UiT

THE ARCTIC UNIVERSITY OF NORWAY

1. Vectorising Code and Machine Learning Basics

FYS-2021 Exercises
Department of Physics and Technology
Faculty of Science and Technology

Vectorising code

Problem 1

We ask the students to use Python as the programming language for this course. This is the standard language used for Machine Learning.

(1a) For students who do not have much experience with Python and the numerical library NumPy (or want to refresh their knowledge), it is recommended to do a few of the "warm-up" exercises that you can find here:

https://www.w3resource.com/python-exercises/numpy/basic/index.php

https://www.w3resource.com/python-exercises/numpy/index-array.php

https://www.w3resource.com/python-exercises/numpy/linear-algebra/index.php

https://www.w3resource.com/python-exercises/numpy/python-numpy-stat.php

This course assumes you have a fair knowledge of how numerical programming, (i.e. working with vectors and matrices programatically), works. In the following exercises, we will look at writing efficient, vectorised code. Assume that we have vectors

$$\boldsymbol{a} \in \mathbb{R}^n \text{ with elements } \boldsymbol{a} = [a_1, \ldots, a_n]^T$$

$$\boldsymbol{b} \in \mathbb{R}^n \text{ with elements } \boldsymbol{b} = [b_1, \ldots, b_n]^{\mathrm{T}}$$

Further assume we have a matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1N} \\ & & \vdots & \\ x_{M1} & x_{M2} & \ldots & x_{MN} \end{bmatrix}$$

The result vector is $\boldsymbol{y}$ which takes its dimensionality based on the problem.

In all problems you should assume that the dimensions of the vectors and matrices involved are such that the matrix multiplications (and inner products) are valid. Note that e.g. `y+=k` is the same as `y=y+k`.

(1b) Write this sum as a mathematical vector operation

$$y = \sum_{i=1}^{n} a_i b_i$$

(1c) Write this for loop as a mathematical vector operation and as vectorised code.

```
for i=1,...,n:
  y+=a_i b_i
```

(1d) Write this for loop as a mathematical vector operation and vectorised code (assume $N = n$)

```
for i=1,...,M:
  for k=1,...,n:
    y_i+=x_ik a_k
```

(1e) *Vectorise* the following algorithm (answer with code). This shouldn't be done with traditional matrix multiplications *alone*:

```
for i=1,...,M:
  for k=1,...,n:
    y+=x_ik a_k
```

(1f) Let $Z \in \mathbb{R}^{P\times M}$ and $z_{ij} \in Z$ be the entry in row i and column j.

Vectorise the following algorithm

```
for j=1,...,N:
  for i=1,...,P:
    for k=1,...,M:
      y_ij+=z_ik x_kj
```

(1g) NB:*This part problem has different notation!*
Suppose you have a dataset with N samples and n features:

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{x}_i = [x_{i1}, \ldots, x_{in}] \in \mathbb{R}^n$$

Similarly, you have parameter vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_P$, $\boldsymbol{w} \in \mathbb{R}^n$ and $\boldsymbol{a} \in \mathbb{R}^P$.

Implement an algorithm, *for every single sample* $\boldsymbol{x}_j = [x_{j1}, \ldots, x_{jn}]$ in the dataset, that calculates:

$$v_i = \sum_{k=1}^{n} w_{ik} x_{jk}$$

$$y_j = \sum_{i=1}^{P} v_i a_i$$

Vectorise these expressions. You want to end up with a vector $\boldsymbol{y} \in \mathbb{R}^N$

Because of the potential lack of relevance, the next problems are considered 'bonus' exercises.

(1h) Implement an algorithm that does matrix multiplication (without using built-in matrix multiplication functions).

(1i) Show that for any matrix $\mathbf{A}$ the product $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is symmetric.

(1j) Implement a algorithm that calculates $\mathbf{A}^T\mathbf{A}$ using the property of symmetry. Generate a huge matrix $\mathbf{A}$ and compare the running time between the implementation you did in (1h) and the built in matrix multiplication function.

(1k) Why vectorised (or matrix-based) computations hold an advantage over traditional 'for' loops in Python?

Github basics

Problem 2

In this task we'll use the basics of Git to clone an existing repo from github and push some of it onto our own repository on github. This tasks assumes Git has been installet. This can be done here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. This task might not present completely accurate information depending on how you've connected your computer to github.
For more information on how to setup github with ssh, have a look at the official documentation: https://docs.github.com/en/authentication/connecting-to-github-with-ssh

(2a) First make a folder called *GitTutorial* and navigate to this folder via a terminal. Then, while inside this folder, use the command **git init** to initialize a local repository in that folder. Verify that a repository has been made by passing **git status**

(2b) Next we'll clone an existing repository from github to your local repository. The repository we'll clone is this: https://github.com/Seilmast/FYS-2021 In the terminal, while still in the **GitTutorial** folder, use the **git clone** and find the link on the github page which references the online repository. This link is found by clicking the green "code" button, and copy the HTTPS link. This should now clone two files into your local repository.

(2c) Now you should have two files locally on your computer. These are called *UploadThis.txt* and *DoNotUploadThis.txt*. We'll first upload both these files to you remote repository. Go through each step in order to do so

1. Go to your github account and make a new repository. When you've made one there should be a link to set up the remote connection. If not, find the SSH link under the big green "code" button.

2. Now, in your terminal, use the **git remote <name> <link>** command to link your remote and local repository. You can define the name yourself. It'll be used to refer to the remote connection. The link should be the SSH link.

3. Next push *UploadThis.txt* to your repository. First use **git add <file>** to prepare the files to be pushed. You may use **git add .** to add all files in the folder, but for now just add the one file manually. Next use **git commit -m <commit message>** to commit all the added files to be pushed. The message can be any string, but good form is to give a brief explanation of the changes made with the given commit. Lastly use **git push <name>** where name is the remote name you defined earlier to push the commit to github. Both files should now be on your repository.

(2d) Often you want to exclude some files from ever being commited, but don't want to do **git add <file>** for every file. Make a file called *.gitignore* in your local repository. Now add the *DoNotUploadThis.txt* to this ignore file, do step 3 from the last task again. This time you can use **git add .**, as git will automatically ignore the files in the *.gitignore* file. This should leave your online repository with only a single textfile.

More on mathematical and Python background

Problem 3

(3a) Follow the notebook: https://github.com/udlbook/udlbook/blob/main/Notebooks/Chap01/1_1_BackgroundMathematics.ipynb. This is a complementary notebook for working with matrices and it prepares you for the next step when we see linear regression.

Problem 4

(4a) Follow the notebook: https://github.com/uitml/MLCourse2023/blob/main/code/NotebookTutorial.ipynb and try to understand each step. This is a simple notebook to refresh your Python skills about arrays and plotting.