How can Machine learning be used in different ways to predict bipolar states?

Joakim I. Frogner

Thesis submitted for the degree of
Master in Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

How can Machine learning be used in different ways to predict bipolar states?

Joakim I. Frogner

Abstract

Contents

List of Figures

List of Tables

Preface

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

Statistics
- Data shows that 5,890,000 adults are diagnosed with bipolar disorder in the USA (2,65% of the adult population) [find better source].

Ways to use the results of this study?

1.2 Thesis overview

[Fill in later]

Chapter 2

Background

2.1 Bipolar disorder

Bipolar disorder is the disorder where you experience extreme mood swings. One day you can feel amazing and everything is fine, but the next day you feel like you don't belong anywhere in this universe. Mood swings in general is not something that you should be concerned about. It is however the extreme cases where your mind turns 180 degrees from day to day that is the main symptom of bipolar disorder. There is not really a specific type of people that get this; they can be of any age and any gender, but most people that suffer from it find out (by having an experience or episode) around age 25 [3].

When talking about bipolar disorder, we often separate between the states *normal*, *mania* and *depression*. The last two are the states we usually talk about, since a normal state isn't that interesting. These two states are very different, but they have some similarities, for example sleeping problems.

When a bipolar person is in a manic state, he/she may do things that they never would have intended doing, like spending a lot of money on items they really don't need, or abusing drugs/alcohol. They may also feel really excited or powerful [2].

A bipolar patient is in a depressive state when he or she is in a bad mood swing. They can stop doing everything they usually like to do, and lie down in bed all day with no motivation to do anything useful. They may feel useless and that they don't belong here, or being guilty of something they may or may not have done. In some cases, a depression may even end up with suicidality, where the person either just thinks of death, or actually attempt suicide (actually 20% of people diagnosed with bipolarity commit suicide [3]).

The frequency of these symptoms can vary. One year they can have these mood swings every day for several weeks at the time, and the next they get them less frequent, like once every month.

We also separate between bipolar disorder type I and II, with the main difference being that the manic episodes are way more aggressive in type I [1].

Statistics say that bipolarity is genetically inheritable, with 23% chance of getting a child with bipolar disorder if one parent is bipolar, and 66% if both parents are [3].

2.2 Machine learning

Machine learning is the field of computer science where you basically throw a lot of data into an algorithm and expect it to give you answers to whatever you prefer, with as little work as possible. This was not the case in the early days of the technology, but nowadays it is a lot easier with all the diffent frameworks and tools available.

Machine learning is a great and almost 'magical' technology, but only if you do it right. First you need to have enough data to feed into the algorithm, and to be efficient when training the model on a large dataset, which you need to be if you want your result quickly, you need good hardware. You can get away with a decent CPU if you just want to test it out on a small dataset, but if you really want to do machine learning, then you need a good GPU. The reason why GPUs are so much better than CPUs on this specific task, is because the CPUs are designed for flexibility and general computing workloads. The GPUs on the other hand, are designed to do the same instructions over and over again in parallel. This makes GPUs a lot more efficient for machine learning, and especially for deep neural networks [16].

Now how do you do the actual machine learning? Well there are many diffent approaches to this, which I will discuss in the next sections, but say you want to use a neural network for your task. Then your next step should be to choose a framework. You can of course do everything yourself, but why reinvent the wheel when there are so many good frameworks and tools already out there?

The programming language **Python** is great for machine learning in my (and many other peoples) opinion. It is structured in a way such that it really looks like pseudo-code, and this is perfect because we don't want to spend time on weird syntax rules in another language. For Python, you have a popular framework called **TensorFlow** which is developed by Google. This allows you to build models easily, and also execute the training and testing. Before you get started with TensorFlow, do a quick google search to see if someone else has already done something similiar to what you are trying to acheive, and if you find something, odds are that your neural network model can be similar, if not identical to it. If not, then you have to sit down and actually make the model yourself.

For the model implementation part, whether you found a model online or want to build it yourself, you can of course do it in TensorFlow, but there is an easier way. **Keras** is also a popular framework that is most commonly used together with TensorFlow. On their documentation website [8], they see their framework as 'A high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK or Teano'.

Follwing their '30 seconds to Keras' guide [8], you can create a 'sequential' model with 'dense' layers, configure its learning process (compile), then fit, train, evaluate and predict with just a few lines of code:

Source Code 2.1: 30 Seconds to Keras

```python
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5, batch_size=32)
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
classes = model.predict(x_test, batch_size=128)
```

So, as long as you know your theory, and can decide which model to use (and either find a good implementation of that model or create it yourself), you can easily do machine learning. One important task you have to do, is to make the dataset ready. This is the boring and tedious part of machine learning, but it has to be done in order for making it possible to train the model on it.

2.3 Machine learning strategies

Picking the right machine learning model can be quite difficult, especially if you don't have any experience from earlier. There are a couple of diffent *strategies* you can choose from when deciding on a machine learning model. These are called *Supervised and unupervised learning*, and you need to look at your dataset and how it is structured to find out which one to use. The following sections will be a description of the strategies, to make the decision easier.

2.3.1 Supervised learning

This is the machine learning strategy where both input and desired output data are provided [15]. You can use this if you want to train a model to classify letters in the alphabet, or anything else where you have a dataset with both input and output data (for the alphabet, images of letters are input data and the actual letters are the output data). If you train this alphabet model, you will be able to input a competely new image of a letter, and the model will classify it to the letter it most likely fits. This kind of supervised learning is called *Classification*, and is *the problem of assigning new observations to the class they most likely belong, based on a classification model built from labeled training data* [11].

Another kind of supervised learning is called *Regression*, and is all about predicting (or estimating) a value. A classic example for regression learning is predicting income, using *features* like home location, job title, field of study, years of education and years of experience. We call these features *categorical* (first three) and *numerical* (last two) [10].

2.3.2 Unsupervised learning

Another strategy is *Unsupervised learning*. This is what you want to use if you have a dataset without the same meaning as in a dataset for supervised learning. The items may not have a fixed answer, like the letters in the alphabet are. It is useful when you have *unlabeled* data, and want to for instance group data together in what we call a *cluster*. It may not be as commonly used as supervised learning, but unsupervised learning can also be very useful in some cases; like grouping addresses together in neighborhoods if you have a unsorted list of addresses as a dataset.

2.3.3 Semi-supervised learning

Now, you may not always want to use one of the strategies above. Looking at your dataset you may want something in between; a combination of labeled and unlabeled data. This is when semi-supervised learning comes in handy. For example if you have a lot of data to give labels to in your dataset, it can be simply too much work. We won't go deep into details about how this works, but it is important to mention it because of its usefulness.

2.4 Machine learning approaches

When you know whether you want to use supervised learning, unsupervised learning or something in between, you need to select an approach. We call them approaches because you use these regardless of the strategy you end up using; most of them (with the exception of *reinforcement learning*, which we will come back to) work in both supervised and unsupervised learning. There are a lot of diffent approaches available, and we will describe some of them, namely those we will use in the main parts of this thesis.

2.4.1 Decision tree learning

In computer science, trees are data structures commonly used to describe something that *branches out* based on different input. For example a tree can be a representation of how the frequency of letters in the alphabet are distributed in a text file, so that the text file can be compressed optimally.

Day	Temperature	Outlook	Humidity	Wind	Run
1	15 C	Sun	Low	Strong	Yes
2	6 C	Rain	High	Weak	No
3	15 C	Rain	Medium	Strong	Yes
4	6 C	Overcast	High	Medium	Yes
5	15 C	Sun	Low	Weak	No
6	12 C	Overcast	Medium	Weak	No
7	12 C	Sun	Medium	Medium	Yes

Table 2.1: Training data set: days a person went out for a run

I won't go into details about how this works, but my point is that tree-structures are very common in most fields of computer science.

In machine learning, we can apply the tree-structures as *decision tree learning*. And in this approach, we set up all the different outcomes (with the training data set) of a specific question in a tree. Let's say you want to predict whether or not a person will run outside on a specific day. Then it makes sense that the training set contains weather information. The different data in the training set is called attributes, and picking these correctly is important for the quality of the prediction.

Table 2.1 contains data about whether a person went outside for a run or not for a week (just an example, not real data). Here the first 4 (excluding "Day") columns (Temperature, Outlook, Humidity and Wind) is the "predictors" and the last column (Run) is the "target". To use this table in decision tree learning, we need to view it as a tree, with one of the predictors as root node and the targets as leaf nodes.

How we choose the tree structure is critical to the performance of the machine learning, and we need to use a good tree building algorithm. The most common algorithm to use in this situation, is the *ID3* algorithm made by J. R. Quinlan. It is a top-down, greedy search through the space of possible branches with no backtracking [5]. The way this happens is by calculating *Entropy* and *Information Gain*. The idea is to recursively choose the predictor that has the highest information gain and generate a tree structure. With an optimal tree, you can create decision rules by simply following the tree with new data.

Random Forest

One known problem with decision tree models, is that they often include a lot of *Variance*. This means that an algorithm is sensitive to small changes in the training set. One method to reduce the variance, is to use Random Forest.

Random Forest is a supervised machine learning strategy, which can be used for both classification and regression learning [14]. It essentially works by combining decision trees, where the tree building algorithm is heavily randomized for all trees. For example, if you want to get movie recommendations using machine learning, using one decision tree will most likely be insufficient. Just think what happens when you ask one friend for movies to watch. What that friend recommends is purely based on movies you like and what he has already watched. You might be lucky and find your next favorite movie, but most likely, asking multiple people for recommendations is going to give a better result. The same goes for machine learning, and decision trees will most likely give a better answer if they are combined in a Random Forest.

2.4.2 Neural networks

General idea

The general idea of machine learning with neural networks, is to make the computer think like a human; it is inspired by they way biological neural networks in the human brain process information [12]. There are a lot of diffent neural networks, but all of them share the same underlying layer based architecture, where data is passed between layers where computation is done. The first layer is the input layer, which simply passes the data to the next layer, which is the hidden layers. The number of hidden layers is competely up to the model and the programmer, and this is where the intermediate processing/computation is done, before the data is passed to the output layer where we perform an activation function to define the output [13].

If you have a lot of hidden layers in a neural network, we call it a *deep* neural network. This is commonly used, and there is a lot of different deep networks with a lot of diffent use cases. Two of the most common neural network models are *Recurrent Neural Networks* and *Convolutional Neural Networks*. These two have their own use cases, which I will describe below.

Figure 2.1 is a visualisation of a basic neural network with the input layer on the left, one hidden layer in the middle and the putput layer on the right hand side. Every node in the input layer is connected with every node in the hidden layer, and every node in the hidden layer is connected to every node in the output layer.

Recurrent Neural Networks (RNN)

This type of neural network is good for predicting something based on a sequence of data, like for example predicting words in a sentence, which can be especially useful for typing on a phone. Also doing predictions based on historical data, like a forecast, is something an RNN can do effectively, which is something the dataset that I'm going to use in this thesis consist of.

One downside to regular Recurrent Neural Networks, is that if the sequence of data is long, the prediction will most likely be off if something that was for example typed in the beginning of a long text is a dependancy for a prediction four chapters later, like the location of the main charater. The workaround for this is something called *Long Short-Term Memory*

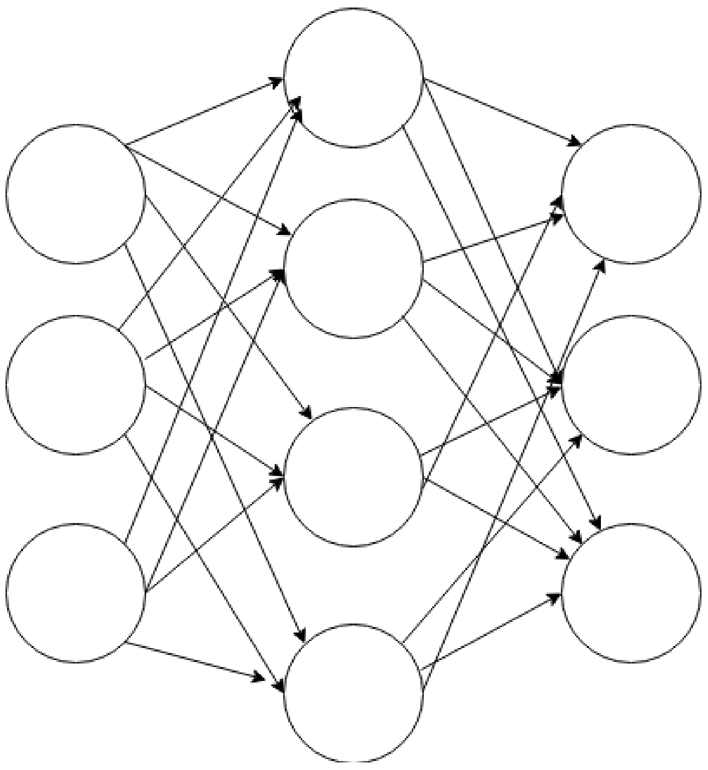


Figure 2.1: Neural network

Recurrent Neural Network (LSTM RNN), and is the idea of having additional logic to avoid the prediction model forgetting important facts.

Convolutional Neural Networks (CNN)

A Convolutional Neural Network, or *CNN* for short, can be used for identifying patterns in data, which can be used for predictions. A common use case for a CNN is image recognition. This is where you train your model to be really good at identifying objects in images, for example the difference between cats and dogs. Then you can input a competely different image to the model, and it will output whether the image is of a cat or a dog. This type of CNN is two-dimentional because an input image is really a two-dimentional array of pixel values, and it is most common to have a 2D CNN. Another way of constructing a CNN, is one-dimentionally, which can be useful for *one-dimentional* data, for example sensor data from gyroscopes or accelerometers [7].

2.5 How can machine learning help people with bipolar disorder?

The usage of machine learning in the medical fields is growing exponentially these days. There are so many use cases of machine learning, and of course it can help in the bipolar field too! Let's say bipolar patients had a device that measured their heart rate among other things 24 hours a day could feed the data into a machine learning model that could give the user live feedback on which bipolar state they are currently in. I think that would be very useful, for both the patients and doctors/nurses. Another use case coudl be if medical institutions could know in advance how many new bipolar patients to expect the next day.

I believe that using machine learning in this field of study could help a lot of people get through their depression or mania, and potentially get rid of the condition competely.

2.6 The dataset

The dataset I will use in this project [6] was collected for another study for motor activity in schizophrenia and major depression. With the data about schizophrenia stripped out, this dataset is sufficient for my thesis. It contains activity level data for 23 bipolar and unipolar patients, and 32 non-depressed contributors. From now on, I will refer to the bipolar/unipolar group as the *condition group*, and the non-depressed group as the *control group*. This is also done in the dataset details [4].

The dataset is in two parts. One part includes sensor data about the condition group and control group, as one file for each person. These files are in two folders: "condition" and "control" (for the two groups respectively), and one file for each person is inside the folders (filename is "GROUP_X.csv" where X is their id and GROUP is either condition or control. Inside the files, there is a list of activity measurements for every minute of the data collection period.

The other part of the dataset is one file including the demographics of each person in the dataset. For the control group this only includes the number of days they were collecting data, their gender (1 for female and 2 for male) and age. For the condition group, it also includes their affliction type (1 for bipolar type II, 2 for unipolar depressive, 3 for bipolar type I), melanch (1 for melancholia, 2 for no melancholia), inpatient (1 for inpatient, 2 for outpatient), edu (education in years), marriage (1 for married / cohabiting, 2 for single), work (1 for working / studying, 2 for unemployed/sick/pension), madrs1 (MADRS score when measurement started) and madrs2 (MADRS score when measurement stopped) [4]. MADRS score (Montgomery-Asberg Depression Rating Scale) is used to grade the current severity of an ongoing depression [4].

2.7 What to do in this project

As stated in my problem statement, I want to perform different types of machine learning on the dataset. There is a couple of things I have in mind that the dataset can be used for:

- Predict whether a given person belongs to the *control group* or the *condition group*.
- Predictions on the patients MADRS score and sleep patterns.

More details on how I will acheive my goal will come in the next part of this thesis.

2.8 Challenges and ethical concerns

In most projects in the medical fields, there are going to be ethical concerns and challenges with privacy. What happens if someone that are not clearified for the data gets access to it? What if the database gets hacked? With new regulations (GDPR), which basically means that users have the right to be "forgotten". However, in this project all data is anonymized (only referenced by an id), so there will be no persons mentioned. If the dataset were not to be anonymized, and the patient's names were in it, things could get problematic if it got into the wrong hands.

Part II

The project

Chapter 3

Planning the project

To predict which group a given person belongs to, we need a good classification model. In order to predict something, we need some label that tells us what is correct. In this case, the filename tells us whether the person is in the *control group* or *condition group*, as described in the previous part.

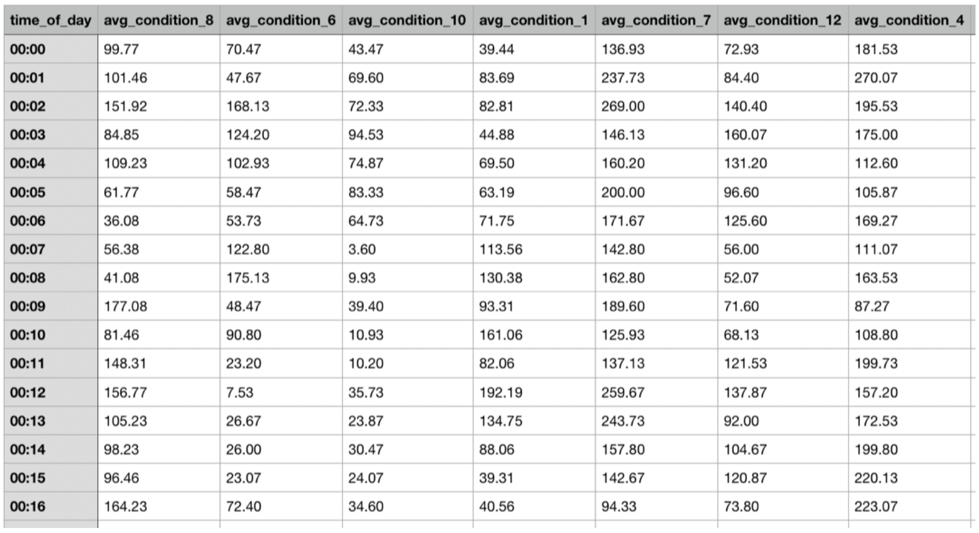
Since the dataset contains activity levels every minute of the recorded days, and the number of days for one participant varied, we were thinking it might be a good idea to calculate the *mean* activity for the participant at each minute every day. This means that we can have a one-dimentional array of recordings for each participant instead of having an inconsistent number of rows.

After structuring the dataset in this way, we ended up with one table where the Y-axis tells the minutes (from 00:00 to 23:59), and the X-axis contains all the participants. The data value at each position is then the mean activity for minute Y and participant X. Table 3.1 is a small part of the dataset structured this way.

Now we just needed a classification model to handle this kind of data, for example a Convolutional Neural Network.

We needed to learn more about CNNs. CNNs are used in image recognition, so we proceeded to implement that. Making it really simple, we wanted to classify images of cats and dogs. We visited a website called https://pythonprogramming.net, which is a good resource for learning almost anything related to programming (using python). There we found a tutorial on 2D CNN for classification between images of cats and dogs [9].

It was both a fun and informative experience implementing this. Especially when we extended the script to allow an image url to predict on. Then we could browse for any image of a cat or a dog, and find out if the model could handle it (in most cases it did!). We even tried inputting images of humans to the model for fun. This experiment gave us a lot of motivation for our task.



time_of_day	avg_condition_8	avg_condition_6	avg_condition_10	avg_condition_1	avg_condition_7	avg_condition_12	avg_condition_4
00:00	99.77	70.47	43.47	39.44	136.93	72.93	181.53
00:01	101.46	47.67	69.60	83.69	237.73	84.40	270.07
00:02	151.92	168.13	72.33	82.81	269.00	140.40	195.53
00:03	84.85	124.20	94.53	44.88	146.13	160.07	175.00
00:04	109.23	102.93	74.87	69.50	160.20	131.20	112.60
00:05	61.77	58.47	83.33	63.19	200.00	96.60	105.87
00:06	36.08	53.73	64.73	71.75	171.67	125.60	169.27
00:07	56.38	122.80	3.60	113.56	142.80	56.00	111.07
00:08	41.08	175.13	9.93	130.38	162.80	52.07	163.53
00:09	177.08	48.47	39.40	93.31	189.60	71.60	87.27
00:10	81.46	90.80	10.93	161.06	125.93	68.13	108.80
00:11	148.31	23.20	10.20	82.06	137.13	121.53	199.73
00:12	156.77	7.53	35.73	192.19	259.67	137.87	157.20
00:13	105.23	26.67	23.87	134.75	243.73	92.00	172.53
00:14	98.23	26.00	30.47	88.06	157.80	104.67	199.80
00:15	96.46	23.07	24.07	39.31	142.67	120.87	220.13
00:16	164.23	72.40	34.60	40.56	94.33	73.80	223.07

Figure 3.1: Dataset structured on mean activity level

However, my data is one-dimentional, and image recognition is two-dimentional. The simple answer then is to use a 1D CNN.

> *A 1D CNN is very effective when you expect to derive interesting features from shorter (fixed-length) segments of the overall data set and where the location of the feature within the segment is not of high relevance. This applies well to the analysis of time sequences of sensor data (such as gyroscope or accelerometer data).* [7]

To learn more about 1D CNNs, we followed a tutorial [7], which used a dataset containing time-sliced accelerometer data from a smartphone on the participants waists. The goal for this CNN is to predict what a given person is doing at the time, given the accelerometer data for that time slice. What the given person is doing is one of the following:

- Standing
- Walking
- Jogging
- Sitting
- Upstairs
- Downstairs

As we followed the tutorial and implemented the model, we learned a lot about how 1D CNNs work, but also where my dataset could provide more data. What if the dataset contained the actual current state of the bipolar patient? Then we could make some automated system that always can tell a patient whether they are normal, manic or depressive. However data collection for this kind of task would be difficult because we can't always know what the patient thinks, nor does the patient themselves. The "tutorial" dataset is different because it is easy to differientiate physical states of the body like standing or walking.

Part III

Conclusion

Chapter 4

Results

Bibliography

[1] *Bipolar 1 Disorder and Bipolar 2 Disorder: What Are the Differences?* Accessed: 14.03.2018. URL: https://www.healthline.com/health/bipolar-disorder/bipolar-1-vs-bipolar-2.

[2] *Bipolar disorder*. Accessed: 14.03.2018. URL: https://familydoctor.org/condition/bipolar-disorder/.

[3] *Bipolar disorder statistics*. Accessed: 14.03.2018. URL: https://www.statisticbrain.com/bipolar-disorder-statistics/.

[4] *Dataset details*. Accessed: 08.05.2018. URL: http://datasets.simula.no/depresjon/.

[5] *Decision Tree*. Accessed: 08.05.2018. URL: http://www.saedsayad.com/decision_tree.htm.

[6] Enrique Garcia-Ceja et al. 'Depresjon: A Motor Activity Database of Depression Episodes in Unipolar and Bipolar Patients'. In: *Proceedings of the 9th ACM on Multimedia Systems Conference*. MMSys'18. Amsterdam, The Netherlands: ACM, 2018. DOI: 10.1145/3204949.3208125. URL: http://doi.acm.org/10.1145/3204949.3208125.

[7] *Introduction to 1D Convolutional Neural Networks*. Accessed: 07.12.2018. URL: https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf.

[8] *Keras documentation*. Accessed: 15.03.2018. URL: https://keras.io.

[9] *Loading in your own data - Deep Learning basics with Python, TensorFlow and Keras p.2*. Accessed: 07.12.2018. URL: https://pythonprogramming.net/loading-custom-data-deep-learning-python-tensorflow-keras/?completed=/introduction-deep-learning-python-tensorflow-keras/.

[10] *Machine Learning for Humans, Part 2.1: Supervised Learning*. Accessed: 11.04.2018. URL: https://medium.com/machine-learning-for-humans/supervised-learning-740383a2feab.

[11] *Machine Learning for Humans, Part 2.2: Supervised Learning II*. Accessed: 11.04.2018. URL: https://medium.com/machine-learning-for-humans/supervised-learning-2-5c1c23f3560d.

[12] *Neural Networks*. Accessed: 21.05.2018. URL: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

[13] *Neural Networks Introduction*. Accessed: 21.05.2018. URL: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc.

[14] *Random forest*. Accessed: 21.05.2018. URL: https://medium.com/Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674.

[15] *What is supervised learning?* Accessed: 11.04.2018. URL: https://searchenterpriseai.techtarget.com/definition/supervised-learning.

[16] *Why are GPUs well-suited to deep learning?* Accessed: 15.03.2018. URL: https://www.quora.com/Why-are-GPUs-well-suited-to-deep-learning.