Module IV : Unsupervised learning: Hebbian and competitive learning

Pawel Chilinski

November 17, 2014

Examples in this document can be found in the source bundle attached to this document (or on github repository: `https://github.com/pawelc/neuralnet/tree/master/src/NeuralNet/exercises/module4`) and can be run after installing torch7 machine learning library: `http://torch.ch/`.

Exercise 1

Implement simple Hebbian rule to perform clustering of the standard Iris Flower input data. Try with 3 clusters.

I implemented the neural Hebbian learning with/without normalising weights after each sample,with/without normalising input data and run exercise in script neuralnet/src/NeuralNet/exercises/module4/ex1.lua. The script accepts various parameters to adjust training of the model. The output from the script is the output of the neuron on each example after training it with unsupervised Hebbian rule, together with the class given in the iris dataset. I run it with different parameters to check how Hebbian learning is affected by different modifications.

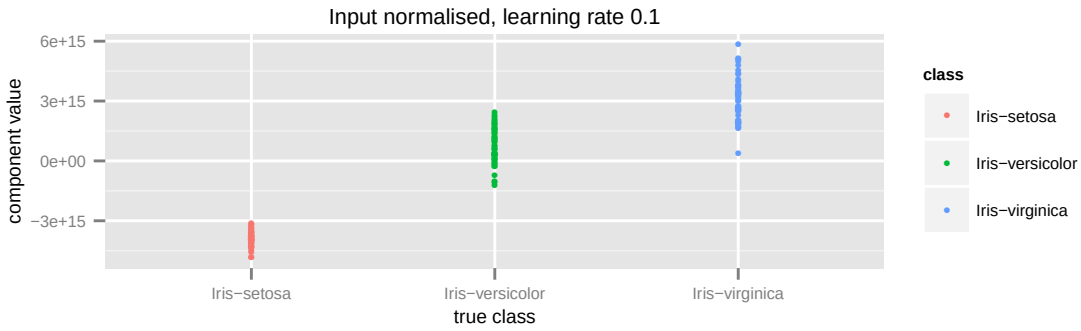
```
> ex1Output <- runLua("module4/ex1.lua")
> ex1Output.normalizedInput <- runLua("module4/ex1.lua -i")
> ex1Output.normalizedInputAndWeights <- runLua("module4/ex1.lua -i -w")
> ex1Output.normalizedInputAndWeightsMoreTrainingEpochs <- runLua("module4/ex1.lua -i -w -e 1000")
> ex1Output.normalizedInputAndWeightsMoreTrainingEpochsBiggerLearningRate <-
+   runLua("module4/ex1.lua -i -w -e 100 -l 0.9")
```

It looks that different settings haven't affected how Hebbian learning separates data into clusters.

We can show how projecting data on the first principal eigenvector reveals clusters in the data. The following figures show the points produced by the model against the true class, cluster around different values in 1D space.

We can see also that increasing the learning rate from 0.1 to 0.9 caused different clustering for each class but still they are separated in the same manner.

The simple Hebbian update rule without input and weight normalisation produces worse result than when we add normalisation to weights and input data. For instance the Iris-setosa and Iris-versicolor overlap after projection in the first plot and in the rest of cases they don't.



Exercise 2

Exercise 2. Implement Sejnowski's covariance rule to perform clustering of the standard Iris Flower input data. Compare results with the Exercise 1.

I implemented the Sejnowski's covariance rule in script neuralnet/src/NeuralNet/exercises/module4/ex2.lua.

```
> ex2Output <- runLua("module4/ex2.lua")
```

The output of the trained single neuron network is presented on the following figure. Outputs from the network are plotted against their true classes. We see that maximum principal component was selected in a similar way as in the case of the Hebbian algorithm output. This plot is closest to the Hebbian algorithm running on the normalised data which makes intuitive sense.

The Sejnowski's covariance rule produces better results that simple Hebbian update rule without weight and input normalisation. For instance the classes of Iris-setosa and Iris-versicolor do not overlap after projection by algorithm and in case of simple Hebbian without normalisation they overlap.

Exercise 3

Derive Oja's learning rule (4.17) using (4.14) from the textbook.
It looks it is already derived in the textbook in (4.15).

Exercise 4

Implement Oja's rule to perform 3-class clustering of the standard Iris Flower input data. Attach comparative analysis of the results.

I implemented Oja's rule in script neuralnet/src/NeuralNet/exercises/module4/ex4.lua:

```
> ex4Output <- runLua("module4/ex4.lua")
```

It can be seen that separation of the clusters is not as good as Sejnowski's covariance rule but the values of first component are much smaller (because of implicit weigh normalisation).

Exercise 5

Implement competitive network with three nodes to perform 3-class clustering of the standard Iris Flower input data. Attach comparative analysis of the results.

I implemented competitive network with three nodes in script neuralnet/src/NeuralNet/exercises/module4/ex5.lua:

```
> ex5Output <- runLua("module4/ex5.lua -i -e 10000 -l 0.01")
```

After training I assign the input to the cluster by selecting the neuron with the biggest value. Each of 3 neurons represent one class. To show the assignment on the plot I draw on horizontal axis the true class and on vertical axis assigned neuron and the points are jittered so they do not overlap.

We can see the Iris-setosa was separated perfectly, the Iris-virginica is not perfectly assigned to one cluster and the Iris-versicolor is almost evenly split between 2 classes. This agree with previous experiments where also red points where separated in most cases to 2 visible separate clusters and green and blue points have overlapping region.