Neural Networks II

Mark A. Austin

austin@umd.edu

ENCE 688P, Spring Semester 2021
University of Maryland

March 24, 2021

Overview

Part 03

Quick Review

Perceptron Model as a Linear Classifier

General Classification Capabilities (Lippmann, 1987)

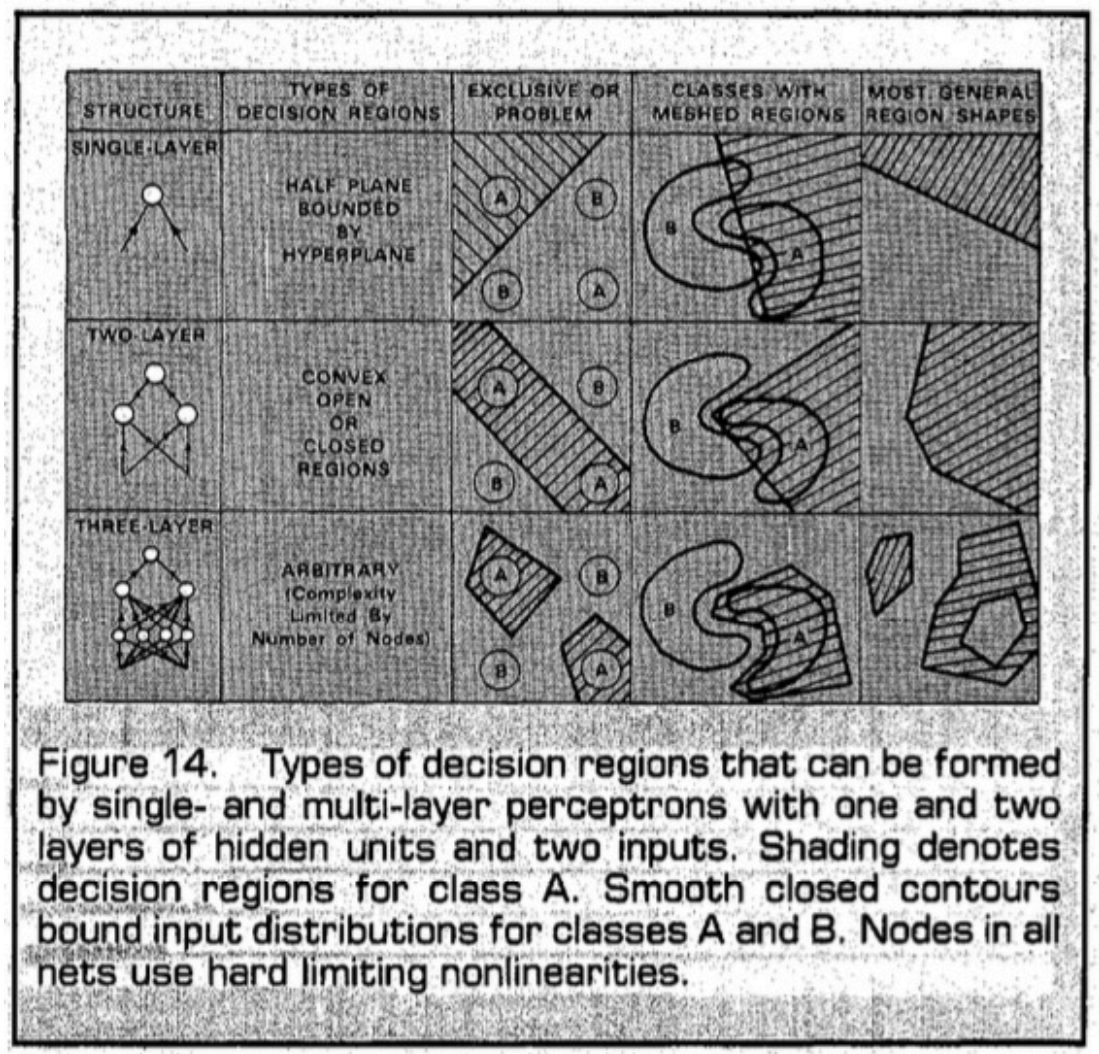
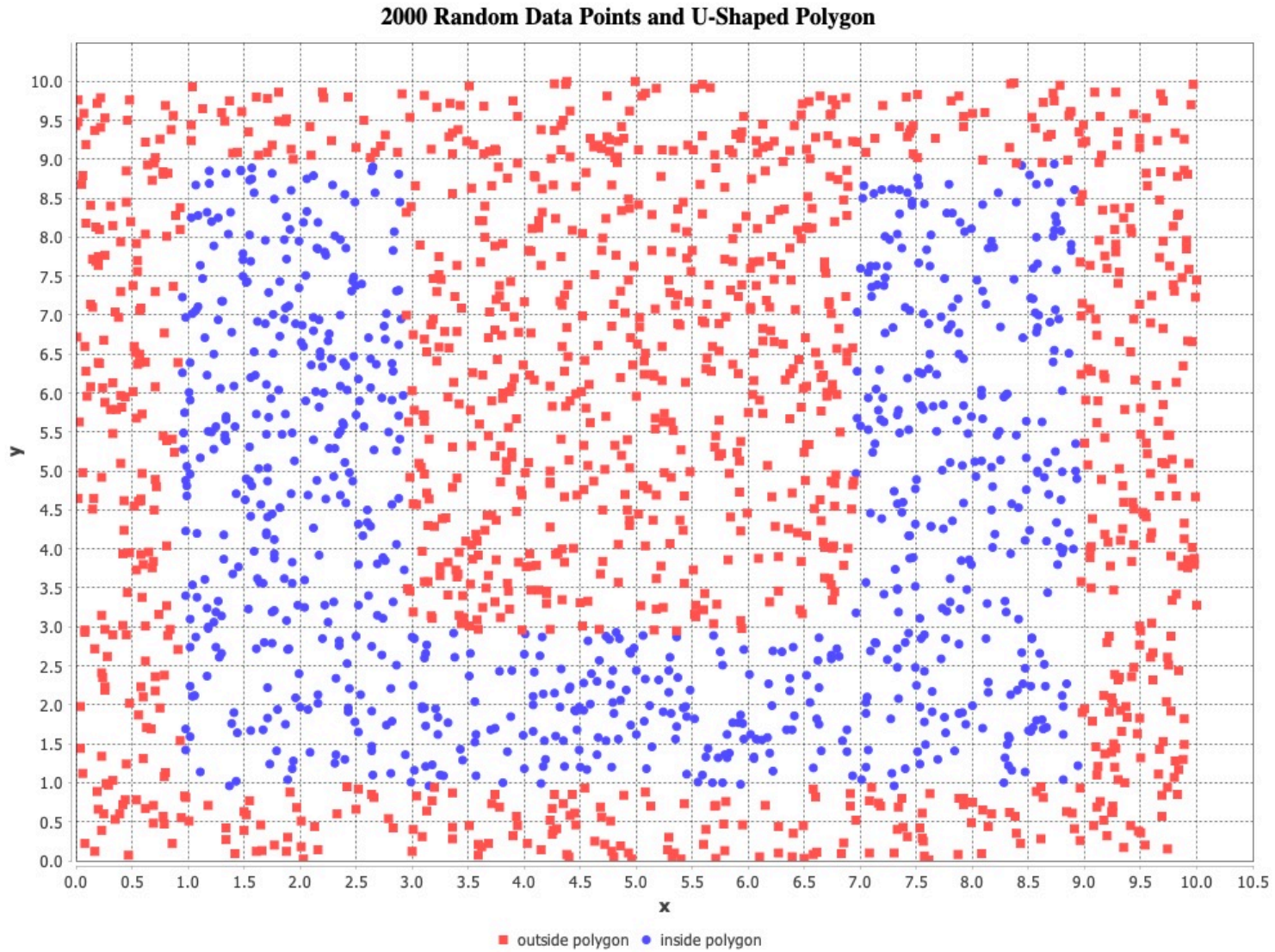


Figure 14. Types of decision regions that can be formed by single- and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth closed contours bound input distributions for classes A and B. Nodes in all nets use hard limiting nonlinearities.

Networks with

Two Hidden Layers

Example 8. Points in U-Shaped Polygon



Example 8. Points in U-Shaped Polygon

Problem Description:

- Consider a 10×10 grid containing a smaller U-shaped polygon.
- We wish to train a neural network to determine whether or not a specific coordinate is inside or outside the letter U shape.
- 2,000 coordinate points are generated at random. The red dots lie outside the U-shaped polygon; the blue dots are inside.
- This is the training data for our neural network.
- A neural network with only one hidden layer cannot capture the non-convex shape. Thus, the purpose of this example is to see use of two layers solves the problem.

Example 8. Points in U-Shaped Polygon

DL4J: Read training dataset ...

```
1       // Read polygon U-Shaped data ...
2
3       double[][] x = DataUtils.readInputsFromFile( "data/polygon-u-shape-data.txt");
4       double[][] t = DataUtils.readInputsFromFile( "data/polygon-u-shape-outcome.txt");
```

DL4J: Scale training dataset to [0,1] range ...

```
1       // Scale coordinates from [0,10] --> [0,1] ....
2
3       for (int i = 0; i < x.length; i = i + 1 ) {
4          x[i][0] = x[i][0]/10.0;
5          x[i][1] = x[i][1]/10.0;
6       }
```

Scaling the dataset from [0,10] range to [0,1] range helps to avoid vanishing gradient problem.

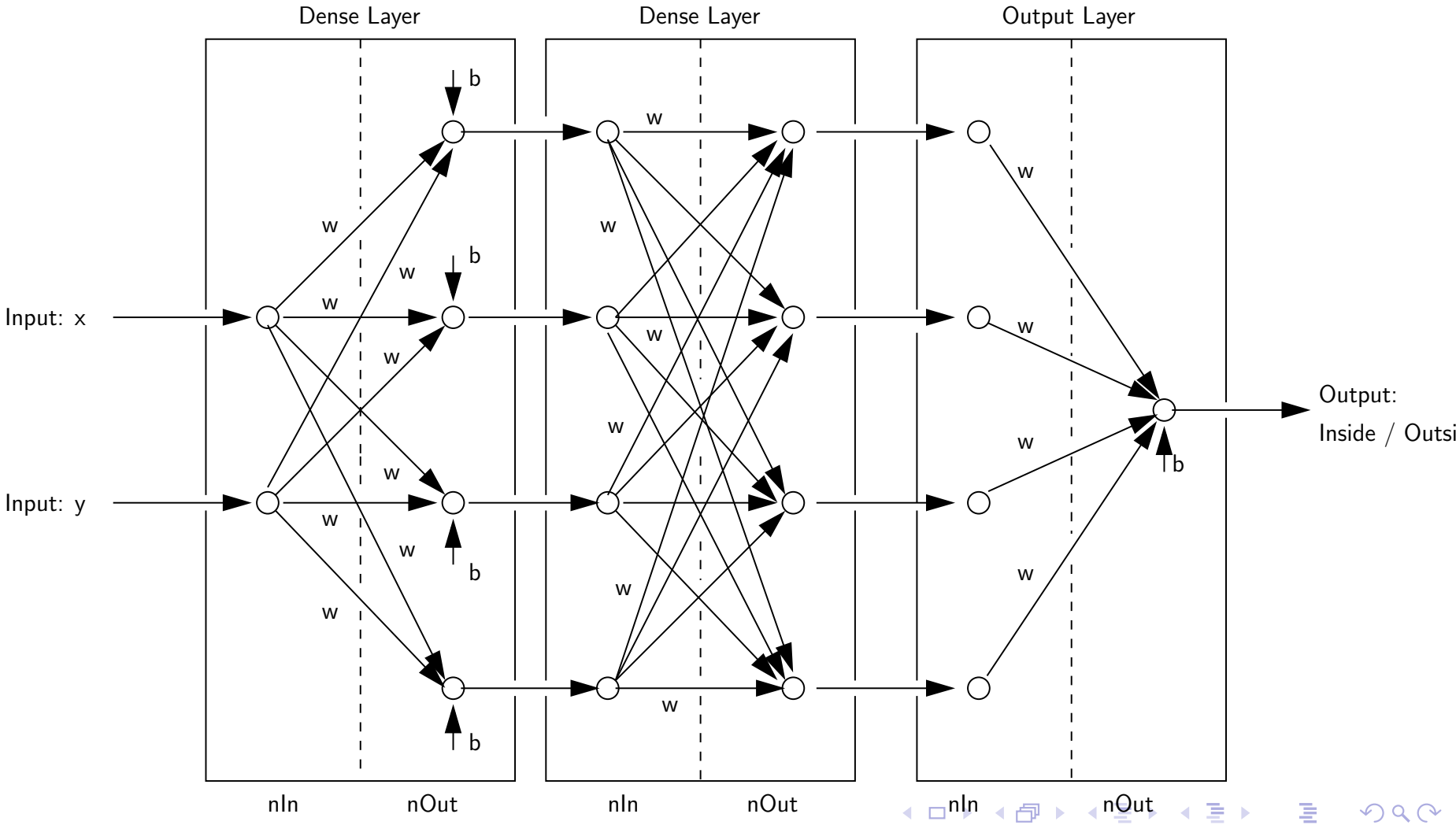
Example 8. Points in U-Shaped Polygon

DL4J: Create Network Configuration:

```
1   MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
2       .updater(new Sgd(0.01))
3       .seed(seed)
4       .biasInit(0) // Init the bias with 0 - empirical value, too
5       .miniBatch(false)
6       .list()
7       .layer( new DenseLayer.Builder()
8            .nIn(2).nOut(4)
9            .activation(Activation.SIGMOID)
10           .weightInit(WeightInit.DISTRIBUTION)
11           .build())
12      .layer( new DenseLayer.Builder()
13           .nIn(4).nOut(4)
14           .activation(Activation.SIGMOID)
15           .weightInit(WeightInit.DISTRIBUTION)
16           .build())
17      .layer( new OutputLayer.Builder(LossFunctions.LossFunction.MSE )
18           .nIn(4).nOut(1)
19           .activation(Activation.SIGMOID)
20           .weightInit(WeightInit.DISTRIBUTION)
21           .build())
22      .pretrain(false)
23      .backprop(true)
24      .build();
```

Example 8. Points in U-Shaped Polygon

DL4J: Network Model (nIn, nOut)



Example 8. Points in U-Shaped Polygon

DL4J: Summary of Network Model (4 nodes on hidden layer)

```
LayerName (LayerType)    nIn,nOut    TotalParams    ParamsShape
=================================================================
   layer0 (DenseLayer)      2,4         12          W:{2,4}, b:{1,4}
   layer1 (DenseLayer)      4,4         20          W:{4,4}, b:{1,4}
   layer2 (OutputLayer)     4,1         5           W:{4,1}, b:{1,1}
-----------------------------------------------------------------
   Total Parameters:  37   Trainable Parameters:  37
```

DL4J: Summary of Network Model (12 nodes on hidden layer)

```
LayerName (LayerType)    nIn,nOut    TotalParams    ParamsShape
=================================================================
   layer0 (DenseLayer)      2,12        36          W:{2,12}, b:{1,12}
   layer1 (DenseLayer)      12,12       156         W:{12,12}, b:{1,12}
   layer2 (OutputLayer)     4,1         5           W:{12,1}, b:{1,1}
-----------------------------------------------------------------
   Total Parameters:  205  Trainable Parameters:  205
```

Example 8. Points in U-Shaped Polygon

DL4J: Training the Network (4 nodes on hidden layers 1 and 2) ...

```
16:38:21.779 Score at iteration 0 is 535.1254272460938
16:38:22.945 Score at iteration 250 is 526.7112426757812
16:38:23.796 Score at iteration 500 is 533.8363037109375
16:38:24.558 Score at iteration 750 is 540.9840087890625
16:38:25.268 Score at iteration 1000 is 538.2188720703125
16:38:25.985 Score at iteration 1250 is 534.0460815429688

  ... lines of output removed ...

16:41:34.715 Score at iteration 98750 is 116.39311981201172
16:41:35.196 Score at iteration 99000 is 115.5108642578125
16:41:35.693 Score at iteration 99250 is 116.2978515625
16:41:36.160 Score at iteration 99500 is 115.38568878173828
16:41:36.621 Score at iteration 99750 is 116.22488403320312
16:41:37.097 Score at iteration 100000 is 115.29683685302734
```

Example 8. Points in U-Shaped Polygon

DL4J: Training the Network (8 nodes on hidden layers 1 and 2) ...

```
16:13:09.127 Score at iteration 0 is 854.4296875

... lines of output removed ...

16:17:47.179 Score at iteration 99750 is 26.462177276611328
16:17:47.988 Score at iteration 100000 is 39.20866775512695
```

DL4J: Training the Network (12 nodes on hidden layers 1 and 2) ...

```
16:28:16.513 Score at iteration 0 is 1169.819091796875

  ... lines of output removed ...

16:33:34.319 Score at iteration 99750 is 18.23417091369629
16:33:35.162 Score at iteration 100000 is 3.900997638702392 6
```

Example 8. Points in U-Shaped Polygon

DL4J: Weights and Bias Values (4 nodes on hidden layers)

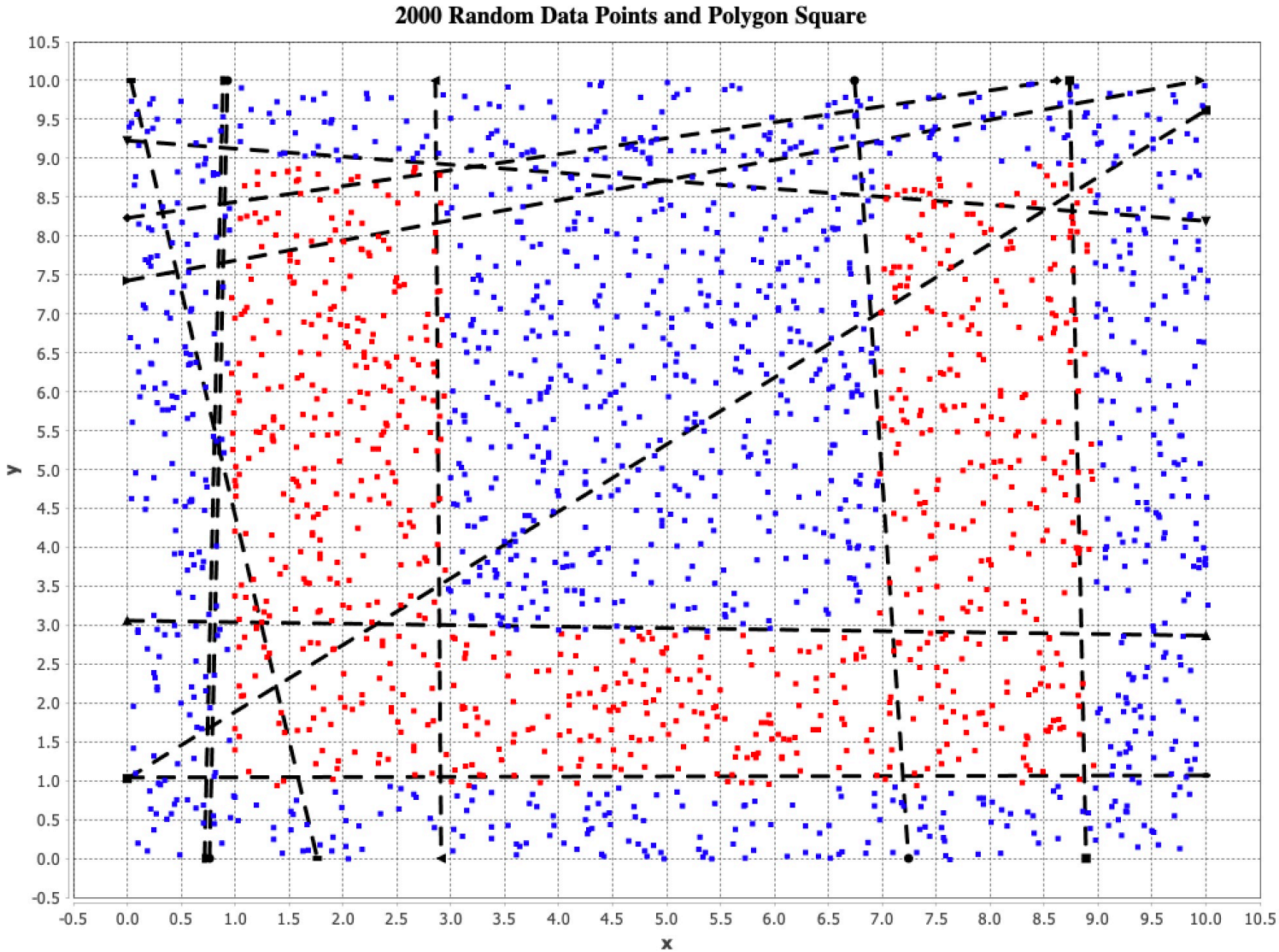
```
================================================================
Layer 0 weights: [[  -33.6689,    38.2139,     0.1660,   -0.0018],
                  [    0.7178,     1.3214,    62.7854,  -82.9784]]
Layer 0 biases:  [[    5.1473,   -31.9166,   -18.6890,    8.3968]]
----------------------------------------------------------------

Layer 1 weights: [[  -16.0345,    82.5613,    -8.2395,  -53.1885],
                  [  -15.8898,    83.7063,    -5.4736,  -52.4685],
                  [  -33.8128,    -0.2496,   -10.7231,    3.3472],
                  [   -5.2890,   -47.5059,    -6.5428,  -75.1653]]
Layer 1 biases:  [[   14.5033,    -0.1769,     1.9136,   48.5209]]
----------------------------------------------------------------

Layer 2 weights:  [   13.1383,    12.1596,     6.3566,    7.4789 ]
Layer 2 biases:      -17.7825
================================================================
```

Example 8. Points in U-Shaped Polygon (4 nodes)

Example 8. Points in U-Shaped Polygon (12 nodes)



Example 8. Points in U-Shaped Polygon

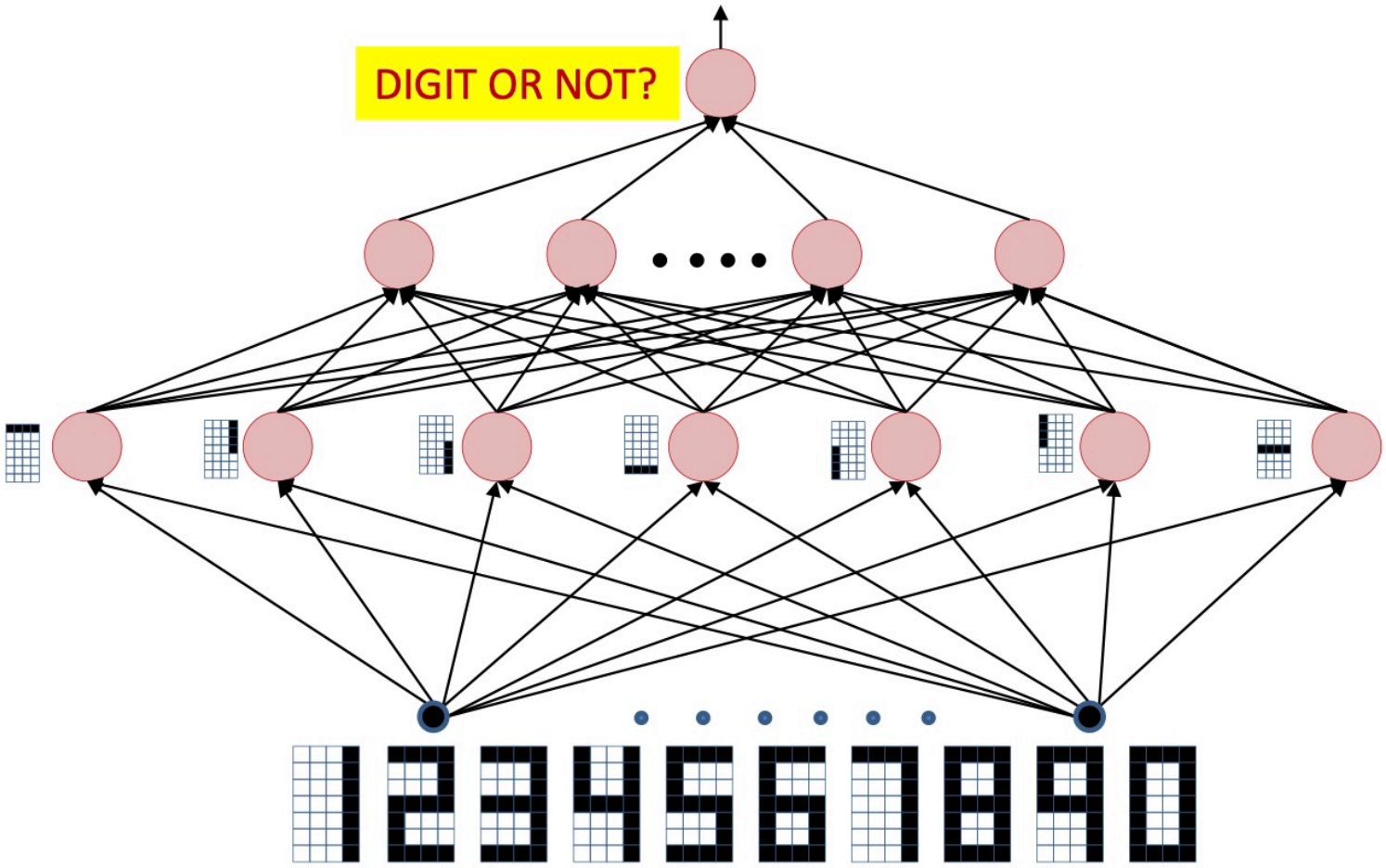
DL4J: Evaluation Metrics (4, 8 and 12 nodes on hidden layers)

Metric	4 nodes	8 nodes	12 nodes
Accuracy:	0.9200	0.9815	0.9985
Precision:	0.9147	0.9817	0.9985
Recall:	0.9217	0.9799	0.9983
F1 Score:	0.9036	0.9769	0.9981

DL4J: Confusion Matrix (4, 8 and 12 nodes on hidden layers)

4 nodes		8 nodes		12 nodes		
0	1	0	1	0	1	
1090	104 |	1180	14 |	1193	1 |	0 = 0 <-- It works!
56	750 |	23	783 |	2	804 |	1 = 1

Example 9. Neural Network for Digit Recognition



Example 9. Neural Network for Digit Recognition

Pixel-Level Synthesis of Data

- Digits are defined on a (6×4) grid. Total number of possible permutations $= 2^{24} = 16.77$ million.
- Need to think differently because this approach is not scalable.

Multi-Layer Organization of Data

- ....
- ....
- ....
- ....

References

- Lippmann R.P., An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April 1987.
- Bhiksha R., Introduction to Neural Networks, Lisbon Machine Learning School, June, 2018.
- Sun J., Fundamental Belief: Universal Approximation Theorems, Computer Science and Engineering, University of Minnesota, Twin Cities, 2020.