Control and Cybernetics

vol. **33** (2004) No. 4

A survey of factors influencing MLP error surface

by

Mirosław Kordos[1] **and Włodzisław Duch**[2,3]

[1]Faculty of Automatic Control, Electronics and Computer Science,
The Silesian University of Technology
Gliwice, Poland
[2]Department of Informatics, Nicholaus Copernicus University
Toruń, Poland
[3]School of Computer Engineering, Nanyang Technological University
Singapore
www.phys.uni.torun.pl/~duch

Abstract: Visualization of neural network error surfaces and learning trajectories helps to understand the influence of numerous factors on the neural learning process. This understanding can be used to improve training and design of MLP networks. The following topics are discussed using a few benchmark datasets for illustration: general error surface properties including local minima, plateaus and narrow funnels, their dependence on network structure, input data, transfer and error functions, consequences of weight initialization, and interesting directions in the weight space. The error surfaces are shown in 3-dimensional PCA-based projections. Finally a possibility of effective weight number reduction is discussed.

Keywords: neural networks, MLP, error surface, visualization, learning trajectory.

1. Introduction

Error surface (ES) $E(\mathbf{W}) = \sum_X ||\mathbf{Y} - M(\mathbf{X};\mathbf{W})||$ of a neural network is defined in the weight space $\mathbf{W}$ (including biases as W_0 components) for a given set of training vectors $\mathbf{X}$, desired output vector $\mathbf{Y}$ and a vector mapping $M(\mathbf{X};\mathbf{W})$ provided by the neural network. Only the multi-layer perceptron (MLP) networks will be considered here, although the same techniques may be used to investigate other types of feedforward networks. An MLP training process can be defined as a search for the global minimum on the hyper-surface $E(\mathbf{W})$, where the learning process creates a trajectory on that surface.

One way to understand better the learning dynamics of the MLP is to visualize both the ES and the learning trajectory using projections of the original weight space on the two- or three-dimensional subspace. The projection directions should preserve most information about the original surface. In two-dimensional visualizations error value is displayed on the vertical axis, with the horizontal axis presenting one interesting direction selected in the weight space. A good choice is either the gradient direction, or the first principal component direction. A sample plot showing the change of the mean squared error (MSE) in the direction of numerical gradient (Duch and Kordos, 2003) is shown in Fig. 1 and in the subspace of two most important PCA components in Fig. 2. The MLP network with a single hidden layer composed of four nodes (4-4-3 architecture) has been trained on the Iris data taken from the UCI repository (Mertz and Murphy, 1999). This data is frequently used for simple benchmarks of classification systems.

The curves in Fig. 1 were created by changing the weight vector **W** in the gradient direction $d\mathbf{W}$ after four consecutive training epochs that started from the minimum found in the gradient direction at the previous step. The first curve has a narrow and deep minimum, indicating that a rather narrow funnel is traversed on the error surface. The second and the subsequent curves reach lower error levels. The ravine on ES is getting wider and wider during the training, finally leading to a broad plateau. That is visible in some of the sections in gradient directions, although not in all, since sometimes the gradient direction deflects from the ravine direction. This should be expected in all problems where separation of vectors that belong to different classes is relatively easy. The error surface should then be insensitive to weight changes that correspond to rotations and shifts of decision borders that do not affect the separation.

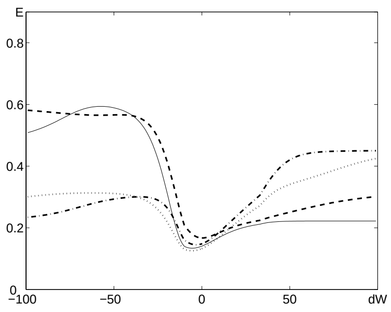


Figure 1. Section of the MLP (4-4-3) error surface for the Iris data in directions of numerical gradients after four consecutive training epochs.

It seems worthwhile to investigate error surfaces in higher dimensional spaces. PCA (Principal Component Analysis) is a natural choice for visualizing the weight space because it provides components from which the original weight space may be reconstructed with the highest accuracy. Moreover, the first two PCA components capture almost all variance of the weight changes. PCA directions were previously used for three-dimensional visualization of learning trajectories by Gallagher (Gallagher, 2000). We propose here to use them also for the visualization of MLP error surfaces.

By using weights from an MLP network with four hidden nodes in a single hidden layer trained on the Iris data two principal components c_1 and c_2 were found, producing the error surface shown on the left side of Fig. 2. The learning trajectory lies on the bottom of one of the ravines. Starting the training from another point could result with the trajectory lying on the bottom of another ravine. Some sample trajectories are shown in Section 7.

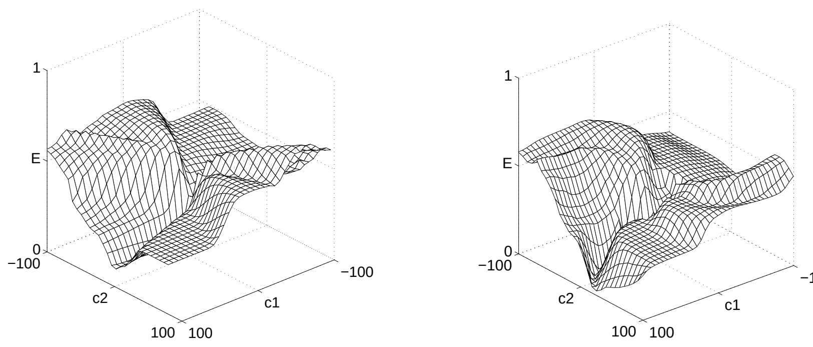


Figure 2. Left: MLP (4-4-3) error surface for the Iris data displayed in the first two PCA directions; right: the same error surface rotated and stretched to show additional aspects of the surface structure.

2. Research methodology

2.1. Training algorithms

The following procedure of determining the projection directions has been used: a network is trained using either standard backpropagation (Haykin, 1994), Levenberg-Marquardt algorithm (Marquardt, 1964), scaled conjugate gradient (Moller, 1993), numerical gradient (Duch and Kordos, 2003), the simplest search-based numerical method that changes one weight at a time (Duch and Kordos, 2003), or its modified version with variable step search (Duch and Kordos, 2004). However, the results of computational experiments do not depend significantly on the training algorithm. Weight vectors $\mathbf{W}(t)$ containing all adaptable network parameters are stored after each training epoch t and collected in the weight matrix $\mathbf{W}_{\max} = [\mathbf{W}(1), \mathbf{W}(1) \ldots \mathbf{W}(t_{\max})]$. Principal components are obtained using Singular Value Decomposition (SVD) applied directly to the $\mathbf{W}_{\max}$ matrix, or to the covariance matrix calculated using the

$\mathbf{W}(t)$ vectors. Interesting directions may also be obtained using independent component analysis (ICA). FastICA algorithm has been used to determine ICA directions. These methods are compared in Subsection 2.5.

2.2. Limitations of PCA projections

Although PCA projections are very useful for visualization of error surfaces they do not show certain aspects of the high-dimensional surfaces:

1. Some ravines in which the training trajectories lie are curved, not straight as shown in the PCA projections.
2. The deepest ravines reach lower error values than those shown in PCA projections.
3. Sometimes shallow local minima close to the ES center are visible in PCA projections, although they do not exist in the original ES.

The curvature and greater steepness of ravines are not shown, because they are not visible in three-dimensional projections. They can be only imagined or visualized if the projection directions will change locally in different fragments of the plot. Such detailed visualizations have not been attempted because they may be rather difficult to interpret. PCA projections may not offer a perfect representation of the ES, but still carry a lot of useful information. The right side of Fig. 2 shows how the ES may look like, addressing the points mentioned above. It was obtained by applying horizontal rotation and vertical stretching to some fragments of the original ES projection in Fig. 2, left.

2.3. Methods of constructing plots

Vertical axis in the plots shows relative error per vector and per class, $E_r(\mathbf{W}) = E(\mathbf{W})/N_v N_c$, where N_v is the number of vectors and N_c is the number of classes in the training set. For all error functions based on Minkovsky's metric $||\cdot||$ the error function is bounded from above by $N_v N_c$, thus the relative error is bounded by 1. Horizontal axes show distances in the weight space in c_1 and c_2 PCA directions corresponding to the first and the second eigenvector of the weight covariance matrix.

MLP networks were trained for data classification. The character of ES is determined by the dataset and network structure, but it does not depend on the training method or network initialization. The exact number of epochs varied depending on the training algorithm and dataset used. Network training was stopped before reaching convergence, when the error reached 10% above the minimum value that can be achieved. The final epochs, when the error was decreasing slowly and weights of output neurons tended sometimes to grow quicker then those of hidden neurons, were rejected. The training was repeated several times for a given method with various random initial weights.

Neither the random weight distribution, nor the training method has significant influence on the shape of ES presented in the space of two main PCA components. Moreover, weights of each layer have comparable contributions in determining PCA directions. The projection of error surface for a given dataset and network structure may differ a bit, depending on the initial weights and training method. It may rotate from one plot to another, its elements may be a bit higher or lower, but the overall structure is well preserved.

To obtain the most reliable ES projection PCA should be calculated from the training cycles ranging from the initial weights to that point when the error begins to change very slowly. However, if only the first half or even fewer training cycles are taken the differences are hardly visible. It is important that the initial training cycles, when the error changes are rapid, not be omitted; otherwise some distortions, that are described in Section 8.2., are likely to occur.

MLPs based on sigmoidal transfer functions are used to generate most of the plots presented here, but ES projections obtained with hyperbolic tangent functions do not differ significantly. Few other types of neural functions have also been tested.

Experiments with over 20 datasets, most of them from the UCI machine learning dataset repository (Mertz and Murphy, 1999), have been made. To be concise only one ES typical for a given situation is shown here; the others are qualitatively similar.

The name of a dataset in figure labels is followed by the number of neurons in successive layers; for example, in Fig. 4 Iris 4-4-3 means that the MLP network with 4 inputs, 4 hidden and 3 output neurons has been trained on the Iris data.

2.4. Typical PCA values

Typically, the first and second PCA directions contain together about 95% of the total variance. There is a strong correlation between the change of the weight $dw_i = w_i(t_0) - w_i(t_{\max})$ during the training and its corresponding entry in the first principal component vector $c_1(w_i)$ (Fig. 3, left). The remaining entries in the principal component vector seem to be uncorrelated with the change of corresponding weights during the training (Fig. 3, right).

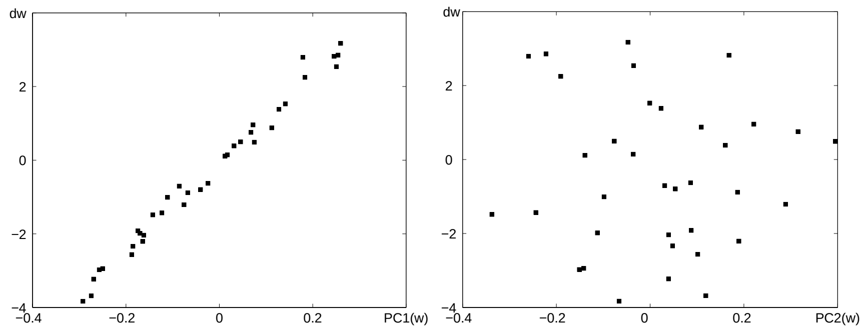
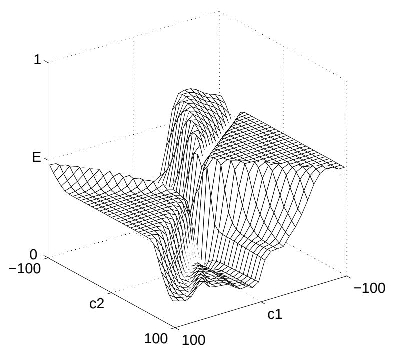


Figure 3. Left: Correlation of $c_1(w_i)$ with the change dw_i of the weight for Iris (4-4-3); right: No such correlation is found for the second component.

2.5. Comparison of SVD on weight matrix, weight covariance matrix, FastICA and two-weight coordinate systems

SVD can be calculated either directly on the weight matrix, or on the weight covariance matrix. The resulting plots are of similar nature, although eigenvalue distribution is different. The weight matrix gives smaller first to second eigenvalue ratio and has larger less significant eigenvalues, but in both cases the first and second PCA directions typically contain about 94–97% of the total variance. Nevertheless, a covariance matrix has two advantages: a projection of the error surface reaches lower error values, and plots differ less from training to training, being less influenced by random initial distribution of weights. For this reason all plots presented here are based on a covariance matrix, except for the two sample ES presented in Fig. 4 for comparison.



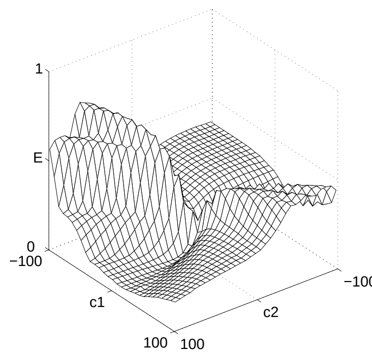


Figure 4. Left: ES of Iris (4-4-3) obtained with PCA on weight matrix for the same training as shown in Figs. 1-2; right: ES of Iris (4-4-3) obtained with FastICA algorithm for the same training as shown in Figs. 1-2.

PCA projections are in the directions of maximum variance, thus even if the data is clearly divided into two clusters, PCA may not reveal that structure. ICA (Independent Component Analysis) projections are in the maximally non-gaussian directions, providing usually a better separation of clusters. ICA approach may therefore show some additional ES properties that are not visible in PCA projections. FastICA algorithm (Hyvarinen and Oja, 2002) has been used here, resulting in very similar projections to those obtained with PCA on the covariance matrix. The global character of both projections is the same. Magnification of ES obtained with ICA projections reveals, however, more details that are hardly visible on the scale of Fig. 4, left in the form of rough ridges. The first ICA direction is almost parallel to the first PCA direction with the cosine between them approximately equal to 0.99, but the second PCA and ICA components are uncorrelated, with the cosine between them being usually below 0.3. Change of various FastICA algorithm parameters does not noticeably change the character of the plots, althought they may differ slightly because FastICA uses some random numbers.

Using a coordinate system based on two most significant weights does not provide so much information as using principal components. Most error surfaces for networks with more than 20 weights in the two-weight coordinate system create four horizontal planes, which are sometimes reduced to two or even one plane. The surfaces have the same nature for every data set and every network structure. More complex shapes of ES projection in two-weight systems are rare for medium to large networks. This is not difficult to understand: changing only two weights cannot have much influence on the error, while changing PCA components leads to coordinated changes of all weights. In networks with significantly more hidden neurons than required to learn the task many neurons perform highly redundant roles. In such cases changing any two weights of the trained network may have no influence on the error and only one horizontal plane may be visible in the ES projection.

3. Influence of the network structure

Networks without hidden layers have very simple ES consisting only of some horizontal or slightly inclined half-planes, situated at various heights, with slopes connecting them (Fig. 5, left).

ES of networks with hidden layers has a starfish structure. An interesting depiction of it was given by Denker et al. (1987): "$E(\mathbf{W})$ surface resembles a sombrero that has been warped in certain symmetric ways: near the middle ($\mathbf{W} = 0$) all configurations have moderately bad E values. Radiating out from the center are a great number of ridges and valleys. The valleys get deeper as they go out, but asymptotically level out. In the best valleys, E is exactly or asymptotically zero, other valleys have higher floors". Pictures presented here confirm that global minima rarely create craters but frequently ravines reaching their minimum in infinity. This corresponds to the infinite growth of (usually output layer) weights when the training is continued.

Each of the h hidden neurons may be labeled by an arbitrary and unique number from 1 to h. Renumerating the network parameters does not change the mapping implemented by the network thus giving $h!$ permutational symmetries. A neural activation function for which $f(-x) = -f(x) + const$ gives further 2^h sign-flip symmetries (Sussmann, 1992). This gives together $2^h h!$ equivalent global minima. A training algorithm converges to that minimum which is easiest to reach from the starting point. Only some of these minima are clearly visible in the PCA projections.

Networks with two hidden layers have more complex ES than those with a single hidden layer, even if they have fewer neurons (Fig. 5, right). Finding optimal solution for networks with more than one hidden layer is more difficult, because the training algorithm may easily end on many high-laying plateaus.

In 3-layer networks with crossover connections the output layer is connected

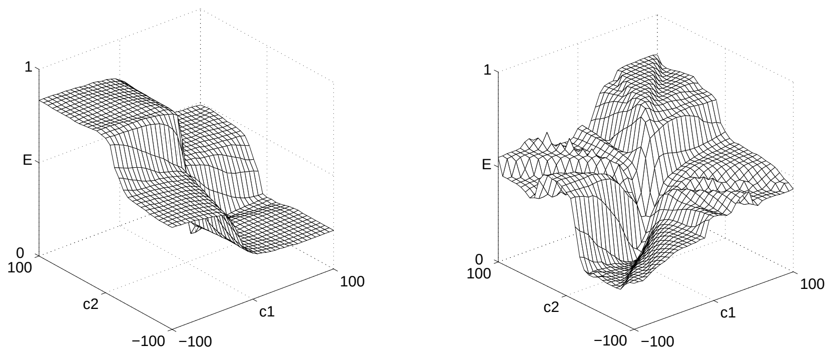


Figure 5. Left: ES of the 2-layer network (Iris 4-3); right: ES of the 4-layer network (Iris 4-4-4-3).

directly to both the input (as in the 2-layer networks) and the hidden layer (as in 3-layer networks). Consequently, their ES displays features of 2-layer networks (low symmetry of ES, Fig. 6, left) and 3-layers networks (complexity of ES).

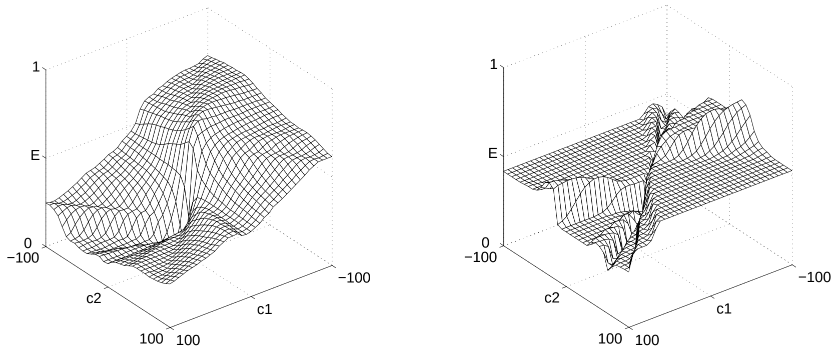


Figure 6. Left: ES of the 3-layer network with crossover connections (Iris 4-4-3); right: ES of the 3-layer network with too many hidden neurons (Iris 4-100-3).

Too few neurons in any hidden layer create a bottleneck, making it impossible for the network to learn the task. ES consists then of some horizontal planes all placed relatively high with some disturbances between them, but has no characteristic ravines leading to global minima (not shown here). The number of global minima visible in PCA projections initially grows with the increase of the number of hidden neurons, but with too many hidden neurons large horizontal planes begin to appear (Fig. 6, right). This effect caused by the weight redundancy is very clear in the two-weight coordinate system, where the projected ES is almost flat becuase many weights must be changed at the same time to change the error.

4. Influence of the training dataset

Similar network structure x-4-2 has been used for several datasets from the UCI repository (Mertz and Murphy, 1999). Generally, the following tendencies can be observed: 1) More complex training data produces more complex ES with more ravines, especially for vectors that are not linearly separable. 2) Equal number of examples in each class leads to a more symmetric ES (Gallagher, 2000).

Wisconsin breast cancer data (Fig. 7, left) has 699 vectors, two classes and only a few overlapping vectors, with good classifiers achieving 97% accuracy in cross-validation tests (Duch et al., 2001), therefore ES is quite simple. Iris (Fig. 2, right) has 3 classes with little overlap, and Ionosphere (Fig. 7, right) has two classes with not much more overlap, and they both give similar ES. Good classifiers reach about 96% of the cross-validation accuracy on these datasets (Duch et al., 2001).

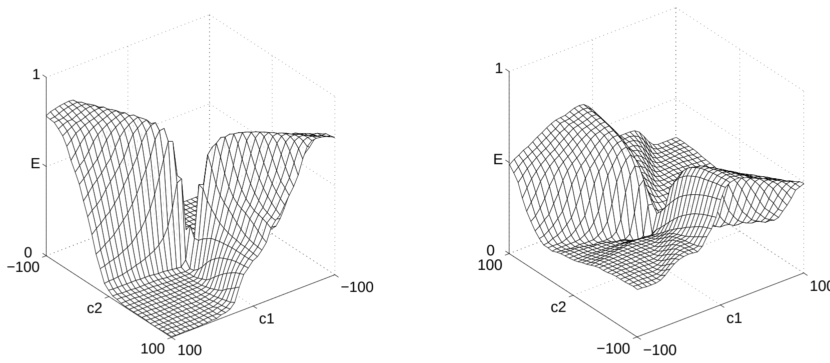


Figure 7. Left: ES of Wisconsin Breast Cancer (10-4-2); right: ES of Ionosphere data (34-4-2).

Appendicitis has only 21 vectors in one class and 85 in the second. It gives a highly non-symmetric ES (Fig. 8, left), and although convergence on this dataset is rather fast, the cross-validation accuracy is less than 90% (Duch et al., 2001). The same dataset with balanced number of vectors, taking all 21 vectors from class 1 and randomly selecting the same number from class 2, produces quite symmetric ES (Fig. 8, right), although with a more complex shape.

XOR is strongly linearly non-separable and therefore has a complex ES (Fig. 9, left). 6-bit parity is even more strongly linearly non-separable, with 32 clusters per class (XOR has only 2). Its ES is very intricate, but with equal number of vectors per class the symmetry is preserved (Fig. 9, right). It is clear that MLP convergence for parity problems is quite difficult to achieve.

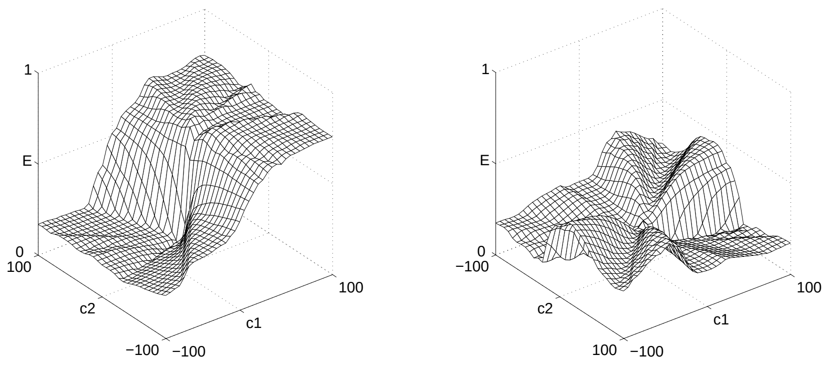


Figure 8. Left: ES of the entire Appendicitis dataset (7-4-2); right: ES of the Appendicitis dataset (7-4-2) with only 42 vectors - all 21 vectors of class 1 and randomly chosen 21 vectors of class 2.

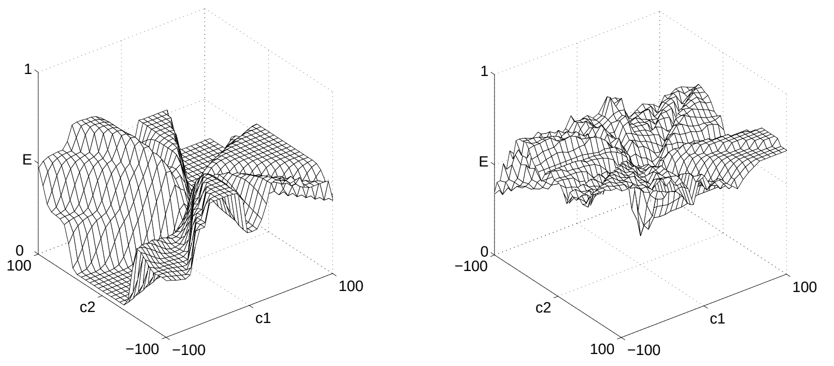


Figure 9. Left: ES of XOR (2-2-2); right: ES of 6-bit parity (6-6-2).

5. Influence of the transfer function type

5.1. Monotone transfer functions

Transfer functions may have very strong influence on the convergence properties of neural networks (Duch and Jankowski, 1999). In this section examples of error surfaces obtained with several transfer functions are presented, such as the staircase function or stretched sigmoid. The purpose of introducing these functions is to prevent the weights from infinite growth during training, and in the case of staircase functions also to simplify the calculations.

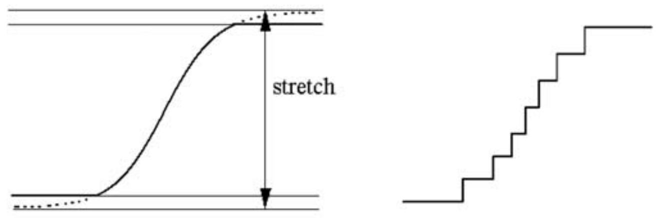


Figure 10. Transfer functions: a) stretched sigmoid, b) staircase function.

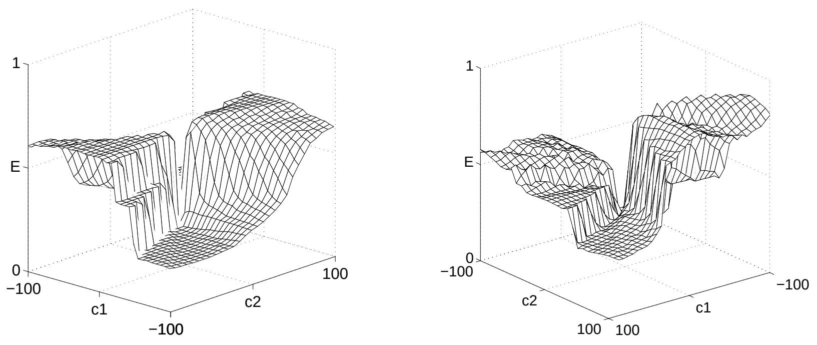


Figure 11. Left: ES of Ionosphere (34-4-2) with stretched sigmoid (stretch=1.3); right: ES of Iris (4-4-3) with staircase transfer function (5 stairs).

The fewer stairs in the staircase function, the more plateaus with sharp edges are found on the surface, and the more difficult training with gradient-based methods becomes. The stretched sigmoid does not cause any sharpness on the error surface. With a small stretch (1.01-1.1) it seems to be a good solution. But with a bigger stretch the function becomes similar to a step function and has a limited usefulness for complex data sets, with large flat areas appearing on the error surfaces (Fig. 11, left). More exotic neural functions may be used (Duch and Jankowski, 1999), but their investigation is beyond the scope of this article.

5.2. Non-monotone transfer functions

Non-monotone transfer functions produce many local minima. Fig. 12 shows ES of XOR (2-2-2) with a sinusoidal transfer function. The training of the network was successful only because all weight remained in the monotone sinusoid interval $(-\pi/2; \pi/2)$. ES visible in this figure has nothing in common with ES of MLP with monotone transfer functions, such as the widely used logistic and hyperbolic tangent sigmoidal functions, where local minima are very rare for

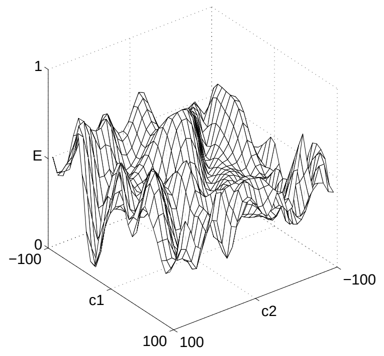


Figure 12. ES of XOR (2-2-2) with sinusoidal transfer function $S = 0.5 + 0.5 * \sin(x)$

real-world datasets, although they may appear as an effect of superpositions of two or more sigmoids. In the standard case ill-conditioning, large flat areas and choosing a wrong ES valley may cause many difficulties with training algorithms.

Basing on the ES shown in Figs. 11 and 12, and the ES obtained with standard sigmoidal transfer functions, it can be concluded that the shape of transfer function is reflected in the features of ES.

6. Influence of the error functions

6.1. Regularization of weights

The regularization term is added to the error function to keep the weights small and assure smooth, gradual changes that increase classification margins and thus improve generalization. In the simplest form the regularization term is the sum of all weights squared $\alpha \sum_i w_i^2$. The training error with the regularization term included reaches higher values even though the number of errors may be smaller. The error surface lies in general higher, especially farther from the center where larger weights $\mathbf{W}$ contribute to the error function. The effect is stronger for large regularization parameter α. A plot for the Wisconsin breast dataset with $\alpha = 0.03$ is presented in Fig. 13. A superposition of the original ES of the dataset and the paraboloid coming from the quadratic term can be observed.

Using MSE error function with desired output signals 0.1 and 0.9 or 0.2 and 0.8 produces very similar ES as with desired outputs 0 and 1, but a global minimum tends to lie close to the ES center in a shallow valley (not shown here).

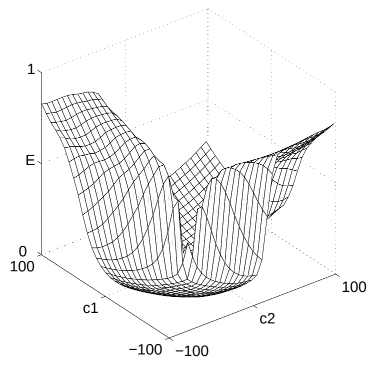


Figure 13. ES of Wisconsin Breast Cancer (10-4-2) with weight regularization, $\alpha = 0.03$.

6.2. Different exponents in the error function

Mean Squared Error function is commonly used, but the error function can use powers for absolute values of the difference between achieved and desired output with different exponents. This has an effect of paying more (or less) attention to small or large differences. An error surface does not depend strongly on the exponent of a power function in the 0.5 to 8.0 range. Two pictures of error surfaces obtained with more extreme values of exponent 0.1 and 32, are shown in Fig. 14. High error exponents successfully reduce weight growth and act as

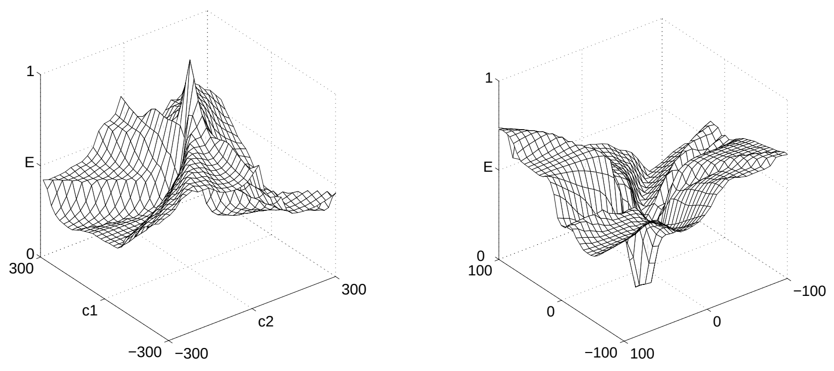


Figure 14. Left: ES of Iris (4-4-3) with the exponent of the error function 0.1; right: the same case with the exponent=32.

a weight regularization method. The learning trajectory remains in the global minimum near the center. For Iris 4-4-3 the module of weight vector never exceeded 25, no matter how long the training lasted, and the network always converged. Low exponents in the error function produce ES with central peak and slopes falling slowly down. With error exponent = 0.1 it is usually enough to reduce the distance error by 20% to achieve the same classification accuracy on a training set, as would require reducing an MSE error by 90%.

6.3. Cross-entropy error function

The cross-entropy error function for a a single output $M(\mathbf{X};\mathbf{W})$ and target values $Y(\mathbf{X}) \in \{0,1\}$ is given by the following formula:

$$E(\mathbf{W}) = -\sum_{\mathbf{X}} (Y(\mathbf{X}) \ln M(\mathbf{X};\mathbf{W}) + (1-Y(\mathbf{X})) \ln(1-M(\mathbf{X};\mathbf{W}))) \tag{1}$$

To avoid numerical problems with logarithms for arguments approaching zero the following modification has been used:

$$E(\mathbf{W}) = -\sum_{\mathbf{X}} \left(Y(\mathbf{X}) \frac{\ln(M(\mathbf{X};\mathbf{W})+\epsilon)}{\ln(1+\epsilon)} + (1-Y(\mathbf{X})) \frac{\ln(1-M(\mathbf{X};\mathbf{W})+\epsilon)}{\ln(1+\epsilon)} \right) \tag{2}$$

where $\epsilon \approx 10^{-10}$ is a small positive number.

The total network error is the sum of all single output errors. Compared to MSE or other power error functions, cross-entropy error functions give similar, though usually more complex ES. Furthermore, the ES changes slower with the distance from the center and therefore the ES with large c_1 and c_2 axes in the range of (-300 to 300) are shown (Fig. 15). Convergence in the final part of

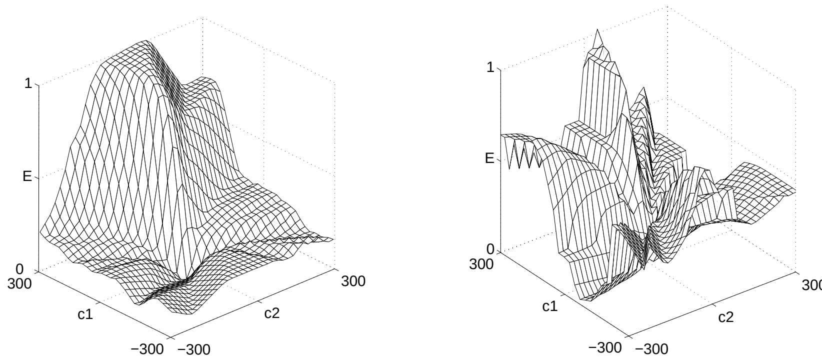


Figure 15. Left: ES of Appendicitis (7-4-2) with cross-entropy error function; right: ES of XOR (4-4-2) with cross-entropy error function (note different range on horizontal axes).

the training may therefore be rather slow. ES values for cross-entropy may be higher then 1 due to the fact that the error is not bounded by $N_v N_c$ as it is in the case of power error functions.

7. Visualization of the learning trajectory

Learning trajectories can be shown in the same plot with their corresponding error surfaces. Although the learning trajectory really does not lie on this two-dimensional section of ES, but somewhere in the weight space, to a certain degree it tends to adhere to the ES projection. The trajectory tends at least to lead to the ravine that was followed by the training algorithm (Fig. 16, right). The beginning of a trajectory lies often over the ES projection and its end under, because the ES projections are often flatter then original ES on which the trajectory lies. The trajectories in n-dimensional weight space frequently create arcs. The mean direction of the arc is usually parallel to the direction of the ES ravine in PCA projections (Fig. 16, left).

Learning trajectories look differently depending on the training algorithm and its parameters. Applying batch backpropagation with small learning rate one obtains trajectories that are very smooth. Increasing learning rate gives more irregular trajectories. Fragments of the backpropagation trajectories may go as well downwards as upwards, while trajectories obtained with some other algorithms go only downwards. Backpropagation trajectories with various learning rates have already been analyzed (Gallagher and Downs, 2003), and therefore they are not discussed here in detail.

ES is always associated with a given set of vectors. Any training that does not calculate the error for every epoch on the entire set leads to ES that changes for each subset of the training vectors. In particular, this is the case for online backpropagation – each point on the trajectory corresponds to another ES. This is why the trajectory is so rough and jagged, but globally it follows its main path. It is impractical to show such a trajectory and the average ES in one picture, because for successful training with online backpropagation the learning rate must be very small, and thus so small trajectory fragments going in different directions would be invisible in the global ES picture scale.

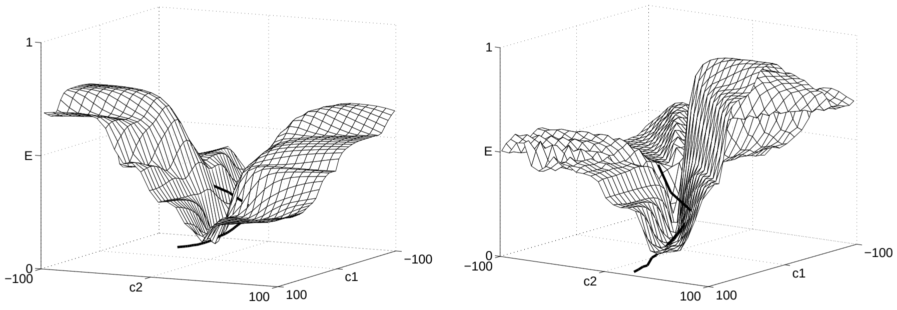


Figure 16. Left: ES and the learning trajectory of Iris (4-4-3) trained with backpropagation; right: trained with Levenberg-Marquardt algorithm.

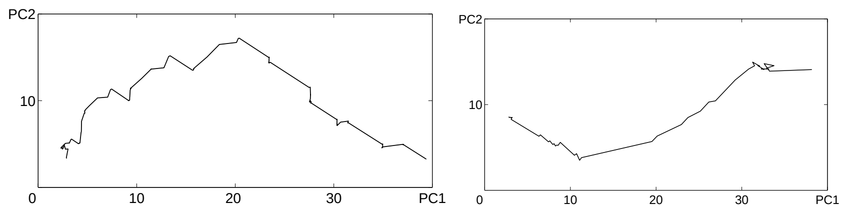


Figure 17. Two-dimensional projections in the first and second PCA direction of Iris (4-4-3) learning trajectories. Left: training with variable step search algorithm; right: training with scaled conjugate gradient.

Fig. 16 shows also the effect of different training algorithms on the error surfaces. First-order backpropagation algorithm (left subfigure) has slower convergence than the second-order Levenberg-Marquardt algorithms (right subfigure). In Fig. 17 trajectories of the other learning algorithms are displayed in two-dimensional projections.

8. Network training acceleration by reduction of effective parameter's number

PCA is frequently used for preprocessing of training data to reduce the number of network inputs. PCA was also proposed for weight pruning (Levin, 1994). In this section the possibility to use PCA to reduce the number of effective training parameters is discussed. After training the network for some number of epochs PCA is performed on the weight matrix. Then, searching for the minimum of error can take place in the reduced space of PCA-determined directions.

8.1. Directions in the weight space

The analysis of directions in a weight space reveals some important properties of ES that can be used to design or improve some neural training algorithms (Duch and Kordos, 2004). Some trends and tendencies are common for many datasets and network structures, differing only in details.

The curves for $\cos\mathbf{W}, \|\mathbf{W}\|, E(t), \cos(\mathbf{W}, \mathrm{PCA})$ shown in Fig. 18 are similar for various training methods. Decrease of the $E(t)$ error is precisely correlated with changes of the weight vector direction $\cos\mathbf{W}$. At the final stage of the training the direction of $\mathbf{W}$ remains almost constant and the error decreases very slowly, although the weights continue to grow. The trajectory is then already in the flat part of the ES ravine. Regularization or very high exponent of the error function may damp the weight growth, nevertheless the error decreases as long as the weight vector changes its direction.

The $\cos(\mathbf{W}, \mathrm{PCA})$ line in Fig. 18 shows cosine of the angle between the resultant PCA direction, calculated as a vector sum of all PCA components, each multiplied by its corresponding eigenvalue. Using only the first and the second PCA component gives almost identical results as using all PCA components. PCA components higher than the 6-th contain only the negligible 10^{-10} of total variance, and thus contribute only a small amount of noise that can safely be rejected. In the example only the weights from the first 100 epochs were included in the weight matrix for PCA calculation. The $\cos(\mathbf{W}, \text{PCA})$ takes the greatest value after about the 50-th epoch, after which PCA and $\mathbf{W}$ directions start to diverge more. The divergence is sometimes even stronger than in Fig. 18. For this reason a large jump along PCA directions during network training only seldom will be successful. PCA directions are very good for ES and even better for trajectory visualization, where a little difference in angles does not matter. However, using PCA for extrapolation of learning trajectories, thus making a jump of several epochs ahead is not an easy task, since the proper direction of the jump must be determined very precisely.

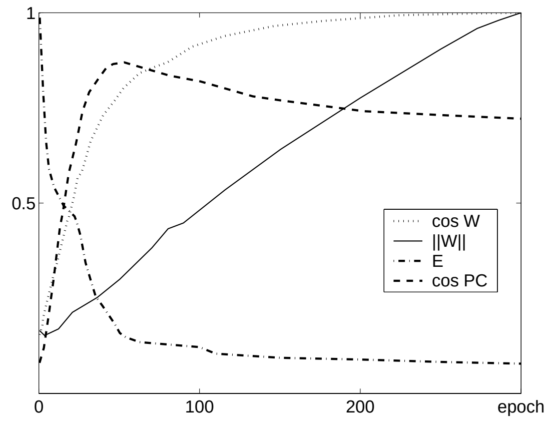


Figure 18. Change of parameters during network training. Vertical axis: normalized $\|W\|$, normalized MSE error, $\cos(\mathbf{W}) = \cos(\mathbf{W}(t) - W(t_{\max})), \cos(\mathbf{W}, \text{PCA})$ = cos between the weight vector $\mathbf{W}$ and the resultant PCA direction. Horizontal axis shows epoch number t.

8.2. PCA-based parameters reduction. A case study

1. Starting from the random weights (error=326) the network is trained on Wisconsin Breast Cancer dataset for several epochs. The training is stopped with error=240.
2. PCA directions are determined.
3. A minimum in the PCA direction is found (using a gradient-based method) with error=43 and a jump is made to that point.
4. No further error decrease in PCA-directions is possible. The network is trained again with a standard algorithm.

5. Again PCA directions are determined on the weight matrix from the last training.
6. Since PCA provides only the direction, there is a free choice of the starting point in the weight space to which the PCA components will be added. Two possibilities of choosing the point are considered:
   a) zero point in the weight space or the starting point of the entire training. The two points are very close to each other, therefore choosing any of them leads to a very similar result: the projection of the ES lifts up. The lowest point on the error surface has now the error=244 (Fig. 19, left).
   b) the last training point (Fig. 19, right). Now the projection of ES is not lifted up, but due to the fact that PCA directions are determined on the weight matrix from only a small part of the training some local PCA directions are obtained. The minimum is situated very close to the last point of the training.
7. In any case the jump can be made only once.

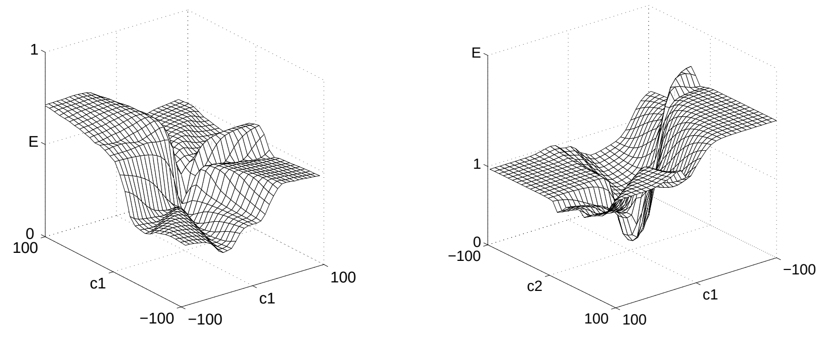


Figure 19. ES for the Wisconsin Breast Cancer (10-4-2) determined from the five training cycles after the jump; left: PCA components added to the zero point; right: PCA components added to the last training point.

The breast cancer dataset was chosen for the case study intentionally, because the dataset is very easy to train (it has very simple ES, Fig. 7). For most datasets such big jumps in PCA directions, decreasing error significantly, are not possible. However, it is always possible to obtain a PCA-based projection of ES on which the last point from the training lies. Almost always a point with error lower than that from the last training point can be found on the ES projection (Fig. 19, right). Still, the method is in most cases impractical because of the high computational costs of calculating PCA that is then used to make only a small move in the reduced space.

8.3. Extrapolating learning trajectories

Although for very complex, artificial problems, the trajectory can change its direction several times, for most real-world problems the learning trajectories create arcs, which are more or less irregular, depending on the training algorithm. Higher PCA components usually vanish as the training progresses, thus an attempt can be made to extrapolate learning trajectory only in two or three PCA directions. Many experiments were conducted with speeding up network training by trajectory extrapolation in up to six PCA directions. The results showed that in order to achieve a significant gain the trajectory must be extrapolated very precisely. That in most cases cannot be done in a single step and some search procedures are required to find the optimal point. Although achieving a smaller error in that way is usually possible, it is in most cases impractical due to the computational cost of calculating PCA every few epochs in order to make only a small extrapolation of the learning trajectory.

9. Conclusions

Although it is impossible to see high-dimensional spaces in three dimensions without any distortions, the PCA-based projections give an interesting insight into many important properties of the error surfaces. The most important ES properties are listed below.

1. ES of networks with hidden layers has a starfish structure.
2. ES depends on the network structure, becoming more complex when more hidden layers are added, and developing large plateaus if the network has many redundant weights.
3. ES are simple for linearly separable (or almost separable) training data, but become more complex for highly nonseparable data that require complex decision borders.
4. Smooth transfer functions lead to simpler ES, staircase or periodic function may create very complicated error landscapes.
5. Error functions do not influence ES strongly for a rather wide range of exponents.
6. Local minima in craters are rare in the standard MLP networks with monotone transfer functions trained with real-world data sets.
7. With MSE error functions global minima are in infinity. Local minima lie in ravines that asymptotically reach higher error values.
8. Difficulties of training algorithms result from bad initialization, choosing a wrong ES ravine, or entering large flat plateaus.

The training method used to generate data for PCA, as long as it converges, does not significantly influence the resulting error surfaces. The learning trajectories of many algorithms create an arc lying in the bottom of one of the ES valleys. This arc may be smooth or it may be rough. ES projections depend on the weights after each epoch. If the training is not successful then the learning trajectory and the ES projection are too flat and too highly situated.

ES has the greatest diversity close to its center. Far from the center the surface changes slowly and flat horizontal areas occupy much place. If the range of random initial weights is too broad then there is a great chance that the starting point will lie somewhere on the flat area, and as a result the network cannot be trained by any gradient-based or local search methods. On the contrary, if all initial weights are zero, the network can be successfully trained because gradients are large in this point. Some gradient methods, such as the backpropagation or numerical gradient algorithm, cannot be initialized with zero weights, but this is a limitation of these algorithms, not of the zero point itself.

In some cases network training can be accelerated by determining PCA components in the weight space after some initial training and then jumping to a minimum found in PCA coordinates, or by extrapolating the learning trajectory in the PCA direction, but a universal solution has not been found so far. Nonlinear techniques, such as principal curves, principal surfaces or kernel PCA, can also be used to display the surfaces and to attempt the training reduction. This is one of the subjects of our current research aimed at better understanding of neural networks and using this understanding to improve network architectures and training methods.

References

Denker, J., Schwartz, D., Wittner, B., Solla, S., and Howard, R., Jackel, L., Hopfield, J.J. (1987) Automatic learning, rule extraction and generalization. *Complex Systems* **1**, 887-922.

Duch, W. and Jankowski, N. (1999) Survey of neural transfer functions. *Neural Computing Surveys* **2**, 163-213.

Duch, W., Adamczak, R. and Grąbczewski, K. (2001) A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks* **12**, 277-306.

Kordos, M. and Duch, W. (2003) Search-based Training for Logical Rule Extraction by Multilayer Perceptron. *Proc. of Int. Conf. on Artificial Neural Networks (ICANN)*, Istanbul, June 2003, 86-89.

Kordos, M. and Duch, W. (2003) Multilayer Perceptron Trained with Numerical Gradient. *Proc. of Int. Conf. on Artificial Neural Networks (ICANN)*, Istanbul, June 2003, 106-109.

Kordos, M. and Duch, W. (2004) Variable Step Search Algorithm for MLP Training. *The 8th IASTED Int. Conf. on Artificial Intelligence and Soft Computing*, Marbella, Spain, Sept. 2004, 215-221.

Gallagher, M. and Downs, T. (2003) Visualization of Learning in Multilayer Perceptron Networks using PCA. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics* **33**, 28-34.

Hyvarinen, A. and Oja, E. (2002) *Independent Component Analysis: A Tutorial.* http://www.cis.hut.fi/projects/ica

LEVIN, A.U., LEEN, T.K. and MOODY, J.E. (1994) Fast Pruning Using Principal Components. *Advances in Neural Information Processing* **6**, 35-42.

MERTZ, C.J. and MURPHY, P.M. (1999) UCI repository of machine learning data-bases. http://www.ics.uci.edu/ mlearn/MLRepository.html

MOLLER, M.F. (1993) A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks*, **6**, 525-533.

RANGANATHAN, A. (2004) *The Levenberg-Marquardt Algorithm.* http://www.cc.gatech.edu/people/home/ananth

SUSSMANN, H.J. (1992) Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks* **5**, 589-593.