Mathematics of data: homework 4

Marco Pietro Abrate 292996

January 11, 2019

1 Projection free convex low-rank matrix optimization

Consider the matrix optimization problem

$$\mathbf{X}^* \in \arg\min_{\mathbf{X}} \{f(\mathbf{X}) : rank(\mathbf{X}) \leq r, \mathbf{X} \in \mathbb{R}^{p\times m}\} \tag{1}$$

for some convex objective function f, where $r < min\{p, m\}$ forces the solution to be low-rank. Convex relaxation methods are extensively studied for solving low-rank matrix optimization problems. The nuclear norm is an effective convex surrogate to obtain (approximatively) low-rank solutions

$$\mathbf{X}^* \in \arg\min_{\mathbf{X}} \{f(\mathbf{X}) : ||\mathbf{X}||_* \leq k, \mathbf{X} \in \mathbb{R}^{p\times m}\} \tag{2}$$

When the projection onto the nuclear norm ball is easy to perform, we can solve this problem by projected gradient methods. However, as the problem dimensions increase, the projection becomes a computational bottleneck.

1.1 Projection onto the nuclear norm ball

Let $\mathbf{Z} = \mathbf{U}\Sigma\mathbf{V}^T$ be the singular value decomposition (SVD) of $\mathbf{Z} \in \mathbb{R}^{p\times m}$. Denote the diagonal of $\Sigma \in \mathbb{R}^{s\times s}$ by a vector $\sigma \in \mathbb{R}^s$, where $s = \min\{p, m\}$. Let σ^{l_1} be the projection of σ onto the l_1-norm ball $\{\mathbf{x} : \mathbf{x} \in \mathbb{R}^s, ||\mathbf{x}||_1 \leq k\}$. It can be shown that the projection of this matrix onto the nuclear norm ball $\chi = \{\mathbf{X} : \mathbf{X} \in \mathbb{R}^{p\times m}, ||\mathbf{X}||_* \leq k\}$ can be computed by projecting σ onto the l_1-norm ball. The projection of $\mathbf{Z}$ onto the nuclear norm ball can be written as

$$proj_\chi(\mathbf{Z}) = prox_\chi(\mathbf{Z}) = \arg\min_{\mathbf{X}} \{\frac{1}{2}||\mathbf{X} - \mathbf{Z}||_F^2 + ||\mathbf{X}||_*\} \tag{3}$$

Now, since $||\mathbf{X} - \mathbf{Z}||_F^2 \geq ||\Sigma_X - \Sigma_Z||_F^2$, finding a solution for (3) is the same as finding a solution for

$$\arg\min_{\Sigma_X} \{\frac{1}{2}||\Sigma_X - \Sigma_Z||_F^2 + ||\Sigma_X||_*\} \tag{4}$$

However, since Σ_X and Σ_Z are diagonal matrices, we can interpret them as vectors and rewrite the problem as follow

$$\arg\min_{\mathbf{x}}\{\frac{1}{2}||\mathbf{x}-\sigma||_2^2+||\mathbf{x}||_1\} \tag{5}$$

This is exactly the projection of σ onto the l_1-norm ball

$$proj_{||\mathbf{x}||_1}(\sigma)=prox_{||\mathbf{x}||_1}(\sigma)=\arg\min_{\mathbf{x}}\{\frac{1}{2}||\mathbf{x}-\sigma||_2^2+||\mathbf{x}||_1\} \tag{6}$$

Which has solution σ^{l_1}. Therefore, Σ^{l_1} is a solution for (4) and, since $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, $\mathbf{U}\Sigma^{l_1}\mathbf{V}^T$ is a solution for (3).
After implementing the projection operator in Python, setting $k=5000$, the computation time, for the 100k and 1M MovieLens datasets, was measured. Running the algorithms 5 times and averaging the times, these are the results

100k ML dataset $\rightarrow$ 0.49985 s
1M ML dataset $\rightarrow$ 40.31346 s

1.2 Linear minimization oracle of nuclear norm

Section 1.1 shows that projection onto the nuclear norm ball requires computing the SVD. The computational complexity of the SVD is $O(\min(m^2p, mp^2))$, and this can easily become a computational bottleneck if m and p are large. This increased the popularity of algorithms that leverage the linear minimization oracle (lmo) insted

$$lmo_\chi=\arg\min_{\mathbf{X}\in\chi}\langle\mathbf{X},\mathbf{Z}\rangle \qquad \text{where} \qquad \langle\mathbf{X},\mathbf{Z}\rangle=Tr(\mathbf{Z}^T\mathbf{X}) \tag{7}$$

Note that $lmo_\chi(\mathbf{Z})$ is not single valued in general. With abuse of terminology, when we say that we compute the lmo, we actually mean that we compute an instance $\mathbf{X}$ such that $\mathbf{X}\in lmo_\chi(\mathbf{Z})$.
The lmo_χ, when χ is the nuclear norm ball $\chi=\{\mathbf{X}:\mathbf{X}\in\mathbb{R}^{p\times m},||\mathbf{X}||_*\leq k\}$ gives the following output

$$-k\mathbf{u}\mathbf{v}\in lmo_\chi\{\mathbf{Z}\} \tag{8}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the left and right singular vectors that corespond to the largest singular value of $\mathbf{Z}$. This can be proved as follow

$$\langle -k\mathbf{u}\mathbf{v}^T,\mathbf{Z}\rangle = \tag{9a}$$
$$= Tr\{\mathbf{Z}^T(-k\mathbf{u}\mathbf{v}^T)\} = \tag{9b}$$
$$\text{(calling } \sigma_{max} \text{ the largest singular value of } \mathbf{Z}) \tag{9c}$$
$$= -kTr(\sigma_{max}\mathbf{v}\mathbf{v}^T) = \tag{9d}$$
$$= -k\sigma_{max}||\mathbf{v}||_2^2 = \tag{9e}$$
$$= -k\sigma_{max} = \tag{9f}$$
$$= -k||\mathbf{Z}||_2 \leq \quad \text{(because } \mathbf{X}\in\chi) \tag{9g}$$
$$\leq -||\mathbf{X}||_*||\mathbf{Z}||_2 \leq \quad \text{(because of Cauchy Schwarz)} \tag{9h}$$
$$\leq \langle\mathbf{X},\mathbf{Z}\rangle \tag{9i}$$

After implementing the *lmo* operator in Python, setting $k = 5000$, the computation time, for the 100k and 1M MovieLens datasets, was measured. Running the algorithms 5 times and averaging the times, these are the results

100k ML dataset $\rightarrow$ 0.07186 s
1M ML dataset $\rightarrow$ 0.22491 s

2 Crime scene investigation with blind deconvolution

In this section, we are asked to identify the license plate of a car involved in a crime scene investigation. Unfortunately, the CCTV image of the car is blurry. We simulate this scence by trying to deblur a plate image found from the internet.
This is an instance of the blind deconvolution problem. The set-up of the problem is as follow: given two unknown vectors $\mathbf{x}, \mathbf{w} \in \mathbb{R}^L$, we observe their circular convolution $\mathbf{y} = \mathbf{x} * \mathbf{w}$, i.e.

$$y_l = \sum_{l'=1}^{L} w_{l'} x_{l-l'+1} \tag{10}$$

where the index $l - l' + 1$ in the sum above is understood to be modulo L. The goal of blind deconvolution is to separate $\mathbf{w}$ and $\mathbf{x}$. We simply assume $\mathbf{w}$ and $\mathbf{x}$ belong to known subspaces of $\mathbb{R}^L$ of dimension K and N, i.e., we can write

$$\mathbf{w} = \mathbf{B}\mathbf{h}, \quad \mathbf{h} \in \mathbb{R}^K \tag{11}$$

$$\mathbf{x} = \mathbf{C}\mathbf{m}, \quad \mathbf{m} \in \mathbb{R}^N \tag{12}$$

In the image deblurring example, $\mathbf{x}$ corresponds to the image we want to recover, $\mathbf{w}$ to a 2D blur kernel. We assume that we know the support of the blur kernel. In this experiment, we use a very rough estimate for the support and simply use a box at the center of the domain.

2.1 Frank-Wolfe for blind deconvolution

Let $\mathbf{b}$ be the L-point normalized discrete Fourier transform (DFT) of the observation $\mathbf{y}$, i.e., $\mathbf{b} = \mathbf{F}\mathbf{y}$, where $\mathbf{F}$ is the DFT matrix. Then, $\mathbf{b}$ can be written as $\mathbf{b} = \mathbf{A}(\mathbf{X})$ where $\mathbf{X} = \mathbf{h}\mathbf{m}^T$ and $\mathbf{A}$ is a linear operator. This reformulation allows us to express $\mathbf{y}$, which is a nonlinear combination of the coefficients of $\mathbf{h}$ and $\mathbf{m}$, as a linear combination of the entries of their outer product $\mathbf{X} = \mathbf{h}\mathbf{m}^T$. Since $\mathbf{X}$ is a rank one matrix, we can use the nuclear norm to enforce approximately low-rank solutions. Then, we can formulate the blind deconvolution problem as follow

$$\mathbf{X}^* \in \arg\min_{\mathbf{X}} \{ \frac{1}{2} ||\mathbf{A}(\mathbf{X}) - \mathbf{b}||_2^2 : ||\mathbf{X}||_* \leq k, \mathbf{X} \in \mathbb{R}^{p \times m} \} \tag{13}$$

where $k > 0$ is a tuning parameter.
We will apply the Frank-Wolfe algorithm to solve the optimization problem

given in (13). The Frank-Wolfe algorithm is one of the earliest algorithm that avoids the proximal operator. Instead, it leverages the *lmo*

$$lmo(\nabla f(\mathbf{Z})) = \arg\min_{\mathbf{X}\in\mathcal{X}} \langle \nabla f(\mathbf{Z}), \mathbf{X} \rangle \tag{14}$$

where $\mathcal{X} = \{\mathbf{X} : ||\mathbf{X}||_* \leq k, \mathbf{X} \in \mathbb{R}^{p\times m}\}$ as in section (1.1). It applies to generic constrained minimization template with a smooth objective function, $min_{\mathcal{X}}\{f(\mathbf{X}) : \mathbf{X} \in \mathcal{X}\}$ as follow

Frank-Wolfe's algorithm

1. Choose $\mathbf{X}_0 \in \mathcal{X}$.
2. For $k = 0, 1, ...$ perform:
$$\begin{cases} \hat{\mathbf{X}}_k := lmo(\nabla f(\mathbf{X}_k)) \\ \mathbf{X}_{k+1} := (1-\gamma_k)\hat{\mathbf{X}}_k + \gamma_k \mathbf{X}_k \end{cases}$$

where $\gamma_k := 2/(k+2)$.

Since this algorithm applies only for smooth objectives. We hereby show that the considered objective function is smooth, in the sense its gradient is Lipschitz continuous. Treating the linear operator as a matrix, without loss of generality, the gradient of f is

$$\nabla f(\mathbf{Z}) = \mathbf{A}^T(\mathbf{A}\mathbf{Z} - \mathbf{b}) \tag{15}$$

Now, considering the norm of the difference of ∇f computed in two points $\mathbf{X}$ and $\mathbf{Y}$ belonging to the domain of f, we can show that ∇f is L-Lipschitz continuous

$$||\mathbf{A}^T(\mathbf{A}\mathbf{X} - \mathbf{b}) - \mathbf{A}^T(\mathbf{A}\mathbf{Y} - \mathbf{b})|| = \tag{16a}$$
$$= ||\mathbf{A}^T\mathbf{A}\mathbf{X} - \mathbf{A}^T\mathbf{A}\mathbf{Y}|| = \tag{16b}$$
$$= ||\mathbf{A}^T\mathbf{A}(\mathbf{X} - \mathbf{Y}) \leq \tag{16c}$$
$$\leq ||\mathbf{A}^T\mathbf{A}|| \cdot ||\mathbf{X} - \mathbf{Y}|| \tag{16d}$$

with $L = ||\mathbf{A}^T\mathbf{A}||$.
Implementing the algorithm in Python, putting $k = 150$ (as the radius of the nuclear norm ball) found as optimal value after a parameter sweep and performing 200 iterations, the resulting deblurred license plate, which appears to be "J209LTL", is shown in Figure 1.

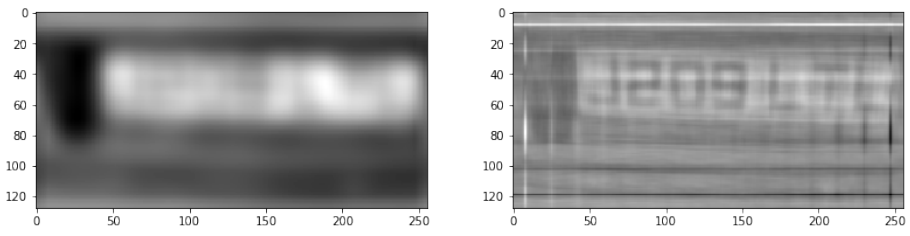

Figure 1: Blurred (left) and deblurred (right) license plate using Frank-Wolfe algorithm.

3 K-means clustering by semidefinite programming

Clustering is an unsupervised machine learning problem in which we try to partition a given data set into k subsets based on distance between data points or similarity among them. The goal is to find k centers and to assign each data point to one of the centers such that the sum of the square distances between them are minimal.
Given a set of n points in a d-dimensional Euclidean space, denoted by

$$S = \{\mathbf{s}_i = (s_{i1}, ..., s_{id})^T \in \mathbb{R}^d, \quad i = 1, ..., n\} \tag{17}$$

the aim is to find an assignment of the n points into k disjoint clusters $\mathcal{S} = (S_1, ..., S_k)$ whose centers are $\mathbf{c}_j (j = 1, ..., k)$ based on the total sum of squared Euclidean distances from each point $\mathbf{s}_i$ to its assigned cluster centroid $\mathbf{c}_j$, i.e.

$$f(S, \mathcal{S}) = \sum_{j=1}^{k} \sum_{i=1}^{|S_j|} ||\mathbf{s}_i^j - \mathbf{c}_j||^2 \tag{18}$$

where $|S_j|$ is the number of points in S_j, and $\mathbf{s}_i^j$ is the i-th point in S_j.
The solution for the abovementioned model-free k-means clustering problem can be approximated with an SDP-relaxation. The resulting optimization problem takes the standard semidefinite programming form

$$\mathbf{X}^* \in \arg\min_{\mathbf{X}} \{\langle \mathbf{C}, \mathbf{X} \rangle : \mathbf{X}\mathbf{1} = \mathbf{1}, \mathbf{X}^T\mathbf{1} = \mathbf{1}, \mathbf{X} \geq 0, Tr(\mathbf{X}) \leq k, \mathbf{X} \in \mathbb{R}^{p \times p}, \mathbf{X} \succeq 0\} \tag{19}$$

where $\mathbf{C} \in \mathbb{R}^{p \times p}$ is the Euclidean distance matrix between the data points. $Tr(\mathbf{X}) \leq k$ enforces approximately low-rank solutions, the linear inclusion constraint $\mathbf{X} \geq 0$ is element-wise non-negativity of $\mathbf{X}$, the linear equation constraints $\mathbf{X}\mathbf{1} = \mathbf{1}$ and $\mathbf{X}^T\mathbf{1} = \mathbf{1}$ require row and column sum of $\mathbf{X}$ to be equal to 1's, and $\mathbf{X} \succeq 0$ means that $\mathbf{X}$ is positive semi-definite. Recall that $Tr(\mathbf{X}) = ||\mathbf{X}||_*$, for any positive semi-definite matrix $\mathbf{X}$.
The problem in (19) can be reformulated as

$$\min_{x \in \chi} f(x) + g_1(A_1(x)) + g_2(A_2(x)) \quad \text{subject to} \quad B(x) \in \mathcal{K} \tag{20}$$

where $f(x) = \langle \mathbf{C}, x \rangle$ is a smooth convex function, g_1 is the indicator function of singleton $\{b_1\}$, g_2 is the indicator function of singleton $\{b_2\}$ and $\mathcal{K}$ is the positive orthant for which computing the projection is easy.
Because of the non-smooth terms g_1 and g_2, Frank-Wolfe can not be used to solve this minimization problem. We will attempt to solve (20) with the HomotopyCGM algorithm with a conditional gradient method, described as

CGM for composite problems

1. Choose $x_0 \in \chi$ and $\beta_0 > 0$.
2. For $k = 0, 1, ...$ perform:

$$\begin{cases} y_k = 2/(k+1), \quad \text{and} \quad \beta_k = \beta_0/\sqrt{k+1} \\ v_k = \beta_k \nabla f(x_k) + A_1^T(A_1 x_k - b) + A_2^T(A_2 x_k - b) + (x_k - proj_{\mathcal{K}}(x_k)) \\ \hat{x}_k := \arg\min_{x \in \chi} \langle v_k, x \rangle \\ x_{k+1} := (1 - \gamma_k) x_k + \gamma_k \hat{x}_k \end{cases}$$

Output: x_k.

3.1 Conditional Gradient Method for clustering fashion-MNIST data

To start, we show that the domain $\chi = \{\mathbf{X} : Tr(\mathbf{X}) \leq k, \mathbf{X} \in \mathbb{C}^{p\times p}, \mathbf{X} \succeq 0\}$ is a convex set. To do so, we consider two points $\{\mathbf{M}; \mathbf{N}\} \in \chi$, and we show that $\alpha\mathbf{M} + (1-\alpha)\mathbf{N}$ is still in χ, with $\alpha \in [0; 1]$. Computing the trace

$$
\begin{aligned}
&Tr(\alpha\mathbf{M} + (1-\alpha)\mathbf{N}) = && && \text{(21a)}\\
&= \alpha Tr(\mathbf{M} + (1-\alpha)Tr(\mathbf{N}) \leq && \text{(since } \mathbf{M} \in \chi\text{)} && \text{(21b)}\\
&\leq \alpha k + (1-\alpha)Tr(\mathbf{N}) \leq && \text{(since } \mathbf{N} \in \chi\text{)} && \text{(21c)}\\
&\leq \alpha k + (1-\alpha)k = && && \text{(21d)}\\
&= k && && \text{(21e)}
\end{aligned}
$$

and showing that the sum is stil positive semi-definite, for any $\mathbf{u} \in \mathbb{C}^p$

$$
\begin{aligned}
&\mathbf{u}[\alpha\mathbf{M} + (1-\alpha)\mathbf{N}]\mathbf{u}^T = && && \text{(22a)}\\
&= \alpha\mathbf{u}\mathbf{M}\mathbf{u}^T + \mathbf{u}\mathbf{N}\mathbf{u}^T - \alpha\mathbf{u}\mathbf{N}\mathbf{u}^T \geq && \text{(since } \alpha \in [0; 1]\text{)} && \text{(22b)}\\
&\geq 0 && && \text{(22c)}
\end{aligned}
$$

we have finally proved that χ is actually a convex set.
Now, we want to rewrite the objective function, call it $h(x)$, to its quadratic penalty form, call it $h_\beta(x)$. We first state that, given a linear inclusion constraint $Tx \in \mathcal{Y}$, the corresponding quadratic penalty function is given by

$$QP_{\mathcal{Y}}(x) = dist^2(Tx, \mathcal{Y}) = \min_{y\in\mathcal{Y}} ||y - Tx||^2 \tag{23}$$

and that the quadratic penalty objective of a minimization problem of the type $\min_x\{f(x) : Ax = b, x \in \chi\}$ is $f_\beta(x) := f(x) + \frac{1}{2\beta}||Ax - b||^2$, where $\beta > 0$ is the penalty parameter.
Considering now $h(x)$, we can write

$$h_\beta(x) = f(x) + \frac{1}{2\beta}||A_1(x) - b||^2 + \frac{1}{2\beta}||A_2(x) - b||^2 + \frac{1}{2\beta}dist^2(x, \mathcal{K}) \tag{24}$$

Then, setting $k^* = \arg\min_{k\in\mathcal{K}} ||k - x||^2$ and writing the squared distance $dist^2(x, \mathcal{K})$ as $||k^* - x||^2$, we can compute the gradient of $h_\beta(x)$ as

$$\nabla h_\beta(x) = \nabla f(x) + \frac{1}{\beta}A_1^T(A_1 - b) + \frac{1}{\beta}A_1^T(A_1 - b) - \frac{1}{\beta}(k^* - x) \tag{25}$$

Finally, noticing that $k^* = \arg\min_{k\in\mathcal{K}} ||k-x||^2 = proj_{\mathcal{K}}(x)$, we can see that the gradient of the penalized objective is equal to v_k/β, as defined in the algorithm.
To compute the gradient of f we can observe that, since $f(\mathbf{X}) = \langle\mathbf{C}, \mathbf{X}\rangle$

$$\nabla f(\mathbf{X}) = \frac{\partial Tr(\mathbf{X}^T\mathbf{C})}{\partial\mathbf{X}} = \mathbf{C} \tag{26}$$

and we can notice that the projection of a matrix $\mathbf{Z}$ onto the set $\mathcal{K}$ is just the matrix itself with all the negative entries set to zero, i.e.

$$proj_{\mathcal{K}}(\mathbf{Z}) = \max(\mathbf{0}, \mathbf{Z}) \tag{27}$$

With that in mind and observing that $\mathbf{Z} - \max(\mathbf{0}, \mathbf{Z}) = \min(\mathbf{0}, \mathbf{Z})$, we can explicitly write the definition of v_k as

$$v_k = \beta_k C + A_1^T(A_1 x_k - b) + A_2^T(A_2 x_k - b) + \min(0, x_k) \tag{28}$$

After implementing the HomotopyCGM algorithm in Python and running it for 10^4 iterations for solving the k-means clustering problem, we get the result shown in Figure 2. The relative residual of the objective function, compared to the true value, is plotted in Figure 3. The final value of the objective function is 42.271, while the optimal value is 57.053. The final objective value is below the optimal value because the solved problem is a SDP-relaxtion of the model-free k-means clustering, which means that the set of solutions is greater than the original one.

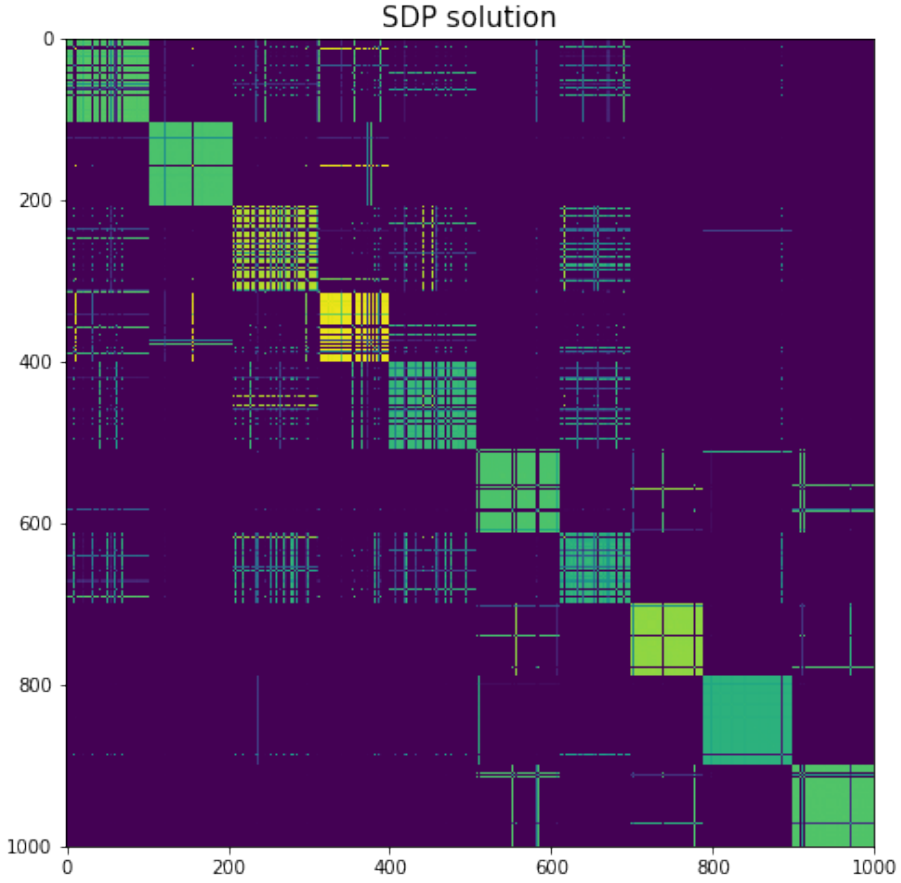

Figure 2: SDP-relaxation to approximately solve the k-means clustering problem.

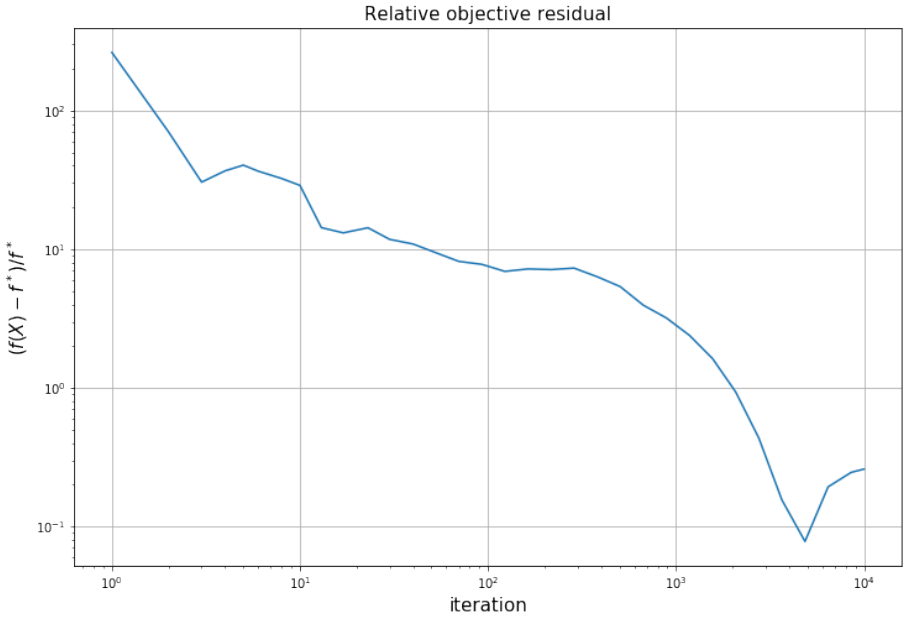

Figure 3: Relative objective residual with respect to the true value f^*.