Deep Learning
CS7015
Programming Assignment 5
Restricted Boltzmann Machines

Shubhangi Ghosh - EE15B129
Monisha J - CS15B053

April 2018

1 Objective

The objective of this assignment was to

- To construct a Restricted Boltzmann Machine(RBM) from scratch using *numpy*
- Train it using the contrastive divergence algorithm upto k steps
- Use it to obtain hidden representations for images in the Fashion-MNIST dataset.

2 Initialization of Weights and Biases

The weights and biases are initialized by the technique suggested in [1].
The weight matrix is initialized from a zero-mean Gaussian with a standard deviation of about 0.01.
The biases for the visible units are initialized with zeros.
The biases for the hidden units are initialized with -4. This encourages sparsity.

3 Training

While training, the hidden states were assumed to be binary, as it is a better choice and offers some inherent regularization[2].

For the visible states, we tried using both binary (as taught in the class) as well continuous states while sampling, and found that continuous states performs better[2].
The training was tracked by measuring the reconstruction error, that is, the squared error between the true input and the sampled input.
A plot of training loss versus epochs is shown in Figure 1.

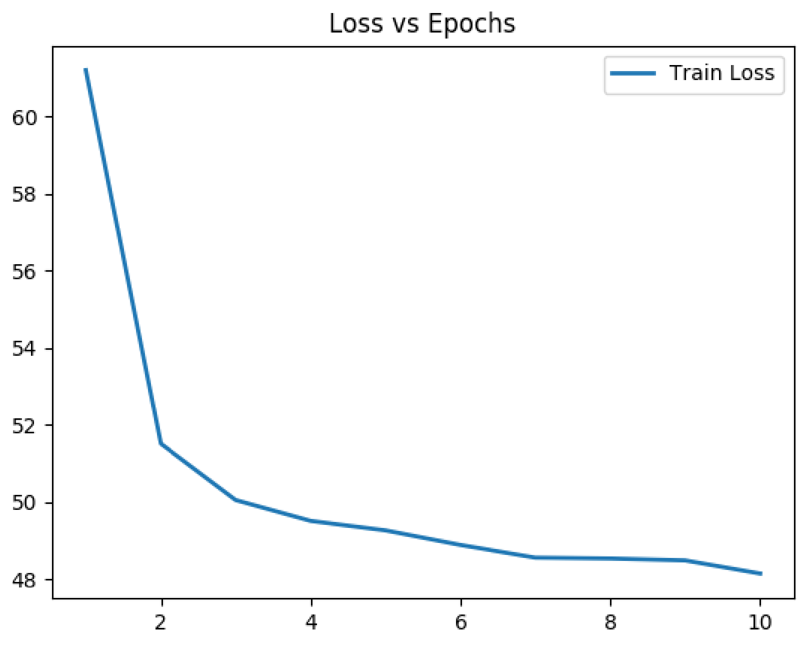


Figure 1: Loss against Epochs

4 Samples Generated

With total number of training iterations as m, where every update step is considered an iteration, and the sampled produced by Contrastive Divergence every $m/64$ iterations were plotted as images. These 64 images are shown in Figure 2.

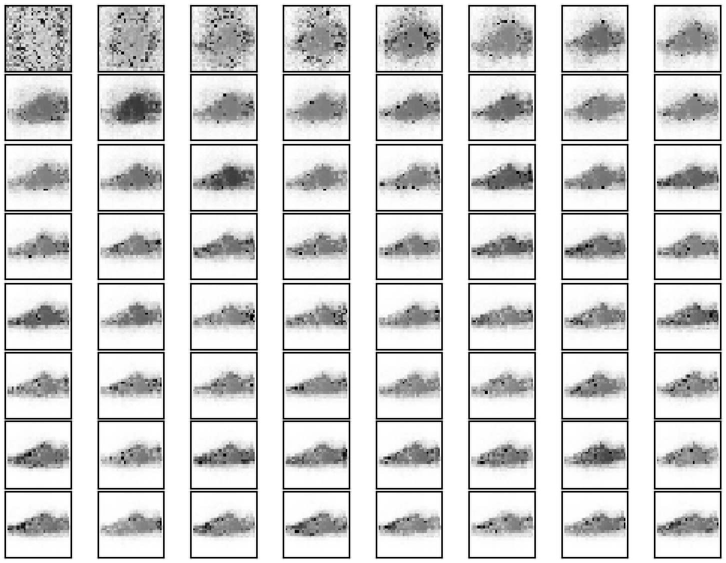


Figure 2: Sampled Generated by Contrastive Divergence

5 Hyper-parameter Configuration

We fixed our hidden state size to 100.
The hyper-parameters listed below were varied to find the best possible fit to our data. The configuration that yielded the best results is listed below :
Epochs : 10
Learning rate : 0.01
Batch Size : 1
k of CD : 1
Momentum : 0.0

6 t-SNE

The hidden representations were computed for around 200 training points. t-SNE[2] was performed using *sklearn* library to convert the n dimensional

representation to two dimensions.
Figure 3 shows some of the training points, while Figure 4 shows some of the test points. Here, data points of the same color belong to the same class. It can be seen that points belonging to the same class lie somewhat close to together in the 2D space. Hence, we can conclude that our model is learning good representations for the images.

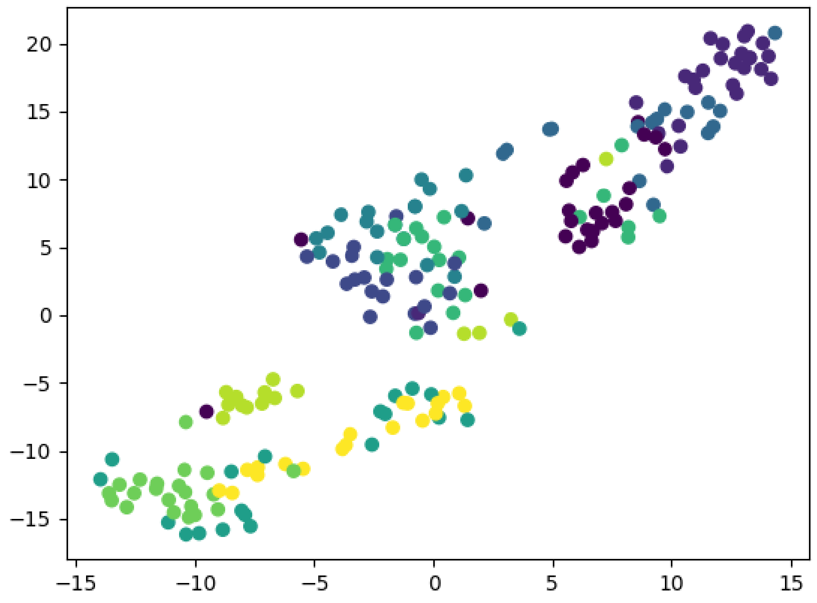


Figure 3: t-SNE plot of training data

7 Sampling

7.1 Contrastive Divergence

In contrastive divergence, we sample h from the input v and then use the sampled h to sample v. This is repeated k times and the final sampled v is used for updating of weights and biases.
k was varied between 1 and 5.
The best suitable value of k was found to be 1.

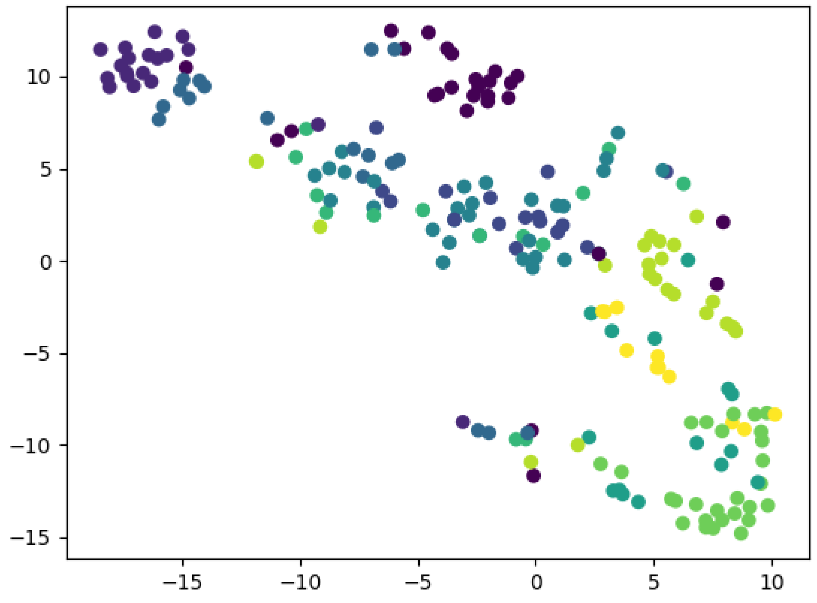


Figure 4: t-SNE plot of test data

7.2 Gibbs Sampling

k	Loss
500	234
1000	199
2000	184

Thus, we see that as we run the longer gibbs chains the loss reduces and the model improves. This is expected as in Gibbs sampling we start from a random image and as we move along the Gibbs chain we get better and better hidden representations(as our model is trained) and thus we get better visible images further along the Gibbs chains as we sample from better hidden representations.

The chain had be run for around 500 iterations before seeing sampled that resembled images from our data. As the training progressed, however, the the number of gibbs steps reduced, and samples could be seen in around 300 iterations itself, and later on in around 100 iterations.

Figures 5a, 5b and 5c show the Gibbs chain at epochs 1, 2 and 3 respectively.

8 Additional Experiments

8.1 Batch Gradient Descent

In addition to stochastic gradient descent, mini-batch based gradient descent was also implemented. Making use of batched updates help in utilising the efficiency of *numpy* matrix operations.
The batch size was varied from 10 to 100. However, the stochastic gradient descent was found to perform the best.

8.2 Momentum based Gradient Descent

Momentum based gradient descent was implemented.
Two variations were tried :

- Setting the momentum to a constant value,
  The best such value was found to be 0.9.

- Varying the momentum as suggested in [1].
  The momentum is set to 0.9 initially, and then reduced to 0.5 in the later epochs.

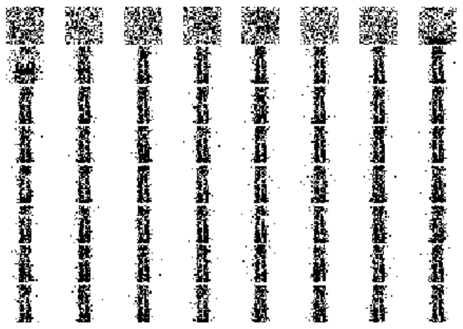
However, the model worked better without using momentum.

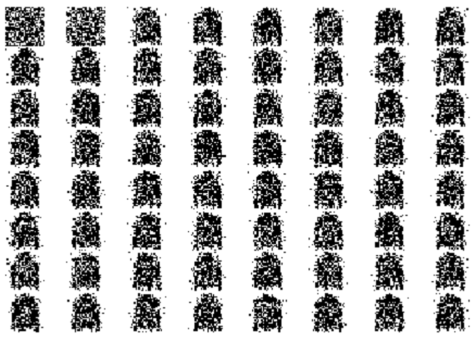
8.3 Weight Decay

Weight Decay[2] refers to L2 regularization applied to the weight update. It helps prevent overfitting.
Weight decay was found to give better performance for our dataset.
The weight decay parameter was tuned, and 0.0001 was found to be the best value. The t-SNE plot of training data when weight decay was used is seen in Figure 6.

8.4 Persistent Contrastive Divergence

Persistent Contrastive Divergence was implemented.
Persistent CD[4] relies on a single Markov chain, which has a persistent


(a) Gibbs chain samples at epoch 1


(b) Gibbs chain samples at epoch 2

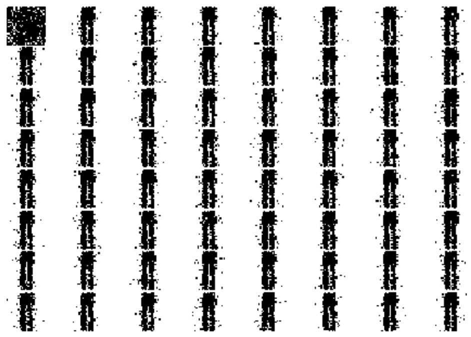

(c) Gibbs chain samples at epoch 3

Figure 5: Pictures of animals

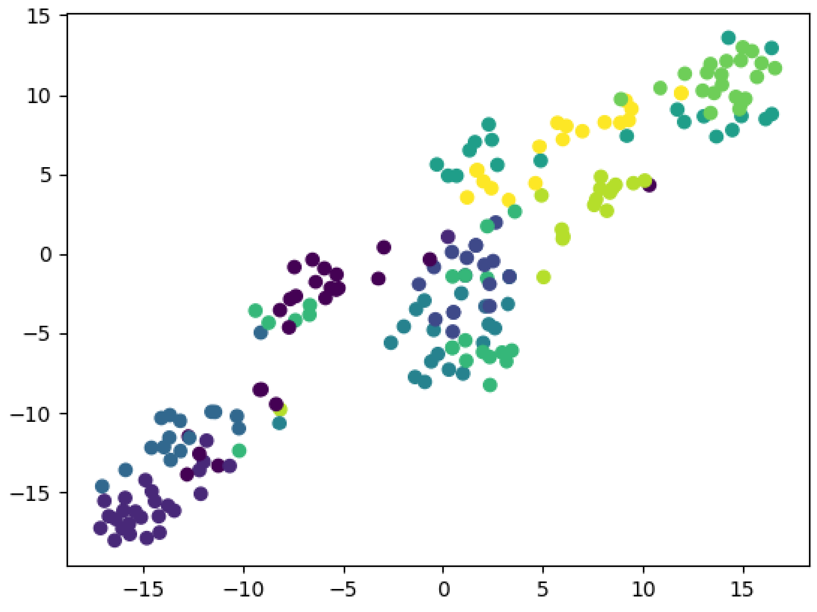


Figure 6: t-SNE plot of training data with weight decay

state, that is, for each parameter update, we extract new samples by simply running the chain for k-steps on the previously extracted sample, rather than starting from a new input sample.
For our dataset, normal CD was found to outperform persistent CD.

References

1. Fischer, Asja, and Christian Igel. ”An introduction to restricted Boltzmann machines.” In Iberoamerican Congress on Pattern Recognition, pp. 14-36. Springer, Berlin, Heidelberg, 2012.

2. Hinton, Geoffrey E. ”A practical guide to training restricted Boltzmann machines.” In Neural networks: Tricks of the trade, pp. 599-619. Springer, Berlin, Heidelberg, 2012.

3. Maaten, Laurens van der, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of machine learning research 9, no. Nov (2008):

2579-2605.

4. Tieleman, Tijmen. "Training restricted Boltzmann machines using approximations to the likelihood gradient." In Proceedings of the 25th international conference on Machine learning, pp. 1064-1071. ACM, 2008.