```cpp
#include "NN.h"

namespace SoilMath
{
    NN::NN()
    {
        beta = 0.666;
    }

    NN::NN(uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)
    {
        // Set the number of neurons in the network
        inputNeurons = inputneurons;
        hiddenNeurons = hiddenneurons;
        outputNeurons = outputneurons;
        // Reserve the vector space
        iNeurons.reserve(inputNeurons + 1); // input neurons + bias
        hNeurons.reserve(hiddenNeurons + 1); // hidden neurons + bias
        oNeurons.reserve(outputNeurons); // output neurons

        beta = 0.666;
    }


    NN::~NN()   {   }

    void NN::LoadState(string filename)
    {
        std::ifstream ifs(filename.c_str());
        boost::archive::xml_iarchive ia(ifs);
        ia >> boost::serialization::make_nvp("NeuralNet", *this);
    }

    void NN::SaveState(string filename)
    {
        std::ofstream ofs(filename.c_str());
        boost::archive::xml_oarchive oa(ofs);
        oa << boost::serialization::make_nvp("NeuralNet", *this);
    }

    Predict_t NN::PredictLearn(ComplexVect_t input, Weight_t inputweights, Weight_t hiddenweights, uint32_t inputneurons, uint32_t
      hiddenneurons, uint32_t outputneurons)
```

```
42      {
43          NN neural(inputneurons, hiddenneurons, outputneurons);
44          neural.SetInputWeights(inputweights);
45          neural.SetHiddenWeights(hiddenweights);
46          return neural.Predict(input);
47      }
48
49      Predict_t NN::Predict(ComplexVect_t input)
50      {
51          if (input.size() != inputNeurons) { throw Exception::MathException("Size of input Neurons Exception!"); }
52
53          iNeurons.clear();
54          hNeurons.clear();
55          oNeurons.clear();
56
57          // Set the bias in the input and hidden vector to 1 (real number)
58          iNeurons.push_back(1.0f);
59          hNeurons.push_back(1.0f);
60
61          Predict_t retVal;
62          uint32_t wCount = 0;
63
64          // Init the network
65          for (uint32_t i = 0; i < inputNeurons; i++) { iNeurons.push_back(static_cast<float>(abs(input[i]))); }
66          for (uint32_t i = 0; i < hiddenNeurons; i++)    { hNeurons.push_back(0.0f); }
67          for (uint32_t i = 0; i < outputNeurons; i++)    { oNeurons.push_back(0.0f); }
68
69          for (uint32_t i = 1; i < hNeurons.size(); i++)
70          {
71              wCount = i - 1;
72              for (uint32_t j = 0; j < iNeurons.size(); j++)
73              {
74                  hNeurons[i] += iNeurons[j] * iWeights[wCount];
75                  wCount += hNeurons.size() - 1;
76              }
77              hNeurons[i] = 1 / (1 + pow(2.71828f, (-hNeurons[i] * beta)));
78          }
79
80          for (uint32_t i = 0; i < oNeurons.size(); i++)
81          {
82              wCount = i;
83              for (uint32_t j = 0; j < hNeurons.size(); j++)
```

```
84              {
85                  oNeurons[i] += hNeurons[j] * hWeights[wCount];
86                  wCount += oNeurons.size();
87              }
88              oNeurons[i] = (2 / (1.0f + pow(2.71828f, (-oNeurons[i] * beta)))) - 1; // Shift plus scale so the learning function can
                  be calculated
89          }
90
91          retVal.OutputNeurons = oNeurons;
92          return retVal;
93      }
94
95      void NN::Learn(InputLearnVector_t input, OutputLearnVector_t cat, uint32_t noOfDescriptorsUsed)
96      {
97          SoilMath::GA optim(PredictLearn, inputNeurons, hiddenNeurons, outputNeurons);
98          ComplexVect_t inputTest;
99          std::vector<Weight_t> weights;
100         Weight_t weight(((inputNeurons + 1) * hiddenNeurons) + ((hiddenNeurons + 1) * outputNeurons), 0);
101         // loop through each case and adjust the weights
102         optim.Evolve(input, weight, MinMaxWeight_t(-50, 50), cat, 1000, 50);
103
104         this->iWeights = Weight_t(weight.begin(), weight.begin() + ((inputNeurons + 1) * hiddenNeurons));
105         this->hWeights = Weight_t(weight.begin() + ((inputNeurons + 1) * hiddenNeurons), weight.end());
106         studied = true;
107     }
108
109 }
```