ELL409 Assignment 1

Shubham Mittal
Electrical Engineering
IIT Delhi
ee1180957@iitd.ac.in

Chirag Bhatt
Mathematics Dept.
IIT Delhi
mt1180750@maths.iitd.ac.in

***Abstract*—In this paper, the primitive machine learning algorithms are applied on a regression problem, binary classification and multi-class classification problem. The observations presented verify the theoretical results in the literature.**

I. Introduction

First the results obtained on the regression problem are presented where detailed discussion and experiments are shown on generalized linear models. We answer questions like when to regularize the model, which regularization is better, what effects it has, and also compare gradient descent and stochastic gradient descent. Then, we discuss the classification problems where we use a discriminant function learnt using regression task (here we use our results and interpretations from the regression problem). Methods from estimation theory like MLE, MAP are also used to estimate the appropriate densities to make a classifier. Each section ends with discussion on non parametric methods like KNN and Parzen Windows. All the experiments are 5-fold cross validated as per the split told in the instructions.

II. Regression Problem - Predicting Temperature

We first use the closed form solution of OLS to polynomial regression problem and get the best degree or dimension of the data such that the metric (MSE) is minimum. Then we begin comparing the solution with that obtained using Gradient Descent. We investigate hyper-parameter tuning, stochastic gradient descent and interpret the results. Further going into higher dimensions and overfitting the data, we analyze various regularization techniques and closely study their effects. We also present our results using MAE loss function. Finally, we conclude this question with the non-parametric methods.

A. Ordinary Least Squares

We first implement polynomial regression using closed form solution and obtain a bias-variance curve (Figure 1). It is evident that the polynomial of degree 2 or 3 are better than lower or higher degrees. Also, since there are 6 features in x (and x is normalized), the number of features increase exponentially as the degree increases which also leads to numerical errors like underflow or overflow. These errors were observed when we train the model on higher degrees (above 10) where the train error starts to rise up (contrary to what we expect). Note that if we used unnormalized data, then going into even degree 2 would lead to sparsity in the features (i.e. many features would be zero) and overflow as well. This stems from the fact that there are binary features and features in the order of 1000 in x. In the closed form solution, Moore Penrose inverse of matrices is used because the degree of polynomial increases, the higher order terms may reduce to zero, leading to singular data covariance matrix.

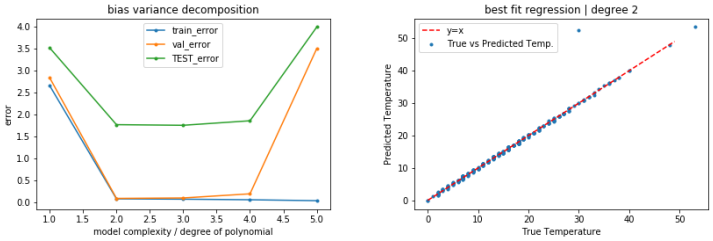

Fig. 1. Polynomial Regression

Next, we compare the closed form solution with that obtained using gradient descent algorithms. For this, we use degree 1 polynomial to have relatively less number of features so that they can be analyzed properly. But the results will definitely follow for x with any number of features.

1) Gradient Descent (GD): Following observations and interpretations were made while implementing gradient descent algorithm :

1) The problem of vanishing and exploding gradients, i.e. gradients become zero or infinite : this problem was observed when the dataset is not normalized and/or the learning rate is too high.
2) Convergence issue : tuning the learning rate is important as if it is too low then the algorithm takes too many epochs to converge. Figure 2 (left) shows the effect of learning rate on the convergence.
3) Convergence to closed form solution : It was observed that the algorithm converges to the closed form solution, provided the hyper-parameters (learning rate and number of epochs) are tuned properly. Also, because the gradient descent algorithm is asymptotic in the sense that it converges to the optimum when the number of iterations goes to infinite, a small difference was observed between the grad. desc. and closed form solution (shown in Table I).
4) Time varying learning rate : It was also observed that reducing the learning rate after some number of epochs doesn't reduce the loss, as it should if we were stuck

in a local minimum. This is because the mse loss is a convex function with the hypothesis $w^T x$, thus there is only one optima.

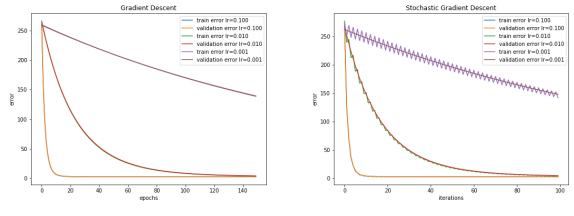

Fig. 2. Optimization algorithms applied on Least Squares

2) Stochastic Gradient Descent (SGD): Following observations and interpretations were made while implementing stochastic gradient descent algorithm :

1) The problem of vanishing and exploding gradient is more enhanced in SGD since we have a noisy gradient computed from a mini-batch of the train data. Also this noisy gradient leads to fluctuations in the train error plot as shown Figure 2(right).
2) Figure 2(right) is obtained using batch size of 200 (train data size is 700) and 50 epochs whereas Figure 2(left) has 150 epochs. Note the difference of the x-axis in both the plots - in SGD the plot is against number of passes through mini-batch of the data, also called as iterations. Hence, we infer that SGD requires lesser number of epochs or passes though complete data and converges faster. Table I shows that weights learnt using SGD differ more than that using GD from the closed form solution.
3) As the batch size is reduced, it was observed that the training time increases and we get more noisier train error plot. Thus concluding that the batch size has to be tuned properly such that it is nether too low (giving high noise) nor too high (going towards GD).

w	Closed Form	Gradient Descent	Stochastic G.D.
Bias Term	[1.47457e+01]	[1.47457e+01]	[1.47638e+01]
w_1	[-4.84591e-03]	[-4.84611e-03]	[8.38333e-03]
w_2	[7.69836e+00]	[7.69835e+00]	[7.64714e+00]
w_3	[-4.78595e-02]	[-4.78597e-02]	[-3.97308e-02]
w_4	[-8.30544e-02]	[-8.30552e-02]	[-1.10427e-01]
w_5	[8.45435e-01]	[8.45444e-01]	[8.43169e-01]
w_6	[6.71323e-02]	[6.71313e-02]	[8.12530e-02]
TrainError	2.665	2.665	2.817
ValError	2.826	2.826	2.835
TestError	3.508	3.508	3.510

TABLE I
ORDINARY LEAST SQUARES

B. Regularized Least Squares

In this section, the effects of L1, L2 and elastic net regularization on linear regression model (with MSE) are shown. For this the original 6 features are used (not going in higher dim. space because of too many features).

1) L1, L2 and Elastic Net Regularization: For L2 regularization, since the regularized cost function is convex the closed form and gradient descent solution are same. Here, the experiments are done using gradient descent since there exists no closed form solution to L1, and hence, elastic net regularization to the best of our knowledge. SGD is not used here as for the given small dataset, no significant advantage was seen over GD as shown in Table I. The results are shown in Figure 3. Following interpretations are made:

1) The train and test error increases as regularization increases since the cost function is being penalized more. Since both L1 and L2 norms are centered around 0, weights reduce to zero as α (regularization constant) increases.
2) The L1 regularization plot shows that it behaving like a feature selector - it pins some weights to zero which means that those features will be ignored in $w^T x$. This is not the case in L2, or rather, it is less pronounced in L2 since most of the weights are near to zero but not exactly zero. Thus L1 helps in selecting the required features.
3) L2 regularization reduces the weights towards zero but it keeps most of the features. As we will see in regularizing the overfitting polynomial regressor, the model has huge values of weights in higher order and applying L2 actually reduces those to zero. Thus L2 helps in getting towards best fit.
4) Elastic Net regularization, hence, combining both L1 and L2 helps in getting sparser solution as well as best fit. The results are shown in Appendix(Figure 15)
5) In L1, some noise during the gradient descent algorithm was also observed. We interpret that to be coming because of the non-differentiable convex function.

It is also mathematically proven in theory that L1 leads to zero weights exactly if a condition (given in Ian goodfellow book) is satisfied. The theory is verified and for that PCA whitening transform is used to make the features statistically independent. The empirical results do verify the theoretical results.

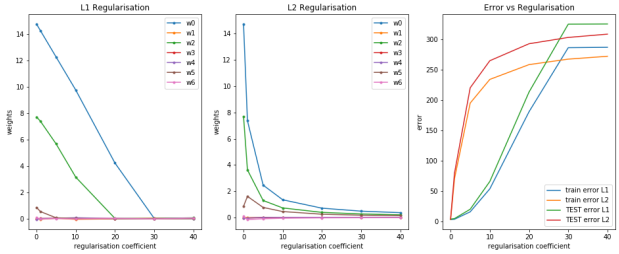

Fig. 3. Comparison between L1 and L2 Regularization

2) Solving the problem of Overfitting: In Figure 1, the 5th degree polynomial overfits the data. Hence, the model is L2 regularized and brought to best fit as shown in Figure 4. It was also observed that the coefficients of higher order terms are

reducing to zero (which were initially large) as regularization increased.

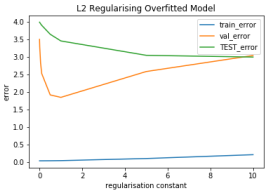

Fig. 4. Regularizing the 5th degree polynomial regression model

C. MAE Loss function

Training the linear regression model using MAE loss function didn't show notable difference from MSE results. Quantifying the goodness of the regression model is difficult and the comparison results are shown in Appendix (Figure 16) and left for the readers to decide which is good. However, following interpretation is made regarding the effect of using different loss function - MSE loss models noise as gaussian distributed whereas MAE loss models the same as laplace distributed.

D. Non Parametric Methods

Approximating a function i.e. making a regressor using knn or parzen windows can be done if we take the average of the output of the training samples near to the test point (k-nn or inside window). Figure 5 (left) show the bias variance decomposition as the model complexity (measured in terms of number of neighbor (knn) and window size (parzen)). Using it, the best-fit model is shown in Figure 5 (right). Comparing it with 1(right), it is visible that the regression model has learnt better from the training data. Also, the non parametric method requires the whole training data to predict output at any point (unlike regression methods), thus these methods are slower and can be considered as base-line algorithms/methods for any ML algorithm.

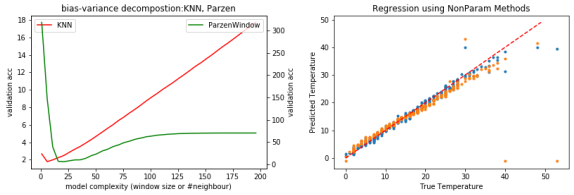

Fig. 5. Non Parametric Methods

One drawback of using parzen windows over KNN is that if no training point falls in the window, then there is no estimate at that test point. In the experiment, in such cases the parzen window - based model outputs -1 (i.e. negative temperature) to clearly tell the users that model has failed.

III. Binary Classification on Health Data

The dataset is visualized in 3D and shown in Appendix (Figure 14). The results like precision, recall, F1 shown in the following subsections are for class-1 and over the complete test dataset.

A. Maximum Likelihood Estimation

In this subsection, the estimation of (assumed) class conditional densities is made and the results are reported. The visualization of the dataset helped in choosing the number of Gaussian components in GMM to be equal to 2. From Table II, we infer that the data is coming from one multivariate Gaussian (for each class conditional density). Also, increasing the number of components (i.e. model complexity) in GMM, the Figure 6 (right) verifies this inference.

	Bayes-Gaussian	**Naive Bayes**	**Bayes-GMM**
Validation Acc	84.69	85.51	84.69
Train Acc	85.10	85.30	85.30
Test Acc	87.14	86.66	86.19
Precision	89.61	90.54	95.
Recall	78.40	76.13	68.67
F1	83.63	82.71	79.72

TABLE II
CLASSIFIERS USING ESTIMATION THEORY

However, interestingly, the ROC curves for the three classifiers (Figure6 (left)) shows that Naive Bayes is the best among three. Thus, we conclude that the data is most likely coming from gaussian class conditional densities such that the features are statistically independent.

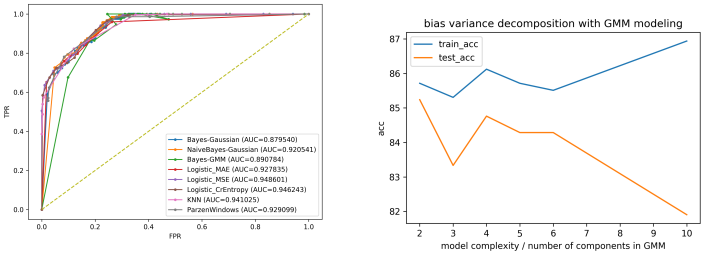

Fig. 6. ROC Curves and Bias-Variance Decomposition

Implementing the MAP estimates showed that the prior or belief, which is imposed on the parameters, is reducing the performance of the classifier. Since the dataset is having sufficient and good variety of sample, we interpret that the imposing prior is not helping or benefiting in getting the estimates.

B. Logistic Regression

In this section, a logistic regressor $\sigma(w^T x)$ is used with different loss functions. After choosing the best lost function (one that gives best test accuracy), we train a logit of polynomial regression $\sigma(w^T \phi(x))$ i.e. going into higher dimensional space where, hopefully, the data becomes more separable and/or the model overfits. As we shall see that overfitting occurs, so we regularize the model to bring it back to best fit.

Table III summarizes the results obtained after training $\sigma(w^T x)$ with different loss functions. Here gradient descent algorithm is used. Based on the performance on the test set (last 4 rows), we conclude that MSE loss function trains the model to the best among the three.

	MAE	MSE	CrossEntropy
Val Acc	0.8408	0.8469	0.8388
Val F1	0.8066	0.8133	0.8007
Train Acc	0.8531	0.8429	0.8449
Test Acc	0.8667	0.8905	0.8857
Precision	0.8736	0.8646	0.8557
Recall	0.8172	0.8925	0.8925
F1	0.8444	0.8783	0.8737

TABLE III
LOGISTIC REGRESSION CLASSIFIERS WITH DIFFERENT LOSS FUNCTIONS

The corresponding ROC curves for the three classifiers (difference being the loss function used for training) are shown in Figure6 (left). Hence, the ROC curves verifies that MSE is indeed the best in the given situation and they also show that regression models outperform the MLE/MAP based classifiers.

Also, for completeness, stochastic gradient descent is used and we obtain similar results, shown in Appendix (Table XII). SGD solution shows some level of deviation from the gradient descent solution, but given a small dataset, there was no notable difference in the training time. However, there was a notable difference in the training time of SGD in question 3.

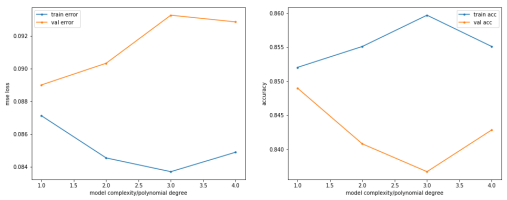

Fig. 7. BiasVariance Decomposition

Next, we try to increase the performance of the model by going into higher dimensions. For that $\sigma(w^T\phi(x))$ is trained where ϕ is polynomial kernel. The results are shown in Figure 7. Note that the training error/acc degrades after degree 3 because of the numerical errors (finite storage capability of digital machines). Thus, it is evident that the model is overfitting and hence, the overfitted 3^{rd} degree polynomial regression model is now regularized. Here the L2 regularization results are shown in Figure 8 but similar results follow in L1 or elastic net.

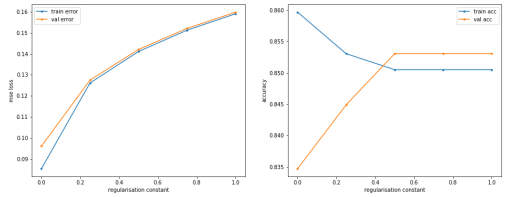

Fig. 8. Regularization of Overfitted Logistic 3^{rd} degree polynomial regressor

It is natural to question that why do we need to first overfit i.e. create a problem and then regularize i.e. solve the same? Answer is that since the dataset given is simple (not many features and data size is small), we start with less complex models and then subsequently increase the complexity to improve the performance. Increasing the model richness may lead to improved performance (Figure 1) up to certain extent but, here, it directly lead to overfitting problem. Regularization is one solution to this problem. However, if we are given complex datasets which has too many features and size is huge (like in Question 3), we start with a rich complex model (that most likely will overfit) and regularize it to reach the best fit. This is discussed in more detail in the section of Question 3.

C. Non Parametric Methods

Figure 7 (left) show the bias variance decomposition by changing the model complexity (measured in terms of number of neighbor (knn) and window size (parzen)). In finding the k-nearest neighbors, for distance metric Infinite (represented as 0) norm, 1-norm and 2-norm are compared as shown in Figure 7(right). However, we state that selecting the correct norm is based on the order of data points in the data set and the validation accuracies from the three norms differ in order of 0.001. Thus, the three norms give nearly same performance.

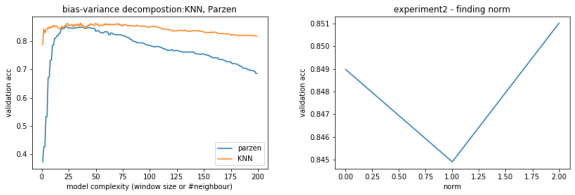

Fig. 9. Bias Variance Decomposition

The performance of the knn and parzen window classifier is given in Table for threshold of 1 as per the ROC curves given in Figure 6. Note that there is no training performance result since these are non parametric classifiers. Here also, the parzen window classifier performs poorly when the window size is too small.

	Val Acc	Test Acc	Precision	Recall	F1
KNN	0.8510	0.8673	0.8585	0.8301	0.8441
Parzen	0.8530	0.8755	0.9171	0.7830	0.8447

TABLE IV
KNN AND PARZEN WINDOW CLASSIFIERS

The AUC scores (shown in Figure 6) conclude that the logistic regression (trained with MSE loss) is the best among the classifiers we have seen in the course.

IV. MULTI-CLASS CLASSIFICATION

The given images of 64×64 present the problem of curse of dimensionality. Since these are gray scale images (having lot of 0s), even normalizing them doesn't produce non-singular data covariance matrix (for MLE, MAP). Hence, we resort to dimensionality reduction techniques like PCA, t-SNE. It was observed that t-SNE is pretty slow (and thus gave better separable data in reduced dimensions), so we used PCA and took the components with most of the data variance. Figure

10 (left) shows that almost 58% data variance is in first two components and remaining 42% in 4094 components (original image dimension being 4096). Thus, taking 2 components, the visualized (and well separated) training data is shown in Figure 10 (right).

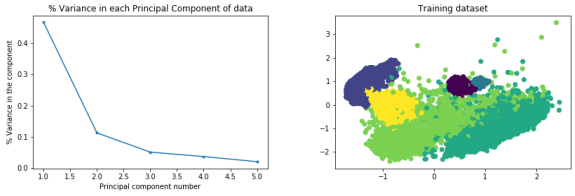

Fig. 10. Medical MNIST Dataset

A. Maximum Likelihood Estimation

The results of implementing a bayes classifier with each class conditional density estimated using MLE are shown in Table V. The per-class performance metrics and confusion matrix for each classifier is given in Appendix. The results are analyzed and compared in the next subsection.

	Bayes Gaussian	**Naive Bayes Gaussian**	**BayesGMM**
Val. Acc	0.9709	0.9642	0.970
Val. MacroF1	0.9706	0.9640	0.9726
Train Acc	0.9711	0.9643	0.9722
Test Acc	0.9694	0.9638	0.9604
Test MacroF1	0.9692	0.9636	0.9513

TABLE V
PERFORMANCE OF CLASSIFIERS

Similar to Question 1, here also the MAP estimates worsen the classifier. Also, results shown in Table V are obtained on small part of dataset (due to enormous time required in estimation). Moreover, increasing the number of components from 2, decreases the accuracy. Thus, similar results (co-incidently) follow here i.e. the data is coming from multivariate gaussian (for each class conditional density)

B. Logistic Regression

Now, a 1-layer neural network is implemented i.e. we assume that $\sigma(w_j^T x)$ approximates $P(Y = j/X)$. The model is trained with different loss functions and the results are compared. After selecting the loss function, attempt is made to improve the performance of the model by going into higher dim. space i.e. training $\sigma(w_j^T \phi(x))$ where ϕ is polynomial kernel.

	MAE	**MSE**	**Cross Entropy**
Val. Acc	0.6431	0.6875	0.8878
Val. MacroF1	0.5271	0.6113	0.8860
Train Acc	0.6433	0.7482	0.9238
Test Acc	0.6425	0.7465	0.9182
Test MacroF1	0.5256	0.6947	0.9180

TABLE VI
LOGISTIC REGRESSION WITH DIFFERENT LOSS FUNCTIONS (FULL BATCH GRADIENT DESCENT)

From Table VI, it is evident that cross entropy loss is the best metric to train the logistic regression model. The per-class performance metrics and the confusion matrix for the Cross Entropy model is given in Table VII and remaining are given in the Appendix. Comparing the results of MLE and logistic regression, we see that accuracy from logistic regression = 91.8% , from naive bayes = 96.3% , from bayes = 96.9%, whereas the number of parameters are $18, 24, 36$. Thus it gives an interpretation that why MLE is giving better results since it has more rich model.

	Precision	**Recall**	**F1**
0	0.989	0.79	0.878
1	0.923	0.999	0.96
2	0.818	1.0	0.9
3	0.948	0.984	0.965
4	0.922	0.886	0.904
5	0.948	0.86	0.901

	0	**1**	**2**	**3**	**4**	**5**
0	0.79	0.0	0.21	0.0	0.0	0.0
1	0.0	0.999	0.0	0.0	0.0	0.001
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.007	0.984	0.006	0.003
4	0.009	0.002	0.005	0.054	0.886	0.044
5	0.0	0.072	0.0	0.0	0.068	0.86

TABLE VII
LOGISTIC REGRESSION OVER CROSS ENTROPY LOSS

From the confusion matrices of the three models, we infer that recall of class AbdomentCt (class 0) and HeadCt (class 5) has increased moving from MAE to Cross Entropy. Also, observing the learning curve (gradient descent path) for the three models in Figure 11, it is interesting to note the spikes in the accuracy plot even when the loss plots are smooth. It questions whether or not we should stop at those spikes, for instance in MAE case (orange plot).

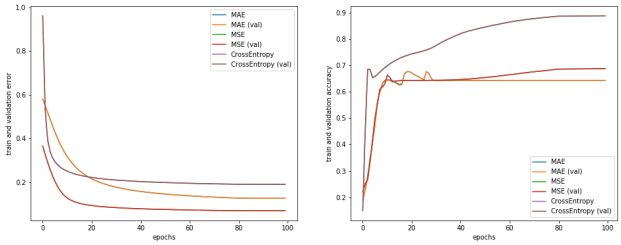

Fig. 11. Metrics of goodness - Loss and Accuracy

Before answering the question, Figure 12 shows the bias variance decomposition for this question obtrained using a polynomial kernel, with cross entropy loss, and increasing the degree. Here, it is even more interesting to see that both accuracy and loss have increased with increase in model complexity. So the second question is whether this is overfitting or not?

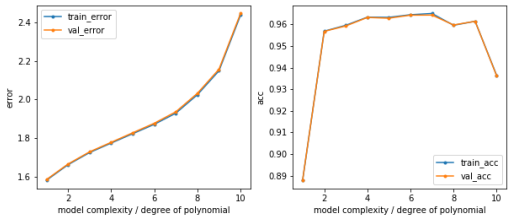

Fig. 12. Bias Variance Decomposition

Now, we shall answer both the questions. Firstly, from the discussion in previous questions, we can understand the increase in the train loss for degrees higher than 5-6 to be due to numerical errors in the computer. Since the loss measures the deviation of predicted values from true values and the accuracy measures the performance of the model, loss is a measure used in the training process to find the "best" parameter values for the model. In short, it is not necessary that high accuracy means low loss. Thus, we should not stop at spikes since we want to train the model well (in terms of loss). Anwering second question - the model has improved (and not overfit) because we assess its performance through accuracy.

	Gradient Descent	**Stochastic GD**
Train Acc	0.9658	0.9592
Test Acc	0.9645	0.9587
Test MacroF1	0.9643	0.9584

TABLE VIII
4^{th} DEGREE POLYNOMIAL LOGISTIC REGRESSION (CROSS ENTROPY)

Thus the best regression model is the logistic 4^{th} degree polynomial regression model. For completeness, results of SGD are also shown in Table in VIII. The per-class performance metrics and confusion matrix for this model are given in Table IX. Even though there is a slight fall in the performance metrics in SGD, but notable difference in the training time was observed.

	Precision	**Recall**	**F1**
0	0.982	0.978	0.98
1	0.953	0.978	0.965
2	0.97	1.0	0.985
3	0.968	0.982	0.975
4	0.983	0.9	0.939
5	0.932	0.951	0.942

	0	**1**	**2**	**3**	**4**	**5**
0	0.978	0.0	0.022	0.0	0.0	0.0
1	0.0	0.978	0.0	0.0	0.0	0.022
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.003	0.0	0.005	0.982	0.01	0.0
4	0.015	0.0	0.004	0.032	0.9	0.048
5	0.0	0.043	0.0	0.0	0.006	0.951

TABLE IX
LOGISTIC 4^{th} DEGREE POLYNOMIAL REGRESSION

C. Non Parametric Methods

Figure 13 (left) show the bias variance decomposition by changing the model complexity. In finding the k-nearest neighbours, for distance metric Infinite (represented as 0) norm, 1-norm and 2-norm are compared as shown in Figure 13(right). However, similar to Question 1, we state that the three norms give nearly same performance.

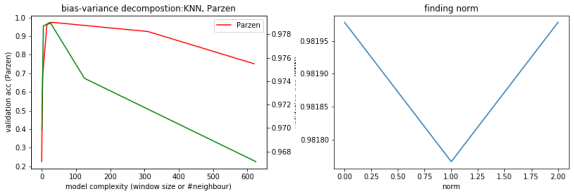

Fig. 13. Bias Variance Decomposition

The results are shown in Table X. Even though the MLE/MAP and non-parametric classifiers perform better than the logistic regression (by 1%), the training (and/or testing) time in the former methods is significantly higher, by almost 10 times in non param method, than the regression method. Thus, this Question presents the need for modern deep learning classifiers for more complex and huge datasets.

	KNN	**Parzen Windows**
Val. Acc	0.9797	0.9737
Val. MacroF1	0.9802	0.9741
Test Acc	0.9788	0.9760
Test MacroF1	0.9790	0.9761

TABLE X
KNN AND PARZEN WINDOW CLASSIFIERS

The per-class precision and confusion matrix of the better non param classifier, KNN, is shown here and the other one's in the Appendix.

	Precision	**Recall**	**F1**
0	0.996	0.996	0.996
1	0.995	0.985	0.99
2	0.994	0.999	0.996
3	0.97	0.978	0.974
4	0.967	0.932	0.949
5	0.953	0.984	0.969

	0	**1**	**2**	**3**	**4**	**5**
0	0.996	0.0	0.004	0.0	0.0	0.0
1	0.0	0.985	0.0	0.0	0.001	0.015
2	0.001	0.0	0.999	0.0	0.0	0.0
3	0.001	0.0	0.002	0.978	0.019	0.0
4	0.002	0.0	0.0	0.03	0.932	0.035
5	0.0	0.004	0.0	0.0	0.012	0.984

TABLE XI
KNN ON MULTI-CLASS DATASET

APPENDIX

A. Binary Classification - Question 1 : Health Dataset Visualization

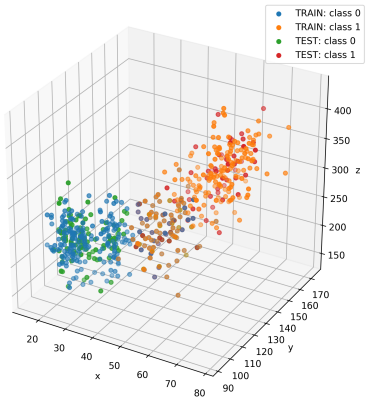

Fig. 14. Health Data

B. Binary Classification - Question 1: Logistic Regression - Stochastic Gradient Descent

	MAE	MSE	CrossEntropy
Val Acc	0.8286	0.8469	0.8469
Val F1	0.8278	0.8179	0.8165
Train Acc	0.851	0.851	0.851
Test Acc	0.8714	0.8714	0.8714
Precision	0.8571	0.8571	0.8571
Recall	0.8478	0.8478	0.8478
F1	0.8525	0.8525	0.8525

TABLE XII
LOGISTIC REGRESSION CLASSIFIERS WITH DIFFERENT LOSS FUNCTIONS (SGD)

C. Regression problem - Question 2: Elastic Net Regularization

Here, it is seen that α, coefficient of L2 penalty penalizes the training loss more than β, coefficient of L1 penalty.

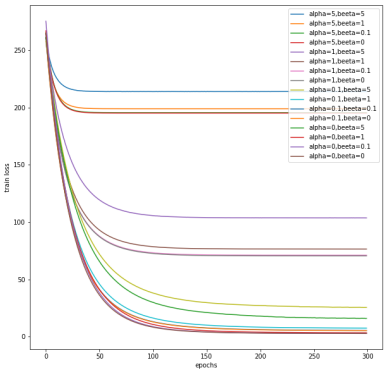

Fig. 15. Elastic Net Regularization

D. Regression problem - Question 2: Linear regression using MAE Loss function

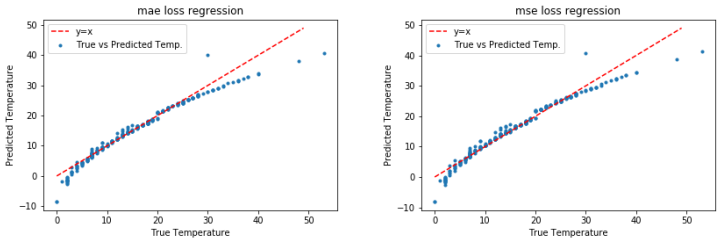

Fig. 16. Linear regression comparison between MAE and MSE

E. Question 3: MLE - Bayes Gaussian

	Precision	Recall	F1
0	0.997	0.986	0.992
1	0.947	0.985	0.966
2	0.986	0.998	0.992
3	0.962	0.984	0.973
4	0.974	0.922	0.947
5	0.95	0.942	0.946

	0	1	2	3	4	5
0	0.986	0.0	0.014	0.0	0.0	0.0
1	0.0	0.985	0.0	0.0	0.0	0.015
2	0.0	0.0	0.998	0.0	0.002	0.0
3	0.001	0.0	0.0	0.984	0.014	0.0
4	0.002	0.0	0.0	0.039	0.922	0.037
5	0.0	0.05	0.0	0.0	0.008	0.942

TABLE XIII

F. Question 3: MLE - Naive Bayes Gaussian

	Precision	Recall	F1
0	0.997	0.986	0.992
1	0.937	0.996	0.966
2	0.986	0.998	0.992
3	0.941	0.981	0.96
4	0.958	0.906	0.931
5	0.964	0.918	0.941

	0	**1**	**2**	**3**	**4**	**5**
0	0.986	0.0	0.014	0.0	0.0	0.0
1	0.0	0.996	0.0	0.0	0.0	0.004
2	0.0	0.0	0.998	0.002	0.0	0.0
3	0.001	0.0	0.0	0.981	0.018	0.0
4	0.002	0.0	0.0	0.06	0.906	0.031
5	0.0	0.06	0.0	0.0	0.022	0.918

TABLE XIV

G. Question 3: MLE - Bayes GMM

	Precision	Recall	F1
0	0.987	0.983	0.992
1	0.946	0.979	0.923
2	0.956	0.998	0.912
3	0.912	0.980	0.983
4	0.984	0.921	0.951
5	0.951	0.932	0.956

	0	**1**	**2**	**3**	**4**	**5**
0	0.936	0.0	0.064	0.0	0.0	0.0
1	0.01	0.945	0.0	0.0	0.0	0.045
2	0.0	0.0	0.993	0.0	0.007	0.0
3	0.011	0.0	0.0	0.985	0.004	0.0
4	0.002	0.0	0.0	0.041	0.920	0.037
5	0.0	0.03	0.0	0.0	0.008	0.962

TABLE XV

H. Question 3: Logistic Regression over MAE Loss

	Precision	Recall	F1
0	0.0	0.0	0.0
1	0.57	1.0	0.726
2	0.495	1.0	0.663
3	0.936	0.984	0.96
4	0.724	0.908	0.806
5	0.0	0.0	0.0

	0	**1**	**2**	**3**	**4**	**5**
0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.006	0.984	0.009	0.0
4	0.0	0.013	0.012	0.067	0.908	0.0
5	0.0	0.664	0.0	0.0	0.336	0.0

TABLE XVI

I. Question 3: Logistic Regression over MSE Loss

	Precision	Recall	F1
0	0.779	0.026	0.051
1	0.782	1.0	0.878
2	0.503	1.0	0.67
3	0.94	0.983	0.961
4	0.843	0.896	0.869
5	0.965	0.6	0.74

	0	**1**	**2**	**3**	**4**	**5**
0	0.026	0.0	0.974	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.007	0.983	0.008	0.002
4	0.008	0.007	0.006	0.062	0.896	0.02
5	0.0	0.242	0.0	0.0	0.158	0.6

TABLE XVII

J. Question 3 : Parzen Windows

	Precision	Recall	F1
0	0.982	0.99	0.986
1	0.999	0.974	0.987
2	0.99	1.0	0.995
3	0.967	0.981	0.974
4	0.978	0.92	0.948
5	0.946	0.991	0.968

	0	**1**	**2**	**3**	**4**	**5**
0	0.99	0.0	0.01	0.0	0.0	0.0
1	0.0	0.974	0.0	0.0	0.001	0.025
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.006	0.0	0.0	0.981	0.012	0.0
4	0.012	0.0	0.0	0.034	0.92	0.034
5	0.0	0.0	0.0	0.0	0.008	0.991

TABLE XVIII
PARZEN WINDOWS ON MULTI-CLASS DATASET