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);
}

double gradDst(int batch_size, int layernumber, int dataset_size, double LR, double maxerror,
    int maxt, MatrixXd *x, MatrixXd *weights, MatrixXd *thetas, double *y)
{
   int times = 0, ctr = 0, idx = 0;
   double error = 100, sumerror = 0;
   MatrixXd deltaW(3, 1);
   double deltaTheta = 0;
   deltaW << 0, 0, 0;
   while (error > maxerror && times < maxt)
   {
      idx = (idx + 1) % dataset_size;
      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);
      ctr = (ctr + 1) % batch_size;
      sumerror += 0.5 * (t - y[idx]) * (t - y[idx]);
      if (ctr == 0)
      {
         weights[layernumber - 1] -= deltaW;
```

Problem 1 continued on next page...

```
            thetas[layernumber - 1](0) += deltaTheta;
            deltaW << 0, 0, 0;
            deltaTheta = 0;
        }
        times++;
        if (idx == 0)
        {
            error = sumerror / dataset_size;
            sumerror = 0;
        }

        cout << "epochs: " << times << endl;
        cout << "error: " << error << endl;
        cout << "w" << weights[0] << endl;
        cout << "theta: " << thetas[0] << endl;
    }
    return error;
}

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 << 6, 0, 0;
    bias << 2;
    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, set
        batch to 1
    gradDst(3, 1, 4, 0.01, 0.001, 1000000, x, weights, biases, y); // batch method , set batch to
        1
}
```

Problem 1 continued on next page...

```
epochs: 117690
error: 0.00100001
w  8.48812
-0.116891
  -2.0642
theta: 2.0642

epochs: 117691
error: 0.00100001
w  8.48814
-0.116871
 -2.06418
theta: 2.06418

epochs: 117692
error: 0.000999998
w  8.48814
-0.116871
 -2.06421
theta: 2.06421
```

Figure 1: Online Learning Method

```
epochs: 119996
error: 0.00100001
w  8.51003
-0.116895
 -2.06968
theta: 2.06968

epochs: 119997
error: 0.00100001
w  8.51003
-0.116895
 -2.06968
theta: 2.06968

epochs: 119998
error: 0.000999997
w  8.51003
-0.116895
 -2.06968
theta: 2.06968
```

Figure 2: Batch Method (batchsize = 100)

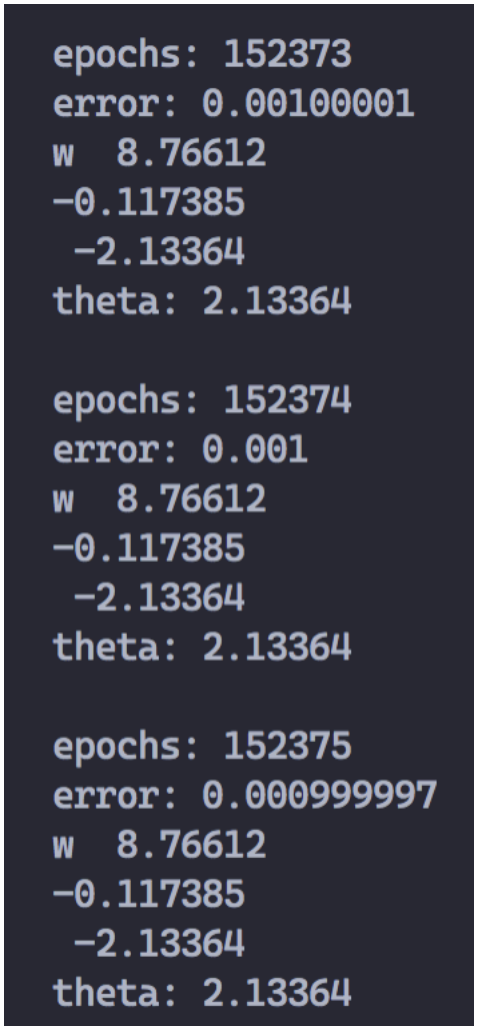

Figure 3: Batch Method (batchsize = 200)