Homework 3: Constrained Optimization

Instructions: Submit a single (zipped) folder containing all code for this homework, whether using `Matlab` or `Python`. This folder will be downloaded, and all commands executed from within the unzipped folder. You are responsible for ensuring your code runs under the instructions below. The only external optimizer you may use is `cvx` (`Matlab`) or `cvxpy` (`Python`). Language-dependent instructions:
Matlab: For each problem number `N`, submit a script `hw3_prN.m`. This script may optionally call other scripts in the folder. Execution of command `run hw3_prN.m` in the command window must generate a solution to problem `N`. The online version of `Matlab` available through UK will be used for evaluation. The version of `cvx` will be `2.2.2` available at `https://github.com/cvxr/cvx`. The `cvx` folder is loaded into `Matlab` (download onto PC then upload through browser), and `cvx_setup.m` is run to enable use of `cvx`.
Python: For each problem number `N`, submit a script `hw3_prN.py`. This script may optionally call other scripts in the folder. Execution of command `python3 hw3_prN.py` in the command window must generate a solution to problem `N`. You may install an identical environment as the evaluation one by executing the command `pip install -r requirements.txt`, where the `requirements.txt` file is included in the homework.

Problem 1: *(25 points)* Write a `Python` or `Matlab` script to solve the following equality constrained minimization problem:

$$\min_{x\in\mathbb{R}^2} -(1-x_1)^2-(1-x_2)^2$$
$$\text{subject to } x_1^2+x_2^2=r$$

where r is set at the beginning of `hw3_pr1.filetype` as a floating point variable with name `r`. The initial condition, if used, should also be set at the beginning as an array of length 2 with name `x0`.

Evaluation: 50% of the points are awarded for finding a valid local minimum. The remaining points are calculated as $\min\left(\frac{0.05t^\star}{t},1\right)\times\frac{\text{problem points}}{2}$, where t is the time taken by your script to solve the optimization problem and $t^\star$ is the time taken by Matlab's `fmincon` for the same initial condition and value of r.

Problem 2: *(25 points)* Write a `Python` or `Matlab` script that uses the `cvx` solver to train a support vector machine – linear max-margin classifier – that distinguishes between two classes in $\mathbb{R}^2$:

$$\begin{aligned} \min_{\mathbf{w},\beta} \quad & \| \mathbf{w} \|^2 \\ & A^T\mathbf{w} + \beta \mathbf{1}_{n_a} \geq \mathbf{1}_{n_a} \\ & B^T\mathbf{w} + \beta \mathbf{1}_{n_b} \leq -\mathbf{1}_{n_b}, \end{aligned}$$

where subscript of $\mathbf{1}$ indicates its length. There are n_a points in the first class, provided as columns of a matrix A. There are n_b points in the second class, provided as columns of a matrix B. The points will be separable by a hyperplane. The script should produce the weights $\mathbf{w}$ – with variable name `w_svm` – of the SVM as an array of size 2 and the bias β – with variable name `beta_svm` as a floating point value.

Problem 3: *(25 points)* Write a `Python` or `Matlab` script that can solve Problem 2 but for the case where the points may not be separable.

Problem 4: *(25 points)* Write a `Python` or `Matlab` script that implements the log-barrier method to solve the following optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \; & (1 - x_1)^2 + (1 - x_2)^2 \\ \text{subject to } & x_1^2 + x_2^2 \leq r \end{aligned}$$

where r is set at the beginning of `hw3_pr4.filetype` as a floating point variable with name `r`. The initial condition, if used, should also be set at the beginning as an array of length 2 with name `x0`.