Sequential Feature Selection

Feature extraction vs. feature selection

Search strategy and objective functions

Objective functions
- Filters
- Wrappers

Sequential search strategies
- Sequential forward selection
- Sequential backward selection
- Plus-l minus-r selection
- Bidirectional search
- Floating search
Feature extraction vs. feature selection

There are two general approaches to dim. reduction

– **Feature extraction**: Transform the existing features into a lower dimensional space
– **Feature selection**: Select a subset of the existing features without a transformation

\[
\begin{bmatrix}
    x_1 \\ x_2 \\ \vdots \\ x_N
\end{bmatrix} \rightarrow
\begin{bmatrix}
    x_{i_1} \\ x_{i_2} \\ \vdots \\ x_{i_M}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x_1 \\ x_2 \\ \vdots \\ x_N
\end{bmatrix} \rightarrow
\begin{bmatrix}
    y_1 \\ y_2 \\ \vdots \\ y_M
\end{bmatrix} = f
\begin{bmatrix}
    x_1 \\ x_2 \\ \vdots \\ x_N
\end{bmatrix}
\]

Feature extraction was covered in previous lectures

– We derived the “optimal” linear features for two objective functions
  - Signal representation: PCA
  - Signal classification: LDA

Feature selection, also called feature subset selection (FSS) in the literature, will be the subject of the last two lectures

– Although FSS can be thought of as a special case of feature extraction (think of a sparse projection matrix with a few ones), in practice it is a quite different problem
– FSS looks at the issue of dimensionality reduction from a different perspective
– FSS has a unique set of methodologies
Feature subset selection

Definition

Given a feature set $X = \{x_i \mid i = 1 \ldots N\}$, find a subset $Y_M$, with $M < N$, that maximizes an objective function $J(Y)$, ideally $P(\text{correct})$

$$Y_M = \{x_{i1}, x_{i2}, \ldots, x_{iM}\} = \arg \max_{M, i_M} J\{x_i \mid i = 1..N\}$$

Why feature subset selection?

Why not use the more general feature extraction methods, and simply project a high-dimensional feature vector onto a low-dimensional space?

Feature subset selection is necessary in a number of situations

Features may be expensive to obtain

- You evaluate a large number of features (sensors) in the test bed and select only a few for the final implementation

You may want to extract meaningful rules from your classifier

- When you project, the measurement units of your features (length, weight, etc.) are lost

Features may not be numeric, a typical situation in machine learning

In addition, fewer features means fewer model parameters

- Improved the generalization capabilities
- Reduced complexity and run-time
Search strategy and objective function

FSS requires

– A search strategy to select candidate subsets
– An objective function to evaluate these candidates

Search strategy

– Exhaustive evaluation of feature subsets involves $\binom{N}{M}$ combinations for a fixed value of $M$, and $2^N$ combinations if $M$ must be optimized as well
  • This number of combinations is unfeasible, even for moderate values of $M$ and $N$, so a search procedure must be used in practice
  • For example, exhaustive evaluation of 10 out of 20 features involves 184,756 feature subsets; exhaustive evaluation of 10 out of 100 involves more than $10^{13}$ feature subsets [Devijver and Kittler, 1982]

– A search strategy is therefore needed to direct the FSS process as it explores the space of all possible combination of features

Objective function

– The objective function evaluates candidate subsets and returns a measure of their “goodness”, a feedback signal used by the search strategy to select new candidates

<table>
<thead>
<tr>
<th>Feature Subset Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information content</td>
</tr>
<tr>
<td>Search</td>
</tr>
<tr>
<td>Objective function</td>
</tr>
<tr>
<td>PR algorithm</td>
</tr>
</tbody>
</table>

Training data

Complete feature set

Final feature subset
Objective function

Objective functions are divided in two groups

- **Filters**: evaluate subsets by their information content, e.g., interclass distance, statistical dependence or information-theoretic measures
- **Wrappers**: use a classifier to evaluate subsets by their predictive accuracy (on test data) by statistical resampling or cross-validation
Filter types

Distance or separability measures

- These methods measure class separability using metrics such as
  - Distance between classes: Euclidean, Mahalanobis, etc.
  - Determinant of $S_W^{-1}S_B$ (LDA eigenvalues)

Correlation and information-theoretic measures

- These methods are based on the rationale that good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other

- Linear relation measures
  - Linear relationship between variables can be measured using the correlation coefficient

$$J(Y_M) = \frac{\sum_{i=1}^{M} \rho_{ic}}{\sum_{i=1}^{M} \sum_{j=i+1}^{M} \rho_{ij}}$$

- Where $\rho_{ic}$ is the correlation coefficient between feature $i$ and the class label and $\rho_{ij}$ is the correlation coefficient between features $i$ and $j$
Relevance Measures Based on Information Theory

**Mutual information**

- **(Shannon) Entropy:**
  \[ H(X) = - \int_x p(x) \log_2 p(x) \, dx \]

- **Conditional entropy:**
  \[ H(Y|X) = \int_x p(x)(- \int_y p(y|x) \log_2 p(y|x)) \, dx \]

- **Mutual information:**
  \[ MI(Y, X) = H(Y) - H(Y|X) = \int_x \int_y p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \, dxdy \]

**Is MI for classification Bayes optimal?**

- \[ \frac{H(Y|X) - 1}{\log_2 K} \leq e_{bayes}(X) \leq 0.5 \times H(Y|X) \]

- **Kullback-Leibler divergence:**
  \[ MI(X, Y) \approx D_{KL}(p(x, y)\|p(y)p(x)), \]
  where \[ D_{KL}(p_1\|p_2) = \int_x p_1(x) \log_2 \frac{p_1(x)}{p_2(x)} \, dx \]
Mutual information

\[ MI(Y, X) = H(Y) - H(Y|X) = \int_x \int_y p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \, dx \, dy \]

**Problem:** \( p(x), p(y), p(x, y) \) are unknown and hard to estimate from the data

**Classification with nominal or discrete features**

- The simplest case – we can estimate the probabilities from the frequency counts
- This introduces a negative bias
- Harder estimate with larger numbers of classes and feature values
Relevance Measures Based on Information Theory

**Mutual information**

\[
MI(Y, X) = H(Y) - H(Y|X) = \int_x \int_y p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \, dx \, dy
\]

**Problem:** \(p(x), p(y), p(x, y)\) are unknown and hard to estimate from the data

**Classification with nominal or discrete features**

- MI corresponds to the Information Gain (IG) for Decision trees
- Many modifications of IG (avoiding bias towards the multivalued features)
  - Information Gain Ratio \(IGR(Y, X) = \frac{MI(Y, X)}{H(X)}\)
  - Gini-index, J-measure, ....
- Relaxed entropy measures are more straightforward to estimate:
  - Renyi Entropy \(H_\alpha(X) = \frac{1}{1-\alpha} \log_2 (\int_x p(x)^\alpha) \, dx\)
  - Parzen window approach
Relevance Measures Based on Information Theory

**Mutual information**

\[
MI(Y, X) = H(Y) - H(Y|X) = \int_x \int_y p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} dx dy
\]

**Problem:** \(p(x), p(y), p(x, y)\) are unknown and hard to estimate from the data

**Regression with continuous features**

- The hardest case
- Possible solutions:
  - Histogram-based discretization:
    - MI is overestimated – depending on the quantization level
    - MI should be overestimated the same for all features
  - Approximation of the densities (Parzen window, ...)
    - Normal distribution \(\rightarrow\) correlation coefficient
    - Computational complexity
  - ...
– Non-linear relation measures

• Correlation is only capable of measuring linear dependence
• A more powerful measure is the mutual information $I(Y; C)$

$$J(Y_M) = I(Y_M; C) = H(C) - H(C|Y_M) = \sum_{c=1}^{C} \int_{Y_M} p(Y_M, \omega_c) \log \frac{p(Y_M, \omega_c)}{p(Y_M)P(\omega_c)} dx$$

• The mutual information between the feature vector and the class label $I(Y_M; C)$ measures the amount by which the uncertainty in the class $H(C)$ is decreased by knowledge of the feature vector $H(C|Y_M)$, where $H(\cdot)$ is the entropy function

• Note that mutual information requires the computation of the multivariate densities $p(Y_M)$ and $p(Y_M, \omega_c)$, which is ill-posed for high-dimensional spaces

• In practice [Battiti, 1994], mutual information is replaced by a heuristic such as

$$J(Y_M) = \sum_{m=1}^{M} I(x_{i_m}; C) - \beta \sum_{m=1}^{M} \sum_{n=m+1}^{M} I(x_{i_m}; x_{i_n})$$
Filters vs. wrappers

Filters

- **Fast execution (+):** Filters generally involve a non-iterative computation on the dataset, which can execute much faster than a classifier training session
- **Generality (+):** Since filters evaluate the intrinsic properties of the data, rather than their interactions with a particular classifier, their results exhibit more generality: the solution will be “good” for a larger family of classifiers
- **Tendency to select large subsets (-):** Since the filter objective functions are generally monotonic, the filter tends to select the full feature set as the optimal solution. This forces the user to select an arbitrary cutoff on the number of features to be selected

Wrappers

- **Accuracy (+):** wrappers generally achieve better recognition rates than filters since they are tuned to the specific interactions between the classifier and the dataset
- **Ability to generalize (+):** wrappers have a mechanism to avoid overfitting, since they typically use cross-validation measures of predictive accuracy
- **Slow execution (-):** since the wrapper must train a classifier for each feature subset (or several classifiers if cross-validation is used), the method can become unfeasible for computationally intensive methods
- **Lack of generality (-):** the solution lacks generality since it is tied to the bias of the classifier used in the evaluation function. The “optimal” feature subset will be specific to the classifier under consideration
Search strategies

Exponential algorithms
– Evaluate a number of subsets that grows exponentially with the dimensionality of the search space
  • Exhaustive Search (already discussed)
  • Branch and Bound
  • Approximate Monotonicity with Branch and Bound
  • Beam Search

Sequential algorithms
– Add or remove features sequentially, but have a tendency to become trapped in local minima
  • Sequential Forward Selection
  • Sequential Backward Selection
  • Plus-l Minus-r Selection
  • Bidirectional Search
  • Sequential Floating Selection

Randomized algorithms
– Incorporate randomness into their search procedure to escape local minima
  • Random Generation plus Sequential Selection
  • Simulated Annealing
  • Genetic Algorithms
Naïve sequential feature selection

One may be tempted to evaluate each individual feature separately and select the best M features

– Unfortunately, this strategy RARELY works since it does not account for feature dependence

Example

– The figures show a 4D problem with 5 classes
– Any reasonable objective function will rank features according to this sequence: \( J(x_1) > J(x_2) \approx J(x_3) > J(x_4) \)
  • \( x_1 \) is the best feature: it separates \( \omega_1, \omega_2, \omega_3 \) and \( \{\omega_4, \omega_5\} \)
  • \( x_2 \) and \( x_3 \) are equivalent, and separate classes in three groups
  • \( x_4 \) is the worst feature: it can only separate \( \omega_4 \) from \( \omega_5 \)
– The optimal feature subset turns out to be \( \{x_1, x_4\} \), because \( x_4 \) provides the only information that \( x_1 \) needs: discrimination between classes \( \omega_4 \) and \( \omega_5 \)
– However, if we were to choose features according to the individual scores \( J(x_k) \), we would certainly pick \( x_1 \) and either \( x_2 \) or \( x_3 \), leaving classes \( \omega_4 \) and \( \omega_5 \) non separable
  • This naïve strategy fails because it does not consider features with complementary information
Sequential forward selection (SFS)

SFS is the simplest greedy search algorithm

- Starting from the empty set, sequentially add the feature $x^+$ that maximizes $J(Y_k + x^+)$ when combined with the features $Y_k$ that have already been selected

1. Start with the empty set $Y_0 = \{\emptyset\}$
2. Select the next best feature $x^+ = \arg \max \limits_{x \in Y_k} J(Y_k + x)$
3. Update $Y_{k+1} = Y_k + x^+$; $k = k + 1$
4. Go to 2

Notes

- SFS performs best when the optimal subset is small
  - When the search is near the empty set, a large number of states can be potentially evaluated
  - Towards the full set, the region examined by SFS is narrower since most features have already been selected

- The search space is drawn like an ellipse to emphasize the fact that there are fewer states towards the full or empty sets
  - The main disadvantage of SFS is that it is unable to remove features that become obsolete after the addition of other features
Example

Run SFS to completion for the following objective function

\[ J(X) = -2x_1x_2 + 3x_1 + 5x_2 - 2x_1x_2x_3 + 7x_3 + 4x_4 - 2x_1x_2x_3x_4 \]

- where \( x_k \) are indicator variables, which indicate whether the \( k^{th} \) feature has been selected \((x_k = 1)\) or not \((x_k = 0)\)

Solution

\[
\begin{align*}
J(x_1) &= 3 \\
J(x_2) &= 5 \\
J(x_3) &= 7 \\
J(x_4) &= 4
\end{align*}
\]

\[
\begin{align*}
J(x_3x_1) &= 10 \\
J(x_3x_2) &= 12 \\
J(x_3x_4) &= 11
\end{align*}
\]

\[
\begin{align*}
J(x_3x_2x_1) &= 11 \\
J(x_3x_2x_4) &= 16
\end{align*}
\]

\[ J(x_3x_2x_4x_1) = 13 \]
Sequential backward selection (SBS)

SBS works in the opposite direction of SFS

– Starting from the full set, sequentially remove the feature $x^-$ that least reduces the value of the objective function $J(Y - x^-)$

  • Removing a feature may actually increase the objective function $J(Y_k - x^-) > J(Y_k)$; such functions are said to be non-monotonic (more on this when we cover Branch and Bound)

1. Start with the full set $Y_0 = X$
2. Remove the worst feature $x^- = \arg \max_{x \in Y_k} J(Y_k - x)$
3. Update $Y_{k+1} = Y_k - x^-; k = k + 1$
4. Go to 2

Notes

– SBS works best when the optimal feature subset is large, since SBS spends most of its time visiting large subsets

– The main limitation of SBS is its inability to reevaluate the usefulness of a feature after it has been discarded
Plus-L minus-R selection (LRS)

A generalization of SFS and SBS

- If L>R, LRS starts from the empty set and repeatedly adds L features and removes R features
- If L<R, LRS starts from the full set and repeatedly removes R features followed by L additions

Notes

- LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities
- Its main limitation is the lack of a theory to help predict the optimal values of L and R

\[
\begin{align*}
1. & \text{ If } L>R \text{ then } Y_0 = \{\emptyset\} \\
& \text{ else } Y_0 = X; \text{ go to step 3} \\
2. & \text{ Repeat } L \text{ times} \\
& x^+ = \arg \max_{x \notin Y_k} J(Y_k + x) \text{;} \quad Y_{k+1} = Y_k + x^+; \quad k = k + 1 \\
3. & \text{ Repeat } R \text{ times} \\
& x^- = \arg \max_{x \in Y_k} J(Y_k - x) \text{;} \quad Y_{k+1} = Y_k - x^-; \quad k = k + 1 \\
4. & \text{ Go to 2}
\end{align*}
\]
Bidirectional Search (BDS)

**BDS is a parallel implementation of SFS and SBS**

- SFS is performed from the empty set
- SBS is performed from the full set
- To guarantee that SFS and SBS converge to the same solution
  - Features already selected by SFS are not removed by SBS
  - Features already removed by SBS are not selected by SFS

1. Start SFS with $Y_F = \{\emptyset\}$
2. Start SBS with $Y_B = X$
3. Select the best feature
   \[
   x^+ = \arg \max_{\substack{x \in F_k^L \setminus Y_F^k \setminus F_k^B \setminus Y_B^k \setminus X \setminus Y_B^k \setminus X}} J(Y_{F_k^L} + x)
   \]
   \[
   Y_{F_{k+1}}^L = Y_{F_k}^L + x^+
   \]
4. Remove the worst feature
   \[
   x^- = \arg \max_{\substack{x \in Y_B^k \setminus Y_F^k \setminus F_k^B \setminus Y_B^k \setminus X \setminus Y_B^k \setminus X}} J(Y_{F_k^L} - x)
   \]
   \[
   Y_{B_{k+1}}^L = Y_{B_k}^L - x^-; \ k = k + 1
   \]
5. Go to 2
Sequential floating selection (SFFS and SFBS)

An extension to LRS with flexible backtracking capabilities

- Rather than fixing the values of L and R, these floating methods allow those values to be determined from the data:
  - The dimensionality of the subset during the search can be thought to be “floating” up and down

There are two floating methods

- Sequential floating forward selection (SFFS) starts from the empty set
  - After each forward step, SFFS performs backward steps as long as the objective function increases

- Sequential floating backward selection (SFBS) starts from the full set
  - After each backward step, SFBS performs forward steps as long as the objective function increases
SFFS Algorithm (SFBS is analogous)

1. $Y = \{\emptyset\}$
2. Select the best feature
   \[ x^+ = \arg \max_{x \notin Y_k} J(Y_k + x) \]
   \[ Y_k = Y_k + x^+; \; k = k + 1 \]
3. Select the worst feature*
   \[ x^- = \arg \max_{x \in Y_k} J(Y_k - x) \]
4. If $J(Y_k - x^-) > J(Y_k)$ then
   \[ Y_{k+1} = Y_k - x^-; \; k = k + 1 \]
   Else
   Go to step 3
   Else
   Go to step 2

*Notice that you’ll need to do book-keeping to avoid infinite loops
Random generation plus sequential selection

Approach

- An attempt to introduce randomness into SFS and SBS in order to escape local minima
- The algorithm is self-explanatory

1. Repeat for a number of iterations
   - Generate a random feature subset
   - Perform SFS on the subset
   - Perform SBS on the subset
Genetic algorithms

GAs are an optimization technique inspired by evolution (survival of the fittest)

- Starting with an initial random population of solutions, evolve new populations by mating (crossover) pairs of solutions and mutating solutions according to their fitness (objective function)
- The better solutions are more likely to be selected for the mating and mutation operations and therefore carry their “genetic code” from generation to generation
  
  • For FSS, solutions are simply represented with an indicator variable [Holland in 1974]

1. Create an initial random population
2. Evaluate initial population
3. Repeat until convergence (or a number of generations)
   - Select the fittest individuals in the population
   - Create offsprings through crossover on selected individuals
   - Mutate selected individuals
   - Create new population from the old population and the offspring
   - Evaluate the new population
**Single-point crossover**

- Select two individuals (parents) according to their fitness
- Select a crossover point
- With probability $P_C$ (0.95 is reasonable) create two offspring by combining the parents

![Diagram of single-point crossover](image)

**Binary mutation**

- Select an individual according to its fitness
- With probability $P_M$ (0.01 is reasonable) mutate each one of its bits

![Diagram of binary mutation](image)
Selection methods

The selection of individuals is based on their fitness value

– We will describe a selection method called Geometric selection
– Other methods are available: Roulette Wheel, Tournament Selection ...

Geometric selection

– The probability of selecting the \( r^{th} \) best individual is given by the geometric distribution

\[
P(r) = q(1 - q)^{r-1}
\]

• where \( q \) is the probability of selecting the best individual (0.05 being a reasonable value)

– Therefore, the geometric distribution assigns higher probability to individuals ranked better, but also allows unfit individuals to be selected

– In addition, it is typical to carry the best individual of each population to the next one; this is called the Elitist Model
GA parameter choices for FSS

The choice of crossover rate $P_C$ is important

- You will want a value close to 1.0 to have a large number of offspring
- The optimal value is inversely proportional to population size [Doak, 1992]

The choice of mutation rate $P_M$ is very critical

- An optimal choice will allow the GA to explore the more promising regions while avoiding getting trapped in local minima
  - A large value (i.e., $P_M > 0.25$) will not allow the search to focus on the better regions, and the GA will perform like random search
  - A small value (i.e., $P \approx 0$) will not allow the search to escape local minima

The choice of $q$, the probability of selecting the best individual, is also critical

- An optimal value of $q$ will allow the GA to explore the most promising solution, and at the same time provide sufficient diversity to avoid early convergence of the algorithm

In general, poorly selected control parameters will result in sub-optimal solutions due to early convergence
## Search strategies – summary

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Complexity</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exhaustive</strong></td>
<td>Always finds the optimal solution</td>
<td>Exponential</td>
<td>High accuracy</td>
<td>High complexity</td>
</tr>
<tr>
<td><strong>Sequential</strong></td>
<td>Good if no backtracking needed</td>
<td>Quadratic $O(N_{EX}^2)$</td>
<td>Simple and fast</td>
<td>Cannot backtrack</td>
</tr>
<tr>
<td><strong>Randomized</strong></td>
<td>Good with proper control parameters</td>
<td>Generally low</td>
<td>Designed to escape local minima</td>
<td>Difficult to choose good parameters</td>
</tr>
</tbody>
</table>

A highly recommended review of the material presented in these two lectures is

Justin Doak
“An evaluation of feature selection methods and their application to Computer Security”
University of California at Davis, Tech Report CSE-92-18