AMATH 582 HW 4: SVM, LDA, And Decision Trees on MNIST

Hongda Li

March 8, 2021

Abstract

The objective is to develope basic practical and theoretical understandings of some classical machine learning methods. We use the MNIST data set and the models are Linear Discriminant Analysis, Principal Component Analysis, Support Vector Machine, and Binary Decision Tree. Matlab is the major tool. PCA is used as a dimensionality reduction technique for the data. Models are trained on the projection onto the principal component mode to increase efficiency. Models are trained to distinguished pair of of digits, or all classifier all of the digits. Metrics used are the errors between the predicted labels and the actual labels and the confusion matrix. The metrics is measure on training, validations, and test set. Theory and intuitions behind the algorithms and practical details are explored in the paper.

1 Introduction and Overview

We were given the Mnist data set, which are images of hand written digits, packed into tensor of size $28 \times 28 \times N$. 60k of them are used for training, and 10k of them are used for testing. Samples for each digits are approximately homogenous.

The classification that we are interested in are SVM: Support Vector Machine, LDA: Linear Discriminant Analysis, and Decision Tree (DTree).

Principal Component Analysis (PCA) is used as a dimensionality reduction technique for the data set. More information on the theory part.

Confusion matrix is used as a performance metrics for all above classifications. And this metrics is being measured for training, validations, and the test set. It's used because it exposes more information than the errors.

In addition, visualization will be made for PCA to demonstrate the low rank property of the data set.

2 Theoretical Background

Most of the ideas discussed below are referencing sources from Mathworks, Sklearn, lecture notes of Math 253 from San Jose State University, The *Book Elements of Statistical Learning* from Springer by Trevor Hastie et al, and the book *Data-Drive Modeling & Scientific Computation* from the University of Washington by Nathan Kutz.

2.1 Linear Discriminant Analysis

Linear discriminant analysis comes in many different types of variance, and it can be regularized as well. The most simple is: LDA, which seeks to look for a linear subspace such that the projection of the labeled data onto the line has maximal distance from the global central, and minimal variance within the clusters. The model assume homogenous Gaussian distribution of the data points. If Non-homogenous Gaussian clusters are involved, quadratic discriminant analysis is needed for the classifier. However if QDA classifier is used, then the covariance matrix for the features for each cluster will have to be diagonal, implying the fact that projecting onto the PCA modes before the analysis will be helpful for QDA.

WLOG, let's assume a binary classification problems with 2 classes: C_1, C_2. The features of each sample is a vector denoted by x_i and the subspace we project them onto is a vector denoted as v, then, the centroid

for one of the cluster projected onto the line can be expressed as[1]:

$$\mu_1 = \frac{1}{|C_1|} \sum_{x_i \in C_1} \left(v^T x_i\right) = \frac{v^T}{|C_1|} \sum_{x_i \in C_1} \left(x_i\right) \tag{1}$$

And conveniently, the centroid of the cluster can be expressed as:

$$m_1 = \frac{1}{|C_1|} \sum_{x_i \in C_1} \left(x_i\right) \tag{2}$$

Consider the distance between μ_1, μ_2 on the projected space, which can be expressed as:

$$\begin{aligned} (\mu_1 - \mu_2)^2 &= (v^T m_1 - v^T m_2)^2 = (v^T (m_1 - m_2))^2 \\ (v^T (m_1 - m_2))^2 &= v^T (m_1 - m_2) \cdot (m_1 - m_2)^T v \\ S_{\text{between}} &= (m_1 - m_2)(m_1 - m_2)^T \in \mathbb{R}^{d \times d} \end{aligned} \tag{3}$$

Where we have define the rank-one matrix that measures the distance between the projected mean of the clusters.

Now consider another matrix which will be helpful for us: WLOG, the variance for the class C_1 can be computed by:

$$\begin{aligned} s_1^2 &= \sum_{x_i \in C_1} \left((v^T x_i - v^T m_2)^2\right) = \sum_{x_i \in C_1} \left(v^T (x_i - m_1)(x_i - m_1)^T v\right) v^T \\ s_1^2 &= \left[\sum_{x_i \in C_1} \left((x_i - m_1)(x_i - m_2)\right)\right] v = v^T S_{\text{within}} v \end{aligned} \tag{4}$$

Then, the problem of maximizing the centroid of the cluster onto the sub-space and minimizing all the variance of the cluster can be summarized as:

$$\max_{\|v\|=1} \left\{ \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2} \right\} = \max_{\|v\|=1} \left\{ \frac{v^T S_{\text{between}} v}{v^T S_{\text{within}} v} \right\} \tag{5}$$

And computationally, this become an augmented Rayleigh Quotient, which can be solved by looking for vector in the Eigenspace of the matrix $S_{\text{between}} S_{\text{within}}^{-1}$.

Take notice that, this formation is from an optimization point of view instead of using Bayes Theorem, and using the idea of Bayes theorem, one can formulate the problem for Quadratic Discriminant Analysis and derive the computational problem.

Take note that, for multi-class, LDA take the squared distance between each cluster from the global centroid of all the data point as a measure for the spread of the clusters.

In addition, we need to notice that LDA can be used both as a dimensionality reduction technique and a classifier. The Eigenspace of the matrix $S_{\text{between}} S_{\text{within}}^{-1}$ is subspace which, maximal separation between the data occurred. However, it doesn't have to be an orthogonal subspace.

2.2 Principal Component Analysis

Principal Component Analysis is used as a dimensionality reduction technique[2]. Consider the Column Data Matrix X and its SVD decomposition of the matrix:

$$X = U \Sigma V^T \implies U^T X = \Sigma V^T \tag{6}$$

Where: U, V are both orthogonal matrices. Assuming that the matrix X is $m \times n$ and in the case of our data set, $m << n$, then the matrix $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times m}$, and $V^T \in \mathbb{R}^{m \times n}$.

The Interpretation of SVD is simple. We decompose the column data matrix into Principal Components that explains the most variance (Columns of the matrix U) and the singular value, which explain the

importance of each mode (Diagonals of matrix Σ), and the profile vector, which describes the composition for each of the sample (Column vector of the matrix V^T, rows of the matrix V).

Conveniently, the principal components of the data matrix is ordered with the singular value and singular values are ordered in descending order.

This means that, we can truncate the rows of the matrix ΣV^T yet still retaining most of the information for each column of the data matrix. This is the "Eckart-Young" Theorem, a major idea behind PCA and dimensionality reduction. Assuming that we now truncate the SVD into using matrices of the size $m \times k, k \times k, k \times n$ for U, Σ, V^T where $k << m$ then it also has the added benefit that the sample spans all the subspace $\mathbb{R}^k$, implying that the covariance matrices for the classes are going to be non-singular. Which is a requirement for Quadratic Discriminant Analysis.

2.3 Support Vector Machine

The idea of SVM is to find the hyper plane between the clusters such that it maximizes the margin between the boundary of the 2 clusters. This is intrinsically a binary classification method[4].

Computationally the problem is phrased as a Quadratic Programming problem. The SVM is computationally phrased as[6]:

$$\min_{\beta,\beta_0} \|\beta\| \text{ s.t: } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i & \forall i \\ \xi_i \geq 0, \sum_i (\xi_i) \leq \text{constant} \end{cases} \tag{7}$$

Where $y_i \in \{1, -1\}$ are the binary label of the sample, and ξ_i is the penalty for cross the boundary for each of the support vector, and β is the vector the defines the hyperplane.

In practice, to get the best separations for different topology of the cluster, kernel function is often used as a way of mapping the same data from a lower dimension to a high dimension, while the theoretical analysis of the dual problem allows improve the speed and make use of only the pair-wise inner product under the new basis[7].

2.4 Binary Decision Tree

The idea of binary tree is a simple yet powerful concepts. A tree is created to partition the feature space on each node of the tree. Each branching of the tree assert one conditions on one of the attribute, (e.g: age older than 18), and each leaf node consists of samples with almost the same labels.

There are many variance of the decision tree, but for our purposes, we will be focusing on the default that comes with the "fictree" command in matlab[3].

Decision tree unlike method described above, doesn't have smooth decision boundary, making it sensitive to noise and extrapolation of the data. However, it's it has a variety of hyper parameters available for tunning and it's a interpretable[5].

3 Algorithm Implementation and Development

The challenging part of machine learning practice in this case is data preprocessing and dimensionality reduction of the data. Hence for this part we will be highlighting the procedures for PCA, unit variance zero mean Standardization, and MinMax Standardization, and the generic training process of various models.

Others parts of the practical implementations that are not listed here involves the procedures of visualizing the results, setting up the parameters, and filtering out certain digits from the training set.

3.1 Data Standardization

There are 2 standardization techniques used in the training process of various models. The zero mean unit variance standardization(Algorithm 1), which is used for the PCA, and the LDA. The MinMax standardization(Algorithm 2), which is used for the SVM. The SVM only works for points that have relative distance due to the penalty of the corresponding optimization problem, hence MinMax standardization is needed.

Algorithm 1: Algorithm 1: Zero Mean Unit Variance Standardization

Input: Data: D
D := Flattten the Tensor: D into a column data matrix, convert the data type as well.
D := D - mean(D, 1)
D := D./std(D, 1)
Return: D

Algorithm 2: Algorithm 2: MinMax Standardization

Input: Data D
Precondition: Algorithm 1 Has been run on D.
D := data - minAll(D)
D := data./maxAll(D)
Return: D

3.2 PCA Dimensionality Reduction

Algorithm 3: Algorithm 3: PCA Dimensionality Reduction

Input: **Data**: D **EnergyLevel**: E
Preconditions: The data, D is processed by Algorithm 1
Perform SVD on dat matrix D to obtain matrices: S, Sigma, V
EnergyS := cumsum(Sigma)/sumAll(Sigma)
Idx := the index of first number in EnergyS such that it's > E
ProjData := Sigma $*$ V^T
ProjData := ProjData(1: Idx, :)
Projector := U(1: Idx, :)^T
Return: ProjData, Projector

Algorightm 3 describes the procedure for PCA dimensionality reduction. Variable "D" is the data matrix of the images, where each images are packed as columns of the matrix D, and "E" is the cumulative energy level we want for all the modes that we are projecting onto. The function then, returns the data described using all first "Idx" principal modes, and return the projector, which is just the matrix U^T. Note, the function "sumAll" does sum up all the elements in the tensor and it's out put is a scalar.

3.3 Training Generic Models

Algorithm 4 refers to the generic procedures of producing a model given the training and test MNIST Data set. Which is important for checking the constancy of the cross validation and the training set, and it's then benchmarked on the test set to see if there are potential for over-fitting.

For our purpose, we trained SVM, LDA, and Decision Tree using the Gaussian Kernel function for SVM, linear discriminant for LDA and they are all trained on the first 66 principal mode which covers just over 50% of the cumulative energy on the singular value spectrum. The choice of using 50% of the energy is aimed for reducing noise in the data, which will help with models that are sensitive to noise such as the SVM and the decision tree. When training the SVM model, all the data point is normalized using Algorithm 2 to transform the features such that the feature lies in $[0, 1]^6 6$.

4 Computational Results

4.1 Visualizing Low-rank Structure Using PCA

The low-rank structure of the MNIST data is exposed by the decay of the singular value. Shown on left side of Fig 1. In addition, some form of clustering on the mode 1, 5, 8 is shown on the right side of Fig 1 where each color represents a digit, and is having their own approximate region in the projected space. 66 principal modes are used for training all the models model, based on the assumption that most of the variances are noises.

Algorithm 4: Algorithm 4: Generic Model Training Process

1: **Input: Training Data and Labels:** TrD,TrL **Testing Data and Labels:** TD, TL
2: Standardize the data: TrD, TrL, TD, and TL appropriate to the model
3: X, X1, Y, Y1 := Permuate the trainig data, 10% for corss validation, 90% for training using TrD, TrL
4: Train the model using X, Y
5: Produce labels using model on X, Y, X1, Y1
6: Produce labels using model on TD, TL
7: **Return:** All the label produced, data set and the model itself

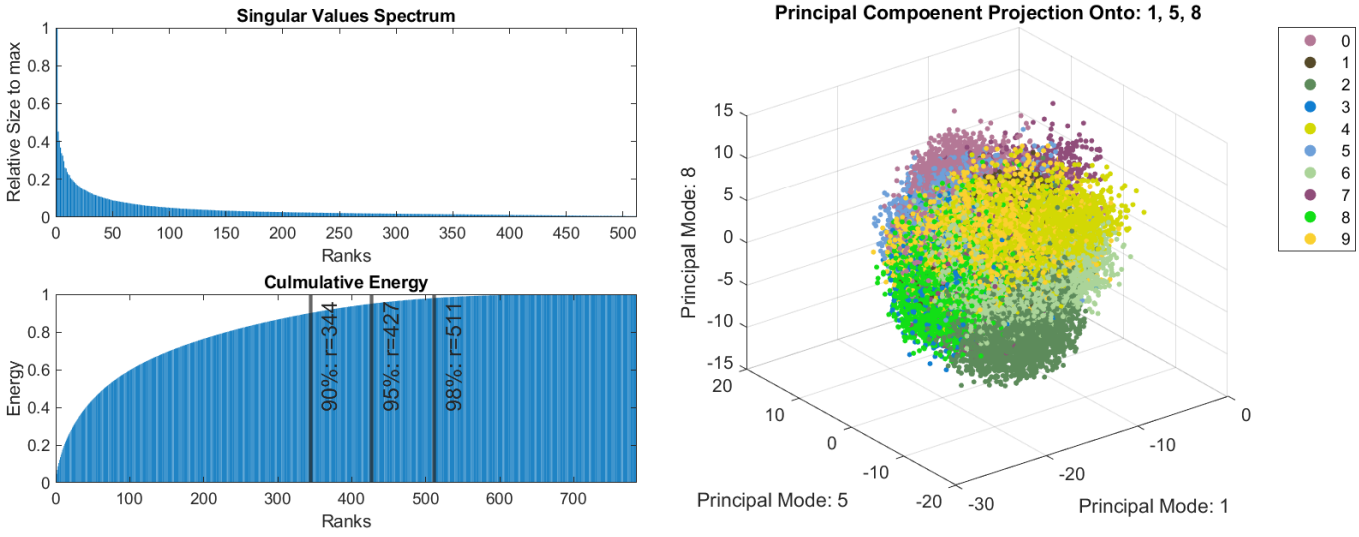

Figure 1: Figure 1: PCA Project and Singular Spectrum

4.2 Performance Comparison on 2,3 Digits Separation

The strategy for looking for the hardest digits to separate is referencing the confusion matrix across the test set of the LDA method on all 10 digits, this is shown in Fig 5. This is achieved by training all the digits on the LDA model on a one vs one setting. Focusing on the misclassified rate for each pairs of digits, it's not hard to see that the digits $4, 9$ are the digits where the model make most of the mistakes. And the digits that the model makes the least amount of mistakes are the digits: $1, 0$. Shown in Fig 2 the LDA misclassified 13.4% of the instances of 9 as 4 and 1.5% instances of 4 as 9 on the test set, but sightly lower on the training set and validation set, indicating a potential for **over fitting**. Between the easiest digits: 0, 1, the LDA models misclassified 0.2% of the digits 0 as 1 and 0.1% of the digits 1 as 0 on the test set (Fig 3). Training error and the test error is extremely close, indicating a very good separations of the distribution between these 2 classes on the PCA modes. LDA shows some efforts when it tries to separate 3 digits, and the confusion matrix is shown under Fig 4

SVM performs better than the LDA method on binary classification of the digits pair $1, 0$ and $4, 9$ and the kernel function is set to "Gaussian". 0.9% of the digit 4 is misclassified as 9 and only 3.9% of the digits 9 is misclassified as 4 on the test set, shown in Fig 7. For the separation between $0, 1$, non of the zero is classified as 1 and only 0.2% instances of 1 is mis-classified(Fig 6). The SVM performs better compare to the LDA methods in both regards.

The Decision tree method performs as good as the SVM models on the $4, 9$ separation but worse on the $0, 1$ separation, this is shown in Fig 9 and Fig 10. The DTree misclassified 0.4% and 0.7% of the "0" as "1" and "1" as "0" on the test set. For $4, 9$, it misclassified 13.9% of 4 as 9 and 14.6% of 9 as 4, higher than training set, indicating a potential **over fit** for the decision tree.

4.3 Performance Comparison on 10 Digits Classifications

Two models that has been benchmarked on all 10 digits are the SVM and the LDA which is shown in Fig 5 and fig 8. Additional testing in the source code indicates that the QDA classifier performs as well as the SVM model. One explanation is that the distribution of samples of each class doesn't have the same amount of variance because LDA is not suited for classes with different amount of variance. Finally, the worst performance of LDA is due to under fitting because it has the same performance across all 3 of the test set and the training set.

5 Summary and Conclusions

The performance improvement on the accuracy and speed for using PCA as a dimensionality reduction technique is drastic. It's implement as extra in the source code, which we demonstrated that the decision tree become useless if the raw digits inputs are use to train the model. One can surmise the improvement if we made the choice of using LDA to find the best subspace on the data projected onto the PCA.

In addition, the importance to rescale the data into the $[0, 1]^n$ unit box in the positive quadrant for SVM is crucial in practice. This is required or the SVM to compute efficiently with a accuracy.

The decision tree model performs decently well only if it's trained on the projection onto the principal modes, which reflects the fact that Decision tree is sensitive to noise and pron to over-fit. To overcome this, one can consider using random forest to improve the accuracy, or Regression tree with a dimensionality reduction technique such as the PCA, or the LDA.

References

[1] Dr. Guanliang Checn. *Math 253: Linear Discriminant Analysis (LDA)*. 2020. URL: https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec11lda.pdf.

[2] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013, p. 387.

[3] *MathWorks: Help Cneter ¿ Documentation ¿ fitctree*. Accessed: 2010-09-30. URL: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm.

[4] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[5] J.Nathan Kutz Steven L. Brunton. *Data-Driven Science and Enineering*. Cambridge University Press, 2019, p. 187.

[6] Joreme Friedman Trevor Hastie Robert Tibshirani. *The Elements of Statistical Learning: Data Ming, Inference, and Prediction 2nd Edition*. Springer, 2017, p. 419.

[7] Joreme Friedman Trevor Hastie Robert Tibshirani. *The Elements of Statistical Learning: Data Ming, Inference, and Prediction 2nd Edition*. Springer, 2017, p. 424.

Appendix A MATLAB Functions

Figure 2: Figure 2: Hardest 2 Digits

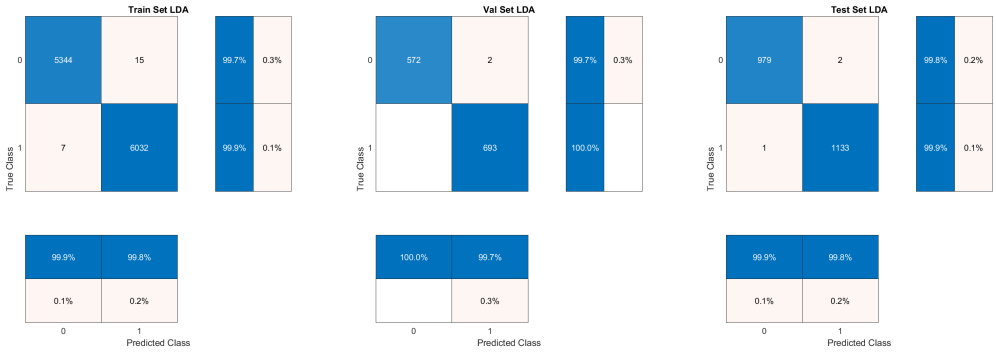

Figure 3: Figure 3: Easiest 2 Digits

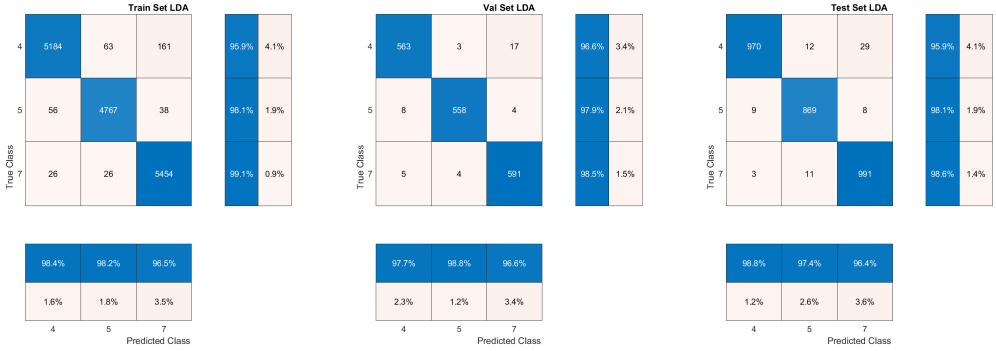

Figure 4: Figure 4: 3 Digits LDA

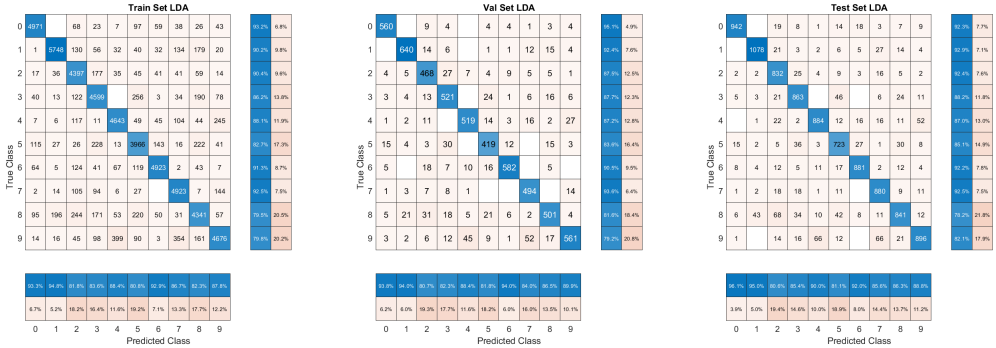

Figure 5: Figure 5: All Digits LDA

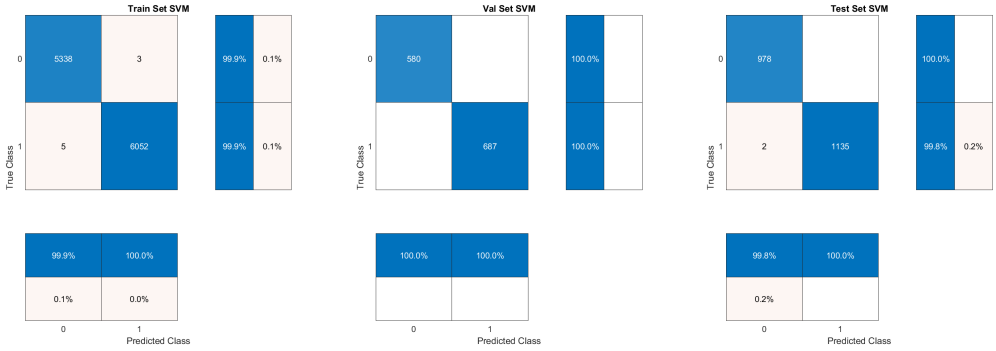

Figure 6: Figure 6: SVM on Easier 2 Digits

Figure 7: Figure 7: SVM on hardest 2 Digits

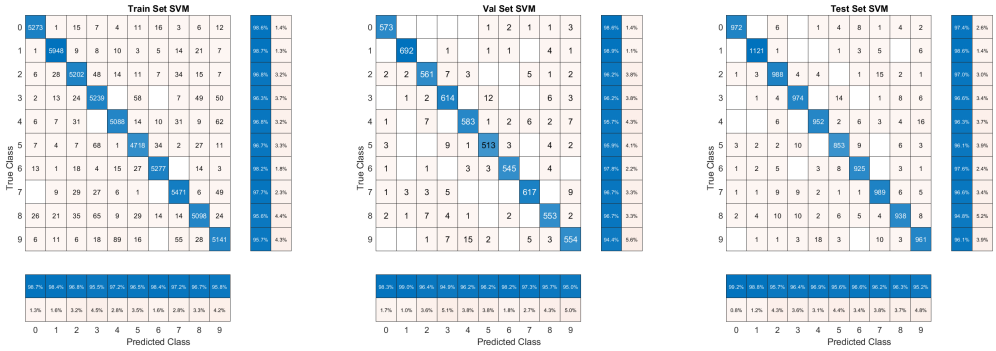

Figure 8: Figure 8: SVM on all 10 Digits

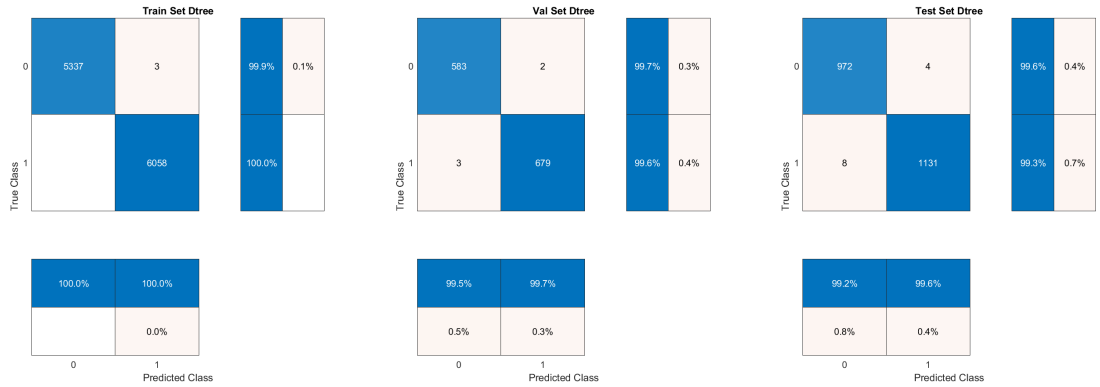

Figure 9: Figure 9: DTree on Easier 2 Digits

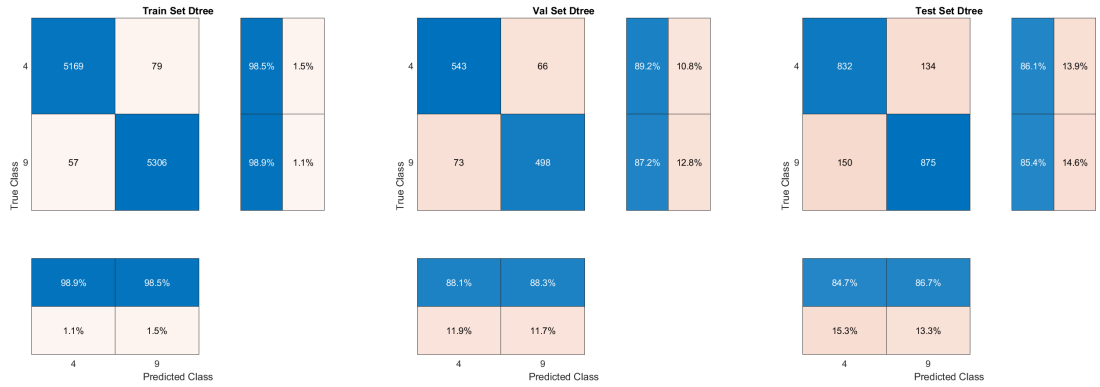

Figure 10: Figure 10: DTree on Hardest 2 Digits