Machine Learning

First assignment solutions

Mobin Nesari
Dr. Hadi Farahani
March 12, 2023

Question 1:

Can gradient descent get stuck in a local minimum when training a logistic regression model? Why?

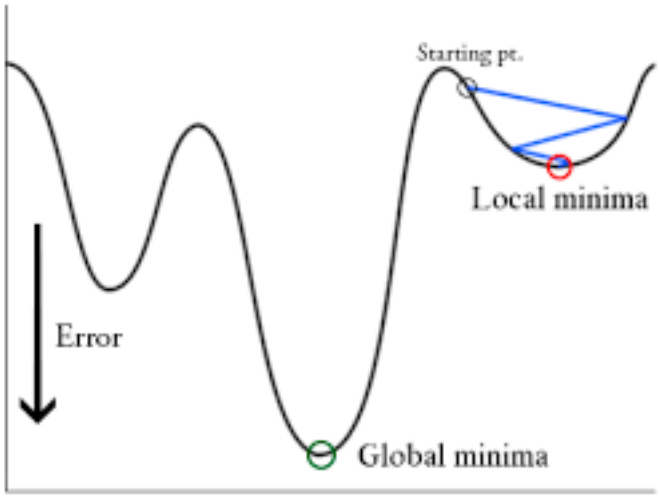
Answer:

The answer is yes. As a matter of fact, solving a logistic regression model requires minimizing a convex loss function. Although we know that in a convex function which means we have one unique global minimum point, due to the shape of loss function, it may has many local optimums.

Now if we check how gradient descent works, we will find out that gradient descent is an iterative approach which makes us one step closer to nearest local optimum. It starts from a initial point and iteratively goes toward minimum. The formula of gradient descent is $\theta = \theta - \alpha \nabla J(\theta)$ where θ is the vector of weights that we want to update, α is the learning rate which controls step size in each iteration, $J(\theta)$ is the loss function, which measures error of the model on the training dataset and $\nabla J(\theta)$ is the gradient of the loss function with respect to the weights.

According to all given facts, if the step size (learning rate) is not sufficient, it will cause gradient descent stuck in local optimum. But there are several ways to avoid getting stuck in gradient descent. One way to avoid getting stuck in gradient descent is to use a good initialization of the weights like random initialization. Another way to escape getting stuck is using a sufficient value for learning rate. The learning rate should be small

enough to avoid over stepping global minima and should be big enough to overshoot local minima. Another solution toward this problem is using other variants of gradient descent which has another attributes like momentum.



Local minima intuition

Question 2:

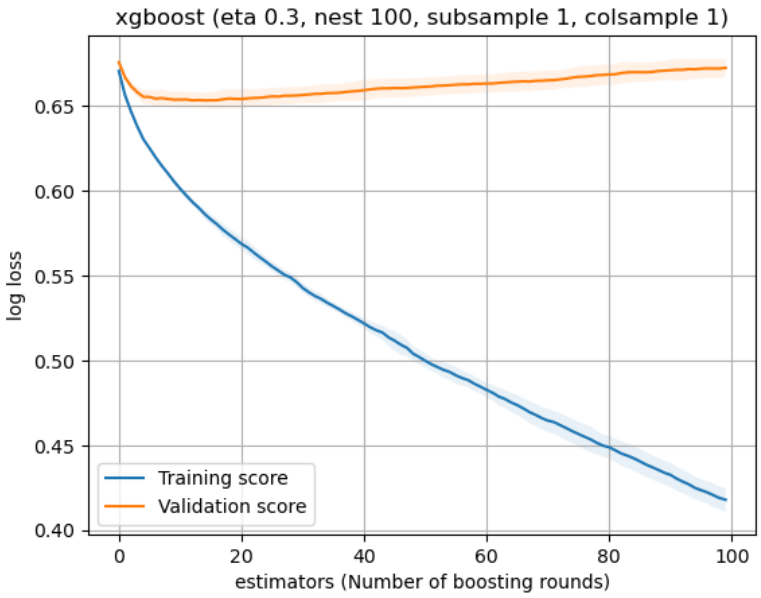
Suppose you are using polynomial regression. You plot the learning curves and you notice that there is a large gap between the training error and the validation error. What is happening? What are three ways to solve this?

Answer:

When there is a big gap between the training error and the validation error, the model can not learn general features and somehow we can say our model is memorizing our data. We call this situation overfitting. First we shall know what is overfitting is and then try to solve it by three famous ways.

In polynomial regression, generalization refers to the ability of the model to accurately predict the output of new, unseen data points. When a polynomial regression model is trained on a set of training data, it learns a function that maps the input features to the output variable. The goal of the model is to find the underlying pattern in the training data that can be used to make accurate predictions on new, unseen data. Accordingly, overfitting occurs when the polynomial regression model becomes too complex and learns the noise or random fluctuations in the training data instead of the underlying pattern. This can happen when the degree of the polynomial is too high relative to the amount of training data, or when the model has too many parameters relative to the complexity of the problem.

When a model overfits the training data, it typically has low training error but high validation error. This is because the model is memorizing the training data and is not generalizing well to new, unseen data. To prevent overfitting, techniques such as **regularization**, **early stopping**, or **cross-validation** can be used. These techniques aim to reduce the complexity of the model and improve its ability to generalize to new data. By reducing overfitting, we can improve the generalization performance of the model and make more accurate predictions on new data.



An instance of overfitting

Now if we want to count three solution for overfitting problem in general, we can count these ones:

I. **Regularization**: Regularization is a technique that adds a penalty term to the loss function to constrain the weights of the model and prevent overfitting. In polynomial regression, one common form of regularization is L2 regularization, which adds a penalty term proportional to the square of the weights. This can help to reduce the complexity of the model and improve generalization performance.

II. **Increase amount of training data**: If the model is overfitting due to lack of data, adding more training data can help to improve the generalization performance. This can be done by collecting more data, or by using techniques such as data augmentation to create synthetic data.

III. **Reduce complexity of the model**: If the model is too complex for the amount of training data, reducing the degree of the polynomial or simplifying the model architecture can help to reduce the variance and improve generalization performance. This can be done by using techniques such as feature selection, dimensionality reduction, or model selection to find the simplest model that fits the data well.

Question 3:

Suppose you are using Ridge regression and you notice that the training error and the validation error are almost equal and fairly high. Would you say that the model suffers from high bias or high variance? Should you increase the regularization hyperparameter α or reduce it?

Answer:

To answer this questions, first we shall study Ridge Regression with its formula and then try to answer given questions.

Ridge regression is a regularized linear regression model in which a penalty term is added to the loss function to prevent overfitting. The penalty term is an L2 regularization term that penalizes the model for having a high weight distribution. The Ridge regression loss function can be written as:

$$L(w) = ||y - Xw||^2 + \alpha ||w||^2$$

Where:

$L(w)$ is the loss function

y is the target variable

X is the input features matrix

w is the vector of model parameters (i.e., weights)

α is the regularization hyperparameter that controls the strength of the penalty term

$||w||^2$ is the L2 norm of the weight vector

The first term in the loss function measures the difference between the predicted output (Xw) and the true output (y), and is the same as the loss function in ordinary least squares (OLS) regression. The second term in the loss function is the penalty term that encourages the model to have smaller weights. The value of α determines the trade-off between minimizing the loss function and minimizing the magnitude of the weights.

Now according to given facts, if the training error and validation error are almost equal and fairly high in Ridge regression, it is an indication that the model is suffering from high bias. High bias means that the model is not complex enough to capture the underlying patterns in the data, and is underfitting the data.

In this case, increasing the regularization hyperparameter (i.e., the strength of the L2 penalty term) may not be the best approach, as it will further constrain the weights and reduce the flexibility of the model. Instead,

we may want to try decreasing the regularization hyperparameter or choosing a less restrictive regularization method, such as L1 regularization or Elastic Net regularization, which can help to increase the flexibility of the model and reduce the bias.

Also, it is important to note that reducing bias may increase the variance of the model, which can lead to overfitting. Therefore, it is important to monitor the validation error and use techniques such as cross-validation to choose the best hyperparameters and prevent overfitting.

Question 4:

Why would you want to use:

- Ridge regression instead of plain linear regression (i.e., without any regularization)
- Lasso instead of ridge regression?
- Elastic net instead of Lasso regression?

Answer:

In practical problems, you would want to use Ridge regression instead of plain linear regression when dealing with a dataset that has high dimensionality, multicollinearity, or when the number of samples is small relative to the number of features.

Multicollinearity is a common problem in linear regression when the input features are highly correlated with each other. This can cause the model to be unstable and result in unreliable coefficient estimates. Ridge regression can help to alleviate this problem by adding a penalty term to the loss function that encourages the model to have smaller weights, reducing the impact of multicollinearity on the model's performance.

In high-dimensional datasets, where the number of features is much larger than the number of samples, the model may overfit the data and have

poor generalization performance. Ridge regression can help to prevent overfitting by adding a regularization term to the loss function that limits the magnitude of the weights.

In addition, Ridge regression can also be useful when there is noise in the data or when there are outliers that can negatively affect the performance of the model. The regularization term helps to reduce the impact of the noise and outliers on the model's performance.

On the other hand, Lasso regression is useful when dealing with high-dimensional datasets that may contain irrelevant or redundant features, and can help to improve the interpretability and predictive performance of the model. It does this by selecting a subset of the most important features for the model, while also regularizing the model to prevent overfitting. However, it may not perform as well as Ridge regression when all the input features are highly correlated with each other (i.e., multicollinearity).

Elastic Net is a regularization technique that combines L1 and L2 regularization, allowing it to handle datasets with correlated features better than Lasso. It selects groups of correlated features instead of just one, improving stability and generalization performance. Additionally, Elastic Net can prevent overfitting by shrinking the magnitude of weights and selecting only the most important features, which is especially useful when the number of features is much larger than the number of samples.

Question 5:

Implement Linear Regression with **Mean Absolute Error** as the cost function from scratch. Compare your results with the Linear Regression module of **Scikit-Learn**.

Answer:

Answer is in Jupyter Notebook file!

Question 6:

Implement Linear Regression using the normal equation as the training algorithm from scratch.

Answer:

Answer is in Jupyter Notebook file!

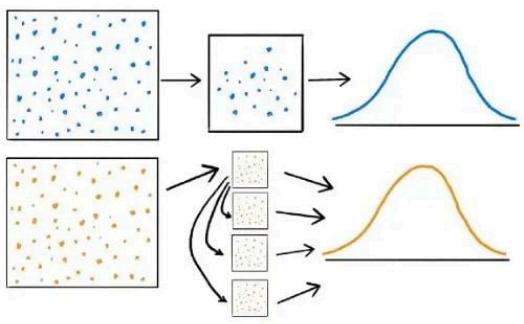
Question 7:

Compare bootstrapping with cross-validation. In which conditions we should use bootstrapping?

Answer:

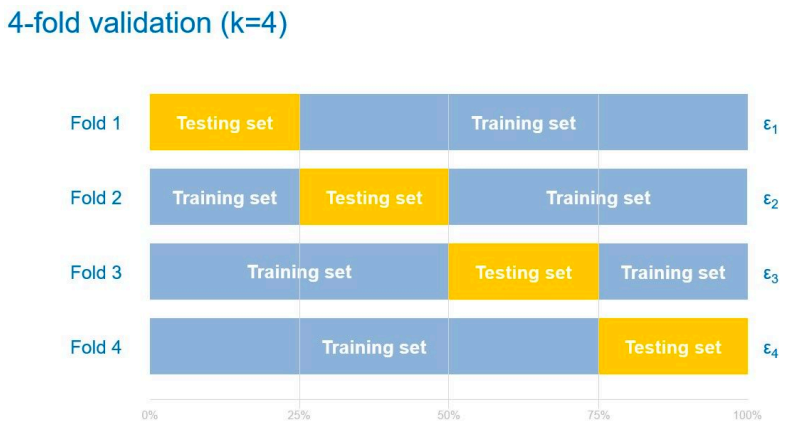
Both Bootstrapping and cross-validation are resampling techniques used for estimating the performance of machine learning models. In this section, first I will introduce each of techniques with their advantages and at the end conclude their use cases.

Bootstrapping involves randomly sampling dataset with replacement to create multiple bootstrap samples. The main advantage of bootstrapping is that it is computationally efficient and can be used for small datasets or when the model training is expensive. It can also be used for testing the stability and variability of the model's performance, as it generates multiple resamples that are used to train and test the model.



Bootstrapping Intuition

On the other hand, cross-validation involves splitting the dataset into multiple folds and using each fold for testing and the rest for training in a rotation. Cross-validation technique is a more robust and reliable method for estimating the model's performance, as it uses multiple non-overlapping folds for training and testing, thus reducing the risk of overfitting. It also provides more accurate estimates of the model's generalization error and can be used for model selection and hyperparamter tuning.



Cross-Validation Intuition

As expressed, bootstrapping is more suitable for small databases and when computational resources are limited, while cross-validation is more suitable for larger datasets and when a more accurate estimate of the model's performance is needed. However, the choice between the two methods ultimately depends on the specific problem and the goals of the analysis.

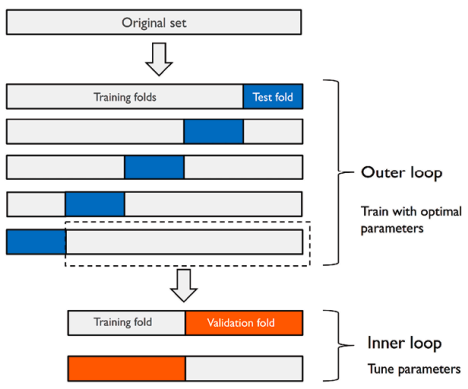
Question 8:

Explain nested cross-validation and 5x2 cross-validation in detail and when we should use them.

Answer:

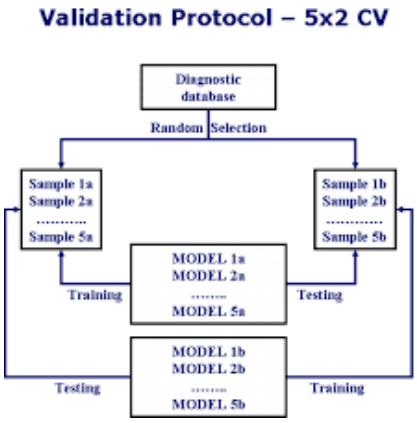
As expressed in previous question, cross-validation is a technique for evaluating the performance of a machine learning model. The main idea is to divide the data into training and validation sets multiple times and compute the average performance score. There are different types of cross-validation techniques, and two of them are nested cross-validation and 5x2 cross-validation.

Nested cross-validation is a technique that involves using cross-validation in two nested loops. The outer loop is used to split the data into training and testing sets, while the inner loop is used to perform another cross-validation to select the best hyperparameters for the model. In the inner loop, the data is divided into training and validation sets, and the model is trained with different hyperparameters. The best hyperparameters are then selected based on the performance on the validation set. The selected hyperparameters are then used to train the model on the outer loop training set, and the performance is evaluated on the outer loop test set. The advantage of nested cross-validation is that it provides an unbiased estimate of the model's performance, as it ensures that the hyperparameters are selected based on the performance on unseen data.



Nested Cross-Validation Intuition

5x2 cross-validation is a variant of k-fold cross validation, where k is set to 2 and the data is divided into 5 pairs of training and testing sets. In each pair, the data is randomly divided into two sets, and the model is trained on one set and evaluated on the other set. This process is repeated 5 times, resulting in 10 performance scores. The final performance score is the average of the 10 scores. The advantage of 5x2 cross-validation is that it provides a more precise estimate of the model's performance, as it uses multiple testing sets. It is particularly useful when the dataset is small and the model has a high variance.



5x2 Cross-Validation Intuition

Overall, nested cross-validation is recommended when hyperparameter tuning is required and when the dataset is small. On the other hand, 5x2 cross-validation is recommended when the dataset is small and the model has a high variance. In general, it is a good measure to use cross-validation to evaluate the performance of a model, as it provides a more reliable estimate of the model's performance than a single train-test split.

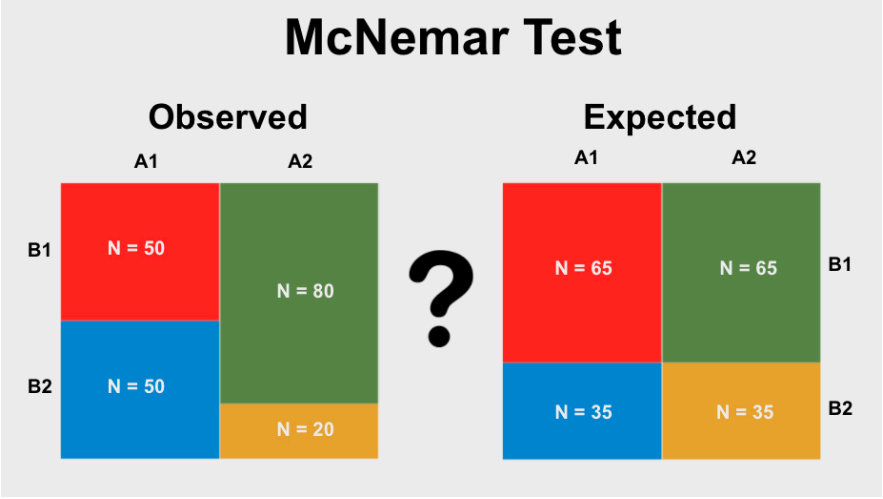
Question 9:

How can we compare different models using statistical significance tests?

Answer:

Statistical Significance test can be used to compare and evaluate the performance of different models. In this section, I will introduce three most famous ones individually and then conclude all given facts in one paragraph.

McNemar test is used to compare the performance of two models on a binary classification task. It compares the number of instances that are correctly classified by one model but misclassified by the other model, and vice versa. If the difference in error rates between the models is statistically significant, then one model is considered better and pioneer than the other one.



McNemar Test Intuition

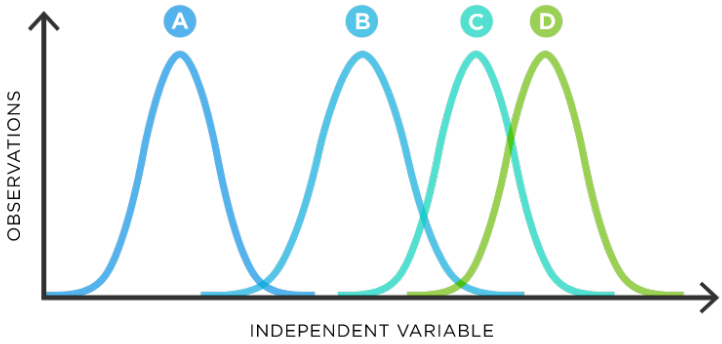
Paired t-test is used to compare the mean performance of two models on a continuous outcome variable, such as RMSE or R-squared. It assumes that the difference between the performance of the two models follows a normal distribution. The test compares the mean difference to zero and determines if the difference is statistically significant.

$$t = \frac{\sum d}{\sqrt{\frac{n(\sum d^2)-(\sum d)^2}{n-1}}}$$

where d: difference per paired value
n: number of samples

Paired T-Test Intuition

Analysis of Variance (ANOVA) is a more general test that can be used to compare the performance of multiple models on a continuous outcome variable. It tests the null hypothesis that there is no significant difference between the means of the different models. If the null hypothesis is rejected, then at least one of the models is considered to have a significantly different mean performance than the others.



When interpreting the results of statistical significance tests, it is important to consider the assumptions of the test and the practical significance of the results. A statistically significant difference between two models does not necessarily mean that one model is always better than the other in practice. It is also important to consider factors such as computational complexity and interpretability when choosing a model for specific task.

Question 10:

Implement Forward and Backward feature selection algorithm from scratch with **MSE** as the metric.

Answer:

Answer is in Jupyter Notebook file!

Question 11:

Suppose the features in your training set have very different scales. Which algorithms (Gradient Descent, Normal Equation, SVD) might suffer from this, and how? What can you do about it?

Answer:

In machine learning and many machine learning algorithms, to bring all the features in same standing is a critical point to consider to achieve a novel model which solve our problem. Hence, we need to do scaling so that one significant number doesn't impact the model just because of their large magnitude.

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and pioneer one.

Among given algorithms and approaches, Gradient Descent is greatly suffering from different scales. The reason behind this fact is in middle of gradient descent algorithm. In gradient descent we derive the cost function and indicates each part of our model's error and trying to amend it by heading toward minimum cost. Therefore, if a feature has very large scale numbers in comparison to other features, then our steps toward minimum

point will be biased by this feature and according to this, model will take much longer to reach the global optimum.

Normal equation, on other hand is robust and is not effected by different scales features. The reason behind this attribute in normal equations is how normal equations work. Normal equation is an analytical approach toward linear regression with a least squares cost function. We can find the value of θ without using iterative methods like gradient descent. The normal equation formula as follows : $\theta = (X^T X)^{-1} . (X^T y)$ where θ is hypothesis parameters that define it the best, X is input features and y is output value. In this approach we calculate parameters directly with higher computational cost but robust to different scales problem.

Singular Value Decomposition (SVD) like normal equation is robust to different scale features. The cause of this attribute is SVD is nothing but some sum and multiplication in some matrices which each column play it's own role in algorithm and other elements won't effect by high value of another feature.

However, different scale problem will be solved easily normalization and standardization. Normalization is used to to transform features to be on a similar scale. The new point in this transformation will be calculated by $\frac{x - x_{min}}{x_{max} - x_{min}}$. Standardization or Z-score normalization is the transformation of features by subtracting from mean and dividing by standard derivation. The calculation formula is as follows: $\frac{x - \bar{x}}{\sigma}$.

Question 12:

In this part, you are going to work with the News Popularity Prediction dataset. You will implement a regression model using the **Scikit-Learn** package to predict the popularity of new articles (the number of times they will be shared online) based on about 60 features. You are expected:

- Perform exploratory data analysis on the dataset.
- Propose 5 different hypothesis tests related to the dataset. At least use 3 different tests.
- Try Ridge and Lasso regression.
- Use various scaling methods and report their effects.
- Add polynomial features and report their effect.
- Try using GridSearchCV with RandomizedSearchCV to tune your model's hyperparameters. ***(Extra Point)***
- Apply the feature selection methods that you have implemented in the above sections.
- Get familiar with and implement the following loss functions from scratch and utilize them with a Linear Regression model and discuss their effect on the performance of the model. ***(Extra Point)***
    - Absolute Error
    - Epsilon-sensetive error
    - Huber

Answer:

Question 13:

Implement batch gradient descent with early stopping for softmax regression from scratch. Use it on a classification task on the Penguins dataset.

Answer:

Answer is in Jupyter Notebook file!