DELIVERABLE 1:

In order to make decisions like thinking creatures, it is necessary to determine on the basis of what exactly the decision is made. We need to know what is going on in the head of a human or other thinking creature. States of internal decision system or mind called mental states.

The main idea of functionalism is that the mental state does not depend on its internal structure, but on how it functions, what role it plays in a system of which it is a part of, for example, completely different states can play the role of pain - the mechanism necessary to determine dangerous impacts for the individual and their termination. This mechanism can be implemented in a variety of ways, such as nerve fiber signal transmission, or a digital variable in a program. This property of functionalism is called Multiple Realizability.

We will never know whether the two different creatures are equivalent since we simply do not have the tool to get into their heads and compare their feelings. We can draw such conclusions only after a thorough analysis of the functions performed. In this case, it calls the functionally equivalent.

Therefore, we do not have truly objective criteria for evaluating the reasonableness of AI, but only subjective ones - it SEEMS only to us that AI behaves as thinking. Yes, we can try to determine that if two creatures in the same situations make the same decisions, then they are equivalent, but it is worth remembering that decision making is a probability, not a clear function, i.e. even the same creature in absolutely identical situations can make different decisions (even with the same probability distribution), which means according to this criterion the same creature is not identical to itself, which is absurd.

It seems that we can never know is AI reasonable or simply pretend to be thinking (and we can also find out if we get into the necessary situation in which the AI manifests its unreasonableness, but at the same time we cannot guarantee that this situation will happen) Nature (animals, human) is imperfect and easy to deceive, but at the same time it is truly perfect in terms of using resources, it does not spend them on what is not needed, and the definition of AI is not the most necessary skill for Cro-Magnon survival . Therefore, the main existing criterion for determining reasonableness is Human testing or Turing test. Figuratively speaking, we have a creature that we know for sure that it is thinking – it's a human being, and we take it as the sole reference (as long as we don't know other thinking creatures), and we can say we compare it. After all, if we want to fake any social animal, for example, a sheep, then the only criterion of success/failure of our actions is whether the other sheep will take our fake for their own.

Turing test attracts with its simplicity and clarity and therefore is very popular. But at the same time, it is not practical, because teaches AI to solve only one problem - to deceive a human. And if we check only one property, then we can optimize it only. While in the modern world, AI requires to solve completely different tasks.

PS: After reading the article you can see that the authors often consider pain as a sample of emotion / inner state, but for some reason, they consider it in a very narrow sense - a finger bruised along the nerve C fibers causes wincing or moaning. But there are many other nonlinear manifestations of pain:

- the pain of offense or empathy;

- the perception or non-perception of pain under hypnosis;

- the experiment when a person is stroked with his hand and an artificial hand in front of him, and then the artificial hand is beaten in his sight;
- walking on coals when a person does not feel pain;
- etc…

Moreover, in all languages, those manifestations are also called “pain” comparable to pain from a bruised finger. All this proves that C fiber has nothing to do with it, and the reason is a belief in pain.

DELIVERABLE 2:

The machine learning system doesn't care what exactly the data in data sets means, only numerical values are important. But what data depends on its grouping. Let's see its features in a classification context

2 base data sets:

1) Iris data set https://archive.ics.uci.edu/ml/data sets/iris

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

2) Wine data set https://archive.ics.uci.edu/ml/data sets/wine

In a classification context, this is a well-posed problem with "well behaved" class structures. A good data set for first testing of a new classifier, but not very challenging.

2 additional data sets chosen.

In theory, machine learning can work with any data set. But some data sets need less preparation. We use the UCI site search, and choose these features:

- Default Task – Classification
- Attribute – Numerical
- Format Type – Matrix
- Instances – 100 to 1000
- Also, we considered how easy we can prepare data sets to machine learning

We chose the next data sets:

3) Glass Identification Data Set https://archive.ics.uci.edu/ml/data sets/glass+identification
4) Haberman's Survival Data Sethttps://archive.ics.uci.edu/ml/data sets/Haberman%27s+Survival

They are not better or worse than each other. They just satisfy to mentioned above criteria

Data separation

The separation of data into sets directly depends on the chosen cross-validation strategy. The most popular solution is K-folds. This is the partition of the entire data set into the K section, each of which is taken in turn as a training set, and all the other data as a test set. Usually K = 5 or 10. In our case, it is necessary to select training data as 10% and 50%, then at 50% K = 2, and there will be only 2 iterations of cross-validation. It's too low and may provide incorrect data. Also, we can use random subsampling. This method with a large number of executions provides a very good result, but has several disadvantages:

1) each following result differs from the previous one, the more performances, the smaller the difference

2) a large number of executions leads to a large waste of computational resources.

The greater the number of iterations in cross-validation, the more reliable the data, so it is desirable that both 10% and 50% of the training data have the same accuracy. Therefore, we still choose random subsampling cross-validation

For testing, you can use the entire Testing set or it's part only – Validation set. The only reason to use part of testing data for validation only - is to save computing resources. The validation set is 10% typically. But if the data set itself is small, then the validation set will be very small and the validation may provide incorrect data. Therefore, it will be reasonable to use Testing set as Validation set for small sets, and 10% to test large ones. The condition between a large and a small set is defined as a comparison of its length with a certain number, for example, 300. This number only affects the degree of saving computing resources, therefore its exact value is not so important.

It should be noted that in the project we use two different checks by various methods.
1) Estimation of the root-mean-square error for finding the optimal regularization parameter. It occurs on the Validation set. We do not need high accuracy. Just calculating error by Validation set
2) Assessment of the classification method itself (Least Mean Squares or Support Vector Machine). High veracity is needed here

Selection of free parameter - regularization parameter λ

By using regularization, we can achieve better prediction accuracy. But to know what value to choose, we will not trust randomness, but find it ourselves. First, we define the criterion by which we will determine the accuracy of the prediction. The most obvious is the number of misclassification errors, but since we have a finite amount of validation data, the result will be highly discrete. For example, for an iris data set with 50% of training data, this is only 75 possible values. Therefore, we need to evaluate which correct predictions are more correct. Since we use the Least Squares Method, we will estimate the mean-square error. More precisely, for less calculation, we can calculate the total quadratic error only without averaging.

Let’s plot curves for correct predictions (Fig.1) and error (Fig.2) depending on the regularization parameter λ for Iris data set and 10% training data. Matlab file: classifier_Iris_plot_curves.m

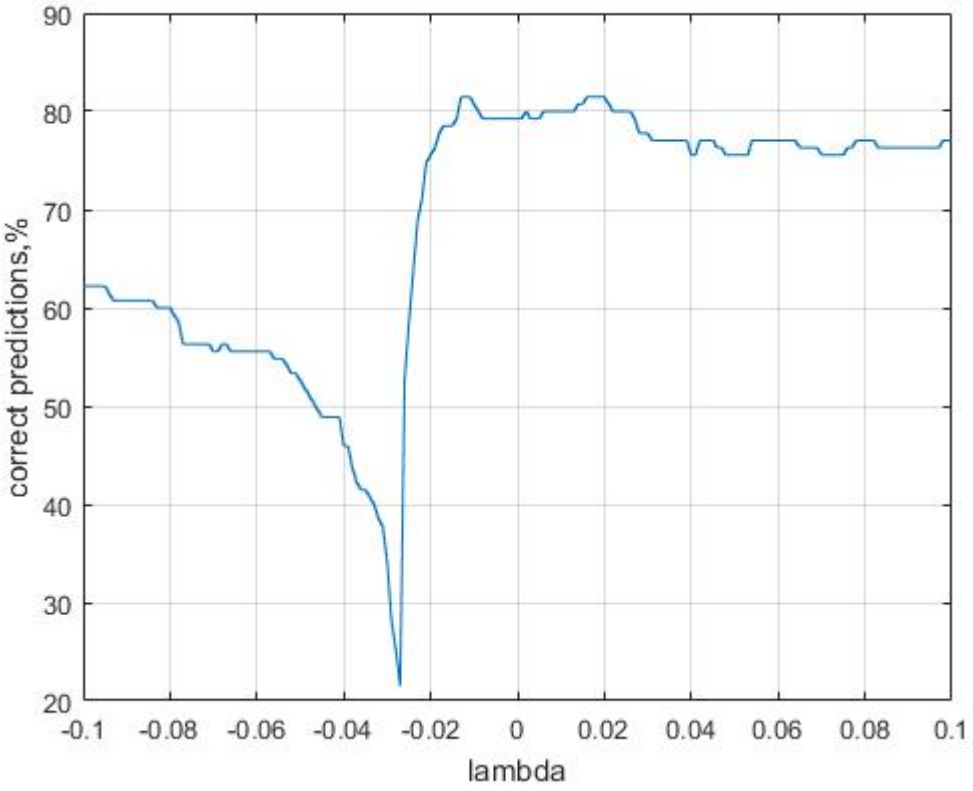

Figure 1 Iris. Correct predictions depending on the regularization parameter λ

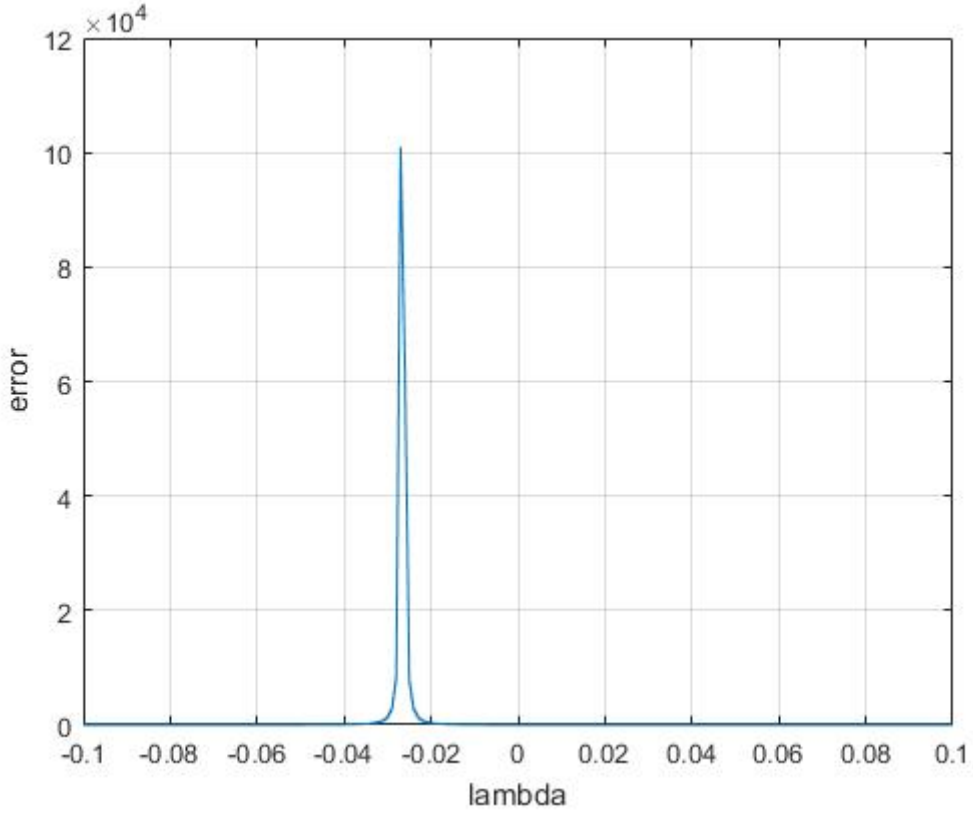

Figure 2 Iris. Error depending on the regularization parameter λ

Let's zoom the last curve

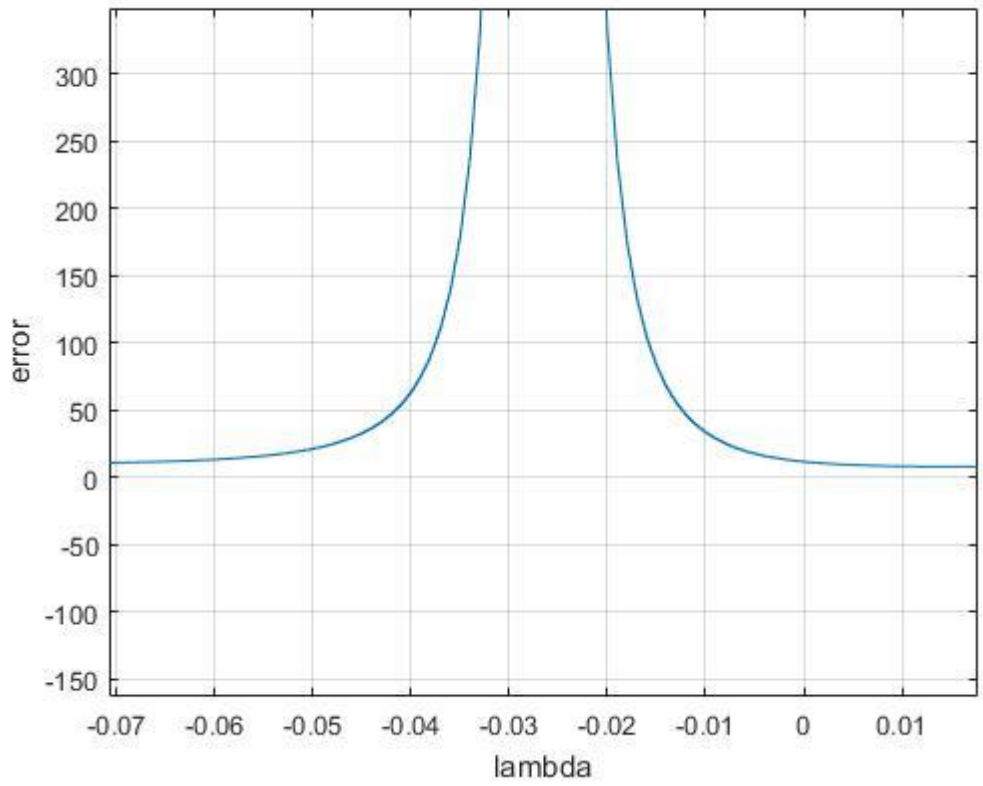

Figure 3 Zoomed curve error depending on the regularization parameter λ

As we can see the error curve has a high peak near zero, which divides both graphs into two parts. We are interested in the right part only because there is a maximum of correct predictions. Moreover in some cases left part can be less than right, and we will search for optimal λ in the wrong place. To prevent this, to begin with, search for peak, and after that search minimum of error curve to the right of the peak. The regularization parameter λ can be both positive and negative

Let's check the same criterion for all data sets

Wine data set

Correct predictions (Fig.4) and error (Fig.5) depending on the regularization parameter λ for Iris data set and 10% training data. Matlab file: classifier_Wine_plot_curves.m

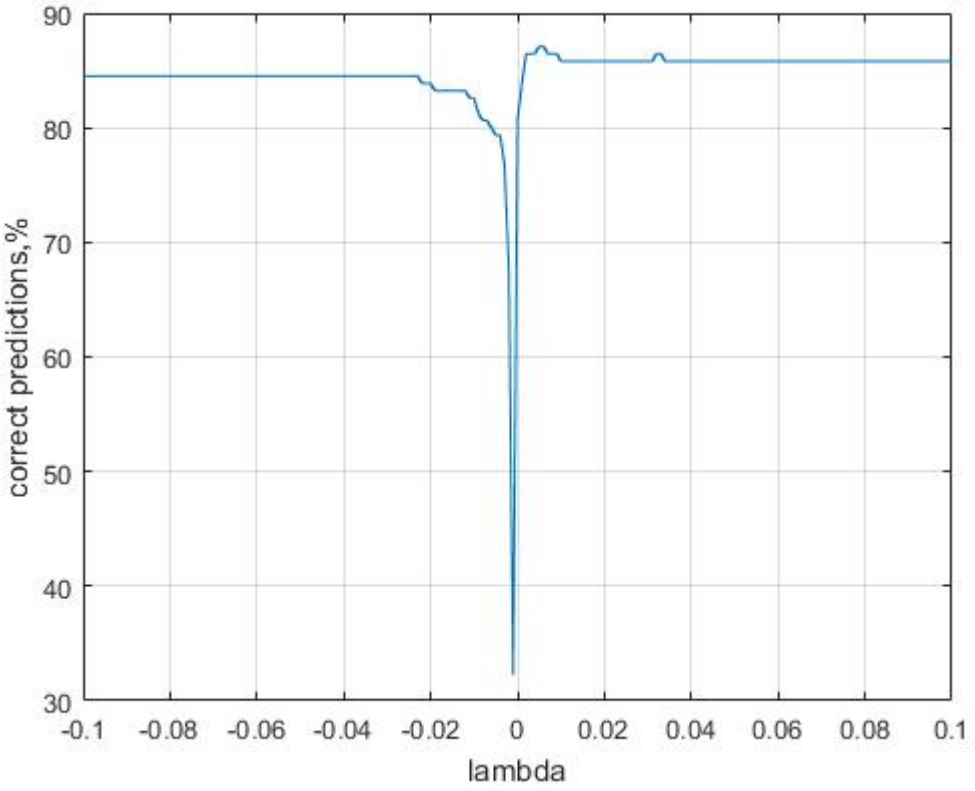

Figure 4 Wine. Correct predictions depending on the regularization parameter λ

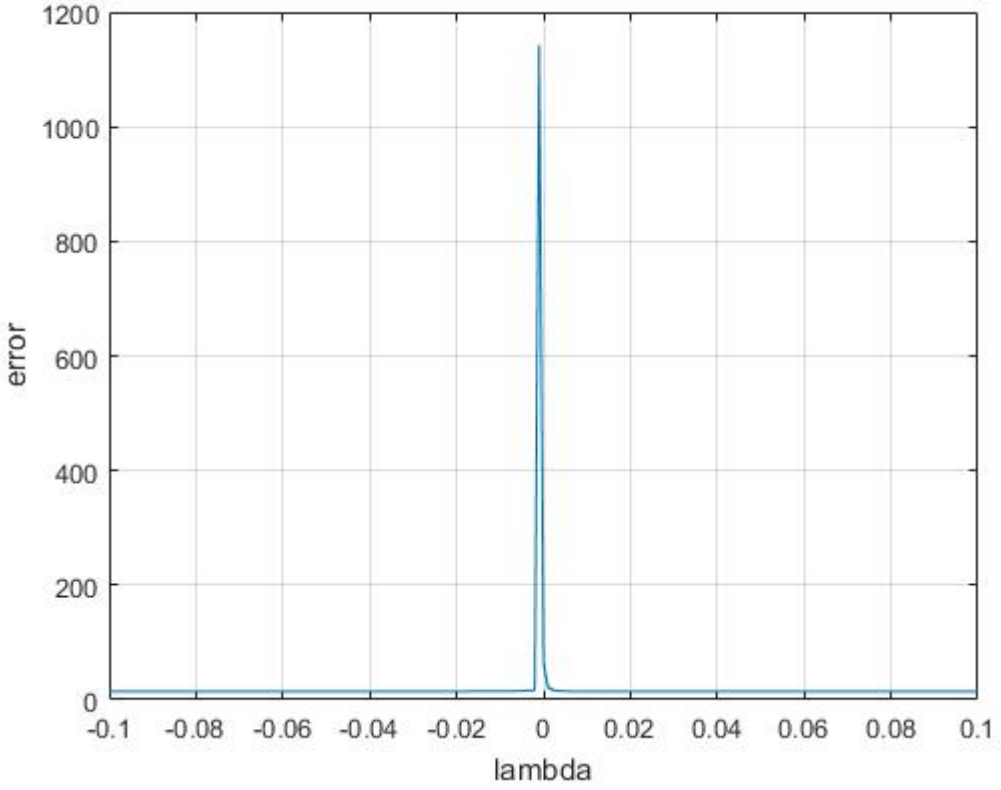

Figure 5 Wine. Error depending on the regularization parameter λ

As we can see, the curve's shape is identical to Iris curves

Glass data set

Correct predictions (Fig.6) and error (Fig7) depending on the regularization parameter λ for Iris data set and 10% training data. Matlab file: classifier_Glass_plot_curves.m

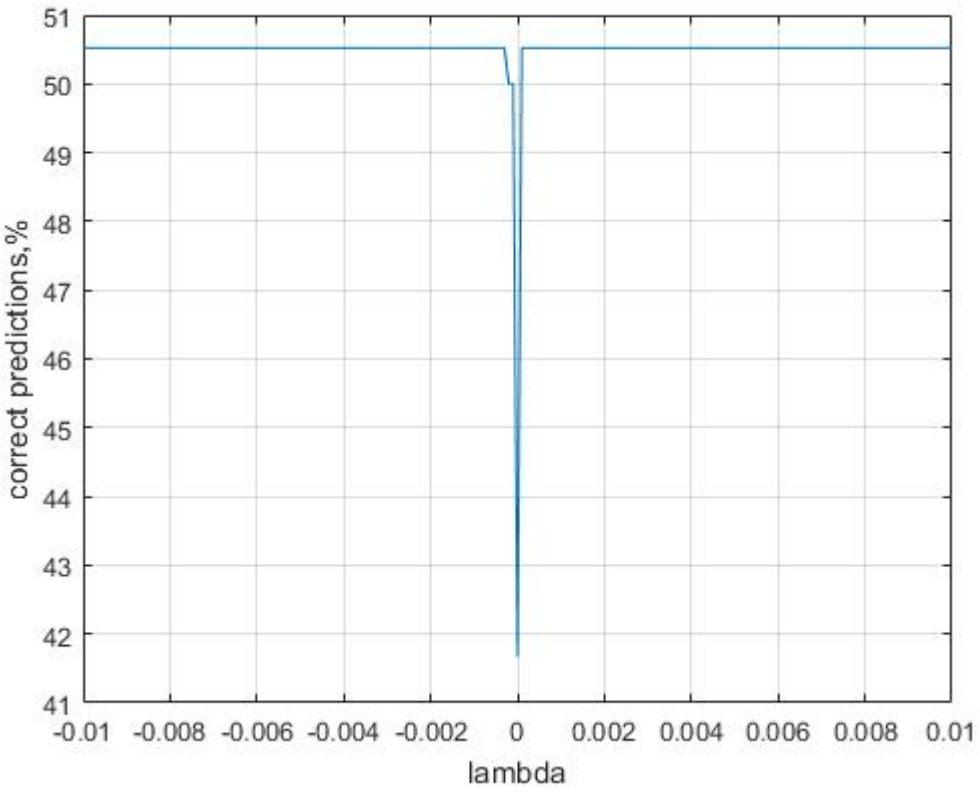

Figure 6 Glass. Correct predictions depending on the regularization parameter λ

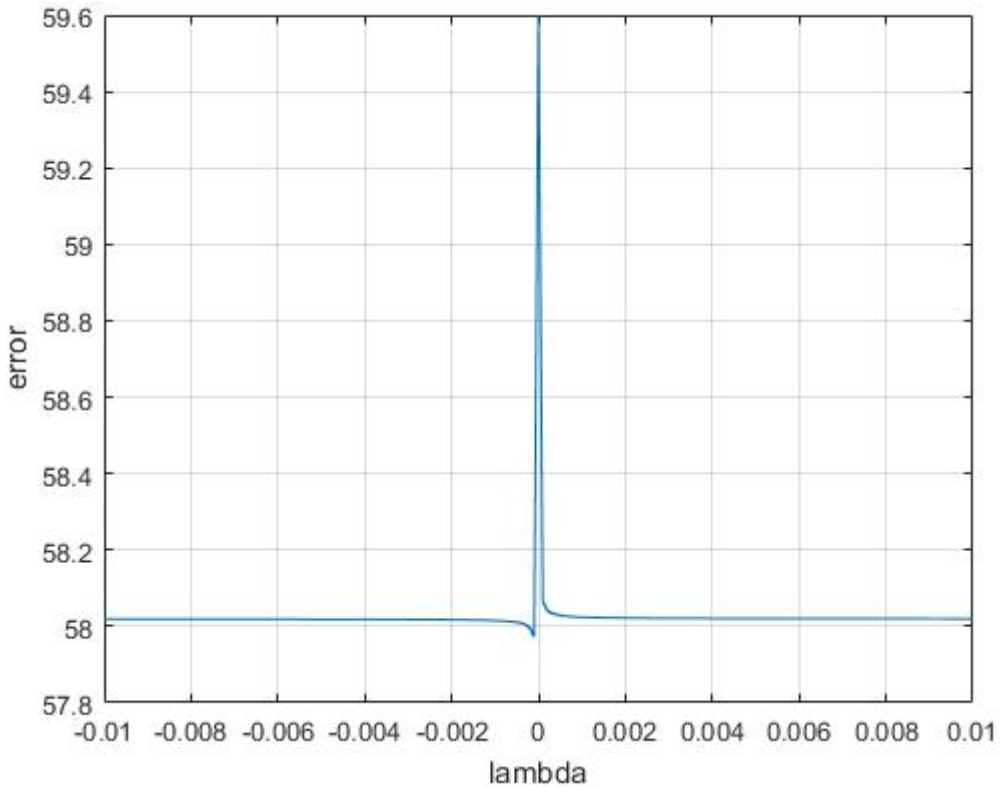

Figure 7 Glass. Error depending on the regularization parameter λ

As we can see, some of it differs from Iris and Wine curves. It has a constant correct prediction ration independent on λ

Haberman data set

Correct predictions (Fig.4) and error (Fig.5) depending on the regularization parameter λ for Iris data set and 10% training data. Matlab file: classifier_ Haberman _plot_curves.m

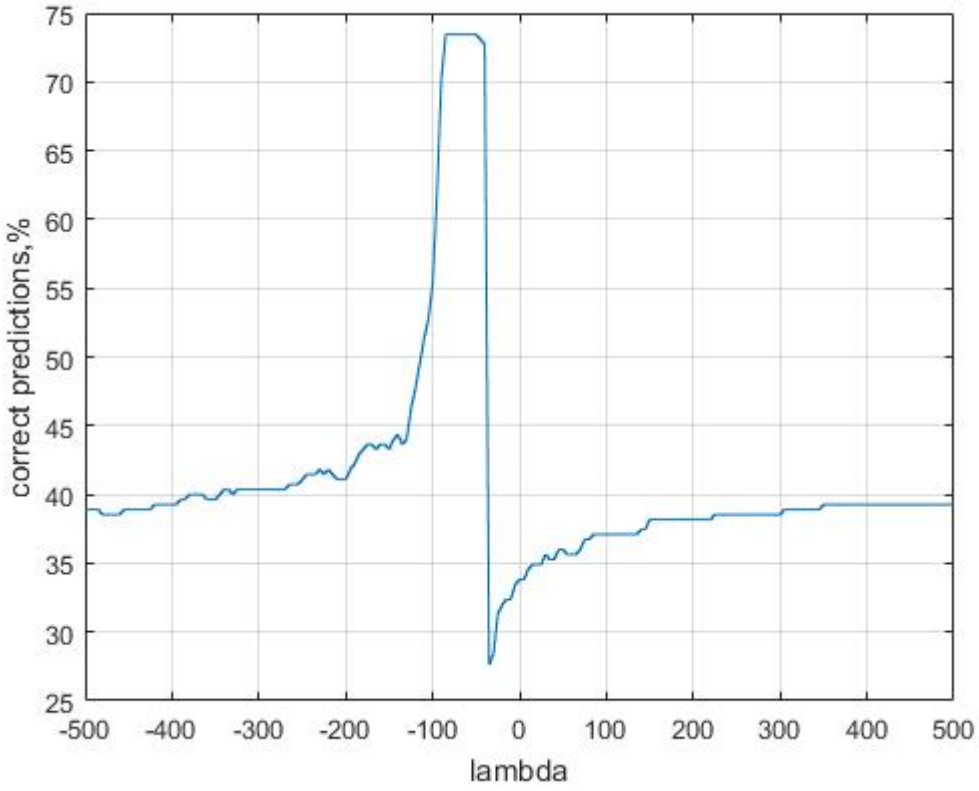

Figure 8 Haberman. Correct predictions depending on the regularization parameter λ

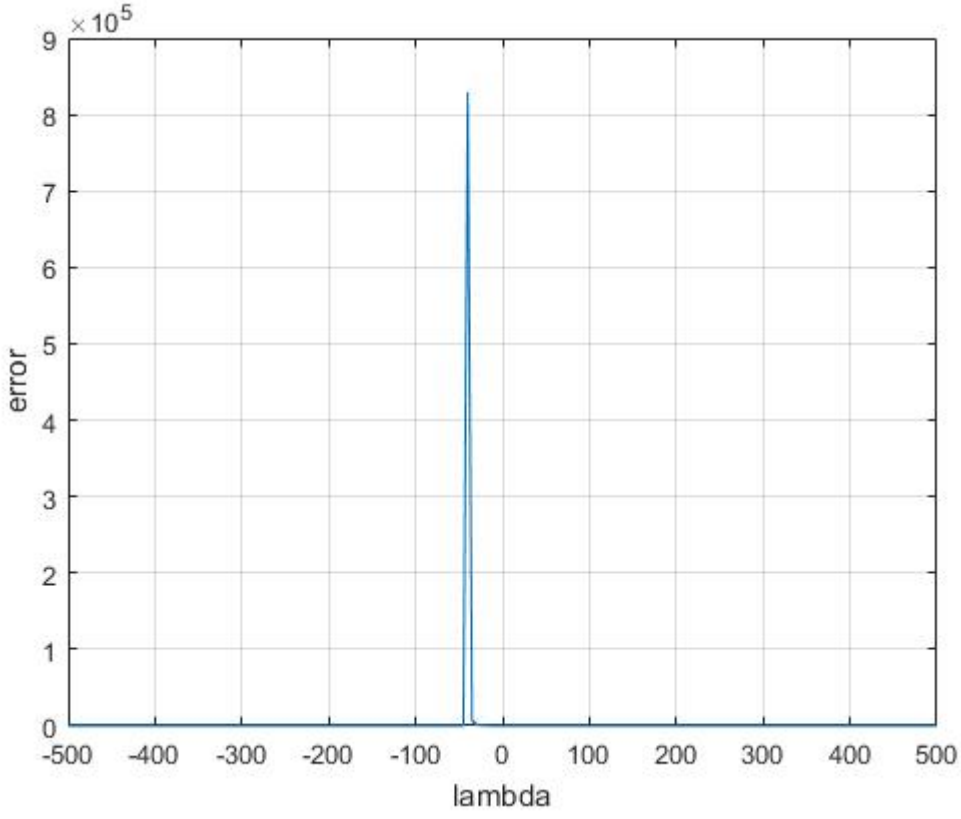

Figure 8 Haberman. Error depending on the regularization parameter λ

It very differs from other data set curves. At first, it has huge λ scale. At second, the maximum prediction is on the left part of the curve. It looks like the Iris curve mirrored along the x-axis.

Let's plot a curve for random separation. Matlab file: classifier_Haberman_plot_curves_random.m

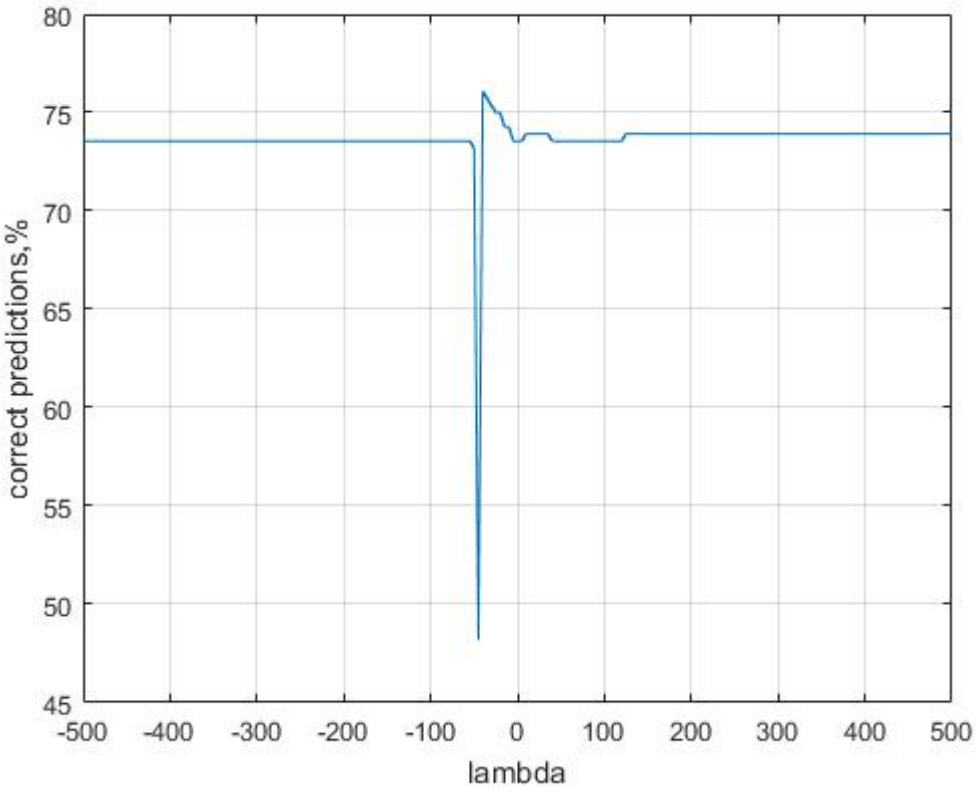

Figure 9 Haberman. Random separation. Correct predictions depending on the regularization parameter λ

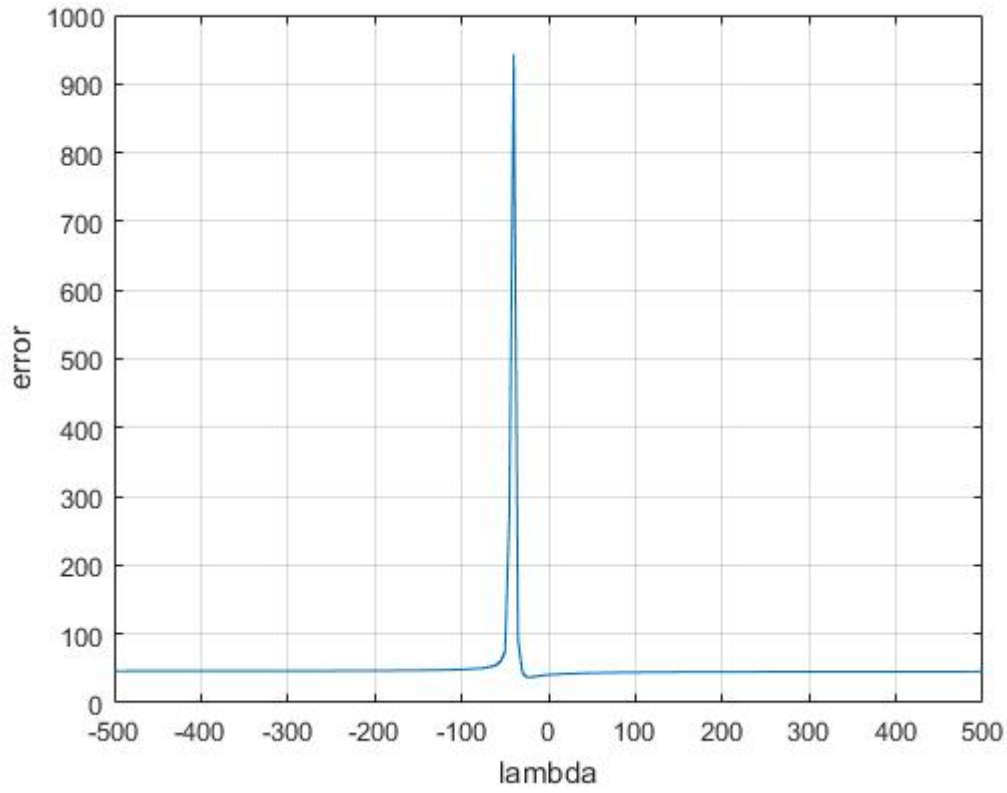

Figure 10 Haberman. Random separation. Error depending on the regularization parameter λ

As we see the situation has changed dramatically. Now prediction is on the right part of the chart. We have no choice, we need to calculate both left and right part minimum error and then compare correct predictions numbers for both lambdas

Also, we need to adjust the scale automatically. Our aim is to research machine learning not make an absolute automatic program, so we still analyze data in semi-manual / semi-automatic mode. So we simply add scale to function arguments.

DELIVERABLE 3

To estimate both methods random subsampling cross-validation used. For reliable comparison, both methods execute with the same random Training/Testing data partition per every iteration. The number of iterations with a different random partition for one Training/Testing ratio is 10

Result for every dataset present as table and chart for a set of Training/Testing ratio

Iris

Matlab file: run_classifier_Iris.m

Percent _train	Percent _test	Correct_ test	Total _test	Correct_percent	Correct_test_ SVM	Correct_percent_ SVM
10	90	112.2	138	81.304	132.8	96.232
20	80	102.5	126	81.349	122.1	96.905
30	70	91.4	111	82.342	108.2	97.477
40	60	81.5	99	82.323	96.4	97.374
50	50	70.8	87	81.379	85.1	97.816

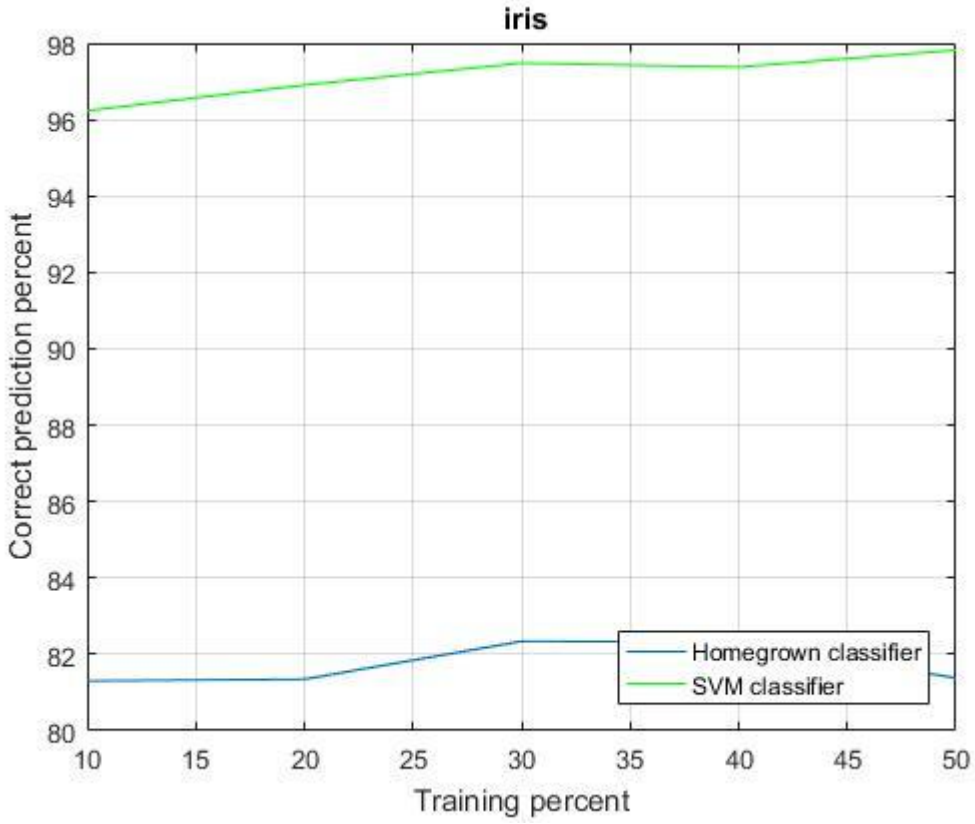

Figure 11 Iris. Comparing prediction accuracy both LSM and SVM methods

Wine

Matlab file: run_classifier_Wine.m

Percent _train	Percent _test	Correct_ test	Total _test	Correct_percent	Correct_test_ SVM	Correct_percent_ SVM
10	90	146.6	157	93.3758	142.9	91.01911
20	80	133.7	141	94.8227	132.1	93.68794
30	70	121.8	127	95.90551	118.3	93.14961
40	60	107.9	112	96.33929	107.4	95.89286
50	50	94	97	96.90722	9.7	96.59794

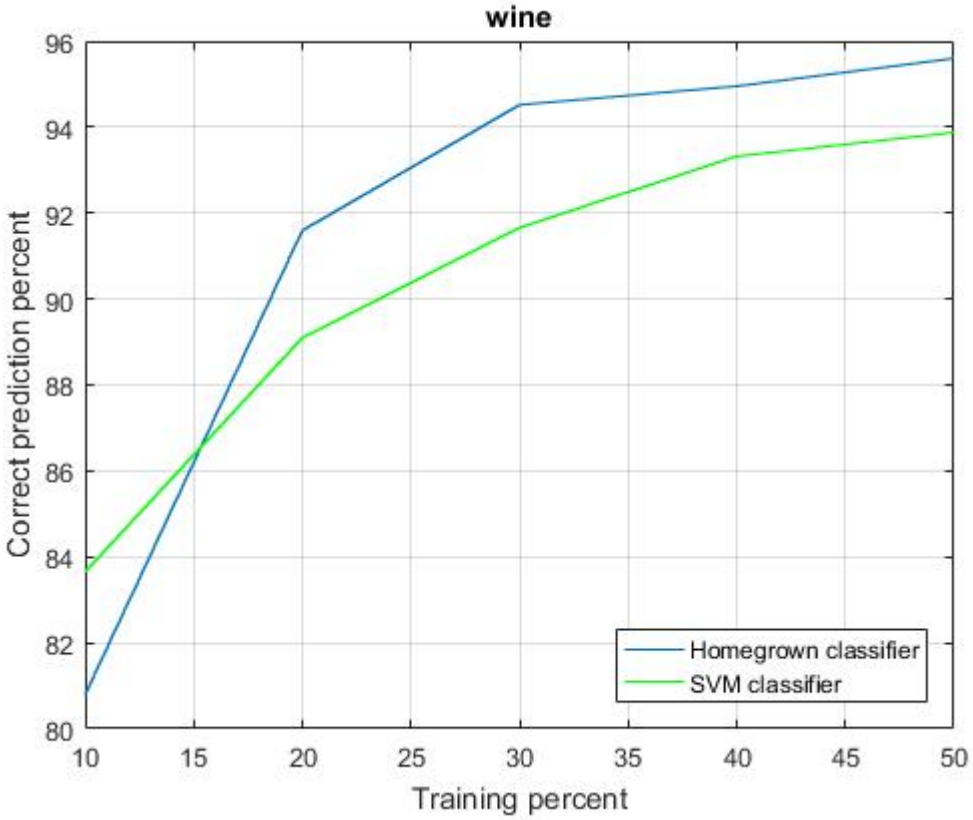

Figure 12 Wine. Comparing prediction accuracy both LSM and SVM methods

Glass

Matlab file: run_classifier_Glass.m

Percent _train	Percent _test	Correct_ test	Total _test	Correct_percent	Correct_test_ SVM	Correct_percent_ SVM
10	90	113.7	195	58.308	114.6	58.769
20	80	98.4	177	55.593	105.8	59.774
30	70	90.1	158	57.025	99.7	63.101

40	60	79.6	141	56.454	88.9	63.05
50	50	72.8	121	60.165	77	63.636

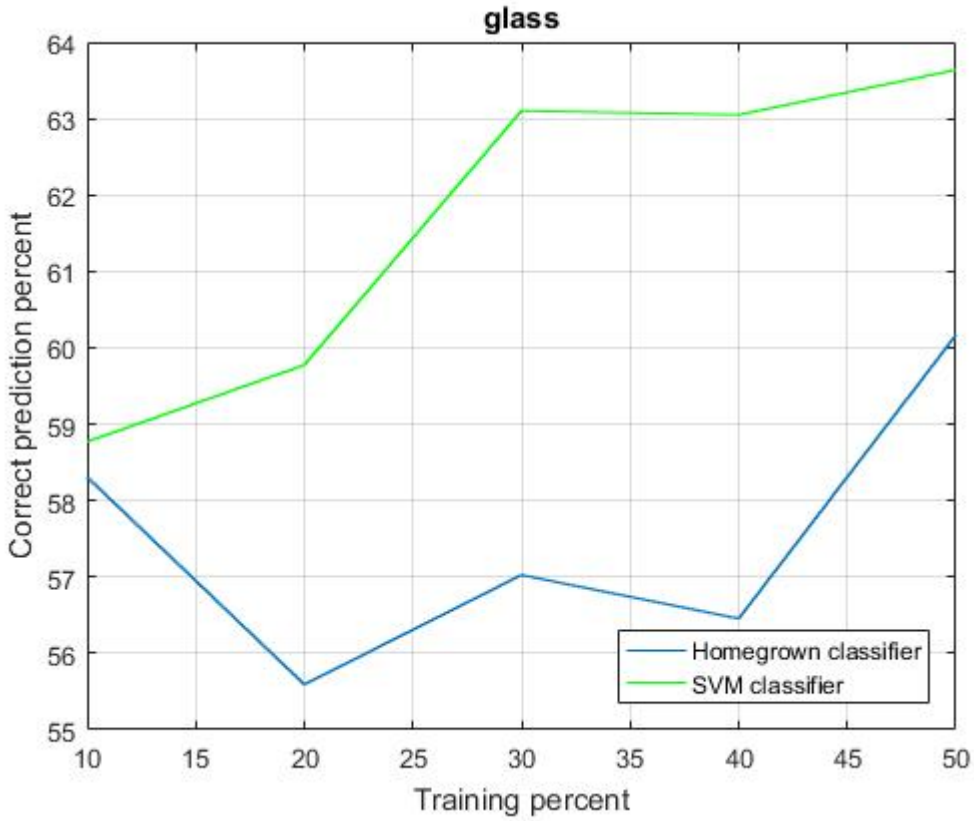

Figure 13 Glass. Comparing prediction accuracy both LSM and SVM methods

Haberman

Matlab file: run_classifier_Haberman.m

Percent _train	Percent _test	Correct_ test	Total _test	Correct_percent	Correct_test_ SVM	Correct_percent_ SVM
10	90	204.4	276	74.058	197.1	71.413
20	80	184.2	246	74.878	182.4	74.146
30	70	162.7	215	75.674	158.3	73.628
40	60	139.8	185	75.568	136.2	73.622
50	50	114.8	153	75.033	111.8	73.072

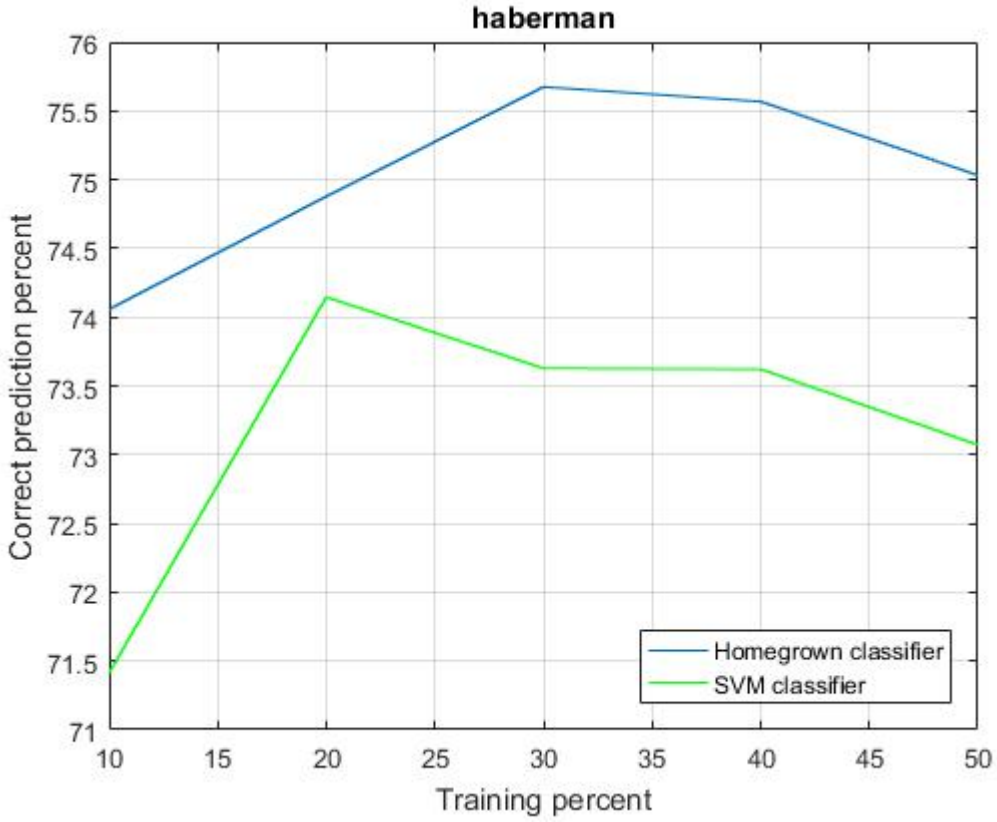

Figure 14 Haberman. Comparing prediction accuracy both LSM and SVM methods

Conclusion

How we can see on graphs, no one method gives a definite advantage. With some data set LSM is better; with some data sets SVM is better

Multiclass SVM discussion

Existing multiclass SVM methods can roughly be divided between two different approaches: the single machine approach, which attempts to construct a multi-class SVM by solving a single optimization problem, and the “divide and conquer” approach, which decomposes the multi-class problem into several binary sub-problems, and builds a standard SVM for each. The second approach, in turn, divides into two different strategies:

1) The “one against all”, which consists of building one SVM per class, trained to distinguish the samples in a single class from the samples in all other classes.
2) The “one against one”, which builds one SVM for each pair of classes.

In practice, both strategies provide good results, and both using used widely.

List of attached Matlab files

Those files plot curves for DELIVERABLE 2. They are standalone scripts

- classifier_Iris_plot_curves.m
- classifier_Wine_plot_curves.m
- classifier_Glass_plot_curves.m
- classifier_Haberman_plot_curves.m
- classifier_Haberman_plot_curves_random.m

Those files start SLM and SVM method comparison with cross-validation. One file for one data set. They are standalone scripts

- run_classifier_Iris.m
- run_classifier_Wine.m
- run_classifier_Glass.m
- run_classifier_Haberman.m

This file runs the one classification iteration many times for every Training/Testing data ration and cross-validation. It's a function script

- classifier_table.m

This file executes the one classification iteration. It's a function script

- classifier

This file search lambda for minimum error value on one part of the error curve. It's a function script

- lambda_search.m

Data set files prepared to machine learning

- iris_formatted.data
- wine.data
- glass_formatted.data
- haberman_formatted.data