Shallow Neural Networks for Mushrooms

Due date: 11:59pm Sunday Dec 12, in Blackboard

We have learned several classification methods. One of them that has caught great attention today is neural networks (NN). In this project, we will implement a simple neural network model to predict whether a particular mushroom that you see is edible or poisonous. By implementing this simple model, you will understand better how a neural network works, and stay healthy.

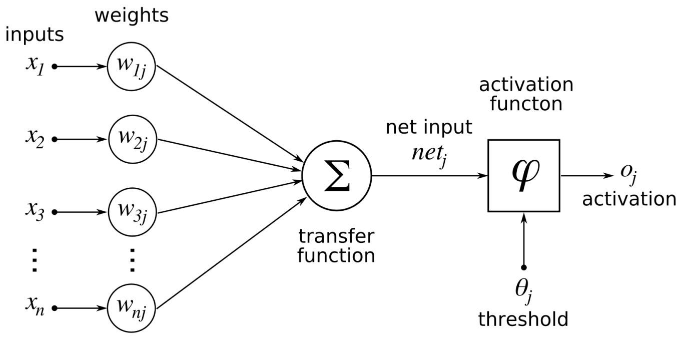

Figure 1. A single neuron

Figure 1 shows the structure of a neuron. A neural network often contains three parts (each part contains one or more neurons): Input layer, (1 or more) Hidden layer, and Output layer. Figure 2 shows an example of the neural network structure.

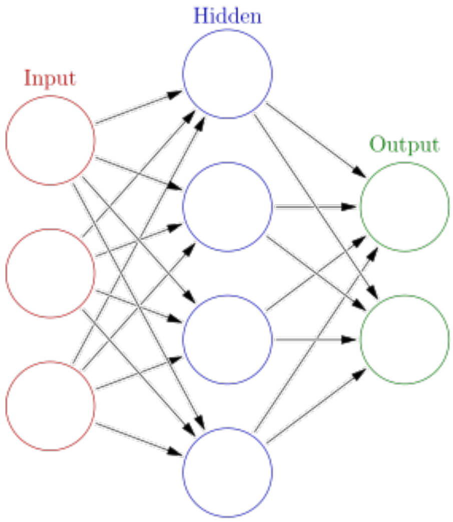

Figure 2. An example of Neural Network

Backpropagation is an algorithm widely used in the training of feedforward neural networks for supervised learning. It efficiently computes the gradient of the loss function with respect to the weights of the network. Assume that, for one output neuron, the squared error function is $E = L(t, y)$, where E is the loss for the output y and target value t. For each neuron j, its output o_j is defined as $o_j = \varphi(net_j) = \varphi(\sum_{k=1}^{n} w_{kj} o_k)$, where φ is an *activation function*. **We use *logistic function* for the**

activation and *squared error function* in this project. If you are not familiar with these terms, we refer you to the Wikipedia page https://en.wikipedia.org/wiki/Backpropagation for more information.

To calculate the partial derivative of the error with respect to a weight w_{ij}, we use the chain rule twice: $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$.

Now let us look at each of the three factors in the above equation. First, $\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (\sum_{k=1}^{n} w_{kj} o_k) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i$. If the neuron is in the first layer after the input layer, o_i is just x_i.

Second, the derivative of the output of neuron *j* with respect to net_j is $\frac{\partial o_j}{\partial net_j} = \frac{\partial \varphi(net_j)}{\partial net_j}$.

Finally, if the neuron is in the output layer, then $o_j = y$ and we have $\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y}$. Moreover, if the logistic function is used as activation and square error as loss function, we can get $\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y} \frac{1}{2} (t - y)^2 = y - t$. For a neuron *j* in an arbitrary inner layer of the network, please refer to https://en.wikipedia.org/wiki/Backpropagation for details.

Introduction to Python

You will write Python code in this project. Python is a popular high-level interpreted programming language with good code readability. For those who are not familiar with Python, this **Begin*ner*'s Guide** (https://wiki.python.org/moin/BeginnersGuide) is a good start. You can also find a good interactive Python tutorial **here** (https://www.learnpython.org/).You can check out **PyCharm** (https://www.jetbrains.com/pycharm/) if you need an IDE, they provide free student licenses.

Data

The data set we use (provided in the directory) is originally from **UCI Machine Learning Repository** (http://archive.ics.uci.edu/ml/datasets/Mushroom). It includes description of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. We have divided it into 3 parts: training set, validation set and testing set. You can find the detailed description and attribute information from both the website and the description file in the data folder.

What to Do

1. Write a program that converts the original dataset (http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/) to a format that fits our NN's input form. Your program should transform original dataset's records (filename is **agaricus-lepiota.data**, provided in the project sub-folder, where each line represents one record with 23 features—see **agaricus-lepiota.names** for details) into the so-called *one-hot* binary code form (the first 2 numbers are classes — *poisonous* or *edible*, while others represent the remaining features—each feature may have multiple values, and you should make sure that your binary code can represent each of the values). Please take a look at **data_representation_example.txt** for some details. Then split the converted dataset into 3 files: **training.txt** for network training, **val.txt** for validation, and **testing.txt** for network testing. (You may name these files by yourself, but make sure to modify the code to fit your files). **Sample_testing.txt** is a sample file showing the exact format that we need.

2. Implement these 3 files: **formulas.py**, **models.py**, and **proj_test.py**.

formulas.py contains the functions that we need for NN training. You need to complete all those 4 functions.
models.py contains the main structure of our NN. You need to complete the evaluation and backpropagation parts.
proj_test.py is a testing file that tests our NN and reports the accuracy. You need to complete the training, validation, and testing parts.

3. After your implementation, run **proj_test.py** and record the results.

4. A report with some partial running results.

GRADING

1. Correctness of models (60%): You need to make sure the logic of your models are correct.
2. Quality of codes: (30%): You will lose points if your code lacks readability. Comments would be helpful for improving your code quality.
3. Write-up (i.e. readme.txt) (10%): Your write-up should include your name, descriptions of your implementation, anything the TA needs to know to run your program, acknowledgment of any part that does not yet work, and names of the students with whom you have discussed the project. NOTE: This is a single-person project. Optionally, you may discuss the project with other student(s), but only at a high level. You need to write all the code yourself. Copying code will be easily detected and subject to academic integrity severe punishments. If you do discuss with other students, you need to write down their names here.

SUBMISSION

You need to submit your project through the Blackboard system.