Sustainable Data Science and Artificial Intelligence

Tutorial #6

**Instructions**

The following C code is for a neural network that learns the to convert BINARY into DECIMAL, as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| [0, 0, 0] 🡪 0 | [0, 0, 1] 🡪 1 | [0, 1, 0] 🡪 2 | [0, 1, 1] 🡪 3 |
| [1, 0, 0] 🡪 4 | [1, 0, 1] 🡪 5 | [1, 1, 0] 🡪 6 | [1, 1, 1] 🡪 7 |

A diagram of a diagram

AI-generated content may be incorrect.

The key hyperparameter (LEARNING\_RATE) are defined near the top as follows:

|  |
| --- |
| **C Programming Language** |
| #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <time.h>  // Hyperparameters  **#define LEARNING\_RATE 0.5**  #define EPOCHS 10000  // Activation function (Sigmoid)  double sigmoid(double x) {  return 1 / (1 + exp(-x));  }  // Derivative of sigmoid  double sigmoid\_derivative(double x) {  return x \* (1 - x);  }  // Random number generator for weight initialization  double random\_weight() {  return ((double)rand() / RAND\_MAX) \* 2 - 1; // Range [-1, 1]  }  int main() {  srand(time(NULL)); // Seed random generator  // Training data: 3-bit binary input and its decimal equivalent (normalized)  double input[8][3] = {  {0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1},  {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}  };  double expected\_output[8];  for (int i = 0; i < 8; i++) {  expected\_output[i] = i / 7.0; // Normalize output (0 to 1)  }  // Initialize weights and biases randomly  double hidden\_weights[3][2], hidden\_bias[2];  double output\_weights[2], output\_bias;  // Random weight initialization  for (int i = 0; i < 3; i++)  for (int j = 0; j < 2; j++)  hidden\_weights[i][j] = random\_weight();  for (int j = 0; j < 2; j++)  hidden\_bias[j] = random\_weight();  for (int k = 0; k < 2; k++)  output\_weights[k] = random\_weight();    output\_bias = random\_weight();  // Training loop  for (int epoch = 0; epoch < EPOCHS; epoch++) {  double total\_error = 0;  for (int i = 0; i < 8; i++) {  // Forward pass: Hidden Layer  double hidden\_output[2];  for (int j = 0; j < 2; j++) {  hidden\_output[j] = sigmoid(  input[i][0] \* hidden\_weights[0][j] +  input[i][1] \* hidden\_weights[1][j] +  input[i][2] \* hidden\_weights[2][j] +  hidden\_bias[j]  );  }  // Forward pass: Output Layer  double output = sigmoid(  hidden\_output[0] \* output\_weights[0] +  hidden\_output[1] \* output\_weights[1] +  output\_bias  );  // Compute error  double error = expected\_output[i] - output;  total\_error += error \* error;  // Backpropagation: Output Layer  double output\_delta = error \* sigmoid\_derivative(output);  // Backpropagation: Hidden Layer  double hidden\_delta[2];  for (int j = 0; j < 2; j++) {  hidden\_delta[j] = output\_delta \* output\_weights[j] \* sigmoid\_derivative(hidden\_output[j]);  }  // Update weights and biases (Gradient Descent)  for (int j = 0; j < 2; j++) {  output\_weights[j] += LEARNING\_RATE \* output\_delta \* hidden\_output[j];  hidden\_bias[j] += LEARNING\_RATE \* hidden\_delta[j];  }  output\_bias += LEARNING\_RATE \* output\_delta;  for (int j = 0; j < 2; j++) {  for (int m = 0; m < 3; m++) {  hidden\_weights[m][j] += LEARNING\_RATE \* hidden\_delta[j] \* input[i][m];  }  }  }  }  // Testing the trained network  printf("\nTesting the trained neural network:\n");  for (int i = 0; i < 8; i++) {  double hidden\_output[2];  // Forward pass - Hidden Layer  for (int j = 0; j < 2; j++) {  hidden\_output[j] = sigmoid(  input[i][0] \* hidden\_weights[0][j] +  input[i][1] \* hidden\_weights[1][j] +  input[i][2] \* hidden\_weights[2][j] +  hidden\_bias[j]  );  }  // Forward pass - Output Layer  double output = sigmoid(  hidden\_output[0] \* output\_weights[0] +  hidden\_output[1] \* output\_weights[1] +  output\_bias  );  printf("Input: (%.0f, %.0f, %.0f) -> Output: %.2f (Expected: %.0f)\n",  input[i][0], input[i][1], input[i][2], output \* 7, expected\_output[i] \* 7);  }  return 0;  } |

Set the LEARNING\_RATE to the following values:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

For example, at learning rate 10:

|  |  |
| --- | --- |
| **Learning\_Rate** | **Outputs** |
| 10 | Testing the trained neural network:  Input: (0, 0, 0) -> Output: 0.02 (Expected: 0)  Input: (0, 0, 1) -> Output: 1.01 (Expected: 1)  Input: (0, 1, 0) -> Output: 2.02 (Expected: 2)  Input: (0, 1, 1) -> Output: 3.01 (Expected: 3)  Input: (1, 0, 0) -> Output: 4.01 (Expected: 4)  Input: (1, 0, 1) -> Output: 5.04 (Expected: 5)  Input: (1, 1, 0) -> Output: 6.01 (Expected: 6)  Input: (1, 1, 1) -> Output: 6.98 (Expected: 7) |

1. Identify which settings give you the closest to the right answer.
2. If the best setting is N, then test N – 0.5. and N + 0.5

**Submission Instructions**

Please submit in the following two ways

|  |
| --- |
| Please e-mail me your completed solution, and with filename:  *Familyname\_FirstName\_TU850\_1\_SDSaAI\_Tutorial6.docx*  e.g.  *Gordon\_Damian\_TU850\_1\_SDSaAI\_ Tutorial6.docx*  e-mail to [Damian.X.Gordon@TUDublin.ie](mailto:Damian.X.Gordon@TUDublin.ie) with subject heading as follows:  [DT850/1] SDSaAI Tutorial #6 |
| Please submit into Brightspace in:   * Assessment   + Assignments     - Tutorial #6 |

For the email, please include the following message:

|  |
| --- |
| **Damian,**  **I am a student in your Sustainable DS & AI (CMPU1040) class in the BSc in DS & AI (DT850/1).**    **Please find attached Tutorial #6**  **Regards,**  **Your Name**  **Student Number.**  **DT580, BSc in DS & AI** |