Computer Vision and Pattern Recognition Seminar

Dynamic Feature Selection for Media Classification

Silvana Podaras*
Florian Schober†

Abstract

The task of automatically processing and classifying digital media is challenging, as in many applications, high-dimensional feature-spaces have to be handled. Feature selection is one method to facilitate this process by reducing the original feature space to a subset of relevant, non-redundant features. The aim of this paper is to give an overview on the broad field of feature selection methods and present the state-of-the-art. The focus lies on filter, wrapper and embedded methods for flat features, and group methods for structured features. All presented methods can not only be used in conjunction with digital media, but also with other types of digital data.

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

Contents

1 Introduction

Automatically classifying and analyzing big amount of digital content has become a common task in the last decades. In case of digital media content, data is already present in quantified form, or eventually has to be quantified before it can be processed. In either case, classification can be a challenge, as the amount of data in many applications is vast: not only that many samples are taken, but those samples often consist of many features. In machine learning, this problem is known as the so-called ”curse of dimensionality”: The complexity of the analysis grows exponentially with increasing dimensionality of the samples. Furthermore, many algorithms tend to over-fit when too much information is given, which is another drawback of big feature sets. [Verikas and Bacauskiene 2002]

Those problems are the motivation for finding methods to gain expressive representations of datasets - for example by removing redundant and irrelevant features. This allows not only a more accurate classification, but also speeds up computation time. There are different strategies on how to reduce the dimensionality of the feature set, and this paper will focus exclusively on one of them: feature selection. The core idea of feature selection is to select a relevant subset of the already existing feature set (instead of introducing new features, or mapping them to another space). The definition of ”relevant features” is of course heavily dependent on the application, but generally speaking, discriminant features are relevant, because they facilitate clear classification. Furthermore, features should not be redundant.

This paper aims to present some of the most common techniques and state-of-the art methods in feature classification. The presented methods do not only apply to media classification (e.g. document classification, text analysis in social media, video analysis), but also to a broad range of problems emerging in different scientific fields (e.g. analysis of genomes in bio informatics).

In chapter 2, a short categorization of features according to their assumed structure is given. Then, methods which are suited for the different feature types are introduced. First, flat features and filter, wrapper and embedded methods are described. Then, applicable methods for structured features and streaming features are presented. Finally, a discussion regarding the presented methods is made in chapter 3.

2 Methods

Although it is hard to make a strict classification of feature-types, in general, three three different types of features can be distinguished:

- Flat Features
- Structured Features
- Streaming Features

The problem with making a clear classification is that the features themselves, being regarded simply as numerical data gained by making measurements, have no structure per se. In fact, the same set of features can be regarded as flat or structured, for example. Dependent on how the underlying problem is modeled, a certain structure can be assumed.

The decision, which method should be taken, is therefore dependent on the assumed properties of the features. Flat Feature methods take no relationship of features into account, whereas Structured Feature methods take a given relationship and use it to preserve structural information.

Since it is not always clear how one feature is related to the other features, structures are sometimes hard to define and might also give worse results if assumed wrong. On the other hand, if the structure tends to represent the correct relationships of the given features, Structured Feature methods tend to outperform Flat Feature methods ([Tang et al. 2014]).

While Flat Feature methods and Structured Feature methods assume a given, finite feature set, Streaming Feature methods work on an infinite or growing feature set. Streaming Feature methods are a more general approach, and find applications in social media.

2.1 Methods for Flat Features

Flat features are features which are assumed to be independent of each other. No intrinsic or group-like structures are induced. For flat features, three different type of 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.

2.1.1 Filter Methods

As the name suggests, filter methods try to filter out relevant features from non-relevant ones. This is done by considering only the data set itself, while the properties of the classifier used afterward are ignored.

This makes filter methods computationally efficient and somewhat flexible, as any classifier can be used on the filtered features.

But this independence has also its drawbacks. As the characteristics of the specific classifiers are not considered, it is not known which one works best with the selected subset. Thus the selected subset does not guarantee optimal performance of the classifier.

Because features are normally evaluated independent of each other, filter methods often fail to remove redundant features. Some methods try to overcome this problem by considering possible correlations between features.

Filter methods perform two phases sequentially: First, they rank the features according to a certain metric. The features with the highest rankings are then chosen to train the classifier.

The following chapters will present some popular filter methods and the metrics they use for ranking features.

Fisher Score When using Fisher score as a criterion for feature selection, each feature is rated and selected independent of the others by calculating its individual fisher score. The intuition behind fisher score is to select features in a way so that in the resulting "feature space", data samples of the same class are close to each other, while the distances between data points of different classes are as big as possible. Algorithmically, this filter process is relatively easy to perform: after computing the fisher score for each feature, the features with the highest scores are selected. (How many features are chosen and the definition of when a score is high enough is an open choice and dependent on the application design.)

Despite its relative simplicity, this method has a major drawback. As no eventual correlation between the features is considered, it is likely that the optimal subset is not selected. For example, if two features score relatively low individually, they eventually would score high in combination and cause better classification results. For the same reason, fisher score cannot handle redundant features. When selecting two features, it is possible that the same accuracy in classification could be achieved with only one of them.

Generalized Fisher score (GFS) is an attempt to overcome this problem. Instead of evaluating one feature after the other individually, several features are considered simultaneously. GFS was tested and compared with other state-of-the art methods, including Fisher score, and proved to perform better in applications like face recognition and number recognition. For face recognition, GFS selected features (in this case pixels) with an asymmetric distribution over the whole face-area, while fisher score selected many pixels which where clustered in non-face areas. [Gu et al. 2012]

Mutual information based Mutual information is an approach to measure a variables dependency on another variable. No mutual information means they are independent, high mutual information means they are dependent on each other.

Some concrete algorithms that are based on this approach are:

- Min redundancy max relevance
- Information gain

Min redundancy max relevance combines measures for redundancy and relevance, and then selects features with high relevance and minimum redundancy.

Information gain measures the amount of information a feature provides when it comes to class prediction. If a feature has good discriminative properties it is chosen.

Relief The original Relief-algorithm 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 is dependent on the number of iterations performed.

The method operates on the whole trainingset of samples and the features each sample consists of. A vector of weights is used, where each weight represents a feature. In each iteration, one of the samples (further referred to as x) is chosen at random. Then the near-hit and near-miss instance of the remaining set in relation to x are selected: The first is the most similar instance from the same class as x, and the latter is the most similar instance of the different class. The distance between the features of x and the near-hit and x and the 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 very similar to each other and are not very discriminative. For each feature f_i, the corresponding weight w_i in the weight-vector is updated dependent on the euclidean distances by applying equation 1:

$$w_i = w_i - (x_i - nearhit_i)^2 + (x_i - nearmiss_i)^2 \tag{1}$$

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 other processing routines.

The original algorithm was further improved. ReliefF [Kononenko et al. 1997] is an 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 has 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 [Lee 2015] In the context of media classification, the ReliefF algorithm was used for automated classification of websites [Jin et al. 2007].

2.1.2 Wrapper Methods

In contrast to filter methods, wrapper methods consider the properties of the classifier which will be used for classification. The feature-set is selected so that it fits the biases and heuristics of the classifier as good as possible.

Wrapper methods basically perform the following two steps iteratively:

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

The search and evaluation steps are repeated until a stopping criterion is met, for example when a desired quality of classification is reached, or until a maximum number of iterations was performed. The subset which performed best is selected to train the actual classifier, and normally, another evaluation with an independent testing set is done before actually using it for classification. (See figure 1)

When choosing a search routine, the structure and size of the search space should be taken into account. As the feature space in the majority of applications is high dimensional, it is not possible to enumerate all possible feature subsets. Greedy algorithms are a popular choice, but tend to get stuck in local optima when exploring big search spaces. in contrary, genetic algorithms are more complex, but are more likely to explore the search space thoroughly and find a global optimum.

When a potential subset is found, its performance 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 major drawback of wrapper methods is the computation time needed, as the subsequent search- and evaluation steps are computationally expensive: Every time a potential feature subset is selected, the classifier has actually to be trained with a training set and evaluated (for example by performing cross validation, or using accuracy estimation.[Kohavi and John 1997])

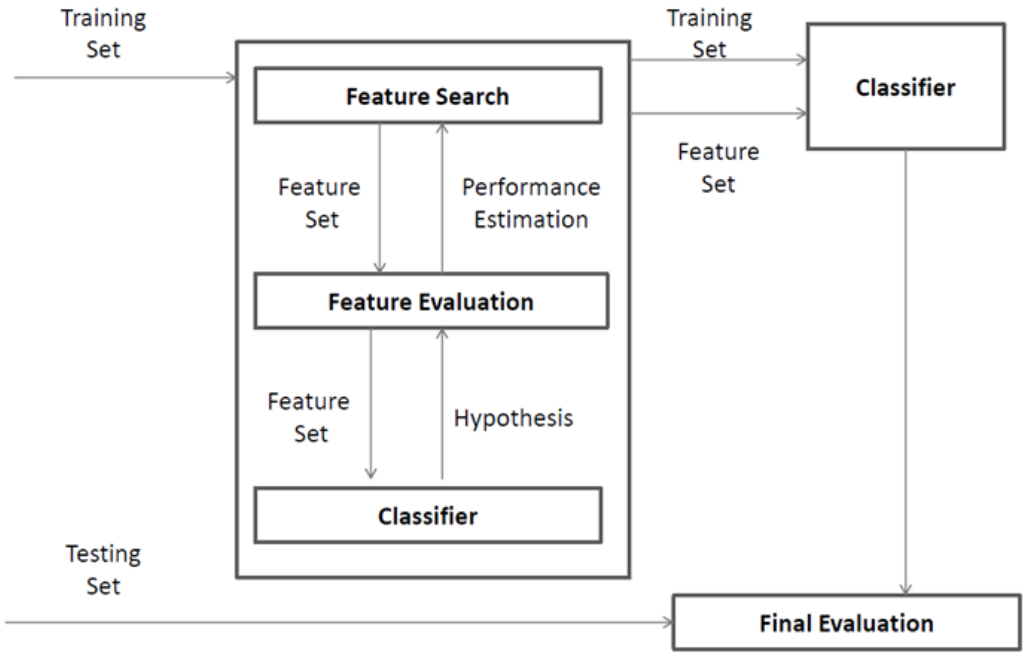

Figure 1: *Reprinted from [Tang et al. 2014]. Basic scheme of a wrapper method. Using a training set, search and evaluation steps are performed iteratively. The search-algorithm provides potential feature-subsets which are evaluated by actually training the classifier. The quality is either heuristically estimated or evaluated by cross validation. After a stopping criterion is met, the actual classifier is trained and evaluated with an independent testing set again before being used for the actual classification task.*

Because the finally selected subset is dependent on the classification algorithm, it will eventually produce biased results when being used with an arbitrary classifier.

Compared to filter methods, wrapper methods have the big advantage of selecting feature subsets which normally produce more accurate classification results. Their better performance thus justifies the long computation time.

This next chapters will go more in-depth about the different search-techniques. First, the general strategies forward selection and backwards elimination are introduced, then search by greedy algorithms and genetic algorithms are explained in more detail.

Forward Selection/backward elimination When using a greedy algorithm in the search-step of a wrapper algorithm, two strategies can be followed when selecting the features: sequential forward selection (SFS) and sequential backward elimination (SBE).

SFS starts with an empty feature set, and adds one feature in each iteration, aiming to create a better subset in each iteration. (The evaluation step is oftern done by cross validation, as mentioned in the previous chapter.) SBE works the other way round: in the first iteration, the full set of features is used and evaluated. Features are then deleted sequentially, until a smaller subset of sufficient quality is gained.

In practice, forward selection is used more often, as it is computationally more efficient to train the classifier often with smaller subsets in the beginning, than performing many trainings with big feature sets.

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). For example, if ten features are selected and perform better compared to all the previous subsets, it is still possible that one feature could be replaced by one which performs even better in combination with the other nine. In other words: the assumption, that the best x selected features must contain a subset of the best $x-1$ 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, and 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.

Hill Climbing Hill climbing, sometimes also referred to as ”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 search outperforms hill-climbing and turns out to be a more robust technique according to Kohvani [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 (the so-called ”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 they are not guaranteed to perform faster than exhaustive search methods. It is not granted that enough sub-trees will be pruned to speed up the search sufficiently, and 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] 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.

Genetic Algorithms Before giving examples of genetic algorithms used for for feature selection, a short introduction into the basic concept of those algorithms has to be made. The theoretical groundwork was laid by [Holland 1992]. As the name implies, the idea is to imitate the process of natural evolution. Parents carry specific genetic information, and a (re-)combination of those genes is passed to their kids in order to create ”better” offspring from generation to generation.

Genetic algorithms 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. For feature selection, chromosomes would 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 for example by using binary values: 1 indicates that a feature is selected for a possible subset, and 0 indicates that it is not selected. (One is not limited to use binary values though. Also, floating-point values in combination with a threshold could be used to indicate if a feature is selected or not)

In the beginning, a desired number of chromosomes is (randomly) initialized. Then two different operations are applied the them to create new "offspring"-chromosomes: crossover (a new chromosome is obtained by combining the values of two other chromosomes) or mutation (randomly changing some of the numerical values in an already existing chromosome).

The resulting new chromosomes are evaluated according to a objective function, which determines the "quality" of the solution a chromosome actually represents. The chromosomes which perform best are taken into the next iteration, where crossover and mutation are applied again. This procedure is repeated until a solution with a desired quality is achieved.

The strength of genetic algorithms 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 for feature selection, otherwise computation time would be way too long in combination with the evaluation step of the underlying wrapper method.

[Oh et al. 2004] criticize that genetic algorithms are weak in fine-tuning local optima, and thus propose a hybrid algorithm for feature selection. While still performing a global search with a genetic algorithm, 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 avoided overfitting of data compared to cross-validation.

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.

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]).

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

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 or assumed and will be used. This approach tends to outperform Methods for Flat Features in certain cases, for example genome analysis in bio-informatics ([Tang et al. 2014]).

Methods for Structured Features can be categorized into 3 groups:

- Graph Methods
- Tree Methods
- Group Methods

The decision which structure to use, is dependent on the relationship of the features. Group and tree methods basically assume simple hierarchical groupings of features, whereas graph methods build a more complex relationship-graph.

2.2.1 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]).

In Figure 2 an example is given of how Lasso, Group Lasso and Sparse Group Lasso would behave in comparison to each other when selecting 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 by the Sparse Group Lasso-method. Some methods that handle this scenario are:

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

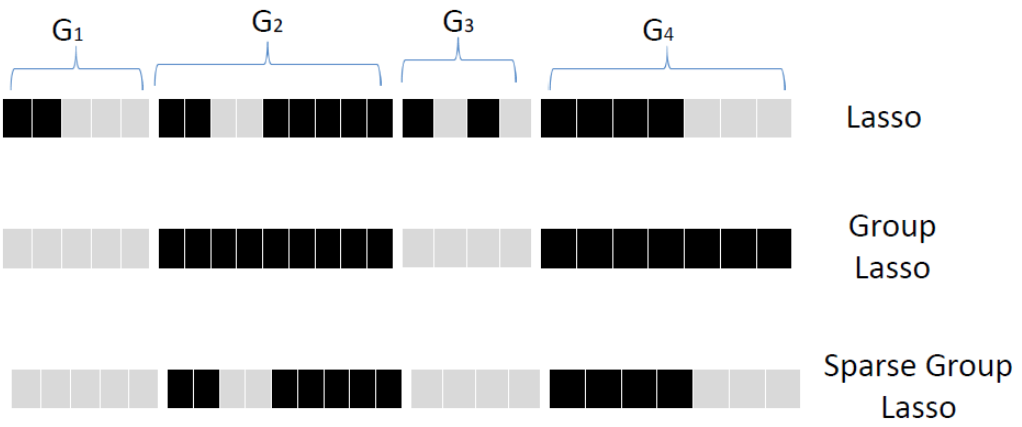

Figure 2: *Reprinted from [Tang et al. 2014]. Illustration of Lasso, Group Lasso and Sparse Group Lasso. Features can be grouped into 4 disjoint groups G1,G2,G3,G4. Each cell denotes a feature and light color represents the corresponding cell with coefficient zero ([Tang et al. 2014]).*

- [Jacob et al. 2009]

2.2.2 Tree Methods

With Tree Methods, features are assumed to have a kind of hierarchical structure, so that features can be grouped into groups, and these groups can again be grouped into groups, until there is only one group left.

This index-tree-structure can be visualized as tree, with all features being leaves (see Figure 3).

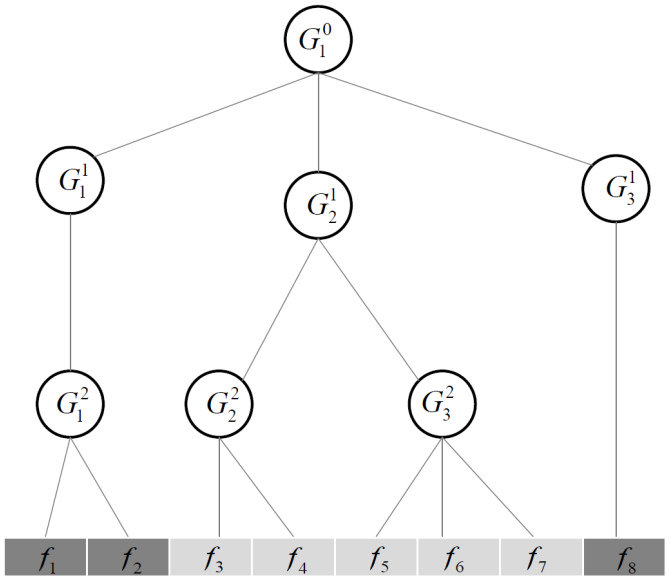

Figure 3: *Reprinted from [Tang et al. 2014]. Illustration of an index tree. E.g. Features f_1 and f_2 can be grouped into G_1^2 ([Tang et al. 2014]).*

Using index-trees, methods like the tree structured group Lasso ([Kim and Xing 2010]) can use this structure to eliminate tree-nodes on a higher level of the hierarchy and therefore eliminate many features at once.

2.2.3 Graph Methods

If data can be considered a graph-like structure, where two related features are connected with an edge, it turns out to increase the performance of classifiers to find connected components and accept/discard whole components at once ([Jacob et al. 2009]).

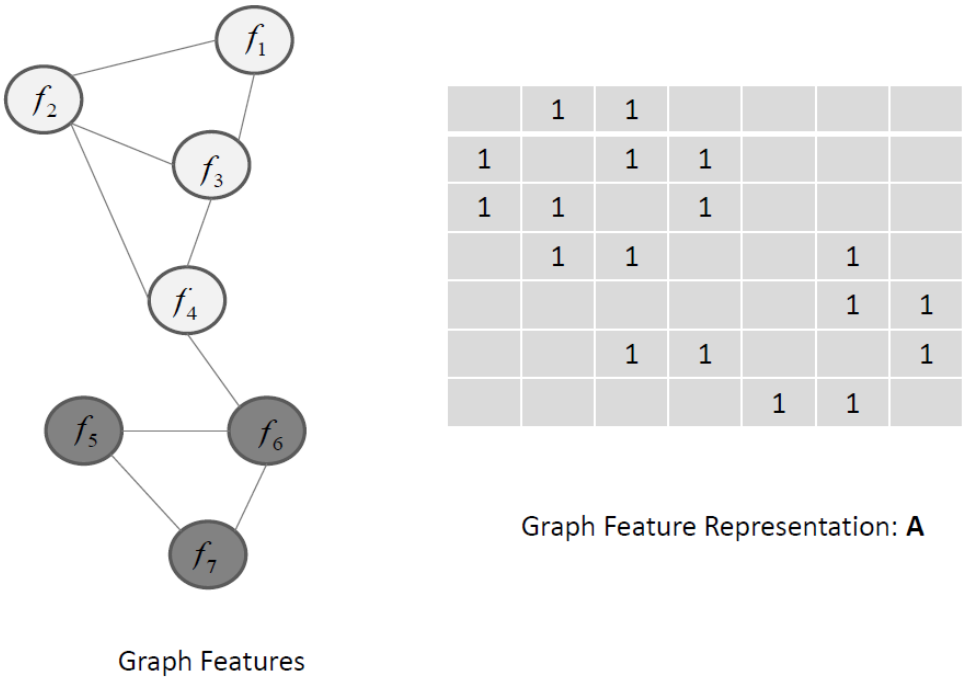

Figure 4: *Reprinted from [Tang et al. 2014]. Illustration of features forming a graph-structurem where A is the adjacency-matrix ([Tang et al. 2014]).*

Figure 4 shows a possible graph of 7 features and its adjacency matrix. In this figure the features f_5, f_6, f_7 are likely to be selected or discarded together.

The graph lasso method ([Jacob et al. 2009]) can be used formulate an energy-equation for the graph that needs to be minimized. Common energy-minimization-techniques can then assign weights to parts of the graph, which can be used to discard non-relevant features.

2.3 Methods for Streaming Features

Streaming Feature methods are operating on a growing or infinite feature set. In some applications - e.g. secial media - new features come up every minute and can or cannot be taken into account. The basic method for streaming features can be described in four steps:

- Step 1: Generate a new feature
- Step 2: Determining whether to select the new feature
- Step 3: Determining whether to remove an already selected feature
- Step 4: Continue with Step 1

Modified from [Tang et al. 2014].

Some methods that are based on this approach are:

- The Grafting Algorithm ([Perkins and Theiler 2003])
- The Alpha-investing Algorithm ([Zhou et al. 2005])

2.3.1 The Grafting Algorithm

This algorithm uses a newly generated feature, and checks wheter to add it or not. In the same step existing features are evaluated and might be removed.

The Grafting Algorithm extends the lasso method and uses its term to decide wheter or not to take a feature. If adds a term to set weights to zero if their mean classification error is too high in relation to the regularization penalty.

This evaluation is done for every feature once a new feature is added. Features with a zero-value weight are then eliminated ([Perkins and Theiler 2003]).

2.3.2 The Alpha-investing Algorithm

This algorithm takes a new feature and assigns an initial α-value to it. All α-values are stored and represent a feature. Everytime a new feature is added, a threshold is updated, based on a socalled p-value ([Zhou et al. 2005]) and based on the α-values of all selected features. Then this threshold is used to select which features stay in the selected set an which do not.

The basic idea is to have a certain wealth (the threshold) that can be spent on features. Everytime a new feature is included, wealth needs to be spent, and the possibility of another feature being unselected is higher. If another feature is then unselected, the wealth increases and can be spent on new features again.

3 Discussion

Feature selection in general is not the only field that addresses the large amount of features. Feature extraction and other mathematical or statistical methods can also reduce an existing feature-set to a new smaller feature set. A big difference between Feature selection and these methods is, that feature selection selects a real subset of features. Other methods might transfer features into other spaces or introduce artificial features that cannot be linked directly to a natural feature that is easy to understand for the user.

Structured Feature methods are often listed as better-perforing in comparison to Flat Featrue methods. Most of the listing articles are aware of the underlying structure and accept this structure as given. In these cases the assumption is usually correct, but there are cases where the underlying structure of features is not completely known and has to be defined. Defining a wrong structure can lead to worse results than Flat Feature methods.

Scalability is an issue that is not addressed by many conventional methods, because most of them need to keep the full data-set in the memory to produce good results ([Tang et al. 2014]). If the dataset is too large, only a few techniques (i.e. Streaming Methods) can still perform good.

Stability defines wheter an algorithm tends to select the same features out of a feature set if different samples are given. This is a very important property that is not yet being addressed by many algorithms ([Tang et al. 2014]). In some cases (e.g. bioinformatics) it is common to test the stability of a method with separate sample-sets and only accepting a result, if all sample-sets produce the same feature-sub-set.

Linked Data among samples is hardly considered in any feature selection method. If relationships are considered, methods usually only look at relationships of features (i.e. Structured Methods) and ignore the possibility of realted samples. When it comes to larger data-sets, more interaction between data-samples can be seen (e.g. Social Media) and should also be taken into account ([Tang et al. 2014]).

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.

GU, Q., LI, Z., AND HAN, J. 2012. Generalized fisher score for feature selection. *CoRR abs/1202.3725*.

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.

PERKINS, S., AND THEILER, J. 2003. Online feature selection using grafting. In *ICML*, 592–599.

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.

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.

ZHOU, D., HUANG, J., AND SCHÖLKOPF, B. 2005. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, ACM, 1036–1043.

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.