ISYE 6740, Summer 2024, Homework 4

100 points

Andrea Ruiz Marquez

6/30/2024

1. Comparing multi-class classifiers for handwritten digits classification. (20 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, Linear 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 mapping 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 a good result is acceptable). Only the best k needs to be reported.
- 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 the proper kernel.
- For KNN and SVM, you can randomly downsample the training data to size $m = 5000$, to improve the computation efficiency.
- Packages may be used for all models in this problem

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

1. (15 points) Report confusion matrix, precision, recall, and F-1 score for each of the classifiers. For the 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 classifiers, **KNN, logistic regression, Linear SVM, kernel SVM, and neural networks**, accounts for 3 points.

KNN

	precision	recall	f1-score	support
0	0.94	0.99	0.96	980
1	0.88	1.00	0.94	1135
2	0.98	0.89	0.93	1032
3	0.92	0.95	0.93	1010
4	0.95	0.90	0.92	982
5	0.93	0.92	0.93	892
6	0.96	0.97	0.96	958
7	0.93	0.93	0.93	1028
8	0.98	0.86	0.92	974
9	0.89	0.92	0.90	1009
accuracy			0.93	10000
macro avg	0.94	0.93	0.93	10000
weighted avg	0.94	0.93	0.93	10000

KNN print

Log Regression

	precision	recall	f1-score	support
0	0.92	0.96	0.94	980
1	0.96	0.97	0.96	1135
2	0.93	0.88	0.90	1032
3	0.87	0.91	0.89	1010
4	0.89	0.92	0.91	982
5	0.86	0.86	0.86	892
6	0.93	0.93	0.93	958
7	0.93	0.90	0.91	1028
8	0.86	0.83	0.85	974
9	0.89	0.88	0.89	1009
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Logistic regression print

```
SVM
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       980
           1       0.96      0.97      0.96      1135
           2       0.92      0.86      0.89      1032
           3       0.87      0.88      0.87      1010
           4       0.89      0.91      0.90       982
           5       0.84      0.84      0.84       892
           6       0.93      0.92      0.93       958
           7       0.92      0.90      0.91      1028
           8       0.84      0.81      0.82       974
           9       0.87      0.87      0.87      1009

    accuracy                           0.90     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.90      0.90      0.90     10000
```

SVM print

```
Kernel SVM
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       980
           1       0.98      0.99      0.98      1135
           2       0.96      0.95      0.96      1032
           3       0.94      0.96      0.95      1010
           4       0.95      0.97      0.96       982
           5       0.94      0.96      0.95       892
           6       0.96      0.97      0.97       958
           7       0.97      0.94      0.96      1028
           8       0.95      0.93      0.94       974
           9       0.95      0.92      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
```

Kernel SVM print

Neural Networks

	precision	recall	f1-score	support
0	0.93	0.97	0.95	980
1	0.97	0.97	0.97	1135
2	0.93	0.90	0.91	1032
3	0.90	0.89	0.90	1010
4	0.91	0.93	0.92	982
5	0.89	0.88	0.89	892
6	0.94	0.94	0.94	958
7	0.93	0.92	0.92	1028
8	0.89	0.87	0.88	974
9	0.90	0.90	0.90	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

Neural Network print

2. (5 points) Comment on the performance of the classifier and give your explanation why some of them perform better than others.

The performance of all the models performed well since the accuracy ranged around 90percent. Some performed better than others such as kernel SVM and KNN which performed better than the linear models.

2. SVM (25 points).

1. (5 points) Explain why can we set the margin $c = 1$ to derive the SVM formulation? Justify using a mathematical proof.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{with} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for } i \end{aligned}$$

.
c=1 in SVM standardizes the margin between the hyperplane and vectors, simplifying while separating classes.

2. (5 points) Using Lagrangian dual formulation, show that the weight vector can be represented as

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i.$$

where

alpha$_i \geq 0$ are the dual variables. What does this imply in terms of how to relate data to w?

The weight vector $\mathbf{w}$ in SVM is a linear combination of the training data points $\mathbf{x}_i$, where $\alpha_i \geq 0$ are the Lagrange multipliers and y_i are the class labels corresponding to $\mathbf{x}_i$. This emphasizes that $\mathbf{w}$ is influenced by the support vectors (non-zero points α_i).

3. (5 points) Explain why only the data points on the "margin" will contribute to the sum above, i.e., playing a role in defining w. Hint: use the Lagrangian multiplier derivation and KKT condition we discussed in class.

These support vectors have non-zero Lagrange multipliers ($\alpha_i > 0$), derived using the KKT conditions.

$$\alpha_i g_i(\mathbf{w}) = 0 \quad \text{and} \quad \alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) = 0.$$

This implies:

$$\begin{cases} \alpha_i = 0, & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0, \\ \alpha_i > 0, & \text{if } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) = 0. \end{cases}$$

This shows how α_i can only be non-zero for points that lie on the margin, which give this conclusion of:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i,$$

4. Simple SVM by hand. Suppose we only have four training examples in two dimensions as shown in Fig. The positive samples at $x_1 = (0, 0)$, $x_2 = (2, 2)$ and negative samples at $x_3 = (h, 1)$ and $x_4 = (0, 3)$.

   (a) (5 points) Given $h > 0$, For what range of parameter h are the training points still linearly separable?
   $x_1 = (0, 0)$ lies at origin
   $x_2 = (2, 2)$ lies on $y = x$.
   $x_3 = (h, 1)$ lies on $y = 1$ at $x = h$.
   $x_4 = (0, 3)$ lies at $y = 3$.

   For linearly separable, there must be a line which separates positive points x_1, x_2 from negative points x_3 and x_4.
   The line $y = x$ goes through $x_1 = (0, 0)$ and $x_2 = (2, 2)$. For $x_3 = (h, 1)$ to be above the line $y = x$
   Therefore,x_1, x_2, x_3, and x_4 to be linearly separable, h must satisfy $h < 1$.

(b) (5 points) Does the orientation of the maximum margin decision boundary change as h changes, when the points are separable? Please explain your conclusion.

The orientation of the maximum margin decision boundary remains consistent regardless of the parameter h, it affects the exact positioning of the points, but does not change the direction hyperplane defined by given support vectors.

3. Neural networks and backpropagation. (15 points)

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

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

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 $z_2^i = \sigma(\beta^T x^i)$. Be sure to show all steps of your proof.

(a) (5 points) Show that the gradient is given by

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

where $u^i = w^T z^i$.

$$\ell(w, \alpha, \beta) = \sum_{i=1}^{m} (y_i - \sigma(w^T z_i))^2$$

$$\frac{\partial \ell}{\partial w} = -2\sum_{i=1}^{m} (y_i - \sigma(w^T z_i))\sigma'(w^T z_i)z_i$$

$$= -2\sum_{i=1}^{m} (y_i - \sigma(w^T z_i))\sigma(w^T z_i)(1 - \sigma(w^T z_i))z_i$$

(b) (10 points) Also, show the gradient of $\ell(w, \alpha, \beta)$ with respect to α and β and write down their expression.

$$\frac{\partial \ell}{\partial \alpha} = \sum_{i=1}^{m} 2(y_i - \sigma(w^T z_i))\sigma'(w^T z_i)w z_{i1}(1 - z_{i1})x_i$$

$$= \sum_{i=1}^{m} 2(y_i - \sigma(w^T z_i))\sigma(w^T z_i)(1 - \sigma(w^T z_i))w x_i z_{i1}(1 - z_{i1}),$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} 2(y_i - \sigma(w^T z_i))\sigma'(w^T z_i) w z_{i2}(1 - z_{i2}) x_i$$
$$= \sum_{i=1}^{m} 2(y_i - \sigma(w^T z_i))\sigma(w^T z_i)(1 - \sigma(w^T z_i)) w x_i z_{i2}(1 - z_{i2}).$$

4. Feature selection and change-point detection. (20 points)

(a) (10 points) Consider the mutual information-based feature selection. Suppose we have the following table (the entries in the table indicate counts) for the spam versus and non-spam emails:

	"prize" = 1	"prize" = 0
"spam" = 1	150	10
"spam" = 0	1000	15000

	"hello" = 1	"hello" = 0
"spam" = 1	145	15
"spam" = 0	11000	5000

Given the two tables above, calculate the mutual information for the two keywords, "prize" and "hello" respectively. Which keyword is more informative for deciding whether or not the email is spam? If any tools are used for your calculation, you must still show your mathematical steps in your report and include code/files used for your calculations.

$$P(\text{"prize"} = 1) = \frac{150 + 10}{150 + 10 + 1000 + 15000} = \frac{160}{17160}$$
$$P(\text{"prize"} = 0) = \frac{1000 + 15000}{150 + 10 + 1000 + 15000} = \frac{16000}{17160}$$
$$P(\text{"spam"} = 1) = \frac{150 + 10 + 145 + 15}{17160 + 16260} = \frac{320}{33420}$$
$$P(\text{"spam"} = 0) = \frac{1000 + 15000 + 11000 + 5000}{17160 + 16260} = \frac{33300}{33420}$$
$$P(\text{"prize"} = 1, \text{"spam"} = 1) = \frac{150}{33420}$$
$$P(\text{"prize"} = 0, \text{"spam"} = 1) = \frac{10}{33420}$$
$$P(\text{"prize"} = 1, \text{"spam"} = 0) = \frac{1000}{33420}$$

$$P(\text{"prize"} = 0, \text{"spam"} = 0) = \frac{15000}{33420}$$

$$H(\text{"prize"}) = -\left[P(\text{"prize"} = 1)\log P(\text{"prize"} = 1) + P(\text{"prize"} = 0)\log P(\text{"prize"} = 0)\right]$$

Mutual Information for prize:

$$I(\text{"prize"}, \text{"spam"}) = H(\text{"spam"}) - H(\text{"prize"}) = 0.67650.6521 = 0.0244$$

$$P(\text{"hello"} = 1) = \frac{145 + 15}{145 + 15 + 11000 + 5000} = \frac{160}{16260}$$

$$P(\text{"hello"} = 0) = \frac{11000 + 5000}{145 + 15 + 11000 + 5000} = \frac{16000}{16260}$$

$$P(\text{"spam"} = 1) = \frac{320}{33420}$$

$$P(\text{"spam"} = 0) = \frac{33300}{33420}$$

$$P(\text{"hello"} = 1, \text{"spam"} = 1) = \frac{145}{33420}$$

$$P(\text{"hello"} = 0, \text{"spam"} = 1) = \frac{15}{33420}$$

$$P(\text{"hello"} = 1, \text{"spam"} = 0) = \frac{11000}{33420}$$

$$P(\text{"hello"} = 0, \text{"spam"} = 0) = \frac{5000}{33420}$$

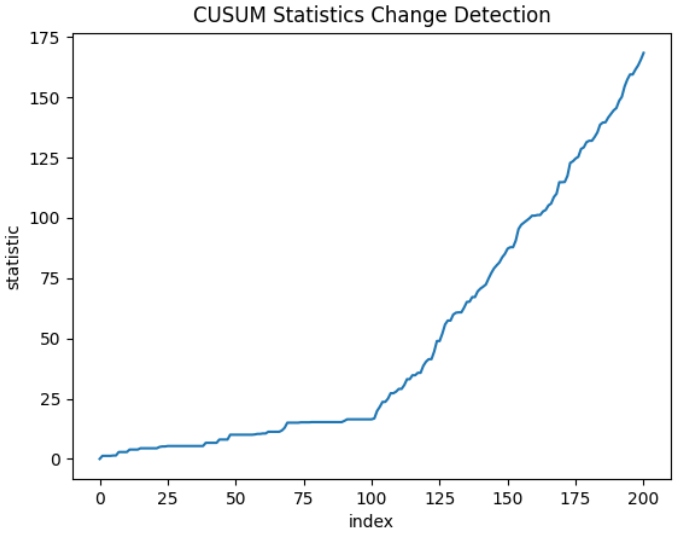
$$H(\text{"hello"}) = -\left[P(\text{"hello"} = 1)\log P(\text{"hello"} = 1) + P(\text{"hello"} = 0)\log P(\text{"hello"} = 0)\right]$$

Mutual Information for hello:

$$I(\text{"hello"}, \text{"spam"}) = H(\text{"spam"}) - H(\text{"hello"}) = 0.67650.6461 = 0.0304$$

Hello has a higher mutual information value than prize.

(b) (10 points) Given two distributions, $f_0 = \mathcal{N}(0.1, 1)$, $f_1 = \mathcal{N}(0.5, 1.5)$, derive what should be the CUSUM statistic (i.e., show the mathematical CUSUM detection statistic specific to these distributions). Plot the CUSUM statistic for a sequence of 150 randomly generated i.i.d. (independent and identically distributed) samples, $x_1, \ldots, x_{100}$ according to f_0 and $x_{101}, \ldots, x_{150}$ according to f_1. Please provide a reasonable estimation based solely on your plot of where a change may be detected.



CUSUM statistic for a sequence of 150 randomly
The estimate could be around sample 100, where the CUSUM statistic shows a marked change.

5. Medical imaging reconstruction (20 points).

In this problem, you will consider an example resembles medical imaging reconstruction in MRI. We begin with a true image image of dimension 50×50 (i.e., there are 2500 pixels in total). Data is cs.mat; you can plot it first. This image is truly sparse, in the sense that 2084 of its pixels have a value of 0, while 416 pixels have a value of 1. You can think of this image as a toy version of an MRI image that we are interested in collecting.

Because of the nature of the machine that collects the MRI image, it takes a long time to measure each pixel value individually, but it's faster to measure a linear combination of pixel values. We measure $n = 1300$ linear combinations, with the weights in the linear combination being random, in fact, independently distributed as $\mathcal{N}(0, 1)$. Because the machine is not perfect, we don't get to observe this directly, but we observe a noisy version. These measurements are given by the entries of the vector

$$y = Ax + \epsilon,$$

where $y \in \mathbb{R}^{1300}$, $A \in \mathbb{R}^{1300 \times 2500}$, and $\epsilon \sim \mathcal{N}(0, 25 \times I_{1300})$ where I_n denotes the identity matrix of size $n \times n$. In this homework, you can generate the data y using this model.

Now the question is: can we model y as a linear combination of the columns of x to recover some coefficient vector that is close to the image? Roughly speaking, the answer is yes.

Key points here: although the number of measurements $n = 1300$ is smaller than the dimension $p = 2500$, the true image is sparse. Thus we can recover the sparse image using few measurements exploiting its structure. This is the idea behind the field of *compressed sensing*.

The image recovery can be done using lasso

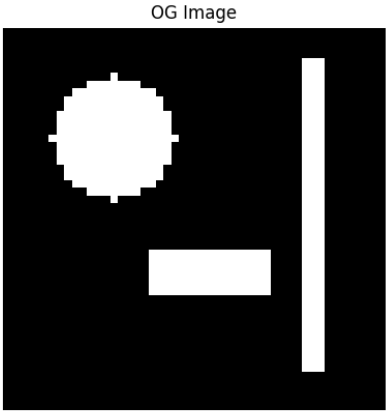
$$\min_x \|y - Ax\|_2^2 + \lambda \|x\|_1.$$

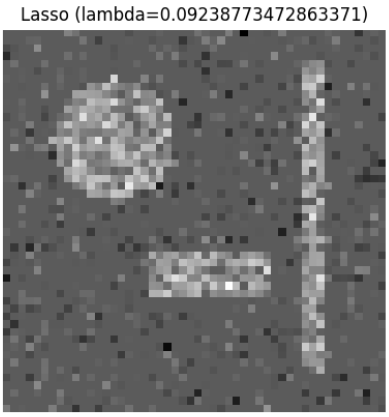
i. (10 points) Now use lasso to recover the image and select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image using your selected lambda values.

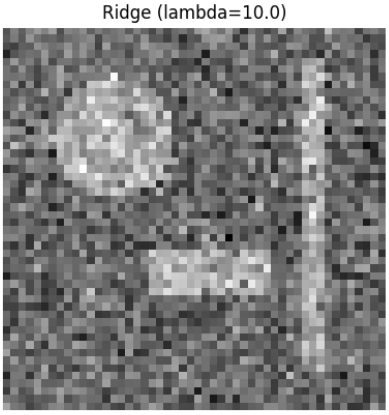
ii. (10 points) To compare, also use ridge regression to recover the image:

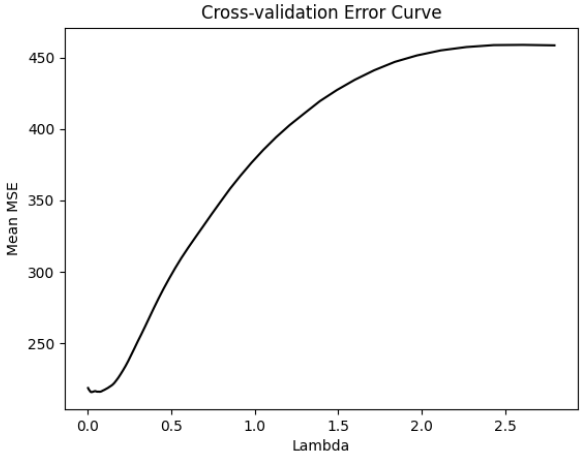
$$\min_x \|y - Ax\|_2^2 + \lambda \|x\|_2^2.$$

Select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image using your selected lambda values. Which approaches give a better recovered image?









Lasso resulted in a better recovered image compared to ridge method.