Sparse Bayesian Classification of Predicate Arguments

Richard Johansson and **Pierre Nugues**
LUCAS, Department of Computer Science, Lund University
Box 118
SE-221 00 Lund, Sweden
{richard, pierre}@cs.lth.se

Abstract

We present an application of Sparse Bayesian Learning to the task of semantic role labeling, and we demonstrate that this method produces smaller classifiers than the popular Support Vector approach.

We describe the classification strategy and the features used by the classifier. In particular, the contribution of six parse tree path features is investigated.

1 Introduction

Generalized linear classifiers, in particular Support Vector Machines (SVMs), have recently been successfully applied to the task of semantic role identification and classification (Pradhan et al., 2005), *inter alia*.

Although the SVM approach has a number of properties that make it attractive (above all, excellent software packages exist), it also has drawbacks. First, the resulting classifier is slow since it makes heavy use of kernel function evaluations. This is especially the case in the presence of noise (since each misclassified example has to be stored as a bound support vector). The number of support vectors typically grows with the number of training examples. Although there exist optimization methods that speed up the computations, the main drawback of the SVM approach is still the classification speed.

Another point is that it is necessary to tune the parameters (typically C and γ). This makes it necessary to train repeatedly using cross-validation to find the best combination of parameter values.

Also, the output of the decision function of the SVM is not probabilistic. There are methods to map the decision function onto a probability output using the sigmoid function, but they are considered somewhat ad-hoc (see (Tipping, 2001) for a discussion).

In this paper, we apply a recent learning paradigm, namely *Sparse Bayesian learning*, or more specifically the *Relevance Vector* learning method, to the problem of role classification. Its principal advantages compared to the SVM approach are:

- It typically utilizes fewer examples compared to the SVM, which makes the classifier faster.
- It uses no C parameter, which reduces the need for cross-validation.
- The decision function is adapted for probabilistic output.
- Arbitrary basis functions can be used.

Its significant drawback is that the training procedure relies heavily on dense linear algebra, and is thus difficult to scale up to large training sets and may be prone to numerical difficulties.

For a description of the task and the data, see (Carreras and Màrquez, 2005).

2 Sparse Bayesian Learning and the Relevance Vector Machine

The Sparse Bayesian method is described in detail in (Tipping, 2001). Like other generalized linear learning methods, the resulting binary classifier has the form

$$\text{sign} f(x) = \text{sign} \sum_{i=1}^{m} \alpha_i f_i(x) + b$$

where the f_i are basis functions. Training the model then consists of finding a suitable $\boldsymbol{\alpha} = (b, \alpha_1, \ldots, \alpha_m)$ given a data set $(\boldsymbol{X}, \boldsymbol{Y})$.

Analogous with the SVM approach, we can let $f_i(x) = k(x, x_i)$, where x_i is an example from the training set and k a function. We have then arrived at the *Relevance Vector Machine* (RVM). There are however no restrictions on the function k (such as Mercer's condition for SVM). We use the Gaussian kernel $k(x, y) = \exp(-\gamma\|x - y\|^2)$ throughout this work.

We first model the probability of a positive example as a sigmoid applied to $f(x)$. This can be used to write the likelihood function $P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha})$. Instead of a conventional ML approach (maximizing the likelihood with respect to $\boldsymbol{\alpha}$, which would give an overfit model), we now adopt a Bayesian approach and encode the model preferences using priors on $\boldsymbol{\alpha}$. For each α_i, we introduce a parameter s_i and assume that $\alpha_i \in N(0, s_i^{-1})$ (i.e. Gaussian). This is in effect an "Occam penalty" that encodes our preference for sparse models. We should finally specify the distributions of the s_i. However, we make the simplifying assumption that their distribution is flat (noninformative).

We now find the maximum of the *marginal likelihood*, or "evidence", with respect to $\boldsymbol{s}$, that is

$$p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{s}) = \int P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\boldsymbol{s}) d\boldsymbol{\alpha}.$$

This integral is not tractable, hence we approximate the integrand using a Gaussian centered at the mode of the integrand (Laplace's approximation). The marginal likelihood can then be differentiated with respect to $\boldsymbol{s}$, and maximized using iterative methods such as gradient descent.

The algorithm thus proceeds iteratively as follows: First maximize the penalized likelihood function $P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\boldsymbol{s})$ with respect to $\boldsymbol{\alpha}$ (for example via the Newton-Raphson method), then update the parameters s_i. This goes on until a convergence criterion is met, for example that the s_i changes are small enough. During iteration, the s_i parameters for redundant examples tend to infinity. They (and the corresponding columns of the kernel matrix) are then removed from the model. This is necessary because of numerical stability and also reduces the training time considerably.

We implemented the RVM training method using the ATLAS (Whaley et al., 2000) implementation of the BLAS and LAPACK standard linear algebra APIs. To make the algorithm scale up, we used a working-set strategy that used the results of partial solutions to train the final classifier. Our implementation is based on the original description of the algorithm (Tipping, 2001) rather than the greedy optimized version (Tipping and Faul, 2003), since preliminary experiments suggested a decrease in classification accuracy. Our current implementation can handle training sets up to about 30000 examples.

We used the conventional one-versus-one method for multiclass classification. Although the Sparse Bayesian paradigm is theoretically not limited to binary classifiers, this is of little use in practice, since the size of the Hessian matrix (used while maximizing the likelihood and updating $\boldsymbol{s}$) grows with the number of classes.

3 System Description

Like previous systems for semantic role identification and classification, we used an approach based on classification of nodes in the constituent tree. To simplify training, we used the soft-prune approach as described in (Pradhan et al., 2005), which means that before classification, the nodes were filtered through a binary classifier that classifies them as having a semantic role or not (NON-NULL or NULL). The NULL nodes missed by the filter were included in the training set for the final classifier.

Since our current implementation of the RVM training algorithm does not scale up to large training sets, training on the whole PropBank was infeasible. We instead trained the multiclass classifier on sections 15 – 18, and used an SVM for the soft-pruning classifier, which was then trained on the remaining sections. The excellent LIBSVM (Chang and Lin, 2001) package was used to train the SVM.

The features used by the classifiers can be grouped into predicate and node features. Of the node features, we here pay most attention to the parse tree path features.

3.1 Predicate Features

We used the following predicate features, all of which first appeared in (Gildea and Jurafsky, 2002).

- *Predicate lemma*.
- *Subcategorization frame*.
- *Voice*.

3.2 Node Features

- *Head word* and *head POS*. Like most previous work, we used the head rules of Collins to extract this feature.
- *Position*. A binary feature that describes if the node is before or after the predicate token.
- *Phrase type* (PT), that is the label of the constituent.
- *Named entity*. Type of the first contained NE.
- *Governing category*. As in (Gildea and Jurafsky, 2002), this was used to distinguish subjects from objects. For an NP, this is either S or VP.
- *Path features*. (See next subsection.)

For prepositional phrases, we attached the preposition to the PT and replaced head word and head POS with those of the first contained NP.

3.3 Parse Tree Path Features

Previous studies have shown that the parse tree path feature, used by almost all systems since (Gildea and Jurafsky, 2002), is salient for argument identification. However, it is extremely sparse (which makes the system learn slowly) and is dependent on the quality of the parse tree. We therefore investigated the contribution of the following features in order to come up with a combination of path features that leads to a robust system that generalizes well.

- *Constituent tree path*. As in (Gildea and Jurafsky, 2002), this feature represents the path (consisting of step directions and PTs of the nodes traversed) from the node to the predicate, for example NP↑VP↓VB for a typical object. Removing the direction (as in (Pradhan et al., 2005)) improved neither precision nor recall.
- *Partial path*. To reduce sparsity, we introduced a partial path feature (as in (Pradhan et al., 2005)), which consists of the path from the node to the lowest common ancestor.
- *Dependency tree path*. We believe that labeled dependency paths provide more information about grammatical functions (and, implicitly, semantic relationships) than the raw constituent structure. Since the grammatical functions are not directly available from the parse trees, we investigated two approximations of dependency arc labels: first, the POSs of the head tokens; secondly, the PTs of the head node and its immediate parent (such labels were used in (Ahn et al., 2004)).
- *Shallow path*. Since the UPC shallow parsers were expected to be more robust than the full parsers, we used a shallow path feature. We first built a parse tree using clause and chunk bracketing, and the shallow path feature was then constructed like the constituent tree path.
- *Subpaths*. All subpaths of the constituent path.

We used the parse trees from Charniak's parser to derive all paths except for the shallow path.

4 Results

4.1 Comparison with SVM

The binary classifiers that comprise the one-versus-one multiclass classifier were 89% – 98% smaller when using RVM compared to SVM. However, the performance dropped by about 2 percent. The reason for the drop is possibly that the classifier uses a number of features with extremely sparse distributions (two word features and three path features).

4.2 Path Feature Contributions

To estimate the contribution of each path feature, we measured the difference in performance between a system that used all six features and one where one of the features had been removed. Table 2 shows the results for each of the six features. For the final system, we used the dependency tree path with PT pairs, the shallow path, and the partial path.

4.3 Final System Results

The results of the complete system on the test sets are shown in Table 1. The smaller training set (as mentioned above, we used only sections 15 – 18

	Precision	Recall	$F_{\beta=1}$
Development	73.40%	70.85%	72.10
Test WSJ	75.46%	73.18%	74.30
Test Brown	65.17%	60.59%	62.79
Test WSJ+Brown	74.13%	71.50%	72.79

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	75.46%	73.18%	74.30
A0	84.56%	85.18%	84.87
A1	73.40%	73.35%	73.37
A2	61.99%	57.30%	59.55
A3	71.43%	46.24%	56.14
A4	72.53%	64.71%	68.39
A5	100.00%	40.00%	57.14
AM-ADV	58.13%	51.58%	54.66
AM-CAU	70.59%	49.32%	58.06
AM-DIR	59.62%	36.47%	45.26
AM-DIS	81.79%	71.56%	76.33
AM-EXT	72.22%	40.62%	52.00
AM-LOC	54.05%	55.10%	54.57
AM-MNR	54.33%	52.91%	53.61
AM-MOD	98.52%	96.73%	97.62
AM-NEG	96.96%	96.96%	96.96
AM-PNC	36.75%	37.39%	37.07
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	76.00%	70.19%	72.98
R-A0	83.33%	84.82%	84.07
R-A1	68.75%	70.51%	69.62
R-A2	57.14%	25.00%	34.78
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	92.31%	57.14%	70.59
R-AM-MNR	40.00%	33.33%	36.36
R-AM-TMP	75.00%	69.23%	72.00
V	98.82%	98.82%	98.82

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

for the role classifier) causes the result to be significantly lower than state of the art (F-measure of 79.4, reported in (Pradhan et al., 2005)).

5 Conclusion and Future Work

We have provided an application of Relevance Vector Machines to a large-scale NLP task. The resulting classifiers are drastically smaller that those produced by the SV training methods. On the other hand, the classification accuracy is lower, probably because of the use of lexicalized features.

The results on the Brown test set shows that the genre has a significant impact on the performance.

An evaluation of the contribution of six parse tree

	P	R	$F_{\beta=1}$
Const. tree	-0.2%	-0.6%	-0.4
Partial	-0.4%	+0.4%	0
Dep. w/ POSs	-0.1%	-0.4%	-0.3
Dep. w/ PT pairs	+0.4%	+0.4%	+0.4
Shallow	-0.1%	+0.4%	+0.1
Const. subpaths	-10.9%	+2.5%	-4.5

Table 2: Contribution of path features

path features suggests that dependency tree paths are more useful for semantic role labeling than the traditional constituent tree path.

In the future, we will investigate if it is possible to incorporate the γ parameter into the probability model, thus eliminating the need for cross-validation completely. In addition, the training algorithm will need to be redesigned to scale up to larger training sets. The learning paradigm is still young and optimized methods (such as for SVM) have yet to appear. One possible direction is the greedy method described in (Tipping and Faul, 2003).

References

David Ahn, Sisay Fissaha, Valentin Jijkoun, and Maarten de Rijke. 2004. The university of Amsterdam at Senseval-3: Semantic roles and logic forms. In *Proceedings of SENSEVAL-3*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines.*

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*. To appear.

Michael E. Tipping and Anita Faul. 2003. Fast marginal likelihood maximisation for sparse bayesian models. In *9th International Workshop on AI and Statistics*.

Michael E. Tipping. 2001. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211 – 244.

R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. 2000. Automated empirical optimizations of software and the ATLAS project.