Lecture 3: Dimensionality reduction – feature ranking and selection

Pavel Kordík, FIT, Czech Technical University in Prague
Feature ranking and selection

- Many algorithms available
- Select most informative variables (features)
- Main application fields
  - Reduce dimensionality of data set
  - Locate optimal subset of features
  - Rank features according to their significance (extremely popular data mining outcome)
Feature Selection

- **Thousands to millions of low level features**: select the most relevant one to build **better, faster, and easier to understand** learning machines.
Feature Selection (FS)

- There are $2^d$ possible sub-features of $d$ features – show why!

- Amaldi and Kann (1998) showed that the minimization problem related to feature selection for linear systems is NP hard.
FS Taxonomy

- **Univariate method**: considers one variable (feature) at a time.
- **Multivariate method**: considers subsets of variables (features) together.

---

- **Filter method**: ranks features or feature subsets independently of the predictor (classifier).
- **Wrapper method**: uses a classifier to assess features or feature subsets.
- **Embedded method**: like wrapper, the search is controlled by the algorithm constructing classifier
Filters vs. Wrappers

- **Main goal:** rank subsets of useful features.

- Danger of over-fitting with intensive search!
Feature subset assessment

Split data into 3 sets: training, validation, and test set.

1) For each feature subset, train predictor on training data.

2) Select the feature subset, which performs best on validation data.

3) Test on test data.
Cross validation

1) Train predictor on all samples except one fold.
2) Select the feature subset, which performs best on the fold not used for training.
3) Repeat until all folds were used for testing.
4) Average results – how?

Split data into K folds: Randomly without replacement.

N variables/features

M samples

$m_1$

$m_2$

$m_3$

$m_4$

$m_5$

$m_6$

$m_7$
Embedded methods

All features → Train classifier → Eliminate useless feature(s) → Performance degradation?

Yes, stop! → No, continue…
Filters, Wrappers, and Embedded methods

All features → Filter → Feature subset → Predictor

All features → Wrapper → Multiple Feature subsets → Predictor

All features → Embedded method → Feature subset → Predictor
Filters

Methods:

- **Criterion:** Measure feature/feature subset “relevance”
- **Search:** Usually order features (individual feature ranking or nested subsets of features)
- **Assessment:** Use statistical tests

Results:

- Are (relatively) robust against overfitting
- May fail to select the most “useful” features
Wrappers

Methods:

- **Criterion:** Measure feature subset “usefulness”
- **Search:** Search the space of all feature subsets
- **Assessment:** Use cross-validation

Results:

- Can in principle find the most “useful” features, but
- Are prone to overfitting
Embedded Methods

Methods:

- **Criterion:** Measure feature subset “usefulness”
- **Search:** Search guided by the learning process
- **Assessment:** Use cross-validation

Results:

- Similar to wrappers, but
- Less computationally expensive
- Less prone to overfitting
Three “Ingredients”

- Statistical tests
- Single feature ranking
- Cross validation
- Performance bounds
- Heuristic or stochastic search
- Exhaustive search
- Feature subset relevance
- Relevance in context
- Nested subset, forward selection/backward elimination
- Single feature ranking
- Performance learning machine

Search

Assessment

Criterion

Embedded
Univariate FS methods

- How relevant is the variable $X_i$ for predicting the output variable $Y$?
  - Relevance
  - Tests
  - Dependence
  - Correlation
  - Pearson coefficient
  - Mutual Information
  - Filters, wrappers, embedded
  - ...
Irrelevant Feature $X_i$

$$P(X_i, Y) = P(X_i) \cdot P(Y)$$

$$P(X_i | Y = 1) = P(X_i | Y = -1)$$

$Y$ - output

density

$X_i$

$Y = 1$

$Y = -1$
Relevant Feature

\[ \mu^{-}, \mu^{+} \]

+ = positive output
- = negative output

\[ X_i \]

\[ \sigma^{-}, \sigma^{+} \]
T-test

• Normally distributed classes, equal variance $\sigma^2$ unknown; estimated from data as $\sigma^2_{\text{within}}$.

• Null hypothesis $H_0$: $\mu_+ = \mu_-$

• $T$ statistic: If $H_0$ is true,

$$t = (\mu_+ - \mu_-)/\left(\sigma_{\text{within}}\sqrt{1/m^+ + 1/m^-}\right) \sim \text{Student}(m^+ + m^- - 2 \text{ d.f.})$$
Univariate Dependence

- Independence:
  \[ P(X, Y) = P(X) P(Y) \]

- Measure of dependence:
  Mutual Information (MI)

\[ MI(X, Y) = \int P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)} \, dX \, dY \]
Empirical Mutual Information
a common way to use MI in practice

- Data \((n) \Rightarrow\) contingency table

\[
\begin{align*}
n_{ij} &= \text{# of times } (i,j) \text{ occurred} \\
n_{ij} &= \sum_j n_{ij} = \text{# of times } i \text{ occurred} \\
n_{ij} &= \sum_i n_{ij} = \text{# of times } j \text{ occurred} \\
n &= \sum_{ij} n_{ij} = \text{dataset size}
\end{align*}
\]

<table>
<thead>
<tr>
<th>(j) (\backslash i)</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>(r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(n_{11})</td>
<td>(n_{12})</td>
<td>...</td>
<td>(n_{1r})</td>
</tr>
<tr>
<td>2</td>
<td>(n_{21})</td>
<td>(n_{22})</td>
<td>...</td>
<td>(n_{2r})</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>(s)</td>
<td>(n_{s1})</td>
<td>(n_{s2})</td>
<td>...</td>
<td>(n_{sr})</td>
</tr>
</tbody>
</table>

- Empirical (sample) probability: \(\hat{p}_{ij} = \frac{n_{ij}}{n}\)
- Empirical mutual information: \(I(\hat{\pi})\)
Mutual Information (MI)

- Consider two discrete random variables \((\iota, \gamma)\)
  
  \[ \pi_{ij} = \text{joint chance of } (i, j), \quad i \in \{1, \ldots, r\} \text{ and } j \in \{1, \ldots, s\} \]
  
  \[ \pi_{i+} = \sum_j \pi_{ij} = \text{marginal chance of } i \]
  
  \[ \pi_{+j} = \sum_i \pi_{ij} = \text{marginal chance of } j \]

- (In)Dependence measured by MI

\[
0 \leq I(\pi) = \sum_{ij} \pi_{ij} \log \frac{\pi_{ij}}{\pi_{i+}\pi_{+j}}
\]

- Also known as \textit{cross-entropy} or \textit{information gain}

- Examples
  
  - Inference of Bayesian nets, classification trees
  
  - Selection of relevant variables for the task at hand
Distance of two features

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
<th>X</th>
<th>Y</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian accuracy</td>
<td>Eq. 3.1</td>
<td>+ s</td>
<td>+ s</td>
<td>Theoretically the golden standard, rescaled Bayesian relevance Eq. 3.2.</td>
</tr>
<tr>
<td>Balanced accuracy</td>
<td>Eq. 3.4</td>
<td>+ s</td>
<td>+ s</td>
<td>Average of sensitivity and specificity; used for unbalanced dataset,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>same as AUC for binary targets.</td>
</tr>
<tr>
<td>Bi-normal separation</td>
<td>Eq. 3.5</td>
<td>+ s</td>
<td>+ s</td>
<td>Used in information retrieval.</td>
</tr>
<tr>
<td>F-measure</td>
<td>Eq. 3.7</td>
<td>+ s</td>
<td>+ s</td>
<td>Harmonic of recall and precision, popular in information retrieval.</td>
</tr>
<tr>
<td>Odds ratio</td>
<td>Eq. 3.6</td>
<td>+ s</td>
<td>+ s</td>
<td>Popular in information retrieval.</td>
</tr>
<tr>
<td>Means separation</td>
<td>Eq. 3.10</td>
<td>+ i</td>
<td>+ +</td>
<td>Based on two class means, related to Fisher’s criterion.</td>
</tr>
<tr>
<td>T-statistics</td>
<td>Eq. 3.11</td>
<td>+ i</td>
<td>+ +</td>
<td>Based also on the means separation.</td>
</tr>
<tr>
<td>Pearson correlation</td>
<td>Eq. 3.9</td>
<td>+ i</td>
<td>+ + i</td>
<td>Linear correlation, significance test Eq. 3.12, or a permutation test.</td>
</tr>
<tr>
<td>Group correlation</td>
<td>Eq. 3.13</td>
<td>+ i</td>
<td>+ + i</td>
<td>Pearson’s coefficient for subset of features.</td>
</tr>
<tr>
<td>$\chi^2$</td>
<td>Eq. 3.8</td>
<td>+ s</td>
<td>+ s</td>
<td>Results depend on the number of samples $m$.</td>
</tr>
<tr>
<td>Relief</td>
<td>Eq. 3.15</td>
<td>+ s</td>
<td>+ s</td>
<td>Family of methods, the formula is for a simplified version ReliefX,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>captures local correlations and feature interactions.</td>
</tr>
<tr>
<td>Separability Split Value</td>
<td>Eq. 3.41</td>
<td>+ s</td>
<td>+ + s</td>
<td>Decision tree index.</td>
</tr>
<tr>
<td>Kolmogorov distance</td>
<td>Eq. 3.16</td>
<td>+ s</td>
<td>+ + s</td>
<td>Difference between joint and product probabilities.</td>
</tr>
<tr>
<td>Bayesian measure</td>
<td>Eq. 3.16</td>
<td>+ s</td>
<td>+ + s</td>
<td>Same as Vajda entropy Eq. 3.23 and Gini Eq. 3.39.</td>
</tr>
<tr>
<td>Kullback-Leibler divergence</td>
<td>Eq. 3.20</td>
<td>+ s</td>
<td>+ + s</td>
<td>Equivalent to mutual information.</td>
</tr>
<tr>
<td>Jeffrey’s-Matusita distance</td>
<td>Eq. 3.22</td>
<td>+ s</td>
<td>+ s</td>
<td>Rarely used but worth trying.</td>
</tr>
<tr>
<td>Value Difference Metric</td>
<td>Eq. 3.22</td>
<td>+ s</td>
<td>+ s</td>
<td>Used for symbolic data in similarity-based methods,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and symbolic feature-feature correlations.</td>
</tr>
<tr>
<td>Mutual Information</td>
<td>Eq. 3.29</td>
<td>+ s</td>
<td>+ + s</td>
<td>Equivalent to information gain Eq. 3.30.</td>
</tr>
<tr>
<td>Information Gain Ratio</td>
<td>Eq. 3.32</td>
<td>+ s</td>
<td>+ + s</td>
<td>Information gain divided by feature entropy, stable evaluation.</td>
</tr>
<tr>
<td>Symmetrical Uncertainty</td>
<td>Eq. 3.35</td>
<td>+ s</td>
<td>+ + s</td>
<td>Low bias for multivalued features.</td>
</tr>
<tr>
<td>J-measure</td>
<td>Eq. 3.36</td>
<td>+ s</td>
<td>+ + s</td>
<td>Measures information provided by a logical rule.</td>
</tr>
<tr>
<td>Weight of evidence</td>
<td>Eq. 3.37</td>
<td>+ s</td>
<td>+ + s</td>
<td>So far rarely used.</td>
</tr>
<tr>
<td>MDL</td>
<td>Eq. 3.38</td>
<td>+ s</td>
<td>+ s</td>
<td>Low bias for multivalued features.</td>
</tr>
</tbody>
</table>
Selection criteria

- **distance** (euclidean distance measure).
- **information** (entropy, information gain, etc.)
- **dependency** (correlation coefficient).
- **consistency** (min-features bias).
- **classifier error rate** (the classifier themselves).
**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.

**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).
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
Consistency measure

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

<table>
<thead>
<tr>
<th></th>
<th>$f_1$</th>
<th>$f_2$</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.
Selection criteria, cont.

**Classifier error rate**

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

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

- computationally very costly!
Steps of the feature selection procedure

Original feature set → Generation → Evaluation → Stopping criterion → Validation

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

Complete/exhaustive
- examine all combinations of feature subset.
  \{f_1,f_2,f_3\} => \{ \{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.
- Problems with stochastic filters, wrappers, embedded

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\}.
FS methods, cont.

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

Evolutionary
- evolutionary algorithms used to find a subset of features
- fitness function can use mentioned selection criteria
Heuristic FS methods

- Best single features under the feature independence assumption: choose by significance tests.

- Step-wise feature selection:
  - The best single-feature is picked first
  - Then next best feature condition to the first
  - ... Beam search, GSFS, PTA(l,r), Floating search

- Step-wise feature elimination:
  - Repeatedly eliminate the worst feature

- Combined feature selection and elimination:
  - Optimal branch and bound:
  - Use feature elimination and backtracking
Problem

- Real world features are seldom independent!
- How can I verify independence?
  - In case of nominal variables?
  - For continuous (ordinal) variables?
- Solution:
  - Select set of most informative features that are mutually independent.
  - ... or at least as independent as possible
Maximum Significant Difference and Independence (MSDI) Algorithm

1. Compute the significance difference ($sd$) of every initial feature.
2. Select the feature with maximum $sd$ as the first feature.
3. Compute the independence level ($ind$) between every candidate feature and the already-selected feature(s).
4. Select the feature with maximum feature significance ($sf = sd \times ind$) as the new feature.
Heuristic – test and dependence

Criterion of Feature Selection

Significance of feature = Significant difference \times Independence

- Pattern separability on individual candidate features
- Noncorrelation between candidate feature and already-selected features
Measurement of Pattern Separability of Individual Features

Statistical Significant Difference

- **Continuous data with normal distribution**
  - Two classes: **t-test**
  - More than two classes: **ANOVA**

- **Continuous data with non-normal distribution or rank data**
  - Two classes: **Mann-Whitney test**
  - More than two classes: **Kruskal-Wallis test**

- **Categorical data**
  - Chi-square test
Independence

Continuous data with normal distribution
- Pearson correlation

Continuous data with non-normal distribution or rank data
- Spearman rank correlation

Categorical data
- Pearson contingency coefficient

\[ \text{independence} = \sqrt{1 - \text{correlation}^2} \]
Algorithms

**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 belong to similar class should stay closer together than those in a different class.
  - near-hit instance = similar class.
  - near-miss instance = different class.
  - \( W = W - \text{diff}(X,\text{nearhit})^2 + \text{diff}(X,\text{nearmiss})^2 \)
Relief algorithm pseudocode

1. selected_subset = {}

2. init. all feature weightage = 0 (eg. for 2 features : \( w_1=0, w_2=0 \))

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 & \( X.\text{class}=H.\text{class} \)
   
   get nearmiss \( M = \) instance in \( D \) where dist\((X,M)\) is closest & \( X.\text{class}<>M.\text{class} \)

   update weightage for all features :
   
   - weightage = weightage -diff\((x,h)\)^2 +diff\((x,m)\)^2
     
     eg. weightage\(_1\) = weightage\(_1\) -diff\((x_1,h_1)\)^2 +diff\((x_1,m_1)\)^2
     
     eg. weightage\(_2\) = weightage\(_2\) -diff\((x_2,h_2)\)^2 +diff\((x_2,m_2)\)^2

4. for \( j = 1 \) to no_of_feature (eg. 2)
   
   if weightage\(_j\) >= Threshold, add feature\(_j\) to selected_subset
Releaf

- \( W = W - \text{diff}(X, \text{nearhit})^2 - \text{diff}(X, \text{nearmiss})^2 \)
  - 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).
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.
- $F(subset) < \text{bound}$, remove from search tree (including all subsets).
- Model of feature set search tree.
Heuristic – information - DTD

- 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

Initial attribute set: \{A1, A2, A3, A4, A5, A6\}

Reduced attribute set: \{A1, A4, A6\}
Decision Trees

- Tree classifiers, like CART (Breiman, 1984) or C4.5 (Quinlan, 1993)

At each step, choose the feature that "reduces entropy" most. Work towards "node purity".
DTD pseudocode

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 $T$

$D = \text{Training Set}$
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.

\[ \therefore \text{The union of all features in the pruned decision tree is returned as } T. \]
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.
Minimum Description Length Method (MDLM)

- Eliminate useless (irrelevant and/or redundant) features
- 2 Subsets: $U$ and $V$, $U \cap V = \emptyset$, $U \cup V = S$

- $\forall v, v \in V$, if $F(u) = v, u \in U$ where $F$ is a fixed non-class dependent function, then features in $V$ becomes useless when $U$ 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 $V$. 
MDLM, cont.

- 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
MDLM pseudocode

MDLM Algorithm. Parameters \((D)\):

1. Set \(MDL = \infty\)

2. For all feature subsets \(L\):

   1.1 Compute \(Length_L = \sum_{i=1}^{q} \frac{P_i}{2} \log \frac{|D_L(i)|}{|D_L|} + h_L\)

      where \(h_L = \frac{1}{2} (N - M)(N + M + 3) \log P + \sum_{i=1}^{q} M(M + 3) \log P_i\),
      \(N\) – 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 \(i\),
      \(|.|\) – denotes determinant.

   If \(Length_L < MDL\) then
   \[ T = L, \quad MDL = Length_L \]

3. Return \(T\)

\(D =\) Training Set
**MDLM implementation**

- **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.
Heuristic – dependence – POE&ACC

- 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 $w_2$. 
POE & ACC pseudocode

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 $T$

$M = \text{Required number of features}$
$w_1 = \text{Weight for POE}$
$w_2 = \text{Weight for ACC}$
POE&ACC implementation

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):

$$\text{ACC}(x) = \frac{\left( \sum^n \text{Corr}(x,y) \right)}{n} \quad \text{where} \quad n = |T|, \quad y \in T$$
Heuristic – dependence – PRESET

- 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)
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
FOCUS pseudocode

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
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.

Trivial implementation of Focus:
Heuristic – consistency

1) FOCUS

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.
RANDOM – consistency – LVF

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
LVF inputs, outputs

LVF Algorithm

- **INPUT**  
  MAX-TRIES  
  D - Dataset  
  N - Number of attributes  
  \( \gamma \) - Allowable inconsistency rate

- **OUTPUT**  
  sets of M features satisfying the inconsistency rate
LVF pseudocode

LVF Algorithm

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

\textbf{FOR} \quad I = 1 \text{ to } \text{MAX-TRIES} \quad \textbf{DO} \quad \text{END FOR}

\[ S = \text{randomSet(seed)}; \]
\[ C = \text{numOfFeatures}(S); \]
\[ \text{IF} \ (C < C_{\text{best}}) \quad \text{THEN} \]
\[ \quad \text{IF} \ (\text{InconCheck}(S,D) < \gamma); \]
\[ \quad \quad S_{\text{best}} = S; \quad C_{\text{best}} = C; \]
\[ \quad \quad \text{print}_\text{Current}_\text{Best}(S) \]
\[ \quad \text{ELSE IF} \ ((C = C_{\text{best}}) \ \text{AND} \ (\text{InconCheck}(S,D) < \gamma)) \]
\[ \quad \quad \text{print}_\text{Current}_\text{Best}(S) \]
\[ \text{END FOR} \]
LVF evaluation

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)
Wrappers

- 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
Heuristic – wrapper (filter)

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 \texttt{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 pseudocode

SFS (Sequential Forward Selection)

SS = 0
BestEval = 0
REPEAT
  BestF = None
  FOR each feature F in FS AND NOT in SS
    SS' = SS ∪ {F}
    IF Eval(SS') > BestEval THEN
      BestF = F; BestEval = Eval(SS')
    IF BestF <> None THEN SS = SS ∪ {BestF}
  UNTIL BestF = None OR SS = FS
RETURN SS
Forward Selection with GS


- Select a first feature $X_{\nu(1)}$ with maximum cosine with the target $\cos(x_i, y) = \frac{x.y}{||x|| ||y||}$

- For each remaining feature $X_i$
  - Project $X_i$ and the target $Y$ on the null space of the features already selected
  - Compute the cosine of $X_i$ with the target in the projection

- Select the feature $X_{\nu(k)}$ with maximum cosine with the target in the projection.

Embedded method for the linear least square predictor
SBS

SBS (Sequential Backward Selection)

• Begins with all features

• Repeatedly removes a feature whose removal yields the maximal performance improvement
SBS pseudocode

**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
Start with all the features.

- Train a learning machine $f$ on the current subset of features by minimizing a risk functional $J[f]$.
- For each (remaining) feature $X_i$, estimate, without retraining $f$, the change in $J[f]$ resulting from the removal of $X_i$.
- Remove the feature $X_{n(k)}$ that results in improving or least degrading $J$.

Embedded method for SVM, kernel methods, neural nets.
Heuristic – inconsistency - ABB

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 description

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 pseoudocode

ABB Algorithm

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

**PROCEDURE** ABB(S,D)

**FOR** all feature \( f \) in S

\[ S_1 = S - f \; ; \text{enQueue}(Q_1, S_1); \]

**END FOR**

**WHILE** notEmpty(Q)

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

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

\[ \text{ABB}(S_2, D); \]

**END WHILE**

**END**
ABB, cont.

**ABB Algorithm**

- **ABB** expands the search space quickly but is inefficient in reducing the search space although it guarantee 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
Random - LVW

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 pseudocode

LVW Algorithm

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

REPEAT
    S<sub>1</sub> = randomSet(); C<sub>1</sub> = numOfFeatures(S<sub>1</sub>);
    err<sub>1</sub> = LearnAlgo(S<sub>1</sub>, D<sub>train</sub>, NULL);
    IF (err<sub>1</sub> < err) OR (err<sub>1</sub> = err AND C<sub>1</sub> < C))
        output the current best;
        k = 0; err = err<sub>1</sub>; C = C<sub>1</sub>; S = S<sub>1</sub>;
    END IF
    k = k + 1;
UNTIL err is not updated for K times;
err<sub>2</sub> = LearnAlgo(S, D<sub>train</sub>, D<sub>test</sub>);
Algorithms 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
## Datasets

<table>
<thead>
<tr>
<th></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
## 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>y</td>
<td>-</td>
<td>y++</td>
<td></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><strong>DTM</strong></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).
In practice...

- No method is universally better:
  - wide variety of types of variables, data distributions, learning machines, and objectives.

- Match the method complexity to the ratio M/N:
  - univariate feature selection may work better than multivariate feature selection; non-linear classifiers are not always better.

- Feature selection is not always necessary to achieve good performance.

**NIPS 2003 and WCCI 2006 challenges:** [http://clopinet.com/challenges](http://clopinet.com/challenges)
What you should remember?

- Taxonomy, search, distances
- Algorithms
  - RelieF
  - B&B
  - DTD
  - SFS
- In WEKA
  - CFS (correlation feature selection – **heuristic** – dependence in our terminology)
  - MIFS (mutual information feature selection – **heuristic** – information in our terminology)
Book of the NIPS 2003 challenge

Feature Extraction, Foundations and Applications
I. Guyon et al, Eds.
http://clopinet.com/fextract-book