CS224N Final Project: Hierarchical Classification Using Markov Networks

Vassil Chatalbashev
Stanford University
vasco@cs.stanford.edu

Abstract

The goal of this project is to apply Markov networks to the hierarchical classification of text documents and empirically test whether the use of hierarchies in that classifier could lead to improved prediction accuracy

1 Introduction

A lot of research in the machine learning community has been concerned with simple true/false classification, or otherwise unstructured classification. Very often, however, in domains like text classification the class labels have an inherent semantic structure, usually in the form of a hierarchy. A good example is the domain of internet search engine classification where documents are assigned to a topic hierarchy in which the children of a node are subtopics of the parent.

1.1 Project Goal

There has been a lot of research on the effectiveness of using class label hierarchies when learning (see bibliography), but for the most part the question of whether the structure of the classification labels can be successfully exploited has no general answer. Other researchers have examined the problem for various classifiers such as Naive Bayes([1], [5]) and SVMs([4]) but the problem has not been examined when Markov networks are used as the classifier. Thus the goal of my project is to empirically test the role of hierarchies in text document classification using Markov networks.

2 Related Work

Dumais[4] gives a very good overview of current research on hierarchical classification. It is important to note that some researchers provide no comparison between hierarchical and flat(i.e. one that ignores the label structure) models which makes it difficult to reason about the role of the hierarchy.

Greiner[2] notes that it is important to distinguish between the semantic contribution of the hierarchy as a source of prior knowledge and their role in changing model complexity by restricting its expressive power. The problem of representation expressivity is very important in machine learning research, and is often formalized as the *bias-variance* tradeoff. The use of the inherent semantic label structure could therefore help by reducing the model complexity and making it less susceptible to overfitting

3 Markov Networks

For more information on Markov networks and graphical models see [8]. Here we present minimal background which is relevant to the project implementation, and is largely borrowed from [6].

3.1 Representation

In the following discussion V denotes a set of random variables, and v denotes a particular assignment.

A Markov network for V defines a joint probability distribution over V. It has two components: an undirected graph which represents various dependencies among the variables, and a set of parameters associated with the graph. For any graph G, a *clique* is a set of nodes V_c, such that each V_i, V_j are connected by an edge in G.

Definition Let $G = (V, E)$ be an undirected graph with a set of cliques $C(G)$. Each $c \in C(G)$ is associated with a set of nodes V_c and a *clique potential* $\phi_c(V_c)$, which is a non-negative function $V_c \rightarrow R$. Let $\Phi = \{\phi_c(V_c)\}_{c \in C(G)}$. The Markov net (G, Φ) defines the distribution $P(v) = \frac{1}{Z}\Pi_{c \in C(G)}\phi_c(v_c)$, where Z is a normalization constant $Z = \sum_{v'} \Pi\phi_c(v'_c)$.

Note that the clique potentials are not restricted except for being non-negative, and thus do not define a probability distribution without the partition function Z. They are best interpreted as a 'compatibility' table between values of the variables in the clique.

Potentials are often expressed using a combination of a small set of indicator functions, also known as *features*, of the form $f(V_c) \equiv \delta(V_c = v_c)$. A benefit of such feature functions is that they allow for the compact representation of compatibilities.

When that representation is used the potentials are more conveniently represented in log-linear form:

$$\phi_c(v_c) = \exp \textstyle\sum_i w_i f_i(v_c) = \exp w_c \cdot f_c(v_c).$$

Note that above each feature f_i has an associated weight w_i. We can then rewrite the log-probability (using the above definitions) as:

$$\log P(v) = \sum_c w_c \cdot f_c(v_c) - \log Z = w \cdot f(v) - \log Z$$

where the last step simply involves rewriting all features and weights in a global column vector.

3.2 Inference

Since a Markov network defines a full joint probability distribution over V, we should be able to answer a variety of questions applicable to the distributions. The task of *inference* involves a number of different queries (for more details see [8],) but we will be mostly interested in answering conditional probability queries, such as "What is the probability that variable Z takes on the value 1 given that we have observed variables E in states e."

Various efficient algorithms for both exact and approximate inference have been developed, which work well for particular network topologies. In general however it can be shown that most interesting inference queries are *NP-hard* for general topology graphs.

For hierarchical classification we will be using tree-structured Markov networks, for which Belief Propagation is an efficient inference algorithm which is guaranteed to converge to the correct posterior probabilities. For more details on the algorithm see [8], [7].

3.3 Learning

In the learning task we are given some training set D where both evidence attributes X and label attributes Y are observed. Our goal is to compute weights w for the potentials Φ. To do that we can use the training set likelihood as our objective function and find a weight setting that maximizes the likelihood if the labels given the other attributes. To avoid overfitting, we introduce a prior over the weights (a zero-mean Gaussian) and use a maximum a posteriori (MAP) estimation.

The MAP objective function is concave and can be maximized using many methods such as various flavors of gradient descent. Here is the log-likelihood functionover the dataset D, including the parameter priors. C denotes a constant which comes from the Gaussian priors but is independent of the weight values:

$$L(w, D) = \sum_{d \in D}(w \cdot f(x_d, y_d) - \log Z(x_d)) - \frac{w \cdot w}{2\sigma^2} + C$$

The gradient is then:

$$\nabla L(w, D) = \sum_{d \in D}(w \cdot f(x_d, y_d) - E_{P_w}[f(x_d, Y_d]) - \frac{w}{\sigma^2}$$

The expected feature counts are computed relative to the distribution defined by the current weights P_w:

$$E_{P_w}[f(x_d, Y_d)] = \sum_{y'} f(x_d, y'_d) P_w(y'|x_d)$$

Thus to compute the gradient we need to be able to run inference on the Markov network in order to compute $P_w(y'|x_d)$.

4 Markov Networks for Hierarchical Classification

4.1 Structure

In order to adapt Markov networks to hierarchical classification we use a tree-structured network whose inner-node potentials have a special structure.

The label nodes of the network have the same edges as the hierarchy we want to represent, and each node represents a particular intermediate category, and the leaf nodes select among the actual labels. Each hierarchy node is also connected to a number of 'evidence' nodes which correspond to the words vocabulary of the documents we are classifying. The actual words used at each level are determined using feature selection (see below.)

In order to enforce only legal assignments in the hierarchy nodes of the network we make use of potentials with special structure. Each node (except the root) has $n+1$ values where n is the number of children of the node. The extra value corresponds to a *sink* state which is entered deterministically when the parent of the node is in a *sink* state or has selected a sibling of the node. As a result the only full legal assignments to the hierarchy nodes of the network correspond to valid paths from root to leaf. Figure 1 illustrates that.

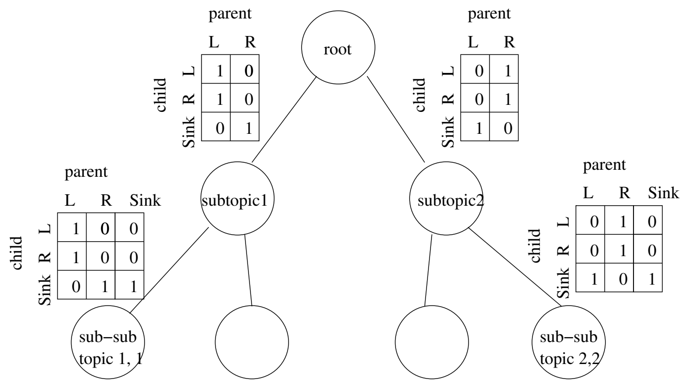

Figure 1: This is an example of the tree-structured network used for hierarchical classification. The 'word' nodes connected to each hierarchy node are ommitted. The potentials illustrate the use of the *sink* state to force a child to be in a consistent state with its parent.

4.2 Learning

As mentioned previously, each hierarchy node is connected to a number of 'word' or 'evidence' nodes which correspond to the words that are used to make a selection at that particular level of the hierarchy. 'Word' nodes take values $\in 0, 1$ to denote presence/absence of the associated word. The potentials between hierarchy and word nodes are represented using *features* and are the potentials we would like to learn. In practice we only learn the potential associated with word presence (i.e. when the value of the word node is 1.)

Note that the inter-hierarchy node potentials are almost deterministic, except when a parent has actually selected that particular child. In that case those potentials place a uniform distribution (after normalizing) on the possible values of the child, but we would also like to learn that distribution. To do that we use a common trick of adding 'dummy' word nodes that are always observed, and effectively using the learned potentials between the hierarchy nodes and the 'dummy' instead of learning just one column of the fixed potentials.

Using the notation introduced previously, here X_d corresponds to the words in a document, and Y_d are the values of all of the hierarchy nodes. So when computing the expectation term in the gradient, for each document we need to instantiate all word nodes accordingly, and run inference (using Belief Propagation) to compute the posterior on Y_d.

5 Feature Selection

In determining which words to use at each level of the hierarchy, we use feature selection using the Information Gain criterion. Yang and Pedersen [3] explored the use of different selection criteria in text categorization and found that IG is among the most effective ones.

Let $\{c_i\}_i^m = 1$ denote the set of subcategories we want to distinguish between. The for a word w, the information gain is computed as:

$$IG(w) = -\sum_i P(c_i) \log P(c_i) + P(w) \sum_i P(c_i|w) \log(c_i|w) + P(\overline{w}) \sum_i P(c_i|\overline{w}) \log P(c_i|\overline{w})$$

where $\overline{w}$ represents all words except w.

In applying feature selection in the hierarchical setting, at each hierarchy level we only take into consideration the leaf classes that are descendants the current node. We then select a fixed number of words with the highest Information Gain for each hierarchy node. Note however that at each level the original vocabulary is used.

Our use of feature selection was motivated mostly by the need to speed up the performance of the classifier, and thus we didn't get to experiment with different numbers of features and their effect on accuracy.

When selecting the initial vocabulary we also apply Zipf's law feature selection in eliminating all words which appear fewer than 10 and more than 1000 times (depending on the training set size.)

6 Implementation

The implementation phase of the project consisted of writing all the Markov network functionality from scratch. We chose to use Python for the relative speed of development, but as a result the implementation is not as fast as it could have been if implemented in a lower-level language like C++. Still it performs well enough to run a number of interesting experiments, and rewriting it in C++ would be easy and would allow us to perform a more thorough experimental evaluation.

To run the code the Python *Numeric* package is required (which is not installed on the leland systems). It is freely available on the web.

A substantial part of the implementation involved the Belief Propagation part of the code. In writing that part, we found Weiss's ([7]) formulation of the algorithm(which is equivalent to the one originally given by Pearl [8]) for the special case of pairwise Markov networks to be very useful.

For the learning part we used batch gradient descent with a decaying learning rate, which was set by repeated trial-and-error.

For the shrinkage parameters, a $\sigma = 1.7$ was found to provide the best bias-variance trade-off, but its selection was somewhat ad-hoc due to time limitations.

7 Experiments

7.1 Setup

For our experiments we used the Reuters-21578 dataset. While its documents are not labeled in a hierarchical fashion, it is easy to identify labels which tend to subsume others, and manually construct hierarchies. We used 3 different 'true' hierarchies, 2 of which(Hierarchy1 and Hierarchy 2) were borrowed from [1]. Our results are not directly comparable to the results in [1] though, since our document selection method may have been slightly different.

Figures 1,2 and 3 present the three hierarchies used. We compared the performance of the 'true' hierarchy vs. a 'flat' model and a 'fake' hierarchy (which is not necessarily 'fake' but is a rather counter-intuitive way of structuring the intermediate categories.)

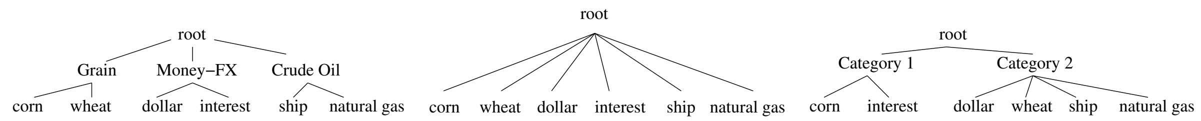

Figure 2: Hierarchy 1: 'Real' hierarchy, flat model, and the 'fake' hierarchy. 696 documents.

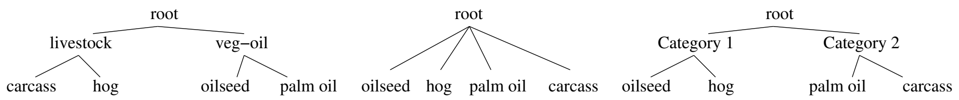

Figure 3: Hierarchy 2: 'Real' hierarchy, flat model, and the 'fake' hierarchy. 149 documents.

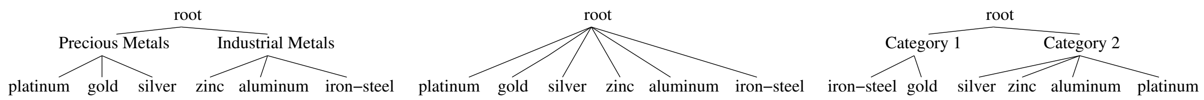

Figure 4: Hierarchy 3: 'Real' hierarchy, flat model, and the fake hierarchy. 313 documents.

Hierarchy	*Features per Node*	*Accuracy*
Hierarchy 1 'true'	40	**0.936**
Hierarchy 1 'fake'	40	0.934
Hierarchy 1 'flat'	160	0.926
Hierarchy 2 'true'	40	**0.773**
Hierarchy 2 'fake'	40	0.750
Hierarchy 2 'flat'	120	0.744
Hierarchy 3 'true'	40	0.898
Hierarchy 3 'fake'	40	**0.904**
Hierarchy 3 'flat'	120	0.879

Table 1: The top-performing classifier for each hierarchy is in bold. Results were obtained using a 70-30 split 15-fold cross-validation.

Node	*Words*
Root	'wheat', 'oil', 'dollar', 'corn', 'tonnes', 'bank', 'gas', 'gulf', 'money', 'dealers', 'stg', 'iran', 'yen', 'iranian', 'usda', 'pct', 'market', 'barrels', 'production', 'grain', 'reserves', 'agriculture'
Grain	'corn', 'wheat', 'maize', 'acres', 'gulf', 'winter', 'nil', 'conservati on', 'epac', 'eep', 'taiwan', 'south', 'differentials', 'dole', 'certificates', 'soft', 'africa', 'hard', 'red', 'ruling', 'duty', 'cash'
Money-Fx	'dollar', 'stg', 'pct', 'yen', 'money', 'mln', 'billion', 'japan', 'dollars', 'tokyo', 'trade', 'band', 'reserves', 'funds', 'bills', 'market', 'assistance', 'repurchase', 'rates', 'currency', 'dealer', 'system'
Crude Oil	'gas', 'iran', 'iranian', 'gulf', 'barrels', 'reserves', 'dlrs', 'production', 'attack', 'natural', 'mln', 'feet', 'cubic', 'year', 'drilling', 'war', 'greece', 'military', 'shipping', 'billion', 'aegean', 'tankers'

Table 2: Feature selection on Hierarchy 1.

7.2 Results

The results confirmed our intuition that the use of a hierarchy should help prediction accuracy, although further tests with more varied datasets (such as the Newsgroups or Yahoo! dataset) would be necessary. It would also be useful to see how the accuracy changes as we vary the number of features at each node.

Table 2 contains examples of some of the words selected in Hierarchy 1 at each level. Note how as we go deeper in the hierarchy the words become more and more specific.

8 Discussion

As we can see from Table 1 above for all three hieararchies used, the hierarchical classifier outperforms the flat one, and even more importantly the flat classifier is always the one with the worst accuracy which implies that the hierarchy helps regardless of how well it captures the intuitive semantic structure of the labels. In Hierarchy 3 the best performing classifier is the one with the 'fake' hierarchy but its accuracy is really close to the 'true' hierarchy classifier.

So our limited experiments show that a hierarchical classifier with the same number of features outperforms the flat classifier, and is successfully able to exploit the structure in the class labels. While more experiments on other datasets are needed to confirm that, these initial results are promising.

9 Future Work

It would be interesting to perform the same experiments on other datasets such as Newsgroups and Yahoo! and also vary the number of features selected at each node of the hierarchy to see what effect that has on the classifier accuracy vs. the baseline flat model (for example in [1] Koller and Sahami observe that hierarchical classification in their models provides the best gain for small feature sets.)

It is also worth exploring hierarchies with more complex structure, as well as hierarchical classificatoin in the multi-label setting.

10 Acknowledgements

Thanks to Ben Taskar for the valuable advice he gave me throughout the project.

References

[1] Daphne Koller, Mehran Sahami. Hierarchically classifying documents using very few words. In "Proceedings of ICML-97, 14th International Conference on Machine Learning", 1997.

[2] Dale Schuurmans, Adam Grove, Russell Greiner. On Learning Hierarchical Classifications

[3] Y. Yang, J.Pedersen. A comparative study on feature selection in text categorization. In "Proceedings of ICML-97, 14th International Conference on Machine Learning", 1997.

[4] Susan Dumais, Hao Chen. Hierarchical Classification of Web Content. In "Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval". 2000

[5] Andrew McCallum, Ronald Rosenfeld, Tom Mitchell, Andrew Y. Ng. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In “Proceedings of ICML-98, 15th International Conference on Machine Learning”. 1998

[6] Ben Taskar, P. Abbeel, D. Koller. Disscriminative Probabilistic Models for Relational Data. In “Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)” .2002

[7] Yair Weiss. Correctness of Local Probability Propagation in Graphical Models with Loops. in “Neural Computation”. 2000

[8] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. 1988.