凸优化期末项目报告

1700010674 严格　　1700010679 聂涵韬

2019 年 1 月 16 日

我们的期末项目选择的是第二个项目”Algorithm for large-scale optimal transport”。针对这一问题，我们首先将其表述为一个LP优化问题，并通过Augmented Lagrangian，Sinkhorn，对偶梯度上升等算法对这一问题进行求解。我们对这些算法用matlab给出了实现，在给定的测试数据集上进行了数值实验，与mosek，gurobi这两个常见的优化包使用传统LP问题算法（内点法、单纯形法）进行求解的数值性能进行了对比。以下，我们将逐个回答给出的问题，并展示我们的数值实验结果。

1　第一题

(a) Solve (2.1) by calling mosek and gurobi directly in Matlab or python. The package “CVX” is not allowed to use here. Compare the performance between the simplex methods and interior point methods.

在文件”OT_gurobi.m”与”OT_mosek.m”中，我们分别调用了mosek和 gurobi来求解这一LP问题。mosek使用内点法，gurobi则使用了对偶单纯形法。我们在(c)问中的数据集上测试了这两种方法，对于这两种方法的数值性能的具体比较将在(c)问详细地回答。

(b) Write down and implement a first-order method, forexample, the alternating direction method of multipliers. 在这里，我们选择了Augmented Lagrangian method 来求解这个LP优化问题。对于优化问题：

$$
\begin{aligned}
minimize \quad & < P, C > \\
subject\ to \quad & P\mathbb{1}_m = a \\
& P^T\mathbb{1}_n = b \\
& P_{ij} \geq 0
\end{aligned}
$$

写出其增广拉格朗日函数:

$$L(P, y, z) =< P, C > +I(P) + \frac{t}{2}y^T(P\mathbb{1}_m - a) + \frac{t}{2}y^T(P\mathbb{1}_n - b)$$

应用Augmented Lagrangian method，我们得到了以下的算法：

Algorithm 1 Augmented Lagrangian

set y,z = 0
initialize P
$i = 0$
$dec = 0$
$value = P^T C$
while $i \leq max\_iteration\ and\ dec \leq tolerence$ **do**
  $P = P - step\_size * C - y - z^T + \frac{t}{2}((P\mathbb{1}_m - a) + (P\mathbb{1}_n - b)^T)$
  $P = max(P, 0)$
  $y = y - t(P\mathbb{1}_m - a)$
  $z = z - t(P\mathbb{1}_n - b)$
  $dec = value - P^T C$
  $value = P^T C$
  $i = i + 1$
end while

其中，在求增广拉格朗日函数的最小值的时候，我们通过对L(P,y,z)做一步梯度下降后投影作为一个近似的最小值。当迭代次数达到事先确定的最大迭代步数或下降值小于给定的精度时，退出迭代并返回数值结果。
在这一方法的基础上，我们尝试了改进，加入了自适应步长，算法如下：

Algorithm 2 AdaGrad

set y,z = 0
initialize P
$i = 0$
$dec = 0$
$r = 0$
$mini = 1e-7$
$value = P^T C$
while $i \leq max\_iteration$ *and* $dec \leq tolerence$ **do**
  $\nabla_P L = (C - y - z^T + \frac{t}{2}((P\mathbb{1}_m - a) + (P\mathbb{1}_n - b)^T)$
  $r = r + \|\nabla_P L\|_F^2$
  $P = P - (step\_size/(mini + \sqrt{r}))\nabla_P L$
  $P = max(P, 0)$
  $y = y - t(P\mathbb{1}_m - a)$
  $z = z - t(P\mathbb{1}_n - b)$
  $dec = value - P^T C$
  $value = P^T C$
  $i = i + 1$
end while

关于这一算法的数值表现也将在1(c)部分呈现。

(c) Test problems

我们结合参考文献，采取了以下几种数据集进行测试：

(1)随机生成300维方阵C与300维向量a，b

(2)Ellipse数据集：考虑从起始集S到目标集T的Optimal Transport问题。原始集合是由300个随机分布在单位圆上的点加上标准差$\sigma = 0.1$，均值为0的随机高斯噪声生成的。S为原始集合沿x轴拉伸1.3倍，沿y轴压缩至0.9倍生成， T为原始集合沿y轴拉伸1.1倍，沿x轴压缩至0.9倍生成。

(3)Caffarelli数据集：考虑从起始集S到目标集T的Optimal Transport问题。S为单位圆中均匀分布的300个点，T是将 S中的点沿平行x轴方向远离y轴平移2个单位长度生成的集合。

(4)image数据集，引用自DOTmark - A Benchmark for Discrete Optimal Transport，Jörn Schrieber, Dominic Schuhmacher, Carsten Gottschlich：计算给定图片之间的Wasserstein divergence。在实际测试时，由于运算能

力有限，我们将图片压缩至16*16进行计算。
参数设置：
对测试集(1)-(3)，参数值为：$stepsize = 8e-2, t = 1e-1, tolerence = 1e-12, max\_iteration = 1.5e4$
对测试集(4)，参数值为：$stepsize = 1.6e-2, t = 1e-1, tolerence = 1e-9, max\_iteration = 2e4$
数值结果如下（结果储存在文件‘numresult.mat’与’imageresult.mat’中）：
(1)随机生成数据集：

	CPU time	Cost	relative error(compare with gurobi)	
AL	21.54	1.1375	1.82e-4	
mosek	2.01	1.1373	8.37e-11	
gurobi	1.36	1.1373	0	

表 1: random_dataset

(2)Ellipse数据集:

	CPU time	Cost	relative error(compare with gurobi)	
AL	25.29	30.1142	1.97e-4	
mosek	1.81	30.1082	3.47e-9	
gurobi	1.93	20.1082	0	

表 2: ellipse_dataset

(3)Caffarelli数据集:

	CPU time	Cost	relative error(compare with gurobi)	
AL	20.99	1200	1.34e-5	
mosek	1.81	1200	1.18e-10	
gurobi	1.96	1200	0	

表 3: Caffarelli_dataset

以上表格中数据都是在10组数据上测试结果的平均。
(4)image数据集：

	CPU time	Cost	relative error(compare with gurobi)	
AL	12.33	2.027e5	5.90e-3	
mosek	1.28	2.015e5	1.39e-8	
gurobi	1.25	2.015e5	0	

表 4: image_dataset

这组数据在450组数据上取平均值

单纯形法和内点法的性能比较： 对比使用内点法的mosek和使用对偶单纯形法的gurobi的性能，可以看出，二者在CPU时间上各有优劣，而在数值精度上对偶单纯形稍有优势，但内点法也相当精确。相较于它们，AL方法速度较慢，精度也不如它们。

对AL算法收敛速度的进一步测试 在集合ellipse_1,random_1,Caffarelli_1上运行AL算法，记录相对误差（相对于gurobi）随迭代步数的下降趋势,纵轴为相对误差，横轴为迭代步数(见下图1，2，3)。观察图像，AL算法较为稳定但下降速度很慢。
使用自适应步长改进后与原算法比较（图4），对收敛速度也没有显著的改善。

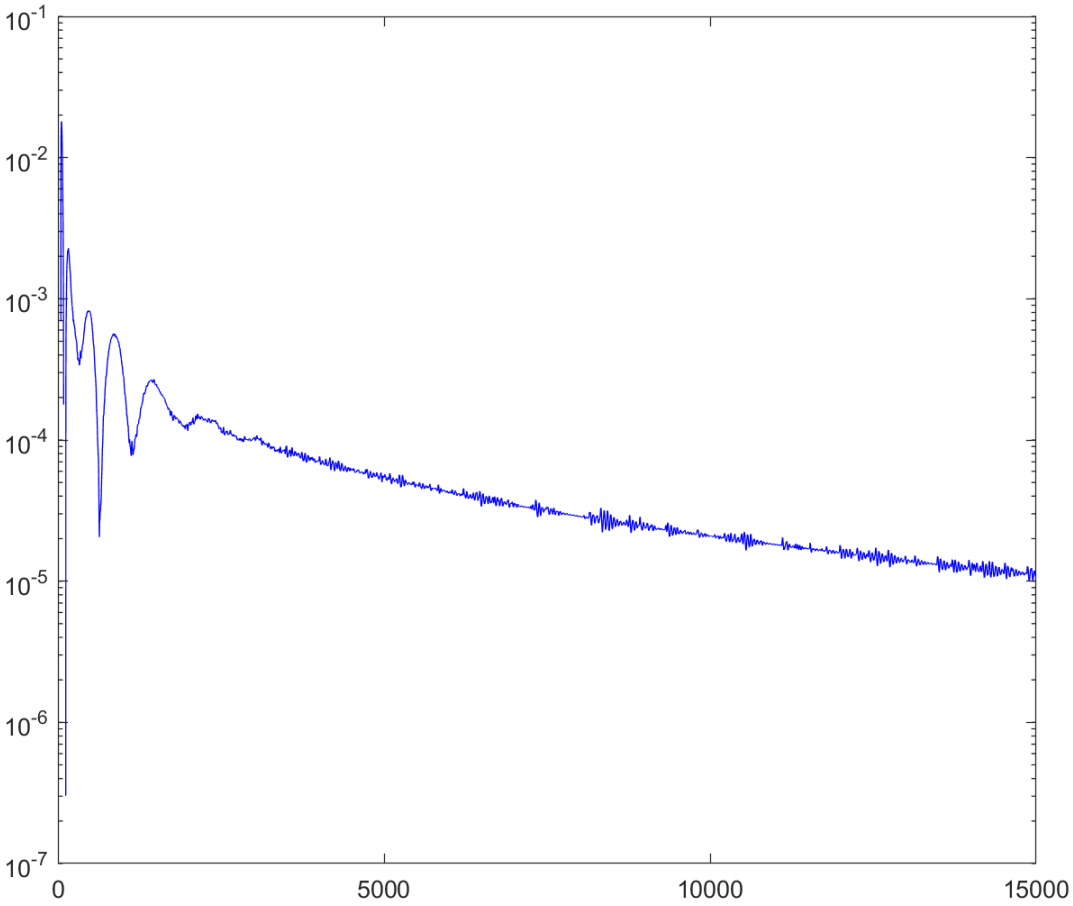


图 1: AL算法在测试集合Caffarelli_1上的相对误差下降曲线

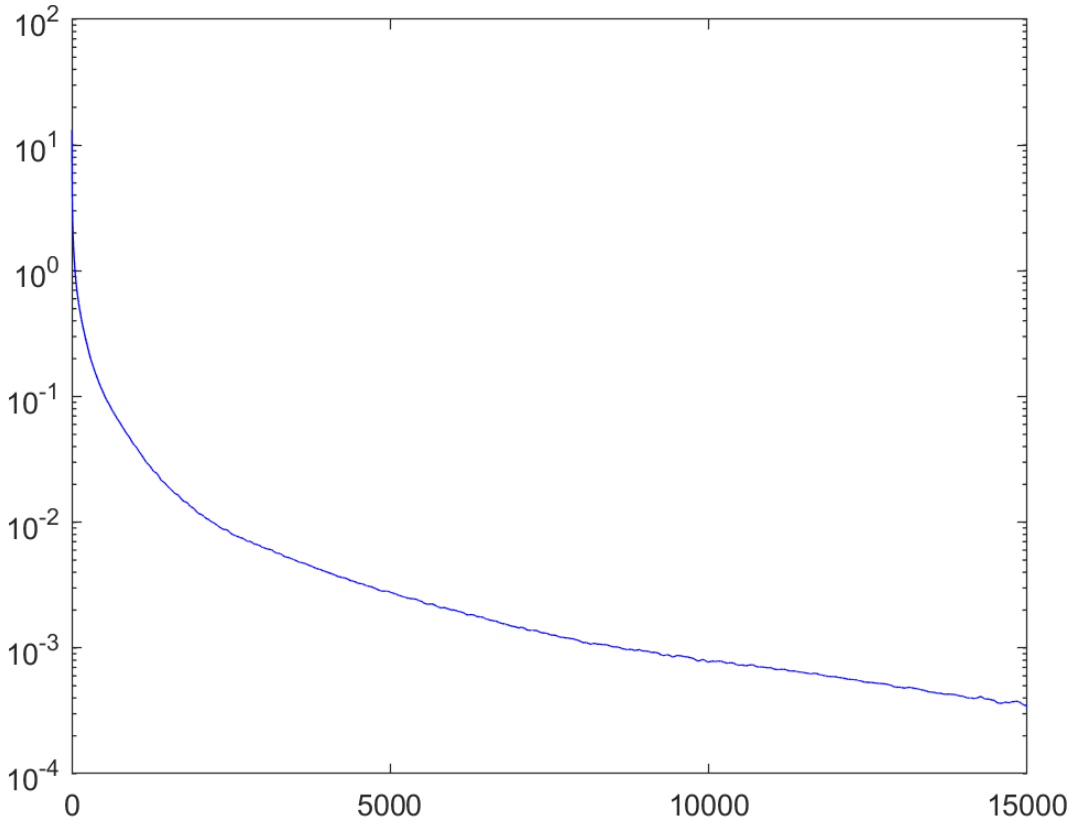


图 2: AL算法在测试集合random_1上的相对误差下降曲线

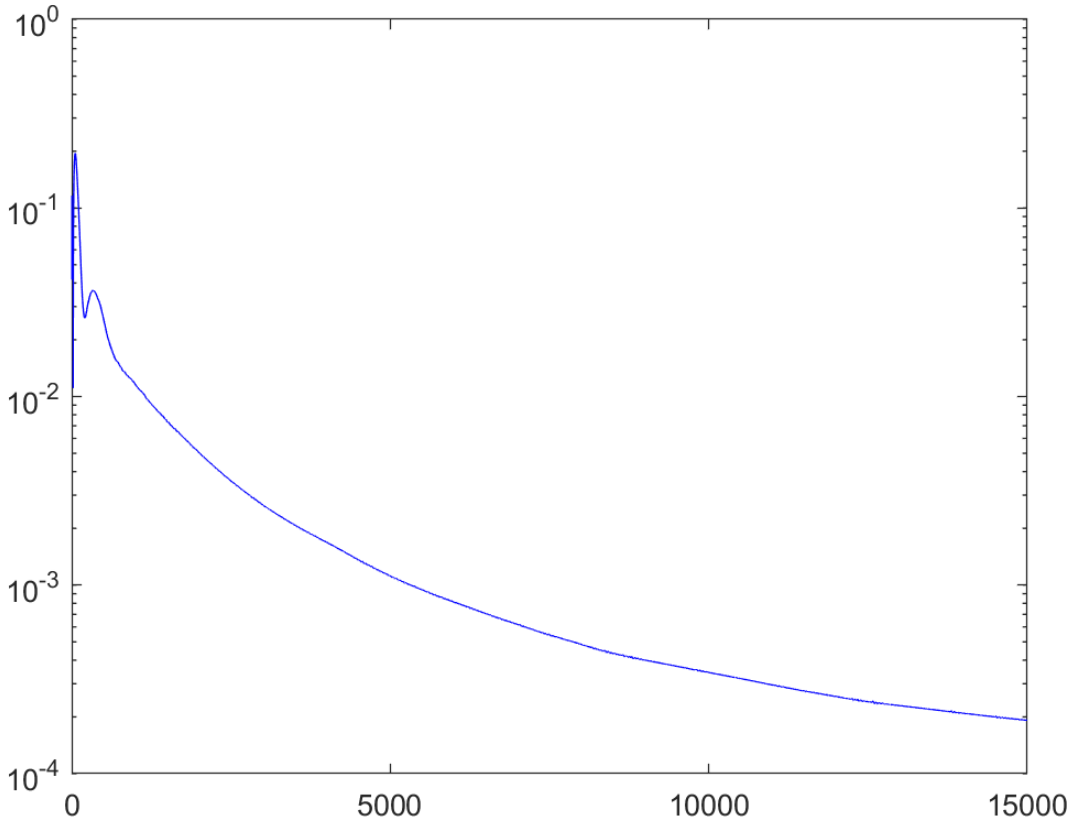


图 3: AL算法在测试集合ellipse_1上的相对误差下降曲线

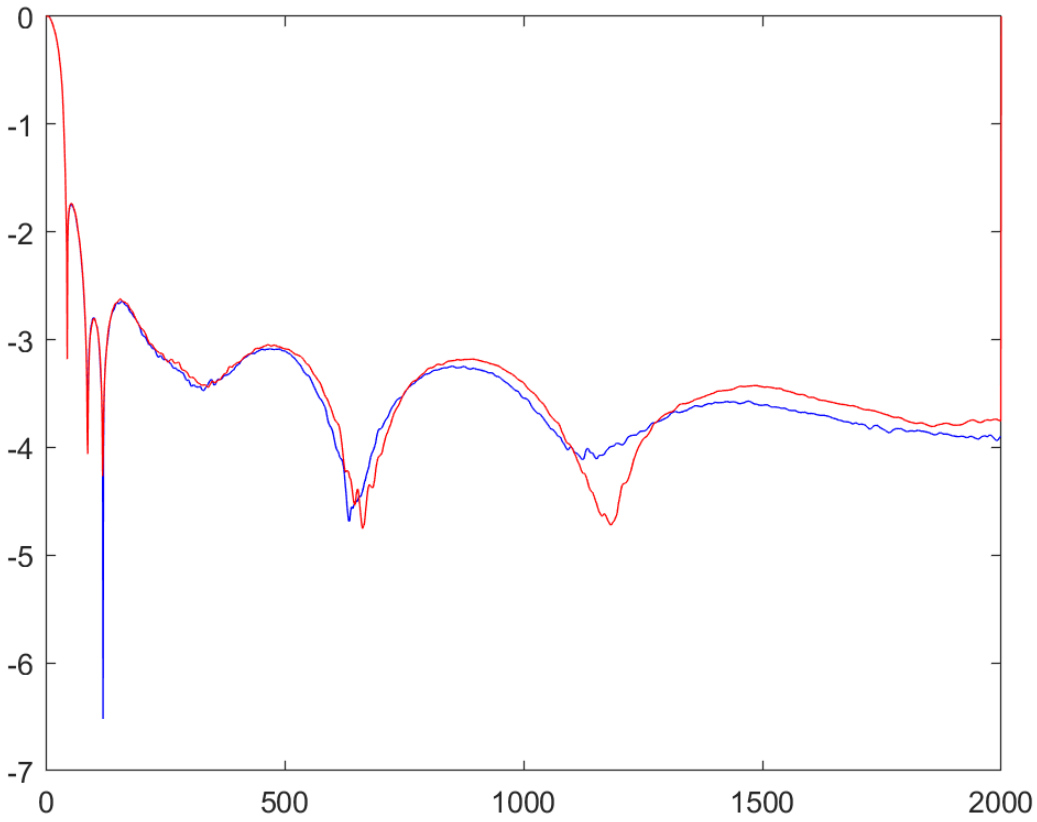


图 4: AL与AdaGrad在测试集合Caffarelli_1上的相对误差比较，红色为AdaGrad，蓝色为AL

2 第二题

(a) Find one of the most important optimization problems from the above references. Write down the background and formulation clearly.

我们认为参考文献中最重要的问题之一带有熵惩罚项的Kantorovich问题(离散情形)，即:

$$\begin{aligned} minimize \quad & L_C^{\epsilon}(a,b) =< P, C > -\epsilon H(P) \\ subject\ to \quad & P\mathbb{1}_m = a \\ & P^T\mathbb{1}_n = b \\ & P_{ij} \geq 0 \end{aligned}$$

其中

$P \in \mathbb{R}^{n\times m}$(n,m与第一问中相反), $a \in \mathbb{R}^n, b \in \mathbb{R}^m, H(P) = -\sum_{ij} P_{ij} log(P_{ij}-1)$

并满足向量a与b的各分量之和均为1。

Kantorovich问题的背景：

Kantorovich问题是Monge问题的松弛。接下来将首先简要介绍Monge问题的背景，然后引出Kantorovich问题及其对偶问题，简要介绍Kantorovich问题及其对偶问题的背景,最后说明增加熵惩罚项的影响。本文讨论的均为离散情形的问题。

(1)Monge问题探讨的是如何衡量两种测度之间的距离。离散情形的Monge问题形式如下：

设

$$\alpha = \sum_{i=1}^{n} a_i \delta_{x_i}$$

是空间$\mathcal{X}$中的一个离散测度，其中$a_i \geq 0$, $\sum_{i=0}^{n} a_i = 1$,δ_{y_i}是狄拉克函数。再设

$$\beta = \sum_{i=1}^{m} b_i \delta_{y_i}$$

是空间$\mathcal{Y}$内的一个测度，其中$b_i \geq 0$, $\sum_{i=0}^{m} b_i = 1$。
现考虑一个映射T：$\{x_1, x_2, ..., x_n\} \to \{y_1, y_2, ..., y_m\}$ 满足

$$b_j = \sum_{i:T(x_i)=y_i} a_i, \quad j = 1, 2, ..., m$$

将其记作$T_{\#}\alpha = \beta$。对任意的$x \in \mathcal{X}, y \in \mathcal{Y}$有代价函数c(x,y)。
Monge问题就是如下的优化问题：

$$\begin{aligned} &minimize \quad \sum_{i=1}^{n} c(x_i, T(x_i)) \\ &subject\ to \quad T_{\#}\alpha = \beta \end{aligned}$$

(2)在Monge问题中，每个x_i只能映往一个y_i。Kantorovich松弛则允许每个x_i可以映往多个y_i。记P_{ij}是从x_i映往一个y_i的成分，记C_{ij}是从x_i映往一个y_i的代价，满足

$$\begin{aligned} a_i &= \sum_{j=1}^{m} P_{ij} \quad i = 1, 2, ..., n \\ b_j &= \sum_{i=1}^{n} P_{ij} \quad j = 1, 2, ..., m \\ P_{ij} &\geq 0, C_{ij} \geq 0 \end{aligned}$$

于是就得到了Kantorovich问题:

$$\begin{aligned} &minimize \quad L_C(a, b) =< P, C > \\ &subject\ to \qquad\qquad P\mathbb{1}_m = a \\ &\qquad\qquad\qquad\qquad P^T\mathbb{1}_n = b \\ &\qquad\qquad\qquad\qquad P_{ij} \geq 0 \end{aligned}$$

(3)Kantorovich问题的对偶问题。

我们可以写出Lagrange对偶函数

$$L(f,g,w)=\inf_{P\in\mathbb{R}^{n\times m}}\sum_{i,j}C_{ij}P_{ij}-f^T(P\mathbb{1}_m-a)-g^T(P^T\mathbb{1}_n-b)-\sum_{i,j}w_{ij}P_{ij}$$

$$=\begin{cases}f^Ta+g^Tb & 若C_{ij}-a_i-b_j-w_{ij}=0,\forall i,j,1\le i\le n,1\le j\le m\\ -\infty & 其它情况\end{cases}$$

其中

$$f\in\mathbb{R}^n,g\in\mathbb{R}^m,w\in\mathbb{R}_+^{n\times m}$$

于是原问题的对偶问题为：

$$\begin{aligned}maximize\ \ & y^Ta+z^Tb\\ subject\ to\ \ & c_{ij}-a_i-b_j-w_{ij}=0,\\ & w_{ij}\ge 0,\\ & \forall i,j,1\le i\le m,1\le j\le n\end{aligned}$$

或者:

$$\begin{aligned}maximize\ \ & <f,a>+<g,b>\\ subject\ to\ \ & f\oplus g\le C\end{aligned}$$

由于线性规划满足Slatter条件，所以对偶问题的解和原问题的解是一样的。

(4)Kantorovich问题及其对偶问题的背景。

对于Kantorovich问题，我们可以有这样一个形象的理解：我们有n家面包店，第i家面包店的存货量为a_i，现在它们要把面包配送给m个客户，第j位客户的需求量为b_i。假设从第i家面包店运送至第j位客户的面包量为P_{ij}，每单位面包的运输成本为c_{ij}。我们需要最小化将面包配送给客户的总成本。

对偶问题也有一个形象的理解：假设现在有了一家快递公司，快递公司将面包从面包店全部运往中转站，再从中转站运往各客户。快递公司向面包和客户分别收取运费，设第i面包店将一单位面包运往中转站的运费为f_i,而中转站将一单位面包运往第j位客户的运费为g_j。我们的对偶问题就是站在快递公司的角度，选择一种定价方案，使得赚取的运费最大化。

约束条件$C_{ij}-a_i-b_j\ge 0$可以这样理解：如果$c_{ij}-a_i-b_j>0$,那么相应的面包店和客户将不使用快递公司的服务，从而快递公司将失去这笔生意，也就是说这样的定价策略是不合理的。

至于f_i，g_j的正负性，这无关紧要，因为若运费为负，则可理解为快递公司倒贴钱，但快递公司会从其他人手中把钱赚回来，以使自身利益最大化。
(5)熵惩罚项的影响。
增加熵惩罚项后，目标函数变为ϵ-强凸函数（因为$\nabla^2 H(P) = -diag(1/P_{ij})$而$P_{ij} \leq 1$），所以保证了该优化问题一定有唯一可行解。
当$\epsilon \to 0$时,$L_C^\epsilon(a,b) \to L_C(a,b)$并且$P^{\epsilon *} \to P*$。所以当$\epsilon$充分小时，我们可以保证算法在有限步内收敛到一个次优解。

(b) Write and implement an algorithm for the optimization problem in 2(a) from the chosen reference. Try to reproduce the numerical results in that reference.

我们选择的是Sinkhorn算法。接下来我们将简要推导Sinkhorn算法，然后测试数据，还原第一篇reference中的数值结果，最后给出简要评价和改进。
（1）Sinkhorn算法的推导
写出问题

$$\begin{aligned} minimize \quad & L_C^\epsilon(a,b) =< P, C > -\epsilon H(P) \\ subject\ to \quad & P\mathbb{1}_m = a \\ & P^T\mathbb{1}_n = b \\ & P_{ij} \geq 0 \end{aligned}$$

的拉格朗日对偶函数:

$$L(P,f,g,w) =< P, C > -\epsilon H(P) - f^T(P\mathbb{1}_m - a) - g^T(P^T\mathbb{1}_n - b) - < W, P >$$

其中$W_{ij} \geq 0$

$$\frac{\partial L}{\partial P_{ij}} = C_{ij} + \epsilon log(P_{ij}) - f_i - g_j - W_{ij}$$

根据KKT条件，在P取到最优解时应有

$$\frac{\partial L}{\partial P_{ij}} = 0$$

可得

$$P_{ij} = exp(\frac{-C_{ij} + f_i + g_j + W_{ij}}{\epsilon})$$

设矩阵$K \in \mathbb{R}^{n \times m}, u \in \mathbb{R}^n, v \in \mathbb{R}^m$满足：

$$K_{ij} = exp(-\frac{C_{ij}}{\epsilon}), i = 1,2,...,n, j = 1,2,...,m$$

$$u_i = exp(\frac{f_i}{\epsilon}), i = 1, 2, ..., n$$

$$v_j = exp(\frac{g_j}{\epsilon}), j = 1, 2, ..., m$$

那么最优解P满足：

$$P = diag(u)Kdiag(v)$$

结合约束条件

$$P\mathbb{1}_m = a$$

$$P^T\mathbb{1}_n = b$$

可以得到

$$diag(u)Kdiag(v)\mathbb{1}_m = a$$

$$(diag(u)Kdiag(v))^T\mathbb{1}_n = b$$

等价于

$$u \odot (Kv) = a$$

$$v \odot (K^Tu) = b$$

于是设计Sinkhorn算法如下,其中后半部分是从u，v还原出P。

Algorithm 3 Sinkhorn

set $k = 0$

initialize $v^{(0)} \in \mathbb{R}^m$

while $k <$iterate_time **do**

$u^{(k+1)} = \frac{a}{Kv^{(k)}}$

$v^{(k+1)} = \frac{b}{K^Tu^{(k+1)}}$

$k = k + 1$

end while

$u^{'} = u \odot min(\frac{a}{u \odot (Kv)}, 1)$

$v^{'} = v \odot min(\frac{b}{v \odot (K^Tu)}, 1)$

$\Delta a = a - u^{'} \odot (Kv)$

$\Delta b = b - v^{'} \odot (K^Tu)$

$P = diag(u^{'})Kdiag(v^{'}) + \frac{\Delta a \Delta b^T}{\|\Delta a\|_1}$

（2）测试数据并复现reference中的数值结果。我们尝试复现第一篇reference《Computational Optimal Transport》第64页的图4.5。现将改图拷贝至此：

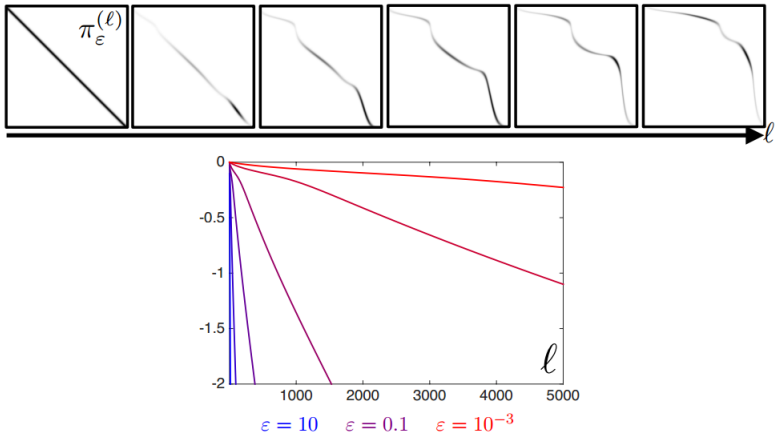


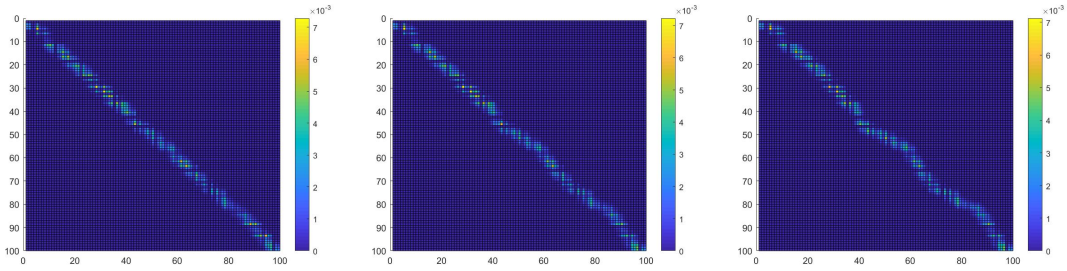
Figure 4.5: Top: evolution of the coupling $\pi_\varepsilon^{(\ell)} = \mathrm{diag}(\mathbf{u}^{(\ell)})\mathbf{K}\,\mathrm{diag}(\mathbf{v}^{(\ell)})$ computed at iteration ℓ of Sinkhorn's iterations, for 1-D densities on $\mathcal{X} = [0,1]$, $c(x,y) = |x-y|^2$, and $\varepsilon = 0.1$. Bottom: impact of ε the convergence rate of Sinkhorn, as measured in term of marginal constraint violation $\log(\|\pi_\varepsilon^{(\ell)}\mathbb{1}_m - \mathbf{b}\|_1)$.

(原文中figure4.5下方注释的最后一行中的b打错了，应为a) 图4.5上方的子图：取[0,1]上的两个随机的测度，取n=100,m=100来近似连续情形,代价矩阵$C \in \mathbb{R}^{n\times m}$,满足

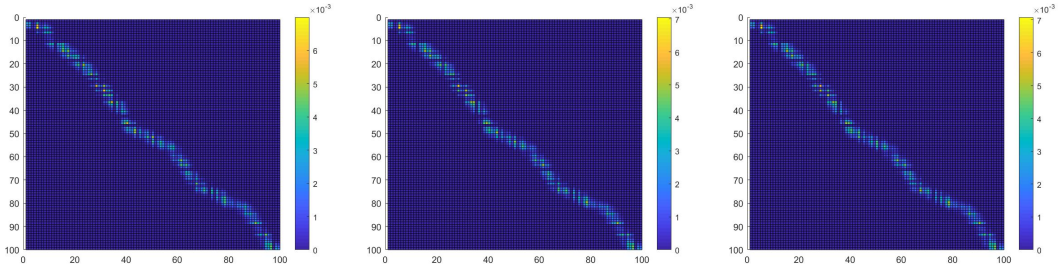
$$C_{ij} = (\frac{i}{n} - \frac{j}{m})^2$$

取ϵ=1e-3,用Sinkhorn算法来解该优化问题，并依次画出经过1,4,16,64,256,1024个Sinkhorn迭代后的P=diag(u)Kdiag(v),如图：从颜色表中可以看到深蓝色区域对应的值为0，这表明解P是稀疏的，并且其非零值的位置都对应代价矩阵中代价小的位置，这和我们的直觉是一致的。

我们得到的图像和reference《Computational Optimal Transport》中的图4.5上方的子图的主要特征是相似的:P的非零值首先集中在对角线上，然后向最优解的方向逐渐移动。但也有一些不一样的地方，对此我们可以做一些解释。由于取的是随机的测度，所以最终的最优解是不一样的，因此最后一张图的形状有明显的差别。我们这里取的ϵ=1e-3,我们尝试过按照reference《Computational Optimal Transport》中的图4.5的注释取ϵ=0.1，但此时解的稀疏性变得不明显，和reference《Computational Optimal Transport》中的图4.5差得很远。此外，我们取了n=100,m=100，用离散测度来近似连续测度，受制于电脑内存有限，这里取得n,m的值并不是很大。



l=1　　l=4　　l=16



l=64　　l=256　　l=1024

图4.5下方的子图：取n=100,m=100,随机向量$a \in \mathbb{R}^n, b \in \mathbb{R}^m$,代价矩阵$C \in \mathbb{R}^{n\times m}$,满足

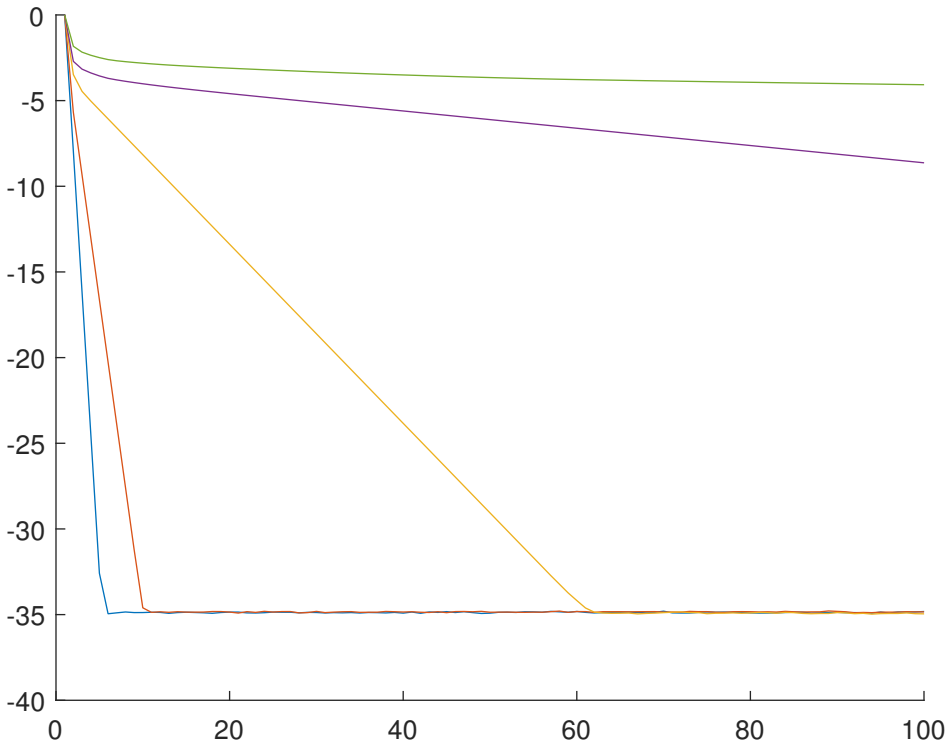
$$C_{ij} = (\frac{i}{n} - \frac{j}{m})^2$$

记

$$P^{(l)} = diag(u^{(l)})Kdiag(v^{(l)})$$

画出依次取ϵ=10,1,0.1,1e-2,1e-3时$log(\|P^{(l)}\mathbb{1}_m - a\|_1)$随l的变化图：

这和reference《Computational Optimal Transport》中的图4.5下方的子图十分相似,只不过我们的Sinkhorn收敛速度貌似更快，猜测可能是由取的n,m的规模差异造成。



从左到右依次为ϵ=10,1,0.1,1e-2,1e-3

(3)Sinkhorn算法的简要评价与改进 Sinkhorn算法在ϵ较大时收敛速度较快，但是由于惩罚项的存在，Sinkhorn得到的最优解仅仅是原 Kantorovich问题的次优解，相比于原Kantorovich问题的最优解有偏差，并且ϵ越大，偏差越多。

另一方面,当ϵ较小时，尽管此时Sinkhorn算法的最优解可以良好的近似原Kantorovich问题的最优解，但出现另一方面的问题。当ϵ较小时，如$\epsilon < 1e{-}4$，Sinkhorn算法就会开始失去数值稳定性，可能发生下溢情况，并且测试表明ϵ越小，越容易发生下溢。

可以对Sinkhorn算法做如下改进，避免数值不稳定性。改进的Sinkhorn算法成为log-domain-Sinkhorn算法。

对一个向量$z \in \mathbb{R}^k$，用

$$min_\epsilon(z) = min(z) - \epsilon log(\sum_i^k e^{z_i - min(z)})$$

来近似

$$-\epsilon log(\sum_i^k e^{z_i})$$

这样就可以有效避免数值下溢。将其代入Sinkhorn算法中。

对于矩阵$A \in \mathbb{R}^{n\times m}$ 定义$Min_\epsilon^{row}(A) \in \mathbb{R}^n$满足

$$Min_\epsilon^{row}(A)_i = min_\epsilon((A_{i1}, A_{i2}, ..., A_{in}))$$

定义$Min_\epsilon^{col}(A) \in \mathbb{R}^m$满足

$$Min_\epsilon^{row}(A)_j = min_\epsilon((A_{1j}, A_{2j}, ..., A_{nj}))$$

log-domain-Sinkhorn算法如下，其中后半部分是为了由f，g还原出P。 log-domain-Sinkhorn确实能够避免下溢，但也付出了高昂的时间代价，实用性不强。

Algorithm 4 Log-domain-Sinkhorn

set $k = 0$

initialize $f^{(0)} \in \mathbb{R}^n, g^{(0)} \in \mathbb{R}^m$

while $k <$iterate_time **do**

$f^{(k+1)} = Min_\epsilon^{row}(C - f^{(k)}\mathbb{1}_{1\times m} - \mathbb{1}_{n\times 1}(g^{(k)})^T) + f^{(k)} + \epsilon log(a)$

$g^{(k+1)} = Min_\epsilon^{row}(C - f^{(k+1)}\mathbb{1}_{1\times m} - \mathbb{1}_{n\times 1}(g^{(k)})^T) + g^{(k)} + \epsilon log(b)$

$k = k + 1$

end while

$K = e^{-\frac{C}{\epsilon}}$

$u = e^{\frac{f^{(k)}}{\epsilon}}$

$v = e^{\frac{g^{(k)}}{\epsilon}}$

$u^{'} = u \odot min(\frac{a}{u\odot(Kv)}, 1)$

$v^{'} = v \odot min(\frac{b}{v\odot(K^T u)}, 1)$

$\Delta a = a - u^{'} \odot (Kv)$

$\Delta b = b - v^{'} \odot (K^T u)$

$P = diag(u^{'})Kdiag(v^{'}) + \frac{\Delta a \Delta b^T}{\|\Delta a\|_1}$

(c) Try to write down and implement an algorithm covered in this course for the optimization problem in 2(a). This algorithm should be different from the one in 2(b)

我们尝试用梯度上升方法解带熵惩罚项的Kantorovich问题的对偶问题。

我们写出对偶问题：

$$maximize \;\; L^\epsilon =< f, a > + < g, b > -\epsilon \sum_{ij} e^{\frac{f_i + g_j - C_{ij}}{\epsilon}}$$

没有约束。Kantorovich对偶问题的约束$f_i + g_j \leq C_{ij}$体现在惩罚项$\epsilon \sum_{ij} e^{\frac{f_i+g_j-C_{ij}}{\epsilon}}$中。由于带熵惩罚项的Kantorovich问题满足Slatter条件，所以对偶问题的解等于原问题的解。

求导,得：

$$\frac{\partial L^\epsilon}{\partial f_i} = a_i - \sum_{j=1}^{m} e^{\frac{f_i+g_j-C_{ij}}{\epsilon}}$$

$$\frac{\partial L^{\epsilon}}{\partial g_j} = b_i - \sum_{i=1}^{n} e^{\frac{f_i+g_j-C_{ij}}{\epsilon}}$$

设计梯度上升方法如下,后半部分是为了由f，g还原出P。

Algorithm 5 GA on dual problem

set $k = 0$

initialize $f^{(0)} \in \mathbb{R}^n, g^{(0)} \in \mathbb{R}^m$

while $k <$iterate_time **do**

$f_i^{(k+1)} = f^{(k)} + step\_size * \frac{\partial L^{\epsilon}}{\partial f_i}(f^{(k)}, g^{(k)})$

$g_i^{(k+1)} = f^{(k)} + step\_size * \frac{\partial L^{\epsilon}}{\partial g_i}(f^{(k)}, g^{(k)})$

$k = k + 1$

end while

$K = e^{-\frac{C}{\epsilon}}$

$u = e^{\frac{f^{(k)}}{\epsilon}}$

$v = e^{\frac{g^{(k)}}{\epsilon}}$

$u^{'} = u \odot min(\frac{a}{u \odot (Kv)}, 1)$

$v^{'} = v \odot min(\frac{b}{v \odot (K^T u)}, 1)$

$\Delta a = a - u^{'} \odot (Kv)$

$\Delta b = b - v^{'} \odot (K^T u)$

$P = diag(u^{'}) K diag(v^{'}) + \frac{\Delta a \Delta b^T}{\|\Delta a\|_1}$

做一个数值测试,取一个较好的结果如下：

n=100,m=100

相对误差等于$(L^{\epsilon} - L^{\epsilon}_{Sinkhorn})/L^{\epsilon}_{Sinkhorn}$

	ϵ	迭代步数	step_size	耗时	L^{ϵ}	相对误差
Sinkhorn	3e-4	1e4		0.17	1.1101237805	0
log-domain-Sinkhorn	3e-4	1e4		4.59	1.1101237805	2.440218e-14
GAonDual	3e-4	1e5	1e-2	10.90	1.1101237769	-3.212515e-09

表 5: 数值试验结果

我们的梯度上升方法收敛速度显著慢于Sinkhorn，迭代步数约为Sinkhorn的4到10倍才能收敛

3 小组分工

本项目由聂涵韬和严格合作完成，小组分工如下：
聂涵韬:增广拉格朗日算法,Sinkhorn,log-domain-Sinkhorn，梯度上升算法推导及实现，2(b)中数值测试及分析。
严格：1(c)中数据集合的构造和处理，1中算法测试和性能分析。
本项目文件的主要内容没有用在其它课程做为课程项目或作业提交。