SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

Regularized Least-Squares Classification for Word Sense Disambiguation

Marius Popescu
Department of Computer Science, University of Bucharest
Str. Academiei 14
70109 Bucharest,
Romania,
mpopescu@phobos.cs.unibuc.ro

Abstract

The paper describes **RLSC-LIN** and **RLSC-COMB** systems which participated in the Senseval-3 English lexical sample task. These systems are based on Regularized Least-Squares Classification (RLSC) learning method. We describe the reasons of choosing this method, how we applied it to word sense disambiguation, what results we obtained on Senseval-1, Senseval-2 and Senseval-3 data and discuss some possible improvements.

1 Introduction

Word sense disambiguation can be viewed as a classification problem and one way to obtain a classifier is by machine learning methods. Unfortunately, there is no single one universal good learning procedure. The *No Free Lunch Theorem* assures us that we can not design a good learning algorithm without any assumptions about the structure of the problem. So, we start by trying to find out what are the particular characteristics of the learning problem posed by the word sense disambiguation.

In our opinion, one of the most important particularities of the word sense disambiguation learning problem, seems to be the dimensionality problem, more specifically the fact that the number of features is much greater than the number of training examples. This is clearly true about data in Senseval-1, Senseval-2 and Senseval-3. One can argue that this happens because of the small number of training examples in these data sets, but we think that this is an intrinsic propriety of learning task in the case of word sense disambiguation.

In word sense disambiguation one important knowledge source is the words that co-occur (in local or broad context) with the word that had to be disambiguated, and every different word that appears in the training examples will become a feature. Increasing the number of training examples will increase also the number of different words that appear in the training examples, and so will increase the number of features. Obviously, the rate of growth will not be the same, but we consider that for any reasonable number of training examples (reasonable as the possibility of obtaining these training examples and as the capacity of processing, learning from these examples) the dimension of the feature space will be greater.

Actually, the high dimensionality of the feature space with respect to the number of examples is a general scenario of learning in the case of Natural Language Processing tasks and word sense disambiguation is one of these examples.

In such situations, when the dimension of the feature space is greater than the number of training examples, the potential for overfitting is huge and some form of regularization is needed. This is the reason why we chose to use Regularized Least-Squares Classification (RLSC) (Rifkin, 2002; Poggio and Smale, 2003), a method of learning based on kernels and Tikhonov regularization.

In the next section we explain what source of information we used and how this information is transformed into features. In section 3 we briefly describe the RLSC learning algorithm and in section 4, how we applied this algorithm for word sense disambiguation and what results we have obtained. Finally, in section 5, we discuss some possible improvements.

2 Knowledge Sources and Feature Space

We follow the common practice (Yarowsky, 1993; Florian and Yarowsky, 2002; Lee and Ng, 2002) to represent the training instances as feature vectors. This features are derived from various knowledge sources. We used the following knowledge sources:

- Local information:
  - the word form of words that appear

near the target word in a window of size 3
  - the part-of-speech (POS) tags that appear near the target word in a window of size 3
  - the lexical form of the target word
  - the POS tag of the target word

- Broad context information:
  - the lemmas of all words that appear in the provided context of target word (stop words are removed)

In the case of broad context we use the bag-of-words representation with two weighting schema. Binary weighting for **RLSC-LIN** and term frequency weighting[1] for **RLSC-COMB**.

For stemming we used Porter stemmer (Porter, 1980) and for tagging we used Brill tagger (Brill, 1995).

3 RLSC

RLSC (Rifkin, 2002; Poggio and Smale, 2003) is a learning method that obtains solutions for binary classification problems via Tikhonov regularization in a Reproducing Kernel Hilbert Space using the square loss.

Let $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ be a training sample with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ for all i.

The hypothesis space $\mathbb{H}$ of RLSC is the set of functions $f : \mathbb{R}^d \to \mathbb{R}$ of the form:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} c_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

with $c_i \in \mathbb{R}$ for all i and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a kernel function (a symmetric positive definite function) that measures the similarity between two instances.

RLSC tries to find a function from this hypothesis space that simultaneously has small empirical error and small norm in Reproducing Kernel Hilbert Space generated by kernel k. The resulting minimization problem is:

$$\min_{f \in \mathbb{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \lambda \|f\|_K^2$$

In spite of the complex mathematical tools used, the resulted learning algorithm is a very simple one (for details of how this algorithm is derived see Rifkin, 2002):

- From the training set $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ construct the kernel matrix $\boldsymbol{K}$

$$\boldsymbol{K} = (k_{ij})_{1 \le i,j \le n} \quad k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Compute the vector of coefficients $\boldsymbol{c} = (c_1, \ldots, c_n)'$ by solving the system of linear equations:

$$(\boldsymbol{K} + n\lambda \boldsymbol{I})\boldsymbol{c} = \boldsymbol{y}$$

$$\boldsymbol{c} = (\boldsymbol{K} + n\lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

  where $\boldsymbol{y} = (y_1, \ldots, y_n)'$ and $\boldsymbol{I}$ is the identity matrix of dimension n

- Form the classifier:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} c_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

The $\mathrm{sign}(f(\boldsymbol{x}))$ will be interpreted as the predicted label (-1 or $+1$) to be assigned to instance $\boldsymbol{x}$, and the magnitude $|f(\boldsymbol{x})|$ as the *confidence* in this prediction.

4 Applying RLSC to Word Sense Disambiguation

To apply the RLSC learning method we must take care about some details.

First, RLSC produces a binary classifier and word sense disambiguation is a multi-class classification problem. There are a lot of approaches for combining binary classifiers to solve multi-class problems. We used *one-vs-all* scheme. We trained a different binary classifier for each sense. For a word with m senses we train m different binary classifiers, each one being trained to distinguish the examples in a single class from the examples in all remaining classes. When a new example had to be classified, the m classifiers are run, and the classifier with the highest confidence, which outputs the largest (most positive) value, is chosen. If more than one such classifiers exists, than from the senses output by these classifiers we chose the one that appears most frequently in the training set. One advantage of one-vs-all combining scheme is the fact that it exploits the confidence (real value) of classifiers produced by RLSC. For more arguments in favor of one-vs-all see (Rifkin and Klautau, 2004).

[1]We didn't use any kind of smoothing. The weight of a term is simply the number of time the term appears in the context divided by the length of the context.

Second, RLSC needs a kernel function. Preliminary experiments with Senseval-1 and Senseval-2 data show us that the best performance is obtained by linear kernel. This observation agrees with the Lee and Ng results (Lee and Ng, 2002), that in the case of SVM also have obtained the best performance with linear kernel. One-vs-all combining scheme requires comparison of confidences output by different classifiers, and for an unbiased comparison the real values produced by classifiers corresponding to different senses of the target word must be on the same scale. To achieve this goal we need a normalized version of linear kernel.

Our first system **RLSC-LIN** used the following kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \frac{< \boldsymbol{x}, \boldsymbol{y} >}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are two instances (feature vectors), $< \cdot, \cdot >$ is the dot product on $\mathbb{R}^d$ and $\| \cdot \|$ is the L_2 norm on $\mathbb{R}^d$.

In the case of **RLSC-LIN** we used a binary weighting scheme for coding broad context. In the **RLSC-COMB** we tried to obtain more information from broad context and we used a term frequency weighting scheme. Now, the feature vectors will have apart form 0 two kind of values: 1 for features that encode local information and much small values (of order of 10^{-2}) for features encoding broad context. A simple linear kernel will not work in this case because its value will be dominated by the similarity of local contexts. To solve this problem we split the kernel in two parts:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} k_l(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2} k_b(\boldsymbol{x}, \boldsymbol{y})$$

where k_l is a linear normalized kernel that uses only the components of the feature vectors that encode local information (and have 0/1 values) and k_b is a normalized kernel that uses only the components of the feature vectors that encode broad context.

The last detail concerning application of RLSC is the value of regularization parameter λ. Experimenting on Senseval-1 and Senseval-2 data sets we establish that small values of λ achieve best performance. In all reported results we used $\lambda = 10^{-9}$.

The results[2] of **RLSC-LIN** and **RLSC-COMB** on Senseval-1, Senseval-2 and Senseval-3 data are summarized in Table 1.

[2]The coarse-grained score on Senseval-3 for both **RLSC-LIN** and **RLSC-COMB** was 0.784

	RLSC-LIN	**RLSC-COMB**
Senseval-1	0.772	0.775
Senseval-2	0.652	0.656
Senseval-3	0.718	0.722

Table 1: Fine-grained score for **RLSC-LIN** and **RLSC-COMB** on Senseval data sets

Because RLSC has many points in common with the well-known Support Vector Machine (SVM), we list in Table 2 for comparison the results obtained by SVM with the same kernels.

	SVM-LIN	**SVM-COMB**
Senseval-1	0.771	0.773
Senseval-2	0.644	0.642
Senseval-3	0.714	0.708

Table 2: Fine-grained score for **SVM-LIN** and **SVM-COMB** on Senseval data sets

The results are competitive with the state of the art results reported until now. For example the best two results reported until now on Senseval-2 data are 0.654 (Lee and Ng, 2002) obtained with SVM and 0.665 (Florian and Yarowsky, 2002) obtained by classifiers combination.

The results are especially good if we take into account the fact that our systems do not use syntactic information[3] while the others do. Lee and Ng (Lee and Ng, 2002) report a fine-grained score for SVM of only 0.648 if they do not use syntactic knowledge source.

These results encourage us to participate with **RLSC-LIN** and **RLSC-COMB** to the Senseval-3 competition.

5 Possible Improvements

First evident improvement is to incorporate syntactic information as knowledge source into our systems.

It is quite possible to substantially improve the results of **RLSC-COMB** using a combination of more adequate kernels (each kernel in the combination being adequate to the source of information represented by the part of the feature

[3]It takes too long to adapt to our systems a parser (to prepare the data for parsing, parse it with a free statistical parser and extract useful features from the parser output)

vector that the kernel uses). For example, we can use a combination of a linear kernel for local information a *string kernel* (Lodhi et al., 2002) for broad context and a *tree kernel* (Collins and Duffy, 2002) for syntactic relations.

Also, instead of using an equal weight for each kernel in the combination we can use weights[4] that reflect the importance for disambiguation of knowledge source that the kernel uses, or we can establish the weight of each kernel experimentally by *kernel-target alignment* (Cristianini et al., 2002).

[4]The weights must sum to one

References

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

M. Collins and N. Duffy. 2002. Convolution kernels for natural language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 625–632, Cambridge, MA. MIT Press.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. 2002. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367–373, Cambridge, MA. MIT Press.

Radu Florian and David Yarowsky. 2002. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of EMNLP'02*, pages 25–32, Philadelphia, PA, USA.

Yoong Lee and Hwee Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP'02*, pages 41–48, Philadelphia, PA, USA.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(February):419–444.

Tomaso Poggio and Steve Smale. 2003. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(January):101–141.

Ryan Rifkin. 2002. *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*. Ph.D. thesis, Massachusetts Institute of Technology.

David Yarowsky. 1993. One sense per collocation. In *ARPA Human Language Technology Workshop*, pages 266–271, Princeton, USA.