FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Rozpoznávání souvislé řeči s využitím neuronových sítí
Student:	Adam Zvada
Vedoucí:	Ing. Miroslav Skrbek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Provedte rešerši metod pro rozpoznávání souvislé řeči s využitím neuronových sítí. Uvažujte rekurentní neuronové sítě a zvažte také možnost použití neuronových turingových strojů. Na základě rešerše a po dohodě s vedoucím práce vyberte vhodné řešení pro robota NAO. Maximálně využívejte existujících knihoven s implementacemi potřebných metod. Navržené řešení otestujte na reálných datech. Rozsah práce upřesněte po dohodě s vedoucím práce.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 13. února 2018

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Informatics

Bachelor's thesis

Continuous Speech Recognition by Neural Networks

Adam Zvada

Supervisor: Ing. Miroslav Skrbek Ph.D

April 20, 2018

Acknowledgements

THANKS

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on April 20, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Zvada, Adam. *Continuous Speech Recognition by Neural Networks.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

In my bachelor thesis Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Contents

List of Figures

Introduction

Chapter 1

Introduction to Speech Recognition

Introduction to Speech Recognition

Chapter 2

TESTIK

TESTIK

Chapter 3

Neural Network

// TODO: Why ANN are awesome :)

While artificial neural networks (ANN) have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence.

An artificial neural network (ANN) is mathematical model, heavily inspired by the way how biological neural networks process information in the human brain. In general we can view ANNs as function of f: X -¿ Y where X is the input to neural network and Y is aproximation of our target function. In general we can view ANNs as function that maps

It's achived by gradule from

What do the do? How do they do it? Supervised learning

How its structred?

While neural networks have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence.

In general we can view ANNs as function of f: X -¿ Y

high level of versatility

ANNs aim to reach high level of versatility as our brain does.

In general ANNs are able to reach high level of versatility as our brain because they are aproximating

Inspired by biological nervous systems, artificial neural networks (ANNs) aim at reaching their versatility through learning

While neural networks have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence.

3.1 Inspiration in Nature

Artificial neural network (ANN) is heavily inspired by the way how biological neural networks process information in the human brain. Even though our

brain is extremely complex and still not fully understand, we just need to know how information is being transferred. The basic building block is nerve cell called *neuron*. It receives, processes, and transmits information through electrical and chemical signals. It's estimated that an average human has 86 billion neurons *.

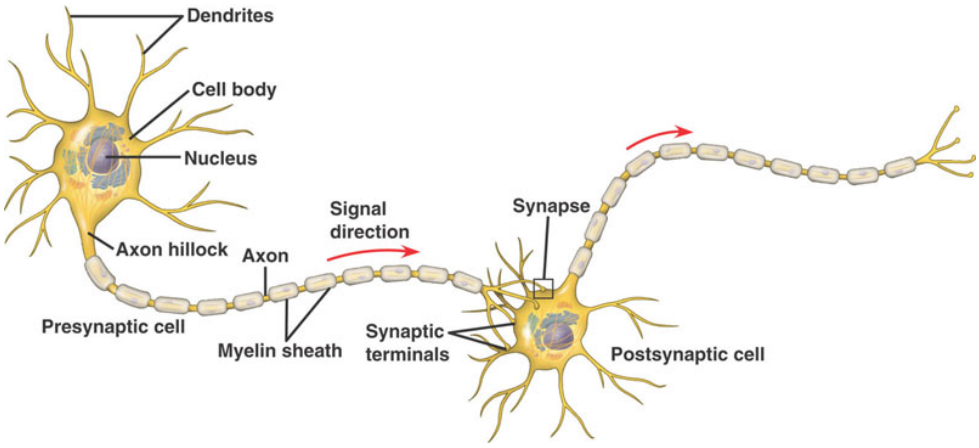

Figure 3.1: Illustration of nerve cell and communication flow

Dendrites are extensions of a nerve cell that propagate the electrochemical stimulation received from other neurons to the cell body. You may think of them as inputs to neuron, whereas neuron's output is called *axon*, a long nerve fiber that conducts electrical impulses away from the cell body. The end of axon is branched to many axon terminals which can be again connected to other dendrites. The connection is managed by *synapses* that can permit the passing of electrical signal to cell body. Once the cell reaches a certain threshold, an action potential will fire, sending the electrical signal down the axon to other connected neurons.

3.2 Artificial Neuron

Artificial neuron is a generic comuptational unit, basic building block for artificial neural network (ANN). It's simplified version of the biological counterpart and we are able to map parts of biological neuron with the artificial one. It takes n inputs represented as a vector $x \in \mathbb{R}^n$ which correspond to dendrites. Generaly artificial neuron produces single output $y \in \mathbb{R}$ as biological neuron where we call it axon. Each neuron's input $i = 1, 2, \ldots, n$ has assigned weight (synapse) $w_1, w_2 \ldots w_n$, they refer to the connection strength between neurons. Weights and same as for synapse are the backbone of learning beacuse in training phases, they keep changing to produce wanted output.(*In this chapter, we will elaborate further.) Inside the artificial neuron, input vector with their

weights are combined and run through an activation function producing some output y. This process is illustrated in $LINK_PRECEPRTON$.

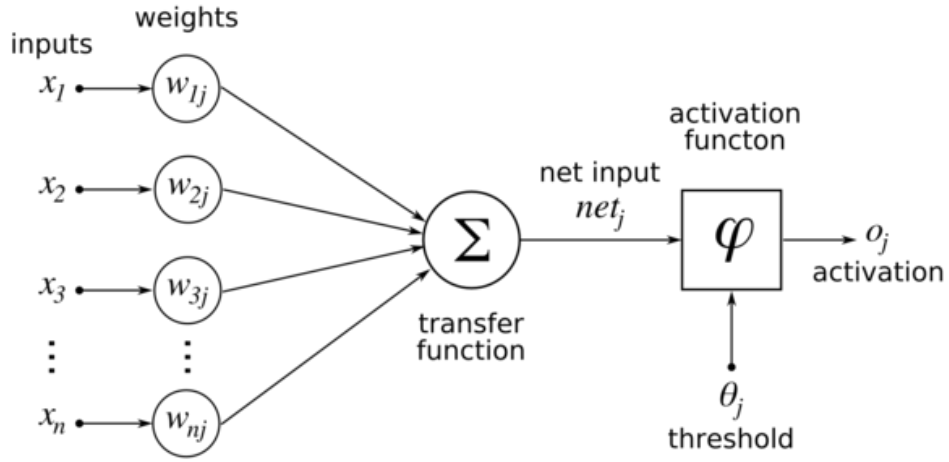

Figure 3.2: Illustration of nerve cell and communication flow

3.3 Perceptron

Perceptron is the simplest ANN with just one neuron and since we covered the basic intuition about artificial neuron we may proceed further and take a look at how output is actually calculated. The equation for a perceptron can be written as

$$y = f(\sum_{i=1}^{N} w_i \cdot x_i + b)$$

where

- x - input vector
- y - predicted output
- f - activation function
- w - weights
- b - bias

Perceptron is a basically linear classifier, therefore the data has to be linearly separable otherwise we would not be able to make the correct prediction. Problems such as speech recognition are not definitely linearly separable, however we can solve non-linear desicions for example by introducing another layer of neurons, thus creating *Multilayered Perceptron*.

3.3.1 Activation Functions

We have stated that biological neuron fires electrical signal to other connected neurons whenever it reaches a certain threshold of incoming electrical impulses. Activation function is based on that concept and inside an artificial neuron it is used for calculating output signal via equation*. It introduces non-linear properties to our ANN and without an activation function would be just a regular linear regression model. Nowdays many diffrent activation function are being used and their performance veries from model to model.

List of some activation function:

- *Sigmoid*
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- *Hyperbolic Tangent*
$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$
- *ReLU*
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$
- *Softmax*
$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}, \quad i = 1, 2 \dots J$$
where i is number of output

* TODO: Activation Functions features... diferentable and non-linear*

3.3.2 Bias

We can think of bias as a value stored inside neuron and being used to calculate it's output. The bias value allows the activation function to be shifted to the left or right, to better fit the data.

3.4 Topology of Artificial Neuron Network

Basic ANN as feedforward model is a directed graph with nodes as neurons and edges with weights representing connection to other neurons. ANN can be divided to three important layers as shown in Fig*. Yellow nodes is an input layer which takes input data, dimension of input vector has to correspond to number of input nodes. Hidden layer as the green nodes is most important to ANN and that is where the training and evaluation happens. Number of

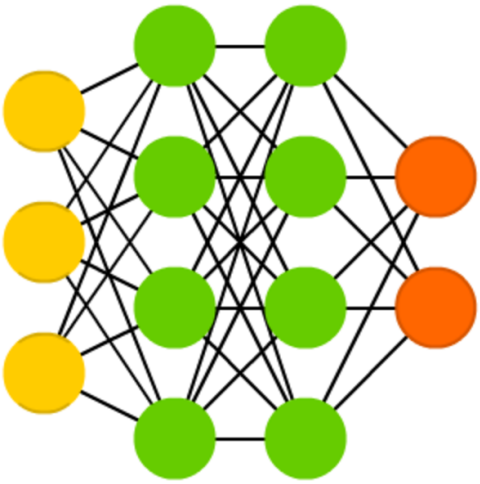

Figure 3.3: Basic topology of fully connected artificial neuron network with input vector of size 3, output vector of size 2 and two hidden layers.

hidden layers and neurons needs to be in a good ratio between its size and its effectiveness. Output layer produces output vector as the prediction for given input.

3.5 Network Evaluation

Evaluation of ANN is done layer by layer.

3.6 Training

The greatest trait of ANN is ability to learn from given data and then make the best aproximate prediction. The aim of the learning process is to find the most optimal values for network's weights and biases while minimizing error on predicated values. For ANN to learn we have to introduce training data consisted of input vector which will be feeded to the network and desired output value (label) for calculating our loss. This aproach is called supervised learning[1].

3.6.1 Loss Function

Loss function compares the prediction from ANN with the desired output and returns the error of the prediction. During a training ANN, the goal is

[1] ANN can be also trained using unsupervised learning.

to minimize given loss fucntion. The most common and most intuitive loss function is Mean squared Error (MSE),

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2.$$

3.6.2 Backpropagation

Backpropagation algorithm is resposible for the ability to learn from given training data. It is an iterative algortihm which for each training data from given training dataset backpropagates the error and adjust the weights and biases accordingly to get desired output.

3.6.2.1 Optimization

Backpropagation requires optimizer to minimize the error on the training data. We will describe backpropagation with using *gradient descent* as the most common optimization algorithm.
Weights and baises are updated using formula,

$$W_{jk}^{l} := W_{jk}^{l} - \alpha\frac{\partial E}{\partial W_{jk}^{l}}$$

$$b_{j}^{l} := b_{j}^{l} - \alpha\frac{\partial E}{\partial b_{j}^{l}}$$

where $\boldsymbol{W_{jk}^{l}}$ is weight with connection between unit j in layer l and unit i in layer $l+1$, $\boldsymbol{b_{j}^{l}}$ is bias associated with unit i in layer $l+1$, $\boldsymbol{\alpha}$ is a learning rate *Reference*, and $\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial W_{jk}^{l}}}$ or $\frac{\boldsymbol{\partial E}}{\boldsymbol{\partial b_{j}^{l}}}$ can be interpreted as minimizing loss function with respect to given weight and bias respectively.

By applying a chain rule twice on the partial derivative of the loss function with respect to a weight, we get

$$\frac{\partial E}{\partial W_{jk}^{l}} = \frac{\partial E}{\partial a_{j}^{l}}\frac{\partial a_{j}^{l}}{\partial z_{j}^{l}}\frac{\partial z_{j}^{l}}{\partial W_{jk}^{l}}$$

where $\boldsymbol{z_{j}^{l}}$ is a sum of weighted inputs to unit j in layer l

$$z_{j}^{l} = b_{j}^{l} + \sum_{k=1}^{K} w_{jk}^{l} a_{k}^{l-1}$$

and $\boldsymbol{a_{j}^{l}}$ is an output of node j in layer l

$$a_{j}^{l} = f(z_{j}^{l}).$$

Let's calculate the last two products of equation *Reference*:

$$\frac{\partial a_j^l}{\partial z_j^l} = f'(z_j^l)$$

$$\frac{\partial z_j^l}{\partial W_{jk}^l} = \frac{\partial W_{jk}^l a_k^l}{\partial W_j k^l} = a_k^{l-1}$$

We introduce a new varibale $\boldsymbol{\delta_j^l}$ which represents the error in unit j in layer l and helps us to better understand and calculate real interested value of $\frac{\partial E}{\partial W_{jk}^l}$ and $\frac{\partial E}{\partial b_j^l}$.

$$\delta_j^l = \frac{\partial E}{\partial z_j^l}$$

We will simplify the error equation on neuron j in output layer L as

$$\delta_j^L = \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L} f'(z_j^L)$$

Now we have enough information to reformulate equation *reference* for output layer to

$$\frac{\partial E}{\partial W_{jk}^l} = \delta_k^L a_j^L.$$

However, to be able to update weights inside the hidden layers, we have to redefine the calculation of δ_j^l. We know that the error produced by an output neuron is just influencing the output value but inside a hidden layer the produced error propagates to all following layers. Therefore we have calculate the δ_j^l where layer l is inside a hidden layer and take into account all δ^{l+1} from following layer $l+1$.

$$\delta_j^l = \frac{\partial E}{\partial z_j^l} = \sum_i \frac{\partial E}{\partial z_i^{l+1}}\frac{\partial z_i^{l+1}}{\partial z_j^l} = \sum_i \frac{\partial E}{\partial z_i^{l+1}}\frac{\partial z_i^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l}$$

$$= \sum_i \delta_i^{l+1} W_{ij}^{l+1} f'(z_j^l)$$

where the sum index i iterates over all neurons in layer $l+1$ and Notice that we have substituted $\frac{\partial E}{\partial z_i^{l+1}}$ with δ_i^{l+1} which is calculated from previous iteration. Finally, we may calculate all weights adjustments through the whole network as

$$W_{jk}^l := W_{jk}^l - \alpha \delta_k^l a_j^l$$

where

$$\delta_k^l = \frac{\partial E}{\partial a_j^L} f'(z_j^L), \quad l = L$$

and

$$\delta_k^l = \sum_i \delta_i^{l+1} W_{ij}^{l+1} f'(z_j^l), \quad l = 2, \ldots, L-1.$$

We won't be exampling the equation for baises adjustments because it follows a similar process shown above with just little changes, resulting to equation

$$b_j^l := b_j^l - \alpha \delta_l^j$$

Using our predefined loss function *ReferenceEquation* we will simplify the error on considered neuron, For example purposes, let us assume that we will use *quadratic loss function*,

$$E = \frac{1}{2}\sum_j (y_j - a_j^l).$$

Algorithm 1 Backpropagation

1: Network evaluation for given input, computing z^l and a^l for each layer
2: **for each** node i in output layer L **do**
3: $\quad \delta_i^L = \frac{\partial E}{\partial a_i^L} f'(z_i^L)$
4: **for each** layer $l = L-1, L-2, \ldots 2$ **do**
5: $\quad$ **for each** node j in layer l **do**
6: $\quad\quad \delta_j^l = \frac{\partial E}{\partial z_j}$
7: $\frac{\partial E}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$
8: $\frac{\partial E}{\partial b_j^l} = \delta_j^l$
9: $w_{jk}^l := w_j k^l - \alpha \frac{\partial E}{\partial w_{jk}^l}$
10: $b_j^l := b_j^l - \alpha \frac{\partial E}{\partial b_j^l}$

Chapter 4

Recurrent Neural Network

Chapter 5

Connectionist temporal classification

Connectionist temporal classification

Conclusion

Appendix A

Acronyms

GUI Graphical user interface

XML Extensible markup language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format