**CreateStructure:**

1. Create Data => IMLData data = new BasicMLData (input );
2. Create Network => var network=new BasicNetwork();
3. Add Layers? => network.AddLayer(new BasicLayer(null,true,i));

=>network.AddLayer(new BasicLayer(new ActivationSigmoid,false,1))

Większość sieci posiada 2-3 warstwy ukryte.

Funkcja aktywacji nie działa w warstwie input(nie przynosi żadnego efektu).

Bool oznacza występowanie Bias, który łączy się z następną warstwą(dlatego w output nie używamy biasu).

1. Ukończenie budowy sieci => network.Structure.FinalizeStructure();
2. Losowanie wag => network.reset();

**Funkcje aktywacji (służy do obliczania wyjścia z danej warstwy):**

1. ActivationBiPolar- wartości true(1)/false(-1), jeżeli wartość jest większa od 0 funkcja zmieni ją na 1 lub odwrotnie. ActivationBiPolar nie może zostać użyta przy treningu z użyciem propagacji.
2. ***ActivationCompetitive***- funkcja która umożliwia wybranie grupy neuronów z największymi wagami. Neurony z największymi wagami są przechowywane w tablicy przesłanej do tej funkcji.
3. LinearActivation
4. ActivationLog- używany tylko w sytuacji gdy input ma wartości pomiędzy -1 i 1.
5. ActivationSigmoid-używany dla wartości dodatnich, wartości ujemne będa zmienione na dodatnie, najprostsza funkcja aktywacji. Nie przewiduje wartości ujemnych na wyjściu.
6. ActivationSoftMax- wyrównuje wszystkie inputy do 1. Używany do sieci klasyfikujących.
7. ***ActivationTANH***-Najczęstsza funkcja aktywacji. Operuje na wartościach dodatnich i ujemnych. Domyślna funkcja aktywacji dla Encog, prosta funkcja rekurencyjna.

forward pass- zaczyna od inputu i porównuje wynik z output.

backward pass- zaczyna od outputu i porównuje wynik z input.

**Propagacja nadzorowana->output jest podany,**

1. Backpropagation-problem polega na tym, że algorytm szuka minimum lokalnego na nie globalnego. Wymaga 2 parametrów. Pierwszy Learning Rate, który odpowiada za wagi. Wartość pierwszego parametru jest dodawana do wag w macierzy.

Drugi parametr to Momentum, który odpowiada za zmiany w wagach.

var train = new Backpropagation ( network,trainingSet, 0.7, 0.3 ) ;

1. Manhattan Update Rule- polega na zmianie wartości w górę lub w dół w zależności od jej aktualnego położenia(jeśli wartość jest bliska 1 wtedy będzie ona zwiększana, jeśli bliska -1 wtedy zmniejszana jeśli bliska 0, wtedy nie będzie zmieniana). Wartość o jaką zwiększy lub zmniejszy sieć podawana jest jako parametr.

var train = new ManhattanPropagation(network , trainingSet ,0.00001) ;

1. Quick Propagation Training-używana do zminimalizowania błędu sieci. Nie wymaga podawania parametru Momentum. Początkowy punkt startowy ustawiony na 2.0 jest optymalny.

var train = new QuickPropagation ( network ,trainingSet , 2 . 0 ) ;

1. ***Resilient Propagation Training***- najbardziej efektywny algorytm treningowy w Encog. Nie wymaga podawania Momentum,Learning Rate lub Upgrade Constans(Stałej dodawania). Algorytm sam dla siebie ustawia odpowiednie parametry. Tworzy nowe parametry dla każdej iteracji.

var train = new Backpropagation ( network,trainingSet) ;

Encog posiada cztery odmiany Resilient Propagation Training:

1. RPROP+
2. RPROP-
3. iRPROP+
4. iPROP

Domyslnie jest używany RPROP+, aby go zmienic, należy:

train.setRPROPType(RPROPType.iRPROPp) ;

1. Scaled Conjugate Gradient Training-również nie wymaga parametrów. Nie jest dostępny dla wszystkich zestawów danych.

var train = new ScaledConjugateGradient (network ,trainingSet ) ;

1. ***Levenberg Marquardt Algorithm Training***-. poprzez zastosowanie współczynnika tłumienia do interpolacji między połączenie Resilient Propagation oraz Backpropagation, tworzona jest metoda hybrydowa.

var train = new LevenbergMarquardtTraining ( network , trainingSet ) ;