TWO-LAYER PERCEPTRON

Table of Contents

Load data 1
Initialize variables and constants 1
Stochastic gradient descent until C < 12% 1
Save data 2
Functions 2

Load data

```
clear;
trainingSet = load('training_set.csv')';
validationSet = load('validation_set.csv')';
```

Initialize variables and constants

```
M1 = 8;
M2 = 6;
learningRate = 0.02;

w1 = randomWithinRange([M1, 2], -0.2, 0.2);
t1 = zeros(M1, 1);
w2 =  randomWithinRange([M2, M1], -0.2, 0.2);
t2 = zeros(M2, 1);
w3 = randomWithinRange([1, M2], -0.2, 0.2);
t3 = 0;

% Save validation error values every tFrequency points
tFrequency = 1e4;
errorCondition = 0.12;
errorArray = [];
t = 0;
```

Stochastic gradient descent until C < 12%

```
while true
    t = t+1;
    patternNumber = randi([1 length(validationSet)]);
    input = trainingSet(1:2, patternNumber);
    target = trainingSet(3, patternNumber);
    V1 = g(-t1 + w1*input);
    V2 = g(-t2 + w2*V1);
    output = calculateOutput(input, w1, t1, w2, t2, w3, t3);
    delta3 = gPrime(-t3 + w3*V2)*(target-output);
    delta2 = gPrime(-t2 + w2*V1).*(w3'*delta3);
    delta1= gPrime(-t1 + w1*input).*(w2'*delta2);
```

```
    if mod(t-1, tFrequency) == 0
        error = validationError(validationSet, w1, t1, w2, t2, w3,
 t3);
        errorArray((t-1)/tFrequency+1) = error;
        if error < errorCondition
            break
        end
    end
    w1 = w1 + learningRate*delta1*input';
    w2 = w2 + learningRate*delta2*V1';
    w3 = w3 + learningRate*delta3*V2';
    t1 = t1 - learningRate*delta1;
    t2 = t2 - learningRate*delta2;
    t3 = t3 - learningRate*delta3;
end
```

Save data

```
csvwrite('w1.csv', w1)
csvwrite('w2.csv', w2)
csvwrite('w3.csv', w3')
csvwrite('t1.csv', t1)
csvwrite('t2.csv', t2)
csvwrite('t3.csv', t3)
```

Functions

```
function error = validationError(validationSet, w1, t1, w2, t2, w3,
 t3)
    pVal = length(validationSet);
    errorSum = 0;
    for i = 1:pVal
        input = validationSet(1:2,i);
        output = calculateOutput(input, w1, t1, w2, t2, w3, t3);
        target = validationSet(3,i);
        errorSum = errorSum + abs(sign(output)-target);
    end
    error = errorSum./(2.*pVal);
end

function g = g(x)
    g = tanh(x);
end

function gPrime = gPrime(x)
    gPrime = 1-(tanh(x)).^2;
end

function output = calculateOutput(input, w1, t1, w2, t2, w3, t3)
    V1 = g(-t1 + w1*input);
    V2 = g(-t2 + w2*V1);
    output = g(-t3 + w3*V2);
end
```

```
function r = randomWithinRange(size, min, max)
    r = (max - min).*rand(size) + min;
end
```

Published with MATLAB® R2020a