COMP3308/3608 Artificial Intelligence

Week 9 Tutorial exercises
Multilayer Neural Networks 2. Deep Learning.

Exercise 1. ***Backpropagation (Homework)***
a) Show that the derivative of the hyperbolic tangent sigmoid function $a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$

is $1-a^2$.

b) Write the weight change rule for this function, using the result from a). See slide 26 from the lecture on backpopagation.

for an output neuron *i*:

$$\Delta w_{ji} =$$

for a hidden neuron *j*:

$$\Delta w_{kj} =$$

Solution:
a) The goal of this exercise is to show that there is a convenient way for computing the derivative of the hyperbolic tangent sigmoid similarly to the sigmoid function where the derivative was *a(1-a), a=f(n).*

$$f(n) = a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$$

$$\frac{\partial f(n)}{\partial n} = \frac{\partial}{\partial n}\left(\frac{e^n - e^{-n}}{e^n + e^{-n}}\right) = \frac{\left(e^n - e^{-n}\right)'(e^n + e^{-n}) - \left(e^n - e^{-n}\right)(e^n + e^{-n})'}{(e^n + e^{-n})^2} =$$

$$= \frac{\left(e^n - (-1)e^{-n}\right)(e^n + e^{-n}) - \left(e^n - e^{-n}\right)(e^n + (-1)e^{-n})}{(e^n + e^{-n})^2} = \frac{(e^n + e^{-n})^2 - \left(e^n - e^{-n}\right)^2}{(e^n + e^{-n})^2} =$$

$$= 1 - \frac{\left(e^n - e^{-n}\right)^2}{(e^n + e^{-n})^2} = 1 - a^2$$

b) for an output neuron *i*:

$$\Delta w_{ji} = \eta \cdot \delta_i \cdot o_j = \eta \left(d_i - o_i\right)(1 - o^2{}_i)\, o_j$$

for a hidden neuron *j*:

$$\delta_j = f'(net_j) \sum_i w_{ji}\, \delta_i = (1 - o^2{}_j) \sum_i w_{ji}\, \delta_i$$

$$\Delta w_{kj} = \eta \delta_j o_k = \eta \left(1 - o^2{}_j\right) \sum_i w_{ji}\, \delta_i\, o_k$$

Exercise 2.

a) A 2-layer feed-forward neural network with 10 input units, 5 hidden units and 3 output units contains how many weights? (Include biases.) Show your work.

Answer:

weights: 10*5 + 5*3 = 65
biases: 5*1 + 3*1=8
total: 65+8 = 73

b) The backpropagation algorithm iterates until it minimizes what?

Answer: The sum of the squared errors of the output values over all the training examples.

c) Is the backpropagation algorithm guaranteed to achieve 100% correct classification for any linearly-separable set of training examples, given a sufficiently small learning rate? Explain briefly.

Answer: No, it will iterate until a local minimum in the squared error is reached.

Exercise 3. Cybenko's theorem (slide 39) states that any continuous function can be approximated by a backpropagation network with 1 hidden layer. Why do we use networks with more than 1 hidden layer?

Answer: This is an existence theorem, i.e. it says that there is a network with 1 hidden layer that can do this (and doesn't tell us how to find this network). However, the theorem doesn't say that a network with 1 hidden layer is optimum in the sense of training time, ease of implementation, or (more importantly) generalization ability (i.e. ability to classify correctly new examples).

Exercise 4. Using Weka

1. Load the iris data (iris.arff). Choose 10-fold cross validation. Run the Naïve Bayes and Multilayer percepton (trained with the backpropagation algorithm) classifiers with the default parameters and compare their performance. Note that multilayer perceptron in Weka is under "Functions".

Which classifier was more accurate? Which one was faster to train?

2. Select "MultilayerPerceptron" as a classifier. In the pane "Choose" (where "MultilayerPerceptron" should appear click to open the options window. If you click "More" you can learn what each of them means. What does "a" in the number of hidden layers mean? Experiment with 1 hidden layer and different number of neurons in it. As you can see the classification performance of the backpropagation network is very sensitive to the architecture. Choose "GUI=true" from the window with the options to visualize the different networks.

Exercises using Matlab and Matlab's Neural Network demos

Exercise 5. Convergence with different learning rates

In Matlab, type `nnd12sd2` to run the demo.

Try different starting positions and different learning rates. What is the influence of the learning rate? What happens if the learning rate is too big or too small?

Answer: The learning rate determines the speed of the training; too big – faster training but oscillations, and overshooting of the minimum; too small – slow convergence

***Exercise 6.* Backpropagation with momentum**

In Matlab, select "Momentum Backpropagation" from Demos -> Toolboxes -> Neural Networks (upper left window)

What is the advantage of using a momentum?

Answer: Speeds up learning by allowing to use a larger learning rate while maintaining the stability.

***Exercise 7.* Generalization**

In Matlab, type nnd11gn or select "generalization" from Demos -> Toolboxes -> Neural Networks (upper left window)

Try a simple function (i.e. choose a small difficulty index) with too many hidden neurons. What is the reason for the poor generalization?

Answer: Overtraining (overfitting).

***Exercise 8.* Convolution in deep NNs**

Follow the convolution example on this website to see how the convolved features are computed:
http://deeplearning.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/

***Exercise 9.* Backpropagation example**

Follow the backpropagation walkthrough example prepared by Josh Stretton and available from Canvas; it is an extended version of the example from the first lecture on slides 33-37. The goal is to understand the forward and backward passes of the backpropagation algorithm and how the weights are updated.