Feature Construction Methods: A Survey

Parikshit Sondhi
Univeristy of Illinois at Urbana Champaign
Department of Computer Science
201 North Goodwin Avenue Urbana, IL 61801-2302
sondhi1@uiuc.edu

Abstract

A good feature representation is central to achieving high performance in any machine learning task. However manually defining a good feature set is often not feasible. Feature construction involves transforming a given set of input features to generate a new set of more powerful features which can then used for prediction. Several feature construction methods have been developed. In this paper we present a survey of past 20 years of research in the area. We describe the major issues involved and discuss the manner in which various methods deal with them. While our understanding of feature construction has grown significantly over the years, a number of open challenges continue to remain.

1 Introduction

Engineering a good feature space is a prerequisite for achieving high performance in any machine learning task. However, given a problem, it is often not clear what the optimal feature representation should be. As a result a common approach is to use all available system variables/attributes as features (essentially a large number of relatively simple features) and leave the problem of identifying the useful feature sets to the learning model. Such a simplistic approach does not always work well. With the advancement of hardware and software technology and availability of ever more data, the number of features used by machine learning systems have grown tremendously over the past decade. While papers in early feature selection literature [Blum and Langley, 1997; Kohavi and John, 1997] rarely used more than 50-100 features, current systems frequently deal with tens of thousands to millions of features.

Consider an example of text categorization. Assume that we need to train a model for classifying a given document as spam and not spam. If we represent a document as a bag of words (unigrams), the feature space consists of a vocabulary of all unique words present in all the documents in the training set. For a collection of 100,000 to 1,000,000 documents, we can easily expect hundreds of thousands of features. If we further extend this document model to include all possible bigrams and trigrams, we could easily get over a million features.

Dealing with such large feature spaces raises complications. First, only a small number of features are actually useful. This is termed as the problem of feature irrelevance. Second, values of many features are correlated. For example if we find a keyword "win" in a spam document, we are also likely to find other keywords like "free", "prize", "money" etc. This is termed as the problem of feature interaction. Such feature interactions often introduce errors and add unnecessary complexity to the learnt concept. For instance its a well established fact that the predictive performance of classifiers like naive bayes or standard concept learning algorithms, such as c4.5 [Quinlan, 1993] degrades when the input contains features that are not directly and independently relevant to the learned concept [John *et al.*, 1994].

In order to deal with the twin problems of feature irrelevance and feature interaction, feature construction methods are used. Feature construction involves transforming a given set of input features to generate a new set of more powerful features which are then used for prediction. This may be done either to compress the dataset by reducing the number of features or to improve the prediction performance. Since the newly generated features take into account the interactions in the previous feature space, they are more meaningful and lead to more concise and accurate classifiers. There is also a better understanding of both the produced classifier and the learnt concept.

In this paper we present a survey of the research done in the area of Feature Construction over the past 20 years. While the early methods were highly restricted in the type of features they could use and the transformations they could perform, recent approaches are much more flexible.

The paper is divided into 5 sections. Section 2 presents a formal definition of the feature construction process and discusses a conceptual overview of the methods described in the literature. Section 3 presents a thorough description of different feature construction methods, categorized based on the techniques used. Section 4 briefly describes the use of feature construction methods for dimensionality reduction. Finally conclusions and open issues are discussed in section 5.

2 Feature Construction Fundamentals

Feature construction methods may be applied to pursue two distinct goals: reducing data dimensionality and improving

prediction performance. Most of the discussion in this review would mainly focus on use of feature construction for improving prediction performance, since it is often the more challenging and less understood aspect. The use of feature construction for dimensionality reduction is covered briefly in section 4. In the following subsections, we start by providing a formal definition of feature construction, present a general overview of the method and then discuss its individual components.

2.1 Formal Definition

Let:

1. $x \in X$ be a member in the input domain.
2. $y \in Y$ be some member in the output domain.
3. S be a set of training examples such that

   $S = \{(x_i, y_i)\}_{i=1..m}$.

4. S' be a set of test examples such that

   $S' = \{(x_i, y_i)\}_{i=1..m'}$.

5. Err_h be the error of hypothesis h compared to the true underlying hypothesis h_T.

We can think of each x as a fixed length vector of feature values in the original feature space i.e.

$x =< f_1, f_2, f_3...., f_n >$

where $f_i \in F_0 \; \forall \; i = 1..n$ and $n = |F_0|$

The goal of any learning machine is to learn a predictor $h : X- > Y$ from S, with low error Err_h. In the feature construction paradigm, each original feature vector x is transformed into a new feature vector

$x' = \phi(x) =< \phi_1(x), \phi_2(x), \phi_3(x)....., \phi_N(x) >$

where $\phi_i \in F_T \; \forall \; i = 1..N$ and $N = |F_T|$.

Each transformed feature value $\phi_i(x)$ is obtained by evaluating some function over all the original f_i's. We want to infer a hypothesis $h' : \phi(x)- > Y$ in the hope that its true error $Err_{h'}$ is less than Err_h. In most practical scenarios Err_h and $Err_{h'}$ will be computed by measuring the performance of h and h' on the test set S'.

2.2 Method: Overview

Conceptually any feature construction method can be thought of as performing the following:

1. Start with an initial feature space F_0 (manual feature construction).
2. Transform F_0 to construct a new feature space F_N (feature transformation).
3. Select a subset of features F_i from F_N (feature selection).
   (a) Ascertain the usefulness of F_i for the prediction task based on some utility criteria.
   (b) If some terminating criteria is achieved:
      i. Go back to step 3.
   (c) Else set F_T = F_i.
4. F_T is the newly constructed feature space.

The initial feature space F_0 consists of manually constructed features that often encode some basic domain knowledge. Different feature construction methods differ in the manner in which they implement each of these steps. Its clear that the three important aspects of any feature construction method are: (a) the method of transformation, (b) the method of selecting a subset of features F_i and (c) the utility criteria for a subset of features. These are discussed below.

2.3 Feature Transformation

When the primary goal of the feature construction is dimensionality reduction, we may stop after constructing a new feature space ($|F_N| < |F_0|$) in step 2, by applying one of the standard methods such as PCA, SVD etc. to F_0.

On the other hand when the focus is on improving prediction accuracy, we typically get $|F_N| >> |F_0|$. A common approach to generating the transformed features is to apply a set of operators (eg. $\{+, -, *\}$) on the original feature values. The choice of operators is based on domain knowledge and the type of features. For example M-of-N expressions are particularly useful for medical classification(murphy and pazzani 1991). Some of the commonly used operators include:

1. Boolean features: Conjunctions, Disjunctions, Negation etc.
2. Nominal features: Cartesian product, M of N etc.
3. Numerical features: Min, Max, Addition, Subtraction, Multiplication, Division, Average, Equivalence, Inequality etc.

Apart from these, hyperplanes, logical rules and bit strings have also been used to construct new feature spaces. The operators are usually applied iteratively. Each new feature $\phi_i \in F_N$ can therefore be represented using a tree of operators and original feature values as shown in figure 1.

A major challenge in feature transformation lies in choosing the right set of operators and applying them appropriately. Given a problem it may not be clear as to which operators are most useful. Choosing arbitrary operators may generate a large number of irrelevant features that can degrade performance [Kira and Rendell, 1992; John *et al.*, 1994]. Same goes for their manner of application. For example when the trees in figure 1 are not depth limited, the size of the resulting feature space F_N becomes infinite. This may make it hard for the selection step to find an optimum feature subset F_T. Recent methods deal with these problems via the use of annotation based approaches which allow domain experts to incorporate domain knowledge without using operators. These are discussed in section 3.4.

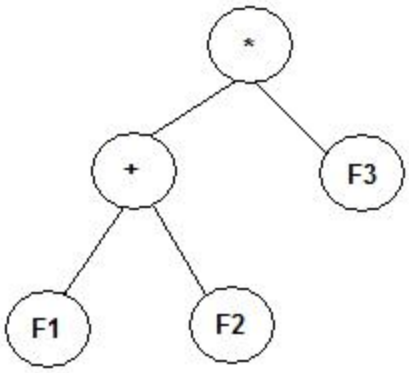

Figure 1: Tree representation of transformed feature $\phi_i \in F_N$

2.4 Feature Selection

Feature selection is a critical step in the feature construction process. Since the transformed feature space F_N is large, we need step 3 to select a subset F_T of F_N. The problem of selecting the optimal subset is NP-hard, and the methods usually perform some sort of sub-optimal greedy search. Frequently used criteria for measuring the utility of a feature space F_i include information gain, correlation coefficient, prediction accuracy on some validation set etc.

A plethora of different selection methods have been presented in the literature (see [Guyon and Elisseeff, 2003] and [Forman, 2003] for a survey). We can loosely classify these methods into three categories: filters, wrappers and embedded methods [Kohavi and John, 1997]. The classification is based on the manner in which the feature subsets are selected. A short description of these classes along with some popular methods is given below.

Filters

Filters select the feature subsets independent of the predictor. They essentially operate as a data preprocessing step before a predictor is trained. Variable ranking approaches, which involve ranking individual features using information theoretic or correlation criteria, and then constructing a subset of high scoring features, belong in this category. Filters have an advantage in that they are faster than wrappers. Moreover they tend to provide a generic (and hence insightful) ordering of features not tuned for a specific learning method. A disadvantage however is that the chosen subset may not be the best suited for the predictor to be used in the next step.

Wrappers

Wrappers are feature selection methods that use the learning method to be used for prediction as a black box to select feature subsets. These methods typically divide the training set into a train and validation set (the test set is separate). For any given feature subset, the predictor is trained on the train set and tested on the validation set. The prediction accuracy on the validation set is considered as the score of the feature subset. Thus we would ultimately want to choose the highest scoring feature subset. Due to repeated train and test cycles for every feature subset, wrappers tend to be much more computationally intensive compared to filters. The goal usually is to traverse the feature space such that the number of subsets to be tested is minimized. An obvious advantage however is that the chosen subset is tuned to the predictor.

Embedded Methods

Embedded methods combine the process of feature selection and model learning. These methods are highly specific to the learning machine. For example we could modify the objective function of an SVM [Vapnik, 1995] to also minimize the number of features along with the error [Guyon and Elisseeff, 2003]. Such methods are often fast and lead to accurate predictors. They are however not directly generalizeable to any predictor.

3 Feature Construction Methods

The task of constructing appropriate features is often highly application specific and labour intensive. Thus building automated feature construction methods that require minimal user effort is challenging. In particular we want methods that:

1. Generate a set of features that help improve prediction accuracy.
2. Are computationally efficient.
3. Are generalizeable to different classifiers.
4. Allow for easy addition of domain knowledge.

A number of different methods have been proposed. In the following subsections we classify these methods based on the techniques they use for defining and searching the feature space. The early methods were largely based on decision trees, while latter approaches have focused more on Inductive Logic Programming and Genetic Programming. GP based methods are flexible in the operators that they can use, while ILP based methods allow for easy incorporation of knowledge from diverse sources. We also present some recent annotation based approaches that eliminate the need for defining operators.

3.1 Decision Tree Related

One of the earliest feature construction algorithms is due to Pagallo [Pagallo, 1989], who created FRINGE, which adaptively enlarged the initial attribute set for learning DNF concepts. In each iteration, new features are constructed by combining pairs of features in the exisiting feature space using $negation$ and and operators. Since these two are a complete set of boolean operators, the overall feature space consists of all boolean functions of the original features. To deal with the problem of a prohibitively large new feature space, FRINGE only combines pairs of features that appear at the fringe of each of the positive branches in the decision tree. The feature generation process is repeated until no new features are generated.

CITRE [Matheus and Rendell, 1989] and DC Fringe [Yang *et al.*, 1991] are two other decision-tree based feature construction algorithms. The algorithms use conjunctions and disjunctions to combine a variety of operands such as root (selects the first two features of each positive branch), fringe

(similar to FRINGE), root-fringe (combination of both root and fringe), adjacent (selects all adjacent pairs along each branch) and all (all of the above).

A problem with decision tree based algorithms is that since, new features are added into the feature space in every iteration, the number of input features that need to be fed to the decision tree construction algorithm become very large making the process computationally inefficient. As a result certain low scoring features are discarded in every iteration. Apart from using the usual decision tree pruning techniques, all the features that were not used while constructing the decision tree may also be discarded.

All the methods discussed earlier only used boolean operators for generating features. In order to come up with a more flexible approach, Markovitch and Rosenstien [Markovitch and Rosenstein, 2002] presented FICUS, a general framework for feature construction. Their approach supplies a set of constructor functions(operators) along with the original feature set and examples to the feature construction algorithm. The algorithm then enriches the original feature space by adding additional promising features. The constructor functions may be either one or more of the commonly used operators, or may be supplied by some domain expert.

The input to FICUS is given using "feature specification language" (FSS). The user can supply information on the type (eg. nominal, continuous etc.), domain and range of basic features and constructor functions. The user can further specify a set of boolean constraints on the kind of features that may be generated or used with certain functions. Thus the new feature space is the set of all features that can be legally generated based on the FSS specification. Traversal of the search space is done via four different operators:

1. Compose: Take one or two features as input and compute a new set of features using all legal functions.
2. Insert: Take two features and create a new one by inserting one into the other.
3. Replace: Take two features and generate a new one by replacing some component of one with the other.
4. Interval: Take one feature and create new boolean features that test whether it lies within some specified range.

The search strategy is a variant of beam search. In each step two sets of features are maintained: a current set of features and a previous set of features that generated the current set via application of the above four operators. Keeping the previous set allows the system to perform one level of backtracking. The utility of a feature set, is computed based on the size and complexity of the decision tree it generates over the set of examples. The utility of individual features in the set is computed using the splitting criteria (information gain). In each iteration, the best set of features is used for generating the new feature set. The authors showed that their method achieved significant performance gains across different domains and classifiers.

While being highly flexible on one hand, FICUS had two potential drawbacks. First, their feature subset selection criteria, did not take into account feature interactions leading to a somewhat narrow search in the feature space. Second, as discussed before, given a problem, the choice of operators is often unclear making it difficult for the users to supply the correct set of constructor functions.

3.2 Genetic Programming Related

Genetic programming is an evolutionary algorithm based technique that starts with a population of individuals, evaluates them based on some fitness function and constructs a new population by applying a set of mutation and crossover operators on high scoring individuals and eliminating the low scoring ones [Koza, 1999]. In the feature construction paradigm, GP is used to derive a new feature set from the original one. Individuals are often tree like representations of features, the fitness function is usually based on the prediction performance of the classifier trained on these features while the operators can be applications specific. The method essentially performs a search in the new feature space and helps generate a high performing subset of features. The newly generated features may often be more comprehensible and intuitive than the original feature set, which makes GP-related methods well-suited for such tasks.

One of the first approaches that utilized genetic programming for feature construction was due to Vafaie and DeJong [Vafaie and de Jong, 1995] who used it for the face recognition task. Feature selection was done by evaluating feature subsets using C4.5 and GP individuals were evolved to construct new features from the selected subset. Bensusan and Kuscu [Bensusan and Kuscu, 1996] also used genetic programming to achieve an improvement in performance of learning incomplete 4-bit parity when compared with backpropagation and C4.5.

Krawiec later showed that using a feature representation enriched by GP-constructed features, can help improve classification accuracy [Krawiec, 2002]. In their approach, an individual is a set of features represented using a fixed length vector of feature definitions. Each feature is defined using an expression tree constructed by combining the values of original features and a set of operators (+,-,*,%, log, LT, GT, EQ etc.). At the time of initiation, the vector contains the variable representing original features. In every evolutionary run, mutation and crossover operators are used to generate new feature sets. The evaluation methodology is based on the wrapper approach. For calculating the fitness of each individual I, the values of all features represented by I are calculated for each training example in the dataset, thus generating a new dataset. A C4.5 decision tree inducer is then used to learn a classification model and its performance on a validation set is used as the fitness score. The authors tested their method on both real and artificial datasets and achieved impressive performance gains. Smith & Bull [Smith and Bull, 2005] later used a similar approach with two phases for creating new features for achieving performance gains with C4.5, K-Nearest Neighbour and Naive Bayes classifiers. They later showed that incorporating parsimony measures [Bojarczuk *et al.*, 2004] could help in improving readability of features involved [Smith and Bull, 2007].

Other related attempts include Otero et. al. [Otero *et al.*, 2002] used genetic programming to construct features for

training C4.5. In each round, they used information gain as a criterion and ultimately evolved a single high scoring feature which was then appended to the original feature set. Ekárt and Márkus [Ekárt and Márkus, 2003] used an interactive genetic programming approach to evolve new features that were only relevant to certain nodes in the decision tree. Every time a new node was to be constructed, the GP algorithm was used to identify new more promising features, relevant to the instances that the node needed to classify. They also used information gain as their fitness criteria. Song et. al. [Song *et al.*, 2003] combined Subset Selection with Genetic Programming for the problem of network intrusion detection. The use of Subset Selection made the training process more efficient and also helped alleviate the problem of overfitting that results due to the use of feature construction methods.

In relatively recent attempts, Shafti and Pérez [Shafti and Pérez, 2007] found that minimum description length (MDL) based fitness functions yield significantly better predictive accuracy that those based solely on entropy or error reduction. Alfred [Alfred, 2008] used a genetic-based feature construction method to improve the descriptive accuracy of data summarization method called Dynamic Aggregation of Relational Attributes (DARA).

A drawback with the evolutionary approaches is the destruction of features which may happen due to the evolutionary process. Krawiec [Krawiec, 2002] proposed a solution to this problem by hiding some of the features from the evolutionary process to preserve them from destruction.

3.3 ILP Based Methods

Inductive Logic Programming(ILP) is used for developing predicate descriptions from examples and background knowledge. ILP based feature construction methods can therefore provide a generalized framework for incorporating background knowledge into the feature generation process. The first use of first-order predicates as features was probably in the LINUS program [Lavrač *et al.*, 1991]. In subsequent literature, the problem of identifying good features with a first order representation has been covered under the propositionalization approaches [Kramer *et al.*, 2000].

In recent attempts Specia et. al. [Specia *et al.*, 2007; 2009] used an ILP based approach to improve the prediction accuracy in the word sense disambiguation task.The general idea is to apply a two step approach:

1. Feature Construction: Use ILP to learn a new set of conjunctive features.
2. Feature Selection: Select a subset of features based on their utility in the prediction process.

Their system learns clauses of the form

$h_i : Class(x, c) <- \phi_i(x)$

where $\phi_i(x) : X -> 0, 1$

is a conjunction over some feature values of an instance x and evaluates to TRUE(1) if the conjunction holds else is FALSE(0). The clauses h_i can be thought of as a rules of the form: "If some feature $\phi(x) = 1$, then the instance x must belong to class c. Once such rules are inferred from the set of examples, the feature space is enriched by adding all the individual conjunctions ϕ_i's as features. Thus they use ILP to identify a subset of conjunctions from the space of all possible conjunctions of initial features. An additional advantage of using ILP is that the conjunctive clauses h_i may also be obtained from sources other than training examples. This allows for easy incorporation of background knowledge in the feature construction process. Once the initial feature set is constructed, they then use a feature selection method on the lines of the wrapper approach discussed earlier. In each iteration, the prediction performance of the feature set is evaluated and two new feature sets are generated. One by dropping the worst performing feature, and another by adding the best performing feature in a different random sample of features from the original feature set. The resulting model showed impressive performance gains in the WSD task.

3.4 Annotation Based Approaches

Annotation based approaches allow users to provide domain knowledge in the form of annotations along with the training examples. The feature space is then learnt based on these annotations. Thus eliminating the need for defining operators.

A recent approach in this category is due to Roth and Small [Roth and Small, 2009], who proposed an interactive feature space construction protocol (IFSC). Rather than predefining a large feature space through the use of operators, their approach allows for a dynamic feature space built based on interactions between the learning machine and a domain expert. During the training process, when the learner presents some instance to the domain expert, the expert uses domain knowledge to supply additional annotations (not just the labels). The learner must now incorporate the additional knowledge via feature space modifications and learn a model. Thus a domain expert can directly encode domain knowledge without the need for defining operators. They applied their method to the problem of named entity recognition while using Semantically Related Word Lists(SRWLs) [Fellbaum, 1998; Pantel and Lin, 2002] as external knowledge. SRWLs group together lexical elements that belong to the same semantic category. For example Compass-Direction = {east, west, north,... etc.}. The protocol works as follows:

The learner starts by learning an initial hypothesis h in the original feature space F_0 and scores all instances using it. A querying function Q is then use to select the scored instances and present them to the domain expert. The goal is to choose a Q such that it minimizes the user interactions while maximizing their impact. The expert looks at the instances and marks some of the features for abstraction. For example in figure 2, the learner makes a mistake while labelling 'Chicagoland' as an organization. The expert marks features pertaining to 'west' for abstraction. Based on this interaction, the feature space is modified by replacing the features $\phi_{east}, \phi_{west}, \phi_{north}$ in the original feature space F_0 by

$$\phi_{Compass-Direction} = \phi_{east} \vee \phi_{west} \vee \phi_{north}$$

in the modified feature space.

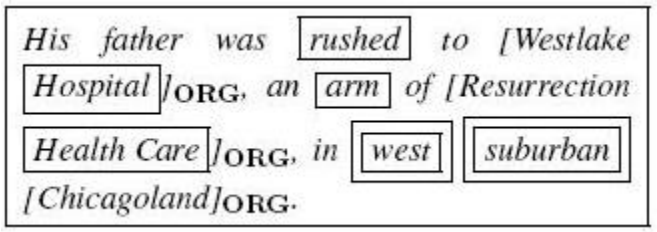

Figure 2: All lexical elements with SRWL membership are boxed. Elements used for the incorrect prediction for Chicagoland are double-boxed. The expert may select any boxed element for SRWL validation.Adapted from Roth and Small.

Another interactive method is due to Raghavan and Allen [Raghavan and Allan, 2007] who present a tandem learning algorithm for feature selection for text classification.The algorithm starts with a small number of labelled instances and in each iteration, recommends instances and features for humans to label. However unlike IFSC, in their case the feature space remains static.

Other methods that allow a domain expert to directly specify information about the feature space via annotations include [Huang and Mitchell, 2006; Zaidan and Eisner, 2007; Druck *et al.*, 2008; Zaidan and Eisner, 2008; Lim *et al.*, 2007]. In particular Lim et. al. [Lim *et al.*, 2007], propose a feature construction approach based on Explanation-Based Learning, for the problem of Chinese character recognition. In their case the knowledge was encoded in the form of 'strokes' that were used to generate the characters(or instances). Features were then generated based on presence/absence of similar strokes across different classes.

4 Feature Construction for Dimensionality Reduction

At times feature construction may be used just to obtain a better, more concise representation of data. This can be done by using certain generic feature construction methods (eg. K-means, SVD, PCA etc.) that mainly focus on transforming the data and not on addition of domain knowledge. In this section we briefly discuss two well known approaches. A more detailed discussion can be found in [Guyon and Elisseeff, 2003].

Clustering

The intuition behind clustering is to replace a group of similar features by a single representative feature. The most popular algorithms include K-means and hierarchical clustering. The features may be clustered based on their values across instances and the value of the representative feature is given by the cluster centroid. For example in a text categorization application, rather than using individual words as features, similar words may be clubbed together to form a single feature. Thus the categorization algorithm would operate on categories of words and is likely to perform better.

Singular Value Decomposition

Singular Value Decomposition (SVD) is a widely used matrix factorization method. It is at the heart of Latent Semantic Indexing (LSI) [Deerwester *et al.*, 1990], which is frequently used in information retrieval applications for characterizing relationships between words. SVD generates a new feature space in which individual features are linear combinations of features in the original space, such that they provide the best possible reconstruction of original data in the least square sense [Duda *et al.*, 2007].

5 Conclusion and Open Issues

In this paper we presented a survey of work done in the area of feature construction. From the early days of simple decision tree based feature constructors to the current interactive methods, our understanding of feature construction has vastly improved. However numerous problems still remain. Some of these are highlighted below:

Overfitting

If we consider the task of feature construction as selecting an optimal subset of features in a potentially infinite feature space, the possibility of overfitting becomes clearly apparent. In a highly expressive feature space with a comparatively small number of examples, there may be many hypotheses that are consistent with the data. Choosing the right one becomes problem. Most systems use some sort of a heuristic search methodology and tend to prefer a suboptimal subset as it is less likely to overfit. To the best of our knowledge, there is currently no literature that specifically explores this problem to identify the best search strategies.

Difficulty in Comparison

While a large number of feature construction methods have been proposed, there has been little or no comparison between them. Most papers present methods that are either meant for specific problems or are evaluated only in certain settings using specific predictors. For example it is not known how the ILP based methods would compare to the Genetic Programming based methods for some problem settings. Moreover performance of many of the ILP/GP/decision tree based methods has only been tested by pairing them up with a decision tree predictor. There has rarely been any evaluation across classifiers.

Incorporating domain knowledge

Incorporating domain knowledge into the feature construction has been and still remains a major challenge. Most methods do this by choosing a set of operators and putting constraints on the feature generation process. This has two major drawbacks. First its not always easy to encode domain knowledge in terms of operators. Second, the use of operators frequently results in a huge feature space which then needs to be searched. Recent approaches deal with this problem by allowing users to supply domain knowledge in the form of annotations which are then used for constructing the feature space. More such methods need to be explored.

References

[Alfred, 2008] Rayner Alfred. A genetic-based feature construction method for data summarisation. In *ADMA '08: Proceedings of the 4th international conference on Advanced Data Mining and Applications*, pages 39–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[Bensusan and Kuscu, 1996] Hilan Bensusan and Ibrahim Kuscu. Constructive induction using genetic programming. In *In Evolutionary Computing and Machine Learning Workshop (ICML-96*. Morgan Kaufmann, 1996.

[Blum and Langley, 1997] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.

[Bojarczuk *et al.*, 2004] Celia C. Bojarczuk, Heitor S. Lopes, Alex A. Freitas, and Edson L Michalkiewicz. A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. *Artificial Intelligence in Medicine*, 30(1):27–48, January 2004.

[Deerwester *et al.*, 1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[Druck *et al.*, 2008] Gregory Druck, Gideon S. Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In Sung H. Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat S. Chua, Mun K. Leong, Sung H. Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat S. Chua, and Mun K. Leong, editors, *SIGIR*, pages 595–602. ACM, 2008.

[Duda *et al.*, 2007] R.O. Duda, P.E. Hart, and D.G. Stork. Pattern classification, new york: John wiley & sons, 2001, pp. xx + 654, isbn: 0-471-05669-3. *Journal of Classification*, 24(2):305–307, September 2007.

[Ekárt and Márkus, 2003] Anikó Ekárt and András Márkus. Using genetic programming and decision trees for generating structural descriptions of four bar mechanisms. *Artif. Intell. Eng. Des. Anal. Manuf.*, 17(3):205–220, 2003.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[Forman, 2003] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, 2003.

[Guyon and Elisseeff, 2003] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[Huang and Mitchell, 2006] Yifen Huang and Tom M. Mitchell. Text clustering with extended user feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 413–420. SIGIR, 2006.

[John *et al.*, 1994] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[Kira and Rendell, 1992] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[Kohavi and John, 1997] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[Koza, 1999] John R. Koza. MIT Press: Cambridge, 1999.

[Kramer *et al.*, 2000] Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. pages 262–286, 2000.

[Krawiec, 2002] Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.

[Lavrač *et al.*, 1991] Nada Lavrač, Sašo Džeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with linus. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, pages 265–281, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[Lim *et al.*, 2007] Shiau Hong Lim, Li-Lun Wang, and Gerald DeJong. Explanation-based feature construction. In *IJCAI07, the Twentieth International Joint Conference on Artificial Intelligence*, pages 931–936, 2007.

[Markovitch and Rosenstein, 2002] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Mach. Learn.*, 49(1):59–98, 2002.

[Matheus and Rendell, 1989] Christopher Matheus and Larry A. Rendell. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 645–650. Morgan Kaufmann, 1989.

[Otero *et al.*, 2002] Fernando E. B. Otero, Monique M. S. Silvia, and Alex A. Freitas. Genetic programming for attribute construction in data mining. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, page 1270, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[Pagallo, 1989] Giulia Pagallo. Learning dnf by decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 639–644. Morgan Kaufmann, 1989.

[Pantel and Lin, 2002] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 613–619, 2002.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition, January 1993.

[Raghavan and Allan, 2007] Hema Raghavan and James Allan. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 79–86, New York, NY, USA, 2007. ACM.

[Roth and Small, 2009] Dan Roth and Kevin Small. Interactive feature space construction using semantic information. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 66–74, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[Shafti and Pérez, 2007] Leila Shila Shafti and Eduardo Pérez. Mdl-based fitness for feature construction. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1875–1875, New York, NY, USA, 2007. ACM.

[Smith and Bull, 2005] Matthew G. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[Smith and Bull, 2007] Matthew Smith and Larry Bull. Improving the human readability of features constructed by genetic programming. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1694–1701, New York, NY, USA, 2007. ACM.

[Song *et al.*, 2003] Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood. A linear genetic programming approach to intrusion detection. In *GECCO*, pages 2325–2336, 2003.

[Specia *et al.*, 2007] Lucia Specia, Ashwin Srinivasan, Ganesh Ramakrishnan, and Maria Das Volpe Nunes. Word sense disambiguation using inductive logic programming. pages 409–423, 2007.

[Specia *et al.*, 2009] Lucia Specia, Ashwin Srinivasan, Sachindra Joshi, Ganesh Ramakrishnan, and Maria Graças Volpe Nunes. An investigation into feature construction to assist word sense disambiguation. *Mach. Learn.*, 76(1):109–136, 2009.

[Vafaie and de Jong, 1995] Haleh Vafaie and Kenneth de Jong. Genetic algorithms as a tool for restructuring feature space representations. In *TAI '95: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, page 8, Washington, DC, USA, 1995. IEEE Computer Society.

[Vapnik, 1995] Vladimir N. Vapnik. The Nature of Statistical Learning Theory. In *Springer*, 1995.

[Yang *et al.*, 1991] Dershung Yang, Larry Rendell, and Gunnar Blix. A scheme for feature construction and a comparison of empirical methods. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 699–704. Morgan Kaufmann, 1991.

[Zaidan and Eisner, 2007] Omar F. Zaidan and Jason Eisner. Using annotator rationales to improve machine learning for text categorization. In *In NAACL-HLT*, pages 260–267, 2007.

[Zaidan and Eisner, 2008] Omar Zaidan and Jason Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In *EMNLP*, pages 31–40. ACL, 2008.