SELF-ORGANIZING NEURAL NETWORKS IN FEATURE EXTRACTION

Mr. Markus Törmä
Institute of Photogrammetry and Remote Sensing
Helsinki University of Technology
Espoo, Finland
markus@mato.hut.fi

Commission II, Working group 3

KEY WORDS: Feature Extraction, Neural Networks, Classification

ABSTRACT

Due to large datavolumes when remote sensing or other kind of images are used, there is need for methods to decrease the volume of data. Methods for decreasing the feature dimension, in other words number of channels, are called feature selection and feature extraction. In the feature selection, important channels are selected using some search technique and these channels are used for current problem. In the feature extraction, original channels are transformed to lower dimensional channels and these are used for problem. Widely used feature extraction method is Karhunen-Löwe transformation. In this study Karhunen-Löwe transformation is compared to transformation made by Kohonen self-organizing feature map. Tests made using artificially generated datasets show that the differences between compared methods are small.

1. INTRODUCTION

Usually remote sensing instruments carry out measurements using several areas of the electromagnetic spectrum. As a result, image provided by a remote sensing instrument consist of several spectral channels. The number of the channels can be seven like in LANDSAT TM-image, but it can go as high as several hundred when spectrometers (e.g. AVIRIS, 224 channels) are used. Important step in data processing before e.g. land use classification is to find relevant channels for the current problem, so that feature dimension would decrease.

We can choose relevant channels using knowledge about spectral properties of the targets represented in the image. For example, if we want to separate land areas from water areas we can use LANDSAT TM channel 4, because the reflectance of water is nearly zero on the near-infrared part of the spectrum. But usually in the more complicated problems we do not have this kind of a priori information, or it is quite time consuming to utilize a priori information to channel selection. In this case, we can perform mathematical feature selection.

The structure of this paper is as follows: chapter 2 represents different approaches for the feature selection and chapter 3 one of these methods, Karhunen-Löwe transformation, called also principal component analysis, is reviewed. In chapter 4 self-organizing neural network called Kohonen self-organizing feature map (SOM) is presented and its use in the feature selection is discussed. Chapter 5 presents experiments made for comparing Karhunen-Löwe transformation and SOM and chapter 6 discusses about results. Finally, chapter 7 represents conclusions.

2. FEATURE SELECTION

The methods for feature selection are divided into two groups: feature selection in feature space and feature selection in transformed space. Feature selection in feature space is made by choosing those features, which contain useful information and deleting those features which contain redundant or unnecessary information. In other words, we have all features in featureset Y and we seek the best subset of Y called X. The best subset of Y is chosen by maximizing some criterion function. In the ideal case this best subset maximizes the probability of correct classification compared to other possible combinations. Usually feature selection in the feature space is simply called feature selection. Feature selection in the transformed space is made by transforming the original measurement vector y to lower dimensional feature vector x. In this case the decrease of redundant and unnecessary information depends on used transformation. Transformation can be any kind of vector function of y, but usually linear transformations are used. Linear transformation can be written as

$$x = Ay, \tag{1}$$

where A is transformation matrix. The problem is how to determine a good matrix A, so that useful information is not destroyed. Feature selection in transformed space is also called feature extraction.

The best subset of all features in the feature selection is chosen using criterion function and search algorithm. Criterion function J to be maximized can based on probability of error, interclass distance, probabilistic distance, probabilistic dependence or entropy. The idea in all these criterion functions is to measure the separability of classes. The best subset could be found by evaluating the value of criterion function for all possible combinations, but this is usually too time consuming. For example, if the dimension of original feature space is 20 and the dimension of transformed feature space is 10, 184756 different cases should be evaluated. The only optimal search algorithm, which implicitly evaluates all possible subsets is called branch and bound-algorithm. Other, nonoptimal algorithms are for example sequential forward selection and sequential backward selection.

The problem in feature extraction is to choose good transformation matrix A. Usually transformation is limited to linear form. Matrix A can be determined by using same criterion functions as in feature selection. In this case optimal matrix A is chosen so that criterion function J(Ay) is maximized. Usually maximization is only possible by using numerical optimization and it is quite time consuming. Another alternative in feature extraction is use Karhunen-Löwe transformation which is discussed in next chapter (Devivjer, 1982).

3. KARHUNEN-LÖWE TRANSFORMATION

Karhunen-Löwe transformation is based on discrete Karhunen-Löwe expansion. In this transformation original information is preserved as well as possible by approximating original feature vector using several terms of expansion.

3.1 Karhunen-Löwe expansion

We have d-dimensional random vectors y and we can represent these vectors without error by the summation of d linearly independent vectors as

$$y = \sum_{i=1}^{d} x_i \phi_i, \tag{2}$$

where x_i are the coefficients of the basis vectors ϕ_i. Basis vectors are orthonormal:

$$\phi_i^T \phi_j = \begin{bmatrix} 1 & for\ i{=}j \\ 0 & for\ i{\neq}j. \end{bmatrix} \tag{3}$$

In this case, the components of vector x can be computed by

$$x_i = \phi_i^T y, \quad i=1,\ldots,d. \tag{4}$$

So, x is orthonormal transformation of y. If we want to decrease the dimensionality of the feature space ($d>m$) we simply do not use all basis vectors ϕ_i but select best basis vectors. The best basis vectors minimize mean squared error between original vector y and its approximation y. The mean squared error can be written as

$$\varepsilon = \sum_{i=m+1}^{d} \phi_i^T \Sigma \phi_i. \tag{5}$$

We notice that discarded basis vectors affect to error. Matrix Σ is autocorrelation matrix of y, also covariance matrix can be used. Optimum choice for basis vectors is those which satisfy

$$\Sigma \phi_i = \lambda_i \phi_i, \tag{6}$$

or eigenvectors of Σ. Combining equations (5) and (6) mean squared error becomes

$$\varepsilon = \sum_{i=m+1}^{d} \lambda_i. \tag{7}$$

Mean squared error is minimized when discarded eigenvectors correspond to smallest eigenvalues (Devivjer, 1982)(Fukunaga, 1990).

3.2 Summary of Karhunen-Löwe transformation

Presented results can be put on algorithmic form:

1. Compute the correlation or covariance matrix Σ of y.
2. Compute the eigenvalues and corresponding eigenvectors of Σ. Normalize the eigenvectors.
3. Form the transformation matrix A from the m eigenvectors corresponding to the largest eigenvalues of Σ.
4. Compute transformed feature vectors using equation (1) (Tou, 1974).

4. SELF-ORGANIZING NEURAL NETWORKS

An artificial neural network (referred as neural network after this) is a parallel, distributed signal or information processing system, consisting of simple processing elements, also called nodes. Processing elements can possess a local memory and carry out localized information processing operations. In the simplest case processing element sums weighted inputs and passes the result through nonlinear transfer function. Processing elements are connected via unidirectional signal channels called connections. The connections are usually weighted and those weights are adapted during training of the network. Learning of the network is based on the adaptation of the weights.

The neural network models can be characterized using their properties like connection topologies, processing element capabilities, learning algorithms, problem solving capabilities etc., and models can differ greatly. Neural networks try to imitate a biological nervous system and its properties like memory and learning. (Lippmann, 1987).

4.1 Self-organizing feature map

Kohonen Self-Organizing feature Map (SOM) is neural network which is trained using competitive learning. Basic competitive learning means that competition process takes place before each cycle of learning. Competition process means that a winning processing element is chosen by some criteria. After the winning processing element is chosen, its weight vector is adapted according to the used learning law (Hecht-Nielsen, 1990).

SOM creates topologically ordered mappings between input data and processing elements of the map. Topologically ordered means that if two inputs are similar, then the most active processing elements responding to inputs are located near each other in the map and the weight vectors of the processing elements are arranged to ascending or descending order, $w_i < w_{i+1}$ all i or $w_i > w_{i+1}$ all i (this definition is valid for 1-dimensional SOM). Motivation behind SOM is that some sensory processing areas of brain are ordered in similar way (Kangas 1994).

SOM is usually represented as a two dimensional matrix (also other dimensions can be used) of processing elements. Each processing element has its own weight vector and learning of SOM is based on the adaptation of these weight vectors (Kohonen, 1990).

The processing elements of the network are made competitive in a self-organizing process and the winning processing element whose weights are updated is chosen by some criteria. Usually this criteria is to minimize Euclidean distance between input vector and weight vector. SOM differs from basic competitive learning so that instead of adapting only the weight vector of the winning processing element also weight vectors of neighboring processing elements are adapted. First, the size of the neighborhood is large making rough ordering of SOM possible and size is decreased as time goes on. Finally, in the end only a winning processing element is adapted making the fine tuning of SOM possible. The use of neighborhood makes topologically ordering process possible and together with competitive learning makes process nonlinear (Kohonen, 1990).

The basic idea is that the weight vectors of the processing elements approximate the probability density function of the input vectors. In other words, there are many weight vectors close to each other in high density areas of the density function and less weight vectors in low density areas.

Mathematically speaking, SOM learns a continuous topological mapping f: $B \subset \mathbb{R}^d \rightarrow C \subset \mathbb{R}^m$. This is nonlinear mapping from d-dimensional space of input vectors to m-dimensional space of SOM. Strict mathematical analysis exists only in simplified cases of SOM. It has been proved difficult to express the dynamic properties of SOM to mathematical theorems (Kohonen, 1990).

4.2 SOM learning algorithm

1. Initialize weights to small random values.
2. Choose input randomly from dataset.
3. Compute Euclidean distance to all processing elements.
4. Select winning processing element j with minimum distance. Winning processing element is also called best matching unit (BMU).
5. Update weight vectors to processing element j and its neighbors using following learning law. The learning law moves weight vector toward input vector.

$$w_j(t+1) = w_j + \alpha(t)(x(t) - w_j(t)), \tag{8}$$

   where gain term α (0<α<1) decreases in time. Also, size of neighborhood decreases in time (only those weight vectors of processing elements are updated, which belong to the neighborhood). Here processing element belongs to the neighborhood, if $d_C(j,i) \leq T$, where d_C is the Chebyshev distance, j is the winning processing element, i is another processing element and T is the threshold which decreases in time.
6. Go to step 2 or stop iteration when enough inputs are presented. (Lippmann, 1987)

4.3 SOM in feature extraction

SOM is usually arranged as a two dimensional matrix (also other dimensions can be used) of processing elements. As a result of learning phase, those processing elements which are spatially close to each other respond in similar way to the presented input pattern. In other words, map is topologically ordered. Also, SOM makes nonlinear transformation from d-dimensional inputspace to m-dimensional mapspace. Mapspace is defined by he coordinates of the processing elements. All these properties are useful in feature extraction.

In feature extraction, original feature vector is presented to SOM and its winning processing element and its mapcoordinates are searched. These mapcoordinates could be used as a transformed features, but usually there is limited number of processing elements and many different inputvectors get same coordinates. This means that if the density function of inputvectors is continuous, the density function of transformed vectors is not continuous.

Better way to make transformation is to use distances computed during the search of the winning processing element. There are two alternatives:

A. Weighted mean of mapcoordinates are computed using inverse distances from inputvector to weightvectors as weights. These mean values are used as a transformed vector.
B. The coordinates of BMU are searched and distances computed from inputvector to BMU (d_A) and the second closest weightvectors (d_B) in row and column direction. Transformed value is coordinate of BMU ±

$d_A / (d_A+d_B)$. Plus is used, if the coordinate of second closest weightvector is greater than the coordinate of BMU.

5. EXPERIMENTS

So that Karhunen-Löwe transformation could be compared to SOM, following experiments were carried out. Datasets were generated using random number generator and feature extraction was made. Transformed datasets were classified and classification error estimated. This process was repeated 50 times using different datasets. Criterion to compare results was minimize the classification error.

5.1 Classification

Classifications were made using the Bayes decision rule for minimum error. A posteriori probability $P(x \mid \omega_j)$ is calculated from a'priori probability P_j and the conditional density function (CDF) $p(x \mid \omega_j)$ using the Bayes theorem

$$P(x|\omega_j) = \frac{p(x|\omega_j)\,P_j}{\sum_{i=1}^{c} p(x|\omega_i)\,P_i}. \qquad \textbf{(9)}$$

where c is the number of classes. When x is to be classified, the a posteriori probabilities are determined for each class and x is assigned to the class with the maximum a posteriori probability.

The value of CDF $p(x \mid \omega_j)$ determines how closely sample x belongs to class ω_j. It is estimated using a nonparametric estimation method called k-nearest neighbor estimation. This method estimates the CDFs locally using small number neighboring samples. The k-nearest neighbor estimate of the CDF of class i is

$$p(x|\omega_i) = \frac{k}{n_i\,v}, \qquad \textbf{(10)}$$

where k is number of neighboring samples, n_i is number of samples in class i and v is the volume of hypersphere which radius is distance between sample x and its kth neighbor (Devivjer, 1982)

5.2 Error estimation

The probability of error is the most effective measure of the performance of a classification system. In practise, the probability of error must be estimated from the available samples. First a classifier is designed using training samples and then it is tested using test samples. The percentage of misclassified test samples is taken as an estimate of the probability of error.

The probability of error is estimated using resubstitution (RES) and leave-one-out (LOO) estimation methods. The resubstitution method uses the same set of samples to train and test the classifier. Because training set and testset are same set, errors estimated using this method are unreliable, but can be used together with other error estimation methods like leave-one-out method. In leave-one-out method each sample is used for train and test, although not at the same time. The classifier is trained using (n-1) samples and tested on the remaining sample (n is the total number of samples). This is repeated n times with different training sets of size (n-1). The error estimate is the total number of misclassified samples divided by n (Devivjer, 1982).

5.3 Datasets

Three different datasets were used. The original dimension of dataset was 8 and number of classes 2. Datasets were generated using random number generator. Number of samples per class was equal to dimension times N, where N = 5, 10 or 100. Then generated samples were classified and classification errors estimated. This was repeated 50 times and each time samples were generated independently. Finally, the statistical descriptors, mean value, median value, standard deviation, minimum and maximum values were computed from classification errors.

In the first dataset, called II, mean of the first class was $M_1 = [0...0]^T$ and mean of the second class was $M_2 = [2.56\ 0...0]$. Covariance matrices for both classes were identity matrices I. In other words, class means differ and covariances are same. Bayes error is about 10%.

In the second dataset, called I4I, mean of both classes were $M_1 = M_2 = [0...0]^T$. Covariance matrix for first class was identity matrix I and for second class 4I. In other words, class means are same and covariances differ. Bayes error is about 9%.

In the third dataset, called IA, mean of the first class was $M_1 = [0...0]$ and mean of the second class was $M_2 = [3.86\ 3.10\ 0.84\ 0.84\ 1.64\ 1.08\ 0.26\ 0.01]$. Covariance matrix for first class was identity matrix I and for second class the diagonal values were $\Sigma = [8.41\ 12.06\ 0.12\ 0.22\ 1.49\ 1.77\ 0.35\ 2.73]$. In this case, both class means and covariances differ. Bayes error is about 1.9% (Fukunaga, 1990).

5.4 Parameters of algorithms

The transformation matrix in Karhunen-Löwe transformation was based on the eigenvectors of the covariance matrix of dataset.

Parameters of SOM were the size of map, size of neighborhood, number of inputvectors presented to algorithm, starting value of α and its decreasing method. In these experiments different sizes of map were used, sizes 9x9, 11x11, 7x11 and 19x19 processing elements. Size of neighborhood in the beginning was more than half of the size of map and decreased linearly until only one weightvector, BMU, was updated. Number of inputvectors presented to the algorithm varied also, it was at least 500 inputvectors per processing element. When the size of map was 9x9 or 7x11 processing elements, the number of inputvectors was 50000 or

100000. When the size of map was 11x11 processing elements, number of inputvectors was 70000 or 140000. When the size of map was 19x19 processing elements, number of inputvectors was 400000.

Parameter for classifier was the number of neighboring samples, k, used to calculate the value of the conditional density function of the class. The bias of density estimate depends on value of k, and value must be determined experimentally. In these experiments value of k varied between 1 and 10.

6. RESULTS

Experiments consisted of about 6000 individual testruns. This is short overview to results.

6.1 Dataset II

First used feature extraction method was Karhunen-Löwe transformation. Classification error was about 33% when dataset II with number of samples in class 40 (N=5, d=8) was classified. In this case RES error varied between 20% - 30% (deviation 4.5% - 5.5%) with varying k 2 - 10 and LOO error varied between 39% - 36% (deviation 7% - 8%) with varying k 2 - 10. When number of samples in class was increased (N=10) classification error was about 34%, RES error varied between 21% - 31% (deviation 3.5% -5%) and LOO error varied between 40% - 36% (deviation 6% - 7.5%). When number of samples in class was again increased (N=100) classification error decreased to about 28%, RES error varied between 19 - 28% (deviation 1% - 1.3%) and LOO error varied between 35% - 30% (deviation 1.6% - 1.9%). Transformed two features contained on an average 36.5% from original information (percentage of two largest eigenvalues from all eigenvalues) when N=5, 33.7% when N=10 and 29.1% when N=100.

The results of SOM case A were independent from number of samples presented to algorithm and the size of map, variation between different combinations were small. When N=5, the average classification error varied between 37% - 39%, RES error varied between 20% - 33% (deviation 4.5% - 6.5%) and LOO error varied between 45% - 40% (deviation 7% - 10%). When number of samples in class was increased (N=10), the average classification error varied between 34.5% - 36%, RES error varied between 20% - 33% (deviation 2.5% - 4.8%) and LOO error varied between 42% - 37% (deviation 4.5% - 6.5%). When number of samples was again increased (N=100), the average classification error varied between 34% - 35%, RES error varied between 21% - 32% (deviation 0.9% - 1.8%) and LOO error varied between 41% - 36% (deviation 1.8% - 2.8%).

Also, the results of SOM case B were independent from number of samples presented to algorithm and the size of map, variation between different combinations were small. When N=5, the average classification error varied between 37% - 38.5%, RES error varied between 20% - 35% (deviation 4% - 6.5%) and LOO error varied between 45% - 41% (deviation 6.5% - 10%). When number of samples in class was increased (N=10), the average classification error varied between 33.5% - 36%, RES error varied between 21% - 33% (deviation 3% - 4.3%) and LOO error varied between 43% - 38% (deviation 4.8% - 6.2%). When number of samples was again increased (N=100), the average classification error varied between 32% - 33%, RES error varied between 20% - 31% (deviation 0.5% - 1.2%) and LOO error varied between 38% - 35% (deviation 1.5% - 2.0%).

6.2 Dataset I4I

Again, first used feature extraction method was Karhunen-Löwe transformation. Classification error was about 45% when dataset I4I with number of samples in class 40 (N=5, d=8) was classified. In this case RES error varied between 24% - 40% (deviation 4.1% - 5.1%) with varying k 2 - 10 and LOO error varied between 51% - 48% (deviation 6.6% - 8%) with varying k 2 - 10. When number of samples in class was increased (N=10) classification error was about 44%, RES error varied between 25% - 38% (deviation 3.2% -4.4%) and LOO error varied between 49% - 47% (deviation 5.2% - 6.3%). When number of samples in class was again increased (N=100) classification error decreased to about 43%, RES error varied between 24 - 38% (deviation 1% - 1.3%) and LOO error varied between 48% - 46% (deviation 1.3% - 1.8%). Transformed two features contained on an average 36.4% from original information when N=5, 33.0% when N=10 and 27.4% when N=100.

SOM case A with N=5, the average classification error varied between 45% - 47.5%, RES error varied between 24% - 42% (deviation 4.5% - 6.5%) and LOO error varied between 55% - 50% (deviation 6.5% - 10%). When number of samples in class was increased (N=10), the average classification error varied between 45% - 46%, RES error varied between 24% - 41% (deviation 3% - 4.5%) and LOO error varied between 52% - 49% (deviation 4.5% - 6.8%). When number of samples was again increased (N=100), the average classification error varied between 44.5% - 45.5%, RES error varied between 24% - 41% (deviation 1.0% - 1.5%) and LOO error varied between 50% - 49% (deviation 1.5% - 2.2%).

SOM case B with N=5, the mean classification error varied between 45% - 47.5%, RES error varied between 24% - 41% (deviation 4% - 6%) and LOO error varied between 55% - 50% (deviation 6.5% - 9%). When number of samples in class was increased (N=10), the mean classification error varied between 45% - 46%, RES error varied between 24% - 40% (deviation 2.5% - 4.5%) and LOO error varied between 51% - 49% (deviation 4% - 6.9%). When number of samples was again increased (N=100), the mean classification error varied between 43% - 44%, RES error varied between 24% - 40% (deviation 0.8% - 1.3%) and LOO error varied between 49% - 47% (deviation 1.5% - 2.6%).

6.3 Dataset IA

Classification error with features extracted using

Karhunen-Löwe transformation was about 10% when dataset IA with number of samples in class 40 (N=5, d=8) was classified. In this case RES error varied between 7% - 10% (deviation 2.8% - 3.5%) with varying k 2 - 10 and LOO error varied between 13% - 10% (deviation 3.5% - 4.7%) with varying k 2 - 10. When number of samples in class was increased (N=10) classification error was about 10.5%, RES error varied between 7% - 10% (deviation 2.5% -3.3%) and LOO error varied between 13% - 11% (deviation 3.4% - 4.1%). When number of samples in class was again increased (N=100) classification error decreased to about 10%, RES error varied between 7% - 10% (deviation 0.5% - 0.8%) and LOO error varied between 12% - 10% (deviation 0.8% - 1.0%). Transformed two features contained on an average 46.6% from original information when N=5, 44.8% when N=10 and 43.0% when N=100.

SOM case A with N=5, the average classification error varied between 13% - 14.5%, RES error varied between 8% - 13% (deviation 4.5% - 6.5%) and LOO error varied between 16% - 13% (deviation 6.5% - 10%). When number of samples in class was increased (N=10), the average classification error varied between 11.5% - 12.5%, RES error varied between 8% - 11% (deviation 3% - 4.5%) and LOO error varied between 14% - 12% (deviation 4.5% - 6.8%). When number of samples was again increased (N=100), the average classification error was 10.5% in all cases, RES error varied between 8% - 10% (deviation 1.0% - 1.5%) and LOO error varied between 13% - 11% (deviation 1.5% - 2.2%).

SOM case B with N=5, the average classification error varied between 13.5% - 15.5%, RES error varied between 7% - 14% (deviation 2.8% - 5.8%) and LOO error varied between 17% - 14% (deviation 3.8% - 5.6%). When number of samples in class was increased (N=10), the average classification error varied between 12% - 13.5%, RES error varied between 8% - 12% (deviation 1.8% - 3.2%) and LOO error varied between 16% - 13% (deviation 2.4% - 4.2%). When number of samples was again increased (N=100), the average classification error varied between 11% - 12.5%, RES error varied between 8% - 12% (deviation 0.6% - 1.2%) and LOO error varied between 13% - 12% (deviation 1.0% - 1.5%).

7. CONCLUSIONS

The classification errors using different feature extraction methods were quite same, differences were small. Main difference was when number of samples per class was small, then the classification errors with features computed using Karhunen-Löwe transformation were smaller than the classification errors with features computed using SOM. Another difference was when dataset II was used, then the classification errors with features computed using Karhunen-Löwe transformation were also smaller. When the number of samples per class increased, the difference decreased. In these cases feature extraction methods based on SOM can be used, because computational time is shorter.

8. REFERENCES

Devivjer, P., Kittler, J., 1982. Pattern Recognition - A Statistical Approach. Prentice-Hall.

Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition. Academic Press, pp. 399-424.

Hecht-Nielsen, R., 1990. Neurocomputing. Addison-Wesley, pp. 63-65.

Kangas, J., 1994. On the analysis of pattern sequences by self-organizing maps. Thesis for the degree of Doctor of Technology, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Kohonen, T., 1990. The self-organizing map. Proceedings of IEEE, 78(9), pp. 1464-1480.

Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing Magazine, 4(2), pp. 4-22.

Tou, J., Gonzales, R., 1974. Pattern Recognition Principles. Addison-Wesley, pp. 271-282.