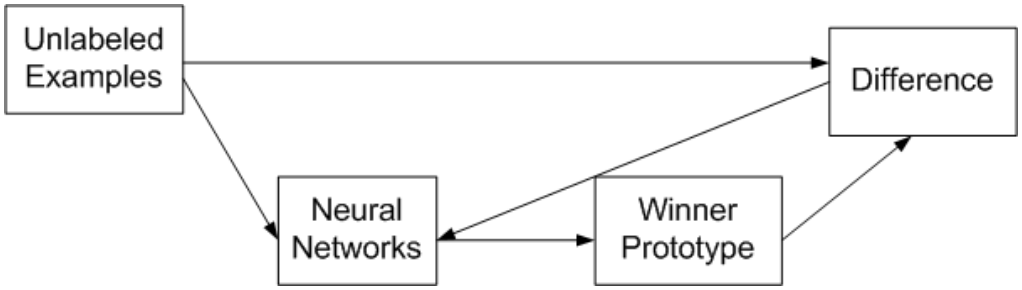
Initial Kohonen Network

Information Security and Machine Learning Lab, Hongik University, South Korea

Introduction

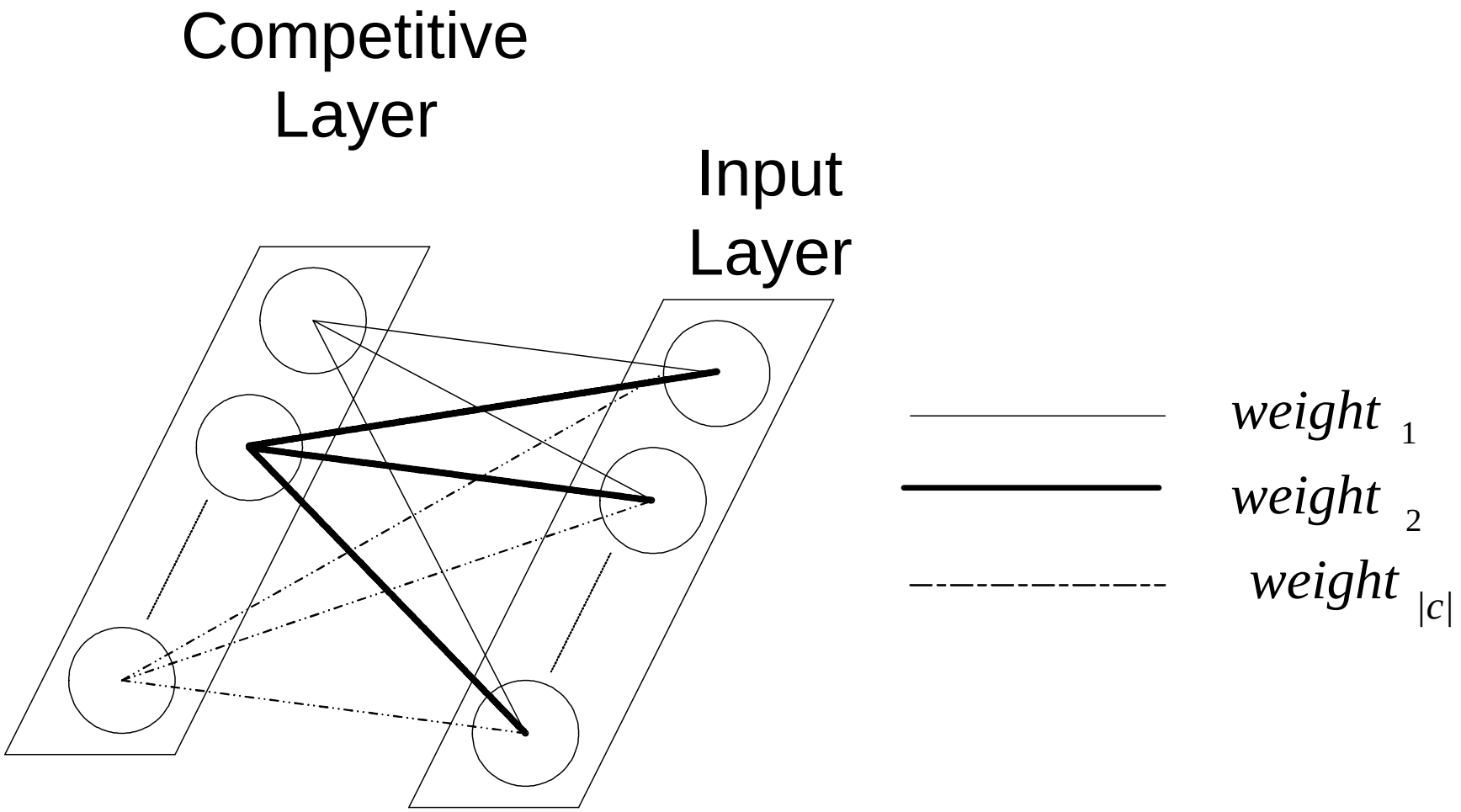
- Initial Kohonen Network is Unsupervised Neural Networks



- Weight Update Rule

$$w_{ji} = w_{ji} + \eta y_j \left(x_i - w_{ji} \right)$$

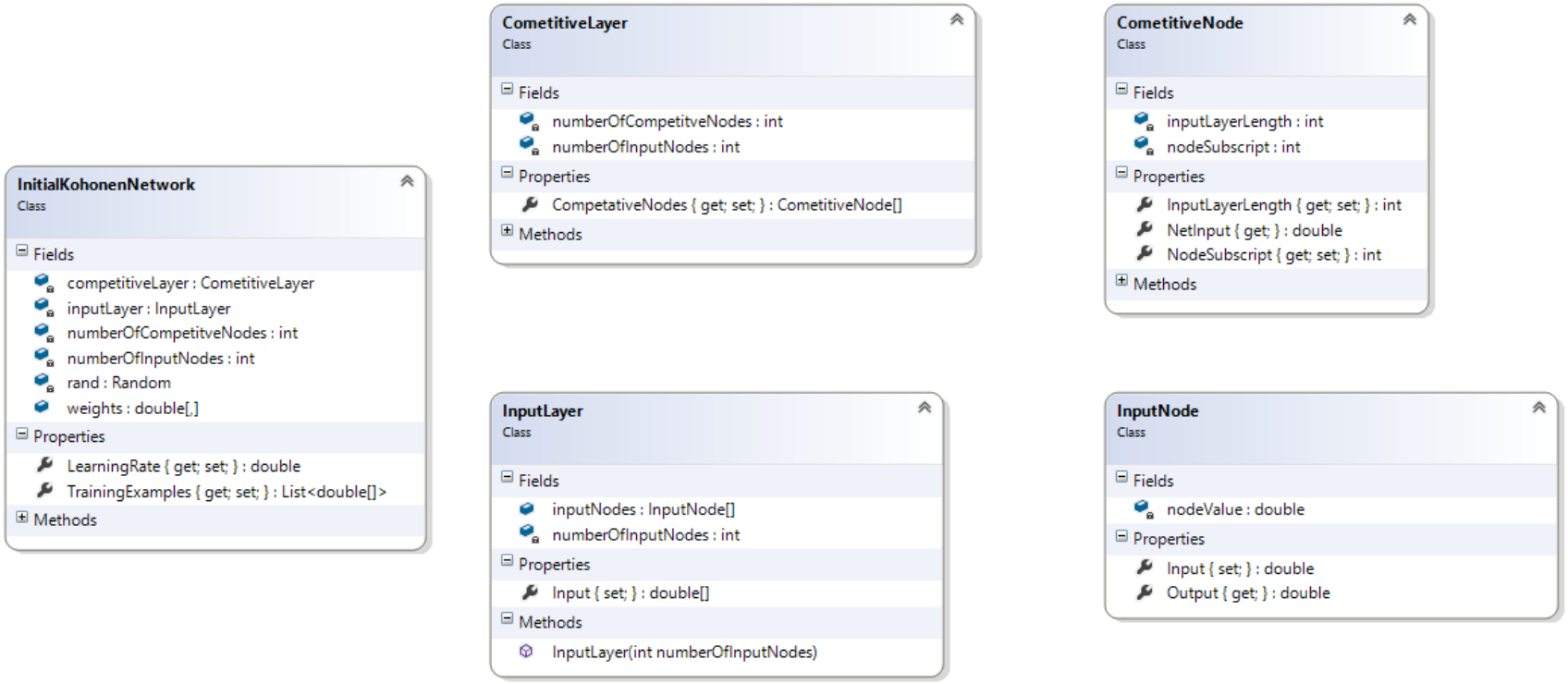
Introduction



Introduction

- Input
  - D-dimensional input vector
- Output
  - Optimized weight vectors
- Winner criteria
  - Net input
- Number of winner Nodes
  - Single

Class Diagram



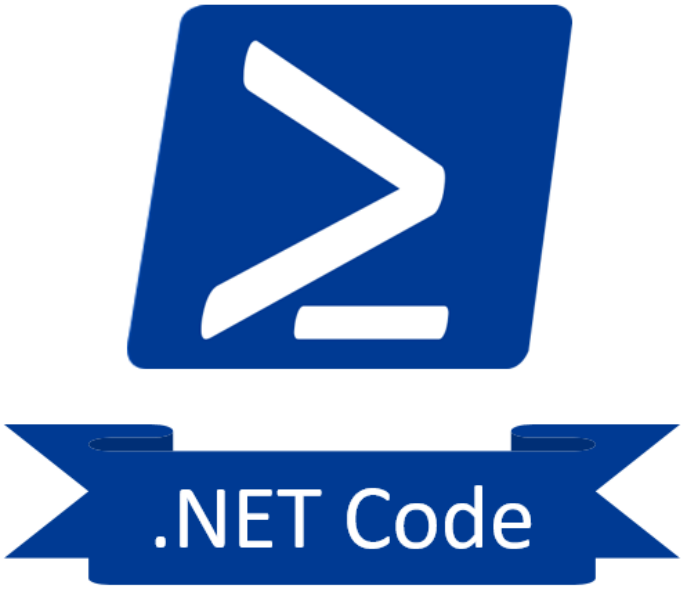
Method Implementation

```
public void Learn()
{
    if (TrainingExamples == null || TrainingExamples.Count == 0)
    {
        throw new Exception("please set training examples to start learning procss of the network");
    }
    PrintWeightMatrix();

    WinnerNode winner = new WinnerNode();
    for (int trainingIndx =0; trainingIndx < TrainingExamples.Count; trainingIndx++)
    {
        double[] trainingExample = TrainingExamples[trainingIndx];
        inputLayer.Input = trainingExample;

        winner.subscript = 0;
        winner.netInput = competitiveLayer.CompetativeNodes[0].NetInput;
        PrintCompetativeNodeStatus(winner.netInput, winner.subscript,trainingIndx);
        //List<CometitiveNode> competativeNodes = new List<CometitiveNode>(competitiveLayer.CompetativeNodes);
        double ithNetInput = 0;
        for (int i = 1; i < numberOfCompetitveNodes; i++)
        {
            ithNetInput = competitiveLayer.CompetativeNodes[i].NetInput;
            PrintCompetativeNodeStatus(ithNetInput, i,trainingIndx);
            if (winner.netInput < ithNetInput)
            {
                winner.subscript = i;
                winner.netInput = ithNetInput;

            }
        }

        for (int j = 0;j < numberOfInputNodes;j++)
        {
            weights[winner.subscript, j] += LearningRate * (trainingExample[j] - weights[winner.subscript, j]);

        }

        Console.WriteLine("Winner node: " + winner.subscript);

        PrintEndLine();
    }
    PrintWeightMatrix();

}
```

Method Implementation

```
public double NetInput
{
    get
    {
        double netInput = 0;
        for(int j=0;j< inputLayerLength;j++){
            netInput += InitialKohonenNetwork.weights[nodeSubscript, j] * InputLayer.inputNodes[j].Output;
        }

        return netInput;
    }
}
```

Demo



PowerShell

Conclusion

- Initial kohonen network can be used for unsupervised clustering in 2D
- Each value is classified in single class