E1 213: Pattern Recognition and Neural Networks **Spring 2021**

Assignment 2: Linear Models

Author: Abijith J. Kamath *Email:* abijithj@iisc.ac.in

1 Introduction

Let patterns be in $\mathbb{R}^d$ and consider the 2-class classification problem - given a pattern $\mathbf{x}$, classify the pattern with a label y. In classification using linear least-squares and logistic regression the training data $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^n$ is used to directly learn a discriminative function that scores the new patterns, and the classification is achieved by thresholding the score.

1.1 Linear Least-Squares Classification

Consider the two-class classification problem of classifying patterns into labels $y \in \{-1, +1\}$. The discriminative function in linear least-squares classification is a function that is linear in the weights, using features of the patterns. In this case, the patterns are directly used as the features, and the discriminative function is given as:

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0 = \mathbf{W}^\mathsf{T} \tilde{\mathbf{x}}, \tag{1.1}$$

where $\mathbf{W} = [w_0, w_1, \cdots, w_d] \in \mathbb{R}^{d+1}$ are the weights to be learnt from the training samples and $\tilde{\mathbf{x}}$ is the augmented feature vector with 1 padded as its first entry. The classification is then done using the sign of the score function $f(\mathbf{x})$.

This can be easily extended to a multi-class classifier by consider similar score functions for each class. The labels are converted to one-hot vectors. The decision rule is to choose the class that gives the largest score.

The weights are obtained by minimising the ℓ^2-norm between the predictions $f(\mathbf{x}^i)$ and $y^{(i)}$:

$$J(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{n} \left(\mathbf{W}^\mathsf{T} \mathbf{x}^{(i)} - y^{(i)} \right)^2. \tag{1.2}$$

The minimiser of $J(\mathbf{W})$ can be obtained in closed form as $\mathbf{W}^* = \mathbf{A}^\dagger \mathbf{y}$, where $\mathbf{A} = [\mathbf{x}^{(1)}\ \mathbf{x}^{(2)}\ \cdots\ ; \mathbf{x}^{(n)}]^\mathsf{T}$ and $\mathbf{y} = [y^{(1)}\ y^{(2)}\ \cdots\ y^{(n)}]^\mathsf{T}$. This directly extends to the multi-class classifier with the labels in $\mathbf{y}$ taken to be one-hot vectors. The closed form solution will then give the complete weight matrix.

1.2 Logistic Regression

Consider the two-class classification problem of classifying patterns into labels $y \in \{-1, +1\}$. The discriminative function in logistic regression is the sigmoid:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{W}^\mathsf{T}\tilde{\mathbf{x}}}}, \tag{1.3}$$

which is treated as the probability of assigning $y = +1$ to the pattern $\mathbf{x}$ ($\mathbf{W}$ and $\tilde{\mathbf{x}}$ have the same meaning as above). If the score is less than $1/2$, the pattern is classified into $y = -1$.

The weights are obtained from the training samples by maximising the likelihood using gradient descent. The weights cannot be obtained in closed form like in Linear Least-Squares classification. The gradient descent updates for the weights is given by:

$$\mathbf{W}^+ = \mathbf{W} + \eta \sum_{i=1}^{n} \left(y^{(i)} - \sigma(\mathbf{W}^\mathsf{T}\tilde{\mathbf{x}}^{(i)})\right) \tilde{\mathbf{x}}^{(i)}, \tag{1.4}$$

where σ is the sigmoid function and η is the step size.

Logistic regression can be extended to multi-class classification by taking the score function for each class to be the softmax function, which is interpreted as assigning the probability of the pattern being in each class and then choosing the class that gives the highest probability.

2 Linear Classification on Synthetic Data

Problem (1.a) Gamma-distributed Classes in 2D

Let the class conditionals be modelled as Gamma distributions with different shape and scale parameters. The 2-class linear least-squares (LS) classifier is trained by min-

imising (1.2) and logistic regression (LOG) is trained using gradient descent with constant step-size using (1.4).

Results: Figure 1 shows the accuracies of classification in confusion matrices and the samples along with the discriminant function. The first and third columns shows the confusion matrices of the linear classifier and logistic regression, respectively with varying training sizes. The second and fourth column shows the samples along with the discriminant function. The accuracies reported are averaged over 10000 realisations and one of the discriminant functions is plotted.

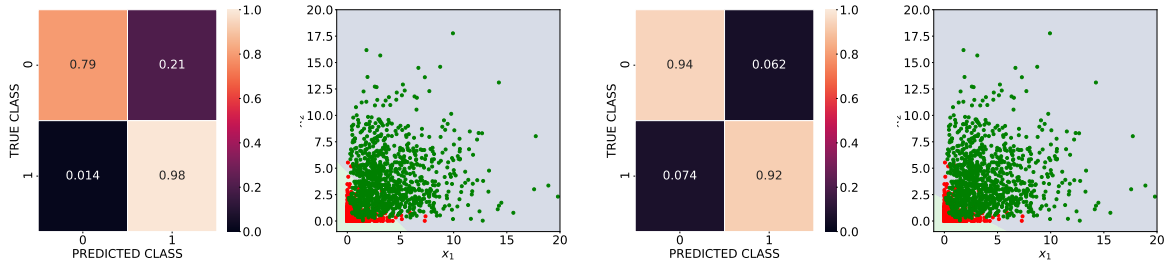
Inferences: It can be seen that the accuracy increases with increasing training size from 10, 50, 100, 500 and 999 in both classifiers. Between the classifiers, logistic regression consistently gives better accuracy. Since the x_1 and x_2 coordinates are independent and Gamma distributed, it is known that the linear classifier has a slope of -1. It can be seen that logistic regression achieves this with fewer training samples. This is reflected in the accuracy.

Problem (1.b) Uniform-distributed Classes in 2D

Let the class conditionals be modelled as uniform distributions with different supports. The 2-class linear least-squares (LS) classifier is trained by minimising (1.2) and logistic regression (LOG) is trained using gradient descent with constant step-size using (1.4).

Results: Figure 2 shows the accuracies of classification in confusion matrices and the samples along with the discriminant function. The first and third columns shows the confusion matrices of the linear classifier and logistic regression, respectively with varying training sizes. The second and fourth column shows the samples along with the discriminant function. The accuracies reported are averaged over 10000 realisations and one of the discriminant functions is plotted.

Inferences: It can be seen that the accuracy of the linear classifier does not vary with increasing training size. Since the least-squares cost equally penalises all points, class 0 is always perfectly classified and the discriminant function always has a large offset owing to the larger spread of class 1. In logistic regression, the accuracy increases with

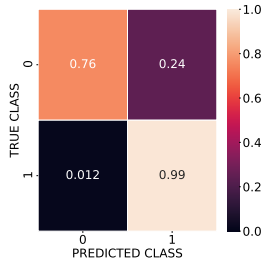


(a) LS, Size 10

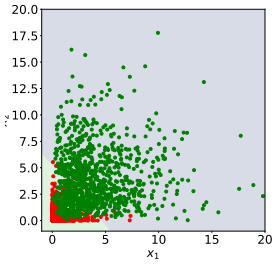
(b) LS Classifier

(c) LOG, Size 10

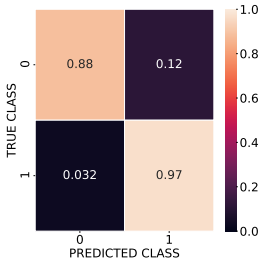
(d) LOG Classifier



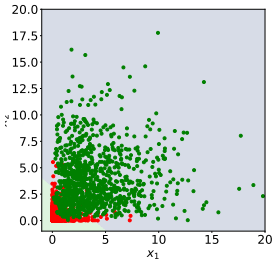
(e) LS, Size 50



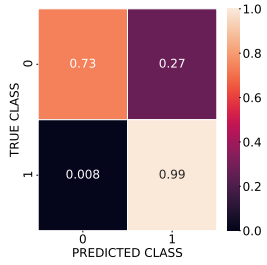
(f) LS Classifier



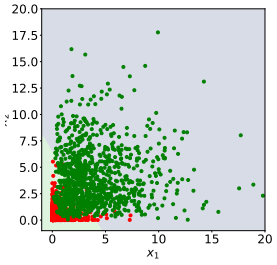
(g) LOG, Size 50



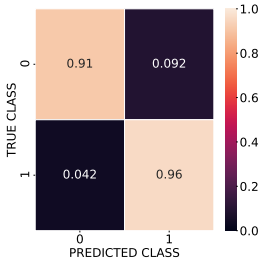
(h) LOG Classifier



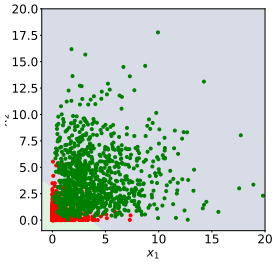
(i) LS, Size 100



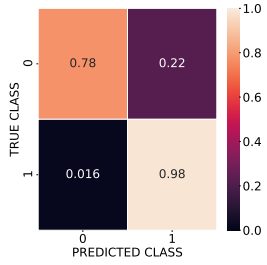
(j) LS Classifier



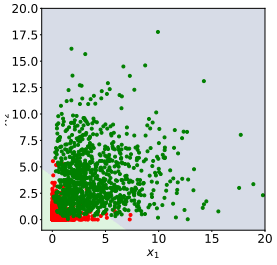
(k) LOG, Size 100



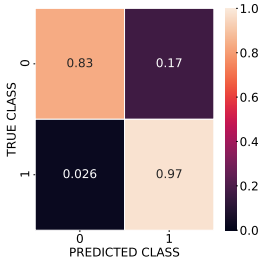
(l) LOG Classifier



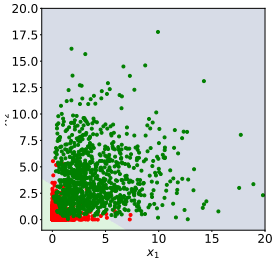
(m) LS, Size 500



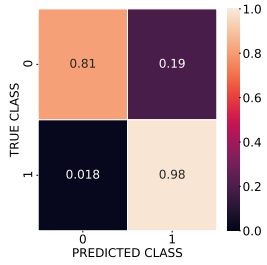
(n) LS Classifier



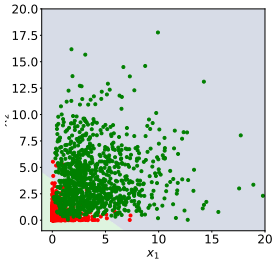
(o) LOG, Size 500



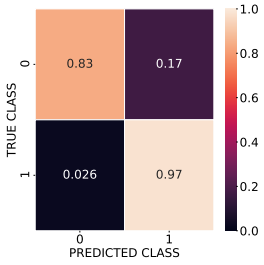
(p) LOG Classifier



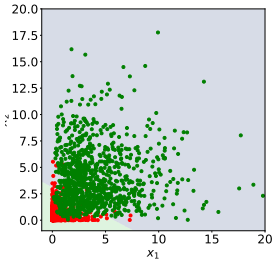
(q) LS, Size 999



(r) LS Classifier

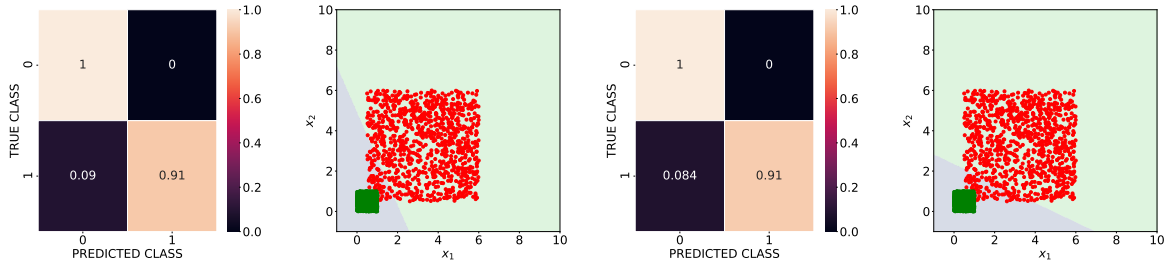


(s) LOG, Size 999



(t) LOG Classifier

Figure 1: Linear Least-Squares Classifier and Logistic Regression on Gamma distributed data.

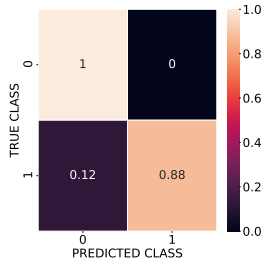


(a) LS, Size 10

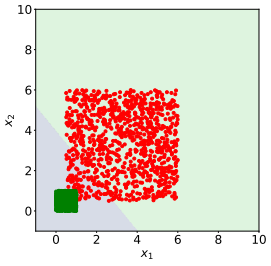
(b) LS Classifier

(c) LOG, Size 10

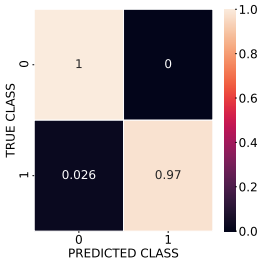
(d) LOG Classifier



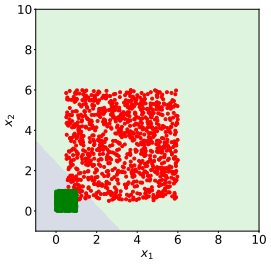
(e) LS, Size 50



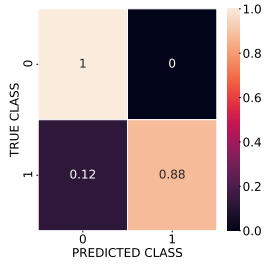
(f) LS Classifier



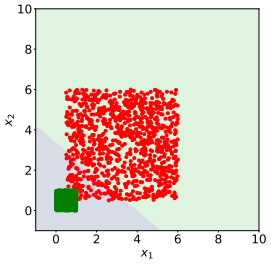
(g) LOG, Size 50



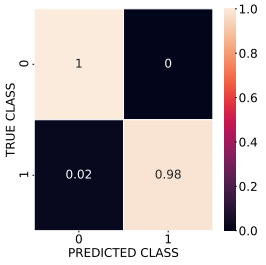
(h) LOG Classifier



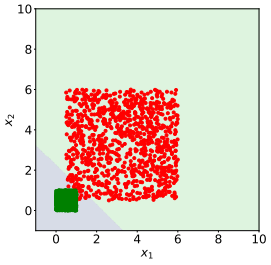
(i) LS, Size 100



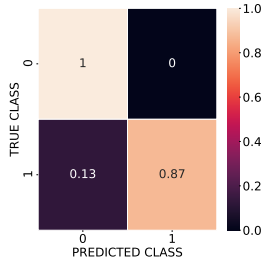
(j) LS Classifier



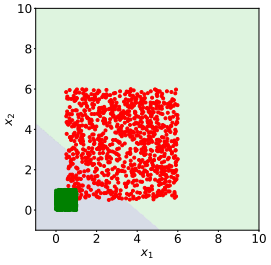
(k) LOG, Size 100



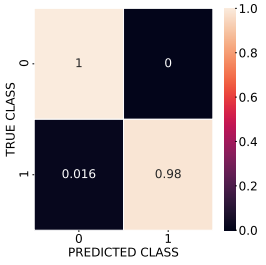
(l) LOG Classifier



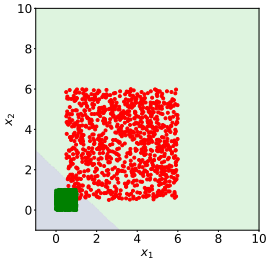
(m) LS, Size 500



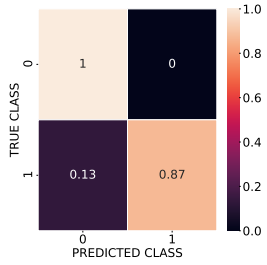
(n) LS Classifier



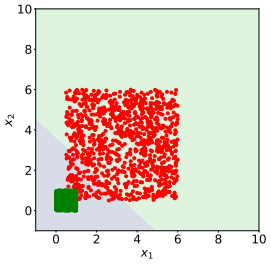
(o) LOG, Size 500



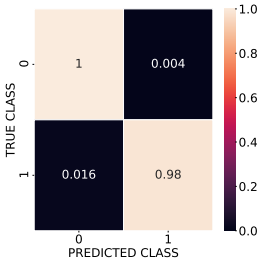
(p) LOG Classifier



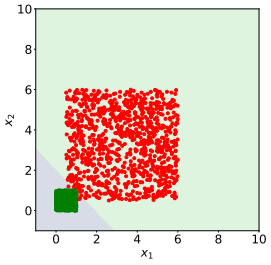
(q) LS, Size 999



(r) LS Classifier



(s) LOG, Size 999



(t) LOG Classifier

Figure 2: Linear Least-Squares Classifier and Logistic Regression on Uniform distributed data.

increasing training size. It is known that the Bayes' classifier for such class conditionals passes through the points of intersection of the squares generated by the limits of the class conditionals. With increasing training size, logistic regression approximately achieves the Bayes' classifier. In this case, logistic regression outperforms linear least-squares classification.

Problem (1.c) Gaussian-distributed Classes in 10D

Let the class conditionals be modelled as Gaussian distributions with different mean and identity covariance. The 2-class linear least-squares (LS) classifier is trained by minimising (1.2) and logistic regression (LOG) is trained using gradient descent with constant step-size using (1.4).

Results: Figure 3 shows the accuracies of classification in confusion matrices. The classifier and the training size used are mentioned in the respective captions. The accuracies reported are averaged over 10000 realisations.

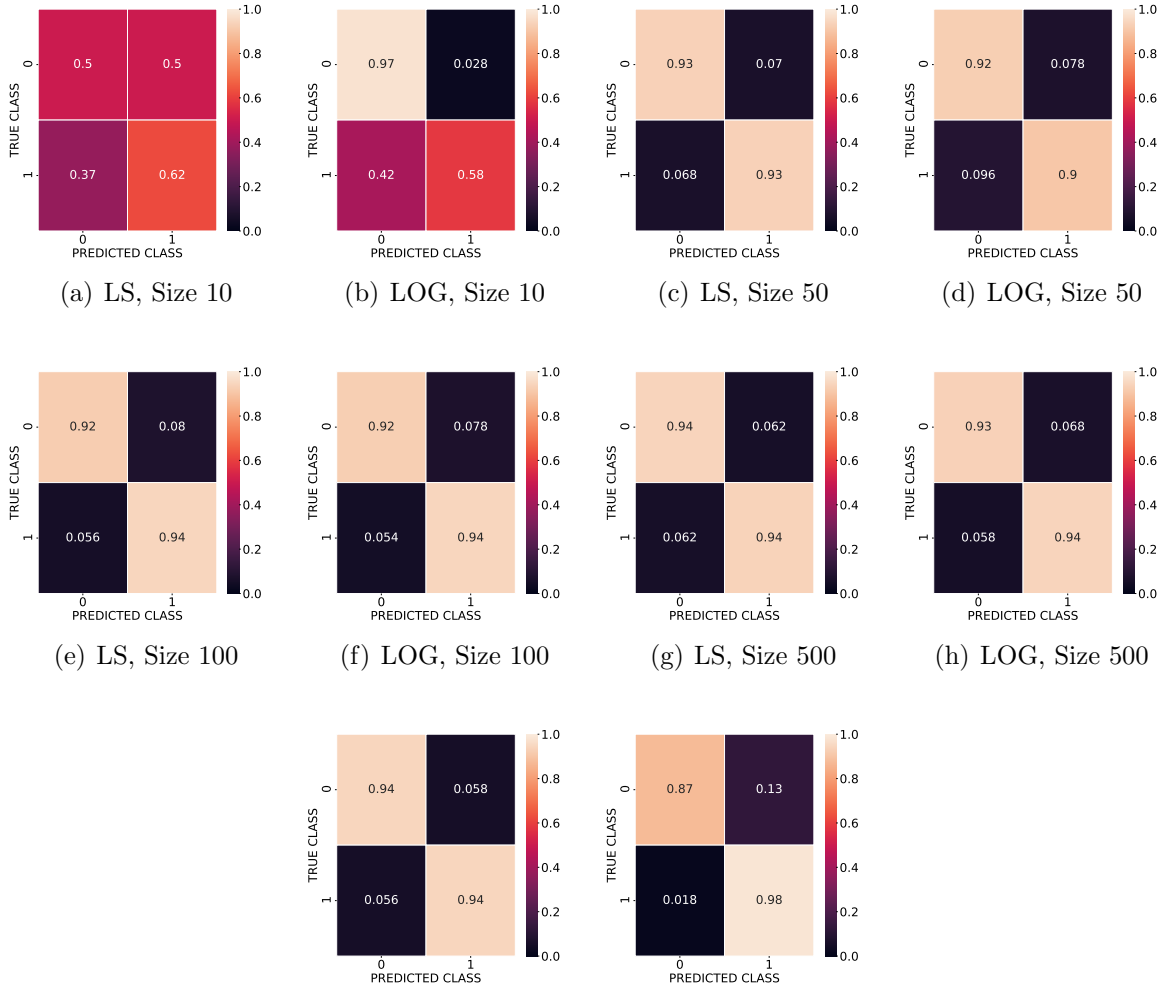
Inferences: It can be seen that the accuracy of both the classifiers increase to the similar values. Since the class conditionals are Gaussian and have identity covariance, the linear least-squares classifier and logistic regression give approximately the same discriminant function.

3 Multi-Class Linear Classification on Iris Dataset

The `iris` data set contains features in $\mathbb{R}^5$ and classifies into three classes – Iris Sentosa, Iris Versicolour and Iris Virginica.

Problem (2.i) One vs. Rest Classifier

In the one vs. rest mode, three binary classifiers are learnt as Iris Sentosa vs. not Iris Sentosa, Iris Versicolour and not Iris Versicolour and Iris Virginica and not Iris Virginica. In this mode of multi-class classification, ambiguities may arise when multiple class flag



(a) LS, Size 10 (b) LOG, Size 10 (c) LS, Size 50 (d) LOG, Size 50

(e) LS, Size 100 (f) LOG, Size 100 (g) LS, Size 500 (h) LOG, Size 500

(i) LS, Size 999 (j) LOG, Size 999

Figure 3: Linear Least-Squares Classifier and Logistic Regression on Gaussian distributed data.

the feature into the true class. The linear least-squares classifier are trained by minimising (1.2).

Results: Figure 4 shows the confusion matrices for each class with varying train-test split fraction of the dataset. The resultant accuracies are shown in the respective titles. The results are averaged over 10000 realisations.

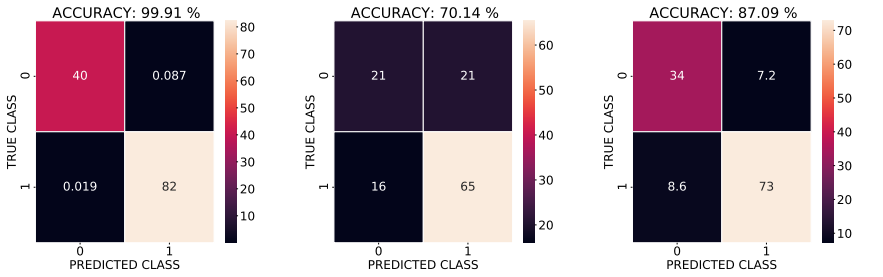
Inference: In general, it can be seen that the accuracy of classification increases in each class with increasing test size. However, this does not give any information to resolve ambiguities. The scores corresponding to each class are lost, and three decisions are obtained. Such multi-class classification is useful when the decision at interest is to know if the class is Iris Sentosa vs. not Iris Sentosa, or so on and the decision of Iris Versicolour and Iris Viriginica are irrelevant.

Problem (2.ii) Multiclass Classifier

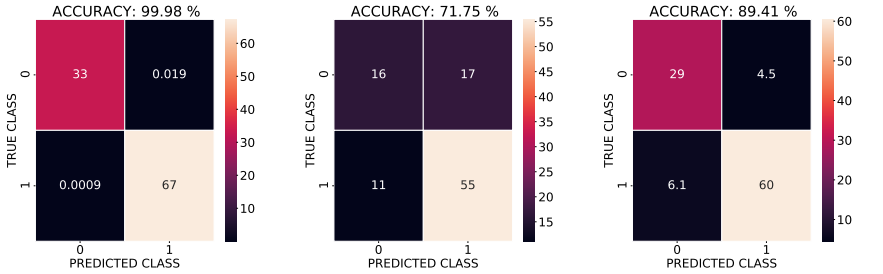
In the multi-class mode, one 3-class classifier using the method described in Section 1.1, by considering labels to be corresponding one-hot vectors to learn a score function that scores classification for each class. In this mode, there is no ambiguity in classifying a new feature in to one of the three classes, as there is always a unique class that provides the highest score (ties maybe resolved arbitrarily). However, this may come at a cost of reduced accuracy in classification.

Results: Figure 5 shows the confusion matrices for each class with varying train-test split fraction of the dataset. The resultant accuracies are shown in the respective titles. The results are averaged over 10000 realisations.

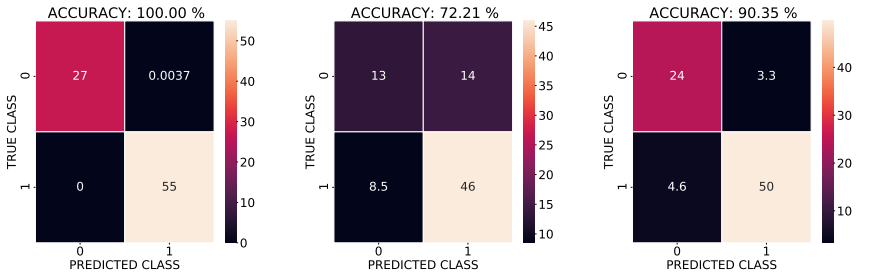
Inference: It can be seen that the accuracy of classification increases with increasing train-test split fraction. In such classification, the decision is not ambiguous – the output for each new feature is a unique class label. However, as compared to accuracies in Figure 4, the accuracies are lower, i.e. an unambiguous decision is made with lesser confidence. Such multi-class classification is useful when the decision at interest is to know exactly which class the new feature belongs, where all classifications are relevant.



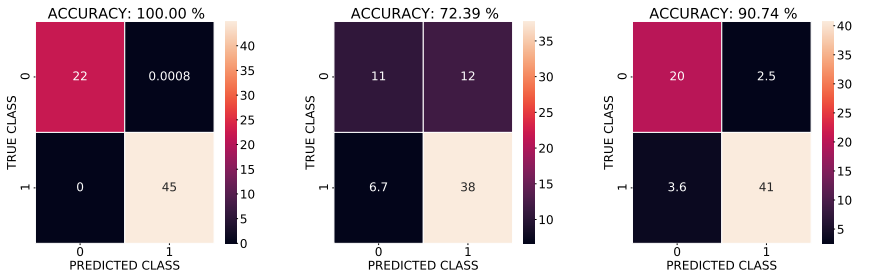
(a) Class 0, Fraction 0.2 (b) Class 1, Fraction 0.2 (c) Class 2, Fraction 0.2



(d) Class 0, Fraction 0.4 (e) Class 1, Fraction 0.4 (f) Class 2, Fraction 0.4

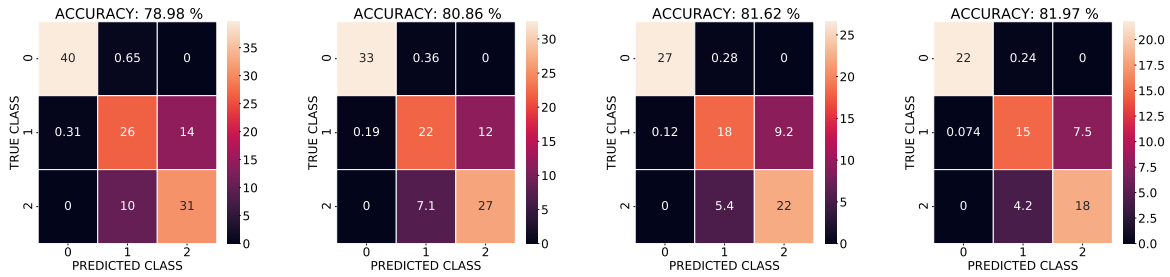


(g) Class 0, Fraction 0.6 (h) Class 1, Fraction 0.6 (i) Class 2, Fraction 0.6



(j) Class 0, Fraction 0.8 (k) Class 1, Fraction 0.8 (l) Class 2, Fraction 0.8

Figure 4: One vs. Rest Linear Least-Squares Classifier with varying train-test split fraction.



(a) Fraction 0.2 (b) Fraction 0.4 (c) Fraction 0.6 (d) Fraction 0.8

Figure 5: Multi-class Linear Least-Squares Classifier with varying train-test split fraction.

4 Linear Classification on Financial Data

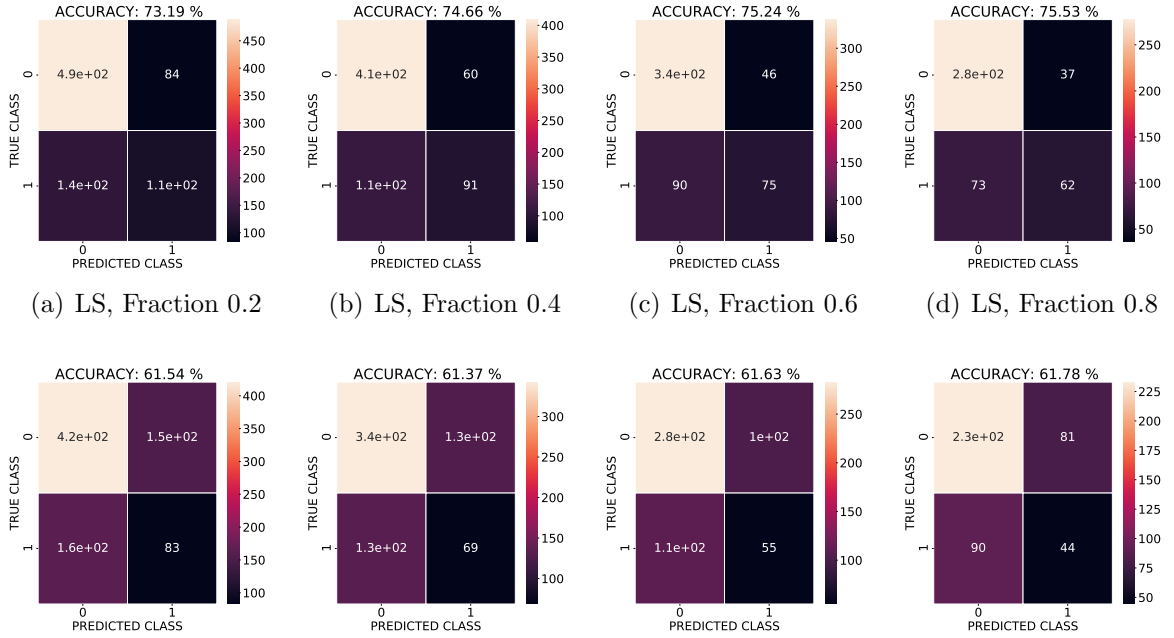
The `german.data-numeric` dataset contains features in $\mathbb{R}^{24}$ related to finance of individuals and classifies into two classes – bad and good. The data provided are in varying ranges, some are scores and some are binary flags. Hence, some data preprocessing maybe needed to obtain useful results.

Implementation: Linear Least-Squares and Logistic Regression are used for classification. Two data preprocessing methods are considered – normalisation and whitening. The linear least-squares classifier is trained by minimising (1.2) and logistic regression (LOG) is trained using gradient descent with constant step-size using (1.4).

Results: Figure 6, 7, 8 shows the confusion matrices of classification using linear least-squares and logistic regression, using no preprocessing, data normalising and data whitening respectively, with varying train-test split fraction. The results are averaged over 10000 realisations.

Inference on no preprocessing: The accuracy of either of the classifiers does not appear to change with changing train-test split fraction. Linear least-squares classifier performs better than logistic regression, for any given train-test split fraction.

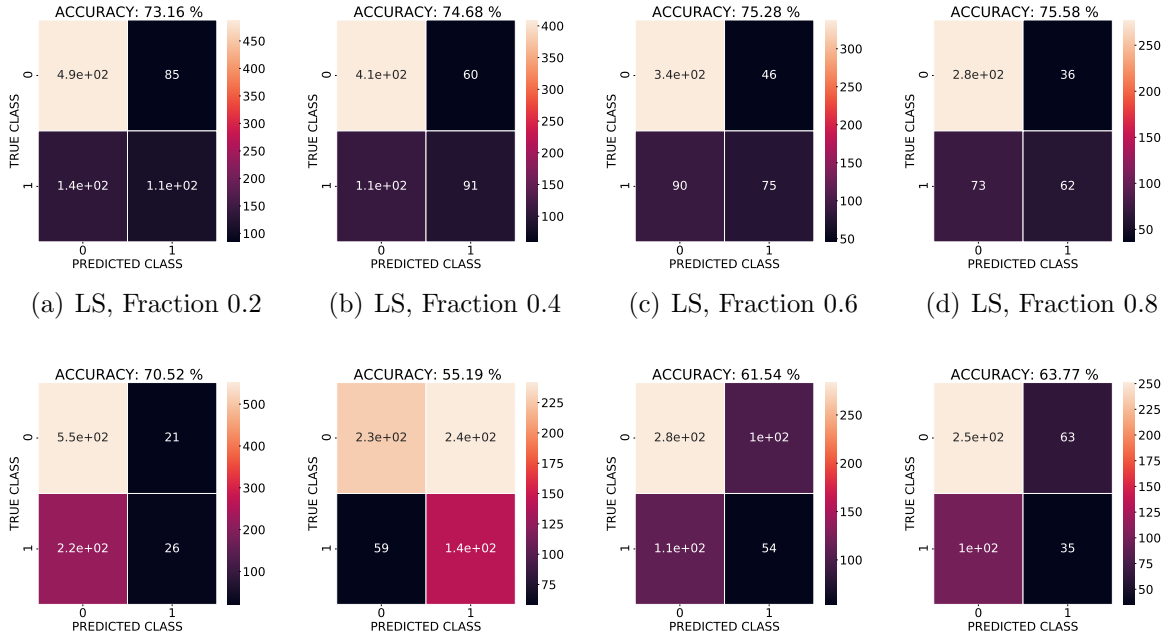
Inference on preprocessing using normalisation: The accuracy of either of the classifiers increases with increase in train-test split fraction. Even in this case, linear least-squares classifier performs better than logistic regression, for any given train-test split fraction. However, there is a noticable trend of increasing accuracy in classification using logistic regression. Logistic regression with train-test split fraction of 0.2 gives an



(a) LS, Fraction 0.2 (b) LS, Fraction 0.4 (c) LS, Fraction 0.6 (d) LS, Fraction 0.8

(e) LOG, Fraction 0.2 (f) LOG, Fraction 0.4 (g) LOG, Fraction 0.6 (h) LOG, Fraction 0.8

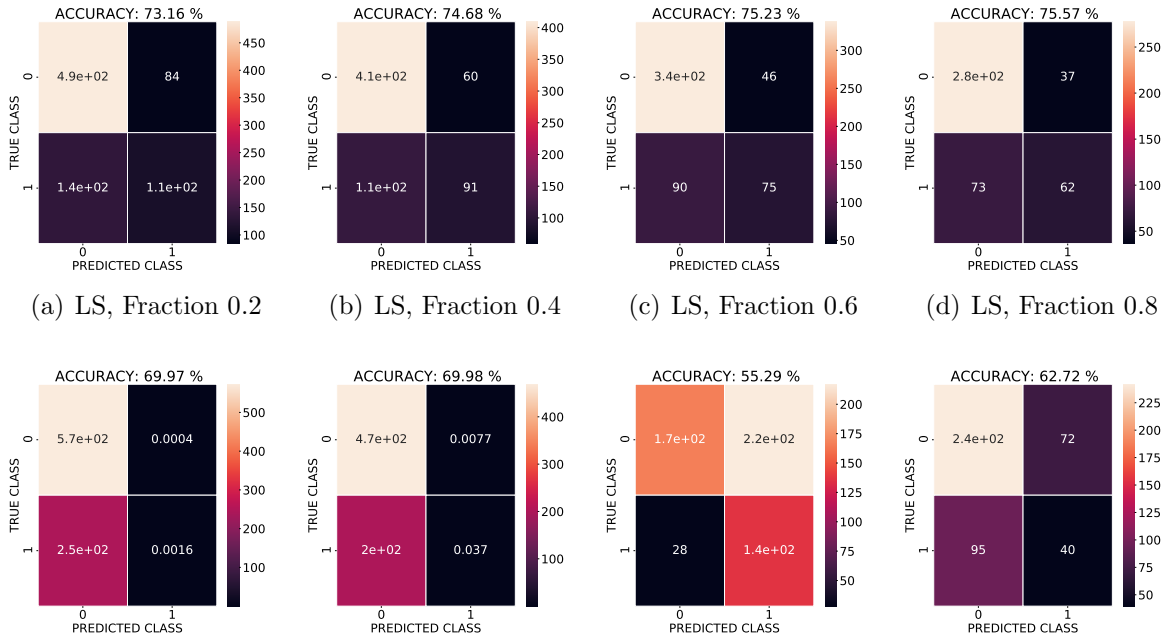
Figure 6: Linear Least-Squares Classifier on German Numeric dataset, with varying train-test split fraction and no data preprocessing.



(a) LS, Fraction 0.2 (b) LS, Fraction 0.4 (c) LS, Fraction 0.6 (d) LS, Fraction 0.8

(e) LOG, Fraction 0.2 (f) LOG, Fraction 0.4 (g) LOG, Fraction 0.6 (h) LOG, Fraction 0.8

Figure 7: Linear Least-Squares Classifier on German Numeric dataset, with varying train-test split fraction and normalising data as preprocessing.



(a) LS, Fraction 0.2 (b) LS, Fraction 0.4 (c) LS, Fraction 0.6 (d) LS, Fraction 0.8

(e) LOG, Fraction 0.2 (f) LOG, Fraction 0.4 (g) LOG, Fraction 0.6 (h) LOG, Fraction 0.8

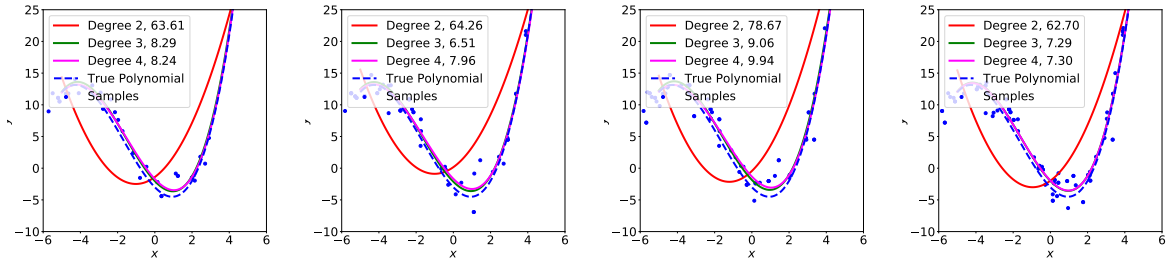
Figure 8: Linear Least-Squares Classifier on German Numeric dataset, with varying train-test split fraction and whitening data as preprocessing.

overall accuracy of 70.52%, however, the false negative rate is large. With higher training sizes, the accuracy in classification with normalisation is higher than in classification without any preprocessing.

Inference on preprocessing using whitening: The accuracy of least-squares classifier shows an increasing trend with increase in train-test split fraction. There is no such clear trend with logistic regression. With lower training sizes, logistic regression gives higher accuracies with higher false negatives, and with higher training sizes, the accuracies are similar to accuracies using normalisation. Either of the preprocessing methods provides better results with higher training size than no preprocessing.

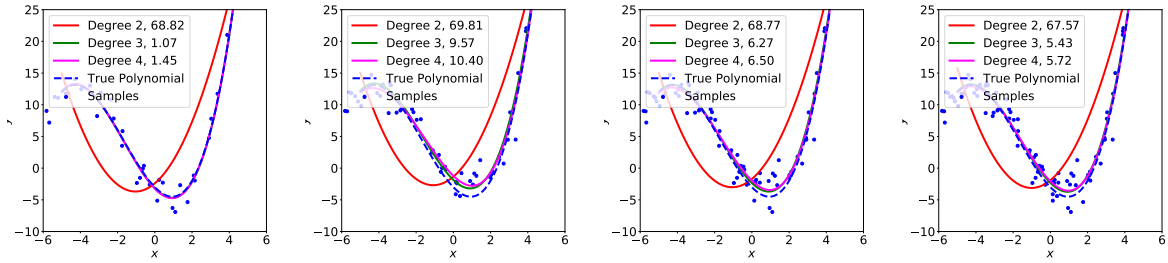
5 Linear Least-Squares Regression in 1D

The regression problem is to learn values in $\mathbb{R}$ for features in $\mathbb{R}^d$. Regression using least-squares is similar to least-squares classification, where the value $f(\mathbf{x})$ is taken to be the value in (1.2). Learning of the weights from the training data is identical to learning in the classification problem, as described in Section 1.1.



(a) Training Samples 40 (b) Training Samples 60 (c) Training Samples 80 (d) Training Samples 99

Figure 9: Linear Least-Squares Regression on synthetic data with random subsampling.



(a) Training Samples 40 (b) Training Samples 60 (c) Training Samples 80 (d) Training Samples 99

Figure 10: Linear Least-Squares Regression on synthetic data with uniform subsampling.

Implementation: Linear least-squares regression is used to fit polynomials of different orders. The features used are $\mathbf{x}$ from the given $1D$ training data and its powers, depending on the order. The training is done using varying training sizes by subsampling the given data randomly and uniformly.

Results: Figure 9 and 10 show samples and regression polynomials learnt from training using random subsampling and uniform subsampling, respectively, for varying training sizes. Polynomials are learnt for orders 2, 3 and 4. The L^2 error between the learnt polynomial and the true polynomial are shown in the legends. These errors are averaged over 10000 realisations. The true polynomials and the samples used are also shown.

Inference: For either case, it can be seen that the errors reduce with increase in training samples. However, the trend is clear in the case of uniform subsampling across all degrees. The errors in the case of random sampling are higher than uniform sampling. These can be attributed to numerical precision errors in the matrix inversion. The feature matrix is known to have a Vandermonde structure and hence the conditioning worsens

when the spacing between the samples reduces. As the training size increases, the error in the degree 3 polynomial is the least.

The errors indicate a method to select the correct order. From the results, the choice of degree 3 is correct. This is in accordance with the minimum description length (MDL) principle. The degree 3 polynomial has the smallest pointwise errors. Hence, the number of bits needed to store the errors, weights and the 1D features is the least.

6 Code Repository

The Python codes to reproduce the results can be found in the GitHub repository `https://github.com/kamath-abhijith/Linear_Models`. Use `requirements.txt` to install the dependencies and the shell scripts to generate the figures.