Chapter 4

Proposed method

In this chapter we describe our IVM, our proposed method for face recognition. First we give an overview of the algorithm. Next we describe the original development of the algorithm found in the literature. Several problems that are not suitable for face recognition are then identified and our attempts and proposals to solve those problems are discussed.

4.1 Introduction

Lawrence et al [17] - [20], [30] invented Informative vector machine, a sparse Gaussian process method for classification and regression. Unlike full Gaussian process methods which take all data points for training IVM selects a set of data points using information theoretic approaches and train its hyperparameters relative to them.

Figure 4.1 describes behavior of IVM in a typical classification problem. If we consider two sets (circles and crosses) which we need to find a decision boundary to separate them. The active data points selection proposed in IVM can be thought of as the filled data elements which can be used for training the algorithm. These data point selection is based on an informative theoretic approach which will be described later in the chapter. This approach of selecting a small subset of data points and makes IVM gain sparsity and faster training.

![Figure 4.1. Learning decision boundary in IVM.](Image from Neil Lawrence’s IVM software)
IVM algorithm has three major components

- Data point selection
- Kernel parameter updates
- Noise parameter updates

Before the data point selection process we first have to identify the probability distributions of the data points. For this two approximation methods can be employed. One is the Assumed density filtering (ADF) [19] - [21], and the other is Expectation Propagation (EP) [30]. In this chapter we describe the ADF approach in developing IVM and before discussing improvements.

4.2 Assumed-density Filtering (ADF) Approximation

In an analogous way to how we developed Gaussian process in the last chapter for a set of a training set \( \{(x_i, y_i), i = 1, 2 \ldots N\} \), we can assume a latent function values \( f = [f(x_1), \ldots, f(x_N)] \) for all N data points which would assume to take a normal distribution \( \mathcal{N}(f|0,K) \) with mean \( 0 \) and kernel matrix \( K \). The relationship between latent functions \( f_n = f(x_n) \) and observables \( (y_n) \) can be describe as a noise model \( p(y_n|f_n) \) [20]. Combining the prior and the noise models can give us the joint distribution over data and latent variables as

\[
p(y, f) = \mathcal{N}(f|0,K) \prod_{n=1}^{N} p(y_n|f_n)
\]

For the convenience of notations this equations can be represented as

\[
p(y, f) = \prod_{n=0}^{N} t_n(f)
\]

where \( t_0(f) = \mathcal{N}(f|0,K) \) and \( t_n(f) = p(y_n|f_n) \).

IVM algorithms trains by taking a data point one at a time from the complete data set based on a data selection process for the applications of ADF approximations which will be discussed later. To assist this process two indexes are maintained. One is \( I \) which stores the indexes currently selected data sets and the other is \( J \) which contains the indexes those are not included for the approximations.

We can approximate the distribution on \( f \) after the inclusion of \( i^{th} \) element as \( q_i(f) \) as

\[
q_i(f) \propto \mathcal{N}(f|0,K) \mathcal{N}(m_i|f_i, B_i^{-1}) \approx p(f|X_i,y_i,0)
\]

This again can be assumed to behave as a normal distribution as

\[
q_i(f) = \mathcal{N}(f|\mu_i,\Sigma_i)
\]

where \( \mu_i \) and \( \Sigma_i \) means and variance after \( i^{th} \) data set inclusion that need to be calculated.
Using ADF as shown in Figure 4.2, the posterior distribution after the $n^{th}$ data point inclusion at $i^{th}$ iteration be represented by $p_i^*(\mathbf{f})$

$$p_i^*(\mathbf{f}) \propto q_{i-1}(\mathbf{f})t_{n_i}(\mathbf{f})$$  \hspace{1cm} (4.5)

![Diagram](#)

$q_{i-1}(\mathbf{f}) \quad p_i^*(\mathbf{f})$

$t_{n_i}(\mathbf{f}) \quad t_{n_2}(\mathbf{f}) \quad \ldots \quad t_{n_{i-2}}(\mathbf{f}) \quad t_{n_1}(\mathbf{f}) \quad \ldots \quad t_{N_i}(\mathbf{f})$

$q_i(\mathbf{f})$

Figure 4.2 Iterative probability approximation procedure of ADF

In order to find the approximation for $q_i(\mathbf{f})$, we can minimize the Kullbach-Liebler divergence between the two distributions, $q_i(\mathbf{f})$ and $p_i^*(\mathbf{f})$

$$KL(p_i^* || q_i) = \langle \log \frac{p_i^*(\mathbf{f})}{q_i(\mathbf{f})} \rangle = - \int p_i^*(\mathbf{f}) \log \frac{p_i^*(\mathbf{f})}{q_i(\mathbf{f})} \, d\mathbf{f}$$  \hspace{1cm} (4.6)

After solving (5) as described in detail in Appendix A we can arrive at relevant moment as

$$\mu_i = \langle \mathbf{f} \rangle_{p_i^*(\mathbf{f})} \hspace{1cm} (4.7)$$

$$\Sigma_i = \langle \mathbf{f} \mathbf{f}^T \rangle_{p_i^*(\mathbf{f})} - \langle \mathbf{f} \rangle_{p_i^*(\mathbf{f})} \langle \mathbf{f} \rangle_{p_i^*(\mathbf{f})}^T \hspace{1cm} (4.8)$$

With the same approximation in (4.5) we can define a normalization constant as

$$Z_i = Z_i(\mu_{i-1}, \Sigma_{i-1}) := \int q_{i-1}(\mathbf{f})t_{n_i}(\mathbf{f}) \, d\mathbf{f}$$  \hspace{1cm} (4.9)

It has been shown in [21] and appendix A that regardless of the noise model updates of the mean and covariance can be written as

$$\mu_i = \mu_{i-1} + \Sigma_{i-1}g_i$$  \hspace{1cm} (4.10)

$$\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1}(g_i g_i^T - 2G_i)\Sigma_{i-1}$$  \hspace{1cm} (4.11)

where $g_i = \nabla_{\mu_{i-1}} \log (z_i(\mu_{i-1}, \Sigma_{i-1}))$ and $G_i = \nabla_{\Sigma_{i-1}} \log (z_i(\mu_{i-1}, \Sigma_{i-1}))$.

Since $t_{n_i}$ is acting on only one element of $\mathbf{f}$ at a time we get $t_{n_i}(\mathbf{f}) = t_{n_i}(\mathbf{e}_n^T)$ where $\mathbf{e}_n$ is the $n^{th}$ unit vector. This will lead to an efficient representation of (10) and (11) as

$$\mu_i = \mu_{i-1} + g_{n_i} \Sigma_{i-1} \mathbf{e}_n$$  \hspace{1cm} (4.12)

$$\Sigma_i = \Sigma_{i-1} - (g_{n_i}^2 - 2G_{n_i}) \Sigma_{i-1} \mathbf{e}_n \mathbf{e}_n^T \Sigma_{i-1}$$  \hspace{1cm} (4.13)

and the normalization constant as

$$Z_{n_i}(\mu_{i-1,n_i}, \Sigma_{i-1,n_i}) = \int t_{n_i}(\mathbf{f})N(\mathbf{f} \mid \mu_{i-1,n_i}, \Sigma_{i-1,n_i}) \, d\mathbf{f}$$  \hspace{1cm} (4.14)
where $\mu_{i-1,n_i}$ and $\zeta_{i-1,n_i}$ are the $n^{th}$ elements at iteration $i$ of $\mu_{i-1}$ and diagonal vector of $\Sigma_{i-1}$,

$$g_{n_i} := \frac{d}{d\mu_{i-1,n_i}} \log(Z_{n_i}(\mu_{i-1,n_i}, \zeta_{i-1,n_i})/d\mu_{i-1,n_i})$$

(4.15)

and

$$G_{n_i} := \frac{d}{d\zeta_{i-1,n_i}} \log(Z_{n_i}(\mu_{i-1,n_i}, \zeta_{i-1,n_i})/d\zeta_{i-1,n_i})$$

(4.16)

With (4.13) one can define

$$v_{n_i} = g_{n_i}^2 - 2G_{n_i}$$

(4.17)

By taking a Gaussian noise model between $y_{n_i}$ and $f_{n_i}$ as

$$p(y_{n_i} | f_{n_i}) = \mathcal{N}(y_{n_i} | f_{n_i}, \beta_{n_i})$$

(4.18)

where $\beta_{n_i}$ is the inverse noise model for $n^{th}$ elements at $i^{th}$ iteration.

This makes the normalization constant

$$Z_{n_i}(\mu_{i-1,n_i}, \Sigma_{i-1,n_i}) = \mathcal{N}(y_{n_i} | \mu_{i-1,n_i}, \zeta_{i-1,n_i} + \beta_{n_i}^{-1})$$

(4.19)

By substituting to (4.15) and (4.16) we get

$$g_{n_i} = \frac{y_{n_i} - \mu_{i-1,n_i}}{\beta_{n_i}^{-1} + \zeta_{i-1,n_i}}$$

(4.20)

$$G_{n_i} = -\frac{1}{2(\beta_{n_i}^{-1} + \zeta_{i-1,n_i})} + \frac{1}{2} g_{n_i}^2$$

(4.21)

### 4.3 Binary Classification

Since this thesis is considering only the binary classification problem below we briefly describe the use of probit noise model $\Phi(\lambda y_{n_i} (f_{n_i} + b))$ where $\lambda, b$ are parameters for slope and bias respectively and $\Phi(z)$ represent the cumulative Gaussian given by

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{t^2}{2}} dt$$

(4.22)

derive update equations as they appear in [22].

Normalization constant for the $n_i$ element of the probit noise model can be written as

$$Z_{n_i}(\mu_{i-1,n_i}, \zeta_{i-1,n_i}) = \int \Phi(\lambda y_{n_i} (f_{n_i} + b)) \mathcal{N}(f_{i-1,n_i} | \mu_{i-1,n_i}, \zeta_{i-1,n_i}) df_{n_i}$$

(4.23)

After rearrangements and integrations we can obtain

$$Z_{n_i}(\mu_{i-1,n_i}, \zeta_{i-1,n_i}) = \Phi(u_{i-1,n_i})$$

(4.24)

where

$$u_{i-1,n_i} = \zeta_{i-1,n_i} (\mu_{i-1,n_i} + b)$$

(4.25)


\[ c_{i-1,n_i} = \frac{y_{n_i}}{\sqrt{\lambda^{-2} + \zeta_{i-1,n_i}}} \]  

(4.26)

If we derive the derivatives of the log partition function with respect to \( \mu_{i-1,n_i} \) we arrive at

\[ g_{n_i} = \frac{c_{i-1,n_i} \mathcal{N}(f_{i-1,n_i} \mid 0, 1)}{\Phi(u_{i-1,n_i})} \]  

(4.27)

and the derivative with respect to \( \zeta_{i-1,n_i} \) will give

\[ G_{n_i} = -\frac{1}{2} g_{n_i} \mu_{i-1,n_i} c_{i-1,n_i} \]  

(4.28)

By combining these results (4.20) and (4.21) for ease of notation with respect to (4.17) we define

\[ v_{n_i} = \left( g_{n_i} g_{n_i}^T - 2 G_{n_i} \right) = g_{n_i} \left( g_{n_i} + u_{i-1,n_i} c_{i-1,n_i} \right) \]  

(4.29)

In [21] another Gaussian approximation is assumed for \( f_{n_i} \) as \( \mathcal{N}(m_{n_i} \mid f_{n_i}, \beta_{n_i}) \) by replacing \( y_{n_i} \) with a parameter \( m_{n_i} \) and inverse variance \( \beta_{n_i} \) that gives the following update equations

\[ m_{n_i} = \frac{g_{n_i}}{\zeta_{i-1,n_i}} + \mu_{i-1,n_i} \]  

(4.30)

\[ \beta_{n_i} = \frac{v_{n_i}}{1 - v_{i-1,n_i} \zeta_{i-1,n_i}} \]  

(4.31)

These two updates introduce two parameters sets \( \mathbf{m} = \{ m_i \mid i = 1..d \} \) and \( \mathbf{\beta} = \{ \beta_i \mid i = 1..d \} \) with the dimension \( d \) of the data points to be selected for training.

### 4.4 Data Point Selection

Unlike in SVM where the sparse solution is found after training the whole data set, IVM uses a greedy selection of data point using information theoretic approaches. It selects data points one by one taking the entropy change into consideration. The original algorithms [18],[20] propose that the data point with the maximum entropy [31] change to be selected, but chose a data point randomly in situation where more than one data point have the same values. Complete random selections and semi-random selection are also been used in data point selection [30].

The entropy of the posterior distribution which is a Gaussian distribution can be written as

\[ H(\mathcal{N}(\cdot, \mu, \Sigma)) = \frac{N}{2} (1 + \log 2\pi) + \frac{1}{2} \log |\Sigma| \]  

(4.31)

The entropy change for the above distribution is,

\[ \Delta H_{i,n_i} = \frac{1}{2} \log |\Sigma_i| + \frac{1}{2} \log |\Sigma_{i-1}| = \frac{1}{2} \log |\Sigma_i \Sigma_{i-1}| \]  

(4.32)
\[ \log | \mathbf{I} - \nu_i \Sigma_i \mathbf{e}_{n_i} \mathbf{e}_{n_i}^T | = \frac{1}{2} \log | (\mathbf{I} - \nu_i \Sigma_i) | \quad (4.33) \]

\[ \log | \mathbf{I} - \nu_i \zeta_{i-1,n_i} \mathbf{I} | = \frac{1}{2} \log | (\mathbf{I} - \nu_i \zeta_{i-1,n_i}) | \quad (4.34) \]

In order to make the IVM algorithm more computationally and storage feasible an efficient representation has been developed. By considering the column based representation of the covariance matrix as \( \Sigma_i = [\mathbf{s}_{i1}, \ldots, \mathbf{s}_{iN}] \) where \( \mathbf{s}_{ij} \) is the \( j \)th column of \( i \)th covariance matrix, an efficient update equation for posterior mean \( (\mathbf{\mu}_i) \) and diagonal posterior covariance columns \( (\zeta_i) \) can be derived as below.

\[ \zeta_i = \zeta_{i-1} - \nu_{i,n_i} \left( \text{diag} (\mathbf{s}_{i-1,n_i} \mathbf{s}_{i-1,n_i}^T) \right) \quad (4.35) \]

\[ \mathbf{\mu}_i = \mathbf{\mu}_{i-1} - \nu_{i,n_i} \mathbf{s}_{i-1,n_i} \quad (4.36) \]

The posterior covariance \( \Sigma_i \) can also be represented in a more efferent way. We note that initial prior covariance id \( \Sigma_0 = \mathbf{K} \) and any subsequent covariance can be represented as

\[ \Sigma_i = \mathbf{K} - \mathbf{M}_i^T \mathbf{M}_i \quad (4.37) \]

where \( \mathbf{M}_i \in \mathbb{R}^{1 \times N} \) is a matrix created by adding \( \sqrt{\nu_{k,n_i}} \mathbf{s}_{k-1,n_i} \) to the \( k \)th column. Multiplying (4.37) from right by \( \mathbf{e}_n \) leads us to a more efficient covariance column vector update

\[ \mathbf{s}_{i-1,n_i} = \mathbf{K}_{n_i} - \mathbf{M}_i^T \mathbf{M}_{i-1} \mathbf{m}_{i-1,n_i} \quad (4.38) \]

Based on the entropy change and the results derived above and the IVM data point selection algorithm as it is given in [18] and [20] takes the form of algorithm 1.

**Algorithm 1**

Require: \( d \) = number of active data points

1. Initialize \( \mathbf{m}, \mathbf{\beta}, \mathbf{\mu} \) to zero vectors. Take \( \zeta_0 = \text{diag}(\mathbf{K}) \) and make \( \mathbf{S}_0 \) an empty matrix.

   Make tow indexes vectors \( \mathbf{J} = \{1, \ldots, N\} \) and \( \mathbf{I} \) an empty vector.

2. For \( k = 1 \) to \( d \) repeat steps 3 to 8

3. For all \( n \in \mathbf{J} \) repeat 3.1 and 3.2

   3.1 Compute \( g_{k,n} \) and \( v_{k,n} \) using (4.27) and (4.29)
Compute $\Delta H_{kn}$ according to (4.34)

4. $n_k = \arg \max_{n \in j} \Delta H_{kn}$

5. Update $m_{nk}$ and $\beta_n$ using (4.30) and (4.31)

6. Compute $\zeta_l$ and $\mu_k$ using (4.35) and (4.36)

7. Append $\sqrt{v_k} s_{k-1,n_k}$ to $M_k$ to form $M_k$

8. Add $n_k$ to $l$ and remove $n_k$ from $j$

The active data selection and the above simplified representations help to lower the computational and storage requirements. According to [20] for an active set size of $d$ the computational complexity becomes $O(d^2 N)$ and the storage requirement $O(dN)$.

### 4.5 Kernel Parameter Updates

After a execution of algorithm 1 a data set, $I$, with dimension of $d$ (pre-specified) will be obtained by data selection process. Apart from that it will also generate updated sets $m$ and $\beta$ relative to the data points selected. These values are helpful in finding the probability of observables values relative to the selected data point $y_I$ as

$$p(y_I) = \mathcal{N}(m_I, 0, K_I + B_I^{-1})$$

(4.39)

where $K_I$ is the kernel matrix for the $I$ data points and hyper parameters, $m_I$ represents $m$ values of $I$ and $B_I$ is made by making $\beta$ its diagonal elements and making other elements zero.

Since (4.39) is a normal distribution by obtaining the log we can easily optimize the parameter of the kernel. Lawrence and et al has proposed the use of scaled conjugate gradients methods as a non linear optimizers [23] which avoids expensive Hessian matrix inversion.

### 4.6 Noise Parameter Updates

To achieve better results, parameters of the noise function need to be updated at each iteration as it updates the kernel parameters. In order to optimize the noise parameter the variational lower bound of the on likelihood is used. ($\theta$ - parameters of the noise model). Variation lower bound would give

$$\sum_{n=1}^N \int q_d(f_n) \log p(y_n|f_n, \theta) p(f_n) df_n - \sum_{n=1}^N \int q_d(f_n) \log q(f_n) df_n$$

(4.40)

The relative term in this bound can be represented as
\[
\sum_{n=1}^{N} \int q_d(f_n) \log p(y_n|f_n, \theta)
\]  
(4.41)

Without much loss of generality the above lower bound can be converted to an upper bound by

\[
L(\theta) = \sum_{n=1}^{N} \log \int q_d(f_n) \log p(y_n|f_n, \theta)
\]

\[
= \sum_{n=1}^{N} \log Z_n
\]  
(4.40)

Algorithm 2 shows the general strategy of optimization where kernel and noise parameters are optimized sequentially with scaled conjugate optimizers [23], [24].

**Algorithm 2**

Require: Require \(d\) active points. T iterations.

1. For \(i = 1\) to \(T\) repeat steps 2 to 4.
2. Select points using Algorithm 1.
3. Optimize kernel parameters by maximizing the approximation to the likelihood using scaled conjugate optimizer.
4. If noise parameter updates are required then
   4.1 Select points using Algorithm 1.
   4.2 Optimize noise parameters by maximizing the approximation to the likelihood using scaled conjugate optimizer.

**4.7 Optimization Strategy**

One major factor that affects the performance of IVM is the data selection process in algorithm 1 which is based on finding the data point with the maximum entropy change given by the equation (4.34). Though this seems convenient from a theoretical point of view implementations create many problems. This is because at the first data selection and in some other selections entropy change of more than one data points takes the same highest value. In such situations no conditions are specified to select a data point and the only option is to select one randomly.
As a consequence of this random selection process inconsistent data sets and hyperparameter values are generated at the end of the training process for the same training data set. With our initial experiments we found that in problems like face recognition where the training set is small and data dimensions are high the resulting trained data sets and hyperparameter values vary a lot and end up giving faulty decision boundaries which gives many classification errors.

Another problem in IVM was that it does not specify an optimal number of data points for a training process in algorithms 1. Different experiments such as digit classifications found in [21] had used a pre-determined number of data points. That approach is not very useful in practical situations and a mechanism to find the optimal number of data points automatically though the training process would make the IVM more efficient.

These problems in IVM were investigated in our research and different strategies were tested to overcome these. Below we describe some of them and some propped solutions.

4.7.1 Common Data Points and Expectation Propagation

One observation of the data point selection and optimization procedure in IVM is that some data points are common or have a higher frequency of repeating than others. This led us to modify the algorithms in such a way that first IVM runs on the training set for specific number of iterations and collects the common data points which are used later as the active dataset to retrain to find a more stable classifier.

The strategy we were adopting was to apply Expectation Propagation (EP) [22], [26] to the common data points as a Gaussian process and optimize hyper parameters. EP was developed by Minka in his PhD thesis [22], which is an unification and generalization of assumed-density filtering and loopy belief propagations. Kuss in his PhD thesis [14] showed that EP can be employed in Gaussian process models as an effective classification mechanism. His work also showed that EP with parameter optimizations can be applied to such problems. These works motivated us to take the common data sets and apply EP with hyper parameter optimizations.

Though this approach seemed effective at first it did not succeed due to several reasons. One major problem was the difficulty of identifying common data sets. Due to random selections active data sets that were selected in different execution of IVM on the same data set under same conditions (iteration and number of active points) were different. This resulted in different data points and their frequencies to be different. This made it difficult to find a mechanism of selecting active data points for further processing.
The other major problem with the above approach was the even if we were able to get some data points to apply Expectation Propagation, the hyperparameters had to be pre-selected so that EP would not diverge. Most of the time, use of hyperparameter values obtained by IVM were not appropriate for EP. Hyper parameter update using gradient methods with IVM resulted in even worse results. This was due to EP only considering a set of data points and hyper parameters were trained only considering those points that would misclassify many data points. Due above reason this approach was abandoned.

4.7.2 Reducing Randomness

As mentioned previously random selections in IVM data point selection is inevitable. Especially at the very first active data point selection more than one data points get the same entropy value since probability of the data sets are not specified. During the iteration process of training several random selections may have to be made since several data points can have the same maximum entropy values.

An approach that was adapted to reduce randomness if complete removal is not possible was to choose the data point with the maximum norm when facing a situation where more than one data point has same highest entropy value. Also we propose to consider data selections in previous iterations. This approach is mainly motivated by markov process. This also requires keeping the history of the data selections during the training process. The approach is given below in the algorithm 3. This approach can be used in place of \( n_k = \arg \max_{n \in J} \Delta H_{kn} \). In any case proper normalization of data is needed for proper convergence.

**Algorithm 3**

1. If it is the initial iteration initialize \( \mathbf{H} \) as an empty vector of dimension \( d \) where \( d \) is the number of vectors to be selected else uses the stored vector. Initialize \( \text{indexes} \) and \( \text{index2} \) to empty matrices.
2. Calculate entropy changes of all data elements not included as active data points.
3. Find the maximum entropy change among data points and \( \text{indexes} \) that has the maximum entropy change value.
4. If it is the very first iteration select a data point which has the maximum norm \((\max_{x \in \text{indexes}} \| \mathbf{x} \|)\). Store the index in \( \mathbf{H} \). If the iteration is not the first one go to 5.
5. If the size of \( \text{indexes} \) is one select the data point of the \( \text{indexes} \) and if size of \( \text{indexes} \) is more than one go to 6.
6. Search in $H$ to find whether the indexes are chosen earlier and let $index2$ contain the previously selected matching data points
   
   6.1 If $index2$ only contains one element then select that data point.
   
   6.2 If $index2$ contains more than one element select a data point from $index2$ randomly.

7. If no matching previous selection in $H$, select an element from indexes based on maximum norm ($\max_{x \in indexes} ||x||$).

Use of algorithm 3 showed us that randomness can be reduced and a constant data selection is achievable most of the time. This is due to the fact we are using maximum norm values to select data point fixes some data selections in iterations.

**4.8 Stopping Condition**

Another drawback of IVM is the lack stopping condition of optimization. This is evident from algorithms 1 and 2 and the approach taken in experiments found in literature is to take empirical solution. Since there was no theoretical background to find a stopping condition several trial and error approaches were tested during our research. Many experiments were conducted with IVM to identify any pattern of convergence among data sets and parameters.

Some of the data tested for convergence were
   
   - Hyperparameters e.g.- variance of kernels
   - Entropy of data points
   - Norm of highest active data point
   - Norm of average active data point

After many experiments we found that none of the above showed any tendency of convergence that can lead to a proper stopping condition. But after several experiments we found that the norm of the ARD matrix has a tendency to converge. Provided the correct number of active points is selected the norm of the ARD matrix seems to converge most of the time.

**4.9 Kernel Functions**

For IVM any positive definite kernel can be used as the covariance function. Unlike in SVM it does not have to follow the Mercer’s theorem explained in the last chapter. All kernels can be used as standard kernel functions or as automatic relevance determination (ARD) [18], [20]
kernel functions. In general ARD kernels can be formed by replacing inner products of feature vectors with parameterized matrix. In other words given a matrix $A$ which we can call the ARD matrix, any inner product of the form $x_i^T x_j$ can be replaced with $x_i^T A x_j$ to form the ARD kernel. The matrix $A$ can be taken as a full function or as a diagonal matrix. In order to save the computational resources in our experiments we have chosen a diagonal matrix. Some of the kernels we have used in our experiment are listed below.

4.9.1 Linear Kernels
The inner products of feature vectors forms the linear kernel,

$$K_{lin}(x_i, x_j) := \theta x_i^T x_j$$

(4.41)

and the ARD version

$$K_{lina}(x_i, x_j) := \theta x_i^T A x_j$$

(4.42)

where $\theta$ the process variance which controls the scale of the output functions.

4.9.2 Radial Basis Function (RBF) Kernels
The most popular kernel in pattern recognition, the RBF kernel originated from the radial basis neural networks.

$$K_{rbf}(x_i, x_j) := \theta \exp\left(-\frac{\lambda}{2} (x_i - x_j)^T (x_i - x_j)\right)$$

(4.43)

The ARD version is

$$K_{rbff}(x_i, x_j) := \theta \exp\left(-\frac{1}{2} (x_i - x_j)^T A (x_i - x_j)\right)$$

(4.44)

The hyperparameters $\lambda$ is the inverse width parameter and $\theta$ is the process variance.

4.9.3 Multi-layer Perceptron (MLP) Kernel
Though not a very commonly used kernel, the multi-layer perceptron is another kernel function that can be robustly used with informative vector machines.

$$K_{mlp}(x_i, x_j) := \theta \sin^{-1}\left(\frac{wx_i^T x_j + b + 1}{\sqrt{(wx_i^T x_i + b + 1)(wx_i^T x_j + b + 1)}}\right)$$

(4.45)

Here the parameters $w$ act as the weight variance and $b$ as the bias variance with $\theta$ as the process variance. The ARD version is given below.
4.9.4 Choosing between Kernels

There is no theory to decide which kernel to be used for a specific problem like face recognition. Ideal solution would be to determine by an experimental process. One aspect to consider is the number of hyper parameter and the number of mathematical operations that are need for each kernels since that can affect the computational times and storage times. In that sense ARD kernels have more storage and computational requirements than their respective non-ARD kernels. Considering all of these facts we can say that computational times and storage times Linear, RBF and MLP in both forms of kernels would have an increasing order.

One possibility with ARD kernels is that after training some values of A may tend to have low values close to zero. In case where A is a diagonal matrix we can judge the importance of the corresponding element in data vector as a feature element for the classification by the relative value of the diagonal element. In this respect element that has a values close to zero can be assumed to be less important than others. In our experiments in chapter 6 we will employ this approach as a mechanism of dimension reduction.

All kernel parameters has to be constrained to positive values and diagonal elements of ARD matrix entries, \( A = \text{diag}(\omega) \) are constrained with the sigmoid transformation [11],[21]

\[
\omega_i = \frac{1}{1 + \exp(-\omega)}
\]

4.10 Method of Prediction

In order to predict the classification of a test data the mean value as \( \mu_* = k_*^{-1}K^{-1}\Sigma B y \) has to be calculated. \( k \) is calculated by taking the kernel functions with the test data point and active data points and \( K \) is the symmetric matrix based on kernel functions of all active data points. \( B \) is the diagonal matrix with \( \beta \) values comprising its diagonal elements. As we saw in the last chapter \( \Sigma = (B + K^{-1})^{-1} \). \( y \) is the corresponding covariance for data points. By substitution we can calculate the mean \( \mu \) of the test data point. For binary classification the test data point can be classified depending on the sign of \( \mu \).
4.11 Summary

A detailed discussion about informative vector machines was presented throughout the chapter. Initial part of the chapter was focused on the development of IVM using assume density filtering (ADF) approximation combined with data point selection and parameter optimizations. Algorithms 1 and 2 were listed as given in the original format. Later section discussed problems in IVM related to optimizations, random data selection and the lack of a proper stopping condition. Several methods that we have considered in attempts to solve these problems are discussed and their shortcomings are described. Finally we proposed an algorithm (algorithm 3) to overcome these problems based on tracking of data points in iterations and taking norm of data points to fix the data point selection. In the final sections we describe different kernels that be used with informative vector machines and also describe about the automatic relevance determination (ARD) kernels which are unique to Gaussian process classification algorithms.