Dynamical Systems and Deep Learning

Coursework II

This is a practical investigation into training restricted Boltzmann machines (RBM) and performing a simple classification task. You should submit your work electronically as specified below in the problem.

Description of the dataset

The dataset to be used is the Caltech 101 Silhouette 28x28 dataset (available in matlab format at [1]), which contains various silhouette images that each belong to 1 of 101 classes. The same dataset is additionally available in csv format here.

Each row in `train_data.csv` and `test_data.csv` is a train and test instance respectively (i.e. visible unit activations are comma-separated columns and data instances are row-separated). The corresponding class label indices for each data instance are given in the rows of `train_labels.csv` and `test_labels.csv`. Also included is a validation data set (`val_data.csv`, `val_labels.csv`), which can be ignored (unless you wish to experiment with techniques to avoid overfitting, such as early stopping [2]). The class name for each class label index is given by the corresponding column in classnames.csv.

RBM implementations

The RBM implementation recommended for this exercise is Matlab Environment for Deep Architecture Learning (Medal), which is available at [3]. Medal includes an example script for an RBM classification task on a digits dataset in `demo/demoBinaryRBM_MNIST.m`, which can be used almost as-is for the below exercise with minimal additional programming required.

You are free to use other languages (e.g. Python, Java) and libraries (including a custom implementation) if you prefer, so long as your submission runs on a standard lab machine.

Description of the Exercise

Construct an RBM with visible units (consisting of both binary sigmoid units for the silhouette patterns, and binary softmax units for the pattern's corresponding class), and binary sigmoid hidden units.

Now experiment with various configurations in order to find one with the lowest class prediction error over the test data set. You may wish to limit your search to using a vanilla gradient descent (i.e. no momentum or weight decay terms) with 1-step contrastive divergence, over the following discrete hyper-parameter space:

number of hidden units $\in \{100, 1000\}$

learning rate $\in \{0.01, 0.1\}$

batch size $\in \{100, 500\}$

number of epochs $\in \{100, 250\}$

However, you are free to experiment with any other parameters and additional techniques (e.g. momentum, weight decay) that you wish (and this will gain additional marks).

Submission instructions

You should submit all of your code, plus instructions for running it. Your code should run on a standard lab machine, and should compute and output a class prediction error percentage. You should submit a pre-trained RBM, so that when we run your code this does not involve training a new network from scratch (in Matlab, you can do this easily by exporting/importing workspace variables to/from .mat files).

You should also submit a short document file, detailing the configurations and learning techniques that you have experimented with, and the best configuration you found.

Notes on using Medal

In Medal, construction of an RBM using a vanilla gradient descent configuration (i.e. without weight decay or momentum terms), with visible units (consisting of both binary sigmoid units for the pattern, and binary softmax units for the pattern's corresponding class), binary sigmoid hidden units,

and the following configuration:

Number of hidden units: 100
Learning rate: 0.1
Batch size: 100
Number of epochs: 100
Number of Gibbs steps during contrastive divergence: 1

looks like:

```
[nObs,nVis] = size(trainData);
nHid = 100;
arch = struct('size', [nVis,nHid], 'classifier',true, 'inputType','binary');
arch.opts = {'verbose', 1, ...
             'lRate', 0.1, ...
              'nEpoch', 100, ...
              'batchSz', 100, ...
              'nGibbs', 1};
r = rbm(arch);
```

You can use the rbm.classify method to compute the class prediction error value (see `demo/demoBinaryRBM_MNIST.m` for a similar example using a different dataset)

Mark Scheme

The marks will be awarded as follows (100 total).

Code runs and outputs a class prediction error:
20 marks

Evidence of experimentation over the hyper-parameter space given in the exercise, using a vanilla gradient descent (i.e. no momentum or weight decay terms) and 1-step contrastive divergence:
up to 50 marks

Finding a good region of hyper-parameter space/relative quality of final prediction error:
up to 10 marks

Evidence of experimentation outside of the given hyper-parameter space, and/or with additional learning techniques (such as momentum and weight decay):
up to 20 marks

References

[1] Practical guide to training restricted Boltzmann machines, Hinton, 2010, https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf
a good general source of information for practical advice relating to training RBMs

[2] Caltech 101 silhouette dataset
https://people.cs.umass.edu/~marlin/data.shtml

[3] Matlab Environment for Deep Architecture Learning (Medal)
https://github.com/dustinstansbury/medal