等式约束熵极大化问题的求解

概述

等式约束熵极大化问题是一个典型的含等式约束的凸优化问题。本文以MATLAB R2017b作为实验平台，使用从可行初始点出发、从不可行初始点出发的牛顿方法和对偶方法来求解熵极大化问题，并对求解结果进行评估。

问题描述

等式约束熵极大化问题的形式如下：

$$min\ f(x)=\sum_{i=1}^{n}x_i log(x_i), s.t. Ax=b, A\in\mathbb{R}^{p*n}$$

目标函数的梯度$\bigtriangledown f(x)$是一个n维向量，二阶梯度$\bigtriangledown f(x)$是$n*n$维矩阵，计算可得：

$$\bigtriangledown f(x)=\ （\frac{\partial f(x)}{\partial x_i}）_{n*1}=\begin{bmatrix} log(x_1)+1 \\ \vdots \\ log(x_i)+1 \\ \vdots \\ log(x_n)+1 \end{bmatrix}$$

$$\bigtriangledown^2 f(x)=\ （\frac{\partial^2 f(x)}{\partial x_i \partial x_j}）_{n*n}=diag(x_i^{-1})$$

函数的程序实现如下所示。

```
% The Objective Function is defined in f.m
function y = f(x)
y = sum(x.*log(x));
% The Gradient of Objective Function is defined in gf.m
function y = gf(x)
y = log(x) + 1;
```

```
% The 2nd Gradient is defined in sgf.m
function y = sgf(x)
y = diag(1./x);
```

求解的初始设定

本文针对上述问题取$n = 100, p = 30$的实例进行求解。我们随机构造矩阵A，并保证其满秩性：

```
% Environment Initialization
n = 100;
p = 30;
% Generating A
A = 0.1 + rand(p, n);
while(rank(A) ~= p)
        A = 0.1 + rand(p, n);
end
```

随机生成n维向量$\hat{x}$并取$b = A\hat{x}$，$\hat{x}$必定符合等式条件。另外生成一个不符合等式条件的点$\tilde{x}$，下文取用这两个点作为可行、不可行初始点。

```
% Common Error
epsilon = 1e-2;
% Generating xAble, xUnable and b
xAble = 0.1 + rand(n, 1);
b = A * xAble;
xUnable = 0.1 + rand(n, 1);
while(max(abs(A * xUnable)) <= epsilon)
        xUnable = rand(n, 1);
end
```

生成这些量时增加了常数0.1，目的是保证函数在实际求解时二阶梯度的下界m不太小，从而充分利用函数的强凸性条件。

从可行点出发的牛顿方法

下降时采取的牛顿方向由如下的方程组确定：

$$\begin{bmatrix} \bigtriangledown^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} d_x \\ w \end{bmatrix} = \begin{bmatrix} -\bigtriangledown f(x) \\ 0 \end{bmatrix}$$

没能针对实际问题优化这个方程组的求解，所以直接用逆矩阵求解下降方向：

$$\begin{bmatrix} d_x \\ w \end{bmatrix} = \begin{bmatrix} \bigtriangledown^2 f(x) & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\bigtriangledown f(x) \\ 0 \end{bmatrix}$$

```
% Descending Direction
AA = [sgf(xUpdated), A'; A, zeros(p)];
bb = [-gf(xUpdated); zeros(p, 1)];
dx = inv(AA) * bb;
dx = dx(1: n, 1);
```

牛顿减小量$\lambda^2 = d_x^T \bigtriangledown^2 f(x) d_x$为整个过程中的下降量，迭代过程使牛顿减小量的值逼近0：

```
% Breaking the Loop
lambdaSquare = dx' * sgf(xUpdated) * dx
if(lambdaSquare <= 2 * e)
        break
end
```

步长的回溯停止条件：$f(x + td_x) < f(x) - \alpha t \lambda^2$

```
% Step Length
t = 1;
while(f(xUpdated + t * dx) > f(xUpdated) - alpha * t * lambdaSquare)
        t = beta * t;
end
```

最后，按照计算得到的步长和方向确定新的序列点。

```
% Deciding the Next Sequential Point
xUpdated = xUpdated + t * dx;
```

从不可行点出发的牛顿方法

下降时采取的牛顿方向由如下的方程组确定：

$$\begin{bmatrix} \bigtriangledown^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_v \end{bmatrix} = -r = \begin{bmatrix} -(\bigtriangledown f(x) + A^T v) \\ -(Ax - b) \end{bmatrix}$$

依然通过直接取逆矩阵相乘的方法求解牛顿方向，如下所示：

```
% Descending Direction
AA = [sgf(xUpdated), A'; A, zeros(p)];
bb = [-gf(xUpdated) - A' * vUpdate; -A * xUpdated + b];
dx = inv(AA) * bb;
dx = dx(1: n, 1);
```

原对偶残差为整个过程中的下降量:

$$r\left(\begin{array}{c} x \\ v \end{array} \right) = \left[\begin{array}{c} r_{dual}(y) = \bigtriangledown f(x) + A^T v \\ r_{pri}(y) = Ax - b \end{array} \right]$$

```
% The Residual Function is defined in r.m
function y = r(x, v, A, b)
y = [gf(x) + A' * v; A * x - b];
```

v的初始值是任意给定的：

```
% Initialization
v = 0.1 + rand(p, 1);
```

迭代过程是优化原对偶残差的过程。原对偶残差最终逼近0，转为可行点（$r_{pri}(y) = 0$）的同时也成为原问题的最优解（$r_{dual}(y) = 0$）.因此，迭代停止条件也由原对偶残差约束。此处采用其2范数作为衡量标准：

```
% Breaking the Loop
if(norm(r(xUpdate, vUpdate), 2) <= 2 * e)
        break
end
```

步长的回溯停止条件：

$$||r(y + td_y)||_2 > (1 - \alpha t)||r||_2, y = \left(\begin{array}{c} x \\ v \end{array} \right)$$

```
% Step Length
t = 1;
while(norm(r(xUpdated + t * dx, vUpdated + t * dv, A, b), 2)
      > (1 - alpha * t)norm(r(xUpdated, vUpdated, A, b), 2)
        t = beta * t;
end
```

最后，按照计算得到的步长和方向确定新的序列点（和对偶变量）。

```
% Deciding the Next Sequential Point
xUpdated = xUpdated + t * dx;
vUpdated = vUpdated + t * dv;
```

改变原问题的结构求解

考虑到原问题是等式约束的优化问题，一种思路是先将原问题转化为对偶问题，再用牛顿方法求解。

$$L(x,v)=\sum_{i=1}^{m}x_i log(x_i)+\sum_{i=1}^{m}v^TA(:,i)x_i-v^Tb$$

$$\because argmin\{x_i:L(x,v)\}=e^{-v^TA(:,i)-1}$$

$$\therefore L_D(v)=-v^Tb-\sum_{i=1}^{m}e^{-v^TA(:,i)-1}$$

原问题转化为

$$\underset{v\geq 0}{max}\ L_d(v)\Rightarrow \underset{v\geq 0}{min}\ f(v)=ev^Tb+\sum_{i=1}^{m}e^{-v^TA(:,i)}$$

其中

$$(\bigtriangledown f(v))_j=eb_j-\sum_{i=1}^{m}A(j,i)e^{-v^TA(:,i)}$$

$$(\bigtriangledown^2 f(v))_{j,k}=\sum_{i=1}^{m}A(j,i)A(k,i)e^{-v^TA(:,i)}$$

函数实现如下所示：

```
% The Objective Function is defined in f.m (An another folder)
function y = f(v, A, b)
y = e * v' * b;
for i = 1: m
        y = y + exp(-v' * A(:, i));
end
% The Gradient of Objective Function is defined in gf.m
function y = gf(v, A, b)
y = e * b;
for j = 1: m
        for i = 1: m
                y(j) = y(j) - A(j, i) * exp(-v' * A(:, i));
        end
end
% The 2nd Gradient is defined in sgf.m
function y = sgf(v, A, b)
y = zeros(m, m);
for j = 1: m
        for k = 1: m
                for i = 1: m
                        y(j, k) = y(j, k) + A(j, i) * A(k, i) * exp(-v' * A(:, i));
                end
        end
end
```

下降时采取的牛顿方向由如下的方程组确定：
$d_{nt} = - \bigtriangledown^2 f(x)^{-1} \bigtriangledown f(x)$

```
% Descending Direction
dnt = -inv(sgf(v, A, b)) * gf(v, A, b);
```

牛顿减小量$\lambda^2 = d_{nt}^T \bigtriangledown^2 f(x) d_{nt}$为整个过程中的下降量，迭代过程使牛顿减小量的值逼近0：

```
% Breaking the Loop
lambdaSquare = dnt' * sgf(xUpdated) * dnt
if(lambdaSquare <= 2 * e)
        break
end
```

步长的回溯停止条件：$f(x+td_{nt}) < f(x) - \alpha t\lambda^2$

```
% Step Length
t = 1;
while(f(xUpdated + t * dnt) > f(xUpdated) - alpha * t * lambdaSquare)
        t = beta * t;
end
```

最后，按照计算得到的步长和方向确定新的序列点。

```
% Deciding the Next Sequential Point
xUpdated = xUpdated + t * dnt;
```

求解情况

对随机生成的可行解使用上述三种办法求解，设迭代终止误差$e = 10^{-8}$，结果如下：

方法	迭代步数	在阻尼牛顿阶段的下降趋势	进入二次收敛阶段的误差量级	在二次收敛阶段的迭代步数
可行点牛顿	8	基本等差下降	1e0	3
不可行点牛顿	8	看上去比等差下降快	1e0	3
对偶牛顿	8	基本等差下降	1e0	3

减小量的图示及分析见附录。
主要现象总结为如下几点：

- 三种牛顿方法下降的步长、途经的序列点各有不同，但收敛速度基本类似；
- 不可行牛顿方法的原对偶残差的缩减确实在一条直线上，残差的方向几乎不变化；
- 回溯法确定步长是主要耗时操作，对参数β的估计决定了迭代次数。在实际求解中，可以通过观察前几步确定下来的步长来认为调整公比β的值，以防止内部迭代次数过多。