Parallel Filter-Based Feature Selection Based on Balanced Incomplete Block Designs

Antonio Salmerón¹, Anders L. Madsen²,³, Frank Jensen², Helge Langseth⁴, Thomas D. Nielsen³, Darío Ramos-López¹, Ana M. Martínez³, Andrés R. Masegosa⁴

¹Department of Mathematics, University of Almería, Spain
²Hugin Expert A/S, Aalborg, Denmark
³Department of Computer Science, Aalborg University, Denmark
⁴Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway
The AMIDST Toolbox for probabilistic machine learning

You can download our open source Java toolbox:

http://www.amidsttoolbox.com

Software demonstration today at 12 in the Lobby!
Outline

1 Motivation

2 Preliminaries

3 Scaling up filter-based feature selection

4 Experimental results

5 Conclusions
Outline

1 Motivation

2 Preliminaries

3 Scaling up filter-based feature selection

4 Experimental results

5 Conclusions
Before **learning a classification model**, a key step is **choosing the most informative variables**, and **excluding the redundant ones**. A good feature selection prevents overfitting and reduces the computational load of the learning process.

Main groups of **feature selection methods**:

- **Wrapper**: normally requiring a model fit for each subset of variables (time consuming).
- **Embedded methods**: also model-dependent, guided by some specific property of the model.
- **Filter**: independent from the model, usually features are ranked according to a univariate score (less computational complexity).
Our proposal

We propose an algorithm for scaling up filter-based feature selection in classification problems, such that:

▶ It makes use of conditional mutual information as filter measure.

▶ It parallelize the workload while avoiding unnecessary calculations.

▶ It uses the balanced incomplete blocks (BIB) designs to reduce disk access and to distribute the calculations.

▶ It is able to significantly reduce the computational load in a multi-core shared-memory environment.
Outline

1 Motivation
2 Preliminaries
3 Scaling up filter-based feature selection
4 Experimental results
5 Conclusions
Entropy and Conditional Entropy

\( \mathbf{X} = \{X_1, \ldots, X_n\} \): discrete variables; \( C \): the class variable.

Entropy of \( \mathbf{X} \in \mathbf{X} \):

\[
H(\mathbf{X}) = - \sum_{x \in \Omega_X} p(x) \log p(x)
\]

(uncertainty in the distribution of \( \mathbf{X} \))

Conditional entropy of \( X_i \) given \( X_j \):

\[
H(X_i|X_j) = - \sum_{x_j \in \Omega_{X_j}} p(x_j) \sum_{x_i \in \Omega_{X_i}} p(x_i|x_j) \log p(x_i|x_j)
\]

(remaining uncertainty in the distribution of \( X_i \) after observing \( X_j \))
Mutual Information of $X_i$ and $X_j$:

$$I(X_i, X_j) = H(X_i) - H(X_i | X_j)$$

(amount of information shared by two variables)

The mutual information is symmetric: $I(X_i, X_j) = I(X_j, X_i)$

Conditional Mutual Information between $X_i$ and $X_j$ given $X_k$:

$$I(X_i, X_j | X_k) = H(X_i | X_k) - H(X_i | X_j, X_k).$$

(amount of information shared by two variables, given a third one)
Information-theoretic filter methods have been analyzed using the **conditional likelihood**:

\[
\mathcal{L}(S, \tau|\mathcal{D}) = \prod_{i=1}^{n} q(c^i|x^i, \tau),
\]

- **S**: features included in the model,
- **τ**: parameters of the distributions involved in the model,
- **D**: a dataset, \( \mathcal{D} = \{(x^i, c^i), i = 1, \ldots, n\} \)
- **q**: the learnt model

The **conditional likelihood** is maximized by minimizing \( I(X \setminus S, C|S) \) (the mutual information between the class and the features not included in the model, given the variables actually included).
Filter measures

We shall make this technical assumption:
For $X_i, X_j \in S$, $X_k \in X \setminus S$, it holds that $X_i$ and $X_j$ are conditionally independent both when conditioning on $X_k$ and on $\{X_k, C\}$.

This allows us to select features greedily, using as a filter measure the following quantity based on the conditional mutual information (cmi):

$$J_{\text{cmi}}(X_i) = I(X_i, C|S) = I(X_i, C) - \sum_{X_j \in S} (I(X_i, X_j) - I(X_i, X_j|C)),$$

where $X_i$ is the candidate variable to include in the model.
Another remarkable filter measure is the joint mutual information (jmi) that is defined as:

$$J_{jmi}(X_i) = \sum_{X_j \in S} I(\{X_i, X_j\}, C) =$$

$$= I(X_i, C) - \frac{1}{|S|} \sum_{X_j \in S} (I(X_i, X_j) - I(X_i, X_j|C)),$$

According to the literature, $J_{jmi}$ is the metric showing the best accuracy/stability tradeoff, and is therefore the one we utilize in the following.
Outline

1 Motivation

2 Preliminaries

3 Scaling up filter-based feature selection

4 Experimental results

5 Conclusions
Filter-based feature selection algorithm

1 Function Filter\((X, C, M)\)

Input: The set of features, \(X = \{X_1, \ldots, X_N\}\). The class variable, \(C\). The maximum number of features to be selected, \(M\).

Output: \(S\), a set of selected features.

begin
   for \(i \leftarrow 1\) to \(N\) do
      Compute \(I(X_i, C)\);
      for \(j \leftarrow i + 1\) to \(N\) do
         Compute \(I(X_i, X_j | C)\);
         Compute \(I(X_i, X_j)\);
      end
   end
   \(X^* \leftarrow \arg \max_{1 \leq i \leq N} I(X_i, C)\);
   \(S \leftarrow \{X^*\}\);
   for \(i \leftarrow 1\) to \(M - 1\) do
      \(R \leftarrow X \setminus S\);
      for \(X \in R\) do
         Compute \(J_{jmi}(X)\) using the statistics computed in Steps 4, 6, and 7;
      end
      \(X^* \leftarrow \arg \max_{X \in R} J_{jmi}(X)\);
      \(S \leftarrow S \cup \{X^*\}\);
   end
   return \(S\);
end
Optimizing the calculation of the scores

To compute all the necessary information theory scores, one possibility is to create a thread for each pair of features. However, for $n$ variables, this requires accessing the dataset $\binom{n}{2}$ times, inducing a significant overhead due to disk/network access.

We propose making groups of variables (blocks), each of which will only access the dataset a single time.

To do this, two key issues:

- Finding an appropriate block size.
- Ensure that every pair of variables appears in exactly one block (to avoid duplicated calculations).
Balanced incomplete block (BIB) designs

What we seek can be done by using Balanced Incomplete Block (BIB) designs.

Given a set of variables $X$, we will say $(X, \mathcal{A})$ is a design if $\mathcal{A}$ is a collection of non-empty subsets of $X$ (blocks).

A design is a $(v, k, \lambda)$-BIB design if:

- $v = |X|$ is the number of variables in $X$.
- each block in $\mathcal{A}$ contains exactly $k$ variables.
- every pair of distinct variables is contained in exactly $\lambda$ blocks.

We have found that $(q, 6, 1)$-BIB designs (blocks of 6 variables) are appropriate for practical use.
Some considerations about BIB designs:

▶ A \((\nu, k, \lambda)\)-BIB might not exist, for some combinations of the parameters.

▶ Finding a BIB design is a NP-complete problem.

▶ To efficiently use BIB designs, we have pre-calculated a number of them, and those are utilized on run-time.

▶ BIB designs can be generated from *difference sets*, avoiding the need of storing the full design.

Check the paper for more details on this!
Example of BIB designs

Figure 1: Example illustrating the use of \((q, 6, 1)\) and \((3, 2, 1)\) designs.
Outline

1 Motivation

2 Preliminaries

3 Scaling up filter-based feature selection

4 Experimental results

5 Conclusions
Experimental setup

- **Goal:** empirical evaluation of the time performance improvement using BIB designs.

- **Shared memory computer with multiple cores:** Intel (TM) i7-5820K 3.3GHz processor (6 physical and 12 logical cores), with 64 GB RAM, running Red Hat Enterprise Linux 7.

- **Time measurements:** averaged over 10 runs with each dataset, elapsed (wall-clock) time.

- **Datasets:** randomly simulated from well-known Bayesian networks, and a real-world dataset from a Spanish bank.
**Tested Bayesian networks**

Table 1: Bayesian networks from which datasets were generated.

| Dataset   | $|\mathcal{X}|$ | $|E|$ | Total CPT size |
|-----------|----------------|------|----------------|
| Munin1    | 189            | 282  | 19,466         |
| Diabetes  | 413            | 602  | 461,069        |
| Munin2    | 1,003          | 1,244| 83,920         |
| SACSO     | 2,371          | 3,521| 44,274         |

$|\mathcal{X}|$: number of variables in the Bayesian network  
$|E|$: number of edges in the Bayesian network  
CPT: total conditional probability table size

In addition, a real-world dataset of 1823 variables from a Spanish bank was also tested.
Speed-up for Munin1 datasets

100,000 cases

500,000 cases

250,000 cases
Speed-up for Munin2 datasets

100.000 cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up

500.000 cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up

250.000 cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up
Speed-up for SACSO datasets

250,000 cases

500,000 cases
Speed-up for Diabetes datasets

![Graph showing average run time in seconds and average speed-up factor against number of threads for 100k cases.](image)
Speed-up for Bank datasets

100k cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up

500k cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up

250k cases

Average run time in seconds
Average speed-up factor
Number of threads
Time
Speed-up
Impact of using the \((q, 6, 1)\)-BIB designs

Bank dataset

100k cases with BIB designs

100k cases, using pairs directly
Outline

1 Motivation

2 Preliminaries

3 Scaling up filter-based feature selection

4 Experimental results

5 Conclusions
Conclusions

▶ A parallel filter-based feature selection algorithm has been proposed.
▶ It makes use of information theoretic measures (conditional mutual information) for filtering.
▶ Also, a two-steps Balanced Incomplete Blocks (BIB) design has been used to distribute and optimize the computations asynchronously \(((q, 6, 1) \text{ and then } (3, 2, 1))\).
▶ For variables with a large number of states, it might be preferable to skip the first step (Diabetes).
▶ The performance improvement when using \((q, 6, 1)\) designs is in most cases substantial.
▶ Speed-up factors of about 4-6 were obtained running on a 6 physical cores computer.
Future work

- Horizontal parallelization: each computing unit holding only a subset of the data over all variables.
- This will support parallelization on a distributed memory system.
Thank you for your attention

You can download our open source Java toolbox:

http://www.amidsttoolbox.com

Acknowledgments: This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209