F20/21DL. Data Mining and Machine Learning

Lab 10. Multilayer Perceptron

Covering Practical work to be done by students in Week 10

The purpose of this lab is:

1. to practice what we have learned so far:
   - Training algorithms for neural networks;
   - error correction learning and gradient descent;
   - theoretical issues related to building and running deep and fully-connected neural networks.
2. understand practical issues of running Neural networks, e.g.
   - the importance of choice of network architectures, activation functions, error functions, and learning parameters
   - the problem of neural network overfitting and ways of mitigating the problem in practice.
3. prepare for the electronic test.
4. to help you to make progress with Python tutorial and your DM & ML portfolio.

1 Understanding the Algorithms deployed in neural network training

Prepare for Test on algorithm for Linear Regression.

1.1 Perceptron

Lets take a few pictures from the small "emotion recognition" data set:

Picture	Cell 33	Cell 42	Cell 48	Cell 58	Face expression
P1	White	Black	White	White	Happy
P8	Black	White	Black	Black	Sad
P10	White	Black	White	Black	Happy

Lets take its reduced version:

Picture		v_{Cell42}		v_{Cell58}	d_{Happy}
P1		1		0	1
P8		0		1	0
P10		1		1	1

Here is the Perceptron that will correspond to this data:

$$\begin{tikzcd}
v_{Cell42} \arrow[r] & w_1 \arrow[dr] & & \\
v_{Cell58} \arrow[r] & w_2 \arrow[r] & \theta, d_{Happy} \arrow[r] & v_{Happy}
\end{tikzcd}$$

1. **Manually** execute the error-correction algorithm (from the lecture slides) on this Perceptron and on this data set, using the following training parameters:

    - Learning rate $\eta = 0.3$
    - $\theta = 0$, suppose we do not train θ
    - Activation function – just identity (no function applied), i.e.: $v_{Happy}(t) = p_{Happy}(t) = w_1(t) * v_{Cell42}(t) + w_2(t) * v_{Cell58}(t)$
    - Random weights initialised at time= 1: $w_1(1) = 1$, $w_2(1) = 1$
    - **Error-signal**: $e_{Happy}(t) = d_{Happy}(t) - v_{Happy}(t)$
    - Error-correction learning rule: $\Delta w_1(t) = \eta * e_{Happy}(t) * v_{Cell42}(t)$, $\Delta w_2(t) = \eta * e_{Happy}(t) * v_{Cell58}(t)$; $w_1(t+1) = w_1(t) + \Delta w_1(t)$; $w_2(t+1) = w_2(t) + \Delta w_2(t)$
    - Time counter: $(t = 1)$ send $P1$; $(t = 2)$ send $P3$, $(t = 3)$ send $P9$, end with computing $w_1(4)$, $w_2(4)$ at step $t = 4$.
    - **Record all intermediate steps and parameters, be ready to answer questions**

2. **Manually** test the resulting trained Perceptron on:

    - Test 1: "a neutral (Happy?) face with noise":
      `White, White, ???`
    - Test 2: "a Happy face with a beard":
      `Black, Black, ???`
    - What is the output v_{Happy} for these two inputs? (Please count Black as 1 and White as 0)
    - Assume that you apply a squashing function to interpret the output of this neural network.
      The function returns 1 if the value of neuron Happy is greater or equal 0.5; and it returns 0 otherwise.
      With this function, what predictions your testing experiment on the two testing examples will give?
      What will be the accuracy of this network on the given test set, assuming that expected classification of both pictures is Happy?

1.2 Deep fully-connected Neural networks

1. Lets try to practice estimating weight matrix sizes for deep fully connected neural networks

2. Suppose we construct a fully-connected neural network to classify the "small emotion recogniton data set" (in its original, non-reduced form).

   Suppose we decided to have two hidden layers, with 8 and 2 neurons, respectively. We also want an output layer to have two neurons, that will stand for "Happy" and "Sad", respectively.

   Draw such a neural network, and calculate the size of weight matrices in each layer. (Please omit the additional bias (weight 0) in the calculations, as it is added uniformly to all layers.) Be ready to answer questions.

2 DM & ML Portfolio

This part is to be completed in groups, and will be assessed during the labs. Marking scheme: this lab will bring you up to 2 points. 1 point for completing the task, 1 additional point for any non-trivial analytical work with the material.

2.1 Python Tutorial and Programming Practice (Prior to the lab)

This part is for your individual programming practice during the week.

- Watch recordings, and run the Python code accompanying tutorial **P6.1 Classification using Perceptron and MLP**.
- Make sure you can run this code using **your chosen data set**. In case you have any issues, contact your lab tutor and ask for help.
- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Decide whether you will be using a computer vision data set in this and next lab. If so, either pick one of your choosing (but not benchmark data sets like MNIST or FASNION MNIST), or use a simple smiley face data set provided on Canvas.
- Use the given code (”Perceptron” function if you use a nominal data set or a Keras library if you use computer vision data set) on a training set, measure the accuracy. Then measure the accuracy using 10-fold cross-validation on the training set. Use the major metrics: accuracy, TP rate, FP rate, precision, recall, F measure, the ROC area if needed.
- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Experiment with various parameters that control the learning. For example: the learning rate, the number and size of layers the number of iterations, batch size, epochs and momentum, and validation threshold. Record the findings in suitable tables:

Multilayer Perceptron	Accuracy	TP	FP	TN	TP	Sensitivity	Specificity	Precision	Recall	Area Under RoC Curve
Architecture 1 (Layers $N \times M \times ...$, learning rate $\eta = ...$, No iterations $= ...$, optimisation algorithm $= ...$, activation functions $= ...$)										
Architecture 2 ...										
Architecture 3 ...										
Architecture 4 ...										
Architecture 5 ...										
Architecture 6 ...										

- Which architecture is the best performing? How do you tell?

Classifier 1 Using the best working Architecture, repeat the experiment, this time using training and testing data sets instead of the cross validation. That is, build the classifier using the training data set, and test the classifier using the test data set. Note the accuracy. Answer the question: Does the classifier generalize well to new data? How do you tell?

Classifier 2 Make new training and testing sets, by moving 30% of the instances from the original training set into the testing set. Note the accuracies on the training and the testing sets.

Classifier 3 Make new training and testing sets, by moving 60% of the instances from the original training set into the testing set. Note the accuracies on the training and the testing sets.

- Finally, analyse **Classifier 1**, **Classifier 2** and **Classifier 3** from the point of view of the problem of classifier over-fitting. Do you notice the effects of over-fitting? How? Note your conclusions in the Jupyter notebook.

  Note: to make conclusions about overfitting, you must compare accuracies of your three classifiers on training and on testing data sets.

- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Put all your results in a suitable form: it can be a table or a series of graphs, that visualise the variations of performance between different classifies.

Multilayer Perceptron	Accuracy	TP	FP	TN	TP	Sensitivity	Specificity	Precision	Recall	Area Under RoC Curve
Classifier 1										
Classifier 2										
Classifier 3										

2.2 During the lab:

- Make conclusions about your experiments with tuning parameters of the *multilayer perceptron.* Make conclusions: what was the influence of various parameters on the classifier's performance? Hypothesise why.
- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Make conclusions about the problem of overfitting of neural networks. What are the ways to stop neural networks from over-fitting? Try a few on your data set and see if it works?
- **The tutors will mark:** quality of your code, completeness of your tables/graphs that summarise the results of your group experiments and your analysis of the tables/graphs, i.e. what sort of conclusions you make, how well the conclusions reflect your understanding of the algorithms.

2.3 After the lab:

- *Group rep:* Make sure all group members have tasks for the week
- *Everyone:* Incorporate the discussion during the lab into your Python code
- *Everyone:* Incorporate all code used in the lab into your Portfolio repository.