DATA 401: Data Science **Fall 2020**

Jonathan Ventura **Hunter Glanz**

Project 3

Neural Networks

Due Dates

The deliverables for this project are simplified and streamlined. There will be no oral presentations, although later in the quarter, we will have a short discussion of the results of the project and the lessons learned.

Deliverable	**Date**
Code	Sunday, November 8 (end of the day)
Project Report	Sunday, November 8 (end of the day)

Logistics

This is a team project. The teams are posted on Canvas.

Learning Objectives

The project has the following learning objectives:

- Build machine learning datasets
- Perform feature engineering and simple data integration tasks when building machine learning datasets
- Use linear regression methods for predictive analysis
- Implement backpropagation for a fully-connected feed-forward neural network
- Build predictive neural network models
- Compare neural network models to linear regression models.

The Assignment

In class we covered in detail the work of fully-connected feed-forward neural networks, and the backpropagation algorithms for training neural networks on input data. In this project you will implement the forward propagation and backpropagation algorithms for such neural networks ***from scratch,*** will use your implementation to train predictive models for regression, and will compare these models with the results of linear regression on the same data.

The dataset provided to you requires some additional cleaning and preprocessing, subject to the instructions below. At the same time, feature engineering is not as major a component of this project as it was in the previous two. You will have to deal with missing values, and you will have to engage in some feature selection and feature engineering of your own, but you will not be using any external datasets, or data extraction techniques.

We document the individual steps of your project below.

Dataset

In 2017 Russian Savings Bank (Sberbank) ran a Kaggle competition asking for predictions of real estate prices. Sberbank provided a training of over 20,000 real estate listings, complete with detailed information about both the properties, and the demographic data related to the location of each listing. It also provided a test set (which we will NOT be using in this project), and a separate file of macroeconomic factors tied to a specific day for the range of dates in the training set.

The real estate data is coming from the property sales listing in Moscow and the Moscow region. All values in the ***sub_area*** column refer to either subdivisions/neighborhoods in Moscow proper (often represented as the name of the closest metro station), or to villages/towns in the Moscow region.

We are making the following files available to you at /datasets/sberbank-russian-housing-market on data401.jventura.net and at https://storage.googleapis.com/data401-datasets/sberbank-russian-housing-market.zip .

File	Purpose
`train.csv`	The training set data for the Sberbank real estate dataset
`macro.csv`	Macroeconomic characteristics data
`data_dictionary.txt`	Data Dictionary

Target (Dependent) Variable

Our overarching goal is to predict ***the price of each house in the dataset*** from other available data.

This variable is called ***price_doc*** and it is available in the last column of the `train.csv` file.

Independent Variables

All other variables in the `train.csv` file are candidate independent variables for your analysis. In addition, if you join `train.csv` and `macro.csv` on the date column, you can use any attributes from `macro.csv` as candidate independent variables. All of this is subject to conditions described below.

Feature Engineering

Given the data from the Sberbank dataset, you will assemble your final dataset as follows:

- Your dataset shall contain ***as many rows as possible***, preferably ALL available rows from `train.csv`. Some rows have ***missing values.*** It is preferable that you figure out how to deal with missing values in the attributes of interest to you to removing rows from the dataset simply because some values are missing. Any decisions for removing rows from the dataset much be explicitly justified in your report (the expectation is that you remove data only if the data point is deemed to be really useless).
- Your dataset shall contain at ***most 50 numeric features.*** Of these, ***no more than 10*** features can be macroeconomic data from the `macro.csv` file.
- Your dataset ***shall not contain*** any ***dummified variables for geographic location*** of the properties (specifically, the ***sub_area*** column). The dataset contains a lot of numeric attributes describing the demographic data of the neighborhood/development/area around the property. Use these features instead of dummyfing city/town, and other geographic data. (You can dummify ***product_type*** variable if you so desire)
- Your dataset ***shall not contain*** the timestamp for the house sale. Instead, you can use the values of up to 10 (see above) macroeconomic characteristics as proxies for the date.
- You can combine columns from the original data files `train.csv` and `macro.csv` as you see fit to create new features.
- You can use ***any utility code you want*** from any package ***in support of your dataset creation*** process.
- Your final dataset shall treat all variables ***as numeric***.

You are welcome to try to optimize the set of features you will be using for predicting the house prices, but strictly speaking, this is not a requirement. You should be able to explain why you selected the attributes for your final dataset, and this explanation shall be found in your report.

Linear Regression Models

You will be using the same dataset for both linear regression and neural network modeling.

For linear regression you are allowed to:

- Develop any code from scratch
- Reuse any linear regression code from Project 1 ***for any of the team members.***
- Use off-the-shelf implementations of linear regression.

Your goal is to obtain the prediction accuracy data that can be easily interpreted. While you will be optimizing the SSE, it makes sense to report your findings in terms of MSE - as it shows the error per prediction.

You can conduct some limited model selection studies to see if removing certain attributes from your dataset improves prediction (it might happen if you have some strange collinearity in your attributes), but strictly speaking this is not required -- our goal is to compare the linear regression performance on the ***same dataset*** as the neural network models will see, and we will not be doing any model selection for neural networks.

Fully-Connected Feed-Forward Neural Networks.

You will implement ***from scratch*** the forward- and back-propagation procedures for a family of fully-connected feed-forward neural networks. In your implementation, you are allowed to rely on general vector and matrix algebra from ***NumPy,*** but you ***are not allowed*** to use any ***scikit-learn, nltk, tensorFlow, pyTorch, keras*** or other packages that involve implementations of machine learning techniques.

To help yourself, you may want to implement a `NeuralNetwork` class, with the following set of hyperparameters:

- ***Layers:*** an Integer value representing the total number of ***hidden layers*** in the network (input and output layers are extra)
- ***Nodes:*** an integer array of size [0,..,Layers+1] containing the dimensions of the neural network. `Nodes[0]` shall represent the input size (typically, 50), `Nodes[Layers+1]` shall represent the number of output nodes (typically, 1). All other values `Nodes[i]` represent the number of nodes in hidden layer `i`.
- ***NNodes:*** a possible alternative to the ***Nodes*** parameter for situations where you want each hidden layer of the neural network to be of the same size. In this case, the size of the output layer is assumed to be 1, and the size of the input layer can be inferred from the dataset.

- **Activations:** *an array of size [0,..,Layers+1] (for the sake of compatibility)* in which `Activations[0]` and `Activations[Layers+1]` are not used, while all other `Activations[i]` values are labels indicating the activation function used in layer `i`. This allows you to build neural networks with different activation functions in each layer.
- ***ActivationFn:*** a possible alternative to ***Activations*** when all hidden layers of your neural network use the same activation function.

(Please note, that without loss of generality, all your hidden layers, and the input layer **shall include an intercept node**. This node ***shall not count*** under the `Nodes[i]` cap. For example if `Nodes[i] = 5`, the layer `i` of the neural network shall contain six nodes: five hidden nodes, and one intercept node, whose value is automatically set to 1. The intercept node is NOT connected to any nodes in the previous layer, but ***is connected*** to all nodes in the layer that follows.)

Generally speaking, you are not required to implement more than one activation function for this project, although it is recommended. However, even if you only implemented one activation function, your implementation of the neural network shall be generic enough to admit additional activation functions as they get implemented in the future.

Your implementation of the backpropagation algorithm shall take additional arguments as needed (e.g., ***the learning rate*** *and* ***the batch size***) and shall produce as output a description of a neural network model (i.e., all NN coefficients).

Your backpropagation algorithm shall implement stochastic gradient descent controlled by the ***batch size*** parameter (the final batch in each epoch can contain fewer data points, if this makes your life easier).

For neural network models, your goal is to maximize the accuracy of prediction. Your hyperparameters (see above) are the number of hidden layers, the number of nodes in each hidden layer, and the assignment of activation functions (if multiple functions are implemented) to the hidden layers. Additionally, you may need to optimize the learning rate to achieve convergence of the NN, and to fiddle with batch size to make your implementation converge in observable time.

Analysis of results.

The final goal of the project is to compare your best Neural Network model to your baseline Linear Regression model, and to determine which model gives you better accuracy. Separately, in addition to comparing accuracy, you may discuss the compute time it takes for each method to produce the result.

Putting it all together.

Submit your dataset on Canvas by the deadline.

Write a report documenting your work. Your report must conform to the following standards.

1. **Formatting.** The report must be written in a form of an academic technical report, and consist of the title, a list of authors (names of all team members alphabetical order, unless another order is warranted) with contact details, the abstract, and the body of the report broken into individual sections as necessary.
2. **Text.** The report must be text-processed, and submitted to the instructors in PDF form. Each team shall continue maintaining the electronic version of the report[1]. The text must be in a serif font (e.g., *Times New Roman*).
3. **Narrative/Structure.** The report must contain the following information, ***organized in a form of coherent and cohesive narrative[2]****:*
    1. An **Introduction** presenting an overview of your work.
    2. Description of the **dataset creation process**: detail how you chose the final features for your datasets, why you chose them. If any new features were produced by combining existing features discuss how and why.
    3. Discussion of your **linear regression modeling implementation**, and any analysis pipelines you created.
    4. Discussion of your **implementation of neural networks.** Give sufficient detail for the instructors to understand your approach. Describe all parameters implemented, as well as other aspects of your implementation, such as activation functions.
    5. Discussion of the **evaluation process** you used to fit linear regression and neural network models. Discussion of any model selection procedures for linear regression and any grid search/hyperparameter turning for Neural Networks.
    6. **Results** of the evaluation process including your comparison.
    7. **Any general observations** you may want to add concerning the differences in the structure of the best predictive and statistical models
    8. Concluding remarks.
4. **Submission.** Submit your report and code on Canvas.

Note: Please note, that the **quality of your report**, including, but not limited to the **quality of your writing** will be a major component of your Project 1 score. This makes the report **an important part** of your assignment, *and not an afterthought it often is in your other coursework.*

Final touches.

Create your code deliverable. It shall contain all code you needed to create your dataset, fit regression models, and evaluate them. Your code must be accompanied by a README file documenting how the code is to be run. If your final dataset differs from the one submitted earlier, include the final dataset with your code submission.

---

[1] This course as well as the Data Science capstone will feature discussions about writing. You will be using samples of your own writing from this class in preparation and during these discussions.

[2] For now, view the notions of coherence and cohesiveness informally. You will, however, encounter the definitions of these terms in the Data Science Capstone sequence.