Introduction to Machine Learning Project 2

Vlad Rosca - s202809

November 2022

1 Introduction

In this project, I continue using the RNA expression dataset, with samples taken from 5 different types of cancer: BRCA, KIRC, COAD, LUAD, PRAD. The attributes contain 20531 genes, whose RNA expression was measured. There are two goals for this project: do a regression analysis and a classification analysis through such models as linear regression, artificial neural network, logistic regression etc.

2 Regression. Part A

For the regression part of the project, I selected one of the genes and tried to predict its values based on the expression values of the other genes. The selected variable is Gene 15897 (where the gene numbers start with 0). This gene is therefore saved in a separate variable and removed from the input vector X before any feature transformations.

2.1 Feature transformation

The gene expression data in the input variable X is centered and standardized for standard deviation to equal 0.

Due to a very big number of attributes, which is a lot bigger than the number of samples, it would be very time-consuming and computationally expensive to train models. Therefore, the input expression data is used to do PCA. Data projected on the principal component space is further used for both regression to predict the selected gene expression, as well as classification. I decided to use the number of principal components that will explain around 90% of variance in the dataset, which was 312 out of 801 principal components (figure 1).

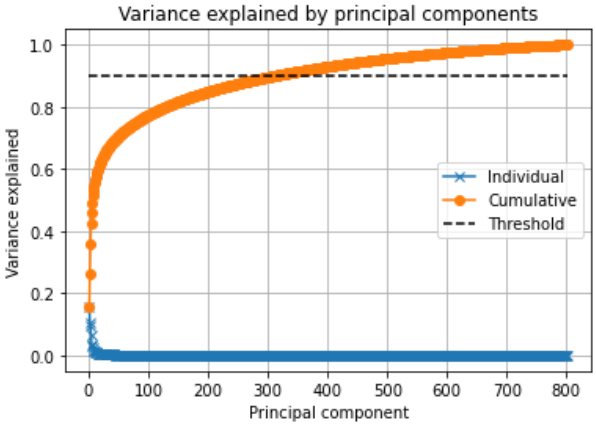

Figure 1: Variance explained by principal components

2.2 Regularization

To start with linear regression, I introduced a regularization parameter λ, and fit a linear model. Using 10-fold cross-validation, I tried to estimate a value for λ that would minimize the test error. I used a range of values between 10^{-1} and 10^{5}, with the step in power value of 0.5. I chose the step 0.5 because it seemed like the regularization parameter was minimal in between the integer powers of 10. The plot showing the test error versus the regularization factor is shown in figure 2. The calculated optimal value for λ is $10^{1.5}$.

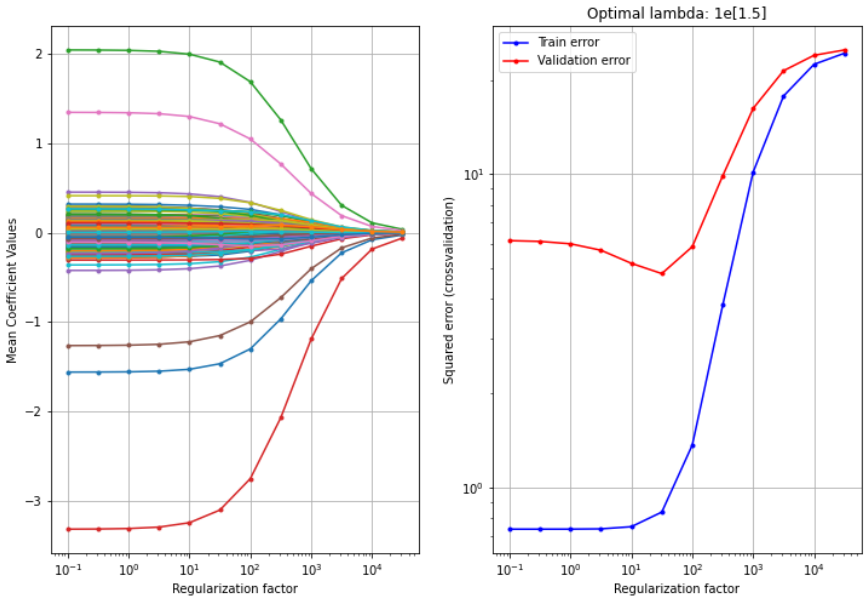

Figure 2: Regularization for linear regression. On the left the coefficients of attributes vs regularization parameter are shown. On the right, the test error versus the regularization factor is shown

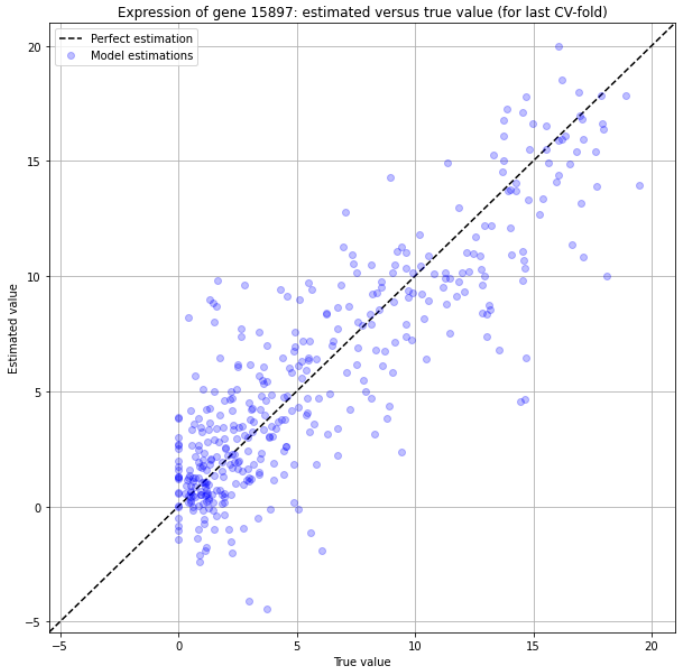

Figure 3: Correlation between the predicted and the true values of gene 15897

2.3 How it works

It's hard to explain how a new data observation is predicted since there are too many attributes (principal components) that are used, and they are a mix of even more gene expression values. However, the plot showing the true values against the predicted values looks like the predictions correlate with the true value and the model is valid. The gene expression can be predicted (figure 3).

3 Regression. Part B

In the part B of regression analysis, a two-layer cross-validation was applied to compare three regression models: a baseline model, a linear regression model, and an artificial neural network model. The method applied $K_1 = K_2 = 5$ since bigger numbers of folds were too time-consuming.

3.1 Baseline model

The baseline model consisted of a linear regression model with no features. Essentially, it computes the mean value for the attribute to be predicted in the given training set. It then makes a prediction by returning all values to the calculated mean value.

3.2 Linear regression model

As a complexity-controlling parameter of the linear regression model, the regularization strength λ was applied, in the same range as described above: from 10^{-1} to 10^5.

3.3 Artificial neural network

For the ANN regression model, the number of hidden neurons h was used as a complexity controlling parameter, and three values were used for that: $h = 1, 5, 20$.

3.4 Results

The results of the two-layer cross-validation analysis are shown in figure 4. The first thing that one can see is that test errors for ANN are noticeably bigger than for linear regression and are similar to the baseline model. This suggests that ANN did not manage to fit the dataset and ended up predicting the mean value instead. Bigger number of hidden neurons made the model make worse predictions, so one neuron was chosen in all cross-validation splits. For the linear regression, the same λ value $10^{1.5}$ was chosen all CV splits.

To evaluate whether there is a statistically significant difference between performances of the generated models, I used a t-test with setup I. The usual value of $\alpha = 5\%$ was used. The p-values and confidence intervals are presented in figure 5. Proving what was said above, the ANN did a very bad job in regression and the difference between the ANN and the baseline models is statistically insignificant. However, the linear regression model is statistically significant with very low p-values towards both ANN and Baseline models. I did try to figure out what went wrong with the neural network, but without success. It tended to converge, so an early stop of training definitely wasn't the issue. I assume an ANN should at least be able to get results as good as a linear regression model, so there was definitely something wrong with it. To look at the ANN results more closely, I plotted the values predicted by the model against the true values (figure 6). This made it very visible that the neural network just predicts the mean value all the time instead of finding a meaningful relationship between the attributes and the predicted variable.

Outer fold	ANN		Linear regression		Baseline
i	h_i	E^{test}_i	Lambda $_i$	E^{test}_i	E^{test}_i
1	1	28.29971313	10^1.5	4.37276471	24.79375374
2	1	25.33745193	10^1.5	5.43776526	20.970088
3	1	22.96499252	10^1.5	4.71770605	25.7258595
4	1	36.67111969	10^1.5	5.84315773	31.23911202
5	1	30.33005142	10^1.5	4.48172668	26.66884021

Figure 4: Comparison of the regression models: ANN, Linear regression, and Baseline. The complexity-controlling parameters λ and h are shown next to the respective test errors for each outer fold.

Models	p-value	Confidence interval	
ANN vs LR	7.335603009004e-11	15.70499952463914	28.0858423112332
ANN vs Base	0.744329527928	-2.05509523089649	1.47171001222644
LR vs Base	2.067156472426e-14	-27.3903496421662	-16.983877412376

Figure 5: P-values and confidence intervals for t-test comparison of the regression models: ANN, Linear regression, and Baseline.

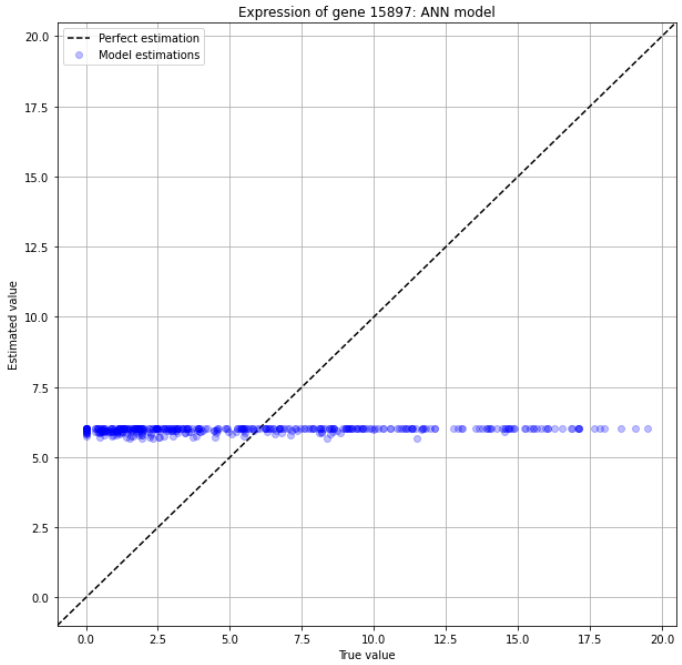

Figure 6: Predicted against true values of gene 15897 in the ANN regression model

4 Classification

For the classification part of the project, I chose to predict the cancer types for the samples provided, using the RNA expression of the genes. This is a multi-class classification problem – there are 5 different types of cancer in the dataset: BRCA, KIRC, COAD, LUAD, PRAD. As mentioned earlier, it is the data projected on the principal component space that will be used to train the models, and out of 801 principal components, 312 of them are used.

Three models were compared in this part: multinomial regression, ANN for multiclass classification, and baseline. Similarly to the linear regression, for multinomial regression regularization strength λ is used as a complexity-controlling parameter. The range is chosen to be $\lambda = [10^{-3}, 10^{3}]$. For the ANN classification model, the range of numbers of hidden neurons was chosen to be $h = [1, 20, 100]$.

The baseline model was set to identify the biggest class (most frequent cancer type), and predict all test data as belonging to the biggest class.

Two-layer cross-validation was used to compare the models. Internal cross-validation cycles were used to determine the best values for λ and h for each outer CV fold. The optimal values were used to train the models again, on the whole training set of the outer fold, and to calculate the test errors.

4.1 Results

The optimal values for λ and h, together with the test errors, are shown in the figure 7. For the multinomial regression, the same regularization values were chosen in every outer CV fold $\lambda = 0.001$. In 4/5 folds, the multinomial regression did not make any mistakes in the test set. For ANN, the

number of hidden neurons varied between $h = 20$ and $h = 100$. The errors look noticeably larger than in the multinomial model but are very small compared to the baseline model.

The models were statistically compared to each other using McNemar test, and p-values with confidence intervals are shown in figure 8. Difference in accuracies of each pair of models is statistically significant, meaning that it's likely none of the models have the same error rate. Thus, multinomial regression is statistically more accurate than the neural network model, and both multinomial and ANN models are statistically more accurate than the baseline model.

Outer fold	Multinomial regression		ANN		Baseline
i	Lambda $_i$	E^{test}_i	h $_i$	E^{test}_i	E^{test}_i
1	0.001	0	100	0.03106	0.64596
2	0.001	0	20	0.03125	0.59375
3	0.001	0	100	0.03750	0.64375
4	0.001	0	100	0.04375	0.63750
5	0.001	0.00625	20	0.07500	0.60625

Figure 7: Comparison of the classification models: Multinomial regression, ANN, and Baseline. The complexity-controlling parameters λ and h are shown next to the respective test errors for each outer fold.

Models	p-value (McNemar)	Confidence interval	
MR vs ANN	1.0768e-09	0.0280676528	0.0568175629
MR vs Base	6.1098e-151	0.5901403989	0.6571653001
ANN vs Base	1.8869e-126	0.5453750124	0.6170555055

Figure 8: P-values and confidence intervals for McNemer test comparison of the classification models: Multinomial regression, ANN, and Baseline.

It is again very hard to understand how the multinomial regression model makes a prediction due to a high number of attributes i.e. the principal components, which also are a combination of many genes.

5 Discussion

Two-layer cross-validation is a powerful algorithm to choose the best parameters for a model and to better estimate its generalization error. Regularization allows to control the complexity of linear regression and logistic regression models by reducing overfitting and improving the predictions on the "unseen" datasets (i.e. reduce the generalization error).

Artificial neural networks, in spite of the potential, are not always the best choice for some datasets as they require a really long time for training. They also have a much bigger selection of parameters, and it is difficult to find the best parameters for a given dataset and purpose (number of hidden neurons, number of hidden layers, loss functions etc.). For example, in my classification case, the ANN model did not have the same optimal number of hidden neurons in every fold. Some folds ened up choosing 20 hidden neurons instead of 100. Thus, more neurons does not always mean the prediction will be more accurate. Perhaps, it contributes to overfitting.

Even though it is impossible to make sense of the coefficients in the linear regression and multinomial regression models, it is still possible to compare which attributes have the biggest effect.

Figures 9 and 10 show the largest coefficients in the respective models. Each table shows 6 different principal components that contribute the most to the prediction of their respective variables. Shared property in these models is having high coefficients for principal components 1, 2, 4, 7. To be fair, this is not surprising because the first PCs always explain the most variance in the data.

I tried to find out how this dataset has been analysed before, but without success.

Attribute (Principal Component)	Coefficient
1	5.59287009
2	-1.47992481
4	1.85211944
5	-3.21086028
7	-1.08672680
8	1.38003826

Figure 9: Attributes with the highest coefficients in the linear regression model

Attribute (Principal Component)	Class	Coefficient
1	BRCA	-0.1137923305
1	KIRC	0.1414573157
2	LUAD	-0.0654930029
2	PRAD	0.1097529362
3	BRCA	-0.1826215261
3	PRAD	0.0856473308
4	LUAD	-0.1325165702
6	COAD	0.0777634707
6	LUAD	-0.0833559744
7	BRCA	-0.0734525101

Figure 10: Attributes with the highest coefficients in the multinomial regression model

6 Exam Problems

The following are the solutions for the exam problems given in the project brief.

Question 1. **Answer D**: The observations of both classes are constantly close to each other, uniformly distributed. That makes the true positive and false positive rates to be close to each other at any threshold and the ROC close to a straight line.

Question 3. **Answer B**: network contains 124 parameters. From 7 inputs to 10 hidden neurons there are 7*10 weights + 10 biases. From 10 hidden neurons to 4 outputs there are 10*4 weights + 4 biases. $7 * 10 + 10 + 10 * 4 + 4 = 124$

Question 4. **Answer D**: I was looking at congestion level 3. To get it right, this has to apply: $b1 > -0.76$, $b1 < -0.16$, $b2 < 0$, and nodes A = true, C = false, D = false. A: $b1 > -0.76$ – true, C: $b1 > -0.16$ – false, D: $b2 > 0.01$ – false. Switch D to true, and the congestion level switches to level 1, just like in the tree.

Question 5. **Answer C**: 5 outer folds. For each outer fold: 4 internal folds * 5 hidden units/regularization strenghs * (1 ANN train + 1 ANN test + 1 regression train + 1 regression test) + 1 ANN outer train + 1 ANN outer test + 1 regression outer train + 1 regression outer test. $5 * (4 * 5 * (20 + 5 + 8 + 1) + 20 + 5 + 8 + 1) = 3570$

7 Report Contributions

All students contributed equally to all sections. The percentage is shown in figure 11.

Contributions	Student
Section	Vlad Rosca s202809
Regression Part A	100%
Regression Part B	100%
Classification	100%
Discussion	100%

Figure 11: Report contributions