Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: September 19, 2017
Total: 100 marks

ASSIGNMENT 2

Instructions

1. Only electronic submissions to this assignment will be accepted. No handwritten, scanned or photographed submissions would be accepted.
2. Submissions are to be made separately in two parts - theory part and code part. Instructions for each of these parts are given below.
3. Submissions will be accepted till October 10, 2017, 2359 hrs, IST.
4. Your submission will be considered complete only when both parts are submitted. If both parts are not submitted by the above deadline, your submission will be considered late.
5. Late submissions will be accepted till October 12, 2017, 2359 hrs, IST.
6. Late submissions will incur a penalty – the maximum marks for a late submission will be 80% of the total marks, even if solutions to all questions are absolutely correct.
7. We will be closely checking your code and theory submissions for instances of plagiarism.
8. You may be penalized if you do not follow the formatting and code submission instructions given below very carefully. We will be using automated scripts to perform part of the evaluation. If your file names, format etc are wrong, the scripts will not detect your answer and you will get a penalty.

Theory Part Submission

1. Your submission should be a single PDF file. No zip/tar/png/jpg files will be accepted.
2. The PDF file should have been compiled using the LaTeX style file provided to you.
3. A sample submission has been include to demonstrate the use of this style file.
4. Your answer to every question should begin on a new page. The style file is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the new question, to enforce this.
5. Submissions for this part should be made on Gradescope `https://gradescope.com`.
6. Late submissions for this part should be made on Gradescope itself.
7. An account has been created for you on this website. Use your IITK CC ID (not GMail, CSE etc IDs) to login and use the "Forgot Password" option to set your password initially.
8. While submitting your assignment on this website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.

9. Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n+1$ otherwise graders might miss your figures and award you less points. Again, the style file should do this automatically but be careful.

10. Your solutions must appear in proper order in the PDF file i.e. your solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n+1$.

11. We may impose a penalty on submissions that significantly deviate from the style file or which do not following the formatting instructions.

Code Part Submission

1. This submission should be a single ZIP file. No PY/PYC/M files will be accepted.

2. The name of the ZIP file should be your roll number. Eg. 17001.zip. If your submission is wrongly named, we may be unable to link it to you and you may lose credit.

3. You may resubmit but do not resubmit more than twice – you may incur a penalty for excessive resubmission. We will simply accept your latest submission.

4. Submissions for the code part should be made to the following URL
`https://www.dropbox.com/request/rpnMBX6kuJQbZqeqxl0b`

5. Late submissions for the code part should be done to the following URL
`https://www.dropbox.com/request/Ns1yJh8T8gSkv4KDgyaX`

6. The first URL will close down at the deadline. The second URL will stay open till the late submission deadline. However, it is your responsibility to make sure that if you are submitting early, you do not submit to the late URL by mistake. If you do that, you will needlessly incur the 80% penalty.

7. The ZIP file should be no more than 500KB in size.

8. Do not include your PDF file from the theory part in this submission.

9. **Do not include any data files (training features etc) in your ZIP archive**. Your archive should only contain code and model files.

10. Your code must be well commented and must execute/compile without need for special packages or installations. If we are unable to execute your code, you may be asked to put up a demonstration and incur a penalty too.

Problem 2.1 (Oh holy ID3, show me how to grow a tree!). Consider the following toy dataset that you can use to learn how to choose an advisor.

S.No.	Name	Size of research group	Like the research area?	Average workload?	# of meetings per week	Good ? advisor
1	Prof. R. Lupin	small	yes	average	$2-3$	no
2	Prof. P. Sprout	large	no	average	$0-1$	yes
3	Prof. R. Hagrid	small	no	average	> 3	no
4	Prof. C. Binns	medium	no	heavy	$0-1$	no
5	Prof. A. Sinistra	large	no	heavy	$0-1$	no
6	Prof. S. Snape	medium	no	heavy	$0-1$	yes
7	Prof. S. Kettleburn	small	no	light	> 3	no
8	Prof. Firenze	small	no	average	$2-3$	no
9	Prof. S. Vector	small	yes	average	$0-1$	yes
10	Prof. R. Hooch	medium	no	light	$0-1$	no
11	Prof. C. Burbage	large	no	average	$0-1$	yes
12	Prof. A. P. W. B. Dumbledore	medium	yes	light	$0-1$	yes
13	Prof. S. Trelawney	small	no	light	$2-3$	no
14	Prof. H. Slughorn	medium	no	heavy	$0-1$	no
15	Prof. Q. Quirrell	large	yes	heavy	$0-1$	no

1. Is the first attribute (name) useful in learning a binary classifier from this data? Why?

2. Is it possible to perfectly classify this data (no matter how complicated the classification algorithm)? Why?

3. Read about the ID3 decision tree learning algorithm. There are many nice resources available for this online. The following website does a nice job with examples `https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm`. Build a decision tree using the ID3 algorithm on the data given above. You may stop splitting nodes that are at depth 2 or more (the root of the tree is depth 0). For any impure leaves, use the majority label at that leaf to take a decision for data points in that leaf. Break any ties arbitrarily.

4. *Fake Bonus*: Can you guess which research areas do I like?

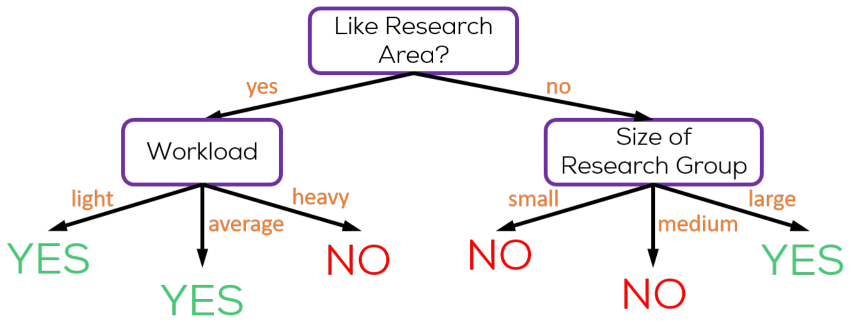

Figure 1: A sample decision tree – not necessarily the one that ID3 would generate :).

Draw the decision tree that you have learnt (see sample above) and include it in your PDF file as an image. You do not need to write/submit code for this problem. However, you do need to justify the following:

1. Why you chose to split a certain node along such and such attribute. Do this by giving numerical values of the information gain corresponding to various attributes at various nodes as you calculated them. Just give the final values – no calculations need be shown.

2. Why you chose not to split certain nodes further and make them a leaf.

Statutory Warning: Don't use this tree to select an advisor in real life. (2+3+10+0=15 marks)

Problem 2.2 (Guess my Grocery List). I run a supermarket called ML-mart that stores L items. The j^{th} item has cost $c_j \in \mathbb{R}$. The i^{th} customer in ML-mart is represented as a d-dimensional real vector $\mathbf{x}^i$. Customer i always buys the same subset of items $S^i \subset [L]$ and pays the bill $b^i = \sum_{j \in S^i} c_j + \epsilon^i$ where $\epsilon^i \sim \mathcal{N}(0, \sigma^2)$ (where σ is the same for all users but not known to you) is some noise in the bill amount. The noise is to account for the fact that maybe the person bought some items, say *mirchi* and *dhaniya* – chillies and cilantro, which are not in the list of L items, in addition to items in the set S^i or else had a coupon for a small discount.

However, something went wrong with my billing machine and it erased all records of which of the L items did the customer buy! The only thing the billing machine stored was the customer ID $\mathbf{x} \in \mathbb{R}^d$ and the bill that customer paid $b \in \mathbb{R}^d$. I have such incomplete records for n customers $(\mathbf{x}^i, b^i)_{i=1,\ldots,n}$. Can you help me complete my bills by figuring out which items the customers bought?

1. Cast the above problem as a latent variable learning problem and develop a sound generative story for the same

2. Write down the complete likelihood expression for the observed data

3. Design a hard-assignment alternating optimization algorithm to solve the problem

4. *Bonus 1*: Design a soft-assignment alternating optimization algorithm for the problem

5. *Bonus 2*: Can you design a hard-assignment alternating optimization algorithm that works in $\mathcal{O}(ndL)$ time? It seems the problem reduces to the problem of finding the largest sum principal matrix of an $L \times L$ matrix (that is a sum of a rank-1 matrix and a diagonal matrix) for which I could neither find any NP-hardness result nor any polytime algorithm. Do let me know if you have any idea.

You may assume that d, n, L are not very large (say $L \approx 10$, $d \approx 100$, $n \approx 1000$) and you do not need to worry about the challenges of the kind we encountered in large-scale multi-label learning (as we discussed in lecture 6). Your solution may use upto $\mathcal{O}\left(2^L \cdot n \cdot d\right)$ time and memory without any hesitation (notice the 2^L). To get you started, here are a few tips

1. Your model $\mathbf{\Theta}$ need not be generative, i.e. it is not necessary to model $\mathbb{P}\left[b, \mathbf{x} \,|\, \mathbf{\Theta}\right]$. You can be discriminative, i.e. just model $\mathbb{P}\left[b \,|\, \mathbf{x}, \mathbf{\Theta}\right]$.

2. Use a linear logistic model conditioned on user features $\mathbf{x}^i$ to account for users selecting items (see lecture 6).

3. Assume that items are bought independently i.e. for any user i and any model $\mathbf{\Theta}$, we have $\mathbb{P}\left[(j_1, j_2) \in S^i \,|\, \mathbf{x}^i, \mathbf{\Theta}\right] = \mathbb{P}\left[j_1 \in S^i \,|\, \mathbf{x}^i, \mathbf{\Theta}\right] \cdot \mathbb{P}\left[j_2 \in S^i \,|\, \mathbf{x}^i, \mathbf{\Theta}\right]$. This may remind you of the the naive Bayes assumption and will play a similar simplifying role here.

4. However, be careful. If you are conditioning on b^i, e.g., looking at $\mathbb{P}\left[(j_1, j_2) \in S^i \,|\, \mathbf{x}^i, b^i, \mathbf{\Theta}\right]$, then the items can no longer be assumed to have been independently bought since the price one has paid restricts which items could be bought together (e.g. if b^i is small then the customer could not have purchased many expensive items together).

5. For elementary problems for which we have already seen solutions in class, you need not explain how to solve them. For example, if your proposed algorithm requires you to solve a few ridge regression/SVM problems, just say that such and such technique from lecture number so and so can be used to solve the ridge regression/SVM problem. However, please use this freedom sensibly – please don't use it to just point to the soft assignment lecture and end your solution there :)

6. Do not worry about whether your algorithm will converge, will it converge to a local minima or a saddle point etc. You are solving an NP-hard problem here. Just develop a sensible algorithm for now.

(5+5+10=20+bonus marks)

Problem 2.3 (The Daft Dual). Consider the following problem formulation for n binary labelled data points $(\mathbf{x}^i, y^i)_{i=1,\ldots,n}$ where $\mathbf{x}^i \in \mathbb{R}^d$ and $y^i \in \{-1, +1\}$

$$\begin{aligned}
\underset{\mathbf{w},\{\xi_i\}}{\arg\min}\ & \|\mathbf{w}\|_2^2 + \sum_{i=1}^n \xi_i^2 \\
\text{s.t. } & y^i \left\langle \mathbf{w}, \mathbf{x}^i \right\rangle \geq 1 - \xi_i, \text{ for all } i \in [n] \qquad (P1) \\
& \xi_i \geq 0, \text{ for all } i \in [n]
\end{aligned}$$

This is a variant of the SVM formulation. Using a technique similar to the one you used for the "Break Free from Constraints" problem in Assignment 1, you can show that $(P1)$ is equivalent to minimizing the regularized squared hinge loss (see lecture 7). However, this question is not about that. We are more interested in solving this optimization problem now.

1. Show that the constraints $\xi_i \geq 0$ are vacuous i.e. the optimization problem does not change even if we remove the all constraints $\xi_i \geq 0$ for all $i \in [n]$.

2. Write the Lagrangian for $(P1)$.

3. Give the derivation for the Lagrangian dual problem for $(P1)$.

4. What are the differences between the dual problem for $(P1)$ and the dual problem for the original SVM problem? The dual for the original SVM problem may be referred to, for example, in [**DAU**] equation 11.38.

5. *Bonus*: Are the positivity constraints $\xi_i \geq 0$ vacuous for the original SVM problem as well? Recall that the original SVM has the term $\sum_{i=1}^n \xi_i$ in its objective instead of $\sum_{i=1}^n \xi_i^2$. Why can/cannot we remove the positivity constraints from the original SVM problem without changing the solution?

It is advisable that you become comfortable with the derivation of the dual of the original SVM problem with slack variables (see [**DAU**] S. 7.7 S. 11.5-11.6, [**SSBD**] S. 15.3-15.5) before attempting this problem. Note that in the lecture, we only looked at a simplified version of the SVM problem without slack variables. (10+2+10+3=25+bonus marks)

Problem 2.4 (My First SVM Solver). In this problem we will design a primal and a dual solver for the SVM problem. We have with us a data set with n data points each with a d dimensional feature vector and a binary label $(\mathbf{x}^i, y^i)$, with $\mathbf{x}^i \in \mathbb{R}^d$ and $y^i \in \{-1, +1\}$. On this dataset we wish to minimize the following function

$$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}),$$

where

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n}\left[1 - y^i\left\langle \mathbf{w}, \mathbf{x}^i \right\rangle\right]_+$$

The Lagrange dual of this problem is the following

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \sum_{i=1}^{n}\boldsymbol{\alpha}_i$$
$$\text{s.t. } 0 \le \boldsymbol{\alpha}_i \le 1, \text{ for all } i \in [n],$$

where $Q \in \mathbb{R}^{n\times n}$ with $Q_{ij} = y^i y^j \left\langle \mathbf{x}^i, \mathbf{x}^j \right\rangle$. If we find an $\hat{\boldsymbol{\alpha}}$ that is (near)-optimal, we can use it to reconstruct a model vector as $\hat{\mathbf{w}} = \sum_{i=1}^{n}\boldsymbol{\alpha}_i y^i \mathbf{x}^i$. You have been supplied with a training data set with 300K data points, each point being 54 dimensional. This is a supervised binary classification problem. However, each feature vector is very sparse. Only around 11-12 of the 54 dimensions are non-zero in any data point. Download this dataset from the URL

`http://web.cse.iitk.ac.in/users/purushot/courses/ml/2017-18-a/material/assn2data.zip`

Your job is to design two SVM solvers to minimize the function $f(\mathbf{w})$ over the entire training set. You may use the training data provided to you in any way to tune your parameters (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 well in terms of minimizing the objective function $f(\mathbf{w})$ on the training set (as well as a secret test set which we have with us and will not reveal to you). Execute the following

1. (Part I) Write a solver to execute the gradient descent (GD) algorithm on the primal problem to minimize $f(w)$ (not SGD, not coordinate descent but vanilla gradient descent).
2. (Part II) Write a solver to execute the stochastic coordinate descent (SCD) algorithm (lec10.pdf page 63) on the dual problem. Since the dual problem has a constraint, you will have to use a projection step.
3. (Part III) For the GD solver, does the current iterate $\mathbf{w}^t$ give better performance or the averaged iterate $\bar{\mathbf{w}}^t = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{w}^\tau$ (See [**SSBD**] section 14.3 and Piazza discussion `https://piazza.com/class/j5toxxryhdx56k?cid=276`). The averaging trick works for SGD but is helpful for GD as well when the function being optimized is non-differentiable. Averaging is not required for SCD since the objective function in the dual is smooth and differentiable.
4. (Part IV) What step length selection strategy worked best for GD?
5. (Part V) On the same graph, with time elapsed (the variable `time_elapsed` in the skeleton code) till now on the x-axis and objective value $f(\mathbf{w})$ (over the entire training set of 300K points) on the y-axis, plot the progress of the GD and SCD algorithms for a comparison. The skeleton code stops the SCD/GD procedure after every few iterations to take the current model $\mathbf{w}^t$ (or the current averaged model $\bar{\mathbf{w}}^t$ if you find that more helpful) and evaluate $f(\mathbf{w}^t)$. Does GD reduce the objective value $f(\mathbf{w})$ faster or does SCD reduce the objective value $f(\mathbf{w})$ faster?
6. (Part VI) There are two ways of measuring time spent – one is wall-clock time (the variable `time_elapsed` in the skeleton) and the other is theoretical time (number of iterations till now * amount of work done for each iteration) where

   (a) number of iterations till now = `tick_vals`* `spacing`
   (b) amount of work per iteration is nd for GD and d for SCD

   Plot the above graphs again using theoretial time on the x-axis. Do you see a difference? Please note that in both set of graphs there will be `n_iter/spacing` points on the graph (see the skeleton code).

You are highly encouraged to read the following paper from ICML 2008 `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf` and use equation (9) while building the solver for Part II. Using equation (9) from the paper will allow you to minimize the objective completely along a coordinate instead of taking just a gradient step. Students using this more advanced step are bound to get much faster solvers than those that implement the mediocre projected stochastic coordinate descent algorithm.

Some points of advice regarding the implementation are given below. There are skeleton files `solver_GD.py` and `solver_SCD.py` included in the assignment package. Do look at them.

1. In the notation of the paper, we will use $D_{ii} = 0$ and $U = 1$ in equation (9).
2. You do not have to implement the "shrinking" step proposed in the ICML 2008 paper.
3. Choose a random coordinate to minimize at each step for Part II. The ICML 2008 paper uses a more involved strategy which you need not implement.
4. Do remember that if implemented naively, coordinate descent can take $\mathcal{O}(nd)$ time at each step. You will have to do careful bookkeeping (see for example lec10.pdf page 90) to make sure each step takes only $\mathcal{O}(d)$ time or so.
5. Remember that the data points you have been given are very sparse. The skeleton file will load the data into a compressed sparse row (CSR) matrix. You may choose to densify the matrix or else keep it sparse. Keeping it sparse will yield speedups.
6. You do not require Shogun for this assignment.
7. You are not allowed to use any sklearn, scipy, pandas etc library functions. The only functions you are allowed to use have already been inserted into the skeleton file.

Submission Instructions

1. You have to submit 2 files, one named `solver_GD.py` and the other called `solver_SCD.py`. The first file should contain all the code you have written for implementing the GD algorithm on the primal problem. The second file should contain all the code your have written for implementing the SCD algorithm on the dual problem.
2. Your `solver_GD.py` should have made decisions with respect to step length and whether to choose the final model or the final averaged model.
3. You have to submit 2 model files, one named `model_GD.npy` and the other called `model_SCD.npy`. Both model files should contain a single d-dimensional real vector (nothing more nothing less – dont include learnt alpha values in the model file). The first model file should contain the model you have learnt on the entire training data using `solver_GD.py`. The second model file should contain the model you have learnt using `solver_SCD.py`.

4. **Do not include any data files (.dat etc) inside your submission**. Only your code files (in .py format) and your model file (`model.npy`) should be there inside your submission ZIP file. We have the training files with us already and can easily include it at our own end to test your code. Your total submission size should not exceed 500KB.
5. Please note that your model files/code files go in the ZIP file to Dropbox. Your plots and inferences go in the PDF file to Gradescope.
6. Submit all your code/model files inside your submission ZIP file. Do not have include sub-directories inside the ZIP file. All files should be present directly inside the ZIP file.

General Instructions

1. All your GD code must reside in a single file called `solver_GD.py`. All your SCD code must reside in a single file called `solver_SCD.py`. If your code does not run on our machines, or takes excessively long time to run, you will get a straight zero in one component of this question.
2. Please be careful not to change the way the skeleton file accepts input and writes output.
3. Your model files must contain a single d-dimensional vector and nothing else. See the skeleton file on how to write the model onto a pickled file. If we cannot read your model file, or use it to perform predictions, you will get a straight zero in another component of this question.
4. Please keep in mind that we may execute your code on datasets with a different number, say 1000, data points. If your code takes decisions (e.g. setting the step length) based on the number of data points in the dataset, keep this in mind.
5. Please keep in mind the file names we have specified. If our scripts cannot locate these files, we will not be able to grade your submission.
6. 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.
7. All plots must be embedded in the PDF file – no stray image files will be accepted. Use the `\includegraphics` command in LaTeX to embed images in your submission PDF file.
8. Your PDF submission must describe neatly what the plotted quantities are as well as the main inference that can be drawn from the plot. E.g. if varying k changes the accuracy, what changes do you observe?

(40 marks)