Project 2 Report

Simon Dayer Khalil Merzouk Arnaud Guibbert
CS-433, Machine Learning, EPFL, Switzerland

***Abstract*—Machine learning keeps growing further and further. It became the main research direction and proved its utility in almost all fields, such as image processing, speech processing, prediction systems, etc. During this project, we are affiliated with one of the numerous scientific laboratories of EPFL to apply these machine learning tools to real research problems.**

I. Introduction

In this project, we are entrusted with a dataset by the Transport and Mobility Laboratory of EPFL "TRANSP-OR", which is active in modeling, optimization, and simulation of transportation systems. Our main goal is to study how people choose their mode of transportation and develop a prediction model with given data on the traveler's trips in Montreal, Canada. A subsidiary goal has also been given by the lab mentor. We were asked to develop a neural network and try to improve the predictions by adding regularization to the model. In order to respond to the demand of the laboratory this paper will be organized as follow:

- First, we prepare the data for the model by importing it, selecting the inputs we are working with, then clean them.
- Secondly, we develop a neural network and all the metrics we need to assess the performances of our model.
- Finally, we test the neural network with and without regularization methods and analyze the results.

II. Experimental Setup

We used Pytorch [1], Pandas [2], Pytorch Lightning [3] and scikit-learn for confusion matrices. In order to ensure reproductibility, we added seeds where it is necessary. GPUs were not used in this project.

III. Data Preparation

A. Importing the data

The first challenge we faced while dealing with a Geo-located dataset is to turn a shapefile file into a csv file to import it into Jupyter notebook. To convert our dataset, we use the geographic information system *"QGIS"*, which allows us to load a shapefile and export it as a csv file. We can then divide the preparation of the data into two main tasks.

B. Data pre-processing

Since datasets can sometimes be messy, a pre-process is needed. The raw dataset[1] contains about 185'000 rows, where each one of them represents one trip of a user. Trips are categorized by three features:

- start and end time
- motivation of trip (work, leisure, etc)
- mode of travel

Nevertheless, more than half of the trips (110'000) don't show any mode of travel. This can be explained by the fact that the data may have come from a survey where some of the users did not give this information. In order to determine a prediction model, all those trips were not taken into account. We also convert all the data into a suitable format. For instance, converting dates into date time format and check that all data types are usable. Then, the next important part of pre-processing for ML model is to extract relevant features. Following this idea, we had to understand more about the context of the data collection. As each trip includes one person for one mode of transportation we average them by hour for each day of the week in order to understand the evolution of the demand of trip during the day. Note that this analysis was made on the whole dataset to be representative.
After plotting the result, we clearly see two different patterns. As the demand vary during weekdays and the weekend , we reiterate the process after splitting the trips between weekdays and weekends.

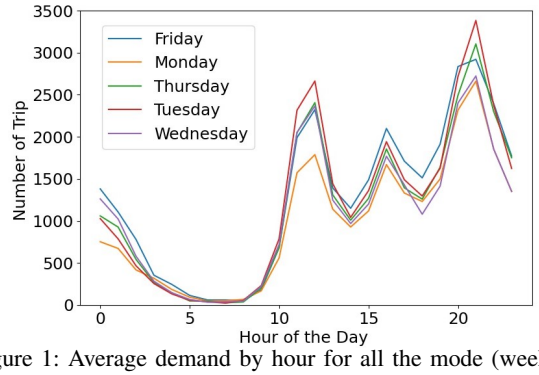


Figure 1: Average demand by hour for all the mode (weekdays)

Based on figure 1, we can now determine two time slots per day where the network is fully congested. During a weekday, we can see two main peaks between 10-13 [hr] and 19-22 [hr]. So, those are chosen to represent the rush hour during the week. Regarding the weekend[2], there are no relevant peaks. We can add that this feature is relevant because some modes of transportation are highly influenced by the congestion (e.g: car or public transportation). Basically, we will be less tented to take a car during peak hour. Finally, we add other features as well that will help our model to learn our dataset. All features and their type are summed up in the table below.

Temperature	continuous
Period Day	one-hot
Rush Hour	one-hot

Duration	continuous
Hour	discrete
Purpose	one-hot vector
Weekend	one-hot

[1] https://donnees.montreal.ca/ville-de-montreal/mtl-trajet.

[2] You may find the same figure for the weekend in the folder Figures

The feature temperature is extracted from an other dataset [3]. Period Day is equal to 1 if the trip is happening during daytime (8-18[hr]), 0 otherwise. The same process is applied for weekend and rush hour features. Regarding the purpose input, the nine different purposes are encoded in a one-hot vector of size nine.

C. Data cleaning[4]

This data cleaning aims to first define the classes that we are going to use for the neural network, then remove outliers and inconsistent points from the data set.

The output is a sequence of 0's and 1 (1 if the travel mode i is used 0 otherwise: one-hot encoding) and the goal is to determine which sequences are relevant and should be kept as a class. One may notice that some travel modes are irrelevant. For example, the travel mode "Taxi/Velo" is not a relevant mode of travel. All the irrelevant travel modes found and the way to manage them are summed up in the Jupyter notebook[4]. After applying this filtration, we end up with the four following classes.

Class	Number of Data points
Velo	14 372
Transport Collectif	19 190
Moto - Voiture	24 744
A pied	11 071

Table I: Distribution of the initial classes

Hence the output of the model will be one of these four classes. As expected, the classes are clearly not balanced, this would be a problem for the training of the model. Now regarding the features, one can notice that some data points are inconsistent. It is possible to find in the data set two points (x_1, y_1) and (x_2, y_2) (where x represents the input and y the class/label) such that $x_1 = x_2$ and $y_1 \neq y_2$. Since the model we are going to develop is a deterministic model, one cannot afford to keep these inconsistent points in the data set. The last task to achieve is to remove outliers for each feature[5]. To achieve this, the Interquartile range method (IQR) was used. This method is applied to each class separately. If we denote by $Q1$ the first quartile and by $Q3$ the third quartile then the decision rule is:

$$x \text{ is a potential outlier if } x \notin [Q1 - 2IQR, Q3 + 2IQR]$$

Where $IQR = Q3 - Q1$. After considering those result, we can wonder how to manage with these outliers. One may replace them by the mean value or simply remove them. In our case, replacing them by the mean value would allow us to keep these data points without changing the original mean. However, it does not make sense from an engineer's point of view. Since there was only a few number[6] of outliers, they were removed from the data set.

[3] https://climat.meteo.gc.ca/historical_data/search_historic_data_f.html

[4] Jupyter notebook: clean_for_nn.ipynb

[5] Only a few features may have outliers. Indeed only the time and the temperature may have outliers because they are results of measurements. The other features are just one-hot encoded, specifying if something is used or not.

[6] here outliers represent 5% of the data set

IV. Neural Network structure

To sum up the pre-processing of the data, we have now 15 features considered as inputs of the model. The purpose of the trip, the rush hour, the period of the day, the weekend affiliation, and the DateTime are one-hot encoded, while the duration of the trip, the temperature, and the hour are continuous/discrete features.

We also have one label column containing the four main classes squeezed into one vector by encoding as integer[7] the four classes from 0 to 3. This will be used in the model as the target, which can also be called the true label. The output of the model will a vector of size four containing the probability to belongs to each class. The prediction will be the class affiliated with the highest probability.

A. Fixed parameters of the neural network

Here we will describe the fixed parameters of the neural network we developed, an other section will follow where the hyperparameters of the network will be tackled. The neural network consisted of three hidden layers, the size of the input layer is equal to 15 and the size of the output layer is equal to 4 (a vector containing the probability to belongs to each class). The number of neurons per hidden layer is a hyperparameter.

Activation and Loss functions

The activation function used for all the layers (except for the last layer) is the Rectified Linear Unit function. This activation function was chosen to avoid the vanishing gradient issue. For the last layer, we used the softmax function to convert the output into probabilities. Given these probabilities, the cross-entropy loss was chosen as the loss function, since it is the most suitable loss function for multi-class classification. The softmax and the cross-entropy loss functions are recalled below:

$$\forall i \in \{1, ..., C\} \quad p_i = \text{softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{C} e^{y_j}}$$

$$\text{Cross-entropy loss:} \quad \mathcal{L}(y_{true}, p) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{C} y^k_{true_i} log(p^k_i)$$

Where N is the number of data points, C the number of classes, and y the vector of the output layer. The expression of the cross-entropy may slightly change if one wants to deal with the unbalanced classes issue. It is possible to weight the contribution of each class to the loss as follows:

$$\mathcal{L}(y_{true}, p) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{C} \frac{\beta_i}{\sum_{i=1}^{C} \beta_i} y^k_{true_i} log(p^k_i)$$

$$\text{Where} \quad \forall i \in \{1, ..., C\} \quad \beta_i = 1 - \frac{N_i}{N}$$

Where N_i represents the number of data points belonging to the class i. β_i increases as the number of data points for class i decreases. Hence points from classes with fewer data points will have a higher impact on the loss: this will counterbalance the fact that there are fewer data points in these classes. More information is provided in the article [4].

[7] The right way to do is to one-hot encode the four classes on a vector of size four, however, the choice to use an integer instead was made to fit with the expected input in the cross-entropy loss function in PyTorch

B. Hyperparameters

The goal is to find the hyperparameters which improve the predictions of the neural network.

Number of neurons per hidden layer

It is one of the key hyperparameter of the neural network. The number of neurons will varies between 10 and 100. The cross-validation will be used to determine which is the best one[8], the parameters of this cross-validation are described in section IV-D.

Regularization parameters

One of the project's goals is to analyze how regularization methods may improve the results. Two regularization options were added to the neural network: L2 and dropout regularization[9]. Given the number of neurons per hidden layers, we developed algorithms[10] which will find the best[8] parameter for each regularization method. These algorithms perform grid search to find the best one. The parameters of the grid search for each regularization method are summed up in the table below. The value 0 is also tested for L2 regularization

	lower bound	upper bound	points	scale
L2	10^{-7}	10^{-3}	5	log
Dropout	0	0.6	5	linear

C. Neural Network Training

Once the neural network is defined, the last thing to do is to train it. The parameters chosen for the training are summed up in the table below.

Optimizer	learning rate	batch size	epochs
Adam	10^{-2}	50	250

Regarding the optimizer, one may use the classical stochastic gradient descent, nevertheless PyTorch is providing more efficient optimizers such as Adam which is able to adapt the learning rate during the training. In this context, Adam optimizer was chosen. Mini-batch is typically chosen between 1 and a few hundred [5].

D. Metrics

To assess the results of the neural network two metrics are used: cross-validation and confusion matrices.

Cross-validation

Cross-validation is here used to estimate the true accuracy of the neural network for different numbers of neurons per hidden layer. It will return the mean and the standard deviation of the accuracy computed on the k folders. Cross-validation may also be used to find the best parameter for regularization. The number of folders k for cross-validation was set to 4 (ratio 75% for the training set): it's a good balance between computational cost and true accuracy estimation.

Confusion matrices

Confusion matrices are useful to analyze the sensitivity of the model regarding each class. Since classes are unbalanced in the data set, it will be a great indicator of how well the neural network handles this issue.

[8] best parameter means the parameter which will maximize the accuracy on the test set

[9] These were selected since they are the most popular regularization methods

[10] Algorithms are named **find_best_lambda** and **find_best_dropout**

V. Results

We will train our different models on two different data sets: the original data set where the four classes are unbalanced, and a subset of this data set where classes are balanced. One generate this subset as follow: if N_{min} denotes the number of data points of the class with less points, then the subset is generated by selecting randomly N_{min} data points in each class. This balanced data set was created in order to evaluate first the behavior of the model on the original data set, then on the balanced one. Only half of each data set was used to get the following results. Two reasons support that choice: the computational cost[11], and because it is sufficient to draw relevant conclusions.

A. Unbalanced Dataset (Original one)

Baseline Results

To get these baseline results, a simple neural network (without regularization) with 3 hidden layers and 10 neurons per hidden layer is trained on the unbalanced data set. Hence the weighted cross-entropy is used. The cross-validation returns the mean[12] accuracy computed on the test and the train sets.

Unbalanced	Train	Test
Accurancy [%]	48.7	47.2

The goal is to first try to overfit the data by increasing the number of neurons per layer and then apply regularization methods to improve the performance of the test set.

Expectations

If we plot the accuracy according to the number of neurons per layer, one should expect that the accuracy on the train set increases as the number of neurons per layer increases. One should also expect a bell shape for the test set without regularization, and a curve that is at least above[13] the test set curve when regularization is introduced.

Validation Result

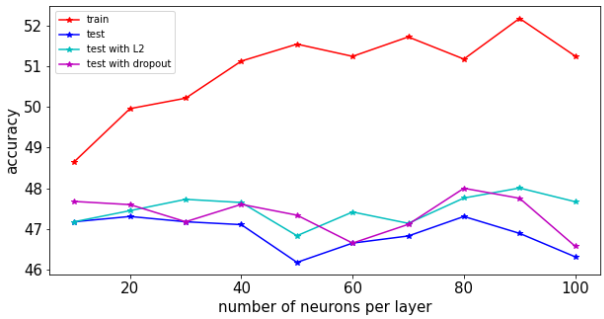


Figure 2: Comparison on the unbalanced data set

In Figure (2), the results of the cross-validation are plotted for 10 different values of the number of neurons per layer, going from 10 to 100. If we add regularization, then before performing cross-validation the algorithms detailed in section

[11] Actually use the whole data set will at least multiply by 2 the computational cost. It lasts 3 days with half of the data set on our computers

[12] Actually it returns also the standard deviation and box plot are available for each curves plotted on the figures (2) and (4). These box plots are stored in the folder Figures

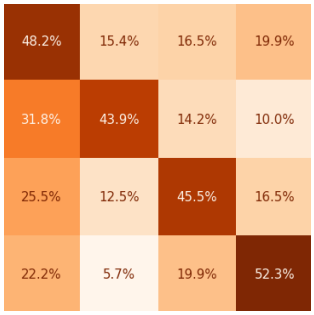
[13] Because one may just take lambda or dropout equal to 0.

IV-B are run to determine the best regularization parameters for each value of the number of neurons per hidden layer[14].

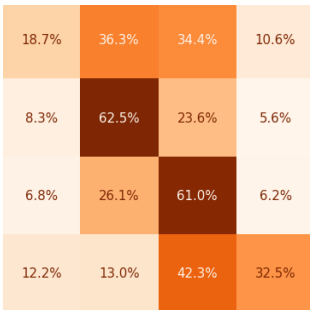
The accuracy on the train set increases while the bell curve does not appear for the test set: it means that the model is already overfitting the data above 10 neurons per layer. Adding regularization improves the accuracy by at most 2%. However, figure 2 does not give any information about the sensitivity of the neural network regarding the classes. Confusion matrices are suited to analyze this.

Unbalanced classes issue

Two confusions matrices[15] are plotted below, they were computed on the train set using a neural network with 30 neurons per hidden layer. On the right, the data set is balanced while on the left it is not.



(a) Balanced set



(b) Unbalanced set

The relevant information of the confusion matrices is stored in their shapes. If it is diagonal then all classes are well predicted. Otherwise, it means that the model is sensitive to one or several classes. As we can see, on the unbalanced data set two classes are well predicted while the predictions for the others are inaccurate. This comes from the fact that the classes "Voiture/Moto" and "Transport Collectif" have a lot more data points than the two others (see the table I in the section III-C). Hence the output of the neural network is, in most of the case, one of the classes with the highest number of points. As those classes are more represented, the probability to predict a true output will increase as well as the accuracy of the class. However, if we want our model to be unbiased and reflect the true accuracy, we should have the same probability to find true outputs in all of the classes. In conclusion, training on an unbalanced dataset (with this configuration) leads to a biased model, so even if the accuracy of the model trained on the balanced set is lower it will still be more reliable.

B. Balanced data set

Baseline Results

Once again, we will use a simple neural network (without regularization) with 3 hidden layers and 10 neurons per layer. The results of the cross-validation are summed up in the next table.

[14]Because of the computational cost these parameters are determined using only one folder over the four of the cross-validation, which actually gives a good hint. The right way to do this would be to determine the best parameter using the four folders. If you have enough computational power feel free to use that method.

[15]On the x-axis you have predicted labels, while on the y axis you have true labels. Labels from the right to the left: "Velo", "Transport Collectif", "Voiture-Moto", "A pied"

Balanced	Train	Test
Accurancy [%]	44.8	41.4

Expectations

The expectations are similar to the ones described previously in section V-A.

Validation Result

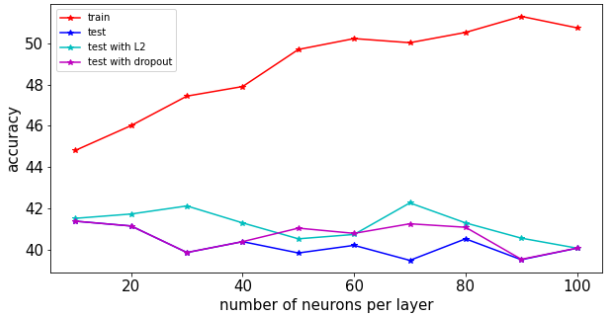


Figure 4: Comparison on the balanced data set

The results of cross-validation on the balanced data set are plotted in the figure (4). Shapes of the curve are similar to the curves of the figure (2). One may notice that the accuracy is slightly inferior to the one computed on the unbalanced data set. The reasons why were tackled in the previous section.

C. Effect of regularization

One of the main goals of this project is to analyze if the regularization improves the results. As we noticed before dropout and L2 improves the mean accuracy on the test set by at most 2%, so technically results are better with regularization. Nevertheless, one may expect more from these methods. One may expect that the accuracy obtained with regularization grows up as the number of neurons per layer increases. Actually, this behavior questions the separability of the data, in short we can even wonder if is it even possible for a neural network to separate the data into four classes.

VI. Discussion

The way of how a computer manages decision making can be very different from humans. When we use Machine Learning algorithms for this purpose it can become a little tricky. Nevertheless, it's worth the effort. Indeed if we can determine the mode of transportation from the purpose, it can help multimodal traffic managers to deal with the demand in public transportation with less money put into surveys. For example, we can think of increasing the number of bus lines going into a commercial zone of the city if we predict that a lot of people were using public transportation to go shopping. Another way to look at our result is that our features may not be relevant enough for our data so the information that our model can learn is limited. We can thus have better results by adding other relevant features as speed or distance. Note that some of the features could be relevant but would also completely change the goal of our prediction model. It depends on the timeline. If we want to be able to predict trips before it has been made, adding speed or distances as a feature can be problematic. For further research, we advise to collect more features and use the Principal component analysis (PCA) to extract the most relevant features.

REFERENCES

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[2] W. McKinney, "Data structures for statistical computing in python," *Proceedings of the 9th Python in Science Conference*, pp. 51–56, 2010.

[3] W. Falcon, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[4] R. Asokan, "Neural networks intuitions: 1.balanced cross entropy," 2019.

[5] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *arXiv:1206.5533*, p. 33, 2012.