Convex Optimization and Applications: Coding Assignment

(points: 100, due on June 15)

Problem 1 (points: 6 + 4 + 6 = 16)

Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{R}^d$ and $\boldsymbol{b} \in \mathbb{R}^d$. Consider the following optimization problems:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \sum_{i=1}^{n} |\boldsymbol{a}_i^\top \boldsymbol{x} - b_i| \qquad \text{and} \qquad \min_{\boldsymbol{x}\in\mathbb{R}^d} \left[\max_{1\leqslant i\leqslant n} |\boldsymbol{a}_i^\top \boldsymbol{x} - b_i| \right].$$

(i) Transform the above problems into linear programs (LP) and implement them in CVX (you can download the package from `http://cvxr.com/cvx/download/`).

(ii) To test the code, create your own data (i.e., $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ and $\boldsymbol{b}$) for $d = 10$ and $n = 200$; needless to say, the data should be such that the programs do work and if possible the results should be interpretable.

(iii) Write CVX scripts that can directly solve the above programs without having to express them as LPs (read the CVX manual to see how this can be done). Compare results with LP codes in Part-(i).

Problem 2 (points: 6 + 4 + 4 + 4 = 18)

Let $A \in \mathbb{R}^{m\times n}$. Recall that for $\boldsymbol{z} \in \mathbb{R}^k$, we have $\|\boldsymbol{z}\|_1 := |z_1| + \cdots + |z_k|$ and $\|\boldsymbol{z}\|_2 := (z_1^2 + \cdots + z_k^2)^{1/2}$. Consider the following variants of the least-squares problem.

Least-squares fitting:	**L$_2$-regularization:**	**L$_1$-regularization:**
$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad (1)$	$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2 \quad (2)$	$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \quad (3)$

(i) Run the attached script "gendata_lasso.m" using $[A, \boldsymbol{b}] =$ gendata_lasso ( ) to get A and $\boldsymbol{b}$. Write CVX scripts that can solve the regression models (1) - (3). Plot the objective at each iteration.

(ii) Vary λ over $\{5, 10, 15, \ldots, 100\}$ for models (2) and (3).

  (a) Let $\boldsymbol{x}^\star(\lambda)$ and $\hat{\boldsymbol{x}}(\lambda)$ be the minimizers of (2) and (3), respectively.
  - Plot $\boldsymbol{x}^\star(\lambda)$ for different values of λ in the same plot.
  - Likewise, in a different figure, plot $\hat{\boldsymbol{x}}(\lambda)$ for different values of λ.
  - Comment on the nature of $\boldsymbol{x}^\star(\lambda)$ and $\hat{\boldsymbol{x}}(\lambda)$ as λ changes.

  (b) Note that the objective functions in (2) and (3) have two components. For (2), graph $\|A\,\boldsymbol{x}^\star(\lambda) - \boldsymbol{b}\|_2$ and $\lambda\,\|\boldsymbol{x}^\star(\lambda)\|_2$ for different values of λ in the same plot. Similarly, for (3).

(iii) Run $[A, \boldsymbol{b}] =$ gendata_lasso $(200, 50, \sigma, 1)$ and solve the three models (see comments in the code gendata_lasso.m to understand what the parameters stand for). Fix a suitable λ from Part-(ii), and vary σ in gendata_lasso over $\{0.1, 0.2, \ldots, 2\}$. Plot the root mean square error versus σ; comment on the results.

(iv) Run $[A, \boldsymbol{b}] =$ gendata_lasso $(200, 50, \sigma, 2)$ and solve the three models. Fix a suitable λ from Part-(ii), and vary σ in gendata_lasso over $\{0.1, 0.2, \ldots, 2\}$. Plot the root mean square error versus σ; comment on the results.

Problem 3 (points: 3 + 3 + 5 + 7 = 18)

The objective of this question is to give you an idea about the Support Vector Machine (SVM), which is one of the most successful binary classification techniques in machine learning. Suppose we are given m labeled data points $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$, where $\boldsymbol{x}_j \in \mathbb{R}^n$ and $y_j \in \{1, -1\}$ for $j = 1, \ldots, m$, which are linearly separable. The SVM training-problem is to find the widest *margin* that separates

$$S_1 := \{\boldsymbol{x}_j \ : \ \text{the corresponding label } y_j = 1\} \quad \text{and} \quad S_2 := \{\boldsymbol{x}_j \ : \ \text{the corresponding label } y_j = -1\}.$$

In other words, the objective is to find the hyperplane $\mathcal{H}_0 := \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \boldsymbol{w}_0^\top \boldsymbol{x} + b_0 = 0\}$ with the widest *margin* that separates S_1 and S_2.

The Margin of a Separating Hyperplane

Suppose a hyperplane $\mathcal{H} := \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \boldsymbol{w}^\top \boldsymbol{x} + b = 0\}$ separates S_1 and S_2, i.e.,

$$\boldsymbol{w}^\top \boldsymbol{x}_j + b > 0 \qquad \forall \boldsymbol{x}_j \in S_1, \tag{4a}$$

$$\boldsymbol{w}^\top \boldsymbol{x}_j + b < 0 \qquad \forall \boldsymbol{x}_j \in S_2. \tag{4b}$$

Then, the margin of $\mathcal{H}$ is defined as twice the minimum distance between $\mathcal{H}$ and data points, i.e.,

$$\operatorname{margin}(\mathcal{H}) := 2 \min_{1 \le j \le m} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_j + b|}{\|\boldsymbol{w}\|_2} = 2 \min_{1 \le j \le m} \frac{y_j \left(\boldsymbol{w}^\top \boldsymbol{x}_j + b\right)}{\|\boldsymbol{w}\|_2}.$$

Note that even if we change $\boldsymbol{w} \mapsto \alpha\, \boldsymbol{w}$ and $b \mapsto \alpha\, b$, where $\alpha > 0$, the hyperplane $\mathcal{H}$ and the binary classification rules given by (4) do not change; thus, $\operatorname{dist}(\boldsymbol{x}_j, \mathcal{H})$ remains unchanged for every $j \in \{1, \ldots, m\}$. Therefore, without loss of generality, we can assume that $\boldsymbol{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are chosen such that

$$\min_{1 \le j \le m} \ y_j \left(\boldsymbol{w}^\top \boldsymbol{x}_j + b\right) = 1;$$

as a result, $\operatorname{margin}(\mathcal{H}) = \dfrac{2}{\|\boldsymbol{w}\|_2}$.

Remark. *(a) The SVM training-problem can be formulated as the following CP:*

$$\begin{aligned} &\textit{Minimize:} && \frac{1}{2}\,\|\boldsymbol{w}\|_2^2 \\ &\textit{Subject to:} && 1 - y_j(\boldsymbol{w}^\top \boldsymbol{x}_j + b) \le 0, \quad j = 1, \ldots, m. \end{aligned} \tag{5}$$

(b) If $\mathcal{H}_0$ is the hyperplane with the widest margin that separates S_1 and S_2, then the classification rule (see Eq. (4)) on a test data-point $\boldsymbol{v} \in \mathbb{R}^n$ is given by:

- *If $\boldsymbol{w}_0^\top \boldsymbol{v} + b_0 > 0$, then $\boldsymbol{v} \in S_1$.*
- *If $\boldsymbol{w}_0^\top \boldsymbol{v} + b_0 < 0$, then $\boldsymbol{v} \in S_2$.*
- *If $\boldsymbol{w}_0^\top \boldsymbol{v} + b_0 = 0$, then a random label is assigned.*

(c) Read about a linear separation of training data-points in a higher dimensional feature space (which could be even infinite dimensional) using the kernel trick; note that this corresponds to an exact separation of training data-points in the original space with possibly a nonlinear decision boundary. ♦

(i) Show that the Lagrange dual of (5) is given by:

$$\begin{aligned} &\text{Maximize:} && \boldsymbol{\lambda}^\top \mathbf{1} - \frac{1}{2}\,\boldsymbol{\lambda}^\top (Y \Sigma Y)\, \boldsymbol{\lambda} \\ &\text{Subject to:} && \boldsymbol{\lambda} \succeq \mathbf{0}, \\ & && \boldsymbol{y}^\top \boldsymbol{\lambda} = 0, \end{aligned} \tag{6}$$

where $\mathbf{1} \in \mathbb{R}^m$ is the vector of all ones, $Y := \operatorname{diag}(y_1, \dots, y_m) \in \mathbb{S}^m$, $\Sigma := X^\top X \in \mathbb{S}^m_+$ and $X := [\boldsymbol{x}_1 \dots \boldsymbol{x}_m] \in \mathbb{R}^{n\times m}$. Show that if $\boldsymbol{\lambda}^\star$ is an optimal solution of (6), then an optimal solution of (5) is given by

$$\boldsymbol{w}_0 = \sum_{i=1}^m \lambda_i^\star y_i \, \boldsymbol{x}_i.$$

(ii) Run "svm_gendata.m" to generate the training data $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$. Solve (6) on this dataset using CVX. Plot the training error at each iteration.

(iii) Consider the Gaussian kernel given by

$$K(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right) \qquad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{7}$$

Define the matrix Σ in (6) as $\Sigma_{i,j} := K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Solve (6) for $\sigma \in \{10^{-2}, 10^{-1}, 0.5, 10, 10^2\}$ and report the training error in each case; comment on the results.

Remark. *If $\boldsymbol{\lambda}^\star$ is the solution of* (6)*, then the kernel-based classification rule is given by the sign of the following expression:*

$$\sum_{i=1}^m \lambda_i^\star y_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

But how do we set b? (Hint: the inequality constraints in (5) *corresponding to the "support vectors" are active).* ◆

(iv) The dataset "data.mat" contains raw features of handwritten digits 0 - 9, and "label.mat" contains the corresponding labels. Both files contain a training set and a test set. Use the Matlab command *reshape(data, 784, size(data, 3)'./255* to preprocess the training and test data.

(a) Solve the SVM model in (5) on the training data corresponding to the digits '6' and '8'. Use the trained model for classifying '6' and '8' from the test data; report the test accuracy.

(b) Perform the same experiment using the model in (6), both with and without the Gaussian kernel in (7). This time use the digits '1' and '7'.

Problem 4 (points: 6 + 6 = 12)

The objective of this question is to give you an idea about the robust principal component analysis technique. Principal Component Analysis (PCA) is used to find an "optimal" representation of data using fewer dimensions. However, PCA is sensitive to outliers. Robust PCA provides a framework which besides being tolerant to noise (outliers) can also handle sparseness in the data.

Suppose we are given a large data matrix $M \in \mathbb{R}^{n_1\times n_2}$ (rows are the dimensions and the columns are the samples), and know that it can be decomposed as

$$M = L_0 + S_0, \tag{8}$$

where L_0 has low rank and S_0 is sparse. However, we do not know column and row spaces of L_0; also, we do not know the locations of the nonzero entries of S_0. The problem is to "approximately" recover the components L_0 and S_0 from a given M. In the robust PCA technique, this is done by solving the following optimization problem:

$$\begin{aligned} \text{Minimize:} \quad & \|L\|_* + \lambda\, \|S\|_{\ell^1} \\ \text{Subject to:} \quad & L + S = M, \end{aligned} \tag{9}$$

where $\|L\|_*$ is the nuclear norm of $L \in \mathbb{R}^{n_1\times n_2}$ (i.e., the sum of the singular values of L), and $\|S\|_{\ell^1}$ is the vector $\|\cdot\|_1$-norm of $S \in \mathbb{R}^{n_1\times n_2}$ (i.e., the sum of the absolute values of the entries of S).

(i) Load the data matrix M using the attached data set "rpca_data.mat" and solve (9) using CVX. In particular, experiment with $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$, and plot the optimal objective value for each λ. Comment on the results.

(ii) Synthetically generate $M = L + S$, where L is of low rank and S is sparse. Solve for L and S using (9) and see if you can successfully recover L and S, possibly by tuning λ. Using results available in the literature on the robust PCA, can you comment on a correct value of the tuning parameter λ?

Problem 5 (points: 4 + 7 + 7 = 18)

The objective of this question is to give you an idea about the max-cut problem. Consider a weighted undirected simple graph $G(V, E, w)$, where V is the vertex set and $E \subseteq V \times V$ is the edge set. A nonnegative weight $w(i, j)$ is assigned to each edge $(i, j) \in E$.

The max-cut problem is finding a subset $U \subseteq V$ such that the total weight of edges between U and $U^c := V \setminus U$ is maximized. In other words, we are interested in finding a subset $U \subseteq V$ such that the weight of a cut with respect to a subset U given by

$$\mathrm{cut}(U) := \sum_{i \in U,\, j \in U^c} w(i, j) \tag{10}$$

is maximized. Note that corresponding to a partition $V = U \cup U^c$ of the vertex set, we can assign a label $x_i \in \{1, -1\}$ to each vertex as follows:

$$x_i := \begin{cases} 1, & \text{if } i \in U \\ -1, & \text{if } i \in U^c. \end{cases} \tag{11}$$

(i) Explain how the convex relaxation of the max-cut problem is given by the following SDP:

$$\begin{aligned} \text{Maximize:} \quad & \frac{1}{4}\,\mathrm{Trace}(LX) \\ \text{Subject to:} \quad & X \in \mathbb{S}^n_+, \\ & X_{ii} = 1, \quad i = 1, \dots, n. \end{aligned} \tag{12}$$

where $L \in \mathbb{S}^n_+$ is the Laplacian corresponding to the given weighted undirected simple graph $G(V, E, w)$.

(ii) Use the attached code "rgg.m" to generate a graph and solve (12) for this graph using CVX. What is the rank of the optimal solution? Using the randomized approximation algorithm, write a script to recover the labels from the optimal solution of (12); this is called "rounding". What can you say about the tightness of the approximation if the rank is one?

(iii) Consider the complete bipartite graph[1] $K_{8,12}$ with the unit weight assigned to each edge. Using CVX, solve (12) for this graph and report the value of the optimal cut and the corresponding labels. Comment on the tightness of the result and the optimal cut.

Problem 6 (points: 6 + 5 + 7 = 18)

Randomly sample m points in the interval $[-1, 1]$ and denote them as $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$. Let X be a discrete random variable that takes these values, and let $p_j := \mathbb{P}(X = a_j)$. Note that $p_1, \dots, p_m \geq 0$ and $p_1 + \cdots + p_m = 1$. The entropy of the probability distribution $\boldsymbol{p} = (p_1, \dots, p_m)$ is defined by

$$\mathcal{E}(\boldsymbol{p}) := -\sum_{j=1}^{m} p_j \log(p_j),$$

where by the definition: $p_j \log(p_j) = 0$ if $p_j = 0$.

[1] A bipartite graph is a graph whose vertex set can be partitioned (mutually exclusive and exhaustive) into two independent sets V_1 and V_2 such that every edge of the graph connects a vertex in V_1 to a vertex in V_2. Furthermore, in a complete bipartite graph, every vertex in V_1 is connected to every vertex in V_2. A complete bipartite graph, where $|V_1| = m$ and $|V_2| = n$, is denoted by $K_{m,n}$.

(i) Consider the following optimization problem:

$$\begin{aligned} \text{Maximize:} \quad & \mathcal{E}(\boldsymbol{p}) \\ \text{Subject to:} \quad & \boldsymbol{p} \geq \mathbf{0}, \\ & p_1 + \ldots + p_m = 1. \end{aligned} \tag{13}$$

Is (13) a convex program? If yes, then solve it using CVX for $m = 10$; also, plot the optimal distribution. Verify the solution using a pen-and-paper calculation.

(ii) Next, we impose the following prior on the distribution:

$$\begin{aligned} -0.1 &\leqslant \mathbb{E}(X) \leqslant 0.1, \\ 0.5 &\leqslant \mathbb{E}(X^2) \leqslant 0.6, \\ -0.3 &\leqslant \mathbb{E}(3X^3 - 2X) \leqslant -0.2, \\ 0.3 &\leqslant \mathbb{P}(X < 0.5) \leqslant 0.4. \end{aligned} \tag{14}$$

Verify using CVX that the above prior specifies a valid distribution (take $m = 20$).

(iii) Under the conditions in (14), write a CVX program to compute the maximum entropy distribution. Verify that the prior is met by the solution; also, plot the optimal distribution.

****