Machine Learning Lab 1 Report

Luis Calderón Robustillo, Andrés Martínez Silva

October 20, 2023

1 Introduction

In this report, a set of basic machine learning algorithms will be described and implemented in prediction models in order to analyse their performance according to different metrics.The goal is to compare the models and reason about which of them is able to compute the best predictions on a given dataset and why. In this process, techniques such as cross-validation and hyperparameter tuning will be used to select among the set of candidate models.

This report is divided into two main sections which tackle two different machine learning problems. In the first Section we train and evaluate a set of regression models, which estimate a function that relates an input vector with a continuous output. In the second section, classification models, which estimate a function that relates an input vector with a discrete output, are evaluated. For this task, two different public data sets will be used: a simplified version of the Year Prediction - Million Songs Database [5] for the regression problem, and the CIFAR-10 Database[1] for the classification problem. All algorithms have been implemented using the scikit-learn Python library for machine learning [3] [4] together with the Google Colab service.

2 Regression models

2.1 Dataset and pre-processing

First of all, we must analyse the dataset. As seen in Figure 1, according to the dataset information, there are more songs as the year approaches 2011, while there are less songs as we go closer to the 1920s.

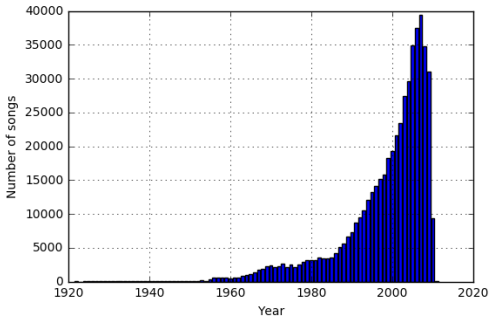

Figure 1: Histogram representing the number of songs in the dataset per year

To continue, we inspect the data visualizing the relations between each dimension of the feature vector and the output. To understand the graphs result, let's first look at how our data is set up. We have 50,000 songs from one specific year. Each song has 90 numbers associated to it that represent different features for that song.

When examining the data, we noticed something interesting. There are more datapoints around the year 2011, but in almost all the graphs, the scores are spread out to the right and left. This means that the scores don't change mainly because of the year, but because we have more data. So, this might make it hard to make predictions based on these scores, because they seem to act independently of the year. In addition, we can analyse the correlation between the features and the outputs. All the values are close to zero, this suggests, how we were predicting before, that there is not much linear relationship between the features and the output. That correlation measures only linear relationships, so the non linear relation-

ship is not captured here.

2.2 Tested algorithms

The methodologies which are going to be studied include Linear Regression, Lasso Regression, Ridge Regression, Huber Regression, RANSAC Regression, and K-Nearest Neighbors (KNN). For each of these methods, we will try to study polynomial solutions if possible. To analyse their performance we are going to use key metrics such as the R-squared ($\mathbf{R^2}$) score and Mean Squared Error (MSE).

Linear regression assumes a linear relationship between input features and the target variable. The goal is to find the best-fitting line that minimizes the prediction error.

Secondly, **Lasso** method is a technique used to select important features from a larger set of variables. This is really useful when there are a lot of potential features, and you want to figure out which ones are actually influential in making predictions.

While Lasso adds a penalty term based on the absolute values of the coefficients, Ridge adds a penalty term based on the squared values of the coefficients. This way, Ridge tends to shrink the coefficients towards zero, but rarely fully eliminates them.

Huber regression is a variation of linear regression that is robust to outliers in the data. It combines elements of both least squares (which minimizes the sum of squared errors) and absolute deviation (which minimizes the sum of absolute errors).

RANSAC is an iterative method used for robust regression in the presence of outliers. It works by fitting models to randomly selected subsets of the data, identifying the inliers that best support the model, and then refining the model using these inliers.

Finally, **K-NN** models use the dataset itself to predict the output. Depending on the number of neighbours chosen the output will be the one provided by the average of the outputs of the nearest neighbours to the input.

2.3 Training and validation results

The results for k-fold cross-validation for the proposed models can be seen in Table 1. While different degrees of polynomial fit on the feature vectors were also tested on all the methods, results were generally poor and so they were discarded from the final metrics.

Method, parameters	MSE $[years^2]$	R^2
Logistic regression	89.43	0.21
Lasso classifier $\alpha = 0.15$	91.23	0.2
Ridge classifier $\alpha = 0.9$	89.43	0.21
Huber	94.62	0.17
RANSAC	1607.29	-13
KNN n=10	86.6	0.24

Table 1: Cross-validation results for the tested methods with no polynomial regression.

To analyse these results we must take into account that, like we said in the introduction of this problem, our data is not an easy dataset to model. Anyway, there are various models which obtain results that can be considered surprisingly good. Concretely, the better models are K-NN and linear regression, obtaining an average of error between 9.46 and 9.3 years, and between 21 % and 24 % percent of variance.

2.4 Test results and conclusions

Finally, we use also validation data for training our models and we use test data to get metrics of our best performing models.

Method, parameters	MSE $[years^2]$	R^2
Logistic regression	87.31	0.21
KNN n=10	81.6	0.27

Table 2: Cross-validation results for the tested methods with no polynomial regression.

According to the results, K-NN model and linear regression seem to be the best two model to use for this problem. On the one hand, with linear regression we obtain a light model. On the other hand, the K-NN model is the data itself, so it will require

more memory while using it. In this way, if precision is prioritized, the K-NN model is the best solution. However, if saving memory is an important matter to keep in mind, the linear regression model would probably be a better choice.

3 Classification models

3.1 Dataset and pre-processing

The original CIFAR-10 dataset consists of 60000 32x32 images in 10 classes -see Figure 2-, with 6000 images per class. 50000 of them are meant for training and rest are meant for testing. The input images can be interpreted as feature vectors of dimension 32x32x3. Due to the computational complexity of using the complete dataset and the size of the feature vectors, a pre-processing stage becomes necessary. The first step is to reduce the training and test sets to a more manageable size. First, it was reduced to a 20 %, then to 35% to improve the results.

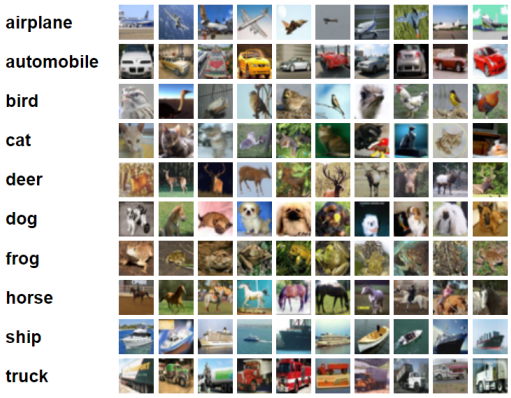

Figure 2: CIFAR-10 classes with some examples

After dataset reduction, GIST descriptors for each image are extracted using a set of parameters and provided functions which allow us to reduce the dimensions of the feature vectors to 256. Additionally, we implement Principal Component Analysis (PCA) to further reduce the size of the feature vector.

The GIST[2] descriptor uses a set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) to represent the dominant spatial structure of a scene. On the other hand, PCA is a technique based on SVD decomposition which is used to supress redundant dimensions in high-dimensional data by projecting the data into a low-dimensional space, all the while preserving the dimensions of maximum variance. Even after GIST feature extraction, PCA becomes necessary to train more complex models as computation times scale very quickly with input feature size. Finally, a standard scaling step is performed which consists on standardizing the input vector by removing the mean and scaling to unit variance.

3.2 Tested algorithms

- **KNN classifier**: for each query sample, the distance to the closest elements of the dataset is computed, and each neighbor takes a vote in order to assign the queried element to an output class.
- **Logistic regression**: the most basic algorithm for linear classification. These models define hyperplanes or decision boundaries that determine whether the queried element belongs to one class or another.
- **Ridge classifier**: classification variant of Ridge regression. The main difference is an initial step where the classifier first converts binary targets to {-1,1}, and then operates as a regression algorithm. Generally faster than logistic regression for multi-class problems.
- **SVM classifier**: these classifiers are characterized by maximum margin parameter ρ which represents the distance between the hyperplanes and the closest samples, also referred as support vectors. SVM methods try to maximize the margin ρ for the different hyperplanes. They are especially efficient in high dimensional spaces and come in many variants, both linear and non-linear, depending on the kernel used.
- **Voting classifier**: classifier that takes majority voting between the predictions of two or more models to make the final prediction.

3.3 Training and validation results

Due to the computational overhead associated with pre-processing and hyperparameter search, the problem was subdivided to select among all the candidate models. We started with a very reduced dataset to conduct a wide search and test different dimensions of PCA. In this stage, two conclusions were made: (1) pre-processing with PCA in addition to GIST greatly reduces computational cost, but it also produces a notable, negative impact in perfomance when using PCA<50. With PCA 50, the perfomance was fairly similar to that of the models trained on the descriptors alone, with the computational times reduced by more than half. With PCA>50, the computation times kept growing with minimal effect on perfomance, so we stuck with PCA=50 combined with GIST. (2) Most of the classifiers scored $\approx$ 50% in accuracy, precision and recall metrics both with GIST only and GIST+PCA. The exception were the non-linear SVM models with RBF (Radial Basis Function) kernel which performed around 10% better in all three metrics. However, when searching over all SVM variations, for those models using linear kernels the cost was much higher (>40 minutes) than for those which excluded them(< 5 minutes), especially for high values of the margin-slack tradeoff coefficient C. As for the voting classifier, it was trained using majority voting between the two best performing models -logistic regression and SVM -, with both accuracy and cost metrics falling in a middle ground between the two which seems logical.

In all experiments, SVM models with RBF kernel seemed to perform slightly better, so the search was narrowed down to find the best value for coefficient C using this specific kernel with an expanded dataset (17500 images, or 35% of the original size). The results for 10-fold cross-validation of the final set of candidate models can be seen in **Table 3.**

3.4 Test results and conclusions

In this stage, the ability of the selected model to make predictions on the test test is evaluated. The test split consists of a total of 1750 images. The model under evaluation is the SVM classifier with RBF kernel and a tradeoff parameter C=2, trained on data which has been pre-processed with GIST and PCA=50, then normalized. We obtain 64.74 % accuracy score which is consistent with cross-validation metrics, and can be considered quite acceptable given the size and complexity of the dataset.

Method, parameters	Accuracy [%]	Cost [s]
Logistic regression	55.48	3.0
KNN n=20	50.29	26.0
Ridge classifier $\alpha = 100$	52.24	1.0
SVM rbf, C=2	64.95	1200.0
Voting classifier	60.35	200.0

Table 3: Cross-validation results for the tested methods using a pre-processed dataset with PCA=50

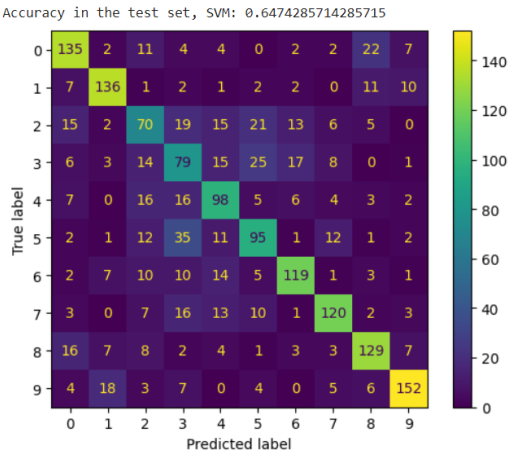

Figure 3: Confusion matrix displaying the prediction results for the selected model in the test stage

In **Figure 3**, the confusion matrix is shown comparing the predicted labels to the true labels. Classes 3 and 5, which correspond to cats and dogs respectively, seem to be confused the most, which makes sense as they are fairly similar in the context of this dataset. Birds are also uncorrectly classified as dogs in some examples, as well as airplanes and ships, being vehicles of similar shape, size and color.

References

[1] *CIFAR-10 Database*. Overleaf. (n.d.) URL: https://www.cs.toronto.edu/%CB%9Ckriz/cifar.html. (accessed: 26.9.2023).

[2] Aude Oliva and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope". In: *International journal of computer vision* 42 (2001), pp. 145–175.

[3] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[4] *Scikit-learn User guide*. https://scikit-learn.org/stable/user_guide.html. Accessed: 2023-10-15.

[5] *Year Prediction - Million Songs Database*. ISMIR. 2011. URL: https://samyzaf.com/ML/song_year/song_year.html. (accessed: 26.9.2023).