

2025_final_project

May 7, 2025

1 2025 Spring ORF307 Final Project

Instructor: B. Stellato

As: Yanjun Liu, Qishuo Yin, Chenyu Yu, and Kaiwen Zhang

1.1 Instructions

Exam files: the exam files are available to download at [THIS LINK](#).

Date and time: From May 8, 2025 at 00:01am to May 11, 2025 at 7:00pm.

- Total time after download: 24 hours. No late submissions allowed. Note that the deadline is either 24 hours after download, or May 11, 2025 at 7:00pm, **whichever comes first**.

1.2 Exam Rules

- You are allowed to use all course materials on the midterm (lecture notes, books, precept materials, code, and homeworks). But you **cannot use internet to search for answers**.
- You have to justify all your answers. If you use code from the course materials, you have to explain what each step means.
- You cannot communicate with anyone during the exam.
- No late submissions allowed. Make sure your submission goes through on time. You can resubmit as many times as you like until your time expires.
- The exam is to be submitted electronically on Gradescope before 7:00pm on the final day.

2 Final Exam

2.1 Introduction

For this final project, you will work with the Fashion MNIST dataset including thousands of images of clothes, which we have provided in CSV format.

You will work with a pre-processed version of this dataset where each label is +1 if the image is a shirt and -1 otherwise.

To load the dataset, ensure the following files are in a folder called `data/` in your working directory: `X_train.csv`, `y_train.csv`, `X_test.csv`, `y_test.csv`. Then run the following code to import the packages and load the data

```
[2]: import numpy as np
np.set_printoptions(precision=4) # Print few decimal places
np.set_printoptions(suppress=True) # Suppress scientific notation
import cvxpy as cp
import pandas as pd
from numpy.linalg import cholesky as llc
import matplotlib.pyplot as plt
```

```
[3]: X_train = pd.read_csv("data/X_train.csv").values
y_train = pd.read_csv("data/y_train.csv").values.ravel()
X_test = pd.read_csv("data/X_test.csv").values
y_test = pd.read_csv("data/y_test.csv").values.ravel()

n_train, m = X_train.shape
n_test, m = X_test.shape

print("-" * 50)
print("Fashion MNIST dataset")
print("-" * 50)
print(f"Number of features: {m}  ({int(np.sqrt(m))} x {int(np.sqrt(m))}
    ↳ pixels)")
print(f"Training set:")
print(f"    • Samples: {n_train}")
print(f"    • Value range: [{X_train.min():.2f}, {X_train.max():.2f}]")
print(f"Test set:")
print(f"    • Samples: {n_test}")
print(f"    • Value range: [{X_train.min():.2f}, {X_train.max():.2f}]")
```

```
-----
Fashion MNIST dataset
-----
```

```
Number of features: 784  (28 x 28 pixels)
```

```
Training set:
```

- Samples: 5000
- Value range: [0.00, 255.00]

```
Test set:
```

- Samples: 1000
- Value range: [0.00, 255.00]

Now, let's visualize a couple of images

```
[4]: def visualize_images(*images):
    n_images = len(images)
    fig, axes = plt.subplots(1, n_images, figsize=(14,6))
    if n_images == 1:
        axes = [axes]
    for i, (ax, img) in enumerate(zip(axes, images)):
        img_resized = img.reshape(28, 28)
```

```

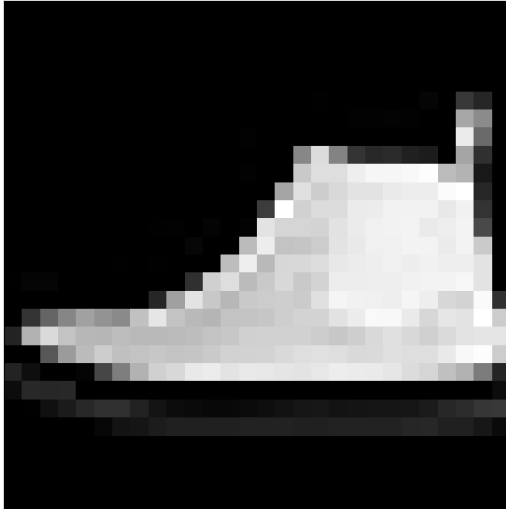
im = ax.imshow(img_resaped, cmap='gray')
ax.set_xticks([])
ax.set_yticks([])
plt.tight_layout()

```

```

[5]: example_boot = X_train[3]
example_shirt = X_train[0]
visualize_images(example_boot, example_shirt)

```



3 (40 points) Build a Support Vector Machine to correctly classify shirts

1. (5 points) Formulate a Support Vector Machine problem to find a hyperplane (a, b) where $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$ to classify shirts. Given sample $x^{(i)}$, your hyperplane should be $a^T x^{(i)} + b > 0$ if label $y^{(i)} = 1$ (shirt) or $a^T x^{(i)} + b < 0$ if label $y^{(i)} = -1$ (not a shirt). You should include a second term in the objective, weighted by parameters $\lambda > 0$ to regularize the ℓ_1 -norm of a .

Note: Remember to normalize the misclassification error by the total number of training samples.

2. (8 points) Run a cross-validation procedure: solve the problem using CVXPY and CLARA-BEL solver for $\lambda = (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05)$ and pick the model that has the lowest test error.

Note, these problems are quite large so it may take a few seconds to solve them. (you may use the argument `verbose=True` to print out the solver progress).

Here is a utility function to check the test error. For example, you can use it as `error(X_test, y_test, a, b)`

```
[6]: def error(X, y, a, b):
      y_pred = np.sign(X @ a + b)
      return np.mean(y_pred != y)
```

3. (7 points) Write the training problem as a linear program in inequality form (get rid of all norms and maxima in the formulation)
4. (10 points) Write the dual problem in standard form. Solve it and find the optimal dual variables for $\lambda = 0.025$.
5. (10 points) As you noticed, solving this problem can be expensive, but you still would like to carefully tune the parameter λ . To avoid unnecessary computations, you would like to understand if, when you vary λ within the interval $[0.025, 0.05]$, the optimal a and b for your SVM would change or not. To check this, you would like to use the optimal dual variables you have found in the previous problem. Using complementary slackness, verify whether the optimal solution for the SVM for $\lambda = 0.05$ and the optimal dual variables you have found for $\lambda = 0.025$ can form an optimal primal-dual pair. From this, what would you expect for the optimal a and b for $\lambda \in [0.025, 0.05]$? Do you need to try out the values of λ in that interval or not?

Hint: To check for non zero entries, remember to allow for a small tolerance to account for numerical errors. You can use a tolerance of $1e-05$.

4 (25 points) Computing adversarial examples

We now would like to verify the robustness of our classifier by looking at adversarial examples. To do so, we will consider all images close to a *representative image* and see if appropriate pixel perturbations could confuse our classifier.

1. (10 points) Formulate the problem of finding the closest image, in terms of the ℓ_1 -norm, to the shirt one at index `example_shirt = 0`, that gets misclassified.

Note: images have pixel intensities between 0 and 255.

2. (5 points) Formulate the problem as a linear program in inequality form without including any norm.
3. (10 points) Solve problem and compare the resulting image with the example shirt above. How many pixels are modified? What is the ℓ_1 -norm distance between the images? Plot the images side by side using the `visualize_images` function above and describe what's different.

Note: To count which pixels are modified, use a tolerance of $1e-05$ to account for small numerical errors.

5 (35 points) Production planning with quality control via SVM

You are a clothing manufacturer that produces a certain amount of shirts (`n_shirts`) daily but can only inspect 40% of them due to resource constraints. Your excellent shirt classifier from part 1 will turn out to be helpful to decide which ones to inspect. The company incurs in various costs:

- Inspection cost: $c = 10$ per item inspected

- False negative (error) cost: $d = 20$ per defective item that reaches a customer (which will not be happy about the purchase)
- True positive (correct) cost (savings): $s = 15$ from catching a defective item before shipping

For each shirt i , you need to decide whether to inspect it using binary variable $w_i \in \{0, 1\}$. We will use the trained SVM problem to suggest to inspect the item if it is not classified as a shirt. Specifically, we transform the output of the SVM into a probability with values between 0 and 1 using the following function. For an image i , the probability of the product to be defective (i.e., not a shirt) is

$$p^{(i)} = \sigma(-(a^T x^{(i)} + b)),$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function.

1. (5 points) Calculate the predicted probabilities for all training shirt images. Use the `X_shirt` images data defined here for convenience.

```
[7]: # get shirt training and testing datasets
shirt_indices_train = np.where(y_train == 1)[0]
X_shirt_train = X_train[shirt_indices_train]
n_shirt_train = X_shirt_train.shape[0]
inspection_capacity_train = int(0.4 * n_shirt_train)

shirt_indices_test = np.where(y_test == 1)[0]
X_shirt_test = X_test[shirt_indices_test]
n_shirt_test = X_shirt_test.shape[0]
inspection_capacity_test = int(0.4 * n_shirt_test)
```

2. (5 points) Run a naive approach on the training dataset. Inspect the items with the largest probability of being defective. Visualize the top-3 items. What are their characteristics?
3. (10 points) Define a mixed-integer optimization problem to minimize the expected cost on the training dataset, i.e., the sum of all the components of the objective, weighted by the defective probabilities. Ensure that the number of inspected products is less than `inspection_capacity`.
4. (15 points) Solve the problem for both training and testing datasets, and compare the expected cost with the naive approach. You can use cvxpy solver SCIPY (which internally calls the HIGHS MIP solver). Which approach is better? Why?

```
[ ]:
```