WSG Diskussionspapiere

WSG 60/97

A Neural Network Classifier for Spectral Pattern Recognition

On-Line versus Off-Line Backpropagation Training

Petra Staufer and Manfred M. Fischer

Institut für Wirtschafts- und Sozialgeographie

Wirtschaftsuniversität Wien

Department of Economic and Social Geography

Vienna University of Economics and Business Administration

WSG Discussion Papers

Abteilung für Theoretische und Angewandte Wirtschafts- und Sozialgeographie
Institut für Wirtschafts- und Sozialgeographie
Wirtschaftsuniversität Wien

Vorstand: o.Univ.Prof. Dr. Manfred M. Fischer
A - 1090 Wien, Augasse 2-6, Tel. (0222) 313 36 - 4836

Redaktion: Univ.Ass. Dr. Petra Staufer

WSG 60/97

A Neural Network Classifier for Spectral Pattern Recognition

On-Line versus Off-Line Backpropagation Training

Petra Staufer and Manfred M. Fischer

WSG-Discussion Paper 60

December 1997

Gedruckt mit Unterstützung
des Bundesministerium
für Wissenschaft und Verkehr
in Wien

WSG Discussion Papers are interim
reports presenting work in progress
and papers which have been submitted
for publication elsewhere.

ISBN 3 85037 071 2

A Neural Network Classifier for Spectral Pattern Recognition
On-line versus Off-Line Backpropagation Training

Petra Staufer[a,*] Manfred M. Fischer[a,b,*]

[a] Department of Economic and Social Geography, Wirtschaftsuniversität Wien
Augasse 2-6, A-1090 Vienna, Austria
E-mail: {petra.staufer, manfred.m.fischer}@wu-wien.ac.at

[b] Institute for Urban and Regional Science, Austrian Academy of Sciences
Postgasse 7/4, A-1010 Vienna, Austria; E-mail: manfred.m.fischer@oeaw.ac.at

Abstract

In this contributon we evaluate on-line and off-line techniques to train a single hidden layer neural network classifier with logistic hidden and softmax output transfer functions on a multispectral pixel-by-pixel classification problem. In contrast to current practice a multiple class cross-entropy error function has been chosen as the function to be minimized. The non-linear differential equations cannot be solved in closed form. To solve for a set of locally minimizing parameters we use the gradient descent technique for parameter updating based upon the backpropagation technique for evaluating the partial derivatives of the error function with respect to the parameter weights. Empirical evidence shows that on-line and epoch-based gradient descent backpropagation fail to converge within 100,000 iterations, due to the fixed step size. Batch gradient descent backpropagation training is superior in terms of learning speed and convergence behaviour. Stochastic epoch-based training tends to be slightly more effective than on-line and batch training in terms of generalization performance, especially when the number of training examples is larger. Moreover, it is less prone to fall into local minima than on-line and batch modes of operation.

Keywords: Pixel-by-pixel classification, feedforward neural networks, network training, backpropagation, gradient descent technique

* *Acknowledgements:* The authors gratefully acknowledge the Department of Economic and Social Geography, Wirtschaftsuniversität Wien, for providing the software routines used in this study. The algorithms are based on the Numerical Recipes in C library [1] modified by *Adrian Trapletti* (supported by a grant of the Kuratorium of the Wirtschaftsuniversität Wien) as necessary to work in an error backpropagation neural network environment.

1 Introduction

The analysis of remotely sensed imagery from earth observation satellite systems such as Landsat-TM, SPOT, IRS-1C etc. is usually achieved by machine-oriented pattern recognition techniques. Unfortunately, many of the commonly used image-processing techniques such as pixel-by-pixel classification based on maximum likelihood (ML) tend to perform poorly in distinguishing between the various categories of urban land cover of interest to land use planners. This is because urban areas comprise a complex spatial assemblage of disparate land cover types — including built structures, numerous vegetation types, bare soil and water bodies — each of which has different reflectance characteristics. Thus, classifying urban land cover is a challenging spectral pattern recognition task requiring novel tools such as computational intelligence techniques.

Neural network models in general and feedforward networks in particular, tend to provide novel, elegant and extremely valuable classes of mathematical tools for spectral pattern recognition (see, e.g., [2] for an overview). Analytical results show that feedforward networks with a single hidden layer are capable to approximate arbitrary mappings sufficiently well in the presence of noise ([3], [4]). But they do not provide more than very general guidance on how this can be done, and what guidance they do offer suggests that network training will be hard.

The focus in this contribution is on the issue of network training. Backpropagation of gradient descent errors [5] is central to much current work on learning in computational neural networks. In fact, the development of this training procedure is one of the main reasons for the renewed interest in computational neural networks and for many success stories in real world application contexts.

On the other hand many enhancements of and variants to gradient descent backpropagation have been proposed (see, for example, [6], [7]). These are mostly heuristic modifications with goals of increased speed of convergence, avoidance of local minima and/or improvement in the network model's ability to generalize, and usually evaluated on artificial benchmark problems. On the other hand, it is a rather surprising fact that standard backpropagation performs better than many fast training algorithms as soon as the learning task achieves a realistic level of complexity and when the size of the training set goes beyond a critical threshold [8]. The relative effectiveness of on-line and off-line versions of backpropagation of gradient descent errors is highly dependent on the problem under consideration [9]. Two versions of off-line backpropagation are considered: stochastic epoch-based learning (with epoch sizes $K^* = 300, 600$ and batch learning. They differ in how often the weights are updated. The batch version updates the weights after all training patterns have been propagated through the network. It stands in contrast to on-line training where the weights are changed after the presentation of each training set pattern and to epoch-based training using K^* patterns randomly chosen from the training set.

There has not yet been sufficient comparative study on real world pattern classification problems to determine whether on-line (pattern based) or off-line versions tend to be superior. The purpose of this contribution is to analyse the efficacy of on-line and off-line gradient descent backpropagation training, on a multispectral pixel-by-pixel classification problem with a challenging level of complexity and a larger size of the training set. A single hidden layer neural network classifier with logistic hidden and softmax output transfer functions is utilized along with a multiple cross-entropy function to be minimized during

the training process. The evaluation is based on two performance indices, training time (measured in terms of CPU seconds) and out-of-sample (generalization) classification accuracy measured in terms of the standard deviation of 10 simulations differing in initial weights.

The remainder of this paper is organized as follows. The next section describes the basic features of the classification task (see Section 2.1) and then of the neural network classifier to be used in the study (see Section 2.2). In Section 3 the network training problem is formulated as a problem of minimizing the multiple class cross-entropy error function (see Section 3.1). We can not solve the non-linear differential equations in closed form, but we approximate a local minimum iteratively with the gradient descent technique (see Section 3.2) and use the backpropagation technique to efficiently evaluate the partial derivatives of the error function (see Section 3.3). Section 4 presents the experimental results evaluation criteria in view. Section 5 summarizes the results achieved and outlines directions for future research.

2 The Spectral Pattern Recognition Task

2.1 The Classification Problem

Spectral pattern recognition deals with classifications that utilize pixel-by-pixel spectral information from satellite imagery. The spectral pattern recognition problem considered in this study is the supervised pixel-by-pixel classification problem in which the classifer is trained with examples of the classes (categories) to be recognized in the data set. This is achieved by using limited ground survey information which specifies where examples of specific categories are to be found in the imagery. Such ground truth information has been gathered on sites which are well representative of the larger area analyzed from space. The image data set consists of 2,460 pixels (resolution cells) selected from a Landsat Thematic Mapper (TM) scene (270×360 pixels) from the city of Vienna and its northern surroundings (observation date: June 5, 1985, location of the centre: $16°23'E$, $48°14'N$; TM Quarter Scene 190-026/4). The six Landsat TM spectral bands used are blue (SB1), green (SB2), red (SB3), near infrared (SB4), and mid infrared (SB5 and SB7), excluding the thermal band with only a 120 meter ground resolution. Thus, each TM pixel represents a ground area of 30m×30m and has six spectral band values varying over 256 digital numbers (8 bit).

`Table 1 to be placed about here`

The purpose of the multispectral classifiction task at hand is to distinguish between the eight classes of urban land use listed in Table 1. The classes chosen are meaningful to photointerpreters and land use managers, but are not necessarily spectrally homogeneous. The classification problem used to evaluate the performance of the above training procedures in a real-world context, is challenging. The pixel-based remotely sensed spectral band values are noisy and sometimes unreliable. Some of the urban land use classes are sparsely distributed in the image. The number of training sites is small relative to the number of land use categories (one site training case). The training sites vary between 154 pixels (class suburban) and 602 pixels (class woodland and public gardens with trees).

The above mentioned six TM bands provide the data set input for each pixel, with values scaled to the interval $[0.1, 0.9]$. This approach resulted in a database consisting of 2,460 pixels (about 2.5 pecent of all the pixels in the scene) that are described by six-dimensional feature vectors, each tagged with its correct class membership. The set was divided into a training set (two thirds of the training site pixels) and a testing set by stratified random sampling — stratified in terms of the eight classes. Pixels from the testing set are not used during network training and serve only to evaluate out-of-sample (generalization) performance accuracy (measured in terms of total classification accuracy) when the trained classifier is presented with novel data. The goal is to predict the correct class category for the test sample of pixels. In remote sensing classification tasks generalization performance can be more important than fast learning.

2.2 The Neural Network Classifier

In the most general terms, a neural network classifier attempts to approximate a pattern classification represented by a mapping

$$\mathcal{F} : \mathcal{R}^N \mapsto \mathcal{R}^C \tag{1}$$

where $\mathcal{R}^N$ denotes the N-dimensional input space and $\mathcal{R}^C$ the C-dimensional output space. The classification function $\mathcal{F}$ is not analytically known, but rather samples

$$S \equiv \left\{ s^1, \ldots, s^K \right\} \tag{2}$$

with

$$S^k \equiv (\mathbf{x}^k, \mathbf{y}^k) \qquad k = 1, \ldots, K. \tag{3}$$

$\mathbf{x}^k \in \mathcal{R}^N$ and $\mathbf{y}^k \in \mathcal{R}^C$ are generated by $\mathcal{F}$, i.e.,

$$\mathcal{F}(\mathbf{x}^k) = \mathbf{y}^k. \tag{4}$$

To approximate $\mathcal{F}$ we consider a single hidden layer feedforward network $\Phi_S : \mathcal{R}^N \mapsto \mathcal{R}^C$. Φ_S consists of a combination of transfer functions

$$\varphi_j : \mathcal{R} \mapsto \mathcal{R} \qquad j = 1, \ldots, J \tag{5}$$

$$\psi_c : \mathcal{R} \mapsto \mathcal{R} \qquad c = 1, \ldots, C \tag{6}$$

that are represented by hidden and output units, respectively, and weighted connections between the input, hidden and output units. The c-th element $c = 1, \ldots, C$ of Φ_S is given by

$$\Phi_S(\mathbf{X}, \mathbf{w}, \mathbf{u})_c = \psi_c \left(\sum_{j=0}^{J} u_{cj} \; \varphi_j \left(\sum_{n=0}^{N} w_{jn} \; x_n \right) \right) \tag{7}$$

where N denotes the number of input units, J the number of hidden and C the number of output elements. $\mathbf{x} = (x_0, x_1, \ldots, x_N)$ denotes the input vector augmented with a bias signal that can be thought of as being generated by a 'dummy' unit (with index zero) where ouput is clamped at 1. The w_{jn} represent input to hidden connection weights, and the u_{cj} hidden to output weights. The symbols $\mathbf{w}$ and $\mathbf{u}$ are convenient

short hand notations of the $(J(N+1)+C(J+1))$-dimensional vectors of the w_{jn} and u_{cj} network weights, respectively (including the biases), i.e., $\mathbf{w} \equiv (w_{01}, \ldots, w_{JN})$ and $\mathbf{u} \equiv (u_{01}, \ldots, u_{CJ})$. $\varphi_j(\cdot)$ and $\psi_c(\cdot)$ are differentiable non-linear transfer (activation) functions of, respectively, the hidden units $(j = 1, \ldots, J)$ and the output elements $(c = 1, \ldots, C)$.

Let us assume

$$\varphi_j(\cdot) = \varphi(net_j) = \frac{1}{1 + \exp(-net_j)} \equiv z_j \quad j = j, \ldots, J \tag{8}$$

with

$$net_j = \sum_{n=0}^{N} w_{jn}\ x_n \qquad j = 1, \ldots, J \tag{9}$$

and

$$\psi_c(\cdot) = \psi(net_c) = \frac{\exp(net_c)}{\sum_{c'=1}^{C} \exp(net_{c'})} \quad c = 1, \ldots, C \tag{10}$$

with

$$net_c = \sum_{j=0}^{J} u_{cj}\ z_j = \sum_{j=0}^{J} u_{cj}\ \varphi(net_j) \quad c = 1, \ldots, C. \tag{11}$$

This specification of the output transfer function, termed softmax transfer function [10], is motivated by the goal of ensuring that the network outputs can be interpreted as probabilities of class membership, conditioned on the outputs z_j $(j = 1, \ldots, J)$ of the hidden units (see [11, 238 pp.]). Without loss of generality, we can assume Φ_S to have a fixed, i.e., predetermined, topology. $N = 6$ representing the spectral bands to be utilized and $C = 8$ representing the a priori given classes, and $J = 14$ specified by [12]) with the assistance of a pruning technique to optimize the complexity of the network model in order to achieve the best generalization. Thus, the approximation Φ_S of $\mathcal{F}$ only depends on the learning samples S and the training algorithm that determines the parameters $\mathbf{w}$ and $\mathbf{u}$ from S, and the model specification.

3 The Network Training Approach

3.1 The Optimization Problem

The process of determining the optimal parameter values of the above network is called training or learning and may be viewed as a problem of minimizing a multivariate error function that depends on the network parameters. The function that is minimized in this study is the multiple cross-entropy error function defined as (see [11] for a derivation in a

Bayesian setting)

$$E(\mathbf{w},\mathbf{u}) = \sum_{k=1}^{K} E^k(\mathbf{w},\mathbf{u}) = -\sum_{k=1}^{K}\sum_{c=1}^{C} y_c^k \ \ln\left[\frac{\Phi(\mathbf{x}^k,\mathbf{w},\mathbf{u})_c}{y_c^k}\right]$$

$$= -\sum_{k=1}^{K}\sum_{c=1}^{C} y_c^k \ \ln\left\{\frac{1}{y_c^k}\ \frac{\exp\left[\sum_j u_{cj}\left(1+\exp\left(-\sum_n w_{jn}\ x_i^k\right)\right)^{-1}\right]}{\sum_{c'}\left[\sum_j u_{c'j}\left(1+\exp\left(-\sum_n w_{jn}\ x_i^k\right)\right)^{-1}\right]}\right\} \quad (12)$$

where Φ_c represents the c-th component of the actual network output as a function of $\mathbf{x}^k$ and the weight vectors $\mathbf{w}$ and $\mathbf{u}$, and may be interpreted as the network's estimate of the class membership probability. y_c^k is the target data which has a 1-of-C coding scheme so that

$$y_c^k = \delta_{cc'} \quad (13)$$

for a training pattern $\mathbf{x}^k$ from class c' where $\delta_{cc'}$ denotes the Kronecker symbol with

$$\delta_{cc'} = \begin{cases} 1 & \text{for } c = c' \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

The function $E(\mathbf{w},\mathbf{u})$ is a non-negative continuously differentiable function on the $(J(N+1)+C(J+1))$-dimensional parameter space, which is a finite dimensional closed bounded domain — and thus, compact. So $E(\mathbf{w},\mathbf{u})$ assumes its minimum value at $(\mathbf{w}^*,\mathbf{u}^*)$ on the weight domain. If the identifier output vectors $\{\mathbf{y}^k\}$ are chosen judiciously to be far apart and the examples for different classes are not too close together, then the minimum mapping with the solution parameter set will be successful in recognizing input feature vectors by class [13]. To solve for a set of locally minimizing weights, we use the differential calculus and put

$$\frac{\partial(\mathbf{w},\mathbf{u})}{\partial u_{cj}} = 0 \quad (15A)$$

and

$$\frac{\partial(\mathbf{w},\mathbf{u})}{\partial w_{jn}} = 0. \quad (15B)$$

Because these non-linear equations cannot be solved in closed form, we approximate a local minimum with an iterative minimization procedure.

3.2 The Gradient Search Strategy

In terms of search mechanisms, gradient based search is appropriate and simple to implement for cases such as $(15A)$ and $(15B)$ where it is known that the $E(\mathbf{w},\mathbf{u})$ is differentiable and bounded. Of course, if gradient search is used, we must be willing to accept locally optimal solutions. In general, only global search mechanisms such as genetic algorithm based ones, which are computationally intensive, may lead to globally optimal solutions (see, for example, [14]).

The idea behind gradient descent procedures is that $E(\mathbf{w}, \mathbf{u})$ being minimized is approximated locally by a non-quadratic function and this approximation function is minimized. For convenience, let us denote $\boldsymbol{\omega} = (\mathbf{w}, \mathbf{u})$, then $E(\boldsymbol{\omega})$ near the point $\omega(\tau)$ with $\tau = 1, 2, \ldots$ can be approximated by the truncated Taylor series

$$E(\boldsymbol{\omega}) \cong E(\boldsymbol{\omega}(\tau)) + (\boldsymbol{\omega} - \omega(\tau))^T \nabla E(\boldsymbol{\omega}(\tau)) \tag{16}$$

where $(\boldsymbol{\omega} - \omega(\tau))^T$ denotes the transpose of $(\boldsymbol{\omega} - \omega(\tau))$ and

$$\nabla E(\boldsymbol{\omega}(\tau)) = \left(\cdots \frac{\partial E(\boldsymbol{\omega}(\tau))}{\partial w_{jn}}, \cdots, \frac{\partial E(\boldsymbol{\omega}(\tau))}{\partial u_{cj}}, \cdots \right) \tag{17}$$

represents the local gradient at $\boldsymbol{\omega}(\tau)$. Note that the local gradient alone determines the minimization direction of the search process.

Gradient descent or steepest descent procedures work as follows:

(i) choose an initial vector $\boldsymbol{\omega}$ in parameter space and set $\tau = 1$,

(ii) determine a search direction

$$\mathbf{d}(\tau) = -\nabla E(\boldsymbol{\omega}(\tau)) \tag{18}$$

and a step size $\eta(\tau)$ by using the one-dimensional linear search, i.e., seeking a positive scalar $\eta = \eta(\tau)$ that minimizes the one-dimensional function

$$E(\boldsymbol{\omega}(\tau) + \eta(\tau)\, \mathbf{d}(\tau)) < E(\boldsymbol{\omega}(\tau)) \qquad \tau = 1, 2, \ldots \tag{19}$$

in the τ-th iteration. In the standard version utilized in this study, the line minimization is replaced by a fixed step size η.

(iii) Update the parameter vector

$$\omega(\tau + 1) = \omega(\tau) + \eta(\tau)\, \mathbf{d}(\tau) \qquad \tau = 1, 2, \ldots \tag{20}$$

(iv) if $dE(\boldsymbol{\omega}/d\boldsymbol{\omega} \neq 0$ then set $\tau = \tau + 1$ and go to (ii), else return $\boldsymbol{\omega}(\tau + 1)$ as the desired minimum.

There are two basic approaches to find the minimum of the global multiple class cross-entropy function (12), off-line learning and on-line learning. They differ in how often the weights are updated. The on-line (incremental or pattern-based) learning updates the weights after every single pattern s^k chosen at random from S. In contrast, off-line learning updates the weights after K^* patterns ($K^* \leq K$) have been propagated through the network. Epoch-based learning refers to the case of $K^* < K$ and batch learning to $K^* = K$.

3.3 The Backpropagation Procedure

At each step τ of the above minimization process partial derivatives of the cross-entropy function with respect to the $\mathbf{w}$- and $\mathbf{u}$-parameters have to be computed. The updating

formulas for the input to hidden connection parameters w_{nj} and the hidden to output connection parameters u_{jc} with a fixed step size η are

$$u_{cj}(\tau+1) = u_{cj}(\tau) + \eta \frac{\partial E(\boldsymbol{\omega})}{\partial u_{cj}(\tau)} \tag{21A}$$

$$w_{jn}(\tau+1) = w_{jn}(\tau) + \eta \frac{\partial E(\boldsymbol{\omega})}{\partial w_{jn}(\tau)}. \tag{21B}$$

There are various approaches for computing the derivatives in equation $(21A)$ and $(21B)$. Backpropagation (see [5]), i.e., propagating errors backwards through the network, is an attractive and computationally efficient technique for evaluating the required partial derivatives (see [11, 141 pp.]). The equations needed to implement training by backpropagating gradient descent errors on a digital computer — where the error function is the cross-entropy function (12), the hidden unit transfer function is the logistic and the output transfer function is the softmax — may be derived by utilizing the chain-rule of differential calculus. They are given as

$$u_{cj}(\tau+1) = u_{cj}(\tau) + \eta \left[\sum_{k=1}^{K^*} \delta_c^k \; z_j^k \right] \tag{22A}$$

and

$$w_{jn}(\tau+1) = w_{jn}(\tau) + \eta \left[\sum_{k=1}^{K^*} \delta_j^k \; x_n^k \right] \tag{22B}$$

where δ_c^k and δ_j^k denote the local error of the c-th output unit and the j-th hidden unit of the network, respectively, after the presentation of the k-th training pattern $(\mathbf{x}^k, \mathbf{y}^k)$, and are defined as

$$\delta_c^k \equiv \Phi(\mathbf{x}^k)_c - y_c^k \tag{23A}$$

and

$$\delta_j^k \equiv \varphi(net_j^k)\,(1 - \varphi(net_j^k)) \sum_{c=1}^{C} u_{cj} \; \delta_c^k. \tag{23B}$$

The calculation of the appropriate cross-entropy error function derivatives by backpropagation errors is clearly attractive. The major difference of the updating rules for the u_{cj}- and w_{jn}-parameters is the evaluation of the local errors. At the output units the error is a function of the desired and the actual output. For the hidden units the local errors are evaluated at the basis of the local errors at the output layer.

It should be noted that the updating formulas $(21A, B)$ and $(22A, B)$ have been written for off-line training. In the case of on-line training they have to be modified to

$$u_{cj}^k(\tau+1) = u_{cj}^k(\tau) + \eta \; \frac{\partial E^k(\boldsymbol{\omega})}{\partial u_{cj}^k(\tau)} \tag{24A}$$

$$w_{jn}^k(\tau+1) = w_{jn}^k(\tau) + \eta \; \frac{\partial E^k(\boldsymbol{\omega})}{\partial w_{jn}^k(\tau)} \tag{24B}$$

and

$$u_{cj}^k(\tau+1) = u_{cj}^k(\tau) + \eta \left[\delta_c^k \; z_j^k\right] \tag{25A}$$

$$w_{jn}^k(\tau+1) = w_{jn}^k(\tau) + \eta \left[\delta_j^k \; x_n^k\right] \tag{25B}$$

respectively, where δ_c^k and δ_j^k are defined by equation $(23A, B)$. The on-line version is not consistent with optimization theory, but nevertheless has been found to be superior to batch learning on real world problems that show a realistic level of complexity and have a training set that goes beyond a critical threshold (see [15], [16]). Both, off-line and on-line versions will be evaluated in Section 4 on the pixel-by-pixel classification problem described in Section 2.

Before doing so the on-line backpropagation training procedure utilized in the study may be summarized by performing the following three major steps.

Step 1: Feedforward Computation: Select a training pattern $(\mathbf{x}^k, \mathbf{y}^k)$ at random from the set of training samples and propagate $\mathbf{x}^k$ forward through the network using equations (8)–(11), thus generating hidden unit activations z_j and output unit activations $\Phi(\mathbf{x}^k, \mathbf{w}, \mathbf{u})_c$ based on current weight settings $\boldsymbol{\omega}^k(w^k, u^k)$.

Step 2: Updating the u_{cj}-Parameters: Compute the δ_c^k for all the output units $c = 1, \ldots, C$ with $(23A)$ to evaluate the required derivatives for u_{cj}-parameter updating according to $(25A)$.

Step 3: Updating the w_{jn}-Parameters: Backpropagate the deltas using $(23B)$ backward to the hidden layer to obtain δ_j^k for each hidden unit $j = 1, \ldots, J$ in the network classifier, and use $(25B)$ to update the required parameters.

In off-line training equation $(25A)$ is substituted by equation $(22A)$ in Step 2, and equation $(25B)$ by equation $(22B)$ in Step 3.

4 On-Line versus Batch Training: A Comparative Study

There is currently no consensus on the relative merits of the on-line and off-line (stochastic epoch-based and batch) versions of gradient descent backpropagation. We, thus, attempt to provide some evidence on how these training procedures differ with respect to the performance criteria:

- training time measured in terms of CPU seconds, and
- out-of-sample (generalization) total classification accuracy.

The supervised multispectral pixel-by-pixel classification problem as described in Section 2.1 serves as testbed for the evaluation, because it is known to pose a difficult real world classification problem (see [12]). The training data set consists of 1,640 pixels and the testing data set of 820 pixels characterized by six-dimensional feature vectors and stratified by eight a priori given classes. A network classifier, with six input units, fourteen

hidden units and eight outputs representing the a priori given classes of urban land cover, was used in this study along with logistic hidden and softmax output transfer functions. The multiple class cross-entropy function is the error function to be minimized.

The network was initialized with random weights in the range $[-0.1, 0.1]$. Learning was stopped if every element of the gradient vector had an absolute value less than 10^{-6}, a condition that is regarded as a realistic test for convergence on a minimum [17] or if the number of iterations exceeded 10^6. Then the test set was presented to the net.

Each experiment (i.e., a combination of training procedure with tuning the step size parameter η in order o do justice to each training procedure) was repeated 10 times, the net being initialized with a different set of random weights before each trial. To enable more accurate comparison, the classifier was initialised with the same 10 sets of random weights in each of the following experiments: on-line gradient descent error backpropagation with an epoch size of 300 training patterns randomly chosen from the training set and $\eta = 0.008$, epoch-based gradient descent error backpropagation with an epoch size of 600 training patterns randomly selected from the training set and $\eta = 0.0001$, and batch training with $\eta = 0.0008$. A systematic search was carried out to determine the optimal combination of training procedure and the step size η. All experiments were done on a SUN Ultra workstation.

Table 2 presents the results for the four experiments. The training times shown are CPU seconds in the sense that they exclude overheads such as scoring on the training and test sets and screen display of progress information. In-sample (out-of-sample) performance is measured in terms of the percentage of training (testing) pixels correctly classified. In addition, the multiple class cross-entropy error function values achieved are summarized. In each case, averages, ranges and standard deviations were calculated over the 10 simulations.

`Table 2 to be placed about here`

The convergence problems frequently cited for gradient descent error backpropagation became apparent for both epoch-based and on-line learning. They failed to converge within 100,000 iterations. This failure is due to the fixed step size which means that weight changes became extremely small as the gradient approaches zero. Although batch training could relatively quickly learn the training set (about 20 times faster than on-line learning, and about 50 times faster than stochastic epoch-based learning) it could not find a local minimum in less than 6,499 iterations. Convergence to a minimum was impossible except with very low η-values such as $\eta = 0.0008$ as reported in Table 2. Such low values make convergence very slow. This can be also observed from the learning curves displayed in Figure 1. To avoid cluttering the graphs Figure 1 shows the learning curves of batch, epoch-based (epoch sizes $K^* = 300, 600$) and on-line training averaged over all the trials.

`Figure 1 to be placed about here`

The results in Table 2 also indicate that differences in generalization are small and much smaller than those between time of learning the training set. Epoch-based training with epoch size of $K^* = 600$ slightly outperforms batch training by one percentage point and on-line training by 2.4 percentage points on average, and leads to more stable solutions as indicated by a lower standard deviation. But generalization performance can vary

between different trials of the same training procedure. Indeed, out-of-sample classification accuracy varies by up to 7.8 percentage points (batch training: 80.49–86.22, epoch-based training with epoch size $K^* = 600$: 83.41–86.59, epoch-based training with epoch size $K^* = 300$: 78.90–86.71, on-line training: 80.98–87.20). Thus, an interesting conclusion from the the comparative study is that better generalization performance is not the result of finding a lower multiple class cross-entropy error function value.

5 Concluding Remarks

On-line and off-line versions of gradient descent error backpropagation were analysed to train a single hidden layer neural network classifier with softmax output transfer functions on a multispectral pixels-by-pixel classification problem with a challenging level of complexity and a larger size of the training set. In contrast to current practice a multiple class cross-entropy error function has been chosen to be minimized. The results of Section 4 provide interesting empirical evidence that batch, epoch-based and on-line training procedures show marked differences in convergence behaviour and learning speed, but only smaller differences in generalization performance. An interesting conclusion from the comparison is that better generalization performance is not the result of inding a lower multiple class cross-entropy error function value.

If the goal is to maximise learning speed on a pixel-by-pixel classification problem, then simple modifications to the batch version of the gradient descent error backpropagation technique such as the adaptive choice of the step size parameter η should be explored. Where high generalization is more important than faster learning and where the training set is very large as it is frequently the case in remote sensing applications, epoch-based training may be more effective than batch training — especially when many training examples possess redundant information in the sense that many contributions to the gradient are very similar. Epoch-based updating makes the search path in the parameter space stochastic when the input vector is drawn at random. In contrast, batch and on-line training appear to be more prone to fall into local minima as indicated from the rather high standard deviations of the cross-entropy error function values. The main difficulty with stochastic epoch-based learning is its apparent inability to converge on a minimum within the 100,000 iterations limit due to the fixed step size. Much work is still needed before epoch-based training can be utilized with the same confidence and ease that batch training currently provides.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge, MA: Cambridge University Press, 2 ed., 1992.

[2] G. G. Wilkinson, "Neurocomputing for Earth observation – recent developments and future challenges," in *Recent Developments in Spatial Analysis: Spatial Statistics, Behavioural Modelling, and Computational Intelligence* (M. M. Fischer and A. Getis, eds.), pp. 289–305, Heidelberg: Springer, 1997.

[3] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown function and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, pp. 551–560, 1990.

[4] H. White, "Connectionist nonparameter regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks*, vol. 3, pp. 535–550, 1990.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart, L. J. McClelland, and the PDP Research Group, eds.), vol. 1, pp. 318–332, Cambridge, MA: MIT Press, 1986.

[6] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[7] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biological Cybernetics*, vol. 59, pp. 257–263, 1988.

[8] R. Rojas, *Neural Networks. A systematic introduction*. Berlin Heidelberg: Springer, 1996.

[9] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. Chichester New York Brisbane Toronto Singapore: Wiley, 1993.

[10] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applications* (F. Fogelman Soulié and J. Hérault, eds.), pp. 227–236, New York: Springer, 1990.

[11] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[12] M. M. Fischer, S. Gopal, P. Staufer, and K. Steinnocher, "Evaluation of neural pattern classifiers for a remote sensing application," *Geographical Systems*, vol. 4, no. 2, pp. 195–224 and 243–244, 1997.

[13] C. G. Looney, "Stabilization and speedup of convergence in training feedforward neural networks," *Neurocomputing*, vol. 10, no. 1, pp. 7–31, 1996.

[14] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural Computation*, vol. 1, no. 4, pp. 425–464, 1989.

[15] Y. Le Cun, "Generalization and network design strategies," in *Connections in Perspective* (M. Pfeifer, ed.), pp. 143–155, Amsterdam: North-Holland, 1989.

[16] W. Schiffmann, M. Jost, and R. Werner, "Comparison of optimized backpropagation algorithms," in *European Symposium on Artificial Neural Networks* (M. Verleysen, ed.), (Brussels), pp. 97–104, 1993.

[17] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 244–256, 1978.

Table 1: Classes and Number of Training/Testing Pixels

	Description	Pixels Training	Pixels Testing
Class c_1	Mixed grass and arable farmland	167	83
Class c_2	Vineyards and areas with low vegetation cover	285	142
Class c_3	Asphalt and concrete surfaces	128	64
Class c_4	Woodland and public gardens with trees	402	200
Class c_5	Low density suburban areas	102	52
Class c_6	Densely built up urban areas	296	148
Class c_7	Water courses	153	77
Class c_8	Stagnant water bodies	107	54
Total Number of Pixels		1,640	820

Table 2: Gradient descent error backpropagation: On-line versus off-line learning*

Learning Mode	Time [CPU seconds]	Cross-Entropy Error Function Value	In-Sample Classification Accuracy [in %]	Out-of-Sample Classification Accuracy [in %]
On-line Learning	8.68±1.05	458.15±87.41	90.78±1.19	82.80±3.11
($\eta = 0.01$)	(0.70)	(50.38)	(0.75)	(2.02)
Epoch-Based Learning				
(epoch size $K^* = 300$;	20.82±3.45	196.33±12.49	94.87±0.40	83.56±3.91
$\eta = 0.008$)	(1.98)	(7.67)	(0.28)	(2.41)
Epoch-Based Learning				
(epoch size $K^* = 600$;	21.88±3.62	312.76±2.66	92.27±0.13	85.28±1.59
$\eta = 0.0001$)	(1.57)	(3.82)	(0.08)	(1.07)
Batch Learning	0.43±0.35	263.35±49.74	93.08±1.32	84.20±2.86
($\eta = 0.0008$)	(0.27)	(29.63)	(0.82)	(1.83)

* Performance values represent mean and variance (standard deviation in brackets) of 10 simulations differing in the initial weights. In-Sample Classification Accuracy: Percentage of training pixles correctly classified. Out-of-Sample Classification Accuracy: Percentage of test pixles correctly classified.

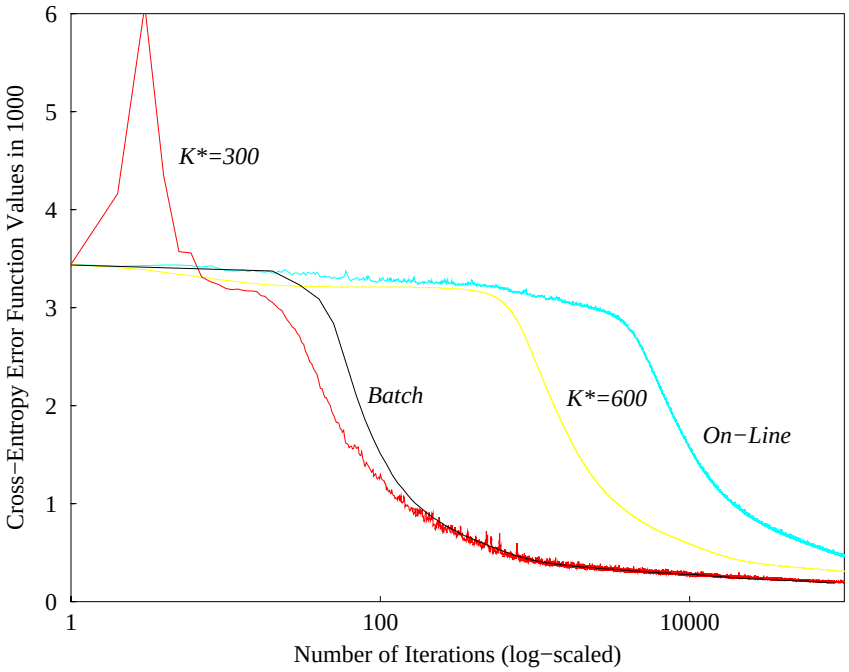

Figure 1: Gradient descent learning curves of on−line, epoch−based (K*=300, 600) and batch modes (average values of 10 simulations differing in initial weights)

List of Major Symbols Used

c	output unit (class) label
C	number of output units (classes)
$\delta_{cc'}$	Kronecker symbol
δ_c	local error of the c-th output unit
δ_j	local error of the j-th hidden unit
$\mathbf{d}$	direction vector
η	step size (scalar)
exp	exponential function
E	total error function
E^k	local error function
$\mathcal{F}$	analytically unknown mapping from input to output space
j	hidden unit label
J	number of hidden units
k	training pattern label
K	number of training patterns
K^*	epoch size
ln	logarithm to base e
n	input unit label
N	number of input units
net_c	net input to the c-th output unit
net_j	net input to the j-th hidden unit
ω^T	transposed parameter vector
$\boldsymbol{\omega}$	vector of all the network parameters
φ_j	transfer function of the j-th hidden unit
Φ	single hidden layer feedforward network function
Φ_c	c-th element of Φ
ψ_c	transfer function of the c-th output unit
$\mathcal{R}$	space of real numbers
S	set of training patterns
τ	iteration step
u_{cj}	connection weight from hidden unit j to output unit c
$\mathbf{u}$	vector of u_{cj}-parameters
$\mathbf{u}^*$	vector of optimal u_{cj}-patterns
w_{jn}	connection weight from input unit n to hidden unit j
$\mathbf{w}$	vector of w_{jn}-parameters
$\mathbf{w}^*$	vector of optimal w_{jn}-patterns
x_n	n-th component of $\mathbf{x}$
x_0	bias signal
$\mathbf{x}$	N-dimensional vector, element of input space X^N
$\mathbf{x}^k$	k-th training input pattern
y_c	c-th component of $\mathbf{y}$
$\mathbf{y}$	C-dimensional vector, element of output space Y^C
$\mathbf{y}^k$	k-th training output pattern
z_j	activation of the j-th hidden unit
$\in$	element of

$\equiv$	definition of
∇	gradient
∂	partial derivative

Petra Staufer received the MA and Ph.D. in geography in 1993 and 1997, both from the University of Vienna and a postgraduate degree in geoinformatics in 1993 from the Technical University of Vienna. Her research interests include neural network modelling and spatial data analysis in the context of geographical information processing (GIS) and remote sensing. She is an assistant professor in the Department of Economic and Social Geography at the Wirtschaftsuniversität Vienna.

Manfred M. Fischer, Professor and Chair at the Department of Economic and Social Geography at the Wirtschaftsuniversität Vienna as well as Director of the Institute for Urban and Regional Research at the Austrian Academy of Sciences, received the M.Sc. and Ph.D. degrees in geography and mathematics in 1974 and 1975, both from the University of Erlangen (Germany), the habilitation degree in 1982 from the University of Vienna. He is author of more than 150 publications. Dr. Fischer has been a leader in the field of spatial analysis and modelling for some 20 years. Specific research areas of current interest include issues of spatial function approximation and learning by computational neural networks, with applications to remote sensing and interregional telecommunication data.