Chapter 7

Kernel Based Extreme Learning Machine for Classification

Contents

This chapter discusses kernel based Extreme Learning Machine (ELM) for classification. Accuracy and learning time are determined when ELM with various kernels are applied on Dataset-I. These parameters are also analysed when the same kernels are used in Support Vector Machine. The comparative study of these measures with Conventional Single Layer Feed Forward Network (SLFN) is also studied.

Section 7.1 introduces important features of ELM. Algorithm of ELM and comparative study of its features with SVM and SLFN is discussed in section 7.2. Section 7.3 deals with the experiments and result analysis and summary is given in Section 7.4.

7.1 Introduction

Extreme Learning Machine is very popular technique over the last decade due to its ease of implementation, fast learning, unification of classification and regression. ELM is single hidden layer feed forward network where weights need not be tuned. Artificial Neural Network (ANN) based classifiers can integrate both structural and statistical information and achieve better performance than that of minimum distance classifiers [144]. However, conventional feed forward neural networks use gradient descent method to train network, which might get the algorithm stuck at local optimum. Also, all parameters of the network need to be tuned iteratively, so learning speed is very slow [55]. ELM is a single layer feed forward neural network (SLFN), which randomly chooses input weights and analytically determines the output weights. When input weights are taken arbitrary and hidden neurons are specified, ELM is considered as linear system and output weights are calculated analytically. This makes learning speed of the network extremely fast. According to Bartlett, ELM tend to have good generalisation performance not only by minimizing training error but also giving smaller norm of weights [8]. The learning algorithm of ELM not only reduces training time but also minimizes norm of weights. Due to these properties, ELM achieves good generalisation with extremely high speed. The universal approximation condition (2.4.1) is necessary and sufficient condition for feature mapping [58]. Even though hidden nodes are given arbitrarily, ELM maintains the universal approximation capability of SLFNs([56], [57], [60]). Huang has shown that maximal margin property of SVM is consistent with minimum norm of output weight of ELM [59]. But, generalisation ability of ELM is similar or better than that of SVM [60]. ELM is suitable for many nonlinear activation functions and kernel functions.

Due to its remarkable efficiency and impressive generalisation performance it has been applied to many fields of classification and regression. Its applications can be found in various domains such as biomedical engineering, robotics, system identification and control. It overcomes the problem of slow training speed and local optimum of conventional neural network learning algorithm. For multi classification problems it achieves good classification accuracy with remarkable learning speed compared to the Support Vector Machine (SVM). Huang *et.al.* have proposed kernel ELM [59]. They have tested Gaussian kernel and Polynomial kernel with ELM on various datasets. But, these kernels are not suitable for all types of applications. Liu *et. el.* have developed a learning algorithm which automatically learn data dependent optimal kernel according to application [80]. Since input weights and biases are randomly chosen, hidden layer output matrix may not be full column rank (non singular matrix), which sometimes makes the linear system unsolvable and lowers the prediction accuracy. To overcome this problem Wang *et.al.* have proposed a new algorithm called effective extreme learning algorithm (EELM) [131]. This algorithm trains the input weights and biases, so that makes the hidden layer output matrix full rank. Deep leaning (DL) is a multilayer network which can extract the significant features learning from lower layer to higher layer ([49], [50]). It is good at extracting features, which uses gradient descent method, that takes too much time in adjusting parameters during the training. So, in DL training speed is very slow. Ding *et. al.* have proposed convolutional extreme learning machine with kernels (CKELM) [33]. The hidden layer of CKELM is not single layer but consists of convolution layers and subsampling layers. It is based on Deep Learning but do not use gradient descent algorithm to adjust parameters. It uses random weights during the training. Thus, CKELM uses features of convolution neural network (CNN) with ELM. So, in CKELM features are extracted with less training time.

In this study, ELM is used with Polynomial kernel functions, Radial Basis Function and Exponential chi-square kernel function to diagnose the skin diseases described in Dataset-I. Using ELM good classification accuracy is obtained with less learning time compare to SVM [100].

7.2 Algorithm of ELM and its Comparative Study

7.2.1 Algorithm of ELM

ELM is a single hidden layer feed forward network in which input weights w_{ji} connecting i^{th} input node to the j^{th} hidden node where $i = 1, 2, ..., n$, where, n be the number of input nodes, $j = 1, 2, ...h$, h is the number of hidden nodes and bias b are assigned randomly. Hidden nodes are crucial but need not be tuned in ELM. They are randomly initiated and remains unchanged ([55], [60]).
Let there be m arbitrary distinct samples $(\mathbf{x}_i, \mathbf{y}_i), \quad i = 1, 2, ..., m, \quad$ where,
$\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^N$, where n is the number of features and N is the number of classes.
Step 1: Randomly assign input weight $w_{ji}, i = 1, 2, ..., n, \ j = 1, 2, ...h$.
Step 2: Hidden layer input nodes h are randomly chosen and output of the hidden layer is initialized using some non linear piecewise continuous function g(x), such as sigmoid function (3.2.2), or we may use some non linear kernel function.
The hidden layer output matrix is

$$H(x) = \begin{bmatrix} g\left(w_{11}^T x_1 + b_1\right) & g\left(w_{21}^T x_1 + b_2\right) & ... & g\left(w_{h1}^T x_1 + b_h\right) \\ g\left(w_{12}^T x_2 + b_1\right) & g\left(w_{22}^T x_2 + b_2\right) & ... & g\left(w_{h2}^T x_2 + b_h\right) \\ \vdots & \vdots & \cdots & \vdots \\ g\left(w_{1n}^T x_n + b_1\right) & g\left(w_{2n}^T x_n + b_2\right) & ... & g\left(w_{hn}^T x_n + b_h\right) \end{bmatrix} \tag{7.2.1}$$

Step 3: Weights $\boldsymbol{\beta_i} = [\beta_{i1}, \beta_{i2}, ..., \beta_{iN}]^T$, connect i^{th} hidden node $i = 1, 2, ..., h$ to N output layer nodes. These weights are obtained by minimizing the error $\|H\boldsymbol{\beta} - \mathbf{y}\|$, with minimum output weights $\|\boldsymbol{\beta}_i\|$.
The objective is to find the least square solution of the system

$$\|H\boldsymbol{\beta} - \mathbf{y}\| = 0. \tag{7.2.2}$$

The output function of ELM is $H\boldsymbol{\beta}$. Where, H maps the data from the n-dimensional input space to h-dimensional hidden layer feature space (ELM feature space), thus $H(x)$ is a feature mapping. For binary classification problem the decision function

of ELM is given by,

$$f(x) = \text{sign}(H\boldsymbol{\beta}) \tag{7.2.3}$$

Therefore, we can say that to minimize the norm of the output weights $\|\boldsymbol{\beta}\|$, is equivalent to maximize the distance of the separating margins of the two different classes in the ELM feature space: $\frac{2}{\|\beta\|}$.
If the number of hidden nodes h is equal to the number of input nodes n, then the system (refer 7.2.2) is square, invertible. The system can be approximated using zero training error. But, in most of the cases the number of hidden nodes is much less than the number of training samples, so H is not a square matrix. In such case, least square solution of the system, with smallest norm is given by Moore-Penrose generalized inverse $H^{\dagger}$ (refer definition 2.1.8) of the rectangular matrix H. The optimum solution i.e. the output layer weight is given by [60]

$$\boldsymbol{\beta} = H^{\dagger}\mathbf{y} \tag{7.2.4}$$

Moore-Penrose generalized inverse can be calculated by different methods like, Orthogonal projection method, Singular Value Decomposition(SVD) etc.[59]. Orthogonal Projection method can be used in two cases: $H^T H$ is nonsingular and $H^{\dagger} = (H^T H)^{-1} H^T$, or when HH^T is nonsingular and $H^{\dagger} = H^T (HH^T)^{-1}$. If some positive value can be added to the diagonal of $H^T H$ or HH^T, then the resulting solution is more stable and more generalised [59].

For multi classification problem the objective function of ELM can be formulated as [59]:

$$\text{Minimize} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\|\xi_i\|^2 \tag{7.2.5}$$

Subject to the constrain,

$$H\boldsymbol{\beta} = y_i^T - \xi_i^T, \qquad i = 1, 2, ..., n.$$

where $\mathbf{y}_i = [y_{i1}, y_{i2}, ..., y_{iN}]$ and $\boldsymbol{\xi}_i = [\xi_{i1}, \xi_{i2}, ..., \xi_{iN}]$ is the training error vector of N output nodes with respect to the training sample $\mathbf{x}_i, \quad i = 1, 2, ..., n$.

The Lagrangian is given by

$$L(\boldsymbol{\beta}, \alpha, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \frac{C}{2}\sum_{i=1}^{n}\|\xi_i^2\| - \sum_{i=1}^{n}\sum_{j=1}^{N}\alpha_{ij}\left(h(x_i)\beta_i - y_{ij} + \xi_{ij}\right) \tag{7.2.6}$$

Applying KKT optimality conditions (refer condition2.2.1),

$$\frac{\partial L}{\partial \beta_j} = 0 \Rightarrow \beta_j = \sum_{i=1}^{n}\alpha_{ij}h(x_i)^T$$

$$\Rightarrow \boldsymbol{\beta} = \boldsymbol{(H)^T\alpha} \tag{7.2.7}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i = C\xi_i, \quad i = 1, 2, ..., n \tag{7.2.8}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \Rightarrow h(\xi_i)\boldsymbol{\beta} - y_i^T + \xi_i^T = 0, \quad i = 1, 2, ..., n. \tag{7.2.9}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, ..., \boldsymbol{\alpha_n}]$.
Using (equation 7.2.7) and (equation 7.2.8) into (equation 7.2.9), we have

$$\left(\frac{I}{C} + HH^T\right)\alpha = \mathbf{y} \tag{7.2.10}$$

where, I is the identity matrix of dimension h, $\mathbf{y} \in R^N$ is the target vector and H is the output matrix of hidden layer, HH^T is called kernel matrix in which $H(x_i).H(x_j)$ is ELM kernel.
Using (equation 7.2.7) and (equation 7.2.10), the output weight vector β is:

$$\boldsymbol{\beta} = H^T\left(\frac{I}{C} + HH^T\right)^{-1}\mathbf{y}. \tag{7.2.11}$$

The system can be solved by Gauss elimination Method, Orthogonal projection method, Iterative method etc.

7.2.2 Comparative Study

In this section we have discussed differences and similarities of various features of traditional SLFN, ELM and SVM.

Table 7.1: Comparative Study of Features of Conventional SLFN, ELM and SVM

	ELM	Conventional SLFN	SVM
Computing Time	Very Low	High	High
Multi classes	Single classifier for m classes	Single classifier for m classes	m or m(m-1)/2 classifiers for for m classes
Affected by Sample Complexity	No	No	Yes
Hidden Layer Nodes	Any Number of Nodes and then fix during entire process	Need to be tuned	Not Require
Generalize performance	Very Good	Poor	Very Good
Learning methodology	Extremely Easy	Easy compare to SVM	Complex
Input weights	Random	Random	Random
Output weights	Determined Analytically	Need to be tuned	Need to be tuned
Parameter adjustments required for	one parameter	many parameters	many parameters

Effect of user Specified Parameter	Least human Intervention	Very sensitive	Sensitive for kernel choice kernel parameters
Universal App. Cond.	Satisfied	Satisfied	Not ensured depends on kernel

7.3 Experimentations and Result Analysis

In this study learning algorithm ELM is applied to the Dataset-I (Appendix-A) to diagnose common skin diseases. For comparative study, conventional SLFN and SVM along with ELM is used. Table (7.2) summarizes learning time and accuracy, when above three learning algorithms are applied on the Dataset-I. The dataset is divided into 70% -30% data partitioned for training and testing purpose respectively. All simulations for SLFN, ELM and SVM are carried out in MATLAB R2015b running in i5-44605 CPU @ 2.90GHz. Results are finalized after 100 trials.

For SLFN, we use Neural Network toolbox of MATLAB. The network is created using newff() matlab inbuilt function defined in Appendix-A. We use sigmoid function(definition 3.2.2) as activation function. Training is carried out using Levenberg-Marquardt algorithm (algorithm 3.2.2). Results are taken for 10, 20 and 47 hidden nodes in hidden layer.

Softwares for ELM and SVM are available only for most popular, standard kernel functions viz., Linear kernel, Polynomial kernel and Radial Basis function. We have incorporated chi-square kernel also in both softwares.

In ELM, classification accuracy is obtained using Polynomial kernel, RBF kernel and Exponential chi-square kernel taking 5 hidden nodes. Parameters are set using grid search algorithm (2.2.2).

For SVM same kernels are used as ELM. In SVM classifications are done using LIBSVM 3.20 with MATLAB interface [19]. For validation 10 fold cross validation is used(2.4.8).

Table 7.2: Performance Result of Conventional SLFN, SVM and ELM

			Learning Time (Seconds)	Classification Accuracy
Conventional SLFN	10 hidden nodes		43.89	91.24 %
	20 hidden nodes		79.52	90.60 %
	47 hidden nodes		14.42	93.70%
ELM	Polynomial Kernel $(\alpha x^T y + a_0)^d$	$\alpha = 1$ $a_0 = 0.8$ $d = 3$	0.0072	88.65%
	RBF Kernel $exp\left(-\gamma \|x - y\|^2\right)$	$\gamma = 0.8$	0.0061	92.20%
	Exponential Chi-Square $exp\left(-\gamma \sum \left(\frac{\|x-y\|^2}{x+y}\right)\right)$	$\gamma = 0.98$	0.0324	92.91%
SVM	Polynomial Kernel $(\alpha x^T y + a_0)^d$	$\alpha = 2$ $a_0 = 10$ $d = 3$	0.0241	90.78%
	RBF Kernel $exp\left(-\gamma \|x - y\|^2\right)$	$\gamma = 0.1$	0.0187	90.78%
	Exponential Chi-Square $exp\left(-\gamma \sum \left(\frac{\|x-y\|^2}{x+y}\right)\right)$	$\gamma = 0.1$	0.1614	90.78%

The simulation results exhibits that the accuracy obtained using conventional SLFN is highest among the three learning algorithms under study. But, learning time is very high compared to ELM and SVM.

Comparative study shows that the highest accuracy obtained by ELM is 92.91% using exponential chi-square kernel, which is better than that of the highest accuracy obtained by SVM. Also, to achieve this accuracy, ELM is taking about 398% less learning time than that of SVM for the dataset under study. So, we observed that ELM has better scalability compared to SLFN and SVM.

7.4 Summary

This chapter discusses Kernel based Extreme Learning Machine (ELM) for classification task of skin Dataset-I (Appendix-A). Weights need not be trained in ELM, therefore learning time of ELM is very less. Also, weights obtained after learning the ELM is unique and hence good generalisation i s achieved. Comparative study of kernel based ELM with SLFN and SVM is made. We observed that the learning time of ELM is extremely less compared to other two classifiers with good classification accuracy.