Modeling and Algorithms for Prof Shit's Project

Hongda Li, Yining Zhou, Xiaoping Shi

January 21, 2024

Abstract

We propose some better algorithm for a problem in detecting structure of probability transition matrices from data.

1 Introduction

We describe an optimization problem introduce by Prof Shi and his student Yining. To start we define the following quantities for the optimization problem.

1. $n \in \mathbb{N}$. It denotes the numer of states for a Markov Chain.
2. $p \in \mathbb{R}^{n\times n}$ denotes the probability transition matrix. It's in small case because it's also the variable for the optimization problem. It supports 2 types of indexing, p_{ij} for $i, j \in \{1, \cdots, n\}$, or p_j with $j \in \{1, \cdots, n^2\}$. More on this later.
3. $\eta_{ij} \geq 0$ for $i, j \in \{1, \cdots, n\}$ is a parameter of the problem.
4. $\hat{p}$ is the empirically measured probability transition matrix. They are the maximal likelihood estimators for the transition probability in the transition probability matrix.
5. λ is the regularization parameter.
6. $\mathbf{C}_n^m$ is the combinatoric term that counts all possible subset of size $n, n < m$ from a super set of size m.

When p is referred to as a vector we may say $p \in \mathbb{R}^{n^2}$, if it's referred to as the matrix, we will use $p \in \mathbb{R}^{n\times n}$. When indexing p using a tuple, or a single number, it's possible to translate between the two type of indexing scheme using the following bijective map:

$$
\begin{aligned}
&(i, j) \mapsto k := i \times n + j \\
&k \mapsto (i, j) := (\lfloor k/n \rfloor, \mathrm{mod}(k, n) + 1).
\end{aligned}
$$

We emphasize, in different programming languages and development environments, the convention of indexing a muti-array using different kind of tuples can be very different. For now we use the aboe indexing, which is a row major index convention (Like Python).

1.1 The Optimization Problem

The optimization problem is posed as

$$\underset{p\in\mathbb{R}^{n^2}}{\operatorname{argmin}}\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}-\eta_{ij}\log(p_{i,j})+\lambda\sum_{i=1}^{n^2}\sum_{\substack{j=1\\j\neq i}}^{n^2}\frac{1}{2}\frac{|p_i-p_j|}{|\hat{p}_i-\hat{p}_j|}\right\} \tag{1.1.1}$$

$$\text{s.t:}\quad\left\{\begin{aligned}&\sum_{i=1}^{n}p_{i,j}=1 \quad \forall i=1,\cdots,n\\ &\qquad p_{i,j}\geq 0 \quad \forall i=1,\cdots,n, j=1,\cdots,n\end{aligned}\right\} \tag{1.1.2}$$

There are 3 parts to the optimization problem posed above. It has a smooth differentiable function, the sume of $\eta_{ij}\log(p_{i,j})$ but its gradient is not Lipschitz. The ias a non-smooth part with $p_i - p_j$ for all indices $1,\cdots,n^2$ and $i \neq j$. If $\hat{p}_i - \hat{p}_j = 0$, then we would ignore the term. Finally, it has a linear constraints on all n^2 variables. $p \in \mathbb{R}^{n^2}$ is a vector with the structure $\Delta_n \oplus \Delta_n \oplus \cdots \oplus \Delta_n$. Each Δ_n is a probability simplex. It's defined as $\Delta_n = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_n = 1\}$. For simplicity we just denote using notation $\Delta_{n\times n} = \Delta_n \oplus \Delta_n \oplus \cdots \oplus \Delta_n$.

2 Modeling

The non-smooth part with the absolute value requires some amount of creativity if we were to use common optimization algorithms.

2.1 Representation of the Non-smooth Part

Claim 2.1.1. The nonsmooth objective can be model as

$$\lambda\sum_{i=1}^{n^2}\sum_{j>i}^{n^2}\frac{|p_i-p_j|}{|\hat{p}_i-\hat{p}_j|}=\|Cp\|_1 \qquad \text{where } C \text{ is } \mathbf{C}_2^{n^2} \text{ by } n^2.$$

The one norm represents the summation of absolute values. The transformation $\mathbf{C}p$ is linear transformation and it's a vector of length $\mathbf{C}_2^{n^2}$. The vector is long and it has a dimension of $(1/2)(1+n^2)n^2$. Each term inside of the summation is a row of the matrix C. Each row of matrix C has exactly 2 non-zero elements in it. Suppose that $i \in \{1,\cdots,\mathbf{C}_2^{n^2}\}$ denoting the index for a specific row of matrix C denotes the index of a specific row, and $j \in \{1,\cdots,n^2\}$ denotes a specific column of matrix C. Mathematically describing the matrix is difficult, but it can be algorithmically defined. A sparse matrix format can be described by as a mapping from (i,j), the set of all indices to the element in the vector. The following algorithm 1 construct such a mapping.

Algorithm 1 Matrix Make Algorithm

Let C be a $\mathbf{C}_2^{n^2}$ by n^2 zero matrix.
for $i = 1, \cdots, n^2$ **do**
    for $j = 1, \cdots, n^2$ **do**
        if $|\hat{p}_i - \hat{p}_j| == 0$ **then**
            Break
        end if
        $C[i \times n^2 + j, i] := \lambda / |\hat{p}_i - \hat{p}_j|$
        $C[i \times n^2 + j, j] := -\lambda / |\hat{p}_i - \hat{p}_j|$
    end for
end for

Remark 2.1.1. In practice, we should use sparse matrix data format such as SCR, SCC in programming languages.

2.2 Modeling it for Sequential Quadratic Programming

To use sequential programming, we model the non-smooth $\|Cp\|_1$ parts of the objective function as a linear constraints. Define $u \in \mathbb{R}_+^{\mathbf{C}_2^{n^2}}$ then the below problem is equivalent to the original formulation:

$$\underset{p,u}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} -\eta_{ij} \log(p_{i,j}) + \sum_{i=1}^{C_2^{n^2}} u_k \right\} \tag{2.2.1}$$

$$\text{s.t:} \left\{ \begin{matrix} -u \leq Cp \leq u \\ p \in \Delta_{n \times n} \\ u \in \mathbb{R}_+^{\mathbf{C}_2^{n^2}} \end{matrix} \right\} \tag{2.2.2}$$

This is a Non-linear programming problem and it has a convex objective. Common NLP packages in programming languages can solve this efficiently. However it's potentially possible that these solvers are not adapated to huge sparse matrix C that has special structure to it.

Remark 2.2.1. To formulate into a linear programming with some relaxations, consider that the non-linear objective $-\eta_{ij} \log(p_{i,j})$ can be discretized.

2.3 Modeling it for Operator Splitting

Operator splitting method aims for objective function of the type $f + g$ where f, g are both convex function and they are proximal friendly. And $\text{ri.dom}(f) \cap \text{ri.dom}(g) \neq \emptyset$. Recall that proximal operator of function f is the resolvent operator on the subgradient $(I + \partial f)^{-1}$. Subgradient is just a generalize type of gradient that can handle continuous function that is not necessarily differentiable. To model the original formulation in such a form, we introduce these quantities:

1. $f_1(x_1) : \mathbb{R}^{n^2} \mapsto \mathbb{R} := \sum_{i=1}^{n^2} -\eta_{i,j}(p_{i,j})$
2. $f_2(x_2) : \mathbb{R}^{\mathbf{C}_2^{n^2}} \mapsto \mathbb{R} := \|x_2\|_1$.
3. $f_3(x_3) = \mathbb{R}^{n^2} \mapsto \bar{\mathbb{R}} := \delta_{\Delta_{n\times n}}(x_3)$

The above 3 functions represent the three parts of the summed objective. Let $f(x_1, x_2, x_3) := f_1(x_1) + f_2(x_2) + f_3(x_3)$. However, they share different variables x_1, x_2, x_3, from different dimension. We want g to represents the constraints that $x_2 = Cp$, and $x_1 = x_3 = p$. Simplifying: $x_2 = Cx_1, x_1 = x_3$. This is a linear system of the form

$$\begin{bmatrix} C & -I & \mathbf{0} \\ I & \mathbf{0} & -I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{0}.$$

Simply denote the above as $Ax = \mathbf{0}$, then g is just a non-smooth function that happens to be an indicator function of a convex set. The convex set is the set of all solutions to the linear system. Therefore setting $g = \delta_{Ax=\mathbf{0}}(x)$, to be the indicator of the linear constraints, we had a complete equivalent representation of the original form of the problem.

The matrix A is a $(\mathbf{C}_2^{n^2} + n^2) \times (3n^2)$ matrix. It's still a sparse matrix. Conviently, if we use $\otimes$ then the above matrix admit kronecker product representation:

2.4 Implementation, Software, and Toolings

References