

CS256 – Midterm Exam Study Guide

By: Zayd Hammoudeh

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

Classification Task of assigning objects to one of several predefined categories.	Training Set A collection of records. Each record contains a set of attributes one of which is the class .	Model A function from the value of record attributes to the class attribute.	Test Set A collection of records used to determine the accuracy of the classification model.	Example Classification Techniques <ol style="list-style-type: none"> 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 <ul style="list-style-type: none"> • Greedy Strategy • Key Decision #1: Attribute to expand next • Key Decision #2: When to stop expanding 	Hunt's Decision Tree Induction Algorithm: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • Discretization – Form an ordinal categorical attribute. <ul style="list-style-type: none"> ◦ 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. <ul style="list-style-type: none"> ◦ 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 $GINI_{SPLIT}$ value. <ul style="list-style-type: none"> ▪ 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 $GINI(t) = 1 - \sum_{j=1}^{n_c} (p(j t))^2$ <ul style="list-style-type: none"> • 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: $\forall j(p(j t) = \frac{1}{n_c})$	$GINI_{SPLIT}$ (Weighted GINI Index) $GINI_{SPLIT} = \sum_{i=1}^k \frac{n_i}{n} \cdot GINI(i)$ <ul style="list-style-type: none"> • 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: $\forall i(GINI(i) = 1 - \frac{1}{n_c})$	Entropy $Entropy(t) = - \sum_{j=1}^{n_c} p(j t) \cdot \log_2(p(j t))$ <ul style="list-style-type: none"> • 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: $\forall j(p(j t) = \frac{1}{n_c})$
--	---	---

Information Gain		Classification Error
$GAIN_{SPLIT}(t) = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} \cdot Entropy(i) \right)$ <ul style="list-style-type: none"> 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 $n = \sum_{i=1}^k n_i$ <p>Key Note: A higher $GAIN_{SPLIT}$ is preferable unlike with the other metrics where a lower value was better.</p> <p>Disadvantage of Information Gain: Tends to prefer splits that result in a large number of partitions, each being small but pure (i.e. overfitting)</p>	<p>Normalizing for Split Size</p> $GainRATIO_{Split} = \frac{Gain_{SPLIT}(t)}{SplitINFO}$ $SplitINFO = - \sum_{i=1}^k \frac{n_i}{n} \cdot \log_2 \left(\frac{n_i}{n} \right)$ <p>SplitINFO penalizes a large split by reducing the gain.</p>	$Error(t) = 1 - \max_j (p(j t))$ <ul style="list-style-type: none"> t – Node in the decision tree j – Class value $p(j t)$ – Probability (i.e. relative frequency) of class value j in node t <p>Minimum Value: 0 when: $\exists j(p(j t) = 1)$</p> <p>Maximum Value: $1 - \frac{1}{n_c}$ when: $\forall j \left(p(j t) = \frac{1}{n_c} \right)$</p>

Stopping Criteria for Decision Tree Induction

<p>Three Stopping Criteria for Decision Tree Induction</p> <ul style="list-style-type: none"> When all records in a node have the same class value When all records in a node have similar attribute values. Early Termination 	<p>Underfitting – When a model is too simple, both training and test errors are large.</p>	<p>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.</p> <ul style="list-style-type: none"> Result: Training error is NOT representative for generalization error. 	<p>Causes of Overfitting</p> <ul style="list-style-type: none"> Noise Insufficient training records (i.e. lack of representative samples)
--	---	--	--

Generalization Error Estimation		
<p>Resubstitution Error Error on the training set.</p> <p>Single Leaf Node Error: $e(t)$ Total Resubstitution Error: $e(T)$</p> $e(T) = \sum e(t)$	<p>Generalization Error Error on the testing data.</p> <p>Single Leaf Node Error: $e'(t)$ Total Generalization Error: $e'(T)$</p> $e'(T) = \sum e'(t)$	<p>Optimistic Estimation Training error is equal to the testing error. $\sum e(t) = \sum e'(t)$</p> <p>Pessimistic Estimation Assign a penalty term to ea. $e'(t) = e(t) + 0.5$</p> <p>Total Pessimistic Error $e'(T) = \sum (e(t)) + N \cdot 0.5$</p> <p>$N$ – Number of leaf nodes.</p> <p>Reduced Error Pruning Use a validation set to estimate the generalization error.</p>

<p>Occam's Razor</p> <p>Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.</p> <p>This is because more complex model has a greater chance of fitting accidentally by errors in the data.</p>	<p>Pre-pruning (Early Stopping Rule)</p> <ul style="list-style-type: none"> Stop the induction algorithm before it becomes a full tree. Typical Stopping Rules: <ul style="list-style-type: none"> All remaining records have the same class value All attribute values are the same. More restrictive conditions: <ul style="list-style-type: none"> 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). 	<p>Post-pruning (Early Stopping Rule)</p> <ul style="list-style-type: none"> 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	Examples of Post-pruning
<p>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) = $(9 + 4 \times 0.5)/30 = 11/30$</p> <p>PRUNE!</p>	<p>– Optimistic error? Don't prune for both cases</p> <p>– Pessimistic error? Don't prune case 1, prune case 2</p> <p>– Reduced error pruning? Depends on validation set</p> <p>Case 1:</p> <p>Case 2:</p>

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 = $0.9 \times (0.8813 - 0.551) = 0.3303$

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	Cheat=Yes 2
Class=No 3	Cheat=No 4

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

Refund	
Yes	No
Class=Yes 0 + 3/9	Class=Yes 2 + 6/9
Class=No 3	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

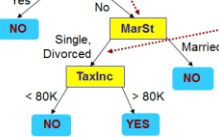
- Pick the most likely of child nodes and use continue down that portion of the tree.

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.

Optimal Tree Induction: NP Hard

Alternate Strategies

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

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.

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.

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

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

Model 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.

$WA = \frac{w_1 \cdot a + w_4 \cdot d}{w_1 \cdot a + w_2 \cdot b + w_3 \cdot c + w_4 \cdot d}$ <p>n – Number of instances covered by rule n_c – Number of positive instances covered by rule.</p>	<p>Proportionality of Cost and Accuracy</p> <ul style="list-style-type: none"> Cost and accuracy are proportional if: $C(Y Y) = C(N N) \text{ and } C(Y N) = C(N Y)$	<p>Sample Size and Model Performance</p> <ul style="list-style-type: none"> Learning Curve – Shows how model accuracy changes (and varies) with sample size. Effects of Small Sample Size: <ul style="list-style-type: none"> Bias in the estimate Variance in the estimate.
--	--	--

Methods for Model Comparison

<p>Holdout – Reserve 2/3 of labelled examples for training and 1/3 for testing.</p> <p>Disadvantages</p> <ul style="list-style-type: none"> 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. 	<p>Random Subsampling – Repeats the holdout method multiple times with replacement.</p> <p>Disadvantages:</p> <ul style="list-style-type: none"> 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. <p>Aggregated Accuracy of k Random Subsamplings</p> $acc_{sub} = \frac{1}{k} \cdot \sum_{i=1}^k acc_i$ <ul style="list-style-type: none"> k – Number of iterations acc_i – Accuracy of the i^{th} iteration. 	<p>Cross Validation – Partition the labelled dataset into k disjoint subsets.</p> <ul style="list-style-type: none"> 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. <p>Aggregated Accuracy of k-Fold Cross Validation</p> $acc_{sub} = \frac{1}{k} \cdot \sum_{i=1}^k acc_i$ <ul style="list-style-type: none"> k – Number of iterations acc_i – Accuracy of the i^{th} iteration. <p>Disadvantages:</p> <ul style="list-style-type: none"> 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.
<p>Bootstrap</p> <ul style="list-style-type: none"> 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. 	<p>Minimum Description Length</p> $Cost(Model, Data) = Cost(Data Model) + Cost(Model)$	

Chapter #05 – Additional Classification Techniques

Rule-Based Classifiers

<p>Classifies records using a collection of “if...then...” rules.</p> <p>Form of Rule:</p> <p>$(Condition) \rightarrow y$</p> <ul style="list-style-type: none"> Condition (Antecedent, LHS) – Conjunction of attributes. y (Consequent, RHS) – Class value. 	<p>Cover – A rule r covers an instance x if the attributes of x satisfy the condition (LHS) of the rule.</p> <p>Coverage of a Rule – Fraction of records that satisfy the antecedent of a rule.</p> $Coverage = \frac{n_R}{N}$ <p>Accuracy of a Rule – For records covered by a rule, it is the fraction of records that have the matching class value.</p>	<p>Mutually Exclusive Rule Set – Rules in the set are independent of each other such that each record is covered by at most one rule.</p> <p>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.</p>	<p>Decision Tree – Can be used to form a mutually exclusive, exhaustive rule set.</p> <p>Rules in a decision tree can be simplified.</p> <p>Effects of rule simplification:</p> <ul style="list-style-type: none"> Problem #1: Rules become non-mutually exclusive. Solution: <ul style="list-style-type: none"> 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.
---	---	---	--

<p>Rule Ordering Schemes</p> <p>Rules Based Ordering – Individual rules are ranked based off their quality.</p> <ul style="list-style-type: none"> 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. <p>Class-Based Ordering – Rules that belong to the same class appear together.</p> <ul style="list-style-type: none"> Advantage: Simplifies rule ordering. Disadvantage: May allow a lower quality rule to have higher priority than a higher quality one. 	<p>Direct Method for Rule Building – Extract rules directly from the data.</p> <ul style="list-style-type: none"> Examples: RIPPER, CN2, Holte’s IR <p>Indirect Method for Rule Building – Extract rules from other classification models (e.g. decision tree, neural network, etc.)</p> <ul style="list-style-type: none"> Examples: C4.5rules 	<p>Sequential Covering Algorithm</p> <ol style="list-style-type: none"> Start with an empty rule set. Grow a rule using the “Learn-One-Rule” function. Remove training records covered by the rule. Repeat steps #2 and #3 until stopping criterion is met. <p>Aspects of Sequential Covering</p> <ol style="list-style-type: none"> Rule Growing Instance Elimination Rule Evaluation Stopping Criterion Rule Pruning 	<p>Rule Growing Strategies</p> <ol style="list-style-type: none"> General to Specific <ol style="list-style-type: none"> Example: Ripper Specific to General <p>CN2 Algorithm</p> <ol style="list-style-type: none"> Start from an empty rule Add conjuncts that minimize the entropy measure. Determine the rule consequent by taking majority class of covered instances.
--	---	---	--

<p>Instance Elimination</p> <ul style="list-style-type: none"> 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. 	<p>Stopping Criterion</p> <p>Compute the information gain with the rule. If the gain is insignificant, discard the rule.</p>	<p>Rule Pruning</p> <ul style="list-style-type: none"> Similar to post-pruning of decision trees. Uses reduced error pruning. <ul style="list-style-type: none"> Remove one of the conjuncts of the rule. Compare error rate on validation set before and after pruning. If error improves, remove the conjunct. 	<p>Rule Simplification – Used to reduce the likelihood of overfitting.</p>
---	---	--	---

Rule Evaluation Metrics

<p>Accuracy = $\frac{n_c}{n}$</p> <p>n – Number of instances covered by rule n_c – Number of positive instances covered by rule.</p>	<p>Laplace = $\frac{n_c + 1}{n + k}$</p> <p>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.</p>	<p>m_estimate = $\frac{n_c + p \cdot k}{n + k}$</p> <p>$n$ – 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.</p>
---	--	--

<p>FOIL Information Gain</p> $Gain(R0, R1) = t \cdot \left(\log_2 \left(\frac{p_1}{p_1 + n_1} \right) + \log_2 \left(\frac{p_0}{p_0 + n_0} \right) \right)$ <p>$R0$ – Initial Rule $R1$ – Modified version of $R0$ with added conjunct t – Number of positive instances covered by both $R0$ and $R1$ (i.e. the intersection) 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 $R1$</p>	<p>RIPPER Algorithm</p> <ol style="list-style-type: none"> For two classes, define one class as positive class and other as negative class. <ol style="list-style-type: none"> In two class problem, negative class is the default class. In multi-class problem, create list of classes ordered by increasing prevalence. <ol style="list-style-type: none"> Select smallest as first as positive class and rest are negative class. Learn rules for the smallest class first. Repeat with next smallest class. 	<p>RIPPER Algorithm – Growing a Rule</p> <ol style="list-style-type: none"> Start from an empty rule set. Add conjuncts as long as they improve FOIL Information Gain (i.e. General-to-Specific). Stop adding conjuncts when the rule starts covering negative examples. Begin pruning the rule immediately (i.e. before generating new rules) using Reduced Error Pruning. Delete conjuncts to maximize v as defined by: $v = \frac{p - n}{p + n}$ <p>p – Number of positive instances covered by the rule. n – Number of negative instances covered by the rule.</p>
---	--	--

RIPPER Algorithm – Building the Rule Set	C4.5rules – Indirect Method	
<ol style="list-style-type: none"> 1. Use Sequential Covering <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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%. 	<ol style="list-style-type: none"> 1. Start from an unpruned decision tree. 2. For each rule $r: A \rightarrow y$, <ol style="list-style-type: none"> 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. 	<ol style="list-style-type: none"> 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. <p>DescriptionLength = $L(\text{error}) + g \cdot L(\text{model})$</p> <ul style="list-style-type: none"> • $L(\text{error})$ – Number of bits required to encode misclassified examples. • $L(\text{model})$ – 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

<p>Instance-Based Classifier – Stores all training records and uses the training records directly to predict the class label of unseen records.</p> <p>Rote-Learner – Memorizes the entire training set and performs classification only if attributes of a record match one the training examples exactly.</p> <p>Nearest Neighbor – Uses k “closest” training records (i.e. nearest neighbors) for performing classification.</p>	<p>Nearest Neighbor Classifier Requirements</p> <ol style="list-style-type: none">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.	<p>Classifying an Unseen Record</p> <ol style="list-style-type: none">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
<p>Voronoi Diagram – Used to depict the decision boundaries for a Nearest Neighbor classifier.</p> <p>Euclidean Distance</p> $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$ <p>Manhattan Distance</p> $d(p, q) = \sum_i p_i - q_i $	<p>Determining the Class from the Nearest Neighbor List</p> <p>Option #1 – Take the majority vote among the k-Nearest Neighbors.</p> <p>Option #2 – Weight the vote according to the distance using the weight factor:</p> $WeightFactor = \frac{1}{d^2}$	<p>Effect of the Value of k</p> <ul style="list-style-type: none">• 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. <p>Attribute Value Scaling Issues</p> <ul style="list-style-type: none">• 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

<p>PEBLs</p> <ul style="list-style-type: none"> • Nearest neighbor algorithm that works with both continuous and nominal features. • Each record is assigned a weight factor. • Number of nearest neighbors, $k = 1$ <p>Weighted Euclidean Distance</p> $d(p, q) = \sqrt{\sum_i w_i \cdot (p_i - q_i)^2}$ <p>w_i – Weight of parameter i</p>	<p>Distance Between Nominal Attributes Value Difference Metric</p> $d(v_1, v_2) = \sum_i \left \frac{n_{1,i}}{n_1} - \frac{n_{2,i}}{n_2} \right $ <p> 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 </p>	<p>Similarity Function Used in PEBLS</p> $d(X, Y) = w_X \cdot w_Y \sum_{i=1}^k d(X_i, Y_i)^2$ <p> 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. </p> <p>$w_X = \frac{\text{Number of Times } X \text{ is Used in Prediction}}{\text{Number of Times } X \text{ Predicts Correctly}}$</p> <p>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.</p>
--	--	--

Bayesian Classifiers

<p>Condition Probability Review $P(C A)$ – Probability of C given A.</p> $P(C A) = \frac{P(C \cap A)}{P(A)}$ $P(A C) = \frac{P(C \cap A)}{P(C)}$	<p>Bayes Theorem</p> $P(A C) = \frac{P(C A) \cdot P(A)}{P(C)}$ <p>Bayes Classifier – A probabilistic framework for solving classification problems.</p>	<p>Bayes Theorem Example</p> $P(SN M) = 0.5$ $P(M) = \frac{1}{50000}$ $P(SN) = \frac{1}{20}$ <p>Hence:</p> $P(M SN) = \frac{P(SN M) \cdot P(M)}{P(SN)} = \frac{0.5 \cdot \frac{1}{50000}}{\frac{1}{20}} = 0.0002$	<p>Requirement of Naïve Bayesian Classifiers</p> <ul style="list-style-type: none"> Consider each attribute (A_1, A_2, \dots, A_n) as independent random variables. Consider the class (C) label as a random variable. <p>Goal is to find: $P(C A_1, A_2, \dots, A_n)$</p>
---	---	--	---

<p>Multi-Attribute Bayes Theorem</p> $P(C A_1, A_2, \dots, A_n) = \frac{P(A_1, A_2, \dots, A_n C) \cdot P(C)}{P(A_1, A_2, \dots, A_n)}$ <p>Classification Approach: Select the value of C that maximizes the above equation.</p>	<p>Naïve Bayesian Simplification</p> $P(A_1, A_2, \dots, A_n C) = P(A_1 C) \cdot P(A_2 C) \cdot \dots \cdot P(A_n C)$ <p>This equation assumes independence among each A_i attribute when the class is given.</p>	<p>Estimating the Class Probability</p> $P(C) = \frac{N_C}{N}$ <p>C – Class value N_C – Number of training records with class value C N – Total number of training records.</p>
---	--	--

<p>Estimating the Attribute-Class Probability</p> $P(A_i C) = \frac{ A_{i,C} }{N_C}$ <ul style="list-style-type: none"> 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. 	<p>Handling Continuous Variables</p> <p>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.</p> <p>Simplest Approach: Use a two-way (i.e. binary) split.</p>	<p>Option #2: Assume attribute follows a normal distribution and use that mean (μ) and standard deviation (σ) to estimate the conditional probability.</p> $P(A_i = a_i C = c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \cdot e^{\frac{-1 \cdot (a_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$ <p>a_i – Continuous value for attribute c_j – Class value μ_{ij} – Mean for value for records with attribute A_i and class value c_j σ_{ij} – Standard deviation for value for records with attribute A_i and class value c_j</p>
	<p>Standard Deviation (σ)</p> $\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n - 1}}$	

Handling Zero Value Conditional Probabilities – Further Conditional Probability Estimation

<p>Original/Standard</p> $P(A_i C) = \frac{ A_{i,C} }{N_C}$ <p>$A_{i,C}$ – Number of records with attribute value A_i and class value C N_C – Number of records with class value C</p>	<p>Laplace</p> $P(A_i C) = \frac{ A_{i,C} + 1}{N_C + k}$ <p>$A_{i,C}$ – 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</p>	<p>m-Estimate</p> $P(A_i C) = \frac{ A_{i,C} + mp}{N_C + m}$ <p>$A_{i,C}$ – 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 $A_{i,C} = 0$. Between 0 and 1. m – Equivalent sample size. Used balance between p and $\frac{ A_{i,C} }{N_C}$</p>
---	---	---

Support Vector Machine (SVM)

<p>Goal: Find a linear hyperplane (i.e. decision boundary) that has maximum separation (i.e. margin) between the two classes.</p> <p>Reason for Maximum Margin: Decision boundaries with large margins tend to have better (i.e. lower) generalization error.</p>	<p>Margin = $\frac{2}{\ \vec{w}\ }$</p> <p>$\ \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</p>	<p>Non-Linearly Separable Datasets</p> <p>Slack Variables</p> <ul style="list-style-type: none"> 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. <p>Higher Order Remapping</p> <p>Problem: Not all classification problems will be linearly separable.</p> <p>Solution: Remap the data into a higher dimensional space (e.g. combine multiple variables at higher order).</p>
--	--	--

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:

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

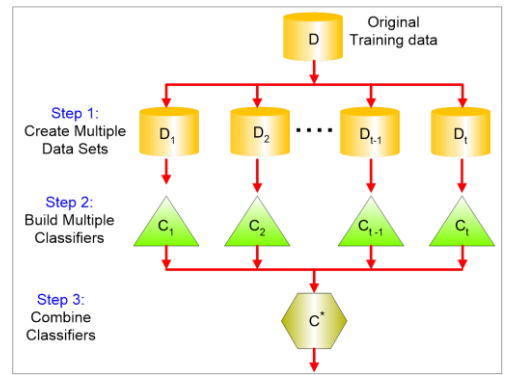
n – Number of training records.

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



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

AdaBoost

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

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

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.

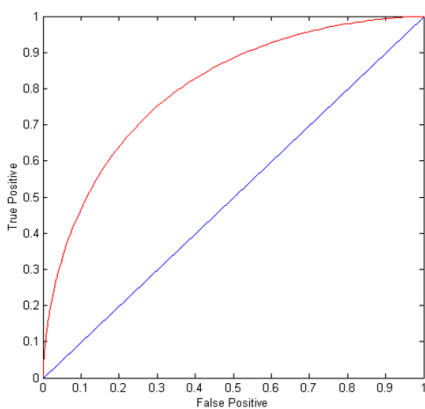
α_j – Importance weight of classifier j

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

$$C^* = \operatorname{argmax}_y \sum_{j=1}^T \alpha_j \cdot \delta(C_j(x) = y)$$

Receiver Operating Characteristics

ROC Curve



Key Terms:

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

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

- 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.

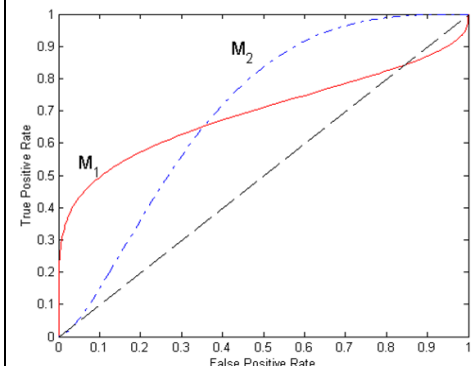
Other Terms:

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

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

- 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).



Chapter #08 – Cluster Analysis: Basic Concepts and Algorithms

<p>Intercluster distance – Distance between objects in two different clusters.</p> <p>Intracluster distance – Distance between objects within the same cluster.</p> <p>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.</p>	<p>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)</p> <p>Hierarchical Clustering – A set of nested clusters organized as a hierarchical tree.</p>	<p>Distinctions between Sets of Clusters</p> <p>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.</p> <p>Fuzzy versus Non-fuzzy – In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1. It resembles probabilistic clustering.</p> <p>Partial versus Complete – Only a portion of the data is clustered.</p> <p>Heterogeneous versus Homogeneous – Clusters of widely different sizes, shapes, and densities. If the data is non-homogeneous, then the clusters will not be homogeneous.</p>
---	---	--

Different Types of Clusters

<p>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.</p>	<p>Center-Based Clusters – A set of objects such that an object in a cluster is closer to the center of its cluster than to the center of any other cluster.</p> <p>Centroid – Average of all the points in the cluster.</p> <p>Medoid – Most representative point in the cluster.</p>	<p>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.</p>	<p>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.</p> <p>Used when the clusters are irregular or intertwined and when noise and outliers are present.</p>	<p>Shared Property Clusters or Conceptual Clusters – Clusters that share some common property or represent a particular concept.</p>
---	--	---	--	---

<p>Objective Function</p> <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. 	<p>K-Means Clustering</p> <ul style="list-style-type: none"> 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) 	<p>K-Means Algorithm</p> <pre> 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 </pre> <p>Runtime: $O(n \cdot K \cdot d \cdot I)$</p> <ul style="list-style-type: none"> n – Number of points/objects in the dataset K – Number of clusters d – Number of attributes I – Number of iterations of the loop.
---	---	---

<p>K-Means Details</p> <ul style="list-style-type: none"> 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. 	<p>Sum of Squared Error</p> $SSE = \sum_{i=1}^k \left(\sum_{x \in C_i} (dist(m_i, x))^2 \right)$ <ul style="list-style-type: none"> 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 <p>A larger SSE indicates worse clustering.</p> <p>Increasing K usually decreases SSE.</p>	<p>Solutions to Initial Centroids Problem</p> <ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> Example: Select the most widely separated. Post Processing Bisecting K-Means. 	<p>Types of Clusters where K-Means is Not Naturally Well Suited</p> <ul style="list-style-type: none"> Different Size Clusters Different Density Clusters Non-globular (e.g. round) shaped clusters. <p>Possible Solution: Use a higher value of K than is expected and merge clusters via post-processing.</p>
--	--	--	---

<p>K-Means and Empty Clusters</p> <ul style="list-style-type: none"> Basic K-Means can yield empty clusters. Solutions: <ul style="list-style-type: none"> Choose a point that contributes most to SEE as the replacement centroid. Choose a point from the highest SSE cluster as replacement centroid. Repeat above steps if there are multiple empty clusters. 	<p>Updating K-Means Incrementally</p> <ul style="list-style-type: none"> In Basic K-Means, centers are updated as points are assigned. In incremental K-Means, centers are updated after each assignment. <ul style="list-style-type: none"> 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. 	<p>K-Means Preprocessing</p> <ul style="list-style-type: none"> Eliminate Outliers Normalize the data (e.g. distance length) <p>K-Means Post-processing</p> <ul style="list-style-type: none"> 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

<p style="text-align: center;">Decision Tree Algorithm</p> <p>Advantages</p> <ul style="list-style-type: none"> • 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) <p>Disadvantages</p> <ul style="list-style-type: none"> • 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. 	<p style="text-align: center;">Rule Based Classifiers</p> <p>Advantages</p> <ul style="list-style-type: none"> • As highly expressive as decision trees <ul style="list-style-type: none"> ○ 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. 	<p style="text-align: center;">Rule Based Classifiers</p> <p>Advantages</p> <ul style="list-style-type: none"> • 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. <p>Disadvantages</p> <ul style="list-style-type: none"> • 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.
<p style="text-align: center;">Bayesian Classifier</p> <p>Advantages</p> <ul style="list-style-type: none"> • Robust to isolated noise points. • Handles missing values by ignoring them in the probability estimate calculations. • Robust to irrelevant attributes. <p>Disadvantages</p> <ul style="list-style-type: none"> • Independence assumption may not hold for some attributes (in such cases, must use a technique known as Bayesian Belief Networks). 		

Chapter #08 – Cluster Analysis

Contiguous Cluster – A set of point such that a point in the cluster is closer (or more similar) to one or more 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 by low density regions.	Prototype Cluster – A cluster is a set of objects that are closer (i.e. more similar) to the prototype (e.g. centroid) that defines that cluster than it is to any other cluster's prototype.
--	--	--

Hierarchical Clustering

Hierarchical Clustering Produces a set of nested clusters organized as a hierarchical tree.	Advantages of Hierarchical Clustering <ul style="list-style-type: none"> Does not assume any particular number of clusters. Hierarchical clusters may correspond to actual/meaningful taxonomies (e.g. phylogenetic dendrogram) 	Types of Hierarchical Clustering <ul style="list-style-type: none"> Agglomerative (Bottom-Up) <ul style="list-style-type: none"> Start with each point as its own cluster. Merge the “closest” pair of clusters in each iteration. Divisive (Top-Down) <ul style="list-style-type: none"> Start with all points in one cluster At each iteration, split the cluster until each point is its own cluster. 	Basic Agglomerative Clustering Algorithm Compute the proximity matrix . Let each point be its own cluster Repeat Merge the two closest/most similar clusters. Update the proximity matrix Until Only a single cluster remains
--	--	---	---

Defining Cluster Proximity

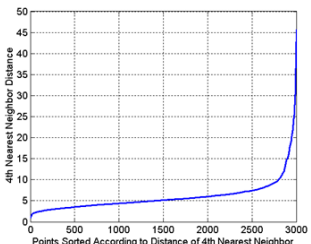
Critical Step in Agglomerative Clustering – Creating the proximity matrix. Need a way to define proximity. Updating the Proximity Matrix – Delete the rows and columns corresponding to the clusters to be merged and add a new row and column for the merged cluster.	Methods for Defining Inter-Cluster Similarity <ul style="list-style-type: none"> Min Distance (Single Linkage) Max Distance (Complete Linkage) Group Average Distance Between Centroid Ward's Method 	Minimum/Single Linkage – Distance between two clusters is based on the two closest (i.e. most similar) points in the clusters. Single Link – Since distance is determined by only one pair of points. Advantage: Can handle non-elliptical shapes. Disadvantage: Sensitive to noise and outliers	Maximum/Complete Linkage – Similarity is based on the two least similar (i.e. most distant) points in the different clusters. Complete Link – Since determined by all points in the two clusters. Advantage: Less susceptible to noise and outliers. Disadvantages: <ul style="list-style-type: none"> Tends to break up large clusters. Biased towards globular clusters.
---	--	---	---

Cluster Average – Proximity of two clusters is the average of the pairwise proximity between points in the two clusters. $P(C_1, C_2) = \frac{\sum_{p_i \in C_1} \sum_{p_j \in C_2} \text{dist}(p_i, p_j)}{ C_1 \cdot C_2 }$ Compromise between single and complete link. Advantage: Less susceptible to noise and outliers. Disadvantage: Biased towards globular clusters.	Ward's Method: Similarity of two clusters is based on the increase in SSE when the two clusters are merged. Advantages: <ul style="list-style-type: none"> Less susceptible to noise and outliers Hierarchical analog of K-Means (can be used to initialize K-Means) Disadvantage: <ul style="list-style-type: none"> Biased towards globular clusters 	Sum of Squared Error $SSE = \sum_{i=1}^k \left(\sum_{x \in C_i} (\text{dist}(m_i, x))^2 \right)$ <ul style="list-style-type: none"> 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 	Time Complexity of Hierarchical Clustering <ul style="list-style-type: none"> Building the Proximity Matrix: $O(N^2)$ Number of Iterations: N Cost to Search Matrix at Each Iteration: $O(N^2)$ Total Runtime: $O(N^3)$. May be able to improve it to $O(N^2 \log(N))$
---	---	---	--

Limitations of Hierarchical Clustering <ul style="list-style-type: none"> Once a decision to merge two clusters is made, it cannot be undone. No global objective function is minimized (make only local/greedy decisions) Different schemes have one or more problems: <ul style="list-style-type: none"> Sensitivity to noise/outliers Difficulty handling different sized clusters and convex shapes Breaking large clusters. 	Divisive Hierarchical Clustering Algorithm Construct a minimum spanning tree (MST) repeat Create a new cluster by breaking the link corresponding to the largest distance until all points are individual clusters (i.e. singleton clusters)	Building a Minimum Spanning Tree Start with a tree consisting of any point repeat Look for the closest pair of points (p,q) such that point (p) is in the tree and the other (q) is not. Add q to the tree by putting an edge between p and q until all points are in the tree
--	---	--

DBSCAN – A Density Based Algorithm

<p>Density – Number of points within a specified volume/radius.</p> <p>Two Parameters of DBSCAN</p> <ul style="list-style-type: none"> • EPS – Neighborhood radius • MinPts – Threshold for the minimum number points around an object (including itself as one object) 	<p>Core Point – Any point that has equal to or more than a specified number of points (<i>MinPts</i>) within its neighborhood (<i>EPS</i>)</p> <p>Border Point – Any point that has fewer points than the specified threshold (<i>MinPts</i>) but is in the neighborhood (<i>EPS</i>) of a core point.</p> <ul style="list-style-type: none"> • Can use difference distance metrics to determine “closest core point” to assign to. <p>Noise Point – Any point that is neither a border point nor a core point.</p>	<p>Lecture Slides DBSCAN Algorithm</p> <p>Label all points as core, border, or noise</p> <p>Eliminate all noise points</p> <pre> current_cluster_label = 0 for all core points do if the core point has no cluster label then current_cluster_label = current_cluster_label + 1 Label the core point with the current cluster label for all points in the Eps-Neighbor of the current point do if the point does not have a cluster label then Label the point with the current cluster label end if end for end if end for </pre>
---	---	---

<p>Strengths of DBSCAN</p> <ul style="list-style-type: none"> • Resistant to noise and outliers • Can handle clusters of different shapes and sizes. <p>Weaknesses of DBSCAN</p> <ul style="list-style-type: none"> • Varying densities • High dimensional data (volume increases at rate r^d for hypersphere) 	<p>Determining EPS</p> <ul style="list-style-type: none"> • For points in a cluster, their k^{th} nearest neighbors are at roughly the same distance. • Noise points have their k^{th} nearest neighbor far away. • Plot sorted distance of k^{th} neighbor and use graph to determine EPS. 	<p>DBSCAN Algorithm</p> <p>Label all points as core, border, or noise</p> <p>Eliminate all noise points</p> <p>Put an edge between all core points within Eps of each other</p> <p>Make each group of connected core points into a separate cluster.</p> <p>Assign each border point to one of the cluster(s) of its associated core point(s).</p>
---	--	---

Cluster Validity

<p>Cluster Validity – Similar to the question of evaluating the “goodness” or the clusters.</p> <p>Reasons to Evaluate Cluster Validity</p> <ol style="list-style-type: none"> 1. Avoid finding patterns in noise 2. Compare clustering algorithms 3. Compare two sets of clusters 4. Compare two clusters. 	<p>Aspects of Cluster Validity Analysis</p> <ol style="list-style-type: none"> 1. Determine Clustering Tendency – Distinguish whether non-random structure exists in the data 2. Compare clustering results to externally known data (e.g. class labels) 3. Evaluate how well the clustering performed without referencing external data. 4. Compare the results of two different sets of cluster analyses to determine which is better. 5. Determine the “correct” number of clusters 	<p>Measures of Cluster Validity</p> <p>External Index – Used to measure the extent to which cluster labels match externally supplied labels.</p> <ul style="list-style-type: none"> • Example: Entropy <p>Internal Index: Used to measure the goodness of a clustering structure without using external information.</p> <ul style="list-style-type: none"> • Example: Sum of squared error (SEE) <p>Relative Index: Used to compare two different clusterings or clusters. Usually uses an external or internal index to compute the relative index.</p>
---	---	--

Proximity Matrix – Symmetric matrix showing the distance between each pair of objects.

Incidence Matrix – Symmetric matrix with a “1” if two objects are in the same cluster and a “0” otherwise.

If the **computed correlation between the two matrices is high**, this indicates that points that are in the same matrix are “close”/“similar” to each other.

May not be useful for density or continuity-based clusters.

Correlation: Between -1 and 1. As proximity goes down, incidence goes up so a good correlation is -1.

Proximity/Similarity Matrix – Can be used to create a visualization for cluster validation. **Sort the rows and columns such that objects belonging to the same cluster are together.**

Block Diagonal Structure – The similarity is non-zero (i.e. 1 or highly similar) inside the blocks of the similarity matrix whose entries represent intra-cluster similarity and 0 elsewhere. (The blocks come from the matrix being sorted)

Ideal Cluster Similarity Matrix

	P1	P2	P3	P4	P5
P1	1	1	1	0	0
P2	1	1	1	0	0
P3	1	1	1	0	0
P4	0	0	0	1	1
P5	0	0	0	1	1

Similarity Matrix Visualization for Well Separated Clusters

The figure is a heatmap visualization of a similarity matrix for 100 points. The x-axis and y-axis are both labeled 'Points' and range from 0 to 100. A color scale on the right indicates similarity values from 0 (blue) to 1 (red). The matrix shows three distinct blocks of high similarity (red) along the diagonal, indicating three well-separated clusters. The first cluster is in the top-left corner (points 0-33), the second is in the middle (points 34-66), and the third is in the bottom-right corner (points 67-99). The off-diagonal blocks show low similarity (blue).

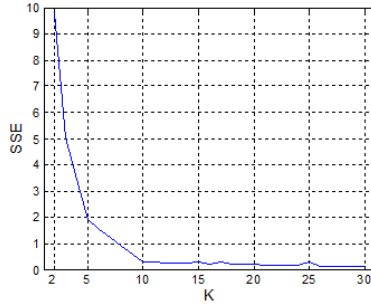
Internal Index – SSE

$$SSE = \sum_{i=1}^k \sum_{x_j \in C_i} dist(x_j, c_i)^2$$

- k – Number of clusters
- x_j – Point in cluster C_i
- c_i – Centroid of cluster C_i
- Good for **comparing two clusters or two clusterings**.
- Can be used to estimate the number of clusters (k).
 - May not work for density based clusters.

Using SSE to Estimate K

Using the graph below, you can see there are about 10 clusters.



Challenges to Assessing Cluster Validity

1. **Limited Scope** – Cluster validity may be limited in its scope of applicability.
2. **Necessity of a Framework** – A framework is needed to interpret the measures of a result. Example: Is a score of “10” good or bad?
 - a. **Example:** Is a score of “10” good or bad?
 - b. **Statistics can be used to build a framework for cluster validity analysis. The more atypical the clustering result, the more likely it represents a valid structure in the data.**
 - c. Can use mean and standard deviation to estimate the likelihood the SSE is due to randomness.
 - d. Can use correlation to estimate whether the correlation is due to randomness.
3. **Using a Framework for Comparison** – While a framework is less important when comparing two clustering results, statistics are required to determine if the difference between two results is significant.

Cluster Cohesion – Measures how closely related are objects within a cluster. **Example:** SSE

Cluster Separation – Measures how distinct or well-separated a cluster is from other clusters.

BSS + WSS = Constant

Between Sum of Squares (BSS)

$$BSS = \sum_{i=1}^K m_i \cdot dist(c_i, c)^2$$

- K – Number of clusters
- m_i – Number of objects in cluster i
- c_i – Centroid of cluster i
- c – Centroid of the entire dataset (i.e. all K clusters combined)

Within Sum of Squares (WSS)

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

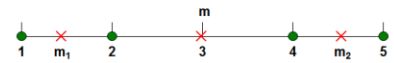
One dimensional version of WSS.

- m_i – Mean of the i^{th} cluster.
- C_i – The i^{th} cluster.

Internal Measures: Cohesion and Separation

- Example: SSE

– BSS + WSS = constant



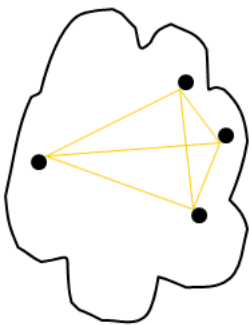
K=1 cluster: $WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$
 $BSS = 4 \times (3-3)^2 = 0$
 Total = 10 + 0 = 10

K=2 clusters: $WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$
 $BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$
 Total = 1 + 9 = 10

Graph Based View of Cluster and Cohesion

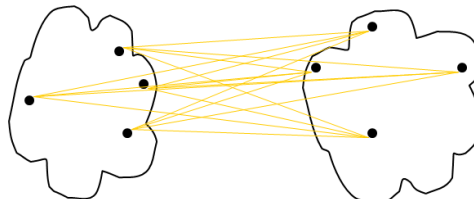
Cohesion – Sum of the weights of all links within a cluster.

$$cohesion(C_i) = \sum_{\substack{x \in C_i \\ y \in C_j}} proximity(x, y)$$



Separation – Sum of the weights between nodes in the cluster to nodes outside the cluster.

$$Separation(C_i, C_j) = \sum_{\substack{x \in C_i \\ y \in C_j}} proximity(x, y)$$



Silhouette Coefficient

- Metric that **combines both cohesion and separation**.
- **Useful for comparing clusters and clusterings.**
- **Range:** -1 (worst) to 1 (best)

Step #1: For a given object i , a represents the average distance between i and all other points in its cluster.

Step #2: For a given object i , b represents the average distance between i and points not in the same cluster as i .

Step #3: Calculate silhouette coefficient (s_i) for point i :

$$s_i = \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

Can calculate the average silhouette for a cluster or clustering.

Chapter #09 – Cluster Analysis: Additional Issues and Algorithms

Data Characteristics

The following is a list of **parameters that can strongly affect cluster analysis**

High Dimensionality <ul style="list-style-type: none"> • Euclidean definition of density – Number of points in a given volume. • As number of dimensions increases, volume increases rapidly making density meaningless. <ul style="list-style-type: none"> ○ For a hypersphere with radius r and d dimensions, volume grows at the rate r^d • Possible Solution: Employ dimensionality reduction techniques. 	Dataset Size <p>Most clustering algorithms only work well on small or medium sized datasets and are unable to handle large datasets.</p> <ul style="list-style-type: none"> • Scalability – Property of a clustering algorithm that quantifies how well it performs on large datasets. 	Sparseness <p>Asymmetric Attribute – Zero values in the data vector are not as important as non-zero values.</p> <p>Possible Solution: Use similarity measures that are more appropriate for asymmetric attributes.</p>	Noise and Outliers <p>Atypical (i.e. outlier) points can significantly degrade the performance of some clustering algorithms.</p> <p>Single Link Clustering Example – Outlier points can cause clusters to be joined that should not be.</p> <p>Possible Solution: Some clustering algorithms like DBSCAN filter noise points.</p>
---	---	--	--

Types of Attributes and Dataset <p>Types of Datasets</p> <ul style="list-style-type: none"> • Structured, graph, ordered <p>Types of Attributes</p> <ul style="list-style-type: none"> • Categorical (nominal or ordinal) or Quantitative (interval or ratio) • Binary, discrete, or continuous <p>Different proximity measures and density measures are needed for different types of data.</p> <p>Dataset analysis is complicated when attributes are of significantly different types (e.g. binary with continuous)</p>	Attribute Scale <ul style="list-style-type: none"> • Attributes may be on vastly different scales (e.g. height, weight). • Differences in attribute scale may cause the clustering to favor a single attribute or set of attributes. • Possible Solution: Normalize or standardize the attributes. 	Mathematical Properties of the Data Space <p>Certain clustering techniques rely on techniques (e.g. mean, density, etc.) that only make sense in a specific mathematical space (e.g. Euclidean space)</p>
--	--	--

Clustering Types

Prototype Based Clusters – A cluster is a set of objects that are closer (i.e. more similar) to the prototype that defines the cluster than to any other cluster's prototype.	Graph Based Clustering – Nodes in the in the graph are objects and the edge represent connections between objects. Example: A cluster is a connected component.	Contiguity Based Clusters – Objects are clustered together only if they are within a specified distance of each other.	Density Based Cluster – A cluster is a dense region of objects that is surrounded by a region of low density.
---	--	--	---

Cluster Characteristics

Data Distribution – Some clustering algorithms assume a particular distribution of the data. Mixture models assume a specific data distribution.	Shape – Some clusters are regularly shaped (e.g. rectangular, globular). <ul style="list-style-type: none"> • DBSCAN work on any shape clusters. • Proto-type based algorithms (e.g. K-Means and some hierarchical clustering metrics – complete link) do not handle different size clusters well. 	Size – Clusters may be the same size or varying sizes. <ul style="list-style-type: none"> • DBSCAN can work on varying size clusters. • K-Means does not work well on different size clusters. 	Differing Density Clusters – Clusters may have the same or varying densities. <ul style="list-style-type: none"> • Both K-Means and DBSCAN do not support differing density clusters.
---	--	---	--

Poorly Separated Clusters – When clusters overlap, some clustering algorithms merge the clusters while others can separate them. <ul style="list-style-type: none"> • Fuzzy clustering – A technique for handling poorly separated clusters. • K-Means may split separated clusters. • DBSCAN – Merges overlapping clusters. 	Relationships between Clusters – In most clustering techniques, there is no consideration of the relationships between clusters. <ul style="list-style-type: none"> • Self-Organizing Maps (SOM) – Directly considers the relationships between clusters during the clustering process. 	Subspace Clusters – Clusters may exist only in a subset of the dimensions. <ul style="list-style-type: none"> • Identifying subspace clusters becomes more difficult as the number of dimensions increases and the number of subspaces increases <p>Possible Solution: Feature selection</p>	
---	--	---	--

General Characteristics of Clustering Algorithms

<p>Order Dependence – For some algorithms, the quality and number of clusters can vary (dramatically) depending on the order in which the data is processed.</p> <ul style="list-style-type: none"> • Examples: SOM 	<p>Nondeterminism – Some clustering algorithms that are non-deterministic rely on an initialization step that involves random choice. For such algorithms, multiple runs may be necessary.</p> <p>Example: K-Means and Fuzzy c-means use random centroid initialization.</p>	<p>Scalability – For large datasets, algorithms with runtimes of $O(N^2)$ may be intractable.</p> <ul style="list-style-type: none"> • Scalability becomes an even bigger problem if data cannot be kept in main memory (i.e. RAM) 	<p>Parameter Selection – Most clustering algorithms have one or more parameters that must be set by the user.</p> <ul style="list-style-type: none"> • Small change in algorithm parameter can have a large effect on algorithm results. • May require trial-and-error to find best parameter values. <p><i>“The fewer parameters, the better”</i></p>
<p>Transforming the Problem to Another Domain – Some clustering problems map the clustering problem from one domain to another.</p> <p>Example: Graph based clustering transforms the problem of finding clusters to the task of partitioning a proximity graph into connected components.</p>	<p>Treating Clustering as an Optimization Problem – Divides the points into a cluster in a way that maximizes some user specified objective function.</p> <p>Intractable Approach: Exhaustive technique of specifying every possible clustering.</p> <p>Better Solution: Use heuristic approaches that provide good but not necessary optimal solutions.</p> <p>Example: K-Means and Fuzzy c-means</p>		

Expanding Prototype-Based Clustering

<p>Option #1 – Objects belong to every cluster with some weight.</p> <p>Addresses the issue that some clusters are equally close to several cluster prototypes.</p> <p>Example: Fuzzy c-means</p>	<p>Option #2 – Model a cluster via a statistical distribution. Each statistical distribution is modelled by a set of statistical parameters (e.g. mean, variance)</p> <p>Generalizes the notion of a prototype and enables the use of established statistical techniques.</p> <p>Example: Mixture Models</p>	<p>Option #3 – Clusters are considered to have fixed relationships.</p> <p>Can simplify the interpretation and visualization of the data.</p> <p>Example: Self-Organizing Maps</p>
--	---	---

Fuzzy Clustering with Fuzzy c-Means

<p>Fuzzy Clustering – Ideal for when clusters are not well-separated.</p> <p>Traditional Set Theory – Each object belongs to a set with a degree of certainty of either 0 or 1.</p> <p>Fuzzy Set Theory – Each object belongs to a set with a degree of membership between 0 and 1.</p>	<p>Definition of the Fuzzy Pseudo Partition</p> <p>All weights for a point i add up to 1.</p> <ul style="list-style-type: none"> • w_{ij} – Weight for object i in cluster j • k – Number of Clusters $\forall i \left(\sum_{j=1}^k (w_{ij}) = 1 \right)$ <p>Each cluster, C_j, contains with non-zero weight at least one point, but does not contain with a weight one all points.</p> <ul style="list-style-type: none"> • m – Number of data objects • w_{ij} – Weight for object i in cluster j $\forall j \left(0 < \sum_{i=1}^m w_{ij} < m \right)$	<p>Fuzzy c-Means (FCM)</p> <ul style="list-style-type: none"> • Similar in structure to standard K-Means • Computing fuzzy pseudo-partition to similar to the step in K-Means where points are assigned to clusters.
--	---	--

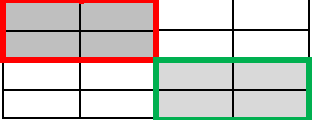
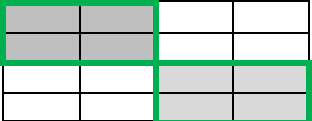
<p>Basic FCM Algorithm</p> <p>Select an initial fuzzy pseudo-partition (i.e. assign values to all w_{ij})</p> <p>repeat</p> <p> Use the fuzzy pseudo-partition to compute the centroids of each cluster</p> <p> Re-compute the fuzzy pseudo partition (i.e. all w_{ij})</p> <p>until Centroids do not change</p>	<p>Strengths and Weaknesses of FCM</p> <ul style="list-style-type: none"> • Unlike K-Means, FCM provides an indication of the degree to which any point belongs to a cluster. • Same strengths and weaknesses as standard K-Means <ul style="list-style-type: none"> ◦ Biased towards globular clusters. ◦ Susceptible to noise and outliers (although less than standard K-Means) • More computationally expensive than K-Means. 	<p>SSE for FCM</p> $FuzzySSE = \sum_{j=1}^K \sum_{i=1}^m w_{ij}^p \cdot dist(x_i, c_j)^2$ <ul style="list-style-type: none"> • m – Number of points in the dataset • K – Number of clusters • c_j – Centroid of the j^{th} cluster. • x_i – i^{th} data point • p – Weight scaling factor between 1 and ∞ • w_{ij} – Weight of point i in cluster j
--	---	--

<p>FCM Step #1 - Initialization</p> <p>Only requirement is that for each object x_i the cluster weights in the fuzzy pseudo-partition add up to 1.</p> <p>Weights are usually randomly initialized.</p>	<p>Step #2 – Calculate the Centroid</p> $c_j = \frac{\sum_{i=1}^m w_{ij}^p \cdot x_i}{\sum_{i=1}^m w_{ij}^p}$ <ul style="list-style-type: none"> • c_j – Centroid of cluster j • x_i – i^{th} data object • w_{ij} – Weight of object i for cluster j 	<p>Step #3 – Calculate the weights</p> $w_{ij} = \frac{\left(\frac{1}{\text{dist}(x_i, c_j)^2} \right)^{p-1}}{\sum_{q=1}^K \left(\frac{1}{\text{dist}(x_i, c_q)^2} \right)^{p-1}}$ <ul style="list-style-type: none"> • c_j – Centroid of cluster j • c_q – Centroid of the q^{th} cluster • x_i – i^{th} data object • w_{ij} – Weight of object i for cluster j • p – Scaling factor between 1 and ∞
---	--	--

Self-Organizing Maps (SOM)

<p>Centroid based. The centroids have a topographical-spatial organization.</p> <p>Many spatial organizations are possible.</p> <ul style="list-style-type: none">• Example: Two dimensional spacing. <p>Centroids that are closer to each other are more related.</p> <ul style="list-style-type: none">• This makes centroids ordered.		<p>Basic Self-Organizing Map Algorithm</p> <p>Initialize the centroids</p> <p>repeat</p> <p> Select the next point $p(t)$.</p> <p> Determine the closest centroid ($m_j(t)$) to $p(t)$</p> <p> Update the closest centroid and all centroids that are within the specified neighborhood.</p> <p>until Centroids do not change much</p> <p>Assign each object to the closest centroid and update the centroids.</p>		<p>Initializing the Centroids</p> <p>Option #1: Select the centroids along the sample range. Generally not a good approach.</p> <p>Option #2: Randomly select a set of points as the initial Centroids.</p>
<p>Selection of the “Next Point”</p> <ul style="list-style-type: none">• Depending on the size of the dataset, some points may be used multiple times while others may not be used at all.• To enhance the influence of certain groups, the probability of selecting certain groups of objects in the dataset may be increased.	<p>Assignment to a Centroid</p> <p>Straightforward use of the distance measure. May use:</p> <ul style="list-style-type: none">• Euclidean distance• Cosine distance	<p>Updating a Centroid</p> $m_j(t+1) = m_j(t) + h_j(t) \cdot (p(t) - m_j(t))$ <ul style="list-style-type: none">• $m_j(t)$ – Current centroid location• $m_j(t+1)$ – Updated centroid location• $h_j(t)$ – Neighborhood function• $p(t)$ – Point selected at time t	<p>Requirements of the Neighborhood Function $h_j(t)$</p> <ul style="list-style-type: none">• Diminishes with time• Enforces a neighborhood effect (i.e. the effect on is strongest on centroids closest to the centroid m_j)	
<p>Example Neighborhood Functions $h_j(t)$</p>				
<p>Gaussian Function</p> $h_j(t) = \alpha(t) \cdot e^{\frac{(-dist(r_j,r_k)^2)}{2\sigma^2(t)}}$ <ul style="list-style-type: none">• r_j – Location of the centroid closest to point $p(t)$• $\alpha(t)$ – Learning rate that decreases monotonically. $0 < \alpha(t) < 1$• $\sigma(t)$ – Typical Gaussian variance parameter. Used to control the width of the neighborhood.	<p>Step Function</p> $h_j(t) = \begin{cases} \alpha(t), & dist(r_j,r_k) \leq Threshold \\ 0, & Otherwise \end{cases}$ <ul style="list-style-type: none">• r_j – Location of the centroid closest to point $p(t)$• $\alpha(t)$ – Learning rate that decreases monotonically. $0 < \alpha(t) < 1$• Threshold – Used to control the width of the neighborhood.	<p>Termination of the Algorithm</p> <ul style="list-style-type: none">• Occurs when the centroids converge to a set of stable values.• Termination make take a long time.• Convergence is not guaranteed if $h_j(t)$ does not approach 0.		
<p>Strengths of SOM</p> <ul style="list-style-type: none">• Facilitates the interpretation and visualization of clustering results since clusters that are neighbors are more related to one another than clusters that are not.		<p>Limitations of SOM</p> <ul style="list-style-type: none">• User must specify multiple settings including: the neighborhood function ($\alpha(t)$, $h(t)$), the grid type, and the number of centroids k.• Often, one SOM cluster does not correspond to a single natural cluster or multiple SOM clusters correspond to a single natural cluster.• SOM lacks a specific objective function.• Not guaranteed to converge if $h(t)$ does not approach 0.		

Grid-Based Clustering

<p>Grid-Based Clustering divides each attribute into a set of contiguous intervals. This forms a set of grid cells.</p> <p>τ – Minimum grid cell density threshold.</p>	<p style="text-align: center;">Basic Grid-Based Clustering Algorithm</p> <ol style="list-style-type: none"> 1. Define a set of grid cells. 2. Assign each object to its associated grid cell and compute the density of each grid cell. 3. Eliminate cells that have a density below a specified threshold, τ. 4. Form clusters from the remaining contiguous/adjacent grid cells. 	<p style="text-align: center;">Defining the Grid Cells</p> <p>Naive Approach – Split the range of values into equal widths. This means all grid cells have equal volume.</p> <p>Equal Frequency Discretization – Divide the values of an attribute into intervals so that each interval contains an equal number of points.</p> <p>The definition of the grid cell has a major impact on the clustering results.</p>
<p style="text-align: center;">Definition of Density</p> $Density = \frac{\#Points}{Volume}$ <p>This is not a straight threshold as used with DBSCAN.</p>	<p style="text-align: center;">Defining Adjacency</p> <p>4-Adjacent Cells – Applicable in two dimensions. Only count cells as adjacent if they are to the left, right, bottom, or top.</p>  <p style="text-align: center;">One Cluster Using 4-Adjacent</p> <p>8-Adjacent Cells – Applicable in two dimensions. Count two cells as adjacent if next or diagonal to each other.</p>  <p style="text-align: center;">Two Clusters Using 4-Adjacent</p>	<p style="text-align: center;">Strengths of Grid-Based Clustering</p> <ul style="list-style-type: none"> • Efficient as it only requires a single pass through the data $O(m)$ • If data is sparse, grid cells only need to be created for non-empty cells. • Basis of many algorithms • Total Runtime: $O(m \cdot lg(m))$ due to adjacency computation with search tree <p style="text-align: center;">Weaknesses of Grid-Based Clustering</p> <ul style="list-style-type: none"> • Very dependent on the value of τ. <ul style="list-style-type: none"> ◦ If τ is too high, clusters are lost. ◦ If τ is too low, then separate clusters may be merged. • Struggles with clusters of different densities. • Performs poorly with high dimensional data. • Rectangular grid cell boundaries do not capture globular clusters well. <ul style="list-style-type: none"> ◦ May discard partially empty adjacent cells unnecessarily although algorithm can be tweaked for that. ◦ Can use finer grid for improved granularity.

Comparing the Clustering Algorithms

	K-Means	DBSCAN	Agglomerative Clustering	Fuzzy c-Means
Inclusion of Data in Output	Keeps all objects	Filters noise/outliers	Keeps all objects	Keeps all Objects
Algorithm Type	Prototype-Based	Density Based	Dependent on Distance Metric	Fuzzy Prototype Based
Supports Different Sizes and Shapes	No Biased towards <i>globular</i> clusters	Yes	No Biased towards <i>globular</i> clusters	No Biased towards globular clusters but less biased than K-Means
Supports High Dimensional Data	Yes	No	Yes	Yes
Use of All Attributes	Yes	Yes	Yes	Yes
Struggles with Different Densities	Yes	Yes	No ????	Yes but less than standard K-Means
Ideal Data Distribution	Spherical (Globular) Gaussian	None	Spherical (Globular) Gaussian for Most Distance Metrics	Spherical (Globular) Gaussian for Most Distance Metrics
Considers Cluster Relationships	No	No	No	No
Time Complexity	$O(m)$	$O(m^2)$	$O(m^3)$ but can get as low as $O(m^2 \log(m))$	$O(m)$
Overlapping/Non-separated Clusters	Splits	Merges	Dependent on Distance Cutoff. Both splits and merges	Fuzzy assignment
"Stable" Results	No Non-deterministic	Yes	Yes	No Non-deterministic and order dependence
Critical Algorithm Parameters	K	$MinPts$ and Eps	Definition of Cluster Proximity	c
Problem Formulation	Optimization (of SSE)	None	None	Optimization

	Self Organizing Maps	Grid Based Clustering		
Inclusion of Data in Output	Keeps all objects	Filters noise/outliers		
Algorithm Type	Prototype-Based	Density Based		
Supports Different Sizes and Shapes	No Biased towards <i>globular</i> clusters	No Biased towards <i>rectangular</i> clusters		
Supports High Dimensional Data	Yes	No		
Use of All Attributes	Yes	Yes		
Ideal Data Distribution	Spherical (Globular) Gaussian	None		
Considers Cluster Relationships	Yes	No		
Time Complexity	$O(m)$	$O(m \cdot lg(m))$		
Overlapping/Non-separated Clusters	Splits	Merges		
"Stable" Results	No Order Dependence	Yes		
Critical Algorithm Parameters	k (Number of Centroids) $\alpha(t)$ $h(t)$ Grid Type	τ Definition of Adjacency (e.g. 4-adjacent, 8-adjacent)		
Problem Formulation	None (No objective function)	None		