ECE 273 Course Projects
Spring 2020

General Instructions

1) **Deadline: June 5, 2020, 11:59 pm. Online Submission on Canvas.**
2) **Guidelines:** You should work *individually* on the project and choose **one topic** from the list provided below. Alternatively, you can define your own project (see announcement from Prof. Pal on Canvas). You are expected to submit a project report, along with figures, as well as MATLAB/Python source codes, and the datasets with proper citations. You are welcome to compose your report in Jupyter Notebook with live code and visualizations. Wherever applicable, you should write your own code. Any plagiarism will be considered a violation of academic integrity and will be dealt with accordingly. Ideally, we should be able to reproduce the exact results as those in your report; otherwise you will lose points. Negative results are welcome, but be sure to discuss them in your report. The report should include the following sections:
   - Objective/Goal of the project.
   - Background and motivation: You should cite relevant references, explain why the problem is important, and its applications. You should also cite the datasets being used.
   - Method/Techniques.
   - Results: Include all figures; you can also use tables to present your results, especially for comparing multiple algorithms.
   - Discussion: This section will provide a critical discussion of your results, and evaluate the strengths and weaknesses of different algorithms and the associated trade-offs.
   - Dataset: In your report, you should include a web link to all datasets you use (if any) and give proper credit. If the datasets are not publicly accessible, submit them with your report in a separate file. If you have more than one file, pack all files into a .zip file. Dont use the formats (like .rar) that require specific software to unpack your file.
   - You will use CVX (for MATLAB) or CVXPY (for Python) which come with inbuilt solvers (optimized for respective platforms) for convex problems. Since CVX is a software package meant for modeling, you will only need to formulate the optimization problem in a standard form using MATLAB/Python syntax. Refer to your syllabus for more details for resources on how to quickly get started with CVX/CVXPY.

List of Project Topics

1. **Topic 1: Quadratically Constrained Quadratic Program (QCQP)**
   References:
   1) Y. Ye and S. Zhang, "New Results on Quadratic Minimization", *SIAM J. Optim.*, vol. 14, no. 1, pp. 245-267, 2003.
   2) Z. Luo, W. Ma, A. So, Y. Ye and S. Zhang, "Semidefinite Relaxation of Quadratic Optimization Problems", *IEEE Sig. Process. Mag.*, vol. 27, no. 3, pp. 20-34, May 2010.
   3) A. Konar and N. D. Sidiropoulos, "Hidden Convexity in QCQP with Toeplitz-Hermitian Quadratics", *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1623-1627, Oct. 2015.
   4) K. Huang and and N. D. Sidiropoulos, "Consensus-ADMM for General Quadratically Constrained Quadratic Programming" *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5297-5310, Oct. 2016.

   Quadratically constrained quadratic programming (QCQP) is an optimization problem in which both the objective funcion and the constraints are in quadratic forms. The general form of QCQP is given by

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^N} \frac{1}{2}\mathbf{x}^T\mathbf{A}_0\mathbf{x} + \mathbf{b}_0^T\mathbf{x} \quad (\mathbf{QCQP}) \\ &s.t.\ \frac{1}{2}\mathbf{x}^T\mathbf{A}_i\mathbf{x} + \mathbf{b}_i^T\mathbf{x} + c_i \leqslant 0 \quad i = 1, 2, \cdots, m \\ &\quad \mathbf{B}\mathbf{x} = \mathbf{h} \end{aligned}$$

where $\{\mathbf{A}_i\}_{i=0}^m$ are real symmetric matrices of dimension N and we have n equality constrains given by $\mathbf{h} \in \mathbb{R}^n$. In Homework 1 Problem 3, we studied conditions for the set $S = \{\mathbf{x} \in \mathbb{R}^N | \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \leqslant 0, \mathbf{g}^T\mathbf{x} = h\}$ to be convex. Additionally, when $\{\mathbf{A}_i\}_{i=0}^m$ are all positive definite, the problem is convex. However, in general, (**QCQP**) is not convex and either non-convex algorithms or proper convex relaxations are necessary. Nonconvex quadratically constrained quadratic programming problems have numerous applications in signal processing, machine learning, and wireless communications.

In this project, you are expected to do all of the followings:

- Consider the general (**QCQP**) and read the references listed above and other related literature therein. Find and summarize all the relaxations of non-convex QCQP such that convex programming methods can be applied.
- The solution of the relaxed version may not be equal to the original non-convex QCQP. Find conditions under which the relaxed convex program will give the same solution (or good approximation) of the original form.
- It is well known that if $\{\mathbf{A}_i\}_{i=0}^m$ are Positive definite, QCQP is convex. But this may not be the only sufficient condition for QCQP to be convex. Read related literature and find other possible (sufficient) conditions for original (QCQP) to be convex.
- Read the above papers to find an application of convex QCQP and implement it using CVX/CVXPY. Test your algorithm on synthetic data.
- **(BONUS)**: There exist many non-convex algorithms for solving (QCQP), you will receive bonus points if you can do a literature review and implement some of them. Compare these algorithms with convex relaxations.

2. **Topic 2: Blind Deconvolution**
   References:
   1) Ahmed, A. and Recht, B. (2014). Blind Deconvolution Using Convex Programming. IEEE Transactions on Signal Processing,vol. 60, no. 3, Mar. 2014..

Blind deconvolution is the problem of recovering two signals given the output of their convolution. The signals can not be recovered unless certain restrictions are imposed on them. In this project, you will study and implement convex algorithms for solving the blind deconvolution problem under suitable conditions, and evaluate their performance guarantees.

- Study the paper by Ahmed, Recht and Romberg to understand the formulation of the blind deconvolution problem. Why is the solution to the problem non-unique in general? Can you explain what kind of ambiguities exist? Under what conditions on the signals (sparsity, subspace structure etc), the problem can permit unique solutions? Explain why.
- Why is the problem non-convex? Explain the main idea in this paper under which the measurements can be expressed as a linear function of a suitable transformed variable. How is this transformed variable related to the constituent signals, and given this transformed variable, how can one obtain the constituent signals? Does the variable transformation make the problem convex? What additional relaxations are necessary to cast it as a convex problem?
- What are the theoretical guarantees of the convex algorithm proposed in the paper? Are they deterministic, or probabilistic? What should be the relation of the number of measurements, sparsity and the dimension of the subspace, so that exact (or stable, in presence of noise) recovery is possible? Clearly state the mathematical guarantees and spell out the assumptions.
- Implement the convex algorithm proposed in the paper. You should be able to recreate the figures. Test the algorithm by generating the phase transition plots which plots the probability of successful recovery as a function of number of measurements, and the sparsity and/or dimension of the subspace. Do you see a sharp region around which the probability changes from close to zero, to close to 1? Compare this with

the theoretical guarantees from the paper.
- Compare performance of the blind deconvolution algorithm against non-blind deconvolution (i.e. where you are given one of the constituent signals). Do a literature survey on alternating minimization techniques for blind deconvolution (which are non-convex), and implement one algorithm of your choice. Compare the performance of convex and non-convex blind deconvolution.
- Design numerical experiments to evaluate the performance of the algorithm when one (or more) conditions (such as sparsity, low rank) are violated. Can you comment on the robustness of the algorithm against such violations?
- **(BONUS)**: Extend the algorithm to perform blind deconvolution in $2D$. One example is image deblurring. Test the algorithm on an image of your choice (by blurring it with a suitable kernel).

3. **Topic 3: Polynomial Neural Networks and Low Rank Matrix Factorization**
References

1) Soltani, Mohammadreza, and Chinmay Hegde. "Towards provable learning of polynomial neural networks using low-rank matrix estimation". *International Conference on Artificial Intelligence and Statistics*. 2018.
2) Candes, Emmanuel J., and Yaniv Plan. "Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements." *IEEE Transactions on Information Theory* 57.4 (2011): 2342-2359.
3) Recht, B., Fazel, M. and Parrilo, P.A., 2010. "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization". *SIAM review*, 52(3), pp.471-501.
4) Goldstein, Tom, Christoph Studer, and Richard Baraniuk. "FASTA: A generalized implementation of forward-backward splitting." *arXiv preprint arXiv*:1501.04979 (2015).

In [1], the authors showed that training a neural network with one hidden layer consisting of fewer number of units compared to the size of input, and with quadratic activation function can be modeled as low-rank matrix recovery problem with noisy quadratic measurements.

$$\min_{\mathbf{L}\in\mathbb{R}^{p\times p}} F(\mathbf{L}) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \mathbf{x}_i^T\mathbf{L}\mathbf{x}_i)^2 \qquad s.t. \qquad rank(\mathbf{L}) \leqslant r \tag{1}$$

where $\{(y_i, \mathbf{x}_i)\}_{i=1}^m, \mathbf{x}_i \in \mathbb{R}^p$ is the input data set, r is the number of hidden units with the activation function $\sigma(z) = z^2$. The ground truth matrix $\mathbf{L}_*$ is constructed from the weights in the first and second layers of the corresponding neural network in the following form:

$$\mathbf{L}_* = \sum_{j=1}^{r} \alpha_j \mathbf{w}_j \mathbf{w}_j^T$$

where $\mathbf{w}_j$ is the weight vector in the j_{th} hidden unit and $\alpha \in \mathbb{R}^r$ is the weight vector of the second layer.

This optimization problem is not a convex problem (Why?), however, there are approaches which first convert the original problem to a simpler convex problem and then solve it as a convex optimization problem. The authors in [2,3] have proposed a convex relaxation of the low-rank constraint based on the nuclear norm of a matrix. In particular, consider the following problem:

$$\mathbf{y} = \mathcal{A}(\mathbf{L}) + \mathbf{n} \tag{2}$$

where $\mathbf{L} \in \mathbb{R}^{p\times p}$ is an unknown matrix, $\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}^m$ is a linear mapping,

$$[\mathcal{A}(\mathbf{L})]_i = \mathbf{x}_i^T\mathbf{L}\mathbf{x}_i$$

and $\mathbf{n} \in \mathbb{R}^m$ denotes additive noise. The desired matrix $\mathbf{L}_*$ (which has p^2 unknown entries) can be recovered from (compressive) measurements (with $m \ll p^2$) by solving the following convex problem:

$$\min_{\mathbf{L} \in \mathbb{R}^{p \times p}} \|\mathbf{L}\|_*$$

$$\begin{aligned} s.t. \qquad & \|\mathcal{A}^*(\mathbf{e})\| \leqslant \lambda \\ & \mathbf{e} = \mathbf{y} - \mathcal{A}(\mathbf{L}) \end{aligned} \tag{3}$$

where $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$,

$$\mathcal{A}^*(\mathbf{e}) = \sum_{i=1}^{m} e_i \mathbf{x}_i \mathbf{x}_i^T$$

$\|.\|$ is the operator norm, $\|.\|_*$ is its dual (i.e the nuclear norm), and λ is a constant which is dependent on the size of $\mathbf{L}$ and noise.

In this project, you are expected to do all of the followings (exlcuding BONUS):

- Read the references and explain why solving the problem (1) is equal to training the corresponded network? Why is it not a convex optimization problem? Why is problem (3) a suitable convex relaxation for problem (2)?
- Generate a random data set $\{\mathbf{x}_i\}_{i=1}^m$ from a normal distribution (you can choose the mean and the covariance), and one ground truth network (Based on the given architecture in [1]) which generates the output y_i corresponding to the input $\mathbf{x}_i$. Implement Algorithm 1 in [1] and test it on the data set you have generated. Try data sets with different sizes and different initialization. Can you recover the ground truth $\mathbf{L}_*$ (or equivalently, the network)? Plot the probability of success by varying the values of m, p and r, as well as different initializations (you can refer to [1] for some representative plots). Comment on your results. Add noise to the measurements and try different noise powers. Report where the algorithm fails and succeeds.
- Formulate the problem of training neural network (or equivalently, finding $\mathbf{L}_*$) as a convex problem, based on [2] and [3]. Use the data set you generated in the last part and solve this convex problem with CVX/CVXPY library. You may need to tune λ by trying different values and cross validating against the training dataset (Read [2] for suitable ways to choose λ). Compare your result with previous solutions and plots, including the need for initialization. Try different sizes of data sets. How accurate is the convex relaxation for this problem? How sensitive is it to the size of data set and noise?
- Generate a new data set where each $\mathbf{x}_i$ can come from two different Gaussian distributions (representing two classes) with two different means (you can keep the covariance same if you wish). Assign each data point a binary label based on its class. (Note that there is no ground truth network in this case). Run the two algorithms that you have implemented in the last two parts (non-convex and convex) to train the polynomial neural network (which is also equivalent to learning a matrix $\mathbf{L}$). Generate some new test data points from your classes and evaluate the performance of the learned network on the test dataset. Are these algorithms able to classify the data? If not, why? Try to justify.
- **(BONUS)** If you were able to solve a binary classification problem on the random data set in previous question, try to apply your algorithm on the MNIST dataset (You can restrict your dataset to only 0 and 1 digits). Is there any way to extend this idea for multi-class classification? How?
- **(BONUS)** Authors in [1] have compared their algorithms with a convex solver proposed in [4] (FASTA). Read this paper and implement this algorithm. Apply FASTA on your generated data set and compare the results with previous parts.

4. **Topic 4: 1-bit Compressed Sensing**
   References:
   1) Plan, Y., and Vershynin, R. (2013). Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. IEEE Transactions on Information Theory, 59(1), 482-494.
   2) Chen, L., Hassani, S. H., and Karbasi, A. (2017). Near-Optimal Active Learning of Halfspaces via Query Synthesis in the Noisy Setting. In AAAI (pp. 1798-1804).

   Compressed Sensing (sparse signal recovery) has been a popular research topic in signal processing since early 90s. It has many applications in image processing, machine learning, communication, radar, and medical imaging. The main goal of Compressed Sensing is to recover a sparse signal $\mathbf{x}$ ($\|\mathbf{x}\|_0 \leqslant s$) from linear measurements of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is a measurement matrix. For many practical applications, infinite-precision linear measurements are not available. Instead, linear measurements are collected with finite precision (quantization). In the extreme case with 1-bit quantization, the noiseless 1-bit compressed sensing measurements is modeled as

$$y_i = \text{sign}(\mathbf{a}_i^T \mathbf{x}), i = 1, 2, \cdots m \tag{4}$$

   In this project, you are expected to do the following:
   - Find two different applications of 1-bit Compressive Sensing. Describe why 1-bit is important in those applications. Examine their noise models and comment on whether you think the noise is modelled reasonably in each application. Cite references in your report
   - Read [1] and compute, by simulation, the Gaussian mean width of the sparse signal set $w(S_{n,s})$. Fix $s = 5$ and plot $w(S_{n,s})$ versus n with its upper bound and lower bound given by Lemma 2.3
   - Let $S^1_{n,s} = \{\mathbf{x} : \mathbf{x} \in S_{n,s}, x_i \geqslant 0\}$, $S^2_{n,s} = \{\mathbf{x} : \mathbf{x} \in S_{n,s}, x_i = c \text{ or } 0\}$ and $S^3_{n,s} = \{\mathbf{x} : \mathbf{x} \in S_{n,s}, x_i = \pm c \text{ or } 0\}$ Plot $w(S^1_{n,s})$, $w(S^2_{n,s})$ and $w(S^3_{n,s})$ with (b). Interpret the result by using Theorem 1.1 in [1]
   - Explain why the relaxation from l_0-norm to l_1-norm is reasonable, by using the Gaussian mean width, (III.1) and Theorem 1.1
   - In [2], the measurement/observation is also $\{+1, -1\}$. Compare the problem in [1] and [2] and describe the mathematical differences between them
   - By using the noise model in [2], implement both algorithms in [1] and [2] and compare them numerically for $s = 1$ to $s = n$ with various n, which one is better and why.
   - **(BONUS)** Develop an algorithm that incorporates the sparse constraint s into the algorithm in [2] and compare your algorithm with [1].