Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: January 25, 2020
Total: 80 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:
`http://web.cse.iitk.ac.in/users/purushot/courses/ml/2019-20-w/material/assn1.zip`
Deadline for all submissions: 15 February, 9:59PM IST
There is no provision for "late submission" for this assignment

1.1 How to submit the PDF 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. Gradescope allows you to submit in groups - please use this feature to make a group submission.
3. To clarify the above, there are currently 39 assignment groups. We wish to have only 39 submissions on Gradescope, i.e. one submission per assignment group, not one submission per student.
4. 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.
5. 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.
6. Do not submit Microsoft Word or text files. Prepare your PDF file using the style file we have provided (instructions on formatting given later).

1.2 How to submit the 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 else files in other languages such as C/Matlab/Java. We will use automated scripts to evaluate your code and will not bother to 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. Name your ZIP file `submit.zip`. 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/XBoSdMP61fx3iFbm7`

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 we will not bother to make any efforts to manually locate your file or else contact you for clarifications. We will simply give your group a 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 simply assign your group zero marks.

15. We will entertain no submissions or mercy appeals 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 (Sparse Recovery)**.** Consider the following formulation for a real-valued regression problem involving n labelled data points $(\mathbf{x}^i, y^i)_{i=1,\ldots,n}$ where $\mathbf{x}^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$. The formulation uses the least-squared loss and the L_1 regularizer instead of the L_2 regularizer.

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{w}\|_1 + \sum_{i=1}^{n} \left(y^i - \langle \mathbf{w}, \mathbf{x}^i \rangle\right)^2 \qquad (P1)$$

Taking $X = [\mathbf{x}^1, \ldots, \mathbf{x}^n]^\top \in \mathbb{R}^{n\times d}$ and $\mathbf{y} = [y^1, \ldots, y^n]^\top \in \mathbb{R}^n$, (P1) can be rewritten more compactly as

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{w}\|_1 + \|X\mathbf{w} - \mathbf{y}\|_2^2$$

The above formulation is called the LASSO and is very useful for regression problems where there are a large number of features i.e. d is very large but there is also a shortage of data, e.g. $n < d$. In such situations, it is sometimes useful to try and learn a *sparse* linear model. A sparse linear model is simply a vector $\mathbf{w} \in \mathbb{R}^d$ that has lots of its coordinates set to zero.

Your Data. We have provided you with data for a 1000 dimensional regression problem with 800 training points in the assignment package (in a file called `train`). Note that we indeed have $d \gg 1$ and $n < d$ in this dataset. If you wish, you may create (held out/k-fold) validation sets out of this data in any way you like. It is known that there exists a very sparse model (with only 20 of the 1000 coordinates with non-zero values) that offers a good solution to the above regression problem. Mathematically speaking, it is assured that there exists an unknown $\mathbf{w}^* \in \mathbb{R}^{1000}$ such that $\|\mathbf{w}^*\|_0 \leq 20$ but $y^i \approx \langle \mathbf{w}^*, \mathbf{x}^i \rangle$ for all i (there is some noise in the labels so we should not expect $y^i = \langle \mathbf{w}^*, \mathbf{x}^i \rangle$ i.e. the labels may not be exactly predicted by $\mathbf{w}^*$).

Finding such a sparse model is challenging since we are not told which 20 coordinates are non-zero and testing all possible $\binom{1000}{20}$ combinations will take too long. To overcome this problem, researchers have devised several solutions but a very successful one involves the LASSO problem (P1). Basically, we solve (P1) which is a convex (but non-differentiable) problem to obtain a model, say $\hat{\mathbf{w}}$. We then look at the top 20 coordinates of $\hat{\mathbf{w}}$ in terms of magnitude and set all the rest to zero to obtain a sparse model $\hat{\mathbf{w}}_{\text{sparse}}$. It can be mathematically proven that under some mild conditions, this sparse vector $\hat{\mathbf{w}}_{\text{sparse}}$ approximates $\mathbf{w}^*$ very well i.e. $\|\hat{\mathbf{w}}_{\text{sparse}} - \mathbf{w}^*\|_2 \to 0$.

Your Task. The following enumerates 4 parts to the question. Parts 1-3 needs to be answered in the PDF file. Part 4 needs to be answered in the Python file.

1. Implement at least two methods to solve the LASSO problem. You may choose your two methods from the list below or propose your own method that you feel is better.

   (a) Primal Subgradient Descent on P1 (note that P1 is non-differentiable)

   (b) Primal Stochastic (possibly with mini-batch) Subgradient Descent on P1

   (c) Primal Stochastic Coordinate Descent on P1

   (d) Primal Proximal Gradient Descent on P1

   Note that we have not included algorithms on the *dual* of (P1) above for sake of simplicity. However feel free to implement a dual method if you like. For each of the 2 (or more) methods you implement, tell us the name of the method and write down all details about how you implemented the method. For example, if you used mini-batch SGD, write down the gradient expression and tell us what batch size and step lengths did you choose. For another example, if you used coordinate descent, write down the formula for the gradient

along a chosen coordinate as well as which method you used to select the next coordinate along which to minimize. For a yet another example, if you used proximal gradient descent, write down the prox function you used for the LASSO. (2 x 10 = 20 marks)

2. Also tell us, for each of the methods you implement, how did you arrive at the best values for the hyperparameters in the methods, 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. (2 x 5 = 10 marks)

3. Plot the convergence curves offered by your chosen 2 methods as we do in lecture notebooks. The x axis in the graph should be time taken and the y axis should be the value of the objective function in (P1). Plot all curves on the same graph (not on different graphs) so that they can be compared. Include this graph in your PDF file submission as an image. Write down which method you think performs the best. (10 marks)

4. Of the methods with which you experimented, choose the one you think is the best method in terms of reducing the objective value as well as recovering $\mathbf{w}^*$ and submit code for that method in `submit.py` (we do not need code for the other method(s), just the one you thought was best). We will evaluate your method on a slightly different dataset than the one we have given you and check how good is the method you submitted. (40 marks)

Parts 1-3 need to be answered in the PDF file. Part 4 needs to be answered in the Python file.

Evaluation Measures and Marking Scheme. Your submitted code will be executed on a secret test dataset. However, the test data set will be similar to the train dataset (it will have roughly same number of data points each with a similar number of features). For the test dataset also, there will be guaranteed to exist some very sparse model (sparsity around 20 as in the train dataset) which is known to perform well on the test dataset.

We will perform 4 experiments. In each experiment, we will offer your code a different timeout (0.1 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 three performance parameters

1. How small an objective value of (P1) does your final model (at timeout) offer? (7 marks)

2. How close (Euclidean distance) is your final model to the secret sparse model? (2 marks)

3. Once we have done the sparsification step on your final model (by looking at the coordinates with largest magnitude), how many of the coordinates actually non-zero in the secret model are revealed? (1 mark)

Thus, the total marks in each experiment is 10 marks for a total of 40 marks for all 4 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 experiment. 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 Gold Model. In order to help you understand the problem better and also so that you may evaluate your learnt model using `eval.py` in the same way as we would evaluate your submission on the test dataset, we have actually included the secret sparse model for the training dataset in the assignment package itself (see the file called `wAstTrain`). Use this model only to check whether the evaluation script is working properly. Be warned that the secret sparse model for the test dataset will be a different one (although it will still be sparse).

Using Resources from the Internet. You are allowed to refer to textbooks, internet sources to find out more about this problem and for specific derivations e.g. the prox function for the L_1 regularizer. 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 paper) as one's own 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. For this assignment, you should not download code available online or code written but persons outside your assignment group either (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 preannounced policies.

(80 marks)

2 How to Prepare the PDF File

Use the following style file to prepare your report.
`https://media.neurips.cc/Conferences/NeurIPS2019/Styles/neurips_2019.sty`

For an example file and instructions, please refer to the following files
`https://media.neurips.cc/Conferences/NeurIPS2019/Styles/neurips_2019.tex`
`https://media.neurips.cc/Conferences/NeurIPS2019/Styles/neurips_2019.pdf`

You must use the following command in the preamble

```
\usepackage[preprint]{neurips_2019}
```

instead of `\usepackage[final]{neurips_2019}` 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 parts 1-5 (and the bonus part 7 if you are attempting it) 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` in the assignment package that has 800 data points, each being 1000 dimensional. You may use this as training data in any way to tune your hyperparameters (e.g. step length) 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. but your job is to do really well in terms of minimizing the objective function of (P1) and approximating $\mathbf{w}^*$ and that too 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 will test your submitted code on a secret test dataset which would be slightly different from the one we have provided you. Thus, your code should not assume it will be given exactly 800 data points (for example, our test dataset may have 900 points or 700 points etc) nor should it assume that each data point has exactly 1000 dimensions (for example, our test dataset may have 800 or 1200 dimensions etc). However, we assure you that our secret test dataset looks similar to the train dataset so that hyperparameter choices that seem good to you during validation, 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, 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. 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.

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.