- ABSTRACT
- INTRODUCTION
- ANN
  - AUTOENCODERS
- SETUP
- EXPERIMENTS (WITH DISCUSSION OF RESULTS)
- SUMMARY

Pattern Classification Using Stacked Autoencoders

PROJECT REPORT

PROJECT MEMBERS

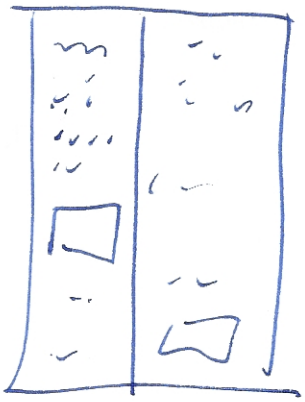
Philipp Grafendorfer
Christina Kastner
Karin Lassnig
Tamara Maier
Alexander Rothschadl

SUPERVISOR

Prof. Helmut Mayer

Department of Computer Sciences
Paris Lodron University of Salzburg

Salzburg, February 18th, 2018



PAPER TEMPLATE → LATEX (EVTL HUF...)

THIS IS NOT A PAPER, IT IS A COLLECTION OF THINGS

- IMPROVE STRUCTURE
- EXPLAIN IMPORTANT CONCEPTS
- ENGLISH CAN BE IMPROVED, BUT IS NOT THE CORE PROBLEM
- INTRODUCTION, DISCUSSION OF RESULTS, SUMMARY

Abstract

This paper is a report of our project "Pattern Classification Using Stacked Autoencoders" carried out during the course "Pattern Recognition II".

At first, we give an overview about the general concept of artificial neural networks. We then switch over to different kinds of autoencoders, which are special neural networks.

For our experiments we used four datasets to train and evaluate our encoders, in which we utilized a particular classifier.

In doing so, we also faced dimensionality reduction as a means of feature selection and denoising in order to prevent the classifier from degenerating to a useless device.

Finally, we documented the results we achieved for each dataset in the form of detailed figures.

WHAT AND WHY INSTEAD OF HOW !!

THERE IS NEARLY NO INFORMATION ON WHAT YOU ARE DOING

MEANINGFUL

Chapter 1

Theoretical Background

1.1 Artificial Neural Networks

Artificial neural networks are a branch of artificial intelligence. They are inspired by biological neural networks, but can also be explained without them. Remarkable advantages over other concepts of artificial intelligence are the high degree of parallel processing and that they are known for yielding comparatively good results.

Components

Artificial neural networks consist of

- Neurons (simple processing units)
- Connections and weights (directed weighted connections between the neurons of consecutive layers)
- Activation function(s) (yield(s) the "strength" of each neuron)
- Learning rule (e.g. backpropagation)

Furthermore, an artificial neural network is characterized by the following layer structure:

- one input layer
- one or more hidden layers
- one output layer

Notation

σ ... activation function
a_j^i ... activation value of the j-th neron in the i-th layer
w_{jk} ... weight from the k-th neuron in the $(i-1)$-th layer to the j-th neuron in the i-the layer
b_j^i ... bias of the j-th neuron in the i-th layer

Contents

Activation function

$$a_j^i = \sigma\left(\sum_k \left(w_{jk}^i \cdot a_k^{i-1}\right) + b_j^i\right)$$

We can choose from different kinds of activation functions. Each of these has its strengths and weaknesses. A good choice is one that assures a fast training process by a fast approximation.

Sigmoid/logit function:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Relu (rectified linear unit) function:

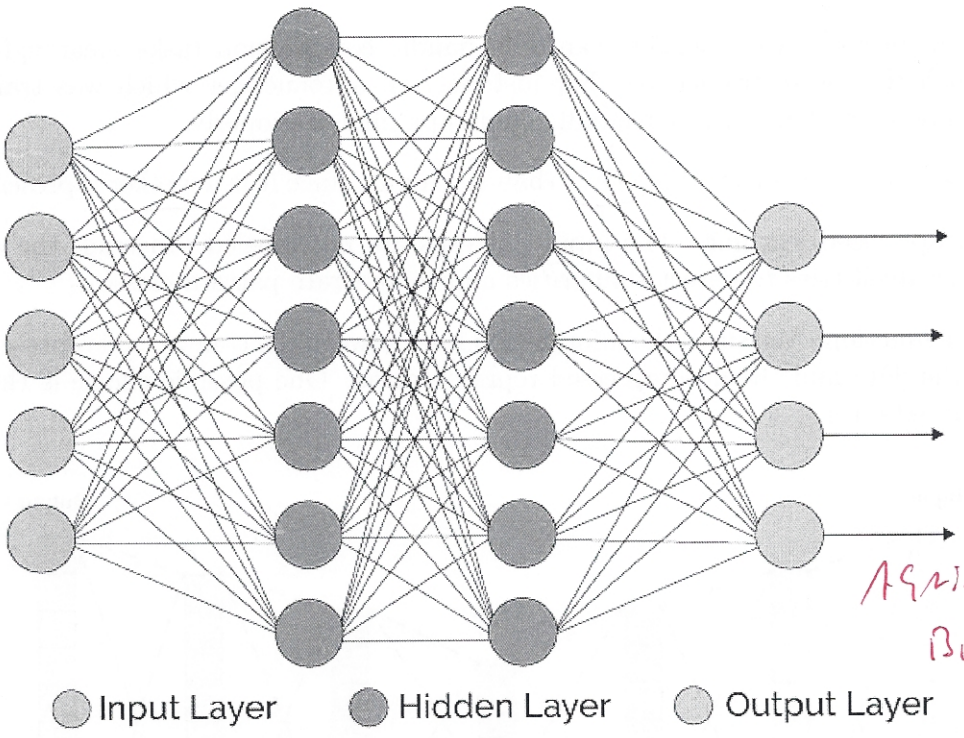
$$\sigma(z) = \max(z, 0)$$

Hyperbolic tangent:

$$\sigma(z) = \tanh(z) = \frac{2}{1+e^{-2z}}$$

Cosine:

$$\sigma(z) = \cos(z)$$



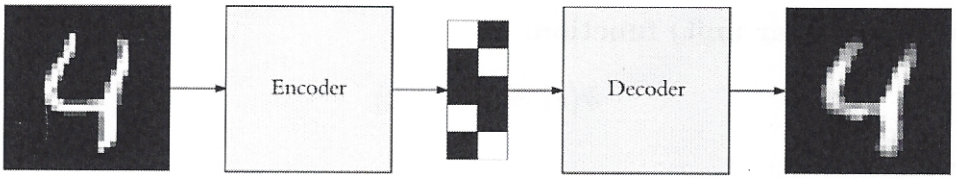
1.2 Autoencoders

An autoenccoder is a special neural network, which is used for the unsupervised learning of efficient codings. The objective is to learn a compressed representation for a set of data and therefore extracting essential features

(→ dimensionality reduction). More precisely, an autoencoder tries to learn an approximation to the identity function with the goal of producing an output that is similar to the input.

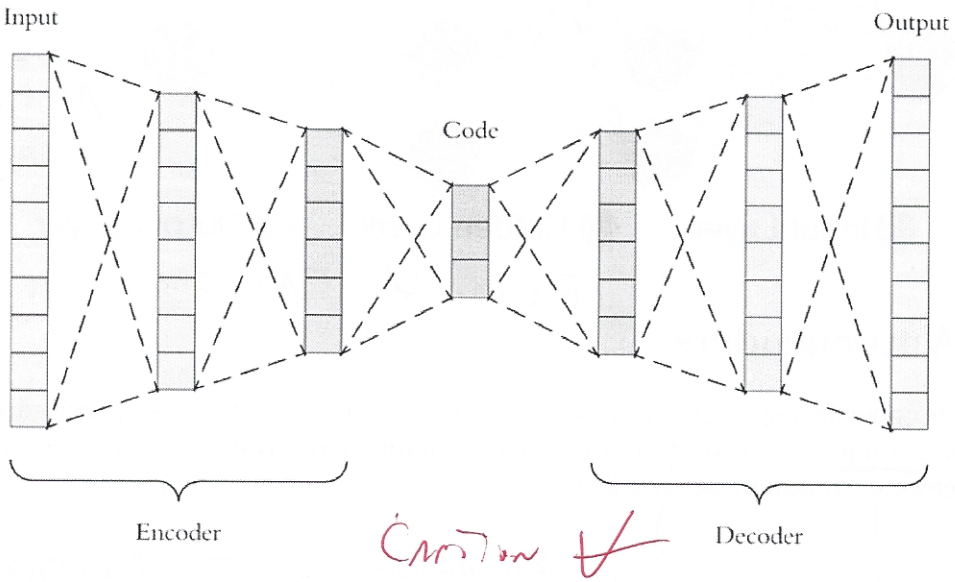
An autoencoder consists of three components:

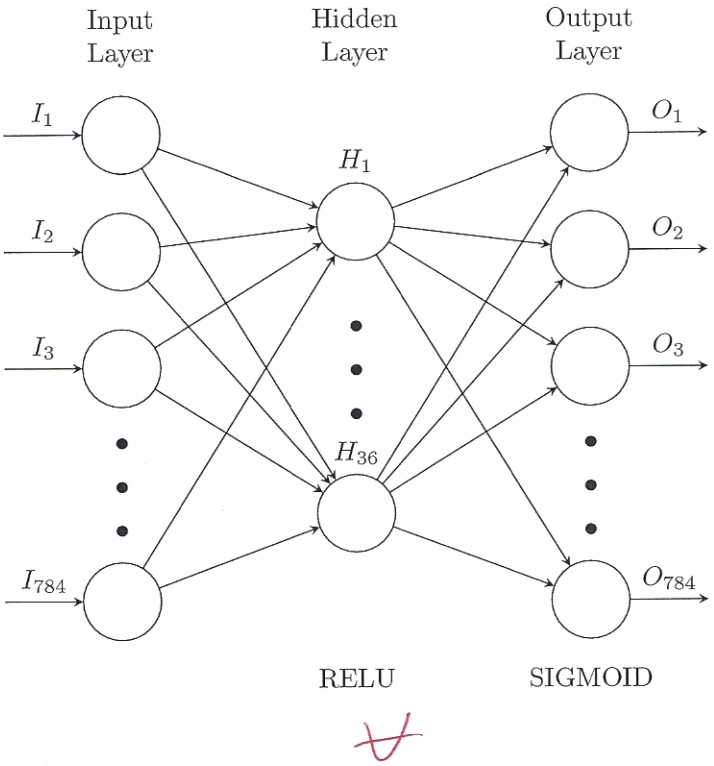
- Encoder: compresses the input and produces the code.
- Code
- Decoder: reconstructs the input using only this code.



Properties of Autoencoders

- Number of layers: Just like a general artificial neural network, an autoencoder consists of one input layer, one ("single-layered") or more ("stacked") hidden layers and one output layer.
- Data-specific: Autoencoders can only handle compression tasks meaningfully for which they were trained to. For instance, an autoencoder which was trained on handwritten digits will not handle landscape photos properly.
- Lossy: The output will be poorer than the input, since it is a weaker representation.
- Unsupervised learning: The input patterns are not labeled. Instead, the autoencoder itself tries to detect similarities and to generate pattern classes.
- Loss function: Measures the information loss between the compressed representation of the data and the decompressed representation. One possible choice is the mean squared error.





1.2.1 Linear Autoencoders

We now want to look at two interesting results regarding linear autoencoders. These are autoencoders with a linear activation function.

A linear autoencoder is optimal under certain conditions. This means, it yields a reconstruction error that is – for a given number of hidden layers – smallest possible. We assume that the decoder is linear and we consider the squared difference error.

Proof: For the proof we will use the following theorem.

Theorem: Let A be an arbitrary matrix with singular value decomposition:

$$A = U\Sigma V^T,$$

where U, V are orthogonal matrices and Σ is a diagonal matrix.

Let $U_{\cdot<k}, \Sigma_{\cdot<k,<}, V^T_{\cdot<k}$ be the decomposition where we only keep the k largest singular values.

Then the matrix B of rank k ($k < \operatorname{rank} A$) that is closest to A[1]

$$B^* = \operatorname{argmin}_{B \text{ such that } \operatorname{rank} B=k} \|A - B\|_F$$

is

$$B^* = U_{\cdot<k}\Sigma_{<k,\leq k}V^T_{\cdot\leq k}.$$

Now we can start with the sketch of the proof.

[1] That means we want to find an approximation B of A that is less complicated. We want to find an approximation that minimizes the Frobenius norm.

When we train the autoencoder, we want to minimize the sum of squared differences:

$$\min_{?} \sum_t \frac{1}{2} \sum_i \left(X_i^{(L)} - \underbrace{\hat{X}_i^{(L)}}_{\text{based on linear autoencoder}} \right)^2 \geq \min_{W^* h(x)} \frac{1}{2} \|X - W^* h(x)\|_F^2,$$

where
$X^{(L)}$ ... training examples
X ... matrix where the columns are the $X^{(L)}$'s
$h(X)$ ... matrix of all hidden layers
W^* ... decoder weights

$$\text{argmin}_{W^* h(x)} \frac{1}{2} \|X - W^* h(X)\|_F^2 = \left(W^* \leftarrow U_{\cdot<k} \Sigma_{<k,<k} h(X) \leftarrow V_{\cdot<k}^T \right)$$

based on the previous theorem, where $X = U\Sigma V^T$ and k denotes the size of the hidden layer.

Let's show that $h(X)$ is a linear encoder.

$$\begin{aligned}
h(X) &= V_{\cdot<k}^T \\
&= V_{\cdot<k}^T (X^T X)^{-1} (X^T X) && \text{(multiplying by identity)} \\
&= V_{\cdot<k}^T (V\Sigma^T U^T U \Sigma V^T)^{-1} (V \Sigma^T U^T X) \\
&= V_{\cdot<k}^T (V \Sigma^T \Sigma V^T)^{-1} V \Sigma^T U^T X && (U^T U = I \text{ because of orthonormality}) \\
&= V_{\cdot<k}^T V (\Sigma^T \Sigma)^{-1} \Sigma^T U^T X && (V(\Sigma^T \Sigma)^{-1} V^T V \Sigma^T \Sigma V^T = I) \\
&= V_{\cdot<k}^T V (\Sigma^T \Sigma)^{-1} \Sigma^T U^T X && (V^T V = I \text{ because of orthonormality}) \\
&= I_{\cdot<k} (\Sigma^T \Sigma)^{-1} \Sigma^T U^T X && (I_{\cdot<k} \ldots \text{identity matrix where we have selected only the first } k \text{ rows.}) \\
&= I_{\cdot<k} \Sigma^{-1} (\Sigma^T)^{-1} \Sigma^T U^T X \\
&= I_{\cdot<k} \Sigma^{-1} U^T X \\
&= \Sigma^{-1}_{\cdot\leq k, \leq k} (U_{\cdot\leq k})^T X && \text{(multiplying by } I_{\cdot\leq k} \text{ selects the first } k \text{ rows)}
\end{aligned}$$

Thus, we have an encoder that was computed by taking all the inputs and linearly transforming them. This means that

$$\underbrace{\Sigma^{-1}_{\cdot<k,\cdot<k} (U_{\cdot<k})^T}_{=:W}$$

is a linear encoder.

Hence, $h(X)$ can be expressed as a linear encoder. $h(X)$ is even the best solution for the minimization problem. Therefore, an optimal pair of encoder and decoder is

$$\underbrace{\Sigma^{-1}_{\cdot<k,\cdot<k} (U_{\cdot<k})^T}_{=W} \quad \text{and} \quad \underbrace{(U_{\cdot<k} \Sigma_{\cdot<k,<k})}_{=W^*} h(X)$$

PROOF REFERENCE.

for the sum of squared errors difference
for an autoencoder with a linear decoder,
where optimality means "has the lowest training reconstruction error". As the training error converges to the generalization error if we have more and more training data, a linear autoencoder will do extremely well for a large amount of training data.

WHERE DOES IT END ?

PROOF -> APPENDIX

If the inputs are normalized as follows:

$$X^{(L)} \leftarrow \frac{1}{\sqrt{T}} \left(\underbrace{X^{(L)}}_{\text{input}} - \underbrace{\frac{1}{T} \sum_{L'-1}^{T} X^{(L')}}_{\text{average of all training inputs}} \right);$$

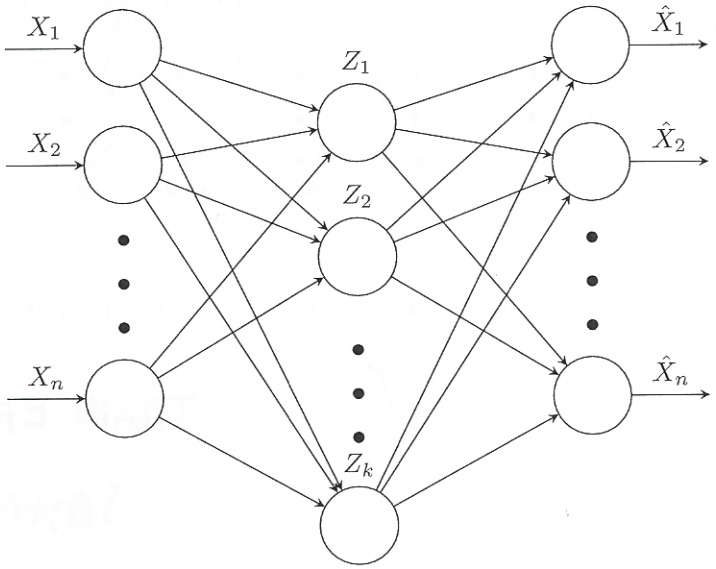
the autoencoder corresponds to the Principal Component Analysis (PCA). To show that an autoencoder with linear activation functions is equivalent to PCA, we look at the simplified case of an autoencoder with only one hidden layer.

At first, we look at the functionality of PCA:

Let X be the original data, z the reduced data and $\hat{X}$ the reconstructed data from the reduced representation. Then we can write PCA as

$$\begin{aligned} Z &= B^T X \\ \hat{X} &= BZ \end{aligned}$$

Now take a look at an autoencoder with the following architecture:



with basically $X \rightarrow Z \rightarrow \hat{X}$.

Since the activation functions are linear, $\mathcal{O}(X) = X$, we can write the autoencoder as

$$\hat{X} = W_1 W_2 X,$$

where W_1, W_2 are the weights of the first and the second layer.

Now, if we set $W_1 = B$, $W_2 = B^T$, we obtain

$$\hat{X} = W_1 W_2 X = W_1 (W_2 X) = W_1 Z = BZ,$$

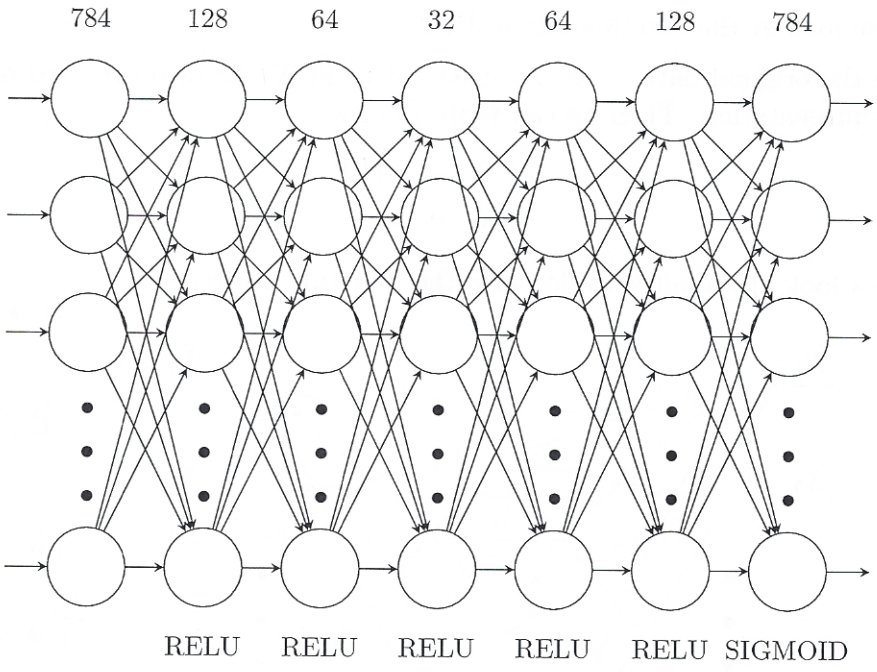
which is the same solution that we had for PCA.

This equivalence is obviously only valid for autoencoders that have a bottleneck layer smaller than the input layer.

1.2.2 Stacked Autoencoders

Stacked autoencoders have multiple hidden layers. We start with the raw inputs to the first hidden (feature) layer and obtain the first-order features. The outputs of the first hidden layer are passed to the second hidden layer, whereupon we get the second-order features and so on – until we reach the last feature layer. From there the values are passed to a classifier. To improve the results we can apply some kind of backpropagation (e.g. gradient descent).

STACKED AUTOENCODER



TRAIN EACH LAYER SEPARATELY

↓

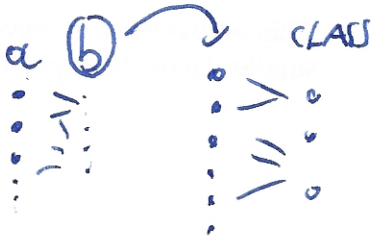
TASH PHILIPP

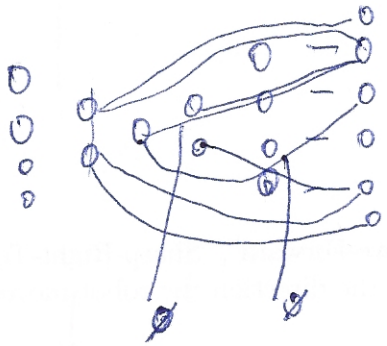
EXPLAIN!!

FROM THIS IT IS NOT CLEAR

HOW YOU TRAIN THE LAYERS

IN ONE STEP OR IN MANY STEPS!?





Chapter 2

Experimental Setup

2.1 Description of Datasets

Semeion

The `Semeion` dataset consists of 1593 handwritten digits between 0 and 9 written by 80 different persons. The handwritten digits were saved in a 16×16 pixel image, whereas a white pixel was scaled to 0 and a black pixel was scaled to 1.
Every person was asked to write each digit twice, first in a normal way and then in a faster way. To summarize the dataset has 256 columns of floats containing 0's and 1's, which will be used as input variables and 10 columns from 0 to 9 representing the output variable. Furthermore, the dataset contains 1593 rows for 1593 handwritten digits.

MNIST

The `MNIST` dataset is a combination of two `NIST`'s databases: Special Database 1 and Special Database 3. These databases also contain handwritten digits between 0 and 9 written by high school students and employees of the United States Census Bureau.
The difference between `Semeion` and `MNIST` is that in `MNIST` the digits were saved in a 28×28 pixel image, whereas a pixel was scaled between 0 and 255. The scaling is indicating the lightness or darkness of a pixel, with higher numbers meaning darker. The `MNIST` dataset has a set of 60000 training examples and 10000 test examples.

Ionosphere

The `Ionosphere` dataset has 351 instances and 34 attributes. It classifies the radar returns from the ionosphere in two classes. If the radar signals show evidence of some type of structure in the ionosphere the instances are classified as "good". The radar signal of instances classified as "bad" passed through the ionosphere.

ROBO

The data was collected as the robot SCITOS G5 navigated through the room following the wall in a clockwise direction. The robot used 24 ultrasound sensors arranged circularly around its "waist" to navigate through the room. The numbering of the ultrasound sensors starts at the front of the robot and increases in clockwise direction.
The dataset consists of 5456 instances and 24 attributes, one for every ultrasound sensor.

The instances are assigned one of the four classes: "Move-Forward", "Sharp-Right-Turn", "Slight-Left-Turn" or "Slight-Right-Turn" depending on the direction the robot moves.

2.2 Our Experiments

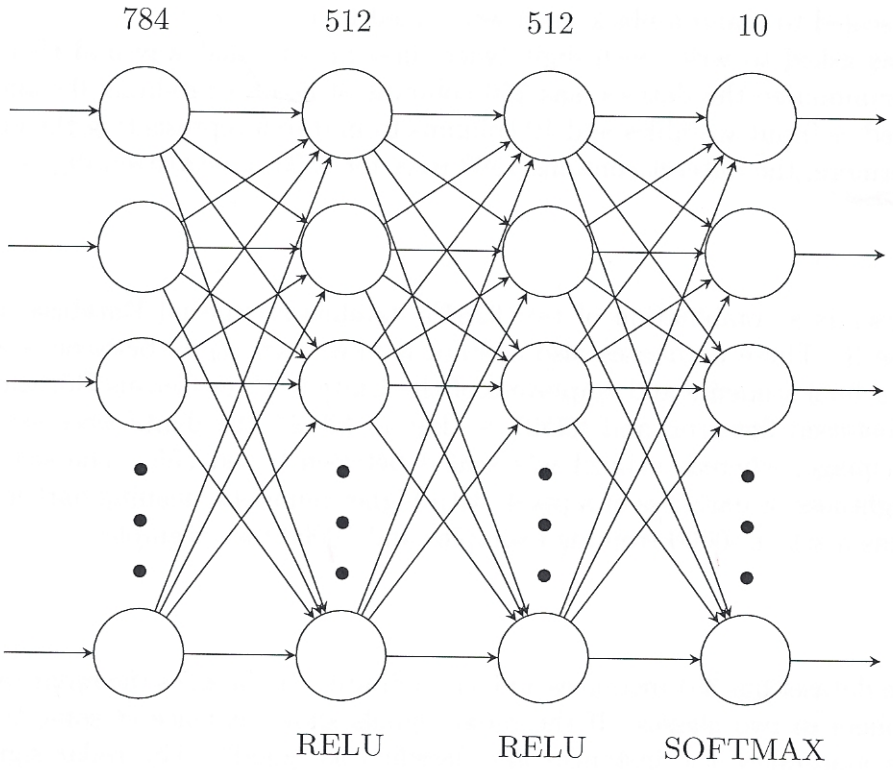
2.2.1 Classifier

For the classifier we introduce a new function, the so-called **softmax function** (also known as **normalized exponential function**). It is a generalization of the logistic function, which squashes a K-dimensional vector $\mathbf{z}$, where $z_1, \ldots, z_K \in \mathbb{R}$, to a normalized vector $\sigma(\mathbf{z})$ of real values in the range $[0, 1]$, where the sum of the coordinates is equal to 1.

$$\sigma \colon \mathbb{R}^K \to [0,1]^K, \quad \sigma(\mathbf{z})_j \mapsto \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}},$$

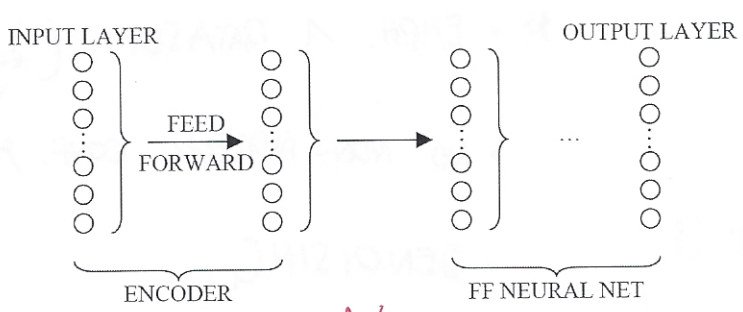
for $j \in \{1, \ldots, K\}$.

CLASSIFIER



2.2.2 Dimensionality Reduction (Feature Selection)

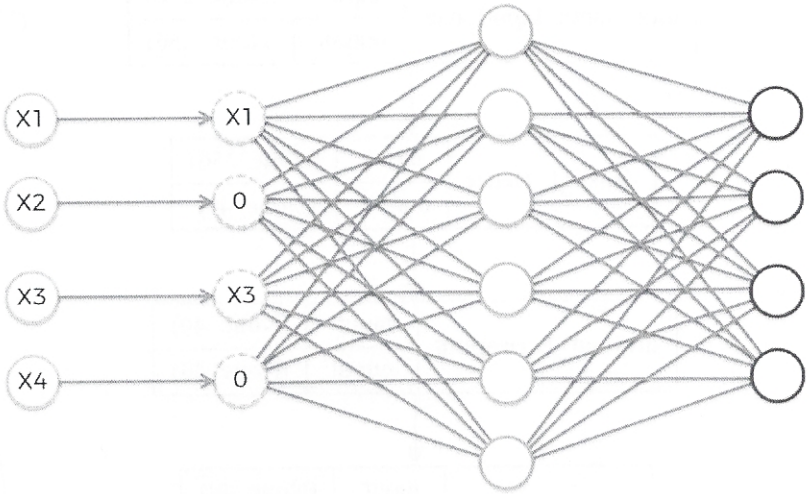
We investigate, how the number of features influences the accuracy.



2.2.3 Denoising

When there are more nodes in the hidden layer than there are inputs, the network is risking to learn the identity function (instead of extracting features), meaning that the output equals the input, making the autoencoder virtually useless. Denoising autoencoders solve this problem by corrupting the data on purpose by randomly turning some of the input values to zero.

ACTUALLY THAT IS THE WHOLE IDEA OF AN AUTOENCODER!!



DENOISING ALSO IMPROVES GENERALIZATION (AT LEAST, IN THEORY)

Chapter 3

Results

Semeion