when the clusters are irregular or intertwined and when noise and outliers are present.

Shared Property Clusters or Conceptual Clusters – Clusters that share some common property or represent a particular concept.

Objective Function

- Clusters are defined by an objective function, and **they either maximize or minimize the function**.
- Enumerating all possible cluster assignments and determining the best one is **NP-Hard**.
- Objective functions can have local and global objectives.
  - **Hierarchical Clustering Algorithms** – Tend to have local objectives.
  - **Partitional Clustering Algorithms** – Tend to have local objectives.
- Parameters for the objective function come from the object data.

K-Means Clustering

- **Partitional clustering** approach.
- Each cluster is associated with a centroid.
- **K** – User specified number of clusters.
- Each point is assigned to the cluster with the closest centroid (i.e. **basic algorithm is exhaustive**)

K-Means Algorithm

```
Select K points (e.g. randomly) to serve as initial centroids.
repeat
    Form K clusters by assigning all points to their closest centroid
    Recompute all K centroids
until Centroids do not change
```

Runtime: $O(n \cdot K \cdot d \cdot I)$

- n – Number of points/objects in the dataset
- K – Number of clusters
- d – Number of attributes
- I – Number of iterations of the loop.

K-Means Details

- **Initial Centers** – Chosen randomly.
- **Closeness/Similarity** – Different measures (e.g. Euclidean distance, cosine similarity, correlation) can be used.
- Most K-Means **convergence** happens in the first few cycles.

Sum of Squared Error

$$SSE = \sum_{i=1}^{k}\left(\sum_{x\in C_i}\left(dist(m_i, x)\right)^2\right)$$

- k – Number of clusters.
- C_i – One of the k clusters.
- x – Point in cluster C_i
- m_i – Mean (i.e. centroid) of cluster C_i

A larger SSE includes worse clustering.

Increasing K usually decreases SSE.

Solutions to Initial Centroids Problem

- Multiple initial runs but probability is still low.
- Sample and use hierarchical clustering to determine initial centroids.
- Select more than K initial values and then select among these the initial centroid
  - Example: Select the most widely separated.
- Post Processing
- Bisecting K-Means.

Types of Clusters where K-Means is Not Naturally Well Suited

- Different Size Clusters
- Different Density Clusters
- Non-globular (e.g. round) shaped clusters.

Possible Solution: Use a higher value of K than is expected and merge clusters via post-processing.

K-Means and Empty Clusters

- Basic K-Means can yield empty clusters.
- **Solutions:**
  - Choose a point that contributes most to SSE as the replacement centroid.
  - Choose a point from the highest SSE cluster as replacement centroid.
  - Repeat above steps if there are multiple empty clusters.

Updating K-Means Incrementally

- In Basic K-Means, centers are updated as points are assigned.
- In incremental K-Means, centers are updated after each assignment.
  - Each assignment updates zero or two centroids.
  - **More expensive**
  - **Introduces an order dependency.**
  - **Never get an empty cluster**
  - Can use weights to change the impact.

K-Means Preprocessing

- Eliminate Outliers
- Normalize the data (e.g. distance length)

K-Means Post-processing

- Eliminate small clusters that may be due to outliers.
- Split loose clusters (i.e. those with relatively high SEE)
- Merge tight clusters (i.e. clusters with relatively low SSE)
- **These steps can also be done with the algorithm is running.**

Bisecting K-Means

```
1: Initialize the list of clusters to contain the cluster containing all points.
2: repeat
3:   Select a cluster from the list of clusters
4:   for $i = 1$ to $number\_of\_iterations$ do
5:     Bisect the selected cluster using basic K-means
6:   end for
7:   Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: until Until the list of clusters contains $K$ clusters
```

Comparison of Classification Algorithms

Decision Tree Algorithm **Advantages** • **Inexpensive to construct** • **Extremely fast at classifying unknown records.** • **Easy to interpret for small sized trees.** • **Accuracy is comparable to other classification techniques for many simple datasets.** (Since everything comes right from the data) **Disadvantages** • **May not generalize well for certain types of functions** (e.g. Parity function requires a complete tree) • **May be insufficient for modelling continuous variables** that do not allow oblique nodes.	**Rule Based Classifiers** **Advantages** • **As highly expressive as decision trees** ○ A decision tree can be expressed via rules based classifier). ○ Allows for more complex models than a decision tree by allowing multiple rules to trigger on a single rule. • **Easy to interpret.** • **Easy to generate.** • **Can classify new records quickly.** • **Performance comparable to decision trees** • **Well suited for handling data sets with imbalanced class distributions.**	**Rule Based Classifiers** **Advantages** • **Lazy Learner** – Does not require the building of a complex model. • **Can create complex decision boundaries** unlike rule-based and decision trees which generally create rectilinear boundaries. **Disadvantages** • **Each unseen records are computationally expensive** (since must be compared to all training records). • **Susceptible to wrong prediction without appropriate proximity measure and preprocessing is done**. • Uses local data to make classification decisions so potentially **susceptible to noise**.

Bayesian Classifier **Advantages** • **Robust to isolated noise points**. • **Handles missing values by ignoring them in the probability estimate calculations**. • **Robust to irrelevant attributes.** **Disadvantages** • **Independence assumption may not hold for some attributes** (in such cases, must use a technique known as Bayesian Belief Networks).		