EasyChair Preprint

№ 4489

Optimization of Regularization and Early Stopping to Reduce Overfitting in Recognition of Handwritten Characters

Naman Jain

October 30, 2020

Optimization of Regularization and Early Stopping to reduce Overfitting in Recognition of Handwritten Characters

[1]Naman Jain

[1] Symbiosis University of Applied Sciences, Indore, India
namanj.1298@gmail.com

Abstract:

Character recognition of handwritten texts is one of the most crucial aspects of any pattern recognition algorithm. The application of digit recognition is very vast, some examples but not limited to, include postal mail sorting, digitizing any hand filled form, bank check processing, signature verification, etc. For the purpose of this research, logistic regression was used to train a model to accurately identify the input digits/alphabets. Also, to improve accuracy of the model, parameter optimization was done on two major factors of logistic regression. The two parameters, the regularization parameter (lambda) and the number of iterations (during training) were closely studied and were optimized to eliminate any chance of Overfitting. Experimental results show that an optimized set of parameters would provide maximum accuracy on the test set and on the Training set if Regularization and Early Stopping were to be applied in a joint manner. The simulation was done on MATLAB using a Gradient Descent based algorithm to minimize the Cost Function. Gradient Descent was chosen as it is guaranteed to find the global minimum of a convex surface, however it can incur a high computational cost. I concluded from the simulation results that machine learning algorithms provide near to 100% accuracy in the training set, but face difficulties and significantly reduced accuracy on the Test Set.

Keywords: Logistic Regression, Regularization, Early Stopping.

1 Introduction

With the rapid growth of Computer Vision, Artificial Intelligence and Machine Learning, human effort in recognizing, learning and making predictions is vastly diminished. Object and Character recognition is a very lucrative field of research, as it has the potential for several industrial and commercial applications. Digit Recognition finds its applications in reading street numbers, bank cheque amounts and even license plates, where on the other hand Alphabet Recognition would be highly efficient when employed to postal offices for mail sorting, digitization of any handwritten form, etc.

The recognition of printed characters may be considered far more straightforward, on the other hand the complexity of handwritten digits (stroke style, thickness, geometrical shape) will indefinitely vary from person to person due to different handwriting styles and traits. It can be vaguely suggested that a training set with all types of handwriting styles would provide heightened accuracy, although it may not be as practical to collect data for each different handwriting style. Hence, the other solution is for a generalized learning algorithm which would accurately predict any handwritten character even if it wasn't trained to identify that particular handwriting style.

Neural Networks (whether artificial or convoluted) have widely been used for image recognition and for image processing. Also, it has been shown by S Chen Et al. [1] that a Neural Network had marginally greater accuracy than a K-NN (Nearest-Neighbor), Random Forest, Gradient Boost or a Decision Tree algorithm, although the Neural Network has to bear a significantly higher computational time to recognize handwritten digits.

For an algorithm to function in unfitting situation, it needs a probability model, which can be very easily implemented through Logistic Regression and by using Support Vector Machines mentioned by Y Chang et Al [2], and H Ahamed Et al. [3] had concluded that for digit recognition an SVM algorithm may have slightly better results on the Test set, although they have reduced accuracy on the training set and a consequential higher computational time as compared to a Logistic Regression Algorithm. Similar statements have also been made previously on other researches, [4,5] in favour of a probabilistic approach over a Discriminant analysis.

In a similar manner, the sigmoid function could be imbedded into the final layer of a Neural Network as previously done by S Makwanaa et al. [6] and also Y Perwej et Al. [7]. This research aims to asses Logistic Regression, its accuracy with and without a regularization factor, the effect of scaling the regularization factor (lambda) and also asses the effect of Early Stopping on the training and test set accuracy.

2 Description of Dataset

2.1 Data Collection

For training a machine learning algorithm, a vast data set is required, which should ideally incorporate all features and Handwriting styles (in this case) for unsurpassed results. The raw data was collected manually and with the assistance of peers (handwritten characters). It is often said, a model with a better dataset is superior to a model with a better Algorithm. The raw database was in the form of .jpg images which were created through a stylus-based application which allowed the users to write down each individual character (be it digit or alphabet) through the screen.

(a) (b) (c)

Fig. 1. (a) Handwritten '1' (b) Handwritten '4' (c) Handwritten 'G'

This method was fairly easy to create datasets and did not consume much time as well. The output image of this application was a .jpg image with approximate dimensions 720×1130 pixels. In this manner, several handwriting styles were collected from various peers and acquaintances, to better fit the Machine Learning Model.

2.2 Data Statistics

Table 1: Dataset for Digit Recognition

Digit Number	#Training Set	#Test Set	Sub-Total
0	80	20	100
1	80	20	100
2	80	20	100
3	73	20	93
4	80	20	100
5	80	20	100
6	80	20	100
7	82	20	102
8	82	20	102
9	83	20	103
Total	800	200	1000

Table 2: Dataset for Alphabet Recognition

'Alphabet'	#Training Set	#Test Set	Sub-Total	'Alphabet'	#Training Set	#Test Set	Sub-Total
A	30	6	36	N	33	6	39
B	32	6	38	O	36	6	42
C	30	6	36	P	34	6	40
D	30	6	36	Q	36	6	42
E	30	6	36	R	38	6	44
F	31	6	37	S	35	6	41
G	30	6	36	T	31	6	37
H	35	6	41	U	33	6	39
I	32	6	38	V	35	6	41
J	31	6	37	W	30	6	36
K	36	6	42	X	31	6	37
L	33	6	39	Y	30	6	36
M	31	6	37	Z	31	6	37
'Alphabet'	#Training Set	#Test Set	Total				
Total	844	156	1000				

2.3 Processing of the Input Dataset

The current form of the data is very bland, and can't be directly fed into a Logistic Regression algorithm. For simplicity, and a massive reduction in computational cost, the images (data) is converted from an RGB 721×1130 format to a grayscale 20×20 image. Hence, the number of pixels per image (datum) to be processed is reduced from 824900 to just 400.

The conversion is done by the MATLAB command:
gray = rgb2gray (imageArray); (to convert image to grayscale)
img =imresize (gray, [20,20]); (to resize the image to 20×20 pixels)

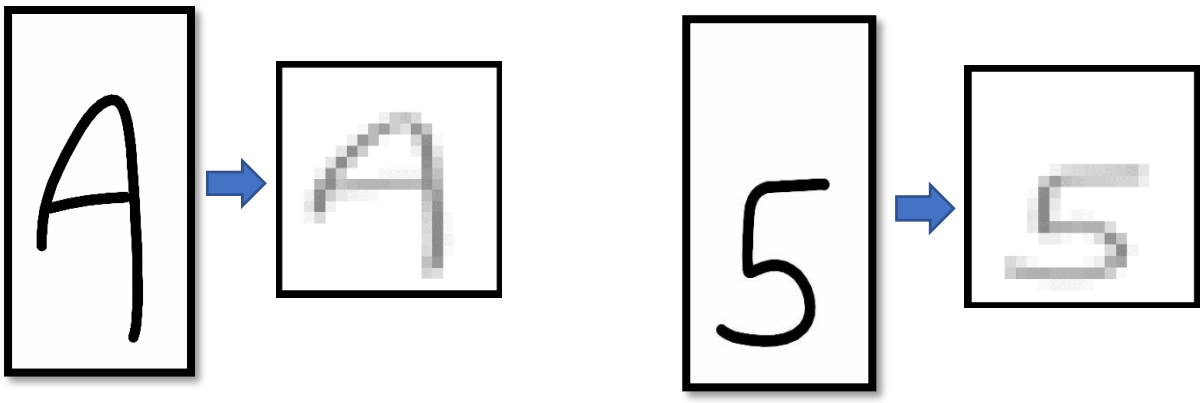

Figure 2: Sample Conversion of RGB image to Grayscale [20×20]

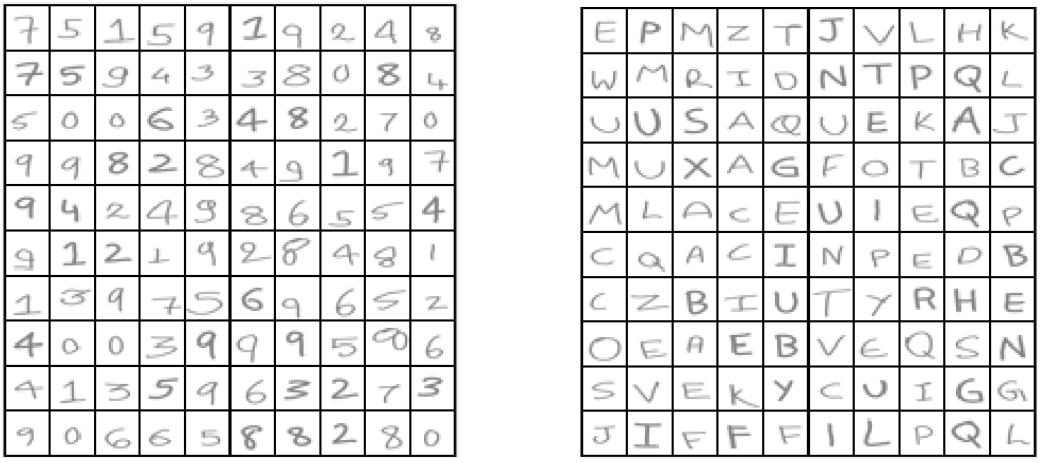

Figure 3: Random Sample of Digits and Alphabets

400 input features can now be fed into a Machine Learning algorithm where each feature will correspond to the weight of the respective pixel in grayscale intensity. Currently, the data is loaded in the form of a 20×20 matrix, which would then be rolled into a single row. Hence the final training set matrix would be having '*m*' rows and 400 columns, where '*m*' is the number of training data.

Hence, the n^{th} data of the entire set, would be a row vector,

$$(X^n) = [X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ \ldots\ldots \ X_{400}] \qquad (1)$$

3 Methodology: Machine Learning Methods

3.1 Logistic Regression

a) Logistic Regression is a probabilistic classification technique which is used to predict outcomes by analysing the relationship between one or more existing independent variables. Unlike linear regression, Logistic regression applies a sigmoid function to the equation, allowing the output to be a probabilistic value, which could then be mapped to several discrete classes. Logistic Regression was also applied previously by J Dong et Al. [8] for failure prediction.

$$Sigmoid\ Function,\quad f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

In this case, the independent variables are the pixel Intensities, and the dependent or unknown variables have to be optimized to train the classifier. The predicated hypothesis equation is:

$$Hypothesis = Sigmoid\ (\theta_0 + x_1\theta_1 + x_2\theta_2 + x_3\theta_3 + x_4\theta_5 + \cdots + x_m\theta_m) \tag{3}$$

Vectorising the Equation:

Let the training set be denoted by 'X', Unknown parameters known as 'θ',

$$X = \begin{bmatrix} 1 & \_\_(X^{(1)})\_\_ \\ 1 & \_\_(X^{(2)})\_\_ \\ . & . \\ . & . \\ . & . \\ 1 & \_\_(X^{(m)})\_\_ \end{bmatrix} \quad \text{and} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_{m+1} \end{bmatrix} \tag{4}$$

Where $(X^{(n)})$ denotes the n^{th} data and is a row vector with 400 elements (grayscale intensity). '1' is padded to each row of the first column to incorporate an unbiased unit θ_0 in the hypothesis.

The hypothesis function, can be written as:

$$Hypothesis = sigmoid(X^T * \theta) \tag{5}$$

Where the Hypothesis, would be a single Column matrix with 'm' rows, one hypothesis (prediction) for each training data. The cost function (error) of the hypothesis function is optimized to find the optimum values of the unknown parameter θ, by which the Cost function would have its minimum value (minimum error, maximum accuracy).

For logistic regression, a Cost function is defined as:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left\{y^i \log\left(h_\theta(x^i)\right) + (1-y^i)\log\left(1-h_\theta(x^i)\right)\right\} \tag{6}$$

In an ideal world, the summation for 'm' terms would be implemented through a looped function, although in MATLAB it can easily be implemented using a vectorised equation.

Vectorised Cost Function:

$$J(\theta) = \frac{1}{m}\{-y^T \log(h) - (1-y)^T \log(1-h)\} \tag{7}$$

Where 'h' is the Hypothesis function, $h = sigmoid(X^T * \theta)$

To find the minimum point of the cost function $J(\theta)$, several algorithms such as Gradient Descent, conjugate Gradient, BFGS, L-BFGS, etc are available. For the purpose of this experiment, a function developed by Carl Edward Rasmussen "*fmincg*" is utilized, which is similar to the in-built function of MATLAB "*fminunc()*".

"*fmincg*" is significantly more accurate when dealing with a higher number of input features (400 in this case) whereas "*fminunc"* is used for much lower number of features. Also, "*fmincg*" reduces computational memory as it is an advanced optimization algorithm.

Both the algorithms can be considered as an advancement towards Gradient Descent, where the model starts from a random initialization and takes iterative steps towards the global minimum of the Cost function. The size of each step is optimized as per the instantaneous gradient, although the number of iterative steps highly affects the rate of convergence of the model.

Similar to Gradient Descent, which is given by:
Repeat for num_out, (where num_out is the number of output terms, ex. 10 for digits)
{
  Repeat for 'n' iterations, or until convergence
  {
    $\theta_k = \theta_k - a \sum_{i=1}^{m} (h_0(x^i) - y^i) x_k^i$ (where a is the rate of convergence)
  }
}
For the case of "*fmincg"* the rate of convergence is automatically optimized by using the value of the instantaneous gradient, allowing the algorithm to converge to the minimum in much fewer iterations.

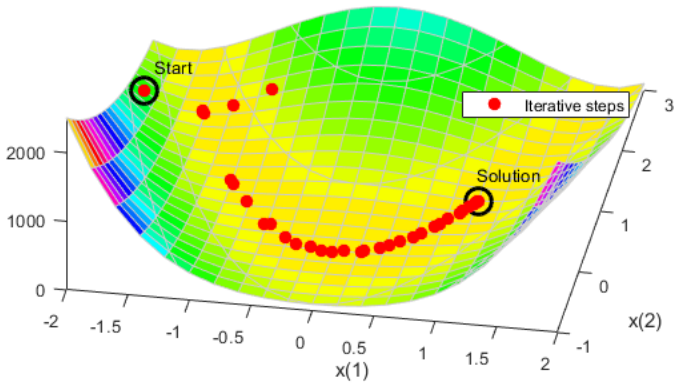

Figure 4: "*fminunc"* on Rosenbrock's "banana function, [9]

The above figure shows the optimization of Rosenbrock's "banana function" which only had 2 input features, *x(1)* and *x(2)*. Since this research deals with 400 input features, it is near impossible with today's technology to visualize the cost function as shown above.

The function "*fmincg()*" requires the cost function, as well as its Gradient in partial derivatives as its input to allow for quick and easier convergence. The Vectorized partial derivative of the cost function is given by:

$$Grad = \frac{\left(X^T * (h - y)\right)}{m} \qquad (8)$$

3.2 Regularization

Overfitting is a common term in machine learning applications, and is highly undesirable. It is defined as a model which fits and recognizes the training data completely, although performs very poorly on the test set of data. As even stated by other authors [10, 11], any learning model is subject even to the slightest of overfit, which should be rectified.

In terms of character recognition, it would mean that if the model is overfitting then it is able to recognize the handwriting styles which are fed into the training set, and would not be able to recognize unknown handwriting styles and provide a mediocre accuracy on the Test Set.

To prevent overfitting from occurring, regularization is used. Regularization is a technique which is used to reduce the coefficients of higher order polynomials in the hypothesis function to reduce their impact, hence improving results on a general case basis. Regularization implemented with Logistic Regression can be written as follows.

Cost Function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left\{y^i \log\left(h_\theta(x^i)\right) + (1-y^i)\log\left(1-h_\theta(x^i)\right)\right\} + \frac{\lambda}{2m}\sum_{i=2}^{m}({\theta_i}^2) \quad (10)$$

Taking Partial Derivative, the Vectorized Gradient is:

$$Grad = \frac{\left({X^T}*(h-y)\right)}{m} + \frac{\lambda}{m}\left(\theta_{2:m}\right) \quad (11)$$

Where, λ is the regularization factor and is usually set to a high value, to reduce the impact of higher order θ terms. Although, selecting the most apt value of λ can be a tough and very computational task.

3.3 Early Stopping

For an iterative algorithm, the performance of the model is measured after each iteration, and is represented in a graphical manner. Up till a certain number of iterations, the accuracy on the training and cross validation set will increase, however after a certain point, the accuracy of the training set will increase and the test set accuracy drastically decreases.

Early stopping refers to running the iterative algorithm only to the point where it provides maximum training set accuracy, as further iterative steps would only result in overfitting the dataset. The Early stop point will be adjusted with respect to the Training time also, to reduce computational cost.

4 Results and Discussion

In this section, the results of recognition model for handwritten digits and alphabets is presented. The results are based on the parameter optimization of lambda, and for the number of iterations of the algorithm to provide maximum training and test accuracy.

4.1 Parameter Optimization. For better results, the model should perform well on the training set and on the test set. Although we have two unknown parameters to optimize,

1) Number of Iterations to run *fmincg()*
2) Value of Regularization factor λ

After some unsuccessful hit and trial approaches, it was decided to administer a nested loop function, which would approximately work such as:

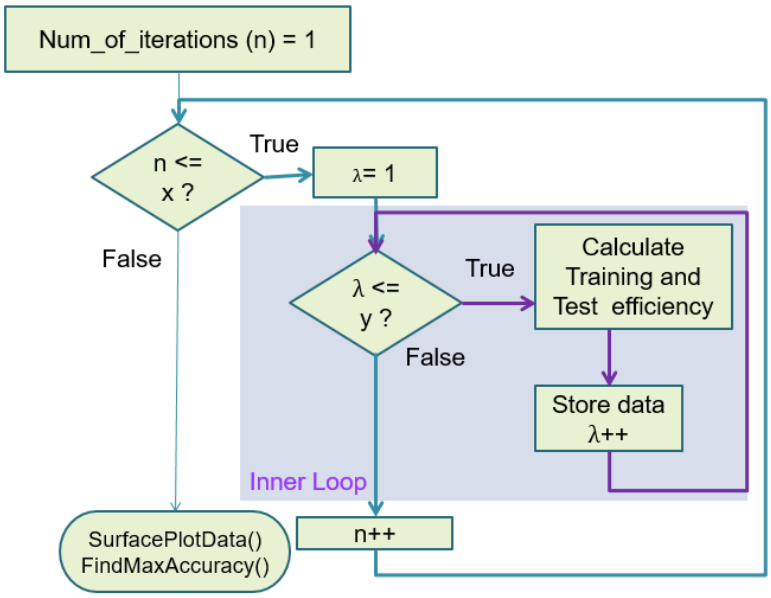

Figure 5: Flow Structure for Parameter Optimization

With the surface plot, the trend of the value of accuracy, with respect to the unknown parameters can be found out, and the best suitable optimized value of λ and the number of iterations can be found out for maximum accuracy on both, training and test set.

4.2 Parameter Optimization for Digits:

The parameter optimization code was run for the dataset of handwritten digits, to determine the best suitable value of the unknown parameters. The following are the obtained 3D surface plots,

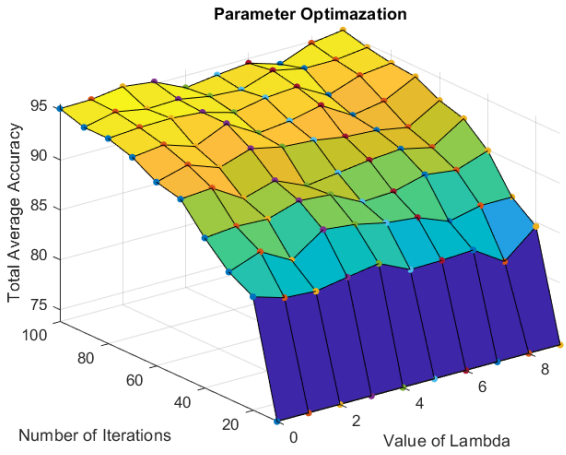

Figure 6: Optimization Plot for Higher Average Accuracy

As seen from the figure 6, it can be said that the average (train and test) accuracy of the model for handwritten digits increases in a logarithmic fashion with an increase in the number of iterations.

Figure 6 contained the accuracy values only up to 100 iterations, hence another plot was needed to identify the equilibrium value for the number of iterations, to maximize accuracy and to reduce computational cost and memory.

Although, it was observed that the training data had a 100% accuracy in some cases, and very low-Test set accuracy (70-80%), which was a clear sign of overfitting of the data. Hence, another plot was done, this time only for the Test Set on the "*Z"* axis shown in figure 7.

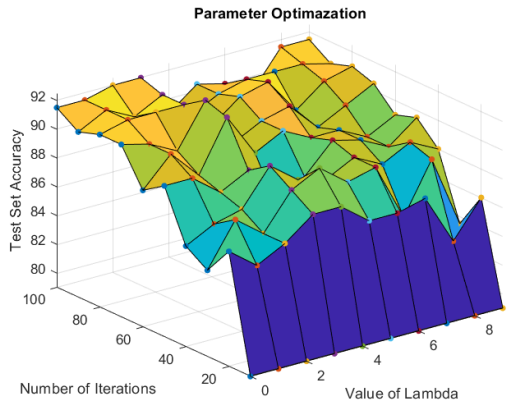

Figure 7: Optimization Plot for Higher Test Accuracy

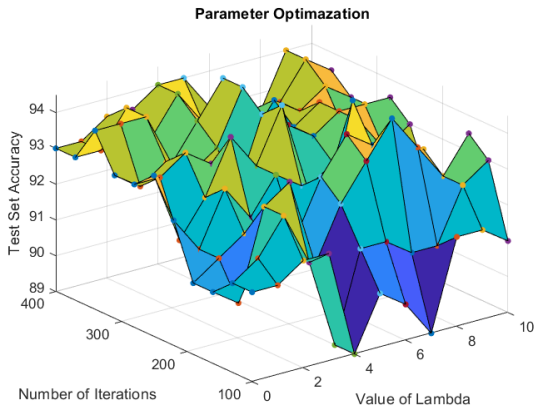

Figure 8: Optimization Plot for Higher Iterations

On observing figure 7, it was found out that the test set had a maximum accuracy of 92.5%, when the model was trained for 70 iterations and had a regularization factor equal to 3.

Also, further increasing the regularization factor above 10, only increased the number of iterations required to achieve the similar, if not lower accuracy to a model with a smaller regularization factor along with lower iterations.

From figure 8, the points with maximum accuracy and minimum number of iterations were selected and their computational time during the training phase is compared:

Table 3: Parameter Optimization for Digits

S.No.	Value of Lambda	Number of Iterations	Training Accuracy (%)	Test Set Accuracy (%)	Computational Time (approx. in seconds.)
1	3	250	97.500	94.0	6
2	5	250	98.125	94.5	6
3	5	310	98.125	94.5	8
4	5	340	98.125	94.5	9
5	5	400	98.750	94.0	10
6	7	160	100.000	94.5	4
7	7	220	100.000	94.5	5

Hence, it was concluded that for digit recognition, the model should be suited to a higher value of lambda which would provide maximum accuracy, reduce the chances of overfitting and have a reduced computational time during the training set. The model was trained for 160 iterations, with $\lambda = 7$.

Although, the difference between other set of parameters is a matter of mere seconds, when implemented on a large scale, handwritten character recognition would require vast amounts of data, approximately 5000 or more. With such vast amounts of data, the computational time would exponentially grow, which would be a major factor in deciding the number of iterations of any Machine Learning Algorithm.

Sample Results after training the model,

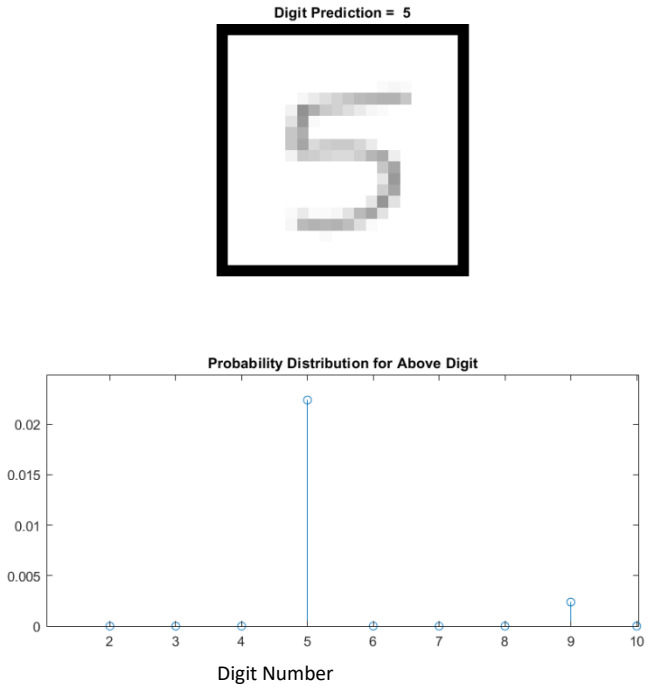

Figure 9: Recognition of Digit '5' by Optimized Model

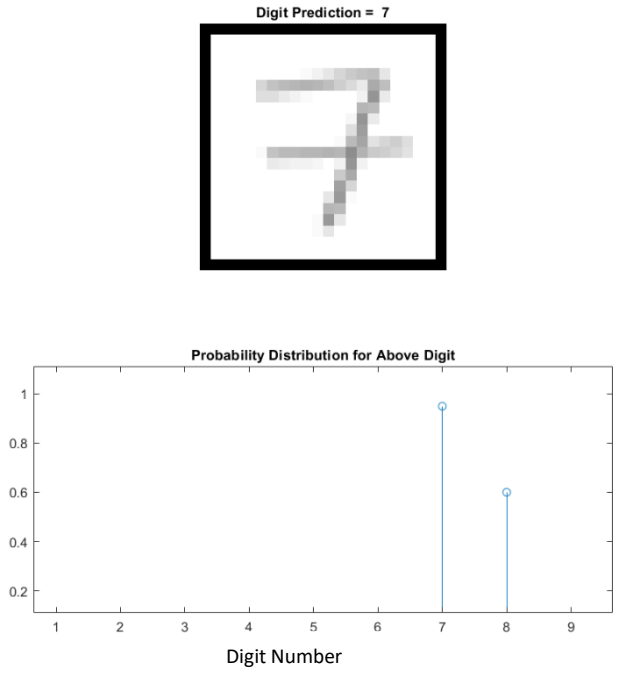

Figure 10: Recognition of Digit '7' by Optimized Model

When provided with unlabelled data, the model was successfully able to identify majority of the digits, and almost always had a confidence level greater than 90% as observed through the probability distribution.

When new handwriting styles were introduced to the dataset, the model had as low as 2% confidence on its top prediction (figure 9, 10), although still managed to get the prediction right as the probability values for the other digits were far lower. Hence it could be concluded that the model was faring well even when datasets with new and distinct features were provided to it.

4.3 Parameter Optimization for Alphabets:

Next, the parameter optimization code was run for the model to recognize handwritten alphabets. Since, for alphabet recognition, the number of output labels are 26, hence the number of iterations could not be increased above a certain extent, otherwise a lot of computer memory would be required.

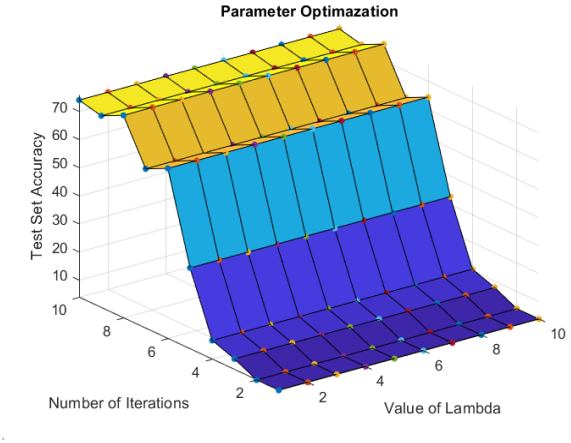

Figure 11: Optimization Plot for Lower Iterations

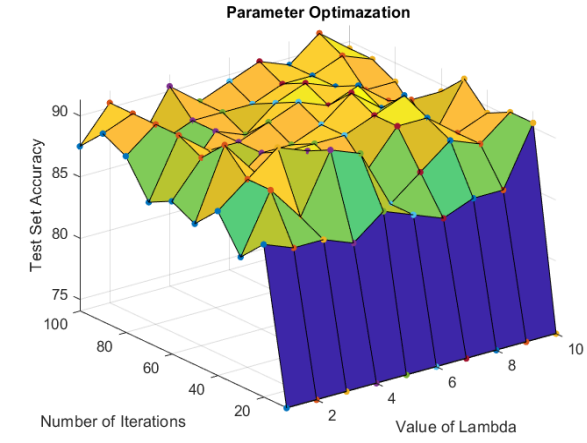

Figure 12: Optimization Plot for Higher Iterations

As observed from figure 11, the model is unable to be accurate during the Test set when trained up to 10 iterations (around 75% training accuracy). The number of iterations were increased and the second plot, figure 12 was obtained. The maximum accuracy obtained from figure 12 is 91.02%. Since, the number of iterations couldn't be increased significantly, the effect of the value of lambda was observed on the test set accuracy. Hence, another plot was developed, with a higher range of values of the regularization factor.

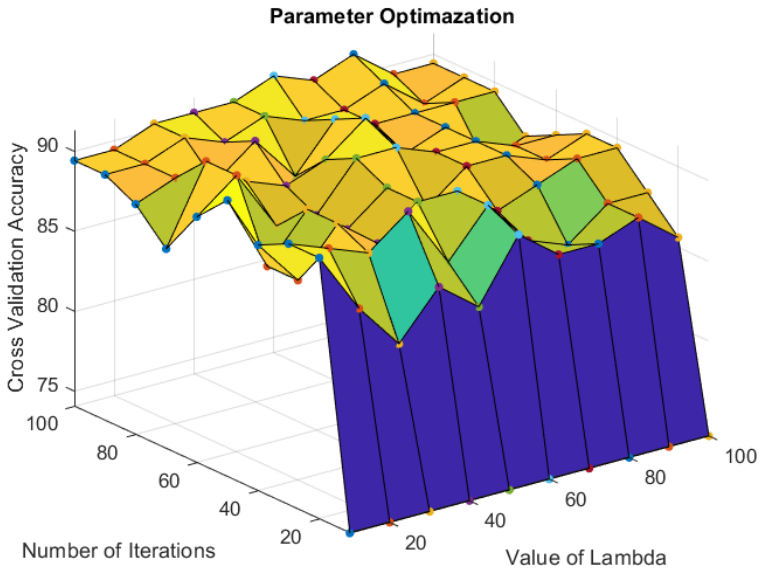

Figure 13: Optimization Plot for higher values of Lambda

The maximum accuracy obtained from figure 13 was also the same, 91.02%. Hence, different parameter sets were compared, which were taken from figure 12 and 13.

Table 4: Parameter Optimization for Alphabets

S.No.	Value of Lambda	Number of Iterations	Training Accuracy (%)	Test Set Accuracy (%)	Computational Time (approx. in seconds.)
1	2	50	99.645	91.02	9
2	3	40	99.645	91.02	8
3	4	30	96.800	91.02	7
4	20	70	100.000	91.02	10
5	60	100	100.000	91.02	11
6	80	100	100.000	91.02	11

From table 4, it was decided that '20' is an optimal value for lambda with 70 iterations to the model. Increasing the value of lambda further would only require more iterations to converge to the same accuracy, also reducing lambda from this optimal value would only reduce the training set accuracy, which would only reduce the effectiveness of the system.

Also, from figure 13, when keeping $\lambda = 20$, if the number of iterations were to be increased above 70, the Test set Accuracy reduced, resulting in overfitting. So, 70 iterations are measured as the early stop value, and further rise in iterations is unwanted, as also supported by past research in [8].
Hence, the model was trained for 70 iterations, with $\lambda = 20$. Below are the sample results and the probability distribution for unlabelled datasets.

In some cases, as shown below in figure 14 ('R' predicted as 'K'), the probability difference between the right and wrong prediction is just marginal, and can be improved by adding more input features to the model or by increasing the amount of training data or by using structural information to recognize characters, [13]. The 3D plots above were created by a limited amount of points, and hence appear jagged in nature. To smoothen the curve and shape of the plots, more data points could be added, to give a better insight to find the optimal set of parameters, although more data points would been a highly increased run-time for the Parameter Optimization Program.
An alternative approach to optimization would make use of a dynamically varying regularization factor, as also practically observed by M Bejani et al. [14] and Y Wang et al. [15], where adaptive regularization schemes were applied to CNN to boost results. Similarly, the regularization factor can be modelled by the Cost function or by the training loss, and could be optimized for better results.

With early stopping applied to a Gradient Descent algorithm, the model is more robust towards label noise, as also supported by [16] and [17]. Similarly, an optimal control approach can be used to implement early stopping with other Algorithms, as done on Gradient Flow by A Effland et al. [18].

Parameter Optimization can easily help to identify the early stop value whilst taking other factors such as Regularization, Dropouts, Rate of Convergence, etc into consideration (by utilizing Multi-dimensional Arrays), as also recently concluded by Y Fedorenko et al. [19] by using a Metagraph approach to eliminate overfitting. Parameter optimization can be utilized in any Machine Learning Model (SVM, CNN, etc) and a greater performance can be achieved by optimized parameters.

Sample Results after training the model,

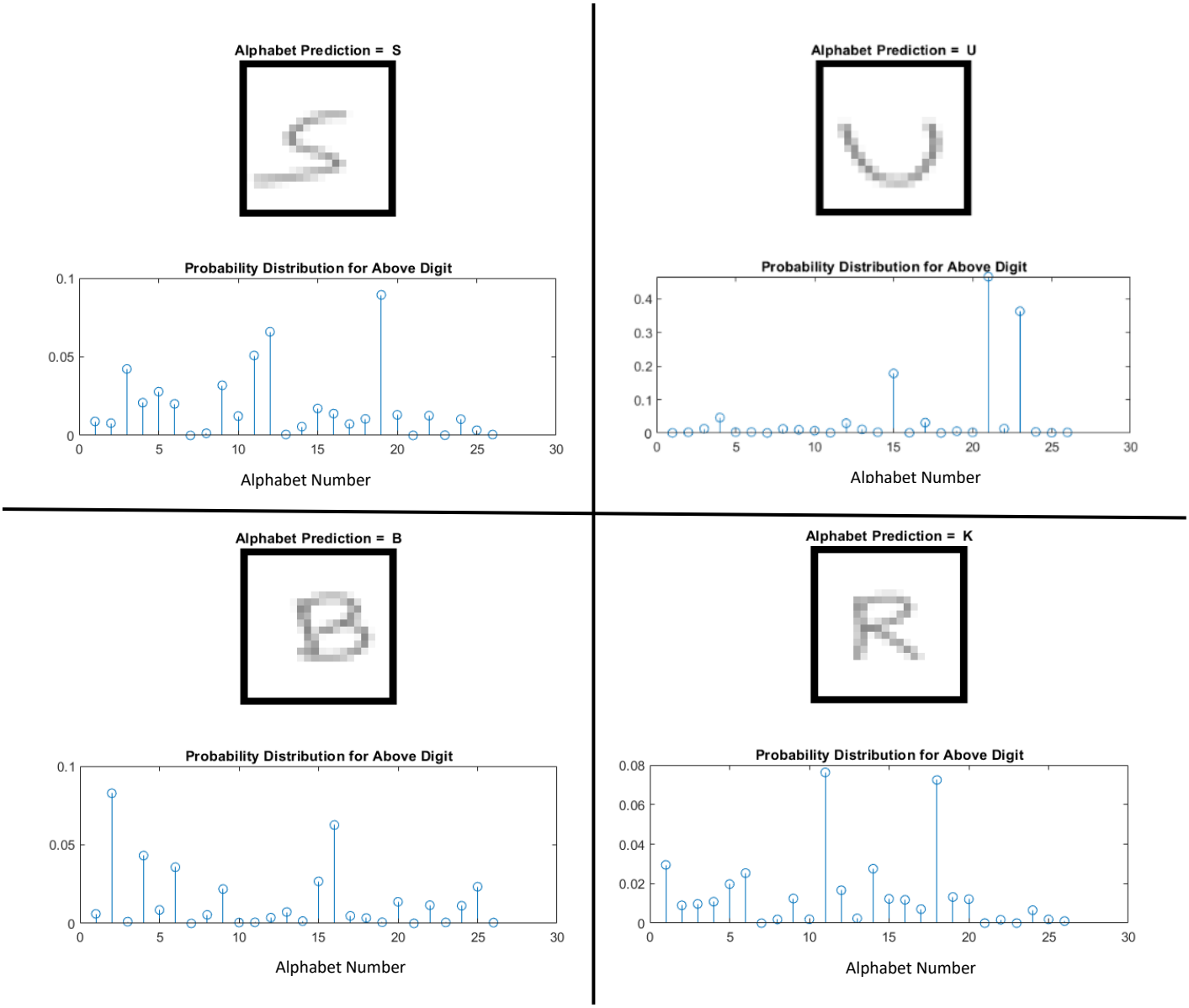

Figure 14: Sample Alphabet Predictions by Optimized Model

Conclusion

In this paper, Parameter optimization of regularized Logistic Regression was done for recognition of handwritten digits and alphabets. The performance of the models with varying the regularization factor λ and the number of iterations of the model have been observed on the basis of their training accuracy, test accuracy and the training time. From the plots for digit recognition, it is also noted that for a regularization factor equal to '0', the model had a lower accuracy at any number of iterations, when compared with a regularized model (λ >0) with the same number of iterations. Hence without regularization, a logistic regression-based model is bound to over-fit for digit recognition with a relatively smaller dataset.

Another conclusion made was that if the model was trained for higher iterations than the early stop value, computational cost and time would increase whilst the accuracy of the model on the training set may or may not increase and the accuracy on the test set would decrease or stay the same. Also, in this paper, it was determined that a probability-based model would be ideal, as even if an output has a low probability (low confidence), accurate predictions can still be made due to the relatively higher probability than the other output labels. Alternatively, if

a discrete threshold-based model was to be selected, a lot of effort in optimizing the threshold value would be required, only adding additional unknown variables into the equation.

It can also be established that handwritten character recognition is very prone to overfitting, and they most appropriate methods to train a model to avoid overfitting are to implement regularization and early stopping (limit number of iterations to optimum). This paper has optimized both mentioned techniques that solve the problem of overfitting, (outside of alterations in the dataset) through a parameter optimization method for best results.

References

[1] Chen, Shengfeng, et al. "Offline Handwritten Digits Recognition Using Machine Learning." Proceedings of the International Conference on Industrial Engineering and Operations Management Washington DC, USA, September 27-29, 2018.

[2] Chang, Yuan-chin Ivar, and Sung-Chiang Lin. "Synergy of Logistic Regression and Support Vector Machine in Multiple-Class Classification." SpringerLink, Springer, Berlin, Heidelberg, 25 Aug. 2004.

[3] Ahamed, Hafiz & Alam, Ishraq & Islam, Md. (2019). Handwritten Digit Recognition System Based on LRM and SVM Algorithm.

[4] Press, S. James, and Sandra Wilson. "Choosing between Logistic Regression and Discriminant Analysis." Taylor & Francis, 5 Apr. 2012.

[5] Linker, R.; Seginer, I. Greenhouse Temperature Modeling: A Comparison between Sigmoid Neural Networks and Hybrid Models. Math. Comput. Simul. 2004, 65, 19–29.

[6] Makwanaa, Sagar, and Kiran Bhowmick. "Hand-Written Digit Recognition Using Logistic Regression." International Journal of Global Technology Initiatives, ISSN 2320-1207, Volume 4, Issue 1- March 2015.

[7] Perwej, Dr. Yusuf & Chaturvedi, Ashish. (2011). Neural Networks for Handwritten English Alphabet Recognition. International Journal of Computer Applications (0975 – 8887). Volume 20– No.7, 1-5. 10.5120/2449-2824.

[8] Dong, Jia-Jyun, et al. "Logistic Regression Model for Predicting the Failure Probability of a Landslide Dam." Engineering Geology Volume 117, Issues 1–2, 10 January 2011, Pages 52-61, https://doi.org/10.1016/j.enggeo.2010.10.004.

[9] "Banana Function Minimization." MATLAB & Simulink Example, www.mathworks.com/help/optim/examples/banana-function-minimization.html.

[10] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In Proc. ICLR, 2014.

[11] Cireşan, Dan Claudiu, et al. "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition." MIT Press Journals, 9 Nov. 2010.

[12] Mahsereci, Maren & Balles, Lukas & Lassner, Christoph & Hennig, Philipp. (2017). Early Stopping without a Validation Set.

[13] Behnke, S, et al. "Recognition of Handwritten Digits Using Structural Information." IEEE Conference Publication, June 1997.

[14] Bejani, Mohammad Mahdi, and Mehdi Ghatee. "Convolutional Neural Network With Adaptive Regularization to Classify Driving Styles on Smartphones."- IEEE Journals & Magazine, IEEE, 18 Feb. 2019.

[15] Wang, Yi, et al. Convolutional Neural Networks With Dynamic Regularization , IEEE, 8 June 2020.

[16] Li, Mingchen, et al. "Gradient Descent with Early Stopping Is Provably Robust to Label Noise for Overparameterized Neural Networks." PMLR, 3 June 2020,

[17] Hu, Wei, et al. "Simple and Effective Regularization Methods for Training on Noisily Labeled Data with Generalization Guarantee." ArXiv.org, 2 Oct. 2019.

[18] Effland, Alexander, et al. "An Optimal Control Approach to Early Stopping Variational Methods for Image Restoration." ArXiv.org, 19 July 2019.

[19] Fedorenko, Yuriy S., et al. "The Analysis of Regularization in Deep Neural Networks Using Metagraph Approach." SpringerLink, Springer, Cham, 2 Oct. 2017.