UMEÅ UNIVERSITY
Department of Computing Science

November 5, 2018

Fundamentals of Artificial Intelligence 5DV121 HT-18

Assignment 5

version 2.0

Namn Kristoffer Gilliusson, Patricia Melin Jonsson
Username krgi0001/c14kgn, pame0014/oi14pmn
Lärare Thomas Johansson, Carl-Anton Anserud, Anders Broberg, Adam Dahlgren Lindström, Ola Ringdahl

Contents

1 Introduction

In this assignment the task is to build a perceptron based classification system that guesses the emotional state of faces presented as input. The program should be able to distinguish if a picture is representing a happy, sad, angry or mischievous face.

There was three files given prior to this assignment. These are images.txt, keys.txt, test.txt. images.txt is 200 images that is used as training data, keys.txt is used as validation for the training data and test.txt is 100 images used for guessing.

The program should be able to be tested on a random number of images and the correctly classified images must be over 65% of all the images.

2 How to run

To run this code all you need to do is to locate the project map and compile it with :

```
javac *.java
```

and run it with :

```
java Faces images.txt keys.txt test.txt
```

3 Theory

A network of artificial neurons that can receive inputs, use this input to change state, and after that produce an output according to that input is what we call an artificial neural network. The neural network uses some kind of activation to produce the output. One type of activation could be the Sigmoid function. The computing system is inspired by biological neural networks, such as those in the human brain. The system is used in machine learning and one example of tasks that the system can learn to achieve is to recognize and identify images. The system is trained with a lot of images and learns to recognize specific characteristics from the learning material that they process.

The neurons in the system have a weight that increases or decreases as the learning proceeds. The neurons may also have a threshold that limits which results to present. Often the artificial neurons are aggregated into layers and the system goes through the layers multiple times.

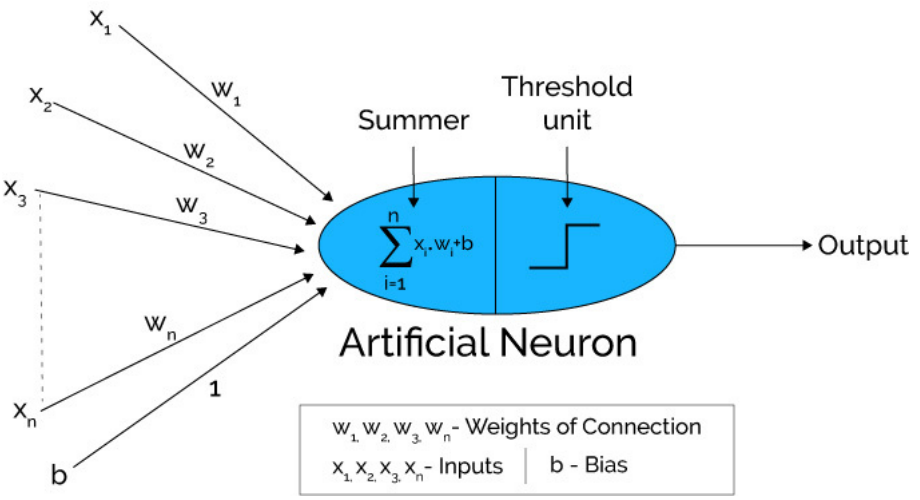

Figure 1: Image of an artificial neural network.

In machine learning, a perceptron is an algorithm that uses supervised learning to classify inputs, for example images. This is a form of linear classification. The algorithm multiplies the input data with weights. After that it subtracts a bias to shift the decision boundary. It does this as many times as there are inputs to train on.

3.1 Perceptron

The preceptron used in this lab uses two vectors which are a pixelVector and a weightVector. The pixelVector is a image of 20x20 size compressed into an vector or 400 size. This vector is filled with the images' grey levels. These levels range from 0 - 31 and in the vector they are divided by 31 to fit the range of 0 - 1. The weightVector is on initialization filled with 400 random 0-1 doubles.

To calculate a result from these grey levels and weights this function is used to multiply and sum the two vectors together.

$$\sum_{i=1}^{n} x_i * w_i + bias \tag{1}$$

The output from the function is then used in an activation function which is this case is the sigmoid function.

$$y = \frac{1}{(1 + e^{-x})} \tag{2}$$

The only thing the sigmoid does is compress the output to a value between 0 - 1.

3.2 Training the perceptron

In this lab there are 4 different perceptrons which should specialize on different type of images. To train these a simple algorithm was used :

1. Shuffle the images around for randomness.
2. Find a correct goal value depending on what image it is training on and what type of perceptron it is.
3. A small learingRate for slow convergence towards the goal.
4. Calculate an error.
5. Update the weights.
6. Repeat until high enough accuracy

To gain a good enough accuracy to pass the 65% grade a learning rate of 0.07 was used. There were no algorithm is use to decide when to stop training. Instead this was check manually after numerous iterations.

4 Difficulties

One of the first problems we faced was that we thought that we should create only one perceptron that would be able to distinguish between all four different face expressions. After some time we understood that we should instead train one perceptron for each face expression. With four perceptrons the problem was much easier to solve.

From the beginning we tried to run the inputs in the order that they were presented. After some time we realized that the images needed to be randomized for each training round in order to make the activation output fairer.

Apart from this, it also took some time to realize that we needed to run the training more than once on the training set to get the accuracy we wanted.