MATH 829: Introduction to Data Mining and Analysis
Lab 2: neural networks

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

March 18, 2016

Using FANN in Python

- Install FANN on your computer - see instructions at `http://leenissen.dk/fann/wp/help/installing-fann/`
- Install the fann2 Python module (`pip install fann2`).
- Test your installation by modelling the XOR function:

Input1	Input2	Output
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

  In order to do so we will:
  1. Create a file containing "training data"
  2. Fit a neural network model.
- Structure of the FANN data files:
  1. First line = Number-of-observations Number-of-inputs Number-of-outputs (separated by a space)
  2. Other lines: One line is an input, next line is the corresponding output (values separated by spaces)

Modelling the XOR function

Create a file named `xor.data` and containing:

```
4 2 1
-1 -1
-1
-1 1
1
1 -1
1
1 1
-1
```

Train a neural network with 1 hidden layer containing 4 hidden nodes:

```
from fann2 import libfann

connection_rate = 1; learning_rate = 0.7
num_input = 2; num_hidden = 4
num_output = 1

desired_error = 0.0001
max_iterations = 100000
iterations_between_reports = 1000

ann = libfann.neural_net()
ann.create_sparse_array(connection_rate, # or ann.create_standard_array
  (num_input, num_hidden, num_output))
ann.set_learning_rate(learning_rate)
ann.set_activation_function_output(
   libfann.SIGMOID_SYMMETRIC_STEPWISE)

ann.train_on_file("xor.data", max_iterations,
  iterations_between_reports, desired_error)

ann.run([-1,-1])
```

The zip data

0. Load the zip data.
1. Convert the outputs to binary vectors in $\{0, 1\}^{10}$.

```
y_train2 = np.zeros((len(y_train), 10))

for i in range(len(y_train)):
    y_train2[i, np.int(y_train[i])] = 1.0
```

2. Load the zip data and write them to a file using

```
def format(value):
    return "%.6f" % value

def write_fann_data(filename, X, y):
    n = X.shape[0]
    p_input = X.shape[1]
    p_output = y.shape[1]
    with open(filename, 'w') as f:
        f.write("%d % d % d\n" % (n,p_input,p_output))
        for i in range(n-1):
            f.write(" ".join(format(x) for x in X[i,:]) + "\n")
            f.write(" ".join(format(x) for x in y[i,:]) + "\n")
        f.write(" ".join(format(x) for x in X[n-1,:]) + "\n")
        f.write(" ".join(format(x) for x in y[n-1,:]))
```

3. Fit neural networks with 1 hidden layer and different number of hidden nodes to the data.

Using Pybrain (XOR function)

```
from pybrain.datasets import SupervisedDataSet
from pybrain.tools.shortcuts import buildNetwork
from pybrain.supervised.trainers import BackpropTrainer

dataModel = [
    [(0,0), (0,)],
    [(0,1), (1,)],
    [(1,0), (1,)],
    [(1,1), (0,)],
]

ds = SupervisedDataSet(2, 1)
for input, target in dataModel:
    ds.addSample(input, target)
# create a large random data set
import random
random.seed()
trainingSet = SupervisedDataSet(2, 1);
for ri in range(0,1000):
    input,target = dataModel[random.getrandbits(2)];
    trainingSet.addSample(input, target)

net = buildNetwork(2, 2, 1, bias=True)

trainer = BackpropTrainer(net, ds, learningrate = 0.001,
    momentum = 0.99)
trainer.trainUntilConvergence(verbose=True,
                              dataset=trainingSet,
                              validationProportion=0.25,
                              maxEpochs=10)

print '0,0->', net.activate([0,0])  # Try other inputs as well...
```