Pattern classification using artificial neural networks

P. Di Stolfo, J. Reissig

February 16, 2014

Abstract

Artificial neural networks (ANNs) are a versatile tool to approximate solutions to nonlinear problems, and, thus, are widely used for pattern classification. In this project, we apply several configurations of ANNs to three different classification problems defined by publicly available data: The best results we achieved are 94.30% for Ionosphere, 61.23% for Red-wine quality, and 90.96% for Semeion. We use `Boone` [6] as the programming framework for classification.

1 Introduction

There are several easy-to-understand methods for pattern classification. Each of them has its specific advantages (e.g., easy to implement, sophisticated) and drawbacks (e.g., has a high computational complexity). Since pattern classification is a very important task and has many applications (e.g., computer-aided diagnostics, authentication, optical character recognition), it is imperative to find methods that are both efficient and yield high quality results. In the Pattern Recognition class (of WS 2013/14), we formed groups and were challenged to perform a series of tests on three datasets available to the public using a given classifier in order to determine which classifier delivers the best results.

In the following, we give an overview of artificial neural networks and explain how a certain training algorithm, namely the backpropagation algorithm, works. Our experimental setup, results, and a discussion will be given in the subsequent sections.

1.1 Artificial neural networks

A large class of methods for training classifiers can merely produce hyperplane discriminants. However, for many applications, linear discriminants are insufficient. Introducing nonlinear functions leads to arbitrarily complex discriminants, but choosing them appropriately poses a serious difficulty. Artificial neural networks (ANNs) combine both nonlinearity and linearity: They implement linear discriminants at base, but in a space where the inputs have been mapped nonlinearly. Via simply algorithms, ANNs learn the kind of nonlinearity from the training data, and, therefore, are among the most versatile tools for pattern classification.

ANNs transform a well-known concept from biology to computer sciences. The human neural system consists of neurons and links (synaptic connections) that save information, and modify the way of how information is transferred. When viewing this system from an abstract point of view, it consists of an input layer (e.g., senses, such as the sense of taste), an output layer (e.g., production of neural signals transmitted to a muscle that make it contract), and a non-transparent interconnection with additional neurons that may be recurrent.

1.1.1 Structure

Usually, ANNs consist of three kinds of layers of neurons with weighted connections between them, namely, the input, hidden, and output layers. If we ignore recurrent networks, we get the widely-used feed-forward networks of $K+1$ layers that only allow for connections between layers i and $i+1$, for $i = 0, \ldots, K$ (see Figure 1 for a sample ANN).

Each neuron represents a scalar value. In order to compute the value of an output neuron, the input values are transformed by certain steps: For each neuron, an activation function (or transfer function) is applied, and the result is used for subsequent computations; usually, there is one activation function for all neurons in an ANN. Each neuron's value is computed as follows: Each input weights is multiplied with the value of the neuron from which the input is received, and summed. Thus, for neuron y_j with activated outputs of the m neurons $x_1, \ldots, x_m$ in the previous layer connected to it, we get the value $H_j := \sum_{i=1}^{m} v_{j,i} x_i$, and after activation, the result is $y_j := f(H_j)$ (see Figure 2). Usually, a bias $w_{j,0}$ is added to the sum.

1.1.2 Classification

In this work, we apply ANNs for solving a classification problem: We want the ANN to tell which class an input vector belongs to. Thus, the number of input neurons matches the input vector's dimension. From the ANN output, one needs to tell which class is predicted by the ANN. This can be implemented as follows: There are as many output neurons as there are classes; a pattern is classified as i if and only if output neuron i has the highest value. As a consequence, the ANN needs to get trained exactly with this encoding of its output.

1.1.3 Training algorithms

As stated, the ANN has to learn the nonlinearity from the traning data. Thus, the concept of learning or training is

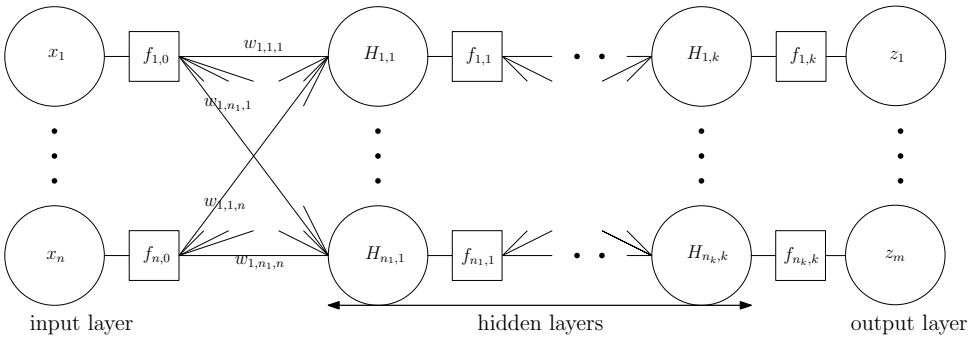

Figure 1: A typical feed-forward ANN.

key to ANNs. The goal of learning is the minimization of the total network error.

The network error usually computes the distance between the target values and the actual values w.r.t. the network weights. Thus, it is of the form $J(z(w)) = \frac{1}{2}(t-z(w))^2$ if w represents all weights of the network, t represents the target values and z the actual values. Most commonly, a single pattern is presented to the network and the error is computed. Then, based on the error, the weights are updated so that the error decreases. We get a sequence $\left(w^{(t)}\right)$ of network weights with decreasing error.

If J is differentiable w.r.t. w, we may use the method of gradient descent to minimize the error by changing the weights by $-\partial J/\partial w$, i.e., the weights are increased by the opposite direction of the greatest ascent, $w^{(t+1)} := w^{(t)} - \eta \partial J/\partial w^{(t)}$ with a factor of proportionality $\eta \in \mathbb{R}$, the so-called learning rate.

Backpropagation algorithm Adapting weights for each layer is not too difficult if we know how to compute the error J with respect to the weight matrix w. This calculation depicts the iterative nature of ANNs. Let us assume that the ANN admits only one hidden layer, as shown in Figure 2. Therefore, it has the layers input, hidden, and output.

For the weight change $\Delta w_{k,j}$ from hidden to output, we need to calculate $\frac{\partial}{\partial w_{k,j}} J(w)$ as shown in Figure 3. The calculation gets more complicated for the weight change $\Delta v_{j,i}$ from input to hidden, as shown in Figure 4.

There are several variants of backpropagation that might help reduce the error more quickly. Standard backpropagation, as explained, uses the rule $w_t := w_{t-1} - \eta \nabla J$. As a drawback, the gradient needs to be recalculated at every step. The so-called gradient reuse method tackles this issue by using the same ∇J as long as the error drops. Another variant modifies the learn rate η. A momentum term Δw_{t-1} is used such that the new weight change $\Delta w_t := -\eta \nabla J + \alpha \Delta w_{t-1}$ involves the previous weight change. Similarly, resilient backpropagation also includes the previous weight change and adapts the learn rate if the error changes its direction.

1.2 `Boone`

`Boone` [6] is short for Basic Object-Oriented Neural Environment. It is a versatile framework for modeling ANNs, and it provides both feed-forward networks and a recurrent network (Hopfield). Several training algorithms are available, such as the backpropagation algorithm and its resilient variant. `Boone` is highly configurable and may be extended easily.

2 Experimental setup

Our task was to apply ANNs for solving a classification problem with three different datasets:

- Ionosphere: A phased array of 16 high-frequency antennas collected radar signals (in $[-1, 1]$) from the ionosphere. The radar returns were manually classified as good or bad [9].
- Semeion: A group of 80 people wrote the digits 0 to 9 by hand. These written digits were scanned and stretched into a box of 16×16 pixels, and each pixel was scaled into a value 0 or 1. The goal is to map each sequence of 0 and 1 to the digit intended by the writer [2].
- Wine quality (short for Red-wine quality): Sixteen ingredients of red wine were examined for their amount within different types of wine. Wine experts evaluated these types and assigned grades from 3 to 8 [7].

Table 1 summarizes the number of patterns, features, and classes of each data set.

We performed data normalization such that all input data lies in $[0, 1]$. For classification, we partitioned each test data file by assigning each pattern to a partition. The partition number of each pattern is the new last entry in each line in the respective data files. Based on these p partitions in a given file, we were advised to perform leave-one-out validation: We perform p classification runs, so that in each of them a different single partition is used as test data and all other $p-1$ partitions are used as training data.

Tasks Since there are many possibilities of customizing ANNs, we were assigned the task of trying different net topologies:

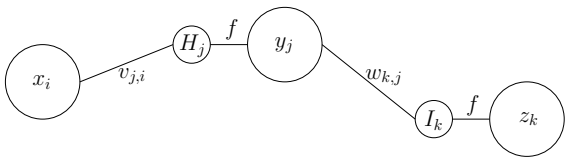

Figure 2: The neurons and weights of a three-layered ANN (involved in the calculations of section 1.1.3.)

$$\begin{aligned}\frac{\partial}{\partial w_{k,j}} J(w) &= \frac{\partial}{\partial w_{k,j}} \sum_r \frac{1}{2}(t_r - z_r(w))^2 \\ &= -\sum_r \frac{1}{2} 2(t_r - z_r) \frac{\partial}{\partial w_{k,j}} \underbrace{z_r(w)}_{\underbrace{f(I_r)}_{= f\left(\sum_s w_{r,s} y_s(w)\right)}} \\ &\qquad\qquad\qquad\qquad\qquad\qquad f'(I_r) \sum_s \underbrace{\frac{\partial}{\partial w_{k,j}} w_{r,s} y_s}_{= \begin{cases} 0, r \neq k \vee j \neq s \\ y_j, r = k \wedge j = s \end{cases}} \\ &= -(t_k - z_k) f'(I_k) y_j\end{aligned}$$

Figure 3: Differentiation of the error function J w.r.t. a weight $w_{k,j}$ from hidden to output layer.

- Number of neurons per layer: We concentrated on a single hidden layer, but tested a higher number of hidden layers as well.
- Activation functions: We examined the performance of the sigmoid function $s(x) := 1/\left(1 + e^{-x}\right)$ against $\tanh$. Since $\tanh$ is surjective on $(-1, 1)$ and s is surjective on $(0, 1)$, it might be advantageous to use either one of them depending on whether the problem values lie between $(-1, 1)$ or $(0, 1)$, respectively.
- Trainers: `Boone` provides both a backpropagation trainer Backprop and a resilient-backpropagation trainer Rprop that we tested both.

Additional tasks In order to attain higher quality results, we tried several ideas that might improve results. In many cases, these ideas are rules of thumb that were determined empirically without any rigorous mathematical argument. The ideas include a way of choosing the number of hidden neurons according to the number of features/classes [4], and "early stopping" [8]. See the next section for details.

3 Results

In order to provide a clear presentation of results, we were advised to use the confusion matrix C with entries $c_{i,j}$ containing the percentage of test patterns of actual class i that got misclassified as patterns of class j. Ideally, C is the identity matrix, meaning that no patterns were misclassified.

The configurations we examined can be partitioned into several groups. Since time for performing experiments is limited, it is almost impossible to try all combinations of reasonable configuration parameters.

We considered the following configurations:

- Hidden = Input, Hidden = Output: The number of neurons on the single hidden layer was determined by the number of input neurons, and by the number of output neurons, respectively.
- DHS: Duda et al. [4] suggest that, as a rule of thumb, the number of hidden neurons might be chosen so that the number of weights equals the number of training points divided by 10. Thus, if i, o and h are the numbers of input, output and hidden neurons, respectively, the number of weights is $i \cdot h + h \cdot o = (i + o)h$ and the number of training points is $n \cdot i$. Therefore, $h = n \cdot i/(10(i + o))$ might be a reasonable choice. We used these parameters in our DHS configuration.
- Functions: This series of trial ran tests the sigmoid against the $\tanh$ using Ionosphere.
- Alternative: In these runs, alternative configurations were considered, i.e., more hidden layers, a high number of steps, and the Backprop trainer.
- Winning configs: These configurations were determined by trial and error, and resemble what we found to be promising.

Each configuration with corresponding results can be seen in detail in Table 2, and the confusion matrices created from winning configurations are listed in Table 3.

4 Discussion

- Data sets: The three datasets, of course, have their peculiarities and vary both in complexity and classification quality.
  - Training Ionosphere was quite simple and fast due to the small dataset cardinality; e.g., with a single hidden layer of 50 neurons, we achieved 92.6%. As the winning configuration shows, a small number of training steps is sufficient. In particular, the winning configuration deteriorated from the 94.30% after 60 steps had

$$
\begin{aligned}
\frac{\partial}{\partial v_{j,i}} J(w) &= \frac{\partial}{\partial v_{j,i}} \sum_k \frac{1}{2}(t_k - z_k(w))^2 \\
&= -\sum_k \frac{1}{2} 2(t_k - z_k) \frac{\partial}{\partial v_{j,i}} \underbrace{z_k(w)}_{\lfloor f(I_k) = f\left(\sum_r w_{k,r} y_r(w)\right) \rfloor} \\
&\qquad f'(I_k) \sum_r w_{k,r} \frac{\partial}{\partial v_{j,i}} \underbrace{y_r(w)}_{\lfloor f(H_r) = f\left(\sum_s v_{s,r} x_s\right) \rfloor} \\
&\qquad\qquad f'(H_r) \underbrace{\sum_s \frac{\partial}{\partial v_{j,i}} v_{s,r} x_s}_{= \begin{cases} 0, r \neq i \vee j \neq s \\ x_j, r = i \wedge j = s \end{cases}} \\
&= -\textstyle\sum_k (t_k - z_k) f'(I_k) w_{k,i} f'(H_i) x_j
\end{aligned}
$$

Figure 4: Differentiation of the error function J w.r.t. a weight $v_{j,i}$ from input to hidden layer.

Data set	data set cardinality	features	classes
Ionosphere	351	34	2
Wine quality	1599	11	6
Semeion	3186	256	10

Table 1: Number of features, input and output dimensions

been reached and did not reach that percentage again even after 3000 steps. We chose the 60 steps manually; "early stopping" like this should rather be performed in a systematic, automatic way [8].

- Semeion was not too hard to classify, either, although the training time was much longer compared to Ionosphere. Even extra-small nets with a single hidden layer with a single neuron took approx. 100 minutes (total training time). Similar to Ionosphere, we achieved a high test data success of 90.96% with 1000 steps.
- Wine presented itself as the most complex data set of all three. Only a fraction of our tests surpassed 60%. In order to eliminate the (possibly negative) influence due to bad test data partitioning, we tried classical leave-one-out validation as well; the results were very similar. Additionally, we tried higher numbers of steps which improved training performance (since training performance is usually not higher than training performance), but that did not improve test performance. In related work, accuracies of 60% to at most 72% were achieved [7], [1]. Thus, we conclude that the nature of Wine data makes a high-quality classification difficult.

- Confusion: The confusion matrices allow us to tell which classes have been identified as similar to each other. It is interesting to see whether ANN classification mistakes are similar to those made by humans. In the case of Ionosphere, there are only two classes, and the second class has twice as many training patterns. Thus, it is natural that an ANN would rather take a 1 for a 2 than a 2 for a 1 (in 12.70% and 1.78% of the cases, respectively).

  In Semeion, the comparison of ANN vs. human gets more exciting since ANNs are expected not to fall for the same visual flaws as humans. In fact, the ANN tends to mix up digits differently than a human does: In our configuration, the 9 is the digit mistakened most often. Humans tend to write 9 and 4 quite similarly; the ANN took the 9 as a 4 and vice-versa only twice (i.e., in 2.53% of all 158 nines), whereas it took the 9 as a 3 eight times (i.e., in 5.03%). Although digits 1 and 7 look quite similar, the ANN did not mix these two up very often. Digit with best recognition is 0 (correct in 96.89% of all zeros).

  A factor that has a strong influence on classification quality is the method of discretization. In a typical digit recognition setting, the raw image data is provided so that a discretization suited for the particular problem can be performed (e.g., [3]). Thus, the given discretization of Semeion might not be optimal for ANNs. What is more, the discretization might cause the ANN to classify digits solely by the number of pixels that are 1, which might be accurate only in a fraction of the cases.

  For the results of Wine, being the most difficult dataset, one would expect the following: The three middle-class wines 5, 6 might be quite difficult to distinct even for a sommelier, but they may easily tell good and bad wines apart. The situation is quite different for the ANN: Due the little number of patterns in class 3, 4, and 8, none of these wines gets classified correctly. What is more, the ANN mixes up 7 with 6 in 65% of cases (but mistakens 6 as 7 in only 5% of the cases).

- Layers, neurons per layer and steps: The number of layers did not have a large impact except for the duration of training to increase dramatically. The classification quality for Ionosphere and Semeion with at least two layers was not higher than with one hidden layer. With tanh and the hidden layer configuration (11, 11), we reached 60.29% (see configuration Alter-

config.	dataset	hidden	trainer	activation	steps	avg. duration	train/test in %
Hidden = Input	Wine	11	Rprop	sigmoid	1000	35 : 05 min	65.71/59.98
	Ionosphere	34	Rprop	sigmoid	1000	05 : 20 min	100.0/92.31
	Ionosphere	34	Rprop	tanh	1000	25 : 20 min	99.13/88.03
Small	Semeion	1	Rprop	sigmoid	1000	267 : 55 min	96.82/89.71
Hidden = Output	Wine	6	Rprop	sigmoid	1000	26 : 40 min	60.43/57.79
	Wine	6	Rprop	tanh	1000	08 : 50 min	57.25/57.47
	Ionosphere	2	Rprop	sigmoid	1000	00 : 40 min	98.98/89.17
	Ionosphere	2	Rprop	tanh	1000	01 : 25 min	99.09/88.32
	Semeion	10	Rprop	sigmoid	100	16 : 05 min	97.04/90.33
DHS	Wine	103	Rprop	sigmoid	1000	190 min	69.48/58.60
	Wine	103	Rprop	tanh	1000	570 min	63.11/59.22
	Wine	103	Rprop	sigmoid	1500	275 min	71.74/57.60
	Wine	103	Rprop	tanh	1500	810 min	63.49/59.54
Functions	Ionosphere	10	Rprop	sigmoid	100	00 : 12 min	99.19/92.59
	Ionosphere	10	Rprop	tanh	100	00 : 48 min	95.01/88.32
	Ionosphere	10	Rprop	sigmoid	500	01 : 20 min	99.92/90.88
	Ionosphere	10	Rprop	tanh	500	04 : 00 min	98.25/90.60
	Ionosphere	10	Rprop	sigmoid	1000	02 : 05 min	99.91/91.17
	Ionosphere	10	Rprop	tanh	1000	06 : 50 min	99.22/90.60
	Ionosphere	10	Rprop	sigmoid	2000	05 : 10 min	99.92/91.17
	Ionosphere	10	Rprop	tanh	2000	18 : 10 min	99.44/89.46
Alternative	Ionosphere	150	Backprop	sigmoid	60	01 : 35 s	98.62/92.31
	Wine	11,11	Rprop	tanh	1000	202 min	62.71/60.29
	Wine	11,11	Rprop	tanh	2000	417 min	63.87/59.91
	Wine	11,11,11	Rprop	sigmoid	10000	540 min	80.61/56.66
	Wine	100	Rprop	sigmoid	16000	25 hours	84.15/56.79
Winning configs	Ionosphere	150	Rprop	sigmoid	60	52 s	99.30/94.30
	Semeion	50	Rprop	sigmoid	1000	40 hours	98.53/90.96
	Wine	11	Rprop	tanh	3500	233 : 15 min	63.80/61.23

Table 2: Trial configurations and results

native). Trial runs with many steps (more than 5000 and even 16000) did not improve nor surpass the results achieved by the runs with moderate number of steps (up to 3500).

- Activation functions: No significant differences in classification quality were noted. It took ANNs with tanh more steps to attain a certain training percentage; see e.g. configuration Functions where the sigmoid net reached 99%/92.6% at 100 steps compared to the tanh net which reached 95%/88.3%. The further development of Functions suggests that the sigmoid is indeed the better choice for Ionosphere, since tanh did not reach the 90.96% test performance of the sigmoid even after 2000 steps.

  In Wine, though, the results derived from nets using tanh were on par with the sigmoid. E.g., in Hidden = Input, we increased the steps up to 4000 for both tanh and the sigmoid in addition to the configurations in Table 2; the nets with tanh had an improved test performance even after 3000 steps; the sigmoid nets deteriorated from almost 60% to 57–58%. The winning configuration for Wine deteriorated after 4000 steps.

  Of course, training duration poses another important criterion for choosing such a function. We noticed that ANNs with tanh instead of the sigmoid finished the experiment run later on average (e.g., see Functions configuration). This difference probably stems from the complexity of evaluating the sigmoid and tanh which is 5 arithmetic operations plus 1 call to `math`'s exp for the former, and 6 operations plus 2 calls to `math`'s exp for the latter. (See `Function.java` of [6].)

- Trainers: As expected [5], Rprop was faster than its competitor Backprop, but classification quality was on par (compare, e.g. the Ionosphere test in configuration Alternative with the winning configuration for Ionosphere).

5 Summary and outlook

With ANNs, it is quite easy to create classifiers that provide high-quality results. Using the Java framework `Boone`, we created ANNs that classified three different datasets by means of a partitioned leave-one-out classification. Our top results for testing include 94.30% for Ionosphere, 90.96% for Semeion and 61.23% for Red wine quality, all achieved with a single-layered network. Additionally, we experimented with two transfer functions, we tried two different training mechanisms and tested how two or more hidden layers affect the quality

Ionosphere: Signal	good	bad
good	87.30 110	12.70 16
bad	1.78 4	98.22 221

Red-wine quality	3	4	5	6	7	8
3	0 0	0 0	90. 9	10. 1	0 0	0 0
4	0 0	0 0	67.92 36	28.30 15	3.77 2	0 0
5	0 0	0 0	74.30 506	24.96 170	0.73 5	0 0
6	0 0	0 0	29.78 190	64.73 413	5.49 35	0 0
7	0 0	0 0	5.03 10	64.82 129	30.15 60	0 0
8	0 0	0 0	5.56 1	55.56 10	38.89 7	0 0

Semeion: Digits	0	1	2	3	4	5	6	7	8	9
0	96.89 156	0 0	0 0	0 0	0.62 1	0 0	0.62 1	0.62 1	1.24 2	0 0
1	0 0	91.98 149	2.47 4	1.23 2	1.23 2	0.62 1	0 0	1.23 2	0.62 1	0.62 1
2	1.89 3	0.63 1	89.94 143	0.63 1	1.89 3	0 0	0 0	1.26 2	3.14 5	0.63 1
3	0 0	1.89 3	1.89 3	89.94 143	1.26 2	0.63 1	0 0	0.63 1	0 0	3.77 6
4	1.24 2	3.73 6	0.62 1	0 0	87.58 141	1.86 3	0.62 1	1.86 3	1.24 2	1.24 2
5	0 0	0.63 1	0 0	0.63 1	0.63 1	94.34 150	1.26 2	0.63 1	0.63 1	1.26 2
6	1.24 2	0.62 1	0 0	0 0	0 0	1.24 2	95.65 154	0 0	1.24 2	0 0
7	0 0	1.27 2	0 0	1.90 3	2.53 4	0 0	0.63 1	90.51 143	1.27 2	1.90 3
8	1.29 2	1.94 3	1.94 3	0 0	1.29 2	0.65 1	0 0	0.66 1	87.74 136	4.52 7
9	1.27 2	1.27 2	0.63 1	5.06 8	1.27 2	2.53 4	0 0	1.27 2	1.90 3	84.81 134

Table 3: Winning configurations, see Table 2 for a detailed configuration.

of results. Even with small and, thus, computationally cheap configurations, we achieved competitive results.

For future projects, some interesting questions turn up. Concerning the comparably low quality of Red-wine classification, it would be important to have a computationally cheap measure of how hard a dataset is, e.g., an uneven distribution of singular values (one very large singlar value compared to all others). Since the classifiers used in class (k-NN classifier, SVM, ...) reached slighlty different results, it would be highly interesting to have an upper bound on the maximum classification quality as well as a kind of maximum quality achievable by an ANN.

For the Semeion dataset, it would be interesting to check how the number of pixels set to 1 in the digital representation of the image influence the network's choice.

There exist several ideas of how the ANN classification results could be improved that have been shown to work quite well. For instance, a problem to be tackled is that not all features have a real significance to the classification but rather confuse the network. In this case, a reduction in dimensionality of features usually helps; this can be achieved by means of singular value decomposition (SVD). As a side note, it would be interesting to see which features are considered as superfluous by the ANN. Since each feature is represented by a particular input neuron, one can run a high number of training steps and check if the weights on edges that leave a certain input neuron all approach zero. It would be worth considering whether these neurons indeed correspond to a feature of a small singular value.

On the other side of the spectrum, if features have too few dimensions, the network is likely to mix up the associated pattern's classes due to their similarity. Thus, networks with hints try to resolve this issue by adding new information to the patterns [4]. Similarly, support vector machines apply this so-called kernel trick in order to increase dimensionality of patterns [1].

Another way of improving ANN classification is to set network weights using evolutionary algorithms. This approach has shown to deliver great results [1], in particular when generalized multi-layer perceptrons (GMLP), where network edges may connect not only neighbouring layers, are used.

References

[1] Tobias Berka and Helmut A. Mayer. "Evolving Artificial Neural Networks for Nonlinear Feature Construction". In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*. GECCO '13. Amsterdam, The Netherlands: ACM, 2013, pp. 1037–1044. ISBN: 978-1-4503-1963-8. DOI: 10.1145/2463372.2463502. URL: http://doi.acm.org/10.1145/2463372.2463502.

[2] Semeion Research Center of Sciences of Communication. *Semeion Handwritten Digit*. 1998. URL: http://archive.ics.uci.edu/ml/machine-learning-databases/semeion/semeion.names.

[3] M. Diem et al. "ICDAR 2013 Competition on Handwritten Digit Recognition (HDRC 2013)". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. 2013, pp. 1422–1427. DOI: 10.1109/ICDAR.2013.287.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN: 0471056693.

[5] Christian Igel and Michael Hüsken. "Improving the Rprop learning algorithm". In: *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*. Citeseer. 2000, pp. 115–121.

[6] A. Mayer and H. Mayer. *Boone*. 2010. URL: http://www.cosy.sbg.ac.at/project/robolab/?projects-boone.

[7] F. Almeida T. Matos P. Cortez A. Cerdeira and J. Reis. "Modeling wine preferences by data mining from physicochemical properties." In: (2009). URL: http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality.names.

[8] Lutz Prechelt. "Early Stopping - but when?" In: *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*. Springer-Verlag, 1997, pp. 55–69.

[9] Vince Sigillito. *Johns Hopkins University Ionosphere database*. 1989. URL: http://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.names.