

Automatic Solar Flare Detection Using MLP, RBF and SVM

Ming Qu¹, Frank Y. Shih¹, Ju Jing², Haimin Wang²

1. College of Computing Sciences

2. Department of Physics

New Jersey Institute of Technology

Newark, NJ 07102

Tel: (973) 596-5654

Fax: (973) 596-5777

(Email: shih@njit.edu)

06 May 2003

Abstract. The focus of the automatic solar flare detection is on the development of efficient feature-based classifiers. The three principal techniques used in this work are Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), and Support Vector Machine (SVM) classifiers. We have experimented and compared these three methods for solar flare detection on the solar H α (Hydrogen-Alpha) images obtained from the Big Bear Solar Observatory in California. The preprocessing step is to obtain the nine principal features of the solar flares for the classifiers. Experimental results show that by using SVM, we can obtain the best classification rate of the solar flares. We believe our work will lead to real-time solar flare detection using advanced pattern recognition techniques.

1. Introduction

Solar flare is intense, abrupt release of energy which occurs in areas on the sun where the magnetic field is changing due to flux emergence or sunspot motion. As a result, large amounts of high-energy electrons are then accelerated. These electrons generate intense X-ray and radio bursts (Zirin, 1988). Typically, the brightness increases for several minutes, followed by a slow decay which lasts between 30 minutes to 1 hour. Solar flares are very important events and they release energies in the range from 10^{20} J to 10^{26} J. The associated energy flux, on the average, is 10^4 times of the flux of normal solar radiator (Stix, 1989). Figure 1 shows a Full disk H α (Hydrogen-Alpha) image with a flare detected on the low-left corner.

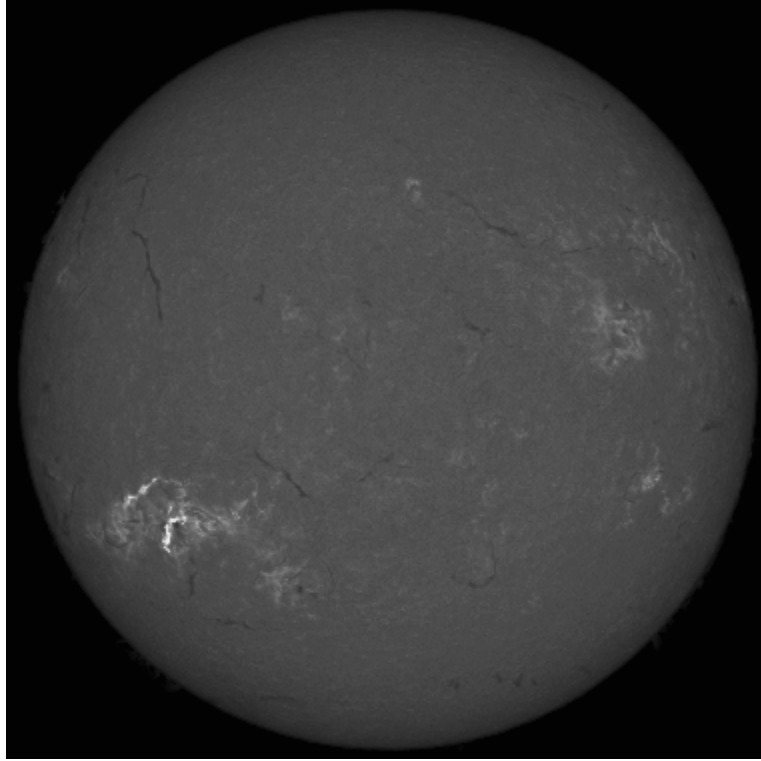


Figure 1. Solar flares on the left side of the image

There are many ways to detect solar flares. When a solar flare first appears on the Sun's surface, there is often a radiation covering the entire range of electromagnetic waves. For example, light curves of radio waves and X-rays would indicate on a set of flares. However, location and properties of flares cannot be demonstrated in light curves. In the mean time, solar flares also emit high velocity charged particles that take one or two days to reach Earth. Our objective is to detect and characterize solar flares automatically in real time. Finally, most solar flares can also be detected using $H\alpha$. We start with solar images observed in $H\alpha$ using an 8-inch telescope in the Big Bear Solar Observatory (BBSO) in California, operated by New Jersey Institute of Technology.

Automatic solar flare detection is the key to space weather monitoring. It is very challenging since the features of the solar flares are complicated. We need a system that can handle the solar flares' complexity and scalability.

Veronig et al. (2000) proposed a method for automatic flare detection by applying a combination of region-based and edge-based segmentation methods. Simple region-based methods are applied for a tentative assignment of flare activity, making use of one of the decisive flare characteristics: the high intensities. Borda et al. (2001) presented a method for automatic detection of solar flares using the neural network technique. The network used is multi-layer perceptron (MLP) with back-propagation training rule. They used a supervised learning technique and required a lot of iterations to train the network.

In this paper, we develop a set of nine features for solar images and use Radial Basis Function (RBF) and Support Vector Machine (SVM) in addition to MLP to perform classification. The paper is organized as follows: section 2 presents the overall system algorithms including Neural Networks, RBF and SVM; section 3 describes

preprocessing and feature analysis; section 4 gives the experimental results; finally section 5 presents the conclusions.

2. System Algorithms

2.1. NEURAL NETWORKS

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for classification (Haykin, 1994). Multi-layer perceptron (MLP) is a type of feed-forward networks. The learning rule is back-propagation (Ripley, 1996). The feed-forward network has three layers: input layer, hidden layer, and output layer. The hidden layer is between the input and the output layers. For example, a simple feed-forward network is shown in Figure 2, where there are two nodes in the input layer, two nodes in the hidden layer, and one node in the output layer. Weights are incorporated into the connections from input nodes to hidden nodes and from hidden nodes to the output node.

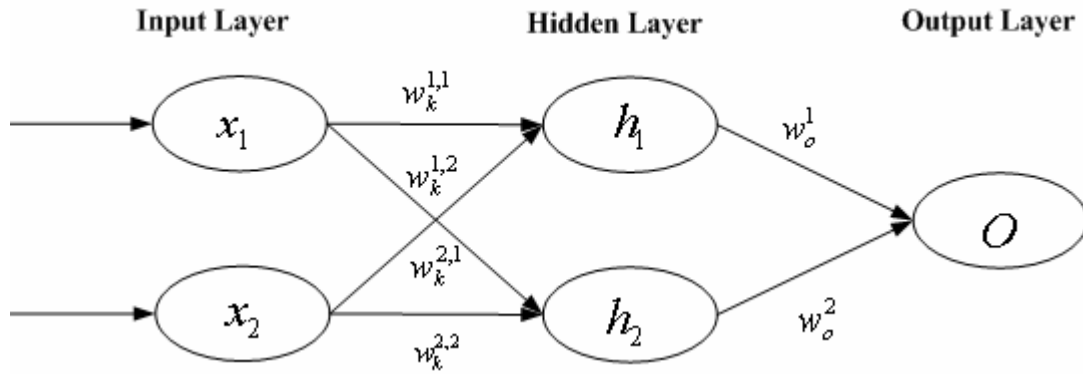


Figure 2. The architecture of MLP

For linear processes, we can use binary thresholding in the hidden and output units. For nonlinear processes, activation functions are used rather than thresholding. In this paper, we adopt the sigmoid function as the activation function

$$h_j = \text{sigmoid}\left(\sum_{i=1}^{N_I} w_k^{i,j} x_i\right) \quad (1)$$

$$O = \text{sigmoid}\left(\sum_{j=1}^{N_H} w_o^j h_j\right) \quad (2)$$

where x_i is the input feature at the input node, $w_h^{i,j}$ is the weight connecting the input node with the hidden node, w_o^j is the weight connecting the j th hidden node with the output node, and O is the output value. The *sigmoid* function is defined as

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

The back-propagation training is conducted to obtain the correct weights. We can compute the mean square error (MSE) between the actual output and the desired output for the given input in the training set. The error function E can be obtained by

$$E = (T - O)^2 \quad (4)$$

where T is the value of the desired target. To reduce the mean square error, it is necessary to calculate the gradient of the error function with respect to each weight. One may then move each weight slightly in the opposite direction to the gradient. The gradient function for the weights in the output layer is shown below.

$$\delta_j = \frac{\partial E}{\partial W_0^j} = \frac{\partial (T - O)^2}{\partial W_0^j} \quad (5)$$

$$W_0^{j+1} = W_0^j - \eta \delta_j \quad (6)$$

From Equation (6), the new values for the network weights are calculated by multiplying the negative gradient with a step size of parameter η (called the learning rate). The weights in the hidden layer are updated using the same procedure. After we calculate all the correct weights, the neural network is completely constructed.

2.2. RADIAL BASIS FUNCTION (RBF)

We can view the design of a neural network as a curve-fitting (approximation) problem in a high-dimensional space. From this viewpoint, learning is accomplished by finding a surface in a multi-dimensional space. This surface is used to interpolate the test data. Radial basis function (RBF) network is a fully connected network and generally is used as a classification tool (Broomhead and Lowe, 1988). In a RBF model, the layer from input nodes to hidden neurons is unsupervised and the layer from hidden neurons to output nodes is supervised (Bishop, 1995). The transformation from the input to the hidden space is nonlinear, and the transformation from the hidden to the output space is linear. The hidden neurons provide a set of “functions” that constitute an arbitrary “basis” for the input patterns. These are the functions known as called radial basis functions. Through careful design, it is possible to reduce a pattern in a high-dimensional space at input units to a low-dimensional space at hidden units.

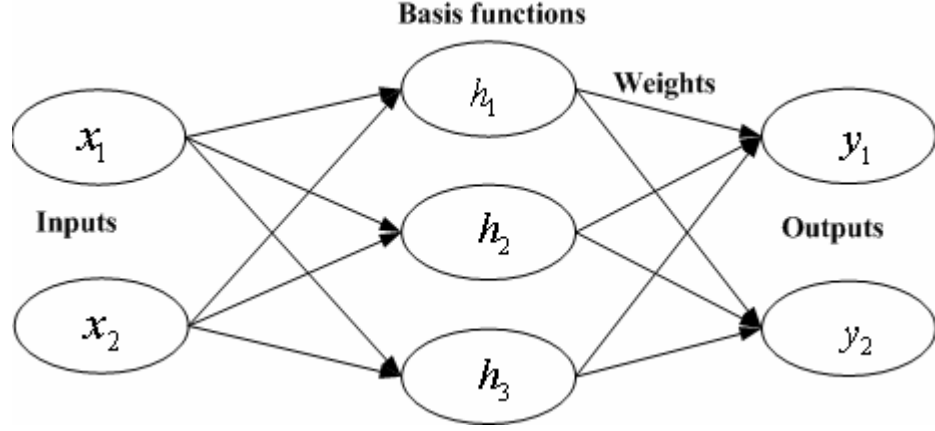


Figure 3. The architecture of a radial basis function neural network. The input vector has d nodes and the outputs vector has c nodes. It is a mapping from $R^d \rightarrow R^c$.

In Figure 3, the network forms the following equation:

$$y_k(x) = \sum \omega_{kj} h_j(x) + \omega_{k0} \quad (7)$$

where $h_j(x)$ is a Gaussian function typically, and ω_{k0} is the bias or threshold. The Gaussian basis function is used as an activation function. That is

$$h_j(x) = \exp\left(-\|x - \mu_j\|^2 / 2\sigma_j^2\right) \quad (8)$$

where x is the d -dimensional input vector with element x_i , and μ_j and σ_j are respectively the center and the standard deviation of the Gaussian basis function. Since the first and second layers of RBF network are unsupervised and supervised respectively, a two-stage training procedure is used for training the RBF model. In the first stage, the input data set is used to obtain the parameters of the activation functions (like μ_j and σ_j). In the second stage, the optimal weights between hidden neurons and output nodes are obtained by minimizing a sum-of-square error function. The following procedures describe the steps of obtaining the optimal weights.

By absorbing the bias parameter, ω_{k0} , into the weights, Equation (7) can be revised as

$$y_k(x) = \sum \omega_{kj} h_j(x) \quad (9)$$

where $h_j(x)$ is an extra basis function with the activation value 1. Equation (9) can be rewritten using the matrix notation as

$$Y(x) = W\Phi \quad (10)$$

where $W = [\omega_{kj}]$ and $\Phi = [h_j(x)]$.

The sum-of-square error function, E , can be described as

$$E = \frac{1}{2} \sum \sum [y_k(x^n) - t_k^n]^2 \quad (11)$$

where x^n is the input data set, and t_k^n is the target value for the output unit k .

By differentiating E with respect to ω_{kj} and setting derivative to zero, the optimal weights can be obtained. The solutions of weights can be expressed using the matrix notation as

$$(\Phi^T \Phi) W^T = \Phi^T T \quad (12)$$

where $T = [t_k^n]$ and $\Phi = [h_j(x^n)]$.

By multiplying $(\Phi^T \Phi)^{-1}$ to Equation (12), the solution for the weights is given as

$$W^T = (\Phi^T \Phi)^{-1} \Phi^T T \quad (13)$$

$$W^T = \Phi^+ T \quad (14)$$

where $\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T$ being the pseudo-inverse of Φ . After computing the optimal weights, the RBF network can be used as a classifier to segment the test data into the corresponding classes, with -1 indicating a non-flare state and 1 indicating a flare state.

RBF is strongly dependent on the quality of the employed learning strategy and the quantity of training images. The aim of an adaptive learning RBF network is to reduce the required knowledge of the system parameters with a minimum amount of performance loss. The RBF network requires knowledge of three different parameters per neuron:

- The center vector μ_j
- The weights ω_{kj}
- The radius σ_j

One unsupervised learning strategy is the self-organizing feature map. When the algorithm has converged, prototype vectors which corresponding to nearby points on the feature map grid have nearby locations in input space. However, the imposition of the topographic property, particularly if the data is not intrinsically two-dimensional, may lead to a sub-optimal placement of vectors. Here we use the K-Means unsupervised learning strategy as follows:

let μ_j be the mean of the data points in set S_j given by

$$\mu_j = \frac{1}{N_j} \sum_{n \in S_j} x^n \quad (15)$$

The initial centers are randomly chosen from the data points, and the nearest μ_j is updated using

$$\Delta \mu_j = \eta(x^n - \mu_j) \quad (16)$$

where η is the learning rate parameter.

The second parameter of the RBF network is the weight ω_k of the output layer. It can be done by using the LMS algorithm. The LMS algorithm was originally developed by Widrow and Hoff in 1960 and is also known as the Widrow-Hoff rule (Mittra and Poor, 1994). The weights can be summarized as follows

$$\Delta W = 2\eta(y_k(x^n) - t_k^n)h(x^n) \quad (17)$$

where η is the learning rate of LMS, and $y_k(x^n)$ and t_k^n are the responses of the RBF network and the desired response. The vector $h(x^n)$ contains the unweighted responses of all neurons in the hidden layer.

The last parameter of the RBF network is the radius or spread of the radial function. We may use an average of the center spread of all RBFs to calculate the radius. After the centers μ_j are established, σ^2 can be derived from the center as

$$\sigma^2 = \frac{1}{M} \sum_{j=1}^M \|y_k(x^n) - \mu_j\|^2 \quad (18)$$

where M is the number of hidden nodes.

2.3. SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) is a generation learning system based on advances in statistical learning theory. It has been successfully applied in text categorization, hand-written character recognition, image classification, biosequences analysis, etc.

Suppose we are given l observations. Each observation consists of a pair: a vector $x_i \in R^n, i = 1, \dots, l$ and the associated “truth” y_i , given to us by a trusted source. It is assumed that there exists density distribution $p(x, y)$ from which these data are drawn, i.e., the data are assumed independently drawn and identically distributed. Now suppose we have a machine to learn the mapping $x_i \rightarrow y_i$. The machine is actually defined by a set of possible mappings $x_i \rightarrow f(x, \alpha)$, where the function $f(x, \alpha)$ is labeled by the adjustable parameter α . For example, a neural network with a fixed architecture and with α corresponding to the

weights and biases, is a learning machine in this sense. $f(x, \alpha)$ function could represent a set of Radial Basis functions or MLPs with a certain number of hidden neurons (Vapnik, 1982). The expectation of the test error for a trained machine is therefore

$$R(\alpha) = \int \frac{1}{2} |y - f(x, \alpha)| p(x, y) dx dy \quad (19)$$

where $R(\alpha)$ is called the expected risk.

The empirical risk $R_{emp}(\alpha)$ is defined to be just the measured mean error rate on the training set (for a fixed, finite number of observations).

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^l |y_i - f(x_i, \alpha)| \quad (20)$$

The empirical risk minimization (ERM) inductive principle is to minimize the risk functional (20) on the basis of empirical data (x_i, y_i) . The ERM principle is based on the law of large numbers converges in probability to the expected risk (strategy followed by MLP and RBF neural networks). By choosing some η such that $0 \leq \eta \leq 1$, the following bound holds (Vapnik, 1998).

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l} \right)} \quad (21)$$

where h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension. The right hand side of Equation (21) is the “bound on the risk”, and the second term on the right hand side is called the “Confidence interval”. When l/h is large, the Confidence interval is small. The actual risk is then close to the value of the empirical risk one. In this case, a small value of the empirical risk guarantees a small value of the expected risk. When l/h is small, a small $R_{emp}(\alpha)$ does not guarantee a small value of the actual risk. In this case, to minimize the actual risk has to minimize the confidence interval. To minimize the right-hand side of the bound risk in Equation (21), one has to make the VC dimension a controlling variable.

In the previous section, we consider classical neural network, which implement the first strategy: Keep the confidence interval fixed and minimize the empirical risk. Below we consider a new type of universal learning machine, the Support Vector Machine, implements the second strategy: Keep the value of the empirical risk fixed and minimize the confidence interval (Vapnik, 1995). SVM is based on the structural risk minimization (SRM) inductive principle. The SRM principle is intended to minimize the risk functional with respect to the both terms, the empirical risks, and the confidence interval (Vapnik and Chervonenkis, 1974). Therefore, SVM is a better strategy than classical neural network such as MLP neural network. The basic idea of a support vector machine is to separate the fixed given input pattern vectors into two classes using a hyperplane in the high dimensional space (Vapnik and Chervonenkis, 1991). This means we always increment the VC-dimension h by one if we fail to separate the input pattern vectors. We do not stop the process until the separation is satisfied.

SVM intends to separate the input pattern using the minimum VC-dimension h , which minimizes the total risk by considering both empirical risk and confidence interval. Figure 4 shows the SVM with a linear hyperplane.

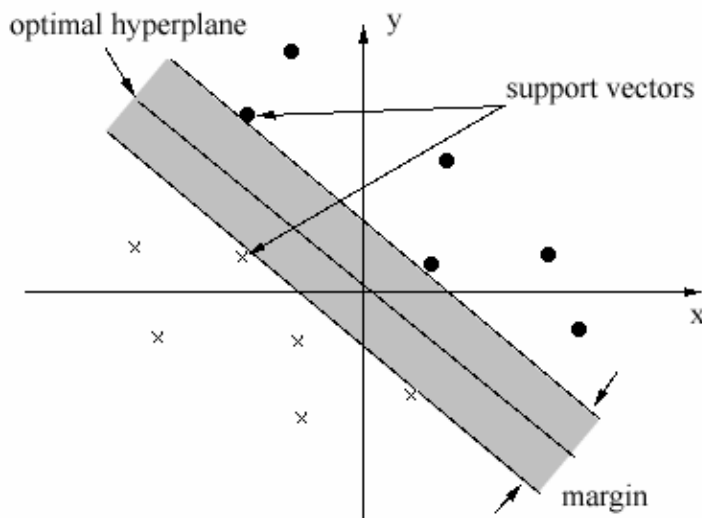


Figure 4. Support vector machine classification with a linear hyperplane that maximizes the separating margin between the two classes.

Consider a two-class classification problem where patterns are represented as an N -dimensional vector x . For each corresponding vector, there exists a value $y \in \{-1, +1\}$.

$$x = \{x_1, x_2, x_3, \dots, x_m\} \in R^N \quad (22)$$

$$y = \{y_1, y_2, y_3, \dots, y_m\} \in \{-1, +1\} \quad (23)$$

The problem is to find a decision function $g(x)$ such that the class label y of any example x can be accurately predicted. Using mathematical terms, we have $g: R \rightarrow \{-1, +1\}$ (Guyon and Stork, 2000).

For linear support vector classifier

$$f(x) = (x \cdot w) + b \quad (24)$$

where $w \in R^N$ and $b \in R$.

The set of labeled training patterns

$$(y_1, x_1), \dots, (y_l, x_l), \quad y_i \in \{-1, +1\} \quad (25)$$

It is linearly separable if there exist a vector w and a scalar b such that the following inequalities are valid for all elements of the training set.

$$w \cdot x_i + b \geq 1 \quad \text{if} \quad y_i = 1, \quad (26)$$

$$w \cdot x_i + b \leq -1 \quad \text{if} \quad y_i = -1. \quad (27)$$

The optimal hyperplane is the unique one which separates the training data with a maximal margin. The distance $\rho(w, b)$ is given by

$$\rho(w, b) = \min_{\{x:y=1\}} \frac{x \cdot w}{\|w\|} - \max_{\{x:y=-1\}} \frac{x \cdot w}{\|w\|} \quad (28)$$

Because the optimal hyperplane (w_0, b_0) is the arguments that maximize the distance $\rho(w, b)$, we have Equation (29) from (26), (27) and (28).

$$\rho(w, b) = \frac{2}{\|w_0\|} = \frac{2}{\sqrt{w_0 \cdot w_0}} \quad (29)$$

It means that the optimal hyperplane is the unique factor that minimizes $w \cdot w$ under the constraints (26) and (27). This problem is usually solved by means of the classical method of Lagrange multipliers.

If we denote the N nonnegative Lagrange multipliers as $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$, the solution is equivalent to determining the saddle point of the function.

$$L(w, b, \alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^N \alpha_i \{y_i (w \cdot x_i + b) - 1\} \quad (30)$$

At the saddle point, L has a minimum for $w = w_0$ and $b = b_0$ and a maximum for $\alpha = \alpha^0$. We have

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial b} = \sum_{i=1}^N y_i \alpha_i^0 = 0 \quad (31)$$

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial w} = w_0 - \sum_{i=1}^N \alpha_i^0 y_i x_i = 0 \quad (32)$$

Substituting Equations (31) and (32) into the right hand side of (30), it is reduced to the maximization of the function

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \quad (33)$$

From (32) it follows that

$$w_0 = \sum_{i=1}^N \alpha_i^0 y_i x_i \quad (34)$$

The parameter b_0 can be obtained from the Kuhn-Tucker condition.

$$b_0 = y_j - w_0 \cdot x_j \quad (35)$$

The problem of classifying a new data point x is now simply solved by looking at the sign of

$$w_0 \cdot x + b_0 \quad (36)$$

For the nonlinear case, we use the kernel functions to transfer input data to feature space. The nonlinear kernel functions are given following:

Polynomial support vector classifier

$$k(x, x') = \langle x, x' \rangle^d \quad (37)$$

Gaussian RBF kernel classifier

$$k(x, x') = \exp\left(-\|x - x'\|^2 / 2\sigma_j^2\right) \quad (38)$$

3. Feature Analysis and Preprocessing

Borda et al. (2001) used seven features for solar flare detection. After extensive investigations, we modify them into nine features as described below.

Feature 1: Mean brightness of the frame. Usually, when flares happen, the image becomes brighter. Let x_j denote the gray level of a pixel and N denote the number of pixels. The mean brightness \bar{x} is represented as

$$\bar{x} = \sum_{j=0}^{N-1} x_j \quad (39)$$

Feature 2: Standard deviation of brightness. Usually, when flares happen, the standard deviation of the brightness becomes large. In Equation (40), std denotes the standard deviation, and V denotes the variance.

$$std = \sqrt{V} \quad (40)$$

$$V = \sum_{j=0}^{N-1} (x_j - \bar{x})^2 \quad (41)$$

Feature 3: Variation of mean brightness between consecutive images. This is the difference of mean values between the current image and the previous image.

Feature 4: Absolute brightness of a key pixel. By comparing consecutive images, we can find the key pixel that has the maximum gray level difference. Usually, when flares happen, the absolute brightness of the key pixel becomes large. The position of the key pixel k is represented as

$$k = where(\max(\text{Im}_{pre} - \text{Im}_{current})) \quad (42)$$

where Im_{pre} denotes the vector of the previous image, and $\text{Im}_{current}$ denotes the vector of the current image.

Feature 5: Radial position of the key pixel.

Features 1-5 focus on the global view of the consecutive images. But these features are insufficient to obtain the information for small solar flares. Therefore, it is necessary to include the local view of the solar images.

Feature 6: Contrast between the key pixel and the minimum value of its neighbors in a 7 by 7 window. When flares occur, this contrast becomes large.

Feature 7: Mean brightness of a 50 by 50 window, where the key pixel is on the center. Since we assume that the key pixel is one of the pixels on the flare, the 50 by 50 window will include most of the flare. This region is the most effective for the solar flare detection.

Feature 8: Standard deviation of the pixels in the aforementioned 50 by 50 window.

Feature 9: Difference of the mean brightness of the 50 by 50 window between the current and the previous images.

See samples of the input nine features and the target value in Table I.

Table I: Samples of the nine features and the target value.

1	2	3	4	5	6	7	8	9	Target value	
						3994.25	2628.22	45	12474.00	431.23
						3635.00	2011.43	1978.60	427.35	-1
3807.42	2458.64	187	16383.00	437.13	6484.00	9460.98	2430.92	7449.55	1	

4. Experimental Results

4.1. EXPERIMENTAL DESIGN

The solar $H\alpha$ images of 2032×2032 pixels were obtained from BBSO. We selected the images of flare events starting from January 1, 2002 to December 31, 2002. The first step taken was to carefully divide the images into two classes: flare state and non-flare state. We tried to cover all the flare events that can be recognized by human eyes. We labeled corresponding images as “flare state” and included the same number of “non-flare state” images. Because some micro-flares are difficult to be recognized even by human eyes, we regarded them as “non-flare state”. Secondly, we computed the nine features for a given solar image and put it into either the training or the testing data set. We used the training data set to train the networks and used the testing data set to obtain the classification rate. Finally, we obtained flare properties using image segmentation techniques, since we are also interested in obtaining the properties of flares such as size, lifetime, and motion.

We developed the programs in Interactive Data Language (IDL) by Research Systems, Inc. The program runs on a DELL Dimension L733r with CPU time 733 Mhz and memory of 256 Mbytes under Windows 2000. There are three steps in our program:

- a) Preprocessing to obtain the nine features of solar flares.
- b) MLP, RBF and SVM training and testing programs used for solar flare detection.
- c) Region growing and edge detection methods for obtaining the flare properties.

For the experiments of solar flare detection, we capture one image per minute and process the current image with comparisons of previous image to obtain the nine features. Firstly, we filter the noise of Full-disk images using a Gaussian function and center these two images using the IDL FIT_LIMB function. Secondly, we compute the mean brightness and standard deviation for the two images. Thirdly, we divide them to obtain the pixel with the maximum gray level difference. It is regarded as the key pixel to compute other features. After preprocessing steps, we use the nine features of the key pixel as the input to the neural network and SVM. At last, we use region growing combined with Canny edge detector to analyze the region of the flares.

In Table II, we estimate time to extract the features and perform the classification and obtain the flare properties. Obviously, the obtained short processing time will give us potential to detect flares in real time.

Table II: Time on Preprocessing step, classification step and obtaining the flare properties for each image.

Job Descriptions	Estimate Time
Preprocessing step	6.3 seconds
Classification step (SVM)	0.03 second
Obtaining flare properties (Combining Region growing and Canny Edge detection)	< 2 seconds

4.2. CLASSIFICATION RATE

We tested the MLP, RBF and SVM methods based on the 240 events including pre-flare, main phase and post-flare and half of them were used as the training set and half of them were used as the testing set. For MLP, after about 1000 iterations for the training set of 120 events, we tested this network with 120 events. The fraction of misclassified events is about 5%. MLP uses supervised training and needs a lot of iterations to obtain the satisfied classification rate. For RBF, we obtain a better classification rate and faster training speed. For SVM, we obtain the best performance among the three methods, such that the misclassification rate is 3.3%. The results are shown in Table III.

Table III: Classification report based on 120 training events and 120 testing events.

Methods based on events	Classification rate	Training time	Testing time
MLP with 1000 iterations	94.2%	60.20 seconds	0.01 second
RBF	95%	0.27 second	0.01 second
SVM	96.7%	0.16 second	0.03 second

The MLP architecture is to perform a full non-linear optimization of the network. Therefore, a lot of iterations to train the system to get the best result are needed. In Table IV, it is demonstrated that the result performs better with more iterations. The number of hidden nodes determines the complexities of the neural networks. We can adjust the number of hidden nodes to obtain the best performance. In Table IV, we show the comparison of MLP with 11 hidden nodes and MLP with other number of hidden nodes. Test correctness of MLP with 11 hidden nodes is comparable to MLP with more hidden nodes. MLP with 11 hidden nodes has better performance than MLP with more or less hidden nodes.

Table IV: Comparison of different MLP iterations and hidden nodes on 120 training events and 120 testing events

Methods based on events	Classification rate	Training time
MLP with 1000 iterations and 11 hidden nodes	94.2%	60.2 seconds
MLP with 100 iterations and 11 hidden nodes	69.2%	6.2 seconds
MLP with 3000 iterations and 11 hidden nodes	94.2%	179.3 seconds

MLP with 1000 iterations and 11 hidden nodes	94.2%	60.2 seconds
MLP with 1000 iterations and 6 hidden nodes	92.5%	46.7 seconds
MLP with 1000 iterations and 20 hidden nodes	94.2%	115.6 seconds

RBF neural network, as compared with the MLP, is the possibility of choosing suitable parameters for each hidden unit without having to perform a full non-linear optimization of the network (Bishop, 1995). Usually, the number of hidden nodes equals the number of classes. For our case, we want to use RBF classifier to separate two classes. We try RBF with different number of hidden nodes and find that RBF with two hidden nodes is good for our pattern classification. The parameters of RBF are center vector, the weight and the radius. The training architecture can be quickly constructed by K-mean algorithm.

SVM is based on the idea of “minimization of the structural risk”. To obtain the best classification rate, we consider the confidence interval and empirical risk. Complex system has high confidence interval and low empirical risk. We test our flare events using linear SVM and non-linear SVMs respectively with Polynomial kernel classifier and Gaussian RBF kernel classifier. Experimental results are shown in Table V.

Table V: Comparison of different SVM training strategies on 120 training events and 120 testing events

Methods based on events	Classification rate	Training time	Testing time
Linear SVM	96.7%	0.16 second	0.03 second
SVM with Polynomial kernel	95%	0.34 second	0.03 second
SVM with RBF kernel	90.83%	0.44 second	0.03 second

Table V shows that linear SVM is better than non-linear kernel SVM for our events. Non-linear SVM can reduce the empirical risk using complex systems, but it has higher confidence interval. Using linear SVM with the nine features of the solar flare, we have both low empirical risk and confidence interval to achieve the best total risk. Therefore, we prefer linear SVM to classify the flare patterns.

5. Conclusions

We have presented the comparisons of flare detection using the three advanced techniques which are Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) and Support Vector Machine (SVM). Through experiments, we have found that SVM is the best for the solar flare detection because it offers the best classification result and the training and testing speed are relatively fast. The second choice is RBF. Because its training data are presented only once, it is the fastest neural network. MLP is not a well-controlled learning machine. The sigmoid function has a scaling factor that affects the quality of the approximation, and the convergence of the gradient-based method is rather slow. The result of MLP is not as good as the RBF and SVM.

After the solar flare detection, the properties of flares are analyzed using region growing and canny edge detector. Because these work are well done by Veronig et al. (2000), so we do not put explanation here. For future study, we will apply SVM, RBF and MLP to more solar phenomena such as Filament, Sunspot, and Plages. We believe that we may create the best model to detect the solar phenomena by combining SVM and RBF classifiers.

Acknowledgements

We thank the referees for very helpful comments which lead to significant improvement of the paper. We acknowledge the valuable discussions from Mr. C. F. Chuang in Computer Vision Laboratory at New Jersey Institute of Technology, and Dr. Y. C. Jiang from the Big Bear Solar Observatory. The work is supported by National Science Foundation (NSF) under grants ATM 0076602 and ATM 0233931.

References

- Bishop, C. M.: 1995, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, p. 164.
- Borda, R. A., Mininni, P. D., Mandrini, C. H., Gómez, D. O., Bauer, O. H., and Rovira, M. G.: 2001, *Solar Phys.* **206**, 347-357.
- Guyon, I. and Stork, D. G.: 2000, *Linear discriminant and support vector machine*, The MIT Press, Cambridge, p. 147.
- Haykin, S.: 1994, in J. Griffin (ed.), *Neural Networks*, Macmillan Publishing Company, NY, USA, p. 3.
- Mitra, U. and Poor, H.V.: 1994, *IEEE Journal on Selected Areas in Communications*, **12(9)**:1460-1470.
- Ripley, B. D.: 1996, *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, p. 143.
- Stix, M.: 1989, *The Sun*, Springer-Verlag Berlin Heidelberg, NY, USA, p. 431.
- Vapnik, N.V. and Chervonenkis, A.J.:1974, *Theory of Pattern Recognition*, Nauka, Moscow.
- Vapnik, N.V.:1982, *Estimation of Dependences Based on Empirical Data*, Addendum 1, New York, Springer-Verlag.
- Vapnik, V.N. and Chervonenkis, A.:1991, *Pattern Recognition and Image Analysis*, 1(3), p.283-305.
- Vapnik, N.V.:1995, *The Nature of Statistical Learning Theory*, Springer, New York, p.126-140.
- Vapnik, N. V.:1998, in S. Haykin (ed.), *Statistical Learning Theory*, John Wiley & Sons, Inc., NY, USA, p. 401.

Veronig, A., Steinegger, M., Otruba, W., Hanslmeier, A., Messerotti, M., Temmer, M., Brunner, G., and Gonzi, S.: 2000, *ESA*. **463**, 455.

Zirin, H.: 1988, *Astrophysics of the sun*, Cambridge University Press, Cambridge, p. 343.