ISYE 6740 Homework 5
Spring 2025
Prof. Yao Xie
Total 100 points

Provided Data:

- Q2: Comparing multi-class classifiers for handwritten digits [mnist_10digits.mat]
- Q6: Medical imaging reconstruction [cs.mat]

As usual, please submit a report with sufficient explanation of your answers to each the questions, together with your code, in a zip folder.

1. Bayes classifier optimality. (10 points)

Prove that the Bayes classifier is optimal in that it minimizes the Bayes risk. Let $X \in \mathbb{R}^d$ be a feature vector, and $Y \in \{1, 2, \ldots, K\}$ be the class label. Let $P(Y = k|X)$ be the posterior probability of class k given the feature X. Let $h(X)$ be any classifier mapping X to one of the class labels. Define the risk of the classifier as (the probability of mis-classification):

$$R(h) = P(h(X) \neq Y).$$

Then show that the Bayesian classifier h^* (defined by picking the maximum posterior) achieves the minimum risk among all classifiers

$$h^* = \arg\min_h R(h)$$

Hint: Note that

$$R(h) = P(h(X) \neq Y) = 1 - P(h(X) = Y)$$

Then note that

$$P(h(X) = Y) = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbb{I}\{h(X) = Y|X)\}]],$$

where the expectation is taken with respect to X, and the conditional expectation, respectively; and $\mathbb{I}$ denotes the indicator function.

$$P(h(X) = Y) = \mathbb{E}_X \left[\mathbb{E}_{Y|X} \left[\mathbb{I}\{h(X) = Y\} \mid X\right]\right]$$

$$P(h(X) = Y) = \mathbb{E}_X \left[\sum_{k=1}^{K} P(Y = k \mid X)\mathbb{I}\{h(X) = k\}\right]$$

$$R(h) = P(h(X) \neq Y)$$
$$R(h) = 1 - P(h(X) = Y)$$
$$R(h) = 1 - \mathbb{E}_X \left[\sum_{k=1}^{K} P(Y = k \mid X) \mathbb{I}\{h(X) = k\} \right]$$

For any classifier $h(X)$, it assigns a class label $h(X)$ to each input X. The best choice for $h(X)$ at each X is the one that maximizes the inner sum:

$$\sum_{k=1}^{K} P(Y = k \mid X) \mathbb{I}\{h(X) = k\}$$

Since $\mathbb{I}\{h(X) = k\}$ selects one term from the summation, the best choice for $h(X)$ is:

$$h^*(X) = \arg\max_k P(Y = k \mid X)$$

This is the Bayes classifier, which chooses the class with the highest posterior probability.

For any other classifier $h(X)$, there exists at least one X where $h(X) \neq h^*(X)$, meaning it selects a class with a lower posterior probability than the Bayes classifier. This results in a strictly lower expected probability of correct classification, thus increasing the risk.

Since the Bayes classifier always picks the most probable class given X, it maximizes $P(h(X) = Y)$ and therefore minimizes the risk $R(h)$.

$$h^* = \arg\min_h R(h)$$

2. Comparing multi-class classifiers for handwritten digits. (10 points)

This question is to compare different classifiers and their performance for multi-class classifications on the complete MNIST dataset. 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. (5 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.

Table 1: Confusion Matrices for Each Classifier

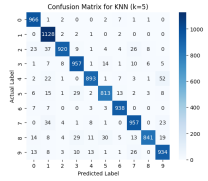
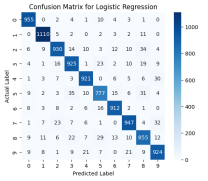
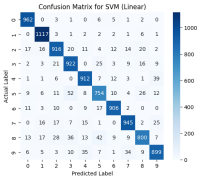
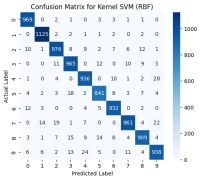
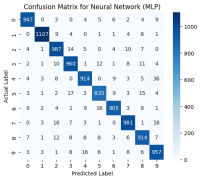
KNN (k=5)	Logistic Regression	SVM (Linear)	Kernel SVM (RBF)	Neural Network (MLP)
				

Table 2: Precision, Recall, and F1-score for Each Classifier

Classifier	Metric	0	1	2	3	4	5	6	7	8	9
KNN (k=5)	Precision	0.9360	0.8903	0.9735	0.9229	0.9571	0.9356	0.9591	0.9220	0.9779	0.8972
	Recall	0.9857	0.9938	0.8915	0.9475	0.9094	0.9114	0.9791	0.9309	0.8634	0.9257
	F1-score	0.9602	0.9392	0.9307	0.9350	0.9326	0.9233	0.9690	0.9264	0.9171	0.9112
Logistic Regression	Precision	0.9531	0.9619	0.9291	0.9042	0.9369	0.8952	0.9431	0.9321	0.8805	0.9103
	Recall	0.9745	0.9780	0.9012	0.9158	0.9379	0.8711	0.9520	0.9212	0.8778	0.9158
	F1-score	0.9637	0.9699	0.9149	0.9100	0.9374	0.8830	0.9475	0.9266	0.8792	0.9130
SVM (Linear)	Precision	0.9422	0.9434	0.8998	0.8789	0.9075	0.8717	0.9438	0.9247	0.9070	0.9044
	Recall	0.9816	0.9841	0.8876	0.9129	0.9287	0.8453	0.9457	0.9193	0.8214	0.8910
	F1-score	0.9615	0.9633	0.8937	0.8956	0.9180	0.8583	0.9447	0.9220	0.8621	0.8977
Kernel SVM (RBF)	Precision	0.9642	0.9766	0.9513	0.9433	0.9435	0.9524	0.9608	0.9639	0.9548	0.9380
	Recall	0.9888	0.9912	0.9457	0.9554	0.9532	0.9428	0.9729	0.9348	0.9333	0.9296
	F1-score	0.9763	0.9838	0.9485	0.9493	0.9483	0.9476	0.9668	0.9491	0.9439	0.9338
Neural Network (MLP)	Precision	0.9673	0.9866	0.9382	0.9421	0.9491	0.9446	0.9638	0.9543	0.9336	0.9246
	Recall	0.9663	0.9753	0.9564	0.9505	0.9308	0.9361	0.9447	0.9543	0.9384	0.9485
	F1-score	0.9668	0.9809	0.9472	0.9463	0.9398	0.9403	0.9541	0.9543	0.9360	0.9364

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

Kernel SVM and Neural Network are the best-performing classifiers due to their ability to model complex, non-linear relationships in the data. Kernel SVM maps inputs to a higher-dimensional space, effectively separating digit classes, while MLP learns hierarchical features, making it highly adaptable to variations in handwriting. KNN performs well but is sensitive to noise and struggles with overlapping digits. Logistic Regression and Linear SVM perform the worst as they assume linear separability, which is insufficient for handwritten digit recognition. Overall, non-linear models like RBF SVM and MLP excel due to their superior feature representation and decision boundaries.

3. 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.

The decision boundary of an SVM is defined as $f(x) = w^T x + b$, where w is the weight vector and b is the bias. The margin is $\frac{1}{\|w\|}$.

For a correctly classified sample (x_i, y_i), the SVM constraint requires $y_i(w^T x_i + b) \geq c$. Since the optimization only depends on the ratio w/c, we can rescale w and b by dividing through by c, without changing the decision boundary: $\frac{y_i(w^T x_i + b)}{c} \geq 1$.

Setting $c = 1$ simplifies the formulation to $y_i(w^T x_i + b) \geq 1$. This results in the accepted margin, which ensures a unique and scale-invariant SVM formulation. Thus, we can always set $c = 1$ without loss of generality.

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?

SVM form can be represented with $\min_{w,b} \frac{1}{2}\|w\|^2$, subject to $y_i(w^T x_i + b) \geq 1, \quad \forall i = 1, \ldots, n$.

The Lagrange multipliers $\alpha_i \geq 0$ for each constraint and construct the Lagrangian are $\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1]$. To find the dual formulation, take the derivative of the Lagrangian with respect to w to get $\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$. Solving for w: $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

This result implies that:

(a) The Lagrange multipliers α_i are nonzero only for support vectors (points on the margin)

(b) Other data points have $\alpha_i = 0$, meaning they do not contribute to w

(c) w is represented as a weighted sum of support vectors x_i, scaled by $\alpha_i y_i$

(d) This is key in kernel SVM, where we use kernels instead of explicitly computing w

Thus, the entire decision boundary is determined only by the support vectors and not by the entire dataset.

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.

To explain why only the data points on the margin contribute to w, we use the Karush-Kuhn-Tucker (KKT) conditions, which are necessary conditions for optimality in constrained optimization problems.

The SVM optimization problem is:

$$\min_{w,b} \frac{1}{2}\|w\|^2, \quad \text{subject to } y_i(w^T x_i + b) \geq 1, \quad \forall i$$

The Lagrangian for this problem is:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1]$$

From the KKT conditions, we obtain the complementary slackness condition:

$$\alpha_i [y_i(w^T x_i + b) - 1] = 0, \quad \forall i$$

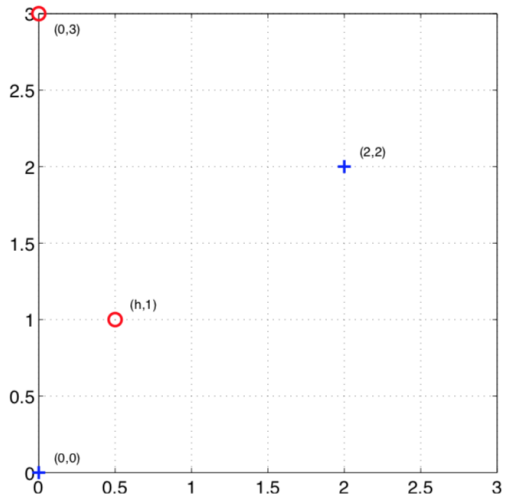
This means that for any given training example (x_i, y_i), either:

- $\alpha_i = 0$, in which case the constraint $y_i(w^T x_i + b) \geq 1$ is strictly satisfied (i.e., the point is not exactly on the margin, and thus does not contribute to w), or
- $y_i(w^T x_i + b) = 1$, meaning that x_i is a support vector, exactly on the margin, and has $\alpha_i > 0$, contributing to w

Thus, only support vectors (the points on the margin) have nonzero α_i and contribute to the weight vector w, while all other points have $\alpha_i = 0$ and do not influence w.

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? Be sure to consider all possible ranges.

For the training points to be linearly separable, there must exist a linear decision boundary that correctly classifies the positive and negative samples.

- The positive points are $(0, 0)$ and $(2, 2)$

- The negative points are $(h, 1)$ and $(0, 3)$

To check separability, we must find a linear decision boundary that separates these groups.

- The two positive points roughly form a line along $y = x$
- The two negative points must be on the opposite side of this decision boundary

The critical observation is that if h is too large, $(h, 1)$ moves to the right, potentially making the dataset non-linearly separable.

To maintain separability, the point $(h, 1)$ should remain below the line that separates the classes. A reasonable choice for the separating line could be:

$$y = x + b.$$

From the given points:

- $(0, 0)$ should satisfy $0 = 0 + b \Rightarrow b = 0$
- $(2, 2)$ should satisfy $2 = 2 + 0$, which is correct
- $(0, 3)$ should be on the negative side, meaning $3 > 0 + b$, which is true for any $b \leq 0$

For $(h, 1)$ to be correctly classified as negative:

$$1 > h + 0 \quad \Rightarrow \quad h < 1$$

Thus, for the data to remain linearly separable:

$$0 < h < 1$$

(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.

If h remains in the range $(0, 1)$, the data points are still linearly separable, and the maximum margin classifier remains the same because the support vectors (points defining the margin) remain unchanged.

- The support vectors in this case are likely $(2, 2)$ and $(0, 3)$, as they determine the widest margin
- The boundary would be roughly a line with slope close to 1 (or a variation of it)

However, if h gets too close to 1, the negative point $(h, 1)$ moves towards the decision boundary, potentially becoming a support vector and slightly tilting the margin.

If $h \geq 1$, the points are no longer separable, and a linear SVM would fail.

Thus, as long as $h < 1$, the orientation remains roughly the same, but if h is close to 1, the margin and decision boundary might shift slightly.

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

$$\begin{aligned}
\ell(w,\alpha,\beta) &= \sum_{i=1}^{m}(y^i - \sigma(w^T z^i))^2 \\
\sigma(u^i) &= \frac{1}{1+e^{-u^i}} \\
\frac{\partial \ell}{\partial w} &= \sum_{i=1}^{m} 2(y^i - \sigma(u^i))\frac{\partial}{\partial w}(-\sigma(u^i)) \\
\frac{d\sigma(u^i)}{du^i} &= \sigma(u^i)(1-\sigma(u^i)) \\
\frac{\partial \sigma(u^i)}{\partial w} &= \sigma(u^i)(1-\sigma(u^i))\frac{\partial u^i}{\partial w} = \sigma(u^i)(1-\sigma(u^i))z^i \\
\frac{\partial \ell}{\partial w} &= -\sum_{i=1}^{m} 2(y^i - \sigma(u^i))\sigma(u^i)(1-\sigma(u^i))z^i
\end{aligned}$$

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

$$\begin{aligned}
\frac{\partial \ell}{\partial \alpha}, \quad \frac{\partial \ell}{\partial \beta} \\
\frac{\partial \ell}{\partial \alpha} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial z_1^i}\frac{\partial z_1^i}{\partial \alpha} \\
\frac{\partial \ell}{\partial z^i} &= -2(y^i - \sigma(u^i))\sigma(u^i)(1-\sigma(u^i))w \\
\frac{\partial z_1^i}{\partial \alpha} &= \sigma(\alpha^T x^i)(1-\sigma(\alpha^T x^i))x^i \\
\frac{\partial \ell}{\partial \alpha} &= \sum_{i=1}^{m} -2(y^i - \sigma(u^i))\sigma(u^i)(1-\sigma(u^i))w_1\sigma(\alpha^T x^i)(1-\sigma(\alpha^T x^i))x^i
\end{aligned}$$

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

5. 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	140	10
"spam" = 0	1100	14000

	"hello" = 1	"hello" = 0
"spam" = 1	150	20
"spam" = 0	12000	6000

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.

$$I(X;Y) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

$P(\text{spam} = 1) = \frac{150}{15250}$, $P(\text{spam} = 0) = \frac{15100}{15250}$, $P(\text{"prize"} = 1) = \frac{1240}{15250}$, $P(\text{"prize"} = 0) = \frac{14010}{15250}$

$P(1,1) = \frac{140}{15250}$, $P(1,0) = \frac{10}{15250}$, $P(0,1) = \frac{1100}{15250}$, $P(0,0) = \frac{14000}{15250}$

$P(\text{spam} = 1) = \frac{170}{18220}$, $P(\text{spam} = 0) = \frac{18000}{18220}$, $P(\text{"hello"} = 1) = \frac{12150}{18220}$, $P(\text{"hello"} = 0) = \frac{6020}{18220}$

$P(1,1) = \frac{150}{18220}$, $P(1,0) = \frac{20}{18220}$, $P(0,1) = \frac{12000}{18220}$, $P(0,0) = \frac{6000}{18220}$

Mutual Information for "prize": $\frac{140}{15250} \log_2 \left(\frac{\frac{140}{15250}}{\left(\frac{1240}{15250} \times \frac{150}{15250}\right)} \right) + \frac{10}{15250} \log_2 \left(\frac{\frac{10}{15250}}{\left(\frac{14010}{15250} \times \frac{150}{15250}\right)} \right) + \frac{1100}{15250} \log_2 \left(\frac{\frac{1100}{15250}}{\left(\frac{1240}{15250} \times \frac{15100}{15250}\right)} \right) + \frac{14000}{15250} \log_2 \left(\frac{\frac{14000}{15250}}{\left(\frac{14010}{15250} \times \frac{15100}{15250}\right)} \right) = 0.0323227 - 0.0024817 - 0.0114383 + 0.0121461 = 0.0305$

Mutual Information for "hello": $\frac{150}{18220} \log_2 \left(\frac{\frac{150}{18220}}{\left(\frac{12150}{18220} \times \frac{170}{18220}\right)} \right) + \frac{20}{18220} \log_2 \left(\frac{\frac{20}{18220}}{\left(\frac{6020}{18220} \times \frac{170}{18220}\right)} \right) + \frac{12000}{18220} \log_2 \left(\frac{\frac{12000}{18220}}{\left(\frac{12150}{18220} \times \frac{18000}{18220}\right)} \right) + \frac{6000}{18220} \log_2 \left(\frac{\frac{6000}{18220}}{\left(\frac{6020}{18220} \times \frac{18000}{18220}\right)} \right) = 0.0033024 - 0.0016442 - 0.0028797 + 0.0028929 = 0.00167$

Mutual Information for "prize": 0.0305
Mutual Information for "hello": 0.00167

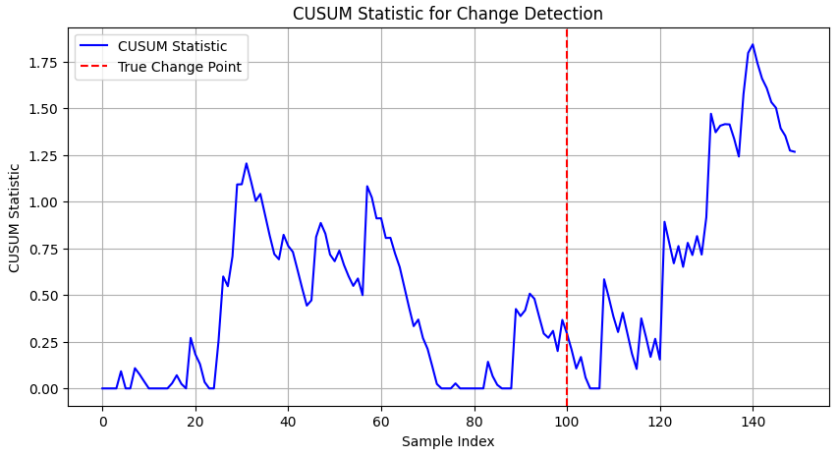
Since $I(\text{"prize"}; \text{spam}) > I(\text{"hello"}; \text{spam})$, the word "prize" is much more informative for classifying spam emails than "hello".

2. (10 points) Given two distributions, $f_0 = \mathcal{N}(0,1)$, $f_1 = \mathcal{N}(0,1.25)$, 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. Using your plot, please provide a reasonable estimation of where a change may be detected.
Note: Be certain to set your random seed to 6740 to ensure reproducible results. Failure to do so will result in point deductions

$$S_n = \max\left(0, S_{n-1} + \log\frac{f_1(X_n)}{f_0(X_n)}\right), \quad \log\frac{f_1(X)}{f_0(X)} = \log\left(\frac{\frac{1}{\sqrt{2\pi(1.25)}}e^{-X^2/(2\times 1.25)}}{\frac{1}{\sqrt{2\pi(1)}}e^{-X^2/(2\times 1)}}\right)$$

$$\log\frac{f_1(X)}{f_0(X)} = \frac{X^2}{2}\left(\frac{1}{1} - \frac{1}{1.25}\right) + \frac{1}{2}\log\frac{1}{1.25}$$

This function accumulates the log-likelihood ratio over time, resetting if it becomes negative.



6. 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.

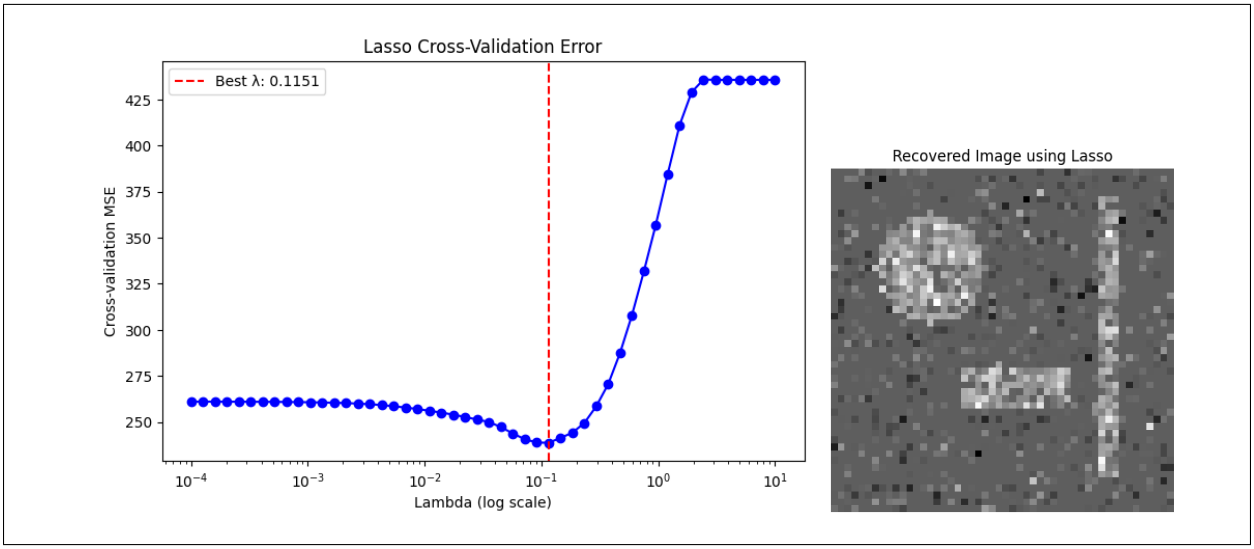
Now the question is: can we model y as a linear combination of the columns of x to recover some coefficient vector that is close to the image? Roughly speaking, the answer is yes.

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

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

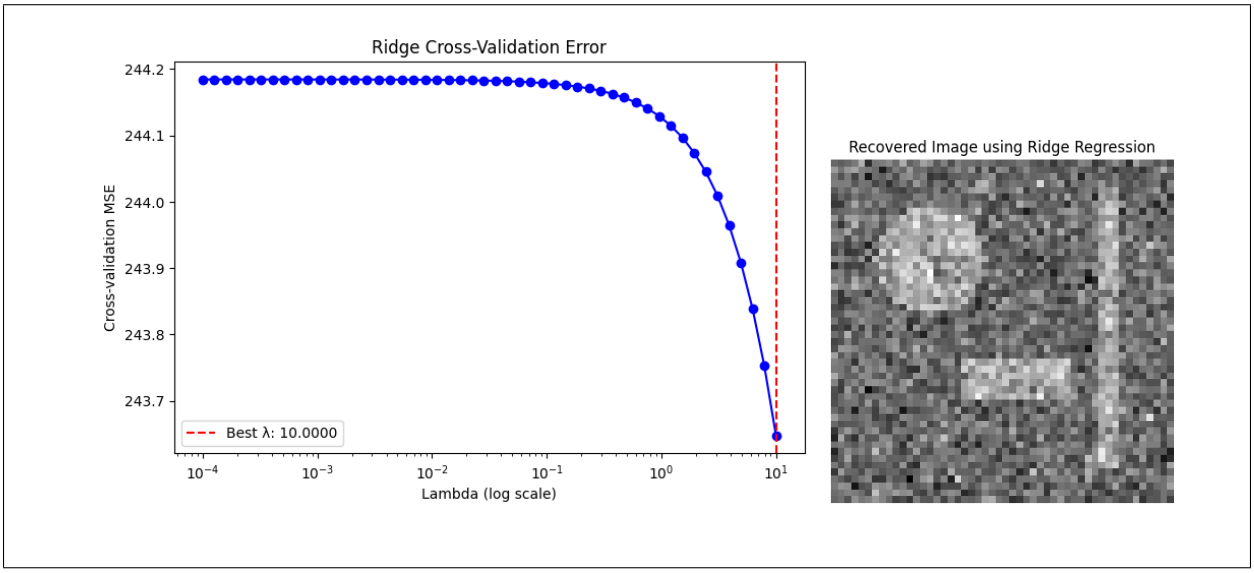
1. (10 points) Now use lasso to recover the image and select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image using your selected lambda values.



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?



Lasso Reconstruction Error (L2 Norm): 13.1099
Ridge Reconstruction Error (L2 Norm): 15.3320
Lasso produces a better recovered image due to sparsity