Project tracks

Global remarks (for both projects)

Your project should combine:

- clear mathematical formulations,
- discussion on algorithmic implementations,
- numerical experiments with insightful visualizations,
- and critical comments about the adopted approach from an optimization point of view, keeping in mind convergence and efficiency.

Students are encouraged to explore variations, test hyperparameters, and include insightful plots and interpretations.

This should be summarized and presented concisely during the oral presentation and in the written report. You should also submit the code that you wrote for this project. General guidelines for your outputs are given on the website of the course.

In the following, we provide two project tracks, choose only one of them:

- I- Hand-Written Digits Classification with Stochastic Gradient Descent
- II - The Hanging Chain Problem with the Conjugate Gradient Method

For both these projects, the list of tasks is here to help you conduct your project, **but your presentation and report should not use the same structure.** You should rather use your own creativity for presenting the work you did, and choose yourself the most interesting aspects to highlight.

Remember that your project should include both theoretical understanding, numerical experiments, and interpretations of the results based on your own understanding of the problem.

Also, note that you should clearly understand everything you submit for this project. It is OK to use LLMs to help with grammar corrections, or to fix some bugs, but you should make sure that you thought about the problem yourself, and every answer you provide should be your own work.

During the oral presentation, we will ask you questions about the content you provided. If you are not able to explain what you did, you might get penalized more than if you had not provided this content at all.

I Hand-Written Digits Classification with Stochastic Gradient Descent

Problem description

In this project, you will study a classical machine learning problem: classification of hand-written digits using logistic regression, solved with stochastic gradient descent (SGD).

Dataset: We consider the NIST dataset available here, which can be downloaded directly from Python using the `sklearn.datasets.load_digits` (see here for more details). This consists of 1797 grayscale images of hand-written digits (0–9). Each image has size 8×8 pixels, and is represented as a vector $a_j \in \mathbb{R}^{64}$. Each image comes with a label $y_j \in \{0, 1, 2, \dots, 9\}$.

Logistic regression model: You will use multi-class logistic regression to classify the images. This is explained in the course; but we briefly recall the main points here for completeness.

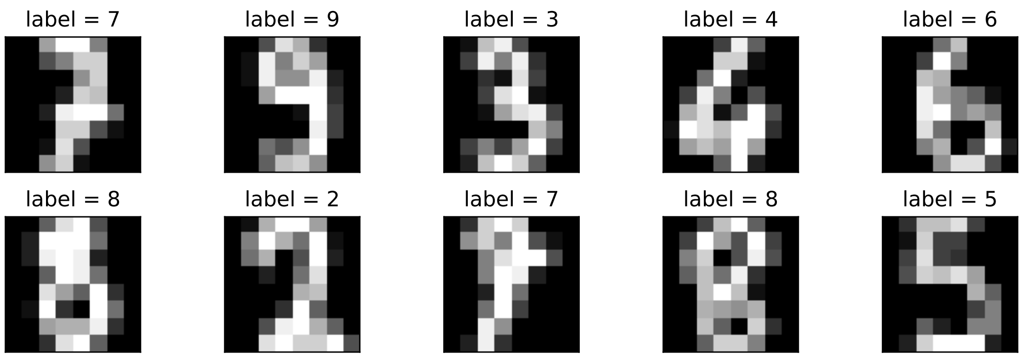

Figure 1: Examples of hand-written digits with associated labels from the NIST dataset available here.

In classification, instead of using the labels directly, we represent them using one-hot vectors

$$p_j \in \mathbb{R}^{10}, \qquad p_{j,l} = \begin{cases} 1 & \text{if } y_j = l, \\ 0 & \text{otherwise.} \end{cases}$$

The predictor is defined by $\varphi(a_j; x)_i = b_i + a_j^\top w_i$, where the parameter x is defined as: $x = (w_0, b_0, \ldots, w_9, b_9)$; and where w_i and b_i are the parameters associated with class i. The predicted class is given by $\hat{y}_j = \arg\max_{i \in 0,\ldots,9} \varphi(a_j; x)_i$.

To find the best parameter x, we train the model on a training dataset with the following optimization problem:

$$\underset{x}{\text{minimize}} \quad \frac{1}{m} \sum_{j=1}^{m} L(p_j, a_j, x)$$

where the training data is $\{(a_j, p_j)\}_{j=1}^m$, and where L is the cross-entropy loss:

$$L(p_j, a_j, x) = \log\left(\sum_{l=1}^{q} e^{\varphi(a_j;x)_l} \right) - \sum_{i=1}^{q} p_{j,i} \varphi(a_j; x)_i.$$

Tasks

1. Logistic regression background:
   - Explain the principle of logistic regression for multi-class classification: why does it make sense to use this loss?
   - Is the problem convex? Does it always have a solution? Is it a smooth problem?
   - Why didn't we use the predictions given by $\arg\max_i \varphi(a_j; x)_i$ directly in the loss, instead of using the cross-entropy loss?
2. Stochastic Gradient Descent:
   - What is the difference between full-batch gradient descent and stochastic gradient descent, and standard gradient descent (sometimes referred to as full-batch gradient descent)?
   - What is the motivation behind stochastic gradient descent for large datasets?
   - Explain the role of mini-batches, and what is the trade-off to consider when choosing the batch size?
3. Implementation:
   - Implement your own SGD algorithm for this problem. For now, start with a constant step-size to make it easier. You may compute gradients using either handwritten formulas or automatic differentiation libraries such as CasADi or PyTorch.
   - You might also consider L_2 regularization in the loss.

4. Evaluation of the approach:
    - Measure classification accuracy on training and test sets and comment the results. How does $L-2$ regularization affect the results?
    - Visualize predictions on test images to confirm that the model behaves correctly.
5. Comparison with the optimal solution:
    - Compute a reference solution using full batch gradient descent, or IPOPT via CasADi.
    - Use it to visualise the convergence of your SGD approach in terms of loss decrease and/or in terms of distance to the optimal value.
6. Influence of a hyperparameter – the step size:
    - Try different step-size strategies, and compare the resulting convergence, as a function of the number of iterations. For example: you might consider constant step-sizes with different values, or decreasing step-sizes proportional to $1/t$ or $1/\sqrt{t}$ or similar, or adaptive strategies like in ADAM.
    - Optionally: you can also discuss the L-smoothness of the problem to motivate the choice of the step-size
7. Influence of a hyperparameter –the batch-size
    - Study the influence of the batch-size on the speed of convergence, and on the computational cost (by looking at the runtime for example).
    - What is the best trade-off between the two?
8. Conclusions regarding the use of SGD for this problem:
    - Based on the discussion above, do you think that SGD is a good choice for this problem compared to full batch gradient descent? Why?
    - What are the advantages and disadvantages of SGD in this context?
    - What do you expect to happen if the dataset becomes larger?

II The Hanging Chain Problem with the Conjugate Gradient Method

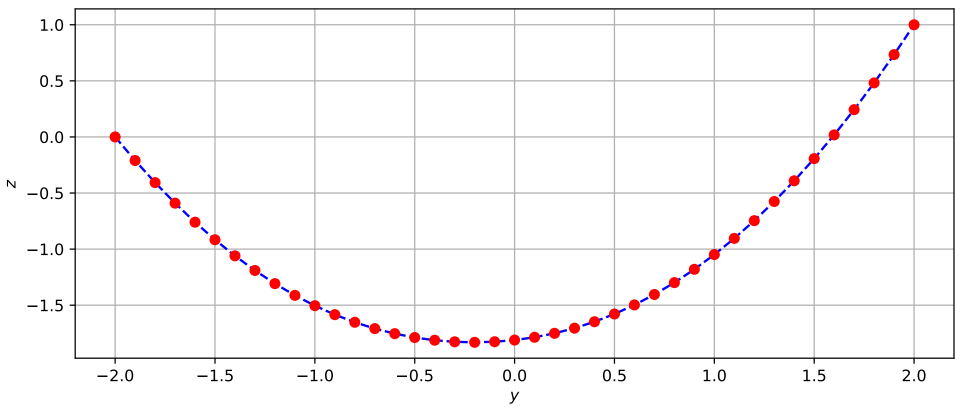

Figure 2: Hanging chain solution for $(y_0, z_0) = (-2, 0)$ and $(y_N, z_N) = (2, 1)$.

Problem description

We want to model a chain of springs attached to two supports and hanging freely in between. The chain consists of $N+1$ masses connected by N massless springs. Each mass i has position (y_i, z_i), $i = 0, \dots, N$. The masses at the two ends of the chain (i.e., masses 0 and N) are fixed to the supports at some positions (y_0, z_0, y_N, z_N), while the other masses are free to move. We would like to find the equilibrium position which minimizes the potential energy of the full system. The potential energy associated with each spring is given by $\frac{1}{2}D\left((y_i - y_{i+1})^2 + (z_i - z_{i+1})^2\right)$, for $i = 0, \dots, N-1$, and with spring constant $D \in \mathbb{R}^+$. The gravitational potential energy of each mass is $m\,g\,z_i$, for $i = 0, \dots, N$, where $g = 9.81$ is the earth gravity constant, and m is the mass. The total potential energy is thus given by

$$V_{\text{chain}}(\boldsymbol{p}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{i=0}^{N-1} \frac{1}{2} D\left((y_i - y_{i+1})^2 + (z_i - z_{i+1})^2\right) + mgz_i,$$

where $\boldsymbol{y} = (y_1, \dots, y_{N-1})$, $\boldsymbol{z} = (z_1, \dots, z_{N-1})$, and the parameters are the positions of the left and right ends of the chain:$\boldsymbol{p} = (y_0, z_0, y_N, z_N)$.

The equilibrium chain position is therefore the solution of the following energy-minimization problem:

$$\underset{\boldsymbol{y},\boldsymbol{z} \in \mathbb{R}^N}{\text{minimize}} \quad V_{\text{chain}}(\boldsymbol{p}, \boldsymbol{y}, \boldsymbol{z}) \tag{1}$$

The goal of this project is to implement an efficient numerical method to solve this problem, and analyse its performance for a large number of masses N.

Tasks

1. Problem analysis:
   - Identify the type of optimization problem. Is it convex?
   - Does it have a unique solution?
   - Put this problem into its standard form by defining the relevant matrices and vectors.
2. Analytical solution:
   - Explain how the solution can be computed explicitly.

- Discuss the computational cost of a naive approach, as a function of N.
- Support your discussion with Python experiments.

3. Visualization:
   - Compute the solution for different endpoint parameters p, using either CasADi + IPOPT, or with the explicit solution you derived.
   - Plot the corresponding equilibrium for the chain.
   - Interpret the physical meaning of the solutions.
4. Gradient descent approach:
   - Derive an efficient expression for the gradient of V_{chain}, and present a gradient descent algorithm to solve this problem.
   - Discuss the computational cost of one iteration as a function of N; and discuss the expected convergence rate of this approach.
   - Implement this gradient descent method, using the efficient gradient expression.
   - Show the convergence of the method and then analyze the computational cost (with the runtime for example). Interpret the results.
5. Conjugate Gradient method:
   - Explain why conjugate gradient (CG) is better suited here.
   - Write the CG iteration equations.
   - Show how matrix–vector products Qp can be computed efficiently without forming the full matrix, similar to what you did for the gradient computation.
   - Implement this algorithm using the efficient matrix–vector product.
   - Compare the convergence with gradient descent.
6. Complexity comparison:
   - Compare the overall computational cost, and the accuracies of the three approaches discussed (naive direct solving, gradient descent, conjugate gradient).
   - What is the best approach for large values of N? Support your discussion with both the theory and a numerical experiments.
7. OPTIONAL: the hanging grid –a 3D extension
   - Extend the problem to a 3D model, where the hanging chain is replaced by a 2D grid of masses ($N \times N$), connected by springs; with some choices for the the fixed boundary conditions.
   - Visualize the solutions for this problem.
   - Discuss again the computational costs of the naive approach and the CG approach as a function of N.