UNIVERSITÀ DEGLI STUDI DI PADOVA

Bilevel optimization for dictionary learning

Fabiana Rapicavoli 2045245

September 2023

Indice

1 Introduction

1.1 Bilevel optmization

The bilevel programming problem is a hierarchical optimization problem, where a subset of the variables is constrained to be a solution of a given optimization problem, parameterized by the remaining variables.
The bilevel programming problem is a multilevel programming problem with two levels. The hierarchical optimization structure appears naturally in many applications when lower level actions depend on upper level decisions [1], so one problem is embedded within another.
In the paper [2], there is a study of class of a bilevel optimization problems, also known as simple bilevel optimization, where they minimized a smooth objective function over the optimal solution set of another convex constrained optimization problem. In this report, just one of the analyzed problems will be tackle: dictionary learning.

1.2 Dictionary learning

The goal of dictionary learning is to learn a concise representation of the input data from a massive dataset. Let A = $\{a_1, ..., a_n\}$ denote a dataset of *n* points with $a_i \in R^m$ for any $i \in$ N = $\{1, ..., n\}$. The aim is to find a dictionary D = $\{d1, ..., dp\} \in R^{m\times p}$ such that each data point a_i can be well approximated by a linear combination of a few basis vectors in D.

Learning a dictionary through bilevel optmization is useful because, in real applications, data points typically arrive sequentially and the underlying representation may evolve, so, after adding a new piece of information in the dataset A, the dictionary should learn just the new data, without starting again. In order to test this theory, in this project an initial dataset A has been learn though frank wolfe algorthm [3] and, then, another dataset A' has been added and the aim is to expand the already-learned dictionary $\hat{D} \in R^{m\times p}$ and the corresponding coefficient matrix $\hat{X} \in R^{p\times n}$ by learning new basis vectors from A' = $\{a'_1, ..., a'_{\hat{n}}\}$, while retaining the learned information in $\hat{D}$.

2 Method and discussion

As already said in the introduction, the goal is to learn a representation of the input data from a massive dataset A and find a dictionary D, such that each data point a_i can be well approximated by a linear combination of a few basis vectors in D.
We can cast this problem as a simple bilevel problem, where the lower-level loss corresponds to samples from seen tasks, while the upper-level loss captures the error on a new task. The goal is to improve the model using the new task while ensuring that it still performs well over the previous tasks.

The method focuses on the case where the lower-level function g is smooth and convex, while the upper-level function f is smooth but not convex optimization problem.

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p}} \min_{\mathbf{X}\in\mathbb{R}^{p\times n}} \frac{1}{2n}\sum_{i\in\mathcal{N}} \|\mathbf{a}_i - \mathbf{D}\mathbf{x}_i\|_2^2$$

such that $\|d_j\|_2 \leq 1$, j = 1, ..., p; $\|x_i\|_1 \leq \delta$, i $\in$ N, where δ was set to 3.

In order to update the dictionary $\tilde{D} \in R^{m\times q}$, with q > p, and the coefficint matrix $\tilde{X} \in R^{q\times n'}$ for the new dataset A' and, at the same time, to enforce $\tilde{D}$ to well defined also the old dataset $\hat{D}$, with the already-learned coefficient matrix $\hat{X}$, the bilevel problem is the following one:

$$\min_{\tilde{\mathbf{D}}\in\mathbb{R}^{m\times q}} \min_{\tilde{\mathbf{X}}\in\mathbb{R}^{q\times n'}} f(\tilde{\mathbf{D}}, \tilde{\mathbf{X}})$$

such that $\|\tilde{x}_k\|_1 \leq \delta$, where δ was set to 3, k = 1, ..., n'; $\tilde{D} \in$ argmin $g(\tilde{D})$, $\|\tilde{d}\|_2 \leq 1$, where the objective function $f(\tilde{D}, \tilde{X}) = \frac{1}{2n'}\sum_{k=1}^{n'}\|a'_k\text{-}\tilde{D}\tilde{x}_k\|_2^2$ is the average reconstruction error on the new dataset A', the lower-level objective $g(\tilde{D}) = \frac{1}{2n'}\sum_{i=1}^{n}\|a'\text{-}\tilde{D}\tilde{x}_i\|_2^2$ is the error on the old dataset A.

In order to create the two datasets A and A', the following procedure has been exploited. We first generate the true dictionary $D^* \in R^{25\times 50}$ consisting of 50 basis vectors in R25, each of which has its entries drawn from a standard Gaussian distribution and is normalized to have unit ℓ2-norm.
We further construct the two dictionaries D^* and D'^* consisting of the first 40 and the last 20 basis vectors in $\tilde{D}^*$, respectively (and hence they share ten bases in common).
The two datasets A = $\{a_1, . . . , a_{250}\}$ and A' = $\{a'_1, ..., a'_{200}\}$ are generated according to the rules:

- $a_i = D{*}x_i + n_i$,
  where i = 1, . . . , 250;

- $a'_k = D'{*}x_k + n'_k$,
  where k = 1, ..., 200;

where $\{x_i\}_{i=1}^{250}$, $\{x'_k\}_{k=1}^{200}$ are sparse coefficient vectors, each with five nonzero entries, whose locations are randomly chosen, and $\{n_i\}_{i=1}^{250}$, $\{n'_k\}_{k=1}^{200}$ are random Gaussian noise vectors, generated as random tensor matrices.

Then, the dictionary D was randomly set as a tensor matrix with dimension 25x40, while the coefficient matrix X was set as a tensor matrix of zeros with dimension 40x250.

The basis vector were normalized to have bounded ℓ_2-norm and impose ℓ_1-norm constraints to encourage sparsity in $\{x_i\}_{i=1}^n$.

The updating was runned using CG-based bilevel optimization (CG-BiO) algorithm and the new dictionary was retrieved with a maximum number of 1000 iterations. A stopping condition was set, checking if the frank wolfe gap is less than a variable given in input. Whether yes, the algorithms stop in advance.

Algorithm 1 CG-based bilevel optimization (CG-BiO)

```
1:  **Input**: Target accuracy $\epsilon_f, \epsilon_g > 0$, stepsizes $\{\gamma_k\}_k$
2:  **Initialization**: Set $\mathbf{x}_0 \in \mathcal{Z}$ s.t. $g(\mathbf{x}_0) - g^* \leq \epsilon_g/2$
3:  **for** $k = 0, \ldots, K-1$ **do**
4:      Compute $\mathbf{s}_k \leftarrow \text{argmin}_{\mathbf{s} \in \mathcal{X}_k} \langle \nabla f(\mathbf{x}_k), \mathbf{s} \rangle$
        $\mathcal{X}_k \triangleq \{\mathbf{s} \in \mathcal{Z} : \langle \nabla g(\mathbf{x}_k), \mathbf{s} - \mathbf{x}_k \rangle \leq g(\mathbf{x}_0) - g(\mathbf{x}_k)\}$
5:      **if** $\langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{s}_k \rangle \leq \epsilon_f$ and
        $\langle \nabla g(\mathbf{x}_k), \mathbf{x}_k - \mathbf{s}_k \rangle \leq \epsilon_g/2$ **then**
6:          Return $\mathbf{x}_k$ and STOP
7:      **else**
8:          $\mathbf{x}_{k+1} \leftarrow (1 - \gamma_k)\mathbf{x}_k + \gamma_k \mathbf{s}_k$
9:      **end if**
10: **end for**
```

Recall that X_g^* denotes the solution set of the lower-level problem. If we assume $x0 \in X_g^*$, then the CG update at iteration k is given by $x_{k+1} = (1 - \gamma_k)x_k + \gamma_k s_k$ where sk $=$ argmin$(\nabla f(x_k), s)$ and the stepisize $\gamma \in [0,1]$.

The standard CG method needs to be initialized with feasible points for the dictionary D and the coefficient matrix X. These points was set as the original frank wolfe algorithm, runned for 1000 iterations computing both the dictionary D and the coefficient matrix X and, then, for other 1000 iteration just over D.

Algorithm 1 Frank-Wolfe (1956)

```
Let $\boldsymbol{x}^{(0)} \in \mathcal{D}$
**for** $k = 0 \ldots K$ **do**
    Compute $\boldsymbol{s} := \underset{\boldsymbol{s} \in \mathcal{D}}{\arg\min} \langle \boldsymbol{s}, \nabla f(\boldsymbol{x}^{(k)}) \rangle$
    Update $\boldsymbol{x}^{(k+1)} := (1-\gamma)\boldsymbol{x}^{(k)} + \gamma \boldsymbol{s}$,   for $\gamma := \frac{2}{k+2}$
**end for**
```

The initialization was computed, using the first dataset A, dictionary D and coefficient matrix X, which requires at most $O(\frac{1}{\epsilon_g})$ iterations.

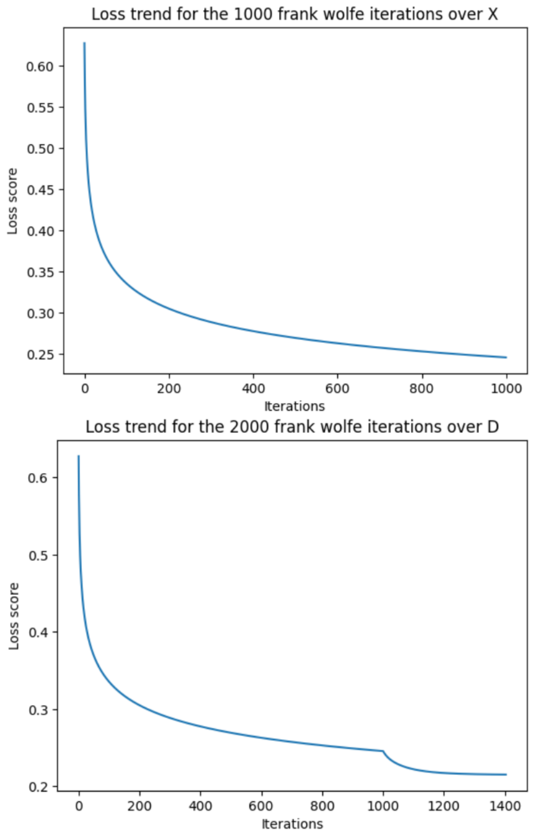

Figura 2.1: Loss trend of the frank wolfe algorithm over the coefficient matrx X and the ditionary D

The loss in figure 2.1 has a decreasing trend for both the coefficient matrx X and the ditio-

nary D results. In the case of X, the algorithm stops after reaching the maximum of possible iterations 1000. In the caso of D, the algorithm stops due to the stopping condition, after 1414 iterations in total and after reaching a frank wolfe gap of value 0.0998.
The trend change direction because, after 1000 iterations, the stepsize change from an adaptive one of 0.1/(k+2), where k is the number of iteration analyzed at that moment, to a fixed one of 0.1.

After that, CG-BIO algorithm was trained with the dataset A', dictionary $\tilde{\mathrm{D}}$ and coefficient matrix $\tilde{\mathrm{X}}$, where the new dictionary $\tilde{\mathrm{D}}$ is the concatenation of the old inizialized dictionary $\hat{D}$ with ten zeros columns, in order to maintain coeherent the multiplications, the old inizialized coefficient matrix $\hat{X}$ was concatenate with ten zeros rows and the new coefficient matrix $\tilde{\mathrm{X}}$ is a random tensor matrix of dimension 50x200.
We assume access to a linear optimization oracle that returns the solution of the subproblem, which is standard for projection-free methods. We repeat the process above until we reach an accuracy of ϵ_{f} for the upper-level objective and an accuracy of ϵ_{g} for the lower-level objective.

The loss in the figure 2.2 has a decreasing trend results and the algorithm stops due to the stopping condition, after 478 iterations and after reaching a frank wolfe gap of value 9.873. The used stepsize has a value equal to $0.3\backslash\sqrt{k+1}$, where k is the number of iteration analyzed at that moment.

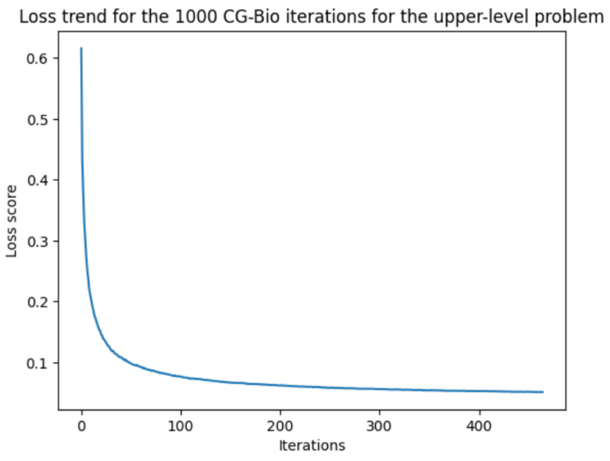

Figura 2.2: Loss trend of the CG-BIO algorithm over the ditionary D

The number of iterations required to find an (ϵ_{f}, ϵ_{g})-optimal solution can be upper bounded by $O(\max\{1\backslash\epsilon_f^2,\ 1\backslash\epsilon_{\mathrm{g}}\epsilon_{\mathrm{f}}\})$. We note that the dependence on the upper-level accuracy ϵ_{f} also matches that in the standard CG method for a single-level problem.

3 Conclusions

In this paper, CG-based method was used to solve the class of bilevel optimization problem of dictionary learning. CG-BiO method finds an $(\epsilon_{\mathrm{f}},\ \epsilon_{\mathrm{g}})$-optimal solution after at most $O(\max\{1\backslash\epsilon_f^2,\ 1\backslash\epsilon_{\mathrm{g}}\epsilon_{\mathrm{f}}\})$ iterations when the upper-level objec-tive f is non-convex.

Bibliografia

[1] Luís Nunes Vicente, CA Floudas, and PM Pardalos. Bilevel programming: Introduction, history and overview. *Encyclopedia of optimization*, 1:178–180, 2009.

[2] Ruichen Jiang, Nazanin Abolfazli, Aryan Mokhtari, and Erfan Yazdandoost Hamedani. A conditional gradient-based method for simple bilevel optimization with convex lower-level problem. In *International Conference on Artificial Intelligence and Statistics*, pages 10305–10323. PMLR, 2023.

[3] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013.