Networked Control Systems and Security - Laboratory

Exercise 1. ***Centralized optimization - Gradient descent algorithm***
Consider the following optimization problem:

$$\min_{x \in \mathbb{R}^p} f(x), \qquad \text{with} \qquad f(x) = \frac{1}{2} x^\top Q x + b^\top x$$

with $x \in \mathbb{R}^p$, $Q \in \mathbb{R}^{p \times p}$, $b \in \mathbb{R}^p$, under the following assumptions:

- Q is symmetric, i.e., $Q^\top = Q$;
- Q is positive semidefinite, i.e., $x^\top Q x \geq 0$ (given the above).

Answer the following questions:

(a) Verify that $f(x)$ is differentiable and, if it is, compute $\nabla f(x)$;

(b) Verify that $f(x)$ is convex;

(c) Verify that $\nabla f(x)$ is L-Lipschitz and determine the Lipschitz constant $L \geq 0$;

(d) Does L depends on Q and/or b? Could you explain why?

(e) Discuss the convergence of the gradient descent algorithm $x(k+1) = x(k) - \rho \nabla f(x(k))$.

(f) Simulate the gradient descent algorithm with $\rho = 0.1/L$, $\rho = 1/L$, $\rho = 3/L$ and comment the results.

Assume that $Q = L$ and $b = 0$, where L is the Laplacian matrix, and aswer the following questions:

(g) Write the update of the gradient descent algorithm in compact form;

(h) Write the update of the i-th component of the gradient descent algorithm;

(i) Can this iteration be considered as distributed?

(j) Compute the set of minimizers. Is it a singleton or not? Why?

(k) Verify that $\nabla f(x)$ is L-Lipschitz and determine the Lipschitz constant $L \geq 0$;

(l) Compare this condition with the standard $\rho < 1/\max_i\{d_i\}$? Which is better? Why?

Exercise 2. ***Centralized optimization - Proximal gradient algorithm***
Consider the following optimization problem:

$$\min_{x\in\mathbb{R}^p} f(x), \qquad \text{with} \qquad f(x) = \frac{1}{2}x^\top Qx + q^\top x \quad \text{and} \quad \mathcal{X} = \{x \in \mathbb{R}^p : Ax = b\},$$

with $x \in \mathbb{R}^p$, $Q \in \mathbb{R}^{p\times p}$, $b \in \mathbb{R}^p$, under the following assumptions:

- Q is symmetric, i.e., $Q^\top = Q$;
- Q is positive semidefinite, i.e., $x^\top Qx \geq 0$ (given the above).

Answer the following questions:

(a) Which assumptions should we make on A and b for $\mathcal{X}$ to be nonempty, closed, and convex?

(b) Write the update of the proximal gradient algorithm $x(k+1) = \Pi_{\mathcal{X}}(x(k) - \rho\nabla f(x(k)))$;

(c) Discuss the convergence of the proximal gradient algorithm;

(d) Simulate the proximal gradient algorithm with $\rho = 0.1/L$, $\rho = 1/L$, $\rho = 3/L$ and comment the results.

Now assume that the cost function is fully separable, i.e., $f(x) = \sum_{i=1}^{p} f_i(x_i)$ and assume that the constraint set satisfies $A = I - \frac{1}{p}\mathbf{1}_p\mathbf{1}_p^\top$ and $b = 0$. Answer the following questions:

(e) Write the update of the proximal gradient algorithm in compact form;

(f) Write the update of the i-th component of the proximal gradient algorithm;

(g) Can this iteration be considered as distributed?

Exercise 3. ***Distributed optimization – Gradient-based algorithms***
Consider a network of n agents connected by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each agent $i \in \mathcal{V}$ has a local cost function $f_i : \mathbb{R}^p \to \mathbb{R}$ and the network aims to solve:

$$\begin{aligned} \min_{x_i \in \mathbb{R}^p} \quad & \sum_{i \in \mathcal{V}} f_i(x_i), \\ \text{s.t.} \quad & x_i = x_j, \quad \forall (i,j) \in \mathcal{E}. \end{aligned}$$

Assume each f_i is convex, differentiable, and L-smooth. Answer the following questions:

(a) Formulate the unconstrained composite optimization problem (centralized);

(b) Derive the proximal gradient algorithm for the composite problem above;

(c) Compute the i-th component of the resulting iterate;

(d) Approximate the iterate using simple (static) average consensus along the edges of $\mathcal{G}$. Is the resulting algorithm distributed? Does it converge to the correct solution?

(e) Approximate the full iterate (gradient step) using dynamic average consensus. What is the resulting algorithm? Does it converge to the correct solution?

(f) Approximate the partial iterate (gradient) using dynamic average consensus. What is the resulting algorithm? Does it converge to the correct solution?

(g) Write the explicit local update of the distributed ADMM;

(h) Simulate and compare algorithms (d), (e), (f), and (g) on a quadratic local cost $f_i(x_i) = \frac{1}{2} x_i^\top M_i x_i + q_i^\top x_i$.

Exercise 4. ***Distributed learning – EMNIST digit classification***
Consider a network of $n = 10$ agents that collaboratively train a linear binary classifier to distinguish EMNIST digit 3 from digit 8, without sharing raw data. Each agent $i \in \mathcal{V}$ holds $N = 100$ private labeled samples, and the network solves distributedly:

$$\min_{x \in \mathbb{R}^k} \sum_{i \in \mathcal{V}} f_i(x), \qquad f_i(x) = \frac{1}{2N} \|X_i x - y_i\|^2$$

(a) Load the EMNIST training set and retain only digits 3 and 8. Assign labels $y_{i,\ell} = -1$ (digit 3) and $y_{i,\ell} = +1$ (digit 8). Partition the first $n \cdot N$ filtered samples equally among the n agents.
Hint: use `torchvision.datasets.EMNIST` *with* `split="digits"` *and filter targets with boolean masks.*

(b) Apply PCA with $k = 50$ components to the combined $n \cdot N$ training images (each flattened and normalized to $[0, 1]^{784}$), and project all images onto the top-k principal components. Why does $k < N$ guarantee that each agent's feature matrix $X_i \in \mathbb{R}^{N \times k}$ has full column rank? Why does this eliminate the need for regularization?
Hint: use `sklearn.decomposition.PCA(n_components=k).fit(raw)` *on the stacked raw images, then* `pca.transform(...)` *to project; keep the fitted* `pca` *object for step (f).*

(c) Each agent i defines the local unregularized least-squares cost:

$$f_i(x) = \frac{1}{2N} \|X_i x - y_i\|^2,$$

where $X_i \in \mathbb{R}^{N \times k}$ is the projected feature matrix and $y_i \in \mathbb{R}^N$ the label vector. Identify M_i and q_i such that $f_i(x) = \frac{1}{2} x^\top M_i x + q_i^\top x + c_i$. Show that M_i is positive definite and determine $L = \max_i \lambda_{\max}(M_i)$.
Hint: $M_i = X_i^\top X_i / N$ *and* $q_i = -X_i^\top y_i / N$.

(d) Build an Erdős-Rényi network with $n = 10$ nodes and edge probability $p = 0.8$, and weight matrix $W = I - \varepsilon L_{\mathcal{G}}$ with $\varepsilon = 0.1$. Compute the step size $\rho = 0.9(1 - \sigma)^2 / L$ where $\sigma = \lambda_2(W)$, and the centralized optimal solution $x^\star = -\left(\sum_i M_i\right)^{-1} \left(\sum_i q_i\right)$ as a benchmark.
Hint: use `networkx.erdos_renyi_graph` *and* `networkx.laplacian_matrix`*;* σ *is the second largest eigenvalue of* W.

(e) Implement the gradient step tracking algorithm, the gradient tracking algorithm, and the ADMM algorithm, and run for $K = 10000$ iterations. Plot $\max_i \|x_i(k) - x^\star\|$ versus k.
Hint: pay attention to the initialization.

(f) After convergence, form $\bar{x} = \frac{1}{n} \sum_{i \in \mathcal{V}} x_i(K)$ and evaluate classification accuracy on the EMNIST test set (digits 3 and 8 only), applying the same PCA transform. The classifier predicts digit 8 if $\bar{x}^\top \tilde{a} \geq 0$, where $\tilde{a}$ is the projected test image. Comment on the results.
Hint: apply `pca.transform` *to the test images before prediction.*