Seminar aus Computer Vision und Mustererkennung

Dynamic Feature Selection for Media Classification

Silvana Podaras*
Florian Schober†

Abstract

TODO Bla bla bla…

1 Introduction

Thanks to the technological development of the recent years, nowadays it is possible to collect and store a vast amount of data. But these possibilities also led to new challenges when evaluating those datasets. Not only the sample size, also the dimensionality of data has increased, which results in the so called curse of dimensionality: The complexity of the analysis (regarding memory consumption and evaluation time) grows exponentially with increasing dimensionality of the samples.

To conquer this problem, various techniques which try to reduce the dimensionality were developed in the field of machine learning. Two categories can be distinguished: feature extraction and feature selection. Feature extraction tries to reduce dimensionality by combining features, thus creating new features with a lower dimensionality. Feature selection on the other hand tries to select a few, important features out of the given set.

The features produced or chosen by either one of those methods are then used for classification and analysis of the sampled data. The application of this techniques in real-life are broad and range from text analysis in social media to the analysis of genomes in bioinformatics.

The goal of this paper is to set a focus on feature selection and present some of the most common techniques and state-of-the art.

The main strategy of feature selection is to chose only a small subset of the available features for analyzing the data, which leads to higher accuracy, lower computational cost and better interpretability of the results.[1] Such a subset would optimally consist only of relevant features. What properties define a relevant feature depends on the application, but generally speaking, the optimal feature should should be discriminant when performing classification (f.ex.: by looking at a certain feature, a clear classification of the current datapoint should be possible.) and should not be redundant (f.ex., two different features should not provide the same kind of information). Possible criteria for evaluating the relevance of features will be covered in more detail in section FOO.

(evtl add: overfitting as another drawback of bg feature sets, see 2001, Feature selection with NN, p1333)

2 Methods

TODO Bla bla…es gibt flat und structured features…

2.1 Methods for Flat Features

If we look at flat features, three different types for feature selection methods can be distinguished:

- Filter methods
- Wrapper methods
- Embedded methods

In this section, a short introduction on those three categories will be given, including a discussion of their benefits and disadvantages.

*e-mail:spodaras@cg.tuwien.ac.at
†e-mail:fschober@cg.tuwien.ac.at

2.1.1 Filter Methods

As the name suggests, filter methods try to filter out relevant features from not-relevant ones. This is done by considering only the actual dataset, the properties of the classifier which will be used afterwards are ignored.

Thus, filter methods are independent of the classification algorithm: Any classifier can be used on the filtered features, which allows to try out different algorithms on the feature set.

This can be seen as an advantage, but on the other side, complete independency can also have its drawbacks. As characteristics of the specific classifiers are not considered, it is not clear which classifier works best with the selected subset. Thus the selected subset does not guarantee optimal performance of the classifier.

Quadratic programming Feature Selction: http://jmlr.org/papers/volume11/rodriguez-lujan10a/rodriguez-lujan10a.pdf

Fisher Score TODO Bla bla bla...

Information Gain TODO Bla bla bla...

Min. Redundancy - Max. Relevance TODO Bla bla bla...

Relief The original algorithm (Relief) was presented by Kira and Rendell [Kira and Rendell 1992], and can be used for binary classification problems. The algorithm is efficient (it runs in polynomial time), robust and tolerant to noisy data, but the quality of the results depends on the number of iterations. The basic idea is to use a weight vector, which has as many entries as there are features. In each iteration, an instance/variable x is chosen at random. Then the near-hit and near-miss instance of the remaining set in relation to x are selected: This is the most similar instance from the same class as x, and respectively the most similar instance of the different class. The euclidean distance between x and the near-hit and near-miss is calculated to determine if a feature is relevant or not.

Intuitively, if a feature is relevant, the near-hits should be closer than the near-misses on average, while for irrelevant features, near-hits and near-misses are independent from each other. For each feature f_i, the corresponding weight w_i in the weight-vector is updated dependent on those distances by applying equation 1:

$$w_i = -(x_i - nearhit)^2 + (x_i - nearmiss)^2 \tag{1}$$

Relevant features thus score values larger than zero, while irrelevant features become zero or negative. After a desired number of iterations, the features whose weights have a value above a certain threshold are chosen for the classification or further processing.

The original algorithm was further developed by [Kononenko et al. 1997]. ReliefF is extension so that the algorithm is able to handle incomplete data and multiple-class problems. RRELIEFF (Regressional ReliefF) [Robnik-Sikonja and Kononenko 1997] adapts the algorithm to handle also linear regression problems.

The family of Relief-algorithms and eventual adaptations have successfully been used on feature selection problems in the recent years. Moore used an adaptation called Tuned ReliefF (TuRF) for genetic analysis [Moore and White 2007], where the worst features are removed systematically in each iteration. By removing a fixed number of features in each iteration, the accuracy of weight-estimation is increasing compared to the results of the original ReliefF algorithm. Eppstein and Haake [Eppstein and Haake 2008] criticize that for truly genome-wide association analysis, where a huge number of features is used (up to hundreds of thousands), all the proposed methods don't scale well and the estimated weights become basically random values. Very large scale ReliefF (VLSReliefF) tries to overcome this limitation by simply applying Relief on subsets of the features, and then combining the results to gain the weights for all features. This process can be paralellized on the GPU to speed it up [] (Bibtex Quote? http://worldcomp-proceedings.com/proc/p2015/PDP2993.pdf)

(Evtl. other application: One application was automated classification of websites [Jin et al. 2007].)

2.1.2 Wrapper Methods

In contrast to filter methods, wrapper methods are dependent on the classifier used after the feature selection. The feature-set is selected so that it fits the biases and heuristics of the classifier as good as possible.

This is achieved by sequentially applying two steps:

- A search routine selects a set of features
- The selected subset is evaluated with the desired classifier

When choosing a search routine, one should take the structure of the search space into account. Especially its size plays a crucial role, as it grows exponentially with increasing the number of features. (The more features there are, the more possible subsets can be created.) As real-life applications can involve thousands of features, the search can not be done randomly, but has to follow a certain strategy. Typical algorithms used are hill-climbing, best-first, or greedy-search, to name a few.

When a potential subset is found, its suitability for the desired classifier has to be evaluated. This can be done for example by simply using a validation set, or by performing cross-validation.

The search and evaluation steps are repeated until a desired quality of the subset is reached. The first step will produce a set of features, which then are used by the classifier. The feature set with the best results (e.g. highest estimation values) will then be used for the actual classification task.

The major drawback of this approach is the computation time needed, as the subsequent search- and evaluation steps are computationally expensive. Compared to filter methods, better performance and more accurate estimates are achieved for the chosen classifier. Also, as the selected subset is dependent on the classifier, it cannot be used with any classifier, as the results will be biased.

This chapter will go more in-depth about different search-techniques, before discussing validation-strategies shortly.

Hill Climbing Hill climbing, sometimes also referred to as ”greedy search” or ”steepest ascend”, is probably the simplest search technique that can be applied. Starting with no features at all, an evaluation function rates the available features and adds the most relevant one to the subset. Then the procedure is repeated, each time adding the most relevant feature of the remaining set to the subset. As soon as the influence of an added feature does not improve the overall quality of the set significantly, the search is stopped. The problem with hill climbing is that it gets stuck in local maxima easily, and fails in finding a globally optimal solution. [Kohavi and John 1997]

Best First Best-first search outperforms hill-climbing, as it is a more robust technique. [Kohavi and John 1997]

Branch and Bound Branch-and-bound (BB) is a general design principle for solving discrete and combinatorial optimization problems. BB-algorithms work by building up a search tree with possible candidate solutions (branch phase). Further, a criterion function has to be designed, which evaluates the ”quality” of possible solutions represented by nodes in the tree. Then, the branches of the tree are explored to find a global optimum for the given function. To avoid that the search is exploring the whole possible space, and thus degenerating to a brute-force-approach, it is limited by so-called bounds, which prune the tree when it becomes unlikely to find an optimum in the current branch.

For feature selection, Narendra et al. [**?**] first proposed a BB-algorithm based on efficient subset selection. The root of the tree represents the full set of features, whereas leaf-nodes represent single features (or subsets of the smallest desired size). Monotonicity is required regarding the criterion function and the subsets. That means, if a current subset is below a bound, then its following child nodes are assumed to be also lower than that bound. Thus, the search in this particular branch can be aborted, and continued at another branch. Additionally, the sorting of the nodes is done with an efficient enumeration scheme to augment finding an optimal solution in a short amount of time.

The problem with BB-methods is that it is not guaranteed to perform better (faster) than exhaustive search methods. It is not granted that enough sub-trees will be pruned to speed up the search sufficiently, in worst-case, the criterion-function is evaluated for every node in the tree. Additionally, evaluating the function close to the root takes longer than evaluating it at the leaves. [**?**] proposed a fast BB approach, which tries to keep the evaluations of the criterion function at a minimum to speed up the search. The algorithm is ”learning” how the removal of features influences the functions value, and tries to ”predict” changes without doing the actual computation. Only if the predicted value comes close to a bound, the criterion function is evaluated, otherwise the algorithm continues descending in the tree.

Adaptive branch and bound for feature selection [**?**] also tries to reduce the number of evaluations of the criterion function, by applying several improvements. Already when building up the tree, the order of the nodes corresponds to the importance of the features. Then, instead of trying to descend sequentially from node to node, an adaptive, ”jumping” search method is used to avoid more-or less redundant computations of the evaluation function. Additionally, the initial bound used in the search and also the initial search level are chosen in an optimal way. (Comment: only tested on 30 features?! Still check this out...)

Genetic Algorithms (SOURCE!!! basically written down by what Silvana knows from se bachelor thesis...) Before explaining how GA can be used for for feature selection, a short introduction into the very basic concept of those algorithms has to be made. As the name implies, genetic algorithms are inspired by the way nature works in real life: parents carry specific genetic information, and a (re-)combination of this information is passed to their kids in order to create better offspring from generation to generation. GA’s imitate this behaviour by encoding the data they should work on or optimize in so-called chromosomes (which could be for examples vectors of numerical values). One is not limited to using numerical data, but this is a very common approach, as numerical values can be processed easily by the typical routines in a GA. A desired number of chromosomes is initialized at the beginning of the algorithm. They can be created randomly (f.ex by creating vectors with random values) or according to specific rules. As the algorithm starts, different operations are applied to create new chromosomes: for example, a new chromosome can be obtained by combining the values of two already existing chromosomes, or by changing random values in an already existing chromosome.

The resulting new chromosomes are then evaluated according to a fitness function, which basically measures how good a chromosome is suited for the underlying problem. The chromosomes then can be ranked, and the best ones are taken again to create new offspring. This procedure is repeated as long until a desired quality/result is achieved.

Just like branch and bound algorithms, GA are generally not suited for big feature spaces, as they try to explore the whole search space until a stopping criterion is met. This takes a lot of computation time when feature spaces are highly dimensional. The strength of GA is that they do not tend to get stuck at local optima (es for example Hill climbing algorithms do), but instead are likely to find global optima.

For feature selection, the chromosomes could correspond to subsets of the total feature set. The values in a chromosome would encode if a feature is selected or not. This can be achieved simply by using binary values: 1 indicates that a feature is selected for a possible subset, and 0 indicates that it is not selected. Mutation and recombination operators would modify the chromosomes, and a fitness function would evaluate their quality for the underlying classifier (remember, each chromosome resembles a subset of features).

But one is not limited to binary values [XY] and [BLA] used GA to find optimal kernel setting for SVM. [quote those two papers found from 2006]

TODO Bla bla bla...

Forward Selection/bachward elimination In practice, two methods which are often used for large datasets are sequential forward selection (SFS) and sequential backward elimination (SBE). When doing forward selection, no feature is selected in the beginning. More and more features are then added to the subset of selected features sequentially, until the desired feature set is achieved. SBE works exactly the other way round, starting with the full set of features first and sequentially deleting features, until a smaller subset of sufficient quality is gained. Both methods have drawbacks: Either features cannot be eliminated once they have been selected (SFS), or they cannot be selected again if they have been discarded once (SBE).

Mao [2] proposes orthogonal forward selection and backward elimination algorithms to overcome this problem. Instead of simply selecting features in a sequential way, they are first mapped onto an orthogonal space. This mapping decorrelates the features, so they can be evaluated and selected individually. After the transformation, one can link back the meaningless (sic?! WTF? why the meaningless ones?) features to the (same number of variables) in the original measurement space. works smtgh like: orthogonalize because features are de-correlated in this space, then select feature that causes may class seperability (SFS). SBS): arrange features in a matrix, lest important ones in the last rows, delete them amd re-arrange them until relevant features remain.

Using a mapping to orthogonal space proved to be very effective for features with high correlation. If the correlation between candidate features is only trivial, orthogonal transforms don't improve the results compared to existing sequential methods.

TODO Bla bla bla...

Greedy TODO Bla bla bla...

2.1.3 Embedded Methods

TODO Bla bla bla...

http://crpit.com/confpapers/CRPITV113Huda.pdf

As both filter and wrapper methods have different strengths and weaknesses, it seems only natural that research tried to combine both ideas to create so-called embedded methods, which are basically a hybrid of the former two. The strategy is to involve the classifier and its properties (like wrapper methods) while selection the features, and being computationally efficient at the same time (like filter methods).

Three types:

- Linear Classifiers (Pruning methods, SVM, logistic regression)
- ID3 / C4.5 with build-in mechanism (WTF?)
- Regularisation models

Regularisation models idea: fit the moddel, but try to have as many weightsas possible small or close to 0, so we can eliminate features with small/zero weights

TODO: more on embedded methods

2.2 Methods for Structured Features

TODO Bla Bla... es gibt die und die methods...

- Graph methods
- Tree methods
- Group methods

2.3 Graph Methods

TODO Bla bla bla...

2.4 Tree Methods

TODO Bla bla bla...

2.5 Group Methods

TODO Bla bla bla...

2.6 Methods for Streaming Features

TODO Bla Bla...

Gender Prediction on Twitter TODO Bla bla...

NOTES AND LINKS:

Dont mix up FS with the classifiers (KNN, Naive bayes, etc)

PCA, Linear discriminant = feature EXTRACTION, brauch ma ned

Feature selection for document processing

3 Discussion

TODO Bla bla bla...

4 Conclusion

TODO Bla bla bla...

References

EPPSTEIN, M., AND HAAKE, P. 2008. Very large scale relieff for genome-wide association analysis. In *Computational Intelligence in Bioinformatics and Computational Biology, 2008. CIBCB '08. IEEE Symposium on*, 112–119.

JIN, X., LI, R., SHEN, X., AND BIE, R. 2007. Automatic web pages categorization with relieff and hidden naive bayes. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, ACM, New York, NY, USA, SAC '07, 617–621.

KIRA, K., AND RENDELL, L. A. 1992. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press, AAAI'92, 129–134.

KOHAVI, R., AND JOHN, G. H. 1997. Wrappers for feature subset selection. *ARTIFICIAL INTELLIGENCE 97*, 1, 273–324.

KONONENKO, I., ŠIMEC, E., AND ROBNIK-ŠIKONJA, M. 1997. Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence 7*, 1 (Jan.), 39–55.

MOORE, J. H., AND WHITE, B. C. 2007. Tuning relieff for genome-wide genetic analysis. In *Proceedings of the 5th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Springer-Verlag, Berlin, Heidelberg, EvoBIO'07, 166–175.

NARENDRA, P. M., AND FUKUNAGA, K. 1977. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on C-26*, 9 (Sept), 917–922.

ROBNIK-SIKONJA, M., AND KONONENKO, I. 1997. An adaptation of relief for attribute estimation in regression. In *Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '97, 296–304.

SOMOL, P., PUDIL, P., AND KITTLER, J. 2004. Fast branch amp; bound algorithms for optimal feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 7 (July), 900–912.