

# Video Tutorial

[https://youtu.be/uLMxC\\_1gIJM](https://youtu.be/uLMxC_1gIJM)

# Download Base Project

[https://github.com/zenzen0014/nn\\_deep\\_learn/tree/master/scripts](https://github.com/zenzen0014/nn_deep_learn/tree/master/scripts)

# Download ATOM

<https://atom.io/>

# Index.html

```
<script type="text/javascript">
  let nn;//neural network

  $("#start").click(function(){

    lr = $("#lr").val();
    e = $("#epoch").val();

    generate_ai(lr, e);
  })

  function generate_ai(lr, e){//lr learning rate, e epoch
    let training_data = [{
      inputs: [0.05, 0.1],
      outputs: [0.99]
    }];
    let start_time = Date.now();
```

# Index.html

```
let start_time = Date.now();

nn = new NeuNet(
  2, 2, 1, 1, // i, h1, h2, T
  [
    [0.2, -0.3], //w1 w2
    [0.15, -0.5] //w3 w4
  ],
  [1, 1], //b1
  [
    [-0.4, 0.3] //w5 w6
  ],
  [0.5], //b2
  [
    [0.25] //w7
  ],
  [1] //b3
)
```

# Index.html

```
for(n = 0; n < e; n++){ //e
    for(i = 0; i < 50; i++){
        a = nn.training(training_data[0].inputs, training_data[0].outputs)
    }
    nn.setlearningRate(lr); // lr
    y = nn.prediction(training_data[0].inputs);
    let elapsed_time = (Date.now() - start_time)/1000;

    if(n == (e-1)){
        $("#mse").val(`${a[0]}.toFixed(4)} %`);
        $("#output").val(`${y[0]}.toFixed(4)}`);
        $("#etime").val(`${elapsed_time} seconds`);
    }
}
</script>
```

# sketch.js

sketch.js

```
class ActivationFunction {  
  constructor(func, dfunc) {  
    this.func = func;  
    this.dfunc = dfunc;  
  }  
}  
  
let sigmoid = new ActivationFunction(  
  x => 1 / (1 + Math.exp(-x)),  
  y => y * (1 - y)  
);
```

# sketch.js

sketch.js

```
class NeuNet {
  constructor(
    ilayer,
    hlayer,
    Hlayer,
    olayer,
    weight_ih = null,
    hbias = null,
    weight_hh = null,
    Hbias = null,
    weight_ho = null,
    obias = null
  ) {
    if (ilayer instanceof NeuNet) {
      let lyr = ilayer;
      this.input_nodes = lyr.input_nodes;
      this.hidden_nodes = lyr.hidden_nodes;
      this.Hidden_nodes = lyr.Hidden_nodes;
      this.output_nodes = lyr.output_nodes;

      this.weight_ih = lyr.weight_ih.copy();
      this.weight_hh = lyr.weight_hh.copy();
      this.weight_ho = lyr.weight_ho.copy();

      this.hbias = lyr.hbias.copy();
      this.Hbias = lyr.Hbias.copy();
      this.obias = lyr.obias.copy();
    } else {
      this.setLearningRate();
      this.setActivationFunction();
    }
  }
}
```

# sketch.js

```
sketch.js

class NeuNet {
  constructor(=) {
    if (ilayer instanceof NeuNet) {=
    } else {
      this.input_nodes = ilayer;
      this.hidden_nodes = hlayer;
      this.Hidden_nodes = Hlayer;
      this.output_nodes = olayer;
      this.weight_ih = new Matrix(this.hidden_nodes, this.input_nodes);
      this.weight_hh = new Matrix(this.Hidden_nodes, this.hidden_nodes);
      this.weight_ho = new Matrix(this.output_nodes, this.Hidden_nodes);
      this.hbias = new Matrix(this.hidden_nodes, 1);
      this.Hbias = new Matrix(this.Hidden_nodes, 1);
      this.obias = new Matrix(this.output_nodes, 1);

      let wih = Matrix.subtract_array(weight_ih, this.hidden_nodes, this.input_nodes);
      let bih = Matrix.fromArray(hbias);
      let whh = Matrix.subtract_array(weight_hh, this.Hidden_nodes, this.hidden_nodes);
      let bhh = Matrix.fromArray(Hbias);
      let who = Matrix.subtract_array(weight_ho, this.output_nodes, this.Hidden_nodes);
      let bho = Matrix.fromArray(obias);

      this.weight_ih = wih;
      this.weight_hh = whh;
      this.weight_ho = who;
      this.hbias = bih;
      this.Hbias = bhh;
      this.obias = bho;
    }
    this.setLearningRate();
    this.setActivationFunction();
  }
}
```

# sketch.js

```
setLearningRate(LearningRate = 0.1) {  
  this.LearningRate = LearningRate;  
}  
  
setActivationFunction(func = sigmoid) {  
  this.ActFunc = func;  
}  
  
prediction(input_array) {  
  let inputs = Matrix.fromArray(input_array);  
  
  let hidden = Matrix.multiply(this.weight_ih, inputs);  
  hidden.add(this.hbias);  
  hidden.map(this.ActFunc.func);  
  
  let Hidden = Matrix.multiply(this.weight_hh, hidden);  
  Hidden.add(this.Hbias);  
  Hidden.map(this.ActFunc.func);  
  
  let output = Matrix.multiply(this.weight_ho, Hidden);  
  output.add(this.obias);  
  output.map(this.ActFunc.func);  
  
  return output.toArray();  
}
```



# sketch.js

sketch.js

```
training(input_array, target_array) {  
  let inputs = Matrix.fromArray(input_array);  
  
  let hidden = Matrix.multiply(this.weight_ih, inputs);  
  hidden.add(this.hbias);  
  hidden.map(this.ActFunc.func);  
  
  let Hidden = Matrix.multiply(this.weight_hh, hidden);  
  Hidden.add(this.Hbias);  
  Hidden.map(this.ActFunc.func);  
  
  let outputs = Matrix.multiply(this.weight_ho, Hidden);  
  outputs.add(this.obias);  
  outputs.map(this.ActFunc.func);  
  
  let targets = Matrix.fromArray(target_array);  
  // Calculate the error ==> ERROR = TARGETS - OUTPUTS  
  let output_errors = Matrix.subtract(targets, outputs);  
  
  // let gradient = outputs * (1 - outputs);  
  let gradients = Matrix.map(outputs, this.ActFunc.dfunc);  
  gradients.multiply(output_errors);  
  gradients.multiply(this.LearningRate);  
  
  // Calculate deltas  
  let Hidden_T = Matrix.transpose(Hidden);  
  let weight_ho_deltas = Matrix.multiply(gradients, Hidden_T);  
  this.weight_ho.add(weight_ho_deltas);  
  this.obias.add(gradients);  
}
```

# sketch.js

sketch.js

```
// let gradient = outputs * (1 - outputs);
let gradients = Matrix.map(outputs, this.ActFunc.dfunc);
gradients.multiply(output_errors);
gradients.multiply(this.LearningRate);

// Calculate deltas
let Hidden_T = Matrix.transpose(Hidden);
let weight_ho_deltas = Matrix.multiply(gradients, Hidden_T);
this.weight_ho.add(weight_ho_deltas);
this.obias.add(gradients);

// Calculate the hidden layer errors
let who_t = Matrix.transpose(this.weight_ho);
let Hidden_errors = Matrix.multiply(who_t, output_errors);

// Calculate hidden gradient
let Hidden_gradient = Matrix.map(Hidden, this.ActFunc.dfunc);
Hidden_gradient.multiply(Hidden_errors);
Hidden_gradient.multiply(this.LearningRate);

// Calculate deltas
let hidden_T = Matrix.transpose(hidden);
let weight_hh_deltas = Matrix.multiply(Hidden_gradient, hidden_T);
this.weight_hh.add(weight_hh_deltas);
this.Hbias.add(Hidden_gradient);

// Calculate the hidden layer errors
let whh_t = Matrix.transpose(this.weight_hh);
let hidden_errors = Matrix.multiply(whh_t, output_errors);
```

# sketch.js

sketch.js

```
// Calculate deltas
let hidden_T = Matrix.transpose(hidden);
let weight_hh_deltas = Matrix.multiply(Hidden_gradient, hidden_T);
this.weight_hh.add(weight_hh_deltas);
this.Hbias.add(Hidden_gradient);

// Calculate the hidden layer errors
let whh_t = Matrix.transpose(this.weight_hh);
let hidden_errors = Matrix.multiply(whh_t, output_errors);

// Calculate hidden gradient
let hidden_gradient = Matrix.map(hidden, this.ActFunc.dfunc);
hidden_gradient.multiply(hidden_errors);
hidden_gradient.multiply(this.LearningRate);

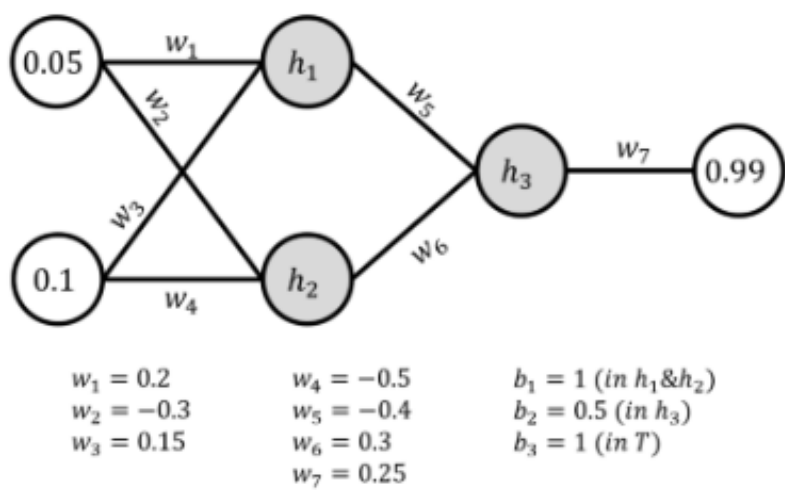
// Calculate input->hidden deltas
let inputs_T = Matrix.transpose(inputs);
let weight_ih_deltas = Matrix.multiply(hidden_gradient, inputs_T);

this.weight_ih.add(weight_ih_deltas);
this.hbias.add(hidden_gradient);

$("#w1").val(this.weight_ih.data[0][0].toFixed(4));
$("#w2").val(this.weight_ih.data[0][1].toFixed(4));
$("#w3").val(this.weight_ih.data[1][0].toFixed(4));
$("#w4").val(this.weight_ih.data[1][1].toFixed(4));
$("#w5").val(this.weight_hh.data[0][0].toFixed(4));
$("#w6").val(this.weight_hh.data[0][1].toFixed(4));
$("#w7").val(this.weight_ho.data[0][0].toFixed(4));

return output_errors.toArray();
}
```

# Run index.html



Learning Rate

0.1

Epoch

200

START

Output

0.9799

MSE

0.0101 %

Elapsed Time

0.544 seconds

W1

0.2522

W2

-0.1955

W3

0.2643

W4

-0.2715

W5

0.7044

W6

1.5971

W7

1.4512

Submit tugas via email :

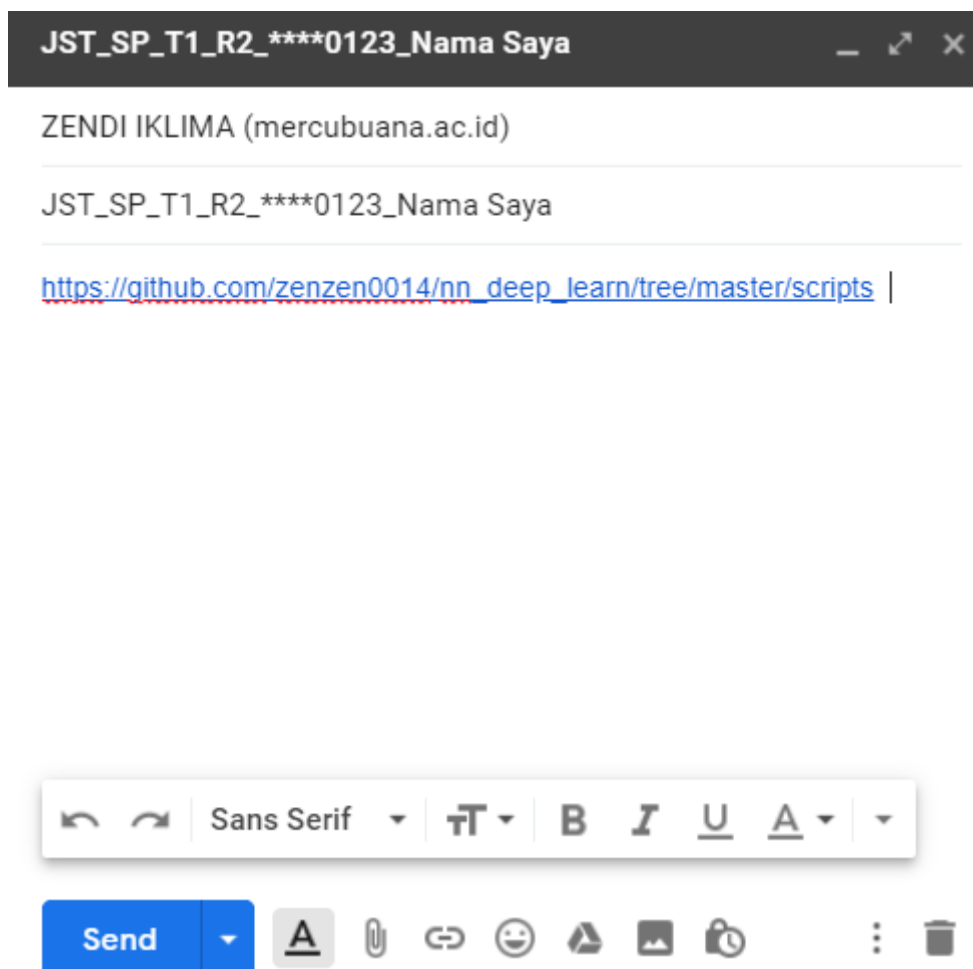
[zendi.iklima@mercubuana.ac.id](mailto:zendi.iklima@mercubuana.ac.id)

Dengan menyertakan link github dan Screenshoot hasil running index.html anda masing-masing dengan mail subject:

**JST\_SP\_T1\_R2\_NIM\_NAMA**

Paling lambat tanggal **30 Agustus 2019 23:59**

*Contoh pengiriman email*



# QUIZ 1

Tentukanlah solusi sebuah fungsi  $f(x) = 2x^2 - 3x + 5$  dengan jumlah data 200, 300 dan 500 data menggunakan MLP dengan arsitektur seperti pada TUGAS 1. running MLP dengan learning rate 0.1 dan 0.3

Submit QUIZ via email :

[zendi.iklima@mercubuana.ac.id](mailto:zendi.iklima@mercubuana.ac.id)

Dengan menyertakan link github dan Screenshoot hasil running index.html anda masing-masing dengan mail subject:

**JST\_SP\_Q1\_R2\_NIM\_NAMA**

Paling lambat tanggal **30 Agustus 2019 23:59**