האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

**Group Members – Bachelor's degree:**

Netta Shafir 205915788

Dor Tal 205501380

Aviv Lahat 208495333

**The Problem:**

$$\arg\min_{x\in\mathbb{R}^n} ||y - Hx||_2 \ s.t\ \vec{1}^T x = 1 \wedge x \geq 0$$

**Transforming the objective:**

Claim: $\arg\min_{x\in\mathbb{R}^n} ||y - Hx||_2 = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}||y - Hx||_2^2$

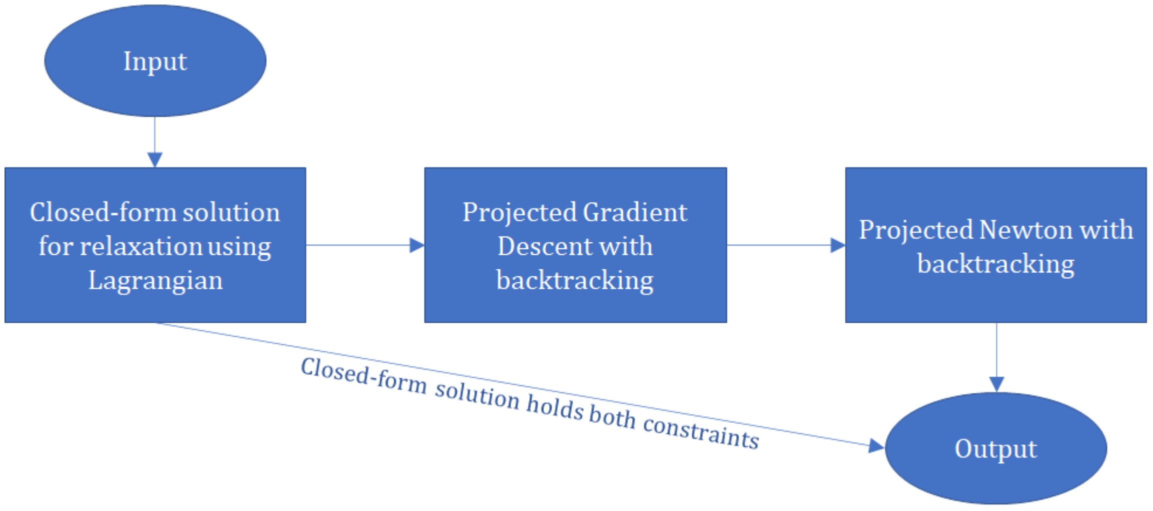
Proof:

$$\nabla\left(\frac{1}{2}||y - Hx||_2^2\right) = 0 \Leftrightarrow y - Hx = 0 \Leftrightarrow \nabla\left(||y - Hx||_2\right) = 0 \Leftrightarrow ||y - Hx||_2 = 0$$

**The New Problem:**

$$\arg\min_{x\in\mathbb{R}^n} \frac{1}{2}||y - Hx||_2^2 \ s.t\ \vec{1}^T x = 1 \wedge x \geq 0$$

**The Solution:**


Input
Closed-form solution for relaxation using Lagrangian
Projected Gradient Descent with backtracking
Projected Newton with backtracking
Closed-form solution holds both constraints
Output

Our algorithm works in 3 steps.

1. Relaxation step:

   We derived a closed-form solution for the problem -

   $$\arg\min_{x\in\mathbb{R}^n} \left|\left|y - Hx\right|\right|_2 \; s.t \; \vec{1}^T x = 1$$

   using Lagrangian function –

   $$L = f_0(x) + \lambda\left(\vec{1} - \vec{1}^T x\right)$$

   <u>Note:</u> since we are coping only with equality constraints in the relaxed problem – lambda can be either positive or negative.

   The constraints term (multiplied by lambda in the LaGrangian) is an affine function w.r.t both x and lambda. Therefore, it is convex w.r.t both. We also know that $f_0$ is convex w.r.t x and lambda ($f_0$ doesn't contain lambda).

   Having said that, we can conclude that the LaGrangian function above is convex w.r.t both x and $\lambda$ as a sum of convex functions. Therefore, by differentiating w.r.t both $x \; and \; \lambda$ and setting the gradient to 0, we are guaranteed to obtain $x^* \; and \; \lambda^*$, for which the optimal minimum of the LaGrangian function above is achieved.

   We derived the function – once w.r.t $x$ resulting in $\nabla f_0(x) = H^T(Hx - y) - \lambda \cdot \vec{1}$, and once w.r.t $\lambda$ resulting in $-\vec{1}^T x + \vec{1}$, then compared to zero to achieve –

   $$H^T Hx - \lambda \cdot \vec{1} = H^T y$$

   $$\vec{1}^T x = 1$$

   In matrix form the above equations for the relaxed constraints can be written as a block matrix –

   $$\begin{bmatrix} H^T H & \overrightarrow{-1} \\ \vec{1} & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} H^T y \\ 1 \end{bmatrix}$$

   We define $\hat{x}^*$ as the optimal solution for the relaxed problem above, and $x^*$ the optimal solution for the original constraint problem.

   Since the loss for a solution of the unconstraint problem is always $\leq$ a solution for the constraint problem, we can conclude that finding an optimal solution $\hat{x}^*$ that also holds the second constraint $\hat{x}^* \geq 0$ means we found an optimal solution $x^*$.

The first step of our algorithm is to search for such $\hat{x}^*$ and check if it holds the second constraint $x \geq 0$. Finding such $\hat{x}^*$ from a closed-form solution promises us the optimum.

2. Gradient Descent step:

   In the case where the solution $\hat{x}^*$ from the Lagrangian method doesn't hold the second constraint for our optimization problem - $x \geq 0$, we use this vector as the initializer for the Gradient Descent step (denoted GD).

   In each iteration of GD, we find the best step size $t$ using backtracking and project the solution vector into the simplex. To find the best method for calculating step size $t$ in GD we tried the backtracking method and momentum method. We chose to continue with backtracking because it had the overall better performance, while the momentum method can miss the local minimum at times and isn't as straightforward as the backtracking method.

   Since GD performance decreases drastically when the condition number of the matrix is large, we check the condition number and the matrix dimensions before running GD. We compute the ratio between the largest and smallest eigenvalues and check the matrix dimensions to ensure that $\#_{rows} \geq \#_{columns}$ so that the Hessian doesn't have an eigenvalue of 0.

   The iterations of GD continue until one of our stop conditions are met – either the improvement by GD is less than $\epsilon$ or GD has done the maximum number of steps.

   When checking the improvement of GD between iterations, we use the original objective to control the tolerance w.r.t the original problem.

3. Newton Method step:

   Once GD is completed, we continue to Newton Method for several reasons. First, in complex functions, GD may improve by less than $\epsilon$ although the solution vector is far from optimum. Second, once our solution vector is close to optimum, Newton can improve the final accuracies.

   We created a projection Newton method which projects every iterations solution into the simplex, similar to what is performed in GD. Likewise, in each iteration of the projected Newton we found the best step size $t$ using a separate backtracking method than GD. In the Hessian calculation, we ensure that $H^T H$ is invertible by computing $H^T H + \epsilon \cdot I$.

To ensure that our projection method works as well as existing Newton methods, we implemented Newton with interior points using an exponential barrier (instead of log as seen in class) and compared performance. To be more precise, we tried solving the following optimization problem:

$$\min_x f_0(x) + \sum_{x_i \in \vec{x}} \frac{1}{t} \cdot e^{-\eta \cdot x_i}$$
$$s.t : \vec{1}^\top x = 1$$

<u>Note:</u> Due to convexity of the exponent function, the new target function above is convex as well.

We decided to set $\eta = 10$, so our exponential barrier uses the function $e^{-10x}$. This function ensures that the loss significantly increases for $x_i < 0$ by sending negative values of $x$ to $\infty$ and positive values to 0 – which is the behavior that we anticipate (Figure 1). Therefore, we use this penalty method along with the standard insertion of additional constraints to the objective.

By setting $\phi(x)$ to be the sum of exponents as seen above we obtain the following optimization problem as we saw in class for log-barriers:

$$\min_x t \cdot f_0(x) + \phi(x)$$
$$s.t : \vec{1}^\top x = 1$$

By solving the following equations, we obtain the desired newton step:

$$\begin{bmatrix} t\nabla^2 f_0(x) + \nabla^2 \phi(x) & \vec{1} \\ \vec{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \mu^* \end{bmatrix} = \begin{bmatrix} t\nabla f_0(x) + \nabla \phi(x) \\ 0 \end{bmatrix}$$

We observed that the projected Newton method and interior points method improved our algorithms performance indifferently and decided on the projected Newton method. We mainly chose this method because it promises a valid solution vector unlike the interior points method, so for unseen samples this method is the way to go.

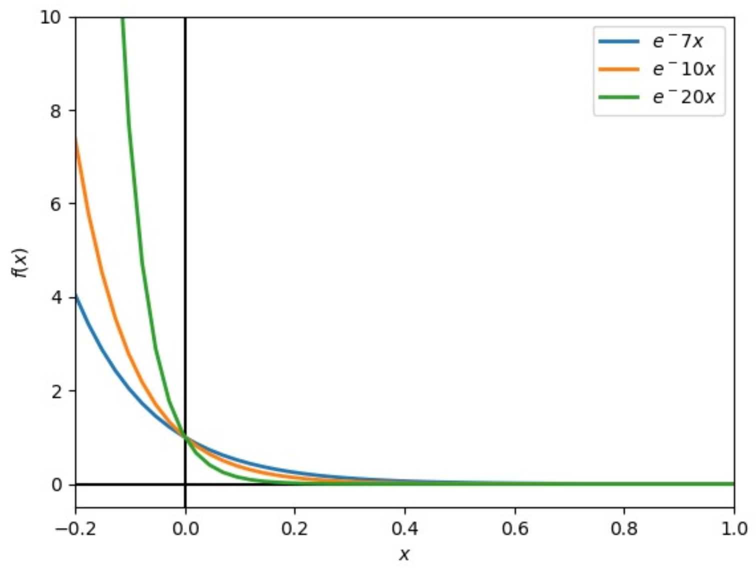


Figure 1: $e^{-10x}$ behaves like the function $\ell$_ shown in recitation 9. When the exponent is raised to a constant approaching ∞ then these functions converge.

**Hyperparameters:**

There are several hyperparameters that we define throughout our algorithm:

Stopping hyperparameters – GD $\epsilon$, GD max steps, Newton $\epsilon$, Newton max steps

Optimization method hyperparameters – GD step size $t$ $\alpha$ and $\beta$, Newton step size $t$ $\alpha$ and $\beta$ Newton $\epsilon$ for Hessian invertibility

In order to choose the hyperparameters that give the best results, we tested each hyperparameter in an isolated test to see which value led to the best overall results.