Introduction to Machine Learning

ECE 580
Fall 2025

HW #2, Due 02/10/25 11:59pm

Submission Instructions

Submit your work to the corresponding assignment in Gradescope. Although Gradescope accepts multiple file formats, they strongly recommend submitting your assignment as a single PDF file.

It is your responsibility to ensure the uploaded file is: 1) the correct file, 2) complete (includes all pages), 3) legible, and 4) submitted on-time as determined by the Gradescope server system clock.

It is your responsibility to submit a multi-page PDF file and tag the pages that correspond to each question. Pages may be tagged after submission, even if the submission deadline has passed. If you are submitting close to the submission deadline, submit your assignment first then immediately return to tag pages.

When code is requested, submit a PDF print-out of your code. Submitting a URL for a cloud-based repository is insufficient.

Late Submissions

Late submissions will be accepted up to 5 days after the submission deadline, with the following point penalty applied if its late submission is not excused: [1]

1 day (0^+ to 24 hours) late: 2 point deduction ($\frac{1}{5}$ letter grade)
2 days (24^+ to 48 hours) late: 5 point deduction ($\frac{1}{2}$ letter grade)
3 days late: 10 point deduction (1 letter grade)
4 days late: 20 point deduction (2 letter grades)
5 days late: 30 point deduction (3 letter grades)
6 or more days late: score = 0 (not accepted for credit)

The late policy is designed to be minimally punitive for submissions up to 3 days late, yet encourages staying current with the coursework for our course by not allowing one assignment's late submission to overlap with the next assignment's submission.

A homework score will not drop below 0 as a result of applying the late penalty point deduction.

[1] One day = one 24-hour period or fraction thereof.

Cross-Validation

Cross-validation is used for both model selection and hyperparameter selection, to ensure the chosen model and/or hyperparameter(s) are not too highly tuned ("overfit") to the data. Here, you are going to explore the impact of cross-validation on your selection of a model that predict a car's price from its characteristics.

1. We will continue with the 13 continuous predictor variables from the Automobile Data Set from the UCI Machine Learning Repository that you used in Homework #1.

(10) (a) For your proposed model, compute the OLS solution[2] with 6-fold cross-validation over the data in `cleaned_automobile_train_dataset.pt`[3]. Let $\hat{w}_{OLS}^{(i)}$ denote the solution for the i^{th} split, for $i = 1...6$.

For each split i, compute the mean square error (MSE) of $\hat{w}_{OLS}^{(i)}$, both for the train subset (130), and for the validation subset (26). Also, compute the average MSE, for both training subset and validation subset, averaged over 6 splits. Print your results.

(10) (b) **(Ridge Regression)** The λ-Ridge-Regression is

$$\hat{w}_{ridge,\lambda} = \arg\min_{w} \frac{1}{n} \sum_{j=1}^{n} (y_j - w^\top \phi(x_j))^2 + \lambda \|w\|_2^2$$

Pick 10 values of λ on a log scale (e.g. $\lambda \in \{2, 1, 0.5, 0.25...\}$. Pick your λ's such that for the largest λ, $0.1\|\hat{w}_{lasso,0}\|_2 < \|\hat{w}_{ridge,\lambda}\|_2 < 0.5\|\hat{w}_{ridge,0}\|_2$ (where $\hat{w}_{ridge,\lambda}$ is computed over the entire training set).

Plot $\|\hat{w}_{ridge,\lambda}\|_2$ against λ (use log scale for λ).

(30) (c) For each λ from (b), perform 6-fold cross-validation, as you did in (a), and compute:

1. cross-validation train MSE of $\hat{w}_{ridge,\lambda}$
2. cross-validation validation MSE of $\hat{w}_{ridge,\lambda}$

Plot the two MSE losses above against λ. (use log scale for x-axis.)

(5) (d) **(Ridge Regression)** Identify the best λ^* from (c) based on validation error. Would your choice of λ have been different based on train error?

Compute $\hat{w}_{ridge,\lambda^*}$ using the entire train dataset, and evaluate the MSE loss on the test dataset (`cleaned_automobile_test_dataset.pt`). Print both the average-6-fold-validation-MSE and the test-MSE.

(10) (e) **(LASSO)** The λ-LASSO is

$$\hat{w}_{lasso,\lambda} = \arg\min_{w} \frac{1}{n} \sum_{j=1}^{n} (y_j - w^\top \phi(x_j))^2 + \lambda \|w\|_1$$

There is no closed-form solution for $\hat{w}_{lasso,\lambda}$, unlike Ridge Regression or OLS. Since we have not learned about optimization yet, we will use sklearn. Demo for solving for $\hat{w}_{lasso,\lambda}$ is provided in starter code.

[2] as you did in HW#1 2(b)

[3] The original "training set" from HW #1 contains 156 samples. Thus each fold should have 13 samples. In each split, there will be 13 validation and 143 training samples

Pick 10 values of λ on a log scale (e.g. $\lambda \in \{2, 1, 0.5, 0.25...\}$). Pick your λ's such that for the largest λ (and only the largest λ), $\hat{w}_{lasso,\lambda}$ is all 0 (where $\hat{w}_{lasso,\lambda}$ is computed over the entire training set).

Plot $\|\hat{w}_{lasso,\lambda}\|_1$ against λ (use log scale for λ).

Plot the [number of entries of $\hat{w}_{lasso,\lambda}$ which are exactly 0], against λ (use log scale for λ).

(30) (f) For each λ, perform 6-fold cross-validation, as you did in (a), and compute:

1. cross-validation train MSE of $\hat{w}_{lasso,\lambda}$
2. cross-validation validation MSE of $\hat{w}_{lasso,\lambda}$

Plot the two MSE losses above against λ. (use log scale for x-axis.)

(5) (g) **(LASSO)** Identify the best λ^* from (f) based on validation error. Would your choice of λ have been different based on train error?

Compute $\hat{w}_{lasso,\lambda^*}$ using the entire train dataset, and evaluate the MSE loss on the test dataset (`cleaned_automobile_test_dataset.pt`).

Print both the 6-fold-cross-validation-MSE and the test-MSE.

Print the number of 0 entries of $\hat{w}_{lasso,\lambda^*}$.

(5) (h) Submit a PDF print-out of your code. (Submitting a URL for a cloud-based repository is insufficient.)

Compressed Sensing Image Recovery

Abstract

Corrupted images may arise from numerous causes – faulty CCDs in an digital imaging device, deterioration of a printed photograph, or even intentional sampling to achieve image compression. If pixels are missing from an image, either by design (compressed sensing) or by accident (a corrupted image), we can estimate the values of the missing pixels via regression. The regression problem, however, may be **ill-posed** because we have fewer **observations than parameters (pixels) to estimate**. We can address the ill-posed nature of the problem by adding a **sparsity-inducing regularization**.

Our task

In our compressed sensing image recovery task, you will apply regularized regression for reconstruction of corrupted images. Local segments in natural images tend to be sparse when modeled using a Discrete Cosine Transform (DCT) basis, meaning that a relatively small number of DCT coefficients are non-zero. We will leverage this domain knowledge by breaking the full image into a series of image chips and imposing a sparsity constraint on the reconstruction each chip through L_1 regularization on the DCT model coefficients. L_1 regularization is chosen because it tends to produce models that are sparse in the coefficients. The overview of the process is shown in Figure 1

The goal of this task is to successfully apply LASSO regularized regression for image reconstruction and investigate the impact of the degree of corruption (proportion of pixels that are missing) on the quality of the reconstruction.

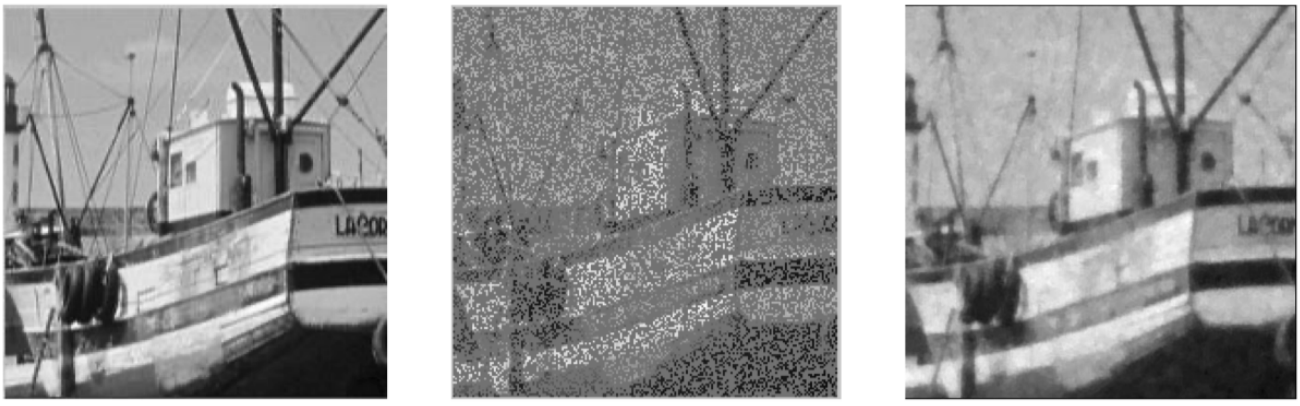

Figure 1: Compressed sensing image recovery overview. From left to right: Original Image; Sampled or corrupted image; Recovered image.

We will break down this task into multiple steps, addressing each step in the subsequent homework questions. In this homework, will will go through the first 4 steps:

- 1) Introduce 1-D DCT
- 2) Introduce 2-D DCT
- 3) Split an image into image blocks
- 4) Applying LASSO on an single image block

Our Data

The image of a Fishing Boat is provided.

We will use the smaller image "fishing_boat" in this assignment (HW #2).

We provide starter code for loading the image.

Introducing DCT

1-D Discrete Cosine Transform (DCT) Model

Mathematical Background

A 1-dimensional signal C (e.g., audio) with N samples can be represented (approximated) as a weighted sum of D 1-dimensional basis functions, each of which also has N samples:

$$\begin{bmatrix} | \\ \mathbf{C} \\ | \end{bmatrix} = \alpha_1 \begin{bmatrix} | \\ \mathbf{T_1} \\ | \end{bmatrix} + \alpha_2 \begin{bmatrix} | \\ \mathbf{T_2} \\ | \end{bmatrix} + \ldots + \alpha_D \begin{bmatrix} | \\ \mathbf{T_D} \\ | \end{bmatrix}$$

$\mathbf{C}$ is a column vector with N elements; Each $\mathbf{T_d}$ is a column vector with N elements.

In vector-matrix notation, $\mathbf{C} = \mathbf{T}\alpha$:

$$\begin{bmatrix} C_1 \\ C_2 \\ \ldots \\ C_N \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ T_1 & T_2 & \ldots & T_D \\ | & | & | & | \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \ldots \\ \alpha_D \end{bmatrix}$$

In this equation, each element of $\mathbf{C}$ is a sample of the 1-D signal; each column of $\mathbf{T}$ is a basis function (the elements of each basis function (column) correspond to the samples of $\mathbf{C}$); each element of α is the weight for the corresponding basis function.

1-D DCT

The 1-D Discrete Cosine Transform (DCT) is a mathematical technique used to represent a signal as a weighted sum of cosine functions oscillating at different frequencies. It is commonly used in image and signal compression applications, such as JPEG, due to its ability to compact most of the signal's energy into a small number of coefficients.

For a 1-D signal $g[x]$ of length N, (g is the input signal, G is the transformed signal, which is also called DCT coefficients), the Discrete Cosine Transform (DCT coefficients $G[u]$) is defined as:

$$G[u] = \alpha_u \sum_{x=1}^{N} g[x] \cos\left(\frac{\pi(u-1)(2x-1)}{2N}\right), \quad u = 1, 2, \ldots, N$$

where:

$$\alpha_u = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } u = 1 \\ \sqrt{\frac{2}{N}}, & \text{if } u > 1 \end{cases}$$

The Inverse Discrete Cosine Transform (IDCT) to reconstruct original 1-D $g[x]$ from DCT coefficient $G[u]$ is given by:

$$g[x] = \sum_{u=1}^{N} \alpha_u \cos\left(\frac{\pi(u-1)(2x-1)}{2N}\right) G[u], \quad x = 1, 2, \ldots, N$$

where the following part is called basis function:

$$\alpha_u \cos\left(\frac{\pi(u-1)(2x-1)}{2N}\right), \quad x = 1,2,\ldots,N$$

We can regard g as $\mathbf{C}$, basis function as $\mathbf{T}$, G as α containing all weights for the corresponding basis function here. Therefore, the IDCT process can also be represented in the form of $\mathbf{C} = \mathbf{T}\alpha$.

Question

1. Demonstrate you understand the basic idea 1-D DCT by solving the below questions:

(10) (a) **Basic Calculation:** Given a signal $g[x] = [1,2,3,4]$, compute the 1-D DCT coefficients manually using the formula for $N = 4$. Use $u = 1,2,3,4$ and assume the normalization factor α_n as provided above. (Answer round to one decimal place)

(5) (b) **Visualizing DCT Basis Functions**: Write Python code to generate and display the 1-D DCT basis functions for $N = 8$.

- Plot all 8 basis functions in a single figure.
- Label the axes appropriately, and include a legend to identify each basis function.

(5) (c) Submit a PDF print-out of your code.
(Submitting a URL for a cloud-based repository is insufficient.)

2. Demonstrate you understand the basic idea 1-D IDCT by solving the following questions:

You are given the following DCT coefficients $G = [14, -3.5, 3.5, 0]$ for a signal, and the ground truth signal $g(x) = [4,6,8,10]$.

(5) (a) **Signal Recovery using Full DCT coefficient**: Perform the inverse DCT (IDCT) using the given DCT coefficients $G = [14, -3.5, 3.5, 0]$ to reconstruct the signal $g_1(x)$.

(5) (b) **Signal Recovery using Sparse DCT coefficient**: Perform the inverse DCT (IDCT) again using the modified DCT coefficients $G_{\text{modified}} = [14, -3.5, 0, 0]$, where the last two coefficients are set to 0. The reconstructed signal is called $g_2(x)$

(5) (c) **Results Comparsion**: Calculate the MSE between the ground truth $g(x)$ and $g_1(x)$, $g(x)$ and $g_2(x)$, find the reconstructed signal with smaller MSE.

2-D Discrete Cosine Transform (DCT) Model

Mathematical Background

A 2-dimensional signal C with $N * M$ samples can also be represented (approximated) as a weighted sum of D 2-dimensional basis functions, each of which also has $N * M$ samples:

$$\mathbf{C} = \alpha_1 \mathbf{T_1} + \alpha_2 \mathbf{T_2} + \ldots + \alpha_D \mathbf{T_D}$$

$\mathbf{C}$ is a N-by-M matrix; each $\mathbf{T_d}$ is a N-by-M matrix; each α_d is a scalar.

The matrices $\mathbf{C}$ and $\mathbf{T_d}$ can be reshaped into column vectors with length NM (this process is termed **"rasterization"**; the matrix is "rasterized").

With $\mathbf{C}$ and $\mathbf{T_d}$ rasterized as column vectors, in vector-matrix notation, $\mathbf{C} = \mathbf{T}\alpha$:

$$\begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_{NM} \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ T_1 & T_2 & \dots & T_D \\ | & | & | & | \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_D \end{bmatrix}$$

In this equation, each element of $\mathbf{C}$ is a sample of the 2 -D signal; each column of $\mathbf{T}$ is a basis function(the elements of each basis function (column) correspond to the samples of $\mathbf{C}$); each element of α is the weight for the corresponding basis function.

2-D DCT

Every image can be expressed as a 2-D matrix, each element within the image is modeled using the discrete cosine transform (DCT). The DCT provides a set of basis vector matrix that are parametrically characterized by the variables x and y which are indices for horizontal and vertical pixel location within the image, and the variables u and v, which can be viewed as spatial frequency in the horizontal direction and spatial frequency on the vertical direction, respectively.

Similar to 1-D DCT, the 2-D DCT basis function for a $P \times Q$ image is given by:

$$\alpha_u \beta_v \cos \frac{\pi(2x-1)(u-1)}{2P} \cos \frac{\pi(2y-1)(v-1)}{2Q},$$

where

$$\alpha_u = \begin{cases} \sqrt{1/P}, & u = 1 \\ \sqrt{2/P}, & 2 \le u \le P \end{cases}$$

and

$$\beta_v = \begin{cases} \sqrt{1/Q}, & u = 1 \\ \sqrt{2/Q}, & 2 \le u \le Q \end{cases}.$$

and the IDCT is given by:

$$g[x,y] = \sum_{u=1}^{P} \sum_{v=1}^{Q} \alpha_u \beta_v \cos \frac{\pi(2x-1)(u-1)}{2P} \cos \frac{\pi(2y-1)(v-1)}{2Q} G[u,v],$$

In the expression for the DCT basis vector matrix for a $P \times Q$ image, x and u take on the integers from 1 to P, and y and v take on the integers from 1 to Q. Each basis image represents the basis for the spatial frequency pair (u,v) over all (x,y) pixel locations.

Question

1. Demonstrate you are able to conduct 2-D IDCT.

(5) (a) **Print the basis functions** for $(u,v) = (3,4)$ and $(u,v) = (5,2)$. Assume $P = Q = 8$.

(5) (b) **Signal Reconstruction**: Given the 2-D DCT coefficients matrix:

$$G = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},$$

write code to 1) display all basis functions and 2) print the reconstructed 2-D signal $g[x,y]$ using the inverse 2-D DCT formula:

$$g[x,y] = \sum_{u=1}^{P} \sum_{v=1}^{Q} \alpha_u \beta_v G[u,v] \cos\left(\frac{\pi(2x-1)(u-1)}{2P}\right) \cos\left(\frac{\pi(2y-1)(v-1)}{2Q}\right),$$

where $x,y \in \{1,2\}$ and $P = Q = 2$.

(5) (c) Submit a PDF print-out of your code.
(Submitting a URL for a cloud-based repository is insufficient.)

LASSO on Image Block

Mathematics Behind Image Recovery

Since an image g (i.e., $\mathbf{C}$) can be easily reconstructed using the given basis functions $\mathbf{T}$ and the DCT coefficients G (i.e.,α) through the formula $\mathbf{C} = \mathbf{T}\alpha$, and since the basis functions $\mathbf{T}$ are fixed for a given image size $P \times Q$ (why?), the problem of image reconstruction can be reframed as finding the DCT coefficients G (i.e.,α).

We can easily calculate the DCT coefficient α using $\alpha = \mathbf{T}^{-1}\mathbf{C}$ when complete image g (i.e., $\mathbf{C}$) is given. However, when we only have corrupted image g' (i.e., $\mathbf{C}'$), i.e., when only samples of C are given, can we approximate the DCT coefficient α?

In fact, sampled image $\mathbf{C}'$ leads to an underdetermined linear system: $\mathbf{C}' = \mathbf{T}\alpha$ (why?), and we need to estimate DCT coefficients α by solving the underdetermined linear system $\mathbf{C}' = \mathbf{T}\alpha$. However, underdetermined systems generally have an infinite number of solutions, which means we need to impose additional constraint(s).

Our constraint is sparsity (i.e., small number of non-zero coefficients) in this case, since natural images tend to be sparse in the DCT domain. Once DCT coefficients are estimated, recover the full image by $\mathbf{C} = \mathbf{T}\hat{\alpha}$ ($\hat{\alpha}$ is sparse DCT coefficients).

Thus, using a simple example to illustrate, the full size image construction process using DCT coefficient can be represented by:

$$\begin{bmatrix} x \\ x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \begin{bmatrix} x \\ x \\ x \\ x \\ x \end{bmatrix}$$

With corrupted image g', we aim to solve an underdetermined linear system to estimate DCT coefficient using pixels from the corrupted image:

$$\begin{bmatrix} x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \begin{bmatrix} x \\ x \\ x \\ x \\ x \end{bmatrix}$$

by putting sparsity constraints on DCT coefficients α, we convert the problem to estimating sparse DCT coefficients

$$\alpha = \begin{bmatrix} \mathbf{0} \\ x \\ x \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

using :

$$\begin{bmatrix} x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ x \\ x \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

LASSO

Casting DCT Image Models to LASSO

We can impose sparsity on DCT coefficients α by L1-norm regularization (LASSO):

$$\min_{\alpha} \|\mathbf{C}' - \mathbf{A}\alpha\|_2^2 + \lambda\|\alpha\|_1$$

Using LASSO in Practice

Although DCT coefficients can be estimated by imposing sparsity, in practice, the DCT coefficients of a large image are often not inherently sparse. To address this, we will divide the large image into smaller blocks. By working with these smaller blocks, each block is more likely to have a limited number of non-zero DCT coefficients. Notably, the non-zero DCT coefficients will vary across blocks, capturing the local features of the image more effectively.

In this homework, we will use 8 pixel by 8 pixel blocks (64 pixels total in each block). Henceforth, we will refer to this as a 8 x 8 block.

Also, in order to leverage standard LASSO routines, it is necessary to cast our 2D image model to a 1D model that is compatible with standard LASSO implementations. This can be accomplished by rasterizing both the image blocks and all the DCT basis blocks. A 2D matrix is rasterized to a 1D vector by stacking all the columns from the matrix on top of each other, in order, to create a vector.

Reconstructing Missing Pixels

To create corrupted image, we remove image pixels from each image block (target variable) and also remove the the basis vector matrix which corresponds to the missing pixels in image block. This process can also called sampling from image (image blocks). And we apply LASSO to estimate the DCT model weights for each image block. The estimated model weights are then used in the model to estimate the missing pixels in each image block. Once we reconstruct every image block, we can combine all image blocks to get the full reconstructed image.

Question

Download the project image "fishing_boat" from Canvas. (They are attached to the HW #2 assignment.)

1. Demonstrate you are able to select a desired K×K block from within an image, and simulate corruption of that block.

(5) (a) Display the original image in gray scale. (Do not display the image in pseudo color.)

(5) (b) Display the first 8×8 image block on the top-left at $(x,y) = (x^*, y^*)$, where

$$x^* = 8 * 3 \qquad y^* = 8 * 6$$

(This indexing presumes the pixels in the image are numbered 1 through N. You will need to subtract 1 when working with 0-indexed arrays.)

Tip1: The fishing_boat image divided by 8×8 blocks looks like Figure 2

Tip2: The example 8×8 block looks like Figure 3

(15) (c) Display your corrupted 8×8 block with $S = 50$, $S = 30$ and $S = 10$ (S: sampling size, i.e., number of remaining pixels after removing the rest of pixels) sensed pixels. (Display the missing pixels in a contrasting color. Do not display the missing pixels in either black or white (which are valid pixel colors for a gray scale image.))

(5) (d) Submit a PDF print-out of your code.
(Submitting a URL for a cloud-based repository is insufficient.)

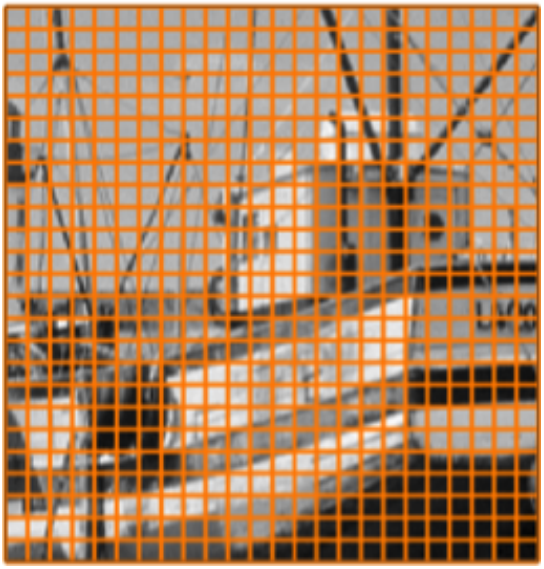

Figure 2: Fishing boat divided by blocks

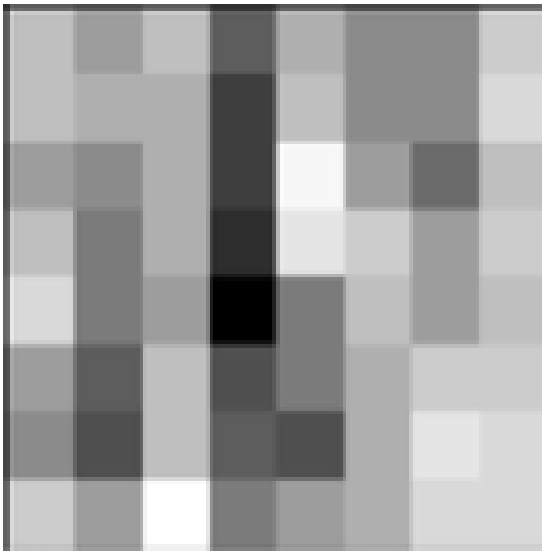

Figure 3: Example 8×8 block.

2. Demonstrate you are able to apply LASSO to estimate model weights for a single corrupted image block.[4]

   Again, see starter code for how to solve LASSO.

   Use $\lambda = 0.01$ for the regularization parameter,[5] and apply LASSO to your corrupted image chip with $S = 30$ sensed pixels (Question 1(c), above) for the given regularization parameter.

(20) (a) For each of the 5 largest magnitude weights, display the corresponding basis function block and provide the u index, the v index, and the estimated weight for that block in the figure (or subfigure) title.

(20) (b) Display the reconstructed image block alongside the original (uncorrupted) image block.

(5) (c) Provide the MSE between the reconstructed image block and the original image block.

[4] We are building up the task step-by-step. This step does not incorporate either a search for the best regularization parameter or cross-validation; a future homework (Homework 3) will include cross-validation to identify the best regularization parameter.

[5] At this phase of the task we are not concerned with the optimality of the regularization parameter value; we a concerned with your progress toward being able to apply LASSO to estimate model weighs and then apply those model weights to reconstruct an image block.

(5) (d) Submit a PDF print-out of your code.
(Submitting a URL for a cloud-based repository is insufficient.)

Total weight = 235