

# ISYE 6740, Summer 2021, Homework 4

## 100 points + 3 bonus points

### 1. Comparing classifiers. (65 points)

In lectures, we learn different classifiers. This question is compare them on two datasets. Python users, please feel free to use **Scikit-learn**, which is a commonly-used and powerful **Python** library with various machine learning tools. But you can also use other similar libraries in other languages of your choice to perform the tasks.

#### 1. Part One (Divorce classification/prediction). (30 points)

This dataset is about participants who completed the personal information form and a divorce predictors scale. The data is a modified version of the publicly available at <https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set> (by injecting noise so you will not get the exactly same results as on UCI website). The dataset **marriage.csv** is contained in the homework folder. There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The last column of the CSV file is label  $y$  (1 means “divorce”, 0 means “no divorce”). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 (“no divorce”) or 1 (“divorce”).

We are going to compare the following classifiers (**Naive Bayes**, **Logistic Regression**, and **KNN**). Use the first 80% data for training and the remaining 20% for testing. If you use **scikit-learn** you can use `train_test_split` to split the dataset.

*Remark: Please note that, here, for Naive Bayes, this means that we have to estimate the variance for each individual feature from training data. When estimating the variance, if the variance is zero or close to zero (meaning that there is very little variability in the feature), you can set the variance to be a small number, e.g.,  $E = 10^{-3}$ . We do not want to have include zero or nearly variance in Naive Bayes. This tip holds for both Part One and Part Two of this question.*

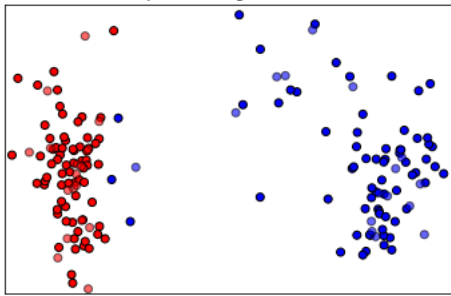
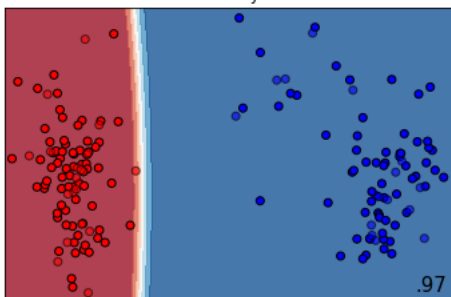
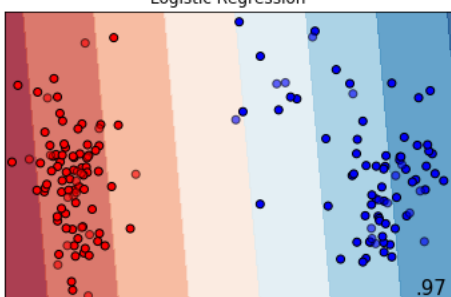
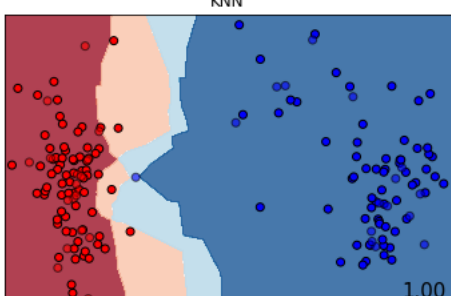
- (a) (15 points) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

Classifier	Testing Accuracy	Training Accuracy
Naïve Bayes	97.06%	97.79%
Logistic Regression	94.12%	100%
KNN	97.06%	97.79%

#### Comments:

In testing, Naïve bayes and KNN perform relatively better than Logistic regression. My guess is that the data may not be linearly seperable and Logistic regression builds a linear classifier while Naïve bayes and KNN have flexible boundaries and can accommodate the classifications better.

- (b) (15 points) Now perform PCA to project the data into two-dimensional space. Build the classifiers (**Naïve Bayes**, **Logistic Regression**, and **KNN**) using the two-dimensional PCA results. Plot the data points and decision boundary of each classifier in the two-dimensional space. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

2D PCA plots	Comments
<p>Input data (light = test)</p> 	<p>While the darker points indicate train data, the lighter tones indicate test data.</p> <p>The data is clustered really well by using just two reduced dimensions. But there are a few blue points (label1) very close to the red (label2).</p> <p>Let us see how different classifiers do the job of classification.</p>
<p>Naïve Bayes</p> 	<p>Naïve bayes classifier has a narrow band of boundary and classified the data with an accuracy of 97%. Shape of the boundary is curved.</p> <p>Although the boundary is very close to the label2 (red) data points, it did not classify the blue points(label1) in the red region correctly.</p>
<p>Logistic Regression</p> 	<p>Logistic regression has a very broad band of boundary and classified the data with an accuracy of 97%. It is a linear boundary.</p> <p>The blue data points that are close to red cluster are not classified correctly with this boundary</p>
<p>KNN</p> 	<p>KNN classifier has a very <b>noisy boundary</b>. The boundary is very jagged. The boundary is flexible and included the blue data points that are closer to the red cluster too.</p> <p>Hence the accuracy is 100%. A better accuracy than Logistic and Naïve bayes.</p>

## 2. Part Two (Handwritten digits classification). (35 points)

This question is to compare different classifiers and their performance for multi-class classifications on the complete MNIST dataset at <http://yann.lecun.com/exdb/mnist/>. You can find the data file **mnist\_10digits.mat** in the homework folder. The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. We will compare **KNN, logistic regression, SVM, kernel SVM, and neural networks**.

- We suggest you to “standardize” the features before training the classifiers, by dividing the values of the features by 255 (thus map the range of the features from [0, 255] to [0, 1]).
- You may adjust the number of neighbors  $K$  used in KNN to have a reasonable result (you may use cross validation but it is not required; any reasonable tuning to get good result is acceptable).
- You may use a neural networks function `sklearn.neural network` with hidden layer sizes = (20, 10).
- For kernel SVM, you may use radial basis function kernel and choose proper kernel.
- For KNN and SVM, you can randomly downsample the training data to size  $m = 5000$ , to improve computation efficiency.

Train the classifiers on training dataset and evaluate on the test dataset.

- (a) (25 points) Report confusion matrix, precision, recall, and F-1 score for each of the classifiers. For precision, recall, and F-1 score of each classifier, we will need to report these for each of the digits. So you can create a table for this. For this question, each of the 5 classifier, **KNN, logistic regression, SVM, kernel SVM, and neural networks**, accounts for 10 points.

Classifier	Confusion matrix (actual vs predicted) Report: Precision, Recall, F1 score & Accuracy										
KNN		0	1	2	3	4	5	6	7	8	9
	0	942	1	0	0	1	12	20	2	2	0
	1	0	1130	2	1	0	0	1	0	1	0
	2	21	154	742	23	11	3	10	34	33	1
	3	2	40	6	905	1	16	3	14	11	12
	4	1	47	0	0	829	0	16	1	1	87
	5	8	32	0	32	8	754	20	4	6	28
	6	15	14	0	0	19	7	903	0	0	0
	7	1	80	2	1	5	0	0	891	0	48
	8	23	49	2	25	14	18	10	11	775	47
	9	10	22	1	6	21	2	2	20	1	924
		precision				recall		f1-score		support	
		0	0.92	0.96	0.94	980					
		1	0.72	1.00	0.84	1135					
		2	0.98	0.72	0.83	1032					
		3	0.91	0.90	0.90	1010					
		4	0.91	0.84	0.88	982					
		5	0.93	0.85	0.88	892					
		6	0.92	0.94	0.93	958					
		7	0.91	0.87	0.89	1028					
		8	0.93	0.80	0.86	974					
		9	0.81	0.92	0.86	1009					
		accuracy			0.88	10000					
		macro avg	0.89	0.88	0.88	10000					
		weighted avg	0.89	0.88	0.88	10000					

Logistic regression	0	1	2	3	4	5	6	7	8	9	
	0	960	0	1	2	0	5	6	3	1	2
	1	0	1112	3	1	0	1	5	1	12	0
	2	8	8	920	20	9	5	10	11	37	4
	3	4	0	17	919	2	22	4	12	21	9
	4	1	2	5	3	914	0	10	2	7	38
	5	10	2	0	40	10	771	17	7	28	7
	6	9	3	7	2	6	20	907	1	3	0
	7	2	7	22	5	8	1	1	950	5	27
	8	10	14	5	21	14	27	7	11	853	12
9	8	8	2	13	31	14	0	24	12	897	
precision recall f1-score support											
0 0.95 0.98 0.96 980											
1 0.96 0.98 0.97 1135											
2 0.94 0.89 0.91 1032											
3 0.90 0.91 0.90 1010											
4 0.92 0.93 0.93 982											
5 0.89 0.86 0.88 892											
6 0.94 0.95 0.94 958											
7 0.93 0.92 0.93 1028											
8 0.87 0.88 0.87 974											
9 0.90 0.89 0.89 1009											
accuracy 0.92 0.92 0.92 10000											
macro avg 0.92 0.92 0.92 10000											
weighted avg 0.92 0.92 0.92 10000											
SVM	0	1	2	3	4	5	6	7	8	9	
	0	959	0	4	1	0	6	6	2	2	0
	1	0	1118	1	3	0	2	3	0	8	0
	2	12	5	939	20	12	4	9	10	19	2
	3	2	3	23	899	3	41	3	12	16	8
	4	1	1	6	0	933	0	6	5	0	30
	5	11	3	4	46	13	770	10	2	25	8
	6	11	2	17	2	18	16	886	0	5	1
	7	1	14	34	10	10	1	0	916	5	37
	8	12	8	20	41	10	31	9	14	819	10
9	6	9	5	8	54	10	0	23	5	889	
precision recall f1-score support											
0 0.94 0.98 0.96 980											
1 0.96 0.99 0.97 1135											
2 0.89 0.91 0.90 1032											
3 0.87 0.89 0.88 1010											
4 0.89 0.95 0.92 982											
5 0.87 0.86 0.87 892											
6 0.95 0.92 0.94 958											
7 0.93 0.89 0.91 1028											
8 0.91 0.84 0.87 974											
9 0.90 0.88 0.89 1009											
accuracy 0.91 0.91 0.91 10000											
macro avg 0.91 0.91 0.91 10000											
weighted avg 0.91 0.91 0.91 10000											
Kernel SVM	0	1	2	3	4	5	6	7	8	9	
	0	967	0	2	0	1	4	4	1	1	0
	1	0	1118	4	2	0	1	3	1	6	0
	2	7	0	973	8	10	0	8	11	15	0
	3	1	0	13	952	0	17	1	14	9	3
	4	1	0	4	0	943	0	7	2	1	24
	5	5	1	3	13	6	841	12	1	6	4
	6	9	3	3	0	7	7	926	0	3	0
	7	0	15	21	4	6	0	0	948	2	32
	8	6	1	4	13	10	10	3	4	917	6
9	7	6	4	11	28	5	0	8	6	934	

	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.96</td><td>0.99</td><td>0.98</td><td>980</td></tr><tr><td>1</td><td>0.98</td><td>0.99</td><td>0.98</td><td>1135</td></tr><tr><td>2</td><td>0.94</td><td>0.94</td><td>0.94</td><td>1032</td></tr><tr><td>3</td><td>0.95</td><td>0.94</td><td>0.95</td><td>1010</td></tr><tr><td>4</td><td>0.93</td><td>0.96</td><td>0.95</td><td>982</td></tr><tr><td>5</td><td>0.95</td><td>0.94</td><td>0.95</td><td>892</td></tr><tr><td>6</td><td>0.96</td><td>0.97</td><td>0.96</td><td>958</td></tr><tr><td>7</td><td>0.96</td><td>0.92</td><td>0.94</td><td>1028</td></tr><tr><td>8</td><td>0.95</td><td>0.94</td><td>0.95</td><td>974</td></tr><tr><td>9</td><td>0.93</td><td>0.93</td><td>0.93</td><td>1009</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.95</td><td>10000</td></tr><tr><td>macro avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>10000</td></tr><tr><td>weighted avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>10000</td></tr></table>						precision	recall	f1-score	support	0	0.96	0.99	0.98	980	1	0.98	0.99	0.98	1135	2	0.94	0.94	0.94	1032	3	0.95	0.94	0.95	1010	4	0.93	0.96	0.95	982	5	0.95	0.94	0.95	892	6	0.96	0.97	0.96	958	7	0.96	0.92	0.94	1028	8	0.95	0.94	0.95	974	9	0.93	0.93	0.93	1009	accuracy			0.95	10000	macro avg	0.95	0.95	0.95	10000	weighted avg	0.95	0.95	0.95	10000																																																									
	precision	recall	f1-score	support																																																																																																																																
0	0.96	0.99	0.98	980																																																																																																																																
1	0.98	0.99	0.98	1135																																																																																																																																
2	0.94	0.94	0.94	1032																																																																																																																																
3	0.95	0.94	0.95	1010																																																																																																																																
4	0.93	0.96	0.95	982																																																																																																																																
5	0.95	0.94	0.95	892																																																																																																																																
6	0.96	0.97	0.96	958																																																																																																																																
7	0.96	0.92	0.94	1028																																																																																																																																
8	0.95	0.94	0.95	974																																																																																																																																
9	0.93	0.93	0.93	1009																																																																																																																																
accuracy			0.95	10000																																																																																																																																
macro avg	0.95	0.95	0.95	10000																																																																																																																																
weighted avg	0.95	0.95	0.95	10000																																																																																																																																
Neural networks	<table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th></tr><tr><td>0</td><td>962</td><td>0</td><td>2</td><td>0</td><td>0</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td></tr><tr><td>1</td><td>0</td><td>1116</td><td>3</td><td>2</td><td>0</td><td>0</td><td>5</td><td>0</td><td>8</td><td>1</td></tr><tr><td>2</td><td>6</td><td>3</td><td>984</td><td>12</td><td>4</td><td>4</td><td>5</td><td>4</td><td>7</td><td>3</td></tr><tr><td>3</td><td>1</td><td>2</td><td>15</td><td>949</td><td>1</td><td>16</td><td>0</td><td>5</td><td>14</td><td>7</td></tr><tr><td>4</td><td>1</td><td>0</td><td>6</td><td>0</td><td>947</td><td>0</td><td>4</td><td>5</td><td>2</td><td>17</td></tr><tr><td>5</td><td>6</td><td>5</td><td>2</td><td>21</td><td>1</td><td>835</td><td>9</td><td>3</td><td>5</td><td>5</td></tr><tr><td>6</td><td>4</td><td>3</td><td>2</td><td>0</td><td>12</td><td>6</td><td>924</td><td>1</td><td>6</td><td>0</td></tr><tr><td>7</td><td>1</td><td>8</td><td>17</td><td>6</td><td>7</td><td>2</td><td>0</td><td>966</td><td>3</td><td>18</td></tr><tr><td>8</td><td>7</td><td>0</td><td>2</td><td>11</td><td>7</td><td>9</td><td>5</td><td>4</td><td>926</td><td>3</td></tr><tr><td>9</td><td>8</td><td>7</td><td>1</td><td>10</td><td>15</td><td>3</td><td>1</td><td>9</td><td>7</td><td>948</td></tr></table>											0	1	2	3	4	5	6	7	8	9	0	962	0	2	0	0	4	4	4	2	2	1	0	1116	3	2	0	0	5	0	8	1	2	6	3	984	12	4	4	5	4	7	3	3	1	2	15	949	1	16	0	5	14	7	4	1	0	6	0	947	0	4	5	2	17	5	6	5	2	21	1	835	9	3	5	5	6	4	3	2	0	12	6	924	1	6	0	7	1	8	17	6	7	2	0	966	3	18	8	7	0	2	11	7	9	5	4	926	3	9	8	7	1	10	15	3	1	9	7	948	
	0	1	2	3	4	5	6	7	8	9																																																																																																																										
0	962	0	2	0	0	4	4	4	2	2																																																																																																																										
1	0	1116	3	2	0	0	5	0	8	1																																																																																																																										
2	6	3	984	12	4	4	5	4	7	3																																																																																																																										
3	1	2	15	949	1	16	0	5	14	7																																																																																																																										
4	1	0	6	0	947	0	4	5	2	17																																																																																																																										
5	6	5	2	21	1	835	9	3	5	5																																																																																																																										
6	4	3	2	0	12	6	924	1	6	0																																																																																																																										
7	1	8	17	6	7	2	0	966	3	18																																																																																																																										
8	7	0	2	11	7	9	5	4	926	3																																																																																																																										
9	8	7	1	10	15	3	1	9	7	948																																																																																																																										
	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.97</td><td>0.98</td><td>0.97</td><td>980</td></tr><tr><td>1</td><td>0.98</td><td>0.98</td><td>0.98</td><td>1135</td></tr><tr><td>2</td><td>0.95</td><td>0.95</td><td>0.95</td><td>1032</td></tr><tr><td>3</td><td>0.94</td><td>0.94</td><td>0.94</td><td>1010</td></tr><tr><td>4</td><td>0.95</td><td>0.96</td><td>0.96</td><td>982</td></tr><tr><td>5</td><td>0.95</td><td>0.94</td><td>0.94</td><td>892</td></tr><tr><td>6</td><td>0.97</td><td>0.96</td><td>0.97</td><td>958</td></tr><tr><td>7</td><td>0.97</td><td>0.94</td><td>0.95</td><td>1028</td></tr><tr><td>8</td><td>0.94</td><td>0.95</td><td>0.95</td><td>974</td></tr><tr><td>9</td><td>0.94</td><td>0.94</td><td>0.94</td><td>1009</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.96</td><td>10000</td></tr><tr><td>macro avg</td><td>0.96</td><td>0.96</td><td>0.96</td><td>10000</td></tr><tr><td>weighted avg</td><td>0.96</td><td>0.96</td><td>0.96</td><td>10000</td></tr></table>						precision	recall	f1-score	support	0	0.97	0.98	0.97	980	1	0.98	0.98	0.98	1135	2	0.95	0.95	0.95	1032	3	0.94	0.94	0.94	1010	4	0.95	0.96	0.96	982	5	0.95	0.94	0.94	892	6	0.97	0.96	0.97	958	7	0.97	0.94	0.95	1028	8	0.94	0.95	0.95	974	9	0.94	0.94	0.94	1009	accuracy			0.96	10000	macro avg	0.96	0.96	0.96	10000	weighted avg	0.96	0.96	0.96	10000																																																									
	precision	recall	f1-score	support																																																																																																																																
0	0.97	0.98	0.97	980																																																																																																																																
1	0.98	0.98	0.98	1135																																																																																																																																
2	0.95	0.95	0.95	1032																																																																																																																																
3	0.94	0.94	0.94	1010																																																																																																																																
4	0.95	0.96	0.96	982																																																																																																																																
5	0.95	0.94	0.94	892																																																																																																																																
6	0.97	0.96	0.97	958																																																																																																																																
7	0.97	0.94	0.95	1028																																																																																																																																
8	0.94	0.95	0.95	974																																																																																																																																
9	0.94	0.94	0.94	1009																																																																																																																																
accuracy			0.96	10000																																																																																																																																
macro avg	0.96	0.96	0.96	10000																																																																																																																																
weighted avg	0.96	0.96	0.96	10000																																																																																																																																

(b) (10 points) Comment on the performance of the classifier and give your explanation why some of them perform better than the others.

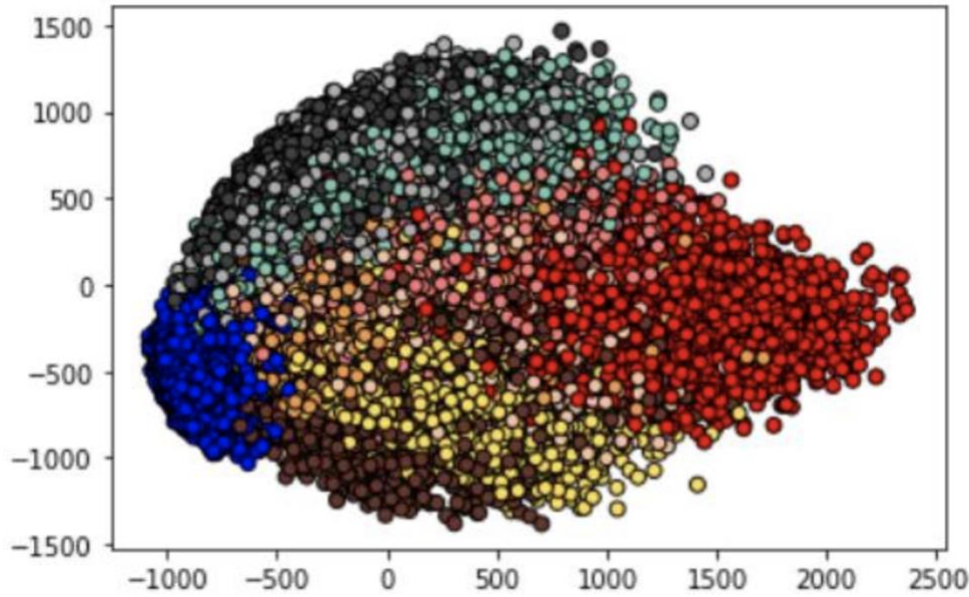
In summary, the performance accuracy is as follows:

- KNN - 88%
- Logistic Regression - 92%
- SVM - 91%
- **Kernel SVM - 95% (Non-linear relationship)**
- **Neural Nets - 96% (Non-linear relationship)**

Neural nets and Radial kernel SVM have outperformed (in both accuracy and precision) KNN, logistic regression and linear SVM classifiers in classification of MNIST handwritten digits.

This may be because the data has a lot of non-linear relationship among the pixels. KNN, logistic and SVM predict directly from the feature extraction.

**Multilayer Perceptron (MLP)** and **radial kernel SVM** captures any **non-linear function/relationship**. So, it is more suitable for learning non-linear models. Neural nets seem to have learnt the complex elements in the images. Also the training time is lower for MLP in comparison to logistic regression which took a lot of time.



This is the 2D PCA of the images and seems like we can draw 9 lines to classify 10 parts of these clusters for the most part. Hence the linear classifiers could classify with 88-92% accuracy. The extra 5% of increased accuracy comes from the flexibility of the non-linear classifier which could better separate the overlapping data points.

## 2. Naive Bayes for spam filtering. (35 points)

In this problem, we will use the Naive Bayes algorithm to fit a spam filter by hand. This will enhance your understanding to Bayes classifier and build intuition. This question does not involve any programming but only derivation and hand calculation.

Spam filters are used in all email services to classify received emails as “Spam” or “Not Spam”. A simple approach involves maintaining a vocabulary of words that commonly occur in “Spam” emails and classifying an email as “Spam” if the number of words from the dictionary that are present in the email is over a certain threshold. We are given the vocabulary consists of 15 words

$$V = \{\text{secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza}\}.$$

We will use  $V_i$  to represent the  $i$ th word in  $V$ . As our training dataset, we are also given 3 example spam messages,

- million dollar offer
- secret offer today
- secret is secret

and 3 example non-spam messages

- low price for valued customer
- play secret sports today
- low price pizza

Recall that the Naive Bayes classifier assumes the probability of an input depends on its input feature. The feature for each sample is defined as  $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}]^T$ ,  $i = 1, \dots, m$  and the class of the  $i$ th sample is  $y^{(i)}$ . In our case the length of the input vector is  $d = 15$ , which is equal to the number of words in the vocabulary  $V$ . Each entry  $x_j^{(i)}$  is equal to the number of times word  $V_j$  occurs in the  $i$ -th message.



1. (5 points) Calculate class prior  $\mathbb{P}(y = 0)$  and  $\mathbb{P}(y = 1)$  from the training data, where  $y = 0$  corresponds to spam messages, and  $y = 1$  corresponds to non-spam messages. Note that these class prior essentially corresponds to the frequency of each class in the training sample. Write down the feature vectors for each spam and non-spam messages.
2. (15 points) In the Naive Bayes model, assuming the keywords are independent of each other (this is a simplification), the likelihood of a sentence with its feature vector  $x$  given a class  $c$  is given by

$$\mathbb{P}(x|y = c) = \prod_{k=1}^d \theta_{c,k}^{x_k}, \quad c = \{0, 1\}$$

where  $0 \leq \theta_{c,k} \leq 1$  is the probability of word  $k$  appearing in class  $c$ , which satisfies

$$\sum_{k=1}^d \theta_{c,k} = 1, \quad c = \{0, 1\}.$$

Given this, the complete log-likelihood function for our training data is given by

$$\ell(\theta_{0,1}, \dots, \theta_{0,d}, \theta_{1,1}, \dots, \theta_{1,d}) = \sum_{i=1}^m \sum_{k=1}^d x_k^{(i)} \log \theta_{y^{(i)},k}$$

Calculate the maximum likelihood estimates of  $\theta_{0,1}$ ,  $\theta_{0,7}$ ,  $\theta_{1,1}$ ,  $\theta_{1,15}$  by maximizing the log-likelihood function above.

(Hint: We are solving a constrained maximization problem and you will need to introduce Lagrangian multipliers and consider the Lagrangian function.)

3. (15 points) Given a test message “today is secret”, using the Naive Bayes classifier that you have trained in Part (a)-(b), to calculate the posterior and decide whether it is spam or not spam.

$V = \{ \text{secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza} \}$   $k=15$

1. Million dollar offer :  $x^1 = [0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0]$

2. Secret offer today :  $x^2 = [\underline{1} \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0]$

3. Secret is Secret :  $x^3 = [\underline{2} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0]$

4. Low price for valued customer :  $x^4 = [0 \text{ } 0 \text{ } \underline{1} \text{ } \underline{1} \text{ } \underline{1} \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0]$

5. Play secret sports today :  $x^5 = [\underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } \underline{1} \text{ } 0 \text{ } 0]$

6. Low price pizza :  $x^6 = [0 \text{ } 0 \text{ } \underline{1} \text{ } \underline{1} \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } \underline{1}]$

$y_1 = y_2 = y_3 = 0$  (Spam)  $\mid$   $y_4 = y_5 = y_6 = 1$  (Not-spam)

① Calculate class Prior  $P(y=0) \& P(y=1)$

$P(y=0) = \frac{3}{6} = \frac{1}{2}$  ;  $P(y=1) = \frac{3}{6} = \frac{1}{2}$

②  $P(x|y=c) = \prod_{k=1}^{15} \theta_{c,k}^{x_k}$ ,  $c \in \{0,1\}$  where  $\theta_{c,k} = \begin{matrix} [0,1] \\ \text{Probability of word} \\ \text{'k' appearing in class 'c'} \end{matrix}$

$\sum_{k=1}^{15} \theta_{c,k} = 1, c \in \{0,1\}$

Log likelihood function of training data

$\downarrow$   
 $l(\theta_1, \theta_2, \dots, \theta_{15}) = \sum_{i=1}^6 \sum_{k=1}^{15} x_k^{(i)} \log \theta_{y^{(i)},k}$



Calculate Maximum Likelihood Estimates of  $\theta_{0,1}$   $\theta_{0,7}$   $\theta_{1,1}$   $\theta_{1,15}$

Solving Constrained maximization problem to find estimator  $\{\hat{\theta}_{c,k}\}$

$$\begin{aligned} \max_{\theta} \ell(\theta_{c,k}, c=0,1; k=1, \dots, 15) &= \sum_{i=1}^6 \sum_{k=1}^{15} x_k^i \log \theta_{y_i,k} \\ \text{Subject to } \sum_{k=1}^d \theta_{c,k} &= 1, c=\{0,1\} \end{aligned}$$

Consider Lagrangian Function to be minimized

$$\ell = - \left[ \sum_{i=1}^6 \sum_{k=1}^{15} x_k^i \log \theta_{y_i,k} + \mu_0 \sum_{k=1}^d (1 - \theta_{0,k}) + \mu_1 \sum_{k=1}^d (1 - \theta_{1,k}) \right]$$

Lagrangian for class ( $y=0$ )  $i = \{1, 2, 3\}$

$$L(x, \theta, \mu) = - \left[ \sum_{i=1}^3 \sum_{k=1}^{15} x_k^i \log \theta_{0,k} + \mu_0 \sum_{k=1}^{15} (1 - \theta_{0,k}) \right]$$

$$\inf_{\theta} L(x, \theta, \mu) \Rightarrow \frac{\partial \ell}{\partial \theta} = 0 \quad \forall \theta_{0,k}$$

$$\text{For } \theta_{0,1} \quad \frac{\sum_{i=1}^3 x_1^i}{\theta_{0,1}} - \mu_0 = 0 \Rightarrow \mu_0 = \frac{\sum_{i=1}^3 x_1^i}{\theta_{0,1}} \Rightarrow \theta_{0,1} = \frac{\sum_{i=1}^3 x_1^i}{\mu_0}$$

$$\text{Similarly all } \theta_{0,k} = \frac{\sum_{i=1}^3 x_k^i}{\mu_0}$$

$\mu$  are lagrangian parameters

Lagrangian for class (y=1) :  $i = \{4, 5, 6\}$

$$L(x, \theta, \mu) = - \left[ \sum_{i=4}^6 \sum_{k=1}^{15} x_k^i \log \theta_{ik} + \mu_1 \sum_{k=1}^{15} (1 - \theta_{ik}) \right]$$

$$\inf_{\theta} L(x, \theta, \mu) \Rightarrow \frac{\partial L}{\partial \theta} = 0 \quad \forall \theta_{ik}$$

$$\text{For } \theta_{11} \quad \frac{\sum_{i=4}^6 x_i^1}{\theta_{11}} - \mu_1 = 0 \Rightarrow \mu_1 = \frac{\sum_{i=4}^6 x_i^1}{\theta_{11}} \Rightarrow \theta_{11} = \frac{\sum_{i=4}^6 x_i^1}{\mu_1}$$

$$\text{Similarly all } \theta_{ik} = \frac{\sum_{i=4}^6 x_k^i}{\mu_1}$$

Splitting the Given constraints :

$$\sum_{k=1}^{15} \theta_{0k} = 1$$

$$\sum_{k=1}^{15} \frac{\sum_{i=1}^3 x_k^i}{\mu_0} = 1$$

$$\mu_0 = \sum_{k=1}^{15} \sum_{i=1}^3 x_k^i$$

$$\mu_0 = 9$$

$$\sum_{k=1}^{15} \theta_{1k} = 1$$

$$\sum_{k=1}^{15} \frac{\sum_{i=4}^6 x_k^i}{\mu_1} = 1$$

$$\mu_1 = \sum_{k=1}^{15} \sum_{i=4}^6 x_k^i$$

$$\mu_1 = 12$$

$\theta_{01} = \frac{\sum_{i=1}^3 x_i^0}{\mu_0} = \frac{3}{9} = \frac{1}{3}$	$\theta_{11} = \frac{\sum_{i=4}^6 x_i^0}{\mu_0} = \frac{1}{12}$
$\theta_{07} = \frac{\sum_{i=1}^3 x_i^1}{\mu_0} = \frac{1}{9}$	$\theta_{15} = \frac{\sum_{i=4}^6 x_i^1}{\mu_0} = \frac{1}{12}$

③  $z = \text{today is secret}$   
vectorizing it as below

$$z = [\overset{1}{1} 0 0 0 0 0 \overset{7}{1} 0 0 0 0 \overset{11}{1} 0 0 0 0]$$

$$P(z|y=0) = \prod \theta_{ok}^{z_k} = \theta_{01} \times \theta_{07} \times \theta_{011} = \frac{1}{9} \times \frac{1}{9} \times \frac{1}{9} = \frac{1}{243}$$

$$P(z|y=1) = \prod \theta_{1k}^{z_k} = \theta_{11} \times \theta_{17} \times \theta_{111} = \frac{1}{12} \times \frac{1}{12} \times 0 = 0$$

From (a) we know  $p(y=0) = p(y=1) = 1/2$

Posterior 
$$p(y=0|z) = \frac{P(z|y=0) \cdot p(y=0)}{P(z|y=0)p(y=0) + P(z|y=1)p(y=1)}$$

$$P(y=1|z) = \frac{P(z|y=1) \cdot P(y=1)}{P(z|y=0)P(y=0) + P(z|y=1)P(y=1)}$$

$$P(y=0|z) = \frac{1/243}{1/243 + 0} = 1 \quad ; \quad P(y=1|z) = \frac{0}{1/243 + 0} = 0$$

So the message "today is secret" is a  
class 0  $\Rightarrow$  SPAM message

Estimated  $\theta_{ck}$  = Probability of word 'k' appearing in class 'c'

$c=0$

$\hat{\theta}_{0,1} = \frac{3}{9} = \frac{1}{3}$	$\theta_{0,6} = 0$	$\theta_{0,11} = \frac{1}{9}$
$\theta_{0,2} = \frac{2}{9}$	$\theta_{0,7} = \frac{1}{9}$	$\theta_{0,12} = 0$
$\theta_{0,3} = 0$	$\theta_{0,8} = \frac{1}{9}$	$\theta_{0,13} = 0$
$\theta_{0,4} = 0$	$\theta_{0,9} = \frac{1}{9}$	$\theta_{0,14} = 0$
$\theta_{0,5} = 0$	$\theta_{0,10} = 0$	$\theta_{0,15} = 0$

$c=1$

$\theta_{1,1} = \frac{1}{12}$	$\theta_{1,6} = \frac{1}{12}$	$\theta_{1,11} = 0$
$\theta_{1,2} = 0$	$\theta_{1,7} = \frac{1}{12}$	$\theta_{1,12} = \frac{1}{12}$
$\theta_{1,3} = \frac{2}{12} = \frac{1}{6}$	$\theta_{1,8} = 0$	$\theta_{1,13} = \frac{1}{12}$
$\theta_{1,4} = \frac{2}{12} = \frac{1}{6}$	$\theta_{1,9} = 0$	$\theta_{1,14} = 0$
$\theta_{1,5} = \frac{1}{12}$	$\theta_{1,10} = \frac{1}{12}$	$\theta_{1,15} = \frac{1}{12}$

### 3. Neural networks. (Bonus: 3 points)

Consider a simple two-layer network in the lecture slides. Given  $n$  training data  $(x^i, y^i)$ ,  $i = 1, \dots, n$ , the cost function used to training the neural networks

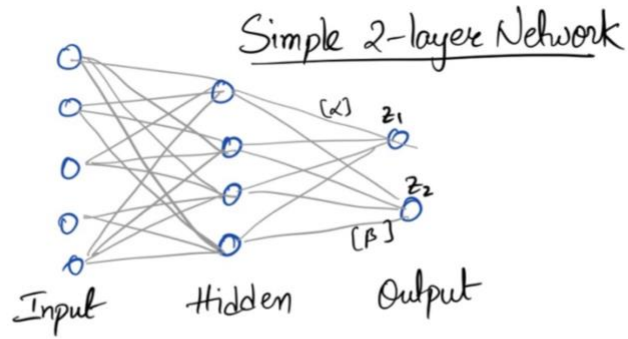
$$\ell(w, \alpha, \beta) = \sum_{i=1}^n (y^i - \sigma(w^T z^i))^2$$

where  $\sigma(x) = 1/(1 + e^{-x})$  is the sigmoid function,  $z^i$  is a two-dimensional vector such that  $z_1^i = \sigma(\alpha^T x^i)$ , and  $z_2^i = \sigma(\beta^T x^i)$ . Show the that the gradient is given by

$$\frac{\partial \ell(w, \alpha, \beta)}{\partial w} = - \sum_{i=1}^n 2(y^i - \sigma(u^i))\sigma(u^i)(1 - \sigma(u^i))z^i,$$

where  $u^i = w^T z^i$ . Also show the gradient of  $\ell(w, \alpha, \beta)$  with respect to  $\alpha$  and  $\beta$  and write down their expression.





Cost function  $l(w, \alpha, \beta) = \sum_{i=1}^n (y^i - \sigma(w^T z^i))^2$

$\sigma(x) = \frac{1}{1+e^{-x}}$  [Sigmoid function]

$$\frac{\partial l(w, \alpha, \beta)}{\partial w} = \sum_{i=1}^n 2(y^i - \sigma(w^T z^i)) \times \frac{\partial \sigma(w^T z^i)}{\partial w}$$

If  $u^i = w^T z^i$ ,  $\frac{\partial \sigma(u^i)}{\partial w} = \frac{\partial \sigma(u^i)}{\partial u^i} \times \frac{\partial u^i}{\partial w}$

Using chain rule for sigmoid

$$\begin{aligned} \frac{d}{dx} \left( \frac{1}{1+e^{-x}} \right) &= \frac{d}{dx} (1+e^{-x})^{-1} \\ &= -1 (1+e^{-x})^{-2} (-e^{-x}) \end{aligned}$$

$$\left| \frac{\partial u^i}{\partial w} = z^i \right.$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1}{1+e^{-x}}\right) \times \left(\frac{e^{-x}}{1+e^{-x}}\right)$$

$$= \left(\frac{1}{1+e^{-x}}\right) \left(1 - \frac{1}{1+e^{-x}}\right)$$

THUS,  $\frac{\partial \sigma(u)}{\partial u} = \sigma(u) \times (1 - \sigma(u))$

so  $\boxed{\frac{\partial l(w, \alpha, \beta)}{\partial w} = \sum_{i=1}^n 2(y^i - \sigma(u^i)) \sigma(u^i) (1 - \sigma(u^i)) z^i}$

For derivative w.r.t  $\alpha, \beta$ .

$$\frac{\partial l(w, \alpha, \beta)}{\partial \alpha} = \sum_{i=1}^n 2(y^i - \sigma(u^i)) \times \frac{\partial(\sigma(u^i))}{\partial u^i} \times \frac{\partial u^i}{\partial \alpha}$$

$$\frac{\partial u^i}{\partial \alpha} = \frac{\partial w^T(z^i)}{\partial \alpha} = \frac{\partial(w^T \sigma(\alpha x^i))}{\partial \alpha} = w^T \frac{\partial \sigma(\alpha x^i)}{\partial \alpha}$$

$$= w^T \frac{\partial}{\partial \alpha} \left( \frac{1}{1+e^{-\alpha x^i}} \right) = w^T (-1) (1+e^{-\alpha x^i})^{-2} \times (x^i) e^{-\alpha x^i}$$

$$= w^T x^i \sigma(\alpha x^i) (1 - \sigma(\alpha x^i))$$

$$\frac{\partial l(w, \alpha, \beta)}{\partial \alpha} = \sum_{i=1}^n 2(y^i - \sigma(u^i)) \times \sigma(u^i)(1 - \sigma(u^i)) \times w^T x^i \sigma(\alpha x^i)(1 - \sigma(\alpha x^i))$$

Similarly for  $\beta$

$$\frac{\partial l(w, \alpha, \beta)}{\partial \beta} = \sum_{i=1}^n 2(y^i - \sigma(u^i)) \times \sigma(u^i)(1 - \sigma(u^i)) \times w^T x^i \sigma(\beta x^i)(1 - \sigma(\beta x^i))$$

Teja 07/09/21