MileStone 1 - Report
EPFL, Spring semester 2024
CS-233 - Introduction to Machine Learning

Staner Karen *(363746)* | **Nissim Ilan** *(358235)* | **Torgeman Youssef** *(360550)*

1 Introduction

In the context of the **CS-233 course Introduction to Machine Learning** we aimed to apply basic machine learning techniques to solve real-life problems. Thus, the goal of Milestone 1 was to recognize which dog breed is in the image and to find the center of the dog in the image. For that, we used three main algorithms, namely Linear Regression, Logistic Regression and K-Nearest Neighbours (KNN). In this implementation, we utilized a subset of the Stanford Dogs Dataset, which consists of 3,256 images of which 2,938 where training images and 327 where test images of different dogs. From the original collection of 120 dog breeds, our implementation is focused on a subset of 20 breeds.

2 Method

2.1 Data Preparation

Prior to applying any Machine Learning algorithm to our data, we proceeded with the following basic steps: loading the data and doing some pre-processing of the data. Our pre-processing routine consisted of two modifications on the data set. The first one being feature normalization, which consisted in scaling the features in each image to make sure that no input feature could dominate the model's learning mechanism due to its scale. The second being adding a column of bias to ensure that the intercept term was possible.

2.2 Validation Set

To select the best hyper-parameters for our Logistic Regression and Linear Regression, we opted to use a Validation Set strategy. This involved randomly partitioning our training set into two subsets: a new training set containing 80% of the original training set and a validation set, which made up the remaining 20%.

2.3 Cross Validation

When fine-tuning our KNN algorithms, we decided to use a 3-fold Cross Validation Approach. We decided to randomly split the training data into three partitions, and iteratively performed three evaluation rounds. In each round, we used a different fold as the test set, while the other two served as a training set.

3 Experiment/Results

3.1 Linear Regression

In the experiment stage, we tested the effect of regularization in our ridge regression model by changing the 'lambda' and computing the test loss (MSE) on the validation set.

From the graph below, it can be seen how the value of the test loss progressively increases as the value of lambda increases from 0 to 10. The data shows that the model achieves the lowest loss value when **lambda is equal to 0**, with a test loss of slightly over **0.005**. This indicates that adding regularization did not lead to improved performance over the validation set. This also suggests that our model's complexity is appropriate for the data and doesn't require regularization to prevent over-fitting.

When testing on the actual test set with **lambda = 0**, we obtain the same result.

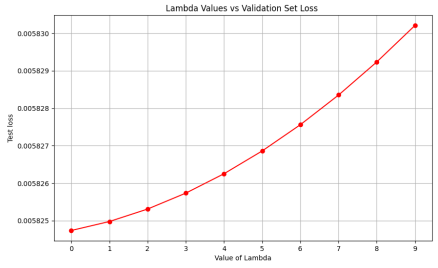

Figure 1: This figure illustrates the validation set loss versus lambda value for ridge regression.

3.2 Logistic Regression

In the logistic regression model, after having converted each label into one-hot encoded vectors, we optimized two parameters using **grid search**: the learning rate (lr) and the maximum number of iterations (max_iters) utilized in the gradient descent.

As demonstrated by the data in Table 1, a low learning rate means slower convergence, possibly requiring many iterations. This in turn increases the computational expenses, which could be impractical for large datasets. Conversely, a high learning rate can cause the training process to diverge or instead converge to a premature, suboptimal solution.

Lr \E	100	500	1000	2000
1e-5	Acc = 39.012% F1 = 0.260496 0.079s	Acc = 62.181% F1 = 0.497729 0.388s	Acc = 68.825% F1 = 0.604232 0.683s	Acc = 78.876% F1 = 0.755021 1.260s
1e-4	Acc = 68.825% F1 = 0.604232 0.070s	Acc = 83.646% F1 = 0.822115 0.340s	Acc = 85.179% F1 = 0.840446 0.581s	Acc = 86.712% F1 = 0.857032 1.353s
1e-3	Acc = 85.349% F1 = 0.842553 0.088s	Acc = 86.882% F1 = 0.858565 0.328s	Acc = 86.371% F1 = 0.852841 0.581s	Acc = 86.201% F1 = 0.850101 1.198s
1e-2	Acc = 84.497% F1 = 0.826216 0.148s	Acc = 86.031% F1 = 0.849627 0.332s	Acc = 86.031% F1 = 0.849455 0.828s	Acc = 85.860% F1 = 0.847905 1.404s
1e-1	Acc = 78.876% F1 = 0.750027 0.084s	Acc = 79.727% F1 = 0.768338 0.301s	Acc = 78.705% F1 = 0.761155 0.660s	Acc = 75.809% F1 = 0.713055 1.205s

Table 1: Report of the accuracy on the validation set per model per task

In this study, we found that the learning rate $lr = 1e-3$ proved to be the most efficient, attaining a high accuracy of **86.882%** and an F1-score of **0.858565** with **500** iterations. The computational time recorded at 0.328 seconds suggest a good compromise between performance and computational efficiency.

To find the most precise number of iteration we implemented early stopping. The goal was to stop the training when the accuracy and F1-score started not increasing after a certain number of iterations. The results indicated that from **350** iterations onwards, with a learning rate of **lr = 1e-3**, the accuracy began to plateau.

When testing on the actual test set, with these parameters we obtained an accuracy of **86.544%** and an F1-Score of **0.854154**.

3.2 KNN

For the K-Nearest Neighbours, the hyperparameter to optimize was K, which specifies the number of nearest neighbours to consider for predictions. To identify the best K, we explored a range of hyperparameters from 1 to 100 within a 3-fold cross validation framework.

3.2.1 Classification

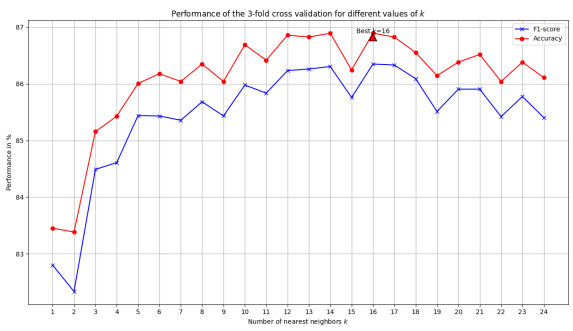

Figure 2: Accuracy and F1-score of the 3-fold cross-validation for KNN classification with the number of neighbours (K) ranging from 1 to 25.

For the classification task, we found that the **K=16** was the value that yielded the best results, achieving an accuracy of **86.89%**. This value strikes a perfect balance-it's neither too high nor too low. If it were too low, the model would make mistakes by focusing too much on small details it was trained on; this is over-fitting. If it were too high, the model might miss important differences between classes; implying under-fitting.

On a different note, the remarkably high accuracy at **K=1** is worth mentioning. Around 80% of classifications made on the basis of one single neighbor were correct. This implies that the features extracted from the images effectively capture the differences between classes. Typically, one single neighbor is sufficient to make accurate predictions. This could reflect a well-clustered class distribution in the feature space. Also, this extraordinary performance for low-value K is also a good indicator that over-fitting is not a concern in this model.

To verify that our results were optimal, we also conducted experiments with different distance metrics (Manhattan Distance (L1-Norm), Euclidean Distance (L2-Norm), L3-Norm and Cosine Distance).

Distance Metric	Optimal K	Maximum Accuracy
Manhattan Distance (L1 Norm)	9	86.79%
Euclidean Distance (L2 Norm)	16	86.89%
L3 Norm	13	86.85%
Cosine Distance	9	86.81%

Table 2: Optimal K and Maximum Accuracy obtained with each distance with 3-fold cross-validation

As demonstrated by the data in Table 2, we realized that the distance metric didn't have much impact.

3.2.2 Regression

For the regression task, we obtained some interesting results. For instance, we found that the optimal number of neighbours was **K=88** with a test loss of **0.00542**.

But, more importantly we noticed that as K increased towards the number of samples, the test loss converged towards a little above 0.005. As K grew bigger and bigger, our prediction became a global averaging of all the target values (i.e in that case the coordinates of the center of each dog). This very low test loss could suggest that the center of each dog is located at the same place.

When testing on the actual test set with **K=88**, we obtained similar results with a test loss of **0.005**.

4 Conclusion

When comparing the results achieved with each algorithm on the real test set, it becomes apparent that their performance levels are similar. Both KNN-Classification and Logistic Regression accomplish accuracies higher than 86%, while both Linear regression and KNN-Regression realize a low test loss of 0.005.