Seminar aus Computer Vision und Mustererkennung

Dynamic Feature Selection for Media Classification

Silvana Podaras*
Florian Schober†

Abstract

(Abstract will be done in the end)

1 Introduction

Thanks to the technological development of the recent years, nowadays it is possible to collect and store a vast amount of data. High-dimensional data is omnipresent in a broad range of real-life applications, such as text analysis in social media or the analysis of genomes in bio informatics. 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 is the cause for the so called "curse of dimensionality": The complexity of the analysis (regarding memory consumption and computation time) grows exponentially with increasing dimensionality of the samples. Furthermore, many algorithms tend to overfit when too many sample points are given, which is another drawback of big feature sets. [**?**]

To conquer this problem, various techniques to reduce dimensionality were developed in the field of machine learning. Two broad categories can be distinguished: feature extraction and feature selection. In feature extraction, features are combined to create "new" features in a lower dimensional space. 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 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.[**?**] Such a subset would optimally consist only of relevant features, whereas redundant or useless features should be removed. What properties define a relevant feature depends on the application, but generally speaking, the optimal feature should be discriminant when performing classification (so when looking at a certain feature, a clear classification of the current data point should be possible) and should not be redundant (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.

2 Methods

Three types of features can be distinguished: flat, structured and streaming features. Dependent on the feature-type, different methods can be used for feature selection.

2.1 Methods for Flat Features

Flat features are features that are assumed to be independent. 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 data set, 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.

Fisher Score

Information Gain

Min. Redundancy - Max. Relevance

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 parallelized on the GPU to speed it up [**?**]

(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. [Narendra and Fukunaga 1977] 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. [Somol et al. 2004] 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 [Nakariyakul and Casasent 2007] 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 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. The theoretical groundwork for genetic algorithms was laid by [Holland 1992]. As the name implies, genetic algorithms try to imitate the process of natural evolution: parents carry specific genetic information, and a (re-)combination of this genes is passed to their kids in order to create "better" offspring from generation to generation.

GAs imitate this behavior by encoding the data they should work on or optimize in so-called chromosomes, which thus represent a possible solution to the underlying problem. A common choice for the data structure of a chromosome is to use a vector of numerical values, which can be processed easily by the typical routines in a GA.

A desired number of chromosomes is initialized at the beginning of the algorithm, and two different operations are applied to create new "offspring"-chromosomes: crossover (a new chromosome can be obtained by combining the values of two other chromosomes) or mutation (changing random values in an already existing chromosome).

The resulting new chromosomes are then evaluated according to a fitness function, which evaluates the "quality" of the solution a chromosome actually represents. 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.

For feature selection, chromosomes could represent 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.

The strength of GA is that they do not tend to get stuck at local optima, but instead are likely to find global optima, even in large datasets. Making an implementation efficient is the key for a successful use of GA for feature selection, as otherwise running time has to be fairly long

to gain acceptable results. Another problem is the choice of parameters for the algorithm (f.ex, when and how often to perform mutations over crossovers), which can influence the outcome to the better or the worse.

[Oh et al. 2004] criticize that GA are weak in fine-tuning local optima, and thus propose a hybrid algorithm for feature selection. While still performing a global search, classical search operations are applied on a local level, which led to significant performance improvements. Still, sequential search approaches tended to be faster regarding convergence. Parallel computing was suggested to overcome this problem, as the evaluation of single chromosomes is independent and could be done simultaneously.

[Frohlich et al. 2003] designed a very specific algorithm suited exclusively for feature-selection for SVMs. Computation time is reduced by considering already existing theoretical error bounds for this kind of classifier, instead of performing cross-validation to estimate the performance of a potential feature-subset. The genetic encoding of data is further used to simultaneously optimize parameters needed for the SVM itself. This method proved to be faster then RFE (Recursive feature elimination) when the number of features selected for a subset was not known beforehand. Furthermore, using error bounds tended less to overfitting of data compared to cross-validation.

Forward Selection/backward elimination In practice, two methods which are often used for large data sets are sequential forward selection (SFS) and sequential backward elimination (SBE).

Both technique work iteratively. Forward selection starts with an empty set, where no feature is selected in the beginning. Sequentially, one feature after the other is added to the subset, so that the new subset maximizes the quality of the subset, which is measured by some criterion function J. This is done until the set has reached a desired size. 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 a major drawback: Either features cannot be eliminated once they have been selected (SFS), or they cannot be selected again if they have been discarded once (SBE). The assumption, that the best five selected features must contain a subset of the best four selected features does not hold in practice. [Nakariyakul and Casasent 2008]

[Pudil et al. 1994] proposes Sequential forward floating selection as improvement (and, respectively, Selective backward floating elimination). A backtracking step is implemented after each sequential addition or deletion, which tries to find eventually better subsets. Both methods show admissible computation-time for small- or medium scaled problems, and perform better than other sequential methods on a variety of problems. However, it should be empathized that they do not always perform better than other methods, but at least ”good enough” on the majority of problems. For very big datasets, they are outperformed by genetic algorithms. [Kudo and Sklansky 2000]

[Mao 2004] proposes *orthogonal* forward selection and backward elimination to overcome problems that occur with SFS and SBE. Instead of simply selecting features in a sequential way, they are first mapped to an orthogonal space. This mapping decorrelates the features, so they can be evaluated and selected individually. After the selection, the features are linked back to the same number of variables in the original measurement space. 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.

2.1.3 Embedded Methods

Embeddd Methods are an approach to combine the computational efficiency of Filter Methods with the quality of Wrapper Methods. By addressing the characteristics of the classifier, Embedded Methods select features that are more suiting for classification than Filter Methods. At the same time, they are not as computational expensive as Wrapper Methods.

Embedded Methods can roughly been characterized into three categories:

- Pruning Methods
- Built-in Methods
- Regression Methods

Pruning Methods typically use all features to train a model, and eliminate features sequentially by setting each feature's coefficient to 0, while simultaneously evaluating the model to keep its performance.

Recursive Feature Elimination using Support Vector Machines is a Pruning Method, that uses SVM(Support Vector Machines) to select the most contributing features.

Since SVM-classifiers are basically a hyperplane that separates data in a high-dimensional space into positive and negative matches, the classification can be reduced to a simple inside-outside test with the plane, which is basically a dot-product with the corresponding hesse-normal-form of the plane.

This method now eliminates all components of the plane, in which the hesse-normal-form of the plane is close to 0, and therefore removing less- and non-contributing features [Brank et al. 2002].

As SVMs mostly work in a space which has a higher dimensionality than the original feature-space, the remaining features of the high dimensional space need to be mapped to features of the low dimensional space to effectively eliminate features.

Built-in Methods use an approach that is fundamentally different to other methods.

C 4.5 is a Built-in Method, that extends the ID3-method ([Salzberg 1994]).

ID3 is a Built-in Method that generates a good decision tree based on all features. Features that are not represented in the tree can then be eliminated ([Quinlan 1986]).

Regression Methods try to analytically force coefficients to be small while still fitting the model. After this computation, coefficients close to 0 can be eliminated.

In general, to classify a feature-vector, Regression Methods minimize an expression where an error term adds up with a penalty ([Tang et al. 2014]). Since the error term can usually be controlled by the user (typical error terms, such as squared distance can be used), Regression Methods differ in the penalty-term.

Some examples of Regression Methods are:

- Lasso Regularization ([Tibshirani 1996])
- Adaptive lasso ([Zou 2006])
- Bridge Regularization ([Knight and Fu 2000],[Huang et al. 2008])
- Elastic Net Regularization ([Zou and Hastie 2005])

The Lasso Regularization is the basic algorithm, that is used by other Regression Methods.

2.2 Methods for Structured Features

Other than Flat Features, Structured Features are features where a certain underlying structure is known and can be used. This approach tends to outperform Methods for Flat Features.

Methods for Structured Features can be categorized into 3 groups:

- Graph Methods
- Tree Methods
- Group Methods

2.3 Group Methods

Since it is often useful to select or discard a group of features at once, features can be grouped into feature-groups. Weights can now be assigned to whole groups by minimization techniques and groups with weights close to 0 can be eliminated. An algorithm that uses this approach is the Group Lasso.

This approach does not necessarily exclude the possibility to select single features inside feature groups as well. As a matter of fact some methods (e.g. the Sparse Group Lasso Regularization) perform feature-group selection and feature selection at once ([Tang et al. 2014]).

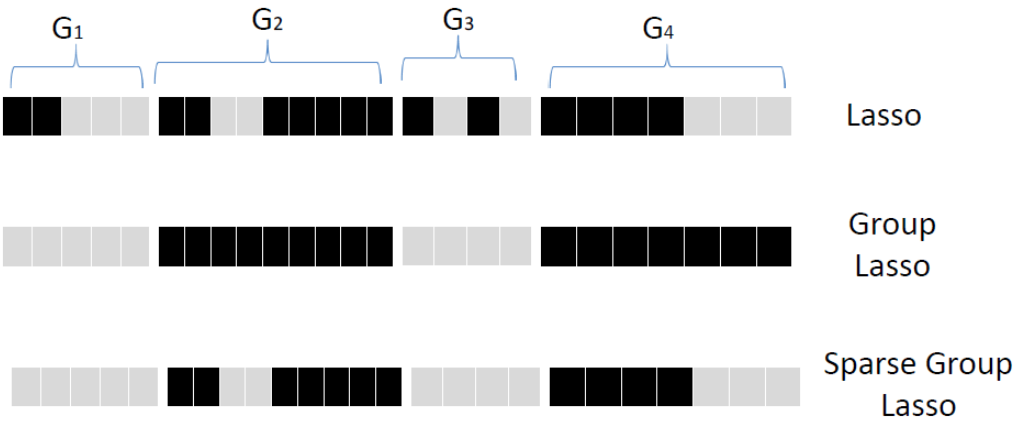

Figure 1: *Reprinted from ([Tang et al. 2014]). Illustration of Lasso, Group Lasso and Sparse Group Lasso. Features can be grouped into 4 disjoint groups G1,G2,G3,G4. Each cell denotes a feature and light color represents the corresponding cell with coefficient zero ([Tang et al. 2014]).*

In Figure 1 you can see some examples of how Lasso, Group Lasso and Sparse Group Lasso can select features.

Sometimes the given data structure might suggest overlapping feature-groups, where a feature can belong to more than one group. This case is no longer handled correctly be the Sparse Group Lasso-method. Some methods that handle this scenario are:

- [Liu and Ye 2010]
- [Kim and Xing 2010]
- [Jenatton et al. 2010]
- [Jacob et al. 2009]

2.4 Graph Methods

2.5 Tree Methods

2.6 Methods for Streaming Features

Gender Prediction on Twitter

3 Discussion

4 Conclusion

(Conclusio will be done in the end)

References

BRANK, J., GROBELNIK, M., MILIC-FRAYLING, N., AND MLADENIC, D. 2002. Feature selection using linear support vector machines. In *Proceedings of the third international conference on data mining methods and databases for engineering, finance and other fields. Bologna, Italy*.

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.

FROHLICH, H., CHAPELLE, O., AND SCHOLKOPF, B. 2003. Feature selection for support vector machines by means of genetic algorithm. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, 142–148.

HOLLAND, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

HUANG, J., HOROWITZ, J. L., AND MA, S. 2008. Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *The Annals of Statistics*, 587–613.

JACOB, L., OBOZINSKI, G., AND VERT, J.-P. 2009. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, ACM, 433–440.

JENATTON, R., MAIRAL, J., BACH, F. R., AND OBOZINSKI, G. R. 2010. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 487–494.

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.

KIM, S., AND XING, E. P. 2010. Tree-guided group lasso for multi-task regression with structured sparsity.

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.

KNIGHT, K., AND FU, W. 2000. Asymptotics for lasso-type estimators. *Annals of statistics*, 1356–1378.

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.

KUDO, M., AND SKLANSKY, J. 2000. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition 33*, 1, 25 – 41.

LEE, LIU, L. W. 2015. Very large scale relieff algorithm on gpu for genome-wide association study. 78 – 84.

LIU, J., AND YE, J. 2010. Moreau-yosida regularization for grouped tree structure learning. In *Advances in Neural Information Processing Systems*, 1459–1467.

MAO, K. 2004. Orthogonal forward selection and backward elimination algorithms for feature subset selection. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34*, 1 (Feb), 629–634.

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.

NAKARIYAKUL, S., AND CASASENT, D. P. 2007. Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognition Letters 28*, 12, 1415 – 1427.

NAKARIYAKUL, S., AND CASASENT, D. P. 2008. Improved forward floating selection algorithm for feature subset selection. In *Wavelet Analysis and Pattern Recognition, 2008. ICWAPR '08. International Conference on*, vol. 2, 793–798.

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.

OH, I.-S., LEE, J.-S., AND MOON, B.-R. 2004. Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 11 (Nov), 1424–1437.

PUDIL, P., NOVOVIČOVÁ, J., AND KITTLER, J. 1994. Floating search methods in feature selection. *Pattern Recogn. Lett. 15*, 11 (Nov.), 1119–1125.

QUINLAN, J. R. 1986. Induction of decision trees. *Machine learning 1*, 1, 81–106.

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.

SALZBERG, S. L. 1994. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning 16*, 3, 235–240.

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.

TANG, J., ALELYANI, S., AND LIU, H. 2014. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, 37.

TANG, J., ALELYANI, S., AND LIU, H. 2014. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, 37.

TIBSHIRANI, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

VERIKAS, A., AND BACAUSKIENE, M. 2002. Feature selection with neural networks. *Pattern Recogn. Lett. 23*, 11 (Sept.), 1323–1335.

ZOU, H., AND HASTIE, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2, 301–320.

ZOU, H. 2006. The adaptive lasso and its oracle properties. *Journal of the American statistical association 101*, 476, 1418–1429.