```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Neural_Network
{
    abstract class Neuron{
        protected List<Synapse> incomingSynapses = new List<Synapse>();
        protected List<Synapse> outgoingSynapses = new List<Synapse>();

        protected double learningRate = 0.3;
        public double delta = 0.0;

        public abstract void learn(TrainingInstance t);
        public virtual void setDelta(TrainingInstance t) { }

        protected double currentOutputVoltage;
        public double getCurrentOutputValue() {
            return activate(excitation());
        }

        public virtual void setStaticOutput(double v){}

        public virtual void addIncomingSynapse(Synapse s, double initWeight) {
            incomingSynapses.Add(s);
            s.weight = initWeight;
        }

        public virtual void addOutgoingSynapse(Synapse s) {
            outgoingSynapses.Add(s);
        }

        protected double excitation() {
            double sum = 0.0;
            foreach (Synapse s in incomingSynapses) {
                sum += s.voltage * s.weight;
            }
            return sum;
        }

        protected virtual double activate(double sum) { return sum; }
        protected virtual double activateDifferentiated(double sum) { return 1; }

        public virtual void calc() {
            currentOutputVoltage = activate(excitation());
            foreach (Synapse s in outgoingSynapses) {
                s.voltage = currentOutputVoltage;
            }
        }
    }
}

```