ELAML
Daniël de Kok
University of Tübingen
SoSe 2018

Le Duyen Sandra Vu
4171456
9CP

Single hidden layer neural network for classification

Neural Networks Introduction

Neural networks are modeled after the human thinking. Which means, if a neuron receives signals until a certain treshold is reached, it fires. There are different neural networks for various cases. This project is an implementation of a simple Vanilla single hidden layer Feed-Forward NN in order to learn classification.
Classification is usually performed supervised, which means that the training data contains the data (features) and the class it belongs to (label). The neural network will hopefully learn a pattern from these data and adjusts the weights to finally classify unlabeled data correctly.

Math

To simulate the human neuron behavior the input will be multiplied with some weights and then squashed into an activation function. For example Tanh which has values between 1 and -1, where 1 represents a neuron firing.
Since this is a single hidden layer neural network, this will be done two times. First the input vector will be multiplied with the weight matrix and the calculated values then go into the activation function. This are the results of the hidden layer. The hidden layer results vector will multiplied with another matrix vector and this will be squashed again by an activation function. The final result will be compared with the desired output.

Err = calculated — desired
hiddenVector = activation_function(inputVector * weightMatrix1 +biashVector1)
outputVector = activation_function(hiddenVector * weightMatrix2 +biasVector2)

(to make the network more flexible some biases can be added)

The goal is to adjust the weights of the neural network until the error between the calculated output and the desired output is as small as possible. So how should the network know in which way it should adjust the weights? The gradient descent is used to find the direction with the smallest resistance to get the error as minimal as possible. In other words the direction with the largest gradient. Imagine you are somewhere on a mountain and looking for the fastest way down.

$W_{new} = W_{old} + (-\eta * \partial Err/\partial W)$

Program

To run the program: cargo build --release
target/release/neural_network filename.txt filename.txt
To generate the documantation:
cargo rustdoc --open -- --no-defaults --passes collapse-docs --passes unindent-comments --passes strip-priv-imports

For more usage information: -h

txt train und test files (including one for inappropriate input data), **args** for reading in terminal arguments, **main** is the program using all the functions from the other rs files, **nn** contains the functions and structs about the neural network, **reader** is for the conversation of the txt data into a 2D vector

Resources

Very helpful nn implementation explained
https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/

Great book and awesome professor
Wolfgang Ertel (2013)
Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung
Springer Vieweg, chapter 9.