Transcription of Text by Incremental Support Vector Machine

Anurag Sahajpal
Department of Computer Science
Bergen University College
N-5020 Bergen, Norway
email: anurag@ii.uib.no

Terje Kristensen
Department of Computer Science
Bergen University College
N-5020 Bergen, Norway
email: tkr@hib.no

Abstract

This paper deals with an on-going work aimed at developing a Suport Vector Machine (SVM) based incremental learning algorithm in the domain of text-based phoneme transcription for Norwegian language. The motivation for this is to reduce the long computation time witnessed in the batch-learning scenario. A standard SVM algorithm is modified in such a way that it incorporates the new data as it becomes available in time. The transcription scheme used is SAMPA for Norwegian. We conclude the paper with a discussion on further research in this direction.

1 Introduction

The basic abstract symbol representing speech sound is the phoneme. In both text-to-speech (TTS) systems [14] or in automatic speech recognition (ASR) [5, 7] systems the phonetic analysis is an important component. Its primary function is to convert the text into the corresponding phonetic sequence. This process is called transcription.

Many authors have studied the process of phoneme transcription by use of artificial neural networks (ANN). Sejnowski gave the first introduction to this field in his classical paper [11]. Phoneme recognition has also been studied by use of time-delay neural networks by Waibel and his colleagues [14] with fairly good results. The ANN approach to the problem of phoneme transcription of a given language can be described as a pattern matching technique where a set of transcription patterns are created to determine how written words are transcribed into text-based phonemes.

A paper about how to use a Support Vector Machine (SVM) to transcribe Norwegian has recently been presented [9]. It demonstrated that a SVM network might be used to transcribe Norwegian text fairly well by using a batch-learning regime. By batch learning we mean a learning algorithm that would require processing of the entire data set when new data is added to existing data set. A key problem is to construct a cumulative learning algorithm that incrementally incorporates new data as it becomes available in time (incremental learning) or across the space (distributed learning) [2]. In this paper we will focus on the incremental learning aspect of SVM.

In what follows, Section 2 discusses the basic theoritical elements of SVM, Sections 3 and 4 describes the SVM-based incremental learning approach, which we plan to use in our domain of text-based phoneme transcription, followed by the description of the domain and experimental results obtained so far. We conclude the paper with a discussion on further research in this direction.

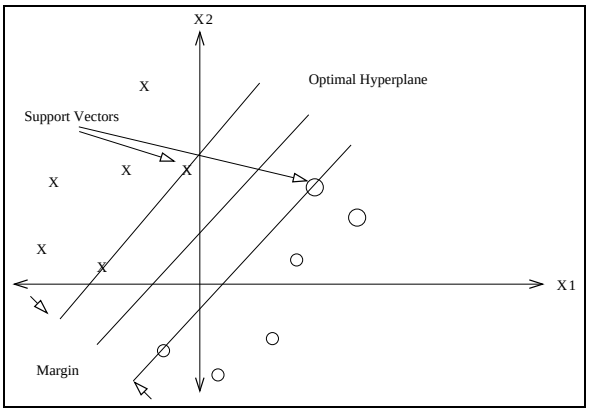

Figure 1: A support vector machine classification defined by a linear hyper plane that maximizes the separating margins between the classes.

2 SVM Theory

Support Vector Machine is a computationally efficient learning technique that is now widely being used in pattern recognition and classification problems. This approach has been derived from some of the ideas of the statistical learning theory regarding controlling the generalization abilities of a learning machine [12, 13].

In this approach the machine learns an optimum hyperplane that classifies the given pattern. By use of kernel functions, the input space, by application of a non-linear function, can be transformed into a higher dimensional space where the optimum hyperplane can be learnt. This gives a flexibility of using one of many learning models by changing the kernel functions.

SVM Classifier

The basic idea of an SVM classifier is illustrated in figure 1. This figure shows the simplest case in which the data vectors (marked by 'X' s and 'O' s) can be separated by a hyperplane. In such a case, there may exist many separating hyperplanes. Among them, the SVM classifier seeks the separating hyperplane that produces the largest separation margin.

In the more general case, in which the data points are not linearly separable in the input space, a non-linear transformation is used to map the data vectors into a high-dimensional space (called feature space) prior to applying the linear maximum margin classifier. To avoid the potential pitfall of overfitting in this higher dimensional space, an SVM uses a kernel function in which the non-linear mapping is implicitly embedded. A function qualifies as a kernel function if it satisfies the Mercer's condition [4].

By the use of a kernel function, the discriminant function in a SVM classifier has the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b \tag{1}$$

Where K (-, -) is the kernel function, $\mathbf{x_i}$ are the support vectors determined from the training data, $\mathbf{x}$ is the pattern vector to be predicted, y_i is the class indicator (e.g. +1 and

-1 for a two class problem) associated with each $\mathbf{x_i}$, N is the number of supporting vectors determined during training, α_i is the Lagrange multiplier for each point in training set and b is a scalar representing the perpendicular distance of the hyperplane from the origin.

Support vectors are elements of the training set that are located either exactly on or inside the decision boundaries of the classifier. In essence, they consist of those training examples that are most difficult to classify. The SVM classifier uses these borderline examples to define its decision boundary between the two classes.

The SVM kernel functions

The kernel function plays a central role of implicitly mapping the input vectors into a higher dimensional feature space, in which better separability is achieved. The most commonly used kernel functions are the polynomial kernel given by:

$$K(\mathbf{x_i}, \mathbf{x}) = (\mathbf{x_i}^T\mathbf{x} + 1)^P, \text{where P > 0 is a constant} \tag{2}$$

or the Gaussian radial basis function (RBF) kernel given by:

$$K(\mathbf{x_i}, \mathbf{x}) = e^{-\|\mathbf{x_i}-\mathbf{x}\|^2/2\sigma^2} \tag{3}$$

where $\sigma > 0$ is a constant that defines the kernel width. Both of these kernels satisfy the Mercers condition mentioned above.

3 Requirements to Incremental SVM for Phonemic Data

We want to explore the possibility to develop better pattern classifiers as multiple data sources become available in time. A standard SVM algorithm may be extended or modified to do such a task.

In what follows we may assume that the phoneme datasets are linearly separated. If they are not, we may find a suitable kernel function to transform them into a feature space where they are supposed to be linear separable. The non-linear classification scheme may always be transformed to a binary linear classification scheme where the computations are done in the low-dimension input space.

In this paper we make the simplification that there are only two data sets of phoneme transcriptions, which become available to the learner at different instances of time (say t_1 and t_2). We further assume that number of classes of phonemes are two. We want to learn a binary classifier using these two data sets of Norwegian phonemic transcriptions, D_1 at time t_1 and D_2 at time t_2, using an incremental learning SVM algorithm. If the learning is exact the classifier should be the same as the one obtained by using a batch learning mode on the data set D = $D_1 \cup D_2$.

A simple approach to incremental SVM learning can be one that involves only addition of support vectors [8]. Such an approach will only work reasonably well when D_1 and D_2 are representative for the entire training set $D_1 \cup D_2$, so that the hyperplane calculated from one is only modified not very much from the hyperplane derived from the whole training set. In [2] it is shown that this may not always be the case by constructing simple artificial data for distributed learning setting for the data sets D_1 and D_2. In general we will have a situation like:

$$SV(D_1 \cup D_2) \neq SV(SV_1 \cup SV_2) \tag{4}$$

where SV is a short notation for Supporting Vectors of the different data sets.

The separating hyperplane is dependent on the support vectors. A simple approach therefore loses important boundary information, which results in that the classifier can result in an arbitrarily high classification error. The SVM learning algorithm therefore has to be modified in a way such that the data sets are incorporated in an exact manner i.e. integrity of the support vectors is preserved.

4 Incremental Learning by Convex Hull Approach

Let L be a learning algorithm for a pattern classification algorithm and let the data sets $D_1, D_2, \ldots D_n$ be given at different times. A sufficient condition for exact learning for incremental case is described below [2]:

$$L(\ldots L(L(D_1) \cup D_2) \ldots \cup D_n) = L(D_1 \cup D_2 \cup \ldots D_n) \tag{5}$$

Equation (5) expresses the requirement for exact learning in an incremental learning setting given N datasets, while equation (6) expresses the sufficient condition for it, namely, the u-closure property for two data sets:

$$L(L(D) \cup D^{'}) = L(D \cup D^{'}) \tag{6}$$

Support Vectors do not satisfy this closure property, but the vertices of the convex hull of the two instances of the classes do. The convex hull of a set of points S, denoted conv(S), is the smallest convex set containing S. conv(S) is defined by :

$$conv(S) = \{\mathbf{x} \in \mathbf{R^n} | \mathbf{x} = \sum_{x \in S} \lambda_i x_i = 1, \lambda_i \geq 0\} \tag{7}$$

The conv(S) is the set of all non-negative linear combinations of points from S. If the set S is finite, the conv(S) is a convex polyhedron given by the intersection of closed halfspaces. The vertices are interesting to us because they uniquely define the convex hull [1, 6]. The union closure property is a well-known theorem for a convex hull. Since such a theorem is satisfied for convex hulls, we can therefore construct an incremental learning algorithm in general by making a transformation of our training set into a convex hull.

Let Vconv(A) denote the vertices that define the convex hull of a convex set A. We then have:

$$Vconv(Vconv(A_1) \cup A_2) = Vconv(A_1 \cup A_2) \tag{8}$$

Let us suppose that two data sets D_1 and D_2 are given in a way such that $D_1 \cup D_2$ is linearly separable and letting $D_i(+)$ and $D_i(-)$, for i = 1, 2, denote the positive and negative instances of the data set D_i. If $D_1 \cup D_2$ is not linear separable, we may transform it to a feature space by a kernel function where this condition is satisfied. Let SVM(D) denote the result of applying the SVM algorithm on data set D. An incremental SVM learning algorithm can then be constructed in the following way:

1. Compute Vconv($D_1(+)$) and Vconv($D_1(-)$)
2. Add Vconv($D_1(+)$) to $D_2(+)$ to obtain $D_2^{'}(+)$
3. Add Vconv($D_1(-)$) to $D_2(-)$ to obtain $D_2^{'}(-)$
4. Compute Vconv($D_2^{'}(+)$) and Vconv($D_2^{'}(-)$).

5. Generate the training set D_{12}, where

$$D_{12} = Vconv(D_2^{'}(+)) \cup VconvD_2^{'}(-))$$

6. Compute SVM (D_{12}).

The above procedure may easily be extended to any number of training data sets. Such an algorithm is exact with respect to the solution found in the incremental mode and will be identical to the results obtained in batch mode [2]. In our case D_1 and D_2 represent data for text-based phoneme transcriptions.

5 SAMPA

The transcription scheme in this paper is based on SAMPA (Speech Assessment Methods Phonetic Alphabet) for Norwegian [10], which is a machine-readable phonetic alphabet and constitutes today one of the ways for encoding of machine-readable phonetic notation.

In the transcription scheme for Norwegian based on SAMPA, the consonants and the vowels are classified into different subgroups defined as follows:

Plosives (6)
Fricatives (7)
Sonorant consonants (5)

where the number in brackets indicate the number of each type.

The vowels are likewise defined by the subgroups:

Long vowels (9)
Short vowels (9)
Diphthongs (7)

In our phoneme database, ordinary and double stress marks are indicated by ! and !!. Table 1 shows some words with location of stress, transcribed by the SAMPA notation.

6 Grapheme to Phoneme Transcription

A grapheme designates the atomic unit in written language, which includes letters, numerals, punctuation marks and other symbols. It is the basic text unit, and for our purpose in this paper, a grapheme simply mean one of the 29 letter-symbols in the Norwegian alphabet. A phoneme, on the other hand, is the theoretical basic unit of sound that can be used to distinguish words in an oral language.

The training data consists of words that are constructed from all the letters (29) of Norwegian. In addition, space is included. Some character combinations, such as double consonants, have been treated in a special way in the transcription algorithm that we have been using. A double tt, for instance, is transcribed to a single t.

The number of phonemes in a language, even in a standard variety, may show some regional variation, and is also to some extent dependent on the chosen method of phonemic analysis. We have used an estimate of 43 phonemes for spoken Norwegian. The SAMPA notation used in our transcriptions is not strictly phonemic, but also uses separate symbols for some allophones, e.g. [@i] and [e] for the lax ("schwa") and the non-lax allophone, respectively, of the phoneme /e/. In addition, our program treats 5 retroflex

Table 1: Some words in the training database with different stress placements.

Words	**Transcription**
apene	!!A:p@n@
lønnsoppgaver	!l2nsOpgA:v@r
politiinspektørene	!puliti:inspk!t2:r@n@
regjeringspartiet	re!jeriNspArti:@
spesialundervisningen	spesi!A:l}n@rvi:sniN@n

allophones ([rd], [rl], [rn] etc.) and corresponding cases where dental consonants do not change into retroflexes ([r-d], [r-l], [r-n] etc.) as separate units. Hence the total number of symbols exceeds the assumed number of phonemes by 10, and thus the total number of phonetic symbols considered in our algorithm is 54.

The letter-to-phoneme correspondence for a given language is generally not one-to-one. This means that the spelling system of that language may be such that the same letter (grapheme) or combination of letters (graphemes) can correspond to different sounds, depending on the context. For example, in Norwegian, the letter “a” correspond to short A sound in word “natt” while the same letter correspond to long A sound (donated by A: in SAMPA) in word “rar”.

The SAMPA notation [10] for Norwegian describes, by way of giving examples, how graphemes correspond to phoneme units. About 36 of phonems are one letter (grapheme) long and the rest 7 diphthongs such as ei, au, etc consists of two letters. In addition to this, in our algorithm, the second group also includes double consonants such as bb, ff, pp, rr, ss, etc and retroflex allophones such as rd, rl etc.

7 Transcription Method

An actual word is considered, and using a window segment on it generates a letter pattern. The phoneme to be predicted is in the middle of the window. The Norwegian language following the transcription regime that we have used totally consists of 54 phonetic symbols or units. Therefore our SVM needs to classify the text into 54 different classes, each for a separate unit.

A certain view (window) is given to the different words, as shown in figure 2. The window size selected in the experiments is seven letters. At any given time the letter in the middle of the window is active. In figure 2 this is ’r’. When ’r’ is active, all the other letters will be inactive.

To handle the beginning and end of each word, and to fill up the space of seven letters, a star (*) is used. The desired output of the network is the correct phoneme associated

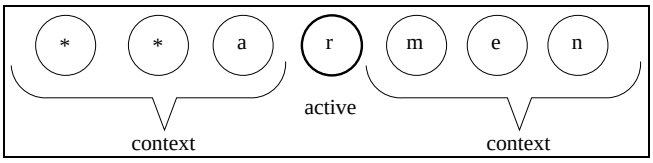

Figure 2: The Norwegian word "armen" in a seven-letter window. The associated phoneme class of the letter r will be the output of SVM.

Input vector : [0 0 0 1 16 5 0]
Target : [2 34]

Figure 3: The input format

with the center or the fourth letter in the window. The other six letters, three on either side of the middle letter, provide the context.

The size of the window may vary. By choosing a seven-letter window we may have the significant information needed to correctly transcribe a Norwegian word. A larger window could give better results because it will generate fewer conflicts in the training session, but will require more computer power and longer training times. The window can also be asymmetrical with unequal number of nodes or positions on each side of the centre node.

The words are stepped through the window letter by letter. At each step the SVM network computes the phoneme that corresponds to the letter in the middle of the window. The parameters of the separating hyper plane are learnt accordingly.

8 Preprocessing and Coding

The original training data consists of binary letter-patterns of words and their transcriptions. Each letter pattern has to be preprocessed before it is fed into the network. A separate program has been developed to do this preprocessing. An internal coding table has been defined in the program to represent each of the 54 phonetic symbol with a unique number in 1 - 54 range. The data after this preprocessing step resembles the input format, as shown in figure 3.

A pattern file is generated, consisting of a huge amount of patterns conforming to the input format, given in figure 3, which illustrates the pattern corresponding to the word ***ape*. The target pattern for the middle letter in the window is given by !!A:, which represents double stress and the phoneme long A vowel. * is represented by 0 in the input.

Each input pattern (or feature vector) in the file, such as the one given in figure 3, has seven components, one for each letter in the window segment. The components represent the letters of the Norwegian language and, at any given time, the component in the middle is active, for which the phoneme is produced. The three components on either side of the active component provide the context. The size of the preprocessed file is about 17 Mbytes by which the SVM is trained.

A “1“ as the 4th attribute expresses the letter a. In the target, the corresponding phoneme long A is mapped to the correct position ie. 34 and the double stress is indicated by 2.

The problem of mapping a letter to its phoneme is divided into two classification problems, one for the stress classification (ordinary, double or no stress - 3 classes) and other for the phoneme classification (54 classes). The stress classification is done separately from phoneme classification. In the final phase of classification, the two output files are merged into a single file to produce a transcription with correct placement of stress.

Table 2: A sample of Norwegian test words generated by SVM

Text	Target	Produced
assisterte	Asi!ste:rt@	Asi!ste:rt@
assortert	Asurt!e:rt	As}rte:rt
attesterte	At@!ste:rt@	At@!ste:rt@
proletariat	prul@tAri!A:t	prul@tAri!A:t
representantene	repr@s@n!tAnt@n@	repr}s@n!tAnt@n@
sivilisasjonene	sivilisA!su:n@n@	sivilisA!su:n@n@

9 Experimental Results

All the experiments have been done on a 2.4 GHz Pentium PC with a 256 Mbytes memory. The result so far obtained pertains to the batch-learning scenario. The basic SVM scheme works only for binary classification, and, in order to deal with a multi-class problem such as we have, a generalized classification scheme has to be used. The classification models for multi-class patterns that we used are One vs One and One vs Rest. The former method of One vs One consists of training one SVM for each pair of classes while the later method consists of training one SVM for each class. Because of highly unbalanced data, One vs Rest was preferred. The kernel function that was used in the experiments was the RBF kernel and the optimal value of the cost parameter C was 200. The training time depended upon the choice of parameters values and size of data. The maximum training time for the batch mode was about 7 days. The program developed used LIBSVM [3] as a toolbox. For the incremental case, SVM algorithm in batch-mode has to be revised to enable it to handle incremental learning. This process has started.

The performance of the network was about 83 % at phoneme level for batch learning. A Sample of words of Norwegian that are produced in batch mode is given in table 2.

10 Conclusion

A SVM algorithm has been developed to transcribe Norwegian text to phonemes in a batch-learning mode and experiments performed. Work towards the basic theortical framework for incremental mode has been started. Future work is going on to modify the SVM algorithm to handle the incremental learning mode as per the convex hull approach described in section 4. The present results indicate an overall good generalization performance of the batch-learning SVM, but the training cycle is long. We expect to see a reduction of time with incremental SVM while maintaining the same generalization accuracy. The experiments are going on and it is yet to be seen how well both regimes of learning compare and how the incremental learning mode scales up to handle high-dimensional input data.

11 Acknowledgement

The authors wish to thank Prof. Helge Dyvik at the Linguistics department, HF faculty, University of Bergen, for his help and feedback on the aspects concerning phonemes in Norwegian in this paper.

References

[1] Barber B.C. and Huhdanpaa H. The Quickhuull Algorithm for Convex Hull. *ACM Transactions on Mathematical Software*, 22(4), 1996.

[2] Caragea D. and Silvescu A and Honavar V. Agents that learn from distributed data sources. In *Fourth International Conference on Autonomous Agents*, 2000.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM : a library for Support Vector Machines. available online at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[4] Haykin H. *Neural Netowrks - a Comprehensive Foundation*. Prentice Hall, 1999.

[5] Jorgensen C. and Lee D.D. and Agabon S. Sub Auditory Speech Recognition Based on EMG Signals. In *IJCNN 2003*, Portland, USA, 2003.

[6] Joseph O'Rourke. *Computational Geometry*. Cambridge University Press, 1998.

[7] Juneja A. and Epsy-Wilson C. Speech Segmentation Using Probabilistic Phonetic Feature Hierarchy and Support Vector Machine. In *IJCNN 2003*, Portland, USA, 2003.

[8] N.A.Syed and H. Liu and K.K.Sung. Incremental Learning with Support Vector Machines. In *KDD Conference*, SanDiego, USA, 1999.

[9] Sahajpal A. and Kristensen T. and Kumar G. Phoneme Transcription by Support Vector Machine. In *11th International Conference on Neural Information Processing*, Calcuta, India, 2004.

[10] SAMPA. available online at http://www.phon.ucl.ac.uk/home/sampa/norweg.htm.

[11] Sejnowski T. J. and Rosenberg C. R. Parallel Networks that Learn to Pronounce English Text. Complex Systems Publications Inc., 1987.

[12] Vapnik V. N. *Statistical Learning Theory*. Wiley, New York, 1998.

[13] Vapnik V. N. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10, sep 1999.

[14] Waibel A. and Hanazawa T. and Hinton G. and Schiko K. and Lang K. Phoneme recognition using time delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Mar 1999.