Appendix A

Abbreviations and Symbols

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARD</td>
<td>Automatic relevance determination</td>
</tr>
<tr>
<td>EP</td>
<td>Expectation Propagation</td>
</tr>
<tr>
<td>FDA</td>
<td>Fisher discriminant analysis</td>
</tr>
<tr>
<td>IVM</td>
<td>Informative vector machines</td>
</tr>
<tr>
<td>IVM*</td>
<td>Modified informative vector machines with algorithms 3 in chapter 4</td>
</tr>
<tr>
<td>LIN</td>
<td>Linear kernels</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer perception kernels</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal component analysis</td>
</tr>
<tr>
<td>POLY</td>
<td>Polynomial kernels</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial basis function</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machines</td>
</tr>
</tbody>
</table>

Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>≈</td>
<td>Approximately equal</td>
</tr>
<tr>
<td>∝</td>
<td>Proportional to</td>
</tr>
<tr>
<td>∞</td>
<td>Infinity</td>
</tr>
<tr>
<td>b</td>
<td>Bias</td>
</tr>
<tr>
<td>A⁻¹</td>
<td>Inverse of a matrix A</td>
</tr>
<tr>
<td>B</td>
<td>Diagonal vector with noise elements</td>
</tr>
<tr>
<td>arg maxₓ f(x)</td>
<td>Value of x that lead to the maximum value of f(x)</td>
</tr>
<tr>
<td>arg minₓ f(x)</td>
<td>Value of x that lead to the minimum value of f(x)</td>
</tr>
<tr>
<td>diag(a₁,a₂,…aₙ)</td>
<td>A matrix whose diagonal elements are a₁,a₂,…aₙ, and off diagonal elements are zero</td>
</tr>
<tr>
<td>eₙ</td>
<td>Vector with value 1 as nth element and all other are zero</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>exp(x)</code></td>
<td>Exponential of <code>x</code></td>
</tr>
<tr>
<td><code>f</code></td>
<td>Latent vectors</td>
</tr>
<tr>
<td><code>H(.)</code></td>
<td>Entropy of a distribution</td>
</tr>
<tr>
<td><code>log(x)</code></td>
<td>Logarithm base10 of <code>x</code></td>
</tr>
<tr>
<td><code>K</code></td>
<td>Kernel matrix</td>
</tr>
<tr>
<td><code>KL</code></td>
<td>Kullbach-Liebler divergence</td>
</tr>
<tr>
<td><code>L</code></td>
<td>Lagrangian function</td>
</tr>
<tr>
<td>`\mathcal{N}(\mathbf{f}</td>
<td>\mu_i, \Sigma_i)`</td>
</tr>
<tr>
<td><code>Q</code></td>
<td>Dual function of a svm</td>
</tr>
<tr>
<td><code>W</code></td>
<td>Weight matrix</td>
</tr>
<tr>
<td><code>w</code></td>
<td>Weight vector</td>
</tr>
<tr>
<td><code>x_i</code></td>
<td>Data element</td>
</tr>
<tr>
<td><code>x^T</code></td>
<td>Transpose of vector <code>x</code></td>
</tr>
<tr>
<td><code>||x||</code></td>
<td>Euclidian norm of vector <code>x</code></td>
</tr>
<tr>
<td><code>&lt;x&gt;_{p(x)}</code></td>
<td>Expectation of <code>X</code> under the probability <code>p(x)</code></td>
</tr>
<tr>
<td><code>y_i</code></td>
<td>Observation of data element <code>x_i</code></td>
</tr>
<tr>
<td><code>Z</code></td>
<td>Normalization constant</td>
</tr>
<tr>
<td><code>\lambda</code></td>
<td>Eigenvalue</td>
</tr>
<tr>
<td><code>\mu</code></td>
<td>Mean of a normal distribution</td>
</tr>
<tr>
<td><code>\Sigma</code></td>
<td>Covariance matrix of a normal distribution</td>
</tr>
<tr>
<td><code>\nabla_\theta</code></td>
<td>Gradient operator in <code>\theta</code> coordinates</td>
</tr>
<tr>
<td><code>\Phi</code></td>
<td>Cumulative Gaussian distribution</td>
</tr>
<tr>
<td><code>\beta</code></td>
<td>Inverse variance of noise</td>
</tr>
<tr>
<td><code>\Sigma_{i=1}^N a_i</code></td>
<td>Summation of elements <code>a_i</code> where <code>i = 1..N</code></td>
</tr>
<tr>
<td><code>\Pi_{i=1}^N a_i</code></td>
<td>Product of elements <code>a_i</code> where <code>i = 1..N</code></td>
</tr>
</tbody>
</table>
Appendix B

In this appendix we provide some results [18], [20] from that are used in development of informative vector machines in chapter 4.

B.1 Moments Calculations from KL

Given a probability distributions \( p(f) \) and \( \mathcal{N}(f|\mu, \Sigma) \) the Kullbach-Liebler divergence \( KL \) between them can be defined as

\[
L(\mu, \Sigma) = KL(p(f) \parallel \mathcal{N}(f|\mu, \Sigma)) = \langle \log \left( \frac{p(f)}{\mathcal{N}(f|\mu, \Sigma)} \right) \rangle_{p(f)}
\]

\[
= \langle \log (p(f)) \rangle_{p(f)} - \langle \log (\mathcal{N}(f|\mu, \Sigma)) \rangle_{p(f)}
\]

(A.1)

Differentiating with respect to \( \mu \) and making equating to zero gives,

\[
\nabla_\mu L(\mu, \Sigma) = \langle \Sigma^{-1}(f - \mu) \rangle_{p(f)} = 0
\]

(A.2)

which simplifies to

\[
\mu = \langle f \rangle_{p(f)}
\]

(A.3)

If we consider the second matrix derivative of equation (A.1) with respect to \( \Sigma \) we get

\[
\Sigma = \langle ff^T \rangle - \langle f \rangle_{p(f)} \langle f \rangle_{p(f)}^T
\]

(A.4)

B.2 Finding Update Equations

In this appendix we describe as in [20] how to arrive at a mean and covariance update irrespective of the noise model as it has employed in chapter 4.

Let \( q(x) \) be an approximated normal distribution over \( x \) with \( \mu \) and \( \Sigma \) the mean and covariance respectively.

\[
q(x) := q(x; \mu, \Sigma) := \mathcal{N}(x; \mu, \Sigma)
\]

(A.6)

With any type of noise model \( t(x) \) and posterior of \( p(x) \propto q(x) t(x) \), the normalization constant can be computed as

\[
Z := Z(\mu, \Sigma) := \int t(x) q(x; \mu, \Sigma) dx
\]

(A.7)

First in order to find how \( \mu \) behave we find the gradient of \( q(x) \) with respect to \( \mu \).
\[ \nabla_\mu q(x) = \Sigma^{-1}(x - \mu)q(x) \quad (A.8) \]

By rearranging we get
\[ xq(x) = \Sigma \nabla_\mu q(x) + \mu q(x) \quad (A.9) \]

Multiplying both sides of (4) by \( Z^{-1}t(x) \) and integrating on \( x \) gives
\[
\langle x \rangle_{p(x)} = \mu + Z^{-1} \Sigma \left[ \nabla_\mu \int t(x)q(x)dx \right]
\]
\[
= \mu + Z^{-1} \Sigma \nabla_\mu Z
\]
\[
= \mu + \Sigma \nabla_\mu \log(Z)
\]
\[
= \mu + \Sigma \mathbf{g} \quad (A.10)
\]

where \( \mathbf{g} := \nabla_\mu \log(Z) \).

The gradient of \( q(x) \) with respect to \( \Sigma \) gives
\[
\nabla_\Sigma q(x) = \frac{1}{2}( -\Sigma^{-1} + \Sigma^{-1}(x - \mu)(x - \mu)^T \Sigma^{-1} )q(x) \quad (A.11)
\]

and it can be rearranged as follows
\[
x x^T q(x) = 2\Sigma [\nabla_\Sigma q(x)] \Sigma + (\Sigma + x \mu^T + \mu x^T - \mu \mu^T)q(x) \quad (A.12)
\]

which again can be multiplied by \( Z^{-1}t(x) \) and integrating on \( x \) gives
\[
\langle xx^T \rangle_{p(x)} = \Sigma + 2\Sigma (\nabla_\Sigma \log(Z)) \Sigma + \langle x \rangle_{p(x)} \mu^T + \mu (\langle x \rangle_{p(x)}^T + \mu \mu^T) \quad (A.13)
\]
\[
= \Sigma + 2\Sigma G \Sigma + \langle x \rangle_{p(x)} \mu^T + \mu (\langle x \rangle_{p(x)}^T + \mu \mu^T) \quad (A.14)
\]

where \( \mathbf{G} := \nabla_\Sigma \log(Z) \).
Appendix C

Kernel Adatron

In our experiments we employed the Kernel Adatron [2] algorithm to solve SVM. For that we combine the dual problem given by the equation (4.3) and its constraint $\sum_{i=1}^{N} a_i y_i = 0$ where $0 \leq a_i \leq C$ for $i = 1, 2, \ldots, N$ with constant $C$ to come up as a Lagrangian to be maximized as

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j x_i^T x_j + \kappa \sum_{i=1}^{N} a_i y_i$$  \hspace{1cm} (B.1)

where $\kappa$ is a lagrangian constant.

By taking the gradient with respect to $a_i$, one will arrive at a gradient update value for learning with learning value $\eta$ as

$$\delta a_k = \eta \frac{\partial L}{\partial a_k} = \eta \left( 1 - y_k \sum_{i=1}^{N} a_i y_i x_i^T x_j + \kappa y_i \right)$$  \hspace{1cm} (B.2)

Further analysis in [3] shows that value for the learning parameter $\eta$ can be taken for Gaussian kernels in the range

$$2 > \eta > 0$$

and for the polynomial kernel from the range

$$\frac{2}{(||x||+1)^d} > \eta > 0$$

where $d$ if the degree of the polynomial kernel.

The basic algorithm of Kernel Adatron is given below.

**Algorithm (Kernel Adatron without bias)**

1. Initialize $a_i^0 = 0. \eta =$ predefined learning parameter value.
2. For $I = 1$ to $m$ repeat steps 3 and 4
3. For point $(x_i, y_i)$ calculate
   $$z_i = \sum_{i=1}^{N} a_i y_i K(x_i, x_j)$$
4. Calculate $\delta a_i^t = \eta (1 - z_i y_i)$
   4.1 If $(a_i^t + \delta a_i^t) \leq 0$ then $a_i^t = 0$
   4.2 If $(a_i^t + \delta a_i^t) > 0$ then $a_i^t \leftarrow a_i^t + \delta a_i^t$
5. Calculate the margin

\[ y = \frac{1}{2} \left( \min_{i(y_j = +1)} z_i - \max_{i(y_j = -1)} z_i \right) \]

If the margin has come to a value close to 1 or the maximum iterations are reached stop, else go to 2.
Appendix D

FACE RECOGNITION USING INFORMATIVE VECTOR MACHINES*
K.A.D.N.K. Wimalawarne¹ and C.R. De Silva¹
¹Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka

INTRODUCTION

In the recent past kernel methods have been successfully applied to face recognition. We present a novel approach in frontal face recognition with informative vector machine, a sparse Gaussian process kernel classifier. Informative vector machine has the ability to provide more sparse solutions than the widely used kernel classifiers like support vector machines.

METHODOLOGY

Our approach of classifying faces was to use Informative vector machines (IVM), a probabilistic classification algorithm. It belongs to the Gaussian process based kernel methods. In its original form IVM had problems due to random data set selections and unavailability of an optimum stopping criteria. In our research we were able to propose a method of reducing randomness with data point selections by using previous selected data points and making the point selection based on maximum norm value. We noticed that the data point sets tend to converge which lead to more stable solutions.

As a kernel method similar to support vector machines (SVM), IVM also employed kernel functions to transform data points to different domains in which the classification becomes efficient. But we found that automatic relevance determination (ARD) version of kernels to be more efficient with IVM since it adds data element level parameters. With ARD kernels we were able to reduce feature space by discarding parameters with low values. By taking the norm of the ARD parameters we were also able to propose a stopping condition for the IVM algorithm.

¹ K.A.D.N.K. Wimalawarne, Student, Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka, Tel:+94-11-2650301 Ext. 3123, email: kisnw@yahoo.com (corresponding author)
* This extended abstract was published in the 13th ERU Symposium of University of Moratuwa, 2007
EXPERIMENTS

To evaluate the performance of IVM in face recognition we have conducted experiments with ORL face database, a standard face database which consists of frontal face images of 40 subjects, 10 from each totaling 400 face images.

We designed our experiment by choosing five images from each subject for the training set and the other five were used as the test set. Images were selected one after the other from their natural ordering for each set. All algorithms applied in this experiment used binary classification with five from each subject as positive and other 195 as negative sets. Experiments were repeated for all 40 subjects.

Our experiments were conducted with IVM using different kernels types. Automatic relevance determination (ARD) version of radial basis functions (RBF), multilayer perceptron (MLP) and linear kernels (LIN) were used as kernels for IVM. For comparisons we employed the standard support vector machines (SVM) with RBF and polynomial kernels (POLY).

RESULTS

Table 1 Experiment results for ORL face database

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg. vectors</th>
<th>% Correct</th>
<th>Avg. dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVM (RBF ARD)</td>
<td>19</td>
<td>99.34</td>
<td>3430</td>
</tr>
<tr>
<td>IVM (MLP ARD)</td>
<td>19</td>
<td>99.21</td>
<td>3911</td>
</tr>
<tr>
<td>IVM (LIN ARD)</td>
<td>19</td>
<td>99.16</td>
<td>2873</td>
</tr>
<tr>
<td>SVM (RBF)</td>
<td>40</td>
<td>99.71</td>
<td>10304</td>
</tr>
<tr>
<td>SVM (POLY)</td>
<td>33</td>
<td>99.42</td>
<td>10304</td>
</tr>
</tbody>
</table>

It’s evident from results in table 1 that none of the IVM classifiers could outperform SVM in accuracy. But all IVM classifiers were able to perform above 99%. It’s also evident that all RBF kernel types can perform better than other type of kernel.

We identified that IVM was very capable of generating sparse results than SVM since it only needed average of 19 data vectors to train the algorithm where SVM needed 33 vectors for RBF kernels and 40 vectors for polynomial (POLY) kernels.

One interesting observation of the experiment was the dimension reduction that was achieved with IVM. A significant reduction of about 70% was possible without any performance loss.
CONCLUSION

Comparable recognition accuracies of IVM compared to commonly used SVM classifier indicates that it can be used in face recognition systems. The most significant result of our research was the discovery of dimension reductions of feature space with frontal face images with IVM ARD classifiers. The combination of sparse solutions and dimension reductions of IVM can help to reduce the storage requirements. It can also help to improving the face recognition speed since the number of computations may become less for a new data element. Further research in face recognition using IVM may lead to solutions for embedded systems and systems with low storage capacities.

REFERENCES


References


