Exercises for EE364b

Stephen Boyd John Duchi Mert Pilanci

May 16, 2022

Contents

1 Subgradients

1.1 For each of the following convex functions, explain how to calculate a subgradient at a given x.

(a) $f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i)$.

(b) $f(x) = \max_{i=1,\ldots,m} |a_i^T x + b_i|$.

(c) $f(x) = \max_{i=1,\ldots,m} \left(-\log\left(a_i^T x + b_i\right)\right)$. You may assume x is in the domain of f.

(d) $f(x) = \sup_{0\le t\le 1} p(t)$, where $p(t) = x_1 + x_2 t + \cdots + x_n t^{n-1}$.

(e) $f(x) = x_{[1]} + \cdots + x_{[k]}$, where $x_{[i]}$ denotes the ith largest element of the vector x.

(f) $f(x) = \inf_{Ay\preceq b} \|x - y\|^2$, *i.e.*, the square of the distance of x to the polyhedron defined by $Ay \preceq b$. You may assume that the inequalities $Ay \preceq b$ are strictly feasible.

(g) $f(x) = \sup_{Ay\preceq b} y^T x$, *i.e.*, the optimal value of an LP as a function of the cost vector. (You can assume that the polyhedron defined by $Ay \preceq b$ is bounded.)

1.2 *Convex functions that are not subdifferentiable.* Verify that the following functions, defined on the interval $[0, \infty)$, are convex, but not subdifferentiable at $x = 0$.

(a) $f(0) = 1$, and $f(x) = 0$ for $x > 0$.

(b) $f(x) = -\sqrt{x}$.

1.3 Explain how to find a subgradient or quasigradient of the following functions at a given point x_0. If the function is concave or quasiconcave, find a subgradient or quasigradient of $-f$. (Please make it clear whether you are finding a subgradient or quasigradient, and whether it is for f or $-f$.) Try to find efficient ways to evaluate a subgradient or quasigradient; in particular, try to avoid ways that require a number of operations that grows exponentially with the problem sizes.

(a) $f(x) = x_{[1]} + x_{[2]}$ on $\mathbf{R}^n$. (Recall that $x_{[i]}$ denotes the ith greatest component of x).

(b) $f(x) = \|x\|_\infty = \max_k |x_k|$

(c) The function of problem 3.24(h) of the book: for p in the probability simplex,

$$f(p) = \min\{\beta - \alpha \,|\, \mathbf{Prob}(x \in [\alpha, \beta]) \ge 0.9\},$$

where x is a random variable taking values $a_1, \ldots, a_n$ with probabilities $p_1, \ldots, p_n$, respectively.

1.4 *Quantiles.* Given a fixed vector $x \in \mathbf{R}^n$, define $f : \mathbf{R} \to \mathbf{R}$ as

$$f(\mu) = \sum_{i=1}^{n} \alpha(x_i - \mu)_+ + (1 - \alpha)(x_i - \mu)_-,$$

where $\alpha \in (0, 1)$. Note that for $\alpha = 1/2$, this is $\|x - \mu\mathbf{1}\|_1/2$.

(a) Find the subdifferential of f at μ.

(b) Assume $0 \in \partial f(\mu^\star)$, *i.e.*, $\mu^\star$ minimizes f. Give an interpretation of $\mu^\star$. You may assume the values of x_i are distinct and ordered, *i.e.*, $x_1 < x_2 < \cdots < x_n$. *Hint.* See exercise title.

1.5 *Subgradient optimality conditions for nondifferentiable inequality constrained optimization.* Consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array}$$

with variable $x \in \mathbf{R}^n$. We *do not* assume that $f_0, \ldots, f_m$ are convex. Suppose that $\tilde{x}$ and $\tilde{\lambda} \succeq 0$ satisfy primal feasibility,

$$f_i(\tilde{x}) \leq 0, \quad i = 1, \ldots, m,$$

dual feasibility,

$$0 \in \partial f_0(\tilde{x}) + \sum_{i=1}^m \tilde{\lambda}_i \partial f_i(\tilde{x}),$$

and the complementarity condition

$$\tilde{\lambda}_i f_i(\tilde{x}) = 0, \quad i = 1, \ldots, m.$$

Show that $\tilde{x}$ is optimal, using only a simple argument, and definition of subgradient. Recall that we do not assume the functions $f_0, \ldots, f_m$ are convex.

1.6 *Optimality conditions for ℓ_1-regularized minimization.* Consider the problem of minimizing

$$\phi(x) = f(x) + \lambda \|x\|_1,$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and differentiable, and $\lambda \geq 0$. The number λ is the regularization parameter, and is used to control the trade-off between small f and small $\|x\|_1$. When ℓ_1-regularization is used as a heuristic for finding a sparse x for which $f(x)$ is small, λ controls (roughly) the trade-off between $f(x)$ and the cardinality (number of nonzero elements) of x.

Show that $x = 0$ is optimal for this problem (*i.e.*, minimizes ϕ) if and only if $\|\nabla f(0)\|_\infty \leq \lambda$. In particular, for $\lambda \geq \lambda^{\max} = \|\nabla f(0)\|_\infty$, ℓ_1 regularization yields the sparsest possible x, the zero vector.

Remark. The value $\lambda_{\max}$ gives a good reference point for choosing a value of the penalty parameter λ in ℓ_1-regularized minimization. A common choice is to start with $\lambda = \lambda^{\max}/2$, and then adjust λ to achieve the desired sparsity/fit trade-off. Useful values of λ typically range between $0.05\lambda^{\max}$ and $0.9\lambda^{\max}$.

1.7 *Coordinate-wise descent.* In the coordinate-wise descent method for minimizing a convex function f, we first minimize over x_1, keeping all other variables fixed; then we minimize over x_2, keeping all other variables fixed, and so on. After minimizing over x_n, we go back to x_1 and repeat the whole process, repeatedly cycling over all n variables. (There are many obvious variations on this, such as block coordinate-wise descent and random coordinate-wise descent.)

(a) Show that coordinate-wise descent fails for the function

$$f(x) = |x_1 - x_2| + 0.1(x_1 + x_2).$$

(In particular, verify that the algorithm terminates after one step at the point $(x_2^{(0)}, x_2^{(0)})$, while $\inf_x f(x) = -\infty$.) Thus, coordinate-wise descent need not work, for general convex functions.

(b) Now consider coordinate-wise descent for minimizing the specific function $\phi(x) = f(x) + \lambda\|x\|_1$, where f is smooth and convex, and $\lambda \geq 0$. Assuming f is strongly convex (say) it can be shown that the iterates converge to a fixed point $\tilde{x}$. Show that $\tilde{x}$ is optimal, *i.e.*, minimizes ϕ.

Thus, coordinate-wise descent works for ℓ_1-regularized minimization of a differentiable function.

(c) Work out an explicit form for coordinate-wise descent for ℓ_1-regularized least-squares, *i.e.*, for minimizing the function

$$\|Ax - b\|_2^2 + \lambda\|x\|_1.$$

You might find the deadzone function

$$\psi(u) = \begin{cases} u - 1 & u > 1 \\ 0 & |u| \leq 1 \\ u + 1 & u < -1 \end{cases}$$

useful. Generate some data and try out the coordinate-wise descent method. Check the result against the solution found using CVX, and produce a graph showing convergence of your coordinate-wise method.

1.8 *Regularization parameter for sparse Bayes network identification.* We are given samples $y_1, \ldots, y_N \in \mathbf{R}^n$ from an $\mathcal{N}(0, \Sigma)$ distribution, where $\Sigma \succ 0$ is an unknown covariance matrix. From these samples we will estimate the parameter Σ, using the prior knowledge that Σ^{-1} is sparse. (The diagonals will not be zero, so this means that many off-diagonal elements of Σ^{-1} are zero. Zero entries in Σ^{-1} can be interpreted as a conditional independence condition, and explains the title of this problem.)

To this end, we solve the (convex) problem

$$\text{maximize} \quad \log\det S - \mathbf{Tr}(SY) - \lambda \textstyle\sum_{i \neq j} |S_{ij}|$$

with variable $S \in \mathbf{S}^n_{++}$, which is our estimate of Σ^{-1}. Modulo a constant and scaling, the first two terms in the objective are the log-likelihood, where matrix Y is the sample covariance matrix

$$Y = \frac{1}{N}\sum_{k=1}^{N} y_k y_k^T$$

(which we assume satisfies $Y \succ 0$). The last term in the objective is a sparsfiying regularizer, with regularization parameter $\lambda > 0$. It does not penalize the diagonal terms in S, since they cannot be zero. We let $S^\star$ denote the optimal S (which is unique, since the objective is strictly concave). It depends on Y and λ.

(a) Suppose we add the additional constraint that S must be diagonal. (In this case S is as sparse as it can possibly be: All its off-diagonal entries are zero.) Find a simple expression for S^{diag}, the optimal S in this case.

(b) Show that there is a (finite) value λ^{diag}, such that $S^\star = S^{\text{diag}}$ if and only if $\lambda \geq \lambda^{\text{diag}}$. Find a simple expression for λ^{diag} in terms of Y.

Hint. See page 641 of the textbook for the derivative of $\log\det S$.

Remark. It is very useful in practice to know the value λ^{diag}. Useful values of the regularization parameter λ are almost always in the range $[0.05, .95]\lambda^{\text{diag}}$.

1.9 *Conjugacy and subgradients.* In this question, we show how conjugate functions are related to subgradients. Let f be convex and recall that its conjugate is $f^*(v) = \sup_x\{v^T x - f(x)\}$. Prove the following:

(a) For any v we have $v^T x \leq f(x) + f^*(v)$ (this is sometimes called Young's inequality).

(b) We have $g^T x = f(x) + f^*(g)$ if and only if $g \in \partial f(x)$.

Note that (you do not need to prove this) if f is closed, so that $f(x) = f^{**}(x)$, result (b) implies the duality relationship that $g \in \partial f(x)$ if and only if $x \in \partial f^*(g)$ if and only if $g^T x = f(x) + f^*(g)$.

1.10 If a function has a unique subgradient at a given point, is the function differentiable at that point? Provide a proof or a counter example.

1.11 In the following, determine the subdifferential set for the given functions at the specified points:

(a) $f(x_1, x_2, x_3) = \max\{|x_1|, |x_2|, |x_3|\}$ at $(x_1, x_2, x_3) = (0, 0, 0)$.

(b) $f(x) = e^{|x|}$ at $x = 0$ (x is a scalar).

(c) $f(x_1, x_2) = \max\{x_1 + x_2 - 1, x_1 - x_2 + 1\}$ at $(x_1, x_2) = (1, 1)$.

1.12 Consider the function $f : \mathbf{R}^2 \to \mathbf{R}$ given by

$$f(x_1, x_2) = \max\left\{\frac{1}{2}\|x\|^2 - x_1, \frac{1}{2}\|x\|^2 + x_1\right\}$$

(a) Determine the subdifferential set $\partial f(x)$ for $x \in \mathbf{R}^2$.

(b) Are the subgradients uniformly bounded over $x \in \mathbf{R}^2$? Would your answer change if x is restricted to lie in the set $X = \{x \in \mathbf{R}^2 \mid \|x\| \leq 1\}$? If yes, provide a bound for the subgradient norms.

1.13 *Does autodifferentiation work correctly?* Calculate a 'gradient' of the following functions using an automatic differentation (autodiff) method at the specified points. Check whether the result is a valid subgradient and give an explanation if there is a mismatch. You may use any programming language and any autodiff package.

(a) $f(x) = \max(x, 0)^2$ at $x = 0$

(b) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 0$

(c) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 10^{-50}$

(d) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 10^{-30}$

(e) $f(x) = \min(|x|, x)$ at $x = 0$

(f) $f(x) = \min(x, |x|)$ at $x = 0$

Hint: You can use Pytorch and Google Colab for autodiff (recommended)[1]*. Please see the following example which calculates the gradient of* $ReLU(x) = \max(x, 0)$ *at* $x = 0$*.*

```
import torch
x = torch.tensor([0.], requires_grad=True)
zero = torch.tensor([0.])
f = torch.max(x,zero)
f.backward()
print(x.grad) #prints the gradient of f with respect to x at its current value
```

1.14 *Computing gradients of expectations.* Suppose $z \in \mathbf{R}$ is a random variable with distribution $\mathbb{P}$ and let $F : \mathbf{R}^{d \times 1}$ be a collection of random functions indexed by z. Define the expected function to be $f(x) := \mathbf{E}_z[F(x, z)]$. We assume that $x \mapsto F(x, z)$ is convex for all z and that $\mathbf{E}_z[|F(x, z)|] < \infty$ for all x. Additionally, f and $x \mapsto F(x, z)$ are differentiable in x.

(a) (3 points) Verify that f is also a convex function.

(b) (3 points) Use the subgradient inequality to show that $\nabla_x f(x) = \mathbf{E}_z[\nabla_x F(x, z)]$. That is, we can exchange the order of differentiation and integration for convex functions.

[1] You can run your python script online on a Google Colaboratory notebook easily: colab.research.google.com

(c) (2 points extra credit) Assume $n = 1$. Abusing notation, the derivative of f is

$$\nabla_x f(x) = \lim_{t \to 0} \frac{f(x+t) - f(x)}{t}.$$

Part (b) shows that we can exchange the limit in this equation with the expectation operator when f is convex (convince yourself that this is true). However, the order of limits and integrals may not be exchanged in general.

Let $\{Y_k\}$ be a sequence of random variables with limit $\bar{Y}$. The Dominated Convergence Theorem[2] says that

$$\lim_{k \to \infty} \mathbf{E}[Y_k] = \mathbf{E}[\bar{Y}]$$

if $|Y_k| \leq |Z|$ for all k and a random variable Z satisfying $\mathbf{E}[|Z|] < \infty$. Show that Part (b) is consistent with measure-theoretic probability by proving that $\nabla_x f(x) = \mathbf{E}_z[\nabla_x F(x, z)]$ using the Dominated Convergence Theorem.

(d) (1 point extra credit) Now suppose that $n > 1$. Use Part (c) to prove $\nabla_x f(x) = \mathbf{E}_z[\nabla_x F(x, z)]$ by restricting f to a scalar function along e_i, a vector from the cardinal basis.

[2] See, for example, Cinlar, 2011 [**?**].

2 Generalized subgradients

2.1 *Non-convex non-differentiable functions, Clarke subdifferentials and Neural Networks.* Let $f : \mathbf{R}^n \to \mathbf{R}$ be a given function that we do not assume to be convex nor to be differentiable (e.g., a deep neural network with ReLU activation functions), so that the subdifferential $\partial f(x) = \{g \in \mathbf{R}^n \mid f(y) \geq f(x) + g^\top (y - x)\ \forall y\}$ is possibly an empty set. In this question, we explore generalized subdifferentials, or Clarke subdifferentials, as we have seen on page 11 of the lecture notes.

Let $D \subset \mathbf{R}^n$ be the set of points at which f is differentiable. We assume that D has (Lebesgue) measure 1, meaning that f is differentiable *almost everywhere*. The Clarke subdifferential of f at x is then defined as

$$\partial_C f(x) = \mathbf{Co}\left\{ \lim_{k\to\infty} \nabla f(x_k) \mid x_k \to x,\, x_k \in D \right\}.$$

The goal of this exercise is to characterize some basic properties of Clarke subdifferentials, relate $\partial_C f(x)$ to $\partial f(x)$ and study some implications of the condition $0 \in \partial_C f(x)$, which is necessary and sufficient for global optimality in the convex case.

We make the following technical assumption: we assume that f is locally Lipschitz, i.e., for any $x \in \mathbf{R}^n$, there exists $\eta > 0$ and $L_x > 0$ such that $|f(y) - f(z)| \leq L_x \|y - z\|_2$ for any y, z such that $\|x - y\|_2, \|x - z\|_2 \leq \eta$. Then, it follows that the function f is differentiable almost everywhere with respect to the Lebesgue measure (this result is sometimes referred to as Rademacher's theorem [**?**]).

Prove the following:

(a) If f is a continuously differentiable function then $\partial_C f(x) = \{\nabla f(x)\}$.

(b) If f is convex then $\partial_C f(x) \subseteq \partial f(x)$. Show that equality actually holds, i.e., $\partial_C f(x) = \partial f(x)$. *Hint: Suppose by contradiction that there exists $g \in \partial f(x)$ such that $g \notin \partial_C f(x)$. Set $h(x) = f(x) - g^T x$. Show that $0 \in \partial h(x)$ and $0 \notin \partial_C h(x)$. Use the hyperplane separation theorem to conclude.*

We say that x is *Clarke stationary* if $0 \in \partial_C f(x)$. If f is convex, then, from (b), we know that x is a global minimizer of f. For a non-convex function f, this property does not extend in general as we explore next.

(c) Suppose that x is a local minimum (resp. maximum) of f, i.e., there exists a radius $\eta > 0$ such that $f(y) \geq f(x)$ (resp. $f(y) \leq f(x)$) for any y such that $\|y - x\|_2 \leq \eta$. Show that x is Clarke stationary. *Hint: suppose by contradiction that $0 \notin \partial_C f(x)$ and conclude by using the hyperplane separating theorem with the convex sets $\partial_C f(x)$ and $\{0\}$.*

(d) Suppose that $\inf_x f(x) > -\infty$ and that $\inf_x f(x)$ is attained. Show that if x is the *unique* Clarke stationary point of f, then x is the unique global minimizer of f.

Finally, we study two examples of non-convex non-differentiable functions: a two-dimensional input function which has a unique Clarke stationary point that is the global minimizer, and, a neural network training loss which has a spurious Clarke stationary point at $(0, \ldots, 0)$.

(e) Consider the function with two-dimensional inputs $f(x_1, x_2) = 10\,|x_2 - x_1^2| + (1 - x_1)^2$. Show that the unique Clarke stationary point of f is $(x_1, x_2) = (1, 1)$ and that it is the unique global minimizer of f.

(f) Consider a supervised learning setting with a neural network parameterization: let $X \in \mathbf{R}^{n \times d}$ be a given data matrix and $y \in \mathbf{R}^n$ be a vector of real-valued observations. For the neural network parameters $u_1, \ldots, u_m \in \mathbf{R}^d$ and $\alpha_1, \ldots, \alpha_m \in \mathbf{R}$, consider the loss function

$$f(u_1, \ldots, u_m, \alpha_1, \ldots, \alpha_m) = \|y - \sum_{i=1}^{m} \sigma(X u_i)\alpha_i\|_2^2\,,$$

where we have introduced the component-wise ReLU activation function σ defined as $\sigma(z) = (\max\{z_1, 0\}, \ldots, \max\{z_n, 0\}) \in \mathbf{R}^n$ for $z = (z_1, \ldots, z_n) \in \mathbf{R}^n$. Show that $0 \in \partial f_C(0, \ldots, 0, 0, \ldots, 0)$.

3 Subgradient methods

3.1 *Minimizing a quadratic.* Consider the subgradient method with constant step size α, used to minimize the quadratic function $f(x) = (1/2)x^T Px + q^T x$, where $P \in \mathbf{S}^n_{++}$. For which values of α do we have $x^{(k)} \to x^\star$, for any $x^{(1)}$? What value of α gives fastest asymptotic convergence?

3.2 *A variation on alternating projections.* We consider the problem of finding a point in the intersection $\mathcal{C} \neq \emptyset$ of convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_m \subseteq \mathbf{R}^n$. To do this, we use alternating projections to find a point in the intersection of the two sets

$$\mathcal{C}_1 \times \cdots \times \mathcal{C}_m \subseteq \mathbf{R}^{mn}$$

and

$$\{(z_1, \ldots, z_m) \in \mathbf{R}^{mn} \mid z_1 = \cdots = z_m\} \subseteq \mathbf{R}^{mn}.$$

Show that alternating projections on these two sets is equivalent to the following iteration: project the current point in $\mathbf{R}^n$ onto each convex set (independently in parallel), and then average the results. Draw a simple picture to illustrate this.

3.3 *Matrix norm approximation.* We consider the problem of approximating a given matrix $B \in \mathbf{R}^{p\times q}$ as a linear combination of some other given matrices $A_i \in \mathbf{R}^{p\times q}$, $i = 1, \ldots, n$, as measured by the matrix norm (maximum singular value):

$$\text{minimize} \quad \|x_1 A_1 + \cdots + x_n A_n - B\|.$$

(a) Explain how to find a subgradient of the objective function at x.

(b) Generate a random instance of the problem with $n = 5$, $p = 3$, $q = 6$. Use CVX, CVXPY, or Convex.jl to find the optimal value $f^\star$ of the problem. Use a subgradient method to solve the problem, starting from $x = 0$. Plot $f - f^\star$ versus iteration. Experiment with several step size sequences.

3.4 *Asynchronous alternating projections.* We consider the problem of finding a sequence of points that approach the intersection $\mathcal{C} \neq \emptyset$ of some convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_m$. In the alternating projections method described in lecture, the current point is projected onto the farthest set. Now consider an algorithm where, at every step, the current point is projected onto any set not containing the point.

Give a simple example showing that such an algorithm can fail, *i.e.*, we can have $\mathbf{dist}(x^{(k)}, \mathcal{C}) \not\to 0$ as $k \to \infty$.

Now suppose that an additional hypothesis holds: there is an N such that, for each $i = 1, \ldots, m$, and each k, we have $x^{(j)} \in C_i$ for some $j \in \{k+1, \ldots, k+N\}$. In other words: in each block of N successive iterates of the algorithm, the iterates visit each of the sets. This would occur, for example, if in any block of N successive iterates, we project on each set at least once. As an example, we can cycle through the sets in

round-robin fashion, projecting at step k onto C_i with $i = k \bmod m + 1$. (When the point is in the set we are to project onto, nothing happens, of course.)

When this additional hypothesis holds, we have $\mathbf{dist}(x^{(k)}, \mathcal{C}) \to 0$ as $k \to \infty$. Roughly speaking, this means we can choose projections in any order, provided each set is taken into account every so often.

We give the general outline of the proof below; you fill in all details. Let's suppose the additional hypothesis holds, and let $x^\star$ be any point in the intersection $\mathcal{C} = \cap_{i=1}^m \mathcal{C}_i$.

(a) Show that
$$\|x^{(k+1)} - x^\star\|^2 \leq \|x^{(k)} - x^\star\|^2 - \|x^{(k+1)} - x^{(k)}\|^2.$$
This shows that $x^{(k+1)}$ is closer to $x^\star$ than $x^{(k)}$ is. Two more conclusions (needed later): The sequence $x^{(k)}$ is bounded, and $\mathbf{dist}(x^{(k)}, \mathcal{C})$ is decreasing.

(b) Iteratively apply the inequality above to show that
$$\sum_{i=1}^{\infty} \|x^{(i+1)} - x^{(i)}\|^2 \leq \|x^{(1)} - x^\star\|^2,$$
i.e., the distances of the projections carried out are square-summable. This implies that they converge to zero.

(c) Show that $\mathbf{dist}(x^{(k)}, \mathcal{C}_i) \to 0$ as $k \to \infty$. To do this, let $\epsilon > 0$. Pick M large enough that $\|x^{(k+1)} - x^{(k)}\| \leq \epsilon/N$ for all $k \geq M$. Then for all $k \geq M$ we have $\|x^{(k+j)} - x^{(k)}\| \leq j\epsilon/N$. Since we are guaranteed (by our hypothesis) that one of $x^{(k+1)}, \ldots, x^{(k+N)}$ is in $\mathcal{C}_i$, we have $\mathbf{dist}(x^{(k)}, \mathcal{C}_i) \leq \epsilon$. Since ϵ was arbitrary, we conclude $\mathbf{dist}(x^{(k)}, \mathcal{C}_i) \to 0$ as $k \to \infty$.

(d) It remains to show that $\mathbf{dist}(x^{(k)}, \mathcal{C}) \to 0$ as $k \to \infty$. Since the sequence $x^{(k)}$ is bounded, it has an accumulation point $\tilde{x}$. Let's take a subsequence $k_1 < k_2 < \cdots$ of indices for which $x^{(k_j)}$ converges to $\tilde{x}$ as $j \to \infty$. Since $\mathbf{dist}(x^{(k_j)}, \mathcal{C}_i)$ converges to zero, we conclude that $\mathbf{dist}(\tilde{x}, \mathcal{C}_i) = 0$, and therefore (since $\mathcal{C}_i$ is closed) $\tilde{x} \in \mathcal{C}_i$. So we've shown $\tilde{x} \in \cap_{i=1,\ldots,m} \mathcal{C}_i = \mathcal{C}$.

We just showed that along a subsequence of $x^{(k)}$, the distance to $\mathcal{C}$ converges to zero. But $\mathbf{dist}(x^{(k)}, \mathcal{C})$ is decreasing, so we conclude that $\mathbf{dist}(x^{(k)}, \mathcal{C}) \to 0$.

3.5 *Alternating projections to solve linear equations.* We can solve a set of linear equations expressed as
$$A_i x = b_i, \quad i = 1, \ldots, m,$$
where $A_i \in \mathbf{R}^{m_i \times n}$ are fat and full rank, using alternating projections on the sets
$$\mathcal{C}_i = \{z \mid A_i z = b_i\}, \quad i = 1, \ldots, m.$$
Assuming that the set of equations has solution set $\mathcal{C} \neq \emptyset$, we have $\mathbf{dist}(x^{(k)}, \mathcal{C}) \to 0$ as $k \to \infty$.

In view of exercise (3.4), the projections can be carried out cyclically, or asynchronously, provided we project onto each set every so often. As an alternative, we could project the current point onto all the sets, and form our next iterate as the average of these projected points, as in exercise (3.2).

Consider the case $m_i = 1$ and $m = n$ (*i.e.*, we have n scalar equations), and assume there is a unique solution $x^\star$ of the equations. Work out an explicit formula for the update, and show that the error $x^{(k)} - x^\star$ satisfies a (time-varying) linear dynamical system.

3.6 *Step sizes that guarantee moving closer to the optimal set.* Consider the subgradient method iteration $x^+ = x - \alpha g$, where $g \in \partial f(x)$. Show that if $\alpha < 2(f(x) - f^\star)/\|g\|_2^2$ (which is twice Polyak's optimal step size value) we have

$$\|x^+ - x^\star\|_2 < \|x - x^\star\|_2,$$

for any optimal point $x^\star$. This implies that $\mathbf{dist}(x^+, X^\star) < \mathbf{dist}(x, X^\star)$. (Methods in which successive iterates move closer to the optimal set are called *Féjer monotone*. Thus, the subgradient method, with Polyak's optimal step size, is Féjer monotone.)

3.7 *Subgradient method for inequality form SDP.* Describe how to implement a subgradient method to solve the inequality form SDP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x_1 A_1 + \cdots + x_n A_n \preceq B, \end{array}$$

with variable $x \in \mathbf{R}^n$, and problem data $c \in \mathbf{R}^n$, $A_1, \ldots, A_n \in \mathbf{S}^m$, $B \in \mathbf{S}^m$.

Generate a small instance of the problem (say, with $n = 4$ and $m = 10$) and solve it using your subgradient method. Check your solution using CVX.

3.8 *Alternating projections for LP feasibility.* We consider the problem of finding a point $x \in \mathbf{R}^n$ that satisfies $Ax = b$, $x \succeq 0$, where $A \in \mathbf{R}^{m \times n}$, with $m < n$.

(a) Work out alternating projections for this problem. (In other words, explain how to compute (Euclidean) projections onto $\{x \mid Ax = b\}$ and $\mathbf{R}^n_+$.)

(b) Implement your method, and try it on one or more problem instances with $m = 500$, $n = 2000$. With $x^{(k)}$ denoting the iterate after projection onto $\mathbf{R}^n_+$, plot $\|Ax^{(k)} - b\|_2$, the residual of the equality constraint. (This should converge to zero; you can terminate when this norm is smaller than 10^{-5}.)

Here is a simple way to generate data which is feasible. First generate a random A, and a random z with positive entries. Then set $b = Az$. (Of course, you cannot use z in your alternating projections method; you must start from some obvious point, such as 0.)

Warning. When A is a fat matrix, the Matlab command `A\b` does not do what you might expect, *i.e.*, compute a least-norm solution of $Ax = b$.

(c) *Factorization caching* is a general technique for speeding up some repeated calculations, such as projection onto an affine set. Assuming A is dense, the cost of computing the projection of a point onto the affine set $\{x \mid Ax = b\}$ is $O(m^2n)$ flops. (See Appendix C in *Convex Optimization*.) By saving some of the matrices involved in this computation, such as a Cholesky factorization (or even more directly, the inverse) of AA^T, subsequent projections can be carried out at a cost of $O(mn)$ flops, *i.e.*, m times faster. (There are several other ways to get this speedup, by saving other matrices.) Effectively, this makes each subgradient step (after the first one) a factor m times cheaper. Explain how to do this, and implement a caching scheme in your code. Verify that you obtain a speedup. (You may have to try your code on a larger problem instance.)

(d) *Over-projection.* A general method that can speed up alternating projections is to over-project, which means replacing the simple projection $x^+ = P(x)$ with $x^+ = x+\gamma(P(x)-x)$, where $\gamma \in [1, 2)$. (When $\gamma = 1$, this reduces to standard projection.) It is not hard to show that alternating projections, with over-projection, converges to a point in the intersection of the sets.

Implement over-projection and experiment with the over-projection factor γ, observing the effect on the number of iterations required for convergence.

3.9 *Subgradient method for total variation in-painting.* A grayscale image is represented as an $m \times n$ matrix of intensities U^{orig} (typically between the values 0 and 255). You are given the values U_{ij}^{orig}, for $(i, j) \in \mathcal{K}$, where $\mathcal{K} \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$ is the set of indices corresponding to known pixel values. Your job is to *in-paint* the image by guessing the missing pixel values, *i.e.*, those with indices not in $\mathcal{K}$. The reconstructed image will be represented by $U \in \mathbf{R}^{m\times n}$, where U matches the known pixels, *i.e.*, $U_{ij} = U_{ij}^{\text{orig}}$ for $(i, j) \in \mathcal{K}$.

The reconstruction U is found by minimizing the total variation of U, subject to matching the known pixel values. We will use the ℓ_2 total variation, defined as

$$\mathbf{tv}(U) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left\| \begin{bmatrix} U_{i+1,j} - U_{i,j} \\ U_{i,j+1} - U_{i,j} \end{bmatrix} \right\|_2 .$$

Note that the norm of the discretized gradient is *not* squared.

(a) Explain how to find a subgradient $G \in \partial\, \mathbf{tv}(U)$. It is sufficient to give a formula for G_{ij}.

(b) Implement a projected subgradient method for minimizing $\mathbf{tv}(U)$ subject to $U_{ij} = U_{ij}^{\text{orig}}$ for $(i, j) \in \mathcal{K}$.

Use it to solve the problem instance given in `subgrad_tv_inpaint_data.m`. You will also need `tv_l2_subgrad.m`, `lena512.bmp`, and `lena512_corrupted.bmp`. Show the original image, the corrupted image, and the in-painted image. Plot $\mathbf{tv}\left(U^{(k)}\right)$ ($U^{(k)}$ is U in the kth iteration) versus k.

The file `subgrad_tv_inpaint_data.m` defines `m`, `n`, and matrices `Uorig`, `Ucorrupt`, and `Known`. The matrix `Ucorrupt` is `Uorig` with the unknown pixels whited out. The matrix `Known` is $m \times n$, with (i, j) entry one if $(i, j) \in \mathcal{K}$ and zero otherwise. The file also includes code to display `Uorig` and `Ucorrupt` as images.

Writing matlab code that operates quickly on large image matrices is tricky, so we have provided a function `tv_l2_subgrad.m` that computes $\mathbf{tv}(U)$ and $G \in \partial\,\mathbf{tv}(U)$ given U. `tv_l2_subgrad.m` uses the `norms` function from `CVX`, so you will need `CVX` installed. A simple (and fast) way to set the known entries of a matrix `U` to their known values is `U(Known == 1) = Uorig(Known == 1)`.

You may need to try several step length sequences to get fast enough convergence. We obtained good results with step sizes like $\alpha_k = 1000/k$ and $\alpha_k = 50/\sqrt{k}$, but feel free to experiment with others. Do not hesitate to run the algorithm for 1000 or more iterations.

Once it's working, you might like to create an animated GIF that shows algorithm progress, say, displaying U every 50 iterations. We used the function `imwrite(U_record,'inpaint.gif','DelayTime',1,'LoopCount',inf)`. Here `U_record` is an $m \times n \times 1 \times r$ matrix, where `U_record(:, :, 1, i)` is the ith intermediate value of U out of the r stored in `U_record`. `imwrite` will project invalid intensity values into the range $[0, 255]$ (with a warning).

3.10 *Strong convexity and smoothness.* We say that a function f is λ-strongly convex over the set C if $\mathbf{dom}\, f \supset C$ and for all $x, y \in C$ and $g \in \partial f(x)$,

$$f(y) \geq f(x) + g^T(y - x) + \frac{\lambda}{2} \|x - y\|_2^2 .$$

That is, the function grows at least quadratically everywhere over C. In a duality relationship you explore in this problem, we say that a function h has L-Lipschitz gradient over the set C if for all $x, y \in C$

$$\|\nabla h(x) - \nabla h(y)\|_2 \leq L \|x - y\|_2 .$$

By a Taylor expansion, this is equivalent to

$$h(y) \leq h(x) + \nabla h(x)^T(y - x) + \frac{L}{2} \|x - y\|_2^2 .$$

(You do not need to prove this equivalence.)

Hint: For this question, use the results of question 1.9.

(a) Let f be λ-strongly convex over a closed convex set C. Letting $I_C(x) = 0$ for $x \in C$ and $+\infty$ otherwise, show that the conjugate $(f + I_C)^*$ is differentiable and has $(1/\lambda)$-Lipschitz continuous gradient.

(b) Show that (if f is λ-strongly convex)

$$\nabla (f + I_C)^*(s) = \operatorname*{argmin}_{x \in C} \left\{ -s^T x + f(x) \right\} .$$

(c) Let f have L-Lipschitz gradient. Show that f^* is $(1/L)$-strongly convex.

As an aside, part (b) shows that the solutions to strongly convex problems are Lipschitz continuous under perturbations of the objective.

3.11 *More general dualities of strong convexity and smoothness.* This question generalizes the results of Ex. 3.10 to non-Euclidean norms and more general geometries. Let $\|\cdot\|$ be a norm on a set $C \subset \mathbf{R}^n$. A function f is λ-strongly convex over the set C with respect to the norm $\|\cdot\|$ if $\mathbf{dom}\, f \supset C$ and for all $x, y \in C$ and $g \in \partial f(x)$,

$$f(y) \geq f(x) + g^T(y - x) + \frac{\lambda}{2}\|x - y\|^2.$$

Let $\|\cdot\|_*$ be the dual norm for $\|\cdot\|$, that is, $\|z\|_* = \sup_{x:\|x\|\leq 1} x^T z$. We say that a function h has L-Lipschitz gradient over the set C with respect to the norm $\|\cdot\|$ if for all $x, y \in C$

$$\|\nabla h(x) - \nabla h(y)\|_* \leq L\|x - y\|.$$

Hint: For this question, use the results of question 1.9.

(a) Let f be λ-strongly convex over a closed convex set C with respect to the norm $\|\cdot\|$. Letting $I_C(x) = 0$ for $x \in C$ and $+\infty$ otherwise, show that the conjugate $(f + I_C)^*$ is differentiable and has $(1/\lambda)$-Lipschitz continuous gradient with respect to the norm $\|\cdot\|_*$.

(b) Let f have L-Lipschitz gradient with respect to the norm $\|\cdot\|$. Show that f^* is $(1/L)$-strongly convex with respect to the dual norm $\|\cdot\|_*$.

3.12 *Mirror descent and adaptive stepsizes.* The mirror descent method iterates

$$g^k \in \partial f(x^k), \quad x^{k+1} = \operatorname*{argmin}_{x \in C} \left\{ \langle g^k, x \rangle + \frac{1}{\alpha_k} D_h(x, x^k) \right\},$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is a convex function. Here $D_h(x, y) = h(x) - h(y) - \nabla h(y)^T(x - y)$ is the Bregman divergence based on the function h, where we assume that h is strongly convex with respect to the norm $\|\cdot\|$ with dual norm $\|\cdot\|_*$. Mirror descent satisfies the following convergence bound: for any $x^\star \in C$, if $D_h(x^\star, x) \leq \frac{1}{2}R^2$ for all $x \in C$, then for every non-increasing stepsize sequence $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$,

$$\sum_{i=1}^k \left[f(x^i) - f(x^\star) \right] \leq \frac{1}{2\alpha_k} R^2 + \sum_{i=1}^k \frac{\alpha_i}{2} \|g^i\|_*^2.$$

We investigate dynamic choices of the stepsize α_k to get reasonable convergence behavior.

(a) Show that for a fixed stepsize α,

$$\inf_{\alpha \geq 0} \left\{ \frac{1}{2\alpha} R^2 + \sum_{i=1}^{k} \frac{\alpha}{2} \|g^i\|_*^2 \right\} = R \left(\sum_{i=1}^{k} \|g^i\|_*^2 \right)^{\frac{1}{2}}.$$

(b) Instead of using a fixed stepsize, suppose at each step k we choose α_k to minimize the current bound:

$$\alpha_k = \operatorname*{argmin}_{\alpha \geq 0} \left\{ \frac{1}{2\alpha} R^2 + \sum_{i=1}^{k} \frac{\alpha}{2} \|g^i\|_*^2 \right\}.$$

Show that with this stepsize sequence, the mirror descent iterates satisfy

$$\sum_{i=1}^{k} \left[f(x^i) - f(x^\star) \right] \leq \frac{3}{2} R \left(\sum_{i=1}^{k} \left\| g^i \right\|_*^2 \right)^{\frac{1}{2}}.$$

(c) Suppose that $\|g^i\|_* \leq G$ for all i. What convergence rate on $f(\bar{x}^k) - f(x^\star)$ does this guarantee for the choice $\bar{x}^k = \frac{1}{k} \sum_{i=1}^{k} x^i$?

3.13 *High dimensional problems, mirror descent, and gradient descent.* We consider using mirror descent versus projected subgradient descent to solve the non-smooth minimization problem

$$\text{minimize } f(x) = \max_{i \in \{1,\ldots,m\}} \{a_i^T x + b_i\} \text{ subject to } x \in \Delta_n = \{z \in \mathbf{R}_+^n \mid z^T \mathbf{1} = 1\}.$$

Implement mirror descent with the choice $h(x) = \sum_{i=1}^{n} x_i \log x_i$ and projected subgradient descent for this problem. (You will need to project onto the simplex efficiently for this to be a reasonable method at all.) You will compare the performance of these two methods.

Generate random problem data for the above objective with a_i drawn as i.i.d. $N(0, I_{n \times n})$ (multivariate normals) and b_i drawn i.i.d. $N(0, 1)$, where $n = 500$ and $m = 50$. Solve the problem using CVX (or `Convex.jl` or CVXPY), then run mirror descent and projected gradient descent on the same data for 100 iterations. Run each method with constant stepsizes $\alpha \in \{2^{-12}, 2^{-11}, \ldots, 2^6, 2^7\}$. Repeat this 25 times, then plot the average optimality gap $f(x^k) - f(x^\star)$ or $f_{\text{best}}^k - f(x^\star)$ as a function of iteration for the best stepsize (chosen by smallest optimality gaps) for each method. Which method gives the best performance?

3.14 *Subgradient methods for Lasso.* Consider the optimization problem

$$\text{minimize} \quad f(x) := \tfrac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1 ,$$

with variables $x \in \mathbf{R}^n$ and problem data $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ and $\lambda > 0$. This model is known as Lasso, or Least Squares with ℓ_1 regularization, which encourages sparsity in the solution via the non-smooth penalty $\|x\|_1 := \sum_{j=1}^{n} |x_j|$. In this problem, we will explore various subgradient methods for fitting this model.

(a) (1 points) Derive the subdifferential $\partial f(x)$ of the objective.

(b) (1 points) Find the update rule of the subgradient method and state the computational complexity of applying one update using big O notation in terms of the dimensions.

(c) (5 points) Let $n = 1000$, $m = 200$ and $\lambda = 0.01$. Generate a random matrix $A \in \mathbf{R}^{m \times n}$ with independent Gaussian entries with mean 0 and variance $1/m$, and a fixed vector $x^* = \big[\underbrace{1, ..., 1}_{k \text{ times}}, \underbrace{0, ..., 0}_{n-k \text{ times}}\big]^T \in \mathbf{R}^n$. Let $k = 5$ and then set $b = Ax^*$. Implement the subgradient method to minimize $f(x)$, initialized at the all-zeros vector. Try different step size rules, including constant step size, constant step length, $1/\sqrt{k}$, $1/k$, Polyak's step length with estimated objective value as shown in lecture slides. Plot objective value versus iteration curves of different step size rules on the same figure.

(d) (3 points) Repeat part (c) using a heavy ball term, $\beta_k(x^k - x^{k-1})$, added to the subgradient, as described on page 25 of lecture slides. Try different step size rules as in part (c) and tune the heavy ball parameter $\beta_k = \beta$ for faster convergence.

3.15 *Line-search for Non-smooth Functions.* In this question, we will examine the feasibility of line-search for choosing the step-size in subgradient descent. Let $f : \mathbf{R}^n \to \mathbf{R}$ be a convex function. At iteration k of subgradient descent, the Armijo line-search selects the *largest* step-size $\alpha_k > 0$ which satisfies

$$f(x_k - \alpha_k g_k) \leq f(x_k) - c\alpha_k \|g_k\|_2^2, \tag{1}$$

where $g_k \in \partial f(x_k)$ and $c \in (0, 1)$ is a relaxation parameter. In practice, this can be achieved by reducing the step-size as $\alpha_k \leftarrow \beta\alpha_k$ for some $\beta \in (0, 1)$ until (1) is satisfied. This is called backtracking.

We will analyze the performance of this backtracking line-search procedure for the following piece-wise linear function.

$$f(x) = \begin{cases} -2x & \text{if } x \leq 0 \\ -\frac{1}{2}x & \text{if } x \in (0, 4) \\ x - 6 & \text{if } x \geq 4. \end{cases}$$

(a) Plot f over the domain $[-2, 6]$ in your favorite plotting software and report the figure. Is f a convex function? Report the minimizer(s) of f.

(b) Since f is piece-wise linear with a finite number of pieces, its subdifferential takes only a finite number of distinct set values. Report each unique subdifferential set of f and the interval over which it is valid.

(c) Suppose we attempt to minimize f using subgradient descent with the Armijo line-search. In particular, suppose that we choose a random subgradient at each iteration and backtrack on α_k until (1) holds.

Suppose $c > 0.25$ and show that there exists an initial point $x_0 \in [-2, 6]$, $x_0 \notin \operatorname{argmin}_x f(x)$ and subgradient $g_0 \in \partial f(x_0)$ such that no step-size $\alpha_0 > 0$ exists for which the Armijo condition holds.

(d) Now let $c \in (0, 1)$. Modify f using knowledge of c to show that there exists a function for which the line-search fails analogously to part (c). As in part (c), enforce $x_0 \notin \operatorname{argmin}_x f(x)$.

3.16 *Finding a point in the intersection of convex sets.* Let Σ be an $n \times n$ diagonal matrix with diagonal entries $\sigma_1 \geq \cdots \geq \sigma_n > 0$, and y a given vector in $\mathbf{R}^n$. Consider the compact convex sets $\mathcal{E} = \{z \in \mathbf{R}^n \mid \|\Sigma^{\frac{1}{2}} z\|_2 \leq 1\}$ and $B = \{z \in \mathbf{R}^n \mid \|z - y\|_\infty \leq 1\}$.

(a) (2 points) Formulate an optimization problem and propose an algorithm in order to find a point $x \in \mathcal{E} \cap B$. *You can assume that $\mathcal{E} \cap B$ is not empty.* Your algorithm must be provably converging (although you do not need to prove it and you can simply refer to the lecture slides).

(b) (2 points) Implement your algorithm with the following data: $n = 2$, $y = (7/4, 0)$, $\sigma_1 = 1$, $\sigma_2 = 0.5$ and $x = (0, 4)$. Plot the objective value of your optimization problem versus the number of iterations.

3.17 *Constrained subgradient method.* Consider the optimization problem

$$\begin{array}{ll} \text{minimize}_{\{x_j\}_{j=1}^J} & f(x_1, \ldots, x_J) := \frac{1}{2}\|b - \sum_{j=1}^J A_j x_j\|_2^2 + \lambda \cdot \sum_{j=1}^J \|x_j\|_2 \,, \\ & \text{s.t. } A_j x_j \geq 0, \, \forall j \in \{1, 2, \ldots, J\} \end{array}$$

with variable $x_1, \ldots, x_J \in \mathbf{R}^n$ and problem data $A_1, \ldots, A_J \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ and $\lambda > 0$. We will apply the subgradient method for constrained optimization given on page 11 of the lecture slides.

Let $J = 3$, $n = 100$, $m = 10$, and $\lambda = .5$. Generate random matrices $A_1, \ldots, A_J \in \mathbf{R}^{m \times n}$ with independent uniformly distributed entries in the interval $\left[0, \frac{1}{\sqrt{m}}\right)$ and, random vectors $x_1, \ldots, x_J \in \mathbf{R}^n$ with independent uniformly distributed entries in the interval $\left[0, \frac{1}{\sqrt{n}}\right)$, then set $b = \sum_{j=1}^J A_j x_j$. Plot convergence in terms of the objective $f(x_1^{(k)}, \ldots, x_J^{(k)})$. Try different step length schedules. Also, plot the maximal violation for the linear constraints at each step.

3.18 *Recovering Discrete Signals via Convex Optimization.* Suppose that x is an n dimensional signal taking values only in $\{-1, +1\}$, i.e., $x \in \{-1, +1\}^n$, and we have observations $y = Ax$. Here, $A \in \mathbb{R}^{m \times n}$ is a matrix whose entries are known. This setting is frequently encountered in wireless communication systems. Typically, the signal x carries digital information and A models the propagation of the signal over a wireless channel. You will try recovering the signal by finding a point $\hat{x}$ that satisfies $\|\hat{x}\|_\infty \leq 1$ and $A\hat{x} = y$. Generate a random matrix A with independent standard Gaussian entries and random signal $x \in \{-1, +1\}^n$ with independent uniformly distributed values in $\{-1, +1\}$ and let $y = Ax$.

(a) Formulate an optimization problem and propose an algorithm to recover a signal from measurements $y = Ax$ obeying the constraint $\|x\|_\infty \leq 1$.

(b) Plot the convergence of the algorithm in part (a) in terms of the Euclidean distance $\|\hat{x} - x\|_2$ for $n = 100$ and $m \in 50, 80, 90$. Plot the original and recovered signals.

4 Stochastic subgradient methods

4.1 *Minimizing expected maximum violation.* We consider the problem of minimizing the expected maximum violation of a set of linear constraints subject to a norm bound on the variable,

$$\begin{array}{ll} \text{minimize} & \mathbf{E}\max(b - Ax)_+ \\ \text{subject to} & \|x\|_\infty \leq 1, \end{array}$$

where the data $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$ are random.

We consider a specific problem instance with $m = 3$ and $n = 3$. The entries of A and b vary uniformly (and independently) ± 0.1 around their expected values,

$$\mathbf{E}\, A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1/2 & 0 \\ 1 & 1 & 1/2 \end{bmatrix}, \qquad \mathbf{E}\, b = \begin{bmatrix} 9/10 \\ 1 \\ 9/10 \end{bmatrix}.$$

(a) *Solution via stochastic subgradient.* Use a stochastic subgradient method with step size $1/k$ to compute a solution x^{stoch}, starting from $x = 0$, with $M = 1$ subgradient sample per iteration. Run the algorithm for 5000 iterations. Estimate the objective value obtained by x^{stoch} using Monte Carlo, with $M = 1000$ samples. Plot the distribution of $\max(b - Ax^{\text{stoch}})$ from these samples. (In this plot, points to the left of 0 correspond to no violation of the inequalities.)

(b) *Maximum margin heuristic.* The heuristic x^{mm} is obtained by maximizing the margin in the inequalities, with the coefficients set to their expected values:

$$\begin{array}{ll} \text{minimize} & \max(\mathbf{E}\, b - \mathbf{E}\, Ax) \\ \text{subject to} & \|x\|_\infty \leq 1. \end{array}$$

Use Monte Carlo with $M = 1000$ samples to estimate the objective value (for the original problem) obtained by x^{mm}, and plot the distribution of $\max(b - Ax^{\text{mm}})$.

(c) *Direct solution of sampled problem.* Generate $M = 100$ samples of A and b, and solve the problem

$$\begin{array}{ll} \text{minimize} & (1/M)\sum_{i=1}^M \max(b^i - A^i x)_+ \\ \text{subject to} & \|x\|_\infty \leq 1. \end{array}$$

The solution will be denoted x^{ds}. Use Monte Carlo with $M = 1000$ samples to estimate the objective value (for the original problem) obtained by x^{ds}, and plot the distribution of $\max(b - Ax^{\text{ds}})$.

Hints.

- Use `x = max( min(x,1), -1 )` to project onto the ℓ_∞ norm ball.
- Use the CVX function `pos()` to get the positive part function $()_+$.

- The clearest code for carrying out Monte Carlo analysis uses a `for` loop. In Matlab `for` loops can be very slow, since they are interpreted. Our for-loop implementation of the solution to this problem isn't too slow, but if you find Monte Carlo runs slow on your machine, you can use the loop-free method shown below, to find the empirical distribution of $\max(b - Ax)$.

```
% loop-free Monte Carlo with 1000 samples of data A and b
M = 1000; noise = 0.1;
Amtx = repmat(Abar,M,1) + 2*noise*rand(M*m,n) - noise;
bmtx = repmat(bbar,M,1) + 2*noise*rand(M*m,1) - noise;
% evaluate max( b - Ax ) for 1000 samples
fvals_stoch = max( reshape(bmtx - Amtx*x_stoch,m,M) );
```

4.2 *Support vector machine training via stochastic subgradient.* We suppose that feature-label pairs, $(x, y) \in \mathbf{R}^n \times \{-1, 1\}$, are generated from some distribution. We seek a linear classifier or predictor, of the form $\hat{y} = \mathbf{sign}(w^T x)$, where $w \in \mathbf{R}^n$ is the weight vector. (We can add an entry to x that is always 1 to get an affine classifier.) Our classifier is correct when $yw^T x > 0$; since this expression is homogeneous in w, we can write this as $yw^T x \geq 1$. Thus, our goal is to choose w so that $1 - yw^T x \leq 0$ with high probability.

A *support vector machine* (SVM) chooses w^{svm} as the minimizer of

$$f(w) = \mathbf{E}\left(1 - yw^T x\right)_+ + (\rho/2)\|w\|_2^2,$$

where $\rho > 0$ is a parameter. The first term is the *average loss*, and the second term is a quadratic regularizer. Finding w^{svm} involves solving a stochastic optimization problem.

Explain how to (approximately) solve this stochastic optimization problem using the stochastic subgradient method, with one sample per subgradient step. In this context, the samples from the distribution are called data or examples, and the collection of these is called the training data. Since this method only processes one data sample in each step, it is called a *streaming* algorithm (since it does not have to store more than one data sample in each step).

Implement the stochastic subgradient method for a problem with $n = 20$, and (x, y) samples generated using

```
randn('state',0)
w_true = randn(n,1); % 'true' weight vector
% to get each data sample use snippet below
x = randn(n,1);
y = sign(w_true'*x+0.1*randn(1));
```

Experiment with the choice of ρ, the step size rule, and the number of iterations to run (but don't be afraid to run the algorithm for 10000 steps).

To view the convergence, you can plot two quantities at each step: the optimality gap $f(w) - f^\star$ and the classifier error probability $\mathbf{Prob}\left(yw^T x \leq 0\right)$. To (approximately) compute these quantities, use a Monte Carlo method, using, say, 10000 samples. (You'll want to compute these 10000 samples, and evaluate the Monte Carlo estimates of the two quantities above, without using Matlab for loops. Also note that evaluation of these two quantities will be far more costly than each step of the stochastic subgradient method.) You can use CVX to estimate $f^\star$.

4.3 *Log-optimal portfolio optimization using return oracle.* We consider the portfolio optimization problem

$$\begin{array}{ll} \text{maximize} & \mathbf{E}_r \log(r^T x) \\ \text{subject to} & \mathbf{1}^T x = 1, \qquad x \succeq 0, \end{array}$$

with variable (portfolio weights) $x \in \mathbf{R}^n$. The expectation is over the distribution of the (total) return vector $r \in \mathbf{R}^n_{++}$, which is a random variable. (Although not relevant in this problem, the log-optimal portfolio maximizes the long-term growth of an initial investment, assuming the investments are re-balanced to the log-optimal portfolio after each investment period, and ignoring transaction costs.)

In this problem we assume that we do not know the distribution of r (other than that we have $r \succ 0$ almost surely). However, we have access to an oracle that will generate independent samples from the return distribution. (Although not relevant, these samples could come from historical data, or stochastic simulations, or a known or assumed distribution.)

(a) Explain how to use the (projected) stochastic subgradient method, using one return sample for each iteration, to find (in the limit) a log-optimal portfolio. Describe how to carry out the projection required, and how to update the portfolio in each iteration.

(b) Implement the method and run it on the problem with $n = 10$ assets, with return sample oracle in the mfile `log_opt_return_sample(m)`. This function returns an $n \times m$ matrix whose columns are independent return samples.

You are welcome to look inside this file to see how we are generating the sample. The distribution is a mixture of two log-normal distributions; you can think of one as the standard return model and the other as the return model in some abnormal regime. However, your stochastic subgradient algorithm can only call `log_opt_return_sample(1)`, once per iteration; you cannot use any information found inside the file in your implementation.

To get a Monte Carlo approximation of the objective function value, you can generate a block of, say, 10^5 samples (using `R_emp=log_opt_return_sample(1e5)`, which only needs to be done once), and then use `obj_hat = mean(log(R_emp'*x))` as your estimate of the objective function. Plot the (approximate) objective value versus iteration, as well as the best approximate objective value obtained up to

that iteration. (Note that evaluating the objective will require far more computation than each stochastic subgradient step.)

You may need to play around with the step size selection in your method to get reasonable convergence. Remember that your objective value evaluation is only an approximation.

4.4 *A subgradient method using averaged gradients.* An alternate method to (sub)gradient descent for optimization is known as *dual averaging*, as it averages gradients (points in the dual space). The method applies to quite general problems, and in this problem, we show how, for any sequence of convex functions $f_i : \mathbf{R}^n \to \mathbf{R}$ we can give bounds on the "online regret" of

$$\sum_{i=1}^{k} \left[f_i(x^i) - f_i(x^\star) \right]$$

over all fixed $x^\star \in C$, where $C \subset \mathbf{R}^n$ is closed and convex.

Let $x_0 \in C$ be an arbitary point and $\alpha_k > 0$ be a non-increasing sequence of stepsizes. Then we iterate the following scheme:

$$\begin{aligned} &\text{choose } g^k \in \partial f_k(x^k) \\ &\text{update } z^k = \sum_{i=1}^{k} g_i \\ &\text{solve } x^{k+1} = \operatorname*{argmin}_{x \in C} \left\{ \langle z^k, x \rangle + \frac{1}{2\alpha_k} \left\| x - x_0 \right\|_2^2 \right\}. \end{aligned}$$

This method is identical to gradient descent with a fixed stepsize if we have $C = \mathbf{R}^n$, $\alpha_k = \alpha$ for all k. More generally, it builds a somewhat global linear approximation to the function, which it regularizes with the squared Euclidean distance. For the remainder of the problem, assume (without loss of generality) that $0 \in C$ and $x_0 = 0$.

(a) Let $h_k(x) = \frac{1}{2\alpha_k} \left\| x \right\|_2^2$ and define

$$h_k^*(z) = \sup_{x \in C} \left\{ z^T x - h_k(x) \right\} = \sup_{x \in C} \left\{ z^T x - \frac{1}{2\alpha_k} \left\| x \right\|_2^2 \right\}.$$

Argue that h_k^* is differentiable and has α_k-Lipschitz gradient, so for any $z_1, z_2 \in \mathbf{R}^n$

$$\left\| \nabla h_k^*(z_1) - \nabla h_k^*(z_2) \right\|_2 \leq \alpha_k \left\| z_1 - z_2 \right\|_2.$$

(b) Argue that $\nabla h_k^*(-z^k) = x^{k+1} = \operatorname{argmin}_{x \in C} \{ \langle z^k, x \rangle + \frac{1}{2\alpha_k} \left\| x \right\|_2^2 \}$.

(c) Let $x^\star \in C$ be arbitrary. Provide justification for each of the lettered steps in the chain of inequalities below.

$$\begin{aligned}
\sum_{i=1}^k f_i(x^i) - f_i(x^\star) &\overset{(i)}{\leq} \sum_{i=1}^k \left\langle g^i, x^i - x^\star \right\rangle = \sum_{i=1}^k \left\langle g^i, x^i \right\rangle - \left\langle z^k, x^\star \right\rangle \\
&= \sum_{i=1}^k \left\langle g^i, x^i \right\rangle - \left\langle z^k, x^\star \right\rangle + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 - \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 \\
&\overset{(ii)}{\leq} \sum_{i=1}^k \left\langle g^i, x^i \right\rangle + h_k^*(-z^k) + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 \\
&\overset{(iii)}{\leq} \sum_{i=1}^k \left\langle g^i, x^i \right\rangle + h_{k-1}^*(-z^k) + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 \\
&\overset{(iv)}{\leq} \sum_{i=1}^k \left\langle g^i, x^i \right\rangle + h_{k-1}^*(-z^{k-1}) + \left\langle \nabla h_{k-1}^*(-z^{k-1}), -g^k \right\rangle + \frac{\alpha_{k-1}}{2} \left\|z^{k-1} - z^k\right\|_2^2 + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 \\
&\overset{(v)}{=} \sum_{i=1}^{k-1} \left\langle g^i, x^i \right\rangle + h_{k-1}^*(-z^{k-1}) + \frac{\alpha_{k-1}}{2} \left\|g^k\right\|_2^2 + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 .
\end{aligned}$$

Hint: The following inequality, which follows from a Taylor expansion, may be useful. If F is a function with L-Lipschitz continuous gradient, then

$$F(y) \leq F(x) + \langle \nabla F(x), y - x \rangle + \frac{L}{2} \left\|x - y\right\|_2^2 .$$

(d) Using the preceding chain of inequalities and that $h_0^*(0) = 0$ (why), prove that for *any* sequence of convex functions f_i and any $x^\star \in C$, the dual averaging procedure gives

$$\sum_{i=1}^k \left[f_i(x^i) - f_i(x^\star)\right] \leq \sum_{i=1}^k \frac{\alpha_{i-1}}{2} \left\|g^i\right\|_2^2 + \frac{1}{2\alpha_k} \left\|x^\star\right\|_2^2 .$$

(e) *A stochastic result.* Suppose that ω_i are drawn i.i.d. according to a distribution $\mathbf{P}$, and that $f_i(x) = F(x, \omega_i)$, where $F(x, \omega_i)$ is convex in x for all ω, and let $f(x) = \mathbf{E}[F(x, \omega)]$. Show that if $\bar{x}^k = \frac{1}{k} \sum_{i=1}^k x_i$, then

$$\mathbf{E}[f(\bar{x}^k)] - f(x^\star) \leq \frac{1}{2k\alpha_k} \left\|x^\star\right\|_2 + \frac{1}{k} \sum_{i=1}^k \frac{\alpha_{i-1}}{2} \mathbf{E}\left[\left\|g^i\right\|_2^2\right] .$$

4.5 *Dual averaging versus gradient descent.* In this question, you will use dual averaging (Ex. 4.4) and stochastic subgradient descent to solve a support vector machine problem (see Ex. 4.2). The objective is of the form

$$f(x) = \frac{1}{N} \sum_{i=1}^N \max\{1 - a_i^T x, 0\}.$$

We consider using the dual averaging method of Ex. 4.4 to minimize the function f over the probability simplex in $\mathbf{R}^n$, *i.e.* $\Delta_n = \{x \in \mathbf{R}^n_+ \mid \mathbf{1}^T x = 1\}$. This constraint set is a heuristic for finding a sparse solution $x^\star$, i.e. one with many zero entries.

(a) Let z be an arbitrary vector in $\mathbf{R}^n$. Show how to compute the proximal operator

$$x(z) = \operatorname*{argmin}_{x \in \Delta_n} \left\{ z^T x + \frac{1}{2} \|x\|_2^2 \right\}.$$

Equivalently, show how to (for any $v \in \mathbf{R}^n$) compute the projection

$$\Pi_{\Delta_n}(v) = \operatorname*{argmin}_{x \in \Delta_n} \left\{ \|x - v\|_2^2 \right\}.$$

(b) Implement projected stochastic gradient descent and stochastic dual averaging. That is, implement a method that at each iteration, draws a single sample of the a_i vectors, computes an associated stochastic subgradient g^k, and then performs one of the following two updates:

$$\text{SGD} \quad x^{k+1} = \operatorname*{argmin}_{x \in \Delta_n} \left\{ \left\| x - (x^k - \alpha_k g^k) \right\|_2^2 \right\}$$

$$\text{Dual averaging} \quad x^{k+1} = \operatorname*{argmin}_{x \in \Delta_n} \left\{ \left\langle z^k, x \right\rangle + \frac{1}{2\alpha_k} \|x\|_2^2 \right\},$$

where $z^k = \sum_{i=1}^k g^i$. For stochastic gradient descent, use the stepsizes $\alpha_k = 1/\sqrt{nk}$, and for stochastic dual averaging use stepsizes $\alpha_k = 1/\sqrt{k}$.
Use the data in `dual_averaging_data.jl` (for `julia` code) or `dual_averaging_data.m` (for Matlab) to generate the matrix $A = [a_1 \ \cdots \ a_N]^T \in \mathbf{R}^{N \times n}$. Run each method for 200 steps, and give two plots: one with the gaps $f(x^k) - f(x^\star)$ as a function of iteration k for each of the methods, the other with the number of non-zero entries in x^k as a function of k. (If your projection does not produce exact zeros, truncate any coordinates with $|x_j| < 10^{-5}$ to zero.) You should see that one of the two methods results in far fewer non-zero entries than the other.

4.6 *Stochastic Kaczmarz method for linear systems.* Consider the Least Squares minimization problem

$$\begin{array}{ll} \text{minimize} & \underbrace{\dfrac{1}{2m} \displaystyle\sum_{i=1}^m (b_i - a_i^T x)^2}_{f(x)}, \\ \text{subject to} & x \in \mathbf{R}^n \end{array}$$

where $a_1, ..., a_m$ are the rows of a data matrix A. We will consider the stochastic subgradient descent iterates

$$x^{t+1} = x^t - \alpha_t g_t, \tag{2}$$

where g_t is a noisy unbiased subgradient of the objective function, i.e., $\mathbf{E}[g^t | x^t] \in \partial f(x^t)$.

(a) (1 point) Let j be a random index chosen from $\{1, ..., m\}$ such that for every index $i \in \{1, ..., m\}$ the probability that $j = i$ is p_i, i.e.,

$$\mathbb{P}[j = i] = p_i \,,$$

for a given discrete probability distribution $p_1, ..., p_m \geq 0$, $\sum_{i=1}^m p_i = 1$. Show that $\frac{(a_j^T x - b_j)}{m p_j} a_j$ is an unbiased subgradient, i.e.,

$$\mathbf{E}\, \frac{(a_j^T x - b_j)}{m p_j} a_j \in \partial f(x) \,,$$

where the expectation is taken over the random variable j.

(b) (1 point) Assume that $b = Ax^*$ for some vector x^*, i.e., $x^* \in \arg\min f(x)$. Define the error vector $e_t = x_t - x^*$, where x_t is the subgradient descent iterate in (2). Consider the constant step size $\alpha_t = \frac{m}{||A||_F^2}$, the unbiased subgradient from part (a) sampled i.i.d. at every iteration, and the probability distribution

$$p_i = \frac{||a_i||_2^2}{\sum_k ||a_k||_2^2} = \frac{||a_i||_2^2}{||A||_F^2} \,.$$

Show that the error vector e_t obeys the time-varying linear dynamical system

$$e_{t+1} = P_t e_t \,,$$

where P_t is a (random) symmetric projection matrix, i.e., $P_t^T P_t = P_t^2 = P_t$ obeying $\mathbf{E}\, P_t = I - \frac{1}{||A||_F^2} A^T A$.

(c) (1 point) Show that

$$\mathbf{E}\, ||e_{t+1}||_2^2 \leq \left(1 - \frac{\sigma_{\min}(A)^2}{||A||_F^2}\right) \mathbf{E}\, ||e_t||_2^2 \,,$$

where $\sigma_{\min}(A)$ is the smallest singular value of A. *Hint: Note that* $\mathbf{E}[||e_{t+1}||_2^2 | e_t] = \mathbf{E}[e_t^T P_t^T P_t e_t | e_t] = \mathbf{E}[e_t^T P_t e_t | e_t] = e_t^T\, \mathbf{E}[P_t] e_t$, *and*

$$e^T A^T A e \geq \sigma_{\min}(A)^2 e^T e$$

for every vector $e \in \mathbf{R}^n$. Apply this bound recursively to obtain a bound on $\mathbf{E}\, ||e_t||_2^2$ involving only $\mathbf{E}\, ||e_0||_2^2$, $||A||_F$, $\sigma_{\min}(A)$.

(d) (1 point) Assuming zero initialization, $x_0 = 0$, how many iterations are needed to obtain $\mathbf{E}\, ||x_t - x^*||_2^2 \leq 10^{-5}$? Your answer should depend on $||x^*||_2$, $||A||_F$, $\sigma_{\min}(A)$. What is the computational complexity (number of real number multiplications) in Big O notation ?

5 Localization methods

5.1 *Kelley's cutting-plane algorithm.* We consider the problem of minimizing a convex function $f : \mathbf{R}^n \to \mathbf{R}$ over some convex set C, assuming we can evaluate $f(x)$ and find a subgradient $g \in \partial f(x)$ for any x. Suppose we have evaluated the function and a subgradient at $x^{(1)}, \ldots, x^{(k)}$. We can form the piecewise-linear approximation

$$\hat{f}^{(k)}(x) = \max_{i=1,\ldots,k} \left(f(x^{(i)}) + g^{(i)T}(x - x^{(i)}) \right),$$

which satisfies $\hat{f}^{(k)}(x) \leq f(x)$ for all x. It follows that

$$L^{(k)} = \inf_{x \in C} \hat{f}^{(k)}(x) \leq p^\star,$$

where $p^\star = \inf_{x \in C} f(x)$. Since $\hat{f}^{(k+1)}(x) \geq \hat{f}^{(k)}(x)$ for all x, we have $L^{(k+1)} \geq L^{(k)}$.

In Kelley's cutting-plane algorithm, we set $x^{(k+1)}$ to be any point that minimizes $\hat{f}^{(k)}$ over $x \in C$. The algorithm can be terminated when $U^{(k)} - L^{(k)} \leq \epsilon$, where $U^{(k)} = \min_{i=1,\ldots,k} f(x^{(i)})$.

Use Kelley's cutting-plane algorithm to minimize the piecewise-linear function

$$f(x) = \max_{i=1,\ldots,m} (a_i^T x + b_i)$$

that we have used for other numerical examples, with C the unit cube, *i.e.*, $C = \{x \mid \|x\|_\infty \leq 1\}$. Generate the same data we used before using

```
n = 20; % number of variables
m = 100; % number of terms
randn('state',1);
A = randn(m,n);
b = randn(m,1);
```

You can start with $x^{(1)} = 0$ and run the algorithm for 40 iterations. Plot $f(x^{(k)})$, $U^{(k)}$, $L^{(k)}$ and the constant $p^\star$ (on the same plot) versus k.

Repeat for $f(x) = \|x - c\|_2$, where c is chosen from a uniform distribution over the unit cube C. (The solution to this problem is, of course, $x^\star = c$.)

5.2 *Chebyshev center cutting-plane algorithm.* Use the Chebyshev center cutting-plane algorithm to minimize the piecewise-linear function $f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i)$ that we have used for other numerical examples, with C the unit cube, *i.e.*, $C = \{x \mid \|x\|_\infty \leq 1\}$. The data that defines the particular function can be found in the Matlab directory of the subgradient notes on the course web site. You can start with $x^{(1)} = 0$ and run the algorithm for 150 iterations. Plot $f(x^{(k)}) - p^\star$ and $f_{\text{best}}^{(k)} - p^\star$ (on the same log plot) versus k.

5.3 *A barrier cutting-plane algorithm.* We consider the standard inequality constrained convex problem,

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

We assume that f_0 is twice differentiable, and that we can evaluate its value, gradient, and Hessian. (It is always possible to choose f_0 to be linear, so this is not a critical assumption.) The constraint functions $f_1, \ldots, f_m$, however, need not be differentiable, but we assume we can evaluate their values, and a subgradient, at any point.

Suppose we have found the value and a subgradient of one (or possibly more) of the constraint functions at points $x^{(1)}, \ldots, x^{(k)}$. This gives us a polyhedral outer approximation of the feasible set, given by the associated standard deep cuts, which we express as

$$g_i^T x \leq h_i, \quad i = 1, \ldots, m_k,$$

where m_k is the number of inequalities in our polyhedral approximation after k steps.

Define $x^{*(k)}$ as the minimizer of

$$(m_k/\alpha^{(k)}) f_0(x) - \sum_{i=1}^{m_k} \log(h_i - g_i^T x)$$

(which we assume exists and is unique), where $\alpha^{(k)} > 0$ is a parameter. The point $x^{*(k)}$ can be computed using an infeasible start Newton method, starting from the previous point $x^{*(k-1)}$ (or any other point, for that matter).

(a) Show that $p^\star \geq f_0(x^{*(k)}) - \alpha^{(k)}$, where $p^\star$ is the optimal value of the original problem. In particular, if $x^{*(k)}$ is feasible for the original problem, it is $\alpha^{(k)}$-suboptimal.

(b) Now we can describe the *barrier cutting-plane method*. At each step, we first compute $x^{*(k)}$. If $x^{*(k)}$ is infeasible, we add a cut (or set of cuts) from a violated constraint to our outer approximation polyhedron, and then increment k. If $x^{*(k)}$ is feasible, we reduce $\alpha^{(k)}$ by some factor μ, typically between 2 and 10, and recompute $x^{*(k)}$ using this new value of $\alpha^{(k)}$. We can (reliably) stop when $x^{*(k)}$ is feasible and $\alpha^{(k)} \leq \epsilon$, our required tolerance.

Apply the barrier cutting-plane method to the linear program in inequality form,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b, \end{array}$$

that we used as our example for the subgradient method notes. (The data that defines the particular problem can be found in the Matlab directory of the subgradient notes on the course web site.) You can initialize the polyhedral approximation of the feasible set with the $(2n)$ inequalities $\|x\|_\infty \leq 1$. Run the algorithm until $x^{*(k)}$ is feasible and $\alpha^{(k)} \leq \epsilon$, with $\epsilon = 10^{-3}$. You can use parameter values

$\alpha^{(1)} = 1$ and $\mu = 5$. Plot $f_{\text{best}}^{(k)} - f^\star$ and $\alpha^{(k)}$ (on the same plot) versus iteration number k. Here $f_{\text{best}}^{(k)}$ is the best objective value encountered so far among feasible points, and $f^\star$ is the optimal value of the original problem.

Hint. Write a function that computes $x^{*(k)}$ using an infeasible start Newton method, starting from the function `acent.m`, available in the Matlab directory of the ACCPM notes on the course web site.

5.4 *Minimum volume ellipsoid covering a half-ellipsoid.* In this problem we derive the update formulas used in the ellipsoid method, *i.e.*, we will determine the minimum volume ellipsoid that contains the intersection of the ellipsoid

$$\mathcal{E} = \{x \in \mathbf{R}^n \mid (x - x_c)^T P^{-1} (x - x_c) \leq 1\}$$

and the halfspace

$$\mathcal{H} = \{x \mid g^T (x - x_c) \leq 0\}.$$

We'll assume that $n > 1$, since for $n = 1$ the problem is easy.

(a) We first consider a special case: $\mathcal{E}$ is the unit ball centered at the origin ($P = I$, $x_c = 0$), and $g = -e_1$ (e_1 is the first unit vector), so $\mathcal{E} \cap \mathcal{H} = \{x \mid x^T x \leq 1,\ x_1 \geq 0\}$. Let

$$\tilde{\mathcal{E}} = \{x \mid (x - \tilde{x}_c)^T \tilde{P}^{-1} (x - \tilde{x}_c) \leq 1\}$$

denote the minimum volume ellipsoid containing $\mathcal{E} \cap \mathcal{H}$. Since $\mathcal{E} \cap \mathcal{H}$ is symmetric about the line through first unit vector e_1, it is clear (and not too hard to show) that $\tilde{\mathcal{E}}$ will have the same symmetry. This means that the matrix $\tilde{P}$ is diagonal, of the form $\tilde{P} = \mathbf{diag}(\alpha, \beta, \beta, \ldots, \beta)$, and that $\tilde{x}_c = \gamma e_1$ (where $\alpha, \beta > 0$ and $\gamma \geq 0$). So now we have only three variables to determine: α, β, and γ. Express the volume of $\tilde{\mathcal{E}}$ in terms of these variables, and also the constraint that $\tilde{\mathcal{E}} \supseteq \mathcal{E} \cap \mathcal{H}$. Then solve the optimization problem directly, to show that

$$\alpha = \frac{n^2}{(n+1)^2}, \quad \beta = \frac{n^2}{n^2 - 1}, \quad \gamma = \frac{1}{n+1}$$

(which agrees with the formulas we gave, for this special case).

Hint. To express $\mathcal{E} \cap \mathcal{H} \subseteq \tilde{\mathcal{E}}$ in terms of the variables, it is necessary and sufficient for the conditions on α, β, and γ to hold on the boundary of $\mathcal{E} \cap \mathcal{H}$, *i.e.*, at the points

$$x_1 = 0, \quad x_2^2 + \cdots + x_n^2 \leq 1,$$

or the points

$$x_1 \geq 0, \quad x_1^2 + x_2^2 + \cdots + x_n^2 = 1.$$

(b) Now consider the general case, stated at the beginning of this problem. Show how to reduce the general case to the special case solved in part (a).

Hint. Find an affine transformation that maps the original ellipsoid to the unit ball, and g to $-e_1$. Explain why minimizing the volume in these transformed coordinates also minimizes the volume in the original coordinates.

(c) Finally, show that the volume of the ellipse $\tilde{\mathcal{E}}$ satisfies $\mathbf{vol}(\tilde{\mathcal{E}}) \leq e^{-\frac{1}{2n}} \mathbf{vol}(\mathcal{E})$.

Hint. Compute the volume of the ellipse $\mathcal{E}$ as a function of the eigenvalues of P, then use the results of parts (a) and (b) to argue that the volume computation can be reduced to the special case in part (a).

5.5 *Finding a point in the intersection of Euclidean balls.* We consider the problem of finding a point in the intersection of m Euclidean balls in $\mathbf{R}^n$, $\mathcal{B}_i = \{z \mid \|z - c_i\|_2 \leq R_i\}$. You will use the following four methods to solve the problem.

- Alternating projections.
- Parallel projections. This is the variation on alternating projections, described in exercise 3.2, in which the next point is the average of the projection of the current point onto each of the sets.
- Analytic center cutting-plane method (with deep cuts).
- Ellipsoid method (with deep cuts).

For the first two methods, use the starting point $x^{(1)} = 0$. For ACCPM, use as initial polyhedron the ℓ_∞-norm ball, centered at 0, of smallest radius, that encloses all the balls. (Be sure to explain how you calculate this radius.) For the ellipsoid method, use as initial ellipsoid the ℓ_2-norm ball, centered at 0, of smallest radius, that encloses all the balls. (Be sure to explain how you calculate this radius.)

The data for the particular problem instance you will solve can be found on the class website in `ball_intersection_data.m`.

Plot the maximum distance of the current point $x^{(k)}$ to the balls, versus iteration k. (Note that the maximum distance is easily computed, and is *not* the same as the distance to the intersection of the balls.) We suggest running the alternating projections algorithms, and the ellipsoid algorithm, for 500 iterations. ACCPM should converge much faster (in terms of iterations).

5.6 *Ellipsoid method for an SDP.* We consider the SDP

$$\begin{array}{ll} \text{maximize} & \mathbf{1}^T x \\ \text{subject to} & x_i \succeq 0, \quad \Sigma - \mathbf{diag}(x) \succeq 0, \end{array}$$

with variable $x \in \mathbf{R}^n$ and data $\Sigma \in \mathbf{S}^n_{++}$. The first inequality is a vector (componentwise) inequality, and the second inequality is a matrix inequality. (This specific SDP arises in several applications.)

Explain how to use the ellipsoid method to solve this problem. Describe your choice of initial ellipsoid and how you determine a subgradient for the objective (expressed as $-\mathbf{1}^T x$, which is to be minimized) or constraint functions (expressed as $\max_i(-x_i) \leq 0$ and $\lambda_{\max}(\mathbf{diag}(x) - \Sigma) \leq 0$). You can describe a basic ellipsoid method; you do not need to use a deep-cut method, or work in the epigraph.

Try out your ellipsoid method on some randomly generated data, with $n \leq 20$. Use a stopping criterion that guarantees 1% accuracy. Compare the result to the solution found using CVX. Plot the upper and lower bounds from the ellipsoid method, versus iteration number.

6 Decomposition methods

6.1 *Distributed method for bi-commodity network flow problem.* We consider a network (directed graph) with n arcs and p nodes, described by the incidence matrix $A \in \mathbf{R}^{p\times n}$, where

$$A_{ij} = \begin{cases} 1, & \text{if arc } j \text{ enters node } i \\ -1, & \text{if arc } j \text{ leaves node } i \\ 0, & \text{otherwise.} \end{cases}$$

Two commodities flow in the network. Commodity 1 has source vector $s \in \mathbf{R}^p$, and commodity 2 has source vector $t \in \mathbf{R}^p$, which satisfy $\mathbf{1}^T s = \mathbf{1}^T t = 0$. The flow of commodity 1 on arc i is denoted x_i, and the flow of commodity 2 on arc i is denoted y_i. Each of the flows must satisfy flow conservation, which can be expressed as $Ax + s = 0$ (for commodity 1), and $Ay + t = 0$ (for commodity 2).

Arc i has associated flow cost $\phi_i(x_i, y_i)$, where $\phi_i : \mathbf{R}^2 \to \mathbf{R}$ is convex. (We can impose constraints such as nonnegativity of the flows by restricting the domain of ϕ_i to $\mathbf{R}^2_+$.) One natural form for ϕ_i is a function only the total traffic on the arc, *i.e.*, $\phi(x_i, y_i) = f_i(x_i + y_i)$, where $f_i : \mathbf{R} \to \mathbf{R}$ is convex. In this form, however, ϕ is not strictly convex, which will complicate things. To avoid these complications, we will assume that ϕ_i is strictly convex.

The problem of choosing the minimum cost flows that satisfy flow conservation can be expressed as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \phi_i(x_i, y_i) \\ \text{subject to} & Ax + s = 0, \quad Ay + t = 0, \end{array}$$

with variables $x,\ y \in \mathbf{R}^n$. This is the *bi-commodity network flow problem.*

(a) Propose a distributed solution to the bi-commodity flow problem using dual decomposition. Your solution can refer to the conjugate functions ϕ_i^*.

(b) Use your algorithm to solve the particular problem instance with

$$\phi_i(x_i, y_i) = (x_i + y_i)^2 + \epsilon(x_i^2 + y_i^2), \qquad \mathbf{dom}\, \phi_i = \mathbf{R}^2_+,$$

with $\epsilon = 0.1$. The other data for this problem can be found in `bicommodity_data.[m|jl]`. To check that your method works, compute the optimal value $p^\star$, using CVX.

For the subgradient updates use a constant stepsize of 0.1. Run the algorithm for 200 iterations and plot the dual lower bound versus iteration. With a logarithmic vertical axis, plot the norms of the residuals for each of the two flow conservation equations, versus iteration number, on the same plot.

Hint. We have posted a function `[x,y] = quad2_min(eps,alpha,beta)`, which computes

$$(x^*, y^*) = \operatorname*{argmin}_{x\geq 0, y\geq 0} \left((x + y)^2 + \epsilon(x^2 + y^2) + \alpha x + \beta y\right)$$

analytically. You might find this function useful.

6.2 *Distributed lasso.* Consider the ℓ_1-regularized least-squares ('lasso') problem

$$\text{minimize} \quad f(z) = (1/2)\left\| \begin{bmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix} - \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \right\|_2^2 + \lambda \left\| \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix} \right\|_1,$$

with optimization variable $z = (x_1, x_2, y) \in \mathbf{R}^{n_1} \times \mathbf{R}^{n_2} \times \mathbf{R}^p$. We can think of x_i as the local variable for system i, for $i = 1, 2$; y is the common or coupling variable.

(a) *Primal decomposition.* Explain how to solve this problem using primal decomposition, using (say) the subgradient method for the master problem.

(b) *Dual decomposition.* Explain how to solve this problem using dual decomposition, using (say) the subgradient method for the master problem. Give a condition (on the problem data) that allows you to guarantee that the primal variables $x_i^{(k)}$ converge to optimal values.

(c) *Numerical example.* Generate some numerical data as explained below, and solve the problem (using CVX) to find the optimal value $p^\star$. Implement primal and dual decomposition (as in parts (a) and (b)), using CVX to solve the subproblems, and the subgradient method for the master problem in both cases. For primal decomposition, plot the relative suboptimality $(f(z^{(k)}) - p^\star)/p^\star$ versus iteration. For dual decomposition, plot the relative consistency residual $\|y_1^{(k)} - y_2^{(k)}\|_2 / \|y^\star\|_2$ versus iteration, where $y^\star$ is an optimal value of y for the problem. In each case, you needn't worry about attaining a relative accuracy better than 0.001, which corresponds to 0.1%.

Generating the data. Generate A_i, B_i, and c_i with entries from a standard Gaussian, with dimensions $n_1 = 100$, $n_2 = 200$, $p = 10$, $m_1 = 250$, and $m_2 = 300$ (these last two are the dimensions of c_1 and c_2). Check that the condition you gave in part (b) is satisfied. Choose $\lambda = 0.1\lambda_{\max}$, where $\lambda_{\max}$ is the value of λ above which the solution is $z^\star = 0$. (See homework 2, exercise 3.) To get reasonable convergence (say, in a few tens of iterations), you may need to play with the subgradient step size.

You are of course welcome (even, encouraged) to also try your distributed lasso solver on problem instances other than the one generated above.

6.3 *Parallel Subgradient Computation.* In this question, you will implement subgradient computations using the parallel computing library Dask.

(a) (0 points) Read the example showing how to modify Python code to utilize data parallelism via Dask here. Install Python and Dask as described on this page.

(b) (2 points) Suppose that $x \in \mathbb{R}^n$ is fixed and given. Compute a subgradient of the function

$$f(x) := \max_{j \in \{1,\ldots,m\}} a_j^T x$$

by using the Python code template below (also available in Canvas/Files/Homeworks). Set $n = 100 \times 10^6$ and $m = 4$. Generate $x \in \mathbb{R}^n$ and $a_1, ..., a_m \in \mathbb{R}^n$ randomly and independently from a standard normal distribution. Your code should return a valid subgradient of $f(x)$ at x. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient using the serial approach.

(c) (3 points) Implement the same subgradient computation in (b) using Dask to see the benefit of data parallelism in terms of the total computation time. You only have to modify your code in (b) using `dask.delayed` as shown in this tutorial. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient with parallelization.

(d) (2 points) Implement the same subgradient calculation in part (b) using `numpy.matmul()` and `numpy.argmax()`. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient with Numpy.

(e) (1 points) Visualize the computation graph for your Dask based implementation using the function `visualize()` for $n = 5$, $m = 4$.

```
from time import time
import dask
import numpy as np
def inprod(x, y):
    return np.dot(x,y)
n, m = 100000000, 4
data = np.random.randn(m,n)
start = time()
output = []
x= np.random.randn(n)
for i in range(data.shape[0]):
    output.append(inprod(data[i,:],x))
index = np.argmax(output)
print("Time spent for the computation without parallelization:",time()-start)
```

7 Monotone operators and operator splitting

7.1 *Cutting plane for nonexpansive operators.* Suppose $F : \mathbf{R}^n \to \mathbf{R}^n$ is nonexpansive, and let $x^\star$ be any fixed point. For $x \in \mathbf{R}^n$, let $z = (1/2)(x + F(x))$ (which would be the next iterate in damped iteration, with $\theta = 1/2$). Show that

$$(F(x) - x)^T(x^\star - z) \geq 0.$$

Draw a picture illustrating this inequality. Thus, by evaluating $F(x)$, we can construct a (deep-cut) cutting plane that separates x from the fixed point set. This means we can find a fixed point of a nonexpansive operator using any localization method, such as the ellipsoid method or the analytic-center cutting-plane method.

7.2 *Diode relation.* The relation $D = \{(x_1, x_2) \mid x_1x_2 = 0,\ x_1 \leq 0,\ x_2 \geq 0\}$ is called the *diode relation*, since it is the V-I characteristic of an ideal diode. Show that D is monotone, and find its resolvent (with $\lambda > 0$) and Cayley operator. Plot the relation D, its resolvent R, and its Cayley operator C.

7.3 *Fixed point set of nonexpansive operator.* Suppose that F is a nonexpansive operator on $\mathbf{R}^n$ with $\mathbf{dom}\, F = \mathbf{R}^n$. Show that its fixed point set, $\mathcal{X} = \{x \mid F(x) = x\}$, is convex.

Hint. Prove, and then use, the following fact: If the sum of the (Euclidean) distances between a point and two others equals the distance between the others, then the point is on the line segment between the two others. (You should draw a picture of this.)

7.4 *Kelley's cutting-plane algorithm with proximal regularization.* The Kelley cutting-plane method was described in an earlier exercise. In this exercise we explore a simple modification of the basic Kelley cutting-plane method that can dramatically speed up convergence.

We consider the problem of minimizing a convex function $f : \mathbf{R}^n \to \mathbf{R}$ over a convex set C, using a subgradient oracle for f. Suppose we have evaluated the function and a subgradient at the iterates $x^{(1)}, \ldots, x^{(k)}$. We define the associated piecewise-linear approximation (and under-estimator) of f as

$$\hat{f}^{(k)}(x) = \max_{i=1,\ldots,k} \left(f(x^{(i)}) + g^{(i)T}(x - x^{(i)})\right).$$

Kelley's cutting-plane algorithm is the iteration

$$x^{(k+1)} = \operatorname*{argmin}_{x \in C} \hat{f}^{(k)}(x).$$

Here we mean any minimizer; the argmin need not be unique. This requires minimizing a piecewise-linear function over C in each iteration.

Kelley's cutting-plane algorithm with proximal regularization is the iteration

$$x^{(k+1)} = \operatorname*{argmin}_{x \in C} \left(\hat{f}^{(k)}(x) + (\rho/2)\|x - x^{(k)}\|_2^2\right)$$

where $\rho > 0$ is a parameter. (Note that $\rho = 0$ gives the standard Kelley cutting-plane algorithm.) With proximal regularization, the argmin is unique. Kelley's cutting-plane algorithm with proximal regularization requires minimizing a quadratic function (with diagonal Hessian) plus a piecewise-linear function over C in each iteration.

The lower and upper bounds on $p^\star = \inf_{x \in C} f(x)$ (used in Kelley's cutting-plane method)

$$L^{(k)} = \inf_{x \in C} \hat{f}^{(k)}(x) \leq p^\star, \qquad U^{(k)} = \min_{i=1,\ldots,k} f(x^{(i)}) \geq p^\star,$$

are of course still valid, although evaluating $L^{(k)}$ now involves solving an additional problem each iteration. (For this reason, the method is often used without calculating $L^{(k)}$.)

Use proximal regularization for Kelley's cutting-plane algorithm in order to solve the lasso problem inside the unit cube,

$$\begin{array}{ll} \text{minimize} & (1/2)\|Ax - b\|_2^2 + \|x\|_1 \\ \text{subject to} & \|x\|_\infty \leq 1, \end{array}$$

with variable $x \in \mathbf{R}^n$, and parameters $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$. We recommend choosing a problem with dimensions $n = 10$ and $m = 50$. One way to generate data is the following:

```
randn('state',0);
rand('state',0);
A = randn(m,n);
z = 2*rand(n,1)-1;
b = A*z;
```

In this way, the optimal point z will be inside the unit cube, so the constraint $\|x\|_\infty \leq 1$ will not be active at the optimal point. You may need to experiment a bit with the regularization parameter ρ.

Compare the results to Kelley's cutting-plane method without regularization.

7.5 *Contraction analysis of projected gradient method.* Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is convex and twice differentiable, with $mI \preceq \nabla^2 f(x) \preceq LI$ for all x. (That is, f is strongly convex, and ∇f has Lipschitz constant L.) Let $\mathcal{C} \subset \mathbf{R}^n$ be a closed convex set, and Π be the Euclidean projection onto $\mathcal{C}$. The projected gradient method, with step size $\alpha > 0$, is the iteration

$$x^{k+1} = F(x^k) = \Pi(x^k - \alpha \nabla f(x^k)).$$

(a) Show that x is a fixed point of F if and only if x minimizes $f(x)$ subject to $x \in \mathcal{C}$.

(b) Find conditions on α, m, and L under which F is a contraction.

(c) Find the value of α that minimizes the Lipschitz constant on F you found in part (b), and give the associated Lipschitz constant.

7.6 *Iterative refinement for linear equations.* Suppose that $A \in \mathbf{S}^n_+$, so choosing x to minimize $f(x) = (1/2)x^TAx - b^Tx$ is the same as solving the linear equations $Ax = b$. Show that the proximal point algorithm, applied to minimizing f, can be expressed as the following iteration, initialized with $x^0 = 0$ (or any other initial point):

$$\begin{aligned} r^k &= b - Ax^k \\ x^{k+1} &= x^k + (A + \rho I)^{-1} r^k \end{aligned}$$

where $\rho = 1/\lambda > 0$ is a parameter. This is called iterative refinement.

Remark. This is used in the following context. Suppose that A has a very high (or infinite) condition number, so carrying out a Cholesky factorization of A is numerically unstable (or impossible). For ρ large enough (and indeed, not too large), we can easily carry out a Cholesky factorization of $A + \rho I$. We store the factorization, so that subsequent solves in iterative refinement are cheaper (by a factor of n, when A is dense). Thus the cost of a few steps of iterative refinement (or more, when n is large) is approximately zero.

7.7 *Optimal parameter choice for Peaceman-Rachford algorithm.* Consider the problem

$$\text{minimize} \quad f(x) + g(x),$$

with variable $x \in \mathbf{R}^n$, where $f, g : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ are convex, closed, and proper functions. This problem is equivalent to solving $0 \in \partial f(x) + \partial g(x)$. The Peaceman-Rachford iteration for solving this problem is

$$z^{k+1} = C_{\partial f} C_{\partial g}(z^k),$$

where

$$C_{\partial f}(z) = 2\left(I + \lambda \partial f\right)^{-1}(z) - z = 2\mathbf{prox}_{\lambda f}(z) - z$$

is the Cayley operator of ∂f, and similarly for $C_{\partial g}$, with $\lambda > 0$. This iteration need not converge. But it does converge if either $C_{\partial f}$ or $C_{\partial g}$ is a contraction. (Note that $C_{\partial f}$ and $C_{\partial g}$ are nonexpansive.)

(a) Assume that f is convex quadratic, $f(x) = (1/2)x^TPx + q^Tx$, with $P \in \mathbf{S}^n_{++}$ and $q \in \mathbf{R}^n$. Find the smallest Lipschitz constant on $C_{\partial f} C_{\partial g}$ in terms of λ and P, without any further assumptions on g. (Your answer can involve the eigenvalues of P, ordered as $\lambda_{\max}(P) = \lambda_1 \geq \cdots \geq \lambda_n = \lambda_{\min}(P) > 0$.)

(b) Find λ^{opt}, the value of λ for which the Lipschitz constant in part (a) is minimized, and give the associated Lipschitz constant for $C_{\partial f} C_{\partial g}$. Express λ^{opt} in terms of the eigenvalues of P. Express the optimal Lipschitz constant in terms of the condition number κ of P, given by $\kappa = \lambda_{\max}(P)/\lambda_{\min}(P)$.

(c) Consider the case $f(x) = \|Ax - b\|_2^2$ and g the indicator function of the nonnegative orthant. (This is the nonnegative least-squares problem.) The optimality conditions for this problem are

$$x \succeq 0, \qquad A^T(Ax - b) \succeq 0, \qquad x_i(A^T(Ax - b))_i = 0, \quad i = 1, \ldots, n.$$

At each iteration of the Peaceman-Rachford algorithm, the point $x^k = R_{\partial g}(z^k)$ satisfies the first optimality condition. We stop the algorithm when $A^T(Ax^k - b) \succeq -\epsilon\mathbf{1}$, and $(x^k)_i(A^T(Ax^k - b))_i \leq \epsilon$ for $i = 1, \ldots, n$, where $\epsilon > 0$ is a tolerance.

Implement the Peaceman-Rachford algorithm in this case, with tolerance $\epsilon = 10^{-4}$.

Generate a random instance of the problem with $m = 500$ and $n = 200$, and plot the number of iterations required versus λ over a range that includes λ^{opt} (from part (b)). The horizontal axis should be logarithmic, showing $\lambda/\lambda^{\text{opt}}$ (say, for 30 values from 0.01 to 100).

Repeat for several random instances, and briefly comment on the results.

7.8 *Solving LPs via alternating projections.* Consider an LP in standard form,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \succeq 0, \end{array}$$

with variable $x \in \mathbf{R}^n$, and where $A \in \mathbf{R}^{m \times n}$. A tuple $(x, \nu, \lambda) \in \mathbf{R}^{2n+m}$ is primal-dual optimal if and only if

$$Ax = b, \quad x \succeq 0, \quad -A^T\nu + \lambda = c, \quad \lambda \succeq 0, \quad c^T x + b^T \nu = 0.$$

These are the KKT optimality conditions of the LP. The last constraint, which states that the duality gap is zero, can be replaced with an equivalent condition, $\lambda^T x = 0$, which is complementary slackness.

(a) Let $z = (x, \nu, \lambda)$ denote the primal-dual variable. Express the optimality conditions as $z \in \mathcal{A} \cap \mathcal{C}$, where $\mathcal{A}$ is an affine set, and $\mathcal{C}$ is a simple cone. Give $\mathcal{A}$ as $\mathcal{A} = \{z \mid Fz = g\}$, for appropriate F and g.

(b) Explain how to compute the Euclidean projections onto $\mathcal{A}$ and also onto $\mathcal{C}$.

(c) Implement alternating projections to solve the standard form LP. Use $z^{k+1/2}$ to denote the iterate after projection onto $\mathcal{A}$, and z^{k+1} to denote the iterate after projection onto $\mathcal{C}$. Your implementation should exploit factorization caching in the projection onto $\mathcal{A}$, but you don't need to worry about exploiting structure in the matrix F.

Test your solver on a problem instance with $m = 100$, $n = 500$. Plot the residual $\|z^{k+1} - z^{k+1/2}\|_2$ over 1000 iterations. (This should converge to zero, although perhaps slowly.)

Here is a simple method to generate LP instances that are feasible. First, generate a random vector $\omega \in \mathbf{R}^n$. Let $x^\star = \max\{\omega, 0\}$ and $\lambda^\star = \max\{-\omega, 0\}$, where the maximum is taken elementwise. Choose $A \in \mathbf{R}^{m\times n}$ and $\nu^\star \in \mathbf{R}^m$ with random entries, and set $b = Ax^\star$, $c = -A^T\nu^\star + \lambda^\star$. This gives you an LP instance with optimal value $c^Tx^\star$.

(d) Implement Dykstra's alternating projection method and try it on the same problem instances from part (c). Verify that you obtain a speedup, and plot the same residual as in part (c).

7.9 *A proximal subgradient method.* Consider the following problem:

$$\text{minimize } f(x) + \varphi(x) \;\; \text{subject to} \;\; x \in X,$$

where $X \subset \mathbf{R}^n$ is a closed convex set, and $f : \mathbf{R}^n \to \mathbf{R}$ and $\varphi : \mathbf{R}^n \to \mathbf{R}$ are convex. We study the convergence of the proximal *sub*gradient method, which iterates

$$\begin{aligned} g^k &\in \partial f(x^k), \\ x^{k+1} &= \mathbf{prox}_{\alpha_k\varphi}(x^k - \alpha_k g^k) = \operatorname*{argmin}_{x\in X} \left\{ (g^k)^T x + \varphi(x) + \frac{1}{2\alpha_k} \left\|x - x^k\right\|_2^2 \right\}. \end{aligned}$$

In particular, we show that the function φ does not hurt convergence over what is attainable when minimizing $f(x)$.

(a) Show that for some $\varphi'(x^{k+1}) \in \partial\varphi(x^{k+1})$, we have

$$\left(g^k + \varphi'(x^{k+1}) + \frac{1}{\alpha_k}(x^{k+1} - x^k)\right)^T (y - x^{k+1}) \geq 0 \quad \text{for all } y \in X.$$

(b) Using the representation from part (a) and choosing $y = x^\star$, show that for any $x^\star \in X$,

$$f(x^k) - f(x^\star) \leq \varphi(x^\star) - \varphi(x^{k+1}) + \frac{1}{2\alpha_k}\left\|x^k - x^\star\right\|_2^2 - \frac{1}{2\alpha_k}\left\|x^{k+1} - x^\star\right\|_2^2 + \frac{\alpha_k}{2}\left\|g^k\right\|_2^2.$$

Hint. The inequality $g^Ty \leq \frac{1}{2\alpha}\|y\|_2^2 + \frac{\alpha}{2}\|g\|_2^2$, valid for any $\alpha \geq 0$, may prove useful.

(c) Show that for a fixed stepsize sequence $\alpha_k = \alpha$ for all k, we have

$$\sum_{i=1}^k \left[f(x^i) + \varphi(x^i) - f(x^\star) - \varphi(x^\star)\right] \leq \frac{1}{2\alpha}\left\|x^1 - x^\star\right\|_2^2 + \frac{\alpha}{2}\sum_{i=1}^k \left\|g^i\right\|_2^2 + \varphi(x^1) - \varphi(x^{k+1}).$$

(d) Let $\bar{x}^k = \frac{1}{k}\sum_{i=1}^k x^i$, and assume that $x^1 \in \operatorname{argmin}_{x\in X}\varphi(x)$ and $\|g^i\|_2 \leq G$ for all i. Give a stepsize α such that

$$f(\bar{x}^k) + \varphi(\bar{x}^k) - f(x^\star) - \varphi(x^\star) \leq \frac{\|x^1 - x^\star\|_2 G}{\sqrt{k}}.$$

8 ADMM

8.1 *Reducing the general ADMM problem to consensus.* The general form ADMM problem is

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array}$$

with variables $x \in \mathbf{R}^n$, $z \in \mathbf{R}^m$. We assume that f and g are convex, and that $c \in \mathbf{R}^p$ (*i.e.*, there are p scalar equality constraints).

A special case is the (two variable) consensus problem, which has the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0, \end{array}$$

with variables $x \in \mathbf{R}^n$, $z \in \mathbf{R}^m$.

Show how to reduce the general form ADMM problem to the consensus problem. (In other words, explain how to solve the general form problem, assuming you can solve the consensus problem.) Explain why the functions in your consensus problem are convex. (You do not have to worry about analysis details, such as showing that functions are closed.) Explain the relation between ADMM for the general form problem, and ADMM for the consensus problem that you form in your reduction.

Remark. We have already seen that ADMM and Douglas-Rachford splitting agree for the consensus problem. The equivalence in this exercise connects the general ADMM form with Douglas-Rachford splitting.

Hint. Define $\tilde{f}(\tilde{x}) = \inf_{Ax=\tilde{x}} f(x)$.

8.2 *Block distributed flow control.* We consider the standard flow control problem, with n flows, given by $f \in \mathbf{R}^n_+$, that travel over m links of a network. The traffic on each link is the sum of the flows that pass over the link, which we express as $t = Rf$, where $t \in \mathbf{R}^m$ is the vector of link traffic, and $R \in \mathbf{R}^{m\times n}$ is the routing matrix, defined as

$$R_{ij} = \begin{cases} 1 & \text{flow } j \text{ passes over link } i \\ 0 & \text{otherwise.} \end{cases}$$

The traffic on each link cannot exceed its capacity: $t \preceq c$, where $c \in \mathbf{R}^m$ is the vector of link capacities. The objective is to maximize a total utility, given by

$$U(f) = \sum_{j=1}^{n} U_j(f_j),$$

where $U_j : \mathbf{R}_+ \to \mathbf{R}$ are concave functions. The flow vector f is the variable; R, c, and $U_1, \ldots, U_n$ are problem data.

Here we consider the case when the m links are partitioned into two groups. This partitioning of the links induces a partitioning of the flows as well, into three groups:

flows that pass only over links in the first group, flows that pass only over links in the second group, and flows that pass over links in both groups.

The goal is to use ADMM to develop a method to solve the flow control problem, using one flow control solver for each group of links. Do this by replicating (*i.e.*, making copies of) the flow variables that pass over links in both groups, and introducing an equality (consistency) constraint between the replicated variables. This gives a consensus problem. For this problem there is a very simple way to obtain a feasible flow vector from the replicated ones (each of which is only known to be feasible for the flows over one of the groups of links): We simply take the (elementwise) minimum.

(a) Give the details of using ADMM to solve the distributed flow control problem as outlined above. Explain what problems are solved in each step, and how they coordinate to solve the whole problem.

(b) Implement your method on the problem instance with data given in the file `dist_flow_ctrl_data.m`, with $U_j(x) = \sqrt{x}$, for $j = 1, \ldots, n$. You can use CVX to solve the two flow control problems required in each iteration. To show convergence of the algorithm, plot $U(f^k) - U^\star$ versus k, where f^k is the feasible flow vector at iteration k obtained using the method described above, and $U^\star$ is the optimal value (found using CVX).

The data file contains R and c. The partitions are encoded as follows. The first m_a rows of R (and entries of c) correspond to the links in the first group, and the rest correspond to the second group. The first n_a entries of f correspond to the flows that pass over links in the first group only; the next n_b entries correspond to the flows that pass over links in the second group only; and the remaining entries of f are those that pass over both groups of links. The data m_a, n_a, and n_b are given in the data file.

8.3 *ADMM for smart grid device coordination.* We consider an electrical grid consisting of N devices that exchange electricity over T time periods. Device i has energy profile $p^i \in \mathbf{R}^T$, with p^i_t denoting the energy consumed by device i in time period t, for $t = 1, \ldots, T$, $i = 1, \ldots, N$. (When $p^i_t < 0$, device i is producing energy in time period t.) Each device has a convex objective function $f_i : \mathbf{R}^T \to \mathbf{R}$, which we also use to encode constraints, by setting $f_i(p^i) = \infty$ for profiles that violate the constraints of device i. In each time period the energy flow has to balance, which means

$$\sum_{i=1}^{N} p^i_t = 0, \quad t = 1, \ldots, T.$$

The optimal profile coordination problem is to minimize the total cost, $\sum_{i=1}^{N} f_i(p^i)$, subject to the balance constraint, with variables p^i, $i = 1, \ldots, N$.

In this problem you will use ADMM to solve the optimal profile coordination problem in a distributed way, with each device optimizing its own profile, and exchanging messages to coordinate all of the profiles.

From this point on, we consider a specific (and small) problem instance. There are three devices: a generator, a fixed load, and a battery, with cost functions described below.

- *Generator.* The generator has upper and lower generator limits: $P^{\min} \leq -p_t \leq P^{\max}$, for $t = 1, \ldots, T$. (Note the minus sign, since a generator's profile is typically negative, using our convention.) The objective is

$$f_{\text{gen}}(p) = \sum_{t=1}^{T} (\alpha(-p_t)^2 + \beta(-p_t)),$$

  where $\alpha, \beta > 0$ are given constants.
- *Fixed load.* The fixed load has zero objective function and the constraint that its power profile must equal a given consumption profile $d \in \mathbf{R}^T$.
- *Battery.* The battery has zero objective function, and charge/discharge limits given by C and D, respectively: $-D \leq p_t \leq C$, for $t = 1, \ldots, T$. The battery is initially uncharged (*i.e.*, $q_1 = 0$), so its charge level in period t is $q_t = \sum_{\tau=1}^{t-1} p_\tau$ (we neglect losses for this problem). The charge level must be nonnegative, and cannot exceed the battery capacity: $0 \leq q_t \leq Q$, $t = 1, \ldots, T+1$. The charge level is extended to time $T+1$ to allow the battery to charge/discharge in time T, subject to the operational constraints.

(a) Use CVX to solve the problem with data given in `admm_smart_grid_data.m`. Plot the (optimal) power profile for the generator and battery, as well as the battery charge level.

(b) Implement ADMM for this problem (you may use CVX to solve each device's local optimization in each ADMM iteration). Experiment with a few values of the parameter ρ to see its effect on the convergence rate of the algorithm. Plot the norm of the energy balance residual, versus iteration. Plot the power profiles of the generator and battery, as well as the energy balance residual, for several values of iteration (say, after one iteration, after 10 iterations, and after 50). Check the results against the solution found by CVX.

8.4 *Radiation treatment planning.* (For some background, and a slightly different formulation, see exercise of same name in *Convex Optimization Additional Exercises*.) The radiation treatment planning problem is to choose the radiation beam levels in a treatment plan, given by $b \in \mathbf{R}^n_+$, where $b_i \in [0, 1]$ is the ith beam level. The beam levels result in a dosage pattern $d \in \mathbf{R}^m$, where d_i is the radiation dosage in voxel i of the patient. These are related by $d = Ab$, where the matrix A depends on the geometry of the equipment and possibly scattering inside the patient. You can assume that A is known and given, with $m > n$. (Although not relevant in this problem, the entries of A are nonnegative.) The goal is to achieve $d^{\min} \preceq d \preceq d^{\max}$, where $d^{\min} \in \mathbf{R}^m_+$ and

$d^{\max} \in \mathbf{R}^m_+$ are given lower and upper target dosages, respectively. (These vary by voxel.) Typically $d^{\min}_i$ is large when voxel i is in the tumor, and $d^{\max}_i$ is small when voxel i is outside the tumor. These target dosages are generally not achievable, so we form a sum of violations objective, which we minimize:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T(d^{\min} - d)_+ + \mathbf{1}^T(d - d^{\max})_+ \\ \text{subject to} & 0 \preceq b \preceq \mathbf{1}, \quad d = Ab, \end{array}$$

with variables b and d.

(a) Explain how to use ADMM to solve the treatment planning problem, with one step handling the linear equality constraint relating b and d, and the other step handling the nonlinear objective in d and the constraints on b. Work out each step *explicitly*, and give the computational cost (in order). You *may not* use CVX for the prox operators (we will deduct points for solutions that do). You can use factorization caching for the step that handles the equality constraint, so give the computational cost for the first iteration and subsequent iterations separately. You can assume that A is dense. (In real problems, A is sparse.)

Hints. Work out the projection onto the equality constraint, and try to express it in terms of $(I + A^TA)^{-1}$, *not* $(I + AA^T)^{-1}$ using the matrix inversion lemma. You may also find it useful to cache A^TA. If your cost per iteration (after the first one) is not $O(mn)$, it's wrong.

(b) Implement ADMM for the problem instance given in `dosage_opt_data.m`. This script calls `line_pixel_length.m` (while creating the matrix A), which you will also need to download.

The problem instance is 2-dimensional, with a rectangular 50×50 array of pixels (voxels), so $m = 2500$. These pixels are divided into 3 groups: a 'tumor' region, two 'critical' regions, and the other pixels. (The tumor and critical regions are rectangular in this problem instance.) In each of these regions, $d^{\min}$ and $d^{\max}$ are constant within their respective regions. There are a total of $n = 400$ beams. Each beam is a simple line, and A_{ij} is the length of line j in pixel i. (In a real problem, the matrix A is more complicated.) The beams are 20 sets of 20 parallel beams, evenly spaced through the center of the area, with angles varying every 9° from 0° to 171°. (You won't need this information, since we compute the matrix A for you.)

In addition to defining the problem data, the script will create several relevant plots. The first plot simply shows the geometry of the tumor and critical regions. The second plot shows the dosage that results from uniform beam levels, and the third plots show histograms of pixel dosages for the tumor region, the critical regions, and the other pixels, along with the respective lower and upper target dosages. (Hopefully, your design will be much better.)

Check your solution using CVX (which might take a minute or two).

Plot the dosage pattern obtained, and compare it to the one obtained with a uniform beam pattern. Do the same for the dosage histograms.

8.5 *Quantile regression.* For $\alpha \in (0,1)$, define $h_\alpha : \mathbf{R}^n \to \mathbf{R}$ as

$$h_\alpha(x) = \alpha \mathbf{1}^T x_+ + (1-\alpha)\mathbf{1}^T x_-,$$

where $x_+ = \max\{x, 0\}$ and $x_- = \max\{-x, 0\}$, where the maximum is taken elementwise. For the connection between this function and quantiles, see exercise 1.4.

(a) Give a simple expression for the proximal operator of h_α.

(b) The *quantile regression problem* is

$$\text{minimize} \quad h_\alpha(Ax - b),$$

with variable $x \in \mathbf{R}^n$ and parameters $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, and $\alpha \in (0,1)$. Explain how to use ADMM to solve this problem by introducing a new variable (and constraint) $z = Ax - b$. Give the details of each step in ADMM, including how one of the steps can be greatly speeded up after the first step.

(c) Implement your method on data (*i.e.*, A and b) generated as described below, for $\alpha \in \{0.2, 0.5, 0.8\}$. For each of these three values of α, give the optimal objective value, and plot a histogram of the residual vector $Ax - b$. Generate A and b using the following code:

```
m = 2000;
n = 200;
rand('state', 3);
A = rand(m, n);
b = rand(m, 1);
```

Hint. You should develop, debug, and test your code on a smaller problem instance, so you can easily (*i.e.*, quickly) check the results against CVX.

9 Sequential convex programming

9.1 *Minimum eigenvalue via convex-concave procedure.* The (nonconvex) problem

$$\begin{array}{ll} \text{minimize} & x^T P x \\ \text{subject to} & \|x\|_2^2 \geq 1, \end{array}$$

with $P \in \mathbf{S}_+^{n \times n}$, has optimal value $\lambda_{\min}(P)$; x is optimal if and only if it is an eigenvector of P associated with $\lambda_{\min}(P)$. Explain how to use the convex-concave procedure to (try to) solve this problem.

Generate and (possibly) solve a few instances of this problem using the convex-concave procedure, starting from a few (nonzero) initial points. Compare the values found by the convex-concave procedure with the optimal value.

9.2 *Circle packing.* The goal is to place N points in a unit box in $\mathbf{R}^2$ so as to maximize the minimum distance between any of the points:

$$\begin{array}{ll} \text{maximize} & D \\ \text{subject to} & \|x_i - x_j\|_2 \geq D, \quad i \neq j \\ & 0 \preceq x_i \preceq 1, \end{array} \tag{3}$$

with variables $x_1, \ldots, x_N \in \mathbf{R}^2$ and $D \in \mathbf{R}$. This problem, and various variations on it, are sometimes called circle-packing problems, since they come down to placing circles in such a way that they don't intersect (except at the boundaries). You can probably guess what good solutions look like.

(a) *Area bound.* Derive an upper bound $\overline{D}$ on the optimal value, using the following argument. For x_i and D feasible, the N disks centered at x_i with diameter D don't overlap, except on their boundaries; these disks also lie inside the box $\{x \mid -(D/2)\mathbf{1} \preceq x \preceq (1 + D/2)\mathbf{1}\}$. Therefore, the total area of the disks is not more than the area of this box.

(b) *Sequential convex programming.* Explain how to (locally) solve this problem using sequential convex programming.

(c) *Numerical example.* Carry out sequential convex programming for the problem above, with $N = 11$, starting from an initial point with $x_i \neq x_j$ for $i \neq j$. Plot D versus iteration, for a few different starting points (put these all on the same plot). Verify that for different starting points, the algorithm can converge to different objective values. Compare the objective values obtained with the bound $\overline{D}$ found in part (a). Plot the final arrangement of the circles (for one of your starting points) using `draw_circle_packing.m`. For your entertainment, you can draw the circles in each iteration, and watch the evolution of the algorithm. The calling sequence is `draw_circle_packing(x,D,iter)`, where `x` is a $2 \times N$ matrix containing the x_i, `D` is D, and `iter` is the iteration counter (which is displayed in the title).

9.3 *Trajectory optimization with avoidance constraints.* In this problem, you must choose N trajectories (say, of some vehicles) in $\mathbf{R}^n$, which are denoted by $p_i(t) \in \mathbf{R}^n$, $t = 1, \ldots, T$, $i = 1, \ldots, N$. The objective is to minimize

$$J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \|p_i(t+1) - p_i(t)\|_2^2,$$

subject to fixed starting and final positions,

$$p_i(1) = p_i^{\text{start}}, \quad p_i(T) = p_i^{\text{final}}, \quad i = 1, \ldots, N.$$

The solution to the problem stated so far is simple: Each trajectory follows a straight line from the starting position to the final position, at uniform speed. Here is the wrinkle: We have *avoidance constraints*, of the form

$$\|p_i(t) - p_j(t)\|_2 \geq D, \quad i \neq j, \quad t = 2, \ldots, T-1.$$

(Thus, the vehicles must maintain a given distance D from each other at all times.) These last constraints are obviously not convex.

(a) Explain how to use the convex-concave procedure to (approximately, locally) solve the problem.

(b) Implement the method for the problem with data given in `traj_avoid_data.m`, which involves three vehicles moving in $\mathbf{R}^2$. Executing this file runs a movie showing the vehicle trajectories when they move in straight lines at uniform speed, with a circle around each vehicle of diameter D. You can use this code to visualize the trajectories you obtain as your algorithm runs.

You should start the convex-concave procedure from several different initial trajectories for which $p_i(t) \neq p_j(t)$, $t = 1, \ldots, T$. For example, you can simply take $p_i(t) \sim \mathcal{N}(0, I)$. Plot the objective J versus iteration number for a few different initial trajectories (on the same plot). Verify that the avoidance constraints are satisfied after the first iteration.

For the best set of final trajectories you find, plot the minimum inter-vehicle distance, $\min_{i \neq j} \|p_i(t) - p_j(t)\|_2$, versus time. Plot the trajectories in $\mathbf{R}^2$. And of course, for your own amusement, view the movie.

Hint. Don't try to write elegant code that handles the case of general N. We've chosen $N = 3$ so you can just name the trajectories ($2 \times T$ matrices) `p1`, `p2`, and `p3`, and explicitly write out the three (*i.e.*, $N(N-1)/2$) avoidance constraints.

10 Conjugate-gradient and truncated Newton methods

10.1 *Conjugate gradient residuals.* Let $r^{(k)} = b - Ax^{(k)}$ be the residual associated with the kth element of the Krylov sequence. Show that $r^{(j)T} r^{(k)} = 0$ for $j \neq k$. In other words, the Krylov sequence residuals are mutually orthogonal. Do not use the explicit algorithm to show this property; use the basic definition of the Krylov sequence, *i.e.*, $x^{(k)}$ minimizes $(1/2)x^T Ax - b^T x$ over $\mathcal{K}_k$.

10.2 *CG and PCG example.* In this problem you explore a variety of methods to solve $Ax = b$, where $A \in \mathbf{S}^n_{++}$ has block diagonal plus sparse structure: $A = A_{\text{blk}} + A_{\text{sp}}$, where $A_{\text{blk}} \in \mathbf{S}^n_{++}$ is block diagonal and $A_{\text{sp}} \in \mathbf{S}^n$ is sparse. For simplicity we assume A_{blk} consists of k blocks of size m, so $n = mk$. The matrix A_{sp} has N nonzero elements.

(a) What is the approximate flop count for solving $Ax = b$ if we treat A as dense?

(b) What is the approximate flop count for an iteration of CG? (Assume multiplication by A_{blk} and A_{sp} are done exploiting their respective structures.) You can ignore the handful of inner products that need to be computed.

(c) Now suppose that we use PCG, with preconditioner $M = A_{\text{blk}}^{-1}$. What is the approximate flop count for computing the Cholesky factorization of A_{blk}? What is the approximate flop count per iteration of PCG, once a Cholesky factorization of A_{blk} if found?

(d) Now consider the specific problem with A_{blk}, A, and b generated by `ex_blockprecond_data.m`. Solve the problem using direct methods, treating A as dense, and also treating A as sparse. Run CG on the problem for a hundred iterations or so, and plot the relative residual versus iteration number. Run PCG on the same problem, using the block-diagonal preconditioner $M = A_{\text{blk}}^{-1}$. Give the solution times for dense direct, sparse direct, CG (to relative residual 10^{-4}, say), and PCG (to relative residual 10^{-8}, say). For PCG break out the time as time for initial Cholesky factorization, and time for PCG iterations.

Hints.

- To force Matlab to treat A as dense, use `full(A)`.
- You do not need to implement the conjugate gradient algorithm; instead use the `pcg` function in Matlab.
- To block precondition with $M = A_{\text{blk}}^{-1}$, first find the Cholesky factorization of A_{blk}, *i.e.*, lower triangular L with $LL^T = A_{\text{blk}}$. The Matlab code to implement block preconditioning is

```
L = chol(A_blk)';
[...] = pcg(A,b,tolerance,MAXITER,L,L');
```

Matlab uses a sparse Cholesky to factor A_{blk}, which is less efficient than using a dense Cholesky factorization on each block separately, but it is efficient enough to make the point.

10.3 *Search directions in the truncated Newton method.* Suppose we use CG to compute an approximate solution of the Newton system $Hv + g = 0$. (Here $H = \nabla^2 f(x) \succ 0$ and $g = \nabla f(x) \neq 0$.)

(a) Suppose we start CG from $v^{(0)} = 0$. Verify that $v^{(1)}$ has the form $-\alpha g$ for some $\alpha > 0$. (Thus, after one step, the search direction is in the direction of the negative gradient.) Verify that $(1/2)v^{(1)T}Hv^{(1)} + g^T v^{(1)} < 0$. Explain why this implies that $g^T v^{(k)} < 0$ for all $k \geq 1$, *i.e.*, all iterates of the CG iteration are descent directions.

(b) Suppose that z is a descent direction, *i.e.*, $g^T z < 0$. Find $\alpha^\star > 0$ that minimizes $(1/2)(\alpha z)^T H(\alpha z) + g^T(\alpha z)$, and verify that the resulting value is negative. Show that if we start the CG process with $v^{(0)} = \alpha^\star z$, all CG iterates are descent directions, *i.e.*, $g^T v^{(k)} < 0$ for all $k \geq 1$.

Hint. Use the fact that, for any starting point $v^{(0)}$, the sequence $(1/2)v^{(k)T}Hv^{(k)} + g^T v^{(k)}$, $k = 0, 1, \ldots$, is nonincreasing.

10.4 *Randomized preconditioners for conjugate gradient methods.* In this question, we explore the use of some randomization methods for solving overdetermined least-squares problems, focusing on conjugate gradient methods. Letting $A \in \mathbf{R}^{m\times n}$ be a matrix (we assume that $m \gg n$) and $b \in \mathbf{R}^m$, we wish to minimize

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}(a_i^T x - b_i)^2,$$

where the $a_i \in \mathbf{R}^n$ denote the rows of A.

Given $m \in \{2^i, i = 1, 2, \ldots\}$, the (unnormalized) Hadamard matrix of order m is defined recursively as

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad H_m = \begin{bmatrix} H_{m/2} & H_{m/2} \\ H_{m/2} & -H_{m/2} \end{bmatrix}.$$

The associated normalized Hadamard matrix is given by $H_m^{(\text{norm})} = H_m/\sqrt{m}$, which evidently satisfies $H_m^{(\text{norm})^T} H_m^{(\text{norm})} = I_{m\times m}$. Moreover, via a recursive algorithm it is possible to compute $H_m x$ in time $O(m \log m)$, which is much faster than m^2 for a general matrix.

To solve the least squares minimization problem using conjugate gradients, we must solve $A^T Ax = A^T b$. In class, we discussed that using a *preconditioner* M such that $M \approx A^{-1}$ can give substantial speedup in computing solutions to large problems. Consider the following scheme to generate a randomized preconditioner, assuming that $m = 2^i$ for some i:

i. Let $S = \mathbf{diag}(S_{11}, \ldots, S_{mm})$, where S_{jj} are random $\{-1, +1\}$ signs

ii. Let $p \in \mathbf{Z}_+$ be a small positive integer, say 20 for this problem.

iii. Let $R \in \{0,1\}^{n+p \times m}$ be a *row selection matrix*, meaning that each row of R has only 1 non-zero entry, chosen uniformly at random. (The location of these non-zero columns is distinct.)[3]

iv. Define $\Phi = RH_m^{(\text{norm})} S \in \mathbf{R}^{n+p \times m}$

We then define the matrix M via its inverse $M^{-1} = A^T \Phi^T \Phi A \in \mathbf{R}^{n \times n}$.

(a) How many FLOPs (floating point operations) are required to compute the matrices M^{-1} and M, respectively, assuming that you can compute the matrix-vector product $H_m v$ in time $m \log m$ for any vector $v \in \mathbf{R}^m$?

(b) How many FLOPs are required to naïvely compute $A^T A$, assuming A is dense (using standard matrix algorithms)?

(c) How many FLOPs are required to compute $A^T A v$ for a vector $v \in \mathbf{R}^n$ by first computing $u = Av$ and then computing $A^T u$?

(d) Suppose that conjugate gradients runs for k iterations. Using the preconditioned conjugate gradient algorithm with $M = (A^T \Phi^T \Phi A)^{-1}$, how many total floating point operations have been performed? How many would be required to directly solve $A^T A x = A^T b$? How large must k be to make the conjugate gradient method slower?

(e) Implement the conjugate gradient algorithm for solving the positive definite linear system $A^T A x = A^T b$ both with and without the preconditioner M. To generate data for your problem, set $m = 2^{12}$ and $n = 400$, then generate the matrix A by setting `A = randn(m, n) * spdiags(linspace(.001, 100, n))` (in Matlab) and `A = randn(m, n) * spdiagm(linspace(.001, 100, n))` (in Julia), and let `b = randn(m, 1)`. For simplicity in implementation, you may directly pass $A^T A$ and $A^T b$ into your conjugate gradient solver, as we only wish to explore how the methods work. (In Matlab, the `pcg` method may be useful.)

Plot the norm of the residual $r^k = A^T b - A^T A x^k$ (relative to $\left\|A^T b\right\|_2$) as a function of iteration k for each of your conjugate gradient procedures. Additionally, compute and print the condition numbers $\kappa(A^T A)$ and $\kappa(M^{1/2} A^T A M^{1/2})$.

Include your code.

[3] *Hint.* To do this in Matlab, generate a random permutation `inds = randperm(m)`, then set `R = sparse(1:(n+p), inds(1:(n+p)), ones(n+p,1)), n+p, m)`, in Julia, set `R = sparse(1:(n+p), inds[1:(n+p)], ones(n+p), n+p, m)`.

11 ℓ_1 methods for convex-cardinality problems

11.1 *Sum-of-norms heuristic for block sparsity.* The basic ℓ_1 heuristic for finding a sparse vector x in a convex set $\mathcal{C}$ is to minimize $\|x\|_1$ over $\mathcal{C}$. We are given a partition of x into subvectors: $x = (x^{(1)}, \ldots, x^{(k)})$, with $x^{(i)} \in \mathbf{R}^{n_i}$. Our goal is to find an $x \in \mathcal{C}$ with the fewest number of subvectors $x^{(i)}$ nonzero. Like the basic sparsity problem, this problem is in general difficult to solve. But a variation on the ℓ_1 heuristic can work well to give a block sparse, if not the block sparsest, vector. The heuristic is to minimize

$$\sum_{i=1}^{k} \|x^{(i)}\|$$

over $x \in \mathcal{C}$.

(a) What happens if the norms above are ℓ_1 norms? Would you expect x to be sparse? Block sparse? (Your answer can be very brief.)

(b) Generate a nonempty polyhedron $\{y \mid Ay \preceq b\}$ using

```
randn('state',0);
m = 50; n = 100;
A = randn(m,n); b = randn(m,1);
```

We will divide x into 20 subvectors, each of length 5. Use the heuristic above, with ℓ_1, ℓ_2, and ℓ_∞ norms on the subvectors, to find block sparse x in your polyhedron. *Hints.* You may find the functions `norms` and `reshape` useful.

11.2 *Sparse Bayes network identification.* Suppose you are given samples $y_1, \ldots, y_N \in \mathbf{R}^n$ from an $\mathcal{N}(0, \Sigma)$ distribution, where $\Sigma \succ 0$. (See page 355 of the textbook.) From these samples you need to estimate the parameter Σ. You'd like the off-diagonals of Σ^{-1} to be sparse, if possible. (A sparse Σ^{-1} corresponds to a compact Bayes network representation, since $(\Sigma^{-1})_{ij} = 0$ means that x_i and x_j are conditionally independent, given the other x_k's.)

(a) Suggest a simple method for finding Σ with the off-diagonals of Σ^{-1} sparse and with the likelihood of Σ nearly maximum. Your method should involve a parameter $\lambda \geq 0$ that adjusts the trade-off between suboptimality in likelihood, and sparsity of Σ^{-1}.

(b) Carry out your method on the data found in `sp_bayesnet_data.m`. The true inverse covariance, Σ^{-1}, is stored in the matrix `Siginv`, and the samples $y_1, \ldots, y_N$ are stored in the matrix `Ys`. (You may need to experiment with different values of λ.) Plot the sparsity patterns of Σ^{-1} and your reconstruction $\hat{\Sigma}^{-1}$.

11.3 *Rank minimization via the nuclear norm.* A simple heuristic for finding a matrix of low (if not minimum) rank, in a convex set, is to minimize its nuclear norm, *i.e.*, the sum

of its singular values, over the convex set. Test this method on a numerical example, obtained by generating data matrices $A_0, \ldots, A_n \in \mathbf{R}^{p\times q}$ and attempting to minimize the rank of $A_0 + x_1A_1 + \cdots + x_nA_n$. You can use $n = 50$, $p = 10$, $q = 10$.

11.4 *Portfolio investment with linear and fixed costs.* The goal is to optimally invest an initial amount B in a portfolio of n assests. Let x_i be the amount (in dollars) of asset i that we purchase. We assume that no short selling is allowed, *i.e.*, $x \succeq 0$. We pay a fixed cost β for each asset we invest in, *i.e.*, for each asset with $x_i > 0$. We also pay a fee that is proportional to the amount purchased, given by $\alpha^T x$, where α_i is the fee (rate) associated with asset i. The budget constraint can be expressed as

$$\mathbf{1}^T x + \beta\,\mathbf{card}(x) + \alpha^T x \leq B.$$

The mean return on the portfolio is given by $\mu^T x$, where μ_i is the mean return of asset i, and the variance of the portfolio is $x^T\Sigma x$, where Σ is the covariance of the price changes. Our goal is to minimize the portfolio standard deviation, subject to meeting a minimum mean return requirement. This can be expressed as the problem

$$\begin{array}{ll} \text{minimize} & (x^T\Sigma x)^{1/2} \\ \text{subject to} & \mu^T x \geq R_{\min}, \qquad x \succeq 0 \\ & \mathbf{1}^T x + \beta\,\mathbf{card}(x) + \alpha^T x \leq B. \end{array} \tag{4}$$

This is a convex-cardinality problem. Once the choice is made of which assets to invest in, the problem becomes convex. But there are 2^n possible subsets of assets to invest in, so exhaustive search over all of these is not practical for $n \geq 20$.

You will focus on a particular instance of this problem, with data given in the file `l1_heuristic_portfolio_data.m` on the class web site.

(a) *Heuristic portfolio investment.* You will use the following ℓ_1-norm based heuristic to approximately solve the portfolio investment problem. First we replace $\mathbf{card}(x)$ with $\gamma\mathbf{1}^T x$ (which is the same as $\gamma\|x\|_1$, since $x \succeq 0$), where $\gamma \geq 0$ is a parameter, to get the problem

$$\begin{array}{ll} \text{minimize} & (x^T\Sigma x)^{1/2} \\ \text{subject to} & \mu^T x \geq R_{\min}, \qquad x \succeq 0 \\ & \mathbf{1}^T x + \beta\gamma\mathbf{1}^T x + \alpha^T x \leq B. \end{array} \tag{5}$$

Solve (5) for, say, 50 values of γ in $[0, 25]$. Note that when you solve the problem for $\gamma = 0$, you are just ignoring the fixed investment costs. The solutions of (5) need not be feasible for the portfolio investment problem (4), since the true budget constraint can be violated.

For each solution of (5), note the sparsity pattern of x. Fix this sparsity pattern (which makes $\mathbf{card}(x)$ constant), and solve the problem (4). This procedure is called *polishing*.

Plot the portfolio standard deviation obtained (after polishing) versus γ. What is the minimum standard deviation, $\sigma_{\min}$, that you obtain? Give the best portfolio found (*i.e.*, the assets purchased, and the amounts for each one). Also plot $\mathbf{card}(x)$, *i.e.*, the number of assets invested in, versus γ.

Hint. To determine the sparsity pattern of x after solving (5), you'll need to use a reasonable (positive) threshold to determine if $x_i = 0$, as in `find(x<1e-3)`.

(b) *A lower bound.* For some values of γ the optimal value of (5) is a lower bound on the optimal value of the original portfolio investment problem (4). Find a simple value $\tilde{\gamma}$ of γ for which this is the case. Compute the corresponding lower bound and compare it to the standard deviation found in (a).

(c) *A more sophisticated lower bound.* Let $x^\star$ denote an optimal point for the original portfolio investment problem (4). Suppose $x_i^\star \leq u_i$ for $i = 1, \ldots, n$. Explain why the optimal value of the problem

$$\begin{array}{ll} \text{minimize} & (x^T \Sigma x)^{1/2} \\ \text{subject to} & \mu^T x \geq R_{\min}, \qquad x \succeq 0 \\ & \mathbf{1}^T x + \beta v^T x + \alpha^T x \leq B, \end{array}$$

where $v_i = 1/u_i$, gives a lower bound on the optimal value of the original portfolio investment problem (4).

A simple choice for u_i is $u_i = B$. Better (*i.e.*, smaller) values can be found as the optimal values of the problems

$$\begin{array}{ll} \text{maximize} & x_i \\ \text{subject to} & \mu^T x \geq R_{\min}, \qquad x \succeq 0 \\ & (x^T \Sigma x)^{1/2} \leq \sigma_{\min} \\ & \mathbf{1}^T x + \beta \tilde{\gamma} \mathbf{1}^T x + \alpha^T x \leq B, \end{array}$$

where $\sigma_{\min}$ is the standard deviation of the portfolio found in part (a). Justify this.

Carry out this procedure for the given problem instance, and compare the resulting lower bound to the results from parts (a) and (b).

12 Optimization with uncertain data

12.1 *Robust geometric program with norm-based uncertainty.* We consider a geometric program in variables $x \in \mathbf{R}^n_+$, with problem data $a \in \mathbf{R}^k_+$, data matrix $B \in \mathbf{R}^{m\times k}$, and exponent vectors $\alpha_i \in \mathbf{R}^n$ and $\beta_{ij} \in \mathbf{R}^n$. For shorthand, we define $x^\alpha = \prod_{j=1}^n x_j^{\alpha_j}$ for vectors $\alpha \in \mathbf{R}^n$. Consider the problem

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^k a_i x^{\alpha_i} \\ &\text{subject to} && \sum_{j=1}^k B_{ij} x^{\beta_{ij}} \leq 1 \text{ for } i = 1, \ldots, m. \end{aligned}$$

Now assume that we have norm-based data uncertainty in our multipliers a_i and B_{ij}, that is, there exist constants $\delta \geq 0$ and $\epsilon \geq 0$ where all we know is that the data could be $a_i \pm \Delta_i$ for a vector $\Delta \in \mathbf{R}^n$ such that $\|\Delta\|_p \leq \delta$, and similarly for the rows of B. We would like to solve the robust formulation

$$\begin{aligned} &\text{minimize} && \sup_{\|\Delta\|_p \leq \delta} \sum_{i=1}^k (a_i + \Delta_i) x^{\alpha_i} \\ &\text{subject to} && \sum_{j=1}^k (B_{ij} + e_j) x^{\beta_{ij}} \leq 1 \text{ for } i = 1, \ldots, m \text{ and all } e \in \mathbf{R}^k \text{ s.t. } \|e\|_p \leq \epsilon. \end{aligned}$$

Here $p \in [1, \infty]$ generate the norms for the uncertainty set. Show how to formulate the robust problem above as a geometric problem.

12.2 *Robust facility locations.* In the facilities placement problem, we would like to choose locations for n facilities, denoted $x_1, \ldots, x_n \in \mathbf{R}^2$, based on the (fixed) locations of m sources of resources $y_1, \ldots, y_m \in \mathbf{R}^2$. We assume that associated with each source/facility pair (y_i, x_j) there is a cost $A_{ij} \geq 0$ that we must pay per unit distance to transport the goods from source y_i to facility x_j. In addition, the facilities must transport goods among themselves, where the cost for transporting per unit distance between x_i and x_j is given by $B_{ij} \geq 0$. Our problem, then, is to minimize the cost of a facility location assignment based on the data $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{n\times n}$, which gives the SOCP

$$\text{minimize} \sum_{i=1}^m \sum_{j=1}^n A_{ij} \|y_i - x_j\|_2 + \sum_{i=1}^n \sum_{j=1}^n B_{ij} \|x_i - x_j\|_2$$

in variables $x_1, \ldots, x_n \in \mathbf{R}^2$.

In some situations, the costs per unit transport from source y_i to facility x_j may be not known precisely (or, perhaps more realistically, the exact amount necessary may not

be known until after the locations have been chosen). In particular, assume that for the pairs i, j such that $A_{ij} > 0$ (there is assumed to be *no* uncertainty in pairs i, j such that $A_{ij} = 0$), the A_{ij} are subject to limited price increases $\Delta_{ij} \geq 0$ with maximum increase $\delta > 0$, and for which the total (summed) increase in cost is bounded by $k\delta$, meaning $\sum_{ij} \Delta_{ij} \leq k\delta$.

(a) Assuming that the worst allowable values of $A_{ij} + \Delta_{ij}$ are realized, Show how to formulate the problem with uncertainty in prices A_{ij} as a (robust) convex optimization problem.

(b) Show how to rewrite your formulation from (a) as a second order cone problem (SOCP).

(c) Using the data in `facilities_placement_helper.jl` or `FacilitiesData.m`, solve the non-robust formulation of the facilities placement problem. Plot the resulting facility locations (using the method `PlotFacilities`, available in `facilities_placement_helper.` or `PlotFacilities.m`). Now, with robustness parameters $k = 4$ and $\delta = .005$ and $\delta = .01$, solve your robust formulation. Plot the resulting facility locations. Can you give, in one or two brief sentences, intuition for any differences you see?

12.3 *Sharpest convex bounds on probabilistic constraints.* Let A be some statement and $1(A)$ be 1 if A is true, 0 otherwise.

(a) Show that for any non-negative convex function $\phi : \mathbf{R} \to \mathbf{R}_+$ satisfying $\phi(x) \geq 1(x \leq 0)$, there exists some $\alpha \in [0, \infty]$ such that $\phi(x) \geq [1 - \alpha^{-1}x]_+ \geq 1(x \leq 0)$ for all $x \in \mathbf{R}$.

(b) Let U be any random variable and $f(x, u)$ an arbitrary function. Show that for any convex function $\phi : \mathbf{R} \to \mathbf{R}_+$ satisfying $\phi(z) \geq 1(z \geq 0)$, there exists $\alpha \in [0, \infty]$ such that

$$\mathbf{Prob}\left(f(x, U) \geq t\right) \leq \mathbf{E}\left[1 + \frac{f(x, U) - t}{\alpha}\right]_+ \leq \mathbf{E}\,\phi(f(x, U) - t).$$

12.4 *Portfolio optimization with chance constraints* Consider a portfolio with n assets described by random return vector $R \in \mathbf{R}^n$, where R_j denotes the relative growth of asset j, with mean returns $\mu \in \mathbf{R}^n_+$ and covariance matrix $\Sigma \in \mathbf{S}^n_+$ between the returns. Let the vector $x \in \mathbf{R}^n_+$ describe the overall proportional portfolio allocation across the set of assets, with $\mathbf{1}^T x = 1$. We would like to maximize our profit while maintaining low risk, that is, for fixed $\epsilon \in (0, 1)$, to guarantee that with probability at least $1 - \epsilon$ we achieve some fixed level of profit. We cast this as the (non-convex) optimization problem

$$\begin{array}{ll} \text{maximize} & t \\ \text{subject to} & \mathbf{Prob}\left(R^T x \leq t\right) \leq \epsilon,\ \mathbf{1}^T x = 1,\ x \succeq 0. \end{array} \tag{6}$$

Solving this problem is hard in general, so we investigate methods to find allocations x that guarantee lower bounds for this problem when we assume that the data R come from a multi-variate normal distribution with mean μ and covariance matrix Σ.

As in the lecture notes, let ϕ be a non-decreasing convex function function $\phi : \mathbf{R} \to \mathbf{R}_+$ with $\phi(0) = 1$. For any such ϕ, random variable Z, and positive α, we know that

$$\mathbf{Prob}\,(Z \geq 0) = \mathbf{E}\,\mathbb{I}(Z \geq 0) \leq \mathbf{E}\,\phi\,(Z/\alpha)\,.$$

In particular, for any $\alpha > 0$, the constraint $\mathbf{E}\,\phi(\alpha^{-1}Z) \leq \epsilon$ is a *safe* approximation to the chance constraint in problem (6), meaning it guarantees $\mathbf{Prob}\,(Z \geq 0) \leq \epsilon$.

(a) Let $\phi(w) = \exp(w)$. Use this ϕ to show that we can generate lower bounds to problem (6) by solving the following problem

$$\begin{array}{ll} \text{maximize} & \mu^T x - \sqrt{2\log\frac{1}{\epsilon}}\,\|x\|_\Sigma \\ \text{subject to} & \mathbf{1}^T x = 1,\ x \succeq 0, \end{array} \tag{7}$$

where $\|x\|_\Sigma^2 = x^T\Sigma x$ is the Mahalanobis norm generated by $\Sigma \in \mathbf{S}_+^n$.

(b) Let $\phi(w) = (1 + w)_+$. Use this ϕ to show that we can generate lower bounds to (6) by solving the following problem

$$\begin{array}{ll} \text{maximize} & t \\ \text{subject to} & \mathbf{E}[(t - R^T x + \alpha)_+] - \alpha\epsilon \leq 0, \\ & \mathbf{1}^T x = 1,\ x \succeq 0. \end{array} \tag{8}$$

(c) Evaluate both lower bounds found in parts (a) and (b) for the problem data (mean μ and covariance Σ) given in `robust_portfolio_data.[jl|m]`. There is an analytic formula for $\mathbf{E}(t - R^T x + \alpha)_+$, but instead we use a Monte Carlo approximation to it to solve problem (8): generate 10^3 sample returns $R \sim \mathcal{N}(\mu, \Sigma)$, and use the empirical expectation to approximately solve the optimization problem (8). Solve parts (a) and (b) using each of $\epsilon = .1, .05, .01, .005$. For each ϵ, report the optimal value achieved for the problems (7) and (8) and the variance $x^{\star T}\Sigma x^\star$ for the resulting solution $x^\star$. Additionally, generate a new set of 10^4 returns R and evaluate $x^T R$ for each return, and plot histograms of the different return values. Give a simple (two sentence) explanation of your results.

13 Model predictive control

13.1 *MPC for output tracking.* We consider the linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \quad t = 0, \ldots, T-1,$$

with state $x(t) \in \mathbf{R}^n$, input $u(t) \in \mathbf{R}^m$, and output $y(t) \in \mathbf{R}^p$. The matrices A and B are known, and $x(0) = 0$. The goal is to choose the input sequence $u(0), \ldots, u(T-1)$ to minimize the output tracking cost

$$J = \sum_{t=1}^{T} \|y(t) - y_{\text{des}}(t)\|_2^2,$$

subject to $\|u(t)\|_\infty \leq U^{\max}$, $t = 0, \ldots, T-1$.

In the remainder of this problem, we will work with the specific problem instance with data

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix},$$

$T = 100$, and $U^{\max} = 0.1$. The desired output trajectory is given by

$$y_{\text{des}}(t) = \begin{cases} 0 & t < 30, \\ 10 & 30 \leq t < 70, \\ 0 & t \geq 70. \end{cases}$$

(a) Find the optimal input $u^\star$, and the associated optimal cost $J^\star$.

(b) *Rolling look-ahead.* Now consider the input obtained using an MPC-like method: At time t, we find the values $u(t), \ldots, u(t+N-1)$ that minimize

$$\sum_{\tau=t+1}^{t+N} \|y(\tau) - y_{\text{des}}(\tau)\|_2^2,$$

subject to $\|u(\tau)\|_\infty \leq U^{\max}$, $\tau = t, \ldots, t+N-1$, and the state dynamics, with $x(t)$ fixed at its current value. We then use $u(t)$ as the input to the system. (This is an informal description, but you can figure out what we mean.)

In a tracking context, we call N the amount of *look-ahead*, since it tells us how much of the future of the desired output signal we are allowed to access when we decide on the current input.

Find the input signal for look-ahead values $N = 8$, $N = 10$, and $N = 12$. Compare the cost J obtained in these three cases to the optimal cost $J^\star$ found in part (a).

Plot the output $y(t)$ for $N = 8$, $N = 10$, and $N = 12$.

14 Branch and bound

14.1 *Branch and bound for partitioning.* We consider the two-way partitioning problem (see pages 219, 226, and 285 in the book),

$$\begin{array}{ll} \text{minimize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array}$$

We can, without any loss of generality, assume that $x_1 = 1$.

You will perform several iterations of branch and bound for a random instance of this problem, with, say, $n = 100$.

To run branch and bound, you'll need to find lower and upper bounds on the optimal value of the partitioning problem, with some of the variables fixed to given values, *i.e.*,

$$\begin{array}{ll} \text{minimize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n \\ & x_i = x_i^{\text{fixed}}, \quad i \in \mathcal{F}, \end{array}$$

where $\mathcal{F} \subseteq \{1, \ldots, n\}$ is the set of indices of the fixed entries of x, and $x_i^{\text{fixed}} \in \{-1, 1\}$ are the associated values.

Lower bound. To get a lower bound, you will solve the SDP

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(WX) \\ \text{subject to} & X_{ii} = 1, \quad i = 1, \ldots, n \\ & \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0 \\ & x_i = x_i^{\text{fixed}}, \quad i \in \mathcal{F}, \end{array}$$

with variables $X \in \mathbf{S}^n$, $x \in \mathbf{R}^n$. (If we add the constraint that the $(n+1) \times (n+1)$ matrix above is rank one, then $X = xx^T$, and this problem gives the exact solution.)

Upper bound. To get an upper bound we'll find a feasible point $\hat{x}$, and evaluate its objective. To do this, we solve the SDP relaxation above, and find an eigenvector v associated with the largest eigenvalue of the $(n+1) \times (n+1)$ matrix appearing in the inequality. Choose $\hat{x}_i = \mathbf{sign}(v_{n+1})\mathbf{sign}(v_i)$ for $i = 1, \ldots, n$.

Branch and bound. Develop a partial binary tree for this problem, as on page 21 of the lecture slides. Use $n = 100$. At each node, record upper and lower bounds. Along each branch, indicate on which variable you split. At each step, develop the node with the smallest lower bound. Develop four nodes.

You can do this 'by hand'; you do not need to write a complete branch and bound code in Matlab (which would not be a pretty sight). You can use CVX to solve the SDPs, and manually constrain some variables to fixed values.

15 Semidefinite relaxations and nonconvex problems

15.1 *Semidefinite representations of nuclear norms* Recall that for a matrix $A \in \mathbf{R}^{m\times n}$ with singular value decomposition $A = U\Sigma V^T$, the nuclear norm $\|A\|_* = \sum_i \sigma_i(A)$ is the sum of singular values of A. Give a semidefinite representation of $\|A\|_*$, that is, write

$$\|A\|_* = \inf_X \{\mathbf{Tr}(CX) \mid X \succeq 0, \mathrm{Lin}(A, X) \succeq 0\}$$

where X is a positive semidefinite matrix (whose size you must specify) and Lin is a linear function in its arguments that you must specify. *Hint:* consider the matrix

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix},$$

where the 0 values are appropriately sized.

15.2 *Approximation of sets by lifts and randomization.* In this question, we explore the possibilities for constructing random vectors that satisfy various (convex or non-convex) quadratic equalities with high probability.

Let $\mathcal{X}_+$ be a *semidefinite lift* of $\mathcal{X}$, meaning that it is compact, has non-empty relative interior, and satisfies

$$\mathcal{X}_+ \supset \mathbf{SHull}(\mathcal{X}) := \mathbf{Co}\{xx^T \mid x \in \mathcal{X}\}.$$

For a collection of matrices $\mathcal{C} \subset \mathbf{S}^n$, we say that $\mathcal{X}_+$ *approximates* $\mathcal{X}$ *over* $\mathcal{C}$ if (i) there exists a constant $\nu < \infty$ such that for all $C \in \mathcal{C}$,

$$\sup_{x\in\mathcal{X}} x^T C x \le \sup_{X\in\mathcal{X}_+} \mathbf{Tr}(CX) \le \nu \sup_{x\in\mathcal{X}} x^T C x \tag{9}$$

and for any matrix $X \in \mathcal{X}_+$, for $t \ge 0$ we have

$$\mathbf{Prob}\left(ZZ^T \notin t \cdot \mathcal{X}_+\right) \le K \exp\left(-ct\right) \quad \text{where } Z \sim \mathcal{N}(0, X). \tag{10}$$

for constants $K, c \in \mathbf{R}_{++}$. As in the lecture on semidefinite relaxations, this means that for $X \in \mathcal{X}_+$, by drawing from a Gaussian distribution we are likely to get a vector Z "near enough" $\mathcal{X}$.

A standard result, which is useful for showing both conditions (9) and (10), is the following. Let $Z \sim \mathcal{N}(0, I)$. Then for any matrix A,

$$\mathbf{Prob}(\|AZ\|_2^2 \ge \mathbf{Tr}(A^T A)(1 + 2\sqrt{t} + 2t)) \le e^{-t} \tag{11}$$

for all $t \ge 0$.

(a) Suppose $\mathcal{X} = \{x \in \mathbf{R}^n \mid \|x\|_2 = 1\}$. Show that $\mathcal{X}_+ = \{X \in \mathbf{S}^n \mid X \succeq 0, \mathbf{Tr}(X) = 1\}$ satisfies the approximation condition (9) with $\nu = 1$ for all $\mathcal{C} \subset \mathbf{S}^n$.

We say that a set $\mathcal{X}$ is *ellipse-like* if it has representation

$$\mathcal{X} = \left\{x \in \mathbf{R}^n \mid x^T S_k x \leq 1, k = 1, \ldots, m\right\}$$

where the matrices $S_k \in \mathbf{S}^n_+$, not necessarily positive definite, with $\sum_k S_k \succ 0$.

(b) Let $\mathcal{X}$ be ellipse-like and define the set

$$\mathcal{X}_+ := \left\{X \in \mathbf{S}^n_+ \mid \mathbf{Tr}(S_k X) \leq 1, k = 1, \ldots, m\right\}. \tag{12}$$

Show that $\mathcal{X}_+$ is compact, convex, and has non-empty interior.

(c) Show that if $X \in \mathcal{X}_+$ as in Eq. (12), then for $Z \sim \mathcal{N}(0, X)$, we have

$$\mathbf{Prob}\left(\text{there is } k \in \{1, \ldots, m\} \text{ s.t. } Z^T S_k Z \geq \mathbf{Tr}(S_k X) \cdot (1 + t)\right) \leq K e^{-ct}$$

for all $t \geq 0$, where you should specify K, c. Argue that this implies inequality (10).

(d) Suppose $\mathcal{X}$ is ellipse-like and $\mathcal{X}_+$ is as in Eq. (12) and assume $\mathcal{X}_+$ satisfies inequality (10). Show the approximation guarantees

$$\sup_{x \in \mathcal{X}} x^T C x \leq \sup_{X \in \mathcal{X}_+} \mathbf{Tr}(CX) \leq \inf_{t \geq 0} \left\{ t \sup_{x \in \mathcal{X}} x^T C x + \frac{m}{\lambda_{\min}} \sqrt{2 \left\|C\right\|_{\mathrm{Fr}}^2 + \left\|C\right\|_*^2} \cdot \sqrt{K} e^{-ct/2} \right\}$$

where $\|\cdot\|_*$ is the nuclear norm and $\lambda_{\min} = \lambda_{\min}(\sum_k S_k)$ is the minimal eigenvalue of $\sum_k S_k$. If in addition $C \succeq 0$ and $\mathcal{X}$ includes the sphere $\{x : \|x\|_2 = 1\}$ give an upper bound of the form (9) where ν depends only logarithmically on m, n, and $\lambda_{\min}$.

15.3 *Near optimal linear estimators.* Let $x \in \mathcal{X} \subset \mathbf{R}^n$ and $A \in \mathbf{R}^{m \times n}$, and consider the problem of estimating a vector x given a noisy observation of the form

$$y = Ax + \xi$$

for $\xi \sim \mathcal{N}(0, Q)$, the normal distribution with covariance Q. We measure the quality of an estimator $\hat{x}$ by its *worst-case risk*, which is the supremum over all $x \in \mathcal{X}$ of its expected loss, that is,

$$\sup_{x \in \mathcal{X}} \mathbf{E}[\|x - \hat{x}(Ax + \xi)\|^2],$$

where $\|\cdot\|$ is the ℓ_2-norm. The *linear* optimal risk is the best worst-case risk of a linear estimator of x based on the observation $Ax + \xi$.

$$\mathrm{Risk}_{\mathsf{Lin}}(\mathcal{X}, Q) := \inf_H \sup_{x \in \mathcal{X}} \mathbf{E}\left[\|H(Ax + \xi) - x\|^2\right],$$

where $\xi \sim \mathcal{N}(0, Q)$.

(a) Assume that the set $\mathcal{X}_+ \subset \mathbf{S}_+^n$ is a semidefinite lift of $\mathcal{X}$ and approximates $\mathcal{X}$ as in the conditions (9) and (10). Give the tightest upper bound you can on

$$\sup_{x \in \mathcal{X}} \mathbf{E}\left[\|H(Ax+\xi) - x\|^2\right],$$

writing the result as a semidefinite problem (where you may treat the constraint $X \in \mathcal{X}_+$ as an abstract semidefinite constraint).

Using your formulation, write an optimization problem for finding a (near) optimal H as in the definition of $\text{Risk}_{\mathsf{Lin}}$.

(b) Argue that if $\mathcal{X} = \mathbf{R}^n$ and $Q \succ 0$, then the optimal $H = (A^TQ^{-1}A)^{-1}A^TQ^{-1}$.

(c) We now consider the set of box constraints

$$\mathcal{X} = \{x \in \mathbf{R}^n \mid \|x\|_\infty \leq 1\}.$$

Show how to write this constraint in the ellipse-like form of question 15.2, that is, as $\cap_k\{x \in \mathbf{R}^n \mid x^TS_kx \leq 1\}$ for matrices $S_k \succeq 0$ with $\sum_k S_k \succ 0$. What is the set $\mathcal{X}_+ = \cap_k\{X \in \mathbf{S}_+^n \mid \mathbf{Tr}(XS_k) \leq 1\}$?

(d) Generate a matrix $A \in \mathbf{R}^{m\times n}$, where $m = 50$ and $n = 25$, with i.i.d. standard normal entries. Using CVX, solve the optimization problem from part (a) with the choice $Q = I$, the identity, so the noise is $\xi \sim \mathcal{N}(0, Q)$, and the choice $\mathcal{X}_+$ is the uncertainty set from part (c). Write down an explicit SDP that you minimize to find H.

Let $H^\star$ be the solution to your SDP (from CVX), and let $\hat{H} = (A^TA)^{-1}A^T$ be an alternative possibility.

For each of $H^\star$ and $\hat{H}$, find the X solving your relaxation from part (a), calling them $X^\star$ and $\hat{X}$. Now, generate 100 sample draws $Z \sim \mathcal{N}(0, X^\star)$ and $Z \sim \mathcal{N}(0, \hat{X})$, set $\hat{x} = \mathbf{sign}(Z)$ for each, and compare the risk of $H^\star$ and $\hat{H}$ on $\hat{x}$ for each of the sampling distributions, that is, $\mathbf{E}[\|H(A\hat{x}+\xi) - \hat{x}\|^2]$. Plot 4 histograms of the attained risk values, one for each of the pairings $H^\star$ with $\mathcal{N}(0, X^\star)$, $\hat{H}$ with $\mathcal{N}(0, X^\star)$, $H^\star$ with $\mathcal{N}(0, \hat{X})$, and $\hat{H}$ with $\mathcal{N}(0, \hat{X})$. What do you observe?