1 Introduction

One of the recurring trends that can be observed today, is the active participation of single consumers in the financial markets [?]. As such a clear development to support this in the banking sector has been created, where consumers are offered stocks, portfolios and etc tailored to their needs. This can be often lacking as individual preferences are not always accounted for.

A metric which can actually be use with a little more certainty to create suggestions tailored to the individual needs of the consumer, is the willingness to take risk of each individual. As the risk of stocks and portfolio options can be calculated based on the expected returns, one could use this metric to asses which financial product would be appropriate for each individual.

The project described here, was created with the purpose of allocating financial products of banks to their customers based on 4 preference questions. The questions should provide in the end a metric that would reflect the willingness to take risk for an individual

Calculating financial risk though, is a very complicated manner. By recognizing that this problem is easy to solve although for an individual case, one might think that his is a problem of classifying individual in certain risk-category groups. The ability to provide classification from a variety of metrics, for a large number of data sets is a main feature exhibited by artificial neural networks.

For this reason, the project described in this paper will try to solve following problem: *Which metrics and how should the metrics be used in order to associate individual willingness to take risk, with the risk of stock options*

2 Theoretical Background

This section will provide a brief introduction into the main theoretical premises utilized in implementing this project. It is structured in the following way:

1. The first subsection will provide a brief introduction into the definition of risk for financial products

2. The second subsection will provide a brief introduction to neural networks and their functions

3. Subsequently the application between the two will be explained

After reading this Section, the reader will be able to grasp the premise of this work.

2.1 Financial Risk

The measurement and control of risk is a major topic even today [?]. [?] states that financial risk is associated with the statistical uncertainty of the final outcome, and is measured by its volatility. Humans will indulge in financial risk at different levels, in order to arrive at financial gains. Some will indulge in highly risky affairs which yield higher rewards, while others will prefer less risk at the downside of smaller returns on their financial investment.

In the case of this research paper, financial risk will be defined as the risk undertaken when a bank customer chooses to invests in a stock option, a portfolio or bond options with the ultimate goal of expanding his monetary gain. The financial risk undertaken therefore, is the loss of the investment in case of risk materialisation [?]. The paper will not consider financial risk per se, rather it will consider itself with categories and the volatility of the stock options given by the bank.

2.2 Neural Networks

Neural networks and deep learning are in essence machine learning algorithms, where deep learning is a more specific type of machine learning [?]. For the purpose of this paper, the definition presented below is derived from [?]. A machine learning algorithm foremost wants to learn something, anything that can be learned from a set of data, therefore a learning algorithm is employed.

[?] defines the learning part of a learning algorithm as ‘A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at task in T, as measured by P, improves with experience E.‘. The task T, is more accurately a task, that is difficult to solve with a fixed program, traditional program, and it is not to be confused with the process of learning, since the task itself is the problem that needs to be solved.

2.2.1 Network Types

The learning process is based on how a machine learning process will process an **example**, that is composed of a collection of **features**, where a vector $xinR^n$ and each x_i of the vector is just another feature of the example. An example of a feature are the values of the individual pixels in an image. Some of the more common tasks that can be solved by machine learning are as follows:

- Classification: The task here is to specify in which category k a certain input belongs to. These type of algorithms are used for example for classifying images to objects. A function $f(x)$is computed based on the data learned.

- Regression: The task here is to predict a numerical value given some input. The function $f(x)$ that is calculated is similar to the one of the classification algorithm with the exception that the output format differs. A simple example of this is the prediction of the claim amount an insured person will make, or the future price of securities.

- Transcription: The task in this instance, is to observe an unstructured representation of some kind of data, and try to transcribe it in textual form. A simple example of this is the transcription of a handwritten image to text.

- Machine translation: In this task the computer tries to translate one language to another, the most common application is the translation of two natural languages e.g. English to French. Here deep learning is quite important.
- Structured output: Any tasks, where the output is a vector, that contains important relationships between the different elements. An example is the transcription above, or parsing of natural languages in trees that describe grammatical structures.
- Anomaly detection: The program will flag data that is not conforming to the usually observed data as an anomaly. Usage is e.g. Credit card fraud, or in anti-virus software.
- Synthesis and sampling: The task is to generate new examples, based on the already existing data. This is interesting as for example the program might be tasked to create a movie script, or create large volumes of music.
- Imputation of missing values: The task is to be able to predict missing values in datasets
- Denoising: A corrupted example is given which was corrupted by an unknown process, and the program hast to predict a clean example through another clean example x
- Density estimation or probability mass function estimation: Advanced use-case where the machine learns a probability density function

2.2.2 Performance Measure *P*

The performance measure evaluates, quantitatively how good the network performs for a specific task. As en example in the case of classification the measure would be the accuracy of the network in predicting correct results. Usually the performance is measured based on the error rate, and to evaluate the performance after the reaching a satisfying error rate, the performance is measured based on a test-set of data that the network never saw before.

2.2.3 The Experience, *E*

Machine learning algorithms, as illustrated above can be split in either **supervised** or **unsupervised** learning algorithms. The experience discussed here is the is the **dataset** which is chosen as an input for the algorithm. In an **unsupervised** learning algorithm, the algorithm experiences the whole dataset and tries to extract useful properties out of it without labeling by a user, for example in a denoising network. The formula for

In a **supervised** learning algorithm, the algorithm experiences the dataset which is associated to a specific **label** or **target**. This is generally used as part of classification algorithms, such as when classifying plants based on the features of their iris. The formula for supervised learning is usually (book 105)

A clear distinction between the two has still to be defined formally, yet this is above the scope of this paper.

2.2.4 Overfitting, underfitting and capacity

One of the most important aspects in neural networks is **overfitting** and **underfitting**. These two terms generally mean that due to outliers in the data the generated model either has a large gap between the training error and the test error, in the case of overfitting, or the training set cannot attain a low error value, as is the case when underfitting. These values can be controlled through an attribute called **capacity**. Capacity generally expresses a wide variety of functions, which will not be explained here, but one example would be to standardize the input data.

3 Literature Review

This section will be reviewing available literature that focuses on creating neural networks to asses risk in different settings. Some of the literature reviewed here does not directly apply to the financial sector, yet yields useful information in what kind of methodology should be applied in order to create the implementation detailed further down in this paper.

3.1 Determinants of Risk

[?] researches the question if it is possible at all possible to investigate individual risk attitudes using a survey. In the paper [?], try to asses the viability of using a hypothetical lottery, and risk self-assessment when measuring risk-attitudes of individuals. Their research is based on [?], and therefore their questions are modelled in a similar fashion. The researchers findings indicate that hypothetical lotteries and self-scaling questions are generally good indicators to measures someones risk attitude. The author though argue, that the heterogeneity of the results point out that it might be better to use more context specific questions based on the problem at hand. The results of this study influenced the decisions taken in Section 4 while creating the survey for this paper.

A wide array of research exists in the social, financial and economical studies on the determinants of risk in individuals and in groupings. For the purpose of this paper, a number of those which address financial risk directly were chosen. [?] states that it aimed to examine risk attitudes based on a large representative survey provided by SOEP. The study was then complemented by a smaller field study which examined the general willingness of an individual to take risk, based on the findings of the previous examination. The authors then expand their findings in order to correlate the general risk with more specific risk types such as smoking, financial risk and etc. In the findings the group concludes that indeed gender, age, height and the parental background correlate with risk attitude an individual displays. In addition to these determinants, choices such as investment in stocks and doing sports have been found to contribute to the willingness of an individual to take risk. [?] used a financial lottery in order to verify these results in the field experiment, this approach will be used by this paper in Section 4.

3.2 Financial risk with NNs

3.3 Various risk with NNs

4 Methodology

As detailed in the relevant literature, the methodology which is appropriate neural network design for the problem detailed in the introduction, is that of a multi-layered perceptron, using logistical regression at its output layer. As detailed in Section 2.2 these types of networks are appropriate for classification tasks. The design of the network to be implemented can be visualised in Figure **??**.

To determine the relevant setup a questionnaire was developed based on the literature reviewed in **??**. The questionnaire is included as part of this paper in the appendix. The first assumption that is going to be tested is if the network can asses through the first three questions posed, the outcome of the fourth question. If the fourth question can be guessed correctly then we could equally assume that with the inclusion of the fourth question as an input the outcome will be even more correct.

The purpose of the fourth question is to classify the persons answering the questionnaire into three distinct classes. One who is risk-averse, one who is of medium risk-aversity/propensity and a class who is more risk-prone than the other ones. The purpose of this, as detailed in Section1 is to be able to determine the appropriate investment packages that should be recommended to a client of a bank.

In order to be able to handle a small dataset, cross-validation is being used. At the network proceeds to plot the MSE of the validation and test loss with respect to the epochs run, which were detailed in Section 2.2. The prediction function will then be tested again on a small subset of the test set in order to be able to also visualize its accuracy.

5 Implementation Details

5.1 Network Design

The chosen language for the implementation is python. This is due to the wide use in neural networks, and in general data-science which is enjoyed by python. As we needed a network with adequate performance the Theano framework was chosen in order to implement the different neural network functionalities. Theano also did provide most of the activation functions by itself, the only exception of this was the Gaussian activation function which was implemented by hand. Since Theano is quite a complex framework the implementation chosen for the task at hand was a simple feed-forward network with backpropagation as depicted in Figure **??**

5.2 Input Pre-processing

Since there can be different types of data, an additional splitting algorithm is employed which will pre-process the dataset in question, in order to first standardize the data, then apply normalization where appropriate and later on split the dataset into the appropriate training, validation and testing sets. Data standardization is needed in order to obtain sane values for binary and categorical datasets. To accomplish this task the **pandas** framework is used in order to do the standardization.

Following this, pandas is again employed in order to normalize all the columns of the dataset. Normalization is needed in order to restrict the data between the values of minimum and maximum values that it can acquire and is employed in statistics in order to reduce outliers. In our case the normalization is crucial in order to also control overfitting and underfitting of the weights and the bias. While standardization is always a normalization, normalization itself is not a standardization. Normalization is used here to eliminate influences of very large values and very small values. At the end of the splitting algorithm, the program will output the shape of the data in order for the user to know how many features the dataset will contain. Depending if the label data is categorical or numerical, the algorithm will also take care to take this under consideration when outputting the amount of features the dataset will have.

5.3 Program Usage

The main scale of the program is located in the file *run.py.* The file can be called using *python run.py* with following arguments:

- -d or –demo: runs the demo function on the MNIST dataset
- -pd or –prediction-demo: runs the MNIST prediction demo
- -j or –json: expects a JSON as input with the task to solve
- -s or –split: expects a CSV as input and then splits the file into appropriate training, validation and test set
- -v or –version: displays the program version

5.3.1 The Demo

The Demo consist of predicting the MNIST Dataset provided from [**?**]. It is used as part of the Tutorial in [**?**], and thus is used to demonstrate and test the custom build network. The Demo loads the MNIST Dataset from a python pickle file and then runs the network. The prediction part of the demo is run automatically when the prefix *-d* is used. Alternatively if someone wants to just test how the predictor works, he can just activate the *-pd* option.

The algorithm should run in a similar fashion as explained by [**?**]. The program is modelled after the algorithms introduced there albeit with a lot of modified parts in order to account for our datasets.

5.3.2 JSON-Tasks

The configuration of the tasks that should run in the neural network is given using a json file which is structured in the following way:

```
{
    "tasks":[
            {
                //Here Task Content
            },...
            {
                //Here Task Content
            }

    ]
}
```

There are 2 key settings in the JSON file, which determine the overall contents of a single task entry. The variable in question is also called setting, and can be set to either train or predict. A Task then has following structure depending on the setting:

- If the setting is set to **train** then following settings apply:
  - *learning_rate*: This specifies the learning rate of the network.
  - *L1_reg*: Used for rate regularization.
  - *L2_reg*: Same as above.
  - *training_epochs*: How many epochs should the network maximally train
  - *datasets*: A collection of datasets that the network should calculate
  - *n_ins*: The amount of input nodes present in the network
  - *n_outs*: The amount of output nodes present in the network
  - *batch_size*: How many batches should be used in each epoch
  - *hidden_layers_sizes*: Specifies the size of the hidden layers. If given as list will produce multiple hidden layers
  - *logfile*: A file which would log the output of the training
  - *activation*: The activation function to be used, currently supports: $tanh(x)$,$relu(x)$,$gausian(x)$,$sigmoid(x)$,$hard\_sigmoid(x)$

  - *delimiter*: Specifies the delimiter to be used in the logfile and the delimiter which is present in the dataset input file

- If the setting is set to **predict**:
  - *best_model*: The pre-calculated model for the neural network
  - *datasets*: The dataset which was used in the training
  - *n_ins*: Same as above
  - *n_outs*: Same as above
  - *hidden_layers_size*: Same as above
  - *result_file*: File to store prediction results
  - *delimiter*: Same as above
  - *activation*: Be sure to use the same activation function used to generate the model

In addition the program can also be run using the GPU, though currently due to the lack of a testable computer this function is still not implemented, as Theano only fully supports NVIDIA-CUDA.

5.3.3 The Dataset

The dataset should be provided as a single CSV. Then the user has to split the dataset into a training set, a validation set and a testing set. To simplify this process for the user, the neural network implements the -s argument which calls the CSV splitter, discussed as part of Section **??**. The splitter takes the file and splits it into an 80/20 training/test file, it then proceeds to apply a similar 80/20 split to the training file in order to provide a fair validation set. In the JSON file which serves as an input to the neural network, the user has to mainly write the name of the original CSV file, the program takes care of it itself.

5.3.4 Activation Functions

6 Future Work