Machine Learning

Milestone 1 H36m Data

Julien Ray (328357)
Yannik Krone (347243)
Gaspard Thoral (345230)

8 November 2022

1 Hyper-parameters optimization

1.1 Linear regression

There are no hyper-parameters in closed form linear regression. The MSE on the test data is 558.059. We will discuss this result in the dedicated section.

1.2 Ridge regression

For ridge regression, the only hyper-parameter to be optimized with the help of cross-validation is the regularization parameter lambda.

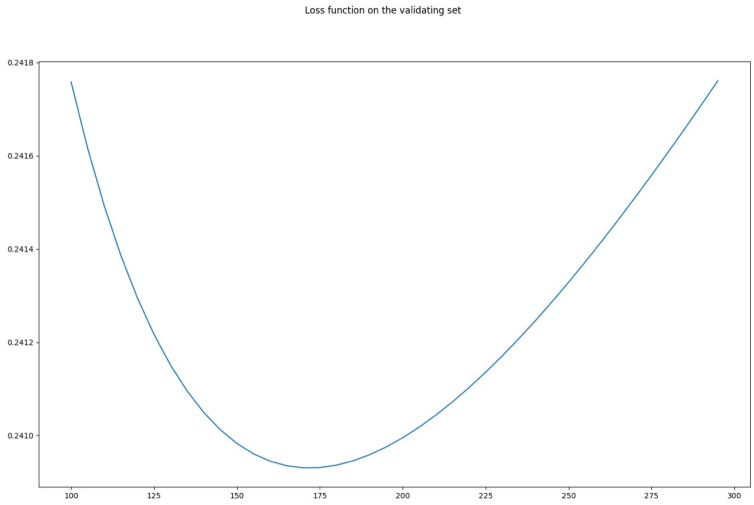

Figure 1: Validation loss based on λ

Figure 1 is the validation loss based on lambda. We can see clearly the best value for lambda is in between 170-175.

This demonstrate the overfitting/underfitting dilemma on the training data: if the regularization factor is too low the ridge regression overfits the training data which results in a higher loss when tested on data that has never been seen before (here the validation set). On the contrary, when the lambda is too big, it puts too much restriction on the model which prevent it from learning properly.

1.3 Logistic Regression

For logistic regression we can optimize the learning-rate as well as the number of iterations.

Figure 2 shows the F1 Score Based on the number of iterations with fixed learning rate 1e-4. It reaches a minimum under 10 iterations and is stable.

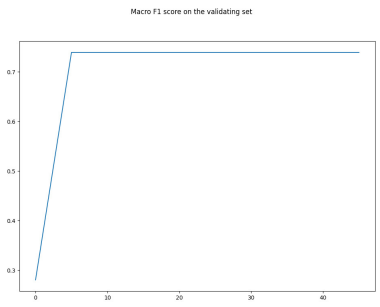

Figure 2: loss based on max iterations

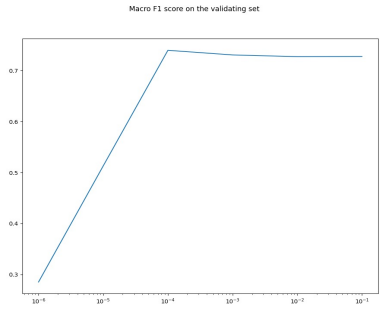

Figure 3: loss based on learning-rate

Figure 3 shows the F1 score based on the learning rate used in gradient descent. There is a maximum value close to 1e-4. The cross validation was done with enough iterations in order to not penalize a lower learning-rate since they would need more time to converge.

2 Results and discussion

linear regression	ridge regression	logistic regression
-	$\lambda = 170$	lr = 1e−4 / max-iterations = 10
MSE = 558.059	MSE = 0.367	Accuracy = 78.94 % Macro F1 = 0.7100

2.1 MSE on ridge/linear regression

There is a big gap in loss between ridge regression with λ 0 and 170. This can be explained by the very high dimensionality of H36m data compared to the number of samples. Without any regularization the model will poorly generalize which results in a very high loss on test data.

2.2 Logistic regression

We got satisfying results on logistic regression with an accuracy close to 80% and F1 score of 0.71. However, if we look closer at the confusion matrix in figure 4, we can see that it struggles to classify class 4 correctly. We think this poor performance is in part due to the lower amount of class 4 data compared to the other classes. (see figure 5). Performances for classes 1 and 2 on the other hand are really good with an accuracy close to 100%.

2.3 Cross-validation limitations

Cross-validation is a fantastic tool but in practice, it can have a few limitations. Indeed, after finding optimal parameters with this method, we tried tweaking

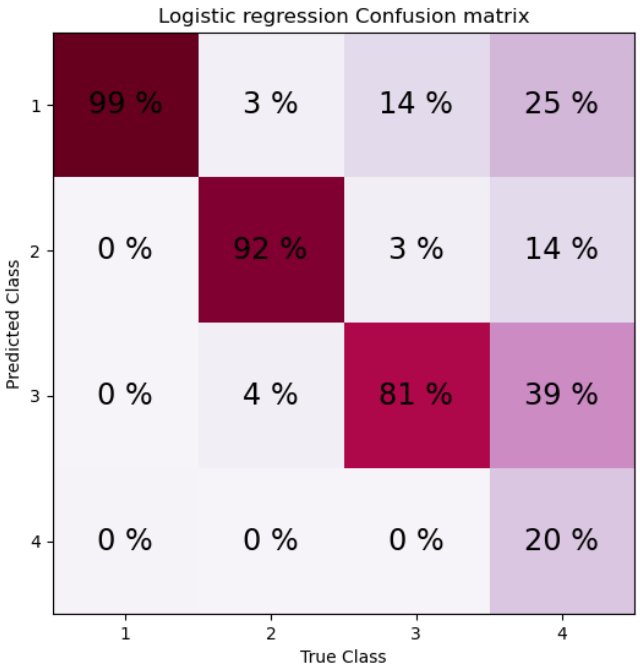

Figure 4: confusion matrix

manually the parameters on the test data to see how good of an optimization cross-validation did. To our surprise, there was some noticeable but not major differences.

Since cross-validation only use training data to validate itself, if there's a difference in features distribution between the test data and the training data, the optimization won't be perfectly accurate. If we look at figure 5 and figure 6 we can see that the distribution between classes is quite different. This may be one of the reason our model performed slightly worse than expected on the test data where the different labels are more evenly distributed.

class	1	2	3	4
n-labels	809	770	550	460

Figure 5: training data
labels count per class

class	1	2	3	4
n-labels	131	222	163	102

Figure 6: test data
labels count per class

2.4 Softmax overflow in logistic regression

Using the default softmax function seen in class caused a few problems for us at first. Indeed, the matrix product data @ weight got very big and caused the exponential function in softmax to overflow. In our project, we implemented a slight variation of softmax to make it numerically stable :

softmax($Vector - C$) is the same as softmax($Vector$). We used this feature to avoid having overflow in our code by doing softmax($predLabels - max(predLabels)$).