Introduction to Data Science 2023
Assignment 5

Daniel Hershcovich, Stella Frank, Thomas Hamelryck, François Lauze, Zi Wang, Phillip Rust

Your solution to Assignment 5 must be uploaded to Absalon no later than April 4th, 22.00 (so you can still enjoy the Easter holiday!)

Guidelines for the assignment:

- **The assignments in IDS must be completed and written individually.** This means that your code and report must be written completely by yourself.
- Upload your report as a single PDF file (no Word .docx file) named `firstname_lastname.pdf`. The report should include all the points of the deliverables stated at the end of each exercise. Adding code snippets to the report is optional, but nice to have especially when you implement things from scratch.
- Upload your Python code. Upload it in a zip archive, even if there is only one file. Expected is either one or several "`.py`" files or a "`.ipynb`" notebook.
- We will grade using Python 3.10 and the specific package versions specified in requirements.txt. Other versions may work, or may break; using these versions is the safest. You can create an appropriate environment with all the requirements using Anaconda or reuse the same environment you had in the previous assignment and install the requirements file.
- To reuse a previous environment, 1) open a terminal or command prompt and navigate to the directory containing the requirements.txt using `cd` (e.g. `cd Desktop/courses/ids/as5`). 2) activate your environment (make sure to replace `name_of_env` with the name you specified in the previous assignment).

  ```
  conda activate name_of_previous_env
  pip install -r requirements.txt
  ```

  Otherwise, if you want to create a new environment you first need to run:

  ```
  conda create --name my_env python=3.10
  ```

  Then use the above two command lines to install the corresponding requirements.

This assignment consists of 7 exercises, each of which is worth 10 points. You will be marked based on 5 exercises of your choice. The total number of points for this assignment (i.e. the denominator) is 50 points. **Note that if you exceed 50 points, you will get more than 100% on the assignment: this is a true bonus!** So the correct strategy is to nominate your 5 best solutions as the marked exercises, and then the 2 other exercises are pure bonus points.

If you choose to do bonus exercises, please **specify at the beginning of your report** which exercises should be considered a bonus, and which are to be marked as part of the assignment.

Gradient descent

Exercise 1 (Gradient descent & learning rates)**.** Apply gradient descent to find the minimum of the function $f(x) = e^{-x/2} + 10x^2$.

Detailed instructions:

1. Manually derive the function $f(x)$ and implement a function that computes this derivative $f'(x)$.
2. Implement and apply gradient descent with learning rates $\eta \in \{0.1, 0.01, 0.001, 0.0001\}$. Your implementation should be done without any external Python packages (you may use NumPy, but it is not needed).
3. For each of the learning rates, do the following:
   (a) Take $x = 1$ as a starting point.
   (b) Visualize the tangent lines and gradient descent steps for the first three iterations (produce 4 plots for your report corresponding to gradient descent with the four learning rates). The first tangent line should be at the initial point $x = 1$.
   (c) Visualize gradient descent steps for the first 10 iterations (another 4 plots; no visualization of tangent lines in this case).
   (d) Run the algorithm until the absolute value of the gradient falls below 10^{-10} or the algorithm has exceeded 10,000 iterations. Report the number of iterations it took the algorithm to converge and the function value at the final iteration (i.e., report 8 values in total, 2 values for each of the 4 learning rates). Discuss briefly which of the learning rates is preferable and why.

Deliverables. 1) 4 plots (1 for each learning rate) visualizing the tangent lines and gradient descent steps when performing 3 iterations, 2) 4 plots (1 for each learning rate) visualizing gradient descent steps for 10 iterations, 3) Number of iterations until convergence and function values at final iteration, 4) discussion about preferred learning rate.

Logistic Regression

Exercise 2 (Logistic regression implementation)**.**
In this exercise, you will implement, run, and test logistic regression on two simple 2-class datasets that are derived from Fisher's Iris dataset and in the Appendix. You are solving a binary classification task.

1. Create a scatter plot for each training dataset with different colors for the two classes. What do you observe?
2. Train a logistic regression model for each of the two Iris datasets described below via `PyTorch` using stochastic gradient descent (SGD) and the following hyper-parameters: batch size $B = 64$, learning rate $\eta = 0.1$, number of steps $N = 10,000$.
3. For both datasets, report the training and test accuracy. Furthermore, report the three parameters of the final logistic regression models.
4. Repeat step 2 and 3, but this time use batch size $B = 4$. Describe how this change affects your results. Also briefly discuss the implications of this change on the number of training epochs. It is not necessary to report the parameters of your models for this part of the exercise.

The datasets we use are modified subsets of the Iris dataset. Each dataset consists of two files, a training and a test set. The first pair consists of `Iris2D1_train.txt` and `Iris2D1_test.txt`. The second pair consists of `Iris2D2_train.txt` and `Iris2D2_test.txt`. Each file contains one entry per row with single whitespaces as column delimiters. You can load the files using `numpy.loadtxt`.

Each entry contains 2 features and a class label. Thus, row n contains (x_{1n}, x_{2n}, y_n). Each dataset is binary, with classes labeled 0 and 1. Both `Iris2D1_train.txt` and `Iris2D2_train.txt` have 70 entries, 40 of class 0 and 30 of class 1. Both `Iris2D1_test.txt` and `Iris2D2_test.txt` have 30 entries, 10 of class 0 and 20 of class 1.

The logistic regression model with parameters $\mathbf{w} = (w_0, w_1, w_2)$ takes the form

$$(x_1, x_2) \mapsto \sigma(w_0 + x_1 w_1 + x_2 w_2) = \sigma(\mathbf{w}(1, x_1, x_2)^T),$$

where σ denotes the logistic (sigmoid) function $\sigma(x) = \frac{1}{1+e^{-x}}$ and w_0 denotes the intercept/bias term.

Deliverables. 1) 2 scatterplots and observations 2) training and test accuracy values (2 values for each of the 2 datasets), 3) the parameters of your logistic regression models (3 values for each of the 2 datasets), 4) the training and test accuracy values when using batch size $B = 4$, 5) discussion regarding training epochs

Exercise 3 (Backpropagation)**.**

In exercise 1, computing the derivative was fairly straightforward. However, for more complex functions such as neural networks, we want to decompose the derivative using the chain rule and compute the gradient step-by-step going backwards through the computational graph. To understand how this works, your task is to compute gradients for intermediate and leaf nodes in the computational graph of our familiar logistic regression model. The graph is shown in Figure 1.

1. Calculate and report the local gradients, indicated with a red question mark, based on the inputs and outputs/activations of each operation (values in blue) and the final gradient (which is equal to 1.00). For each unique operation, also write down the function and its derivative (which you use to compute the local gradients). E.g., for a "$+c$" operation, i.e. a function $f(x) = x + c$ with constant c, the derivative is $\frac{\partial f}{\partial x} = 1$.
2. What patterns do you see for backpropagation through addition and multiplication operations, i.e., is there a specific way gradients are routed through all addition operations (and likewise for multiplication operations)?

3. Assume we are running gradient descent on this computational graph. Report how the weights $\mathbf{w} = (w_0, w_1, w_2) = (1.00, 0.50, 0.50)$ would change after one update step of gradient descent with learning rate $\eta = 0.1$. Do we need the local gradients of our input features x_1 and x_2?

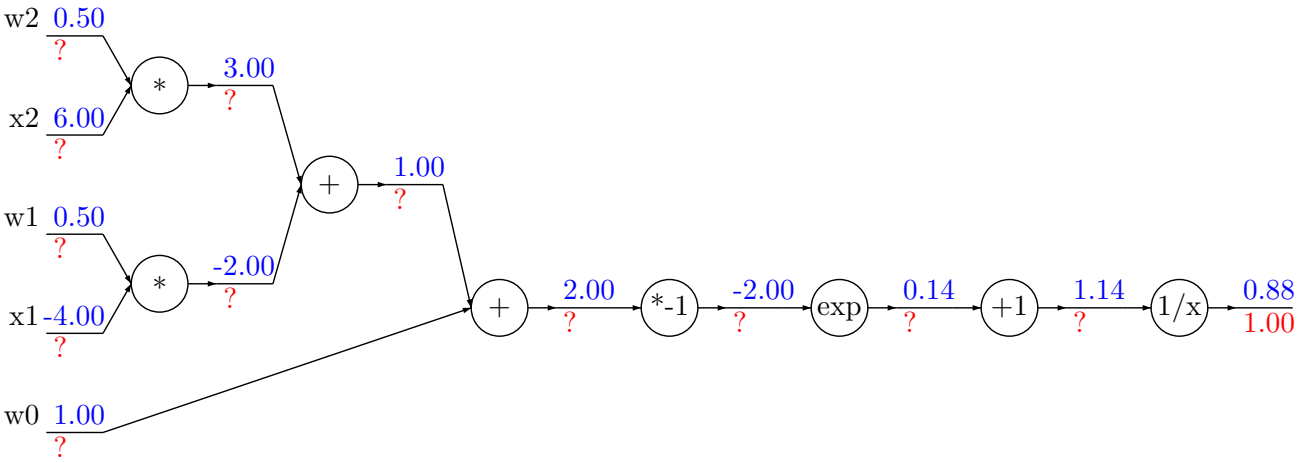

Figure 1: Computational graph for logistic regression model with $\mathbf{w} = (w_0, w_1, w_2) = (1.00, 0.50, 0.50)$ and input features $x_1 = -4.00$ and $x_2 = 6.00$. The sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is decomposed into four operations.

Deliverables. 1) Local gradients of the computational graph and gate functions and function derivatives used to compute them, 2) Discussion of patterns in backpropagation, 3) updated weights w and single-sentence answer whether local gradients are needed.

Note: When reporting the local gradients, you can re-use our computational graph for your report by replacing the question marks in the `computational_graph.svg` file (the file has a lot of numbers, but you can search using `cmd/ctrl + f`) and using the `svg` package for LaTeX. You can also draw your own graph or simply report the values in text form as long as it is clear which reported value corresponds to which local gradient in the graph.

Exercise 4 (Logistic regression loss-gradient)**.**

1. Show that the gradient of the in-sample function $E_{in}(\mathbf{w})$, as defined in the lecture, is given by

$$\begin{aligned}\nabla E_{in}(\mathbf{w}) &= -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n \mathbf{x}_n}{1+e^{y_n \mathbf{w}^\top \mathbf{x}_n}} \\ &= \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n \theta\left(-y_n \mathbf{w}^\top \mathbf{x}_n\right).\end{aligned}$$

2. Argue that a misclassified example contributes more to the gradient than a correctly classified one.

Deliverables. 1) Derivation of $E_{in}(\mathbf{w})$, 2) discussion about contribution of misclassified vs. correctly classified examples.

Computer Vision

Exercise 5 (Image Classification)**.** In this exercise, you will train, validate and test a ResNet (short for Residual Network) image classification model [1] on the custom FashionMNIST dataset described below. ResNet is based on the idea of residual learning, which allows for very deep neural networks to be trained successfully without encountering the vanishing gradient problem.

1. (Create a copy of this Google Colab notebook.)
2. Use the Colab notebook to train a `FashionMnistResNet18` on our custom FashionMNIST dataset with the hyper-parameters batch size $B = 256$, number of epochs $E = 100$, learning rate $\eta = 0.1$. Evaluate your model every 100 training steps.
3. Create a plot containing the training and validation loss curves; use the running mean loss values `mean_train_loss` and `mean_eval_loss` for this. Also create a second plot that shows the development of the training and validation accuracies throughout the training process. Tip: Make use of the `metrics_dict` returned by the training function.
4. Describe the plots created in step 2. Based on your observations, would you recommend changing the number of training epochs? If so, why?
5. Report the test accuracy of your model at the best validation loss and plot the confusion matrix on the test dataset. What are your takeaways from the confusion matrix?

Deliverables. 1) a plot containing the training and validation loss curves 2) a plot showing the development of the training and validation accuracies throughout the training process 3) a description of the plots 4) report test accuracy at best validation loss 5) plot and discuss the confusion matrix on the test dataset.

Exercise 6 (Data Augmentation)**.** In this exercise, you will explore simple data augmentation techniques to make better use of your limited training data.

1. Augment your training data by randomly flipping 33% of the images horizontally. Tip: Explore available image transformations in PyTorch at `https://pytorch.org/vision/main/transforms.html`. Re-run your experiments from exercise 5.2 with your augmented training dataset. Create a plot that compares the development of the validation accuracy of your model throughout training when using augmentation versus when not using augmentation. Also report the test accuracy of both models at the best respective validation loss. What do you see?
2. Repeat the experiment above, but this time flipping 50% of the images vertically. Based on your results and your knowledge of the FashionMNIST dataset, discuss the usefulness of this transformation.

Deliverables. 1) a plot showing the development of the validation accuracies throughout the training process with and without horizontal flipping 2) the test accuracy at best validation loss with and without horizontal flipping 3) a discussion of the results (horizontal flipping versus no augmentation) 4) a plot showing the development of the validation accuracies throughout the training process with and without vertical flipping 5) the test accuracy with and without vertical flipping 6) a discussion on the usefulness of vertical flipping for FashionMNIST.

Natural Language Processing

Exercise 7 (Text Classification)**.** In this exercise, you will implement, train and test a binary text classifier for the Tweets Sentiment Analysis dataset described below.

1. (Create a copy of this Google Colab notebook.)
2. Tokenize the training set with a simple whitespace tokenizer and plot a histogram of the 30 most common tokens.
3. Implement a continuous-bag-of-words (CBoW) classifier with randomly initialized 100-dimensional word embeddings and one hidden layer with ReLU activation. Train it on the training set using the AdamW optimizer with batch size $B = 256$, number of epochs $E = 30$, learning rate $\eta = 0.01$, and evaluation on the validation set every 50 steps. Report the test accuracy of your model at the best validation loss.
4. Create a plot containing training and validation loss curves, and create a second plot of the development of the training and validation accuracies throughout the training process. Discuss the results in the context of overfitting.
5. Visualize the learned embeddings for 2000 most common tokens using PCA in a 2-dimensional scatter plot. Assign the corresponding tokens as labels to the 100 points that are farthest away from the center of all the points, alongside their respective coordinates. What do you observe?
6. Replace the CBoW classifier with a pre-trained `DistilBERT` checkpoint. Fine-tune the new classifier for $E = 3$ epochs with learning rate $\eta = 5e{-}5$, batch size $B = 32$, and evaluation on the validation set every 250 steps. Evaluate the model at the best validation loss on the test set. Report 2 plots and the test accuracy as in steps 2 and 3. Compare the results with CBoW; what is the difference?

Deliverables. 1) a histogram plot of the 30 most common tokens 2) a plot containing the training and validation loss curves and the test accuracy of your model 3) a plot showing the development of the training and validation accuracies throughout the training process 4) a short discussion of the plots 5) the test accuracy of the CBoW classifier 4) a 2-dimensional scatter plot of the learned embeddings using PCA, and your observation 5) 2 plots, the test accuracy of your model and a short comment on the results.

Appendix: Data material

The Iris dataset

The Iris dataset is a classical one, used by R. Fisher. From Wikipedia: "*The dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.*"

For the assignment, `Iris2D2_train.txt` and `Iris2D2_test.txt` were made of the 100 first entries, keeping only features 0 and 1 ('sepal length', 'petal length'). `Iris2D1_train.txt` and `Iris2D1_test.txt` were made of the 100 last entries, keeping only features 0 and 2 ('sepal length', 'sepal width'). Classes 1 and 2 for these last entries were remapped to 0 and 1. Each of the 100 entries where randomly permuted and divided into 70 training and 30 testing entries.

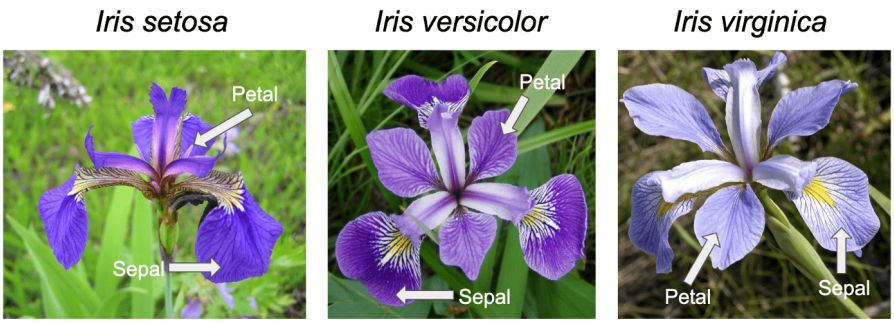

Figure 2: Iris dataset: the three species of Iris

FashionMNIST

FashionMNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The labels are:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

For the assignment, we provide three custom splits: `fashion_mnist_training.pt` (5000 examples), `fashion_mnist_validation.pt` (1000 examples), and `fashion_mnist_test.pt` (1000 examples). They can be loaded via our custom PyTorch dataset found in the corresponding notebook.

Figure 3: Fashion MNIST dataset

Twitter Sentiment Analysis

The Twitter Sentiment Analysis dataset [2] has three sentiment labels, namely negative (-1), neutral (0), and positive (+1). It contains two columns: the tweet text and the label.

In this assignment, we use a subset involving two sentiment labels, namely negative (0) and positive (+1). Each row in the files is a pair of (tweet text, label).

We provide three custom splits:

- `mini_twitter_binary_training.csv` (17241 examples),
- `mini_twitter_binary_validation.csv` (2155 examples),
- `mini_twitter_binary_test.csv` (2155 examples).

They can be loaded using `pandas.read_csv` and our custom PyTorch dataset found in the corresponding Colab notebook.

clean_text	**category**
proud our modi doing really good development for our nation and peoples support him 😊	1
yes modi drama was stupid tho	0
mission shakti rahul gandhi says well done drdo wishes modi happy world theatre day ⚡vikram sarabhai	1
the modi government' jammu kashmir policy has been disaster further alienating the people the valley	0

Table 1: Example tweets from the preprocessed twitter sentiment dataset

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[2] S. Hussein. Twitter sentiments dataset, 2021.