Abstract

In this paper, we describe a new method for training SVM on large data sets. Vector Quantization is applied to reduce a large data set by replacing examples by prototypes. Training time for choosing optimal parameters is greatly reduced. Some experimental results yield to demonstrate that this method can reduce training time by a factor of 100, while preserving classification rate. Moreover this method allows to find a decision function with a low complexity when the training data set includes noisy or error examples.

1 Introduction

The Support Vector Machines (SVM) [10] are a powerful tool to solve pattern recognition problems. However the use of the SVM is difficult with large data sets, even by using fast SVM algorithms dedicated to solve large problems [1]. One reason is that a model selection (parameters search) must be done. The training process is time consuming if the parameters are far from the optimal values. Moreover the optimal solution is in various ranges for different implementations [6]. Another reason is that the complexity of the decision function generally increases with the size of database [9].

A method using Vector Quantization (VQ) to reduce the size of the training data set is proposed. We experimentally show that the proposed method finds the optimal solution (or a closer one) faster than the classical grid search [1]. We also show that our method can find a decision function with reduced complexity when the data set includes noisy or error examples.

In section 2 and 3, we briefly describe the VQ technique and SVM. In section 4 the proposed approach is detailed. Numerical experiments are shown in section 5. Finally discussions and conclusions are proposed in section 6.

2 Vector Quantization

Vector Quantization (VQ) maps a vector \( \mathbf{x} \) of dimension \( k \) to another vector \( \mathbf{y} \) of dimension \( k \) that belongs to a finite set \( \mathbf{C} \) (codebook) of output vectors (codewords) [4]. Thus the vector quantizer \( Q \) is defined as follows: \( \mathbf{Q} : \mathbb{R}^k \rightarrow \mathbf{C} \), where \( \mathbf{C} = \{y_1, y_2, \ldots, y_N\} \) and \( y_i \in \mathbb{R}^k \). The vector quantizer \( Q \) can be decomposed into a vector encoder operation \( \varepsilon \) and a vector decoder operation \( D \). The encoder \( \varepsilon \) is a mapping from \( \mathbb{R}^k \) into an index set \( \mathcal{I} \), and the decoder \( D \) maps the index set \( \mathcal{I} \) into the codebook \( \mathbf{C} \): \( \varepsilon : \mathbb{R}^k \rightarrow \mathcal{I} \) \( D : \mathcal{I} \rightarrow \mathbf{C} \). Thus the overall operation of the VQ can be viewed as the composition of these two operations: \( Q(\mathbf{x}) = D.\varepsilon(\mathbf{x}) = D(\varepsilon(\mathbf{x})) \).

Every \( n \) point vector quantizer is associated with a partition of \( \mathbb{R}^k \) into \( n \) regions or cells, \( \mathbf{R}_i \) for \( i \in \mathcal{I} \). The \( i \)th cell is defined by \( \mathbf{R}_i = \{x \in \mathbb{R}^k : Q(x) = y_i\} \). To minimize the distortion, we use the following formula to determine the distance between two codewords: \( d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{k} (x_i - y_i)^2 \).

3 Support Vector Machines

The SVM of Vapnik has become widely established as one of the leading approaches to pattern recognition and machine learning [10]. It expresses predictions in terms of a linear combination of kernel functions centered on a subset of the training data, known as support vectors (sv).

Given the training data \( (x_i, y_i), i = 1, \ldots, n \), \( x_i \in \mathbb{R}^k \), \( y_i \in \{-1, +1\} \), SVM maps the input vector \( \mathbf{x} \) into a high-dimensional feature space \( \mathbf{H} \) through some mapping function \( \phi : \mathbb{R}^d \rightarrow \mathbf{H} \), and constructs an optimal separating hyperplane in this space. The mapping \( \phi(\cdot) \) is performed by a kernel function \( \mathbf{K}(\cdot, \cdot) \) which defines an inner product in \( \mathbf{H} \). The separating hyperplane given by an SVM is:

\[
\mathbf{w} \cdot \phi(\mathbf{x}) + b = 0
\]  

The optimal hyperplane is characterized by the maximal distance to the closest training data. The margin is inversely
Figure 1. Experimentals results
proportional to the norm of $w$. Thus computing this hyperplane is equivalent to solving the following optimization problem:

$$\text{Minimiser: } V(\mathbf{w}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C \left( \sum_{i=1}^{n} \xi_i \right)$$

(2)

where the constraint $\forall i = 1 : y_i [\mathbf{w} \cdot \phi(\mathbf{x}_i) + b] \geq 1 - \xi_i$, $\xi_i \geq 0$ requires that all training examples are correctly classified up to some slack $\xi$ and $C$ in (2) is a parameter allowing trading-off between training errors and model complexity. This optimization is a convex quadratic programming problem. Its whole dual is to maximize the following optimization problem:

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

(3)

subject to $\forall i = 1 : \sum_{i=1}^{n} y_i \alpha_i = 0$, $0 \leq \alpha_i \leq C$. The optimal solution $\alpha^*$ specifies the coefficients for the optimal hyperplane $\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}_i)$ and defines the subset $SV$ of all support vectors. An example $\mathbf{x}_i$ of the training set is a support vector if $\alpha_i^* \geq 0$ in the optimal solution. The formula (1) gives the decision function:

$$f(\mathbf{x}) = \sum_{i \in SV} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

(4)

where the threshold $b$ is computed via the unbounded $sv$. An efficient algorithm SMO [8] and many refinements [1] were proposed to solve (3).

SVM being binary classifiers, several binary SVM classifiers are induced for a multi-class problem. A final decision is taken from the outputs of all binary SVM [5].

4 The hybrid method

The idea of our method is to train SVM with different parameters from a small data set representative of the training set. The VQ scheme has been used to perform the reduction of the initial training set. The aims are 1) to reduce training time and 2) to allow earlier the reject of bad parameters. At each iteration the quantized training set size is increased and new bad parameters can be rejected. This is iterated until the error rate converges.

The kernel chosen for SVM is RBF kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ and the multi-class method is one-against-all [5]. The proposed scheme is the following:

Let $\mathbf{T} = \{(x_i, y_i)\}_{1 \leq i \leq n}$ be the training set and $\mathbf{V} = \{(x'_i, y'_i)\}_{1 \leq i \leq n'}$ the testing set and $P : \{(C_1, \sigma_1)\}_{1 \leq k \leq m, 1 \leq i \leq l}$ a set of parameters.

step 1: The training set is split into $m$ sub data sets $SB_j = \{x_i : y_i = j\}$ of size $s_j$ where $m$ is the number of classes in the training set. $\alpha$ is initialized at 2.

step 2: For each $SB_j$ a codebook $C_j$ of size $2^\alpha$ is created applying VQ ($C_j = SB_j$ if $s_j \leq 2^\alpha$). The union of all codebooks (with reference class label) yields to define a new training data set $T_\alpha = \bigcup_{j=1}^{m} C_j \times \{j\}$.

step 3: SVM are trained on databases $T_\alpha$ for each parameter $(C, \sigma)$ in $P$. Empirical $e$ error is then computed for all decision functions $f_{T_\alpha, C, \sigma}$ on the testing set $\mathbf{V}$. The minimal empirical error $e^*$ and the minimal number of support vectors $nsv^*$ are determined on all this test sets.

step 4: All the $(C_k, \sigma_k)$ for which the associated decision function $f_k$ has an empirical error $e_k \geq \varphi_1 (e^*)$ or $nsv_k \geq \varphi_2 (nsv^*)$ are removed from $P$.

step 5: $\alpha$ is incremented and steps 2 to 4 are repeated until empirical error converges ($\forall i = 1 : e_\alpha^* \geq e_{\alpha-1}^*(1 - \varepsilon)$, $\varepsilon = 1e^{-3}$) or $T_\alpha = \mathbf{T}$.

step 6: The SVM are trained on the training set $\mathbf{T}$ only on parameters remaining in $P$ after all previous steps. If no results are obtained in a given time, this last step is aborted.

We assume that step 2 provides good prototypes of examples in the training set. Thus prototype $\{x_i, y_i\} \in T_\alpha$ is representative of all its nearest examples $\{x_j, y_j\} \in T$ and $d_{\min}(x_i, x_j) \leq \rho d(x_k, x_j)$. Thus the topology of the data between the new and the initial training set is preserved. Performances obtained from the training process of SVM applied to $\mathbf{T}$ and $T_\alpha$ are very closed. Many criteria to estimate the performance of SVM exist [2]. Single validation estimate $e$ and support vector count $nsv$ are the most commonly used. $\varphi_1$ and $\varphi_2$ allow the rejection of bad parameters (in our study $\varphi_1(\mathbf{x}) = \varphi_2(\mathbf{x}) = 2\mathbf{x}$).

5 Experiments

Three well known databases are used: shuttle, satimage and letter. All training and testing data sets are normalised between 0 and 1. The svmtorch package is used to solve multi-class SVM [3]. Table 1 provides description of the data set. Figures 1(a) and 1(c) show the influence of the criteria $e$ and $nsv$ with respect to $(C, \sigma)$ for the shuttle database.

<table>
<thead>
<tr>
<th>Problem</th>
<th>training</th>
<th>testing</th>
<th>class</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>satimage</td>
<td>4435</td>
<td>2000</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>letter</td>
<td>15000</td>
<td>5000</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>shuttle</td>
<td>43500</td>
<td>14500</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1. Data Sets Description.

Figure 1(e) shows that training time is strongly linked to the number of support vector $nsv$. The reason is that the
SMO algorithm uses shrinking and caching techniques [1] to reduce computational time. Those techniques are not efficient when nsv increases. In contrast figures 1(b) and 1(d) show the found values with our method (α = 4). Ones note that each region of interest is better identified. Note that surfaces are coarser with the small training set. The reason is that SVM algorithm has found optimal decision function using a great number of sv when the training set size is large and parameters far from the optimal ones. The obtained rate of the decision function is satisfactory, but illustrates an over-fitting phenomenon. Figure 1(f) shows the rejections of a pair parameters along the iterations. Worst parameters are early rejected. Table 2 presents classification rates and training times using our method and a classic grid search. The proposed approach greatly reduces the training time on huge databases. Moreover the classified rate is almost the same.

<table>
<thead>
<tr>
<th>Problem</th>
<th>SVM + 6 steps</th>
<th>SVM (classic)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>rate</td>
</tr>
<tr>
<td>satimage</td>
<td>8.5</td>
<td>10.7</td>
</tr>
<tr>
<td>letter</td>
<td>19</td>
<td>6.34</td>
</tr>
<tr>
<td>shuttle</td>
<td>3.8</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 2. error rate and training time (hours).

Our method has also been applied to do pixel classification problem. The latter is a part of a segmentation scheme [7]. The training and testing data sets are built from four microscopic cytology images (574*752 pixels) of bronchial tumours. For each image, a segmentation is realised by a human expert. This gives the classes (background, cytoplasm or nucleus) of all pixels. Our method has been applied to this training set which contains 1,726,592 examples each. Figure 2 shows that the classification rate is not improved while the training set size grows. However the complexity of the decision function grows linearly with the size of the training set. The reason is that this data set is greatly noised. The number of examples bounds in the margin grows with data set size. The comparison of classifiers in [7] has show that SVM best performed but very slowly (the number of support vectors was close to 1000). The decision function found with our method is 100 times faster (only 9 support vectors) and gives one identical segmentation (the small decrease of classified rate is erased by the next morphological operations, cf [7]).

6 Conclusions

A new method for the training process of SVM is proposed. The main idea is based on the use of VQ to construct example prototypes and then generate a smaller training set. We have shown that this method greatly reduces training time. Moreover this method can provide decision function of low complexity for some data sets. Future research work will investigate the extension of this method for kernel with more parameters [2]. Indeed grid search cannot work with more that two parameters. We must also measure the influence of this method with different multi-class techniques and other learning kernel classifiers [9].

References