Feature Selection for Classification
by M. Dash and H. Liu

Group 10

Stanlay Irawan HD97-1976M
Loo Poh Kok HD98-1858E
Wong Sze Cheong HD99-9031U

Slides: http://www.comp.nus.edu.sg/~wongszec/group10.ppt
Feature Selection for Classification

Agenda:

• Overview and general introduction. (pk)
• Four main steps in any feature selection methods. (pk)
• Categorization of the various methods. (pk)
• Algorithm = Relief, Branch & Bound. (pk)
• Algorithm = DTM, MDLM, POE+ACC, Focus. (sc)
• Algorithm = LVF, wrapper approach. (stan)
• Summary of the various method. (stan)
• Empirical comparison using some artificial data set. (stan)
• Guidelines in selecting the “right” method. (pk)
(1) Overview.

- various feature selection methods since the 1970’s.
- common steps in all feature selection tasks.
- key concepts in feature selection algorithm.
- categorize 32 selection algorithms.
- run through some of the main algorithms.
- pros and cons of each algorithms.
- compare the performance of different methods.
- guideline to select the appropriate method.
Feature Selection for Classification

(2) What is a feature?

<table>
<thead>
<tr>
<th>TRS_DT</th>
<th>TRS_TYP_CD</th>
<th>REF_DT</th>
<th>REF_NUM</th>
<th>CO_CD</th>
<th>GDS_CD</th>
<th>QTY</th>
<th>UT_CD</th>
<th>UT_PRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>04/05/93</td>
<td>25119</td>
<td>10002J</td>
<td>001M</td>
<td>10</td>
<td>CTN</td>
<td>22.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>05/05/93</td>
<td>25124</td>
<td>10002J</td>
<td>032J</td>
<td>200</td>
<td>DOZ</td>
<td>1.370</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>05/05/93</td>
<td>25124</td>
<td>10002J</td>
<td>033Q</td>
<td>500</td>
<td>DOZ</td>
<td>1.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>13/05/93</td>
<td>25217</td>
<td>10002J</td>
<td>024K</td>
<td>5</td>
<td>CTN</td>
<td>21.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>13/05/93</td>
<td>25216</td>
<td>10026H</td>
<td>006C</td>
<td>20</td>
<td>CTN</td>
<td>69.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>13/05/93</td>
<td>25216</td>
<td>10026H</td>
<td>008Q</td>
<td>10</td>
<td>CTN</td>
<td>114.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>14/05/93</td>
<td>25232</td>
<td>10026H</td>
<td>006C</td>
<td>10</td>
<td>CTN</td>
<td>69.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>14/05/93</td>
<td>25235</td>
<td>10027E</td>
<td>003A</td>
<td>5</td>
<td>CTN</td>
<td>24.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>14/05/93</td>
<td>25235</td>
<td>10027E</td>
<td>001M</td>
<td>5</td>
<td>CTN</td>
<td>24.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>22/04/93</td>
<td>24974</td>
<td>10035E</td>
<td>009F</td>
<td>50</td>
<td>CTN</td>
<td>118.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>27/04/93</td>
<td>25033</td>
<td>10035E</td>
<td>015A</td>
<td>375</td>
<td>GRS</td>
<td>72.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>20/05/93</td>
<td>25313</td>
<td>10041Q</td>
<td>010F</td>
<td>10</td>
<td>CTN</td>
<td>26.000</td>
</tr>
<tr>
<td>21/05/93</td>
<td>00001</td>
<td>12/05/93</td>
<td>25197</td>
<td>10054R</td>
<td>002E</td>
<td>25</td>
<td>CTN</td>
<td>24.000</td>
</tr>
</tbody>
</table>
Feature Selection for Classification

(3) What is classification?

- main data mining task besides association-rule discovery.
- predictive nature - with a given set of features, predict the value of another feature.

- common scenario:
  - Given a large legacy data set.
  - Given a number of known classes.
  - Select an appropriate smaller training data set.
  - Build a model (e.g., Decision tree).
  - Use the model to classify the actual data set into the defined classes.
Feature Selection for Classification

(4) Main focus of the author.
• survey various known feature selection methods
• to select subset of relevant feature
• to achieve classification accuracy.
  Thus: relevancy -> correct prediction

(5) Why can’t we use the full original feature set?
• too computational expensive to examine all features.
• not necessary to include all features
  (ie. irrelevant - gain no further information).
Feature Selection for Classification

(6) Four main steps in a feature selection method.

- **Generation**: select feature subset candidate.
- **Evaluation**: compute relevancy value of the subset.
- **Stopping criterion**: determine whether subset is relevant.
- **Validation**: verify subset validity.

**Diagram Process**

- **Original feature set** → **Generation** → **Evaluation** → **Stopping criterion**
  - *no* → **Generation**
  - *yes* → **Validation**

Selected subset of feature
Feature Selection for Classification

(7) Generation

- select candidate subset of feature for evaluation.
- Start = no feature, all feature, random feature subset.
- Subsequent = add, remove, add/remove.
- categorise feature selection = ways to generate feature subset candidate.
- 3 ways in how the feature space is examined.
  (7.1) Complete
  (7.2) Heuristic
  (7.3) Random.
Feature Selection for Classification

(7.1) Complete/exhaustive

- examine all combinations of feature subset.
  \{f_1,f_2,f_3\} \Rightarrow \{\{f_1\},\{f_2\},\{f_3\},\{f_1,f_2\},\{f_1,f_3\},\{f_2,f_3\},\{f_1,f_2,f_3\}\}
- order of the search space \(O(2^p)\), \(p - \#\) feature.
- optimal subset is achievable.
- too expensive if feature space is large.

(7.2) Heuristic

- selection is directed under certain guideline
  - selected feature taken out, no combination of feature.
  - candidate = \{\{f_1,f_2,f_3\}, \{f_2,f_3\}, \{f_3\}\}
- incremental generation of subsets.
- search space is smaller and faster in producing result.
- miss out features of high order relations (parity problem).
  - Some relevant feature subset may be omitted \{f_1,f_2\}.
Feature Selection for Classification

(7.3) Random

• no predefined way to select feature candidate.
• pick feature at random (ie. probabilistic approach).
• optimal subset depend on the number of try
  - which then rely on the available resource.
• require more user-defined input parameters.
  - result optimality will depend on how these parameters are defined.
  - eg. number of try
Feature Selection for Classification

(8) Evaluation

• determine the relevancy of the generated feature subset candidate towards the classification task.

  \[ \text{Rvalue} = J(\text{candidate subset}) \]

  if (Rvalue > best_value) best_value = Rvalue

• 5 main type of evaluation functions.

  (8.1) distance (euclidean distance measure).
  (8.2) information (entropy, information gain, etc.)
  (8.3) dependency (correlation coefficient).
  (8.4) consistency (min-features bias).
  (8.5) classifier error rate (the classifier themselves).
Feature Selection for Classification

(8.1) Distance measure

• $z^2 = x^2 + y^2$

• select those features that support instances of the same class to stay within the same proximity.

• instances of same class should be closer in terms of distance than those from different class.

(8.2) Information measure

• entropy - measurement of information content.

• information gain of a feature: (eg. Induction of decision tree)
  \[ \text{gain}(A) = I(p,n) - E(A) \]
  \[ \text{gain}(A) = \text{before A is branched} - \text{sum of all nodes after branched} \]

• select A if gain(A) > gain(B).
Feature Selection for Classification

(8.3) **Dependency measure**

- correlation between a feature and a class label.
- how close is the feature related to the outcome of the class label?
- dependence between features = degree of redundancy.
  - if a feature is heavily dependence on another, than it is redundant.
- to determine correlation, we need some physical value.
  value = distance, information
(8.4) Consistency measure

- two instances are *inconsistent* if they have *matching feature values* but group under *different class label*.

<table>
<thead>
<tr>
<th></th>
<th>f₁</th>
<th>f₂</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance 1</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
</tr>
<tr>
<td>instance 2</td>
<td>a</td>
<td>b</td>
<td>c₂</td>
</tr>
</tbody>
</table>

- select \{f₁,f₂\} if in the training data set there exist no instances as above.
- heavily rely on the training data set.
- min-feature = want smallest subset with consistency.
- problem = 1 feature alone guarantee no inconsistency (eg. IC #).
Feature Selection for Classification

**Filter approach**

- Original feature set
  - Feature selection
    - Evaluation fn
  - Selected feature subset
  - Classifier
    - Evaluation fn <> classifier
    - Ignored effect of selected subset on the performance of classifier.

**Wrapper approach**

- Original feature set
  - Feature selection
    - Classifier
    - Evaluation fn = classifier
    - Take classifier into account.
    - Loss generality.
    - High degree of accuracy.
Feature Selection for Classification

(8.5) Classifier error rate.

• wrapper approach.
  
  \[
  \text{error\_rate} = \text{classifier(feature subset candidate)}
  \]
  
  if (error\_rate < predefined threshold) select the feature subset

• feature selection loss its generality, but gain accuracy towards the classification task.

• computationally very costly.
### Feature Selection for Classification

Comparison among the various evaluation methods.

<table>
<thead>
<tr>
<th>method</th>
<th>generality</th>
<th>time</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance</td>
<td>yes</td>
<td>low</td>
<td>-</td>
</tr>
<tr>
<td>information</td>
<td>yes</td>
<td>low</td>
<td>-</td>
</tr>
<tr>
<td>dependency</td>
<td>yes</td>
<td>low</td>
<td>-</td>
</tr>
<tr>
<td>consistency</td>
<td>yes</td>
<td>moderate</td>
<td>-</td>
</tr>
<tr>
<td>classifier error rate</td>
<td>no</td>
<td>high</td>
<td>very high</td>
</tr>
</tbody>
</table>

- **generality** = how general is the method towards different classifiers?
- **time** = how complex is the method in terms of time?
- **accuracy** = how accurate is the resulting classification task?
### Feature Selection for Classification

(10) Author's categorization of feature selection methods.

<table>
<thead>
<tr>
<th>Measures</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Heuristic</td>
</tr>
<tr>
<td><strong>Distance</strong></td>
<td>Relief</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td>Decision Tree Method (DTM)</td>
</tr>
<tr>
<td><strong>Dependency</strong></td>
<td>Probability of Err &amp; Ave Correlation Coefficient Method (POE+ACC)</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>Focus</td>
</tr>
<tr>
<td><strong>Classifier Error Rate</strong></td>
<td>SBS, SFS</td>
</tr>
</tbody>
</table>
Feature Selection for Classification

(11.1) Relief [generation=heuristic, evaluation=distance].

• **Basic algorithm construct:**
  - each feature is assigned cumulative weightage computed over a predefined number of sample data set selected from the training data set.
  - feature with weightage over a certain threshold is the selected feature subset.

• **Assignment of weightage:**
  - instances belongs to similar class should stay closer together than those in a different class.
  - near-hit instance = similar class.
  - near-miss instance = different class.
  - \( W = W - \text{diff}(X, \text{nearhit})^2 + \text{diff}(X, \text{nearmiss})^2 \)
Feature Selection for Classification

1. selected_subset = {}

2. init. all feature weightage = 0  (eg. for 2 features : w₁=0, w₂=0)

3. for i = 1 to no_of_sample
   - get one instance X from the training data set D.
   - get nearhit  H = instance in D where dist(X,H) is closest & X.class=H.class
   - get nearmiss  M = instance in D where dist(X,M) is closest & X.class<>M.class
   - update weightage for all features :
     - weightage  = weightage  -diff(x,h)² +diff(x,m)²
     - eg. weightage₁ = weightage₁ -diff(x₁,h₁)² +diff(x₁,m₁)²
     - eg. weightage₂ = weightage₂ -diff(x₂,h₂)² +diff(x₂,m₂)²

4. for j = 1 to no_of_feature (eg. 2)
   - if weightage₁ >= Threshold, add feature₁ to selected_subset
Feature Selection for Classification

Classification: Male / Female

- **Hair Length (cm)**
- **Shoe Size**

\[
\begin{align*}
\text{feature} & \quad x & \quad w & \quad -(x-\text{hit})^2 & \quad +(x-\text{miss})^2 & \quad =w \\
\text{shoe size} & \quad x_1 & \quad 0 & \quad -(4-5)^2 & \quad +(4-1)^2 & \quad -1+9 \\
\text{hair length} & \quad x_1 & \quad 0 & \quad -(2-1)^2 & \quad +(2-3)^2 & \quad -1+1 \\
\text{shoe size} & \quad x_2 & \quad 8 & \quad -(2-1)^2 & \quad +(2-5)^2 & \quad +16 \\
\text{hair length} & \quad x_2 & \quad 0 & \quad -(5-5)^2 & \quad +(5-4)^2 & \quad +1 \\
\end{align*}
\]

* if (threshold=5), the feature “shoe size” will be selected.
Feature Selection for Classification

- \[ W = W - \text{diff}(X,\text{nearhit})^2 - \text{diff}(X,\text{nearmiss})^2 \]
  - try to decrease weightage for instances belong to the same class
    (*note: their dist. diff. should be small).
  - try to increase weightage for instances belong to diff class
    (*note: their dist. diff. should be large).
  - If \( W \leq 0 \), then sign of irrelevancy or redundancy.
  - If \( W > 0 \), then instances in diff. class is further apart as expected.

- Disadvantages:
  - applicable only to binary class problem.
  - insufficient training instances fool relief.
  - if most features are relevant, relief select all (even if not necessary).

- Advantages:
  - noise-tolerant.
  - unaffected by feature interaction
    (weightage is cumulative & det. collectively).
Feature Selection for Classification

(11.2) Branch & Bound. [generation=complete, evaluation=distance]

• is a very old method (1977).

• Modified assumption:
  - find a minimally size feature subset.
  - a bound/threshold is used to prune irrelevant branches.

• \( F(\text{subset}) < \text{bound} \), remove from search tree (including all subsets).

• Model of feature set search tree.
Feature Selection for Classification

\[ F = \{ f_1, f_2, f_3 \} \]
Category IV - Generation Heuristic / Evaluation Information

2 Methods:

• 1) Decision Tree Method (DTM)
  – Run C4.5 over training set.
  – The features that are selected are the union of all features in the pruned decision tree produced by C4.5.
  – An information based function selects the feature at each node of the decision tree
Category IV - Generation Heuristic / Evaluation Information

DTM Algorithm. Parameters ($D$)

1. $T = \emptyset$
2. Apply C4.5 to training set, $D$
3. Append all features appearing in the pruned decision tree to $T$
4. Return $T$

$D =$ Training Set
Category IV - Generation Heuristic / Evaluation Information

C4.5

• Uses Information based Heuristic for node selection.

$$ I(p, n) = - \left( \frac{p}{p+n} \right) \log_2 \left( \frac{p}{p+n} \right) - \left( \frac{n}{p+n} \right) \log_2 \left( \frac{n}{p+n} \right) $$

  - p = # of instances of class label 1
  - n = # of instances of class label 0

• Entropy - “a measure of the loss of information in a transmitted signal or message”.

$$ E(F_i) = \left( \frac{p_0 + n_0}{p+n} \right) I(p_0, n_0) + \left( \frac{p_1 + n_1}{p+n} \right) I(p_1, n_1) $$

  - p_i = # of instances with feature value = x, class value = 1 (positive)
  - n_i = # of instances with feature value = x, class value = 0 (negative)

• $$ E(C) = \frac{6+2}{16} I(6,2) + \frac{1+7}{16} I(1,7) = 0.677421 $$
Category IV - Generation Heuristic / Evaluation Information

- Feature to be selected as root of decision tree has minimum entropy.
- Root node partitions, based on the values of the selected feature, instances into two nodes.
- For each of the two sub-nodes, apply the formula to compute entropy for remaining features. Select the one with minimum entropy as node feature.
- Stop when each partition contains instances of a single class or until the test offers no further improvement.
- C4.5 returns a pruned-tree that avoids over-fitting.

\[ \therefore \text{The union of all features in the pruned decision tree is returned as } T. \]
Hand-run of CorrAL Dataset:

- Computation of Entropy across all features for selecting root of the decision tree:

<table>
<thead>
<tr>
<th>Feature - $F$</th>
<th>$E(F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.850603</td>
</tr>
<tr>
<td>B1</td>
<td>0.882856</td>
</tr>
<tr>
<td>B0</td>
<td>0.882856</td>
</tr>
<tr>
<td>A1</td>
<td>0.882856</td>
</tr>
<tr>
<td>A0</td>
<td>0.882856</td>
</tr>
<tr>
<td>C</td>
<td>0.677421</td>
</tr>
</tbody>
</table>

\( \therefore \) C is selected as the root because it has the lowest entropy.
Category IV - Generation Heuristic / Evaluation Information

C (E=0.677421)

B0 (E=0.737517)

Null

Instances:
4,7,8,10,12,13,15,16
Features:
A0,A1,B0,B1,I

Instances:
1,2,3,5,6,9,11,14
Features:
A0,A1,B0,B1,I

DTM returns \{ A_0, A_1, B_0, B_1, C \}
2) Koller and Sahami’s method
   - Intuition:
     • Eliminate any feature that does not contribute any additional information to the rest of the features.
   - Implementation attempts to approximate a Markov Blanket.
   - However, it is suboptimal due to naïve approximations.
1 Method:

- Minimum Description Length Method (MDLM)
  - Eliminate useless (irrelevant and/or redundant) features
  - 2 Subsets: \( U \) and \( V \), \( U \cap V = \emptyset \), \( U \cup V = S \)

\[ \forall v, \ v \in V, \text{ if } F(u) = v, \ u \in U \text{ where } F \text{ is a fixed non-class dependent function, then features in } V \text{ becomes useless when is } U \text{ becomes known.} \]

- \( F \) is formulated as an expression that relates:
  - the \# of bits required to transmit the classes of the instances
  - the optimal parameters
  - the useful features
  - the useless features

- Task is to determine \( U \) and \( V \).
Uses Minimum Description Length Criterion (MDLC)
- MDL is a mathematical model for Occam’s Razor.
- Occam’s Razor - principle of preferring simple models over complex models.

MDLM searches all possible subsets: $2^N$

Outputs the subset satisfying MDLC

MDLM finds useful features only if the observations (the instances) are Gaussian
Category V - Generation Complete / Evaluation Information

MDLM Algorithm. Parameters (D):

1. Set $MDL = \infty$

2. For all feature subsets $L$:

   1.1 Compute $Length_L = \sum_{i=1}^{i=q} \frac{P_i}{2} \log \frac{|D_L(i)|}{|D_L|} + h_L$
   
   where $h_L = \frac{1}{2} (N - M)(N + M + 3) \log P + \sum_{i=1}^{i=q} M(M + 3) \log P_i$,
   
   $N$ – total number of features,
   $M$ – number of features in the candidate subset,
   $P$ – total number of instances in $D$,
   $P_i$ – number of instances with class label $i$,
   $q$ – total number of class labels,
   $D_L$ – covariance matrix formed from all the useful feature vectors,
   $D_L(i)$ – covariance matrix formed from the useful feature vectors of class $i$,
   $\log$ – denotes determinant.

   $T = L, MDL = Length_L$

3. Return $T$  
   $D$ = Training Set
Category V - Generation Complete / Evaluation Information

• Suggested implementation
  – For all feature subsets:
    • 1. Calculate the covariance matrices of the whole feature vectors for all classes: $D_L$
    • 2. Calculate the covariance matrices of the whole feature vectors for each separate class: $D_L(i)$
    • 3. Obtain the covariance matrix for useful subsets as sub-matrixes of $D_L$ and $D_L(i)$
    • 4. Compute the determinants of the sub-matrices $D_L$ and $D_L(i)$
    • 5. Compute $Length_L$ given 1, 2, 3, 4 as in step 2 of the algorithm

• Return subset that has the minimum description length.
• Hand-run of CorrAL dataset returns \{C\} with minimum description length of 119.582.
Category VII - Generation Heuristic/Evaluation Dependence

2 methods

• 1) POE + ACC (Probability of Error and Average Correlation Coefficient)
  – First feature selected is feature with smallest probability of error ($P_e$).
  – The next feature selected is feature that produces minimum weighted sum of $P_e$ and average correlation coefficient $ACC$.
  – $ACC$ is mean of correlation coefficients of all candidate features with features previously selected at that point.
  – This method can rank all the features based on the weighted sum.
  – Stopping criterion is the required number of features.
  – The required parameters are the number of features and the weights $w_1$ and $w_2$. 
Category VII - Generation Heuristic/Evaluation Dependence

POE + ACC Algorithm Parameters \((M, w_1, w_2)\)

1. \(T = \emptyset\)
2. Find feature with minimum \(P_e\) and append to \(T\)
3. For \(i = 1\) to \(M-1\)
   - Find the next feature with minimum \(w_1(P_e) + w_2(ACC)\)
   - Append it to \(T\)
4. Return \(T\)

\(M = \text{Required number of features}\)
\(w_1 = \text{Weight for POE}\)
\(w_2 = \text{Weight for ACC}\)
Category VII - Generation Heuristic/Evaluation Dependence

- To calculate $P_e$
  - First compute the a priori probability of different classes
  - For each feature, calculate the class-conditional probabilities given the class label.
  - Then for each feature value, find the class label for which the product of a priori class probability and class-conditional probability given the class label is a maximum
  - Finally count the number of mismatches between the actual and predicted class values and select the feature with minimum mismatches

- To calculate ACC:
  - Compute correlation coefficient of the candidate feature $x$, with each feature previous selected. (Correlation coefficient measures the amount of linear association between any 2 random variables) :

$$\text{ACC}(x) = \left( \sum^n \text{Corr}(x,y) \right) / n \quad \text{where} \quad n = \left| T \right|, \ y \in T$$
Hand-run of CorrAL Dataset:

- **A priori class probabilities of $D$:**
  - for class $0 = 9/16$, class $1 = 7/16$

- **For feature $C$: class-conditional probability calculation:**

  \[
  \begin{array}{|c|c|c|}
  \hline
  & class = 0 & class = 1 \\
  \hline
  P ( C=0 ) & 2/9 & 6/7 \\
  P ( C=1 ) & 7/9 & 1/7 \\
  \hline
  \end{array}
  \]

- Calculating product of a priori class probability and class-conditional probability given the class label:

  \[
  \begin{array}{|c|c|c|}
  \hline
  & x = 0 & x = 1 \\
  \hline
  P ( C=0 | Class = x ) & 2/9 \times 9/16 = 0.125 & 6/7 \times 7/16 = 0.375 \\
  P ( C=1 | Class = x ) & 7/9 \times 9/16 = 0.4375 & 1/7 \times 7/16 = 0.0625 \\
  \hline
  \end{array}
  \]

- Thus when $C$ takes value of 0, the prediction is class $= 1$ and when $C$ takes the value of 1, the prediction is class $= 0$. 
Category VII - Generation Heuristic/Evaluation Dependence

• Using this, the number of mismatches between the actual and predicted class values is counted to be 3 (instances 7, 10 and 14)
\[ \therefore P_e \text{ of feature } C = \frac{3}{16} \text{ or } 0.1875. \]

• According to the author, this is the minimum among all the features and is selected as the first feature.

• In the second step, the \( P_e \) and \( ACC \) (of all remaining features \{ \( A_0 \), \( A_1 \), \( B_0 \), \( B_1 \), \( I \) \}) with feature \( C \) are calculated to choose the feature with minimum \( [w_1(P_e) + w_2(ACC)] \)

• Stop when required number of features have been selected.

• For hand-run of CorrAL, subset \{ \( C \), \( A_0 \), \( B_0 \), \( I \) \} is selected.
Category VII - Generation Heuristic/Evaluation Dependence

2) PRESET

- Uses the concept of a rough set
- First find a reduct and remove all features not appearing in the reduct (a reduct of a set P classifies instances equally well as P does)
- Then rank features based on their significance measure (which is based on dependency of attributes)
Category XI - Generation Complete/Evaluation Consistency

3 Methods:

• 1) Focus
  – Implements the Min-Features bias
  – Prefers consistent hypotheses definable over as few features as possible
  – Unable to handle noise but may be modified to allow a certain percentage of inconsistency
Category XI - Generation Complete/Evaluation Consistency

Focus Algorithm. Parameters \((D, S)\)

1. \(T = S\)

2. For \(i = 0\) to \(N-1\)
   
   For each subset \(L\) of size \(i\)
   
   If no inconsistency in the training set \(D\) then
   
   \(T = L\)
   
   return \(T\)

\(D = \) Training Set

\(S = \) Original Feature Set
Category XI - Generation Complete/Evaluation Consistency

• Focus performs breath-first generation of feature subsets:-
  – It first generates subsets of size one, then two, and so on.
  – For each subset generated, check whether there are any inconsistencies.
  – A subset is inconsistent when there are at least two instances in the dataset having equal values for all the features under examination. Eg, for subset \( \{A_0\} \), instances 1 and 4 have the same \( A_0 \) instance value (ie:- 0) but different class labels ( 0 and 1 respectively)
  – Continues until it finds the first subset that is not inconsistent or when the search is complete.
Hand-run of CorrAL Dataset:

- Consistent feature sets are:
  - \{ A_0, A_1, B_0, B_1 \}
  - \{ A_0, A_1, B_0, B_1, I \}
  - \{ A_0, A_1, B_0, B_1, C \}
  - \{ A_0, A_1, B_0, B_1, I, C \}

- However Focus returns the smallest consistent subset that is \{ A_0, A_1, B_0, B_1 \}.

- Trivial implementation of Focus:
  - http://www.comp.nus.edu.sg/~wongszec/cs6203_focus.pl
  - To run, type: perl cs6203_focus.pl
Category XI - Generation Complete/Evaluation Consistency

2) Schlimmer’s Method
   - Variant of Focus: Uses a systematic enumeration scheme as generation procedure and the inconsistent criterion as the evaluation function
   - Uses a heuristic function that makes the search for the optimal subset faster.

3) MIFES_1
   - Also very similar to Focus: Represents the set of instances in the form of a matrix.
CATEGOR Y XII (Consistency – Random)

LVF Algorithm

• Las Vegas Algorithm
• Randomly search the space of instances which makes probabilistic choices more faster to an optimal solution
• For each candidate subsets, LVF calculates an inconsistency count based on the intuition
• An inconsistency threshold is fixed in the beginning (Default = 0)
• Any subsets with inconsistency rate > threshold, REJECT
CATEGOR Y XII (Consistency – Random)

LVF Algorithm

• INPUT MAX-TRIES
  D - Dataset
  N - Number of attributes
  γ - Allowable inconsistency rate

• OUTPUT sets of M features satisfying the inconsistency rate
LVF Algorithm

\[ C_{\text{best}} = N; \]

**FOR** I = 1 to MAX-TRIES

S = randomSet(seed);

C = numOfFeatures(S);

**IF** (C < \( C_{\text{best}} \))

**IF** (InconCheck(S,D) < \( \gamma \));

\( S_{\text{best}} = S; \ C_{\text{best}} = C; \)

print_Current_Best(S)

**ELSE IF** ((C = \( C_{\text{best}} \)) **AND** (InConCheck(S,D) < \( \gamma \)))

print_Current_Best(S)

**END FOR**
CATEGORY XII (Consistency – Random)

LVF Algorithm

ADVANTAGE
• Find optimal subset even for database with Noise
• User does not have to wait too long for a good subset
• Efficient and simple to implement, guarantee to find optimal subset if resources permit

DISADVANTAGE
• It take more time to find the optimal subset (whether the data-set is consistent or not)
FILTER VS WRAPPER

FILTER METHOD
Consider attributes independently from the induction algorithm

• Exploit general characteristics of the training set (statistics: regression tests)

• Filtering (of irrelevant attributes) occurs before the training
FILTER VS WRAPPER

WRAPPER METHOD

- Generate a set of candidate features
- Run the learning method with each of them
- Use the accuracy of the results for evaluation (either training set or a separate validation set)
WRAPPER METHOD

• Evaluation Criteria (Classifier Error Rate)
  ≈ Features are selected using the classifier
  ≈ Use these selected features in predicting the class labels of unseen instances
  ≈ Accuracy is very high

• Use actual target classification algorithm to evaluate accuracy of each candidate subset

• Generation method: heuristics, complete or random

• The feature subset selection algorithm conducts a search for a good subset using the induction algorithm, as part of evaluation function
WRAPPER METHOD

DISADVANTAGE

• Wrapper very slow
• Higher Computation Cost
• Wrapper has danger of overfitting
CATEGOR Y XIII: CER – Heuristics

SFS (Sequential Forward Selection)

- Begins with zero attributes
- Evaluates all features subsets w/ exactly 1 feature
- Selects the one with the best performance
- Adds to this subsets the feature that yields the best performance for subsets of next larger size
- If EVAL() is a heuristics measure, the feature selection algorithm acts as a filter, extracting features to be used by the main algorithm; If it is the actual accuracy, it acts as a wrapper around that algorithm
SFS (Sequential Forward Selection)

SS = 0
BestEval = 0
REPEAT
  BestF = None
  FOR each feature F in FS AND NOT in SS
    SS’ = SS \cup \{F\}
    IF Eval(SS’) > BestEval THEN
      BestF = F; BestEval = Eval(SS’)
    IF BestF <> None THEN SS = SS \cup \{BestF\}
  UNTIL BestF = None OR SS = FS
RETURN SS
SBS (Sequential Backward Selection)

• Begins with all features

• Repeatedly removes a feature whose removal yields the maximal performance improvement
SBS (Sequential Backward Selection)

SS = FS
BestEval = Eval(SS)

REPEAT
    WorstF = None
    FOR each feature in F in FS
    SS’ = SS - {F}
    IF Eval(SS’) >= BestEval THEN
        WorstF = F; BestEval = Eval(SS’)
    IF WorstF <> None THEN SS = SS - {WorstF}
UNTIL WorstF = None OR SS = 0
RETURN SS
ABB Algorithm

• Combat the disadvantage of **B&B** by permitting evaluation functions that are not monotonic.

• The bound is the inconsistency rate of dataset with the full set of features.
ABB Algorithm

- Legitimate test: Determine whether a subset is a child note of a pruned node, by applying Hamming distance.
- `InConCal()` calculates the consistency rate of data given a feature subsets by ensuring:
  - No duplicate subset will be generated
  - No child of pruned node (Hamming distance)
ABB Algorithm

\[ \delta = \text{inConCal}(S, D); \]

**PROCEDURE** ABB(S,D)

FOR all feature \( f \) in S

\( S_1 = S - f \); enQueue(Q, S_1);

END FOR

WHILE notEmpty(Q)

\( S_2 = \text{deQueue}(Q); \)

IF (\( S_2 \) is legitimate \( \land \) inConCal(\( S_2, D \)) \( \leq \) \( \delta \))

ABB(S_2, D);

END WHILE

END
ABB Algorithm

- **ABB** expands the search space quickly but is inefficient in reducing the search space although it guarantees optimal results.

- Simple to implement and guarantees optimal subsets of features.

- **ABB** removes irrelevant, redundant, and/or correlated features even with the presence of noise.

- Performance of a classifier with the features selected by **ABB** also improves.
LVW Algorithm

• Las Vegas Algorithm

• Probabilistic choices of subsets

• Find Optimal Solution, if given sufficient long time

• Apply Induction algorithm to obtain estimated error rate

• It uses randomness to guide their search, in such a way that a correct solution is guaranteed even if unfortunate choices are made
LVW Algorithm

Err = 0; k = 0; C = 100;

REPEAT
  S₁ = randomSet(); C₁ = numOfFeatures(S₁);
  err₁ = LearnAlgo(S₁, D_{train}, NULL);
  IF (err₁ < err) OR (err₁ = err AND C₁ < C))
    output the current best;
    k = 0; err = err₁; C = C₁; S = S₁;
  END IF
  k = k + 1;
UNTIL err is not updated for K times;
err₂ = LearnAlgo(S, D_{train}, D_{test});
CATEGORİY XIII: CER – Random

LVW Algorithm

- LVW can reduce the number of features and improve the accuracy
- Not recommended in applications where time is critical factor
- Slowness is caused by learning algorithm
EMPIRICAL COMPARISON

• Test Datasets
  ≈ Artificial
  ≈ Consists of **Relevant** and **Irrelevant** Features
  ≈ Know beforehand which features are relevant and which are not

• Procedure
  ≈ Compare Generated subset with the known relevant features
## CHARACTERISTIC OF TEST DATASETS

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>CORRAL</th>
<th>PAR3+3</th>
<th>MONK3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Irrelevant</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Correlated</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Redundant</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Noisy</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
**RESULTS**

- Different methods work well under different conditions
  - **RELIEF** can handle noise, but not redundant or correlated features
  - **FOCUS** can detect redundant features, but not when data is noisy

- No single method works under all conditions

- Finding a good feature subset is an important problem for real datasets. A good subset can
  - Simplify data description
  - Reduce the task of data collection
  - Improve accuracy and performance
RESULTS

- Handle Discrete? Continuos? Nominal?
- Multiple Class size?
- Large Data size?
- Handle Noise?
- If data is not noisy, able to produce optimal subset?
Feature Selection for Classification

Some Guidelines in picking the “right” method?
Based on the following 5 areas. (i.e. mainly related to the characteristic of data set on hand).

• Data types - continuous, discrete, nominal
• Data size - large data set?
• Classes - ability to handle multiple classes (non binary)?
• Noise - ability to handle noisy data?
• Optimal subset - produce optimal subset if data not noisy?
# Feature Selection for Classification

## Comparison table of the discussed method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Generation</th>
<th>Evaluation</th>
<th>Contin.</th>
<th>Discrete</th>
<th>Nominal</th>
<th>Large Dataset</th>
<th>Multiple Classes</th>
<th>Handle Noise</th>
<th>Optimal Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>B &amp; B</td>
<td>complete</td>
<td>distance</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>-</td>
<td>y</td>
<td>-</td>
<td>y++</td>
</tr>
<tr>
<td>MDLM</td>
<td>complete</td>
<td>information</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>-</td>
<td>y</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>Focus</td>
<td>complete</td>
<td>consistency</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y++</td>
</tr>
<tr>
<td>Relief</td>
<td>heuristic</td>
<td>distance</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>DTM</td>
<td>heuristic</td>
<td>information</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>POE+ACC</td>
<td>heuristic</td>
<td>dependency</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>-</td>
<td>y</td>
<td>-</td>
<td>n</td>
</tr>
<tr>
<td>LVF</td>
<td>random</td>
<td>consistency</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y*</td>
<td>y**</td>
</tr>
</tbody>
</table>

- method does not discuss about the particular characteristic.
- y++ if certain assumptions are valid.
- y* user is required to provide the noise level.
  y** provided there are enough resources.

*note: "classifier error rate" not included (ie. Depend on specify classifier).