Robustness of Selective Desensitization Perceptron Against Irrelevant and Partially Relevant Features in Pattern Classification

Tomohiro Tanno, Kazumasa Horie, Jun Izawa, and Masahiko Morita

University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, Japan
tanno@bcl.esys.tsukuba.ac.jp, horie@bipl-sdnn.org,
izawa@emp.tsukuba.ac.jp, mor@bcl.esys.tsukuba.ac.jp

Abstract. Recent practical studies have shown that a selective desensitization neural network (SDNN) is a high-performance function approximator that is robust against redundant input dimensions. This paper examined the classification performance of a single-output-SDNN, which we refer to as a selective desensitization perceptron (SDP), through a numerical experiment on binary classification problems that include some irrelevant features and partially relevant features and compared these results with multilayer perceptron (MLP) and support vector machine (SVM) classification methods. The results show that SDP was highly effective not only in dealing with irrelevant features but also in a dataset including a partially relevant feature, which is irrelevant in most of the domain but affects the output in a specific domain. These results indicate that the previously observed SDNN's high-performance in the practical problems might be originated from the fact that SDP does not require a precise feature selection with taking account of the various degrees of feature relevance.

Keywords: Binary Classification, Selective Desensitization Perceptron, Irrelevant Feature, Partially Relevant Feature.

1 Introduction

When one uses a classifier for solving a practical problem with a multidimensional dataset, it is typically unknown whether the dataset includes redundant dimensions and which dimensions are actually redundant for this classification problem. Therefore, the user of a classifier often uses all features available even though many irrelevant features may be included. However, irrelevant features are known to have harmful effects on learning processes, such as increases in learning time and degradation of classification accuracy. For example, although support vector machines (SVMs) with radial basis function (RBF) kernel are known to exhibit excellent performance with binary classification problems, their accuracy severely decreases when the dataset includes irrelevant features [1, 2]. To deal with this, multiple methods of feature selection have been proposed in

order to improve the performance via removal of irrelevant features [2, 3]. However, these methods often require prior and technical knowledge, time, and effort. Besides, the accuracy may not be improved by feature selection when some features seem to be irrelevant but are actually relevant in a specific condition. For example, the features may be irrelevant in most of the domain but relevant in a certain range of their (or other feature's) respective domains. We refer to these features as "partially relevant features" here. These features may degrade the classification accuracy like irrelevant features, but they should not be completely removed because they may be critical to solve the pattern in a certain domain.

For example, a multilayer perceptron (MLP) is a layered neural network known to be relatively robust to irrelevant features, i.e., the degradation of the classification accuracy is small. This is because its structure enables it to ignore their effects by decaying the synaptic weight of the input [4]. However, because it completely ignores the feature, it is predicted that the MLP cannot consider the partial relevance of the feature.

In contrast, a selective desensitization neural network (SDNN) is another layered neural network that exhibits some superiority as a function approximator. A study on continuous-state reinforcement learning showed that SDNN performed well when approximating a value function that included some redundant dimensions [5]. Furthermore, Nonaka et al. showed that SDNN could accurately approximate a complicated function that outputs a constant value in a specific domain; this can be considered as the case that the features are partially irrelevant to the output [6]. Although the SDNN is a function approximator, the output layer of SDNN is comprised of a set of simple perceptrons, each of which can be regarded as a binary classifier. Considering these properties of SDNN combined with its previously reported robustness, we hypothesized that a single-output-SDNN, which we refer to as a selective desensitization perceptron (SDP), is advantageous in binary classification for dealing with data that includes irrelevant and partially relevant features.

To test this idea, we systematically examined the robustness of SDP against irrelevant and partially relevant features. We conducted a numerical experiment on binary classification problems that included some irrelevant features or partially relevant features, and then compared the performances of SDP, MLP, and RBF kernel SVM.

2 Selective Desensitization Perceptron

SDP is a single-output-SDNN that can be applied as a binary classifier. It is constructed by applying manipulations of pattern coding and selective desensitization to a simple perceptron. The structure of an SDP is shown in Fig.1. Pattern coding and selective desensitization are performed from the first to the second layer and the second to the third layer, respectively. The third to the fourth layer comprises a simple perceptron for binary classification, which is the only part that SDP trains.

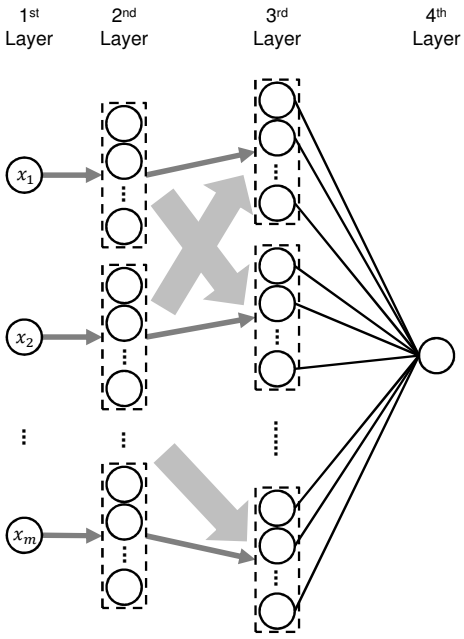

Fig. 1. Structure of an SDP with m dimensional input.

2.1 Pattern Coding

Pattern coding converts each analog input value into high-dimensional binary vectors known as code patterns. We specifically adopt the following procedures for all the input variables using different random sequences.

(1) After we quantized the analog input value x into q bins, we assigned an n-dimensional vector whose elements take only $+1$ or -1 to each bin. (2) We set P_1, the pattern vector of the first bin, by selecting $+1$ and -1 randomly for each element so that half of the elements take positive values and the rest take negative values. (3) For P_2, the pattern after the first bin (that is, P_1), we randomly selected r elements from those with positive value and the other r elements from those with negative value and then flipped the signs of these $2r$ elements. (4) Subsequently, we repeated this process for P_k based on the pattern of $P_{(k-1)}$ until we completed the same for the last bin.

This method results in a coded pattern that gradually changes from the first bin to the last bin, while ensuring that the correlation between two consecutive patterns is high and that the correlation between two patterns that are apart is near zero. If the code patterns fulfill these conditions, the values of the parameters, n, q, and r, do not affect the performance of SDNN (and SDP) significantly. A large n and q are always preferable if we disregard computational costs [6].

2.2 Selective Desensitization

Selective desensitization integrates two binary code patterns into one ternary $(-1, 0, +1)$ pattern by modifying one with the other. When there are two code patterns, $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$, x_i is desensitized if $y_i = -1$; that is, X is modified with Y into

$$X(Y) = (\frac{1+y_1}{2}x_1, ..., \frac{1+y_n}{2}x_n). \tag{1}$$

This makes $X(Y)$ a ternary pattern whose half of the elements are zero and the other half are the same as X. Note that different code patterns should be used for respective input variables to avoid undesirable bias in the modification (for example, $X(Y)$ never contains an element -1 if $X = Y$). Likewise, $Y(X)$, or Y modified with X, can be considered, and we use both $X(Y)$ and $Y(X)$ as the input pattern to the perceptron (Fig.1).

In general, if the input dimension m is greater than two, selective desensitization is conducted for all combinations of modified and modifier variables, so that $m(m-1)$ ternary patterns are created in the third layer (Fig.1).

2.3 Simple Perceptron

The output unit is a threshold element receiving signals from the third layer and emits 0 or 1, which works as a simple perceptron. In mathematical terms,

$$z = f(\sum_{i=1}^{s} w_i p_i + h), \tag{2}$$

where s is the input size (number of elements in the third layer); p_i and w_i are the i-th input signal (output of the third layer) and its weight, respectively; h is the threshold input; and $f(u)$ denotes the Heaviside step function taking 1 for $u > 0$ and 0 otherwise. Only this part is trained, with an error-correcting rule generally used for simple perceptrons. Specifically, the input weights and the threshold are updated as

$$w_i(t+1) = w_i(t) + (d - z)p_i, \tag{3}$$

$$h(t+1) = h(t) + (d - z), \tag{4}$$

where d is the target signal (0 or 1).

3 Numerical Experiment

We conducted an experiment on binary classification problems that included one or more irrelevant features and/or partially relevant features and compared the performance of SDP, MLP, and SVM. Here, an irrelevant feature is an element in the input vector that does not affect the output at all. A partially relevant

feature, on the other hand, does not affect the output in most of its domain, but affects the output in a specific domain.

A dataset was generated using a binary (black and white) image of size 101×101 (10201 pixels in total) which represents the correct classification boundary (Fig.2). For each pixel, the horizontal-vertical coordinates (normalized to $[0, 1]$) were used as the first and second components of the feature vector, and the luminance (white is 1 and black is 0) was used as the class number. Those two features were the relevant features that determined the output class. One or more random values of range $[0, 1]$ were included in each feature vector as the third or later components, which were the irrelevant features.

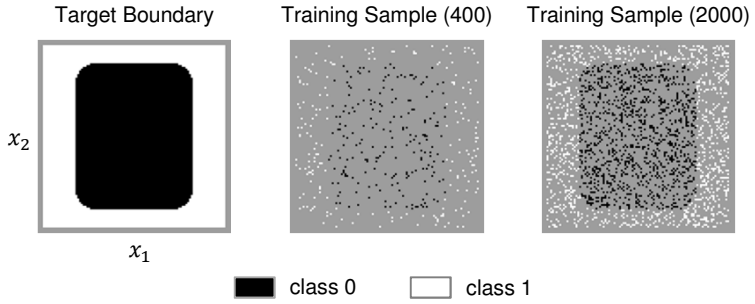

Fig. 2. The target class boundary and examples of the training sample set. The (x_1, x_2) coordinates were the two feature values that determined the output class value shown by the luminance.

The experiment was conducted as per the following procedures:

1. A certain number (k) of pixels on a binary image was pseudo-randomly selected with no overlap. The training sample set was created using these pixel points.
2. Each classifier learned the training samples.
3. Each classifier made predictions of the output class of 50000 unlearned points. Here, the rate of misclassified points was evaluated as the generalization error.
4. The above steps were repeated for 10 trials with different pseudo-random sequences.

The training parameters of classifiers are described below:

SDP: The pattern coding parameters were $n = 5000$, $q = 101$, and $r = 100$. The model was trained until every training sample was correctly classified.

MLP: One hidden layer of 100 units was applied between the input and the output layer. For hidden and output layers, hyperbolic tangent ($tanh$) was used

as the activation function. Training was done with a standard backpropagation algorithm until the mean squared error of training reached below 0.005.

SVM: The RBF kernel was used. Kernel parameters were searched for the values of $C = 2^{-5}, 2^{-3}, ..., 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$ for every learning trial by five-fold cross-validation and grid search. LIBSVM was used [1].

3.1 Results

First, we examined the influence of irrelevant features. Fig.3 shows the mean generalization error of each classifier against the number of irrelevant features included in the input. The number of training samples was 400 for (a) and 2000 for (b). The error of SVM increased considerably as the number of irrelevant features increased, while the error of SDP and MLP did not.

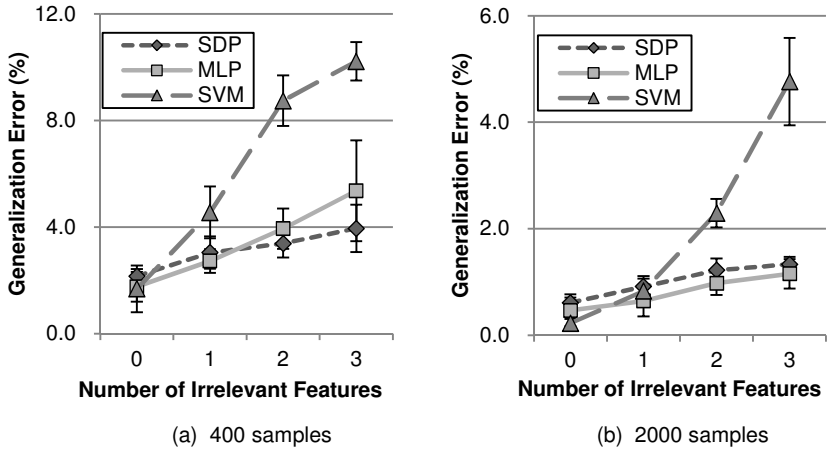

Fig. 3. Mean generalization error against the number of irrelevant features. The number of samples was 400 in (a) and 2000 in (b). Error bars indicate standard deviation.

Then, we introduced a partially relevant feature. Here, the third feature x_3 was the partially relevant feature; specifically, the output class was always 0 when x_3 was less than 0.1, and always 1 when x_3 was greater than 0.9. Otherwise the output was determined by the first two relevant features. In other words, x_3 is relevant to the output near the domain of $x_3 = 0.1$ and $x_3 = 0.9$, and irrelevant otherwise.

Fig.4 and Fig.5 show the results of learning data with one partially relevant feature and one irrelevant feature. The number of training samples was 2000. In Fig.4, the mean generalization error was compared with the case of two irrelevant features. The error of MLP and SVM increased substantially as one of

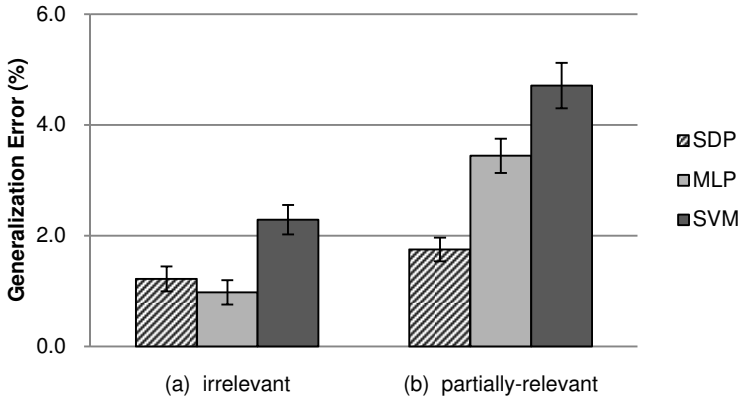

Fig. 4. (a) Mean generalization error for the problem with two irrelevant features. (b) Mean generalization error for the problem with one partially relevant feature and one irrelevant feature. The number of training samples was 2000. Error bars indicate standard deviation.

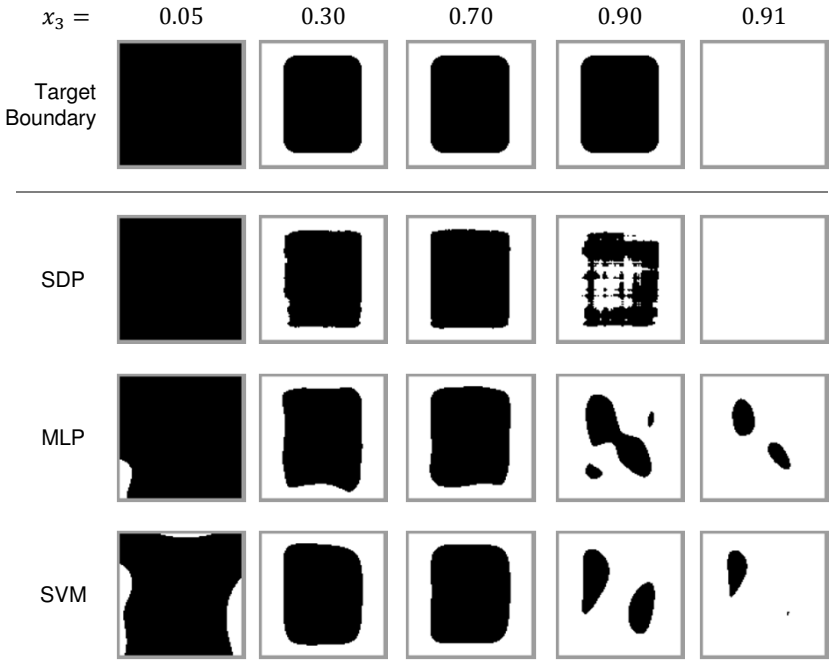

Fig. 5. Cross-sectional views of the class boundary constructed by each classifier, each of which is plotted in the (x_1, x_2) coordinate. The value for x_4 was fixed at 0.50.

the irrelevant features became partially relevant, while the increase was much smaller for SDP. Fig.5 shows some examples of the boundaries that each classifier constructed according to its predictions. It shows that MLP and SVM could not make accurate predictions, especially, when the output should be always 0 or 1 (where $x_3 < 0.1$ or $x_3 > 0.9$). In contrast, SDP showed more accurate predictions for the entire domain of x_3.

4 Discussion

These results show that SDP is a more robust classifier than MLP and SVM for irrelevancy and partial-relevancy in classification problems. Here, we discuss how these three are influenced by irrelevant features, and why SDP is more robust.

SVM with an RBF kernel is known to be susceptible to irrelevant features [2], and this was confirmed in our experiment (Fig.3). This is considered to be because its structure does not allow it to ignore a specific input variable. Because the input space became larger with the addition of irrelevant features, training samples became sparse and the training of SVM was directly affected.

MLP is known to be relatively robust to irrelevant features [4], a trend which was also shown in our experiment (Fig.3). Because MLP determines the output according to the combination of the weighted sum of the input variables, it can ignore a specific variable by setting the synaptic weight for that variable to zero. However, Fig.3 indicates that MLP cannot be trained to ignore the irrelevant features with fewer training samples. Furthermore, because MLP can only ignore the feature completely, the error increased when the feature was partially relevant to the output (Fig.4). Fig.5 shows how the output of MLP was affected by the value of x_3 (partially relevant feature) in the domain where x_3 is irrelevant to the output.

In contrast to SVM and MLP, the influences of irrelevant features and partially relevant features were very small for SDP (Fig.3 and Fig.4). Fig.5 shows that the predictions of SDP were highly accurate in the entire domain of x_3, which indicates that SDP can remove the influence of a specific input variable without completely ignoring it. One possible reason for this is the mapping to the high-dimensional space by manipulations of pattern coding and selective desensitization. Because the weights in SDP are connected from each element of the high-dimensional vectors to the output, SDP can delete the influence of a certain variable by making the weighted sum of the elements zero, which can be achieved without setting the weight itself to zero. For example, consider the case that $P(x_3 = 0.30) = [+1, -1, ...]$ and $P(x_3 = 0.70) = [-1, +1, ...]$, which are the vectors (code patterns) that the values of 0.30 and 0.70 are converted into, and suppose that the third and later elements are all desensitized and that the weights for the first and second elements are equal. Then the weighted sum becomes zero for both $x_3 = 0.30$ and $x_3 = 0.70$, but if $P(x_3 = 0.05) = [+1, +1, ...]$, the weighted sum will not be zero for $x_3 = 0.05$, meaning that the output is affected by x_3 around 0.05 but not around 0.30 and 0.70. This enables SDP to remove the influence of the variable only in a specific domain.

Although SDP converts the inputs to higher dimensional code patterns, the computational cost does not increase much because it does not train hidden layer(s) and also because the input pattern for the simple perceptron is simple containing only $+1$, -1, and 0. Furthermore, the time for training (including parameter setting) of SDP was much smaller than that of MLP and SVM. However, because the computational cost of SDP increases in proportion to the square of the input dimension, SDP is especially suitable for problems with relatively low dimension (less than several dozens).

For these reasons, SDP is a useful and a practical classifier that can be applied when there is not much prior knowledge about a problem, or how each feature is related to the output. This characteristic of SDP is potentially one of the factors that contributed to achieving high performance and practicability in several previous studies using SDNN, such as discriminating multiple hand motions from surface electromyogram signals at high accuracy in real-time without precise adjustment of the number and position of sensors [7, 8].

5 Conclusion

We compared the performance of SDP, MLP, and an RBF kernel SVM on binary classification problems that include some irrelevant features and partially relevant features (features that are irrelevant in most of the domain but affect the output in a specific domain).

The experimental results showed that SVM performed worse as more irrelevant features were included in the input, whereas MLP and SDP showed only a slight degradation in performance. The performance of MLP degraded when a partially relevant feature was included in the input. In contrast, SDP showed a much smaller increase in classification error. Thus, we conclude that SDP is highly robust to both the entire and the partial irrelevance of features.

According to this result, we suggest that using SDP allows us to keep relevant, irrelevant, and partially relevant dimensions in the given dataset with less worry of losing important information by unintentionally removing relevant or partially relevant features. In other words, SDP is a very practical binary classifier that can be easily used without prior knowledge of the problem, or highly technical feature selection methods. These advantages of SDP might be a reason for the previously reported high practicability of SDNN as a function approximator since it is composed of a set of multiple SDPs.

In future research, we plan to carry out experiments on higher dimensional problems with various types of features, such as those with different redundancies and distributions, to further clarify the robustness of SDP in detail.

References

1. Chang, C. C., Lin, C. J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2.3(27), 1–27 (2011)

2. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., Vapnik, V.: Feature selection for SVMs. In: Leen, T. K., Dietterich, T. G., Tresp, V. (eds) NIPS 2000. Advances in Neural Information Processing Systems, vol. 13, pp. 647–653. MIT Press, Cambridge (2001)
3. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research 3(May), 1157–1182 (2003)
4. Gasca, E., Sanchez, J. S., Alonso, R.: Eliminating redundancy and irrelevance using a new MLP-based feature selection method. Pattern Recognition 39(2), 313–315 (2006)
5. Kobayashi, T., Shibuya, T., Morita, M.: Q-learning in continuous state-action space with noisy and redundant inputs by using a selective desensitization neural network. Journal of Advanced Computational Intelligence and Intelligent Informatics 19(6), 825–832 (2015)
6. Nonaka, K., Tanaka, F., Morita, M.: Empirical comparison of feedforward neural network on two-variable function approximation (in Japanese). IEICE TRANSACTIONS on Information and Systems J94(12), 2114–2125 (2011)
7. Kawata, H., Tanaka, F., Suemitsu, A., Morita, M.: Practical surface EMG pattern classification by using a selective desensitization neural network. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds) ICONIP 2010. LNCS, vol. 6444, pp. 42–49. Springer, Heidelberg (2010)
8. Horie, K., Suemitsu, A., Morita, M.: Direct estimation of hand motion speed from surface electromyograms using a selective desensitization neural network. Journal of Signal Processing 18(4), 225–228 (2014)