Feature Generation by Recursive Induction

A M.S.c research proposal by Lior Friedman
Under the supervision of Prof. Shaul Markovitch

November 16, 2014

1 Introduction

In recent decades, we have seen an increasing prevalence of machine learning techniques used in a wide variety of fields such as medical diagnosis, vision, and biology. Most machine learning methods assume a given set of labeled examples, represented by a set of pre-defined features. These methods have proven to be successful when a collection of good, distinguishing features is available. In many real-world applications, however, the given set of features is not sufficient for inducing a high quality classifier.

One approach for overcoming the difficulty resulting from an insufficiently expressive set of features, is to generate new features. Most feature generation algorithms produce new features by combining the original ones in various ways. For example, the LFC algorithm (Ragavan et al., 1993) combines the original feature using logical operators. The LDMT algorithm (Utgo and Brodley, 1991) uses linear combinations of the original features to construct more informative ones. The FICUS algorithm (Markovitch and Rosenstein, 2002) presents a general framework for using any set of constructors to combine features.

Recently, there has been a strong trend of utilizing *Deep Learning* (LeCun et al., 1998; Bengio, 2009) as a feature generation technique. These methods essentially form combinations and transformations on pre-defined features in a semi-supervised manner, thus yielding new, more predictive features. There are many good examples of this, such as the ones presented in Plötz et al. (2011) and Kim et al. (2013).

While feature combination has proven to enhance the performance of induction algorithms, there are many cases where a mere combination of existing features is not enough. To that extent, a different approach for generating features has been devised. This approach aims to incorporate additional knowledge from external sources in order to construct new and informative features. Gabrilovich and Markovitch (2006) for example, present a method for generating features that are based on Wikipedia concepts. Jarmasz (2012) presents a method for utilizing lexical links between words to generate features.

In the case where this external knowledge is organized as a set of relations between entities, several methods can be used. One such method is using *Inductive Logic Programming* (Muggleton, 1991) to generate features, in a process referred to as *upgrading* (Van Laer and De Raedt, 2001). This process is demonstrated through the ICL algorithm (same paper), an algorithm that uses a refinement graph to search for logical formulae which serve as features. The SGLR algorithm (Popescul and Ungar, 2007) extends this

idea to numerical features, generated using aggregation-based techniques, and then utilizes regression for inference.

In this work, we present a new methodology for using relational knowledge for feature generation. Our algorithm uses common induction algorithms over sets of feature values, resulting in classifiers which are then used as new features. Such classifier-based features can effectively capture complex relationships that are difficult to discover using existing feature-generation methods.

As an example of the potential use of such an approach, consider the following: Suppose we are given medical records of patients containing name, country of origin as well as medically relevant information. Our task is to identify patients with high risk of developing a *genetic* disease, which is more common in warm countries as well as countries with a lower GDP. The original set of features is not sufficient for inducing the target concept. Assume, however, that we have access to an external knowledge base, containing, among others, relations specifying information about countries and connecting last names with country of origin. Our method would first formulate a new induction problem where the positive examples are last names of high-risk patients, and then, through a recursive process, formulate a new problem where the positive examples are countries of origin corresponding to last names of high risk patients. The resulting classifier could then be used as a regular binary feature when classifying patients, allowing us to capture the genetic component of the disease through the last name. While traditional feature generation methods could find some relation-based trends, they would struggle to find the above relationship, as it is too complex.

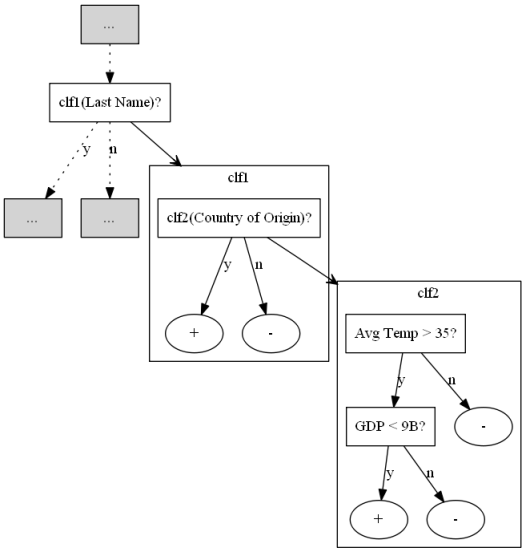

Figure 1: An single node in the decision tree. Note that the split is made by calling an additional classifier (which is created using relational data) on the value of a feature.

2 Background

One of the earliest methods of utilizing relational information, officially defined by Muggleton (1991) and used even before then in the well-known FOIL algorithm (Quinlan,

1990), is *Inductive Logic Programming(ILP)*. This approach induces a set of first-order formulae that define a good separation of the given training set. Originally, ILP methods searched the space of first-order formulae to find the appropriate classifier. Over time, effort has been made to allow the usage of traditional propositional algorithms such as ID3 (Quinlan, 1986) over this domain.

One major attempt at adding relational knowledge to induction methods was *propositionalization* (Kramer and Frank, 2000): Since we wish to allow the use of first-order predicates in propositional methods, we create possible (non recursive) first-order predicates in a process known as refinement search (van der Laag and Nienhuys-Cheng, 1998). We define a graph of possible first-order formulae by selecting an initial relation and allowing one of two possible refinement operators: The first operator is binding a single variable in the formula by assigning a constant value to it, and the second is binding a single variable using another relation.

These operators define a search space. For each formula, we can ask whether a given object satisfies it, giving us a binary query for the data. We often prefer to only use formulae with one unbound variable, as it simplifies the satisfaction check. It is important to note that each application of refinement operators yields a formula that is subsumed by the parent, meaning that any object satisfying the child formula will also satisfy its parent.

A major setback of this process is that it generates an impractically large number of features, most of which irrelevant. To this end, *upgrade* methods such as ICL (Van Laer and De Raedt, 2001) were suggested, where instead of creating predicates a-priori, we do so during the training phase, allowing us to avoid searching non-promising refinements.

As a continuation to this trend, Popescul and Ungar (2007) suggest SGLR, an upgrade method which allows the generation of nominal attributes by using aggregation operators. Essentially, instead of simply checking whether a first-order predicate is satisfied, we perform aggregation over all objects which satisfy it. In addition, SGRL offers an initial insight into the issue of improving the search process by using Akaike's information criteria (AIC, see Burnham and Anderson (2002)) to select features based on a combination of complexity and predictive power.

Newer advances in the relational approach can be seen in the field of *Statistical Relational Learning* (Blockeel, 2013; Nath and Domingos, 2014) as well as the task of *Collective Classification* (Laorden et al., 2012; Kajdanowicz et al., 2013). The general field of Statistical Relational Learning (and reasoning) deals in tasks where there is a complex, relational structure within the domain. Common tasks in the field are: Collective classification, link prediction and entity resolution between datasets. Of these, the most relevant to our domain is the task of collective classification, where relational information is used to infer labels of objects based on related objects. However, this relational information is not used for feature generation, but as a way to link objects within the target domain.

In recent years, a new resource in the form of Semantic Linked Data has begun to take form, as part of the Semantic Web project (see survey, Bizer et al. (2009)). This resource has led to the creation of several new approaches designed to utilize this new, ontology-based representation of knowledge (Lösch et al., 2012; Rios et al., 2014).

To this end, there have been several efforts in utilizing Linked Data for Feature Generation (Cheng et al., 2011; Paulheim and Fümkranz, 2012). While these techniques offer some insight towards applying the knowledge within such an extensive knowledge base

as features, they add them in an unsupervised manner, leading to the creation of an impractically large number of features, most of which are irrelevant, as seen in propositionalization methods. Due to this, they have difficulties handling complex relationships.

An interesting domain can be found in a well-known and well-explored task within the field of Natural Language Processing (NLP): the task of text classification. this task involves assigning categories or labels to documents, based on their content. The potential labels/categories are given in advance, and we have a collection of labeled documents to use as a training set. This task has many practical applications, such as spam filtering, sentiment analysis and topic identification.

Until recent years, text classification systems represented text almost exclusively as a *Bag of Words*, thus creating a vector space representation (Wu and Salton, 1981; Salton and McGill, 1983). While this method offered simplicity, it had inherent limitations in terms of representation power.

A major breakthrough came in the form of the *explicit sematic analysis* (Gabrilovich and Markovitch, 2006) method, which used semantic concepts extracted from knowledge sources such as Wikipedia as features. This technique allowed for a richer representation of the text, and has shown improvement over the Bag-of-Words representation for the task of text classification, especially in shorter texts.

The goal of our work is to utilize Linked Data as a knowledge source for automatically generating features based on meaningful semantic concepts, thus improving existing machine learning algorithms. We intend to do so in a supervised manner to allow a better guided search over candidate features, based on a combination of complexity and predictive power. We will place focus on the domain of text classification as a natural application of our approach.

3 Proposed Solution

Let O be a set of objects. Let $Y = \{0, 1\}$ be a set of labels (we assume binary labels for ease of discussion). Let $S = \{(o_1, y_1), \ldots, (o_m, y_m)\}$ be a set of labeled examples such that $o_i \in O, y_i \in Y$. Let $F = \{f_1, \ldots, f_n\}$ be a *feature map*, a set of *feature functions* $f_i : O \to Dom_i$. This definition implies a training set represented by feature vectors: $\{(\langle f_1(o_i), \ldots, f_n(o_i)\rangle, y_i) | (o_i, y_i) \in S\}$.

Given a set of relations $\bar{R} = \{R_1, \ldots, R_t\}$ with arity of n_j $(j = 1 \ldots k)$, we can assume w.l.o.g that the first argument is a key. For each relation R_j we define $n_j - 1$ new binary relations where each of the first elements is the key and the second one is another column. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be such set of binary relations, where R_i is defined over $K_i \times D_i$. These relations can thus be seen as functions $R_i : K_i \to D_i$.

Definition 3.1. A *supervised feature generation algorithm* A using relations is an algorithm that given $\langle S, F, \mathcal{R}\rangle$, creates a new feature map $F_{\mathcal{R}} = \{f'_1, \ldots, f'_l\}$.

We would like the new hypothesis space, defined over $F_{\mathcal{R}}$, to be one that is both rich enough to provide hypotheses with a lower loss than those in the original space, as well as simple enough that the learning algorithm used will be able to find such good hypotheses given training data. We would also like there to be connections between F and $\mathcal{R}$, meaning some of the feature values of features in F (when applied to objects in S) also exist within relations in $\mathcal{R}$.

We propose a general supervised feature generation algorithm using additional knowledge on feature values. Given an original feature f_i with domain Dom_i, our algorithm will formulate a new learning task trying to separate values in Dom_i appearing in positive examples of the original learning task from those appearing in negative ones. The result of the new learning task will be a classifier $h_i : Dom_i \rightarrow \{0, 1\}$ that can label feature values of f_i. We can then define a new binary feature $f'_i(x) = h_i(f_i(x))$. We name this algorithm *Binary-RFG*, as it generates binary features (For pseudo-code, see appendix A).

In order to explain how we create such h_i, let us consider a single step of the feature generation algorithm. Given a feature f_i, we define the set of all its value in the training set S as $v_i(S) = \{v|(o, y) \in S, f_i(o) = v\}$. In the intro example, for instance, $v_i(S)$ will be the set of all last names of patients. We now formulate a new learning problem with the new training set $\hat{S}_i = \{(v, label(v))|v \in v_i(S)\}$. The labeling function can be, for example, defined as the majority label: $label(v) = majority(\{y_k|\,(o_k, y_k) \in S, f_i(o_k) = v\})$.

To define a feature map over the new training set $\hat{S}$, we look for all relations in $\mathcal{R}$ where the domain of the key argument contains v_i: $\mathcal{G}(S, \mathcal{R}) = \{r \in \mathcal{R}|v_i(S) \subseteq \{x_1|(x_1, x_2) \in r\}\}$. In the intro example, one such r can be a relation mapping last names to countries of origin. We then use $\mathcal{G}$ as a feature map for $\hat{S}_i$. Solving this new learning problem on $\hat{S}_i, \mathcal{G}(S, \mathcal{R})$ yields our classifier h_i. Note that during the process of learning h_i, we can once again call the feature generation procedure to generate useful features for *that* task, hence the recursive aspect of the process.

Through the above process, we can define a tree of possible domains to search over as follows:

Definition 3.2. Domain:
S is a domain. If $\bar{S}$ is a domain, then for any function $G_{\bar{F},\bar{R}} : O \rightarrow Dom_i$, $\bar{S}' = \{(v_i, label(v_i)|(o, y) \in \bar{S}, v_i = G_{\bar{F},\bar{R}}(o)\}$ is also a domain.

The result is a process that bears some similarity to that of search over a refinement graph. Given sufficient resources, we could potentially search this tree exhaustively during the learning process. In most practical cases, however, an exhaustive search proves intractable, and we must select a search strategy over this tree.

The unique contributions of this method are as follows:

- By moving our domain space when constructing a new problem, we essentially look at the problem from another perspective, which allows for the discovery of complex relationships.
- The process of re-labeling allows noise reduction and emphasizes more general trends within the data that may be harder to otherwise notice.
- We can exploit the power of existing, well-developed learning algorithms when we create a classifier, and possibly use different ones as we take recursive steps.

We can also consider a *local* variation of our algorithm, where instead of finding features that perform well on the entirety of the data, we find features which are good on specific subsets. To do so, we use the above method to find a single binary feature, and apply it to the objects. Then, we split the objects to those labeled positive and

those labeled negative. This process can then be repeated on each subset to increase the granularity of the generated features at the cost of a smaller, potentially biased training set.

Finally, we note that the local variation yields an algorithm that can easily be made into an *anytime algorithm*: an algorithm that given more allotted computation time to learn, will produce results of higher quality (Zilberstein, 1996). You may see the anytime version for the case of decision trees as the classifier in appendix B

4 Research Questions

Listed below are several interesting and potentially challenging research questions we intend on exploring during our work:

- *Cutoff strategies and threshold criteria.* We must define good stopping criteria for determining whether a resulting classifier created by the algorithm is likely to yield a good feature. We intend to explore several criteria, including Information-Gain ratio (Quinlan, 1986) and Akaike's corrected information criterion (see Burnham and Anderson (2002)). We must also consider the case of limited resources as a consideration, evaluating the gain from an additional recursive step against the computational cost of doing so.

- *Expansions to non-binary features.* The algorithm as written generates binary features. While it is not hard to arrive at non-binary features by applying aggregation based methods as in SGLR (Popescul and Ungar, 2007), it may be wise to explore other options to do so.

- *Expansion of considered feature values.* At its current form, the algorithm is not designed to make use of numeric features. Such feature values must be handled differently than the ones we currently consider. We intend to consider several approaches to this.

- *Labeling recursive problems.* A prevalent and highly interesting question is that of labeling examples within the constructed problem. In other words, how do we define $label(v)$ for $v_i(S)$? We intend on comparing the following methods in an empirical manner:

  - Use only elements $v \in Dom_i$ which are consistent, meaning $|\{y|R(o) = v \wedge (o, y) \in S\}| = 1$. The resulting label is thus that label.
  - Take the majority of labels for all objects in $o \in O$ where $R(o) = v$. Generally any weighted averaging over the labels works here, but majority is the most immediate.
  - Choosing a middle ground: Take a weighted average of labels, but only if this average is significantly (in the statistical sense) different than the result of a random labeling. Intuitively, this means we intend on choosing elements where the majority leans significantly towards a specific label. This helps eliminate weak majority trends in the data by treating them as noise.

Another aspect to consider is that we may choose to change the set of labels itself when moving between domains.

- *Anytime expansion for other settings.* In this work, we consider the anytime setting using decision trees as a classifier. This choice yields a conceptually simple algorithm and means our generated features are highly relevant. To the best of our knowledge, there are few works regarding usage of external knowledge for anytime learning, with the most recent work found being the one presented in Lindgren (2000). Therefore, there is a highly vested interest in applying this anytime approach to other cases in an effective, efficient manner.

5 Evaluation

In order to measure the performance and significance of this approach, we intend to empirically evaluate its performance for the domain of document classification. We intend to look at several datasets, ranging from classical, often used datasets such as 20 newsgroups (Lang, 1995) and OHSUMED (Hersh et al., 1994) to more modern ones such as Reuters Corpus Volume I(RCV1) (Lewis et al., 2004) and TechTC (Davidov et al., 2004). For Semantic linked data, we first intend to use YAGO2 (Hoffart et al., 2013). We will consider the usage conceptNet5 (Speer and Havasi, 2012) as well.

We intend to compare the accuracy of our approach against the achievable accuracy without the use of feature generation, as well as the accuracy achievable by other feature generation methods such as *ESA* (Gabrilovich and Markovitch, 2006) and *FeGeLOD* (Paulheim and Fümkranz, 2012).

We will place focus on decision trees and the basic, local algorithm, an instance which allows easier adaptation as well as interpretability. In addition, we intend on measuring the anytime performance of its anytime variation compared to traditional anytime algorithms that do not make use of additional feature generation.

References

Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

Hendrik Blockeel. Statistical relational learning. In *Handbook on Neural Information Processing*, pages 241–281. Springer, 2013.

Kenneth P Burnham and David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach.* Springer, 2002.

Weiwei Cheng, Gjergji Kasneci, Thore Graepel, David Stern, and Ralf Herbrich. Automated feature generation from structured knowledge. In *Proceedings of the 20th ACM*

international conference on Information and knowledge management, pages 1395–1404. ACM, 2011.

Dmitry Davidov, Evgeniy Gabrilovich, and Shaul Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 250–257. ACM, 2004.

Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, volume 6, pages 1301–1306, 2006.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

William Hersh, Chris Buckley, TJ Leone, and David Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR94*, pages 192–201. Springer, 1994.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

Mario Jarmasz. Roget's thesaurus as a lexical resource for natural language processing. *arXiv preprint arXiv:1204.0140*, 2012.

Tomasz Kajdanowicz, Przemyslaw Kazienko, and Marcin Janczak. Collective classification techniques: an experimental study. In *New Trends in Databases and Information Systems*, pages 99–108. Springer, 2013.

Yelin Kim, Honglak Lee, and Emily Mower Provost. Deep learning for robust feature generation in audiovisual emotion recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3687–3691. IEEE, 2013.

Stefan Kramer and Eibe Frank. Bottom-up propositionalization. In *ILP Work-in-progress reports*, 2000.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

Carlos Laorden, Borja Sanz, Igor Santos, Patxi Galán-García, and Pablo G Bringas. Collective classification for spam filtering. *Logic Journal of IGPL*, page jzs030, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Tony Lindgren. Anytime inductive logic programming. In *Computers and Their Applications*, pages 439–442, 2000.

Uta Lösch, Stephan Bloehdorn, and Achim Rettinger. Graph kernels for rdf data. In *The Semantic Web: Research and Applications*, pages 134–148. Springer, 2012.

Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4): 295–318, 1991.

Aniruddh Nath and Pedro Domingos. Learning tractable statistical relational models. *StaR-AI*, 2014.

Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM, 2012.

Thomas Plötz, Nils Y Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1729, 2011.

Alexandrin Popescul and Lyle H Ungar. 16 feature generation and selection in multi-relational statistical learning. *STATISTICAL RELATIONAL LEARNING*, page 453, 2007.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3): 239–266, 1990.

Harish Ragavan, Larry Rendell, Michael Shaw, and Antoinette Tessmer. Complex concept acquisition through directed search and feature caching. In *IJCAI*, pages 946–951, 1993.

Miguel Rios, Lucia Specia, Alexander Gelbukh, and Ruslan Mitkov. Statistical relational learning to recognise textual entailment. In *Computational Linguistics and Intelligent Text Processing*, pages 330–339. Springer, 2014.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1983.

Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

Paul E Utgo and Carla E Brodley. Linear machine decision trees. Technical report, Citeseer, 1991.

Patrick RJ van der Laag and Shan-Hwei Nienhuys-Cheng. Completeness and properness of refinement operators in inductive logic programming. *The Journal of Logic Programming*, 34(3):201–225, 1998.

Wim Van Laer and Luc De Raedt. How to upgrade propositional learners to first order logic: A case study. In *Machine Learning and Its Applications*, pages 102–126. Springer, 2001.

Harry Wu and Gerard Salton. A comparison of search term weighting: Term relevance vs. inverse document frequency. *SIGIR Forum*, 16(1):30–39, May 1981. ISSN 0163-5840. doi: 10.1145/1013228.511759. URL `http://doi.acm.org/10.1145/1013228.511759`.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17 (3):73, 1996.

Appendix A Binary-RFG algorithm

Algorithm 1 Binary-RFG

$S = \{(o_i, y_i)\}$- set of labeled objects. We will mark Ob the objects and y the appropriate labels.
F- A set of feature-functions.
$\mathcal{R}$- A set of relations.
n- Number of features to generate.
cond- Stopping criteria.
classifiers- An array of the classifier to be used at every level of the recursive process.

```
function B-RFG(S, F, R, n, classifiers, cond)
    if cond(S,F) or n < 1 then
        return classifiers[0](S,F)
    end if
    featureChosen= pickFeature(S,F)
    newObjects= featureChosen(Ob)
    newLabels= label(S,newObjects)
    newF= {R_i(newObjects)|R_i ∈ R, newObjects ∩ D_i ≠ ∅}
    newClassifier= B-RFG((newObjects,newLabels), newF, R, 1, classifiers[1:], cond)
    newFeature= function(x): return newClassifier(featureChosen(x))
    return newFeature∪ B-RFG(S, F, R, n-1, classifiers, cond)
end function
```

Appendix B Anytime Local-Hierarchy-Binary-RFG Algorithm

Algorithm 2 Anytime Tree-RFG

```
function Anytime-Improved-Tree(currentTree, S, F, R, cond)
    if timeout then
        return currentTree
    end if
    nodeToImprove= pickNode(currentTree, S)
    newClassifier= B-RFG(nodeToImprove.S, F, R, 1, [decisionTree], cond)
    newFeature= function(x): return newClassifier(featureChosen(x))
    posSet= {(x, y) ∈ nodeToImprove.S|newFeature(x)==1}
    negSet= S-posSet
    newNode= makeNode(newFeature, nodeToImprove.S, posSet, negSet)
    currentTree.replace(nodeToImprove, newNode)
    return Anytime-Improved-Tree(currentTree, S, F, R, cond)
end function
```