CS 440/ECE 448 Artificial Intelligence
Assignment 3: Naive Bayes Classification
Section R3 (3 Credit)

Haoen CUI[*], *Guohao (Holden) DOU*[†], *Chuchao LUO*[‡]

DUE November 27, 2017

Contents

[*]Haoen CUI's Email hcui10@illinois.edu
[†]Guohao (Holden) DOU's Email gdou2@illinois.edu
[‡]Chuchao LUO's Email chuchao2@illinois.edu

1 Part 1: Digit Classification

1.1 Single Pixels as Features (For Everybody)

1.1.1 Implementation

We treated this problem as a special case of *pixel group as features* where the pixel groups are simply disjoint and of size 1 by 1. Please see the next section for details (e.g choice of Laplace smoothing constant, classification rates, and confusion matrices).

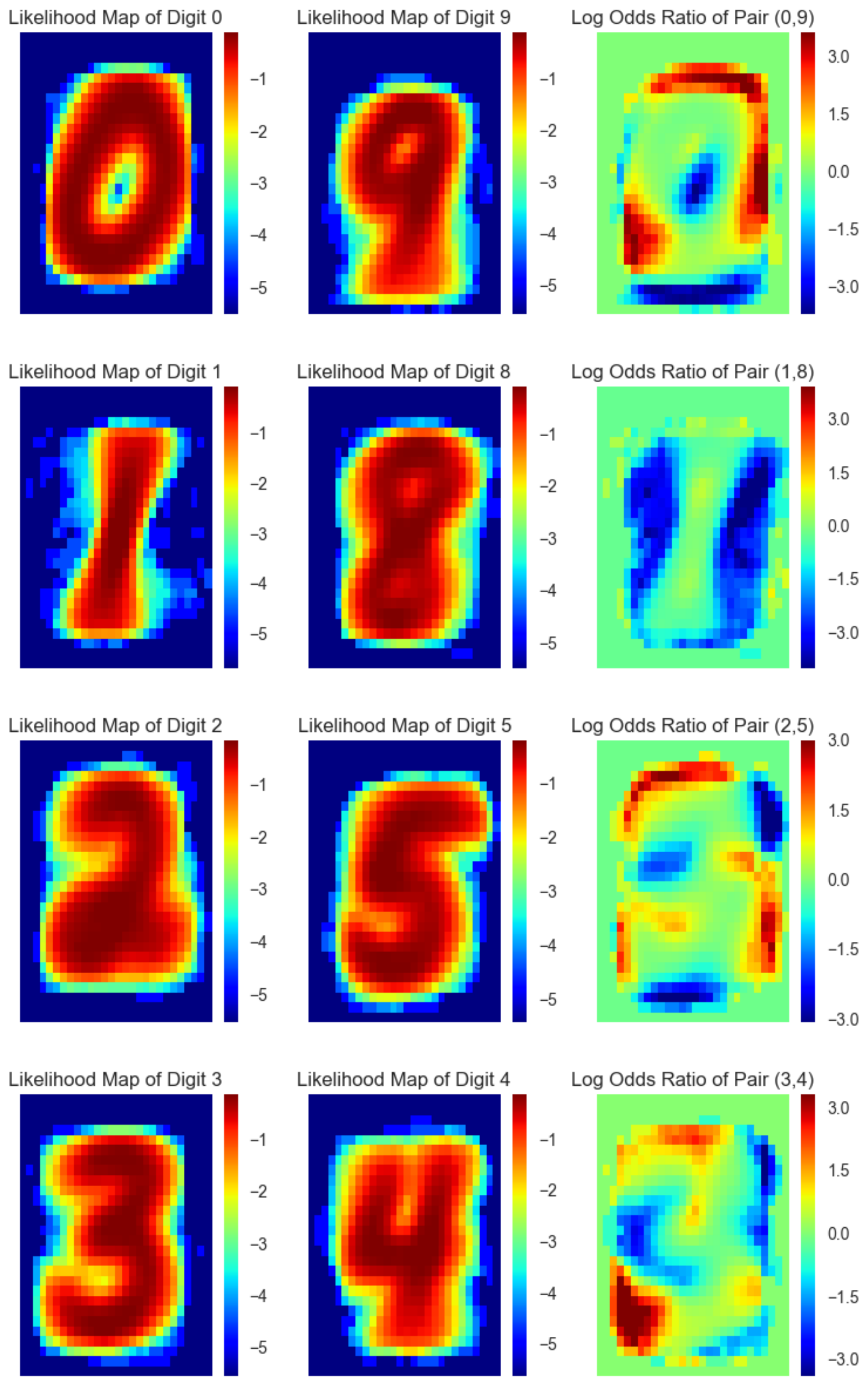
1.1.2 Posterior Probabilities: Highest and Lowest

Test examples with largest and smallest posterior probabilities for each digit are shown below.

```
## ['======================== Digit = 0 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '              ####           |                             ',
##   '            #######          |         ##########          ',
##   '           ##########        |        ############         ',
##   '          ############       |      ###############        ',
##   '         ######## ####       |     #################       ',
##   '        ########  #####      |     ##########  ######      ',
##   '        #####      ####      |     ########      ####      ',
##   '       #####        ###      |       #####       #####     ',
##   '       #####        ###      |       ####         #####    ',
##   '       ####         ###      |       ####          ####    ',
##   '       ####         ###      |       ####          ####    ',
##   '       ####         ###      |       ####          ####    ',
##   '       ####        ####      |       ####          ####    ',
##   '       ####       ####       |       #####         ####    ',
##   '       #####     #####       |       ######        ####    ',
##   '       ######   ######       |        #####        ####    ',
##   '        #############        |         ######      ####    ',
##   '        ############         |          ###############    ',
##   '         ##########          |           ##############    ',
##   '           ######            |            ############     ',
##   '                             |             ##########      ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 1 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |            ###              ',
##   '                             |            ####             ',
##   '              ###            |            ####             ',
##   '              ####           |            ####             ',
##   '              ####           |            ####             ',
##   '              ####           |            ####             ',
##   '              ####           |            ####             ',
##   '              ###            |            ####             ',
##   '             ####            |            ####             ',
##   '             ####            |           ######            ',
##   '             ####            |           ######            ',
##   '             ####            |           ## ###            ',
##   '             ###             |              ####           ',
##   '            ####             |               ####          ',
##   '            ####             |               ####          ',
##   '            ####             |                #####        ',
##   '           ####              |          ###########        ',
##   '           ####              |         ############        ',
##   '          #####              |         ############        ',
##   '          #####              |          ##########         ',
##   '          #####              |                             ',
##   '           ###               |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 2 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '          ######             |                             ',
##   '          ########           |                             ',
##   '         ##########          |                             ',
##   '         ####  #####         |                             ',
##   '         ##     ####         |              #########      ',
##   '                 ###         |         ##############      ',
##   '                 ###         |       #################     ',
##   '                  ##         |     ###################     ',
##   '                  ##         |    ####################     ',
##   '                  ##         |    ###########  #######     ',
##   '                 ###         |    ##########  #######      ',
##   '                 ###         |    ####      ########       ',
##   '           ##### ###         |    ###      #########       ',
##   '         ##########          |    ###     ##########       ',
##   '        ###########          |    ##      ########         ',
##   '        #############        |            ########         ',
##   '        ###  ###########     |            #########        ',
##   '        #########  ####      |            ###########      ',
##   '        ########             |             ###########     ',
##   '         #####               |               #########     ',
##   '                             |                #######      ',
##   '                             |                   ####      ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 3 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '         ########            |                             ',
##   '         ##########          |                             ',
##   '            ########         |                             ',
##   '                ####         |          ############       ',
##   '                 ####        |        ###############      ',
##   '                ####         |        ################     ',
##   '              #####          |      #####         ####     ',
##   '             #####           |      ####          ####     ',
##   '            #####            |      ###          ####      ',
##   '           #######           |                  ####       ',
##   '           ########          |                 #####       ',
##   '           ##  ####          |               ######        ',
##   '                ###          |               #####         ',
##   '                ###          |               #####         ',
##   '                ###          |                 ###         ',
##   '               ####          |                ###          ',
##   '              #####          |              #####          ',
##   '     #############           |             ######          ',
##   '     ############            |            ######           ',
##   '      ##########             |           #####             ',
##   '                             |         #######             ',
##   '                             |        #######              ',
##   '                             |        #####                ',
##   '                             |                             ']]
## ['======================== Digit = 4 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |        ###                  ',
##   '           #       ##        |        ###                  ',
##   '          ###      ##        |        ###    ####          ',
##   '          ###      ##        |        ###    ####          ',
##   '         ####    ####        |       ####    ####      #   ',
##   '        ####     ####        |       ####    ####     ###  ',
##   '        ####     ####        |       ####    ####     ###  ',
##   '        ###      ####        |       ####    ####     ###  ',
##   '        ###     #####        |       ####    ####    ####  ',
##   '        ##    #######        |       ####     ###  ######  ',
##   '      #############          |       ####     ##########   ',
##   '       ############          |       #### #############    ',
##   '          ########           |       ################      ',
##   '              ###            |       ##############        ',
##   '              ###            |         ###########         ',
##   '              ###            |                 ####        ',
##   '             ####            |                 ####        ',
##   '             ###             |                 ####        ',
##   '             ###             |                  ###        ',
##   '             ###             |                   ##        ',
##   '             ###             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 5 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                   ####      |       #####                 ',
##   '            ###########      |      #######                ',
##   '            ##########       |      #######                ',
##   '            ########         |      #######                ',
##   '           ####              |      ######                 ',
##   '           ###               |      #######                ',
##   '          ###                |       #########             ',
##   '          ######             |           #######           ',
##   '         ########            |              ######         ',
##   '         ########            |                #####        ',
##   '          #    ###           |                   ####      ',
##   '               ###           |                    ####     ',
##   '               ###           |                     ####    ',
##   '                ##           |            ####     #####   ',
##   '       ###      ###          |            ##############   ',
##   '       ###     ###           |             #############   ',
##   '       ###     ###           |              ############   ',
##   '       ####   ####           |                   ######    ',
##   '        #########            |                             ',
##   '           #####             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 6 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                ###          |        ###                  ',
##   '              #####          |        ####                 ',
##   '             #####           |       ####                  ',
##   '            #####            |       ####                  ',
##   '            #####            |       ###                   ',
##   '           #####             |       ###        ###        ',
##   '          #####              |       ###        #####      ',
##   '          ####               |      ####       ######      ',
##   '          ####               |      ####       ######      ',
##   '         #####               |      ####       #######     ',
##   '         ####                |      ###        #######     ',
##   '         ####   #####        |      ###        #######     ',
##   '        #####  ######        |      ####       #######     ',
##   '        #############        |      ####       #######     ',
##   '        #### ########        |       ####      ######      ',
##   '        #############        |       ####       #####      ',
##   '        ###########          |        ####      ####       ',
##   '         #########           |         ##############      ',
##   '          #######            |           ##########        ',
##   '            ####             |            #######          ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 7 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |           ######    ####    ',
##   '        #### ######          |           ###############   ',
##   '        ############         |           ###############   ',
##   '        ############         |            ##############   ',
##   '          ###    ###         |                  ######     ',
##   '                ####         |                 #######     ',
##   '                ####         |                 #####       ',
##   '               ####          |               ######        ',
##   '               ####          |          ##########         ',
##   '               ####          |         ##########          ',
##   '              ####           |         #############       ',
##   '              ####           |         #############       ',
##   '              ###            |         #############       ',
##   '             ####            |        ######               ',
##   '             ###             |        #####                ',
##   '            ####             |       #####                 ',
##   '            ###              |      #####                  ',
##   '           ###               |      #####                  ',
##   '           ###               |      #####                  ',
##   '           ###               |      ###                    ',
##   '           ###               |                             ',
##   '                             |                             ']]
## ['======================== Digit = 8 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                ####         |          #####              ',
##   '              #######        |        #########            ',
##   '            #########        |      ###########            ',
##   '           ##########        |      #############          ',
##   '          ###### ####        |      ############# #        ',
##   '          ####   ####        |      ################       ',
##   '          ###     ####       |      ######   #######       ',
##   '          ###    #####       |       ##############        ',
##   '          ############       |       ############          ',
##   '          ##########         |        ###########          ',
##   '           #######           |         ###########         ',
##   '          ######             |          #############      ',
##   '         ########            |          #############      ',
##   '         ########            |          ###############    ',
##   '        ####  ####           |            ##############   ',
##   '        ####  ####           |            ##############   ',
##   '        #### ####            |             #############   ',
##   '        #########            |               ###########   ',
##   '         ########            |                ##########   ',
##   '          ######             |                   #####     ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ']]
## ['======================== Digit = 9 ========================',
##  '---- Highest Posterior ------------- Lowest Posterior -----',
##  ['                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |                             ',
##   '                             |               #######       ',
##   '            ######           |           ############      ',
##   '           ########          |         ###############     ',
##   '         ###########         |      ##################     ',
##   '        #####   ####         |     ############## ####     ',
##   '        ####   #####         |    ######           ###     ',
##   '       ####    #####         |    #####            ###     ',
##   '       ####    #####         |    ####           #####     ',
##   '       ####  ######          |    ####        ########     ',
##   '        ###########          |    ####################     ',
##   '         #########           |    ####################     ',
##   '          ########           |     #############  ####     ',
##   '             ####            |       ########     ####     ',
##   '             ####            |                    ####     ',
##   '             ###             |                    ####     ',
##   '            ####             |                    ####     ',
##   '            ####             |                    ####     ',
##   '            ###              |                    ####     ',
##   '            ###              |                    ####     ',
##   '            ###              |                    ####     ',
##   '            ###              |                             ',
##   '                             |                             ']]
```

1.1.3 Visualization of Likelihoods and Odds Ratios

We interpreted `pairs of digits that have the highest confusion rates` as `pairs without misclassifications`. Hence, we chose pairs `(0, 9)`, `(1, 8)`, `(2, 5)`, and `(3, 4)` (randomly out of all eligible pairs). The plots are shown below. We tried several different color maps, but none of them match exactly with the example on the assignment page. The following are the closest version we can get. Though the color maps are different, they convey the same information. Also, note that the smoother used to generate these results will be described as a special case with disjoint kernel size `(1, 1)` in the next section.



1.2 Pixel Groups as Features (For Four-Credit Students)

1.2.1 Implementation

- See `deserializer.py` for data loading utilities.
- See `fc_utils.py` for the "featurize" function. It basically does convolution of different stride and kernel size over the 28 * 28 matrix.
- See `train_fc.py`, `cv_fc.py`, `test_fc.py` ("fc" stands for four-credit) for detailed implementation.

1.2.2 Choice of Smoothing Constant: 10-Fold Cross Validation

We used 10-fold cross validation to select the smoothing constant. To be specific,

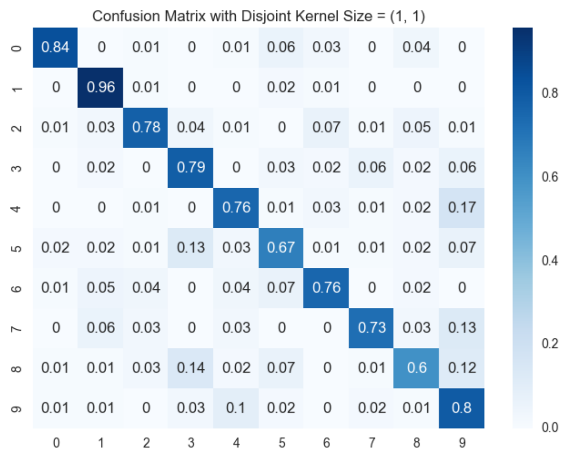
- We randomly assigned a fold number (out of 1 to 10) to each record in the training set.
- Then, we iterate through all potential smoothers on the each fold.
- For a given smoother and a selected fold number, the selected fold will serve as the validation set and the remaining training set will be used to train the Naive Bayes model. We can calculate a performance measure (in this case: overall misclassification rate). Thus, we will get 10 measures for each smoother.
- We summarized the performance of each smoother using the average of its 10 measures.
- Finally, we selected the best smoother based on the aggregated averages.
- **IMPORTANT NOTICE**: testing set is not being used in the entire cross validation phase. It is only used once for testing purposes. Relevant results (e.g. plots and prototypicals) are generated based on these tests.

We considered smoothers in the list `smoothers = [0.1, 0.5, 1, 2, 4, 8]`. The best smoothers selected for different kernel sizes are shown below. The entire assignment used the same cross-validation methodology. Hence, we will not repeat this section again.

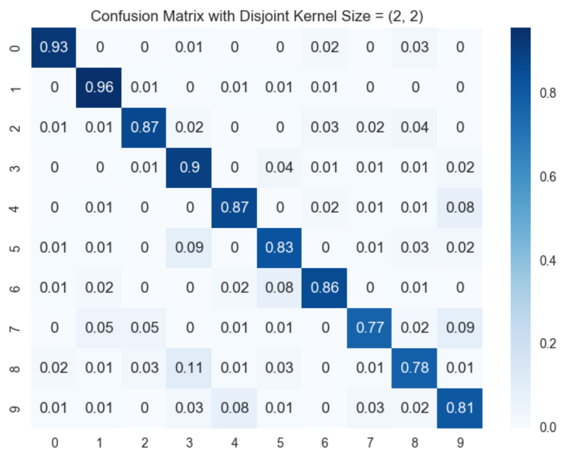
Table 1: Choice of Smoothing Constant Using 10-Fold Cross Validation

Kernel Type	Kernel Size	Best Smoother Value
Disjoint	(1, 1)	0.5
Disjoint	(2, 2)	0.1
Disjoint	(2, 4)	0.1
Disjoint	(4, 2)	0.1
Disjoint	(4, 4)	0.1
Overlapping	(2, 2)	0.1
Overlapping	(2, 4)	0.1
Overlapping	(4, 2)	0.1
Overlapping	(4, 4)	0.1
Overlapping	(2, 3)	0.1
Overlapping	(3, 2)	0.1
Overlapping	(3, 3)	0.1

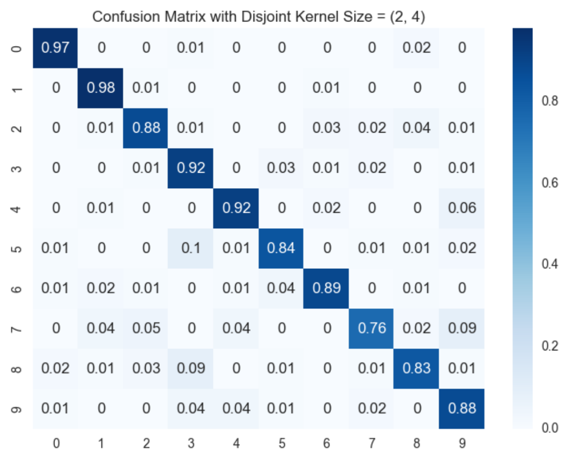
1.2.3 Accuracy on Test Set: Classification Rate and Confusion Matrix



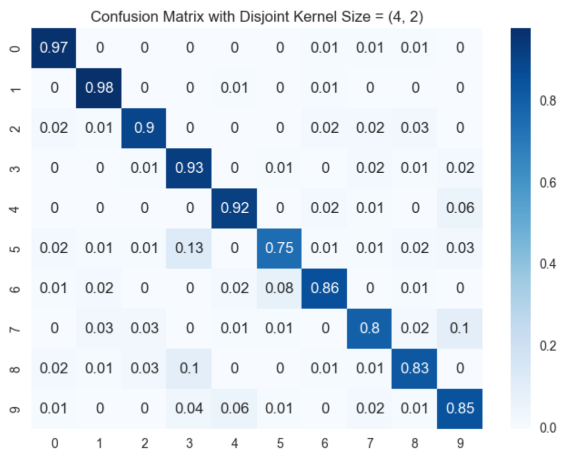
	0	1	2	3	4	5	6	7	8	9
0	0.84	0	0.01	0	0.01	0.06	0.03	0	0.04	0
1	0	0.96	0.01	0	0	0.02	0.01	0	0	0
2	0.01	0.03	0.78	0.04	0.01	0	0.07	0.01	0.05	0.01
3	0	0.02	0	0.79	0	0.03	0.02	0.06	0.02	0.06
4	0	0	0.01	0	0.76	0.01	0.03	0.01	0.02	0.17
5	0.02	0.02	0.01	0.13	0.03	0.67	0.01	0.01	0.02	0.07
6	0.01	0.05	0.04	0	0.04	0.07	0.76	0	0.02	0
7	0	0.06	0.03	0	0.03	0	0	0.73	0.03	0.13
8	0.01	0.01	0.03	0.14	0.02	0.07	0	0.01	0.6	0.12
9	0.01	0.01	0	0.03	0.1	0.02	0	0.02	0.01	0.8



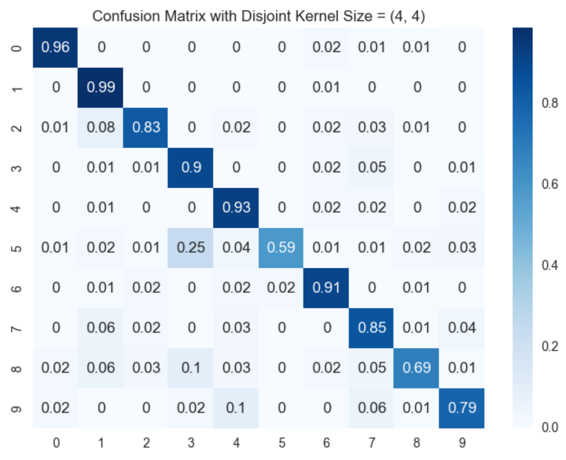
	0	1	2	3	4	5	6	7	8	9
0	0.93	0	0	0.01	0	0	0.02	0	0.03	0
1	0	0.96	0.01	0	0.01	0.01	0.01	0	0	0
2	0.01	0.01	0.87	0.02	0	0	0.03	0.02	0.04	0
3	0	0	0.01	0.9	0	0.04	0.01	0.01	0.01	0.02
4	0	0.01	0	0	0.87	0	0.02	0.01	0.01	0.08
5	0.01	0.01	0	0.09	0	0.83	0	0.01	0.03	0.02
6	0.01	0.02	0	0	0.02	0.08	0.86	0	0.01	0
7	0	0.05	0.05	0	0.01	0.01	0	0.77	0.02	0.09
8	0.02	0.01	0.03	0.11	0.01	0.03	0	0.01	0.78	0.01
9	0.01	0.01	0	0.03	0.08	0.01	0	0.03	0.02	0.81



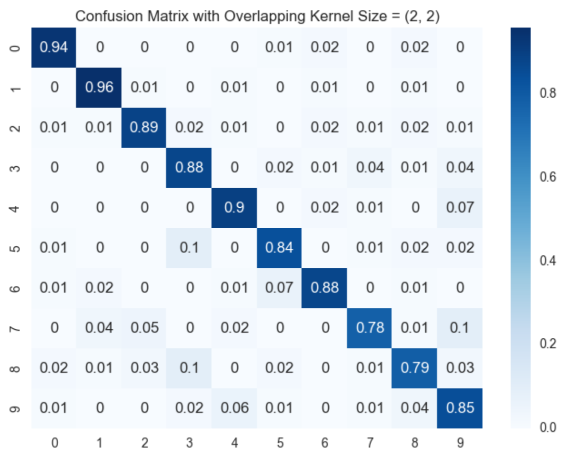
	0	1	2	3	4	5	6	7	8	9
0	0.97	0	0	0.01	0	0	0	0	0.02	0
1	0	0.98	0.01	0	0	0	0.01	0	0	0
2	0	0.01	0.88	0.01	0	0	0.03	0.02	0.04	0.01
3	0	0	0.01	0.92	0	0.03	0.01	0.02	0	0.01
4	0	0.01	0	0	0.92	0	0.02	0	0	0.06
5	0.01	0	0	0.1	0.01	0.84	0	0.01	0.01	0.02
6	0.01	0.02	0.01	0	0.01	0.04	0.89	0	0.01	0
7	0	0.04	0.05	0	0.04	0	0	0.76	0.02	0.09
8	0.02	0.01	0.03	0.09	0	0.01	0	0.01	0.83	0.01
9	0.01	0	0	0.04	0.04	0.01	0	0.02	0	0.88



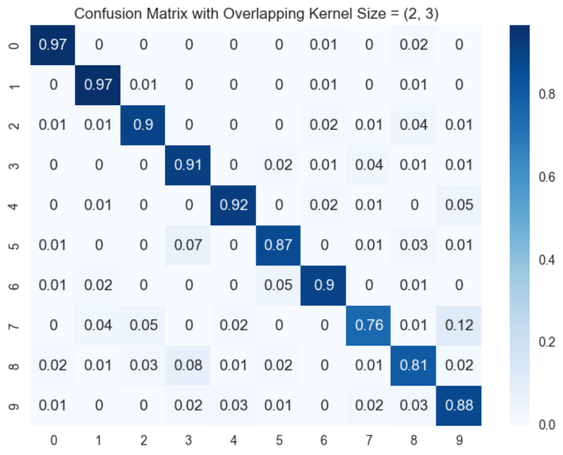
	0	1	2	3	4	5	6	7	8	9
0	0.97	0	0	0	0	0	0.01	0.01	0.01	0
1	0	0.98	0	0	0.01	0	0.01	0	0	0
2	0.02	0.01	0.9	0	0	0	0.02	0.02	0.03	0
3	0	0	0.01	0.93	0	0.01	0	0.02	0.01	0.02
4	0	0	0	0	0.92	0	0.02	0.01	0	0.06
5	0.02	0.01	0.01	0.13	0	0.75	0.01	0.01	0.02	0.03
6	0.01	0.02	0	0	0.02	0.08	0.86	0	0.01	0
7	0	0.03	0.03	0	0.01	0.01	0	0.8	0.02	0.1
8	0.02	0.01	0.03	0.1	0	0	0.01	0.01	0.83	0
9	0.01	0	0	0.04	0.06	0.01	0	0.02	0.01	0.85

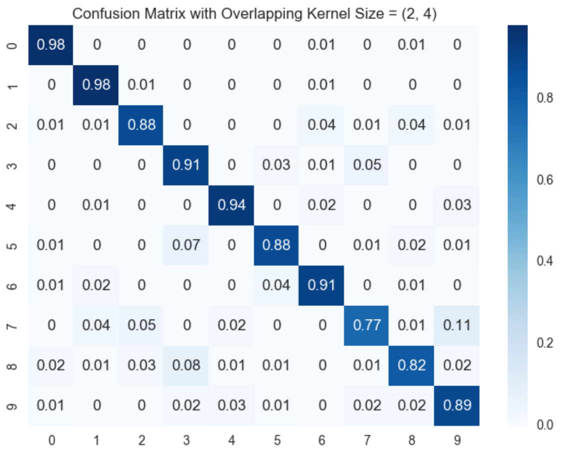


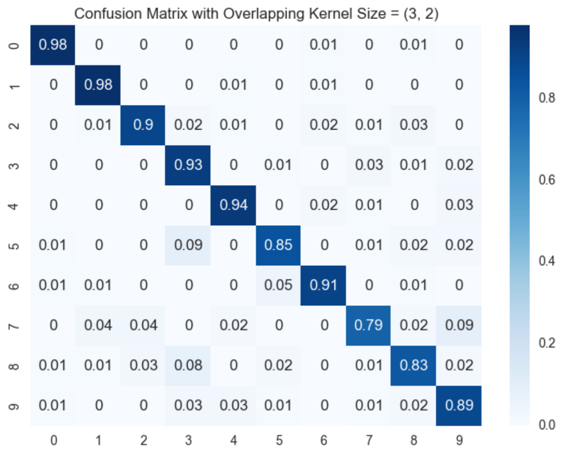
	0	1	2	3	4	5	6	7	8	9
0	0.96	0	0	0	0	0	0.02	0.01	0.01	0
1	0	0.99	0	0	0	0	0.01	0	0	0
2	0.01	0.08	0.83	0	0.02	0	0.02	0.03	0.01	0
3	0	0.01	0.01	0.9	0	0	0.02	0.05	0	0.01
4	0	0.01	0	0	0.93	0	0.02	0.02	0	0.02
5	0.01	0.02	0.01	0.25	0.04	0.59	0.01	0.01	0.02	0.03
6	0	0.01	0.02	0	0.02	0.02	0.91	0	0.01	0
7	0	0.06	0.02	0	0.03	0	0	0.85	0.01	0.04
8	0.02	0.06	0.03	0.1	0.03	0	0.02	0.05	0.69	0.01
9	0.02	0	0	0.02	0.1	0	0	0.06	0.01	0.79

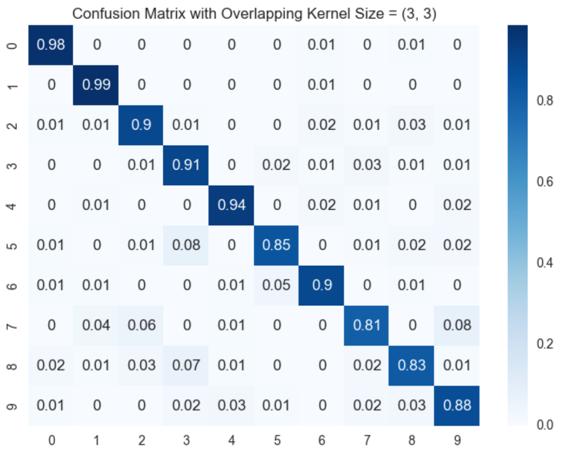


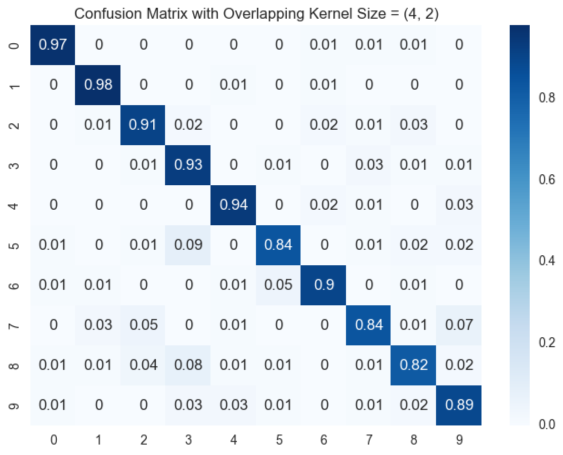
	0	1	2	3	4	5	6	7	8	9
0	0.94	0	0	0	0	0.01	0.02	0	0.02	0
1	0	0.96	0.01	0	0.01	0	0.01	0	0.01	0
2	0.01	0.01	0.89	0.02	0.01	0	0.02	0.01	0.02	0.01
3	0	0	0	0.88	0	0.02	0.01	0.04	0.01	0.04
4	0	0	0	0	0.9	0	0.02	0.01	0	0.07
5	0.01	0	0	0.1	0	0.84	0	0.01	0.02	0.02
6	0.01	0.02	0	0	0.01	0.07	0.88	0	0.01	0
7	0	0.04	0.05	0	0.02	0	0	0.78	0.01	0.1
8	0.02	0.01	0.03	0.1	0	0.02	0	0.01	0.79	0.03
9	0.01	0	0	0.02	0.06	0.01	0	0.01	0.04	0.85











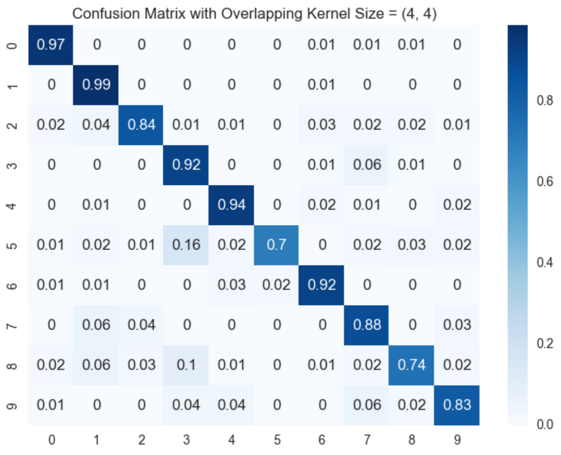


Table 2: Overall Accuracy on Different Kernels with Best Smoothers

Kernel Type	Kernel Size	Overall Accuracy
Disjoint	(1, 1)	77.0 %
Disjoint	(2, 2)	85.8 %
Disjoint	(2, 4)	88.6 %

Kernel Type	Kernel Size	Overall Accuracy
Disjoint	(4, 2)	87.9 %
Disjoint	(4, 4)	84.6 %
Overlapping	(2, 2)	87.1 %
Overlapping	(2, 4)	89.6 %
Overlapping	(4, 2)	90.2 %
Overlapping	(4, 4)	87.4 %
Overlapping	(2, 3)	88.8 %
Overlapping	(3, 2)	90.0 %
Overlapping	(3, 3)	90.0 %

1.2.4 Trends for Different Feature Sets

The following are some general trends we found,

- with same kernel size, overlapping kernels tend to perform better than disjoint kernels because overlapping ones contain more features.
- with the same kernel type, increase in kernel size does not necessary translate to better performance because at some point using additional features may be merely over-fitting.
- by trying out different kernels and tuning the smoother, one can achieve a much higher performance than merely using the default 1 by 1 pixels.

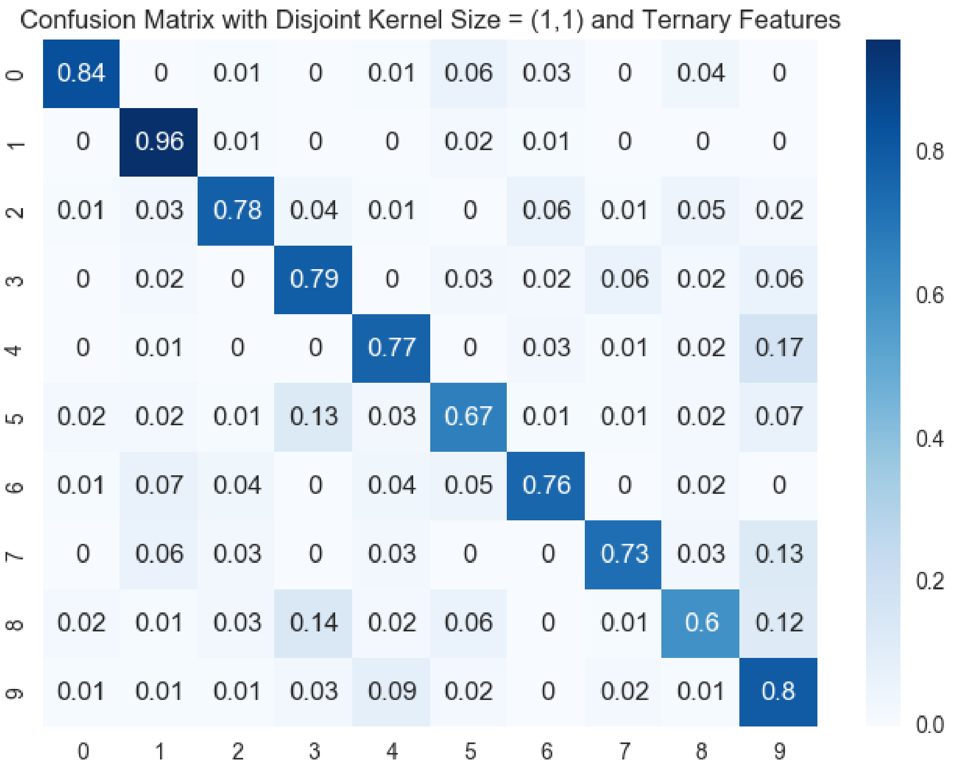
1.2.5 Running Time for Different Feature Sets

Both training and testing with disjoint features turn out to be significantly faster than those for overlapping features. This is because when using disjoint features, feature space is smaller than the default size 28 by 28. The larger the kernel size is, the fewer the number of features will be. On the other hand, overlapping kernels significantly boost our feature space. The smaller the kernel size is, the larger the feature space will be.

1.3 Extra Credit

1.3.1 Ternary Features

This part is relatively easy. We merely changed the way we read in files to incorporate ternary features. Then, we run the same process as the one we used for binary features with disjoint 1 by 1 kernel. Finally, we ended up with the same best smoother (`= 0.5`) and the resulting overall classification rate is 77.1 %. There is not much improvements.



	0	1	2	3	4	5	6	7	8	9
0	0.84	0	0.01	0	0.01	0.06	0.03	0	0.04	0
1	0	0.96	0.01	0	0	0.02	0.01	0	0	0
2	0.01	0.03	0.78	0.04	0.01	0	0.06	0.01	0.05	0.02
3	0	0.02	0	0.79	0	0.03	0.02	0.06	0.02	0.06
4	0	0.01	0	0	0.77	0	0.03	0.01	0.02	0.17
5	0.02	0.02	0.01	0.13	0.03	0.67	0.01	0.01	0.02	0.07
6	0.01	0.07	0.04	0	0.04	0.05	0.76	0	0.02	0
7	0	0.06	0.03	0	0.03	0	0	0.73	0.03	0.13
8	0.02	0.01	0.03	0.14	0.02	0.06	0	0.01	0.6	0.12
9	0.01	0.01	0.01	0.03	0.09	0.02	0	0.02	0.01	0.8

Figure 1: Ternary Confusion Matrix

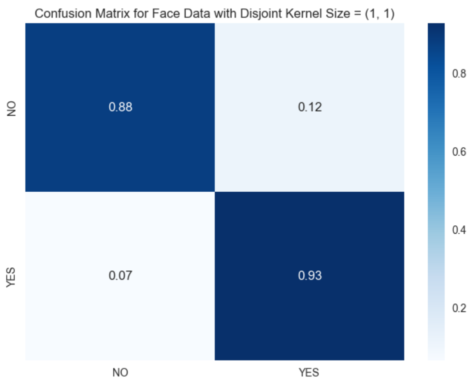
1.3.2 Naive Bayes Classifier on Face Data

Here we used exactly the same methodologies as the previous sections except for a few minor changes in matrix dimensions and sample sizes. A fixed smoother of 1 is used in this case.

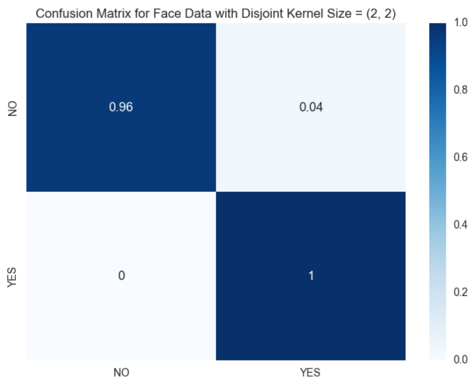
- See `deserializer_face.py` for data loading utilities.

- See `train_ec.py` and `test_ec.py` for detailed implementation.

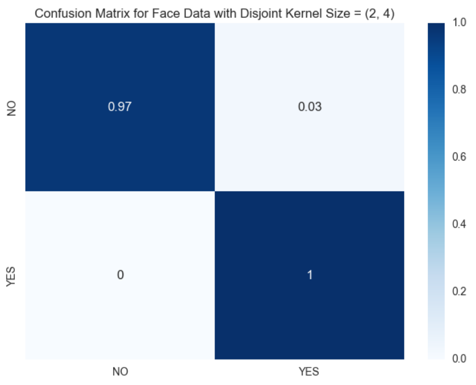
Detailed accuracy report on testing set is shown below.



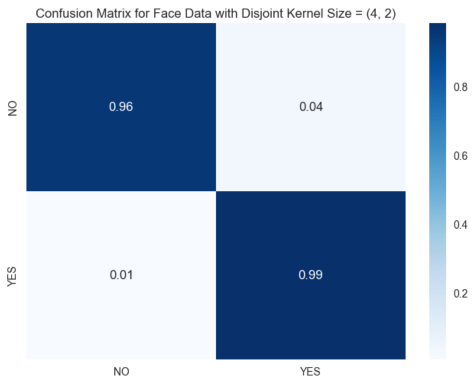
Confusion Matrix for Face Data with Disjoint Kernel Size = (1, 1)
NO
YES
0.88
0.12
0.07
0.93
NO
YES
0.8
0.6
0.4
0.2



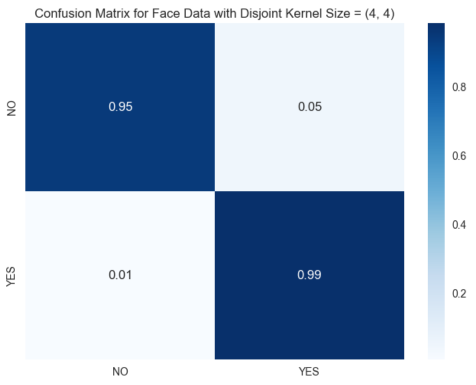
Confusion Matrix for Face Data with Disjoint Kernel Size = (2, 2)
NO
YES
0.96
0.04
0
1
NO
YES
1.0
0.8
0.6
0.4
0.2
0.0



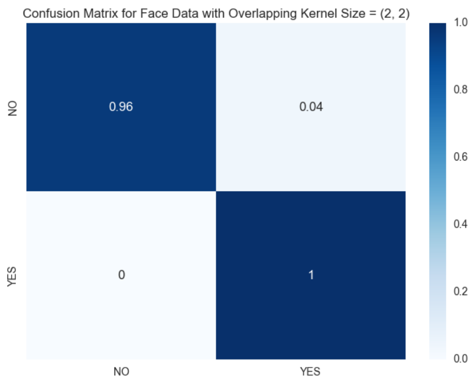
Confusion Matrix for Face Data with Disjoint Kernel Size = (2, 4)
NO
YES
0.97
0.03
0
1
NO
YES
1.0
0.8
0.6
0.4
0.2
0.0



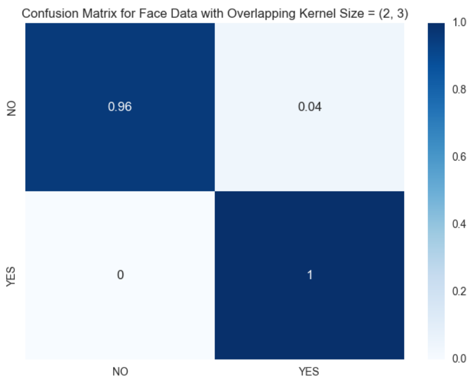
Confusion Matrix for Face Data with Disjoint Kernel Size = (4, 2)
NO
YES
0.96
0.04
0.01
0.99
NO
YES
0.8
0.6
0.4
0.2



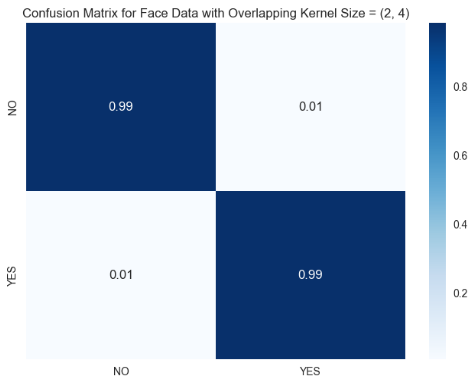
Confusion Matrix for Face Data with Disjoint Kernel Size = (4, 4)
NO
YES
0.95
0.05
0.01
0.99
NO
YES
0.8
0.6
0.4
0.2



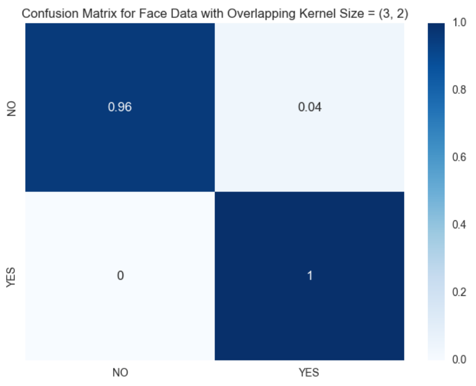
Confusion Matrix for Face Data with Overlapping Kernel Size = (2, 2)
NO
YES
0.96
0.04
0
1
NO
YES
1.0
0.8
0.6
0.4
0.2
0.0

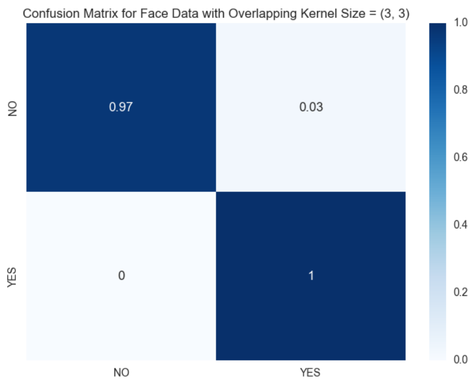


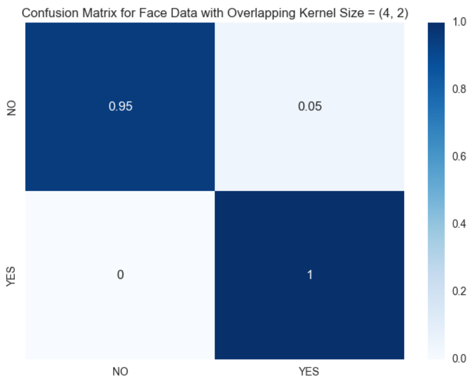
Confusion Matrix for Face Data with Overlapping Kernel Size = (2, 3)
NO
YES
0.96
0.04
0
1
NO
YES
1.0
0.8
0.6
0.4
0.2
0.0



Confusion Matrix for Face Data with Overlapping Kernel Size = (2, 4)
NO
YES
0.99
0.01
0.01
0.99
NO
YES
0.8
0.6
0.4
0.2







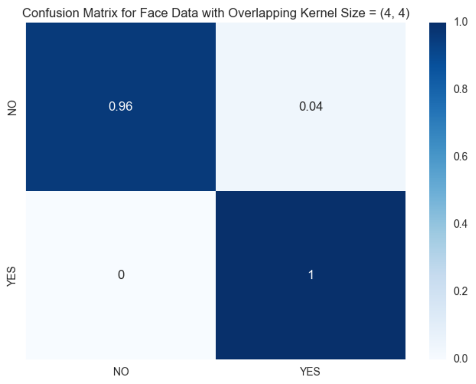


Table 3: Overall Accuracy with Different Kernels on Face Data

Kernel Type	Kernel Size	Overall Accuracy
Disjoint	(1, 1)	90.7 %
Disjoint	(2, 2)	98.0 %
Disjoint	(2, 4)	98.7 %
Disjoint	(4, 2)	97.3 %
Disjoint	(4, 4)	96.7 %
Overlapping	(2, 2)	98.0 %
Overlapping	(2, 4)	98.7 %
Overlapping	(4, 2)	97.3 %
Overlapping	(4, 4)	98.0 %
Overlapping	(2, 3)	98.0 %
Overlapping	(3, 2)	98.0 %
Overlapping	(3, 3)	98.7 %

2 Part 2: Audio Classification

2.1 Binary Classification: Hebrew Words of “Yes” and “No” (For Everybody)

2.1.1 Implementation

See `YESNO.py` for detailed implementation. Smoothing is implemented in `getData()` function.

2.1.2 Classification Rate and Confusion Matrix

We used the same logic of 10-fold cross validation to select the best smoother. The potential list we considered is `smoothers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`. The best smoother we selected is `best_smoother = 2` and its average classification rate on the 10 folds is 98.15%. With this smoother, the confusion matrix is shown below.

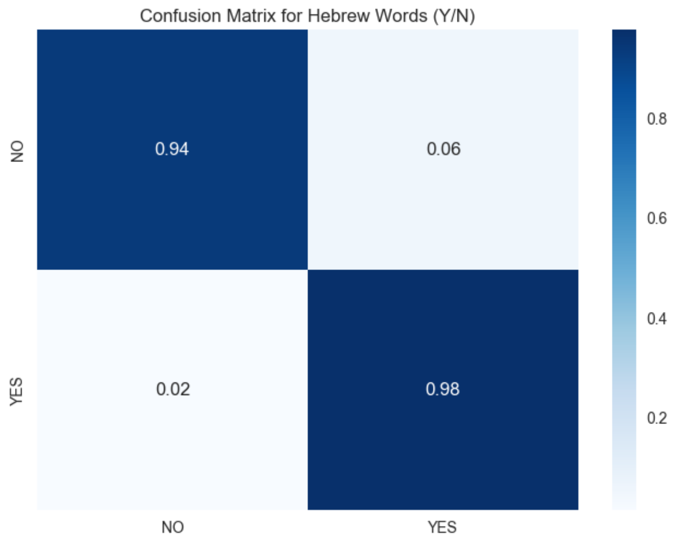


Figure 2: Confusion Matrix for Hebrew Words (Y/N)

2.2 Multi-Class Classification: Audio Digits 1-5 Spoken by Four Different Speakers (For Four-Credit Students)

2.2.1 Implementation

The same methodology as the previous sections. See `numberClassifier.py` for detailed implementation.

2.2.2 Accuracy: Classification Rate and Confusion Matrix

Again, smoother is chosen based on cross validation described above. The final overall accuracy with the best smoother is 85 % and the confusion matrix is shown below.

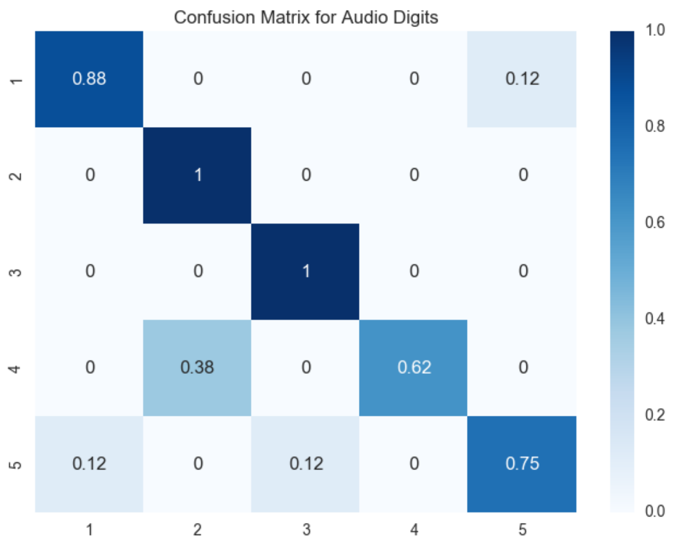


Figure 3: Confusion Matrix for Audio Digits

2.3 Extra Credit

2.3.1 Binary Classification on Unsegmented Data

We designed some utilities stored in `splitData.py`. The main `split()` function

- First strips garbage data at the beginning and the end of a voice data.
- Then iterates the rest of the data.
  - When the number of high energy grid is more than `startThreshold`, we start to record data.
  - When the number of high energy grid is less than `endThreshold`, or when the time limit is exceeded (10 columns), we stop recording data.
  - If the data is not long enough (less than 10 columns), use all zero column to pad two side of that data to 10 columns.

Using the functions in `splitData.py` we designed, we run the binary Naive Bayes classifier on the splitted data again. Meanwhile, smoother is selected using cross validation as described above. The overall accuracy is 96 % and the confusion matrix is shown below.

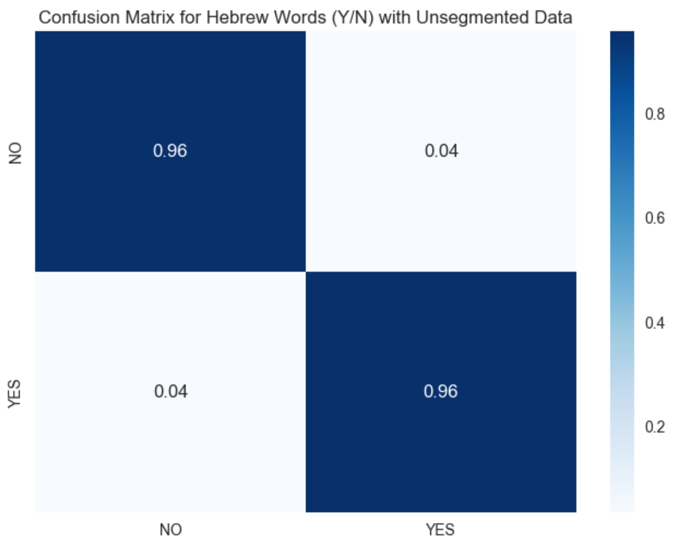


Figure 4: Confusion Matrix for Hebrew Words (Y/N) with Unsegmented Data

2.3.2 Alternative Method (RNN) on Hebrew Yes-No Corpus

See `train.py`, `test.py`, `rnn.py` and `tensorize.py` for RNN implementation. It simply treats every sound as a sequence of different energy through time, which is pretty much what recurrent neural nets are made for. An overall accuracy of 95% could be achieved. Confusion matrix is listed below.

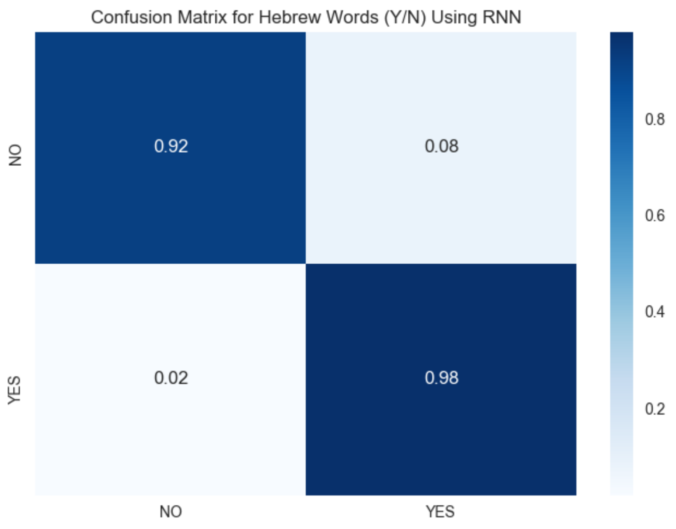


Figure 5: Confusion Matrix for Hebrew Words (Y/N) Using RNN

2.3.3 Average-Column Method on Hebrew Yes-No Corpus

We implemented the method as described in the assignment along with smoother selection using cross validation explained above. The confusion matrix is shown below. As noted in the assignment, our result was not as good as the baseline but the computational complexity (e.g. time taken) was lower.

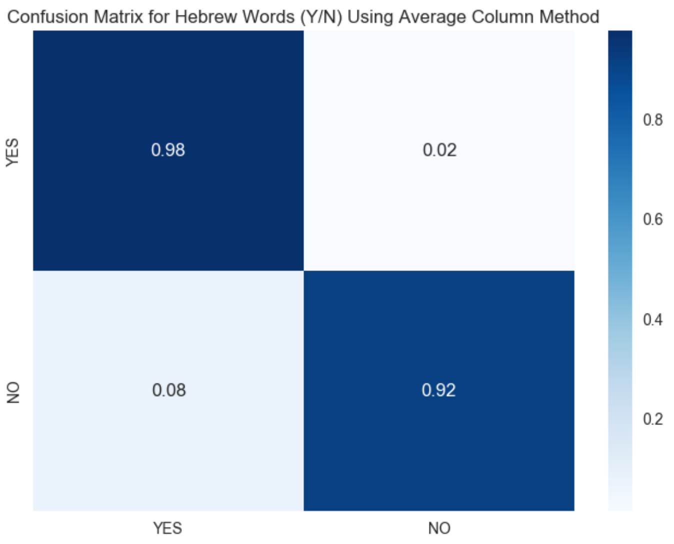


Figure 6: Confusion Matrix for Hebrew Words (Y/N) Using Average Column Method

Appendix

In this appendix section, we explain what is contained in the submission package so that one can refer to the raw files in case that the report above was not clear on any points (which should not be the case).
Note that

- "`.`" in the following table stands for the path to the submission folder, i.e. "`PATH/TO/CS440MP3`".
- All `.pkl` pickle files are removed from the submission package due to size limit.

Table 4: Selected Annotations of Submitted Files

File	Usage or Comments
`./part1/log`	Execuion logs
`./part2/log`	Execuion logs
`./*/*.pkl`	Training results
`./part1/cv_fc.py`	Cross validation utilies for Part 1.2 (4 credit)
`./part1/deserializer.py`	Data loading utilities for Part 1.2 (4 credit)
`./part1/deserializer_face.py`	Data loading utilities for Part 1.3.2 (extra credit)
`./part1/fc_utils.py`	Featurize (convolution) functions for Part 1.2 (4 credit)
`./part1/odds.py`	Utilities to compute odds ratios
`./part1/ternary2binary.py`	Convert data from ternary representation to binary representation
`./part1/test.py`	Generate testing outputs based on training results for Part 1.1 (3 credit)
`./part1/test_fc.py`	Generate testing outputs based on training results for Part 1.2 (4 credit)
`./part1/test_ec.py`	Generate testing outputs based on training results for Part 1.3 (extra credit)
`./part1/train.py`	Model training utilities for Part 1.1 (3 credit)
`./part1/train_fc.py`	Model training utilities for Part 1.2 (4 credit)
`./part1/train_ec.py`	Model training utilities for Part 1.3 (extra credit)
`./part1/disj_rec.txt`	Smoother cross validation results for disjoint kernels
`./part1/ovlp_rec.txt`	Smoother cross validation results for overlapping kernels
`./part2/YESNO.py`	Generate testing outputs for Part 2.1 (3 credit)
`./part2/YESNO_avg_col.py`	Generate testing outputs for Part 2.3.3 (extra credit)
`./part2/cv.py`	Cross validation utilies for Part 2
`./part2/dataLoader.py`	Data loading utilities for Part 2
`./part2/ec2.py`	Generate testing outputs for Part 2.3.1 (extra credit)
`./part2/numberClassifier.py`	Generate testing outputs for Part 2.2 (4 credit)
`./part2/rnn.py`	RNN training and testing for Part 2.3.2 (extra credit)
`./part2/splitData.py`	Data splitting utilities for Part 2.3.1 (extra credit)
`./part2/tensorize.py`	Tensorize utilities for Part 2.3.2 (extra credit)
`./part2/test.py`	Generate testing outputs for Part 2.2 (4 credit)
`./part2/train.py`	Model training utilities for Part 2.1 (3 credit)
`./part2/utils.py`	General utilities for Part 2
`./visualization/viz.py`	Python code for visualizations
`./report`	Reports and supporting documents to generate the report

Statement of Individual Contribution

Table 5: Statement of Individual Contribution

	NetID	Contribution
Haoen CUI	hcui10	visualization, report, and ideas generation
Guohao DOU	gdou2	part 1 (algorithm design and programming) and ideas generation
Chuchao LUO	chuchao2	part 2 (algorithm design and programming) and ideas generation