Task 14

Exercises

1. What Linear Regression training algorithm can you use if you have a training set with millions of features?

Answer: For a training set with millions of features, the Normal Equation approach becomes computationally infeasible due to the matrix operations involved. In such cases, Stochastic Gradient Descent (SGD) or Mini-Batch Gradient Descent are more appropriate. These algorithms can handle large datasets efficiently by updating the model parameters iteratively, rather than solving the equation directly.

2. Suppose the features in your training set have very different scales. What algorithms might suffer from this, and how? What can you do about it?

Answer: Algorithms that rely on distance measurements or gradient-based optimizations, such as Linear Regression, Logistic Regression, Support Vector Machines (SVMs), and k-Nearest Neighbors (k-NN), can suffer from features with different scales. The algorithms may give undue weight to features with larger scales, leading to suboptimal performance. To address this issue, you can normalize or standardize the features. Normalization scales the features to a range of [0, 1], while standardization scales the features to have a mean of 0 and a standard deviation of 1.

3. Can Gradient Descent get stuck in a local minimum when training a Logistic Regression model?

Answer: No, Logistic Regression has a convex cost function, which means there are no local minima, only a single global minimum. Gradient Descent will not get stuck in a local minimum for Logistic Regression; it will always converge to the global minimum, given a proper learning rate.

4. Do all Gradient Descent algorithms lead to the same model provided you let them run long enough?

Answer: In theory, yes. If the learning rate is properly tuned and the algorithms run for an adequate number of iterations, Batch Gradient Descent, Stochastic Gradient Descent (SGD), and Mini-Batch Gradient Descent should all converge to the same model. However, practical differences can arise due to the randomness in SGD and Mini-Batch Gradient Descent and the learning rate schedule.

5. Suppose you use Batch Gradient Descent and you plot the validation error at every epoch. If you notice that the validation error consistently goes up, what is likely going on? How can you fix this?

Answer: If the validation error consistently goes up, it is likely that your model is overfitting the training data. To fix this, you can:

- Reduce the complexity of the model (e.g., by using fewer features or a simpler model).
- Increase regularization (e.g., add an L2 regularization term in Linear/Logistic Regression).
- Use early stopping to halt training when validation error starts to increase.

6. Is it a good idea to stop Mini-batch Gradient Descent immediately when the validation error goes up?

Answer: No, it is not always a good idea to stop immediately because the validation error can fluctuate due to the stochastic nature of Mini-batch Gradient Descent. A better approach is to use early stopping with a patience parameter, which allows some tolerance for fluctuations before stopping.

7. Which Gradient Descent algorithm (among those we discussed) will reach the vicinity of the optimal solution the fastest? Which will actually converge? How can you make the others converge as well?

Answer:

- Stochastic Gradient Descent (SGD) will reach the vicinity of the optimal solution the fastest due to frequent updates, which lead to quicker movement towards the minimum.
- Batch Gradient Descent will actually converge given enough time, as it moves steadily towards the minimum.
- Mini-Batch Gradient Descent balances the trade-offs between SGD and Batch Gradient Descent, reaching the vicinity quickly and converging relatively well.

To make SGD and Mini-Batch Gradient Descent converge as well, you can use learning rate schedules (e.g., gradually decreasing the learning rate) or momentum techniques.

8. 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: A large gap between the training error and the validation error indicates high variance (overfitting). Three ways to address this are:

1. Reduce the degree of the polynomial to decrease model complexity.
2. Increase the size of the training set to provide more data for training.
3. Add regularization (e.g., Ridge or Lasso Regression) to penalize large coefficients and reduce overfitting.

9. 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: If both the training error and the validation error are almost equal and fairly high, the model suffers from high bias (underfitting). In this case, you should reduce the regularization hyperparameter α to allow the model more flexibility to fit the data better.

10. 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?**

Answer:

- Ridge Regression instead of plain Linear Regression (i.e., without any regularization)?
  Ridge Regression helps prevent overfitting by adding an L2 regularization term that penalizes large coefficients, which is especially useful when dealing with multicollinearity or when you have a large number of features.
- Lasso instead of Ridge Regression?
  Lasso (L1 regularization) not only helps prevent overfitting but also performs feature selection by driving some coefficients to exactly zero, thus producing a simpler and more interpretable model.
- Elastic Net instead of Lasso?
  Elastic Net combines both L1 and L2 regularization, making it more robust when dealing with correlated features. It balances the benefits of Ridge (L2) and Lasso (L1) regularization, often providing better performance when features are highly correlated.

11. Suppose you want to classify pictures as outdoor/indoor and daytime/nighttime. Should you implement two Logistic Regression classifiers or one Softmax Regression classifier?

Answer: You should implement two Logistic Regression classifiers because you have two binary classification problems (outdoor vs. indoor and daytime vs. nighttime). Softmax Regression is used for multiclass classification, where each instance belongs to one of three or more classes.