e-PGPathshala

Subject : Computer Science

Paper: Machine Learning

Module: Support Vector Machines-II

Module No: CS/ML/18

Quadrant I – e-text

Welcome to the e-PG Pathshala Lecture Series on Machine Learning. In this module we will be discussing more details about support vector machine, the basics of which we have already discussed in the last module.

Learning Objectives:

The learning objectives of this module are as follows:

- To understand the concept margins of SVM
- To learn more about soft margin
- To classify multiple classes using SVM
- To understand some applications of SVM

18.1 Introduction

SVM has roots in statistical learning and is considered one of the most important discoveries in machine learning since it also works well for high dimensional data. Most importantly it represents the decision boundary using a subset of training examples called support vectors. It is based on the concept of maximal margin hyper planes. Margin of a classifier is defined as the minimum distance to any example and in SVM the decision boundary which maximizes the margin is chosen.

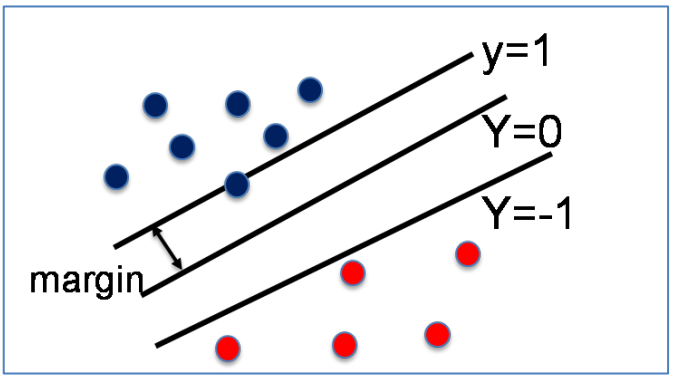

Figure 18.1 Margin and SVM

In this module we will discuss more in detail about soft margin, the use of SVM to solve multi-class problems and finally some applications of SVM.

18.2 Soft Margin

Consider a two class classification problem that uses a linear model

$$y(x) = w^T(x) + b$$

followed by a threshold function. Assume that training data are linearly separable. As we have already discussed there are multiple hyper planes that separate the data points. There are many possibilities for such hyperplanes. The task is to choose the optimal separating hyperplane. There are distance of $(d_+ + d_-)$ where d_+ is the shortest distance of a positive example from the hyper plane and d_- is the shortest distance of a negative example from the hyper plane.

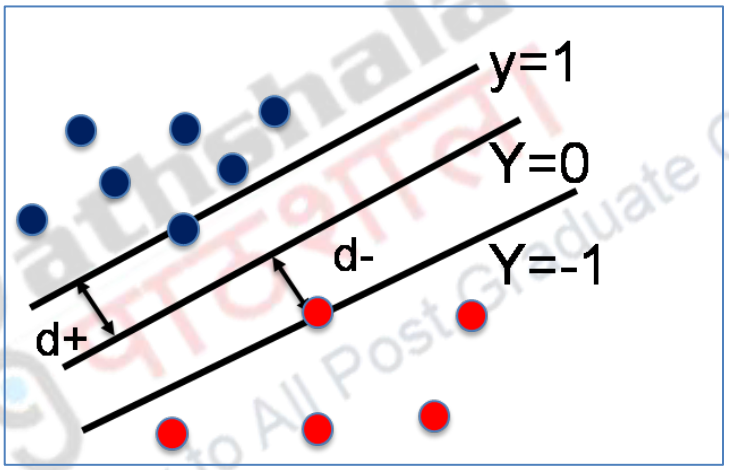

Figure 18.2 Optimal Separating hyperplane

Now if the problem is not linearly separable then we need to introduce slack variables and we need to minimize:

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

18.2.1 What is Soft Margin?

The SVM must not result in too many misclassifications. Therefore we introduce soft margin where a user specified parameter can control roughly, how many examples are allowed to violate the separating hyper plane and how far across the plane the examples can be considered. Setting this parameter is a tradeoff where we still want to try to achieve a large margin with respect to the correctly classified examples. Hence, the soft margin parameter specifies a trade-off between hyperplane violations and the size of the margin.

Maximum margin allows SVM to select among multiple candidate hyper planes. Soft margin accepts some misclassifications of the training examples. A soft margin can be obtained by adding a constant factor to the kernel function output, defining a priori, an upper bound on the size of the training set weights.

18.2.2 Linear, Soft-Margin SVMs

Soft –Margin SVM tries to maintain slack variables ξ_i to zero while maximizing margin. It does not minimize the *number* of misclassifications but the sum of distances from the margin hyper planes. In the case of soft margin, we allow "error" ξ_i in classification. We use "slack" variables ξ_1 , ξ_2 …….. ξ_n (one for each sample) (Figure 18.3).

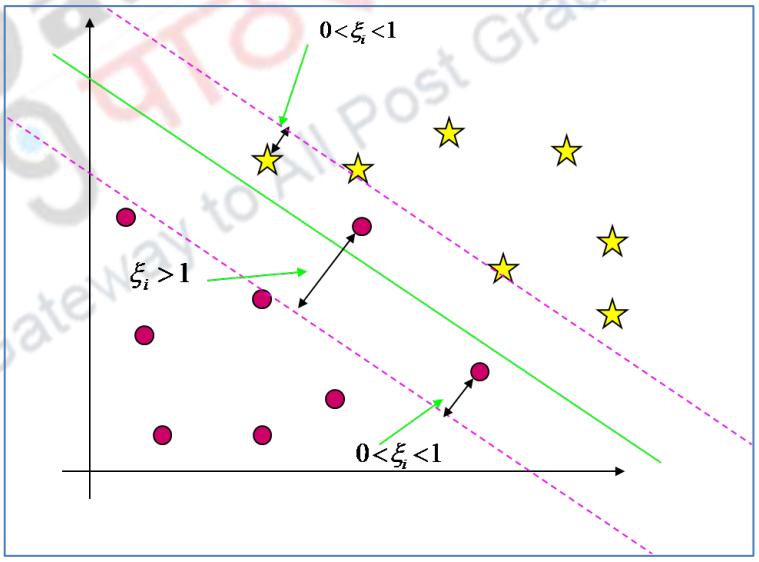

Figure 18.3 Use of Slack Variables

ξ_i Is the deviation error from ideal place for sample i: If $0< \xi_i < 1$ then sample i is on the right side of the hyperplane but within the region of the margin. If $\xi_i > 1$ then sample i is on the wrong side of the hyper plane.

The primal optimization problem now can be explained. We change the constraints to

$$y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$$

Instead of

$$y_i(w^t x_i + b) \geq 1 \quad \forall i$$

As tradeoff parameter $C \to \infty$, we get closer to the hard-margin solution.

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \forall x_i$$
$$\xi_i \geq 0$$

C>0 is a constant. It is a kind of penalty on the term $\sum_{i=1}^{n} \xi_i$. It is a tradeoff between the margin and the training error. It is a way to control overfitting along with the maximum margin approach.

18.2.3 The "C" Problem

"C" plays a major role in controlling over fitting (Figure 18.4). A larger value of C means that less number of training samples are not in ideal position. But smaller margin affects performance and a large enough value of C may lead to over fitting that is a complex classifier that fits only the training set. This means less training error that affects the classification performance positively. A smaller value of C means that more training samples are not in ideal position. A small C may lead to under fitting. This means more training error that affects

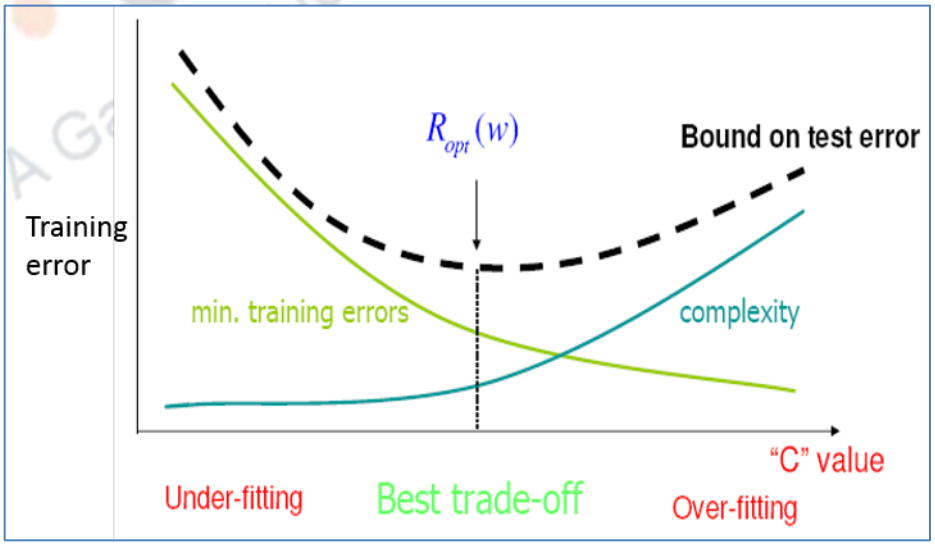

Figure 18.4 Tradeoff of C

the Classification Performance negatively. From the figure you can see that the more complex the model is (shown by blue line) the less is the minimum

training error (shown by the green line) that is we have good classification performance but only for the given training samples. Now the dotted line shows the bound on the test error, so we need to choose a value of C that gives minimum testing error.

18.2.4 Soft vs Hard Margin SVMs

The robustness of soft and hard margin SVMs is shown in Figure 18.5.

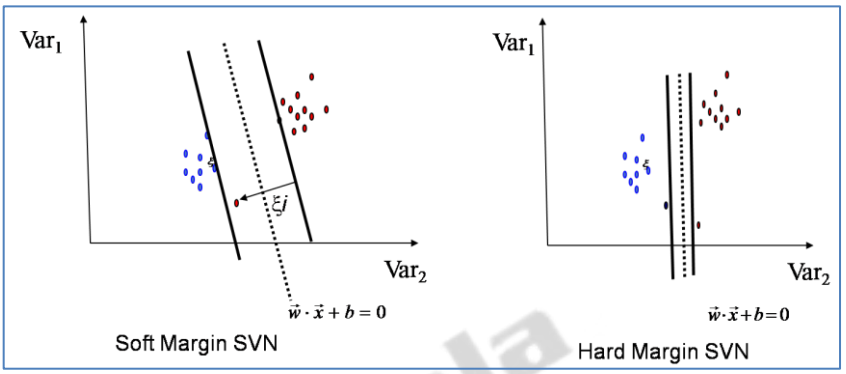

Figure 18.5 Soft versus Hard Margin SVMs

18.2.4.1 Hard Margin SVM

Considering the objective function discussed in section 18.2.2 we have an extra term in our objective function that is equal to the sum of the slack variables ξi's. Our aim is to minimize the objective function and hence we need to keep the ξi values small resulting in a hard margin and overfitting. Remember that the ξ term is multiplied by C which indicates that as C increases we care less about the size of the margin, and more about keeping the ξi's small. Thus, the value of C trades between how large of a margin we would prefer, as opposed to how many of the training set examples violate this margin (and by how much).

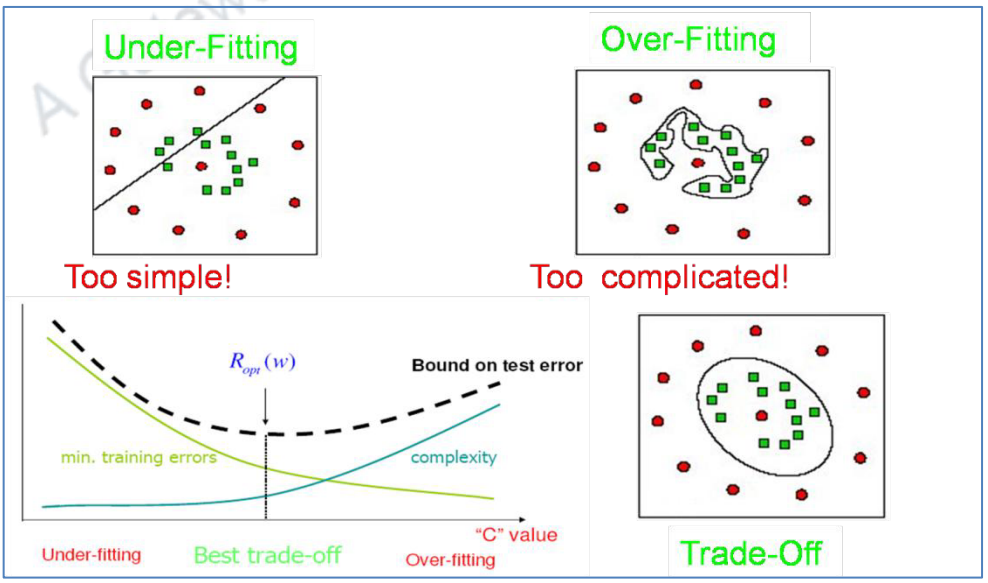

Figure 18.6 Under Fitting and Over Fitting

Soft Margin SVM

Soft margins allow errors in classification. Objective function is still valid, but constraints need to be relaxed to tackle the non-linear separable case, so the linear separator is modified so that it does not satisfy all the constraints. In other words it allows softening of the constraints and results in under fitting (Figure 18.6).

18.2 Classification with SVMs

The classifier is modelled as a separating hyper plane. The most "important" training points are support vectors and they define the hyper plane. When we need to classify a given data point x using SVMs, we can score its projection normal to the hyper plane:

$\mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$

We can decide the class based on where the example lies with respect to the hyper plane. Depending on whether the score is < or > we can decide the class. We can also set a confidence threshold t in order to control the overfitting issue . When the score > t: we say the example belongs to the class and when score < -t we say the example does not belong to the class.

18.3.1 Recipe and Model selection procedure:

In most of the real-world applications of SVM we combine what we learned about the kernel trick and the soft margin and use them together:

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{constrained to } 0 \le \alpha_i \le C \quad \forall i \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

We solve for α using a Quadratic Programming software.

$$w = \sum_{j=1}^{n} \alpha_j y_j \phi(x_j) \quad (\textit{No need to find "w" because we may not know } \phi(x))$$

To find "b" we take any $0< \alpha_i <C$ and solve

$$\alpha_i[y_i(w^t x_i + b) - 1] = 0$$

$$\Rightarrow y_i\left(\sum_{j=1}^{n} \alpha_j y_j (\phi(x_j))^t \phi(x_i) + b\right) = 1 \Rightarrow b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(x_j, x_i)$$

The Classification function will be:

$$g(x) = sign(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b)$$

18.3.2 Model selection procedure

We have to decide which Kernel function and “C” value to use.

Let us assume that the Gaussian radial basis or a low degree polynomial kernel is used. We start checking which set of parameters C or σ (the standard deviation) if we choose Gaussian radial basis are the most appropriate by using K-Fold Cross-Validation. The steps of this cross-validation are to first randomly divide all the available training examples into *K* equal-sized subsets. Then we use all but one subset to train the SVM with the chosen parameters. We then use the held out subset to measure classification error. We repeat Steps 2 and 3 for each subset. We then average the results to get an estimate of the generalization error of the SVM classifier. The SVM is tested using this procedure for various parameter settings. In the end, the model with the smallest generalization error is adopted. Then we train our SVM classifier using these parameters over the whole training set.

18.4 Multi Class SVM for more than two classes

There are two basic approaches to solve *q*-class problems (q>2) with SVMs. The first approach we will discuss is the One versus Others approach.

18.4.1 One vs. Others:

This approach works by constructing a “regular” SVM ω_i for each class *i* that separates that class from all the other classes (class “ i” positive and “not i” negative). Then we check the output of each of the q SVM classifiers for our input and choose the class i where the corresponding SVM has the maximum output.

18.4.2 Pairwise (one vs one):

Here we construct a binary SVM for each pair of classes (so we construct q(q-1)/2 SVMs) as shown in Figure 18.7. Then we use “max-wins” voting strategy: we test each SVM on the input and each time an SVM chooses a certain class we add vote to that class. Then we choose the class with highest number of votes.

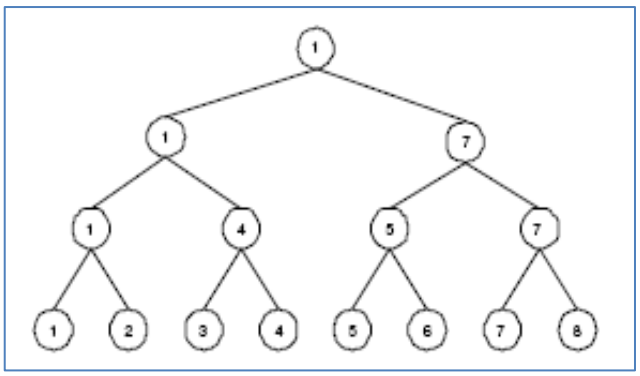

Figure 18.7 Advanced Pairwise SVM

Both the methods mentioned above give in average comparable accuracy results. However the second method is relatively slower than the first method. Sometimes for certain application one method is preferable over the other. More advanced methods to improve pairwise comparison includes using decision graphs to determine the class selected in a similar manner to knockout tournaments. There are several other strategies that can be used for selecting the final classification based on the output of the binary SVMs.

18.4.3 Basis of Multi-Label Classification

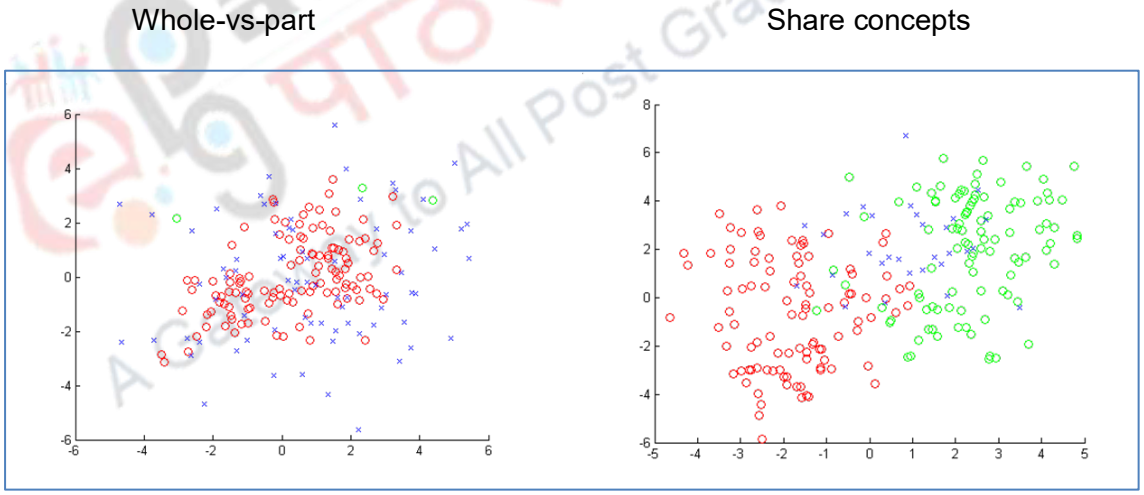

Figure 18.8 Basis of Multi-label Classification

The basis of multi-label classification could be whole-part where an element may belong to both the whole and the component. This perspective is common for parent-child relationship where we add “Other” category, and do binary classification to distinguish the child from the other category. Since the classification boundary is non-linear, kernel methods may be more effective. The basis of multi-label classification can also be shared concepts (Figure 18.8).

When the basis of the multi-label classification is shared concepts, training can be conducted in the following ways:

Mode-S – We label the multi-label data with the class to which the data most likely belonged, using some perhaps subjective criterion.

Mode-N –In this case we consider the multi-label data as a new class separate from all the other classes

Mode-X – Here we use the multi-label data more than once, using each example as a positive example of each of the classes to which it belongs. This means that a training sample can belong to more than one class

The testing for the share concepts can be carried out as follows:

P-cut - Label input testing data by all of the classes corresponding to positive SVM scores. If no scores are positive, label that data to the class with top score.

S-cut -Train a threshold for each class by cross validation, and label input testing data by all of the classes for which scores of the input data were higher than the threshold..

R-cut -For any given test instance, always assign it *r* labels according to the descending confidence scores. Here *r* can be learned from training data.

18.4.4 Evaluation Criteria

As we have already discussed previously F_1 score combines precision and recall into one measure.

$$F_1 = \frac{2pr}{p+r}$$

F_1-score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p}+\frac{1}{r}}$$

$$p = \frac{TP}{TP+FP}. \qquad r = \frac{TP}{TP+FN}.$$

Here as we know p is precision and r is the recall, and TP is true positive, FP is false positive and FN is false negative. The above measures are used for one class at a time.

Now we are going to discuss how to evaluate multi-label classification. There are many parameters that are used to evaluate multi-label classification performance. When we have more than one class we need to combine multiple performance measures into one quantity. There are basically two methods of combining namely micro-averaging and macro-averaging. In micro-averaging where we collect decisions for all classes, compute a contingency table and then evaluate. Macro-averaging is where we compute performance for each class, then average the value. With the above basis we outline the evaluation measure of combining performance measures.

Here we use n to define the number of classes. The first method is Micro-F1 which measures the overall classification performance.

$$MicroF1 = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)}$$

The second method is Macro-F1 which measures the classification accuracy on the category level. This reflects the classifier's capability of dealing with rare categories.

$$MacroF1 = (\sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}) / n$$

18.5 SVM Applications

SVM has been successfully used to solve many real-world problems. Some of the applications include text categorization, image classification, bioinformatics (protein classification, cancer classification), hand-written character recognition, etc..

18.5.1 Application 1: Cancer Classification

The SRCBT dataset can be used to illustrate the application. This is a high dimensional data set where the number of samples (patients) is > 1000 and the number of features n > 100. In such examples sometimes the samples are imbalanced with less number of positive samples. Many irrelevant features may be present where we need to perform feature selection. In the linear case, w_i^2 gives the ranking of dim i. However, SVM is sensitive to noisy (mis-labeled) data.

18.5.2 Application 2: Text Categorization

Now let us take up the classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content. There are many uses for this task such as email filtering, web searching, sorting documents by topic, etc. A document can be assigned to more than one category, viewed as a series of binary classification problems, one for each category. In this application, the representation of text is an important aspect. Normally vector space model (aka bag-of-words representation) used by information retrieval is used. Typical features are term frequency, and inverse document frequency where inverse term frequency is used to give less weightage to terms that occur in all classes and hence does not contribute to the classification.

$$IDF(w_i) = log\left(\frac{n}{DF(w_i)}\right)$$

In this case a document is represented by a vector indexed by a pre-fixed set or dictionary of terms. The values of an entry can be binary or weights

$$\varphi_i(x)=tf_i \log(idf_i) / k$$

For documents, removal of stops words and stemming is carried out as a pre-processing step. In order to account for varying length documents, normalization is carried out. Therefore can be represented Doc x => φ(*x*). The distance between two documents is φ(*x*)·φ(*z)* where *K*(*x,z*) = φ(*x*)·φ(*z*) is a valid kernel. SVM can be used with *K*(*x,z*) for discrimination.

Why should we use SVM for text categorization? It is a typical multi-class multi-label classification problem. Documents are represented in such a way that it results in high dimensional input space, has few irrelevant features (dense concept) and the document vectors are sparse. In general text categorization problems are linearly separable. SVM, with some additional heuristic, has been regarded as one of the best classification scheme for text data, based on many benchmark evaluations.

18.5.3 Application 3: Facial Expression Recognition

For this application we refer to the work discussed in the paper “Facial Expression Recognition Using Support Vector Machines by Philipp Michel and Rana El Kaliouby”. Three basic problems a facial expression analysis approach needs to deal with are face detection in a still image or image sequence where we assume a full frontal view of the face. The data from facial expression is extracted using an automatic tracker that extracts the position of facial features from the video stream or an image as the case may be. For each expression, a vector of feature displacements is calculated by taking the

Euclidean distance between feature locations in a neutral state of the face and a “peak” frame representative of the expression. We use the SVM method to construct the classifier and the vectors of feature displacements for the previous stage as our input. A set of 10 examples for each basic emotion (in still images) was used for training, followed by classification of 15 unseen examples per emotion. Libsvm was used as the underlying SVM classifier. At first they used the standard SVM classification using linear kernel and they got 78% accuracy. Then with subsequent improvements including selection of a kernel function (they chose RBF) and the right “C” customized to the training data, the recognition accuracy boosted up to 87.9%.

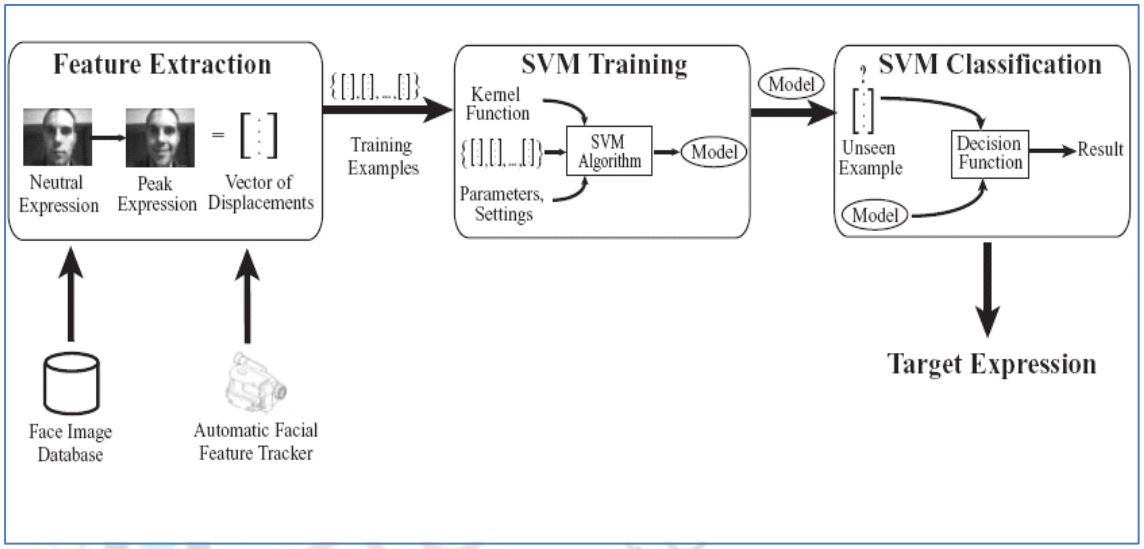

Figure 18.9 Classification of Facial Recognition

18.6 Advantages of SVM

The Kernel maps to a very-high dimensional space. In general, the search space has a unique minimum. The training that needs to be carried out is extremely efficient and similarly classification is extremely efficient. The two choices that are to be made are about the Kernel function and the cost. In typical domains, very good accuracy can be achieved. It is generally an extremely robust method.

18.7 Weakness of SVM

However in spite of many advantages SVM has some disadvantages. It is sensitive to noise where a relatively small number of mislabeled examples can dramatically decrease the performance. In general it is a binary classification mechanism where only two classes are considered. It is expensive in terms of both memory and computational time, especially for multiclass problems. The selection of the kernel function and the right value for the “Trade-off” parameter “C” is not a straightforward issue.

18.8 Software: Popular implementations

SVM is a technique that has many implementations available for use. One such implementation is SVM*light*: http://svmlight.joachims.org/ . This implementation is by Joachims . It is the most widely used SVM classification and regression package. It is distributed as a C++ source and binaries for Linux, Windows, Cygwin, and Solaris. The kernels provided are polynomial, radial basis function, and sigmoid (tanh).

The next software is LibSVM : http://www.csie.ntu.edu.tw/~cjlin/libsvm/ . LIBSVM (Library for Support Vector Machines), is developed by Chang and Lin and is also a widely used software developed in C++ and Java. It supports multi-class classification, weighted SVM for unbalanced data, cross-validation and automatic model selection. It has interfaces for Python, R, Splus, MATLAB, Perl, Ruby, and LabVIEW. Kernels provided include linear, polynomial, radial basis function, and neural (tanh).

Summary

- Explained max margin
- Discussed optimal separating hyper planes
- Explained soft margin SVMs and multi class SVMs
- Listed various applications
- Discussed cancer classification and text categorization
- Listed advantages and Weaknesses of SVM