# ECE174-MiniProject1

Jay Paek

November 2022

## 1 Problem 1

Given

$$\mathbf{A} \in \mathbb{C}^{m \times n}, \mathbf{y} \in \mathbb{C}^m$$

show that $\hat{\mathbf{x}}$ solves $min||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ if it is a solution to the following system of equations:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}$$

such that $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{C}^n$

### 1.1 Problem 1.1

The solution interpreted in terms of the problem implies that $||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ is minimized.
However, since

$$min||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 \geq 0 \quad \forall\, \mathbf{x}$$

the minimum possible value is 0, which can only be achieved when

$$\mathbf{y} - \mathbf{A}\mathbf{x} = 0$$

This further implies that a solution exists when the following can hold true:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \qquad \textbf{or} \qquad \mathbf{y} \in \mathfrak{R}(\mathbf{A})$$

But what if $\mathbf{y} \notin \mathfrak{R}(\mathbf{A})$?
Let $\mathbf{B} \in \mathbb{C}^{m \times r}$ be a matrix such that the r columns of $\mathbf{B}$ is the basis of $\mathfrak{R}(\mathbf{A})$ such that $\mathbf{dim}(\mathbf{A}) = \text{r} \leq \text{m}$
Hence,

$$\mathbf{B} = \begin{bmatrix} b_1 & b_2 & ... & b_r \end{bmatrix} \textit{ such that } \mathbf{span}\{b_1, b_2, ..., b_r\} = \mathfrak{R}(B) = \mathfrak{R}(A),\ b_1, b_2, ..., b_r \in \mathbb{C}^m$$

$\because \mathfrak{R}(A) = \mathfrak{R}(B)$, $\exists \mathbf{z} \in \mathbb{C}^r$ such that:

$$\mathbf{Ax} = \mathbf{Bz}$$

We can change the question to $min||\mathbf{y} - \mathbf{Bz}||_2^2$. If $\exists \mathbf{z}$ such that $\mathbf{z}$ minimizes $||\mathbf{y} - \mathbf{Bz}||_2^2$, then we can guarantee that $\mathbf{x}$ exists as well. Knowing that $||\mathbf{y} - \mathbf{Bz}||_2^2$ is minimized when

1. $\mathbf{y} - \mathbf{Bz}$ is orthogonal to every other vector in $\mathfrak{R}(A)$
2. a vector is orthogonal to every other vector when it is orthogonal to all the vectors of a basis of $\mathfrak{R}(A)$

the $\mathbf{z}$ that solves to $\mathbf{min}||\mathbf{y} - \mathbf{Bz}||_2^2$ must satisfy:

$$\mathbf{B}^T\mathbf{Bz} = \mathbf{B}^T\mathbf{y}$$

as proved in class.
Note that $\mathbf{B}^T\mathbf{y} \in \mathbb{R}^r$, so if $\mathfrak{R}(\mathbf{B}^T\mathbf{B}) = \mathbb{R}^r$, then $\mathbf{z}$ is guaranteed to exist.
We know that $\mathbf{B}^T\mathbf{B} \in \mathbb{R}^{r \times r}$, which means the matrix is r column vectors in $\mathbb{R}^r$.
$\mathfrak{R}(\mathbf{B}^T\mathbf{B}) = \mathbb{R}^r$ if and only if the columns of $\mathfrak{R}(\mathbf{B}^T\mathbf{B})$ are linearly independent.
Therefore, $\mathbb{N}(\mathbf{B}^T\mathbf{B}) = \{0\}$
Let $z_0 \in \mathbb{N}(\mathbf{B}^T\mathbf{B})$

$$\mathbf{B}^T\mathbf{B}z_0 = 0$$

$$z_0^T\mathbf{B}^T\mathbf{B}z_0 = 0$$

$$(\mathbf{B}z_0)^T\mathbf{B}z_0 = 0$$

$$||\mathbf{B}z_0||_2^2 = 0$$

Since the norm squared of a vector MUST be greater than or equal to 0.
$||\mathbf{B}z_0||_2^2 = 0$ only when $\mathbf{B}z_0 = 0$

$$\therefore \mathbb{N}(\mathbf{B}^T\mathbf{B}) = \{0\}$$

So, the only solution to this system is:

$$z_0 = 0$$

which proves that the columns of $\mathbf{B}^T\mathbf{B}$ are linearly independent and $\mathfrak{R}(\mathbf{B}^T\mathbf{B}) = \mathbb{R}^r$. Since $\mathbf{min}||\mathbf{y} - \mathbf{Bz}||_2^2$ has a solution $z_0$, $\exists \hat{\mathbf{x}}$ that minimizes $||\mathbf{y} - \mathbf{Ax}||_2^2$.

## 1.2 Problem 1.2

For the case where $\mathbf{y} \in \mathfrak{R}(\mathbf{A})$, $\exists \hat{\mathbf{x}}$ that minimizes $||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ when

$$\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$$

However, for the case where $\mathbf{y} \notin \mathfrak{R}(\mathbf{A})$, $\hat{\mathbf{x}}$ minimizes $||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ such that $\mathbf{A}\hat{\mathbf{x}} = \mathbf{B}\mathbf{z}$, where $\mathbf{B}$ is the basis of $\mathfrak{R}(\mathbf{A})$, such that $\mathbf{z}$ is the solution to

$$\mathbf{B}^T\mathbf{B}\mathbf{z} = \mathbf{B}^T\mathbf{y}$$

From 1.1, we know:

1. $\mathbf{B}^T\mathbf{B} \in \mathbb{R}^{r \times r}$ such that $r = \mathbf{rank}(A)$.
2. The columns are linearly independent.
3. $\mathfrak{R}(\mathbf{B}) = \mathbb{R}^r$

From this information, we can conclude that $\mathbf{B}^T\mathbf{B}$ is a basis of $\mathbb{R}^r$.
Let $\tilde{\mathbf{B}} = \mathbf{B}^T\mathbf{B}$ and $\tilde{\mathbf{y}} = \mathbf{B}^T\mathbf{y}$, then we can represent the equation as

$$\tilde{\mathbf{B}}\mathbf{z} = \tilde{\mathbf{y}}$$

We can tell that $\tilde{\mathbf{y}}$ is in the $\mathfrak{R}(\tilde{\mathbf{B}})$, and by the Unique Representation Theorem, there is only one such linear combination of the basis vectors to represent $\tilde{\mathbf{y}}$. This implies that only one such $\mathbf{z}$ exists that solves $\mathbf{B}^T\mathbf{B}\mathbf{z} = \mathbf{B}^T\mathbf{y}$. Since $\mathbf{z}$ is unique, then $\mathbf{B}\mathbf{z}$ is unique.
Let $\mathbf{B}\mathbf{z} = \hat{\mathbf{z}}$, then

$$\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{z}}$$

We know that $\hat{\mathbf{z}} \in \mathfrak{R}(\mathbf{A})$ since it is a linear combination of the bases of $\mathbf{A}$.
There are a total of four cases exists (I love using enumerate):

1. $\mathbf{y} \in \mathfrak{R}(\mathbf{A})$ and $\mathbb{N}(A) = \{0\}$ : One $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$
2. $\mathbf{y} \in \mathfrak{R}(\mathbf{A})$ and $\mathbb{N}(A) \neq \{0\}$ : Infinite $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$
3. $\mathbf{y} \notin \mathfrak{R}(\mathbf{A})$ and $\mathbb{N}(A) = \{0\}$ : One $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{z}}$
4. $\mathbf{y} \notin \mathfrak{R}(\mathbf{A})$ and $\mathbb{N}(A) \neq \{0\}$ : Infinite $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{z}}$

We can simplify this into two cases:

1. $\mathbb{N}(A) = \{0\}$ : One $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$
2. $\mathbb{N}(A) \neq \{0\}$ : Infinite $\hat{\mathbf{x}}$ exists where $\mathbf{A}\hat{\mathbf{x}} = \hat{\mathbf{z}}$

## 1.3 Problem 1.3

Given that $\mathbf{rank}(A) < n$, it implies that $\mathfrak{R}(A) \neq \mathbb{R}^n$

This does not immediately mean that $\mathbf{y} \notin \mathfrak{R}(A)$, it is merely that $\mathbf{y} \notin \mathfrak{R}(A)$ is not guaranteed.

In case 1, where $\mathbf{y} \in \mathfrak{R}(A)$, then the solution is $\hat{\mathbf{x}}$ that solves $\mathbf{A}\hat{\mathbf{x}} = \mathbf{y}$ (obviously).

Even if $\mathbf{y} \notin \mathfrak{R}(A)$, $min||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ can still be solved because we know $||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2$ is minimized when $\mathbf{y} - \mathbf{A}\mathbf{x}$ is orthogonal to every vector in $\mathfrak{R}(A)$, which is the same as being orthogonal to a basis of $\mathfrak{R}(A)$.

From 1.1, we represent $\mathbf{A}\mathbf{x}$ as a linear combination of a basis of $\mathbf{A}$

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{z}$$

and find that $\mathbf{z}$ is a solution to the following:

$$\mathbf{B}^T\mathbf{B}\mathbf{z} = \mathbf{B}^T\mathbf{y}$$

We already proved that a solution exists to the normal equation based on $\mathbf{B}$, the basis of $\mathbf{A}$, regardless of the $\mathbf{rank}(\mathbf{A})$. In order to solve for $\hat{\mathbf{x}}$, I will solve for $\mathbf{z}$ in the normal equation based on the basis.

After attaining $\mathbf{z}$, find $\hat{\mathbf{x}}$ such that $\mathbf{A}\hat{\mathbf{x}} = \mathbf{B}\mathbf{z}$. Then, $||\mathbf{y} - \mathbf{A}\hat{\mathbf{x}}||_2^2$ will be minimized.

However, $\mathbf{A}$ is not full rank, which implies that it is not a basis of $\mathfrak{R}(A)$.

Since the columns of $\mathbf{A}$ are not linearly independent, there is no unique representation to any vector $\mathfrak{R}(A)$.

Therefore, there will be infinite $\hat{\mathbf{x}}$ such that $||\mathbf{y} - \mathbf{A}\hat{\mathbf{x}}||_2^2$ is minimized. I just need to find one such $\hat{\mathbf{x}}$ where $\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{z}$ is satisfied.

However, in my program, I will use the pinv function from numpy in order to find the solution to the normal equation without finding the basis in order to solve for $\mathbf{x}$

# 2 Least-Square Multi-Class Classifier

## 2.1 One vs All Classifier

Figure 1 is the evaluation for the one versus all classifier on the training data. This table is for problem 3.

```
prediction_results = np.argmax(dot(train_in_aug, prediction_weights),axis=1)
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5682 | 7 | 18 | 14 | 24 | 43 | 64 | 4 | 61 | 6 | 5923 |
| 1 | 2 | 6548 | 40 | 15 | 19 | 31 | 14 | 12 | 55 | 6 | 6742 |
| 2 | 99 | 264 | 4792 | 149 | 108 | 11 | 234 | 91 | 192 | 18 | 5958 |
| 3 | 42 | 167 | 176 | 5158 | 32 | 125 | 56 | 115 | 135 | 125 | 6131 |
| 4 | 10 | 99 | 42 | 6 | 5212 | 50 | 39 | 23 | 59 | 302 | 5842 |
| 5 | 164 | 95 | 28 | 432 | 105 | 3991 | 192 | 36 | 235 | 143 | 5421 |
| 6 | 108 | 74 | 61 | 1 | 70 | 90 | 5476 | 0 | 35 | 3 | 5918 |
| 7 | 55 | 189 | 37 | 47 | 170 | 9 | 2 | 5426 | 10 | 320 | 6265 |
| 8 | 75 | 493 | 63 | 226 | 105 | 221 | 56 | 20 | 4412 | 180 | 5851 |
| 9 | 68 | 60 | 20 | 117 | 371 | 12 | 4 | 492 | 38 | 4767 | 5949 |
| totals | 6305 | 7996 | 5277 | 6165 | 6216 | 4583 | 6137 | 6219 | 5232 | 5870 | 60000 |

| | |
|---|---|
| Error Rate | 0.14226666666666665 |
| Accuracy for 0 | 0.9593111598851933 |
| Accuracy for 1 | 0.9712251557401365 |
| Accuracy for 2 | 0.8042967438737831 |
| Accuracy for 3 | 0.8412983200130484 |
| Accuracy for 4 | 0.8921602191030469 |
| Accuracy for 5 | 0.7362110311750599 |
| Accuracy for 6 | 0.9253126056100034 |
| Accuracy for 7 | 0.8660814046288907 |
| Accuracy for 8 | 0.7540591351905657 |
| Accuracy for 9 | 0.8013111447302068 |

Figure 1: One Versus All Classifier on Training Data

Figure 2 is the evaluation for the one versus all classifier on the testing data.

```
prediction_results = np.argmax(dot(test_in_aug, prediction_weights),axis=1)
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 944 | 0 | 1 | 2 | 2 | 7 | 14 | 2 | 7 | 1 | 980 |
| 1 | 0 | 1107 | 2 | 2 | 3 | 1 | 5 | 1 | 14 | 0 | 1135 |
| 2 | 18 | 54 | 813 | 26 | 15 | 0 | 42 | 22 | 37 | 5 | 1032 |
| 3 | 4 | 17 | 23 | 880 | 5 | 17 | 9 | 21 | 22 | 12 | 1010 |
| 4 | 0 | 22 | 6 | 1 | 881 | 5 | 10 | 2 | 11 | 44 | 982 |
| 5 | 23 | 18 | 3 | 72 | 24 | 659 | 23 | 14 | 39 | 17 | 892 |
| 6 | 18 | 10 | 9 | 0 | 22 | 17 | 875 | 0 | 7 | 0 | 958 |
| 7 | 5 | 40 | 16 | 6 | 26 | 0 | 1 | 884 | 0 | 50 | 1028 |
| 8 | 14 | 46 | 11 | 30 | 27 | 40 | 15 | 12 | 759 | 20 | 974 |
| 9 | 15 | 11 | 2 | 17 | 80 | 1 | 1 | 77 | 4 | 801 | 1009 |
| totals | 1041 | 1325 | 886 | 1036 | 1085 | 747 | 995 | 1035 | 900 | 950 | 10000 |

| Error Rate | 0.13970000000000005 |
|---|---|
| Accuracy for 0 | 0.963265306122449 |
| Accuracy for 1 | 0.9753303964757709 |
| Accuracy for 2 | 0.7877906976744187 |
| Accuracy for 3 | 0.8712871287128713 |
| Accuracy for 4 | 0.8971486761710794 |
| Accuracy for 5 | 0.7387892376681615 |
| Accuracy for 6 | 0.9133611691022965 |
| Accuracy for 7 | 0.8599221789883269 |
| Accuracy for 8 | 0.7792607802874744 |
| Accuracy for 9 | 0.7938553022794846 |

Figure 2: One Versus All Classifier on Testing Data

Some observations to note for these classifiers:

- 9 is often mistaken for 4 (80 instances) and also often mistaken for 7 (77 instances)
- 5 is often mistaken for 3 (72 instances)
- From the error rate table, 5 seems to have the lowest accuracy rate, possibly because it is always one small stroke away from being another number.
- 8 also seems to be a very hard number to identify. Possibly because of the number's structure complexity, being two loops in one image.
- 1 has the highest accuracy rate, probably because of its simplicity to detect, one straight vertical line.

## 2.2 One vs One Classifier

Figure 3 is the evaluation for the one versus one classifier on the training data. This table is for problem 3.

```
: prediction_results = collapse_results(sign(dot(train_in_aug, prediction_weights)))
analyze_multi(prediction_results, train_expected[0])
```

| Predicted --> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5806 | 2 | 15 | 8 | 11 | 19 | 22 | 6 | 33 | 1 | 5923 |
| 1 | 2 | 6623 | 36 | 17 | 7 | 16 | 2 | 11 | 21 | 7 | 6742 |
| 2 | 51 | 68 | 5521 | 49 | 57 | 21 | 42 | 44 | 92 | 13 | 5958 |
| 3 | 26 | 42 | 119 | 5579 | 9 | 161 | 18 | 48 | 90 | 39 | 6131 |
| 4 | 14 | 18 | 20 | 5 | 5586 | 11 | 14 | 16 | 8 | 150 | 5842 |
| 5 | 44 | 48 | 39 | 138 | 23 | 4967 | 93 | 10 | 46 | 13 | 5421 |
| 6 | 27 | 16 | 36 | 2 | 32 | 84 | 5690 | 0 | 30 | 1 | 5918 |
| 7 | 10 | 76 | 53 | 7 | 69 | 9 | 0 | 5881 | 5 | 155 | 6265 |
| 8 | 35 | 195 | 42 | 107 | 48 | 142 | 37 | 25 | 5155 | 65 | 5851 |
| 9 | 22 | 14 | 17 | 82 | 155 | 30 | 3 | 137 | 31 | 5458 | 5949 |
| totals | 6037 | 7102 | 5898 | 5994 | 5997 | 5460 | 5921 | 6178 | 5511 | 5902 | 60000 |

| Error Rate | 0.06223333333333336 |
|---|---|
| Accuracy for 0 | 0.9802464967077494 |
| Accuracy for 1 | 0.9823494512014239 |
| Accuracy for 2 | 0.9266532393420611 |
| Accuracy for 3 | 0.9099657478388518 |
| Accuracy for 4 | 0.9561793906196508 |
| Accuracy for 5 | 0.9162516140933408 |
| Accuracy for 6 | 0.9614734707671511 |
| Accuracy for 7 | 0.938707102952913 |
| Accuracy for 8 | 0.8810459750470006 |
| Accuracy for 9 | 0.917465120188267 |

Figure 3: One Versus One Classifier on Training Data

Figure 4 is the evaluation for the one versus one classifier on the testing data.

```
prediction_results = collapse_results(sign(dot(test_in_aug, prediction_weights)))
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 961 | 0 | 1 | 1 | 0 | 6 | 8 | 3 | 0 | 0 | 980 |
| 1 | 0 | 1120 | 3 | 3 | 1 | 1 | 4 | 1 | 2 | 0 | 1135 |
| 2 | 9 | 18 | 936 | 12 | 10 | 5 | 10 | 10 | 22 | 0 | 1032 |
| 3 | 9 | 1 | 18 | 926 | 2 | 20 | 1 | 7 | 21 | 5 | 1010 |
| 4 | 2 | 4 | 6 | 1 | 931 | 1 | 7 | 4 | 3 | 23 | 982 |
| 5 | 7 | 5 | 3 | 30 | 8 | 800 | 17 | 2 | 15 | 5 | 892 |
| 6 | 6 | 5 | 12 | 0 | 5 | 19 | 908 | 1 | 2 | 0 | 958 |
| 7 | 1 | 16 | 17 | 3 | 11 | 1 | 0 | 955 | 1 | 23 | 1028 |
| 8 | 7 | 17 | 8 | 23 | 10 | 36 | 10 | 10 | 840 | 13 | 974 |
| 9 | 6 | 5 | 1 | 11 | 30 | 12 | 0 | 21 | 3 | 920 | 1009 |
| totals | 1008 | 1191 | 1005 | 1010 | 1008 | 901 | 965 | 1014 | 909 | 989 | 10000 |

| Error Rate | 0.07030000000000003 |
|---|---|
| Accuracy for 0 | 0.9806122448979592 |
| Accuracy for 1 | 0.986784140969163 |
| Accuracy for 2 | 0.9069767441860466 |
| Accuracy for 3 | 0.9168316831683169 |
| Accuracy for 4 | 0.9480651731160896 |
| Accuracy for 5 | 0.8968609865470852 |
| Accuracy for 6 | 0.9478079331941545 |
| Accuracy for 7 | 0.9289883268482491 |
| Accuracy for 8 | 0.8624229979466119 |
| Accuracy for 9 | 0.9117938553022795 |

Figure 4: One Versus One Classifier on Testing Data

Some observations denotes similar patterns as the one versus all classifier.

- Error rate is much better then One v All. Basically halved the rate.
- 5 is often mistaken for 3 (30 instances), and 9 is often mistaken for 4 (30 instances). We saw this pattern in the one versus all classifier.
- Although error and accuracy rates improved, error rate trends remained. 5 and 8 still have the lowest accuracy, while 1 and 0 has the highest.
- The One versus One classifier has significantly better performance in exchange for more computing power and time. However, these inconveniences are very small.

# 3 Problem 3

## 3.1 Randomized Feature Spaces

All randomized feature spaces have the same L-dimensional randomizing matrix and same additive random bias.

### 3.1.1 Identity Function

Figure 5 depicts the confusion matrix and error rates for the One versus All classifier evaluated on training data.

```
prediction_results = np.argmax(dot(train_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5682 | 7 | 18 | 14 | 24 | 43 | 64 | 4 | 61 | 6 | 5923 |
| 1 | 2 | 6548 | 40 | 15 | 19 | 31 | 14 | 12 | 55 | 6 | 6742 |
| 2 | 99 | 264 | 4792 | 149 | 108 | 11 | 234 | 91 | 192 | 18 | 5958 |
| 3 | 42 | 167 | 176 | 5158 | 32 | 125 | 56 | 115 | 135 | 125 | 6131 |
| 4 | 10 | 99 | 42 | 6 | 5212 | 50 | 39 | 23 | 59 | 302 | 5842 |
| 5 | 164 | 95 | 28 | 432 | 105 | 3991 | 192 | 36 | 235 | 143 | 5421 |
| 6 | 108 | 74 | 61 | 1 | 70 | 90 | 5476 | 0 | 35 | 3 | 5918 |
| 7 | 55 | 189 | 37 | 47 | 170 | 9 | 2 | 5426 | 10 | 320 | 6265 |
| 8 | 75 | 493 | 63 | 226 | 105 | 221 | 56 | 20 | 4412 | 180 | 5851 |
| 9 | 68 | 60 | 20 | 117 | 371 | 12 | 4 | 492 | 38 | 4767 | 5949 |
| totals | 6305 | 7996 | 5277 | 6165 | 6216 | 4583 | 6137 | 6219 | 5232 | 5870 | 60000 |

| | |
|---|---|
| Error Rate | 0.14226666666666665 |
| Accuracy for 0 | 0.9593111598851933 |
| Accuracy for 1 | 0.9712251557401365 |
| Accuracy for 2 | 0.8042967438737831 |
| Accuracy for 3 | 0.8412983200130484 |
| Accuracy for 4 | 0.8921602191030469 |
| Accuracy for 5 | 0.7362110311750599 |
| Accuracy for 6 | 0.9253126056100034 |
| Accuracy for 7 | 0.8660814046288907 |
| Accuracy for 8 | 0.7540591351905657 |
| Accuracy for 9 | 0.8013111447302068 |

Figure 5: One versus All Classifier with Identity Function on Training Data

Figure 6 depicts the confusion matrix and error rates for the One versus All classifier evaluated on testing data.

```
prediction_results = np.argmax(dot(test_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 944 | 0 | 1 | 2 | 2 | 7 | 14 | 2 | 7 | 1 | 980 |
| 1 | 0 | 1107 | 2 | 2 | 3 | 1 | 5 | 1 | 14 | 0 | 1135 |
| 2 | 18 | 54 | 813 | 26 | 15 | 0 | 42 | 22 | 37 | 5 | 1032 |
| 3 | 4 | 17 | 23 | 880 | 5 | 17 | 9 | 21 | 22 | 12 | 1010 |
| 4 | 0 | 22 | 6 | 1 | 881 | 5 | 10 | 2 | 11 | 44 | 982 |
| 5 | 23 | 18 | 3 | 72 | 24 | 659 | 23 | 14 | 39 | 17 | 892 |
| 6 | 18 | 10 | 9 | 0 | 22 | 17 | 875 | 0 | 7 | 0 | 958 |
| 7 | 5 | 40 | 16 | 6 | 26 | 0 | 1 | 884 | 0 | 50 | 1028 |
| 8 | 14 | 46 | 11 | 30 | 27 | 40 | 15 | 12 | 759 | 20 | 974 |
| 9 | 15 | 11 | 2 | 17 | 80 | 1 | 1 | 77 | 4 | 801 | 1009 |
| totals | 1041 | 1325 | 886 | 1036 | 1085 | 747 | 995 | 1035 | 900 | 950 | 10000 |

| | |
|---|---|
| Error Rate | 0.13970000000000005 |
| Accuracy for 0 | 0.963265306122449 |
| Accuracy for 1 | 0.9753303964757709 |
| Accuracy for 2 | 0.7877906976744187 |
| Accuracy for 3 | 0.8712871287128713 |
| Accuracy for 4 | 0.8971486761710794 |
| Accuracy for 5 | 0.7387892376681615 |
| Accuracy for 6 | 0.9133611691022965 |
| Accuracy for 7 | 0.8599221789883269 |
| Accuracy for 8 | 0.7792607802874744 |
| Accuracy for 9 | 0.7938553022794846 |

Figure 6: One versus All Classifier with Identity Function on Testing Data

Figure 7 depicts the confusion matrix and error rates for the One versus One classifier evaluated on training data.

```
predictions = dot(train_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5806 | 2 | 15 | 8 | 11 | 19 | 22 | 6 | 33 | 1 | 5923 |
| 1 | 2 | 6623 | 36 | 17 | 7 | 16 | 2 | 11 | 21 | 7 | 6742 |
| 2 | 51 | 68 | 5522 | 48 | 57 | 21 | 42 | 44 | 92 | 13 | 5958 |
| 3 | 25 | 42 | 119 | 5578 | 9 | 163 | 18 | 48 | 90 | 39 | 6131 |
| 4 | 14 | 18 | 20 | 5 | 5586 | 11 | 14 | 16 | 8 | 150 | 5842 |
| 5 | 44 | 47 | 39 | 137 | 23 | 4970 | 93 | 10 | 45 | 13 | 5421 |
| 6 | 27 | 16 | 36 | 2 | 31 | 83 | 5692 | 0 | 30 | 1 | 5918 |
| 7 | 9 | 75 | 53 | 8 | 66 | 9 | 0 | 5886 | 5 | 154 | 6265 |
| 8 | 35 | 194 | 42 | 108 | 48 | 142 | 37 | 25 | 5155 | 65 | 5851 |
| 9 | 22 | 14 | 17 | 82 | 156 | 31 | 3 | 137 | 30 | 5457 | 5949 |
| totals | 6035 | 7099 | 5899 | 5993 | 5994 | 5465 | 5923 | 6183 | 5509 | 5900 | 60000 |

| | |
|---|---|
| Error Rate | 0.06208333333333338 |
| Accuracy for 0 | 0.9802464967077494 |
| Accuracy for 1 | 0.9823494512014239 |
| Accuracy for 2 | 0.9268210808996308 |
| Accuracy for 3 | 0.9098026423095743 |
| Accuracy for 4 | 0.9561793906196508 |
| Accuracy for 5 | 0.916805017524442 |
| Accuracy for 6 | 0.9618114227779655 |
| Accuracy for 7 | 0.9395051875498803 |
| Accuracy for 8 | 0.8810459750470006 |
| Accuracy for 9 | 0.9172970247100353 |

Figure 7: One versus One Classifier with Identity Function on Training Data

Figure 8 depicts the confusion matrix and error rates for the One versus One classifier evaluated on testing data.

```
predictions = dot(test_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 961 | 0 | 1 | 1 | 0 | 6 | 8 | 3 | 0 | 0 | 980 |
| 1 | 0 | 1118 | 4 | 2 | 1 | 1 | 5 | 1 | 3 | 0 | 1135 |
| 2 | 9 | 18 | 937 | 13 | 10 | 4 | 10 | 9 | 22 | 0 | 1032 |
| 3 | 9 | 2 | 18 | 926 | 2 | 19 | 1 | 7 | 21 | 5 | 1010 |
| 4 | 4 | 2 | 7 | 2 | 930 | 1 | 7 | 4 | 3 | 22 | 982 |
| 5 | 6 | 5 | 3 | 30 | 8 | 798 | 18 | 2 | 15 | 7 | 892 |
| 6 | 7 | 5 | 12 | 0 | 5 | 20 | 907 | 0 | 2 | 0 | 958 |
| 7 | 1 | 16 | 18 | 3 | 8 | 1 | 0 | 957 | 1 | 23 | 1028 |
| 8 | 7 | 16 | 9 | 22 | 9 | 36 | 11 | 10 | 841 | 13 | 974 |
| 9 | 6 | 5 | 1 | 11 | 28 | 12 | 0 | 22 | 3 | 921 | 1009 |
| totals | 1010 | 1187 | 1010 | 1010 | 1001 | 898 | 967 | 1015 | 911 | 991 | 10000 |

| Error Rate | 0.07040000000000002 |
|---|---|
| Accuracy for 0 | 0.9806122448979592 |
| Accuracy for 1 | 0.9850220264317181 |
| Accuracy for 2 | 0.9079457364341086 |
| Accuracy for 3 | 0.9168316831683169 |
| Accuracy for 4 | 0.9470468431771895 |
| Accuracy for 5 | 0.8946188340807175 |
| Accuracy for 6 | 0.9467640918580376 |
| Accuracy for 7 | 0.9309338521400778 |
| Accuracy for 8 | 0.8634496919917864 |
| Accuracy for 9 | 0.9127849355797819 |

Figure 8: One versus One Classifier with Identity Function on Testing Data

Some observations to note:

- The confusion matrices and error rates for the One versus All classifiers are dead similar.
- The one versus one classifier performed better on the training data, but a little worse on the testing data. However, the performances are still very similar.
- False positive and false negative patterns from the 'clean data' classifier appear in the randomized feature classifier as well.

### 3.1.2 Sigmoid Function

Figure 9 depicts the confusion matrix and error rates for the One versus All classifier with a sigmoid feature function evaluated on training data.

```
prediction_results = np.argmax(dot(train_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, train_expected[0])
```

| Predicted --> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5763 | 2 | 10 | 13 | 8 | 24 | 42 | 3 | 51 | 7 | 5923 |
| 1 | 1 | 6620 | 44 | 9 | 16 | 6 | 8 | 12 | 23 | 3 | 6742 |
| 2 | 35 | 76 | 5455 | 59 | 71 | 7 | 44 | 91 | 109 | 11 | 5958 |
| 3 | 20 | 50 | 87 | 5583 | 10 | 112 | 21 | 67 | 100 | 81 | 6131 |
| 4 | 8 | 50 | 21 | 2 | 5499 | 7 | 37 | 14 | 28 | 176 | 5842 |
| 5 | 60 | 58 | 17 | 133 | 42 | 4860 | 111 | 27 | 68 | 45 | 5421 |
| 6 | 41 | 22 | 26 | 2 | 31 | 76 | 5689 | 1 | 30 | 0 | 5918 |
| 7 | 24 | 98 | 46 | 15 | 104 | 8 | 2 | 5801 | 13 | 154 | 6265 |
| 8 | 24 | 125 | 58 | 97 | 46 | 125 | 46 | 17 | 5226 | 87 | 5851 |
| 9 | 31 | 30 | 13 | 80 | 169 | 25 | 6 | 155 | 51 | 5389 | 5949 |
| totals | 6007 | 7131 | 5777 | 5993 | 5996 | 5250 | 6006 | 6188 | 5699 | 5953 | 60000 |

| | |
|---|---|
| Error Rate | 0.06858333333333333 |
| Accuracy for 0 | 0.9729866621644437 |
| Accuracy for 1 | 0.9819044793829724 |
| Accuracy for 2 | 0.915575696542464 |
| Accuracy for 3 | 0.9106181699559615 |
| Accuracy for 4 | 0.9412872304005477 |
| Accuracy for 5 | 0.896513558384062 |
| Accuracy for 6 | 0.9613044947617438 |
| Accuracy for 7 | 0.9259377494014366 |
| Accuracy for 8 | 0.8931806528798496 |
| Accuracy for 9 | 0.9058665321902841 |

Figure 9: One versus All Classifier with Sigmoid Function on Training Data

Figure 10 depicts the confusion matrix and error rates for the One versus All classifier with a sigmoid feature function evaluated on testing data.

```
prediction_results = np.argmax(dot(test_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 962 | 0 | 1 | 0 | 0 | 2 | 10 | 1 | 4 | 0 | 980 |
| 1 | 0 | 1122 | 2 | 2 | 1 | 1 | 4 | 1 | 2 | 0 | 1135 |
| 2 | 6 | 12 | 924 | 16 | 12 | 2 | 11 | 18 | 31 | 0 | 1032 |
| 3 | 2 | 4 | 10 | 938 | 2 | 15 | 4 | 18 | 13 | 4 | 1010 |
| 4 | 2 | 9 | 4 | 0 | 914 | 1 | 8 | 2 | 6 | 36 | 982 |
| 5 | 10 | 4 | 0 | 17 | 11 | 807 | 10 | 9 | 16 | 8 | 892 |
| 6 | 14 | 5 | 3 | 2 | 8 | 13 | 913 | 0 | 0 | 0 | 958 |
| 7 | 3 | 26 | 15 | 4 | 8 | 3 | 1 | 932 | 3 | 33 | 1028 |
| 8 | 9 | 9 | 11 | 13 | 11 | 23 | 10 | 5 | 872 | 11 | 974 |
| 9 | 7 | 7 | 1 | 12 | 33 | 9 | 0 | 17 | 13 | 910 | 1009 |
| totals | 1015 | 1198 | 971 | 1004 | 1000 | 876 | 971 | 1003 | 960 | 1002 | 10000 |

| Error Rate | 0.0706 |
|---|---|
| Accuracy for 0 | 0.9816326530612245 |
| Accuracy for 1 | 0.9885462555066079 |
| Accuracy for 2 | 0.8953488372093024 |
| Accuracy for 3 | 0.9287128712871288 |
| Accuracy for 4 | 0.9307535641547862 |
| Accuracy for 5 | 0.9047085201793722 |
| Accuracy for 6 | 0.9530271398747391 |
| Accuracy for 7 | 0.9066147859922179 |
| Accuracy for 8 | 0.8952772073921971 |
| Accuracy for 9 | 0.9018830525272548 |

Figure 10: One versus All Classifier with Sigmoid Function on Testing Data

Already, we can see that the feature function significantly helps improve the accuracy and precision of the accuracy.

Figure 11 depicts the confusion matrix and error rates for the One versus One classifier with a sigmoid feature function evaluated on training data.

```
predictions = dot(train_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5876 | 2 | 5 | 2 | 6 | 6 | 8 | 0 | 17 | 1 | 5923 |
| 1 | 2 | 6680 | 21 | 9 | 11 | 0 | 1 | 9 | 4 | 5 | 6742 |
| 2 | 19 | 13 | 5793 | 20 | 26 | 4 | 15 | 25 | 40 | 3 | 5958 |
| 3 | 5 | 12 | 67 | 5874 | 3 | 68 | 5 | 30 | 40 | 27 | 6131 |
| 4 | 5 | 8 | 14 | 0 | 5724 | 1 | 15 | 7 | 7 | 61 | 5842 |
| 5 | 16 | 9 | 9 | 55 | 10 | 5258 | 37 | 4 | 13 | 10 | 5421 |
| 6 | 16 | 8 | 9 | 0 | 6 | 36 | 5831 | 0 | 12 | 0 | 5918 |
| 7 | 3 | 26 | 33 | 7 | 30 | 2 | 1 | 6103 | 6 | 54 | 6265 |
| 8 | 7 | 33 | 32 | 33 | 12 | 35 | 19 | 5 | 5649 | 26 | 5851 |
| 9 | 12 | 10 | 7 | 40 | 57 | 15 | 1 | 48 | 30 | 5729 | 5949 |
| totals | 5961 | 6801 | 5990 | 6040 | 5885 | 5425 | 5933 | 6231 | 5818 | 5916 | 60000 |

| Error Rate | 0.02471666666666672 |
|---|---|
| Accuracy for 0 | 0.9920648320108053 |
| Accuracy for 1 | 0.9908039157520023 |
| Accuracy for 2 | 0.972306143001007 |
| Accuracy for 3 | 0.9580818789756973 |
| Accuracy for 4 | 0.9798014378637453 |
| Accuracy for 5 | 0.9699317469101641 |
| Accuracy for 6 | 0.9852990875295708 |
| Accuracy for 7 | 0.9741420590582601 |
| Accuracy for 8 | 0.9654759870107674 |
| Accuracy for 9 | 0.9630189947890402 |

Figure 11: One versus One Classifier with Sigmoid Function on Training Data

Figure 12 depicts the confusion matrix and error rates for the One versus One classifier with a sigmoid feature function evaluated on testing data.

```
predictions = dot(test_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 968 | 0 | 2 | 1 | 1 | 3 | 2 | 1 | 2 | 0 | 980 |
| 1 | 0 | 1125 | 3 | 1 | 0 | 1 | 1 | 1 | 3 | 0 | 1135 |
| 2 | 7 | 1 | 989 | 5 | 8 | 0 | 5 | 7 | 9 | 1 | 1032 |
| 3 | 2 | 1 | 10 | 969 | 1 | 8 | 1 | 4 | 11 | 3 | 1010 |
| 4 | 2 | 0 | 10 | 2 | 946 | 1 | 3 | 2 | 3 | 13 | 982 |
| 5 | 6 | 4 | 5 | 16 | 4 | 839 | 4 | 3 | 7 | 4 | 892 |
| 6 | 8 | 3 | 2 | 0 | 5 | 11 | 925 | 0 | 4 | 0 | 958 |
| 7 | 1 | 10 | 19 | 2 | 5 | 0 | 0 | 977 | 2 | 12 | 1028 |
| 8 | 4 | 1 | 9 | 9 | 5 | 8 | 5 | 4 | 925 | 4 | 974 |
| 9 | 5 | 4 | 6 | 9 | 16 | 3 | 0 | 7 | 3 | 956 | 1009 |
| totals | 1003 | 1149 | 1055 | 1014 | 991 | 874 | 946 | 1006 | 969 | 993 | 10000 |

| | |
|---|---|
| Error Rate | 0.03810000000000002 |
| Accuracy for 0 | 0.9877551020408163 |
| Accuracy for 1 | 0.9911894273127754 |
| Accuracy for 2 | 0.9583333333333334 |
| Accuracy for 3 | 0.9594059405940594 |
| Accuracy for 4 | 0.9633401221995926 |
| Accuracy for 5 | 0.9405829596412556 |
| Accuracy for 6 | 0.965553235908142 |
| Accuracy for 7 | 0.9503891050583657 |
| Accuracy for 8 | 0.9496919917864476 |
| Accuracy for 9 | 0.9474727452923687 |

Figure 12: One versus One Classifier with Sigmoid Function on Testing Data

The sigmoid feature function did not disappoint. It halved the error rates on all data sets across both classifiers. Here are some key observations:

- 9 continues to be mistaken for 4 and 7, and vice versa.
- 5 continues to be mistaken for 3. Trends from the basic multi-class classifiers continues to emerge in the randomized feature space classifiers.
- 5 and 8 continue to have the lowest accuracy rates while 1 has the highest.
- Despite being the most accurate, it seems to struggle identifying 7, in which is determines it as a 2.

### 3.1.3 Sinusoidal Function

Figure 13 depicts the confusion matrix and error rates for the One versus All classifier with a sinusoidal feature function evaluated on training data.

```
prediction_results = np.argmax(dot(train_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, train_expected[0])
```

| Predicted --> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5763 | 2 | 10 | 13 | 8 | 24 | 42 | 3 | 51 | 7 | 5923 |
| 1 | 1 | 6620 | 44 | 9 | 16 | 6 | 8 | 12 | 23 | 3 | 6742 |
| 2 | 35 | 76 | 5455 | 59 | 71 | 7 | 44 | 91 | 109 | 11 | 5958 |
| 3 | 20 | 50 | 87 | 5583 | 10 | 112 | 21 | 67 | 100 | 81 | 6131 |
| 4 | 8 | 50 | 21 | 2 | 5499 | 7 | 37 | 14 | 28 | 176 | 5842 |
| 5 | 60 | 58 | 17 | 133 | 42 | 4860 | 111 | 27 | 68 | 45 | 5421 |
| 6 | 41 | 22 | 26 | 2 | 31 | 76 | 5689 | 1 | 30 | 0 | 5918 |
| 7 | 24 | 98 | 46 | 15 | 104 | 8 | 2 | 5801 | 13 | 154 | 6265 |
| 8 | 24 | 125 | 58 | 97 | 46 | 125 | 46 | 17 | 5226 | 87 | 5851 |
| 9 | 31 | 30 | 13 | 80 | 169 | 25 | 6 | 155 | 51 | 5389 | 5949 |
| totals | 6007 | 7131 | 5777 | 5993 | 5996 | 5250 | 6006 | 6188 | 5699 | 5953 | 60000 |

| | |
|---|---|
| Error Rate | 0.06858333333333333 |
| Accuracy for 0 | 0.9729866621644437 |
| Accuracy for 1 | 0.9819044793829724 |
| Accuracy for 2 | 0.915575696542464 |
| Accuracy for 3 | 0.9106181699559615 |
| Accuracy for 4 | 0.9412872304005477 |
| Accuracy for 5 | 0.896513558384062 |
| Accuracy for 6 | 0.9613044947617438 |
| Accuracy for 7 | 0.9259377494014366 |
| Accuracy for 8 | 0.8931806528798496 |
| Accuracy for 9 | 0.9058665321902841 |

Figure 13: One versus All Classifier with Sinusoidal Function on Training Data

Figure 14 depicts the confusion matrix and error rates for the One versus All classifier with a sinusoidal feature function evaluated on testing data.

```
prediction_results = np.argmax(dot(test_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 962 | 0 | 1 | 0 | 0 | 2 | 10 | 1 | 4 | 0 | 980 |
| 1 | 0 | 1122 | 2 | 2 | 1 | 1 | 4 | 1 | 2 | 0 | 1135 |
| 2 | 6 | 12 | 924 | 16 | 12 | 2 | 11 | 18 | 31 | 0 | 1032 |
| 3 | 2 | 4 | 10 | 938 | 2 | 15 | 4 | 18 | 13 | 4 | 1010 |
| 4 | 2 | 9 | 4 | 0 | 914 | 1 | 8 | 2 | 6 | 36 | 982 |
| 5 | 10 | 4 | 0 | 17 | 11 | 807 | 10 | 9 | 16 | 8 | 892 |
| 6 | 14 | 5 | 3 | 2 | 8 | 13 | 913 | 0 | 0 | 0 | 958 |
| 7 | 3 | 26 | 15 | 4 | 8 | 3 | 1 | 932 | 3 | 33 | 1028 |
| 8 | 9 | 9 | 11 | 13 | 11 | 23 | 10 | 5 | 872 | 11 | 974 |
| 9 | 7 | 7 | 1 | 12 | 33 | 9 | 0 | 17 | 13 | 910 | 1009 |
| totals | 1015 | 1198 | 971 | 1004 | 1000 | 876 | 971 | 1003 | 960 | 1002 | 10000 |

| Error Rate | 0.0706 |
|---|---|
| Accuracy for 0 | 0.9816326530612245 |
| Accuracy for 1 | 0.9885462555066079 |
| Accuracy for 2 | 0.8953488372093024 |
| Accuracy for 3 | 0.9287128712871288 |
| Accuracy for 4 | 0.9307535641547862 |
| Accuracy for 5 | 0.9047085201793722 |
| Accuracy for 6 | 0.9530271398747391 |
| Accuracy for 7 | 0.9066147859922179 |
| Accuracy for 8 | 0.8952772073921971 |
| Accuracy for 9 | 0.9018830525272548 |

Figure 14: One versus All Classifier with Sinusoidal Function on Testing Data

Figure 15 depicts the confusion matrix and error rates for the One versus One classifier with a sinusoidal feature function evaluated on training data.

```
predictions = dot(train_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5874 | 2 | 6 | 6 | 3 | 5 | 9 | 1 | 15 | 2 | 5923 |
| 1 | 1 | 6678 | 23 | 7 | 9 | 0 | 1 | 9 | 8 | 6 | 6742 |
| 2 | 24 | 15 | 5796 | 20 | 28 | 4 | 12 | 24 | 32 | 3 | 5958 |
| 3 | 8 | 8 | 59 | 5901 | 2 | 58 | 5 | 23 | 45 | 22 | 6131 |
| 4 | 4 | 6 | 11 | 0 | 5728 | 1 | 10 | 12 | 7 | 63 | 5842 |
| 5 | 22 | 4 | 10 | 61 | 7 | 5245 | 37 | 3 | 20 | 12 | 5421 |
| 6 | 14 | 6 | 9 | 1 | 11 | 34 | 5834 | 0 | 9 | 0 | 5918 |
| 7 | 3 | 23 | 30 | 3 | 28 | 2 | 1 | 6110 | 5 | 60 | 6265 |
| 8 | 9 | 27 | 23 | 40 | 14 | 42 | 18 | 10 | 5649 | 19 | 5851 |
| 9 | 14 | 7 | 6 | 38 | 70 | 19 | 1 | 52 | 18 | 5724 | 5949 |
| totals | 5973 | 6776 | 5973 | 6077 | 5900 | 5410 | 5928 | 6244 | 5808 | 5911 | 60000 |

| Error Rate | 0.024349999999999983 |
|---|---|
| Accuracy for 0 | 0.9917271652878609 |
| Accuracy for 1 | 0.9905072678730347 |
| Accuracy for 2 | 0.972809667673716 |
| Accuracy for 3 | 0.9624857282661883 |
| Accuracy for 4 | 0.9804861348853132 |
| Accuracy for 5 | 0.967533665375392 |
| Accuracy for 6 | 0.9858060155457925 |
| Accuracy for 7 | 0.9752593774940144 |
| Accuracy for 8 | 0.9654759870107674 |
| Accuracy for 9 | 0.962178517397882 |

Figure 15: One versus One Classifier with Sinusoidal Function on Training Data

Figure 16 depicts the confusion matrix and error rates for the One versus One classifier with a sinusoidal feature function evaluated on testing data. The sinusoidal feature function shows similar trends to the

```
predictions = dot(test_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 969 | 0 | 2 | 0 | 0 | 3 | 3 | 1 | 2 | 0 | 980 |
| 1 | 0 | 1125 | 4 | 2 | 0 | 1 | 1 | 0 | 2 | 0 | 1135 |
| 2 | 7 | 0 | 999 | 7 | 3 | 1 | 1 | 9 | 5 | 0 | 1032 |
| 3 | 2 | 0 | 5 | 980 | 1 | 6 | 0 | 6 | 7 | 3 | 1010 |
| 4 | 4 | 0 | 3 | 0 | 950 | 0 | 4 | 2 | 2 | 17 | 982 |
| 5 | 3 | 0 | 2 | 9 | 1 | 861 | 7 | 1 | 6 | 2 | 892 |
| 6 | 5 | 2 | 2 | 0 | 2 | 3 | 939 | 0 | 5 | 0 | 958 |
| 7 | 1 | 6 | 16 | 1 | 3 | 0 | 0 | 989 | 0 | 12 | 1028 |
| 8 | 6 | 3 | 2 | 6 | 4 | 2 | 2 | 4 | 939 | 6 | 974 |
| 9 | 7 | 4 | 1 | 9 | 11 | 2 | 0 | 7 | 4 | 964 | 1009 |
| totals | 1004 | 1140 | 1036 | 1014 | 975 | 879 | 957 | 1019 | 972 | 1004 | 10000 |

| | |
|---|---|
| Error Rate | 0.02849999999999997 |
| Accuracy for 0 | 0.9887755102040816 |
| Accuracy for 1 | 0.9911894273127754 |
| Accuracy for 2 | 0.9680232558139534 |
| Accuracy for 3 | 0.9702970297029703 |
| Accuracy for 4 | 0.9674134419551935 |
| Accuracy for 5 | 0.9652466367713004 |
| Accuracy for 6 | 0.9801670146137788 |
| Accuracy for 7 | 0.9620622568093385 |
| Accuracy for 8 | 0.9640657084188912 |
| Accuracy for 9 | 0.9554013875123885 |

Figure 16: One versus One Classifier with Sinusoidal Function on Testing Data

identity function and the sigmoid function. However, it showed the lowest error rates i.e. best performance in both classifiers across all data sets.

- The 3, 5; 4, 9; 2, 7; and 7, 9 pair has more false positives.
- No new trends, but this classifier perform the best out of the functions.

### 3.1.4 ReLU Function

Figure 17 depicts the confusion matrix and error rates for the One versus All classifier with a ReLU feature function evaluated on training data.

```
prediction_results = np.argmax(dot(train_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5680 | 3 | 19 | 20 | 23 | 41 | 62 | 3 | 67 | 5 | 5923 |
| 1 | 2 | 6569 | 41 | 8 | 17 | 24 | 15 | 11 | 49 | 6 | 6742 |
| 2 | 91 | 187 | 4926 | 134 | 109 | 13 | 185 | 96 | 184 | 33 | 5958 |
| 3 | 48 | 123 | 161 | 5203 | 26 | 152 | 42 | 104 | 139 | 133 | 6131 |
| 4 | 13 | 95 | 41 | 6 | 5277 | 38 | 41 | 17 | 47 | 267 | 5842 |
| 5 | 132 | 61 | 27 | 354 | 96 | 4247 | 163 | 36 | 183 | 122 | 5421 |
| 6 | 77 | 51 | 54 | 5 | 66 | 110 | 5512 | 1 | 39 | 3 | 5918 |
| 7 | 48 | 161 | 35 | 33 | 148 | 10 | 4 | 5533 | 17 | 276 | 6265 |
| 8 | 59 | 381 | 60 | 197 | 110 | 188 | 66 | 20 | 4604 | 166 | 5851 |
| 9 | 52 | 52 | 22 | 105 | 345 | 20 | 3 | 387 | 46 | 4917 | 5949 |
| totals | 6202 | 7683 | 5386 | 6065 | 6217 | 4843 | 6093 | 6208 | 5375 | 5928 | 60000 |

| Error Rate | 0.12553333333333339 |
|---|---|
| Accuracy for 0 | 0.9589734931622489 |
| Accuracy for 1 | 0.974339958469297 |
| Accuracy for 2 | 0.8267875125881168 |
| Accuracy for 3 | 0.8486380688305334 |
| Accuracy for 4 | 0.9032865457035262 |
| Accuracy for 5 | 0.7834347906290352 |
| Accuracy for 6 | 0.9313957418046638 |
| Accuracy for 7 | 0.8831604150039905 |
| Accuracy for 8 | 0.7868740386258759 |
| Accuracy for 9 | 0.8265254664649521 |

Figure 17: One versus All Classifier with Sinusoidal Function on Training Data

Figure 18 depicts the confusion matrix and error rates for the One versus All classifier with a ReLU feature function evaluated on testing data.

```
prediction_results = np.argmax(dot(test_in_feature, prediction_weights),axis=1)
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 945 | 1 | 1 | 2 | 0 | 8 | 13 | 2 | 7 | 1 | 980 |
| 1 | 0 | 1104 | 2 | 2 | 4 | 2 | 4 | 0 | 17 | 0 | 1135 |
| 2 | 13 | 42 | 828 | 33 | 16 | 0 | 30 | 20 | 44 | 6 | 1032 |
| 3 | 6 | 18 | 15 | 886 | 5 | 22 | 8 | 21 | 23 | 6 | 1010 |
| 4 | 1 | 17 | 5 | 0 | 891 | 2 | 12 | 1 | 7 | 46 | 982 |
| 5 | 21 | 12 | 7 | 58 | 17 | 689 | 23 | 12 | 37 | 16 | 892 |
| 6 | 16 | 10 | 10 | 0 | 23 | 16 | 875 | 0 | 8 | 0 | 958 |
| 7 | 4 | 36 | 18 | 7 | 15 | 1 | 1 | 893 | 0 | 53 | 1028 |
| 8 | 15 | 46 | 7 | 25 | 23 | 34 | 12 | 11 | 783 | 18 | 974 |
| 9 | 17 | 12 | 3 | 16 | 64 | 2 | 1 | 42 | 7 | 845 | 1009 |
| totals | 1038 | 1298 | 896 | 1029 | 1058 | 776 | 979 | 1002 | 933 | 991 | 10000 |

| Error Rate | 0.1261 |
|---|---|
| Accuracy for 0 | 0.9642857142857143 |
| Accuracy for 1 | 0.9726872246696036 |
| Accuracy for 2 | 0.8023255813953488 |
| Accuracy for 3 | 0.8772277227722772 |
| Accuracy for 4 | 0.9073319755600815 |
| Accuracy for 5 | 0.7724215246636772 |
| Accuracy for 6 | 0.9133611691022965 |
| Accuracy for 7 | 0.8686770428015564 |
| Accuracy for 8 | 0.8039014373716633 |
| Accuracy for 9 | 0.8374628344895937 |

Figure 18: One versus All Classifier with Sinusoidal Function on Testing Data

One particular observation is that the performance of the classifier with the ReLU function does not perform as well as the classifier with the sigmoid or sinusoidal feature function.

Figure 19 depicts the confusion matrix and error rates for the One versus One classifier with a ReLU feature function evaluated on training data.

```
predictions = dot(train_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, train_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5812 | 1 | 14 | 11 | 9 | 25 | 18 | 4 | 27 | 2 | 5923 |
| 1 | 1 | 6637 | 39 | 10 | 5 | 6 | 2 | 13 | 19 | 10 | 6742 |
| 2 | 51 | 49 | 5585 | 43 | 47 | 13 | 43 | 41 | 74 | 12 | 5958 |
| 3 | 25 | 35 | 105 | 5623 | 7 | 149 | 15 | 47 | 85 | 40 | 6131 |
| 4 | 18 | 23 | 19 | 1 | 5585 | 8 | 15 | 16 | 8 | 149 | 5842 |
| 5 | 37 | 39 | 32 | 131 | 24 | 5013 | 78 | 8 | 42 | 17 | 5421 |
| 6 | 35 | 13 | 29 | 1 | 29 | 82 | 5704 | 0 | 25 | 0 | 5918 |
| 7 | 11 | 56 | 55 | 3 | 75 | 11 | 2 | 5919 | 10 | 123 | 6265 |
| 8 | 25 | 136 | 35 | 99 | 32 | 112 | 39 | 22 | 5283 | 68 | 5851 |
| 9 | 23 | 11 | 19 | 76 | 148 | 29 | 2 | 128 | 30 | 5483 | 5949 |
| totals | 6038 | 7000 | 5932 | 5998 | 5961 | 5448 | 5918 | 6198 | 5603 | 5904 | 60000 |

| Error Rate | 0.05593333333333328 |
|---|---|
| Accuracy for 0 | 0.9812594968765828 |
| Accuracy for 1 | 0.9844259863541975 |
| Accuracy for 2 | 0.937395099026519 |
| Accuracy for 3 | 0.9171423911270592 |
| Accuracy for 4 | 0.9560082163642588 |
| Accuracy for 5 | 0.9247371333702269 |
| Accuracy for 6 | 0.9638391348428523 |
| Accuracy for 7 | 0.9447725458898644 |
| Accuracy for 8 | 0.9029225773372074 |
| Accuracy for 9 | 0.9216675071440579 |

Figure 19: One versus One Classifier with Sinusoidal Function on Training Data

Figure 20 depicts the confusion matrix and error rates for the One versus One classifier with a ReLU feature function evaluated on testing data.

```
predictions = dot(test_in_feature, prediction_weights)
prediction_results = collapse_results(sign(predictions))
analyze_multi(prediction_results, test_expected[0])
```

| Predicted ---> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Totals |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 964 | 0 | 1 | 0 | 2 | 6 | 4 | 2 | 1 | 0 | 980 |
| 1 | 0 | 1117 | 2 | 4 | 1 | 1 | 2 | 1 | 7 | 0 | 1135 |
| 2 | 15 | 7 | 939 | 9 | 12 | 3 | 10 | 11 | 26 | 0 | 1032 |
| 3 | 6 | 0 | 16 | 927 | 3 | 17 | 4 | 7 | 25 | 5 | 1010 |
| 4 | 2 | 1 | 10 | 0 | 937 | 0 | 5 | 3 | 2 | 22 | 982 |
| 5 | 9 | 4 | 1 | 26 | 9 | 807 | 14 | 2 | 15 | 5 | 892 |
| 6 | 7 | 3 | 6 | 0 | 7 | 14 | 919 | 0 | 2 | 0 | 958 |
| 7 | 3 | 14 | 15 | 6 | 10 | 0 | 0 | 950 | 2 | 28 | 1028 |
| 8 | 9 | 11 | 8 | 20 | 9 | 31 | 8 | 9 | 856 | 13 | 974 |
| 9 | 7 | 7 | 5 | 11 | 29 | 8 | 0 | 17 | 7 | 918 | 1009 |
| totals | 1022 | 1164 | 1003 | 1003 | 1019 | 887 | 966 | 1002 | 943 | 991 | 10000 |

| Error Rate | 0.06659999999999999 |
|---|---|
| Accuracy for 0 | 0.9836734693877551 |
| Accuracy for 1 | 0.9841409691629956 |
| Accuracy for 2 | 0.9098837209302326 |
| Accuracy for 3 | 0.9178217821782179 |
| Accuracy for 4 | 0.9541751527494908 |
| Accuracy for 5 | 0.9047085201793722 |
| Accuracy for 6 | 0.9592901878914405 |
| Accuracy for 7 | 0.9241245136186771 |
| Accuracy for 8 | 0.8788501026694046 |
| Accuracy for 9 | 0.9098116947472745 |

Figure 20: One versus One Classifier with Sinusoidal Function on Testing Data

There are no new trends in the confusion matrix. The ReLU feature function allows the classifiers to perform a little better than the identity function. I supposed this is because the ReLU function is just the identity function when $x > 0$.

The sinusoidal and sigmoid feature functions performed the best on the both the training and testing data. Next, ReLU, and then finally the identity function. Images for 0 and 1 were easiest to match while images for 5 and 8 were the hardest to match through most of the classifiers.
The Identity function did little to nothing in order to change the results for classifications, but the other functions performed noticably better than the original multi-class classifiers.

## 3.2 Performance based on Varied Feature Spaces

Figure 21 describes the change in error rate as the dimension of the randomized feature space, L, changes.

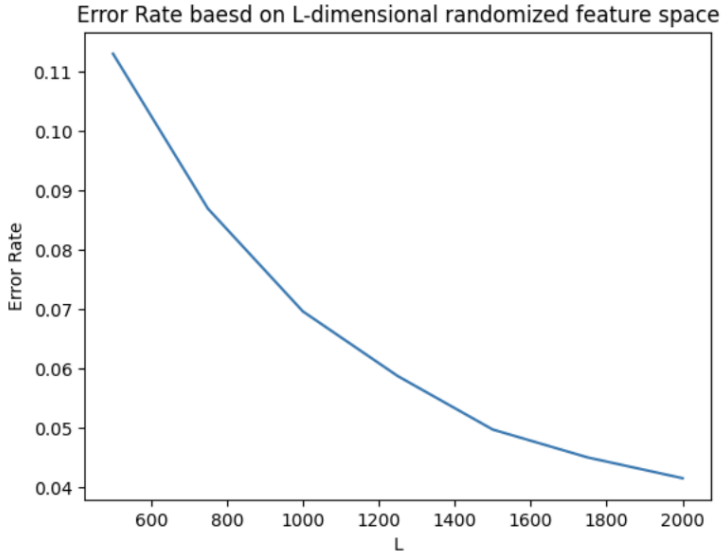


Figure 21: L vs Error rate

From the graph, as L increases, error rate decreases. However, the rate at which the error rate decreases decreases as well. The graph looks similar to a exponential function that will stabilize around one value with larger L. At some point, sacrificing time and computing power would not be worth for minuscule changes in error rate.

### 3.3 Robustness of Classifier

Figure 22 shows the change in error rate as the norm of the Gaussian noise changes. The following was tested on a one versus all classifier trained on clean training data (without any noise) tested on noisy test data. The norm of the noise was controlled by altering the variance of the Gaussian distribution while maintaining the mean as 0.

```
train_in_feature = np.hstack((np.ones((train_in.shape[0], 1)), train_in))
test_in_feature = test_in
prediction_weights = make_one_all(train_in_feature, train_expected)
evaluate_robustness()
```

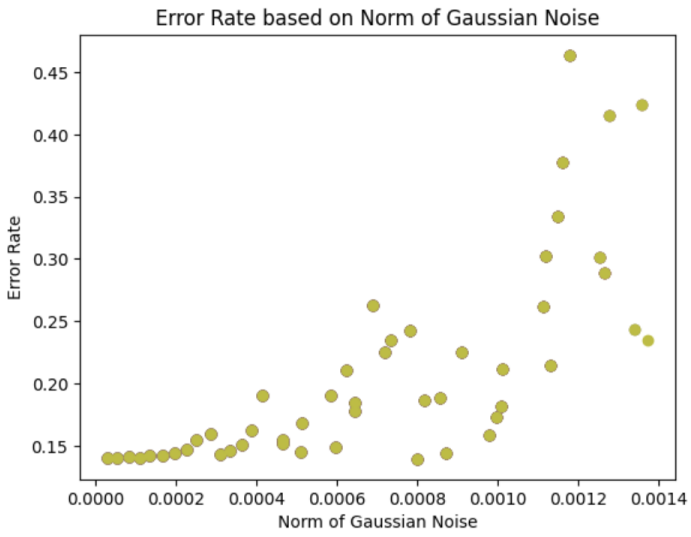


Figure 22: Norm of Gaussian Noise Vector vs Error Rate

According to the scatter plot, almost immediately, the classifier begins to break. As soon as the norm goes beyond 0.0002, the error rate increase exponentially.

Figure 23 shows the same process but with the training data modified by a sinusoidal feature function.

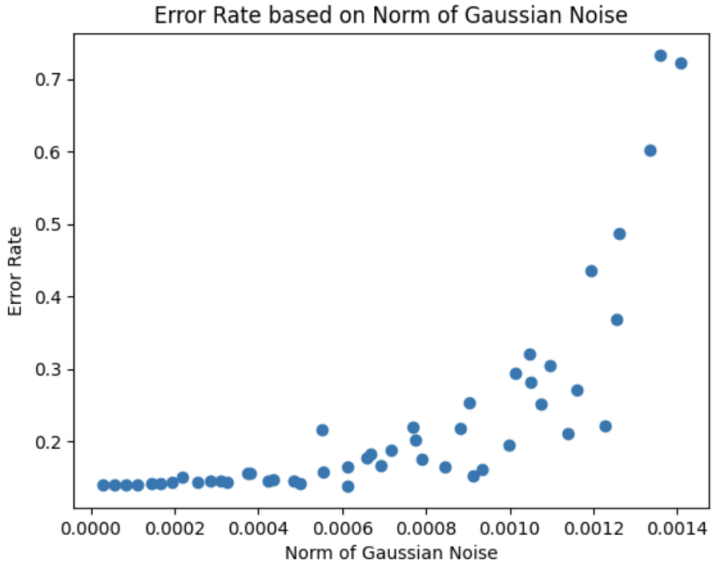


Figure 23: Norm of Gaussian Noise Vector vs Error Rate

The points look more "in line" than the previous figure. The points remain stabilized through the domain, and the classifier doesn't seem to break at least until the norm of the noise is 0.0008.

The trends of the error rates seem to follow the strength of the classifier as evaluated in section 3.1. With the sinusoidal function have the best performance, it also performs well with noise.

## 3.4 Robustness on Binary Classifier

Figure 24 shows the change in the confidence values as the norm of the Gaussian noise changes. The following was tested on a binary classifier trained on clean training data (without any noise) for the number '4' tested on noisy image of a '4'. The norm of the noise was controlled by altering the variance of the Gaussian distribution while maintaining the mean as 0.

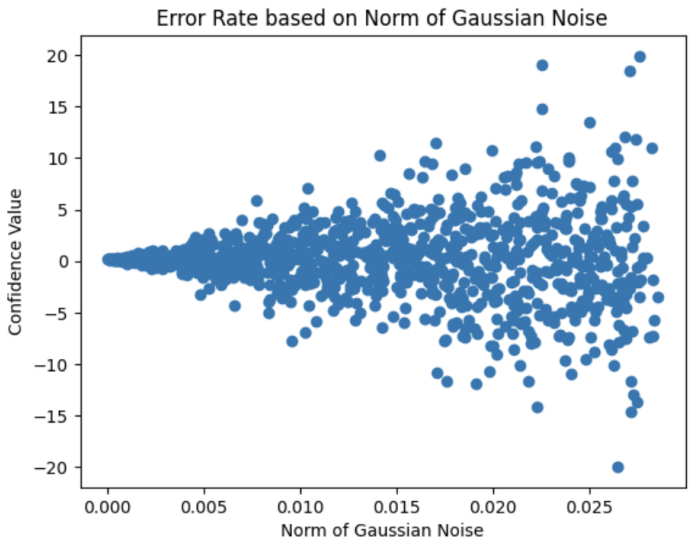


Figure 24: Norm of Gaussian Noise Vector vs Confidence Values

As the norm of the noise increases, the confidence values increase in range. A value that was once bound between -1 and 1 slowly becomes sporadic as soon as it escapes these bounds. This binary classifier is very prone to noise, seeing that values begin to change very quickly.

## 3.5 Varied Randomized Feature Spaces

Figure 25 shows the change in error rate as the variance of the Gaussian randomization changes. The following was tested on a one versus all classifier trained on a randomized feature space tested on clean test data. The feature space was 1000 dimensions from a Gaussian distribution with a sinusoidal feature function. The variance was altered from 0 to 10 (0 causing the data to be a 0 matrix hence 0.9 error rate) From the following graph, I can see that there is no correlation between variance of the Gaussian random features and error rates.

```
plt.scatter(*zip(*var_vs_errs))
plt.title("Error Rate based on Variance of Randomize Feature")
plt.xlabel("Variance of Randomize Feature")
plt.ylabel("Error Rate")
```

```
Text(0, 0.5, 'Error Rate')
```

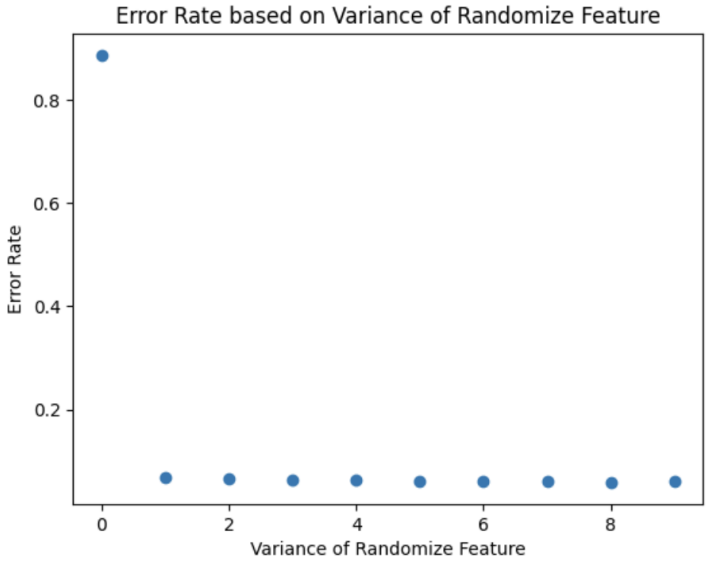


Figure 25: Variance of Noise Vector vs Error Rate

Due to hardware limitations, I could not make further computations with higher variances, but I can still conclude that low valued variances of the randomized feature does not affect the error rate.

The following figure is 10 error rates from a poisson random feature with $\lambda = 1$

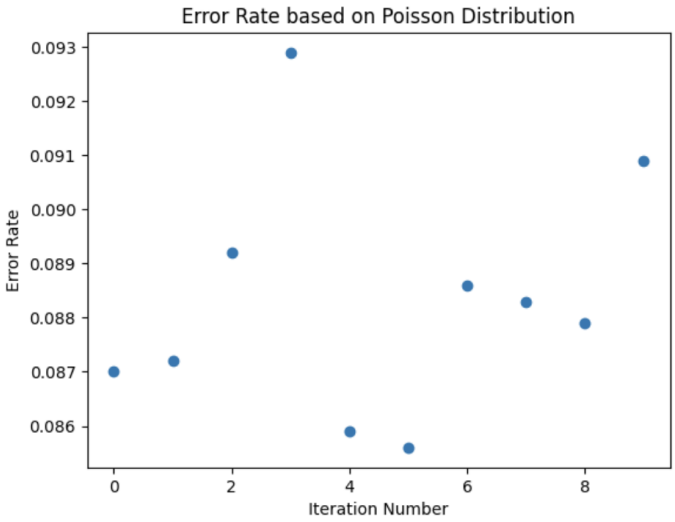


Figure 26: Poisson Distribution vs Error Rate

Poisson distribution will absolutely destroy the classifier. Although poisson random distributions do not model real-world noise accurately, I just wanted to see what it would look like.

From this project I learned that I need a new computer. Hardware limitations prevented me from making further experimentations.