Tutorial 7: Artificial neural networks; support vector machines

Artificial neural networks

1. Answer the following:
   (a) What's an activation function and what is its purpose?
   (b) What is a hidden layer in an artificial neural network?
   (c) What is a softmax function, and when is it used?

Spiral

2. Follow this link to an interactive demo of a neural network trained on the spiral pattern. On that page you can change the network's architecture, number of hidden layers, neurons per layer, activation functions, learning rates, etc. You can start and stop the training process, observe how the parameters of the model change, and how this affects the output. Try to get a feel for what you need to do to get a network to classify the spiral pattern correctly.

3. Download `MLP.py`, `datasets.py` and `nn_learner1.py` from the Tutorial 7 website. Start Canopy, click on 'Editor', then open `nn_learner1.py`. This script trains a neural network on a spiral pattern, which in its default configuration is probably not going to do very well. Your job is to modify the neural network configuration (you can use what you learned from the previous exercise). The multi-layer perceptron library provided in `MLP.py` contains all the code for training with backprogation. The `MLP.MLP()` call creates a neural network. Here's how you can change its architecture:

   - The *layer_sizes* parameter is an array that specifies the number of inputs, number of neurons in the hidden layer and number of network outputs of the network. For instance, to create a network with 2 inputs, 4 neurons in the hidden layer, and 1 output, set *layer_sizes=np.array([2, 4, 1])*. To add a hidden layer with 7 neurons, set *layer_sizes=np.array([2, 4, 7, 1])*
   - The *f_activation* parameter is a string that selects the activation function for all hidden neurons. The choices you have are *'logsig'*,*'tanh'*, and *'relu'*.
   - The *f_output* parameter is a string that selects the activation function of the output neurons. The choices you have are *'logsig'*,*'tanh'*,*'relu'*,*'lin'*, and *'softmax'*. You can use *'softmax'* only when the output layer has more than

one neuron - the network will optimise the cross-entropy cost as opposed to mean squared error.

- The *sigma* parameter is a number that specifies the variance of the network weights and biases initialisation. When neural network is initialised, its weights and biases are set to random values drawn from gaussian distribution of zero mean and the specified variance. Changing this parameter changes the starting state.

The `train()` method of an instance of the `MLP` object trains the neural network using backpropagation. The method shows its progress by printing as well as plotting the cost and classification error over the training epochs. Here's how you can control the training process:

- The *num_epochs* parameter is an integer that specifies the total number of epochs to train for.
- The *alpha* parameter is a float that specifies the learning rate.

After the training, an image representation of the classification on the training data is shown.

Quiz question Write down and submit the specification of the architecture, number of epochs, the learning rate, as well as the train and test error of your best network for the spiral problem (no need to submit values of the weights and biases, just the configuration of the architecture). What are the reason behind your choices? For instance, if you decided to add more hidden layers, why did you do that? If you decided to use a different activation function, why do you think it might help?

Iris

4. Train a neural network on the 2-dimensional version of the Iris dataset. To use the Iris dataset, change the *dset = datasets.dataset_spiral()* line to *dset = datasets.dataset_iris(attributes=2)*. This dataset uses only two attributes of the 4-dimensional Iris dataset. You will have to change the number of outputs of your network, because this Iris dataset represents data of 3 classes. You might want to change the architecture and training time of your network, as the dataset should be easier to classify than the spiral (although you might not be able to get a 0 training error). Like before, at the end of the training a visualisation of the classification regions will display.

   Quiz question Write down and submit the specification of the architecture, number of epochs, the learning rate, as well as the train and test error of your best network for the 2-attribute Iris problem.

5. Train a neural network on the 4-dimensional Iris dataset. To specify that you want a Iris dataset, change the *dset = datasets.dataset_iris(attributes=2)* line to *dset = datasets.dataset_iris(attributes=4)*. Now the number of inputs is 4. Train a neural network on this classification task.

   Quiz question Write down and submit the specification of the architecture, number of epochs, the learning rate, as well as the train and test error of your best network for the 4-attribute Iris problem. Does the model train better/worse on this version of the dataset? If so, why do you think this happens?

Digits

6. Train a neural network on the set of 8x8 pixel images of digits from 0-4. To use the digits dataset, set *dset = datasets.dataset_digits(n_class=5)*. The input is a set of 64-dimensional vectors representing images with digits corresponding to 5 classes. Modify your network architecture accordingly and try to find a configuration that learns the classification well. After training, a sample of 32 images with the test labels predicted by your classifier will be shown.

   Quiz question Write down and submit the specification of the architecture, number of epochs, the learning rate, as well as the train and test error of your best network for the 5-class digits problem.

7. If you want, you can try to train your network on all 10 classes of the digits. Just set *dset = datasets.dataset_digits(n_class=10)*.

Support Vector Machines

8. Answer the following:

   (a) What's a feature space?

   (b) What's the relation of the kernel function to the feature space?

9. Repeat the classification of the spiral, Iris, and digits datasets using a Support Vector Machine. To get started, download `svm_learner1.py` from the Tutorial 7 website. This code uses *sklearn*'s Python library for Support Vector Machine training. You can control your model with the following parameters passed in to the *svm.SVC()* method:

   - The value of the *C* parameter is a penalty for misclassification. Set it to a small value to allow some training error, increase it to increase the penalty for training error.

- To use the linear kernel set the kernel argument to *kernel='linear'*
- To use the Radial Basis Function kernel, set the kernel argument to *kernel='rbf'* and specify the value of the *gamma* argument (a positive float) to parameterise the kernel function.
- To use the polynomial kernel, set the kernel argument to *kernel='poly'* and specify the value of the *degree* argument (a positive integer greater than 1) to parameterise the kernel function.

For more information on *sklearn's* SVM API refer to http://scikit-learn.org/stable/modules/svm.html. This implementation of SVM uses the LIBSVM https://www.csie.ntu.edu.tw/~cjlin/libsvm/ library.

You can change datasets in the same way as you did in the previous exercise. You don't need to specify the number of inputs and outputs in the SVM model when switching datasets - the *sklearn's* library figures it all out.

After the SVM training, the script will print the train and test errors as well as show a plot of the classification (with the exception of the 4-attribute Iris dataset, for which there is no visualisation).

Quiz question Write down and submit the C parameter value, the kernel function and its parameter value, as well as the train and test error of your best SVM for each problem. Did you find working with SVM easier/harder than with a neural network? Why?