Indian Journal of Fibre & Textile Research
Vol. 41, September 2016, pp. 270-277

Pattern classification of cotton yarn neps

Abul Hasnat, Anindya Ghosh[a], Azizul Hoque[b] & Santanu Halder[c]
Government College of Engineering and Textile Technology, Berhampore 742 101, India

Received 12 October 2014; revised received and accepted 23 December 2014

In this study, two types of cotton yarn neps, viz. seed coat and fibrous neps, have been classified by means of two standard classifiers, namely support vector machine and probabilistic neural network using the features extracted from the images of neps. At first, the region of interest is located in the captured images using *k*-means clustering algorithm, from which six features are extracted. These extracted features are used as dataset (both training and testing) for classifiers. A *K*-fold cross validation technique has been applied to assess the performance of the two classifiers. The results show that the neps classification accomplished by means of image recognition through these classifiers achieves nearly 96-97% accuracy for the test data set. Experimental results show that the required time for training probabilistic neural network is significantly less as compared to that of support vector machine.

Keywords: Cotton yarn, Fibrous nep, Pattern classification, Probabilistic neural network, Seed coat nep, Support vector machine

1 Introduction

Now-a-days, fabric defect identification using image processing[1-3] is becoming an important area in textile industry, where it is possible to automate the inspection of fabric defects during weaving[4-6]. Nep is one type of yarn defects which is defined as a knot (or cluster) of highly entangled cotton fibres consisting either entirely of fibres or a seed-coat fragment attached with fibres. Neps are created when fibres become tangled in the process of cotton harvesting, ginning and spinning. Generation of neps causes a very serious problem because the neps that remain in yarns often persist into the final fabrics and adversely affects its quality. For almost all types of yarn or fabric, neps are considered as serious defects. A significant occurrence of neps level in the yarn often causes a fabric to be down-graded or rejected. By and large, cotton yarn neps may be classified into two types, namely seed coat and fibrous neps. So far, no hands-on state-of-the-art technology has been adopted for the identification of these two types of neps in cotton yarns. An online system of identifying the seed coat and fibrous neps would be helpful to the spinners to track back their causes of generation in the downstream processes, which in turn increases the possibility of improving the process control system of a spinning industry.

The present study endeavors to recognize fibrous and seed-coat neps in cotton yarns by means of two standard pattern recognition systems or classifiers, namely support vector machine (SVM)[7-9] and probabilistic neural network (PNN)[10-11] using the features extracted from gray level images of neps. The proposed work may be divided into four steps, namely capturing the images of cotton yarn neps using LEICA microscope, detection of region of interest from the captured image, feature extraction from the region of interest, and finally classification of the features by SVM and PNN.

2 Materials and Methods

2.1 Capturing Images of Yarn Neps

20's Ne carded yarns spun from J-34 cotton fibres of 4 micronaire with 25 mm span length (2.5%) were collected from the spinning industry. The images of yarn neps were captured using a LEICA camera (Model EZ-4D) with a magnification of ×40. Samples were illuminated by three halogen lights positioned approximately 20 cm above directly and to the right and left of the sample to supply illumination in diagonal directions of 45°. Altogether 270 images were captured for the experimentation. All the captured images of size 2048 × 1536 × 3 pixels were

[a]Corresponding author.
E-mail: anindya.textile@gmail.com
[b]Present address: Kalyani University, Kalyani 741 245, India
[c]Present address: Government College of Engineering and Leather Technology, Kolkata 700 098, India

converted into digitized gray images with grey level intensities ranging from 0 to 255 and stored as two-dimensional gray scale matrixes.

2.2 Extraction of Region of Interest

Each digitized gray image was enhanced by applying a median filter to eliminate undesired noise. From the filtered image $I(a,b)$ of size $p \times q$ (where a = 1, 2,...,p; b = 1,2,...,q), the region of interest was extracted by applying k-means clustering algorithm. To locate the position of the nep in the image, the average pixel intensity value of i^{th} row, A_i and the row h with minimum average pixel intensity were calculated using the following equations:

$$A_i = \frac{\sum_{b=1}^{q} I(i,b)}{q} \quad \text{... (1)}$$

$$A_h = MIN\{A_i\} \quad \text{... (2)}$$

where $i = 1, 2.., p$.

In this way, h was located and as a preprocessing step the image was cropped up to 30% of height on both sides of the located row h. It was observed from the experimental result that the region of interest lies within the 60% of cropped image.

In the next step, using pixel intensity as input, k-means clustering algorithm was applied to remove the background pixels from the captured images. The k-means clustering algorithm[12,13] is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given set of n vectors into k (k is an integer) number of clusters. The main idea is to define k centroids $G = \{g_1, g_2, .. g_i, .., g_k\}$, one for each cluster; taking each of vectors belonging to the given set, associating it to the nearest centroid g_i, and then including it in the cluster C_i. Thus, the first step is completed and an early k grouping is done. At this point, we need to re-calculate k new centroids of the clusters, resulting from the previous step. After having these k new centroids, a new grouping has to be done for each vector of the set and once again it is re-associated to nearest new centroid and corresponding clusters. This step is repeated and as a result the k centroids change their location step by step until no more changes occur. Thus, the algorithm converges and gives the final clusters.

In this study, clustering was done by using the pixels intensity values and all pixels which belong to the background cluster were set to a value of 255.

In the next step, from the clustered image the average pixel intensity of j^{th} column, A_j and the column v with minimum average pixel intensity were calculated using the following equations:

$$A_j = \frac{\sum_{a=1}^{p} I(a, j)}{p} \quad \text{... (3)}$$

$$A_v = MIN\{A_j\} \quad \text{... (4)}$$

where $j = 1,2....q$.

After locating the column v in the clustered image, 15% width of the image from both sides of this column was cropped. It was observed from the experimental result that the region of interest lies within this 30% of the clustered image. In this cropped image, only the pixels having intensities of < 255 were considered.

2.3 Feature Extraction

From the extracted region of interest of each image, an array $Arr(i)$ where $i = 1,2,...,N$, was constructed containing pixels intensity values of < 255. Six simple features viz. minimum, maximum, average, standard deviation (SD), median and mode were determined from each array using the following equations:

$$\text{Mimimun} = MIN_{i=1}^{N}\{Arr(i)\} \quad \text{... (5)}$$

$$\text{Maximum} = MAX_{i=1}^{N}\{Arr(i)\} \quad \text{... (6)}$$

$$\text{Average} = \frac{\sum_{i=1}^{N} Arr(i)}{N} \quad \text{... (7)}$$

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{N} (Arr(i) - \text{Average})^2}{N}} \quad \text{... (8)}$$

The median is the middle value of the sorted array (if array size is even the average of the middle two values is considered). Mode is the element having maximum frequency in the array. Fig. 1 shows the region of interest of an image and the features extracted from it. These six features were extracted from each of the 270 captured images.

2.4 Pattern Classifiers (SVM and PNN)

If we consider a set $I=\{x_i\}; i\in\{1,2,3,4,..n\}$ of n input data, $x_i=\{f_1, f_2,.., f_m\}$ of m dimensional space, a set of classes $W=\{w_1, w_2,...w_c\}$ where $c\le n$, then a classification problem is defined as mapping $F: I\to W$ where each X_i in I is assigned to one of the classes of *W*. This mapping function is called decision function. The basic problem in classification is to find c decision functions $d_1(x), d_2(x),..d_c(x)$ with the property that, if a pattern x belongs to class i, then $d_i(x)>d_j(x)$; $i, j=1,2,...c; j\neq i$, $d_i(x)$ is some similarity measure between x and class i, such as distance or probability concept. Classification is a supervised learning where the classifier is trained using training dataset consisting features as input and expected class of each of the data points as output. Based on this training some internal parameters of the classifier are adjusted accordingly. For testing, a new set of data points are fed into the classifier which assigns a class to each of the testing data points. There are many classifiers like *k*-Nearest Neighbor, SVM, artificial neural network (ANN), PNN, etc. In this work, SVM and PNN classifiers are selected for pattern classification due to the fact that the complexity of SVM is indifferent to the dimension of input data and the training time of PNN is insignificant because there is no weight updating process involved in its training.

SVM is a mathematical model derived from the statistical learning theory developed by Vapnik[7]. SVM is supervised learning model with associated learning algorithms that analyze multidimensional data and recognize the patterns. As it is a supervised learning model, for a set of training data, each vector assigned to one of two categories, its training algorithm builds a model that assigns new examples into one category or the other. SVM model is a representation of the examples as points mapped in the hyper space, the examples of the separate categories are divided by hyper plane with a clear gap that is as wide as possible. After training, new examples are mapped into that same space and then classified in to one of the categories, based on which side of the gap they fall on. In addition to performing linear classification, SVM can efficiently perform non-linear classification using the kernel function, implicitly mapping their inputs into high-dimensional feature spaces.

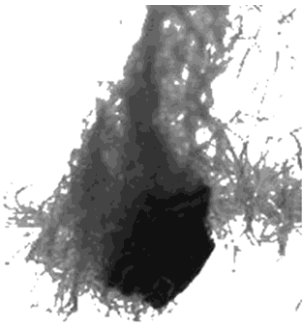


Fig. 1—Region of interest of an image and features extracted from it [minimum 6, maximum 171, average 99.33, standard deviation 49.75, median 112, mode 11]

Consider the problem of separating the set of data points $\{(x_i, y_i)\}: i=1,2,..n$ where for $y_i=+1,\ w^Tx_i+b>0$ and for $y_i=-1,\ w^Tx_i+b<0$ (Fig. 2). With a scale transformation of both *w* and *b*, these two equations are equivalent to $w^Tx_i+b\ge 1$, when $y_i=+1$ and $w^Tx_i+b\le -1$, when $y_i=-1$. These two equations can be combined as $y_i(w^Tx_i+b)\ge 1$. Therefore, the corresponding margins can be defined by $w^Tx^+ +b=1$ and $w^Tx^- +b=-1$. The margin width, *M* is given by

$$M=\frac{2}{\|w\|} \qquad \text{... (9)}$$

In case of maximal margin classifier, the margin width *M* is maximized and the problem could be formulated as

Maximize $\frac{2}{\|w\|}$,

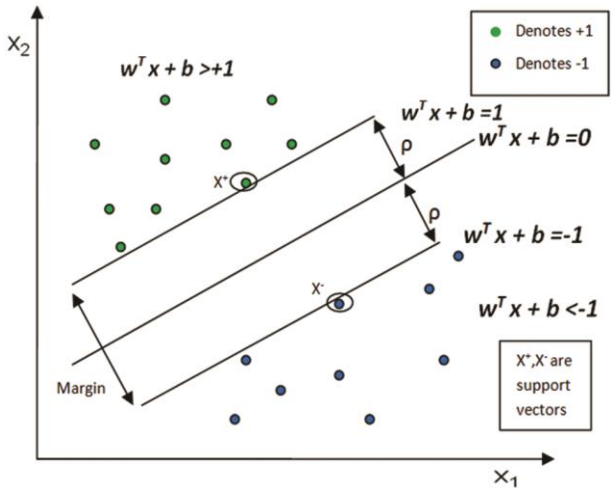


Fig.2—Support vector machine as classifier

Subject to $y_i(w^T x_i + b) \geq 1$.

Alternatively, the optimization problem becomes

Minimize $f = \frac{1}{2}\|w\|$

subject to $y_i(w^T x_i + b) \geq 1$.

The main problem with the maximal margin classifier is that it always produces perfectly consistent hypothesis, which is a hypothesis with no training error. In real data, where noise can always be present, this can result in a brittle estimator. These problems can be overcome by using the soft-margin optimization, where we need to introduce slack variables (ξ_i) to allow the margin constraints to be violated, as shown below:

$y_i(w^T x_i + b) \geq 1 - \xi_i$,

$\xi_i \geq 0$.

Thus, the soft-margin optimization problem becomes

$$\text{Minimize } f = \frac{1}{2}\|w\| + \rho\sum_{i=1}^{l}\xi_i \quad \text{... (10)}$$

subject to $y_i(w^T x_i + b) \geq 1 - \xi_i$

where ρ is a pre-specified value.

On the other hand, PNN is a feed forward neural network, which is derived from the Bayesian network and a statistical algorithm called Kernel Fisher discriminant analysis. It was introduced by Specht[10] in early 1990s. The PNN architecture is feed forward in nature, but differs in the way that learning occurs in ANN. PNN is supervised learning algorithm but includes no weights in its hidden layer. Instead each hidden node represents an example vector, with the example acting as the weights to that hidden node. These are not adjusted at all.

PNN is mainly used in classification problems. A schematic architecture of PNN is illustrated in Fig. 3. When an input is presented, the hidden layer computes the distance from the input vector to the training input vectors. This produces a vector where its elements indicate how close the input is to the training inputs. The summation layer sums the contribution for each class of inputs and produces its net output as a vector of probabilities. Finally, a compete transfer function on the output of the summation layer picks the maximum of these probabilities and indicates that particular class as the output. The function of each layer of PNN is discussed below.

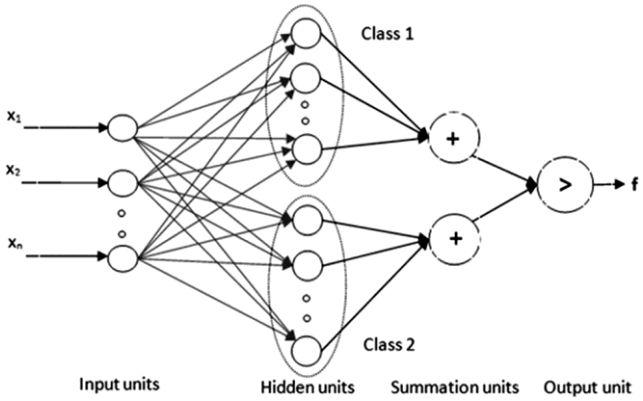


Fig. 3—A schematic representation of the PNN architecture

(i) Input Layer

The input layer contains m nodes (in this work m = 6) for each of the input features of vector, $x_i = \{f_1, f_2, .., f_6\}$. These are fan-out nodes that branch at each feature input node to all nodes in the hidden layer so that each hidden node receives the complete input feature vectors.

(ii) Hidden Layer/Pattern Layer

This layer contains one neuron for each vector in the training data set. It stores the values of the predictor variables for the vector along with the target value. A hidden neuron computes the Euclidean distance of the test case from the neuron's center point [that is the stored vector, $x^{(P)}$] and then maps it to radial basis function (RBF) as given in the following equation:

$$f(x) = \frac{1}{\sqrt{(2\pi\sigma^2)^m}} \exp^{[-\|x - x^{(P)}\|^2 / 2\sigma^2]} \quad \text{... (11)}$$

The σ values can be taken to be one-half the average distance between the feature vectors in the same group or at each exemplar it can be one-half the distance from the exemplar to its nearest other exemplar vector.

(iii) Summation Layer

The summation layer neurons compute the maximum likelihood of pattern, x being classified into class c_k; $k = 1, 2$ by summarizing and averaging the output of all neurons that belong to the same class. The actual target category of each training case is stored with each hidden neuron; all the weighted values coming out from hidden neurons (of specific class) are fed only to the summation neuron that

corresponds to the hidden neuron's category. The k^{th} summation node sums up the values received from the k^{th} group of hidden nodes using the following equation:

$$p_k(x) = [\frac{1}{\sqrt{(2\pi\sigma^2)^m}}]\frac{1}{P}\sum_{P=1}^{P}\exp^{[-\left\|x-x^{(P)}\right\|^2/2\sigma^2]} \quad \text{... (12)}$$

where P is the number of hidden nodes for a particular class.

(iv) Output Layer

The decision layer/output unit classifies the pattern, x in accordance with the Bayes's decision rule based on the output of all the summation layer neurons using the following equation:

$$\hat{c}(k) = \arg\max_{k=1}^{2}\{p_k(x)\} \quad \text{... (13)}$$

Thus, the PNN recognizes the test vector in k^{th} class.

2.5 Cross Validation

A *K*-fold cross validation technique[14] has been applied to assess the accuracy of both the classifiers. In *K*-fold cross validation, the initial dataset is randomly partitioned into *K* mutually exclusive subsets or folds D_1, D_2, … , D_K, each of approximately equal size. The training and testing are performed *K* times. In the i^{th} iteration, partition D_i is reserved as the test set and the remaining partition are collectively used to train the model. In this method, each data point is used for the same number of times for training and once for testing. Therefore, the validation of the model becomes more accurate and unbiased. The *K*-fold cross validation method is schematically depicted in Fig. 4, where $1, 2, ..., K$ represent the fold corresponding to testing dataset.

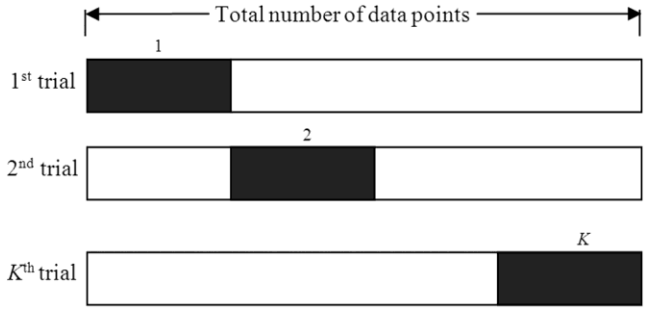


Fig. 4—Schematic representation of *K*-fold cross-validation

3 Results and Discussion

Altogether 270 captured images of cotton yarn neps have been considered for experimentation and regions of interest are extracted from all the images using the method as described before. In Table 1, second and third column show some captured images and their corresponding regions of interest respectively. The last column shows the extracted features from the region of interest of the captured images.

Table 2 depicts only a subset of 60 data points choosing 30 from each category of neps, in which f_i, $i = 1,2..6$ are the extracted features corresponding to minimum, maximum, average, standard deviation, median and mode respectively. The last column of Table 2 shows associated type of the respective nep (1 for fibrous neps and 2 for seed coat nep). Each of the extracted features is normalized in the range [0 1]. The final dataset consists of six normalized features representing each of the 270 images as inputs and their respective associated class as output. A 10-fold cross validation technique has been applied for cross validation for which both PNN and SVM classifiers are trained with 9 of the folds and tested on the fold left out. Therefore, training and testing are done for 10 times for both the classifiers. The expected generalization accuracies referring to training as well as testing are estimated as average accuracy ± standard deviation of 10 cycles.

In case of PNN, the value of σ is tuned to be 0.07 by means of trial and error. For SVM, the value of ρ is tuned to be 20 and radial basis function is used as a kernel function with a kernel width equals to 2. The expected generalization accuracies of training and testing for PNN classifier are found to be 99.42 ± 0.39 and 95.92 ± 3.68 respectively, whereas the same for SVM are found to be 97.12 ± 0.47 and 97.03 ± 3.82 respectively. For both the classifiers, training accuracy is found to be better than the testing accuracy because the latter is done on the unseen data.

As far as the accuracy of testing dataset is concerned, SVM gives higher accuracy in classification than the PNN. This may be ascribed to the better generalization capacity and ability of handling noisy data by SVM. On the other hand, PNN gives higher accuracy on the training dataset because it simply stores a training vector in the respective class by creating a new node in the hidden layer and there is no weight update or convergence process

Table 1— Extracted features from the region of interest of few captured images

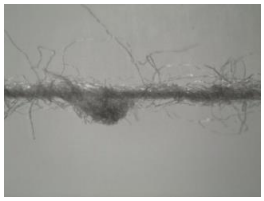
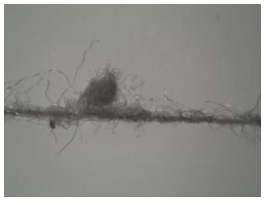
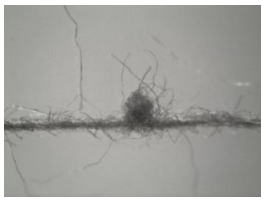
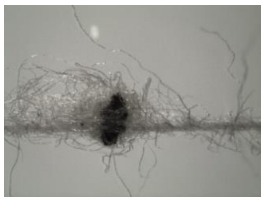
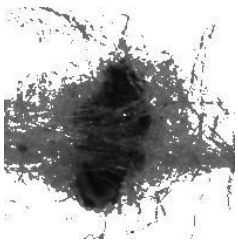
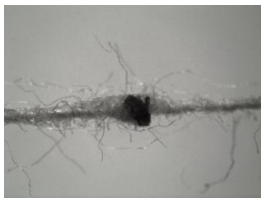
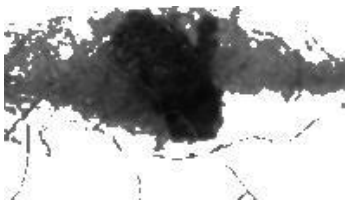
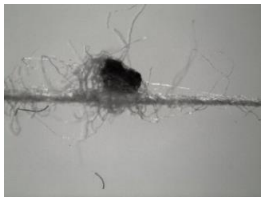
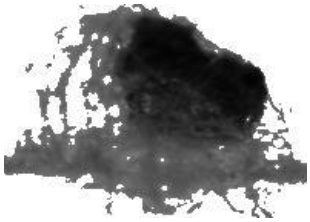
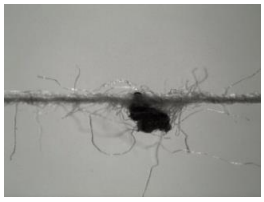
Captured image	Region of interest	Extracted features
		Minimum =56 Maximum =128 Average =98.89 SD =14.96 Median =99 Mode = 98
		Minimum =35 Maximum =113 Average =84.63 SD =14.24 Median =83 Mode = 76
		Minimum = 43 Maximum =132 Average = 95.93 SD =20.67 Median = 95 Mode = 83
		Minimum = 8 Maximum =121 Average = 77.37 SD =33.73 Median = 88 Mode = 114
		Minimum = 7 Maximum = 122 Average = 69.77 SD =35.29 Median = 76 Mode = 23
		Minimum = 4 Maximum =119 Average =70.14 SD =35.58 Median = 84 Mode = 85
		Minimum = 3 Maximum = 108 Average = 48.08 SD =32.21 Median = 44 Mode = 11

Table 2—A sample of extracted features from images

SL	f_1	f_2	f_3	f_4	f_5	f_6	Type
1	32	162	122.66	27.084	128	147	1
2	25	128	92.205	20.81	91	85	1
3	49	135	98.735	21.376	99	84	1
4	35	137	98.832	23.564	97	80	1
5	45	144	110.8	21.72	113	136	1
6	33	120	78.141	21.455	77	56	1
7	49	139	103.79	21.001	105	105	1
8	49	149	96.736	24.129	92	82	1
9	31	155	104.99	32.492	110	128	1
10	24	122	80.799	24.182	82	82	1
11	63	149	115.55	21.114	116	145	1
12	48	126	93.466	20.044	93	123	1
13	68	147	113.14	18.717	111	95	1
14	55	141	106.45	20.096	108	105	1
15	26	136	92.899	27.592	97	104	1
16	59	139	101.46	20.928	98	80	1
17	71	148	122.09	18.58	125	148	1
18	36	149	93.434	28.676	92	95	1
19	62	133	106.19	17.252	107	129	1
20	65	131	103.74	16.093	103	93	1
21	44	127	95.238	17.74	95	94	1
22	26	135	84.869	25.565	81	57	1
23	36	124	87.821	20.007	88	81	1
24	37	129	93.147	20.924	93	85	1
25	56	132	104.73	18.329	107	111	1
26	50	127	96.762	16.487	97	97	1
27	56	115	90.57	13.801	91	94	1
28	44	109	82.544	16.771	83	106	1
29	26	112	74.132	22.312	77	87	1
30	37	131	91.148	22.537	92	84	1
31	4	102	44.423	29.626	44	13	2
32	11	110	64.054	26.829	68	69	2
33	6	141	78.281	39.671	83	12	2
34	12	130	89.742	31.06	96	126	2
35	14	124	82.045	34.126	92	124	2
36	6	151	90.424	42.672	99	13	2
37	9	108	60.654	26.744	62	12	2
38	7	112	69.777	26.078	70	63	2
39	5	121	65.718	33.268	72	10	2
40	5	111	61.797	28.61	65	63	2
41	19	97	66.558	17.724	68	62	2
42	14	103	78.325	19.754	82	80	2
43	8	97	66.314	20.257	71	74	2
44	9	120	78.859	33.141	91	107	2
45	7	131	82.998	33.561	91	97	2
46	11	118	70.372	29.834	72	97	2
47	10	135	93.51	31.294	100	111	2
48	16	104	78.171	19.409	81	98	2
49	13	101	73.364	20.197	77	77	2
50	23	110	76.259	20.585	78	87	2
51	13	134	84.898	33.47	91	108	2
52	8	124	82.077	24.027	82	79	2
53	17	104	75.206	19.853	79	79	2
54	10	103	74.555	22.067	81	93	2
55	7	112	64.983	25.947	67	61	2
56	26	115	85.578	18.376	87	89	2
57	12	123	86.737	26.951	92	122	2
58	5	119	88.246	20.872	89	86	2
59	6	110	72.184	24.005	72	64	2
60	11	115	82.147	21.908	84	84	2

f_1–minimum, f_2–maximum, f_3–average, f_4–standard deviation, f_5–median, f_6–mode, Type 1–fibrous nep, Type 2–seed coat nep.

involved. As a consequence, PNN does not require significant amount of time for training the system. Thus, PNN works quite faster compared to SVM classifier.

4 Conclusion

The present study holds the key of effective classification of cotton yarn neps. It has been observed that both PNN and SVM are able to classify fibrous and seed coat neps in the cotton yarn with a reasonable degree of accuracy. SVM gives higher accuracy in classification on the testing dataset in comparison to PNN. The time required for the training of PNN is awfully small as the training process does not need any weight update process but it simply adds a node in the hidden layer for each of the training vector. The proposed method of classifying seed coat and fibrous neps is a guide to improve the process control system in a cotton spinning industry.

References

1 Pratt W K, *Digital Image Processing*, 2nd edn (Wiley Inter Science, New York), 1991.

2 Umbaugh S E, *Computer Vision and Image Processing* (Prentice-Hall, Upper Saddle River, New Jersey), 1998.

3 Gonzalez R C & Woods R E, *Digital Image Processing* (Addison Wesley, New York), 1993.

4 Tsai S, Lin C H & Lin J J, *Text Res J*, 65 (1995) 123.

5 Kumar A & Pang G, *IEEE Transact Industrial Appl*, 38 (2002) 425.

6 Ghosh A, Guha T, Bhar R B & Das S, *Int J Cloth Sci Technol*, 23 (2011)142.

7 Vapnik V N, *Statistical Learning Theory* (John Wieley & Sons, New York), 1998.

8 Cristianini N & Taylor J S, *An Introduction to Support Vector Machine and other Kernel based Learning Methods* (Cambridge University Press, Cambridge), 2000.

9 Schlkopf B & Smola A J, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond* (MIT Press, Massachusetts), 2001.

10 Specht D F, *IEEE Transact Neural Networks*, 3 (1990) 109.

11 Ibrahiem M M, Emary E & Ramakrishnan S, *World Appl Sci J*, 4 (2008) 772.

12 Hartigan J A, *Clustering Algorithms* (Wiley, New York), 1975.

13 Gan G, Ma C & Wu J, *Data clustering: Theory, Algorithms and Applications* (SIAM Press, Philadelphia), 2007.

14 Han J & Kamber M, *Data Mining Concepts and Techniques*, 2nd edn (Morgan Kaufmann Publishers, San Francisco), 2006.