Dan Wilder

University of Missouri - St. Louis
CS 4340 Machine Learning
Uday K. Chakraborty, PhD (Instructor)

Project #1: Perceptron Learning Algorithm (PLA)

This project required implementing the perceptron learning algorithm in a language of the student's choosing. I chose the C programming language. The requirement was to choose 10 data points for training and 5 for testing. The data sets were to be of two dimensions, points on a plane. The specific values of these points were left to the discretion of the student. I chose the points arbitrarily, except that I intentionally made them linearly separable. Tables 1 and 2 summarize this. The PLA also requires some other initialization. The choice for the PLA positive constants, **c** and **k,** were both set to 1. The initial choice for all the weights were also to set to 1.

Point	**True Class**
(1, 2)	1
(2, 4)	1
(3, 3)	1
(-1, 4)	1
(-3, -4)	-1
(-2, -2)	-1
(-5, 1.5)	-1
(2.5, -3)	1
(1.5, -0.5)	1
(-2.5, 3.75)	-1

Table 1 : A linearly separable training set

The program used the training data to generate a solution. The final solution equation also known as the decision boundary was **2 + 6.5x + 3.75y = 0.** This equation was used to classify the test data set.

Point	**True Class**	**Classified As**
(0, 0)	1	1
(2, -1)	1	1
(-3, 3)	-1	-1
(-1, -1)	-1	-1
(-2, 1)	-1	-1

Table 2: Linearly separable test data set classified according to the decision boundary.

Some statistics about the run were collected. Specifically, it took 5 iterations over the training set with 7 weight updates before the solution was discovered. The final weight vector was <2, 6.5, 3.75>. The misclassification error for both data sets was 0%. This was expected as the data were linearly separable. If the data weren't linearly separable, this implementation would terminate after 100,000 iterations and print the resulting misclassification errors.

```
Iterations: 5

Weight Updates: 7

Training Misclassification Error: 0.00%

Weight Vector:
        w[0] = 2.0000
        w[1] = 6.5000
        w[2] = 3.7500

Decision Boundary: (equivalent expressions)
        *) 2.0000 + 6.5000(x) + 3.7500(y) = 0
        *) y = -1.7333(x) + -0.533333

Test Data Results:
        Point #1 (0.00, 0.00)
                Classify as 1; ACTUAL = 1
        Point #2 (2.00, -1.00)
                Classify as 1; ACTUAL = 1
        Point #3 (-3.00, 3.00)
                Classify as -1; ACTUAL = -1
        Point #4 (-1.00, -1.00)
                Classify as -1; ACTUAL = -1
        Point #5 (-2.00, 1.00)
                Classify as -1; ACTUAL = -1

Testing Misclassification Error: 0.00%
sentient Project_1 $
```

Figure 1: Output of program running with described parameters and data sets

To help visualize this information, I used the following URL https://www.desmos.com/calculator. The training and test data points were plotted along with the the final solution equation. The corresponding line properly separated the points into two groups according to their class.

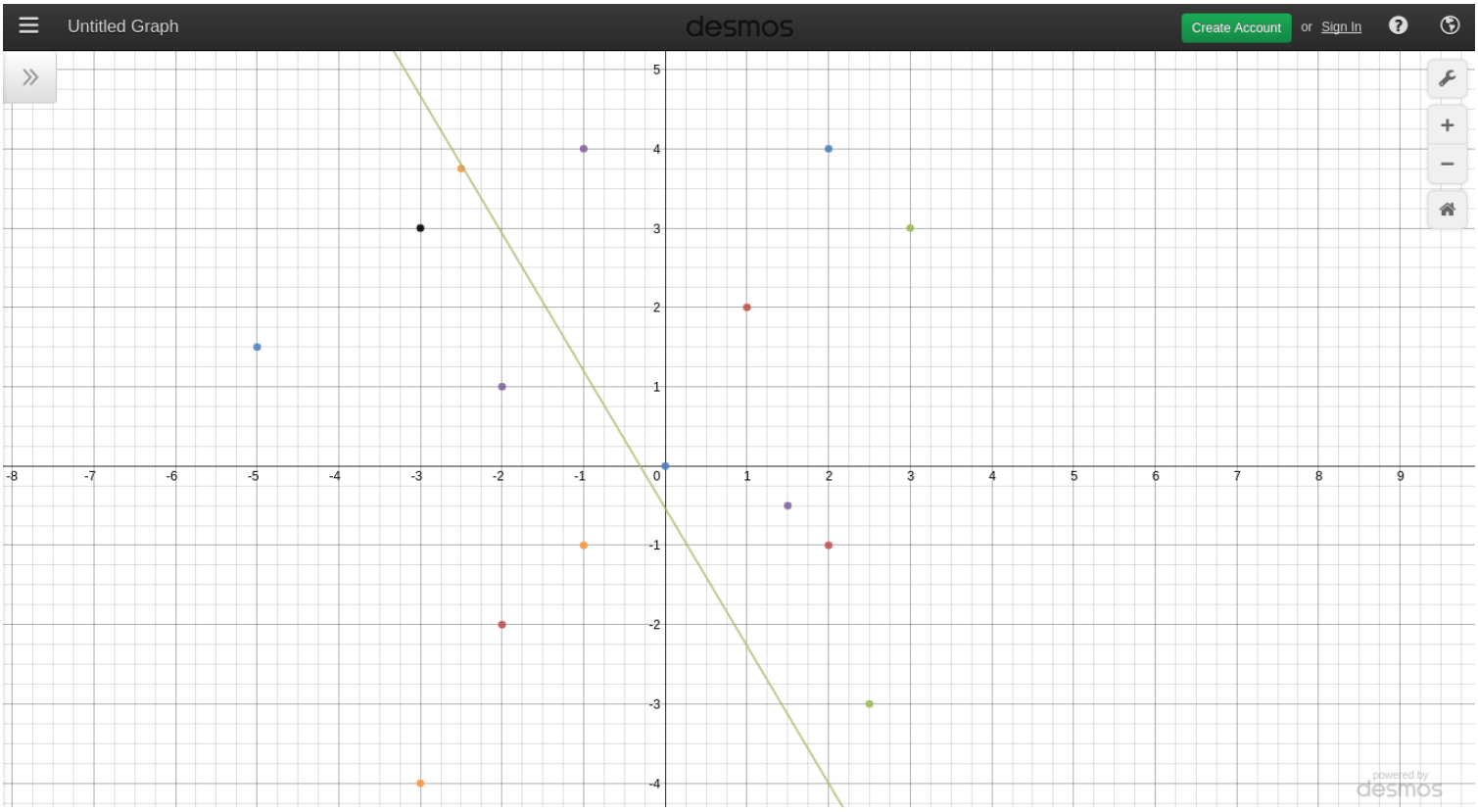

Figure 2: Training and test data separated properly

Source Code

```
/* Author:                 Dan Wilder
 *
 * School:                 University of Missouri - St. Louis
 * Semester:               Fall 2015
 * Class:                  CS 4340 Machine Learning
 * Instructor:             Uday K. Chakraborty, PhD
 *
 * Project #1:     Perceptron Learning Algorithm (PLA)
 *
 *   This is a C code implementation of the PLA. 10 data points are used
 * for training and 5 for testing. The values of these points were left
 * to the discretion of the student. I chose the points in such a way that
 * they were linearly separable. Aside from that constraint, points were
 * chosen arbitrarily. Positive constants c and k as well as initial weights
 * were also chosen. At the end of execution, the decision boundary
 * (final solution equation) will be printed to stdout along with some
 * statistics about the run and classification of test points using the
 * derived solution.
 *
 *   ~ Dan Wilder
 */
```

```
#include <stdio.h>

// Parameters:
#define MAX_ITERATIONS          100000
#define c               1
#define k               1

// Feature vector; M=2 dimensions; i.e. point on a plane
struct point {
  double x1, x2;
  int d; // True class
};

int main() {

  struct point trainingData[10] = {
    { .d = 1, .x1 = 1, .x2 = 2  },
    { .d = 1, .x1 = 2, .x2 = 4 },
    { .d = 1, .x1 = 3, .x2 = 3 },
    { .d = 1, .x1 = -1, .x2 = 4 },
    { .d = -1, .x1 = -3, .x2 = -4 },
    { .d = -1, .x1 = -2, .x2 = -2 },
    { .d = -1, .x1 = -5, .x2 = 1.5 },
    { .d = 1, .x1 = 2.5, .x2 = -3 },
    { .d = 1, .x1 = 1.5, .x2 = -0.5 },
    { .d = -1, .x1 = -2.5, .x2 = 3.75 }
  };

  struct point testData[5] = {
    { .d = 1, .x1 = 0, .x2 = 0 },
    { .d = 1, .x1 = 2, .x2 = -1 },
    { .d = -1, .x1 = -3, .x2 = 3 },
    { .d = -1, .x1 = -1, .x2 = -1 },
    { .d = -1, .x1 = -2, .x2 = 1 }
  };

  // Storage for TEST data classification: 1 or -1
  int classify[5];

  // Weight Vector
  double w[3] = {1,1,1};

  // Statistics
  int weightUpdates = 0;
  int iteration = 0;
  double trainingError = 0.0;
  double testingError = 0.0;

  // Other variables
  int misclassified, sign, j;
  struct point sample;

  /* ALGORITHM */
  do {
    misclassified = 0;

    // Loop over sample set
    for (j = 0; j < 10; j++) {
      sample = trainingData[j];
```

```
    sign = (w[0] + w[1]*sample.x1 + w[2]*sample.x2 >= 0) ? 1 : -1;

    // Point misclassified -> update weight vector
    if (sign != sample.d) {
      ++misclassified;
      w[1] = w[1] + c * sample.d * sample.x1;
      w[2] = w[2] + c * sample.d * sample.x2;
      w[0] = w[0] + c * sample.d * k;
      ++weightUpdates;
    }
  }

  trainingError = misclassified / 10.0;
  ++iteration;
 } while (misclassified && iteration < MAX_ITERATIONS);

 // Using derived solution, classify test data set
 misclassified = 0;
 for (j=0; j < 5; ++j) {
  sample = testData[j];

  classify[j] =
    (sample.x2 >= ((w[1]/-w[2])*sample.x1 + (w[0]/-w[2]))) ? 1 : -1;

  if (classify[j] != sample.d)
    ++misclassified;
 }
 testingError = misclassified / 5.0;

 /* PRINT RESULTS */
 printf("\nIterations: %d\n\nWeight Updates: %d\n", iteration, weightUpdates);
 printf("\nTraining Misclassification Error: %.2lf%%\n", trainingError*100);

 printf("\nWeight Vector:\n\tw[0] = %.4lf\n\tw[1] = %.4lf\n\tw[2] = %.4lf\n",
  w[0], w[1], w[2]);

 printf("\nDecision Boundary: (equivalent expressions)\n");
 printf("\t*) %.4lf + %.4lf(x) + %.4lf(y) = 0\n", w[0], w[1], w[2]);
 printf("\t*) y = %.4lf(x) + %lf\n", -w[1]/w[2], -w[0]/w[2]);

 printf("\nTest Data Results:\n");
 for (j=0; j < 5; ++j) {
  printf("\tPoint #%d (%.2lf, %.2lf)\n\t\tClassify as %d; ACTUAL = %d\n",
    j+1, testData[j].x1, testData[j].x2, classify[j], testData[j].d);
 }
 printf("\nTesting Misclassification Error: %.2lf%%\n", testingError*100);

 return 0;
}
```