COMP 551 - PROJECT 2 REPORT

Multi-Class Logistic Regression and Gradient Descent

Group 8
Zhengwen Fu (260856256), Linda Cai(260720706), Shuntian Xiao (260823600)
November 22, 2020

Abstract

In this project, a multi-class logistic regression algorithm and a mini-batch gradient descent optimizer with momentum are implemented for multi-class classification. The learning algorithm with the optimizer is then applied to two datasets for two corresponding supervised machine learning tasks: handwritten digit recognition and iris species taxonomy. The best combination of hyperparameters for the model is searched first. Then, the hyperparameters are altered one at a time to analyze the effect of each hyperparameter on the model. The result shows that the rate of convergence as well as divergence is influenced by the learning rate, whose optimal value resides somehow in an equilibrium state; the amount of oscillations can be controlled by the momentum, whose optimal value inclines towards a higher proportional magnitude; lastly, the efficiency of weights-update as well as the oscillation level is monitored by the batch-size, whose optimal value stays close to the turning point of its training time graph. Furthermore, the termination condition of the optimizer is carefully designed to give better performance in less time. Then, the learning algorithm with the optimizer is compared against k-nearest neighbours algorithm which is shown to be more accurate on less linear datasets and much slower in predicting time in the result. Finally, the effect of the L2 regularization on the algorithm is also studied. The result demonstrates that the algorithm performs best without the L2 regularization penalty.

1 Introduction

Multi-class classification tasks refer to classification tasks with more than two class labels. In machine learning, there are many classification algorithms to tackle multi-class classification, for instance, multi-class logistic regression (i.e. softmax regression) [1] and k-nearest neighbours (i.e. KNN) [1]. As the no free lunch theorem [2] states for machine learning, there is no model that can perform well for every problem. Hence the goal is to grasp the dependency of the result on the hyperparameter tuning, as well as to perceive the subtle performance differences across several classifiers for specific multi-class classification problems.

In this project, a multi-class logistic regression algorithm [1] and a mini-batch gradient descent optimizer with momentum [1] are implemented for multi-class classification. Softmax regression is a generalization of logistic regression based on the softmax function instead of the logistic function. While the logistic regression is for binary classification, the softmax regression extends to multi-class classification. One-hot encoding [1] is used here to convert the categorical labels into computable vectors. Moreover, the mini-batch gradient descent with momentum is the optimizer which enables the cost of the model to converge to a minimum iteratively with a noisy-fast estimate of the gradient and a carefully chosen termination condition.

The learning algorithm with the optimizer is applied to two datasets. One dataset is for the recognition of handwritten digits [3] and the other is for the taxonomy of iris [4]. 5-fold cross-validation [5] is used as the validation technique to investigate the performance of the model in terms of the accuracy. A simple grid search is applied first to select the best combination of hyperparameters for the model. Then, the hyperparameters are altered one at a time to analyze the effect of each hyperparameter on the model in terms of validation accuracy, training accuracy, and training time. Besides, the termination condition of the optimizer is designed to stop when the validation accuracy did not improve for consecutive 20 iterations. Furthermore, with the optimal hyperparameters found before, this discriminative softmax regression algorithm with the optimizer is compared against the non-parametric KNN algorithm from Scikit-learn [3] with optimal k. Their performance differences for two datasets in validation accuracy, training accuracy, and training time are analyzed. Moreover, the effect of L2 regularization [6] on the algorithm in terms of validation accuracy is also studied.

2 Datasets

The digits dataset is from Scikit-Learn [3], and it contains 1798 data points where each datapoint is a 8×8 gray-scale image of a handwritten digit from one of the ten label classes where each class refers to a digit (0 to 9). There are approximately 180 samples in each class, and every feature is an integer from 0 to 16.

The iris dataset from Openml [4] is a dataset smaller than the digits dataset, and it contains 3 classes of iris with 50 data points for each class. Each label class refers to one of three types of iris plant: Iris-setosa, Iris-versicolour, or Iris-virginica. There are four features in this dataset: sepal length in cm, sepal width in cm, petal length in cm, and petal width in cm. There are no missing values in the features. The string labels of this dataset are converted into numerical values for easy one-hot encoding later.

3 Results

3.1 Hyperparameters of the Optimization Procedure

Hyperparameter tuning assists to enhance the model by exploring an optimal value for the performance. In this project, the learning rate α, the momentum β and batch-size $\mathbb{B}$ are investigated as the targeted hyperparameters. The optimal combinations of them giving the best validation accuracy for the datasets are found by using the simple grid search, as shown in Table 1. Then, by altering one hyperparameter at a time, the hyperparameters are analyzed based on the changes in training accuracy, validation accuracy, and training time.

	Optimal Validation Accuracy	Learning Rate α	Momentum β	Batch-Size $\mathbb{B}$
Digits Dataset	0.975	0.4	0.8	21
Iris Dataset	1.000	0.6	0.8	31

Table 1: Optimal accuracy and corresponding combination of hyperparameters for datasets

3.1.1 Learning Rate (Plots in Appendix A)

Through a close examination of the validation and training performance graphs in Digits dataset (Fig.1, Fig.2), the performance is steady in the range of $\alpha \in [0.1, 0.9]$, and similarly for the Iris dataset (Fig.4, Fig.5). When the α is extremely small, the performance on both the test and training sets behave poorly. Usually, an overly small learning rate may take too long to converge therefore reaching only to a sub-optimal solution. As α increases, the performance improves and stabilizes. Towards α equal to 1, the Iris dataset is unaffected because it is linearly separable and thus large α still results in finite step convergence; however, the Digits dataset reveals a consequence with an overly large α: the oscillation in the cost function leads to divergence, which yields inaccurate weights. Such eventual decline in quality of result conveys a defining trait of the learning rate hyperparameter: generally it requires some form of equilibrium in order to obtain the optimal value. The optimal α's in the plots (Fig.1, Fig.4) are close to the α's in the optimal sets in Table 1: $\alpha_{plot,Digit} = \alpha_{set,Digit} = 0.4$ and $\alpha_{plot,Iris} = 0.5, \alpha_{set,Iris} = 0.6$. For the training time analysis, the graph shows fluctuations over relatively small ranges of training time: [0.025,0.225] in the Digits dataset (Fig.3) and [0.0065,0.0100] in the Iris dataset (Fig.6).

3.1.2 Momentum (Plots in Appendix B)

In the Digits dataset, the graphs in Fig.7 and Fig.8 convey a steady growth of training and validation accuracy when the momentum β increases. Unlike the learning rate α, which can hurt the performance if assigned an extremely small value, momentum is a hyperparameter that attempts to enhance the optimization process, which explains why the performance curve at $\beta = 0$ starts off high. As β reaching near the value of 1, the curve suddenly shows a sharp decline. The greater the momentum, the more influence by the recent gradient weight. So an exceedingly large momentum causes a loss in accuracy due to overly high bias towards the latest gradient components. A similar pattern is portrayed by the Iris dataset in a more bumpy version. The optimal β's in the plots (Fig.7, Fig.10) are close to the β's in the optimal sets in Table 1: $\beta_{plot,Digit} = \beta_{set,Digit} = 0.8$ and $\beta_{plot,Iris} = 0.7, \beta_{set,Iris} = 0.8$. Despite that the training time graphs differ (Fig.9 and Fig.12) and fluctuate with mini hills, these mini hills' downhill slopes seem slightly longer than the uphill slopes during certain arbitrary intervals. So the training time perhaps is decreasing and thus increasing the momentum to some extent indeed accelerates the learning process.

3.1.3 Batch-Size (Plots in Appendix C)

With a small batch size, the path of gradient descent is driven by noisy estimates. The training and validation curves of the Digits dataset (Fig.13, Fig.14) demonstrate how noises can lead to stalling, as a batch size of 1 yields the lowest accuracy. As $\mathbb{B}$ increases, both performance curves increase drastically during the approximate interval of [0,10], and then stabilizes because the magnitude of $\mathbb{B}$ is sufficient to help the model converge. The optimal $\mathbb{B}$'s in the plots (Fig.13, Fig.16) are close to the $\mathbb{B}$'s in the optimal sets in Table 1: $\mathbb{B}_{plot,Digit} = \mathbb{B}_{set,Digit} = 21$ and $\mathbb{B}_{plot,Iris} = 21, \mathbb{B}_{set,Iris} = 31$. By investigating the training time v.s $\mathbb{B}$ plots in (Fig.15, Fig.18), the time graphs from the two distinct datasets finally cohere, as they share a close graphical representation and rate of decay, though at different size of training time scales. Furthermore, for both datasets, the optimal values of batch size are close to where the turning points locate in the respective time graphs. An optimal $\mathbb{B}$ ought to reduce the noise while preserving an acceptable test and training accuracy.

3.1.4 Common Characteristic

In each plot for altering one hyperparameter, it can be observed that the optimal point in the plot is not necessarily the value of that hyperparameter in the optimal set, but generally very close. This can be explained by the mini-batch gradient descent. In this step, the data are shuffled before splitting and the parameters are optimized by the mini batches of random data points. Hence a certain degree of randomness and noise are introduced to the results. Thus, it is not guaranteed that the results for different trials are exactly the same.

3.2 Termination Condition

The termination condition of optimization for the best validation accuracy is achieved by inputting a validation dataset to the optimizer. Then, in the loop of the optimizer, the set of parameters giving the best validation accuracy is always recorded and the loop breaks when the validation accuracy has not been improved in the past 20 iterations. This strategy turns out to be really useful, because it decreases the training time by avoiding redundant optimization and returns the model with the best validation accuracy.

3.3 Comparison Against Another Classifier (Plots in Appendix D)

As shown in Table 1, Fig.19, and Fig.22, in terms of accuracy, the K-nearest neighbours is better than the softmax regression in digit dataset and is worse in iris dataset. This shows that the decision boundaries of the iris dataset are more linear and KNN can be accurate on less linear datasets. In terms of predicting time, the K-nearest neighbours classifier is much slower than the softmax regression, because KNN is a lazy learner which doesn't learn any specific parameter and predicts the label by finding the most similar example in the training set.

3.4 Investigating Regularization (Plot and Table in Appendix E)

As shown in Fig.25 and Table 2, the model achieves the best validation accuracy when lambda equals 0, i.e. with no L2 regularization penalty. This is actually expected, because the bases for the model are meaningful simple features and thus the chance of overfitting is low.

4 Discussion and Conclusion

The learning rate, momentum, and batch size are the critical hyperparameters for the performance of the softmax regression. The mini-batch gradient descent inevitably adds certain randomness to the results. A carefully designed termination condition for the optimizer improves both the efficiency and the accuracy of the model. The softmax regression is much faster in prediction comparing to KNN, while KNN can deal with datasets with less linear decision boundaries. L2 regularization penalty is not necessary for the normal softmax regression.

For future work, it would be a good idea to visualize the datasets and evaluate how linear their approximate decision boundaries are, since the softmax regression is accurate on datasets with linear decision boundaries. Besides, as the plots for the hyperparameters reveal a lot of characteristics to investigate, the details of hyperparameter tuning may can be further studied through the help of weight space visualization.

5 Statement of Contributions

Zhengwen Fu contributed to the gradient descent, softmax regression, performance analysis and report writing.
Shuntian Xiao contributed to the K-fold cross validation, grid search, and report editing.
Linda Cai contributed to the hyperparameters analysis, research, and report writing.

References

[1] K. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2012. [Online]. Available: https://books.google.ca/books?id=RC43AgAAQBAJ 1

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997. 1

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 1, 2

[4] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013. [Online]. Available: http://doi.acm.org/10.1145/2641190.2641198 1, 2

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001. 1

[6] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006. 1

A Learning Rate Plots

Digits Dataset

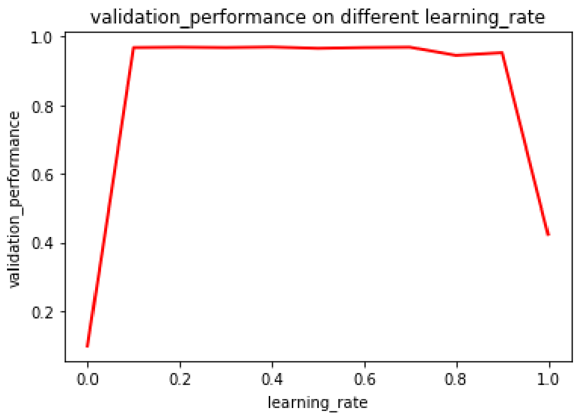

Figure 1: learning rate is 0.4 with validation performance 0.9693887341380378

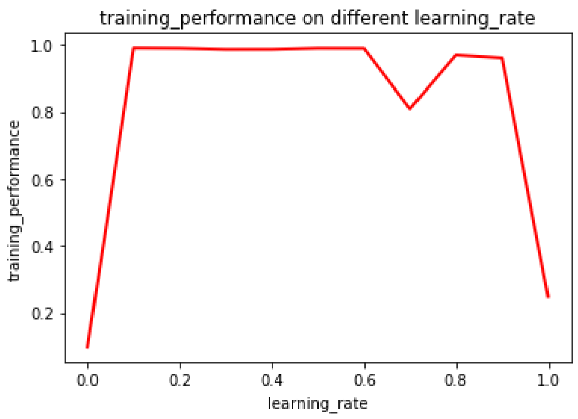

Figure 2: learning rate is 0.1 with training performance 0.9904012086685773

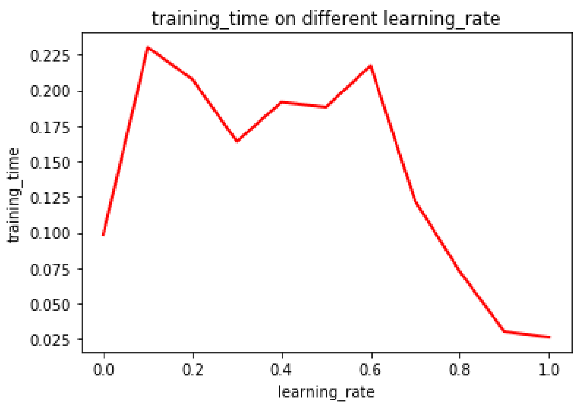

Figure 3

Iris Dataset

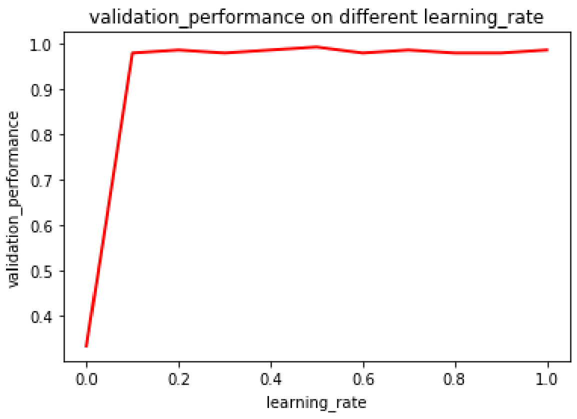

Figure 4: learning rate is 0.5 with validation performance 0.9933333333333334

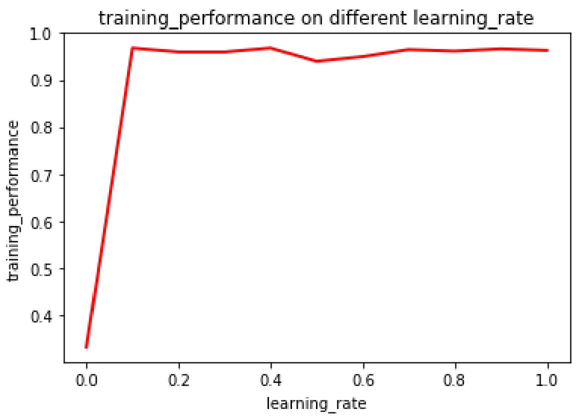

Figure 5: learning rate is 0.1 with training performance 0.9683333333333334

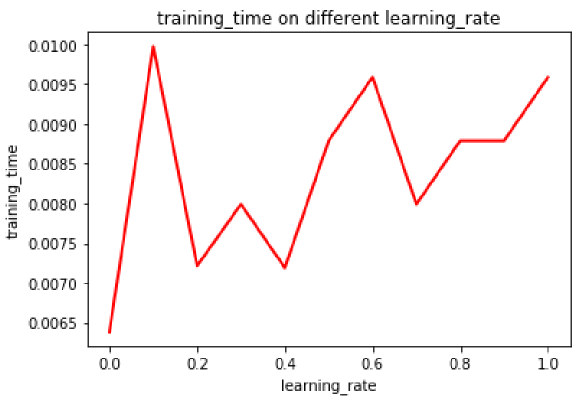

Figure 6

B Momentum Plots

Digits Dataset

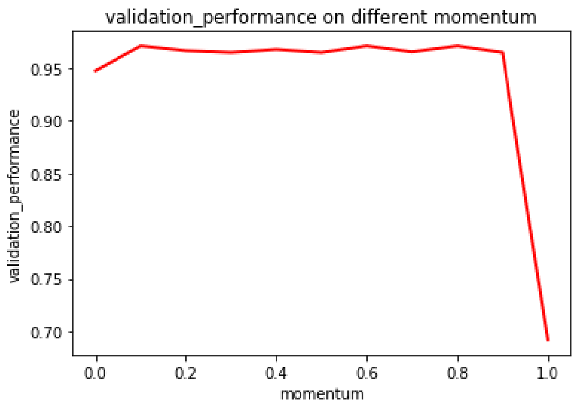

Figure 7: momentum is 0.7999999999999999 with validation performance 0.9705060352831941

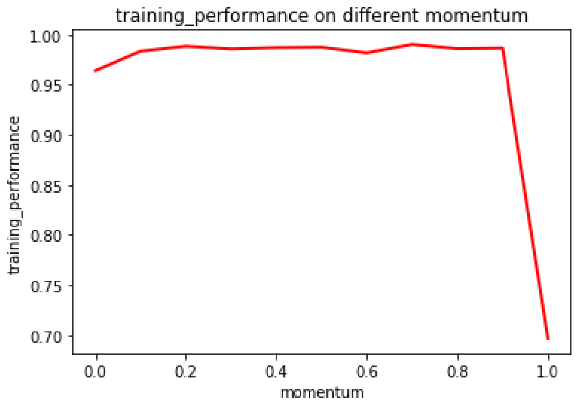

Figure 8: momentum is 0.7 with training performance 0.9904003375909671

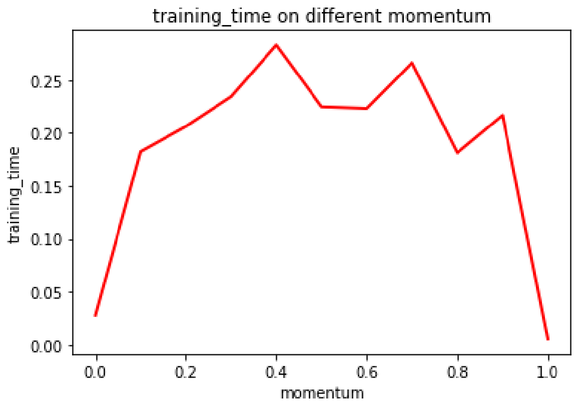

Figure 9

Iris Dataset

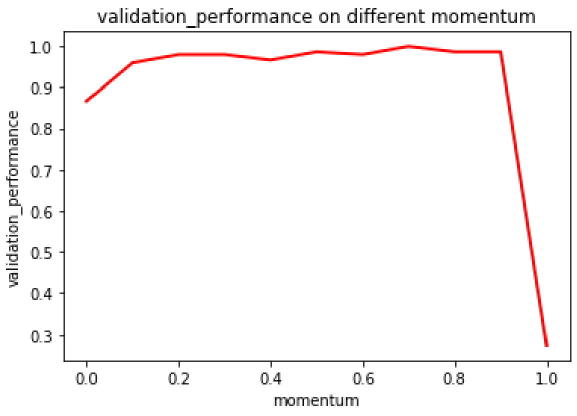

Figure 10: momentum is 0.7 with validation performance 1.0

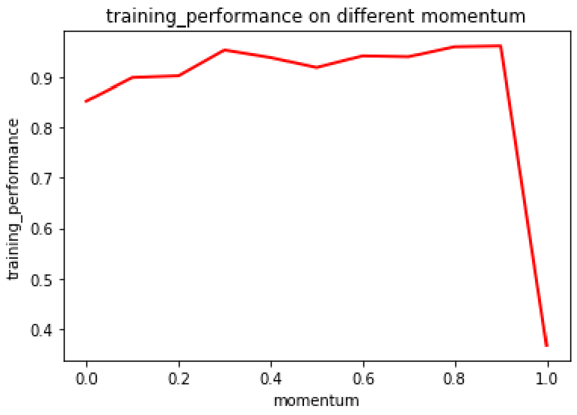

Figure 11: momentum is 0.8999999999999999 with training performance 0.9633333333333333

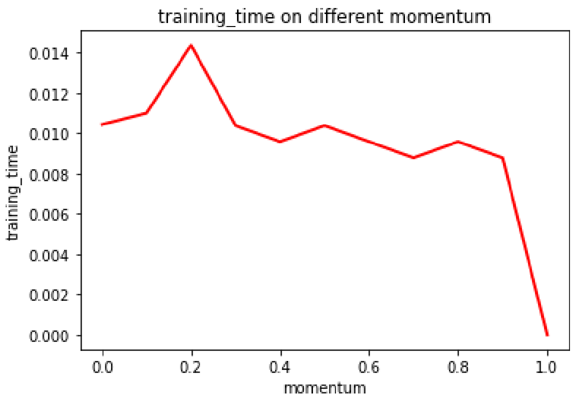

Figure 12

C Batch Size Plots

Digits Dataset

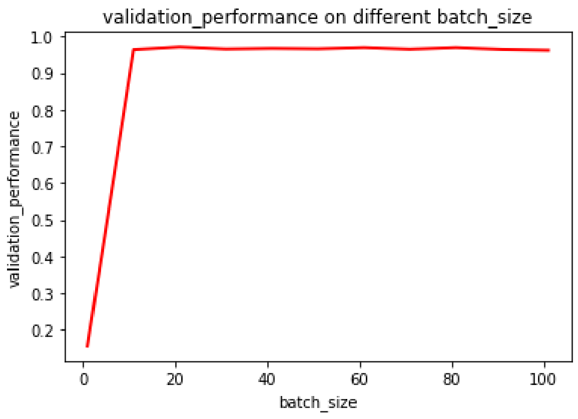

Figure 13: batch size is 21 with validation performance 0.9710662333642835

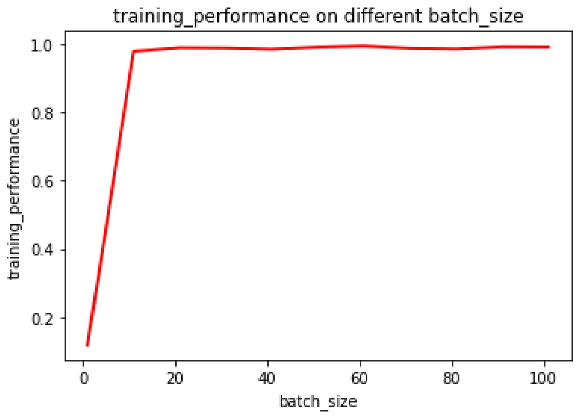

Figure 14: batch size is 61 with training performance 0.99401743897737544

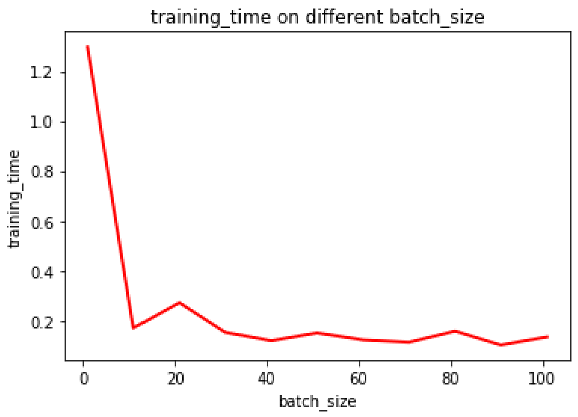

Figure 15

Iris Dataset

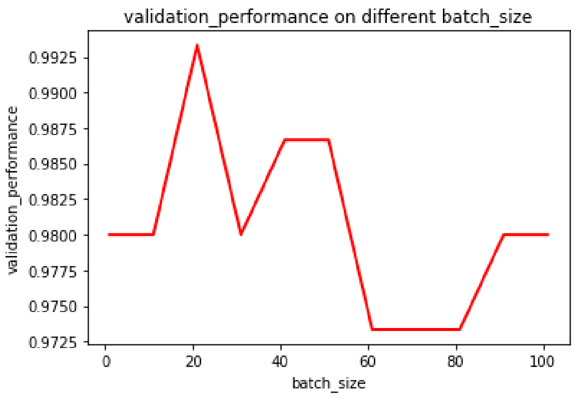

Figure 16: batch size is 21 with validation performance 0.9933333333333334

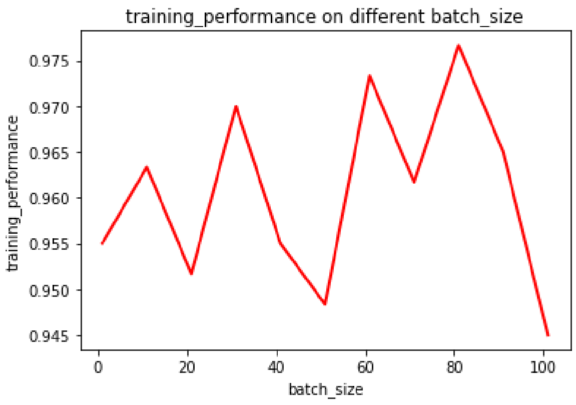

Figure 17: batch size is 81 with training performance 0.9766666666666666

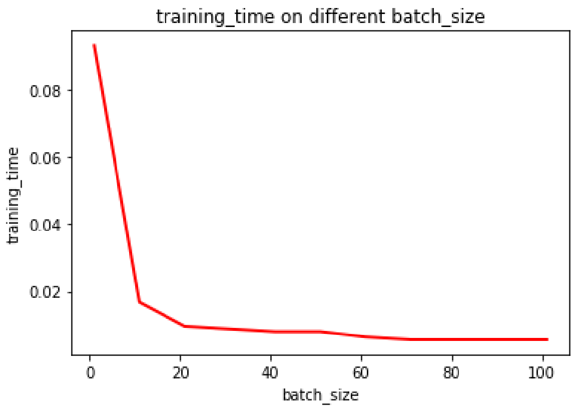

Figure 18

D KNN Classifier Plots

Digits Dataset

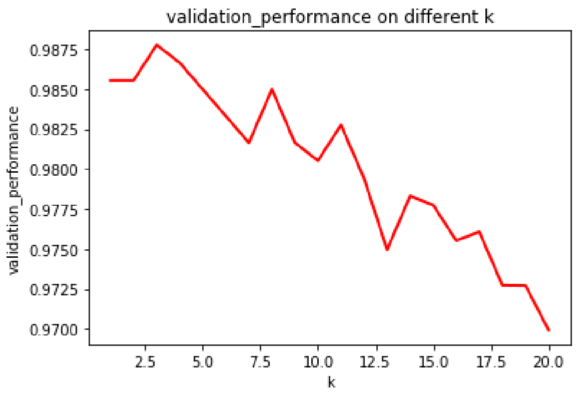

Figure 19: The optimal value for k is 3 with validation performance 0.9877514701330856

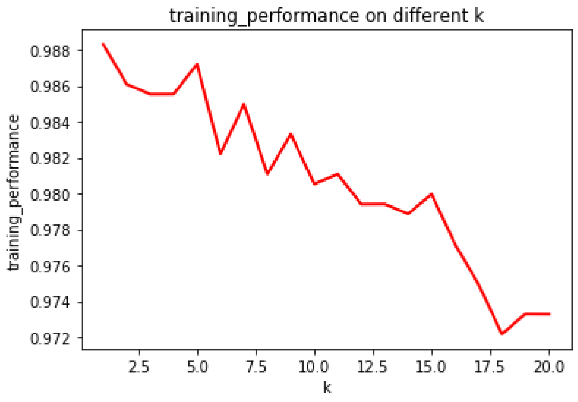

Figure 20: The optimal value for k is 1 with training performance 0.988310120705664

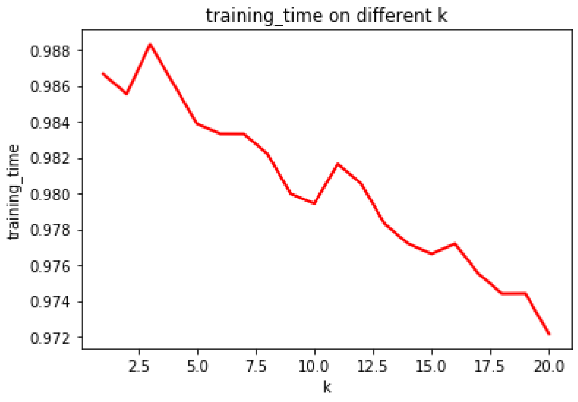

Figure 21

Iris Dataset

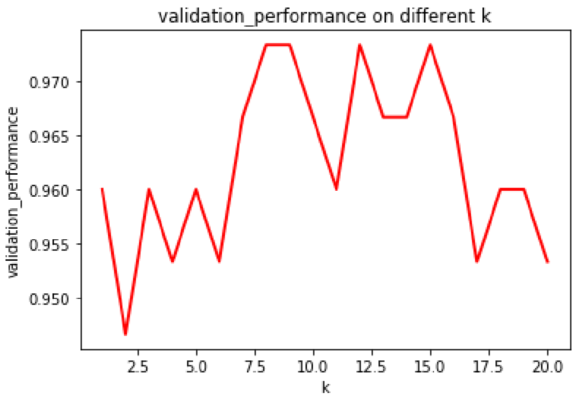

Figure 22: The optimal value for k is 8 with validation performance 0.9733333333333334

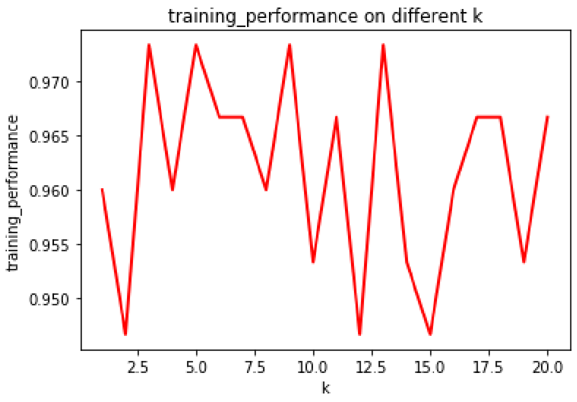

Figure 23: The optimal value for k is 3 with training performance 0.9733333333333334

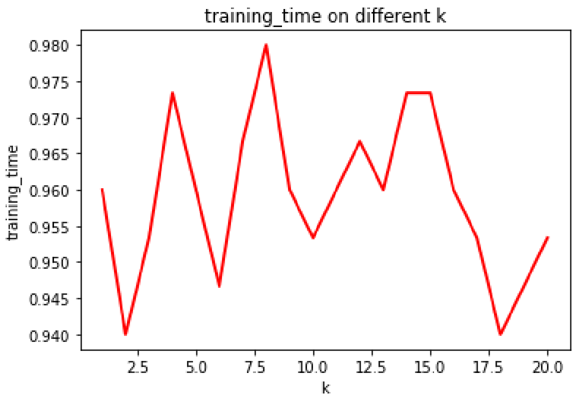

Figure 24

E L2 Regularization Plot & Table

Digits Dataset

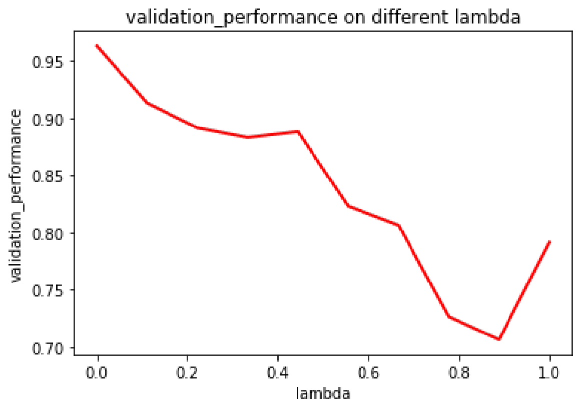

Figure 25: The optimal value for lambda is 0 with validation performance 0.9632721202003339

	Optimal Validation Accuracy
With L2 Regularization	0.963
Without L2 Regularization	0.963

Table 2: Optimal validation accuracy with or without L2 Regularization for Digits dataset