ISYE 6740 Homework 5
Fall 2024
Prof. Yao Xie
Total 100 points

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, 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, SVM, kernel SVM, and neural networks**, accounts for 3 points.

   Answer:

   KNN and Logistic Regression:

```
Confusion Matrix:
 [[ 966    1    0    0    0    2    9    1    1    0]
 [   0 1130    1    2    0    0    2    0    0    0]
 [  21   42  921    9    3    1    6   22    7    0]
 [   2    7    7  941    1   25    1   11    9    6]
 [   1   23    1    0  904    0   11    2    1   39]
 [   7    9    1   23    7  821   10    1    5    8]
 [  10    8    1    0    4    5  930    0    0    0]
 [   0   43    5    1    8    1    0  944    0   26]
 [  21   11    9   21   12   28    8   10  834   20]
 [   6    9    2    8   23    5    2   17    3  934]]
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       980
           1       0.88      1.00      0.93      1135
           2       0.97      0.89      0.93      1032
           3       0.94      0.93      0.93      1010
           4       0.94      0.92      0.93       982
           5       0.92      0.92      0.92       892
           6       0.95      0.97      0.96       958
           7       0.94      0.92      0.93      1028
           8       0.97      0.86      0.91       974
           9       0.90      0.93      0.91      1009

    accuracy                           0.93     10000
   macro avg       0.93      0.93      0.93     10000
weighted avg       0.93      0.93      0.93     10000
```

```
Confusion Matrix:
 [[ 957    0    1    4    1    9    4    3    1    0]
 [   0 1110    5    2    0    2    3    2   11    0]
 [   6    9  930   16   10    3   12    9   33    4]
 [   4    1   17  922    1   24    2   10   20    9]
 [   1    3    7    3  921    0    7    4    6   30]
 [   9    2    3   35    8  779   15    6   31    4]
 [   8    3    7    2    7   16  912    2    1    0]
 [   1    7   24    6    6    1    0  949    1   33]
 [  10   11    6   20    8   28   14   10  855   12]
 [   9    8    1    9   21    6    0   21    8  926]]
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       980
           1       0.96      0.98      0.97      1135
           2       0.93      0.90      0.91      1032
           3       0.90      0.91      0.91      1010
           4       0.94      0.94      0.94       982
           5       0.90      0.87      0.89       892
           6       0.94      0.95      0.95       958
           7       0.93      0.92      0.93      1028
           8       0.88      0.88      0.88       974
           9       0.91      0.92      0.91      1009

    accuracy                           0.93     10000
   macro avg       0.93      0.93      0.93     10000
weighted avg       0.93      0.93      0.93     10000
```

SVM and Kernel SVM:

```
Confusion Matrix:
 [[ 955    0    3    1    1    6    6    2    3    3]
 [   0 1118    1    3    1    3    3    0    6    0]
 [  11   10  927   20   15    3    9   11   24    2]
 [   3    3   20  910    3   32    2   12   15   10]
 [   1    1    9    0  933    0    5    4    1   28]
 [  12    3    5   42   15  772   11    2   22    8]
 [  13    2   18    2   17   16  886    1    3    0]
 [   1   15   32    5   11    3    0  922    4   35]
 [  12    7   21   46   14   37    8   15  804   10]
 [   4    9    4    8   57   10    0   36    7  874]]
              precision    recall  f1-score   support

           0       0.94      0.97      0.96       980
           1       0.96      0.99      0.97      1135
           2       0.89      0.90      0.89      1032
           3       0.88      0.90      0.89      1010
           4       0.87      0.95      0.91       982
           5       0.88      0.87      0.87       892
           6       0.95      0.92      0.94       958
           7       0.92      0.90      0.91      1028
           8       0.90      0.83      0.86       974
           9       0.90      0.87      0.88      1009

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

```
Confusion Matrix:
 [[ 966    0    1    0    2    4    5    1    1    0]
 [   0 1120    3    2    0    1    4    1    4    0]
 [   8    2  964    8   13    0    8   11   18    0]
 [   1    1   14  943    0   25    1   12   10    3]
 [   1    1    4    0  939    0    8    3    1   25]
 [   5    2    3   19    9  831   12    1    6    4]
 [   9    3    3    0    8    5  927    0    3    0]
 [   0   15   20    3    8    0    0  949    2   31]
 [   6    1    7   15    9   15    6    5  903    7]
 [   6    7    2   12   35    4    0   11    7  925]]
              precision    recall  f1-score   support

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

```
Confusion Matrix:
 [[ 960    0    7    2    0    3    3    2    2    1]
 [   0 1116    0    4    1    1    4    2    7    0]
 [  11    6  975   12    4    2    2   11    8    1]
 [   2    3   14  952    0   16    0    6   10    7]
 [   6    1    4    1  930    2    9   10    2   17]
 [   6    5    6   13    1  823   13    5   12    8]
 [   6    2    5    0    9   14  919    0    2    1]
 [   2    8   13    9   10    2    0  966    4   14]
 [   9    5   10   11    7   14    9    6  901    2]
 [   3    5    0   12   23    9    1   17   13  926]]
              precision    recall  f1-score   support

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

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

   Answer:

   The linear kernel SVM performs poorly compared to more flexible models, because it draw a straight line to separate classes but the dataset is complex. The RBF kernel SVM has the greatest performance because it can handle non-linearly data. Neural networks can learn hierarchical representations and model the complexities of handwritten digits very well.

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.

   Answer:

   For the maximum margin classfier:

$$\max_{w,b} \frac{2c}{\|w\|} \quad \text{subject to} \quad y^i(w^\top x^i + b) \geq c, \forall i$$

   The reason that we set the margin c=1 is because, c is a constant, it is much easier to set it as c instead of changing the scale of w and b. If we change the optimization variable $w = \frac{w}{a}$, $b = \frac{b}{a}$, choose $a = \frac{1}{c}$, we can get a cleaner problem: $\max_{w,b} \frac{1}{\|w\|}$.

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?

   Answer:

   Dual problem of support vector machines

$$\min_{w,b} \|w\|^2 \quad \text{s.t.} \quad y^i(w^\top x^i + b) \geq 1, \forall i$$

Convert to standard form:

$$\min_{w,b} \frac{1}{2} w^\top w \quad \text{s.t.} \quad 1 - y^i(w^\top x^i + b) \leq 0, \forall i$$

The Lagrangian function:

$$L(w, b, \alpha) = \frac{1}{2} w^\top w + \sum_{i=1}^{m} \alpha_i \left(1 - y^i(w^\top x^i + b)\right)$$

Deriving the dual problem:

$$L(w, b, \alpha) = \frac{1}{2} w^\top w + \sum_{i=1}^{m} \alpha_i \left(1 - y^i(w^\top x^i + b)\right)$$

Taking derivative and setting to zero:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} \alpha_i y^i x^i = 0$$

$$w = \sum_{i=1}^{m} \alpha_i y^i x^i$$

w is a linear combination of the data(x^i),each x^i is weighted by the Lagrange multiplier α and the class label y.

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.

   Answer:

   The Lagrangian function:

$$L(w, b, \alpha) = \frac{1}{2} w^\top w + \sum_{i=1}^{m} \alpha_i \left(1 - y^i(w^\top x^i + b)\right)$$

   Based on the KKT conditions:

$$g_i(w) \leq 0$$

$$\alpha_i \geq 0$$

$$\alpha_i g_i(w) = 0$$

$$g_i(w) = 1 - y^i \left(w^\top x^i + b\right)$$

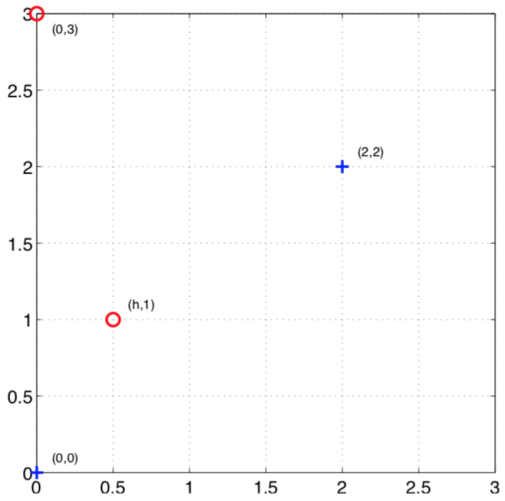
   When $\alpha = 0$, then $1 - y^i \left(w^\top x^i + b\right) < 0$

   When $\alpha > 0$, then $1 - y^i \left(w^\top x^i + b\right) = 0$

   We also know that when $y^i \left(w^\top x^i + b\right) = 1$, the points are on the margin. Thus, α is not equal to 0, which can contribute to the sum above.

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) For what range of parameter $h > 0$, the training points are still linearly separable?

Answer:

The blue points are on the line y=x, then we can separate red points and blue points using the line y=x. In this case, $0 < h < 1$. When h becomes larger, the blue points stay at the left side of the line between the two red points. In this case $h > 4$.

(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.

Answer:

Yes, when $0 < h < 1$, the slope of the decision boundary > 0. When the $h > 4$, the slope of the decision boundary < 0.

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 proofs.

1. (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 $u^i = w^T z^i$.

Answer:

Because $\sigma(x) = \dfrac{1}{1 + e^{-x}}$ is the sigmoid function.

$$\frac{d}{du}\sigma(u) = \sigma(u)\left(1-\sigma(u)\right)$$

$$\frac{\partial}{\partial w}\sigma\left(w^T z^i\right) = \sigma\left(w^T z^i\right)\left(1-\sigma\left(w^T z^i\right)\right) z^i \text{ then plug in the following euqation}$$

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

$$= \sum_{i=1}^{m} 2\left(y^i - \sigma\left(w^T z^i\right)\right)\left(-\sigma\left(w^T z^i\right)\right)\left(1-\sigma\left(w^T z^i\right)\right) z^i = -\sum_{i=1}^{m} 2\left(y^i - \sigma\left(u^i\right)\right)\sigma\left(u^i\right)\left(1-\sigma\left(u^i\right)\right) z^i$$

2. (10 points) Also, show the gradient of $\ell(w,\alpha,\beta)$ with respect to α and β and write down their expression.

Answer:

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \alpha} = \frac{\partial \ell}{\partial z_1^i}\frac{\partial z_1^i}{\partial \alpha}$$

$$= \left(-\sum_{i=1}^{m} 2\left(y^i - \sigma\left(u^i\right)\right)\sigma\left(u^i\right)\left(1-\sigma\left(u^i\right)\right) w_1\right)\left(\sigma\left(\alpha^T x^i\right)\left(1-\sigma\left(\alpha^T x^i\right)\right) x^i\right)$$

$$= -\sum_{i=1}^{m} 2\left(y^i - \sigma\left(u^i\right)\right)\sigma\left(u^i\right)\left(1-\sigma\left(u^i\right)\right) w_1\sigma\left(\alpha^T x^i\right)\left(1-\sigma\left(\alpha^T x^i\right)\right) x^i$$

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \beta} = \frac{\partial \ell}{\partial z_2^i}\frac{\partial z_2^i}{\partial \beta}$$

$$= \left(-\sum_{i=1}^{m} 2\left(y^i - \sigma\left(u^i\right)\right)\sigma\left(u^i\right)\left(1-\sigma\left(u^i\right)\right) w_2\right)\left(\sigma\left(\beta^T x^i\right)\left(1-\sigma\left(\beta^T x^i\right)\right) x^i\right)$$

$$= -\sum_{i=1}^{m} 2\left(y^i - \sigma\left(u^i\right)\right)\sigma\left(u^i\right)\left(1-\sigma\left(u^i\right)\right) w_2\sigma\left(\beta^T x^i\right)\left(1-\sigma\left(\beta^T x^i\right)\right) x^i$$

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

1. (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.

Answer:

$$I(U;C) = \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}}$$

Based on the equation above, we can get the mutual information for prize:

$$I(U;C) = 0.03296$$

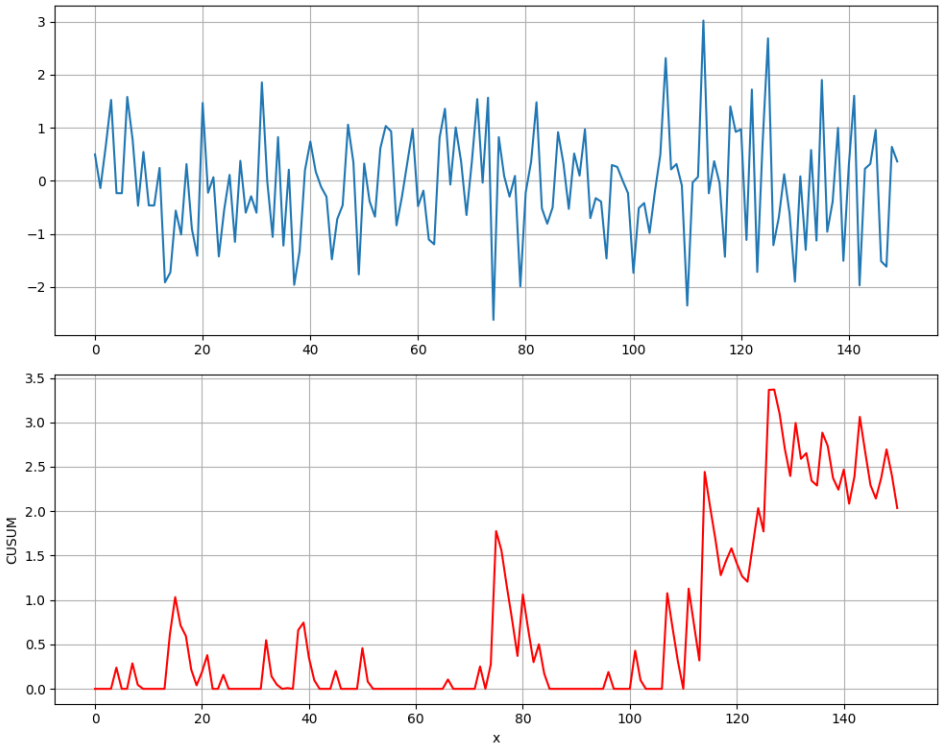
For hello:

$$I(U;C) = 0.00195$$

Prize is more informative.

2. (10 points) Given two distributions, $f_0 = \mathcal{N}(0,1)$, $f_1 = \mathcal{N}(0,1.5)$, 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.

Answer:



Based on the CUSUM plot, the change is detected after 100.

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.

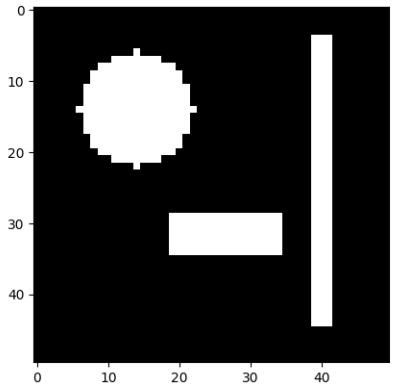
Key points here: although the number of measurements $n = 1300$ is smaller than the dimension $p = 2500$, the true image is sparse. Thus we can recover the sparse image using few measurements exploiting its structure. This is the idea behind the field of *compressed sensing*.

The image recovery can be done using lasso

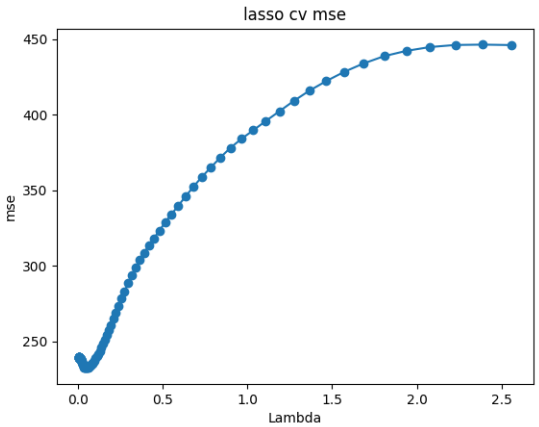
$$\min_x \|y - Ax\|_2^2 + \lambda\|x\|_1.$$

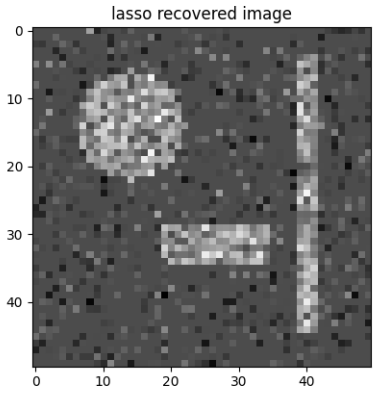
1. (10 points) Now use lasso to recover the image and select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image using your selected lambda values.

   Answer:



The best lambda for Lasso regression is 0.044752





2. (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 approach gives a better recovered image?

Answer:

The best lambda for Ridge regression is 136