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)

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

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.

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 TODO Bla bla bla…

The original algorithm (Relief) was presented by Kira and Rendell [1992, src!], and can be used for binary classification problems. Its advantages are fastness and robustness, 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 and the weights in the weight-vector are then updated based on the following metric: [insert formula here] Equation (foo) takes the (euclidean) distance of the current instance x to its near-hit and near miss instances. This is the most similar instance from the same class, and respectively the most similar instance of a different class of the set compared to x. Reformulate this wonderful sentence fromWiki: Thus the weight of any given feature decreases if it differs from that feature in nearby instances of the same class more than nearby instances of the other class, and increases in the reverse case. After a desired number of iterations, the features whose weights have a value above a certain threshold are chosen for further processing/classification.

ReliefF is a further development of the algorithm made by Kononenko et al [src], to extend Relief to multiple-class problems.

ReliefFF ever further development (EPIC naming -_)

ReliefF (+ overview on Relief algorithms): http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf

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

Those two 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.

TODO: Das hier ohne Doppelpunkt ausformulieren

Search routine: The size of the search space depends on the number of features: The more there are, the more possible subsets of them exist. As real-life applications normally involve a lot of features, the search can not be done randomly, but has to follow a certain strategy, too. Typical algorithms used are hill-climbing, best-first, or greedy-search (forward selection/backward elimination paper!), to name a few.

Evaluation: The quality of the selected subset can be evaluated by different methods, using a simple validation set or cross-validation are two common methods.

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.

Wrapper methods were already introduced shortly in chapter FOO. As stated, they consist of two different steps, a search- and a evaluation-step, which are applied sequentially in each iteration until a stopping criterion (usually sufficient quality in classification) is met.

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

TODO Bla bla bla...

Hill Climbing TODO Bla bla bla...

Best Fit TODO Bla bla bla...

Branch and Bound ...usw

TODO Bla bla bla...

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 rearrange 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

(some stuff on genetic/evolutionary algorithms: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.8297rep=rep1type=

Feature selection for document processing http://thesai.org/Downloads/IJARAI/Volume3No11/Paper$_3 - A_Multistage_Feature_Selection_Model_for_Document.pdf$

3 Discussion

TODO Bla bla bla...

4 Conclusion

TODO Bla bla bla...

References