due April 26, Friday 23:59pm

EE364b Homework 3

3.0 *A randomized Kaczmarz solver for Least Squares.* Consider the Least Squares minimization problem

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

where $a_1^T, ..., a_m^T$ are the rows of a data matrix $A \in \mathbf{R}^{m\times n}$. We will consider the stochastic subgradient descent iterates

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

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) 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 $\frac{(a_j^T x - b_j)}{p_j} a_j$ is an unbiased subgradient, i.e.,

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

where the expectation is taken over the random variable j.

(b) 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 (1). Consider the constant step size $\alpha_t = \frac{1}{||A||_F^2}$, 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) 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. Apply this bound recursively to obtain a bound on $\mathbf{E}\,||e_t||_2^2$ involving only $\mathbf{E}\,||e_0||_2^2$, $||A||_F$, $\sigma_{\min}(A)$.
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*

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

for every vector $e \in \mathbf{R}^n$.

(d) 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 ?

3.1 *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) $\pm\,0.1$ 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 $\max(b - Ax^{\mathrm{ds}})$.

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 $\max(b - Ax)$.

3.3 *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.

Hint. Note that projecting $x^{(k)}$ onto the constraints involves projecting onto a simplex, or solving the optimization problem

$$\begin{array}{ll} \text{minimize} & (1/2)\|z - x^{(k)}\|_2^2 \\ \text{subject to} & \mathbf{1}^T z = 1, \quad z \succeq 0, \end{array} \tag{2}$$

with variable z. One way to solve the problem is to introduce a dual variable ν for the equality constraint and write the (partial) Lagrangian as

$$L(z, \nu) = (1/2)\|z - x^{(k)}\|_2^2 + \nu(\mathbf{1}^T z - 1),$$

with $\mathbf{dom}\, L(z, \nu) = \mathbf{R}_+^n \times \mathbf{R}$ (in other words, the inequality constraint $z \succeq 0$ is implicit). Consider the single variable function $g(\nu)$ obtained by minimizing $L(z, \nu)$ over z. The value $\nu^\star$ that maximizes $g(\nu)$ can be found using bisection, and the solution $z^\star$ to problem (2) can be found given $\nu^\star$.

(b) Implement the method and run it on the problem with $n = 10$ assets, with return sample oracle `log_opt_return_sample` in the file `log_opt_return_sample.m`. This function called with argument m returns an $n \times m$ matrix whose columns are independent return samples. We have also provided a function to project onto the simplex in `projToSmplx.m`.

You are welcome to look inside `log_opt_return_sample.m` 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.

(c) *Allowing short sales.* Repeat part (b) without the constraint $x \geq 0$. In other words, solve the optimization problem

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

using the same algorithm as in part (b) (though the projection onto the constraints will be different, because the constraints are different). As in part (b), plot the (approximate) objective value versus iteration, as well as the best approximate objective value obtained up to that iteration.