Homework 3: Supervised Learning

Contents

> ⚠ **Warning**
>
> The submission of the homeworks has **NO** deadline. You can submit them whenever you want, on Virtuale. You are only required to upload it on Virtuale **BEFORE** your exam session, since the Homeworks will be a central part of the oral exam.
>
> You are asked to submit the homework as one of the two, following modalities:
>
> - A PDF (or Word) document, containing screenshoots of code snippets, screeshots of the results generated by your code, and a brief comment on the obtained results.
> - A Python Notebook (i.e. a `.ipynb` file), with cells containing the code required to solve the indicated exercises, alternated with a brief comment on the obtained results in the form of a markdown cell. We remark that the code **SHOULD NOT** be runned during the exam, but the student is asked to enter the exam with all the programs **already executed**, with the results clearly visible on the screen.
>
> Joining the oral exam with a non-executed code OR without a PDF file with the obtained results visible on that, will cause the student to be rejected.

Exercise 1: Logistic Regression on a Toy 2D Dataset

1. Generate two Gaussian clusters in $\mathbb{R}^2$ and associate them with a class depending on which cluster each point lies on, e.g.:
   - Class 0 centered at $(-2, -2)$ with variance 1,
   - Class 1 centered at $(2, 2)$ with variance 0.5.
2. Plot the dataset in 2D using `plt.scatter` so that each cluster is colored according to its class.
3. Implement logistic regression **from scratch** as did during class:

$$f_\Theta(x) = \sigma(\Theta^T x), \qquad \ell(\Theta; x, y) = -\big[y \log f_\Theta(x) + (1-y)\log(1 - f_\Theta(x))\big].$$

4. Train it using simple Gradient Descent on the full dataset. **Note:** the computation of $\nabla \mathcal{L}(\Theta; X, Y)$ for this choice of ℓ is given in the teaching note.
5. Visualize the learned **decision boundary**:
   - plot the line $\{\Theta^T x = 0\}$,
   - overlay with the dataset.
6. Comment on why the decision boundary is linear.

Exercise 2: SGD on Logistic Regression

Use the same synthetic dataset used in the previous exercise, but train logistic regression using **SGD** with the following choices:

1. Try different batch sizes:
   - $N_{\text{batch}} = 1$,
   - $N_{\text{batch}} = 10$,
   - $N_{\text{batch}} = N$ (full GD).
2. For each setting:
   - Plot the loss vs epoch,
   - Plot the classification accuracy vs epoch.
3. Compare the stability and speed of convergence over the choice of different batch sizes.
4. Why the gradients become noisier for small batches? Why larger batches give smoother curves?

Exercise 3: Evaluation Metrics on a Synthetic Dataset

Using the logistic regression model trained above:

1. Compute predicted probabilities $\hat{y}_i = f_\Theta(x_i)$.
2. Convert them to binary predictions $\hat{y}_i \in \{0, 1\}$ using threshold 0.5.
3. Compute:
    - Confusion matrix (TP, FP, FN, TN),
    - Accuracy,
    - Precision,
    - Recall,
    - F1-score.
4. Modify the threshold to:
    - 0.3,
    - 0.7, and repeat.
5. Comment on:
    - How lower thresholds increase recall and lower precision,
    - How higher thresholds increase precision and reduce recall,
    - Why classification metrics depend on the application (as discussed in class).

Exercise 4: Logistic Regression on a Real Dataset

Reproduce the pipeline from the book using the **Pima Indians Diabetes Dataset**, available at https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

This dataset contains medical measurements and a binary label indicating diabetes diagnosis.

1. Preprocess the data as done in class:
    - Download `diabetes.csv`.
    - Extract features X and labels y.
    - **Standardize the features** (mean 0, variance 1).
      Explain why normalization is required, connecting to:
        - conditioning,

        - stable optimization,
        - meaningful gradient magnitudes.
    - Add a bias column of 1s.
2. Implement logistic regression from scratch (sigmoid + BCE + gradient), and optimize using **SGD** with:
    - a batch size of 32, a learning rate of 10^{-3}, and 200 epochs.
    - for each epoch, track full-dataset BCE loss and full-dataset accuracy.
    - in a plot, visualize the behavior of Loss vs epoch and Accuracy vs epoch. Comment the results.
3. Train the same model using **Adam** (using the formulas from class):

$$\Theta_{k+1} = \Theta_k - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}.$$

   with a learning rate of 10^{-3}, a batch size of 32, and 200 epochs. Then, plot SGD vs Adam loss and accuracy curves.

4. For each method, evaluate:
    - Final accuracy,
    - Confusion matrix,
    - Precision, Recall, F1.
5. Discuss on which method converge faster, which oscillate more, and how this relates to adaptive learning rates discussed in class.

Optional Extension: From Logistic Regression to a Simple Neural Network

This optional section mirrors the final part of the supervised learning chapter.

Implement a neural network with:

- One hidden layer,
- ReLU activation,
- Sigmoid output layer.

You may reuse the implementation shown in the main text. Then:

1. Train the neural network on the **same** Kaggle dataset.
2. Track:

    - BCE loss vs epoch,
    - Accuracy vs epoch.
3. Compare and discuss the performance against logistic regression.

> **Note**
>
> This extension is *not mandatory* for passing the exam but can improve your understanding of model capacity.