Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: September 03, 2022
Total: 60 marks

ASSIGNMENT 1

1 What should I submit, where should I submit and by when?

Your submission for this assignment will be one PDF (.pdf) file and one ZIP (.zip) file. Instructions on how to prepare and submit these files is given below.

Assignment Package:
`https://web.cse.iitk.ac.in/users/purushot/courses/ml/2022-23-a/material/assn1.zip`
Deadline for all submissions: 17 September, 9:59PM IST
Code Submission: `https://forms.gle/ySJ4rmbnEJoVoXFY9`
Report Submission: on Gradescope
There is no provision for "late submission" for this assignment

1.1 How to submit the PDF report file

1. The PDF file must be submitted using Gradescope in the *group submission mode*. Note that this means that auditors may not make submissions to this assignment.

2. Make only one submission per assignment group on Gradescope, not one submission per student. Gradescope allows you to submit in groups - please use this feature to make a group submission.

3. Link all group members in your group submission. If you miss out on a group member while submitting, that member may end up getting a zero since Gradescope will think that person never submitted anything.

4. You may overwrite your group's submission (submitting again on Gradescope simply overwrites the old submission) as many times as you want before the deadline.

5. Do not submit Microsoft Word or text files. Prepare your report in PDF using the style file we have provided (instructions on formatting given later).

1.2 How to submit the code ZIP file

1. Your ZIP file should contain a single Python (.py) file and nothing else. The reason we are asking you to ZIP that single Python file is so that you can password protect the ZIP file. Doing this safeguards you since even after you upload your ZIP file to your website, no one can download that ZIP file and see your solution (you will tell the password only to the instructor). If you upload a naked Python file to your website, someone else may guess the location where you have uploaded your file and steal it and you may get charged with plagiarism later.

2. We do not care what you name your ZIP file but the (single) Python file sitting inside the ZIP file must be named "submit.py". There should be no sub-directories inside the ZIP file – just a single file. We will look for a single Python (.py) file called "submit.py" inside the ZIP file and delete everything else present inside the ZIP file.

3. Do not submit Jupyter notebooks or files in other languages such as C/Matlab/Java. We will use automated scripts to evaluate your code and will not run code in other formats or other languages (we will simply give such submissions a zero).

4. Password protect your ZIP file using a password with 8-10 characters. Use only alphanumeric characters (a-z A-Z 0-9) in your password. Do not use special characters, punctuation marks, whitespaces etc in your password. Specify the file name properly in the Google form.

5. Remember, your file is not under attack from hackers with access to supercomputers. This is just an added security measure so that even if someone guesses your submission URL, they cannot see your code immediately. A length 10 alphanumeric password (that does not use dictionary phrases and is generated randomly e.g. 2x4kPh02V9) provides you with more than 55 bits of security. It would take more than 1 million years to go through all $> 2^{55}$ combinations at 1K combinations per second.

6. Make sure that the ZIP file does indeed unzip when used with that password (try `unzip -P your-password file.zip` on Linux platforms). If we are unable to unzip your file, you will lose marks.

7. Upload the password protected ZIP file to your IITK (CC or CSE) website (for CC, log on to `webhome.cc.iitk.ac.in`, for CSE, log on to `turing.cse.iitk.ac.in`).

8. Fill in the following Google form to tell us the exact path to the file as well as the password `https://forms.gle/ySJ4rmbnEJoVoXFY9`

9. Do not host your ZIP submission file on file-sharing services like Dropbox or Google drive. Host it on IITK servers only. We will be using a script to autodownload your submissions and if not careful, Dropbox or Google Drive URLs may send us an HTML page (instead of your submission) when we try to autodownload your file. Thus, it is best to host your code submission file locally within IITK servers.

10. While filling in the form, you have to provide us with the password to your ZIP file in a designated area. Write just the password in that area. For example, do not write "Password: helloworld" in that area if your password is "helloworld". Instead, simply write "helloworld" (without the quotes) in that area. Remember that your password should contain only alphabets and numerals, no spaces, special or punctuation characters.

11. While filling the form, give the complete URL to the file, not just to the directory that contains that file. The URL should contain the filename as well.

    (a) Example of a proper URL:
    `https://web.cse.iitk.ac.in/users/purushot/mlassn1/submit.zip`

    (b) Example of an improper URL (file name missing):
    `https://web.cse.iitk.ac.in/users/purushot/mlassn1/`

    (c) Example of an improper URL (incomplete path):
    `https://web.cse.iitk.ac.in/users/purushot/`

12. We will use an automated script to download all your files. If your URL is malformed or incomplete, or if you have hosted the file outside IITK and it is difficult for us to download automatically, then your group will either lose a lot of marks or else get a straight zero in these cases.

13. Make sure you fill in the Google form with your file link before the deadline. We will close the form at the deadline.

14. Make sure that your ZIP file is actually available at that link at the time of the deadline. We will run a script to automatically download these files after the deadline is over. If your file is missing, we will assign your group zero marks.

15. We will entertain no submissions over email, Piazza etc. All submissions must take place before the stipulated deadline over the Gradescope and the Google form. The PDF file must be submitted on Gradescope at or before the deadline and the ZIP file must be available at the link specified on the Google form at or before the deadline.

Problem 1.1 (Odd-even rules don't seem to add anything new)**.** Melbo was a bit perturbed that the arbiter PUF model for verification was broken so easily using a simple linear model. In an effort to make an unbreakable PUF, Melbo came up with a simple but promising idea. Recall that a PUF is a chain of say k multiplexers, each of which either swaps the lines or keeps them intact, depending on what is the challenge bit fed into that multiplexer. The multiplexers each have delays which are hard to replicate but consistent. If the top signal reaches the finish line first (whether it is blue or red does not matter), the response is 0 else if the bottom signal reaches first (again, color does not matter), the response is 1.

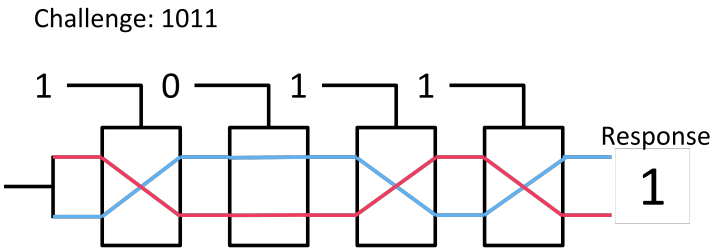

Figure 1: A simple arbiter PUF with 4 multiplexers

Melbo's puffier PUF takes 3 PUFs and feed the same challenge into all of them. However, since the 3 PUFs have 3 different set of multiplexers with different delays, the 3 PUFs need not give the same response even if they are given the same challenge. If an odd number of PUFs (i.e. any one PUF or all three PUFs) have the response 1, Melbo's puffier PUF would give a response 1 and if an even number of PUFs (i.e. no PUF or any two PUFs) have the response 1, Melbo's puffier PUF would give a response 0. This is often called the XOR operation (which stands for eXclusive OR) and Melbo's puffier PUF is also known as a XOR-PUF.

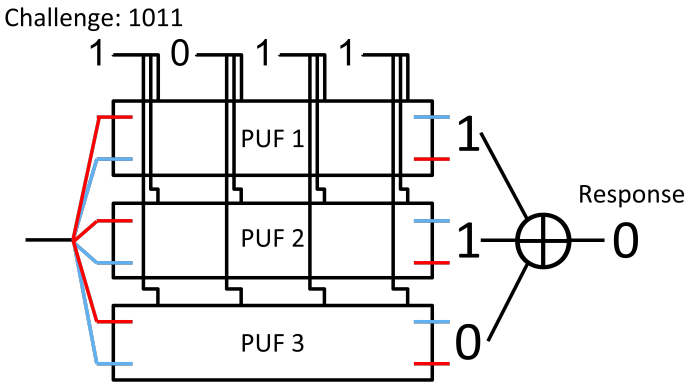

Figure 2: An XOR PUF with 3 PUFs, each with 4 multiplexers

Melbo thinks that with multiple PUFs and such a complicated odd-even rule in place, there is no way any machine learning model, let alone a linear one, can predict the responses if given a few thousand challenge-response pairs. Your job is to prove Melbo wrong! You will do this by showing that even in this case, there does exist a linear model that can perfectly predict the responses of a XOR-PUF and this linear model can be learnt if given enough challenge-response pairs (CRPs).

Your Data. We have provided you with data from a XOR-PUF which has 3 PUFs but each PUF has 8 multiplexers i.e. the challenges will be 8-bits long. The same challenge is fed to all

3 PUFs. The training set consists of 10000 CRPs and the test set consists of 20000 CRPs. If you wish, you may create (held out/k-fold) validation sets out of this data in any way you like. Your job will be to create a new feature vector from the 8-bit challenge so that a linear model on that feature vector is able to predict the response. However, a twist in the tale is that unlike in the Arbiter PUF case where the new feature vector was also 8-dimensional if the challenges were 8-dimensional, here the new feature vector may need to have a different dimensionality than the challenges. Recall that we saw that if we properly encode the challenge as a 8-dim vector, say $\mathbf{x} \in \mathbb{R}^8$, then for any PUF, there exists a linear model parameterized as $(\mathbf{w} \in \mathbb{R}^8, b)$ so that the response is always

$$\frac{1 + \text{sign}(\mathbf{w}^\top \mathbf{x} + b)}{2}$$

Your Task. The following enumerates 6 parts to the question. Parts 1,2,3,5,6 need to be answered in the PDF file containing your report. Part 4 needs to be answered in the Python file.

1. By giving a mathematical derivation, show the exists a way to map the binary digits $0, 1$ to signs $-1, +1$ as say, $m : \{0, 1\} \to \{-1, +1\}$ and another way $f : \{-1, +1\} \to \{0, 1\}$ to map signs to bits (not that m and f need not be inverses of each other) so that for any set of binary digits $b_1, b_2, \ldots, b_n$ for any $n \in \mathbb{N}$, we have

   $$\text{XOR}(b_1, \ldots, b_n) = f\left(\prod_{i=1}^{n} m(b_i)\right).$$

   Thus, the XOR function is not that scary – it is essentially a product. (5 marks)

2. Let $(\mathbf{u}, a), (\mathbf{v}, b), (\mathbf{w}, c)$ be the three linear models that can exactly predict the outputs of the three individual PUFs sitting inside the XOR-PUF. For sake of simplicity, let us hide the bias term inside the model vector by adding a unit dimension to the original feature vector so that we have $\tilde{\mathbf{u}} = [\mathbf{u}, a], \tilde{\mathbf{v}} = [\mathbf{v}, b], \tilde{\mathbf{w}} = [\mathbf{w}, c], \tilde{\mathbf{x}} = [\mathbf{x}, 1] \in \mathbb{R}^9$. The above calculation shows that the response of the XOR-PUF can be easily obtained (by applying f) if we are able to get hold of the following quantity:

   $$\text{sign}(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot \text{sign}(\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot \text{sign}(\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}})$$

   To exploit the above result, first give a mathematical proof that for any real numbers (that could be positive, negative, zero) $r_1, r_2, \ldots, r_n$ for any $n \in \mathbb{N}$, we always have

   $$\prod_{i=1}^{n} \text{sign}(r_i) = \text{sign}\left(\prod_{i=1}^{n} r_i\right)$$

   Assume that $\text{sign}(0) = 0$. Make sure you address all edge cases in your calculations e.g. if one or more of the numbers is 0. (5 marks)

3. The above calculation tells us that all we need to get hold of is the following quantity

   $$(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}})$$

   Now show that the above can be expressed as a linear model but possibly in a different dimensional space. Show that there exists a dimensionality D such that D depends only on the number of PUFs (in this case 3) and the dimensionality of $\tilde{\mathbf{x}}$ (in this case $8 + 1 = 9$) and there exists a way to map 9 dimensional vectors to D dimensional vectors as

$\phi : \mathbb{R}^9 \to \mathbb{R}^D$ such that for any triple $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}})$, there always exists a vector $\mathbf{W} \in \mathbb{R}^D$ such that for every $\tilde{\mathbf{x}} \in \mathbb{R}^9$, we have $(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \mathbf{W}^\top \phi(\tilde{\mathbf{x}})$. (10 marks)

Hint: First try solving this for the simpler case where there are only 2 PUFs. If we expand the terms of $(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}})(\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) = \left(\sum_{j=1}^9 \tilde{u}_j \tilde{x}_j\right) \left(\sum_{j=1}^9 \tilde{v}_j \tilde{x}_j\right)$, we get an expression of the form $\sum_{j=1}^9 \sum_{k=1}^9 \tilde{u}_j \tilde{v}_k \tilde{x}_j \tilde{x}_k$. Thus, if we create a $9^2 = 81$-dimensional function that maps

$$\tilde{\mathbf{x}} = (\tilde{x}_1, \ldots, \tilde{x}_9) \text{ to } \phi(\tilde{\mathbf{x}}) = (\tilde{x}_1\tilde{x}_1, \tilde{x}_1\tilde{x}_2, \ldots \tilde{x}_1\tilde{x}_9, \tilde{x}_2\tilde{x}_1, \ldots, \tilde{x}_9\tilde{x}_9),$$

then we are done since we can now get $(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) = \mathbf{W}^\top \phi(\tilde{\mathbf{x}})$ by taking

$$\mathbf{W} = (\tilde{u}_1\tilde{v}_1, \tilde{u}_1\tilde{v}_2, \ldots \tilde{u}_1\tilde{v}_9, \tilde{u}_2\tilde{v}_1, \ldots, \tilde{u}_9\tilde{v}_9)$$

Closely understand the trick in this simpler case and then extend it to the case of 3 PUFs to solve this part of the problem. Give detailed calculations for your solution.

4. Write code to solve this problem by learning the linear model $\mathbf{W}$. Given a test challenge $\tilde{\mathbf{x}}_t$, you will first map it to D dimensions, then apply the linear model $\mathbf{W}^\top \phi(\tilde{\mathbf{x}}_t)$ and then do simple operations to obtain the XOR-PUF response. You may use any formulation e.g. SVM with hinge loss, squared hinge loss, logistic regression etc., to learn the linear model. You may use any solver e.g. primal gradient/subgradient descent, SGD, MBSGD, SDCA etc to solve the problem so long as you implement the solver yourself i.e. you are not allowed to simply call e.g., an sklearn function, to solve the problem. Submit code for your chosen method in `submit.py`. Note that your code will need to implement at least 3 methods namely

   (a) `get_renamed_labels()` that should set the convention on how to map 0/1 labels as given in the dataset to $-1/+1$ as required by binary classification solver like SVM.

   (b) `get_features()` that should implement the $\phi()$ function to map the raw data features to the final features over which the linear model would be applied. Note that even in the arbiter function example, we did have such a map that used flip and cumulative product operations (see the DH code notebooks).

   (c) `solver()` that implements the actual solver.

   We will evaluate your method on a different dataset than the one we have given you and check how good is the method you submitted (see below for details). Note that you will have to implement the map ϕ as well. For this, you may find the Khatri-Rao product to be very useful (`https://en.wikipedia.org/wiki/Khatri%E2%80%93Rao_product#Column-wise_Kronecker_product`). The KR-product is implemented in the Scipy package which you can freely use without penalty (however, use of any other Scipy routine is not permitted and any such use will incur penalties). Although the KR-product is define in a column-wise manner, you may find the row-wise version more useful which can be managed by transposing the matrix before applying the KR-product. (30 marks)

5. For the method you implemented, describe in your PDF report what were the hyperparameters e.g. step length, policy on choosing the next coordinate if doing SDCA, mini-batch size if doing MBSGD etc and how did you arrive at the best values for the hyperparameters, e.g. you might say "*We used step length at time t to be $\eta/\sqrt{t}$ where we checked for $\eta = 0.1, 0.2, 0.5, 1, 2, 5$ using held out validation and found $\eta = 2$ to work the best*". For another example, you might say, "*We tried random and cyclic coordinate selection choices and found cyclic to work best using 5-fold cross validation*". Thus, you must tell us among which hyperparameter choices did you search for the best and how. (5 marks)

6. Plot the convergence curves in your PDF report offered by your chosen method as we do in lecture notebooks. The x axis in the graph should be time taken and the y axis should be the test classification accuracy (i.e. higher is better). Include this graph in your PDF file submission as an image. (5 marks)

Parts 1,2,3,5,6 need to be answered in the PDF file containing your report. Part 4 needs to be answered in the Python file.

Evaluation Measures and Marking Scheme. We created two XOR-PUFs – a public one and a secret one. Both had 8-bit challenges and used 3 PUFs each but the PUFs all have different delays so a model that is able to predict the public XOR-PUF delays is expected to do very poorly at predicting the secret XOR-PUF delays and vice versa. The train/test sets we have provided you with the assignment package were created using CRPs from the public XOR-PUF. We similarly created a secret train and secret test set using CRPs from the secret XOR-PUF. We will use your code to train on our secret train set and use the learnt model to make predictions on our secret test set. Note that the secret train/test set may not have the same number of points as the public train/test set that we have provided you. However, we assure you that the public and secret PUFs look similar otherwise so that hyperparameter choices that seem good to you during validation (e.g. step length, stopping criterion etc), should work decently on our secret dataset as well. We will perform 5 experiments. In each experiment, we will offer your code a different timeout (0.2 sec, 0.5 sec, 1 sec, 2 sec, 5 sec) and look at the model obtained by your code after each of these timeouts. We will award marks based on two performance parameters

1. How small a hinge loss does your final model (at timeout) offer? (3 marks)
2. How small a misclassification error rate does your final model offer? (3 marks)

Thus, the total marks in each experiment is 6 marks for a total of 30 marks for all 5 experiments. We will run your code 5 times for each experiment and use the average performance parameters so as to avoid any unluckiness due to random choices inside your code in one particular trial. For more details, please take a look at `eval.py` in the evaluation package. Once we receive your code, we will simply run `eval.py` to obtain the performance statistics and use those to award marks to your submission.

Warning Regarding Secret Dataset. In order to help you understand how we will evaluate your submission using `eval.py`, we have included a dummy secret train and secret test set in in the assignment package itself (see the file called `secret`). However, note that these are just subsets of the train and test dataset we provided you. The reason for providing this dummy secret dataset is to allow you to check whether the evaluation script is working properly or not. Be warned that the secret dataset on which we actually evaluate your submission will be a different one.

Using Internet Resources. You are allowed to refer to textbooks, internet sources, research papers to find out more about this problem and for specific derivations e.g. the XOR-PUF problem. However, if you do use any such resource, cite it in your PDF file. There is no penalty for using external resources but claiming someone else's work (e.g. a book or a research paper) as one's own work without crediting the original author will attract penalties.

Restrictions on Code Usage. You are prohibited from using machine learning libraries such as sklearn, scipy, libsvm, keras and other such packages in any manner while solving this problem. The usage of any machine learning libraries for whatever reason will result in heavy penalties. This is because these packages contain powerful solvers which if used as a function call, render this assignment all but trivial. The only exception to this is the Khatri-Rao product which you are allowed to use from scipy. Please note that you are not allowed to use any other routine from the scipy library. For this assignment, you should also not download any code available online or use code written by persons outside your assignment group (we will relax this restriction from the next assignment onward). Direct copying of code from online sources or amongst assignment groups will be considered and act of plagiarism for this assignment and penalized according to pre-announced policies.

(60 marks)

2 How to Prepare the PDF File

Use the following style file to prepare your report.
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.sty`

For an example file and instructions, please refer to the following files
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.tex`
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.pdf`

You must use the following command in the preamble

```
\usepackage[preprint]{neurips_2022}
```

instead of `\usepackage{neurips_2022}` as the example file currently uses. Use proper LaTeX commands to neatly typeset your responses to the various parts of the problem. Use neat math expressions to typeset your derivations. Remember that all parts of the question need to be answered in the PDF file. All plots must be generated electronically - no hand-drawn plots would be accepted. All plots must have axes titles and a legend indicating what the plotted quantities are. Insert the plot into the PDF file using proper LaTeX `\includegraphics` commands.

3 How to Prepare the Python File

The assignment package contains a skeleton file `submit.py` which you should fill in with the code of the method you think works best among the methods you tried. You must use this skeleton file to prepare your Python file submission (i.e. do not start writing code from scratch). This is because we will autograde your submitted code and so your code must have its input output behavior in a fixed format. Be careful not to change the way the skeleton file accepts input and returns output.

1. The skeleton code has comments placed to indicate non-editable regions. **Do not remove those comments**. We know they look ugly but we need them to remain in the code to keep demarcating non-editable regions.

2. We have provided you with data points in the file `train.dat` in the assignment package that has 10000 data points, having an 8-bit challenge and a 1 bit response. You may use this as training data in any way to tune your hyperparameters (e.g. step length, or

coordinate choice schedule) by splitting into validation sets in any fashion (e.g. held out, k-fold). You are also free to use any fraction of the training data for validation, etc. Your job is to do really well in terms of coming up with an algorithm that can learn a model to predict the responses in the test set accurately and also perform learning as fast as possible.

3. The code file you submit should be self contained and should not rely on any other files (e.g. other .py files or else pickled files etc) to work properly. Remember, your ZIP archive should contain only one Python (.py) file. This means that you should store any hyperparameters that you learn (e.g. step length etc) inside that one Python file itself using variables/functions.

4. We created two 3-way XOR-PUFs – a public one and a secret one. Using the public XOR-PUF, we created CRPs that were split into the train and test set we have provided you with the assignment package. We similarly created CRPs using the secret XOR-PUF and created a secret train and secret test set with it. We will use your code to train on our secret train set and use the learnt model to test on our secret test set. Both the public and secret XOR-PUFs have 8-bit challenges and use 3 PUFs each. However, the delays in the PUFs are not the same so a linear model that is able to predict the public XOR-PUF delays will do poorly at predicting the secret XOR-PUF delays and vice versa. Moreover, note that the secret train set is not guaranteed to contain 10000 points and the secret test set is not guaranteed to contain 20000 points although the number of points (for example, our secret test set may have 19000 points or 21000 points etc). However, we assure you that the public and secret PUFs look similar otherwise so that hyperparameter choices that seem good to you during validation (e.g. step length, stopping criterion etc), should work decently on our secret dataset as well.

5. Certain portions of the skeleton code have been marked as non-editable. Please do not change these lines of code. Insert your own code within the designated areas only. If you tamper with non-editable code (for example, to make your code seem better or faster than it really is), we may simply refuse to run your code and give you a zero instead (we will inspect each code file manually).

6. You are allowed to freely define new functions, new variables, new classes in inside your submission Python file while not changing the non-editable code.

7. You are not allowed to use any machine learning libraries such as sklearn, scipy, keras, pytorch, pandas etc in your submission Python file. Do not use the sys library either. You are expected to implement your method yourself from scratch. You are allowed to use only basic Python libraries such as numpy, time and random which have already been included for you. The only exception to this is the Khatri-Rao product which you are allowed to use from scipy. Please note that you are not allowed to use any other routine from the scipy library. Do take care to use broadcasted and array operations as much as possible and not rely on loops to do simple things like take dot products, calculate norms etc otherwise your solution will be slow and you may get less marks.

8. The assignment package also contains a file called `eval.py` which is an example of the kind of file we will be using to evaluate the code that you submit. Before submitting your code, make sure to run `eval.py` and confirm that there are no errors etc.

9. You do not have to submit `eval.py` to us – we already have it with us. We have given you access to the file `eval.py` just to show you how we would be evaluating your code.