4.1 Consider the Bregman divergence $D(x, y) = \sum_i x_i \log(x_i/y_i) - (x_i - y_i)$, which is known as the generalized KL divergence.

Show that the projection with respect to this Bregman divergence on the simplex, i.e., $\Delta_n = \{x \in \mathbb{R}_+^n \mid 1^T x = 1\}$, amounts to a simple renormalization $y \rightarrow y/\|y\|_1$.

4.2 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 $(x_1^{(0)}, x_2^{(0)})$, 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 $\tilde{x}$. You will now show that the fixed point $\tilde{x}$ is optimal, i.e., minimizes $\phi$. Thus, coordinate-wise descent works for $\ell_1$-regularized minimization of a differentiable function.

(i) Show that $\tilde{x}$ satisfies

$$0 \in \frac{\partial f}{\partial x_i}(\tilde{x}) + \lambda \mathcal{I}_i, \quad i = 1, \ldots, n,$$

where $\mathcal{I}_i$ is the subdifferential of $\| \cdot \|$ at $\tilde{x}_i$: $\mathcal{I}_i = \{-1\}$ if $\tilde{x}_i < 0$, $\mathcal{I}_i = \{+1\}$ if $\tilde{x}_i > 0$, and $\mathcal{I}_i = [-1, 1]$ if $\tilde{x}_i = 0$.

(ii) Show that the condition in (i) implies that $\tilde{x}$ minimizes $\phi$.

(c) Work out an explicit form for coordinate-wise descent for $\ell_1$-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.

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

\[
\min_{x} f(x) = \max_{i \in \{1, \ldots, m\}} \{a_i^Tx + b_i\} \quad \text{subject to } x \in \Delta_n = \{z \in \mathbb{R}_+^n \mid z^T1 = 1\}.
\]

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 \(f(x^k) - f(x^*)\) or \(f_{\text{best}}^k - f(x^*)\) as a function of iteration for the best stepsize (chosen by smallest optimality gaps) for each method. Which method gives the best performance?

4.4 SGD versus dual averaging. A support vector machine problem has an objective of the form

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

We consider using the dual averaging method to minimize the function \(f\) over the probability simplex in \(\mathbb{R}^n\), i.e., \(\Delta_n = \{x \in \mathbb{R}_+^n \mid 1^Tx = 1\}\). This constraint set is a heuristic for finding a sparse solution \(x^*\), i.e., one with many zero entries. 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:

**SGD** \(x^{k+1} = \arg\min_{x \in \Delta_n} \left\{ \|x - (x^k - \alpha_k g^k)\|_2^2 \right\}\)

**Dual averaging** \(x^{k+1} = \arg\min_{x \in \Delta_n} \left\{ \left\langle z^k, x \right\rangle + \frac{1}{2\alpha_k} \|x\|_2^2 \right\}\).
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.py` (for Python) to generate the matrix \( A = [a_1 \cdots a_N]^T \in \mathbb{R}^{N \times n} \). Run each method for 200 steps, and give two plots: one with the gaps \( f(x^k) - f(x^*) \) 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.