CSCE 970: Pattern Recognition **Spring Semester, 1999**

Topic Lecture 5: Feature Vector Selection

Lecturer: Stephen Scott *Scribe: Menghua Fu*

1 Introduction

In these lectures, we discussed the Feature Vector Selection algorithm. Feature Vector Selection is usually more effective than the previous discussed Scalar Feature Selection because the former considers mutual correlations between features. Feature Vector Selection can be either optimal or suboptimal. For optimal methods, we need to form all possible vector combinations of l features out of the m originally availables. It does not require much thought to see that the computational burden is the major limiting factor of optimal approach. As result, some suboptimal algorithms are used instead. In the main section, we will discuss the three widely used suboptimal searching techniques: Sequential Backward Selection, Sequential Forward Selection and Floating Search Method. After that, we will priefly discuss Feature Linear transformation.

2 Review of Feature Vector Selection

The objective of Feature Vector Selection is to select a subset of l features out of m original features(1≤m). In the previous lectures, we discussed the Scalar Feature Selection in which we treated features separately. The advantage of Scalar Feature Selection is simple in computation, but it may not be effective enough for complex problems and for features with high mutual correlations. In Feature Vector Selection algorithm, we take into consideration of feature relationships. Therefore, we expect that Feature Vector Selection is more powerful than Scalar Feature Selection in practical applications. The followings are outlines of three widely used algorithms discussed in class.

1. **Sequential BackWard Selection**
    - Starts with *all* features selected.
    - Choose the locally best one to eliminate, *i.e.* the one that leaves a vector maximizing C(.).
    - Repeat until only l features remain.
2. **Sequential Forward Selection**
    - Start with the *single* best feature selected.
    - Choose the locally best one to add, i.e. the one that leaves a vector maximizing C(.).

- Repeat until l features have been selected.

3. **Floating Search Method**
   - Forward: Select best element from Y_{m-k} to add to X_k and check if we can exchange it for another element in X_k to increase C(X_k).
     (a) If **YES**: Go to Backward.
     (b) If **NO**: Add new elements to X_k, yielding X_{k+1}, increment k, and repeat.
   - Backward: Select best element from X_k to eliminate and check if it yeilds a better X_{k-1}.
     (a) If **YES**: Delet element from X_k, yielding X_{k-1}, decreament k, and repeat.
     (b) If **NO**: Go to Forward.

Sequential Backward and Forward Selection algorithms suffer from the so-called nesting effect. On the other hand, in Floating Search Method, the pre-selected feature can be discarded or re-selected later.

3 The Feature Vector Selection

Feature Vector Selection selects a subset features out of original available as well as considers the feature mutual correlation(unlike the previous discussed Scalar Feature Selection). For the optimal Feature Vector Selection algorithm, we need to try all the possible feature combinations. Obviously, this is usually very complicated. For instance, if $m=20$ and $l=5$, the number of vector combinations is as high as 15,504! As result of it, we usually have to decrease the computation burden in practice. If C(.) is monotonic, *i.e.* C(x_1, x_2, ... x_k) $\leq$ C(x_1, x_2, ... x_k, x_{k+1}), dynamic programming [1] and Branch & Bound (see Chapter 15 for detail) can be used.

If C(.) is not monotonic, some other suboptimal methods can be used to optimize(reduce) computational work. It should be noticed that suboptimal methods do not always guarantee to get the best subset of features. Suboptimal methods are usually used in practice because of their less computation burden. The three widely used techniques are: Sequential Backward Selection, Sequential Forward Selection and Floating Search Method. The former two suffer from the nesting effect. That is, once a feature is discarded in backward method, there is no way for it to be reconsidered again. The opposite is true for the forward procedure; Once a feature is chosen, there is impossible for it to be discarded later on. On the other hand, Floating Search provides the flexibility to reconsider previously discarded feature and drop the pre-selected feature later.

3.1 Sequential Backward Selection

This method starts with all features selected. At each step we drop out one feature from the "best" combinations until we obtain a vector of exactly l features. The number of combination(time complexity) searched via this method is $1+\frac{1}{2}*((m+1)*m-l*(l+1))$. The execution of Sequential Backward Selection can be demonstrated by the following simple example. Here, we assume $m=4$ and $l=2$ with four features: x_1, x_2, x_3, x_4.

- Adopt a class separability criterion, C(For how to calculate C, please see the previous lectures), and compute its value for the feature vector $[x_1, x_2, x_3, x_4]^T$.
- Eliminate one feature and for each of the possible resulting combinations, that is, $[x_1, x_2, x_3]^T$, $[x_1, x_2, x_4]^T$, $[x_1, x_3, x_4]^T$ and $[x_2, x_3, x_4]^T$, compute the corresponding criterion value. Select the combination with the best value, say $[x_1, x_2, x_3]^T$.
- From the selected three-dimensional feature vector eliminate one feature and for each of the resulting combinations, $[x_1, x_2]^T$, $[x_1, x_3]^T$ and $[x_2, x_3]^T$, compute the criterion value and select the one with the best value. Now, the number of features selected is 2, we stop the execution.

3.2 Sequential Forward Selection

On the opposite, this method starts with a single best feature selected. At each step, we add one new feature into the "best" combinations until we get a vector of l features. The number of combinations(time complexity) for Sequential Forward Selection is $l*m-l*(l-1)/2$. The execution of this method can be demonstrated by the same example as:

- Computer the criterion value for each of the features. Select the feature with the best value, say x_1.
- From all possible two-dimensional vectors that contain the winner from the previous step, that is, $[x_1, x_2]^T$, $[x_1, x_3]^T$ and $[x_1, x_4]^T$. Compute the criterion value for each of them and select the best one, say, $[x_1, x_3]^T$. Now, we get two features so the execution stops here.

3.3 Floating Search Method

There are two schemes used to implement Floating Search Method [5]. One is the forward selection and the other is the backward selection. The relationale at the heart of the method is summarized as follows:

At the next step, the k+1 best subset X_{k+1} is formed by "borrowing" an element from Y_{m-k}. Then, return to the previously selected lower dimension subsets to check whether the inclusion of this new element improves the criterion. If it does, the new element replaces one of the previously selected features. The steps of the algorithm, when maxization of C is required are:

- **Step I: Inclusion**

  $X_{k+1} = \arg MAX_{y \in Y_{m-k}} C(\{X_k, Y\})$; that is, choose that element from Y_{m-k} which, combined with X_k, results to the best value of C. Let $X_{k+1} = \{X_x, X_{k+1}\}$.

- **Step II: Test**
  1. Find the feature that has the least effect on the cost when it is removed from X_{k+1}. That is, find x_r such that $x_r = \arg MAX_{x_r \in X_{k+1}} C(X_{k+1} - \{x_r\})$.
  2. If r=k+1, change k=k+1 and go to Step I.
  3. If $C(X_{k+1} - \{x_r\}) < C(X_k)$ AND r is not equal to k+1, go to Step I; that is, if removal of X_r does not improve upon the cost of the previously selected best group of k, no further backward search is performed.
  4. If k=2 put X_k=$X_{k+1} - \{x_r\}$ and $C(X_k) = C(X_{k+1} - \{x_r\})$; go to Step I.

- **Step III: Exclusion**
  1. $X_k^{'} = X_{k+1} - \{x_r\}$; that is, remove x_r.
  2. $x_s = argMAX_{y \in X_k^{'}} C(X_k^{'} - \{y\})$; that is, find the least significant feature in the new set.
  3. If $C(X_k^{'} - \{x_s\}) < C(X_{k-1})$ then $X_k = X_k^{'}$ and go to Step I; no further backward search is performed.
  4. Put $X_{k-1}^{'}$=$X_k^{'}$-$\{x_s\}$ and k=k-1.
  5. If k=2 put X_k=$X_k^{'}$ and C(X_k)=C($X_k^{'}$) and go to Step I.
  6. Go to Step III.

3.4 Comparisons of Complexity and Performance

1. **Time Complexity**

   Obviously, the time complexity of Floating Search Method is larger than those of the other two. We have given the time complexities for backward and forward algorithms. The complexity of Floating Search Method is quite complicated. Compared with Sequential Backward Selection, Sequential Forward Selection is more efficient when $m>>l$. For example, when m=10*l, the time complexity for backward algorithm is:

   $(O_{Backward}) = 1 + 0.5 * ((m + 1) * m - l * (l + 1))$
   $= 1 + 0.5 * ((10 * l + 1) * 10 * l - l^2 - l)$
   $= 1 + 50 * l^2 + 5 * l - 0.5 * l^2 - 0.5 * l$
   $= 49.5 * l^2 + 4.5 * l + 1$

   The time complexity for forward algorithm is:
   $(O_{Forware}) = l * m - l * (l - 1)/2$
   $= l * (10 * l) - l * (l - 1)/2$
   $= 10 * l^2 - 0.5 * l^2 + 0.5 * l$
   $= 9.5 * l^2 + 0.5 * l$

For detail information about the complexity and performance of feature selection, please read the paper written by Jian and Zongker [3].

2. **Performance**

   First of all, all of these three algorithms are suboptimal methods. Since Floating Search Method is free of nesting effective, it is generally believed that Floating Search Method is better than other two.

3.5 Feature Generation

We can also transform the existing features into new ones so that the classes can be well separated by the new transformed features. A common practice is linear transformation: y=A^TX, where X is the original m-vector, y is new l-vector and A^T is l x m transformation matrix. We can learn linear transformation matrix by neural network. The Auto-Associative Networks [2] [4] can be simply experessed as:

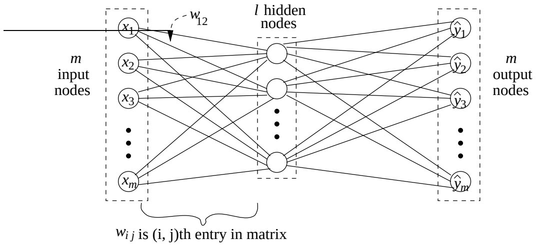

Figure 1: The Auto-Associative Networks

References

[1] Bellman R.E. “Dynamic Programming”, Princeton University Press, 1957.

[2] Bourland H., Kamp Y “Auto-association by multilayer perceptrons and singular value decomposition” *Biological Cybernetics*, Vol. 59, pp.291-294, 1988.

[3] Jain A., Zongker D. “Feature selection: Evaluation, application, and small sample performance” *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 19(2), pp.153-158, 1997.

[4] Kramer M.A. “Nonlinear principal component analysis using auto-associative neural networks” *AIC Journal*, Vol. 37(2), pp.233-243, 1991.

[5] Pudil P., Novovicova J., Kittler J. “Floating search methods in feature selection”, *Pattern Recognition Letters*, Vol. 15, pp.1119-1125, 1994.