Project B: Classifying Images

Last modified: 2023-11-14 09:58

Status: RELEASED.

Due date: Thu. Nov. 30, 2023 by 11:59pm ET (Boston time)

Jump to:

Starter Code   Problem 1: MLP Classifier   Problem 2: Open Challenge   Rubric

Turn-in links:

- Team formation (by Tue 11/7): https://forms.gle/afXQ9BBJARMQmrFV7
- PDF report turned in to: https://www.gradescope.com/courses/596466/assignments/3629695/ *(will open after Thu 11/9)*
- ZIP file of test-set predictions for Problem 1 Leaderboard: https://www.gradescope.com/courses/596466/assignments/3629696 *(will open after Thu 11/9)*
- ZIP file of test-set predictions for Problem 2 Open-Ended Leaderboard: https://www.gradescope.com/courses/596466/assignments/3629697/ *(will open after Thu 11/9)*
- Reflection on Project B (at very end): https://forms.gle/YcnADakeE8xYRExx9

Overview

This is a four week project with lots of open-ended programming. Get started right away!

Suggested intermediate deadlines:

- by Tue. 11/7: Form teams (ideally, same as last time). Signup here: https://forms.gle/afXQ9BBJARMQmrFV7
- by Thu. 11/14: Complete Problem 1 code/experimentation + leaderboard submission
- by Tue. 11/21: Complete Problem 1 writeup
- by Tue. 11/28: Complete Problem 2 code/experimentation + leaderboard submission
- by Thu. 11/30: Complete Problem 2 writeup

Team Formation

By start of class on Tue 11/07, you should have identified your partner and signed up here:

- ProjectB Team Formation Form

Even if you decide to work alone, you should fill this form out acknowledging that.

In this project, you are encouraged to work as a team of 2 people. If you prefer, you can work individually. Individuals still need to complete all the parts below and will be evaluated no differently than teams. We strongly recommend working in pairs to keep your workload manageable.

If you need help finding a teammate, please post to our "Finding a Partner for Project B" post on Piazza.

What to Turn in

What to Turn In

Each team will prepare *one* PDF report covering all problems.

- Each submission must be associated with *all* team members' names
- Suggested length 5 pages (upper limit is 7 pages)
- We hope the report is a stand-alone representation of your work that you could show to a future employer as evidence of your ML skills.
- Should be **human-readable**. No code. Not a jupyter notebook export.
- This document will be manually graded according to our rubric
- Can use your favorite report writing tool (Word or G Docs or LaTeX or ....)
- Should have each subproblem marked via the in-browser Gradescope annotation tool)

Each team will prepare a ZIP file of test-set predictions for each of Problem 1 and Problem 2.

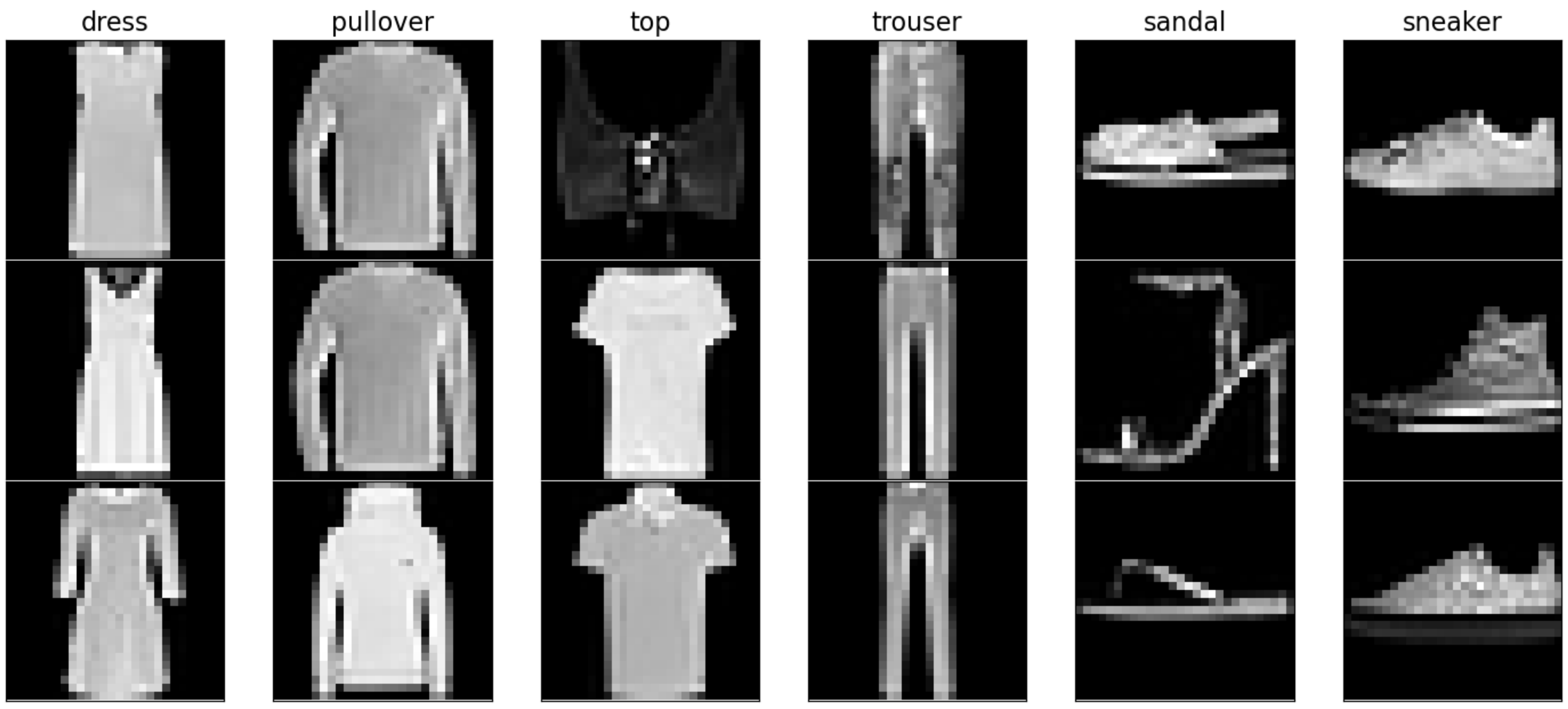
- Each submission must be associated with *all* team members' names
- Each submission ZIP will contain just one plain text file: `yhat_test.txt`
    - This file should have one line for each image of the provided `test` set, in the same order.
    - Each line contains just the text class name (e.g. 'sneaker') representing the predicted class.
    - Use the standardized class names in Dataset: FashionMNIST-6

Each *individual* will turn in a *reflection form* (after completing the report).

- Link: https://forms.gle/YcnADakeE8xYRExx9

Dataset: FashionMNIST-6

You will be constructing a classifier to map a given grayscale image to the type of clothing it depicts.



ProjB dataset: FashionMNIST-6, a custom subset of FashionMNIST. Each image is 28x28 pixels, showing the silouhette of an item of clothing. There are 6 possible classes. We show you 3 example images of each.

We've built a customized subset of the Fashion MNIST dataset by Xiao, Rasul, and Vollgraf of Zalando Research.

The provided data is available here: https://github.com/tufts-ml-courses/cs135-23f-assignments/tree/main/projectB/data_fashion

The `data_fashion` folder contains images of the following 6 classes

1. dress (original class id 3)
2. pullover (original id 2)
3. top (original id 0)
4. trouser (original id 1)
5. sandal (original id 5)
6. sneaker (original id 7)

These classes represent "silhouette types" of the images in the fashion catalog of one of Europe's largest online fashion stores.

We have separated images (and labels) into a training, validation, and test set. These sets have 2102, 600, and 600 examples respectively.

Note 1: We will not release the `test` data images until Thu 11/16. This is to prevent "leakage" and ensure you focus on proper pipeline development. Even without this, you can still complete all but the last step of Problem 1 and 2. You can assume `test` and `valid` are very similar in composition.

Note 2: The original FashionMNIST dataset has more classes and more examples. We've customized this subset to make the problem interesting but also a bit simpler (only 6 classes to track, not 10).

Features (inputs to your classifier)

Each set (train/valid/test) has all its images stored in one CSV file named `[set]_x.csv` file.

Each 28x28 image is stored as one *row* of 784 numbers. The first 28 elements in the row represent the row of pixels at the top of the image from left to right, followed by each subsequent row from top to bottom. Each element is a greyscale pixel value from 0 (black) to 255 (white).

Labels (outputs)

Each provided set (train/valid) has all class labels stored in one CSV file named `[set]_y.csv`.

Each row of this file contains the string name of the class of the corresponding image (image stored in the same row in `[set]_x.csv`).

Leaderboard Performance Metric

You will build classifiers that can produce for each test image (indexed by i) a single predicted class $\hat{y}_i \in$ `{'dress', 'pullover', 'top', 'trouser', 'sandal', 'sneaker'}`.

We will then use *balanced accuracy* to assess the overall classifier quality.

For a test set of size N and $C = 6$ classes, we compute balanced accuracy as

$$\text{BalAcc}(y_{1:N}, \hat{y}_{1:N}) = \frac{1}{C} \sum_{c=1}^{C} \frac{\text{TP}_c(y_{1:N}, \hat{y}_{1:N})}{N_c(y_{1:N})}$$

where $\text{TP}_c(\cdot)$ counts *true positives* for class c (number of correctly classified examples whose true label is c), and $N_c(\cdot)$ counts the total number of examples with true label c.

Balanced accuracy ranges from 0.0 - 1.0, with higher values indicating a better model. A random classifier that picks one of the C labels uniformly at random would get a score of 1/C.

In practice, please report *rounded* to 3 decimal places.

Starter Code

Link to starter code: https://github.com/tufts-ml-courses/cs135-23f-assignments/tree/main/projectB

Aside from the data, the starter code contains

- `load_and_plot_data.py` shows how to load data and plot images
- `save_rand_predictions.py` shows how to save predictions in the format required by our leaderboard
- `calc_bal_acc.py` shows how to compute balanced accuracy given a saved set of predictions

We deliberately do not provide too much starter code, so that you can practice working from scratch.

Problem 1: MLP Classifier

Your challenge is to train an MLP classifier that can distinguish the 6 different classes of clothes while overcoming a challenging training set.

In your PDF report, include the following sections:

1A : Dataset Exploration

Table 1A with caption

Describe the composition of your training and validation set with a table and corresponding caption.

- *Rows*: There should be one row for each class label, in the same order as listed above in the Dataset: FashionMNIST-6 section
- *Cols*: Each column should report the count of images in one of the provided sets. There should be one column for the *train* set, and another column for the *validation* set.

Your caption should state the major challenge you can foresee in building classifiers for the provided training data.

1B : Model Development (including training and hyperparameter selection)

Develop an MLP with **at most 1 hidden layer** that can solve the 6-class fashion problem as well as possible.

Stipulations:

- You should use `MLPClassifier` with at most one hidden layer (but any number of units)
- You can only use the provided `train` data for model fitting. No augmentation/duplication allowed here in 1B.
- You can only use the provided `valid` data for hyperparameter search

In your report, provide brief paragraphs answering the following questions

- What (if any) preprocessing do you perform on the provided images? Why?
- What performance metric are you optimizing for your hyperparameter search? Why?
- How did you select the MLP's architecture? (how many units? what activation function for hidden and final layers?)
- How did you select the MLP's optimization? (including algorithm, plus optimization-specific hyperparameters like learning rate or batch size)

Strive for descriptions that are concise yet complete, and could be reproduced by a classmate.

Table 1B with caption

Include a table reporting your chosen performance metric on the training and validation sets for at least 5 possible hyperparameter configurations. If possible, strive to show 5 configurations that span under- and over-fitting regimes for one key hyperparameter (holding others fixed to their best values).

In a brief caption, provide the major takeaways from your hyperparameter search.

1C : Model Analysis

Table 1C with caption

For your final model from 1B, provide ONE table where you

- visualize performance on the validation set with a confusion matrix. True labels are rows. Predicted labels are cols.
- use the title of the table to report your model's balanced accuracy on the validation set

Be sure the confusion matrix is **human readable** by using the class names for each possible decision (e.g. rows and columns should be labeled with text names like 'sneaker' or 'sandal'. Always use the same order as in Dataset: FashionMNIST-6)

In a caption below the confusion matrix, write a few sentences describing the model's limitations and hypothesizing why these limitations occur.

1D : Training Set Modification

Make a plan to modify the training set by **duplicating** (but not altering) some of its images and corresponding labels to address the issues you find in 1C.

In your report, describe in a coherent paragraph

- *how* you edited the training set via duplication. Be sure to report the total number of images of each class.
- *why* you chose this strategy. What do you expect will happen?

1E : Duplicated-Data Model Development

Repeat **exactly** your full development pipeline from 1B (including preprocessing (if any), training, and hyperparameter search) on the modified training set from 1D.

Table 1E with caption

Include a table reporting your performance metric on your modified train set as well as the original validation set for at least 5 possible hyperparameter configurations.

In a brief caption, provide the major takeaway messages of your hyperparameter search.

1F : Modified Model Analysis

Table 1F with caption

For your best model from 1E, provide one table where you

- visualize performance on the validation set with a confusion matrix (same style to 1C)
- use the title of the table to report your model's balanced accuracy on the validation set

In the caption, write 1-2 sentences describing:

- Contrasts between these results and the results from 1C.
- Why the results do or do not support your modification procedure

1G : Submit to Leaderboard and Record Test-Set Performance

Apply your best classifier pipeline from 1E to each test image in `test_x.csv`. Make a hard prediction for each image into one of the 6 possible classes, storing the *class name* (not integer id). Follow the What-to-turn-in instructions and submit your predictions to the Problem 1 Leaderboard.

In your report, include a brief summary paragraph stating your ultimate test set performance. Discuss if your performance is notably better or worse than you found in your own experiments on the validation set, and reflect on why you think that might be.

Problem 2: Open-Ended Challenge

In this *open-ended* problem, you will apply new methods to the same dataset.

Here are some concrete examples of methods you could try:

- Data augmentations (horizontal flip, translation, rotation, etc.).
    - This can be done with basic NumPy array operations or a Python package (skimage, torchvision, etc.) that handles image transformations.
- Feature transformations, such as
    - A pretrained deep network such as MobileNet V3.
    - Dimensionality reduction techniques such as PCA.
- Try a different multiclass classifier in sklearn (nearest neighbor, random forest, XGBoost, etc.).
    - Be sure you understand enough about this classifier to define a reasonable hyperparameter search strategy.
- Try an MLP with **2 or more** hidden layers
    - Beware! Larger models would incur longer training times and are prone to overfitting, so be sure to carefully think about your hyperparameters.

For full credit, we expect that at least one of the following is substantially different from Problem 1. Any of the above suggestions would be considered substantially different.

- data preprocessing / augmentation
- feature representation (e.g. features from pretrained neural net)
- classifier method

Furthermore, your chosen method must be plausibly motivated by improving your classifier's performance over Problem 1.

Data remixing rules. You can use any provided images in `train` or `valid` to develop any part of your model. You are free to combine the training and validation sets however you like. Be sure you have a way to avoid overfitting. If you do remix the provided labeled data, please be sure in 2A below to report *exactly* what you've done and why.

Data sourcing rules. In this problem, you are free to use any images from the provided train set or validation set to develop your model however you see fit. However, you can **only** use the provided data (or augmentations of that data). You should not use any additional Fashion MNIST data from external sources.

Code restrictions. You can import any Python packages or libraries that you have legal license to use. You should give credit in your report (via an academic citation) to any resources that substantially help your progress.

You should write the primary pipeline used to train your model yourself. You should be able to walk another person in the class thru the code used to do all model development, and also be able to articulate the strengths and weaknesses of your approach.

2A : Method Description

Describe the new method(s) that you use so that another student in the class could clearly understand your motivation and reproduce your implementation. Include a hypothesis for why each new method should improve performance.

Please write one paragraph about each aspect in order:

- (i) data augmentation / preprocessing
- (ii) feature representation
- (iii) classification method

Remember, at least one of these must be investigated, but not all are required. If you did not investigate one item of the above, just write "No investigation was performed."

If you did a substantial investigation that did not work out, feel free to describe that here as well, in a clearly marked paragraph called **Attempt that did not make it to the final model**.

2B : Model Development (including training and hyperparameter search)

Well-written paragraph about your model development pipeline for the method outlined in 2A, with enough detail that another student could roughly reproduce your work, describing:

- What hyperparameters were searched? What concrete values? Why?
- How did you search hyperparameters? (e.g. if you use CV, specify how splits were made and number of folds)
- What performance metric is your search optimizing? Why?
- How did you select the model's architecture?
- How did you perform your model's training (optimization)?

Strive for descriptions that are concise yet complete.

Table 2B with caption:

Include a table reporting your chosen performance metric on your own training and validation sets for at least 5 possible hyperparameter configurations. If possible, strive to show 5 configurations that span under- and over-fitting regimes.

In a brief caption, provide the major takeaways from your hyperparameter search. What seems to be working on this data? Why?

2C : Model Analysis

Table 2C with caption

For your final model from 2B, provide ONE table where you

- visualize heldout performance with a confusion matrix
- use the title of the table to report your model's balanced accuracy on this dataset

In the caption, write 1-2 sentences describing:

- Contrasts between these results and both of your previous results from 1C and 1E.
- Why the results do or do not support your hypothesis from 2A.

Caveat: If you only used data from `train` to fit models, you can use the provided `valid` set here. However, if you called fit using a remix of provided `train`/`valid` datasets, be sure you are reporting error on a heldout set that was never used in a call to `fit`.

2D : Submit to Leaderboard and Record Test-Set Performance

Apply your best classifier pipeline from 2C to each test image in `test_x.csv`. Make a hard prediction for each image into one of the 6 possible classes, storing the *class name* (not integer id). Follow the What-to-turn-in instructions and submit your predictions to the Problem 2 Leaderboard.

In your report, include a summary paragraph stating your ultimate test set performance. Discuss if your performance is better or worse than in Problem 1, and reflect on why you think that might be.

Rubric for Overall Performance

The final grade for this project will be a weighted average:

- 88% : your report performance, using the rubric below
- 10% : your leaderboard submissions, using the rubric below
- 2% : completion of your reflection on the project

Rubric for Evaluating PDF Report

This should match the rubric outline on Gradescope for the PDF report. If for any reason there is a conflict, the official problem weights on Gradescope will be used.

- 60 points for Problem 1 Report
    - 5 for 1A
    - 15 for 1B
    - 10 for 1C conf mat analysis
    - 15 for 1D
    - 5 for 1E
    - 10 for 1F conf mat analysis
    - 1G (leaderboard submission) graded separately
- 40 points for Problem 2 Report

    - 15 for 2A
    - 15 for 2B
    - 10 for 2C
    - 2D (leaderboard submission) graded separately

Rubric for Evaluating Leaderboard Submissions

You'll submit 1 sets of predictions to our leaderboard, which will be graded as follows:

- 85% of points represent if you achieved a "reasonable" score (e.g. a standard pipeline trained using good practices)
- 15% of points awarded if you are within tolerance of the top 3 submissions in this class

Partial credit is possible in both cases via linear interpolation.

MIT License / Source on github / Powered by Pelican / ✨