Classification and Representation Learning

Tutorial 1 : Multilayer Perceptron

Hoel Le Capitaine

Academic year 2017-2018

The goal of this exercise is to implement a multi-layer perceptron to perform classification on the given files:
nonlinear-classification.data and circular-classification.data.

- the MLP is composed of 2 input neurons x_i, one output neuron o_k, and K hidden neurons in a single layer (y_j).
- the output neuron sums its inputs with K weights W^{out} and a bias b^{out}, and uses a threshold transfer function on this result: it predicts = +1 if it is positive, -1 otherwise.
- each of the K hidden neurons receives 2 weights from the input layer, giving a $2 \times K$ matrix W^{hidd}, and has a bias b^{hidd}
- the hidden neurons use a logistic transfer function

Implementation

```
from numpy import *
import matplotlib.pyplot as plt
# Load the data set
data = loadtxt('nonlinear_classification.data')
# Separate the input from the output
X = data[:, :2]
T = data[:, 2]
N, d = X.shape
# Parameters
eta = 0.05 # learning rate
K = 15 # Number of hidden neurons
# Weights and biases
max_val = 0.1
W_hid =random.uniform(-max_val,max_val,(d,K))
b_hid =random.uniform(-max_val,max_val,K)
W_out = random.uniform(-max_val, max_val, K)
b_out = random.uniform(-max_val, max_val, 1)
```

Feedforward

The feedforward pass is already implemented in the method **`feedforward`**

```
# Logistic transfer function for the hidden neurons
def logistic(x):
    return 1.0/(1.0 + exp(-x))
# Threshold transfer function for the output neuron
def threshold(x):
    data = x.copy()
    data[data > 0.] = 1.
    data[data < 0.] = -1.
    return data
def feedforward(X, W_hid, b_hid, W_out, b_out):
    # Hidden layer
    Y = logistic(dot(X, W_hid) + b_hid)
    # Output layer
    O = threshold(dot(Y, W_out) + b_out)
    return Y, O
```

Backpropagation

The goal is to implement the backpropagation algorithm by comparing the desired output t_k with the prediction o_k

```
errors = []
for epoch in range(2000):
  nb_errors = 0
  for i in range(N):
        x = X[i, :]
        t = T[i]
        y, o = feedforward(x, W_hid, b_hid, W_out, b_out)
        if t != o:
            nb_errors += 1
        delta_out = (t — o)
        delta_hidden = 0.0 # TODO
        W_out += 0.0 # TODO
        b_out += 0.0 # TODO
    for k in range(K):
        W_hid[:, k] += 0.0 # TODO
    b_hid += 0.0 # TODO
    if nb_errors == 0:
      break
```

Questions

1. Visualize with the plot function the two datasets `nonlinear_classification.data` and `circular_classification.data` with different colours for the two classes (use your previous code)
2. Create the file `mlp.py` with the necessary code, and complete the backpropagation algorithm to classify `nonlinear_classification.data` or `circular_classification.data`. The learning rate can be fixed to 0.05. Learning should stop when no more examples are miscorrectly classified, or when 2000 epochs have already been made. Try different values for the number of hidden neurons (e.g. 2, 5, 10, 15, 20, 25) and observe how correctness and speed of convergence evolve.

Questions

3. Observe more precisely the behaviour of the network during learning by calling the graphical function `plot_classification` every five epochs.
4. The weights are initialized randomly between -1 and 1. Try to initialize them to 0. Does it work? Why?
5. For a fixed number of hidden neurons (e.g. $K = 15$), launch 10 times the same network. Does performance change? What is the mean and variance of the number of epochs needed? Vary the learning rate. How does performance evolve?

Questions

6. Instead of the logistic function, use a linear transfer function for the hidden neurons. How does performance evolve? Is the non-linearity of the transfer function important for learning?
7. Use this time the hyperbolic tangent function as a transfer function for the hidden neurons (method tanh() in NumPy). Does it improve learning? The derivative of the tanh function is given by $f'(x) = 1 - f(x)^2$
8. Use the Recitfied Linear Unit (ReLU) transfer function $f(x) = \max(0, x)$. (Find its derivative) What does it change? Conclude on the importance of the transfer function for the hidden neurons.