Advanced Machine Learning

Homework 4

Sai Vineet Reddy Thatiparthi

15 March 2019

The program is attached along with the submission. All are python Jupyter Notebooks.

1 **Train a VAE using partition (i) of the data (note that, you will not be using class labels in this training)**

Proof. I have attached the code of the fully running VAE. It is trained on the first partition (70 percent).. □

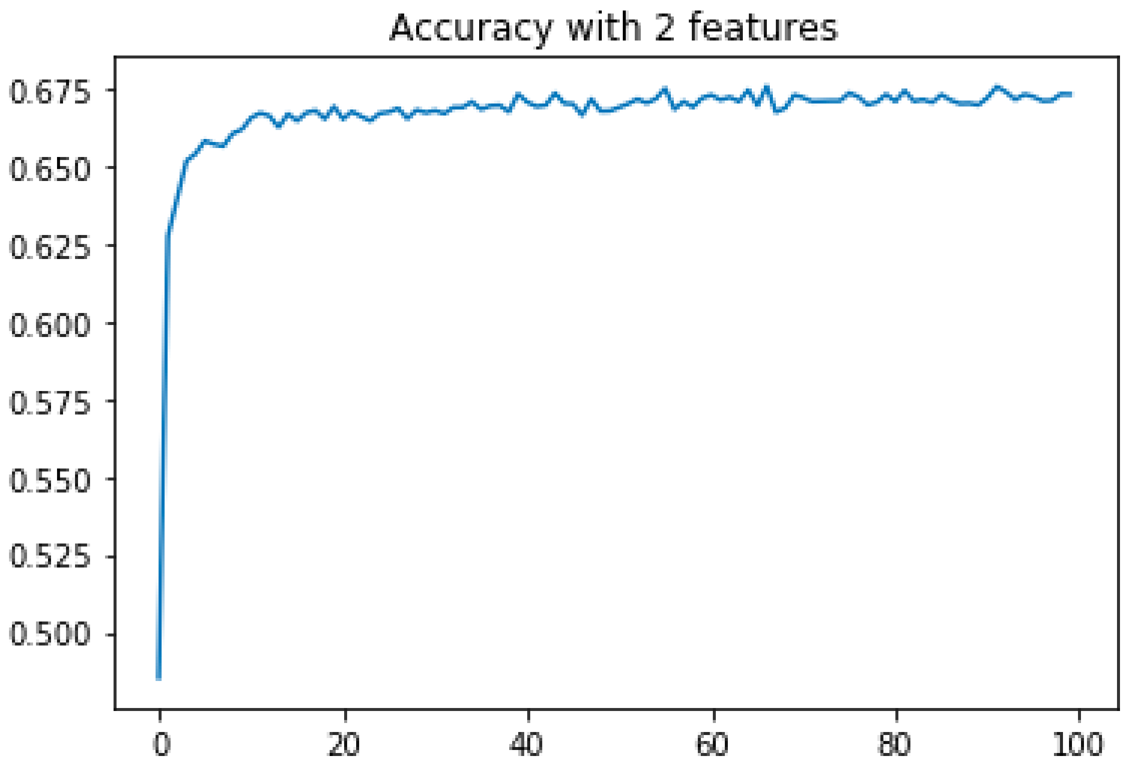
2. **Now, using the network obtained above to create a dataset from partition (ii). Specically, take the patterns from partition (ii) and use the encoder output to generate inputs. The corresponding class labels are the desired output. Train a simple feed-forward network (1 hidden layer) with this data inputs are the outputs of the encoder and class labels are desired output. Use this network to assess the performance on partition (iii) of the dataset. You may want to play around with the (hyper-)parameters to get good performance. Document the performance you get with the different network configurations you tried.**

Proof. I stored the encoded outputs of the 2nd partition, and trained a FFNN on it. At first, the accuracy was really bad, getting only about 50 percent. On closer inspection, changing the number of features the Varitional Auto-Encoder has to encode the MNIST database in the latent space played a huge role in my final accuracy of this FFNN. It is probably because, if more features are present in the latent space, less loss is incurred during the decoding phase. I tried different configurations - each of which I have explained and tabulated their results in a table below.

Attempt 1: Features - 2 The classification accuracy was about 70 (figure 1) percent after 100 epochs. The accuracy moved a little higher when the number of neurons in the hidden layer were increased. However, even intuitively, it understandable why the neural net is having difficulty classifying as the encoded images have not much information as we are trying to cram everything into just 2 features.

Attempt 2: Features - 3 When the features were changed to 3 in the latent space, the performance was better than the previous model. The neural network was getting an accuracy of 73 percent (figure 2). This is probably because the neural network had an extra dimension in the latent space that it could work with, so more information could be preserved. To see if I could do better than this, we increased the number of features even more.

Figure 1: 2 features

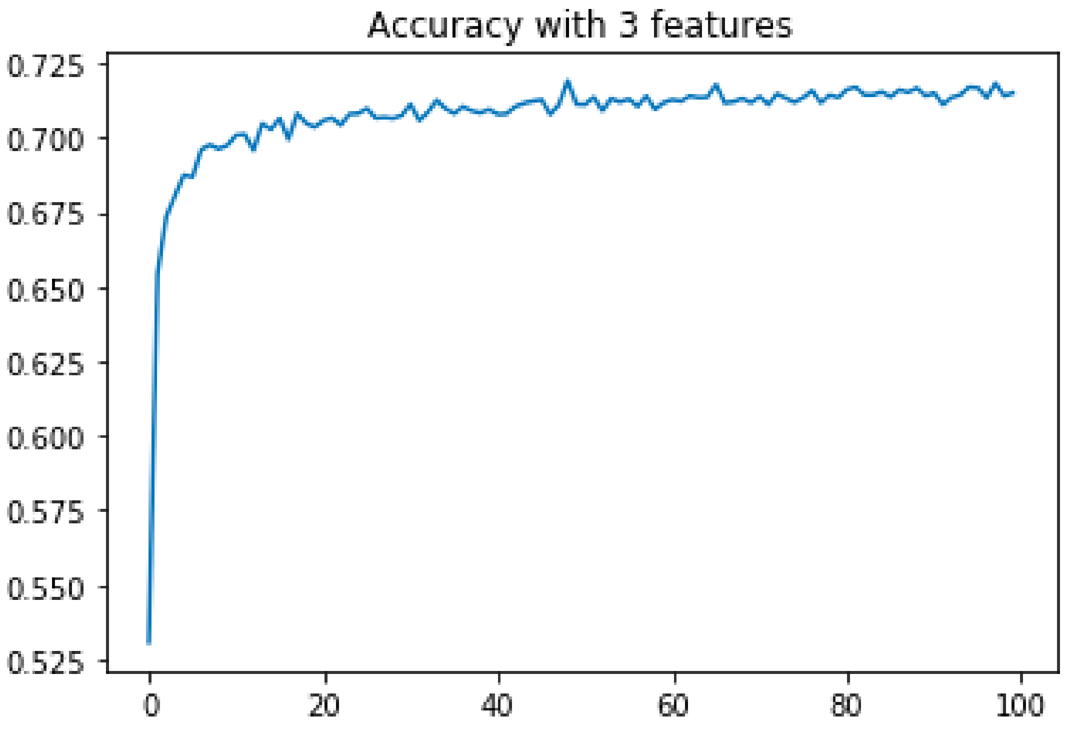


Attempt 3: Features - 5 When the features were changed to 5 in the latent space, the performance was the best out of all the different models and configurations I tried. The neural network was achieving an accuracy of almost 90 percent (figure 3), which is a considerable improvement over the last two configurations. What one could infer from these results it that the accuracy exponentially increases with increase in the number of features that the network can work with in the latent space. It also makes intuitive sense as we lose less information if we encode an image into a large number of features as we can reproduce the images, better.

Latent Features	**Hidden Neurons**	**Accuracy**
2	64	*67.5*
2	256	*70*
3	64	*71.5*
3	256	*72.69*
5	64	*84.89*
5	256	*88.67*
2	**CNN**	80.72

Attempt 4: Features - 2, CNN When the features were set to 2, I created a CNN that took the encoded images as input and output the class they belonged to. This method yielded a lot more success as in 15 epochs, an accuracy of 79 was achieved, which is a lot more than the equivalent

Figure 2: 3 features



regression version above (number of features are the same - attempt 1).

Conclusion - With the increase in the latent features, the image quality increases. This allows the encoder to perform a better job at encoding the salient features of a given image into the latent space. This also gives the FFNN classifier a better image to work with as well.

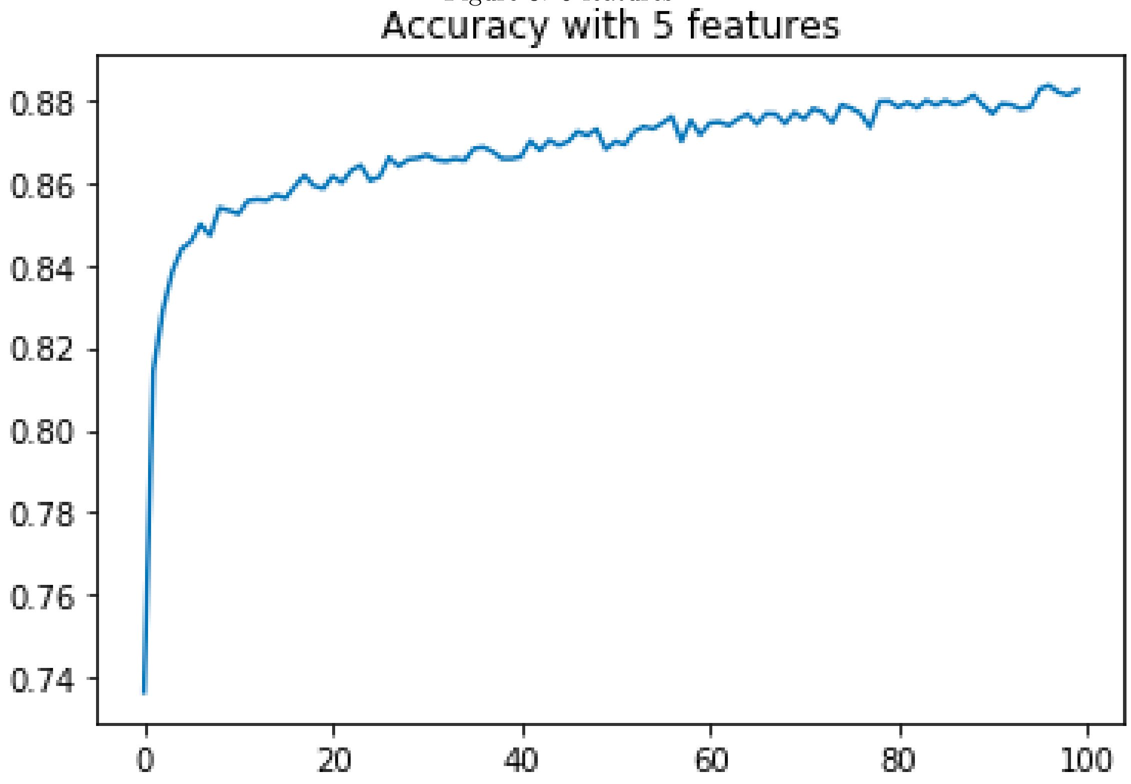
Another interesting thing I noted was that a CNN outperformed every other network despite using the fewest features. Maybe one should use CNN to perform the classification and not a simple FFNN. □

3. **Separately, use partition (i) to train a feed-forward network in a supervised manner. The network produces the class label directly. Assess the performance on partition (iii). Again, you may want to play around with the hyper-parameters to get good performance. Document the performance you get with the different network configurations you tried.**

Proof. This was a simple task that could be achieved using two different ways - a normal neural network (regression) or a convolutional neural network.

Regression - The model was a very simple neural network with 1 hidden layer with 64 neurons in them. In 5 epochs, an accuracy of 98 percent was achieved! Increasing the hidden neurons even a bit changed the testing accuracy to 99 percent. There was not much need to fiddle around

Figure 3: 5 features



with the hyperparameters as a good result was being obtained regardless on the first try itself. The classifier had the entire full resolution image to work with, which is why it performed so good.

CNN - I tried to implement a CNN as images were being considered. With a very simple a bare bones CNN, a similar accuracy of 97 percent was achieved in just 5 epochs. Increasing the number of convolutions increased the accuracy a bit every time this was done.

□

4. **Compare the performance you get in part (b) with what you get in part (c) and provide crisp comments on the utility of training as outlined in (b).**

Proof. From the look of it, one can see that supervised learning in question 3 performed a lot better than the unsupervised model in question 2. The supervised model achieved higher accuracy than the unsupervised model in fewer epochs. The supervised model tried to approximate a function to the data while the unsupervised model tried to find a mapping of the input data (the only thing that it had to work with).

However, one has to see that despite having a lower accuracy, the unsupervised model had no class labels to work with and dealt with encoded images that did not hold much information. The unsupervised model also dealt with mostly lossy images as we were trying to encode them into very few features. Keeping this in mind, the final accuracy of the VAE run FFNN was so far away from the final model is probably this. However, the number of parameters in the unsupervised model were far fewer in than the number of features in the supervised model. In short, unsupervised model's computational complexity was lower than that of the supervised model.

Unsupervised learning might seem not so useful in this scenario, but when one does not have access to labelled data, this approach is far better than any other method as it tries to make sense of data despite not having any labels. When one has access to labels, supervised learning is probably the best approach, but in the absence of these labels, unsupervised is the way to go. □