Artificial Intelligence and Knowledge Engineering
Assignment 5: Neural networks

Przemysław Dolata, Julita Bielaniewicz

May 2023

1 Assignment goal

The aim of this exercise is to familiarize yourself with neural networks - one of the most commonly used models in machine learning. By completing the tasks, you will learn how to apply neural networks to a practical regression problem. You will learn the basics of the model logic and examine the impact of its most relevant hyperparameters, as well as understand how to recognize typical phenomena that may cause a decrease in prediction quality.

2 Introduction

The term „neural networks" encompasses an extremely broad class of information-processing systems with a common feature of structure: they consist of *neurons*, generally organised in *layers*, connected to each other by connections with assigned *weights*. In this exercise, you will focus on the MLP (*Multilayer Perceptron*) model. This model is an unidirectional neural network with one input layer, one or two hidden layers, and one output layer. The size of the input layer is defined by the dimension of the data vector X, similarly, the size of the output layer depends on the dimension of the result vector Y; the selection of the number and size of hidden layers is the responsibility of the network designer.

The MLP model, like many other neural networks, can be taught by back-propagation in a supervised mode, and this is precisely the approach you will use in this task. The learning base will consist of a certain set of texts with labels; the goal of the learning process will be to find a function that maps text X into a label Y, in other words, allowing the automatic labelling of future texts.

When selecting neural networks for a specific problem, it is crucial not only to choose the network architecture and the *hyperparameters* describing its structure and learning process, but also the cost function that will be optimised during learning. The choice of this function is generally defined by the type of problem to be solved and the available data. From this point of view, the tasks are most often divided into:

- classification, where we search for the mapping of $f(\theta, X) \rightarrow Y$, where Y is a nominal variable; nominal variables are not ordered, examples are e.g. animal species, vehicle brand,
- regression, where the mapping $f(\theta, X) \rightarrow Y : Y \in \mathbb{R}$ is searched for, i.e. Y is an ordinal variable, i.e. subject to ordering, e.g. age, length

Classification tasks usually use a logistic function (*logistic regression*), while in regression problems one of the most frequently used cost functions is the mean squared error. In both variants, the learning process consists of minimizing the cost function. The value of this function is therefore the most important signal that should be observed during model training to properly assess the progress of this process.

The training of neural models using the simple or stochastic gradient method is an iterative process. During each iteration a*forward propagation* is performed so that the model is queried on a sample from the training set, then the value of the error function is calculated, which finally allows you to perform *backpropagation*, i.e. calculating the value of the ∇ gradient for each neuron in the network using *the chain rule*. Then the weights of the neurons are updated according to the formula

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla$$

where η is a hyperparameter called *learning rate*). However, there are more complicated optimization methods that update the weights in a more sophisticated way. Performing this procedure for all samples in the training set is called *epoch*; as a rule, training a neural model requires repeating the process over many epochs.

MLP networks, like many other learning algorithms, require the data to have a vector representation. Therefore, if the data does not have such a structure, it is crucial to convert them to such a form. In general, this process is called *feature extraction*, and in the context of natural language processing applications, it is often referred to as *tokenization*, i.e. breaking text into a sequence of uniquely marked „tokens" (e.g. individual words or syllables). Since the process of full tokenization is quite complicated, for the purposes of this task you will carry out a simpler variant of the text data vectorization based on the dictionary. The procedure will consist of determining the D set of the most interesting words and assigning them unique indexes; then each text in the database will be converted into the vector $x \in \mathbb{N}^D$ containing the number of occurrences of the word with index i in position x_i. Remember to perform the initial data processing in the same way in all of the stages of working with the network - training, testing and processing new data (e.g. vectorization based on the same dictionary).

3 Tasks

As part of this list, you will tackle the problem of predicting the level of ridiculousness of a given joke from the Jester set. This collection contains about 25,000 funniness ratings for 100 different jokes that range from (-10.0, 10.0).

To simplify the task and allow you to focus on examining the neural network itself, the feature extraction module is attached to this manual. Your task will be to train a simple MLP neural network and test the key hyperparameters. Use the knowledge gained during the implementation of the previous list regarding the division of data into training and validation sets. Use the Weka package or Python capabilities (mainly `sklearn and matplotlib`) to complete the tasks.

It is crucial to start your experiments with as raw neural network as possible. If you are using Python and the `sklearn` package, specifically the classes `sklearn.neural_network.MLPRegressor`, make sure you pass the following hyperparameter values:

- `solver = 'sgd'`
- `alpha = 0.0`
- `learning_rate = 'constant'`

For Weki and module `MultilayerPerceptron` set:

- `learningRate = 0.001`
- `momentum = 0.9`
- ...and make sure the data label column is in numeric format - otherwise the network will start in classification mode!

Naturally, you don't have to stick to the given values - see: list of tasks, point 6.

Detailed task list:

1. Prepare a training and a validation set using the feature extraction procedure code attached to the list. If you intend to use Weka, it is recommended to perform a one-time data conversion and an export to one of the compatible formats. (5 points)
2. Test the operation of the basic MLP model with the default configuration of hyperparameters by training it on data from the Jester set. Trace the behavior of the model over time by visualizing the value of the cost function as a function of the number of epochs, paying attention to the values for the training set and the validation set. (20 points)
3. Investigate the effect of *learning rate* on performance: repeat learning for 3 different parameter values. Adjust the length of the learning process (number of epochs) if necessary. Plot the results as in the previous exercise. What happens when the learning rate is too slow? What if too high? (30 points)
4. Investigate the impact of MLP model size on performance: perform at least 3 experiments for models that differ in the number of neurons. When does the model stop fitting the data well? When does it begin to overfit the training set? (30 points)
5. Choose the best model from the above experiments and test it in practice: find (or write your own) joke text, transform it into a vector using the feature extraction method used in the tasks, and then query the neural model. Does the prediction match your expectations? (15 points)
6. $\star$ Examine any other parameter, such as regularization. What is its effect? What problem does it help to solve? How does it affect the results? (bonus up to 10 points)

The result of your work should be the implementation of experiments in Python or Weka, and a report containing the results of these experiments in the form of tables and/or graphs, along with a brief overview. Be sure to include the full set of hyperparameters used for each experiment. Send the report to the teacher at least 24 hours before handing over the list.

4 Literature

1. Materials from the lecture course led by prof. Piasecki (Lecture no. 8)
2. Jester Dataset
3. Tadeusiewicz R., M. Szaleniec - *The Lexicon of Neural Networks*
4. `scikit-klearn` user manual, „Neural network models (supervised)"