Graphaware Assessment Report: California Housing Value Predictions

Andrea Paudice

March 21, 2021

Abstract

This report summarizes the design, the implementation and the analysis of a regularized linear regression algorithm for the prediction of the value of a house based on a bunch of features. The analysis is limited due to time constraints.

1 Dataset description

For this assessment project, the California Housing Prices dataset has been used. It contains information from the 1990 California census and the columns are as follows:

- *longitude*, a measure of how far west a house is; a higher value is farther west.
- *latitude*, a measure of how far north a house is; a higher value is farther north.
- *housing median age*, median age of a house within a block; a lower number is a newer building.
- *total rooms*, total number of rooms within a block.
- *total bedrooms*, total number of bedrooms within a block.
- *population*, total number of people resid- ing within a block.
- *households*, total number of households, a group of people residing within a home unit, for a block.
- *median income*, median income for house- holds within a block of houses (measured in tens of thousands of US Dollars).
- *median house value*, median house value for households within a block (measured in US Dollars).
- *ocean proximity*, location of the house with respect to the ocean/sea.

The goal is to predict the **median house value** in a given block of houses given a vector of the above features (excluding *median house value*) representing the block. The overall data consist of 20433 records, but I will use a much smaller sample.

Remark: I decided to use a smaller version of this dataset as it offers a nice trade-off between a complex real-world dataset and one allowing to illustrate several aspect of a typical machine learning problem.

2 Proposed algorithm

In order to predict the median house value from the given features, a regularized linear regression model with the squared loss has been adopted.

Training algorithm. Given a set of observations $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ and a scalar $\lambda > 0$, the algorithm seeks for a vector $w \in \mathbb{R}^d$ minimizing the following cost

$$L(w) = \frac{1}{2n} \sum_{i=1}^{n} (x_i^T w - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2, \tag{1}$$

where the notations $x^T y$ and $\|x\|_2$ denote the standard inner product and the ℓ_2-norm in $\mathbb{R}^d$ respectively. In (1) the factor $1/2$ has only be introduced for mathematical convenience and does not affect the optimal solution. Notice that when $\lambda \to 0$ leads (1) towards the ordinary *Least Square Regression* (LSR) where we seek for a vector w minimizing the empirical risk with respect to the squared loss. On the other hand a large λ leads the solution towards vectors with small norm, but possibly a large empirical risk. Thus λ allows to control the trade-off *bias-complexity* and needs to be tuned from the data as I will show later.
As for the solution, notice that (1) is a convex and differentiable function in w so the minimum can be found by equation its gradient to 0. To this end define $X \in \mathbb{R}^{n \times d}$ as the matrix with the instances $x_1, dots, x_n$ stacked as rows and $y \in \mathbb{R}^n$ as the vector of the labels. Equipped with this notation observe that (1) can be rewritten as

$$L(w) = \frac{1}{2n} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2. \tag{2}$$

The gradient is given by

$$\nabla L(w) = \left(\frac{1}{n} X^T X + \lambda I_d \right) w - \frac{1}{n} X^T y, \tag{3}$$

where with X^T I denote the transpose of X and I_d denotes the identity matrix in $\mathbb{R}^{d \times d}$. Thus the optimal solution can be computed in closed form and is

$$w^* = \left(X^T X + n\lambda I_d \right)^{-1} X^T y. \tag{4}$$

Notice that (2) can be minimized also via *Gradient Descend*, however considering the size of the dataset, I preferred to use the closed form solution rather the approximation provided by gradient descend. Notice also that computing (4) takes $O(d^3)$ times, which for medium-sized datasets, as the one considered in this project, is reasonable.

3 Preprocessing

To illustrate several aspect of a typical machine learning project I use a very small fraction of the data for training, i.e. 50 samples, while leaving 1000 samples for testing. The fraction of the leftovers, will be used later on to simulate a data campain to enlarge the trainin set.
Since the dataset contains rows with missing values and they represent about the 1% of the overall dataset, I decided to remove them. The labels have been re-scaled so that each unit corresponds to 1000 dollars for readability; otherwise the error values are too large. The dataset contains also a categorical feature *ocean proximity* taking 5 different values: <1H OCEAN, INLAND, ISLAND, NEAR BAY, NEAR OCEAN. I decided to use a *1-hot-encoding* in order to no impose any ordering among the values. Another possibility is to treat this feature as ordinal and map each value to an integer with value proportional to its distance from the ocean. However, since there is some ambiguity among values like <1H OCEAN, NEAR BAY and NEAR OCEAN I preferred to stick with the 1-hot-encoding. As a result,

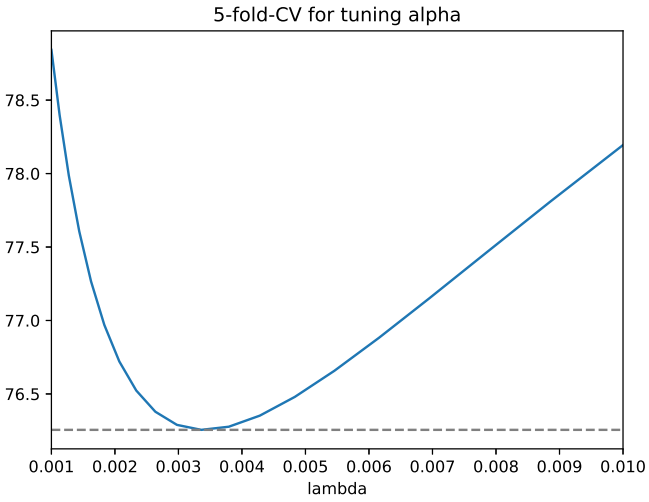

Figure 1: 5-fold-cross-validation for tuning λ.

the feature *ocean proximity* as been replaced with 5 binary features that for each feature vector have the property that just 1 of them takes the value 1 (the one corresponding to the value of *ocean proximity* in the given vector). As for normalization, I decided to re-scale all the features in the range $[0, 1]$ so that the (1) becomes easier to optimize. Normalization has been carried out on the training data and then applied to the test data in all the procedures reported below.

4 Hyperparameter evaluation

To tune the value of λ I resorted to 5-fold-cross validation. After few attempt to find an appropriate range of values to test, I focused on the range $[0.001, 0.01]$ where I tested 20 equally spaced values. The resulting cross-validation plot is reported in Figure 1 where the optimal value of alpha is about 0.0035.

5 Results

I tested the learned model on the test data and reported the *Root Mean Squared Error* (RMSE)

$$R(w) = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}(x_i^T w - y_i)^2}$$

as a performance metric. In order to evaluate the quality of the found solution I also compared the results against 3 baselines:

- **mean**, this is simply the average label in the training data.
- **median**, this is the median label in the training data.
- **LSR**, this is the ordinary LRS, i.e. the predictor learned solving (1) with $\lambda = 0$.

Predictor	RMSE
Linear RLSR	75.43
RLS	81.19
mean	116.53
median	122.28

Table 1: RMSE on the test data.

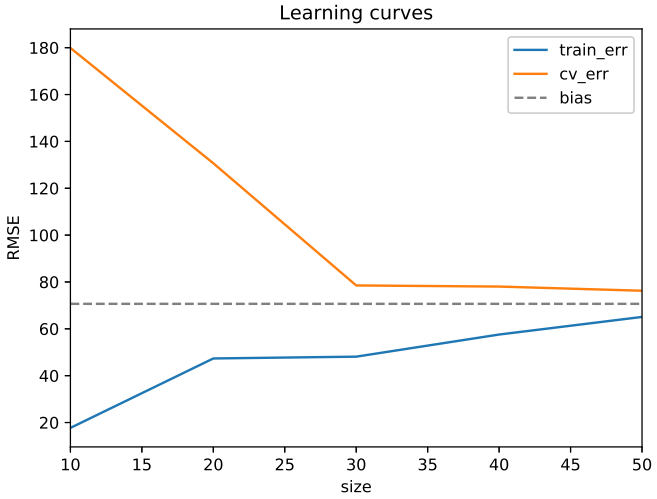

Figure 2: Learning curves of the learned model using an internal 5-fold-cross-validation.

Results are given in Table 1 where I referred to the proposed method as Linear LSR. It is possible to see that indeed Linear RLS is the best model.
In the following I also report the *learning curves* for the learned model w^* in order to seek catches for performance improvements. This curves are produced by taking slice of the training data of fixed size and then performing a 5-fold-CV to evaluate the risk of the predictor using both the training error and the CV-error. Results are given in Figure 2 where it is possible to see that the training error and the cv error close to each other.
This fact reveals that the model does not suffer of overfitting. On the other hand, to reduce the **bias** it may be worth to us by using a more complex model.

Improvements. To reduce the bias, I increased the complexity of the model by adopting a quadratic expansion of the features. Specifically, I mapped each feature vector $x \in \mathbb{R}^d$ to

$$(1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, x_1 x_d, \ldots, x_{d-1} x_d)$$

obtaining a new vector of $2d + \frac{d(d-1)}{2} + 1$. This model is much more complex than the previous one and indeed requires a larger training set to work Ill. Indeed, with the sample training set size used above, its RMSE, after tuning its λ via cross-validation to the optimal value of 0.05, is roughly 81.15, much worse than regularized linear regression. A look at the learning curve in Figure 3 reveals the overfitting: there is a large gap between the training and the cv error.

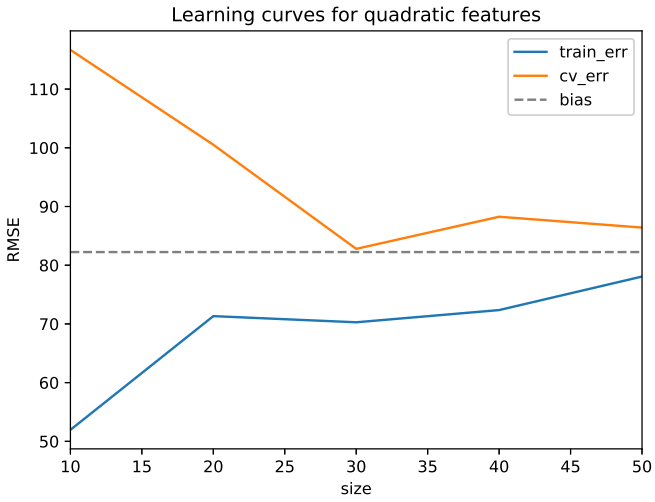

Figure 3: Learning curves of the quadratic RLSR using an internal 5-fold-cross-validation.

Predictor	RMSE
Quadratic RLSR	64.58
Linear RLSR	68.68
RLS	68.66
mean	115.72
median	118.93

Table 2: RMSE on the test data.

To afford such complexity an increase in the sample size is needed. Below the same experiment is reported, this time over a training set of 1500 examples. Tuning of λ has been performed via 5-fold-cv for both regularized linear regression and regularized regression with quadratic features. Results are given in table 2 where the best model is now the quadratic RLSR that achieves to reduce the overall RMSE to 67.72.
For completeness, in Figure 4 is reported the learning curve of the learned quadratic model which reveals that now the model does not overfit anymore.

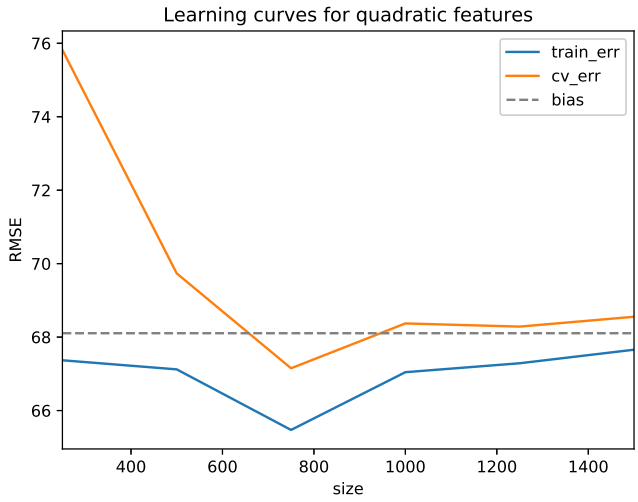

Figure 4: New learning curves of the quadratic RLSR using an internal 5-fold-cross-validation.