Machine Learning

Homework3

Shervin Halat

98131018

1.

In order to compare the two mentioned classification methods, it should be noted that although the two methods have many basic similarities (such as using Bayes' decision rule) but also, they differ remarkably in the way of implementation. In the following, some major and significant similarities and dissimilarities are mentioned.

First of all, it should be noted that Logistic Regression is a discriminative model (Discriminative models learn the posterior probability) as against Naïve Bayes which is generative model (Generative classifiers learn a model of joint probabilities). Although, many believed that discriminative model is always superior to generative model but it is shown by many experiments that depending on different situations generative models may overcome discriminative ones. Naïve Bayes model is a much faster method as it uses biases like independent features but at the same time the error of Naïve Bayes is higher than Logistic Regression. Thus, as the number of training examples (m) is increased, one would expect generative naive Bayes to initially do better, but for discriminative logistic regression to eventually catch up to, and quite likely overtake, the performance of naive Bayes.

Besides, as the number of training examples grows toward infinity, the Naïve Bayes and Logistic Regression converge toward identical classifiers. Thus, in these situations it is suggested to use Naïve Bayes as it reaches to the output faster (Naïve Bayes reaches its asymptotic faster (O(log n)) than the Logistic Regression (O (n))).

Also, for low amount of data NB can perform better when there's a low amount of training data as it is a generative model! But when sufficient amount of data is available, Logistic Regression outperform compared to Naïve Bayes.

2.

Due to huge amount of increase in text data, Naïve Bayes classification has become an important issue nowadays. Due to the naïve assumption of independent features, some kind of issues such as Zero Frequency Problem may appear. This problem mainly results from lack of sufficient amount of data. For example, if the conditional probability given class C_i of a specific feature equals to zero, regardless of adequacy of other probabilities, the conditional probability given class C_i of the feature vector would results in zero. That is, assuming **$P(W_k|C) = 0$** ("c" denotes class and "d" denotes feature vector and "w" denotes different values of each feature value) and according to Naïve Bayes assumption:

$$P(d|c) = P(w_1|c)\ P(w_2|c)..\ P(w_d|c) = \prod_{1<k<d} P(wk|c)$$

The P(d|c) would become zero regardless of other conditional probabilities. Thus, **$P(C|d) = 0$** which doesn't seem logical.

To tackle the issue mentioned above, smoothing methods were suggested. These methods firstly, would generally improve the accuracy of the model and secondly, Accommodate the generation of common and non-informative feature values. So the main purpose of the smoothing is to provide a non-zero probability to unseen feature values and improve the accuracy of probability estimator.

The main difference between different smoothing methods are the way each one computes **$P(W_k|C_i)$.**

Different smoothing methods are as follows:

(P(w|C) denotes the maximum likelihood estimation of word w in collection C)

a) Laplace smoothing:

$$p(w|c_i) = \frac{1 + c(w, c_i)}{|V| + \sum_{w' \in V} c(w', c_i)}$$

where $c(w, c_i)$ is the frequency of word w in category c_i, and |V| is the size of vocabulary. In other words, the Laplace smoothing virtually adds equal number of data to each value of each feature!

b) Jelinek-Mercer (JM) smoothing:

This method involves a linear interpolation of the maximum likelihood model with the collection model using a coefficient "λ"

$$p_\lambda(w|c_i) = (1 - \lambda)\frac{c(w, c_i)}{\sum_{w' \in V} c(w', c_i)} + \lambda p(w|C)$$

c) Dirichlet (Dir) smoothing:

The conjugate prior for the Bayesian analysis is the Dirichlet distribution with parameters (μp(w1|c), μp(w2|c), μp(w3|c),….., μp(w1|c))

$$p_\mu(w|c_i) = \frac{c(w, c_i) + \mu p(w|C)}{\sum_{w' \in V} c(w', c_i) + \mu}$$

d) **Absolute Discounting (AD) smoothing:**

It lowers the probability of seen words by subtracting a constant from their counts. It is similar to JK method but differs in that it discounts the probability by subtracting instead of multiplying.

$$p_\delta(w|c_i) = \frac{max(c(w,c_i) - \delta, 0) + \delta|c_i|_u p(w|C)}{\sum_{w' \in V} c(w', c_i)}$$

Where δ is a discount constant and σ = δ $|d|_u$ / $|d|$, so that it equals to one. Here, $|d|_u$ is the number of unique terms in d and $|d|$ are the total number of terms.

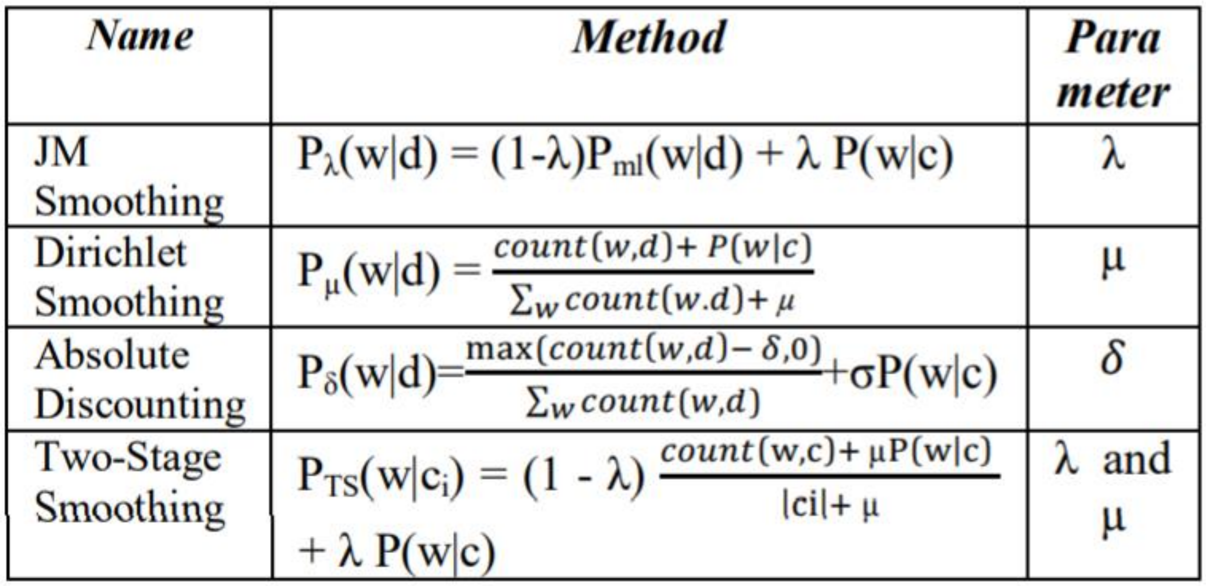
e) **Two-stage (TS) smoothing:**

It combines the Dirichlet Smoothing with the Interpolation method.

$$p_{\lambda,\mu}(w|c_i) = (1-\lambda)\frac{c(w,c_i) + \mu p(w|C)}{\sum_{w' \in V} c(w', c_i) + \mu} + \lambda p(w|C)$$

Laplace Smoothing is replaced by various sophisticated smoothing methods like JK Smoothing, Dirichlet Smoothing, Two-Stage Smoothing, and Absolute Discounting.

Summary of Smoothing Techniques:



Name	*Method*	*Para meter*
JM Smoothing	$P_\lambda(w\|d) = (1-\lambda)P_{ml}(w\|d) + \lambda\, P(w\|c)$	λ
Dirichlet Smoothing	$P_\mu(w\|d) = \frac{count(w,d)+ P(w\|c)}{\Sigma_w count(w.d)+ \mu}$	μ
Absolute Discounting	$P_\delta(w\|d)=\frac{\max(count(w,d)-\delta,0)}{\Sigma_w count(w,d)}+\sigma P(w\|c)$	δ
Two-Stage Smoothing	$P_{TS}(w\|c_i) = (1 - \lambda)\, \frac{count(w,c)+ \mu P(w\|c)}{\|ci\|+ \mu} + \lambda\, P(w\|c)$	λ and μ

References:

1) https://www.ntu.edu.sg/home/gaocong/papers/wpp095-yuan.pdf
2) https://pdfs.semanticscholar.org/ffd3/5579c2c6390606076b593eddd8faf5085a2f.pdf

3.

???

4.

Assuming Naïve Bayes classifier with continuous features, with assumption of Gaussian distribution for each feature of each class:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi}\sigma_{i,k}} e^{-(x_i-\mu_{i,k})^2/2\sigma_{i,k}^2}$$

Figure 1:

Considering assumption described above, the “?” test data in figure 1 would be assigned to the class “A”.

The reason is as follows:

The two classes “A” and “B” are distributed with the same mean and standard deviation, since, both are distributed cyclic with common centers. Thus,

P(X|A) = P(X|B)

What’s more, as both distributions are cyclic, the covariance matrixes are both diagonal therefore, features for both distributions are independent.

P(X|A) = P(X1|A) * P(X2|A)

P(X|B) = P(X1|B) * P(X2|B)

Besides, for Naïve Bayes classifier we compare:

$$p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Hence, the only parameters that differ in figure 1 are the prior probabilities which is obvious from the figure that:

P(A) >> P(B)

Thus, for figure 1:

P(A|?) >> P(B|?)

Figure 2:

Considering assumption described above, the "?" test data in figure 2 would be assigned to the class "B".

The reason is as follows:

As mentioned before in Naïve Bayes we compare:

$$p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

and in this example for "?" the probability of **P(?$_{x1}$|A) is approximately zero!** Thus, regardless of other probabilities' values, **P(A|?) is zero!** Contrarily, we know that P(B|?) is anything except zero. Thus, "?" belongs to class "B".

The reason for **P(?$_{x1}$|A) = 0** is that the variance of X1 for class A is approximately equal to zero. Then, the normal distribution for this feature for class "A" becomes an impulse function which is zero in all values except the impulse point!

5.

The cost function with regularization parameter for logistic regression is as follows:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \, log(h_\theta(x^{(i)}) + (1 - y^{(i)}) \, log(1 - h_\theta(x^{(i)})) \right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Similar to regularization in linear regression, we can avoid overfitting training data by implementing the second sum. It should be noted that the value of lambda should be reasonable and it shouldn’t be so small or so large. For example, considering the figure of the question, for very small amount of **λ** the model is overfitted and as a result probability of training data has reached the value of 1 hence, the average log-probability has reached zero. But as we continuously increase **λ**, the model is less overfitted and bios of the model is increased and as a result the training data error increases or in other words the probability of training labels decreases and hence the average log-probability of training data lastly diminishes to a specific value (when **λ** is reaches infinity)

6.

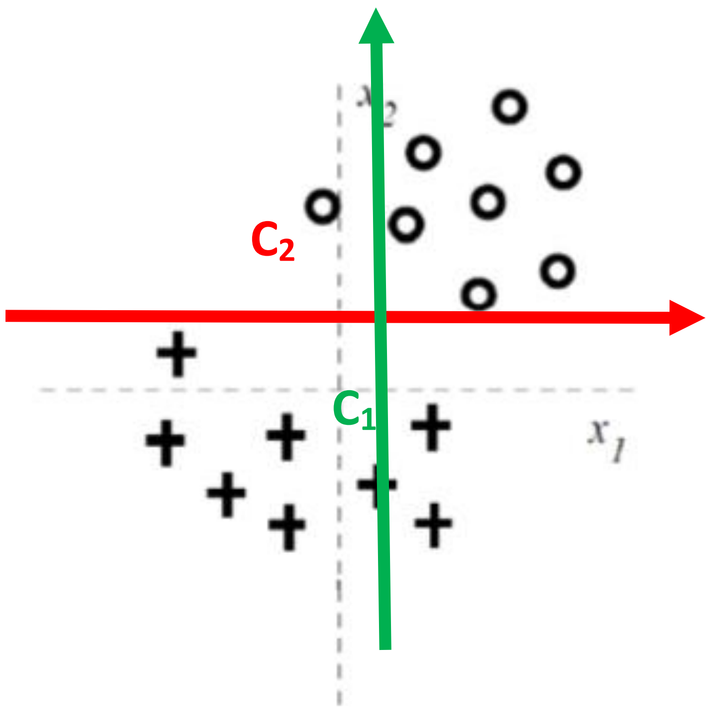
Generally, considering high enough value for regularization parameter results in negligible value of corresponding Θ which at last results in the elimination of corresponding feature from the decision boundary equation. Therefore, in this case the decision boundary would become something like the following in each case: ("C" is constant)

$$X1 = C_1$$

or

$$X2 = C_2$$

As it is obvious from the figure below, the discriminability of feature X2 (Red line) is more than feature X1 (Green Line)



7.

a.

In the case of continuous variables, Gaussian distribution for each feature of each class has been assumed as follows:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi}\sigma_{i,k}} e^{-(x_i-\mu_{i,k})^2/2\sigma_{i,k}^2}$$

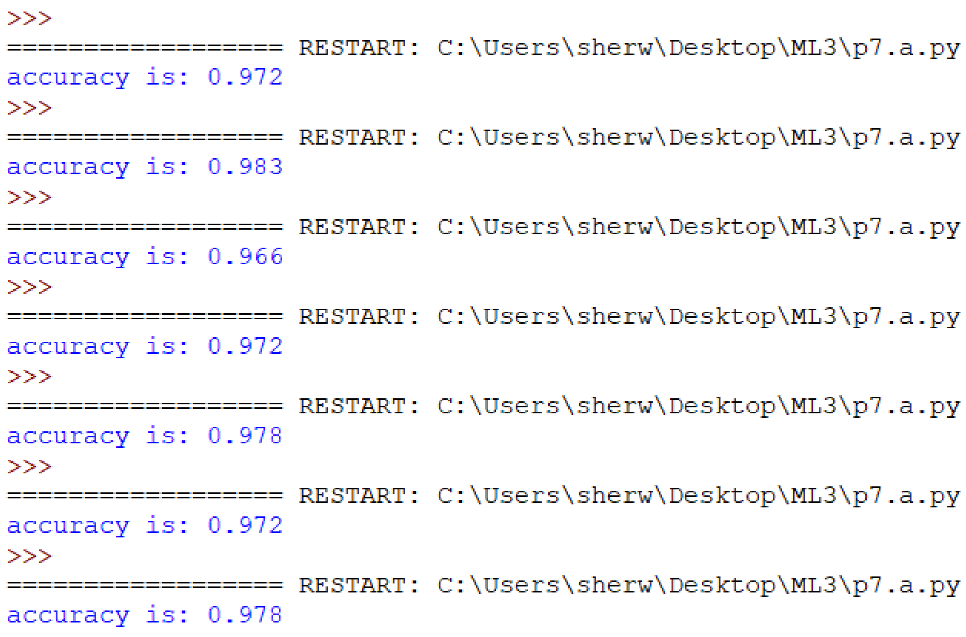
The prior probabilities are the same as discrete type.

In order to overcome the issue of smallness of multiplication of many probabilities, the logarithm of the equation below was taken.

$$\log(P(\text{class}_i|\,\mathbf{data})) \propto \log(P(\text{class}_i)) + \sum_j \log(P(\text{data}_j|\text{class}_i))$$

The implementation of this problem is in the script p7.a

According to the code results, the accuracy obtained using 6-fold cross validation for multiple runs is as follows:



```
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.972
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.983
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.966
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.972
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.978
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.972
>>> 
================== RESTART: C:\Users\sherw\Desktop\ML3\p7.a.py
accuracy is: 0.978
```

As it can be figured out from the results above, the accuracy is something around 97%.

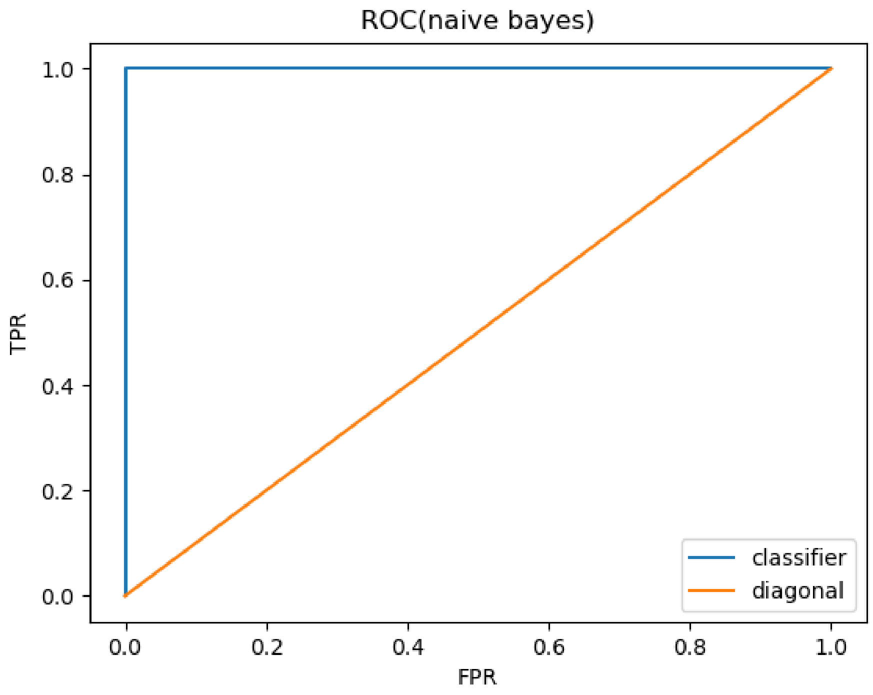
b.

To plot the ROC curve two **classes 1 and 2** were considered and 70% of the dataset was considered as Training Data hence, **30% Test Data**.

(1: Negative class, 2: Positive class)

The related script is **p7.b**.

The obtained ROC curve is as follows:



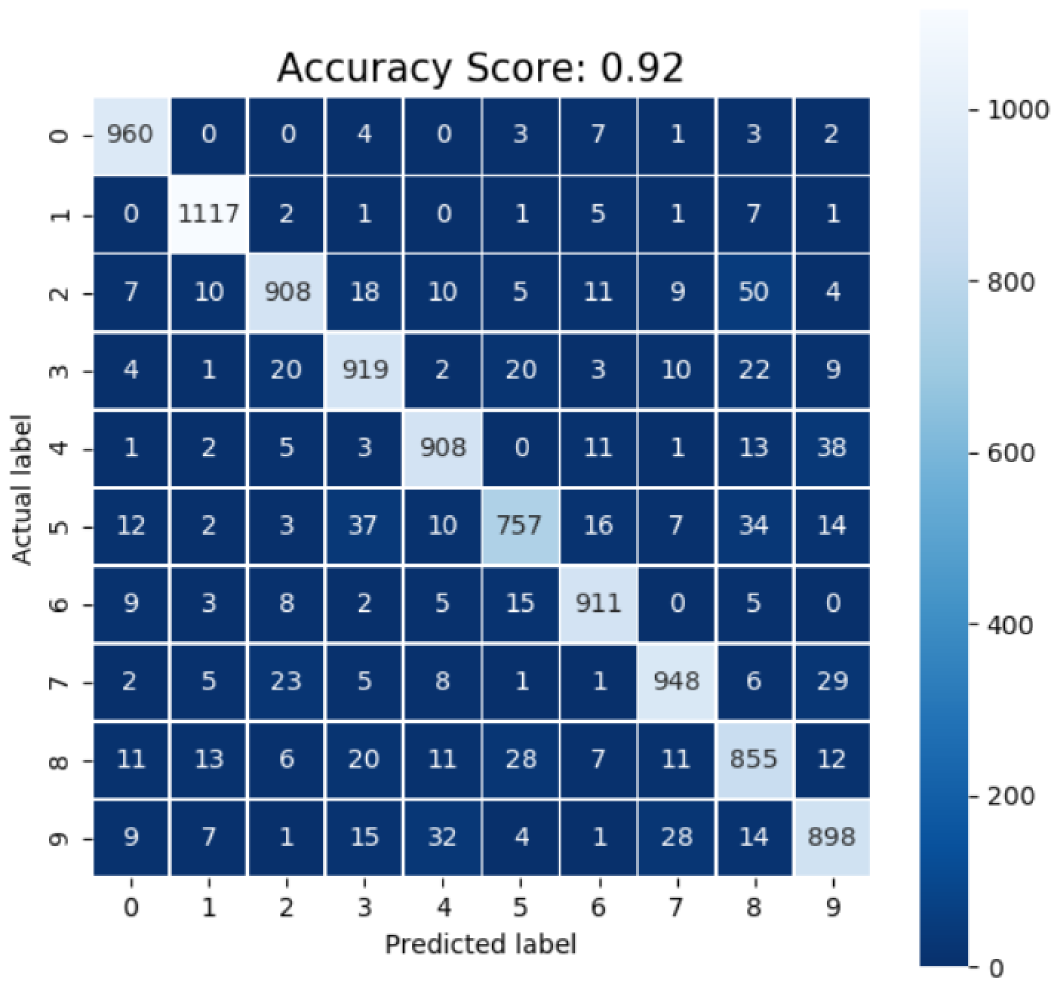
Considering the accuracy obtained in part “a” the ROC curve above was expectable.

8.

a.

One to seven (1 / 7) of the MNIST dataset was considered as Training et and 6 to 7 parts as Test Set. Therefore, 60000 samples as Training samples. (Related script is **p8.a.OVR**)

Confusion Matrix:



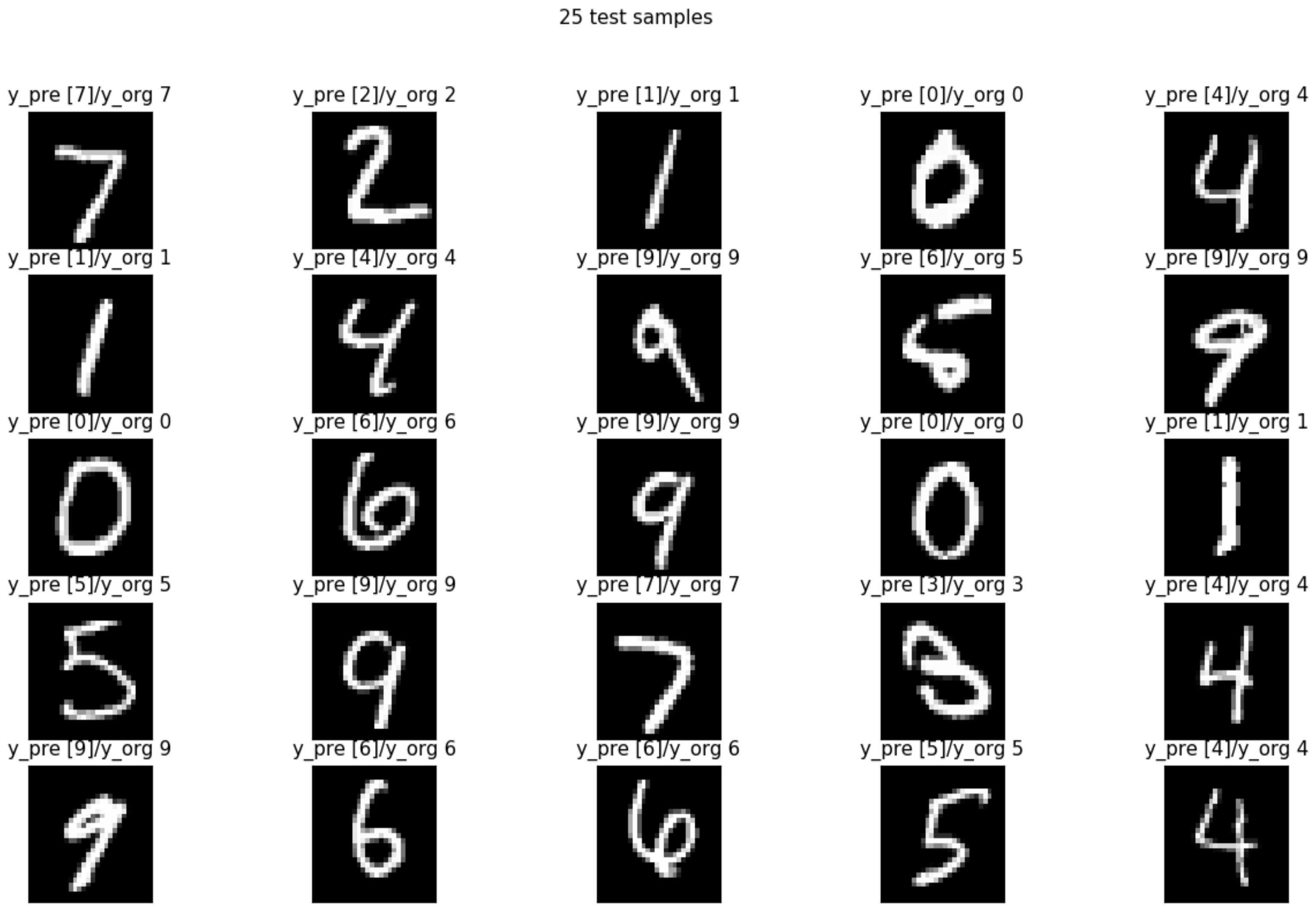
	0	1	2	3	4	5	6	7	8	9
0	960	0	0	4	0	3	7	1	3	2
1	0	1117	2	1	0	1	5	1	7	1
2	7	10	908	18	10	5	11	9	50	4
3	4	1	20	919	2	20	3	10	22	9
4	1	2	5	3	908	0	11	1	13	38
5	12	2	3	37	10	757	16	7	34	14
6	9	3	8	2	5	15	911	0	5	0
7	2	5	23	5	8	1	1	948	6	29
8	11	13	6	20	11	28	7	11	855	12
9	9	7	1	15	32	4	1	28	14	898

Test Set accuracy = 91.8% (92%)

Training Set accuracy = 92.3%

b.

Related script is **p8.b**



c.

???

d.

Generally, one of the remarkable issues in most machine learning classification methods is the model’s bios toward the classes with superior prior probabilities. This issue may be due to a topic named imbalanced classes (or imbalanced data) which lastly leads to imbalanced learning. In summary, imbalanced classes are classes which haven’t approximately equal data samples or in other words one class has large amount of data compared to another class. In this situation classification models may reach to an optimum accuracy by considering all test data as samples of class with large amount of training data which is obviously is not logical and acceptable. The mentioned issue, which is called Imbalanced Learning, is exactly what may happen by implementing one-vs-all method since we consider all classes as negatives and only one class as positive by which we label a large amount of data as negative class as against positive class which contains small amount of data.

In order to tackle the mentioned issue, it is suggested to define costs values in case of misclassification.