Fundamentals of Artificial Intelligence (57205HT16): Assignment 2 - Happy, Sad, Mischievous or Mad?

Marc Coquand
id14mcd
mcoquand@gmail.com

Linus Lagerhjelm
id14llm
id14llm@cs.umu.se

Supervisor: Thomas Johansson
Alexander Sutherland
Thomas Hellström

October 10, 2016

Contents

1 Introduction

This report contains the documentation and explanation of a face recognition program. The program can recognize if a set of drawn faces are happy, sad, mischievous or mad.

1.1 Running the program

The implementation is written in python 2.7 which means that a python 2.7 shell is required to run the implementation. Required program arguments are a search path to a `training.txt`, search path to an answer file `training-facit.txt` and a test file `test-file.txt`. The format of the training file should be the id of the image followed by the image on a new line. The answer file should be formatted as the image id follow by the correct answer (1,2,3 or 4).

Should the user fail to provide the required arguments, the program will immediately terminate with a massage explaining that not enough input arguments were provided.

Example usage of the program might look like:

```
python faces.py training-file.txt training-facit.txt test-file.txt
```

2 Problem description

2.1 Perceptron Learning Algorithm

The program written implements an algorithm for supervised learning called *Perceptron Learning*. *Supervised Learning* is a technuque for machine learning where the ANN is exposed to input and produces a guess based on the knowledge aquired from previous guesses. Since the desired output from the ANN, in supervised learning, is known for every input, the ANN is slightly modified with each iteration to eventually produce good outputs for every set of inputs.

The ANN in the produced program is composed by four different Perceptrons

where each one represents one of the different facial expressions that are to be identified in the assignment.

Each perceptron is implemented in *Python* as a class composed by a list of *401* weights that mapps to the *400* pixel values that makes up an image. The additional weight is provided as a bias for the perceptron. A bias is used in order to shift the result of the activation function (more on that later) towards either the left or the right. I.e. to make the output more extreme so that it will produce a more distinct answer.

Training of the network occurs in the *Tutor class* where the provided set of training images is repetedly shown for each perceptron. The output from a perceptron, also known as *activation*, is computed using the formula:

$$a_i = act(\sum_j x_j * w_{j,i})$$

where x is the input, w is the weight and act is the activation function which is the sigmoid function:

$$sigmoid(x) = 1/(1 + e^{-x})$$

$$\begin{aligned} e_i &= y_i - a_i \\ \nabla w_{j,i} &= \alpha e_i x_j \\ w_{j,i} &\leftarrow w_{j,i} + \nabla w_{j,i} \end{aligned}$$