Artificial Intelligence II
Assignment 1 Report

Andreas - Theologos Spanopoulos (sdi1700146@di.uoa.gr)

October 5, 2020

Exercise 1

The Ridge Regression loss function is defined as:

$$J(w) \;=\; \mathrm{MSE}(w) + \lambda \frac{1}{2} \sum_{i=1}^{n} w_i^2 \;=\; \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)} \right)^2 + \lambda \frac{1}{2} \sum_{j=1}^{n} w_j^2 \qquad (1)$$

where λ is the regularization parameter. We know that for Linear Regression the hypothesis function $h_w(x)$ is a linear product of the vectors

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Which is calculated as follows:

$$h_w(x) \;=\; w^T x \;=\; \sum_{j=0}^{n} w_j x_j$$

Thus we can compute the partial derivative of $h_w(x)$ w.r.t. any parameter w_k as follows

$$\frac{\partial h_w(x)}{\partial w_k} \;=\; \frac{\partial}{\partial w_k} \left(\sum_{j=0}^{n} w_j x_j \right) \;=\; x_k$$

Now that we have defined some basic terms, we can go ahead and compute the gradient of the loss function $J(W)$:

$$\frac{\partial J}{\partial w} = \begin{bmatrix} \frac{\partial J(w)}{\partial w_0} \\ \frac{\partial J(w)}{\partial w_1} \\ \frac{\partial J(w)}{\partial w_2} \\ \vdots \\ \frac{\partial J(w)}{\partial w_n} \end{bmatrix}$$

Since the loss funtion $J(w)$ is "symmetric" w.r.t. its weights, it is enough to compute only one partial derivative $\frac{\partial J(w_j)}{\partial w_j}$ and then the results generalizes

$$\begin{aligned}
\frac{\partial J(w)}{\partial w_k} &= \frac{\partial}{\partial w_k}\left(\frac{1}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)^2 + \lambda\frac{1}{2}\sum_{j=1}^{n} w_j^2\right) \\
&= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w_k}\left[\left(h_w(x^{(i)}) - y^{(i)}\right)^2\right] + \lambda\frac{1}{2}\frac{\partial}{\partial w_k}\sum_{j=1}^{n} w_j^2 \\
&= \frac{2}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)x_k^{(i)} + \lambda w_k
\end{aligned}$$

for $k = 1, 2.., n$. For $k = 0$, the partial derivate w.r.t. w_0 is the same as the above equation, with the only difference that the last term is 0 (regularization does not apply to the bias parameter).

Therefore, we can now compute the gradient of the loss function as follows:

$$\frac{\partial J}{\partial w} = \begin{bmatrix} \frac{2}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right) \\ \frac{2}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)x_1^{(i)} + \lambda w_1 \\ \frac{2}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)x_2^{(i)} + \lambda w_2 \\ \vdots \\ \frac{2}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)x_n^{(i)} + \lambda w_n \end{bmatrix} = \begin{bmatrix} \frac{2}{m}\sum_{i=1}^{m}\left(w^T x^{(i)} - y^{(i)}\right) \\ \frac{2}{m}\sum_{i=1}^{m}\left(w^T x^{(i)} - y^{(i)}\right)x_1^{(i)} \\ \frac{2}{m}\sum_{i=1}^{m}\left(w^T x^{(i)} - y^{(i)}\right)x_2^{(i)} \\ \vdots \\ \frac{2}{m}\sum_{i=1}^{m}\left(w^T x^{(i)} - y^{(i)}\right)x_n^{(i)} \end{bmatrix} + \lambda \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

This finally leaves us with

$$\nabla J = \frac{2}{m}\left(X^T\left(Xw - Y\right)\right) + \lambda[0 \;\; w_1 \;\; w_2 \;\; ... \;\; w_n]^T$$

where

- X is the **design** matrix of shape $m \times n$, where every row consists of a training example $x^{(i)}$.
- Y is the **label** matrix of shape $m \times 1$, where every row i consists of a label (target value) $y^{(i)}$ for the specific training example $x^{(i)}$.

Exercise 2

This exercise is pretty straightforward. I implemented manually the different gradient descent algorithms. Also, I implelented manually a Grid Search class in order to tune the hyper-parameters. I have added plenty of comments and text in the notebook, to make clear every step I choose to take, and the reason for taking this step. If there is one thing that could be added, it's an explanation of why the model is not underfitting/overfitting. I will do that very briefly here:

- **Why the model is not underfitting**

  We can take a look at the learning curves inside the Notebook to gain insight. If the model was underfitting, then

  1. The training loss would be converging (increasingly) to a non-minimum value.
  2. The validation loss would be very close to the training loss.

  The second bullet matches our graphs, but the first one doesn't.

  The main reasons why a model would underfit are:

  - There are not enough features, that is, the model is too simple.
  - The regularization parameter λ is set too high.

  None of those cases apply to our occasion.

- **Why the model is not overrfitting**

  Again by taking a look at the learning curves inside the Notebook, we can deduce that the model is not overfitting, because if it was, then

  1. The model would not be able to generalize, that is, there would be a notable gap between training loss and the validaiton loss.

  This is not our case, as we can clearly see that in the graphs.

  The main reasons why a model would overfit are:

- There are too many features, that is, the model is too complicated.
- The regularization parameter λ is set too low.
- There are not enough training examples $x^{(i)}$.
- There is no diversity in the training set.

None of those cases apply to our occasion.

Exercise 3

This assignment is also pretty straightforward as well. This time, not a lot of explanation is needed though. I implemented some custom functions (that use regexes) to preprocess the data. Then I used the tfidf from sklearn. Note that I actually tried to construct my own features, but I couldn't get more than 69.2% f1 score. So, after 2 days of banging my head against the wall, I decided to use tdidf, which levels the f1 score at almost 80%. Also note that I implemented my own Grid Search function, as GridSearchCV() from sklearn wasn't working properly. Everything else regarding the implementation can be found in the Notebook.

Resources

https://www.google.com/search?channel=fs&client=ubuntu&q=sklearn

https://www.kaggle.com/enespolat/grid-search-with-logistic-regression

https://www.google.com/search?channel=fs&client=ubuntu&q=nltk