Profielwerkstuk

A report on Machine Learning, Neural Networks and the evolution of Neural Networks through Augmenting Topologies - describing the underlying methods and its particular application to making a computer teach itself how to play a video game

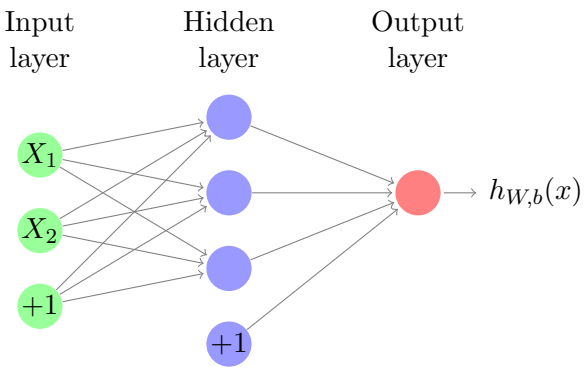
by Maxim van den Berg, Sam van Kampen, Wilco Stam and Robin Wacanno

23-12-2016

Abstract

Lekker abstract

Contents


Input layer
Hidden layer
Output layer
X_1
X_2
+1
+1
$h_{W,b}(x)$

Listing 1: Data definitions (nnrv.m)

```
W1 = stdnormal_rnd(2,3);
W2 = stdnormal_rnd(3,1);
b1 = stdnormal_rnd(1,3);
b2 = stdnormal_rnd(1,1);




exx1 = [1; 0; 1; 1; 0; 0; 0; 1; 1; 0; 1; 0; 0; 0];
exx2 = [0; 1; 0; 1; 0; 0; 1; 1; 1; 0; 0; 1; 1; 0];

exy =  [0; 0; 0; 1; 0; 0; 0; 1; 1; 0; 0; 0; 0; 0];
```

Listing 2: Implementation of the XNOR gate in a neural network (nnt1.m)

```
% An implementation of the XNOR-gate using a neural network
% Uses data values defined in nnrv.m
%
% by Maxim van den Berg, Sam van Kampen, Robin Wacanno and Wilco Stam

% The learning rate
alpha = 1;

% The effect of weight decay is negligible in this network, and is therefore
    zero.
lambda = 0;


maxi = input("maxi: ");
m = rows(exx1);

function s = f(z)
  s = 1./(1 + exp(-z));
end


%
disp("W2 (begin) =");
disp(W2);
disp("W1 (begin) =");
disp(W1);
disp("b2 (begin) =");
disp(b2);
disp("b1 (begin) =");
disp(b1);

% Compute starting values, so that they can be displayed
a1 = [exx1, exx2];
z2 = a1 * W1 + b1;
a2 = f(z2);
z3 = a2 * W2 + b2;
h = f(z3);
a3 = h;
disp("h (begin) =");
```

```
disp(h);


for i = 1:maxi

% Compute the current output of the neural net
a1 = [exx1, exx2];

z2 = a1 * W1 + b1;
a2 = f(z2);

z3 = a2 * W2 + b2;
h = f(z3);
a3 = h;

% Start computing deltas for backprop

%delta[laagnummer]
delta3 = -(exy - h)   .* (a3 .* (1 - a3)); % f'(z) = f(z)(1 - f(z)).
delta2 = (delta3 * W2') .* (a2 .* (1 - a2));

gradW2 = a2' * delta3;
gradW1 = a1' * delta2;
gradb2 = sum(delta3);
gradb1 = sum(delta2);


W2 = W2 - alpha * (gradW2 * 1/m) + lambda * W2);
W1 = W1 - alpha * (gradW1 * 1/m) + lambda * W1);

b2 = b2 - alpha * (gradb2 * 1/m);
b1 = b1 - alpha * (gradb1 * 1/m);

end


disp("h =");
disp(h);

disp("round(h) =");
disp(round(h));

disp("J / cost =");
disp(delta3);

disp("W2 =");
disp(W2);
disp("W1 =");
disp(W1);

disp("b2 =");
disp(b2);
disp("b1 =");
disp(b1);
```

(temporary titles)

1 Introduction

2 [temp]

2.1 Background information

2.1.1 Solving mathematical problems with a computer

- History
- Basic idea (fiting a formule to a data set and such)

> "[Machine Learning is] The field of study that gives computers the ability to learn without being explicitly programmed."
> Arthur Lee Samuel (1959), a pioneer in the field of computer gaming, artificial intelligence, and machine learning.
>
> "Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."
> Tom M. Mitchell, a famous computer scientist and university professor.

2.1.2 Neural networks - a simulation of the brain [temp]

- Biology lesson about the brain and how neural networks work alike, especially evolving neural networks

2.1.3 Examples of applications of machine learning and neural networks

- Examples of neural network implementation and what they have achieved

2.2 Machine learning

2.2.1 Mathematical theory and model

- Gradient descend
- formulas and alpurrr
- Explanation vectorized implementation

2.2.2 Implementation in the programming language Octave

- First without vectorization.
- Then with vectorized implementation - Determining the amount of iterations and the value for alpha.

2.3 Neural networks

2.3.1 Mathematical theory and model

- formulas and alpurrr
- Explanation vectorized implementation

2.3.2 Implementation in the programming language Octave

- First without vectorization.
- Then with vectorized implementation - Determining the amount of iterations and the value for alpha and labda.

2.3.3 Alternative algorithms to gradient descent

2.4 Evolving neural networks and self-learning artificial intelligence

2.4.1 Self improving neural networks

neural networks are braaaains, etc. etc. etc.

2.4.2 The application of neural networks based artificial intelligence to videogames

- theory - input handling - Which games is is applicable to?

2.4.3 Implementation in the programming language Octave

2.5 Experiment: The application of neural networks based artificial intelligence to videogames

2.5.1 Choosing an emulator and transfering the program to the appropiate language

2.5.2 The setup and time partition of the experiment

2.5.3 [after the experiment] Analysis of the data

- Did it work?
- How long did it take?
- Application on other levels and games
- How can the program be improved

3 Conclusion