Neural Networks 64 (2015) 19–28

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2015 Special Issue

Deep learning of support vector machines with class probability output networks

Sangwook Kim[a], Zhibin Yu[a], Rhee Man Kil[b,*], Minho Lee[a,*]

[a] School of Electronics Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, Republic of Korea

[b] College of Information and Communication Engineering, Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon, Gyeonggi-do 440-746, Republic of Korea

ARTICLE INFO

Article history:
Available online 7 October 2014

Keywords:
Deep learning
Support vector machine
Class probability output network
Uncertainty measure

ABSTRACT

Deep learning methods endeavor to learn features automatically at multiple levels and allow systems to learn complex functions mapping from the input space to the output space for the given data. The ability to learn powerful features automatically is increasingly important as the volume of data and range of applications of machine learning methods continues to grow. This paper proposes a new deep architecture that uses support vector machines (SVMs) with class probability output networks (CPONs) to provide better generalization power for pattern classification problems. As a result, deep features are extracted without additional feature engineering steps, using multiple layers of the SVM classifiers with CPONs. The proposed structure closely approaches the ideal Bayes classifier as the number of layers increases. Using a simulation of classification problems, the effectiveness of the proposed method is demonstrated.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Despite the early recognition of the importance of deep architecture (Tesauro, 1992; Utgoff & Stracuzzi, 2002), deep learning research has not been prevalent because there has been no useful learning method applicable for existing learning machines except for the models proposed by Fukushima (1980), LeCun et al. (1989). The Restricted Boltzmann Machine (RBM), initially invented by Smolensky in 1986 (Smolensky, 1986), is a generative stochastic neural network that can learn probability distribution over its set of inputs. With the proposal of Hinton and colleagues to utilize the RBM network with contrastive divergence (Hinton, Osindero, & Teh, 2006), deep architectures using the RBM network have become popular for pattern recognition. It has been successful in solving several engineering problems including speech and video signal processing (Arel, Rose, & Karnowski, 2010; LeCun et al., 1989). The essence of deep learning methods is their ability to automatically extract features through deep (or multiple) layers. Since the feature extraction is critical to the performance of machine learning and pattern recognition techniques, it has been used in several domain-specific, feature-engineering techniques. However, deep learning methods such as deep belief networks provide superior performance without additional feature engineering (Bengio, 2009).

Although well-trained deep networks provide good performance, the learning algorithm requires accurate configuration and user-determined hyper-parameters. For example, a RBM network uses hyper-parameters such as the learning rate, momentum, weight norm regularization, initial value of the weights, number of hidden units, and size of each mini-batch (Hinton, 2010). If these hyper-parameters are not selected appropriately, the training cannot be performed efficiently and will suffer from an overfitting problem or more critically, no convergence at all. Consequently, the model will provide a significantly reduced test accuracy with a perfect training accuracy. This usually happens when there is insufficient training data or the model is highly biased to the training data. Without careful learning parameter selections, usual machine learning models result in an unsuccessful generalization performance. To alleviate the overfitting problem, a number of methods have been proposed. For example, the cross-validation technique divides the training dataset into two separate sets—training and validation. By testing the models using the validation set after the training set, the model with the best generalization performance in the validation set is selected.

Boosting is a supervised machine learning meta-algorithm for reducing bias to achieve generalized learning (Schapire,

* Corresponding authors. Tel.: +82 31 299 4959; fax: +82 53 950 6436.
E-mail addresses: rmkil@skku.edu (R.M. Kil), mholee@knu.ac.kr, mholee@gmail.com (M. Lee).

http://dx.doi.org/10.1016/j.neunet.2014.09.007
0893-6080/© 2014 Elsevier Ltd. All rights reserved.

2003). These methods assume the availability of a weak learning algorithm that, given labeled training examples, produces a weak classifier or hypothesis. The objective of boosting is to create a strong classifier consisting of weak classifiers while treating the weak classifiers as the "black box". The principal concept of boosting is to choose training sets for the base learner in such a fashion as to force it to infer something new about the data each time it is called. This can be accomplished by choosing training sets for which we can reasonably expect the performance of the preceding base classifiers to be inadequate (Schapire & Freund, 2012). For example, the AdaBoost method learns datasets by re-weighting the distribution of the samples, i.e., the method evaluates the accuracy of the classifiers and the hard samples receive more weight in the later stages than the easy samples.

Support vector machine is one of the most representative models of shallow networks and it delivers a high generalization performance (Cortes & Vapnik, 1995). It utilizes the structural risk minimization (SRM) principle and attempts to avoid the overfitting problem by acquiring the decision hyper-plane. This is optimal for the maximum margin between classes. To benefit from the generalization effects of SVMs, Kim et al. proposed a deep network using SVMs (Kim, Kavuri, & Lee, 2013), but still the criteria for classifiers' uncertainty are not included. Moreover, the output of SVM did not represent the conditional probability for the given pattern. There existed another effort in which the SVM was also used as the final classification unit of a Convolutional Neural Network rather than the components of the network to obtain more discrimination power of classification (Tang, 2013). Wiering et al. introduced the back-propagation-like optimization method for SVM networks (Wiering, Schutten, Millea, Meijster, & Schomaker, 2013). However, it had the fixed structure of just two layers of SVMs and actually did not have the deep structure. In this context, the class probability output network (CPON) was proposed for classification problems using the conditional probability estimates for a given pattern (Park & Kil, 2009). In the CPON, the output distribution of the discriminative classifier is identified by the beta distribution parameters and provides the *p*-value, a measure of hypothesis testing for classification problems. Furthermore, the CPON provides the uncertainty measure representing the degree of uncertainty when the decision of classification is made (Kim, Kil, & Lee, 2011).

In this paper, we propose a new deep network model using SVMs with CPONs in which the network is optimized using the SRM principle and the CPON provides the *p*-value, the measure of hypothesis testing, for the classification problems. This direction of classification model is in line with the concept of stacked generalization (Wolpert, 1992) in which the stacked generalization is able to minimize the generalization error rate. For stacked generalization models, Ting and Witten claimed that best performances were able to be obtained when the higher-level model combines the confidence levels (or class probabilities), not just the predicted values of lower level ones (Ting & Witten, 1999). For the probabilistic scaling of classifier's output, one popular method is to use the sigmoid type scaling functions using the first (Platt, 1999) and second (Hastie, Tibshirani, Jordan, Kearns, & Solla, 1998) order polynomials of the classifier's output. In these methods, the parameters of the sigmoid type scaling function are estimated from the classifier output samples. These probabilistic scaling methods showed the improved performances for some cases of classification problems, but they do not provide consistent improvement of the classification performances because the scaling method does not always estimate the posterior probability of class membership accurately if the distribution of the classifier's output does not fit the suggested scaling function. In this context, we consider a new method of class probability output network (CPON) in which the posterior probability of class membership is estimated using the beta distribution parameters under the assumption that the output of classifier lies within the finite range and the distribution of classifier's output is usually uni-modal; that is, the distribution has one value that occurs with the greatest frequency. This assumption is quite reasonable for many cases of classification problems with the proper selection of kernels for the given data. The good features of beta distribution are that any symmetric or asymmetric uni-modal distribution is well fitted to the beta distribution using two parameters, and that these parameters are easily estimated from the mean and variance of data. The proposed model also provides the uncertainty measure for the decision of continuing deep layer learning. Using the uncertainty measure, the training patterns are divided into the certain and uncertain patterns. In our method, the uncertain patterns are used for retraining of classifiers. This direction is similar with the boosting methods (Freund, 1995; Schapire, Freund, Bartlett, & Lee, 1998) in which weak classifiers are linearly combined to reduce the variance of estimator (or classifier) while the proposed model is using the concept of stacked generalization to achieve the ideal Bayes classification. In the proposed model, the uncertain patterns are propagated to the upper layer of the deep network architecture and trained in the layer of SVMs with CPONs. Then, the uncertain patterns are selected using the uncertainty measure. These procedures are repeated until the total uncertainty measure is sufficiently small. As a result, the proposed deep network model (using the deep SVMs with CPONs) becomes very close to the ideal Bayes classifier as the number of layers increases. Boosting methods just combine weak classifiers to create a strong classification. In these methods, each classifier's weight is evaluated and the importance of the sample is adjusted based on the accuracy in the training iterations. The proposed model evaluates the uncertainty of the decision on the sample with a probabilistic concept, rather than weak classifiers. Moreover, conventional boosting methods with cascading architecture consider only the rejection ratio and sample importance in the training process. The proposed model considers the uncertainty level as well as SRM in the training process. Further, boosting methods use primitive input data in the training process for all the weak classifiers in the model. However, the proposed model utilizes the previous layer by projecting the input data to a new feature space that is constructed by CPONs with SVMs. These feature extraction steps imply an abstraction that does not exist in boosting. The effectiveness of the proposed method is also demonstrated in this paper through the simulation of classification problems.

The remainder of this paper is organized as follows: Section 2 reviews pattern classification models. Section 3 describes the structure of the proposed model. Experimental results are given in Section 4. Finally, concluding remarks are presented in Section 5.

2. Pattern classification models

For pattern classification problems, a deep structure network consisting of multiple layers of SVMs with CPONs is proposed. Pattern classifiers provide an output $\hat{y}$ as a discriminant value for the given input pattern $\mathbf{x}$. First, let us consider a binary classification problem. The discriminant function h can be formulated as:

$$\hat{y} = h(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in X$ (an input space, an arbitrary subset of $\mathcal{R}^{\boldsymbol{d}}$) represents the d dimensional input pattern and $\hat{y} \in \mathcal{R}$ represents the output of the discriminant function h as an estimate for the output pattern $y \in Y$ (an output space, a set of $\{-1, +1\}$).

Assume we have K classes. One method of implementing multi-class discriminant functions is to construct K discriminant functions for each class as follows:,

$$\hat{y}_k = h_k(\mathbf{x}), \quad \text{for } k = 1, \ldots, K. \tag{2}$$

From these class outputs, the decision is made using the maximum discriminant value such as

$$\text{class} = \arg\max_k \hat{y}_k. \tag{3}$$

In general, the discriminant function h_k can be constructed by a linear combination of kernel functions; that is,

$$h_k(\mathbf{x}) = \sum_{j=1}^{m_k} \omega_{kj}\phi_{kj}(\mathbf{x}), \tag{4}$$

where m_k, ω_{kj}, and ϕ_{kj} represent the number of kernel functions, the jth weight value, and the kernel function for the kth discriminant function, respectively.

These discriminant functions are trained for a set of n training samples,

$$\mathcal{S} = \{(\mathbf{x_i}, y_i) | i = 1, \ldots, n\},$$

where $\mathbf{x}_i$ and y_i represent the ith input and output samples that are drawn randomly and independently from a population of samples with the probability distribution of $P(\mathbf{x}, y)$. We assume that the d dimensional input pattern $\mathbf{x}_i \in \mathcal{R}^{\boldsymbol{d}}$ and the output pattern $y_i \in [-1, +1]^K$ are given as a pair of samples.

For these samples, the goal of learning for each class is to construct an estimation function h_k that minimizes the expected risk

$$R(h_k) = \int_{X \times Y} L(y_k, h_k(\mathbf{x})) dP(\mathbf{x}, y_k), \tag{5}$$

where y_k and L represent the target value and loss function for the kth class, respectively.

In classification problems, the loss function L is given by

$$L(y_k, h_k(\mathbf{x})) = \begin{cases} 1 & \text{if } y_k \cdot h_k(\mathbf{x}) \leq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

To minimize the expected risk of Eq. (5), it is necessary to identify the probability distribution $P(\mathbf{x}, y_k)$, however, this is usually unknown. Rather, we usually find h_k by minimizing the empirical risk $R_{emp}(h_k)$ evaluated by the mean of loss function values for the given samples using:

$$R_{emp}(h_k) = \frac{1}{n} \sum_{i=1}^{n} L(y_{ki}, h_k(\mathbf{x}_i)), \tag{7}$$

where y_{ki} represents the ith output sample for the kth class.

A method to resolve this problem is to employ the structural risk minimization (SRM) principle embodied by SVM (Cortes & Vapnik, 1995). To describe SVM, let us consider the discriminant function for a binary classification as the form of Eq. (4) with the additional bias term ω_0:

$$h(\mathbf{x}) = \sum_{j=1}^{m} \omega_j K\left(\mathbf{x}, \mathbf{x}_j\right) + \omega_0, \tag{8}$$

where m represents the number of kernel functions and K represents Mercer's kernel, a continuous, symmetric, and positive definite kernel.

In SVM, the problem of learning binary classification is given by

$$\min_{\omega_j} \frac{1}{2} \sum_{j=1}^{m} \omega_j^2 + C \sum_{i=1}^{n} \xi_i, \tag{9}$$

subject to

$$y_i \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \; i = 1, \ldots, n$$

where C represents the positive regularization constant indicating the tradeoff between the empirical error and the complexity term, and ξ_i represents the slack variable for the ith pattern indicating the tolerance of the margin for the separating hyper-plane.

This problem can be converted to the following optimization problem using the duality principle (Vapnik, 1998):

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{10}$$

subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0,$$

where α_i represents the Lagrangian multiplier for the ith sample.

This problem can be solved by the quadratic optimization problem (Luenberger, 1968). Then, only the support vectors; that is, the samples corresponding to the non-zero values of α_i, remain to constitute the solution of the hyper-plane as described in the Karush–Kuhn–Tucker (KKT) theorem (Ben-Tal & Zowe, 1982; Guignard, 1969). Let $\hat{\alpha}_i$ represent the solution of Eq. (10). The final form of the hyper-plane is determined by

$$h(\mathbf{x}) = \sum_{i \in I} y_i \hat{\alpha}_i K(\mathbf{x}, \mathbf{x}_i) + \hat{\omega}_0, \tag{11}$$

where I represents the index set of support vectors and $\hat{\omega}_0$ represents the solution of the bias term. Here, the number of support vectors is much smaller than the number of samples and this yields a sparsity of the kernel functions. However, in general, this output value is not related to the probability of the class. In this context, the class probability output network (CPON) (Park & Kil, 2009) was proposed to obtain the posterior probability of class membership from the distribution of the classifier's output data. In the proposed CPON, the output distribution is identified by the beta distribution parameters under the assumption that the output of the classifier lies within the finite range and the distribution of the classifier's output is usually uni-modal. This assumption is quite reasonable for many cases of classification problems with the proper selection of kernels for the given data. The positive features of the beta distribution are that any symmetric or non-symmetric uni-modal distribution is well fitted to the beta distribution using two parameters a and b. The beta distribution is given by

$$f_Y(y|a, b) = \frac{1}{B(a, b)} y^{a-1}(1-y)^{b-1}, \tag{12}$$

where a and b represent positive constants for the beta density function, and $B(a, b)$ represents the beta function defined by

$$B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx. \tag{13}$$

In the proposed model, we assume that the values of the classifier's output are normalized between 0 and 1. Thus, y represents the normalized output of Eq. (11).

This beta function is related to the gamma function Γ through the following identity:

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}, \tag{14}$$

where the gamma function is defined by

$$\Gamma(a) = \int_0^{\infty} \lambda e^{-\lambda x} (\lambda x)^{a-1} dx, \quad \lambda > 0.$$

The cumulative distribution function (CDF) of the beta random variable Y is described by

$$F_Y(y|a,b) = \frac{1}{B(a,b)} \int_0^y x^{a-1}(1-x)^{b-1}\,dx. \tag{15}$$

For the construction of CPON, the SVM is used as a nonlinear mapping function from the input to output space because this classifier provides sparse representation of training patterns using the SRM principle. Then, the output is normalized between 0 and 1 using a linear scaling method. For the normalized classifier's output distribution, the positive and negative classes are approximated by the beta distribution parameters. In this training of classifiers, the beta parameters and kernel parameters such as kernel widths are adjusted such that the classifier's output distributions approach to the ideal beta distributions. For the detailed training procedure, refer to Park and Kil (2009).

When the CPON is trained, the classification for an unknown pattern can be determined by the beta distribution for each class. First, for the unknown pattern, the normalized output y for the classifier is computed. Here, if the normalized value is greater than 1, we set that value to 1; if the value is less than 0, we set that value to 0. Then, the following conditional probability of the + class is given the normalized SVM output y; thus, the output of CPON $F^+(y)$ is determined by:

$$\begin{aligned} F^+(y) &= P\left(+|Y^+ \le y \text{ or } Y^- \ge y\right) \\ &= \frac{F_{Y^+}(y)P(+)}{F_{Y^+}(y)P(+) + (1 - F_{Y^-}(y))P(-)}, \end{aligned} \tag{16}$$

where $F_{Y^+}(y)$ and $F_{Y^-}(y)$ represent the CDFs of the positive and negative output using the CDF of the beta distribution of Eq. (15), respectively.

In the case that both class probabilities are equal; that is, $P(+) = P(-)$, the output of CPON $F^+(y)$ becomes

$$F^+(y) = \frac{F_{Y^+}(y)}{F_{Y^+}(y) + (1 - F_{Y^-}(y))}. \tag{17}$$

This output implies the p-value ratio of the testing hypotheses of the positive and negative classes as follows:

$$F^+(y) = \frac{p\text{-value of testing } H^+}{p\text{-value of testing } H^+ + p\text{-value of testing } H^-}, \tag{18}$$

where H^+ and H^- represent the hypotheses that the given instance belongs to the positive and negative classes, respectively. Then, the final decision can be made using the conditional class probability for the given pattern as follows:

$$\text{class} = \begin{cases} \text{positive} & \text{if } F^+(y) > 0.5 \\ \text{negative} & \text{otherwise.} \end{cases} \tag{19}$$

If the value of $F^+(y)$ is greater than 0.5, the probability of error for the + class is greater than the probability of error for the − class when the decision is −. The opposite is also true. Therefore, the decision should be +. Thus, $F_{Y^+}(y)$ represents the degree of confidence for the + class. Furthermore, in the case that both + and − labels are observed with equal probabilities, this decision becomes the ideal Bayes decision for pattern classification problems. For a detailed description of applying CPONs to pattern classification problems, refer to Park and Kil (2009), Rosas, Kil, and Han (2010).

The suggested CPON provides an effective method of estimating conditional probabilities for pattern classification problems. However, in practice, the conditional class probabilities are estimated from a limited amount of data. Consequently, the estimated conditional class probabilities are not accurate and include uncertainties. In practice, the CPON output is determined by

$$\hat{F}^+(y) = \frac{\hat{F}_{Y^+}(y)}{\hat{F}_{Y^+}(y) + (1 - \hat{F}_{Y^-}(y))}, \tag{20}$$

where $\hat{F}^+(y)$ and $\hat{F}^-(y)$ represent the estimated CDF values of the positive and negative classes from the data, respectively.

Then, we can determine the degree of uncertainty for the classification decision. This can be achieved by estimating the confidence intervals for the conditional class probabilities. These confidence intervals can be determined with the following procedure using the Kolmogorov–Smirnov (K–S) statistic (Rohatgi, Saleh, & Ehsanes, 2001):

- First, find the distance measures $D^{\pm}_{n,a}$ for the positive and negative classes; they are

$$D^+_{n,a} = \frac{K_\alpha}{\sqrt{n^+}} \quad \text{and} \quad D^-_{n,a} = \frac{K_\alpha}{\sqrt{n^-}}, \tag{21}$$

where n^+ and n^- represent the sample size of the positive and negative classes, respectively, and K_α represents the value that satisfies $H(K_\alpha) = 1 - \alpha$, where $H(t)$ represents the CDF of the K–S statistic:

$$H(t) = \frac{\sqrt{2\pi}}{t} \sum_{i=1}^{\infty} e^{-(2i-1)^2\pi^2/(8t^2)}. \tag{22}$$

- Set the variables $u^{\pm}$ as follows:

$$u^+ = F_{Y^+}(y) \quad \text{and} \quad u^- = F_{Y^-}(y). \tag{23}$$

- Determine $100(1-\alpha)$ percent confidence intervals for the CDFs of the positive and negative classes:

$$F^*_{U^+}\left(u^+\right) - D^+_{n,\alpha} \le F_{y^+}(y) \le F^*_{U^+}\left(u^+\right) + D^+_{n,\alpha}, \tag{24}$$

and

$$\begin{aligned} &1 - F^*_{U^-}\left(u^-\right) - D^-_{n,\alpha} \\ &\quad \le 1 - F_{y^-}(y) \le 1 - F^*_{U^-}\left(u^-\right) + D^-_{n,\alpha}, \end{aligned} \tag{25}$$

where $F^*_{U^+}(u^+)$ and $F^*_{U^-}(u^-)$ represent the empirical CDFs of the uniform distribution for the positive and negative classes, respectively.

The two-sided confidence intervals of Eqs. (24) and (25) can be described by one-sided confidence intervals as follows:

- For the positive class, with a probability of $1 - \alpha/2$,

$$F_{Y^+}(y) \le F^*_{U^+}\left(u^+\right) + D^+_{n,\alpha} \quad \text{or} \tag{26}$$

$$F_{Y^+}(y) \ge F^*_{U^+}\left(u^+\right) - D^+_{n,\alpha}. \tag{27}$$

- For the negative class, with a probability of $1 - \alpha/2$,

$$1 - F_{Y^-}(y) \le 1 - F^*_{U^-}\left(u^-\right) + D^-_{n,\alpha} \quad \text{or} \tag{28}$$

$$1 - F_{Y^-}(y) \ge 1 - F^*_{U^-}\left(u^-\right) - D^-_{n,\alpha}. \tag{29}$$

Let $F^*_{U^+}(u^+) \ge 1 - F^*_{U^-}(u^-)$. Then, from Eqs. (27) and (28), we can find the value of α_0 such that these two boundaries are met as follows:

$$F^*_{U^+}\left(u^+\right) - D^+_{n,\alpha_0} = 1 - F^*_{U^-}\left(u^-\right) + D^-_{n,\alpha_0} = x_0. \tag{30}$$

With the above condition,

$$F_{Y^+}(y) \ge x_0 \ge 1 - F_{Y^-}(y). \tag{31}$$

This implies that with a probability of $1 - \alpha_0/2$,

$$F_{Y^+}(y) \ge 1 - F_{Y^-}(y). \tag{32}$$

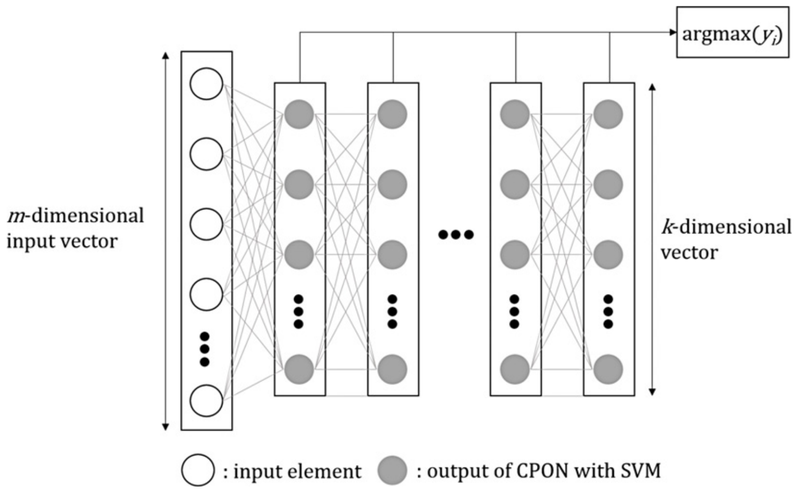

Fig. 1. Structure of the proposed model.

Thus, with a probability of $1-\alpha_0/2$, the true CDF for the positive class is greater than or equal to the true CDF for the negative class. In the case of $1-F^*_{U^-}\left(u^-\right) \geq F^*_{U^+}\left(u^+\right)$, the same argument can be applied; Therefore, from Eqs. (26) and (29),

$$F^*_{U^+}\left(u^+\right)+D^+_{n,\alpha_0}=1-F^*_{U^-}\left(u^-\right)-D^-_{n,\alpha_0}=x_0. \tag{33}$$

Then, with a probability of $1-\alpha_0/2$,

$$F_{Y^+}(y) \leq 1-F_{Y^-}(y)\,. \tag{34}$$

From this description of confidence level $1-\alpha_0/2$, the uncertainty measure $\alpha_0/2$ is determined as follows:

Algorithm 1. Determining the uncertainty measure

Step 1. From the output of CPON for the positive and negative classes, determine the empirical CDFs. After the CPON is trained sufficiently for the given patterns,

$$F^*_{U^+}\left(u^+\right) \approx \hat{F}_{Y^+}(y) \quad \text{and} \tag{35}$$

$$1-F^*_{U^-}\left(u^-\right) \approx 1-\hat{F}_{Y^-}(y)\,. \tag{36}$$

Step 2. From the boundary condition x_0, determine the value of K_{α_0}:

- If $F^*_{U^+}\left(u^+\right) \geq 1-F^*_{U^-}\left(u^-\right)$,

$$K_{\alpha_0}=\frac{F^*_{U^+}\left(u^+\right)+F^*_{U^-}\left(u^-\right)-1}{1/\sqrt{n^+}+1/\sqrt{n^-}}. \tag{37}$$

- Otherwise,

$$K_{\alpha_0}=\frac{1-F^*_{U^+}\left(u^+\right)-F^*_{U^-}\left(u^-\right)}{1/\sqrt{n^+}+1/\sqrt{n^-}}. \tag{38}$$

Step 3. From the value of K_{α_0}, determine the uncertainty value $\alpha_0/2$:,

$$\alpha_0=1-H(K_{\alpha_0}). \tag{39}$$

This uncertainty measure $\alpha_0/2$ represents how well the CDF values of the positive and negative classes are separated. For example, if the value of α_0 is 0.1, it is implied that with a probability of $1-0.1/2(=0.95)$, one CDF value is greater than or equal to the other CDF value. Therefore, we may think that classification decision is reasonably certain. That is, for the given uncertainty measure α_0, the value of $1-\alpha_0/2$ represents the probability of achieving the ideal Bayes decision of pattern classification problems under the assumption that both + and − patterns appear equally.

3. Proposed model

For pattern classification problems, a layered structure where each layer consists of SVMs connected with CPONs is proposed. The overall structure is shown in Fig. 1. In this structure, one-against-the-rest models (Weston & Watkins, 1999) are used to perform the K-class classification task; that is, K one-against-the-rest classification models (implemented by K SVMs with CPONs) are used for the classification decision. Then, the decision is made by selecting the label with the maximum output. In the proposed method, that would be the maximum conditional probability estimate of Eq. (20) for the given pattern. The dimension of the input layer is the same as that of the input pattern and the K probability estimates are obtained from the K one-against-the-rest classifiers. From the second layer, the input is given by the K probability estimates and the output is determined from by the K probability estimates for the given input. The detailed structure of each layer is presented in Fig. 2. At each layer, the CPONs provide the measure of uncertainty α_0 of Eq. (39) for the given pattern. If the given pattern is regarded as uncertain (for example, $\alpha_0 > 0.1$) by the uncertainty checker, that pattern is projected onto the feature space that is constructed by CPON outputs and used as the input pattern for the next layer of SVMs. This projection is propagated to the deep layers until the pattern meets the required certainty of the classification results or reaches the top layer. If the classification for the given pattern is regarded as certain, the classification is performed in that layer.

In this model, the CPONs are used to select the parameters of the classifiers according to a test on the empirical distribution and to determine the depth of the network by verifying the values of the uncertainty measures. To confirm the validity of the SVM model with specific parameters, p-values of testing the output distributions in the CPONs are used. Because the CPON is capable of modeling the probabilistic output distribution of the discriminative classifier, it can evaluate the goodness of SVM parameters (such as kernel width of radial basis function) by performing hypothesis testing of the output distribution to fit the ideal beta distribution (Park & Kil, 2009). The parameter selection of the SVMs is described in the following algorithm:

Algorithm 2. Parameter Selection of the SVMs

Step 1. Create a pool of parameters for the SVMs
Step 2. For every set of parameters in the pool, repeat the following steps:

- Train the SVM for the given training data using the set of parameters.

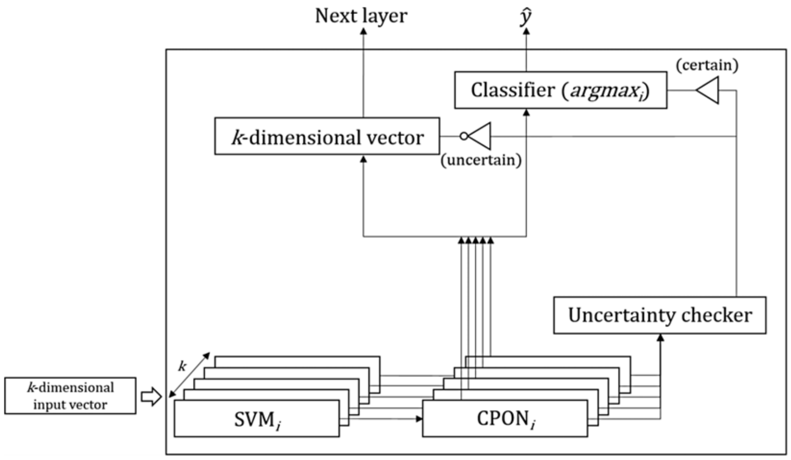

Fig. 2. Processing in each layer in the proposed model.

- For the trained SVM, construct the corresponding CPON.
- For each output distribution of positive or negative class of the SVM, calculate the corresponding p-value of the K–S test for the goodness of fit to the beta distribution. Then, p-values p^+ and p^- are evaluated for the positive and negative classes, respectively.

Step 3. Select the SVM with CPON with the maximum product value of $(p^+ \cdot p^-)$ among the trained SVMs with CPONs.

The parameter of SVM with RBF kernel is the width of the kernel, σ, and the RBF kernel function K is represented as

$$K(x, x_i) = e^{\frac{-\|\mathbf{x}-\mathbf{x}_i\|}{\sigma}}, \qquad (40)$$

where $\mathbf{x}_i$ is ith support vector of the machine.

For the detailed procedure of calculating the p-values of the K–S test, refer to Park and Kil (2009). Finally, the CPONs are used to determine whether an additional layer is required by verifying the values of the uncertainty measures that are calculated by Algorithm 1. The construction of deep layer SVMs with CPONs are described in the following algorithm:

Algorithm 3. Construction of deep layer SVMs with CPONs

Step 1. Let $\mathbf{x}_{ij}^l$ and y_{ij}^l be the jth input and output patterns to the SVM for the ith class of the lth layer, respectively.

Step 2. For a K-class classification problem, K SVMs are constructed using their p-values from the K–S test for beta distributions. For each class, train the multiple SVMs with the different parameters and select the trained SVM with the maximum p-value using Algorithm 2. After this step, K selected SVMs are set as components of the lth layer.

Step 3. For each input pattern $\mathbf{x}_{ij}^l$, a series of K SVM output y_{ij}^l, $i = 1, \ldots, K$ is obtained.

Step 4. Calculate the values of the uncertainty measures $\alpha(y_{ij}^l)$ for each output of the SVM using the Algorithm 1. Then, determine the uncertain interval $[y_{i-}^l, y_{i+}^l]$ for $\alpha(y_{ij}^l) > \theta$. An example of a θ value is 0.05. If the output patterns are lying in the uncertain interval, these patterns will be used to train the SVMs with CPONs of the next layer, where the inputs for the next SVM layer consist of the outputs of the previous layer of CPONs with SVM as shown in Fig. 2. Deeper layers use randomly selected 10% certain data obtained from the previous layer of SVM training together with uncertain and inaccurate data. It helps generalization performance in the learning of next layer.

Step 5. In each layer, determine the sample means of uncertainty measures; they are

$$E_i^l = \frac{1}{n_i^l} \sum_{j=1}^{n_i^l} \alpha(y_{ij}^l), \quad i = 1, \ldots, K \qquad (41)$$

where n_i^l represents the number of patterns for the ith class selected at the lth layer.

Step 6. If $E_i^l < \bar{\theta}$, where an example value of $\bar{\theta}$ is 0.9, the deep learning process will be ended in the current layer, else if check whether the number of training data for the next layer is enough or not, based on the theoretical criteria given by the PAC learning framework in Eq. (45). If the number of training data satisfies the PAC learning criterion, go to step 1 with $l \leftarrow l + 1$. Otherwise, terminate the deep learning process.

In this algorithm, the decision for stacking layers is made by verifying the values of the uncertainty measures. Then, in principle, the average probability of achieving a Bayes decision for the ith classifier at the first layer is approximately $1 - E_i^l/2$ as E_i^l is an unbiased estimate of the average value of $\alpha(y_{ij}^l)$. If the proportion of the number of patterns lying in the uncertain interval at the first layer is given by δ_1, the mean of uncertainty measures is bounded as follows:

$$E_i^l \le \theta(1 - \delta_1) + \delta_1 = \delta_1(1 - \theta) + \theta. \qquad (42)$$

At the second layer, if the proportion of the number of patterns lying in the uncertain interval is given by δ_2, the total mean of uncertainty measure T_i^2 is bounded by

$$T_i^2 \le \theta(1 - \delta_1) + \delta_1(\theta(1 - \delta_2) + \delta_2) = \delta_1\delta_2(1 - \theta) + \theta. \qquad (43)$$

Note that $T_i^1 = E_i^1$. The total mean of uncertainty measure of the ith classifier at the lth layer is bounded by:

$$T_i^l \le \prod_{j=1}^{l} \delta_j (1 - \theta) + \theta. \qquad (44)$$

That is, T_i^l is approximately bounded by θ for a sufficiently large number of layers as the value of δ_j is usually significantly less than 1. This implies that the proposed network is achieving the approximate ideal Bayes decision for a sufficiently large number of layers since Eq. (32) or Eq. (34) holds with a probability of $1 - T_i^l/2$.

Fig. 3 shows the uncertainty measures using the Diabetes dataset (Bache & Lichman, 2013) as an example case. The dataset is described in more detail in the experimental section. From the

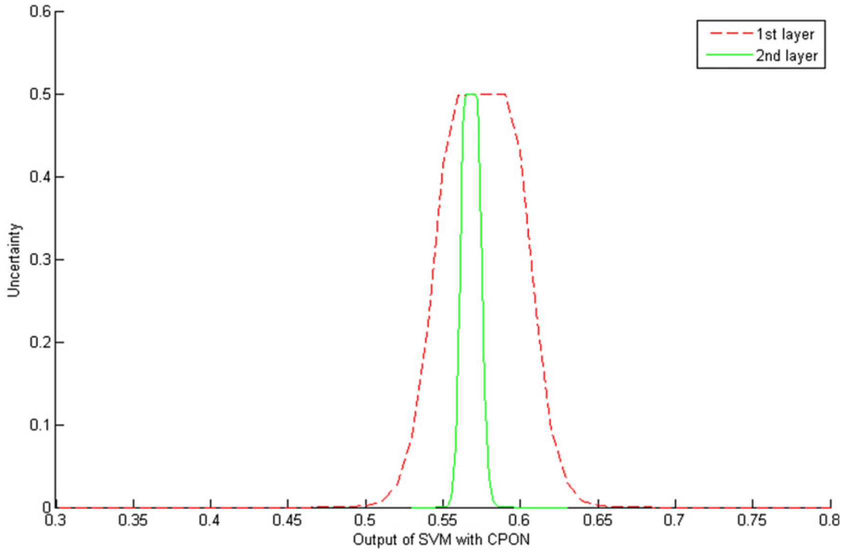

Fig. 3. Uncertainty measures on diabetes dataset.

first SVM layer with CPON, the interval of uncertain data can be represented by a Gaussian-like curve of which the mean is approximately 0.57. The second SVM layer with CPON attempts to focus on the uncertain data located in the interval of the first layer SVM output. This intensively analyzes the uncertain data only after processing the first layer SVM ranged between 0.54 and 0.62. This shows deeper layers process the uncertain data that are identified by the previous layers.

In the proposed model, one difficulty of stacking layers is the smaller number of propagated patterns as the number of layers increases. One method of addressing with this problem is decreasing the threshold value θ. Then, the number of patterns propagating to the upper layers increases. The usual value θ is between 0.01 and 0.1. However, at the top layer, the number of patterns may be overly small to train the network. One guideline for this learning can be obtained from the concept of probably approximately correct (PAC) learning. In PAC learning, the lower bound n_- of the sample complexity for binary classification problems (Anthony, 1997) is determined by:

$$n_- = \frac{1-\epsilon}{\epsilon}\log\frac{1}{\eta}, \tag{45}$$

where ϵ represents the probability of error and η represents the term related to the confidence level $1-\eta$. For example, if $\epsilon = 0.1$ and $\eta = 0.05$, the lower bound of the sample complexity is determined by $n_- = 27$. This implies that PAC learning is not possible if the number of patterns is less than n_-. Thus, if the number of patterns is less than n_-, additional deep layer learning is not recommended

Upon construction of the SVMs with CPONs, the final classification for the given pattern is determined by the following algorithm:

Algorithm 4. Pattern Classification

```
Step 1. Set $l = 1$.
Step 2. For the given test pattern $\mathbf{x}^l$, a series of $K$ SVM output
        $y_i^l$, $i = 1, \ldots, K$ is obtained at the $l$th layer.
Step 3. For the test data, if all of $K$ SVM output values are lying
        in the uncertain intervals and $l$ is not the top layer, set
        $\mathbf{x}^{l+1} = [y_1^l, \ldots, y_K^l]$ and go to Step 2. Otherwise, make a
        decision of class label $\hat{t}$ as:
        $\hat{t} = \arg\max_i y_i^l$.    (46)
```

In this algorithm, the layer that defines the class label for the given pattern is determined by verifying the uncertain intervals of the classifiers at the same layer. If a classifier output exists that is not determined to be an uncertain pattern at some layer level, the decision is made at that layer.

As described previously, the uncertainty measure of CPON is not only for model selection in the training phase, but also for depth selection in both training and test phases. Cross-validation techniques make it possible to select a model that has a low generalization error on the validation set in the training phase. However, for the test data, cross-validation-like techniques cannot provide the certainty of outputs nor difficulty of the sample. This is because we cannot define an accuracy measure on data, especially in the feature extraction stage. Moreover, we cannot make an inference whether the classification is convinced. Conventional cross-validation techniques are not particularly suitable for deep learning. The use of CPON enables the model to utilize variable depths rather than a fixed depth of abstraction.

In Figs. 4 and 5, the scatterplots for the output values of the second and the third layers are shown. Figs. 4 and 5 were trained and tested with the Wisconsin Breast Cancer and PIMA Indians Diabetes datasets, respectively. In these scatterplots, horizontal and vertical axes represent the output of binary classifiers Y1 and Y2. Red and blue circles represent outputs of the class 1 and 2, respectively. As shown in these figures, the deeper layers exhibits better distinguishable distribution of new feature space for both examples.

4. Experimental results

For the simulation of the proposed model, the Wisconsin Breast Cancer dataset, PIMA Indians Diabetes dataset, BUPA Liver Disorder dataset, and Ionosphere dataset from the UCI Machine Learning Repository (Bache & Lichman, 2013) and the MNIST dataset (LeCun, Bottou, Bengio, & Haffner, 1998) were tested. The Wisconsin Breast Cancer dataset consists of 699 samples of two classes and has 443 and 238 samples for class 1 and 2, respectively. There are ten attributes as illustrated in Table 1. For the classification simulation, the 1st attribute, "sample code number" is excluded, i.e., only the last nine attributes are used. The Diabetes dataset consists of 768 samples of two classes, has 500 and 268 samples for each class; there are eight attributes. The BUPA Liver Disorder dataset consists of 345 samples of two classes, 145 and 200 samples, and six-dimensional input data. The Ionosphere dataset consists of two

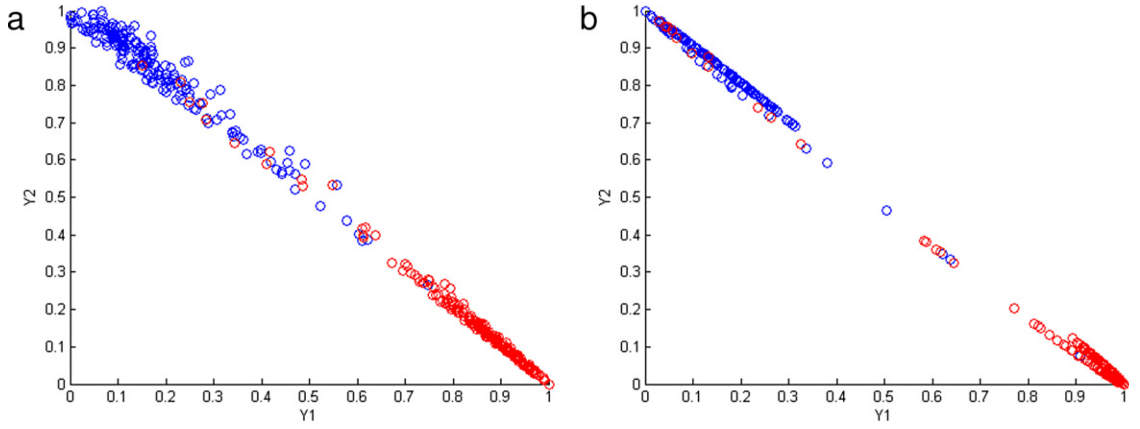

Fig. 4. Plot of the output values of the proposed model on the Wisconsin Breast Cancer dataset.

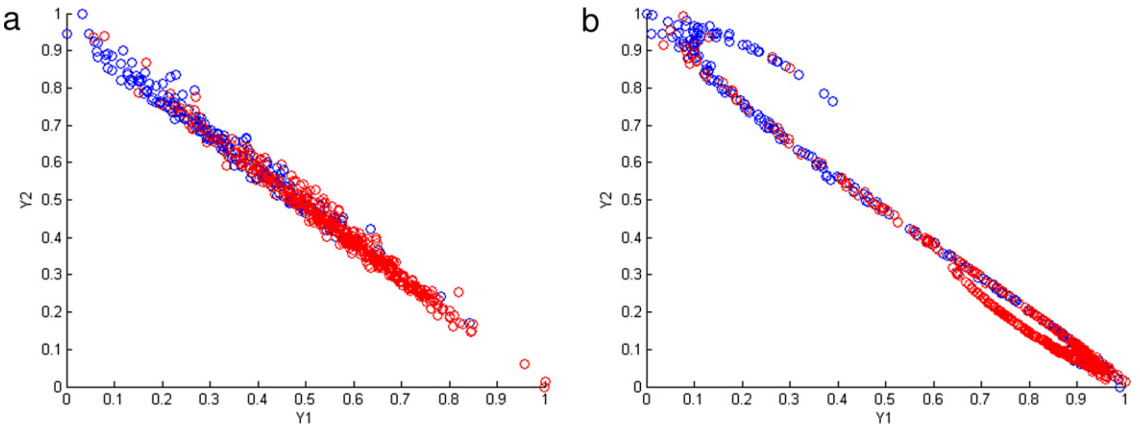

Fig. 5. Plot of the output values of the proposed model on the PIMA Indians Diabetes dataset.

Table 1
Description of the datasets used in experiments.

Dataset name	Size of dataset	Input dimension	Number of classes
Wisconsin breast cancer	699	10	2
PIMA Indians diabetes	768	8	2
Liver disorders	345	6	2
Ionosphere	351	34	2
MNIST	70,000	784	10

classes, 225 and 126 samples. The data dimension is 34. We divided each dataset into 90% for training and 10% for test data for the UCI datasets. Finally the MNIST dataset of handwritten digits has ten classes from digit "0" to "9" and approximately 6000/1000 samples for training/test per class. Therefore, a total of 60,000 and 10,000 samples are used to train and test models, respectively. Each image sample has 28 × 28 dimensions. The description of the datasets is summarized in Table 1.

For the SVM implementation, LIBSVM was used (Chang & Lin, 2011). The pool of RBF kernel widths were constructed as:

$$\sigma_{RBF} = \sigma_{data} \times 2^{-6+\frac{i}{10}}, \quad i = 1, \dots, 50, \tag{47}$$

where σ_{data} is the standard deviation of the data and is approximated as the average of standard deviation of each dimension. The coefficient C for the regularization term in Eq. (9) was tested on $\{2^{-8}, 2^{-6}, \dots, 2^{6}, 2^{8}\}$ and θ, which is the threshold for uncertainty level is set to 0.1.

The ratio of the certain data, uncertainty measures, and accuracies on the certain and overall data were evaluated in each layer. These are presented in Table 2.

In Table 2, the total value of uncertainty of lth layer, T^l simply accumulates uncertainty values of T_i^l for all classes, as:

$$T^l = \sum_i^K T_i^l, \tag{48}$$

where K is the number of classes. Notice that in Table 2, uncertainties on the data decrease as the model advanced deeper. As illustrated in the table, the ratios of certain data increased as the depth of the layer increased. Because certain data of deeper layers contained more difficult samples, the accuracy on the certain data may decrease in each deeper layer. However, the number of certain data increased and the recognition accuracies were safely maintained without an overfitting phenomena. For the Wisconsin Breast Cancer dataset, 4 layers are stacked for the training data, but in test phase, the 3rd and 4th layers fail to guess the correct label of hard uncertain examples and the overall accuracy did not grow from 98.55%.

To confirm the efficiency of the proposed model compared to Naïve cross-validation techniques, a baseline model was constructed as follows:

Algorithm 5. Training of baseline model with cross-validation

```
Step 1. Build and stack SVM layers with cross-validation using the
        classification accuracy.
Step 2. Feed-forward using the outputs of the SVMs
Step 3. If classification accuracy does not increase, abort the
        algorithm. Otherwise, make the new dataset with feed-
        forwarded features and go to Step 1.
```

Table 2
Ratio of certain data, uncertainty measures and accuracies on test dataset.

	The number of layer	1	2	3	4
Wisconsin breast cancer	The ratio of certain data (%)	94.21	98.55	98.55	98.55
	Total value of uncertainty	0.291	0.208	0.201	0.200
	Accuracy on certain data (%)	100	100	0	0
	Accuracy on overall data (%)	94.20	98.55	98.55	98.55
Diabetes	The ratio of certain data (%)	79.22	94.81	97.40	–
	Total value of uncertainty	0.503	0.293	0.243	–
	Accuracy on certain data (%)	83.61	75	100	–
	Accuracy on overall data (%)	66.23	77.92	83.11	–
Liver disorder	The ratio of certain data (%)	60	74.29	–	–
	Total value of uncertainty	0.663	0.414	–	–
	Accuracy on certain data (%)	76.19	60.0	–	–
	Accuracy on overall data (%)	45.71	77.14	–	–
Ionosphere	The ratio of certain data (%)	83.33	86.11	88.89	
	Total value of uncertainty	0.475	0.276	0.232	–
	Accuracy on certain data (%)	100	0	100	–
	Accuracy on overall data (%)	83.33	83.33	97.22	-
MNIST	The ratio of certain data (%)	53.55	79.21	84.93	94.70
	Total value of uncertainty	1.66	1.32	1.22	1.10
	Accuracy on certain data (%)	92.95	94.97	90.27	90.92
	Accuracy on overall data (%)	96.80	98.60	98.82	98.84

Table 3
Comparison of Naïve baseline, Platt's scaling, and the proposed model.

Dataset name	Naïve baseline	Platt's scaling	The proposed model
Wisconsin breast cancer	98.55	96.67	98.55
PIMA Indians diabetes	80.51	79.22	83.11
BUPA Liver disorders	57.14	54.13	77.14
Ionosphere	88.89	84.15	97.22
MNIST	93.26	94.43	94.93

Table 3 displays the comparison results among the proposed model, Platt's scaling, and Naïve baseline model. For the relatively easy datasets such as Wisconsin Breast Cancer, the proposed method exhibits the same accuracy. However, for the other datasets, the proposed model delivered superior accuracies than both the Naïve baseline approach and Platt's scaling (Platt, 1999). The Platt's scaling method shows the lowest accuracy on all the datasets except the MNIST recognition.

Note that in the Table 3, for the MNIST dataset, training is not done for entire 60,000 samples. Since the over fitting problem is usually severe with the limited number of training dataset, we try to show the generalization performance of the proposed method compared to other methods using only randomly selected 6000 samples.

The test accuracy of several models on the datasets are given in Table 4. For comparison, Naïve Bayes classifier (Metsis, Androutsopoulos, & Paliouras, 2006), single-layer SVM, single-layer SVM with CPON, deep belief networks (DBN) (Lee, Ekanadham, & Ng, 2008; Le Roux & Bengio, 2008), and the proposed model were evaluated. For the MNIST dataset, the training of Naïve Bayes classifier was not successful because the model requires the positive variance for each dimension but there are many background pixels which are constant values as zero, i.e., zero variances. For the DBN, we used the indicator target vectors to encode class categories. For example, categories 1, 2, and 3 were represented as vectors [1 0 0], [0 1 0] and [0 0 1], respectively. And the real-valued inputs are normalized with the range from 0 to 1, element-wise. The number of hidden layers of DBN was set to three. The number of hidden nodes were selected from the pool using cross-validation and as a result, they are set to four, three, and two for the Breast Cancer, Diabetes, and Liver Disorder datasets. For the Ionosphere dataset which has 34 dimensionality, the hidden nodes were set to fifteen, seven and five. For the MNIST dataset, the hidden nodes were set to 300, 100 and 50.

These accuracies were calculated on the entire test datasets. As described in the table, the proposed model showed the best accuracy on test datasets except for the Ionosphere. For the Ionosphere dataset, DBN shows the best performance; the proposed model followed. But for the relatively small datasets with low dimensionality, DBN was not able to achieve the comparable results to the best accuracy. Single-layer SVM with CPON shows better performance than single-layer SVM without CPON.

5. Concluding remarks

In this paper, a deep network structure using SVMs with CPONs was proposed to provide adequate depth of the structure and robust classification accuracy. In the proposed SVMs with CPONs, the classifiers were optimized using the SRM principle. The conditional class probability for the given pattern was estimated using the parameters of the beta distribution. Furthermore, the uncertainty measure was defined using the confidence level for the probability estimate and applied to determine the depth of the deep network structure. The proposed model of layered structure closely approached to the ideal Bayes classifier as the number of layers increases. Through the classification simulation, the effectiveness of the proposed method was demonstrated. Because the proposed model is a general classification model of

Table 4
Comparison of test accuracies on the datasets.

Dataset	Breast cancer	Diabetes	BUPA liver disorder	Ionosphere	MNIST
Naïve Bayes	96.5	76.62	57.14	77.78	Not available
Single-layer SVM	96.54	74.03	69.6	88.89	95.3
Single-layer SVM with CPON	96.85	77.92	75.26	94.44	96.82
Deep belief networks	98.42	70.7	58.77	**98.45**	96.54
Proposed model	**98.55**	**83.11**	**77.14**	97.22	**98.84**

achieving the Bayes classifier, this model can be effectively applied to classification problems with a variety of data distributions to obtain robust and high classification performances. In future work, we are considering an incremental deep learning based on the proposed model for solving real-world problems. Furthermore, dynamic models based on the proposed deep structure are under investigation.

Acknowledgments

This work was partly supported by the ICT R&D program of MSIP/IITP. [10041826, Development of emotional features sensing, diagnostics and distribution s/w platform for measurement of multiple intelligence from young children] and by the Industrial Strategic Technology Development Program [10044009] funded by the Ministry of Knowledge Economy (MKE, Korea) (50%).

References

Anthony, M. (1997). *Computational learning theory*. Cambridge University Press.

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, *5*, 13–18.

Bache, K., & Lichman, M. (2013). UCI machine learning repository. URL http://archive.ics.uci.edu/ml.

Ben-Tal, A., & Zowe, J. (1982). A unified theory of first and second order conditions for extremum problems in topological vector spaces. In *Optimality and stability in mathematical programming* (pp. 39–76). Springer.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, *2*, 1–127.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*, 27.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, *121*, 256–285.

Fukushima, K. (1980). Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*, 193–202.

Guignard, M. (1969). Generalized Kuhn–Tucker conditions for mathematical programming problems in a Banach space. *SIAM Journal on Control*, *7*, 232–241.

Hastie, T., Tibshirani, R., Jordan, M., Kearns, M., & Solla, S. (1998). Classification by pairwise coupling. *Advances in Neural Information Processing Systems*.

Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Momentum*, *9*, 926.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Kim, S., Kavuri, S., & Lee, M. (2013). Deep network with support vector machines. In *Neural information processing* (pp. 458–465). Springer.

Kim, H.-G., Kil, R. M., & Lee, S.-Y. (2011). Uncertainty measure for selective sampling based on class probability output networks. In *Neural information processing* (pp. 774–781). Springer.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*, 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area V2. *Advances in Neural Information Processing Systems*, *20*, 873–880.

Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, *20*, 1631–1649.

Luenberger, D. G. (1968). *Optimization by vector space methods*. John Wiley and Sons.

Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). Spam filtering with naive bayes-which naive Bayes? In *CEAS* (pp. 27–28).

Park, W. J., & Kil, R. M. (2009). Pattern classification with class probability output network. *IEEE Transactions on Neural Networks*, *20*, 1659–1673.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, *10*, 61–74.

Rohatgi, V. K., Saleh, A., & Ehsanes, M. (2001). Nonparametric statistical inference. In *An introduction to probability and statistics* (2nd ed.) (pp. 598–662).

Rosas, H., Kil, R. M., & Han, S. (2010). Automatic media data rating based on class probability output networks. *IEEE Transactions on Consumer Electronics*, *56*, 2296–2302.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification* (pp. 149–171). Springer.

Schapire, R. E., & Freund, Y. (2012). *Boosting: foundations and algorithms*. MIT Press.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 1651–1686.

Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory.

Tang, Y. (2013). Deep learning using linear support vector machines. In *Workshop on challenges in representation learning*, ICML.

Tesauro, G. (1992). Practical issues in temporal difference learning. In *Reinforcement learning* (pp. 33–53). Springer.

Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, *10*, 271–289.

Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation*, *14*, 2497–2529.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *ESANN*, Vol. 99 (pp. 61-72).

Wiering, M., Schutten, M., Millea, A., Meijster, A., & Schomaker, L. (2013). Deep support vector machines for regression problems.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*, 241–259.