CSCI964 Computational Intelligence: Lab #1

Mei Wangzhihui 2019124044

Task 1

```
#include <iostream>
#include <eigen3/Eigen/Core>

using namespace std;
using namespace Eigen;

double SigmoidFunc(double x)
{
    return 1.0 / (1.0 + exp(-x));
}

void Sigmoid(VectorXf &src, VectorXf &dst)
{
    float *src_data = src.data();
    float *dst_data = dst.data();
    for (int i = 0; i < src.size(); ++i)
    {
        dst_data[i] = SigmoidFunc(src_data[i]);
    }
}

double forwardProp(int numberOfLayers, MatrixXd input, MatrixXd *weights, MatrixXd *thetas)
{
    MatrixXd output = input;
    // output.transposeInPlace();
    for (int i = 0; i < numberOfLayers; i++)
    {
        output = (output * weights[i] - thetas[i]).unaryExpr(&SigmoidFunc);
    }
    return output(0);
}

void gradDst(const int batch_size, const int layernumber, const int dataset_size, const int
     maxepoch, const double LR, const double maxerror, MatrixXd *x, MatrixXd *weights, MatrixXd
     *thetas, double *y)
{
    int epoch = 0, ctr = 0, idx = 0;
    double error = 0, lasterror, deltaerror, sumerror = 0;
    MatrixXd deltaW(3, 1);
    double deltaTheta = 0;
    deltaW << 0, 0, 0;
    //Calculate the initial mean error
    for (int i = 0; i < dataset_size; i++)
    {
        double t = forwardProp(layernumber, x[i], weights, thetas);
        sumerror += 0.5 * (t - y[i]) * (t - y[i]);
    }
    lasterror = sumerror / dataset_size;
    do
    {
        // One Epoch
```

Problem 1 continued on next page...

```
        epoch++;
        for (int i = 0; i < batch_size; i++)
        {
            double t = forwardProp(layernumber, x[idx], weights, thetas);
            deltaW += LR * (t - y[idx]) * t * (1 - t) * x[idx].transpose();
            deltaTheta += LR * (t - y[idx]) * t * (1 - t);
            idx = (idx + 1) % dataset_size;
        }
        // Adjust The Weight and theta
        weights[layernumber - 1] -= deltaW / batch_size;
        thetas[layernumber - 1](0) += deltaTheta / batch_size;
        sumerror = 0;
        //Calculate the Mean Error
        for (int i = 0; i < dataset_size; i++)
        {
            double t = forwardProp(layernumber, x[i], weights, thetas);
            sumerror += 0.5 * (t - y[i]) * (t - y[i]);
        }
        error = sumerror / dataset_size;
        deltaerror = abs(error - lasterror);

        lasterror = error;
        deltaW << 0, 0, 0;
        deltaTheta = 0;

        cout << "\nepochs: " << epoch << endl;
        cout << "error: " << error << endl;
        cout << "deltaerror:" << deltaerror << endl;
        cout << "w" << weights[0] << endl;
        cout << "theta: " << thetas[0] << endl;
    } while (deltaerror > maxerror && epoch < maxepoch);
}

int main()
{
    MatrixXd x0(1, 3), x1(1, 3), x2(1, 3), x3(1, 3);
    MatrixXd weight(3, 1), bias(1, 1);
    x0 << 0, 0, 1;
    x1 << 0, 1, 1;
    x2 << 1, 0, 1;
    x3 << 1, 1, 1;
    weight << 0, 0, 0;
    bias << 0;
    double y0 = 0, y1 = 0, y2 = 1, y3 = 1;
    MatrixXd x[] = {x0, x1, x2, x3};
    double y[] = {y0, y1, y2, y3};
    MatrixXd biases[] = {bias};
    MatrixXd weights[] = {weight};
    //gradDst(1, 1, 4, 0.01, 0.001, 1000000, x, weights, biases, y); // online learning
    gradDst(3, 1, 4, 1000000, 0.01, 0.000000001, x, weights, biases, y); // batch method
}
```

Set initail $W = (0, 0, 0)$ initial $\theta = 0$ max $\delta_{error} = 1E - 9$ learning rate $LR = 0.01$

Problem 1 continued on next page...

```
epochs: 322449
error: 0.000391416
deltaerror:1.60764e-09
w  7.10954
-0.122093
 -1.71753
theta: 1.71753

epochs: 322450
error: 0.000391414
deltaerror:1.2596e-09
w  7.10954
-0.122101
 -1.71754
theta: 1.71754

epochs: 322451
error: 0.000391413
deltaerror:9.99997e-10
w  7.10955
-0.122101
 -1.71754
theta: 1.71754
```

Figure 1: Online Learning Method (batchsize = 1)

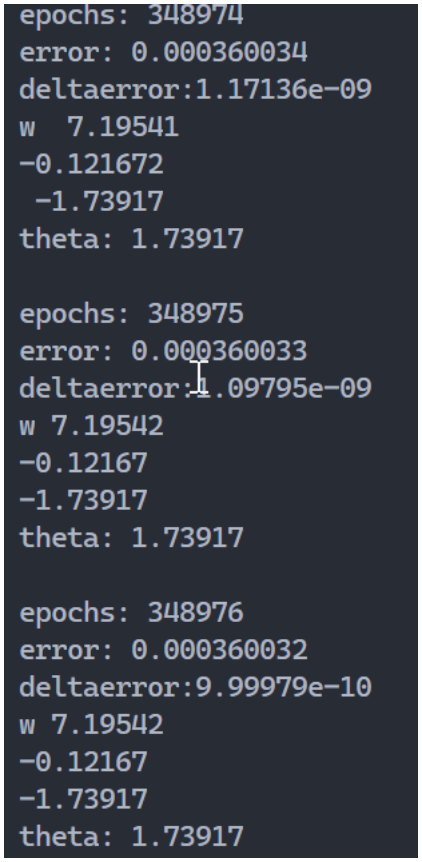

Figure 2: Batch Method (batchsize = 3)