Abstract

A popular method for feature selection are filters based on the estimation of the mutual information between the features and the target. If the data is very high dimensional, even simple, iterative methods require substantial computational time. In this work we propose an early stopping method for feature selectors that reduces the complexity of the feature selector by orders of magnitude without any loss of predictive performance. We demonstrate the practical use of early stopping on high dimensional image classification tasks.

1. Introduction

There are two major aims of feature selection. On the one hand feature selection uncovers which variables are important in the process that generates the target variable. On the other hand it allows to make predictions about the target variable based on the selected features. Due to computational constraints, many Machine Learning algorithms can only be applied to very high dimensional datasets, if prior feature selection is performed. Further, it reduces overfitting as it sparsifies the data. In this paper we concentrate on feature selection for prediction.

There is a vast number of approaches to tackle the problem of feature selection, for an overview see [5]. Usually feature selection methods are classified into filters, wrappers and embedded methods. While wrappers use the predictor as a black box and embedded methods perform feature selection in the training phase, filters are completely independent of the predictor. This independence makes them much more modular, after filtering any predictor can be applied to the selected features. Further, in non-sparse ultra-high dimensional settings, wrappers and embedded methods are often computationally very expensive if not infeasible.

In this work, we focus on filter methods which rely on mutual information [3]. These methods try to select features that approximately maximise the mutual information between the features and the target. As the evaluation of a large number of candidate subsets is intractable in high dimension, many methods rely on an iterative selection procedure. Therefore the algorithm keeps a vector of scores about the features quality and updates the scores each time a feature is selected.

The idea of ‘early stopping’ is to stop updating the scores at some iteration and select features in future iterations according to a fixed vector of scores. We illustrate this method by applying it to the popular filter method CMIM (Conditional Mutual Information Maximisation) [4]. We show that without much loss of information, early stopping allows to save a considerable amount of computational time and sometimes even improves predictive performance.

2. Algorithm

2.1. Variable Selection Based on Mutual Information

Following the argumentation in [4] and [2] one would ideally like to choose $K$ features $\nu(1), \ldots, \nu(K)$ such that the mutual information of the selected features and the target $I(X_{\nu(1)}, \ldots, X_{\nu(K)}; Y)$ is maximised. This quantity is very difficult to estimate, but it can be decomposed as

$$I(X_{\nu(1)}; \nu(K); Y) = I(X_{\nu(K)}; Y | X_{\nu(1):\nu(K-1)}) + I(X_{\nu(1):\nu(K-1)}; Y)$$

$$= I(X_{\nu(1)}; Y) + \sum_{k=2}^{K} I(X_{\nu(k)}; Y | X_{\nu(1):\nu(k-1)}).$$

Instead of maximising these sums jointly with respect to $\nu(1), \ldots, \nu(K)$ one can maximise each summand separately. As a result, one obtains a greedy iterative procedure that approximately maximises the mutual infor-
2.3. Early Stopping

Roughly speaking, we select the feature that adds most information about the target to the set of features selected so far.

To apply early stopping we replace $X_{\nu(k-1)}$ by $X_{\nu((k-1)g)}$ for some $g < K$ which means that the information contained in features selected after iteration $g$ will not be considered when selecting features in the following iterations.

2.2. The CMIM Algorithm

The difficulty of above procedure lies in the estimation of mutual information conditioned on a large set of variables, which is problematic in terms of computational complexity as well as estimation error. In contrast, mutual information conditioned on only one variable is easier to estimate. Therefore a large number of approaches have been proposed to approximate expression (2), an overview over different methods and empirical evaluations can be found in [2].

In our work, we concentrate on the CMIM Algorithm (Conditional Mutual Information Maximisation, [4]) to illustrate the effects of early stopping. CMIM is reported to be faster than other mutual information based methods while having competitive predictive power [2].

The solution CMIM proposes in order to estimate (2) is to neglect higher order interactions between the features. As

$$I(X_n; Y | X_{\nu(1)}, \ldots, X_{\nu(K)}) \leq I(X_n; Y | X_{\nu(K)})$$

holds for any $k \leq K$, we can bound the term on the left by the minimum of the terms on the right over all $k \leq K$. We obtain the CMIM procedure:

$$\nu(1) = \text{ArgMax}_n I(X_n; Y)$$  \hspace{1cm} (1)

$$\nu(k) = \text{ArgMax}_n \{ I(X_n; Y | X_{\nu(1)}, \ldots, X_{\nu(k-1)}) \}$$  \hspace{1cm} (2)

Thus, we select the feature that adds most information to any (single) feature from the set of features.$\nu(k)$ is updated in each iteration according to the chosen feature. We noticed that in late iterations the updates change only little, but the calculation of the updates is computationally very expensive. Therefore we propose to stop the update of the scores and to replace the $\text{Min}_{1 \leq k \leq g}$ by $\text{Min}_{1 \leq k \leq K}$ for some $g < K$. This makes the expression at iteration $g$ independent of $k$ and allows us to select all remaining features without further calculation of mutual information. The pseudocode of CMIM with early stopping is given in Figure 1.

![Figure 1. CMIM with early stopping](image)

2.4. Computational Complexity

Naive implementation. The part of CMIM with early stopping that is dominating in complexity are the $N \times g$ evaluations of $I(X_n; Y | X_{\nu(k)})$. Estimation of conditional mutual information can be done with linear complexity in the number of samples, for instance by using binning methods [8]. Therefore, the final complexity of CMIM with early stopping is $O(NgM)$. If the if statement in Algorithm 1 in line 9 and 13 is left out, one
obtains the original CMIM algorithm with a complexity $\Omega(\mathcal{NKM})$. We will see in the experimental part that without losing too much information, $g$ can be chosen one order of magnitude smaller that $K$, so that the gain in computational time is considerable.

**Fast CMIM implementation.** We used an implementation of CMIM due to [4] that performs a lazy evaluation of the scores, i.e. does not update scores of features one can already be sure will not be selected in that iteration. This exploits the fact that the individual scores can only decrease along the iterations due to the min operator. The result is exactly the same as for the slower naive version which updates all scores (see [4] for further details) and one can apply early stopping equally to both implementations.

As a result, the number of calls to the conditional mutual information estimator under lazy evaluation is not constant for different iterations. We actually expect that later iterations become progressively more costly, as evaluations that have been postponed at first become necessary. Figure 2 shows the cumulative number of calls to the MI estimator for the MNIST data considered in Section 3 and supports this: on average, later iterations are computationally more costly than earlier rounds. In contrast, Figure 3 shows the cumulative number of scores that were actually modified in the different iterations of CMIM. Obviously the number of modified scores in later rounds decreases substantially. Both of these observations support the use of early stopping, as later iterations become more costly but less efficient.

![Figure 2. Cumulative number of calls of the conditional mutual information estimator along iterations.](image)

![Figure 3. Cumulative number of decreased scores along iterations.](image)

### 3. Experimental Results

#### 3.1 Experimental Setup

Our methods are evaluated on multiclass image classification tasks by comparing the test error of a classifier (here AdaBoost.MH [9] with a number of iterations belonging to \{400, 800, 1600\}) trained on the selected features. The features for the image data were computed using a library of feature extractors that were collaboratively designed in the MASH Project [1]. We consider as datasets: first, a subsample of the handwritten digit MNIST dataset [7] (6000 samples, 10 classes, 48416 features), and second, 2 classes subsampled from the CIFAR-10 dataset [6] (1979 samples, 8185 features). Note that our purpose is not to beat the state of the art on these datasets: we used a 10% subsample of the total MNIST and only a 20% subsample of 2 classes of 10 for CIFAR in order to reduce the overall computational load. The goal here is to compare the performance of feature selection methods with and without early stopping on real data where the number of features is large, using (after feature selection) a classifier known to be performant. We underline some important aspects of the feature space. First, the data is not sparse when represented by these features (i.e. all features are generally non-zero for any given training image). Second, a large proportion of features contain information relative to the labels, but at the same time the feature space is highly redundant. This is a typical setting where variable selection is advisable in general.
Table 1. MNIST data. Classification error rates (and standard deviations) on test data averaged over 10 repetitions of the experiments. The learning algorithm is multiclass AdaBoost applied on decision stumps using only selected features. The last figure in each column is the performance obtained without early stopping. Best performance column-wise is in bold.

<table>
<thead>
<tr>
<th># selected feat.</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>early ↓ stopping at ↓ 400 Adaboost.MH iterations</td>
<td>10.1 (0.3)</td>
<td>2.9 (0.2)</td>
<td>9.5 (0.3)</td>
<td>3.9 (0.2)</td>
<td>2.4 (0.1)</td>
<td>2.5 (0.2)</td>
<td>8.9 (0.3)</td>
<td>2.4 (0.1)</td>
<td>2.3 (0.1)</td>
</tr>
<tr>
<td>1000</td>
<td>4.5 (0.1)</td>
<td>2.7 (0.2)</td>
<td>3.7 (0.2)</td>
<td>2.4 (0.1)</td>
<td>2.4 (0.1)</td>
<td>2.3 (0.1)</td>
<td>3.6 (0.1)</td>
<td>2.1 (0.1)</td>
<td>2.2 (0.2)</td>
</tr>
<tr>
<td>2500</td>
<td>4.4 (0.2)</td>
<td>2.6 (0.2)</td>
<td>3.7 (0.2)</td>
<td>2.4 (0.1)</td>
<td>2.4 (0.1)</td>
<td>2.3 (0.1)</td>
<td>3.6 (0.1)</td>
<td>2.1 (0.1)</td>
<td>2.3 (0.1)</td>
</tr>
</tbody>
</table>

Table 2. CIFAR data (2 classes). Setup as in the previous table.

<table>
<thead>
<tr>
<th># selected feat.</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>early ↓ stopping at ↓ 400 Adaboost.MH iterations</td>
<td>11.3 (0.5)</td>
<td>9.1 (0.3)</td>
<td>11.2 (0.5)</td>
<td>9.4 (0.5)</td>
<td>12.2 (0.6)</td>
<td>9.4 (0.5)</td>
</tr>
<tr>
<td>1000</td>
<td>11.7 (0.5)</td>
<td>9.3 (0.6)</td>
<td>11.4 (0.7)</td>
<td>9.1 (0.3)</td>
<td>11.0 (0.8)</td>
<td>8.6 (0.6)</td>
</tr>
<tr>
<td>2500</td>
<td>10.3 (0.5)</td>
<td>9.0 (0.4)</td>
<td>10.6 (0.9)</td>
<td>8.9 (0.3)</td>
<td>10.6 (0.6)</td>
<td>8.9 (0.3)</td>
</tr>
</tbody>
</table>

3.2 Results

As results we give classification errors as a function of the number of selected features, Adaboost.MH iterations and score updates in Tables 1 and 2. The results strongly support that early stopping for feature selection is advantageous. First, by stopping the CMIM score updates after 100 iterations for a final selection of 1000 or 2500 features, as discussed in Section 2.4, we reduce the computational complexity between one and two orders of magnitude with respect to the complete algorithm. Second, we notice that the classification performance with early stopping does not decrease, often it does even increase significantly. This could be due to early stopping counterbalancing an overfitting effect resulting from an estimation error in the mutual information estimation.

Acknowledgements. This work was supported by the European Community’s Seventh Framework Programme FP7 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No 247022 - MASH.

References