EINDHOVEN UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF BIOMEDICAL ENGINEERING

Computer-aided diagnosis

8DC00 Medical Image Analysis

J.P. van Ingen 1019503
E.G. van Pelt 1342592
T.R.S. Tuerlings 1018213
H. Versluijs 1023803

Group 21

Contents

1 Introduction

Breast cancer is besides skin cancer the most common type of cancer in women. This year in the United States there are 276,480 and 48,530 woman diagnosed with invasive and non-invasive breast cancer respectively. Not only woman, also around 500 men are diagnosed with the invasive type. Invasive means that the tumor grows into healthy tissue, whereas a non-invasive tumor does not grow into tissue. The survival rate of a patient with non-invasive cancer is around 91% over 5 years. For an invasive tumor this is around 84%. But if the cancer has spread out in other parts of the body the survival rate over five years is 27% [1].

Breast cancer is characterized by the size of the sell nuclei of the tumor. These sizes are evaluated by pathologist. The current routine requires a qualitative measurement, by a tissue analysis with the aid of a microscope. However, a quantitative evaluation of the size would be much more useful to diagnose breast-cancer. Unfortunately, this takes too much time to do manually. On the basis of the nuclei size a diagnosis is set. The bigger the nuclei of a tumor cell, the more aggressive the tumor is [2]. A disadvantage of this manual evaluation is could result in inter-user variability. This means that different pathologist might give a slightly different indication of nuclei size, which leads to different diagnoses being made. Therefore a computer is preferred. Firstly, because it can calculate nuclei size quantitatively, which allows for a more accurate diagnosis. Performing this quantitative evaluation by a computer will be much faster than doing it manually. Another advantage is that is done automatically so there is no inter-user variability, which means the results will have greater precision.
While this project focuses on the application of evaluating breast tumors, the used methods and algorithms can also be used in other clinical settings. A few clinical examples that also require size measurements are measuring the left ventricular size, or measuring vessel diameters for diagnosing aneurysms. The logistic regression method used for classification is useful for diagnosing all kinds of things. Think for example of diagnosing an aortic stenosis or retinopathy. The only problem in analysis of these medical diseases is the availability of ground truth data, which is necessary to provide a training set in the supervised models that are used in this project. Obtaining this ground truth data can be done by letting several pathologists segment a set of nuclei manually, to obtain the sizes. The inter-user variance can be computed to evaluate the quality of these ground truths.

The goal for this project is to design an automatic method for measuring nuclei size using a linear regression model. After that a classification model is made to classify the nuclei size into large and small. This is done by a logistic regression model.
The linear regression model is trained two times, once for the total training set and once for only a part of the training set. It is expected that the model will be trained better with using more data, since there is less chance of over-fitting the training set. Thus, the error between the predicted area and the ground truth will be larger when using only part of the training set. For the logistic regression model, the hyper-parameters of the model are initialized in different ways. Thus, the effects of different initial values will be analyzed and discussed, and the optimal values will be found. The classification accuracy of a model trained with these optimal parameters will be reported. Also the effect of using a smaller training set will be examined.

This project makes use of a Python script, written within the Spyder environment. With the use of GitHub repository: MIA-group-21, the code was easily accessible for all the group members.

2 Method

2.1 Data

The dataset that is used contains a set of small microscopy RGB images of histopathology slides of a nuclei. The images have a size of 24 x 24 pixels. For linear regression, there are 20730 images in the test set, and 21910 images in the training set. A validation set is not needed, since only one model is being trained, that doesn't require an optimal set op (hyper)parameters, so there is no need to compare performance of different models (for which the validation loss is typically used). For logistic regression, the training set has been further split into a validation set

and a smaller training set. Here there are 14607 train images, 20730 test images and 7303 validation images used.
All the images are derived from Veta et al. (2015) [3].

2.2 Linear regression

Before performing a linear regression model, the data of the nuclei sizes are sorted to get the 300 smallest nuclei sizes and the 300 largest nuclei sizes. The corresponding images are then visualized in two plots.
After that, the data is prepared for linear regression. The images are reshaped to obtain feature vectors. Every row represents an image and every column represents a feature. The feature set consists of the the raw pixel values for the width and height of the image, for every color channel of the (RGB) image, so there are 24 x 24 x 3, 1728 features.
With the use of these feature vectors, a linear regression model is implemented. The linear regression parameters were computed with the use of the total training set and a second time with the use only one-fourth of the training set. For both sets of parameters, the area size of the nuclei of the test set was predicted. The error was calculated using equation 1. To normalize the data size the division step is taken.

$$J = \sum_{i} \frac{(predicted_y - test_y)^2}{(test_x)_{shape}[0]} \tag{1}$$

The predicted area vs the ground truth area of the training set, for both the total training set and one-fourth of the training set, are visualized in a plot.

2.3 Logistic regression

To classify the nuclei sizes a logistic regression model is trained. Two classes are made: ”large” with the label $y = 1$ and ”small” with the label $y = 0$. The model aims to classify each nucleus in one of these classes. First, the datasets are reshaped into feature vectors in the same way as for linear regression. The only difference here is that the features are normalized, by scaling and recentering the feature vectors with regard to the training set. Then a logistic regression model is trained using the implemented values for the learning rate (mu), the batch size, number of iterations, as well as the initial values for the model parameters (Theta). Each iteration, the set of model parameters Theta is updated with the gradient of the loss function. To do this, the analytical expression of the gradient is used, which speeds up the process. The negative log-likelihood is calculated using equation 2, with p the sigmoid function applied to training X and Theta.

$$J = \sum_{i=1}^{N} train_y * log(p) + (1 - train_y) * log(1 - p) \tag{2}$$

After each iteration, the validation loss is computed and plotted together with the training loss to visualize the model performance.

The initial values for mu, batch size, number of iterations and Theta are selected by trial and error, trying different combinations and looking at the plots of training and validation loss. The mu is the learning rate, also called the step-size. This determines how much theta is increased or decreased each iteration, depending on the gradient. The batch size controls the number of training images that is used to calculate the gradient. Using only part of the training set to compute this gradient, means the stochastic gradient descent method is used. This method is used to speed up the calculations. The number of iterations tells how often the gradient is computed and thus how often the Theta is updated, before the model will stop training. Choosing initial values for Theta requires some experimenting, since the values must be close to the optimal values, to allow for a fast and reliable training of the model. In the results, the optimal set of these parameters will be chosen, that is: the set where the validation loss is the lowest.
Having thus trained the model as well as possible, its classification accuracy will be determined. This is done by trying the model on the provided test set and reporting the test loss.

The final experiment will be to evaluate the effect of the size of the training set. The model will be trained for different sizes of the training set (a large training set, and various reduced training sets). The training losses will be reported, together with the validation losses, to check for over-fitting.

2.4 Code

For this project, a Python script has been written. This script, called cad_project.py, contains two functions;

1. Nuclei_measurement.
   Nuclei_measurement is used to perform a linear regression model which calculates the area of the cell nuclei, by using the ls_solve function from registration.py. The function nuclei_measurement consists of two parts, in the first the whole trained model is used to predict the area of the test dataset, whereas in the second part only every fourth training sample is used.

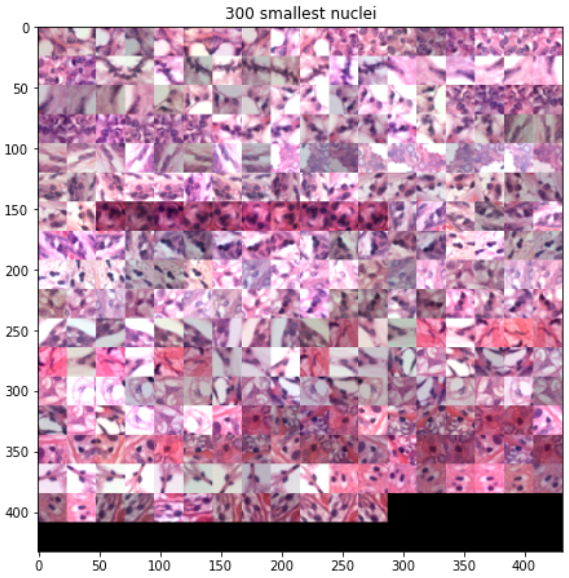
2. Nuclei_classification.
   This function is used to train a logistic regression model to classify the nuclei into either ”large” or ”small”. The logistic regression model makes use of an analytically expression of the gradient of the loss function, using the cad.r_agrad function from cad.py, to train the model faster. The losses are computed with the negative loglikelihood function implemented in cad.lr_nll. Furthermore, different initial values for the parameters of the training process are set to compare the effects of varying these.

3 Results

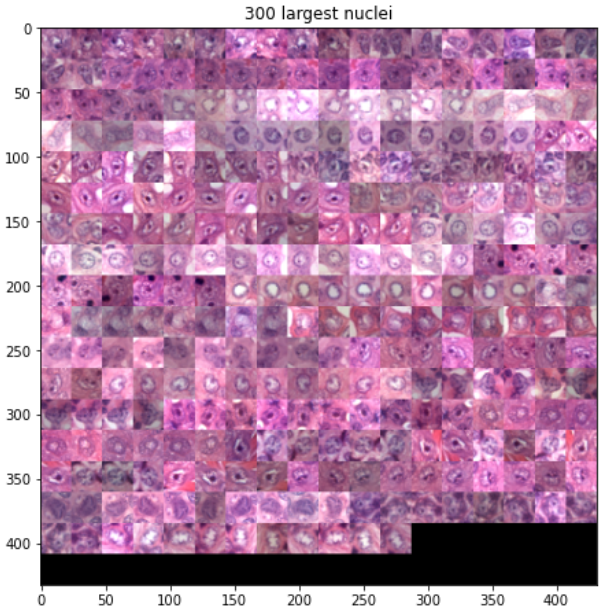
3.1 Linear regression

After running the code: nuclei_measurement(), a graph of the 300 smallest and 300 largest nuclei from the training dataset was obtained. This can be seen in figure 1.

Furthermore, the predicted area of the nuclei using full sample of the training set is visualized in comparison to the predicted area of the nuclei using a smaller training sample, namely every fourth training sample. The results of the are shown in figure 2.
The graph shows a similar kind of distribution of the points. However, the second graph, 2b, shows more outliers compared to the first graph, 2a. This can for example be seen from the fact that there are more data-points around the predicted area of 200 in graph 2b. The test error of both sets are calculated, the test error of the



(a) The 300 smallest nuclei from the training dataset.



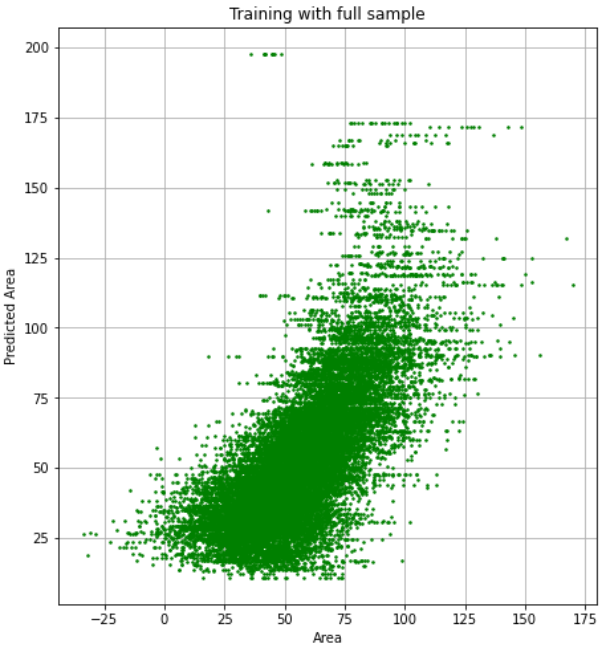
(b) The 300 largest nuclei from the training dataset.

Figure 1: Graph of the nuclei

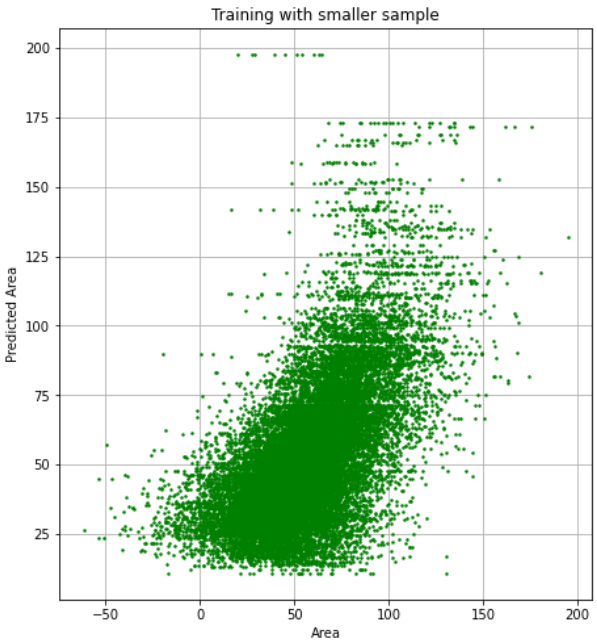
full sample of training set is 374.9, and for the smaller training set the error is 592.1.

3.2 Logistic regression

Several experiments have been done. First, the effect of the parameters mu, the batch size and the number of iterations was tested. The loss functions of different initial settings can be seen in figures 3, 4 and 5. First, for the lower learning rates, mu = 0.001 and 0.0005 (Fig. 3a,3c), both loss functions have a peaky appearance. Whereas for the learning rates mu = 0.0001 and 0.00001 (Fig. 3b,3d), the validation loss function has a more smoothly progress. For the



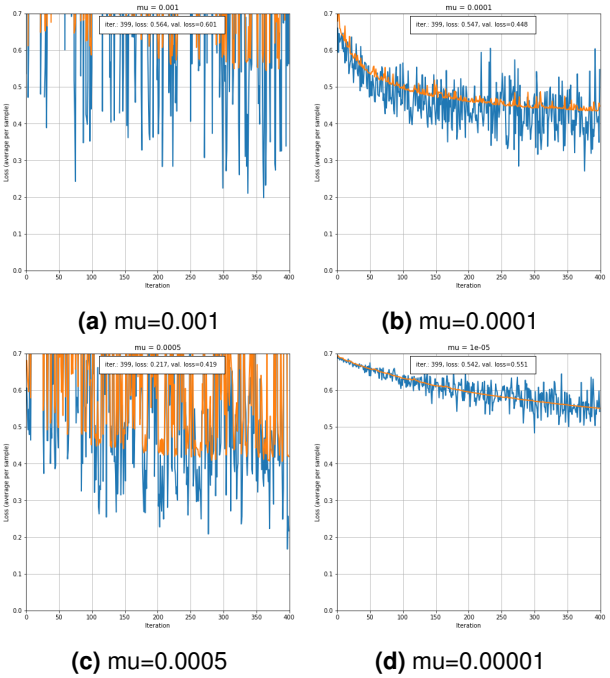
(a) The predicted vs the true area of the nuclei for the full sample of the training set.



(b) The predicted vs the true area of the nuclei for every fourth training sample of the training set.

Figure 2: The two trained models

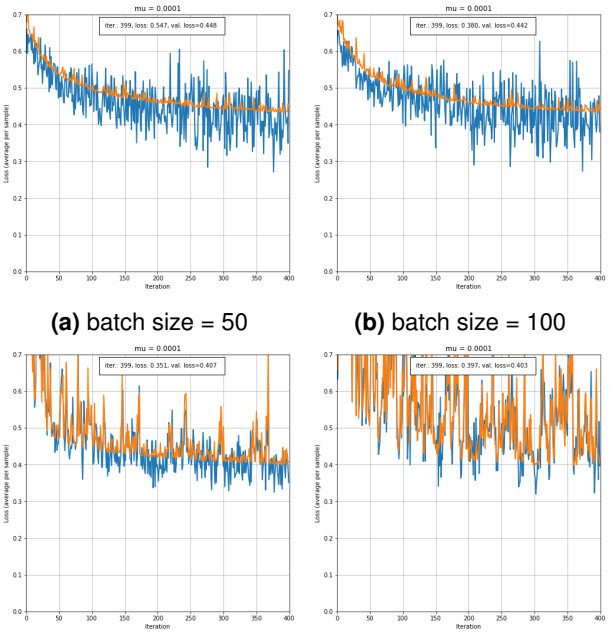
batch sizes, the validation loss is smooth for 50 (Fig. 4c) and 100 (Fig. 4d), and peaky in the figures 4c and 4d for batch sizes 150 and 200 respectively. When varying of the amount of iterations, it can be notified that the figures 5c and 5d the graphs do not look very differently from each other. Only the training loss has a more roughly peaky appearance in figure 5d. However, for both iterations (1000 and 4000) the training loss function ends around 0.7. For an amount of 40 iterations, fig. 5a, both the training and validation losses still have a decreased slope at the end of the graph. Also, the losses have decreased slightly.



(a) mu=0.001 **(b)** mu=0.0001

(c) mu=0.0005 **(d)** mu=0.00001

Figure 3: The validation loss with varying mu

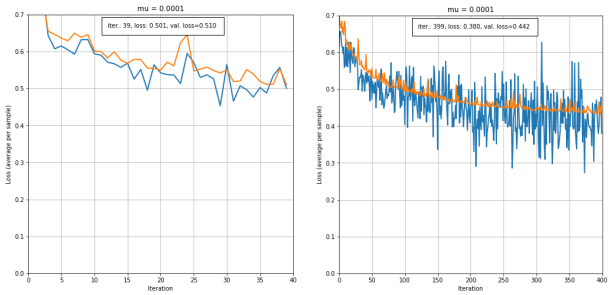


(a) batch size = 50 **(b)** batch size = 100

(c) batch size = 150 **(d)** batch size = 200

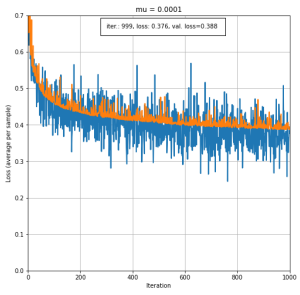
Figure 4: The validation loss with varying batch size

Then the best performed initial values were searched. All results can be found in the appendix A.1. First the parameters was fixed to; number of iterations = 300, mu = 0.001, batch size = 30, and the optimal value for Theta has been found. To do this, a rough search has been performed in

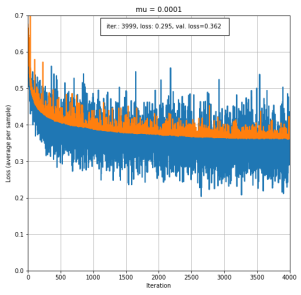


(a) 40 iterations

(b) 400 iterations



(c) 1000 iterations



(d) 4000 iterations

Figure 5: The validation loss with number of iterations

advance, where the Theta array of ones is multiplied to a number in the range of 0.001 to 10, giving various Theta initial arrays.The best results, the result with the lowest validation losses, were found for theta = 0.001 (validation loss = 0.566) and theta = 0.4 (validation loss = 0.462). For higher values of theta, often no validation loss could be computed (NaN), (appendix A.1, 2). Then, a more sophisticating research is performed around those values. The Theta of ones is now multiplied by; 0.001 to 0.009 and 0.3 to 0.5. The best result was found for theta = 0.003 (validation loss = 0.459). However, there is very little difference compared to theta = 0.009 (validation loss = 0.481 or theta = 0.3 or 0.008 (both with a validation loss = 0.487), (appendix A.1, 3).

Next, this optimal value theta = 0.003 has been used in the search for the optimal value for the number of iteration. This has been done varying in the range of 300-700. The results are in the appendix A.1, 4. The best result, looking from the highest number in the range, is found for iterations = 650 (validation loss = 0.448). Looking at the lowest numbers in the range, the best result is obtained for iterations = 300 (validation loss = 0.459).

Now the optimal initial values of Theta (0.003) and the number of iterations (300) are used to search mu. Therefore, mu is varied from 0.00001 to 0.1 by increasing with a factor of 10. The best result is for mu = 0.001, which has a validation loss of 0.468 (appendix A.1, 5). After the optimal mu is set, only the batch size with the best performance needs to be found. This is done by fixating the other optimal values and varying the batch from between 10 and 100. The best result was found for batch = 20 (validation loss = 0.464). So, the optimal set of initial values are: iterations = 300, mu = 0.001, batch size = 20, and a Theta consisting of an array with initial values 0.003.

Using these values, the classification accuracy is computed as 0.497.
Finally, the effect of using a smaller training set has been investigated. The results are shown in Table 1.

Table 1: Results of reduced training sets

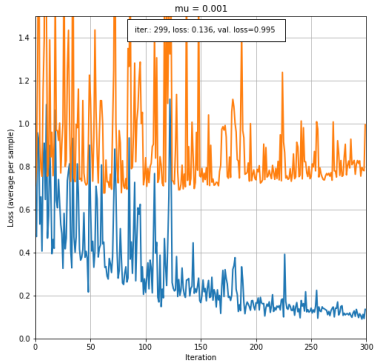
Reduction by	Training loss	Validation loss
0.5%	0.136	0.995
5%	0.034	0.721
50%	0.150	0.512

The loss functions can be seen in the images, 6a, 6b and 6c, were the blue graph is the training loss and the orange graph shows the validation loss. It is noticeable that for the reduction of 0.5%, the training loss is still declining and the validation loss function is slightly increasing (6a). For reducing with 5%, the losses consist of less peaks and the training loss is almost 0.0 and shows no decline. By the reduction of 50% (6b) of the dataset, the graph shows a noisy appearance, where the training loss function peaks through the peaks of the validation loss.

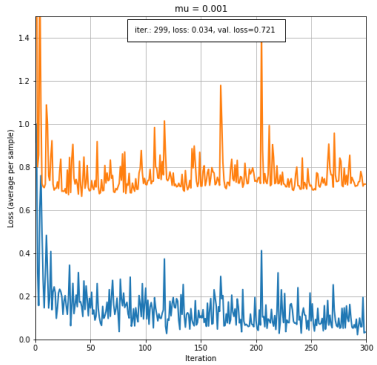
4 Discussion

4.1 Linear regression model

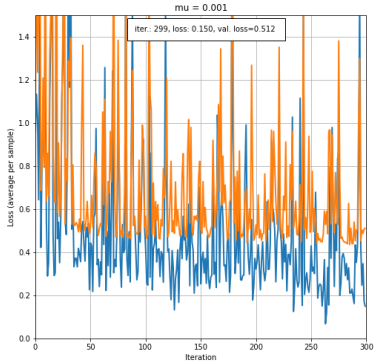
The error for the predicted area vs the ground truth with the model trained using the total training set was 374.9 and with the model trained using one-fourth of the training set was 592.1.
Next to the errors, it was visualised in the plots of the predicted area vs the real area that the points were least scattered when the whole training set was used. Therefore, it can be concluded that using more data will result in a more



(a) reduction by 0.5%



(b) reduction by 5%



(c) reduction by 50%

Figure 6: Effects of reducing the training size

accurate prediction. This corresponds with the hypothesis mentioned in the introduction. However, when using more data it will take more time to train the model. This is something that needs to be taken into account.

4.2 Logistic regression model

Discussion of loss curves It is important to set the right initial values for mu, the batch size, the number of iterations and for the parameter Theta. For mu (Fig. 3) it can be seen that a smaller learning rate will not be able to reach the solution in a reasonable time (Fig. 3d), whereas a bigger the learning rate here shows a more optimal effect. When the learning rate is to large, it is not effective at all, as it 'jumps' across the solution and never hits the target properly. When varying the batch size (Fig. 4), it can be seen that the large the batch size is, the more expensive it is to compute. The number of iterations (Fig. 5) is also important as the solution needs to be found in this range. When the number of iterations is large, a smaller number is preferred if this reaches the same outcome. When there are too few iterations, the optimal solution is not reached (Fig. 5a).

The acquisition of mu, the batch size and the number of iterations for the actual results were set as was described in the results section. When mu is set too low, the minimum of the loss function cannot be found within a small number of iterations. It calculates the gradient for small steps, so it will take a long time and amount of iterations before the minimum of the loss function is found. On the other hand, when the mu is set too high, it will lead to faster convergence to the lowest error.

However, this can result in a unstable curve were the optimal solution will be missed. The parameter Theta needs to be updated more within the amount of iterations. In the results it was found that setting mu too small (0.00001) lead to a higher validation loss. This means that the optimal value of Theta couldn't be reached in time, since the steps are too small.

An increased batch size results in a slower model. This is because of the fact that for a larger training data set, the gradient descent is calculated. However, at the same time this means that the updating of theta is more accurate, so the function will faster converge.

A low number of iterations is computationally less expensive so reduces the training time, however a disadvantage is that it may cause the training process to stop before the optimal solution is reached. By increasing the number of iterations, the loss function will descent. Nevertheless, the computer needs to calculate more

gradients and do more updates on the parameter Theta. Therefore, it will take more time when having more iteration steps. It was thought that more iterations lead to a better performance of the model, however there was no big difference, for the amount of iterations of 650 and 300, the validation losses were 0.448 and 0.459 respectively.

Discussion of initial values After running and evaluating the code, it can be concluded that the initial values should be set on: iterations = 300, mu = 0.001, batch = 20, theta = 0.003. It is important to note however, that these values aren't set in stone. Running the model twice with the same initial values, results in different values for the training and validation loss. This is due to the fact that the batch of examples, used to update Theta, differs per iteration and thus with each training of the model. So, the performance of the model could be slightly different, which lead to a slightly different outcome in the validation loss. Another discussion point is that now only the validation losses have been used to compare between different sets of values. However, since the loss curves can be quite peaky, it would have been wise to compare these curves as well. This would have led to e.g. a lower value for mu, which reduces the peaks (as described before).

Instead of using a set number of iterations to train the model, also a stopping condition could be used. This stopping condition could be defined as the validation error being below a certain value, e.g. 0.5, depending on how well the model is supposed to be trained. For a better model, a lower value can be used as stopping condition. The stopping condition thus ensures that the model reaches a certain level of performance (specified by the requirement for the validation loss), but doesn't continue iterating when this level is reached, which leads to a lot more computations without getting a (much) better model. Thus, no more computational power will be required than needed.

From the graphs that are made for different sizes of the training set (6a, 6b,6c), it can be seen that specifically for the reduction of 0.5% 6a and of 5% 6b over-fitting occurs. As the validation loss is not decreasing but slightly increasing, whereas the training loss decreases. So, the training data is still learning and the validation data is not. The training error is smaller than the validation error for all reduction factors (1). As expected, this difference is largest for the biggest reduction (factor 0.005). Since the amount of data in the training set is reduced and not in the validation set, the validation set requires more to have a better performance.

4.3 Further research

In the first place it is recommended to have more data to train the linear regression model. Because if there is more data the predicted area will be closer to the real area. To predict the area as precise as possible is very important if this model would be used to diagnose a patient.

For a better result of logistic regression, we recommend to do a more extensive search on the optimal set of hyper-parameters. This can be done by varying all parameters with each other, using nested for loops. However, the computational power and time was lacked upon in this project, so trying out this suggestion didn't succeed. Improvements that were already mentioned above are evaluating the loss curves to find optimal values for the parameters and using a stopping condition.

Another improvement can be to define functions that automatically compare the training results to select the set of parameters with the lowest validation error. Especially, when doing a broader search for the parameter values, this eliminates the need for a lot of manual work. Writing such a function is easy and can be done quite fast, so it will certainly be more time effective. Therefore, in a future project this must certainly be done.

References

[1] "Breast Cancer: Statistics — Cancer.Net."

[2] A. I. Baba and C. Câtoi, "TUMOR CELL MORPHOLOGY," 2007.

[3] M. Veta, P. J. Van Diest, S. M. Willems, H. Wang, A. Madabhushi, A. Cruz-Roa, F. Gonzalez, A. B. Larsen, J. S. Vestergaard, A. B. Dahl, *et al.*, "Assessment of algorithms for mitosis detection in breast can-

cer histopathology images," *Medical image analysis*, vol. 20, no. 1, pp. 237–248, 2015.

Appendices

A Results

A.1 A Results logistic regression

Table 2: Coarse search for optimal value of theta, with mu = 0.001, iterations = 300, batch size = 30

iterations	mu	batch	theta	training loss	validation loss
300	0.001	30	0.001	0.422	0.566
300	0.001	30	0.01	0.758	0.763
300	0.001	30	0.1	0.652	0.665
300	0.001	30	0.2	0.381	0.911
300	0.001	30	0.4	0.372	0.462
300	0.001	30	0.6	0.632	0.612
300	0.001	30	0.8	0.562	0.834
300	0.001	30	1	1.21	0.939
300	0.001	30	2	nan	nan
300	0.001	30	3	nan	nan
300	0.001	30	6	nan	nan
300	0.001	30	8	nan	nan
300	0.001	30	10	nan	nan

Table 3: Fine search for optimal value of theta, with mu = 0.001, iterations = 300, batch size = 30

iterations	mu	batch	theta	training loss	validation loss
300	0.001	30	0.0001	0.272	0.504
300	0.001	30	0.001	0.225	0.662
300	0.001	30	0.002	1.760	1.182
300	0.001	30	0.003	0.288	0.459
300	0.001	30	0.004	0.972	1.356
300	0.001	30	0.005	1.470	1.132
300	0.001	30	0.006	0.186	0.937
300	0.001	30	0.007	0.511	0.685
300	0.001	30	0.008	0.371	0.487
300	0.001	30	0.009	0.631	0.481
300	0.001	30	0.3	0.480	0.487
300	0.001	30	0.4	0.886	0.946
300	0.001	30	0.5	0.722	0.552

Table 4: Search for optimal value number of iterations, with mu = 0.001, batch size = 30, theta = 0.003

iterations	mu	batch	theta	training loss	validation loss
300	0.001	30	0.003	0.288	0.459
350	0.001	30	0.003	0.704	1.101
400	0.001	30	0.003	0.984	0.945
450	0.001	30	0.003	0.227	0.460
500	0.001	30	0.003	0.349	0.805
550	0.001	30	0.003	0.680	0.608
600	0.001	30	0.003	0.775	0.507
650	0.001	30	0.003	0.561	0.448
700	0.001	30	0.003	0.348	0.469

Table 5: Search for optimal value number of mu, with iterations = 300, batch size = 30, theta = 0.003

iterations	mu	batch	theta	training loss	validation loss
300	0.00001	30	0.003	0.586	0.605
300	0.0001	30	0.003	0.394	0.469
300	0.001	30	0.003	0.474	0.468
300	0.01	30	0.003	nan	nan
300	0.1	30	0.003	nan	nan

Table 6: Search for optimal value number of batch size, with iterations = 300, mu = 0.001, theta = 0.003

iterations	mu	batch	theta	training loss	validation loss
300	0.001	10	0.003	0.097	0.546
300	0.001	20	0.003	0.140	0.464
300	0.001	30	0.003	0.311	0.508
300	0.001	50	0.003	0.627	0.769
300	0.001	75	0.003	1.354	nan
300	0.001	100	0.003	nan	nan