Word-Similarity Based Language Modeling using Support Vector Regression Machines

Ahmad Emami and Gideon S. Mann
Johns Hopkins University
Baltimore, Maryland 21218
ahmad@clsp.jhu.edu, gsm@cs.jhu.edu

Abstract

Prior language models which rely on word similarity for smoothing have used clustering alone to estimate probabilities for word bigrams using a class-based model. In this paper we describe a radically new approach which uses support vector regression machines to directly estimate conditional bigram probabilities without resorting to intermediate class models. A key part of our method is the projection of words into a high-dimensional continuous valued space using word clustering methods.

1 Introduction

Language models $[P(W)]$ are a key component of modern speech recognition systems and give a measure of how probable a given sequence of words is *a priori*. The language model has other applications as well including machine translation, handwriting recognition and OCR. The other major component in speech recognition systems, the acoustic model $[P(A|W)]$, estimates the probability of a acoustic signal having been generated by a given word sequence. Using the noisy channel model and Bayes Rule, these two terms find the most likely word sequence given the acoustic data.

$$\hat{W} = \underset{W}{argmax}\ P(W|A) = \underset{W}{argmax}\ P(A|W)P(W)$$

The language model is applied one word at time to predict the next word given all prior words. Ideally, the language model would use the entire history to make its prediction. However, data sparsity is a crippling problem for language models. As a result a Markov assumption is made, so that the model predicts the next word not on the entire history, but merely the previous N - 1 words. Real speech recognition systems typically use N = 3 or 4. In this paper we restrict ourselves to a simpler model (a *bigram model*), where the next word is predicted solely based on the previous word(N=2).

$$P(w_i|\text{history}) = P(w_i|w_{i-1})$$

Though the MLE can be computed simply, even a bigram model suffers from sparse data. For large vocabularies (V = 100K), there are 10 billion (10^{10}) parameters to estimate. Even a small vocabulary (1K) has 1 million parameters to estimate, necessitating an enormous amount of training data. Word sequences assigned zero probability are problematic for a speech recognizer because this prevents the machines from ever considering those word sequences.

Determining how to estimate unseen word sequences is a key problem of language modeling. Interpolation estimates are commonly used which combine bigram, unigram and 0-gram probabilites to estimate word sequences (Jelinek, Mercer 1980) .

$$P(w_i|\text{history}) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i) + \lambda_3 \frac{1}{|V|}$$

2 Class-Based Language Models

One of shortcoming of the Jelinek-Mercer interpolation model is that it ignores potential information from similar words. For example, nouns often follow verbs, but only very infrequently do verbs follow verbs. Person names should occur with the same classes of verbs, as should common nouns. Traditional smoothing models neglect this kind of semantic relateness.

People have tried to create models which reflect this semantic relatedness by building class-based language models (Brown et al. 1992, Jardino 1994, Kneser and Ney 1993, Lafferty and Mercer 1993, Bellegarda et al. 1996). In these models, clustering methods are applied to build words class from which class probabilities are estimated.

The bottom-up clustering method proposed by (Brown et. all 1992) is typical of these methods. In bottom-up clustering the individual words of the vocabulary are first assigned to individual classes, each class containing only one word. In a class-based language model, the probability is defined as:

$$P(w_i|w_{i-1}) = P(w_i|\phi(w_i))P(\phi(w_i)|\phi(w_{i-1}))$$

This will be maximized when each word is in its own class. However the number of classes is too big in order to be able to estimate the transition probabilities $P(\phi_0|\phi_1)$ reliably. Therefore we must

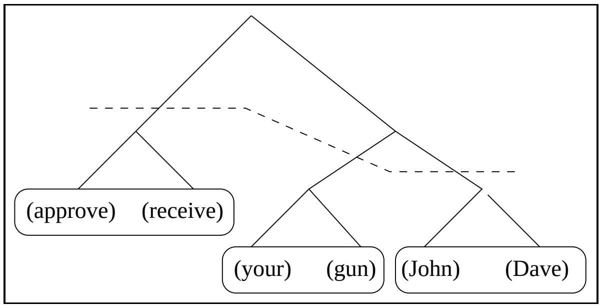

Figure 1: Typical use of word clustering : word classes

reduce the number of classes, and we will do so by merging the classes two at a time in a hierarchical manner. Every merge results in a loss in the log likelihood and we will first merge the two classes among all possible pairs whose merging will lead to the least reduction in the log likelihood. After each merge the probability distributions involving the 2 old nodes and the new node should be updated as well the estimated reductions in log likelihood. This process in continued until only one class is left.

It can be proven that when changing class membership only, the change in log likelihood is equal to the change in mutual information of classes predicting each other, which is:

$$MI = \sum_{\phi_0} \sum_{\phi_1} P(\phi_1, \phi_0) log\Big(\frac{P(\phi_1, \phi_0)}{P(\phi_0) \cdot P(\phi_1)}\Big)$$

Brown et al. (1992) describe an algorithm which performs this greedy search and builds a tree in $O(|V|^3)$, where $|V|$ is the size of the vocabulary. Unfortunately, this computation time restricts use of the method to small vocabularies.

Once the tree has been built, cuts across the tree can then be made in order to derive classes for each word $\phi(w)$ (Figure 1). Now, transition probabilities $P(\phi_0|\phi_1)$ and posterior distributions over words $P(w|\phi_0)$ can be estimted from training data. Variations of this model have shown improvements in perplexity when interpolated with traditional language models.

3 How To Pose LM as a Support Vector Regression Task

Boser et al. (1992) proposed support vector machines as a method for solving statistical estimation problems which overcome limitations of classical statistics for parameter estimation on small sample sizes. SVMs are "large margin" classifiers which find the optimal hyperplanes which seperates two classes. This hyperplane can be found given l vectors x_i with the assigned class y_i by finding the saddle point of

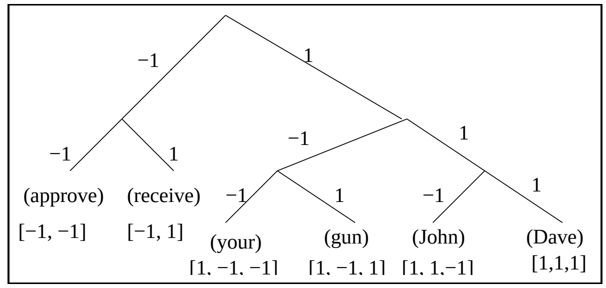

Figure 2: Word clustering used for projection into a high-dimension space

the Lagrange functional (Vapnik 2000) :

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - C \sum_{i=1}^{l} \alpha_i \{[x_i \cdot w] - b]y_i - 1\}$$

In SVMs, each input vector x is projected from a small dimensional space into a much higher dimensional space where the hyperplane will easily seperate the data. The mathematics of the projection and of the SVM formulation allows for optimizations to be made using a Kernel function which computes distance between elements in the high dimensional space. This transformation using Kernel functions allows the minimization to be performed efficiently.

Statistical learning theory predicts that since SVMs take into account both the empirical risk ($\sum_{i=1}^{l} \alpha_i \{[x_i \cdot w] - b]y_i - 1\}$) and the structural risk ($\frac{1}{2}(w \cdot w)$) they should perform well even for small training sets. This theoretical prediction has been borne out in the past decade as more people have begun using SVMs for statistical learning. These experiments have demonstrated good performance on a wide variety of machine learning tasks and the SVMs have exhibited remarkable generalization ability. The recent success of SVMs make them appealing candidate to apply to the task of language modeling.

In order to pose the problem of language modeling as a support vector machine problem, first the data needed to be projected into a continuous domain. Inspired by past work on class-based language models, we chose to project words into a continuous space using the bottom-up word clustering described in Section 2.

This method builds a tree from word co-occurance statistics, and then we assign each word a vector according to the path taken from the top of the tree to the leaf containing that word, with a -1 for every left branch taken, and a 1 for every right branch (Figure 2). By doing this we can construct a vector for each word, in the Figure, "approve" would be mapped to the vector [-1,-1] and "John" to [1,1,-1].

As a result of clustering, not all leaves are at the same depth (i.e the tree is not complete). In order to normalize the vectors, we added 0's to fill out vectors which had missing values (i.e. they were at the lowest possible depth).

The next observation was that class based method were inappropriate for the task. Ultimately, what is desired is probability estimates, not word clusters. As Vapnik (2000) sagely suggests:

> *When solving a given problem, try to avoid solving a more general problem as an intermediate step.*

With this goal in sight, an obvious way to use Support Vector Machines presents itself: perform regression over the continuous space and thereby directly estimate probabilities. Support Vector Regression Machines find the hyperplane with the minimal distance to the training points and the least structural complexity:

$$L(w, \xi^+, \xi^-) = \frac{1}{2}(w \cdot w) + C\Big(\sum_{i=1}^{l} \xi^+ + \sum_{i=1}^{l} \xi^-\Big)$$

where ξ^+ and ξ^- represent positive and negative error respectively (for details see Vapnik 2000). Instead of regressing over conditional probabilities, we decided to regress over pointwise Mutual Information values. In this context, the MI statistic can be seen as a measure of how much more likely are you to see the word y, given that you've already seen the word x compared to the expectation of seeing the word x *a priori*.

$$MI(x, y) = log\frac{p(x|y)}{p(x)}$$

This is a reasonable value over which to smooth since one would expect that semantically similar words x_i and x_j would have similar degrees of increased (or decreased) prediction given a predictor y, while they might have vastly different prior (and therefore posterior) probabilities. Once this value is estimated it can be used to derive the predicted conditional probability:

$$\hat{p}(x|y) = p(x)\widehat{MI}(x, y)$$

4 Experiment 1 : MI estimation

Our experiment was done on the top 1000 most frequent words in the WSJ 87 corpus using bigrams, estimating conditional probabilities for each predictor individually. The tree we got from the bottom-up hierarchical word clustering, described in the previous section, had a maximum depth of 19, so this resulted in 19 dimensional vectors (with 0's padding out vectors with missing values). We used the support vector regression machine package 'mySVM' (Rüping 00), which is based on SVM^{light} (Joachims 1999). After tuning on the word 'Reagan', we decided to use the polynomial kernel $(x * y + 1)^3$ with C=1 and $\epsilon = 1$, which we tuned on one word ('Reagan').

Figures 3 - 6 show an example of what the Support Vector Regression Machine learns for bigrams (of, X). In the graph, the words have been projected from a 19-dimensional space (constructed from the tree projection). The major troughs in the smoothed MI picture correspond to verb clusters : the first (words 131 - 143) is ("keep meet reduce raise take seek go give provide allow receive continue begin") and the trough at the end is ("is remains makes includes means was has isn't seems wasn't doesn't hasn't does"). The highest peak (words 65-75) is composed of pronouns ("her their those these our my your them us him me"), most of which are possesive pronouns. As can be seen, the conversion of the Mutual information values into probabilities (Figure 4 to Figure 6) has changed the curve, though not dramatically. The major peak now is "the".

The curve generated by the Support Vector Regression Machine shows exactly the kind of generalization we hoped for. It has generalized over syntactic class and semantic relatedness to provide more accurate estimates of mutual information than were previously available.

Another way to analyze the performance of the SVRM is to see how the original compares with the estimate. Figure 7 illustrates the deviation of the original MI estimate to the estimated MI. As can be seen from the figure, the SVRM machine has effectively fit the data, picking out an underlying trend amoung the noise.

5 Experiment 2 : Language Modeling

To test the estimted mutual information derived in the above method, we computed the perplexity of a held-out set of test data. The perplexity is calculated as [1]:

$$PP(W) = 2^{LP(W)}$$

Where LP is the entropy of the data given the model :

$$LP(W) = -\frac{1}{n}\sum_{i=1}^{n} logP(w_i|\text{history})$$

We tested two models, a bigram model :

$$P_B(x|y) = \begin{cases} \frac{c(x,y)-\alpha}{c(y)} & \text{if c(x,y)} > 0 \\ \lambda P(x) & \text{otherwise} \end{cases}$$

And the case when our estimated $P_E(x|y)$ was interpolated with the bigram.

[1] From (Jelinek 1997)

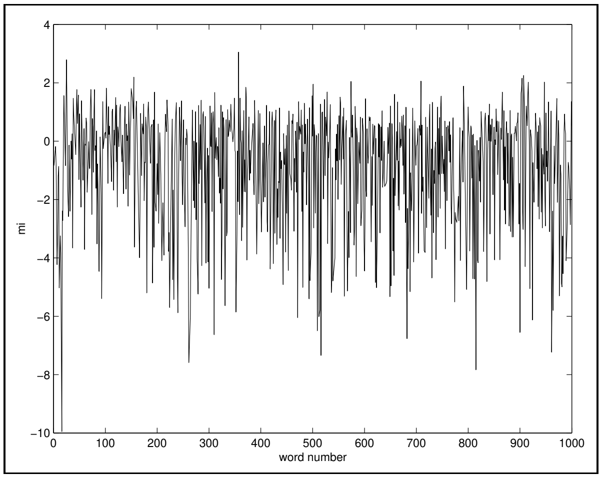

Figure 3: MLE of $MI(of, X)$ from Training Corpus

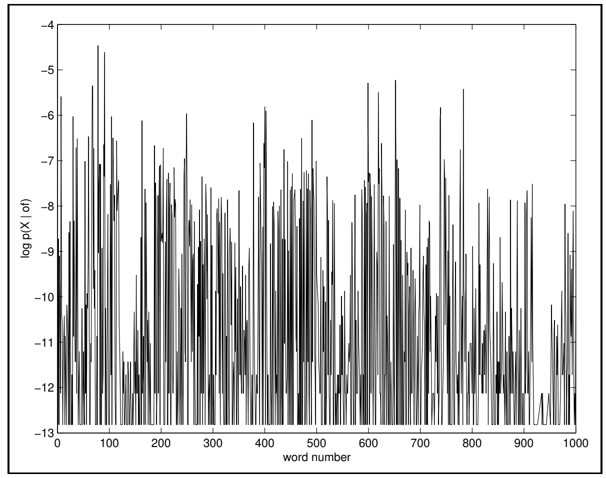

Figure 5: MLE of $logP(X|of)$ from Training Corpus

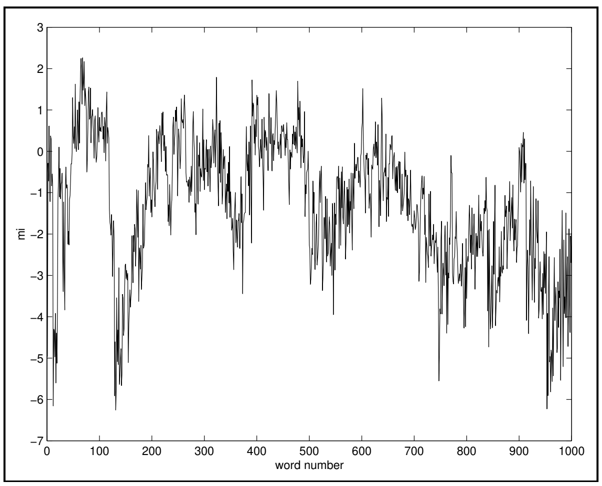

Figure 4: SVRM Estimate of $MI(of, X)$

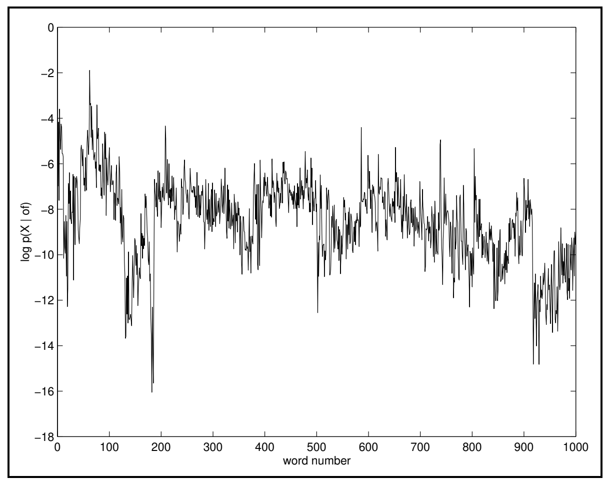

Figure 6: SVRM Estimate of $logP(X|of)$

$$P_I(x|y) = \begin{cases} \frac{c(x,y)-\alpha}{c(y)} & \text{if c(x,y)} > 0 \\ \lambda P_E(x|y) & \text{otherwise} \end{cases}$$

In both of these absolute discounting models, λ is chosen to form a distribution over the space. With the above definitions, the raw bigram model (P_B) had a perplexity of 62.48 on the held-out test data while the estimated model (P_I) has a perplexity of 62.80. Even though there is not a gain from the use of the estimated MI information, these results in addition to those described in the previous section demonstrated that method might be fruitful avenue of research.

6 Conclusions

In this paper we have presented a radically new method for using Support Vector Regression Machines in language modeling. First we demonstrated how to project words into a continuous space using hierarchical clustering techniques. Next, we showed how to generalize over this space using mutual information and then how to convert these mutual information values back into conditional probabilities. Finally, we gave some initial experimental results of using these estimates for language modeling.

This "kernel" of an idea opens up a tremendous amount of future research pathways. First, the projection method is clearly sub-optimal in that the space and the kernel functions are oddly constructed. Each dimension has a very different level of importance for the task, so treating them as equivalent is inappropriate. Clearly, some other projection method could be used before the smoothing step. One projection method which seems especially suitable is Latent Semantic Analysis (Deerwester 1990, Landauer and Dumias 1997). LSA projects a word into a high-dimensional space (200+ vectors) where each dimension has a unique semantic content.

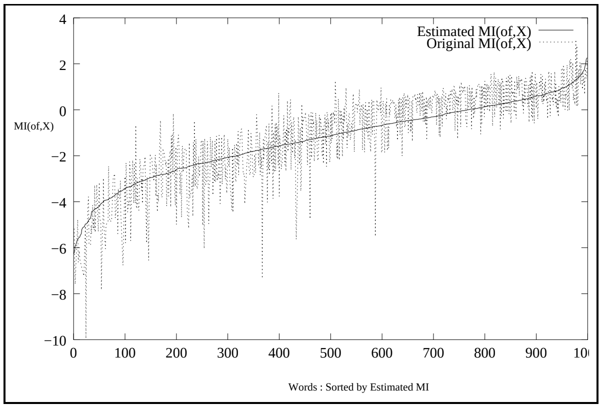

Figure 7: Original MI plotted against Estimated MI

Designing a support vector machine more suited to the task of estimating conditional probabilities is another major line of inquiry. It seems like what is really desired is to adjust the conditional probabilities so that the mutual information is as smooth as possible – but this doesn't necessarily imply that the conditional probabilities are also as smooth. Moreover, constraints that the sum of the probability over the entire space must sum to 1 changes the nature of the smoothing which is appropriate.

Finally, missing data should be trated in a manner different from their current treatment. Because of the nature of the regression, values for missing data is estimated in a form of interpolation from "nearby" training items. This is not entirely appropriate since missing values are missing for a reason – they didn't occur in the text. If the words are frequent, this may be significant. For this reason, it might make senese to treat missing values differently. We have tried a few experiments with setting missing values to $-log(\frac{1}{P(x)p(y)})$, but this did not appear to help. Ultimately what is needed is a direct reformulation of the problem so that the problem can be solved directly.

References

J.R. Bellegarda, J.W Butzberger, Y.L Chow, N.B Coccaro, and D. Naik. 1996. A novel word clustering algorithm bases on latent semantic analysis. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 172–175, Atlanta, GA.

B. Boser, I. Guyon, and V.N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA.

P.F. Brown, V.J. Della Pietra, P.V. deSouza, R.L. Mercer, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467 – 479.

S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

M. Jardino. 1994. A class bigram model for very large corpus. In *International Conference on Spoken Language Processing*, pages 867–870.

F. Jelinek and R.L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *Proc. of the Workshop on Pattern Recognition in Practice*, Amsterdam, the Netherlands: North Holland, May.

F. Jelinek. 1997. *Statistical Methods for Speech Recognition.* Boston: MIT Press.

T. Joachims. 1999. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola (ed.), editors, *Advances in Kernel Methods - Support Vector Learning.* Boston : MIT Press.

R. Kneser and H. Ney. 1993. Improved clustering techniques for class-based language modelling. In *European Conference on Speech Communication and Technology*, pages 973–976.

J.D. Lafferty and R.L. Mercer. 1993. Automatic classification using features of spelling. In *Proc. 9th Ann. Conf. of the University of Waterloo Centre for the new OED and Text Research*, pages 89–103. Oxford : Oxford University Press.

T.K. Landauer and S.T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211 – 240.

S. Rüping. mysvm, http://www-ai.cs.uni-dortmund.de/software/mysvm/#joachims/99a.

V. N. Vapnik. 2000. *The Nature of Statistical Learning Theory.* New York City: Springer.