DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2018

A comparison of machine learning algorithms for automatic classification of neurons by their morphology

JOAKIM LILJA

MARCUS ÖSTLING

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

A comparison of machine learning algorithms for automatic classification of neurons by their morphology

JOAKIM LILJA, MARCUS ÖSTLING

Degree Project in Computer Science
Date: June 6, 2018
Supervisor: Alexander Kozlov
Examiner: Örjan Ekeberg
Swedish title: En jämförelse av maskin inlärningsalgoritmer för automatiserad klassificering av neuroner utifrån deras morfologi
School of Electrical Engineering and Computer Science
KTH - Royal Institute of Technology
Stockholm, Sweden

Abstract

Classification of neurons has been a studied topic in neuroscience for several years and with the increase of data, new methods are encouraged to help with the classification. This study compares different machine learning algorithms to see which are better suited for classification of large data sets of morphological reconstructions in multidimensional feature space. Ten algorithms were compared on a data set of over 10 000 samples of mice neurons. Further, each classifiers ability to classify each available cell type were also investigated. The results show that Random Forest had the best overall mean accuracy followed by Multi-layer Perceptron with 83% and 78% respectively. However, observing the classification of individual cell types all the algorithms varied in accuracy and Random Forest was not considered the best. In conclusion, machine learning algorithms are a viable source when classifying neurons but more research needs to be performed to reach the higher accuracy results.

Sammanfattning

Klassificering av nervceller har varit ett studerat ämne inom neurovetenskap i flera år och med ökningen av data uppmuntras nya metoder att hjälpa med klassificeringen. I denna studie jämförs olika maskininlärningsalgoritmer för att se vilka som passar bättre för klassificering av stora datamängder av morfologiska rekonstruktioner i multidimensionell karakteristisk rymd. Tio algoritmer jämfördes på en datamängd med över 10 000 prover av nervceller från möss. Vidare undersöktes också varje algoritms förmåga att klassificera varje enskild celltyp. Resultaten visar att Random Forest hade den bästa övergripande medelprecisionen följt av Multi-layer Perceptron med 83% respektive 78%. Fortsättningsvis observerades klassificeringen av enskilda celltyper att alla algoritmerna varierade i precision och Random Forest ansågs inte vara den bästa i varje fall. Sammanfattningsvis är maskininlärningsalgoritmer användbara verktyg för att klassificera nervceller, men mer forskning behövs göras för att nå högre precision.

Contents

1 Introduction **1**
- 1.1 Purpose . . . 2
- 1.2 Problem statement . . . 2
- 1.3 Scope . . . 2
- 1.4 Outline . . . 3

2 Background **4**
- 2.1 Terminology . . . 4
- 2.2 Neuroscience . . . 4
  - 2.2.1 Neuron . . . 5
- 2.3 Classifier . . . 6
  - 2.3.1 Multi-layer Perceptron (MLP) . . . 6
  - 2.3.2 Decision Tree CART (DT CART) . . . 7
  - 2.3.3 Random Forest . . . 7
  - 2.3.4 Linear Discriminant Analysis (LDA) . . . 8
  - 2.3.5 Support Vector Classification (SVC) . . . 8
  - 2.3.6 K-nearest neighbors (K-nn) . . . 8
  - 2.3.7 Naive Bayes Gaussian . . . 9
  - 2.3.8 Logistic Regression . . . 9
- 2.4 Related work . . . 10
- 2.5 Software . . . 11

3 Methods **12**
- 3.1 Data collection . . . 12
- 3.2 Data formatting . . . 13
- 3.3 Training the classifiers . . . 14
- 3.4 Model assessment . . . 15

4 Results **16**
- 4.1 Model comparison . . . 16
- 4.2 Cell-type comparison . . . 17

5 Discussion **19**
5.1 Discussion of the results 19
5.2 Future work and Improvements 21
5.3 Reliability 22
5.4 Limitations 22

6 Conclusion **23**

Bibliography **24**

A Morphological features used **27**

B Table: Model assessment correct prediction percent **28**

C Table: Cell type hit percent **29**

D Github **31**

Chapter 1

Introduction

Even after a century of research of electrical circuits in the brain, the answer to how many neuron types that exist is still a challenging question and it is still not resolved how to best describe a neuron and what features to best define them [25]. However, in recent years the amount of data available is ever growing, even in neuroscience. This makes it tiresome to manually classify neurons and as a response, several machine learning techniques have been studied in the last years to help with the classification of neurons.

The main machine learning methods used in previous research when classifying neurons based on their characteristics are supervised classification and unsupervised classification [25], where the typical characteristics are physiology, biochemistry and morphology [1]. Supervised classification is a method to train a model where the cell type is connected to each sample of morphological features which are then used to predict samples where the cell type is unknown. This means that the cell-types are predetermined before the classifications are being conducted. Unsupervised classification is another method to train a model where the cell type for each sample is unknown and the model learns by clustering together samples based on the chosen characteristic.

The neuron classification is an important subject as neurons have different structures depending on their functionality which makes it one reason why we want to classify them [10]. This will also help increase the knowledge of neurobiology and the connections of morphology and function. To cite A. Tsiola "The understanding of any neural circuit requires the identification and characterization of all its components" [25]. Looking at research available today not many have tested different machine learning methods on large neuron data sets, thus more studies in this area are needed to further help automated classification of neurons to ultimately help connect neuronal

types with behavior, computation, and eventually cognition.

1.1 Purpose

The purpose of this research was to analyze and test the performance of how different machine learning algorithms were able to learn to classify a neuron's type based only on its morphology.

1.2 Problem statement

There are not many other studies before this one that has tested different machine learning algorithms for neuron classification and certainly not on large data sets thus it was an important subject to further help the development of machine learning in neuroscience. The problem this study want to address can be summarized with the following question:
"Given ten carefully chosen machine learning algorithms based on previous research, which are better suited for classification of large data sets of morphological reconstructions in multidimensional feature space?".

1.3 Scope

This study was testing how well different machine learning algorithms can classify neurons based on their morphology. This paper did not compare how the algorithms were implemented or tried to implement them in an optimal way but merely to test already implemented algorithms on the same data and see how good they were against each other. For simplicity, only algorithms provided by SciKit Learn was used. The parameters were fine-tuned to get better results rather than just test them by their default parameters and see how they compare. How the fine-tuning was carried out will be explained in the method, section 3.3. There were tools available to help extract several morphological features from neuron reconstructions such as L-Measure [21] but in this study, only morphological data available at neuromorpho.org was used during testing of the algorithms. Neuromorpho.org currently holds 20 morphological parameters but with tools such as L-Measure more could be used and more precise classification could be made but that is not within the scope of this paper. Further, this study will only handle predetermined cell-types and not try to find new ones thus the machine learning method used in this study will be supervised learning as previously mentioned it is the method used for this purpose.

1.4 Outline

This paper consists of six chapters. The first chapter is an introduction to the subject, what the purpose was and why this was an important subject. Chapter two gives a background to the paper in the form of explaining the used algorithms, terminology, neuroscience and other related work. Chapter three explains the methods used, how the data was collected, formatted, how training the classifiers were performed and how they were evaluated. Chapter four presents the results in a plot to see how the algorithms compare to each other. Chapter five discuss the results, future work to be made and what improvements could have been done in this paper. The last chapter is a short conclusion of the results.

Chapter 2

Background

This chapter explains the terminology used, a brief introduction to neuroscience and also the different machine learning algorithms used in the study. Lastly, a section with previous related work for this subject is presented.

2.1 Terminology

API
Application Programming Interface

Model
A model is the result of a machine learning algorithm trained on known data, it is the model that is used to predict unknown data.

Morphology
Morphology is the study of an organisms form and structure [15].

Drosophila melanogaster
A species of fly.

2.2 Neuroscience

Neuroscience is the science about the nervous systems structure and how its function generates behavior. Neuroscience is a branch of biology that specifies in genetics, molecular and cell biology, systems anatomy and physiology, behavioral biology, and psychology [16]. This section will only describe the branch of neurons from neuroscience.

2.2.1 Neuron

The nervous system can be divided into two major branches. The first being supporting cells called neuroglia and nerve cells (neurons) [16]. Neurons typically have four regions, the soma (cell body), dendrites, axon and presynaptic terminals which defines its morphology. The soma is the center of the cell. Dendrites spread like branches on a tree and mainly handles incoming signals from other neurons. Axon often extends from the soma and outputs signals to other neurons, the length of an axon can range from 0.1 mm up to 2 m [17]. An illustration of a neuron can be found in figure 2.1.

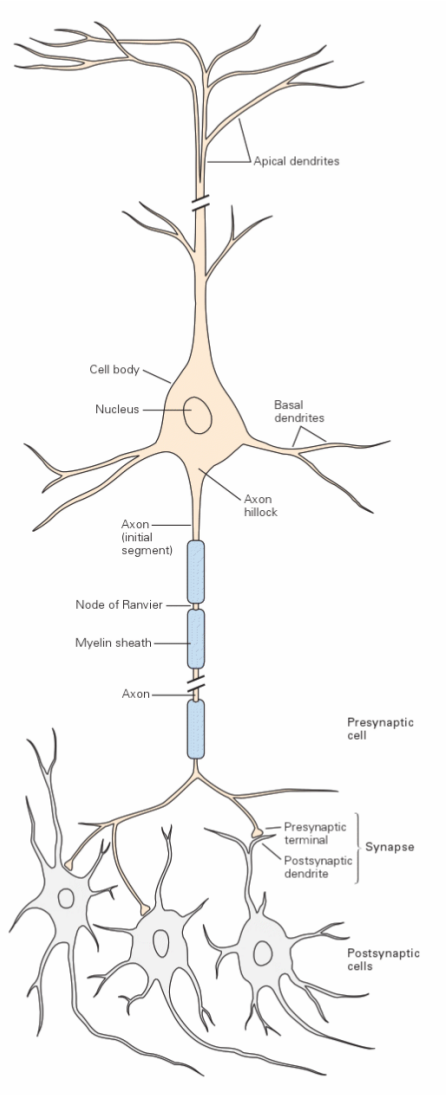

Figure 2.1: A neuron
An illustration of a neuron from [17].

2.3 Classifier

A classifier in machine learning is a model trained on known data to classify new data into a set of known classes. The following classifiers task is to classify every new neuron to the correct type based only on the neurons morphology. The following classifiers explained are the ones used in this paper, where three are described under subsection 2.3.4 because of their similarities.

2.3.1 Multi-layer Perceptron (MLP)

To explain the Multi-layer perceptron knowledge about an artificial neuron and perceptron is needed. An artificial neuron has one or more input and each input is given a weight, these inputs are then multiplied by its corresponding weight and are then all summed together. If the neurons total sum is greater than a given threshold the neuron outputs a one (1) otherwise a zero (0). Figure 2.2 shows an image of an artificial neuron [9].

$$output = \begin{cases} 1 & if \ \sum_i x_i w_i > threshold \\ 0 & otherwise \end{cases}$$

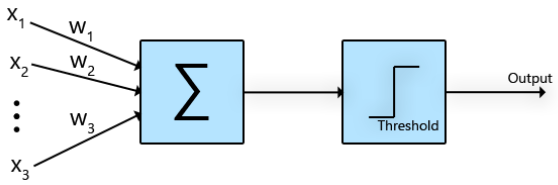

Figure 2.2: An Artificial Neuron, with inputs x_i, weights w_i

A perceptron is a collection of artificial neurons with a set of n inputs that are connected to m neurons with each of the connections having a weight ($w_{n,m}$). In the perceptron, all the neurons are independent of each other and so are also the weights. The output from the perceptron is a vector of zeros and ones, each value representing whether the corresponding neuron's output. The output pattern is compared with the known correct values for this specific input to then identify which of the neurons were correct and which were false. When a neuron outputs a false value a change to the weights for this neuron is made in order to make the neuron output a correct value with future inputs. How much each weight is changed depends on a given parameter called the learning rate, which affects how fast the perceptron learns [9].

Finally, a Multi-layer perceptron uses multiple layers of perceptron where

the output from one perceptron becomes the input for another perceptron. This is important because the normal perceptron can only learn linear dependencies between the neurons while the multi-layer perceptron can learn nonlinear dependencies [9].

2.3.2 Decision Tree CART (DT CART)

The decision tree is a classification method that uses the features of the data as nodes. The tree grows during its training by for each node in the tree chooses a feature to split the tree at. To choose the feature to split at the Entropy is calculated to make sure the split is made at the feature with the highest information gain. Together with the chosen feature, a threshold is added to determine how to divide the feature, as shown in figure 2.3. The tree ends when the information gain is null, all nodes are pure or if the given maximum depth is reached. After the tree is trained the classification is conducted by looking at the unknown data point's feature and for each node in the tree follow the path until a class is reached [2].

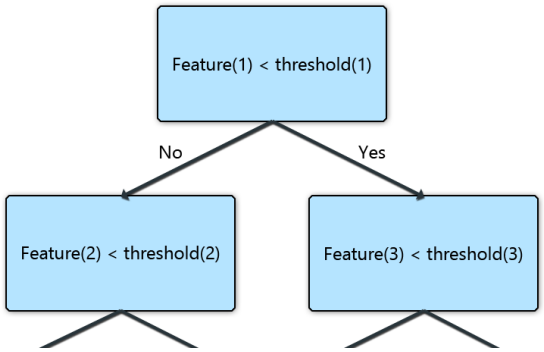

Figure 2.3: Decision tree: feature and threshold

2.3.3 Random Forest

Random forest is a classification method that uses multiple decision trees in order to classify new data points. There is a slight difference in the decision trees that random forest uses, the difference is that the feature chosen at each node is random instead of the best choice and a random subset of the training set is used to train each tree. To determine the class of an unknown data point a majority vote of all the trees in the forest are made. Depending on each of the trees probability estimates each tree will be weighted and the impact on the voting will differ, high weighted trees votes are prioritized [2] [19].

2.3.4 Linear Discriminant Analysis (LDA)

The main goal of LDA is to project the data in a reduced dimension or axis in a way that maximizes the separation of classes. To do this LDA needs to calculate a within-class scatter of the data set by using the mean of the classes, mean of the entire data set and the covariance of each class with itself. With these values the amount of spread that exists in the data set is calculated, this and with the help of the covariance matrix the scatter within the data set is calculated. The scatter itself is calculated by multiplying the probability of a class with the covariance which then gives us the within-class scatter. A between the class scatter is calculated by looking at the difference in means of the classes. In short, LDA maximizes the distance between the means and minimizes the variation within each class and then creates a new dimension or axis that is lower than the previous one and projects the data into that dimension in such a way that it maximizes the separation of the classes [9] [26] [7].

2.3.5 Support Vector Classification (SVC)

Support Vector Classification is a support vector machine used for classification and its goal is to find the best way to separate classes using a hyperplane. A hyperplane is a subspace with one less dimension than its ambient space which divides the space in two. To find the best separating hyperplane a margin is used. The separation will be at its best when the width of the margin is maximized, which is performed during training. The classification of an unknown data point is set by determining which side of the hyperplane the unknown data point is.

In order to handle more complex data which is not linearly dividable, a kernel is used. The kernels task is to move the calculations of the vectors into a higher dimension where they are linearly separable. The kernels used in this paper are linear, polynomial and Radial basis function (RBF) [2]. For simplicity, the kernels were handled as separate classifiers.

2.3.6 K-nearest neighbors (K-nn)

The algorithm K-nearest neighbors use K known data points to classify a new unknown data point. Suppose there are some data points inside an input space of training data, K-nearest neighbor wants to look at k known data points in the training data which are close to it. To do this the distance to the neighbors need to be calculated which could be expensive depending on what space the training set is in. The K-nearest neighbors classification in this paper is done by measuring the Euclidean distance between the new point

and all the other known data points and then assigning the new point to the same class as the most common K nearest points [8] [9].

2.3.7 Naive Bayes Gaussian

Naive Bayes Gaussian is one classifier in the family Naive Bayes. These classifiers use Bayes' theorem shown below in equation 2.1 by determining the probability of an outcome with a certain set of conditions.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

The word naive in a Naive Bayes classifier is based on the conditional independence of causes which might be hard to accept because one might think the presence of a cause is often not independent from another presence of cause. Simply the occurrence of one cause cannot change the probability of other causes. It has been shown that dependencies can clear one another thus making the Naive Bayes result in high performance and even possible.

Naive Bayes Gaussian is a continuous distribution known by its mean and variance making Gaussian a good classifier for continuous values where the probabilities can be modeled using Gaussian distribution. Gaussian is often better to use in more generic testing and classification.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.2}$$

Using the maximum likelihood, shown in equation 2.2, approach the mean and variance can be calculated of the conditional probabilities. With calculating the maximum likelihood its index i refers to the samples in the data set which leads to the model being trained. For a more thorough explanation one can read the referenced material [2].

2.3.8 Logistic Regression

If it is possible to find a linear model with an accuracy better than the predetermined threshold, then a certain problem is linearly separable. A linear model uses separating hyperplanes to classify samples, Logistic regression is one in the family of linear classifiers.

Logistic Regression classifies a given data point based on the probability that it belongs to a class. Because the probability for an event to occur must be bounded between 0 and 1 the logistic function, equation 2.3,

$$P(X) = \frac{e^X}{1 + e^X} \tag{2.3}$$

was introduced. The logistic function is then fitted to the known data point by estimating the maximum-likelihood.

Like in linear regression, weights are used to give different points a certain weight to measure how important it is in the classification. While training the model some features might be less relevant than others thus leading to weight adjustment to get better results [2] [9] [7].

2.4 Related work

The study *towards the automatic classification of neurons* [1] summarizes 27 different studies that have tested machine learning techniques in some way on some form of neuron data. Only two of them have tested on data sets over 800 neurons and the goal in these studies was not to specifically compare the machine learning techniques but to find a specific way to objectively classify what cell types exist and focus more on unsupervised learning.

The study *Comparison between supervised and unsupervised classifications of neuronal cell types: A case study* [4] compares supervised and unsupervised machine learning techniques and specifically in their supervised classification they tested the algorithms ability to distinguish pyramidal cells and interneurons, which are two neuron cell types, in 65 different morphological features. The neurons were digitally constructed using a program called Neurolucida Explorer which gives a good 3D construction and a lot of morphological features, thus the 65 features used in this study. The machine learning techniques tested were MLP, logistic regression, C4.5m Naive Bayes and K-nn. The study used filters to rank the different morphological features. Without any filters MLP performed best with over 87% correctness and with filters used, logistic regression performed best with over 91% correctness.

Another study [3] presents a possible solution for classifying interneurons in a systematic way. The study also shows that 10 different supervised classification models can automatically classify neurons in accordance with experts. 2,886 morphological features were created using Neurolucida Explorer and the tests were run on 241 neurons. One particular test was to classify interneurons in 10 different classes or subtypes where the top four results are from SVC-polynomial kernel, Naive Bayes, K-nn with 3 neighbors and random forest all performing with over 52% accuracy.

Morphological Neuron Classification Using Machine Learning [25] was a study with the goal to provide a way to extract and classify neurons based on their

morphology and also to assess the classifying algorithms. The study used several, both supervised and unsupervised, machine learning algorithms on 430 neuron samples from rats. The result showed that supervised algorithms were better than unsupervised and specifically the algorithm LDA performed the best with an overall mean accuracy of 90%. Other well-performing algorithms were Multi-layer perceptron and Decision tree C5.0.

The previous work gives insight into what algorithms have been tested before and how they have performed. This will help compare the results in this report with the results in previous work. Based on the previous work the best performing algorithms were chosen for this report. Because of the simplicity to use SciKit Learn and add additional algorithms, a few other algorithms were also added from previous work to this study for extra reference.

2.5 Software

All the implementation was written in Python3 using the libraries scikit-learn (sklearn) [18], numpy [13], pandas [14] and matplotlib [11]. Sklearn is the core library used in this study. It is a library filled with a lot of machine learning algorithms and it was from here all algorithms were taken from. Numpy makes handling vectors, matrices, and linear algebra operations easier. Pandas is a library for data structures and was used to read and write .csv files (e.g. Excel, Google sheet files). Finally, matplotlib was used to plot the graphs for the results. All code can be accessed at GitHub which can be found in Appendix D.

Chapter 3

Methods

This chapter explains the methods used for data gathering and formatting. It follows by an explanation of how the classifiers have been trained and how the models were evaluated.

3.1 Data collection

The data used in this research was gathered from the website neuromorpho.org [12], which is a website that collects digitally reconstructed neurons from laboratories worldwide. Data was gathered using a python-script (see Appendix D) written to access the API provided by neuromorpho.org. This website was chosen because it contains morphological information of each neuron in large data sets. The data was also freely available and gathered from all over the world, which was needed for the research. The data chosen to collect from the API was both all the meta-data, containing information about the neuron e.g. its cell type and neuron's name, and all the morphology data. A complete list of the morphological data can be found in Appendix A. The meta-data was needed to determine the cell type and that the reconstruction of the neuron was complete or at least moderate complete. Furthermore, the study limited the data gathering for a specific species, namely mouse. The mouse species was chosen because of its large amount of neuronal reconstruction data available. One goal in this study was to test the classifiers on large amounts of data which had not been accomplished before and mice were the largest data set available, except for rats and Drosophila melanogaster, with over 17 000 samples. However, mice were ultimately chosen because of their more recent samples than those from rats and their similarities with humans [6].

3.2 Data formatting

Some samples from the data set were missing essential information about the neuron, e.g. what type of neuron it is. To avoid unfair classification, e.g some morphological feature missing leading to worse or better classification by a classifier, these samples needed to be handled. Because of a large amount of data available a decision to remove all samples that were missing information about the cell type or the neurons morphology was made. The categorical data was handled by encoding the strings into integers so that it was easier to work with. Feature scaling was used on all the morphological features so that all features were able to give an equal contribution to the model. The feature scaling was performed using SKlearn's function StandardScaler [23].

Each neuron often had multiple cell types which made it hard to choose one and the majority of the neurons were pyramidal cell types which caused a rather uneven distribution. With the help of the project's supervisor (Alexander Kozlov, Dept. Neuroscience) useful cell types were extracted, removing the unnecessary ones, and also solving the challenge with vast pyramidal distribution. If available, pyramidal cells were first divided into subtypes, e.g thick-tufted and midbrain-projecting. Further partitioning was needed and thus pyramidal cells were divided based on their primary brain region. Some unusual cell types were grouped into an others category. After this the cell types were a lot better distributed, the final distribution is shown in the graph 3.1. Because the project's supervisor advised it, the morphological features Overall Depth and Average Diameter, which could contain misleading values, were removed.

There exist techniques to further handle uneven distribution, e.g resampling. Resampling consists of numerous methods to help balance uneven data. These methods are widely-employed in areas such as chemical and biomedical engineering [5]. This paper will not use any resampling methods or other similar methods to balance the data. Instead, it was studied how well the chosen classifiers were able to classify the data even though it was unbalanced. This will be more connected to a real scenario as data is seldom balanced.

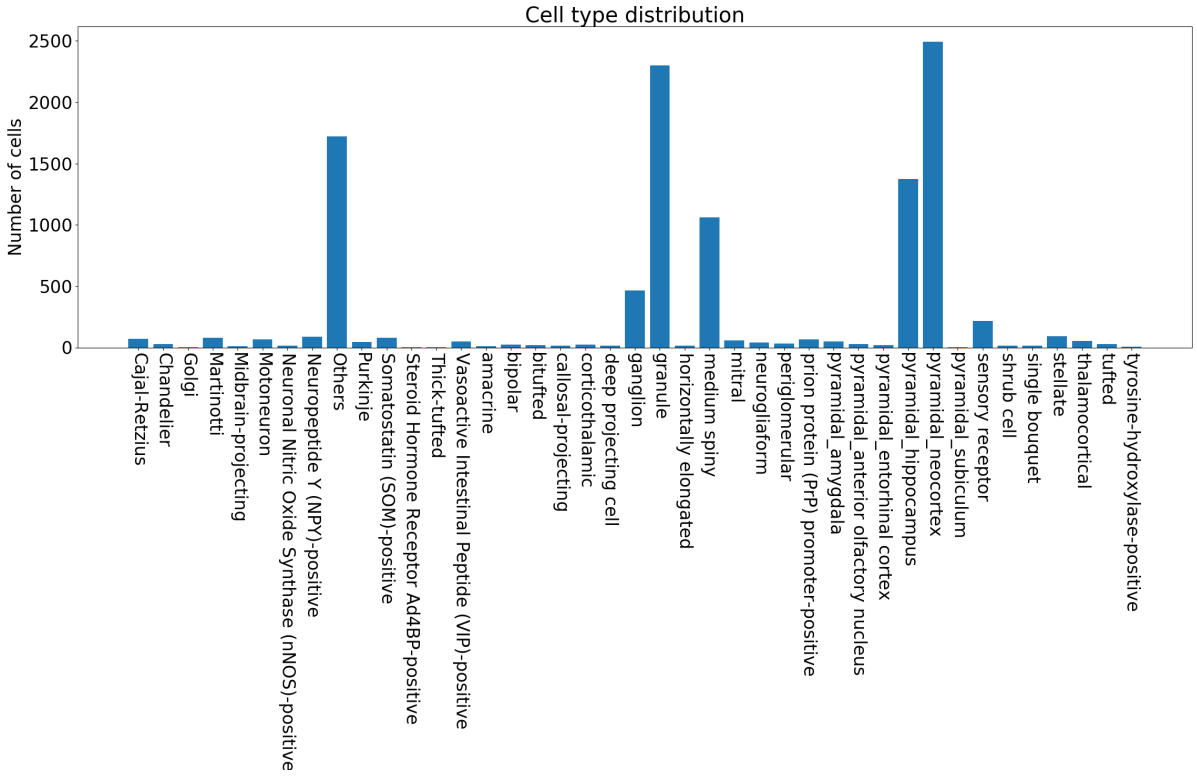

Figure 3.1: Cell type distribution

3.3 Training the classifiers

Each classifier had several parameters that adjust how they behave and how accurate the results would be. The parameters were manually tested and run 20 times, which is an arbitrary number of runs due to time restrictions, to see if the accuracy changed from previous parameter settings. The parameters that gave the best accuracy for each classifier was kept to the main testing with 1000 iterations. The number 1000 was chosen to reach a higher order of statistical correctness but still restricted to the run-time of nearly 72 hours. The final parameters can be found in the code on GitHub, see Appendix D. The data set was randomly divided into two different sets, $2/3$ of the data set was used as for training set and the remaining samples were used for the test set. Dividing into a test and a train data set was executed to make it possible to assess the model [9]. All the classifiers were trained on the same set of training data and assessed of the same set of test data. The training and the testing were run 1000 times for each model.

3.4 Model assessment

To evaluate the classifiers they were all trained a thousand times on randomly divided partitions of the training and test data, each time the classifier has trained the accuracy was calculated and stored according to equation 3.1, a correct sample was a sample where the model prediction was equal to the known cell type [9]. After all the classifiers were tested the mean and the standard deviation was calculated from the previously stored accuracy.

$$Accuracy = \frac{Amount\ of\ correct\ classified\ samples}{Total\ amount\ of\ samples} \tag{3.1}$$

Each model was also assessed in how they manage to classify each cell type, e.g. one model might be able to classify 100% of type 1 but can't classify any other types, while another model might classify 50% of every type. This was built in python together with the accuracy calculations by storing for every model and every cell type how many that this model classified correctly and how many that was falsely classified. Much like the confusion matrix which is a matrix of the known cell type and the predicted cell type [9]. The confusion stores for each known cell type what the actual predicted cell type was. For this study, it was chosen to not use the confusion matrix because the training set was randomly selected and could in some cases miss some cell types which led to the confusion matrix varying in size, instead a unique solution to this problem was developed and chosen. The unique solution was a list for each classifier containing the amount of correct classified and miss-classified cell types.

Chapter 4

Results

This section will first present how well each classifier performed in sense of overall accuracy. The results will be presented both in a diagram and a table. The next part will further present how each classifier performed for each cell type.

4.1 Model comparison

The different models were assessed as described in section 3.4 and the result was presented in graph 4.1 showing the mean accuracy of each algorithm together with the standard deviation. The result showed that Random Forest was the algorithm that had the highest mean accuracy when classifying neurons based on their morphology with a mean of 83.4% correct prediction after 1000 tests. Close after Random Forest, both Multi-layer Perceptor and SVC (RBF) performed with a mean correct prediction of 78.2% and 77.4% respectively. The algorithm with the lowest accuracy when classifying neurons was Naive Bayes (Gaussian) with only a mean of 33.3% correct prediction.

Figure 4.1 presents the mean and standard deviation results for each model and for the exact mean percentage for each model see Appendix B.

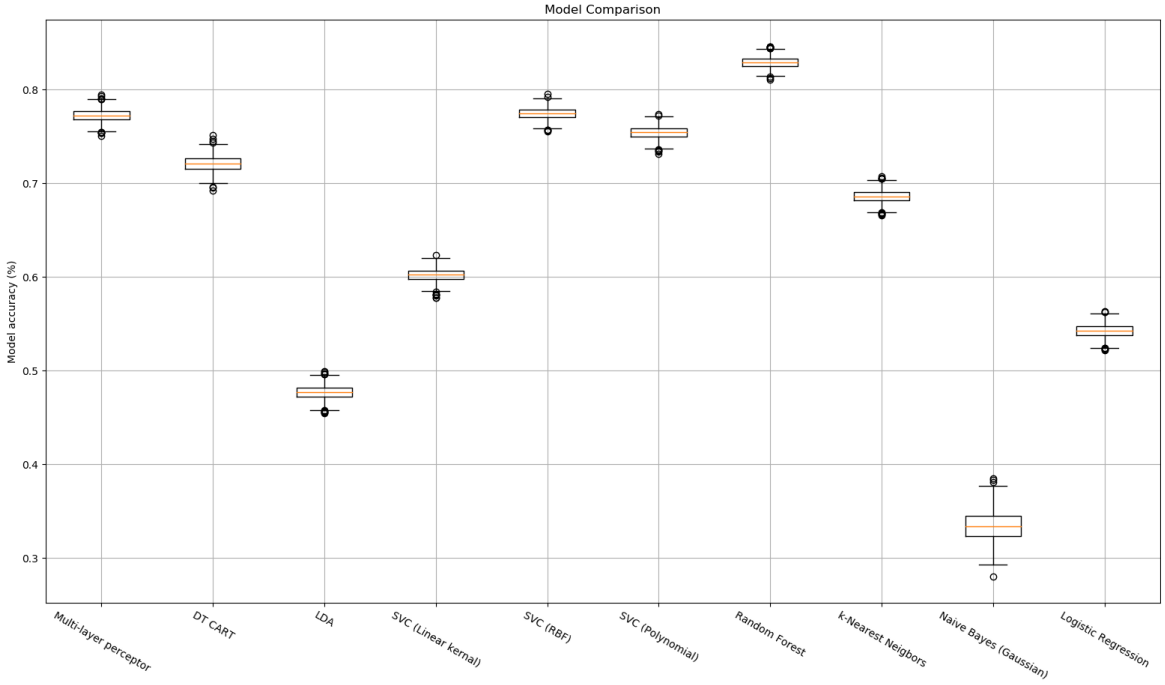

Figure 4.1: Model comparison
The standard deviation for each classifier (The circles represent outliers).

4.2 Cell-type comparison

The percentage of correct predictions for each cell type was also studied. Some cell types were harder to predict than others as the mean values for each cell types was varying ranging from 0% to 90.23%, as shown in the graphs 4.2 with corresponding ids in table 4.3. It was two cell types that no algorithm managed to predict, these two were Thick-tufted (number 30) and Steroid Hormone Receptor Ad4BP-positive (number 40). The cell types that most algorithms were able to predict was ganglion (number 10), granule (number 13) and Purkinje (number 0). The percentage of correct predictions for each cell type can be found in Appendix C.

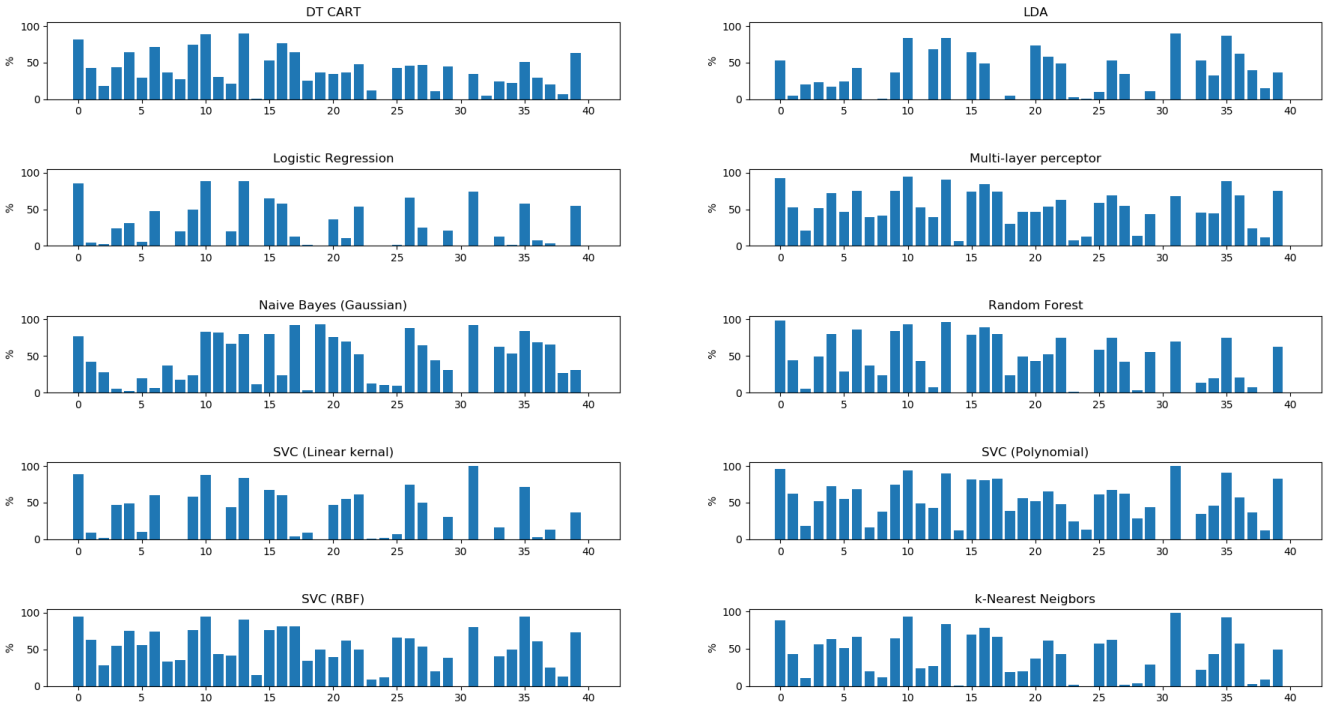

Figure 4.2: Cell type hit ratio
The Y-axis represents the accuracy in percent for each specific cell type and the numbers on the X-axis corresponds to the cell type id in figure 4.3

id	Cell type	Number of cells	id	Cell type	Number of cells
0	**Purkinje**	44	21	**corticothalamic**	25
1	**Motoneuron**	65	22	**neurogliaform**	39
2	**Somatostatin (SOM)-positive**	78	23	**Neuronal Nitric Oxide Synthase (nNOS)-positive**	15
3	**Neuropeptide Y (NPY)-positive**	86	24	**tyrosine-hydroxylase-positive**	5
4	**Others**	1719	25	**Vasoactive Intestinal Peptide (VIP)-positive**	49
5	**Martinotti**	79	26	**prion protein (PrP) promoter-positive**	67
6	**pyramidal_neocortex**	2493	27	**amacrine**	10
7	**stellate**	91	28	**pyramidal_entorhinal cortex**	21
8	**thalamocortical**	55	29	**mitral**	60
9	**medium spiny**	1063	30	**Thick-tufted**	1
10	**ganglion**	463	31	**Midbrain-projecting**	11
11	**periglomerular**	33	32	**pyramidal_subiculum**	2
12	**callosal-projecting**	15	33	**single bouquet**	13
13	**granule**	2298	34	**bitufted**	21
14	**Golgi**	4	35	**bipolar**	22
15	**Cajal-Retzius**	72	36	**shrub cell**	15
16	**pyramidal_hippocampus**	1375	37	**horizontally elongated**	14
17	**pyramidal_amygdala**	48	38	**deep projecting cell**	13
18	**Chandelier**	30	39	**sensory receptor**	217
19	**pyramidal_anterior olfactory nu**	27	40	**Steroid Hormone Receptor Ad4BP-positive**	1
20	**tufted**	28			

Figure 4.3: Cell type id

Chapter 5

Discussion

This section will discuss the different results for each model and also their capacity to classify certain cell types. Next, some notes of future work will be described followed by a section of reliability. Last some discussion of limitations.

5.1 Discussion of the results

This study mainly compared ten different machine learning algorithms to see how well they were to classify neuron cell types based on their morphology. Even though data was collected from neuromorpho.org with a limited number of morphological features some classifiers got an impressive result. Random forest had a mean of 83% correct predictions over 1000 iterations. Multi-layer perceptron, DT CART, SVC (RBF) and SVC (Polynomial) all had a mean of over 70% correct predictions which all were considered acceptable results. The other five had lower scores and should not be considered as good results. Looking back at the related work all previous studies had a lot more available morphological features leading to higher prediction scores for the same classifiers. As the available morphological features in this study were far less the results are not in alignment with previous studies but were still considered high for the sake of available features, even though one would like to at least get over 90% scores. One key point to take away in this was that the currently available number of features from neuromorpho.org was not enough to get the desired scores. If possible, maybe neuromorpho.org could try to add more morphological data for each neuron to further help the research in this area. Maybe there were some features more crucial than others? As the previous work mentioned, they had to cut some morphological features out to receive higher results, thus some features are obviously redundant. Neuromorpho.org could instead try to add features and find this

fine line and which features are most useful when determining cell-types out of morphological data. It was still worth to point out that a mean of 83% is impressive considering the morphological features were few in numbers. When comparing these results with previous work the top two performing algorithms from the second mentioned study was MLP with 87% correctness and with filters Logistic Regression scoring over 91% correctness. The next study had four classifiers with even top results where one being Random Forest and another SVC (polynomial). The final study mentioned had Linear Discriminant as the best performing with a mean of 90% but also MLP and Decision Tree C5.0 had high correctness. MLP, Random Forest and SVC (polynomial) were mentioned in previous related work which also was the best performing algorithms, with SVC (RBF), in this study when looking at overall accuracy. It seems that MLP, Random Forest and SVC (nonlinear) are more suited when classifying neuron cell-types based on this study and previous work before this.

The result from this study also shows that each of the machine learning algorithms performance differed depending on the cell types. Although Random Forest had the overall highest accuracy one might not always want to use it. E.g. Random Forest could only predict on average 7.70% of the cell type "callosal-projecting" where Naive Bayes (Gaussian), which had the lowest overall accuracy, was able to predict on average 66.35% of this cell type. When choosing an algorithm one might want to consider which cell types that are in focus. It was also worth noting that in the results it was shown that no classifier was able to classify two cell types, Steroid Hormone Receptor Ad4BP-positive and Thick-tufted. These two cell types are also the types with the least amount of cells, both with only 1 cell each, which might suggest that the problem with classifying these cell types lies in the unbalanced distribution of the data and not in the classifiers. Observing the mean accuracy for the individual cell types in Appendix C, both the nonlinear SVCs had the highest value closely followed by Multi-layer Perceptron. Furthermore, this shows that even with high overall accuracy a classifier might misclassify some of the cell types.

Seeing how random forest performed best in overall classification it was not best when observing individual cell-types but placed in fourth place. This shows that the term best differs depending on what evaluation method is used and what one might want to look for when classifying neurons. In the study *machine learning and its applications to biology* [24] this statement was backed as they found that caution should be taken when comparing one machine learning algorithm over another. The study also shows that many studies published as successful have been shown to be overoptimistic, this

because of chosen evaluation method [24]. Therefore one might be careful when choosing evaluation method and choose one that best fits the goal of the analysis.

Looking further ahead this study could help contribute to more than just the understanding and development of neuroscience. Higher understanding of the neuronal system could lead to better prostheses for all beings. Prostheses could be controlled directly by the brain and be constructed to make the user feel through prostheses. These developments are built on advances in neuroscience among advances in other areas as well. Medical neurotechnologies also have the potential to help minimize impact in symptoms from Parkinson's Disease, depression and help regain lost function caused by spinal cord or nerve damage [22].

5.2 Future work and Improvements

As there are not many studies on large data sets we encourage any future work to keep doing these comparisons with large data sets instead of small ones. Obviously, large data sets make it more statistical significant but it is currently hard to get large data sets that are complete with all the cell feature data needed. As with all research, it would also be encouraged to do these studies on human samples. Unfortunately, there are currently not that many human samples available at neuromorpho.org which was one reason this study didn't use human samples. The data on neuromorpho.org doesn't keep all features of a cell and thus any future work could use programs such as L-measure that was previously mentioned to extract more features. This could lead to a more correct prediction from classifiers as seen in the related work section where several studies had more features available than this study, leading to better results. Also, some previous studies found that some morphological features actually worsen the results thus any future work related to specifically neuromorpho.org could test if certain classifiers perform better with some features removed. Resampling as mentioned earlier and also other balancing methods could be used in the future for further testing how balanced data from neuromorpho.org compare with different classifiers. Any future work could also compare other supervised methods other than the ones in this study. One subject that was not discussed in this paper was unsupervised methods that also could be interesting for future work.

An encouragement is made for future work in more depth to study different classifiers ability to determine different cell types and figure out why some are better than others in this area.

5.3 Reliability

The data set consisted of over 10 000 samples which have never been studied before when looking at neuron classification (at least not what could be found) which was a plus. The classifiers were taken from a third party library which was considered reliable as previous work has used the same library but also companies such as Spotify were using the library [20] leaving any error from the writers part minimal. The classifiers were trained and run 1000 times which could be argued as reliable in statistical references. One part of the study that could affect the reliability is the parameter testing for each classifier. To find the best parameters only 20 iterations were run for each parameter to find a better result, which was not much considered the main run was for 1000 iterations. In the sense of looking at data sets with few amounts of morphological features, this study is reliable.

One weakness of this study is in its reproducibility due to missing the parameter "random-state" for several of the algorithms used. The random-state parameter would have made the algorithms use the same random generated values if the tests were run again and ensured that the algorithms determinate. But, because of a thousand runs and with all outliers close to the standard deviation a reproduction of the test would probably have similar results.

5.4 Limitations

As the study aimed to use large data sets and specifically data from neuromorpho.org the biggest limitation was the available species for large data sets. Rats and Drosophila melanogaster had a larger data set but rats consisted of older data and ultimately mice were chosen because of their similarities with humans [6]. Because neuromorpho.org was chosen as a source for neuron data only 20 features were available. In reality, there are a lot more as some researchers have found almost 3000 morphological features using specialized tools [3]. Preferably neuromorpho.org would add more features in the future for more easier to use data.

Chapter 6

Conclusion

The study showed that Random Forest, Multi-layer Perceptron and SVC (non-linear kernel) were better suited for classification of large data sets of morphological reconstructions in multidimensional feature space. However, all algorithms could be viable and the choice of algorithm depends on which cell type one wants to classify due to the prediction accuracy of each type was varying.

Bibliography

[1] Rubén Armañanzas and Giorgio A. Ascoli. "Towards the automatic classification of neurons". eng. In: *Trends in Neurosciences* 38.5 (May 2015), pp. 307–318. ISSN: 0166-2236.

[2] Giuseppe Bonaccorso. Packt Publishing, 2017. ISBN: 978-1-78588-962-2. URL: https://app.knovel.com/hotlink/toc/id:kpMLA00001/machine-learning-algorithms/machine-learning-algorithms.

[3] Javier DeFelipe et al. "New insights into the classification and nomenclature of cortical GABAergic interneurons". In: *Nature Reviews Neuroscience* 14 (2013), pp. 202–216.

[4] Luis Guerra et al. "Comparison between supervised and unsupervised classifications of neuronal cell types: A case study". In: *Developmental Neurobiology* 71.1 (), pp. 71–82. DOI: 10.1002/dneu.20809. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dneu.20809. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dneu.20809.

[5] Guo Haixiang et al. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.12.035. URL: http://www.sciencedirect.com/science/article/pii/S0957417416307175.

[6] National Human Genome Research Insitute. *Why Mouse Matters*. June 2010. URL: https://www.genome.gov/10001345/importance-of-mouse-genome/.

[7] Gareth James et al. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. ISBN: 1461471370, 9781461471370.

[8] "k-Nearest Neighbor Algorithm". In: *Discovering Knowledge in Data*. Wiley-Blackwell, 2014. Chap. 7, pp. 149–164. ISBN: 9781118874059. DOI: 10.1002/9781118874059.ch7. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118874059.ch7. URL:

https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118874059.ch7.

[9] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. 2nd. Chapman & Hall/CRC, 2014. ISBN: 1466583282, 9781466583283.

[10] F.E.A. Martini. *Anatomy and Physiology' 2007 Ed.2007 Edition*. Rex Bookstore, Inc., 2007. ISBN: 9789712348075. URL: https://books.google.se/books?id=joJb82gVsLoC.

[11] *matplotlib*. URL: https://matplotlib.org/.

[12] *NeuroMorpho.org*. URL: http://neuromorpho.org.

[13] *numpy*. URL: http://www.numpy.org/.

[14] *pandas*. URL: https://pandas.pydata.org/.

[15] Oxford University Press. *Definition of morphology in English*. visited on 2018-04-30. URL: https://en.oxforddictionaries.com/definition/morphology.

[16] Dale Purves et al. *Neuroscience third edition*. Sunderland, 2004. ISBN: 0-87893-725-0.

[17] James Schwartz et al. *Principles of Neural Science, Fifth Edition*. McGraw-Hill Publishing, 2012. ISBN: 9780071810012.

[18] *scikit-learn*. URL: http://scikit-learn.org/stable/.

[19] *scikit-learn: Random Forest*. URL: http://scikit-learn.org/stable/modules/ensemble.html#random-forests.

[20] *scikit-learn: Testimonials*. URL: http://scikit-learn.org/stable/testimonials/testimonials.html.

[21] Ruggero Scorcioni, Sridevi Polavaram, and Giorgio A. Ascoli. "L-Measure: A Web-Accessible Tool for the Analysis, Comparison, and Search of Digital Reconstructions of Neuronal Morphologies." In: *Nature protocols* 3.5 (2008), pp. 866–876.

[22] Robert K Shepherd. *Neurobionics : the biomedical engineering of neural prostheses*. eng. 2016. ISBN: 1-118-81603-X.

[23] *SKlearn StandardScaler*. URL: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler.

[24] Adi L Tarca et al. "Machine learning and its applications to biology". In: *PLoS computational biology* 3.6 (2007), e116.

[25] Xavier Vasques et al. "Morphological Neuron Classification Using Machine Learning". eng. In: *Frontiers in Neuroanatomy* 10 (Nov. 2016). ISSN: 1662-5129.

[26] Andrew R. Webb and Keith D. Copsey. “Linear Discriminant Analysis”. eng. In: *Statistical Pattern Recognition*. Chichester, UK: John Wiley Sons, Ltd, Nov. 2011, pp. 221–273. ISBN: 9780470682272.

Appendix A

Morphological features used

Feature
Soma Surface
Number of Stems
Number of Bifurcations
Number of Branches
Overall Width
Overall Height
Total Length
Total Surface
Total Volume
Max Euclidean Distance
Max Path Distance
Max Branch Order
Average Contraction
Total Fragmentation
Partition Asymmetry
Average Rall's Ratio
Average Bifurcation Angle Local
Fractal Dimension

Table A.1: List of morphological features

Appendix B

Table: Model assessment correct prediction percent

Classifier	**Mean percentage after 1000 runs**
Multi-layer perceptor	78.198 %
DT CART	71.969 %
LDA	47.164 %
SVC (Linear kernel)	59.900 %
SVC (RBF)	77.392 %
SVC (Polynomial)	75.612 %
Random Forest	83.398 %
k-Nearest Neighbors	69.216 %
Naive Bayes (Gaussian)	33.315 %
Logistic Regression	54.477 %

Table B.1: Correct prediction percent

Appendix C

Table: Cell type hit percent

id	Cell type	Number of cells after 1000 runs	DT CART	LDA	Logistic Regression	Multi-layer perceptor	Naive Bayes (Gaussian)	Random Forest	SVC (Linear kernal)	SVC (Polynomial)	SVC (RBF)	k-Nearest Neigbors	MEAN
0	**Purkinje**	14601	81.99%	52.47%	84.91%	92.93%	76.43%	98.50%	88.61%	95.59%	94.49%	88.66%	**85.46%**
1	**Motoneuron**	21666	42.28%	4.79%	4.18%	53.00%	41.77%	43.70%	9.37%	62.05%	63.00%	43.33%	**36.75%**
2	**Somatostatin (SOM)-positive**	26024	18.39%	19.84%	2.98%	21.34%	27.77%	5.39%	1.47%	17.75%	28.12%	10.81%	**15.39%**
3	**Neuropeptide Y (NPY)-positive**	28658	43.64%	22.83%	23.88%	51.84%	5.46%	48.91%	46.94%	51.56%	54.72%	56.05%	**40.58%**
4	**Others**	572409	64.37%	17.40%	31.09%	71.93%	2.36%	80.00%	48.45%	72.23%	74.82%	62.81%	**52.55%**
5	**Martinotti**	26441	29.26%	24.54%	5.96%	46.98%	19.65%	28.68%	10.51%	55.58%	55.40%	51.32%	**32.79%**
6	**pyramidal_neocortex**	830377	71.29%	43.23%	47.20%	75.43%	6.57%	85.62%	59.72%	68.74%	73.90%	66.03%	**59.77%**
7	**stellate**	30102	36.33%	0.00%	0.00%	39.85%	36.53%	36.82%	0.00%	16.22%	33.16%	19.19%	**21.81%**
8	**thalamocortical**	18434	26.93%	0.88%	19.57%	41.37%	17.04%	23.35%	0.00%	37.88%	35.57%	12.09%	**21.47%**
9	**medium spiny**	354110	74.01%	36.72%	49.65%	75.06%	23.56%	83.69%	57.88%	74.67%	76.61%	63.91%	**61.58%**
10	**ganglion**	154136	89.13%	84.00%	88.66%	94.47%	82.96%	93.45%	87.55%	93.77%	94.62%	93.66%	**90.23%**
11	**periglomerular**	10975	30.39%	0.00%	0.00%	52.68%	81.82%	42.65%	0.00%	48.79%	43.03%	23.40%	**32.28%**
12	**callosal-projecting**	5055	21.35%	68.68%	20.34%	39.03%	66.35%	7.70%	44.04%	42.61%	40.97%	26.45%	**37.75%**
13	**granule**	766911	89.60%	84.04%	88.30%	90.33%	79.72%	96.61%	83.96%	90.11%	90.88%	82.71%	**87.63%**
14	**Golgi**	1363	0.29%	0.00%	0.00%	6.16%	11.37%	0.00%	0.00%	12.47%	14.67%	0.15%	**4.51%**
15	**Cajal-Retzius**	23826	53.20%	64.28%	65.24%	73.99%	80.33%	78.65%	67.50%	81.93%	76.15%	68.64%	**70.99%**
16	**pyramidal_hippocampus**	459184	76.55%	48.87%	58.11%	84.01%	23.45%	89.57%	60.38%	80.49%	81.83%	78.46%	**68.17%**
17	**pyramidal_amygdala**	16021	63.86%	0.01%	12.93%	74.17%	92.12%	80.07%	4.09%	82.40%	81.56%	66.06%	**55.73%**
18	**Chandelier**	10055	25.17%	4.88%	1.63%	30.22%	3.33%	24.17%	9.18%	38.36%	34.11%	18.20%	**18.93%**
19	**pyramidal_anterior olfactory nucleus**	8973	36.50%	0.00%	0.00%	46.50%	93.60%	49.70%	0.00%	56.21%	49.53%	19.24%	**35.13%**
20	**tufted**	9402	34.92%	73.56%	36.65%	46.64%	75.65%	43.49%	47.30%	52.30%	39.80%	37.28%	**48.76%**
21	**corticothalamic**	8322	36.61%	57.65%	10.97%	53.80%	69.92%	52.14%	54.90%	65.60%	62.08%	60.71%	**52.44%**
22	**neurogliaform**	12965	47.94%	49.31%	53.96%	63.26%	52.61%	74.87%	61.70%	47.83%	49.21%	42.48%	**54.32%**
23	**Neuronal Nitric Oxide Synthase (nNOS)-positive**	4952	12.44%	3.17%	0.28%	7.31%	12.84%	0.79%	0.55%	23.97%	8.56%	1.39%	**7.13%**
24	**tyrosine-hydroxylase-positive**	1698	0.08%	0.65%	0.00%	12.60%	10.37%	0.00%	1.47%	12.78%	11.84%	0.00%	**4.98%**
25	**Vasoactive Intestinal Peptide (VIP)-positive**	16365	43.16%	10.03%	1.20%	58.92%	9.27%	58.34%	7.00%	60.72%	65.85%	57.35%	**37.19%**
26	**prion protein (PrP) promoter-positive**	22411	46.07%	52.52%	65.55%	68.65%	88.59%	75.03%	74.72%	67.42%	65.08%	62.47%	**66.61%**
27	**amacrine**	3309	46.99%	34.21%	24.87%	54.85%	65.03%	42.55%	50.35%	62.16%	53.55%	1.84%	**43.64%**
28	**pyramidal_entorhinal cortex**	6985	10.94%	0.00%	0.00%	13.51%	44.41%	2.92%	0.00%	28.06%	19.57%	3.89%	**12.33%**
29	**mitral**	20179	45.16%	11.48%	20.45%	43.36%	31.29%	55.50%	30.20%	43.85%	38.64%	28.90%	**34.88%**
30	**Thick-tufted**	353	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.00%**
31	**Midbrain-projecting**	3673	34.60%	89.46%	74.57%	68.12%	92.30%	69.48%	99.81%	99.78%	79.88%	98.48%	**80.65%**
32	**pyramidal_subiculum**	693	5.34%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.53%**
33	**single bouquet**	4315	24.24%	53.16%	12.56%	45.28%	62.13%	13.86%	16.66%	34.81%	40.16%	21.55%	**32.44%**
34	**bitufted**	7039	21.85%	32.87%	1.55%	44.20%	53.32%	19.16%	0.23%	45.83%	49.95%	42.86%	**31.18%**
35	**bipolar**	7211	50.85%	86.55%	57.98%	88.35%	84.09%	75.26%	71.18%	90.81%	94.37%	92.65%	**79.21%**
36	**shrub cell**	5061	28.95%	61.85%	7.59%	69.47%	68.72%	20.35%	2.98%	56.97%	61.19%	56.91%	**43.50%**
37	**horizontally elongated**	4692	19.88%	39.30%	3.43%	24.49%	66.13%	7.46%	13.00%	37.06%	24.68%	2.58%	**23.80%**
38	**deep projecting cell**	4387	7.13%	15.41%	0.91%	11.63%	26.40%	0.43%	0.27%	11.69%	12.86%	9.00%	**9.57%**
39	**sensory receptor**	72334	62.98%	36.30%	55.07%	74.83%	31.28%	62.29%	36.57%	82.33%	73.20%	48.80%	**56.36%**
40	**Steroid Hormone Receptor Ad4BP-positive**	333	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.00%**
	MEAN	**14601**	**37.92%**	**31.34%**	**25.18%**	**49.04%**	**44.31%**	**43.20%**	**30.45%**	**51.30%**	**49.94%**	**39.52%**	

Figure C.1: Cell type hit ratio

Appendix D

Github

https://github.com/MarcusJoakimKex2018/AutomaticClassification