WARSAW UNIVERSITY OF TECHNOLOGY
FACULTY OF MATHEMATICS
AND INFORMATION SCIENCE

Neural Networks

Lecture 14

Counter-propagation Neural Networks for Image Compression

The data compression

The data compression yields to the data reduction which is to be send or stored usually with the possibility of its full reproduction (decompression).

Image data compression is used to encode large amounts of image data for transmission over a limited-capacity channels. Recently, neural networks algorithms have been developed for data compression, yielding superior performance over classical techniques.

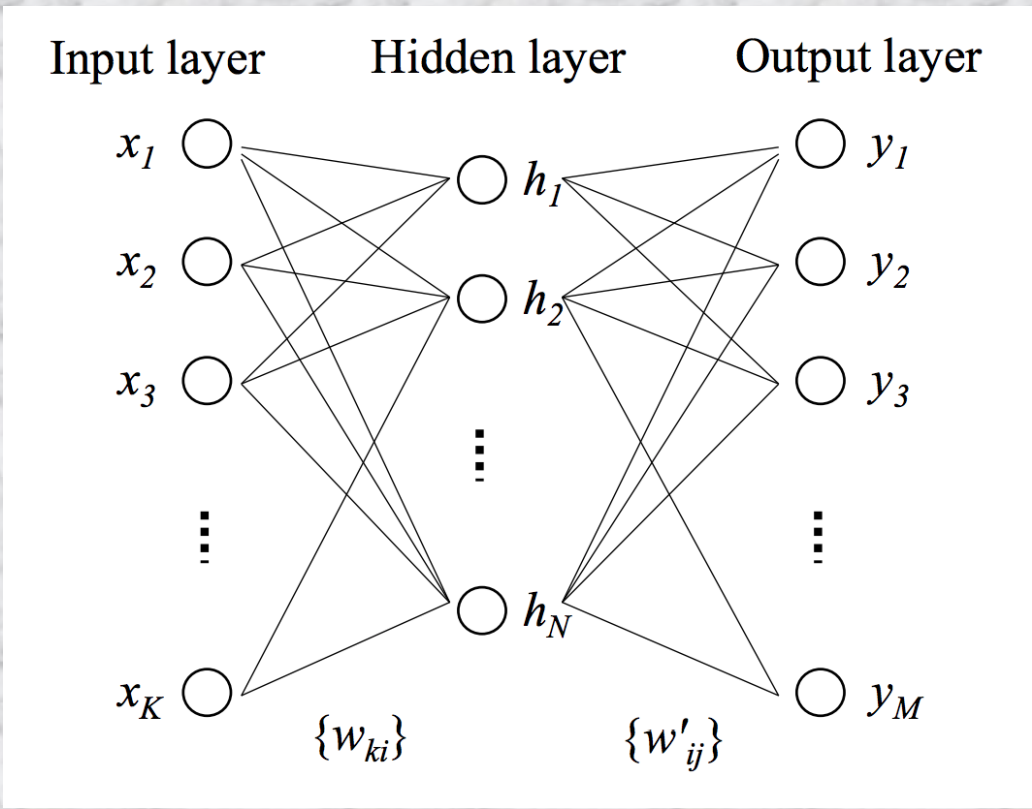
The data compression

For the efficient compression, the image data are passed through a network producing the binary vectors as their compressed version. After decompression the image data and output data are expected to be very close.

The compression ratio depends greatly on the tolerated amount of error. Algorithm is made of two major steps, namely network training (or learning) and processing (data compression and decompression)

The data compression

The model of multilayer neural network for a data compression



The data compression

The input data consist of *N* vectors composed of *n*-elements (i.e. *n* = 64) are sequentially applied to the input layer. The number of elements of an input vector is equal to the number of elements in this layer (i.e. 8x8) and also equal to number of pixels in the sample of a picture.

Assuming that each pixel has a gray level between 0 and 255 (0 – means absolutely dark, and 255 means absolutely white) the pixel can be stored with one byte (8 bits) of information.

The data compression

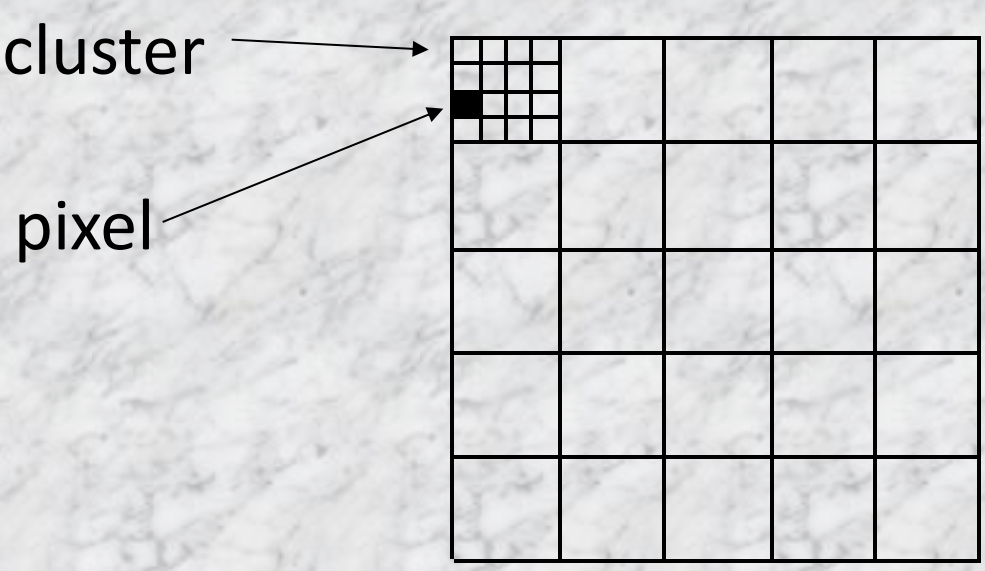
The hidden layer is composed from q neurons ($q<<n$, i.e. $q=16$) and realizes data compression. In the above mentioned example we get reduction factor equal to 4. The input layer and hidden layer can be treated as a transmitter.

In the output layer there is n neurons again, and the decompression has place reproducing the original 64 elements pattern samples.

The output layer can be treated as a receiver. Usually the most popular system of a network learning is the is the back-propagation algorithm.

The data compression

The input picture is divided into clusters composed of pixels.



The data compression

It was shown that neural network, called the counter-propagation network, can perform for some applications even better than the back-propagation one. The architecture of the ***counter-propagation.*** network is a combination of the self-organizing map of Kohonen and the outstar structure of Grossberg.

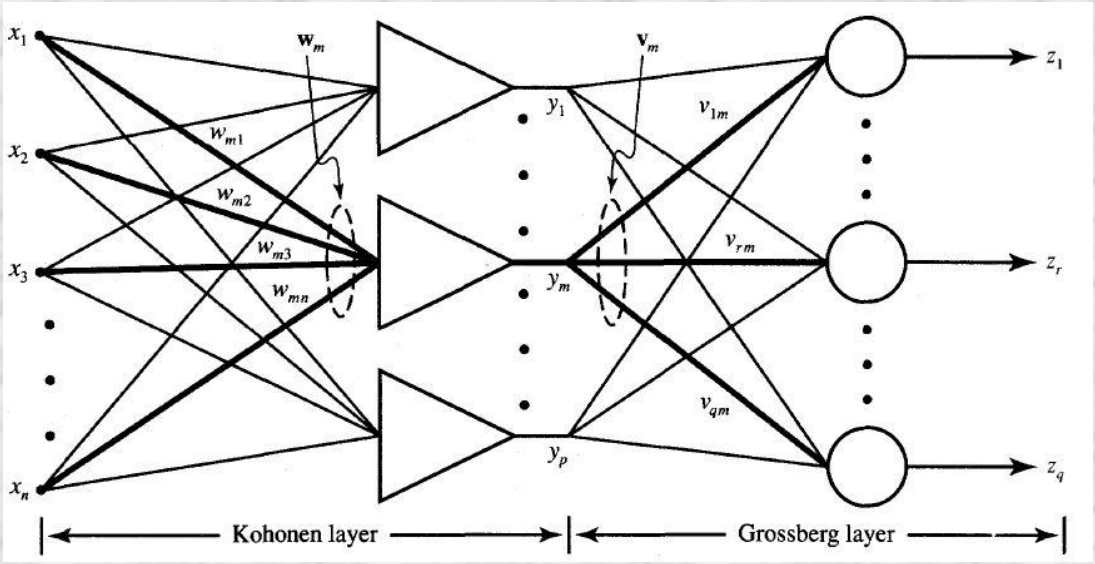
The data compression

The **counter-propagation** (CPN) network is composed of two feedforward layers: the Kohonen layer and the Grossberg layer. The number of neurons in these layers can differ. There is a total interconnection between layers; every Kohonen neuron is connected with each Grossberg neuron. The Kohonen layer is connected to the network input, and is operating under the rule ***winner takes all*** (WTA).

CPN is useful in pattern mapping and associations, data compression, and classification.

The data compression

The architecture of counter-propagation neural network



It has two layers similar to feedforward networks
BUT it has different learning strategy

The data compression

For the normalized vector input signal $\boldsymbol{X} = [x_1, x_2, \ldots, x_k]$ each Kohonen neuron K_i obtains at its input the weighted sum

$$NET_i = \sum_{j=1}^{k} w_{ij} x_j \qquad \text{for } i = 1,2,\ldots,N,$$

where

N is the number of Kohonen neurons,

w_{ij} is the connection value between *jth* input node and *ith* Kohonen neuron,

x_j has the value of normalized component of the real input.

The data compression

The WTA winner is the neuron with the maximal value of NET_i. This neuron generates on its output the signal equal to 1 blocking the rest of neurons and activates the Grossberg neurons via the weights v_{ij} .

The data compression

Grossberg neurons G_i obtain the weighted sum

$$G_i = \sum_{j=1}^{N} v_{ij} K_j$$

v_{ij} is the connection value between j^{th} Kohonen neuron and i^{th} Grossberg neuron,,

N is the number of Kohonen neurons,

K_j is the output value of j^{th} Kohonen neuron ($K_j \in \{0,1\}$).

The data compression

Because only one of Kohonen neurons is active (output signal equal to 1) the Grossberg layer generates at output the vector signal equal to the weight vector $\boldsymbol{V}_k$ determined by the weights values between the Kohonen neuron *k* and Grossberg neurons.

The Kohonen layer operates as a vector classifier, activating the neuron representing the class of similar input vectors. The Grossberg layer generates the vector ***Y*** associated with this class.

The data compression

Every Grossberg neuron realizes the outstar function. When the winning neuron in the Kohonen layer has the number K_j, the output vector signal from the Grossberg layer is the binary value of K_j (for example, for $K_j = 5$ the output vector $\boldsymbol{Y}$ = [1 0 1 0 0 0 0 0]).

The number of Grossberg neurons depends on the number of Kohonen neuron. For $0 < \boldsymbol{N} \leq 256$ we have $\boldsymbol{M} = 8$ (one byte) Grossberg neurons and for $\boldsymbol{N} > 256$ (but of course $\boldsymbol{N} \leq 65536$), $\boldsymbol{M} = 16$ (two bytes). Generally, for $\boldsymbol{N} \in < 2^{8n}; 2^{8(n+1)} >$ we have $\boldsymbol{M} = 8(n+1)$, $n \in \boldsymbol{N} \cup \{0\}$ and it needs $n+1$ bytes to write a vector to a file.

The data compression

Learning

Before compression it is necessary to learn the network. The learning procedure can be based in practice on any learning image (recommended are images with varied gray levels). The learning picture is segmented into clusters composed of 2*2, 3*3 or 4*4 pixels. Each pixel has a gray level between 0 and 63.

Dividing a picture into *n* rows and *m* columns generates *n***m* clusters. Each cluster is represented by a *k*-dimensional vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_k]$ where *k* is the number of pixels in the cluster. The vector $\boldsymbol{X}$ has to be normalized before use.

The data compression

Elements of $\boldsymbol{X}$ have the values depending on the pixels gray level. The Grossberg layer generates the number of the winning Kohonen neuron. This information defines the cluster number, which is able to reconstruct the components of the vector $\boldsymbol{X}' \cong \boldsymbol{X}$.

The data compression

At the first stage the weights for **the Kohonen layer** are generated. The method consists of the division of learning vectors into classes composed of similar vectors. At the beginning there is only one Kohonen neuron. Its weights are assumed to be identical with the components of the first learning vector $\boldsymbol{W}_1 = \boldsymbol{X}^1$. Next, for each following learning vector $\boldsymbol{X}^i$ the products $\boldsymbol{X}^i\boldsymbol{W}_k$ are calculated where $\boldsymbol{W}_k$, (k=1,2,...), are the input weights of every Kohonen neuron already formed.

The data compression

If $$(1 - X^i W_k) < B$$

where B ($B \in (0;1)$), defines the difference between the learning vectors, then i^{th} learning vector is included into the class k, and the weights of k^{th} neuron are modified.

If the above condition is not satisfied for any existing Kohonen neuron, the new Kohonen neuron is added with weights equal to the components of $\boldsymbol{X}^i$ representing this new class.

The algorithm is repeated until the last learning vector is included in the proper class.

The data compression

In **the Grossberg layer** we have the supervised learning procedure. For every Kohonen neuron the special vector $\boldsymbol{d}_i$, $i = 1,2,...,N$ is generated. This vector is the binary notation of the number of Kohonen neuron (for example for the neuron 6 we have $\boldsymbol{d}_6$ = [0 1 1 0 0 0 0 0].

The data compression

The weights $\boldsymbol{V}_k$ are calculated according to the outstar rule

$$v_{jk}(t+1) = v_{jk}(t) + \eta\left(d_j - v_{jk}(t)\right)$$

or its simplified version $v_{jk} = d_j$

where η is a learning coefficient,

v_{jk} is the connection value between k^{th} Kohonen neuron and *ith* Grossberg neuron.

The data compression

Data compression

The picture for compression is divided into clusters, identically as at the learning stage, and the vectors representing these clusters are built, and next normalized. These vectors are the input signals for the network. The Kohonen layer determines the winning neuron.

The data compression

It generates the output signal 1 activating the Grossberg neurons, which generate the output signal ***Y*** equal to the number of the winning Kohonen neuron. This number (in the ASCII code) together with the original length of vector ***X*** (also in the ASCII code) is written to the file.

The data compression

Data decompression

Decompression is an inverse procedure. Each stored binary vector $\boldsymbol{Y}_j$, $j = 1,2,...,n*m$, corresponds with j^{th} compressed cluster and with the $\boldsymbol{W}_i$ weight vector of i^{th} Kohonen neuron representing the class of similar vectors (clusters). The weight vector $\boldsymbol{W}_i$ is multiplied by the vector length and the appropriate cluster is reconstructed.

The data compression

The coding performance can be measured by using the signal to noise ratio (SNR) which is defined as follows

$$\mathrm{SNR} = 10\log_{10}\frac{\sum_{i=1}^{N}\left(\sum_{j=1}^{M} a_{ij}^{2}\right)}{\sum_{i=1}^{N}\left(\sum_{j=1}^{M}(a_{ij}-b_{ij})^{2}\right)}$$

where a_{ij} and b_{ij} are *(i,j)th* pixel values of the original image and the coded image, respectively.

The data compression
Example of computer simulation

The learning image

The data compression
Example of computer simulation

The images are segmented into small elements made of 4 (2 * 2), 9 (3 * 3) or 16 (4 * 4) multi level pixels. Each pixel has 64 gray levels.

Vectors are normalized.

For the image of size 80 x 80 pixels and cluster of the size 2 x 2 pixels there were 1600 learning vectors and for 4x4 sub images - 400 learning vectors.

The data compression
Example of computer simulation

The compression ratio depends from the size of a cluster and the length of output vector ***Y***. The not compressed cluster needs as many bytes as the total number of its pixels. The compressed cluster needs *M mod 8* bytes for an output vector plus one byte to store the original length of the compressed vector.

The data compression
Example of computer simulation

Image size 80*80 pixels.

For B = 0.1, cluster 2x2 pixels (1600 clusters)

- 1 Kohonen neuron, compression ratio 2.

For B=0.0016, cluster 4*4 pixels (400 clusters)

- 256 Kohonen neurons, compression ratio 8.

The data compression
Example of computer simulation

original picture

B=0.1 ; 2x2 ; 1 neuron

B=0.1 ; 4x4 ; 1 neuron

B=0.0016 ; 4x4 ; 16 neurons

The data compression
Example of computer simulation

The picture before the compression

The data compression
Example of computer simulation

The picture after the decompression, **B=0.1**; **2x2 1neuron**

The data compression
Example of computer simulation

The picture after the decompression, **B=0.1**;**4x4**, **1neuron**

The data compression
Example of computer simulation

The picture after the decompression, **B=0.0016**; **4x4**; **256 neurons**

The data compression Summary

In the range up to 256 Kohonen neurons the higher number of neurons increases the image quality (after decompression); the compression ratio equal to 2 (for the cluster $k = 2*2$) and compression ratio equal to 8 (for the cluster $k = 4*4$).

Increase the number of the Kohonen neurons above the 256 decreases the compression ration from $k/2$ to $3k/8$.

Increase of k needs significant increase of the number of Kohonen neurons to maintain the proper compression quality.