Single and Multi-layer perceptrons for classifications and predictions of time series

Henrik Åhl

January 2, 2016

Department of Astronomy and Theoretical Physics, Lund University

Project supervised by Mattias Ohlsson

LUND UNIVERSITY

And this all works extremely well. Or, at least it works.
— Mattias Ohlsson, 2015

1 Introduction

Neural networks can be used for a wide range of purposes, ranging from function approximation to identifying patterns in images. Although the networks themselves can be constructed to be however complicated the creator wishes, relatively simple networks can be shown to be able to solve complicated tasks. Especially single-layer perceptrons, as are the main study in this report, are able to perform surprisingly complex tasks.

2 Problems

2.1 Competitive networks for clustering

In using the VQ model for classification, it is apparent that the method indeed is able to correctly find the different class centers in many cases. In the few occurences where it is not the case, the ambiguity stems from outputs getting stuck between two classes, with equal tendencies to shift towards either one. It is however possible that these happenings often will eventually vanish given more computed epochs, due to the inherent flucutations of the system.

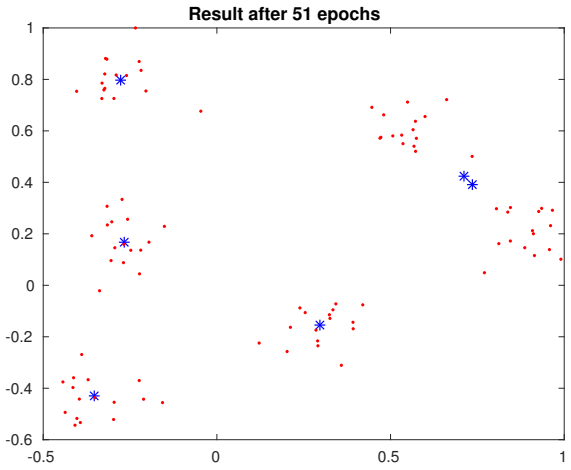

Figure 1: Clustering result with the VQ algorithm. Note the ambiguous nodes in the top right corner.

When the default settings are changed, so that the bias learning rate is set to zero, so called dead neurons appear – neurons that get stuck far away from clusters due to the algorithm favoring the update of neurons close to data clusters. It is also apparent that the value of the bias learning rate is

significantly lower than the ordinary one, as the neurons otherwise become extremely likely to not move at all, resulting in a cluster of its own of dead neurons.

Furthermore, when superfluous neurons are introduced, they severely affect the performance of the VQ method. As there in that case tends to be several "winning neurons", the neurons counteract each other, resulting in them instead merely encircling or approaching the very cluster centers they are meant to find. It is also the case that more neurons give rise to heavier interference, and thus also worse overall performance.

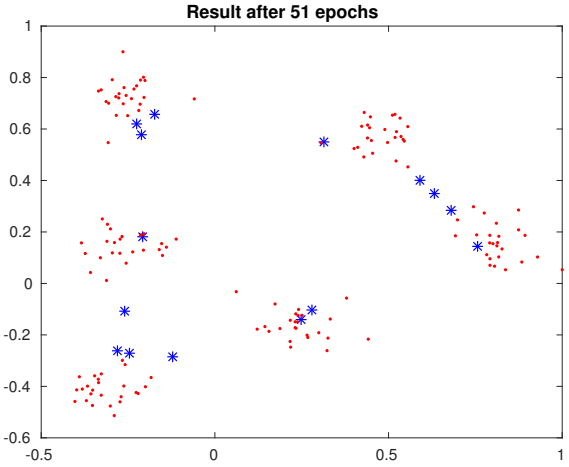

Figure 2: Clustering result with the VQ algorithm and superfluous neurons. Especially consider the upper right corner, which like in the previous case attracts ambiguous neurons.

2.2 ORL face data

Interestingly enough, the competitive network manages to group different faces into categories fairly adequately; in almost all 80 cases, the network manages to find the corresponding two pictures of the same person. Moreso, with 6 output nodes used, the network groups a significant amount of the occuring faces with similar attributes into the same categories. The most defining trait with determines grouping seems to be the skin tone, or the brightness of the photograph, as it is highly related to the color of a majority of the pixels in every picture. However, also people with other similarities are frequently grouped together, such as people with glasses or people with a similar haircut, which ought to be because similarly colored pixels appear in similar places, which causes the network to identify the shared attribute.

Also when performed several times, the network tends to give similar results, grouping people into roughly the same categories as before.

There does however appear to be a significant bias towards beloning to the first groups (i.e. laying close to node 1 etc.), which suggests that some data points again get stuck due to conflicting algorithmic interests.

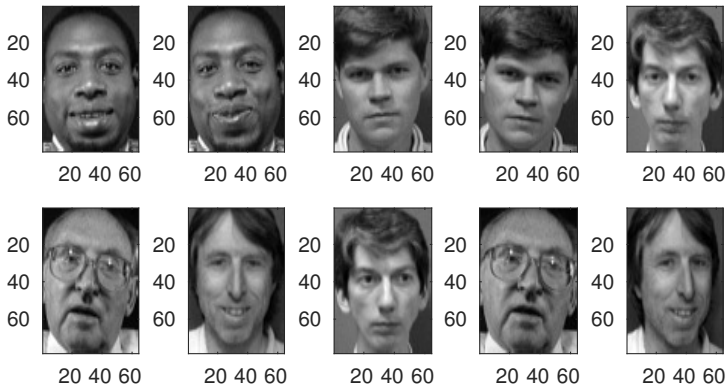

Figure 3: Cluster category produced by the VQ algorithm. Note the similarities in the overall tone of the images.

When further performing the analysis on as many as 40 nodes, i.e. as many nodes as there are different people, and iterating over many epochs, the network is fairly able to group together the corresponding two pictures of two people, being able to do so correctly in as many as 10/40 cases, where several of the remaining fraction still contains correct matchups, though with 3 or more individuals beloninging to that group. Again, however, there is a strong bias towards the lowest labeled nodes in the network.

As the cluster centers can be identified as the "average" person within a cluster, it can be amusing to witness the result, here exemplified in fig. 4. As to be expected, the average individual corresponds to an overlap between the images belonging to that specific cluster.

Figure 4: Example image corresponding to the weight vector defining a cluster center.

2.3 SOFM

With the SOFM method, clusters in the data set tend to attract a concentration of output nodes in one or two dimensions. The higher the variance between data points, the further away their corresponding nodes will appear in weight space. These two attributes of the SOFM algorithm can easily be seen in fig. 5.

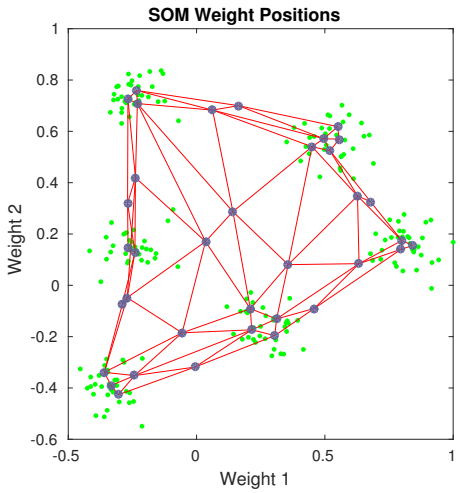

Figure 5: Weight positions for the SOFM algorithm. Clusters of data tend to attract clusters of nodes.

From a quick analysis of the SOFM performance in grouping faces together, under an image size reduction to 50 %, it seems to be, from a subjective viewpoint, worse than the simple clustering method. Although it indeed manages to group corresponding face pairs in the same category to roughly the same performance as in the clustering case, there cannot be said to be any clear similarities between pairs. In the clustering case, for example, people with glasses were likely to end up in the same category, which cannot be said to be the effect of the SOFM.

Notably enough however, when the image reduction is skipped completely, the SOFM net shows significant improvements, even passing the performance of the VQ net. The grouping in this case differs from the VQ net in that it seems to find similar features that the former could not, such as expressions (smiles etc.) and face shapes – at least to a higher degree. Also the previously occuring group of men with beards and glasses again appear, as shown in fig. 6, showing that image reduction does affect the performance of the network.

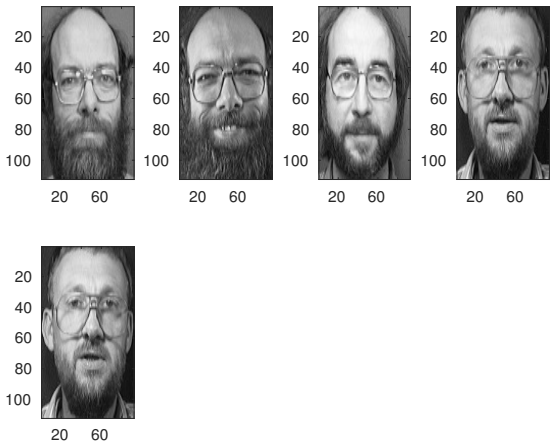

Figure 6: Cluster produced of men with similar facial attributes. Note here that the skin tone is not as defining of a factor as in previous cases.

The distance between output nodes also indeed seems to be related to the features of the images. As in the previous case, the most defining attribute seems to be the overall brightness of the image. In particular in the comparision between fig. 7 and fig. 8 this behaviour of the network can be seen, as the two clusters correspond to points far from each other in weight space.

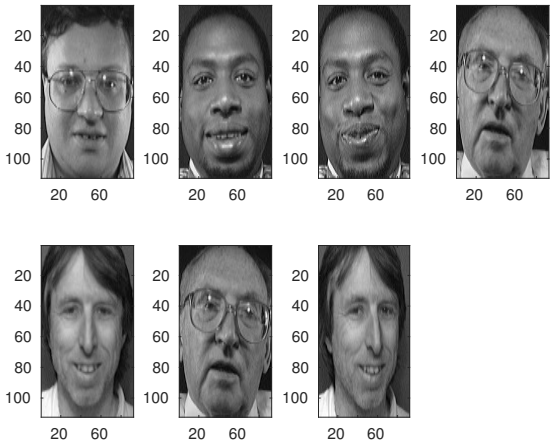

Figure 7: Dark-face category.

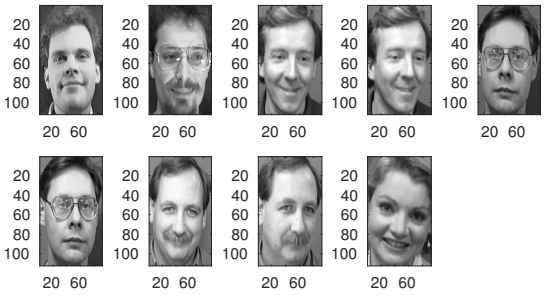

Figure 8: Light-face category.

2.4 LVQ

It seems clear from the analysis that at least six nodes are required to somewhat sufficiently classify the data points. This ought to be because of the need to define two areas – one related to class 1, the other to class 2. The corresponding decision boundary is then made up by weighting of the nodes determining "cluster centers". This is shown in fig. 9, where the nodes are seen distributed throughout the data points in order to delimit the two classes, in effect defining a decision boundary with a closed geometry.

With a slightly lower learning rate than the default one, the best network trained managed to classify 92 % of the data points correctly, roughly on par with the 15-node, two-layer network result in homework assignment 1.

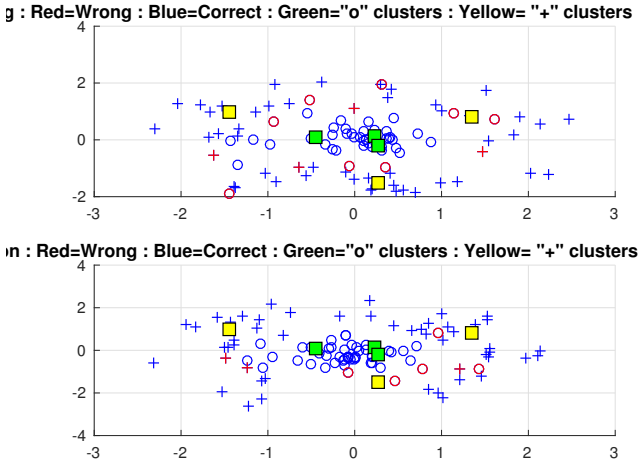

Figure 9: Visualization of training and validation performance. The cluster centers belonging to the two classes respectivelly are spread out in order to properly weight the classification boundary in place.

2.5 Time-Delay Networks

Using an Elman net with one delay feedback into the loop, it can be seen by using an alternating number of neurons, that the ability of the net to predict time series quickly deteriorates the more nodes there are. For four and five hidden nodes, the network simply crashes completely, not being able to predict anything to a satisfying degree. Instead, for two and three hidden nodes, the network performs comparatively equally ($NMSE \sim 0.13$), producing a valid prediction of the time series. For a network with one hidden node, the output is on average slightly worse ($NMSE \sim 0.24$), mostly due to a slight shift in time, suggesting that the network indeed does a viable prediction, but is not able to align it phase-wise with actual data. This one-node Elman net performs roughly as well as the earlier sunspot predictors used in homework assignment one, which had a $NMSE$ on about the same level.

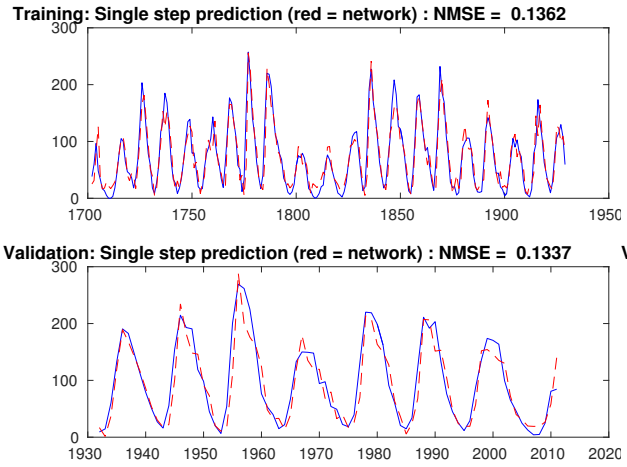

Figure 10: Elman network performance with two hidden nodes.

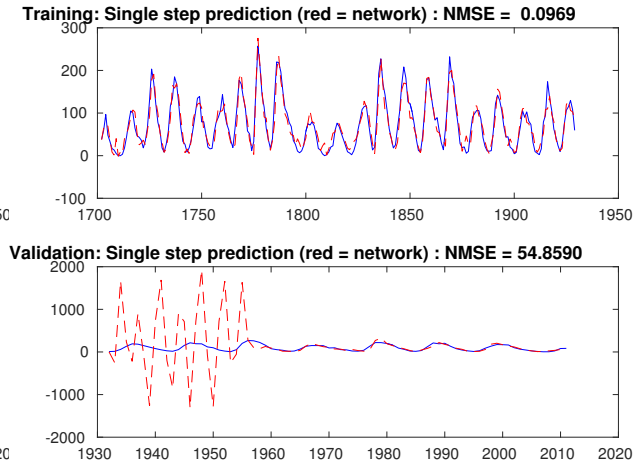

Figure 11: Elman network performance with five hidden nodes. Extremely bad prediction of the time series although the training performance is better than the two node net.

Interestingly enough, from the analysis, the optimal time delay appears to be the one where the previous value is evaluated, i.e. a time delay of one. Contrary to what was achieved in assignment one then, the most determining previous value for the Elman net is the one occuring at $t' = t - 1$, whereas we in the earlier assignment saw that a set with circa four of the preceding twelve outputs were the best at predicting the sunspot time series. (At least for me.)

2.6 Hopfield Networks

It can be shown that 0.138 patterns can be stored for each node, as done by Hertz et al 1991 [3]. Using this rule of thumb and the fact that our images are represented by 7×5 nodes, we see that we can theoretically store at most 6 complete (random) patterns in our network. We must however realize that our patterns are not fully random, but correlated in appearance; consider for example the similarities between the letters D and E in low resolution. Consequently, all of our letters will not be stable, as we can most notably see in fig. 12, where the stored letter E converges towards the representation of a B, with letters A–E stored in the network.

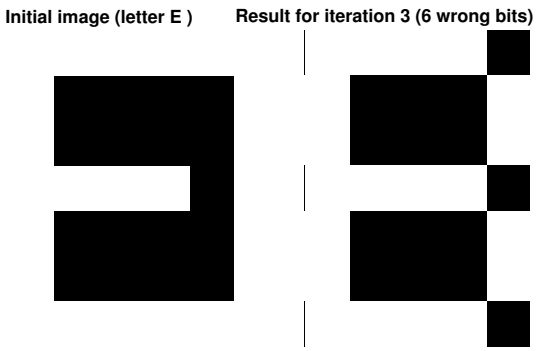

Figure 12: Convergence for the letter E, which classifies as a B after a few iterations.

Since the images are so compressed, even slight distortions will greatly affect the retrieval ability of the network. Although the similarities between the letters A and B are few, the letter A will occasionally converge towards a B even with as low of a distortion probability as 0.1. Also the fact that inverted patterns are minima in the effective energy landscape is illustrated in fig. 13, as the B is inverted relative to the stored letter.

In effect then, if the stored patters share similarities, the stability of patters certainly cannot be guaranteed, whereas if the images are also distorted, the retrieval rate is even further decreased.

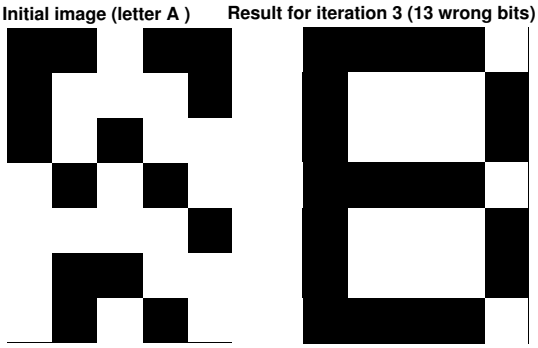

Figure 13: Convergence for the distorted letter A ($p = 0.1$), which classifies as an inverted(!) B after a few iterations.

When ten letters are stored in the network, the limitations become even more apparent as there even without distortions will be huge instabilities, resulting in many spurious states. In fact, the only letters not converging towards spurious states, even without distortion, are the letters I and J, where J nontheless converges towards the stable I. All in all then, $9/10$ stored patterns will in effect be unstable. Notably, the letters B, E and F all converge towards the same spurious state, as do the letters C, D and G.

In the Matlab model however, all stored letters are stable. However, as with the original model, distorted patters are likely to end up in spurious states, even with low probabilities ($p = 0.1$) of flipping pixels. Out of all the ten letters, five are on average stable.

3 Concluding remarks

References

[1] Mattias Ohlsson, *Lecture notes, Artificial Neural Networks*, Department of Astronomy and Theoretical Physics, Lund University, 2015.

[2] Mattias Ohlsson, *Exercise 2: Self-Organization and Feedback Networks*, Department of Astronomy and Theoretical Physics, Lund University, 2015.

[3] Hertz, J., Krogh, A., and Palmer, R.G.; *Introduction to the theory of neural computation*, Redwood City, CA: Addison-Wesley, 1991.