浙江大学

本科实验报告

课程名称：应用运筹学基础

实验名称：编程实现单纯形法和对偶单纯形法

姓　　名：胡晨旭 蒋仕彪 刘明锐 章启航

学　　号：3170104243 3170102587 3170105696 3170104343

专　　业：求是科学班（计算机）

指导教师：张国川

2020 年 01 月 07 日

应用运筹学基础-实验

1. Simplex 实现

1.1 原理说明

- **线性规划（Linear Programming）**

$$\begin{array}{ll}\min\limits_x & c^Tx+d \\ \text{s.t.} & Ax\le b \\ & Px=q\end{array} \Rightarrow \begin{array}{ll}\max\limits_x & c^Tx \\ \text{s.t.} & Ax\le b \\ & Px=q \\ & x\ge 0\end{array} \Rightarrow \begin{array}{ll}\max\limits_x & c^Tx \\ \text{s.t.} & Ax=b \\ & x\ge 0\end{array}$$

  - 由于仿射函数既是凸函数又是凹函数，所以优化问题是 min 还是 max 问题不大；常数 d 对优化问题的解没有影响，一般也可以去掉。
  - 对于 $x\le 0$ 的变量可以取反代替，无限制的变量可以取 $x=x_0-x_1$ 代替。
  - 对于 $Ax\ge b$ 的约束可以取反代替，等式 $Ax=b$ 可以拆成 $Ax\ge b$ 和 $Ax\le b$ 代替。
  - 对于 $Ax\le b$ 这个约束，可以通过添加非负变量将其松弛成等式。所以我们总能将线性规划转成右侧的形式。

- 本代码中，将中间的形式（去除 $Px=q$）视为标准输入形式，随后在计算时使用右侧形式。

- **单纯形法（Simplex Method）**（以 $\max$ 模型为例）
  1. 将 A 中的**基变量**消成单位矩阵，并把目标函数**基变量**的系数都消成 0。
  2. 以字典序顺序找一个 $c_j>0$ 的非基变量 j **入基**。
  3. 以字典序顺序考察 $b_i>0$ 的行，确定一行 i 使得 $\frac{b_i}{A_{i,j}}$ 最小（对于 j 最紧的限制）。
  4. 拿第 i 行去消，把每一行的 $A_{?,j}$ 以及目标函数的 c_j 都消成 0。即第 i 行的基变量**出基**。
  5. 重复步骤 $2\sim 4$ 直到检验数全大于 0.
  - 根据Bland's法则，按照字典序入基出基，一定能够停止。

- **单纯形表（Simplex Tableau）**
  - 变量分成基变量和非基变量：设 $c^T=\begin{bmatrix}c_B^T & c_N^T\end{bmatrix}$，$A=\begin{bmatrix}A_B & A_N\end{bmatrix}$，$x=\begin{bmatrix}x_B^T & x_N^T\end{bmatrix}^T$
  - 显然我们有 $A_Bx_B+A_Nx_N=b$ 且 $z=c_B^Tx_B+c_N^Tx_N$
  - 通过简单的线性变换有：

$$\begin{array}{c|cc|c} & c_B^T & c_N^T & 0 \\ \hline x_B & A_B & A_N & b\end{array} \Rightarrow \begin{array}{c|cc|c} & c_B^T & c_N^T & 0 \\ \hline x_B & I & A_B^{-1}A_N & A_B^{-1}b\end{array} \Rightarrow \begin{array}{c|cc|c} & 0 & c_N^T-c_B^TA_B^{-1}A_N & -c_B^TA_B^{-1}b \\ \hline x_B & I & A_B^{-1}A_N & A_B^{-1}b\end{array}$$

- 这其实就是单纯形法的形式化迭代过程

- **基可行解**（以 $\max$ 模型为例）
  - 这里使用一个不同于课上讲的方法。
  - 对于存在 $b_i<0$ 的问题，将松弛变量作为基并不是一个可行解，因此需要找到一个基可行解。
  - 设松弛变量 $x_0\ge 0$，辅助目标 $\max\limits_x -x_0$，并修改限制 $Ax-1x_0=b$（1 是全1向量）。
  - 第一步找到 $b_i<0$ 中绝对值最大的行 i 出基，x_0 入基。可以发现这样之后有 $b\ge 0$，可以对该线性规划跑单纯形法。
  - 若结果 $x_0>0$，有原问题无可行解；否则将 x_0 删除，则已经找到原问题的一可行解。

- 将 x_0 删除时若 x_0 是非基变量，可直接删除对应列；否则需要先将 x_0 出基，取任意 i 入基。由于 $x_0=0$，这一次转轴不会造成影响。
- **对偶单纯形法**（以 $\max$ 模型为例）
  1. 取一组检验数全都 ≤ 0 的基。
  2. 以字典序顺序找一个 $b_i<0$ 的行 i，将基变量 i 出基。
  3. 我们想找某个非基变量从 0 变成大于 0 来增加 b_i。在这一行中，以字典序顺序找一个 $A_{i,j}<0$ 的 j 使得 $|\frac{c_j}{A_{i,j}}|$ 最小（这样 j 消完目标函数后检验数依然全都 ≥ 0），让 j 入基。
  4. 重复步骤 $2\sim 3$ 直到约束右侧的值均 ≥ 0。
  - 根据Bland's法则，按照字典序入基出基，一定能够停止。

1.2 程序设计

- **转化为标准型**

```
//将读入的问题翻译成标准形式
//这里标准形式是max z=... s.t. sigma(aijxj)<=bi xi>=0
//totn为转化后变量数，totm为转化后限制数
int totn=n,totm=0;
//min转max取负
r(j,n)c[j]=-c[j];
//考虑每个变量
r(j,n)if(e[j]==-1){
  //如果<=0，取负
  r(i,m)a[i][j]=-a[i][j];
  c[j]=-c[j];
}else if(e[j]==0){
  //如果无限制，转为x=x1-x2
  //x1占用原编号，x2占用编号为mus[j]
  mus[j]=++totn;
  r(i,m)a[i][mus[j]]=-a[i][j];
  c[mus[j]]=-c[j];
}
//输入目标函数
r(j,totn)LP::a[0][j]=c[j];
//考虑每个限制
r(i,m){
  if(d[i]<=0){// < 或 = 需要增加一个形如aix<=bi的限制
      ++totm;
      r(j,totn)LP::a[totm][j]=a[i][j];
      LP::a[totm][0]=b[i];
  }
  if(d[i]>=0){// = 或 > 需要增加一个形如aix>=bi的限制,即-aix<=-bi
      ++totm;
      r(j,totn)LP::a[totm][j]=-a[i][j];
      LP::a[totm][0]=-b[i];
  }
}
//输入矩阵大小
LP::n=totn;LP::m=totm;
```

  - 这里使用前文中提到的替代方法，将读入的各种限制转化为标准形式。
    - 为了统一性，在数组 A 中我们将目标函数储存在 $A_{0,j}$，将b储存在 $A_{i,0}$。
- **基可行解**

```
bool init(){
  int fx=1;
  rep(i,2,m)if(a[i][0]<a[fx][0])fx=i;
  if(sgn(a[fx][0])>=0)return 1;//没有负行，已经得到初解
  pivot(fx,n+1);//取出abs最大的负行，和辅助变量转轴
  for(;;){//以m+1行为目标函数，做单纯形
      int x=0,y=0;
      rep(j,1,n+1)if(sgn(a[m+1][j])==1&&idn[j]<idn[y])y=j;//找到可以入基的变量，系数为正，取字典序最小
      if(!y)break;//没有，代表已经得到最优解
      rep(i,1,m)if(sgn(a[i][y])==1){//找到对应的出基变量
          if(!x)x=i;
          else{
              db temp=a[i][0]*a[x][y]-a[x][0]*a[i][y];//多个选择时，选增量最小的出基变量
              if(sgn(temp)==-1||(sgn(temp)==0&&idm[i]<idm[x]))//还是相同时取字典序最小
                  x=i;
          }
      }
      if(!x)assert(0);//不可能存在：init返回无界
      pivot(x,y);//转轴
  }
  if(-a[m+1][0]<-1e-6)return 0;//如果辅助变量最后不为0，代表无解
  //将辅助变量转到n+1来舍弃
  rep(i,1,m)if(idm[i]==n+m+2){//如果辅助变量在基变量位
      int fy=1;
      rep(j,2,n+1)if(abs(a[i][j])>abs(a[i][fy]))fy=j;
      pivot(i,fy);//找系数非0位转轴，转到非基变量
      //由于已知辅助变量为0，因此转到非基变量不会导致问题
      break;
  }
  rep(j,1,n)if(idn[j]==n+m+2){//如果辅助变量不在n+1
      swap(idn[j],idn[n+1]);
      rep(i,0,m)swap(a[i][j],a[i][n+1]);//交换到n+1
      break;
  }
  //成功找到初解
  return 1;
}
```

- 运用上述方法找到一个基可行解。
- 这里将松弛变量存储在 $A_{i,n+1}$ 的位置，辅助目标在 $A_{m+1,j}$。
- idn 与 idm 用于记录目前该行对应基变量、该列对应变量的编号。这种记录方法可以使得基变量对应的单位矩阵在 A 中省略。

- **单纯形法**

```
bool spx(){//单纯形，以0行为目标函数
  for(;;){
      int x=0,y=0;
      rep(j,1,n)if(sgn(a[0][j])==1&&idn[j]<idn[y])y=j;//找到可以入基的变量，系数为正，取字典序最小
      if(!y)return 1;//没有，代表已经得到最优解
      rep(i,1,m)if(sgn(a[i][y])==1){//找到对应的出基变量
          if(!x)x=i;
```

```
          else{
              db temp=a[i][0]*a[x][y]-a[x][0]*a[i][y];//多个选择时，选增量最小的出基变量
              if(sgn(temp)==-1||(sgn(temp)==0&&idm[i]<idm[x]))//还是相同时取字典序最小
                  x=i;
          }
      }
      if(!x)return 0;//找不到出基变量：答案无界
      pivot(x,y);//转轴
  }
}
```

- 按照上述操作跑单纯形法。
- 注意到使用 idn 与 idm 找到字典序最小的入基变量和出基变量。

- **对偶单纯形法**

```
bool dual(){//对偶单纯形，以0行为目标函数
  for(;;){
      int x=0,y=0;
      rep(i,1,m)if(sgn(a[i][0])==-1&&idm[i]<idm[x])x=i;//找到必须出基的变量，系数为负，取字典序最小
      if(!x)return 1;//没有，代表已经得到可行解（可行+对偶可行=最优）
      rep(j,1,n)if(sgn(a[x][j])==-1){//找到对应的入基变量
          if(!y)y=j;
          else{
              db temp=a[0][j]*a[x][y]-a[0][y]*a[x][j];//多个选择时，选减量最小的入基变量
              if(sgn(temp)==-1||(sgn(temp)==0&&idn[j]<idn[y]))//还是相同时取字典序最小
                  y=j;
          }
      }
      if(!y)return 0;//找不到入基变量：对偶无界，原问题无解
      pivot(x,y);//转轴
  }
}
```

- 若初始解检验数全都 ≤ 0，程序会选择使用对偶单纯形法。
- 可以发现，对偶单纯形法和普通单纯形法实现上一一对应。

- **转轴（入基+出基）**

```
void pivot(int x,int y){//转轴
  swap(idm[x],idn[y]);//交换行列对应的变量
  //修改行数据
  db t=a[x][y];
  rep(j,0,n+1)a[x][j]/=t;//本行除以对应列的系数，使a[x][y]=1
  a[x][y]=1/t;//现在y列编号为原本的x行编号，因此这个位置值为基变量系数1除以t
  rep(i,0,m+1)if(i!=x){//修改其他行
      t=a[i][y];//取出系数，使相减后对应位a[i][y]=0
      a[i][y]=0;//现在y列编号为原本的x行编号，因此这个位置值为基变量系数0除以t
      rep(j,0,n+1)a[i][j]-=t*a[x][j];//相减
  }
}
```

- 转轴将 x 行的基变量出基，将 y 列的非基变量入基。

- 由于使用了 idn 与 idm ，在转轴时需要特殊处理 x 行和 y 列，其他位置的行为和消元法一致。

注意：程序中的计算流程是先检查是否直接就是对偶可行解，是的话就使用对偶单纯形法求解，不是就使用普通单纯形法求解。

2. Simplex 测试

2.1 测试方法

- 数据生成

  数据生成的思路是首先先随机生成一个向量，以此向量为可行解不断添加约束条件。假设随机生成的向量为$\hat{x}$，随机生成一组约束系数为a，再随机生成一个偏移量o，就得到了一个约束：$a \cdot x \leq a \cdot \hat{x} + o$。（生成无解数据的时候仅需要添加若干组反向的约束，让$\hat{x}$不在约束的空间内）。

```
# randomize a feasible solution
feasible = np.random.rand(n,1) * SCALE - 0.5 * SCALE if 1 else
np.random.randint(0,100,(n,1))

# randomize the cost
cost = np.random.rand(n,1) * SCALE if 1 else np.random.randint(0,100,(n,1))

# randomize the constraint (a,b,d,e)
a = np.random.rand(n,m) * SCALE - 0.5 * SCALE if 1 else
np.random.randint(0,100,(n,m))
d = np.random.randint(0,3,(1,m)) - 1
d = np.random.randint(0,3,(1,m)) - 1
b = np.sum(feasible * a, 0) - d * np.random.rand(1,m) * SCALE if 1 else
np.random.randint(0,100,(1,m))
for i in range(n):
    tmp = random.randint(1,10)
    e[i,0] = 0 if not tmp % 3 else (1 if feasible[i,0]>0 else -1)
```

- 标准答案计算

  我使用了python的科学计算库scipy中的scipy.optimization.linprog作为标准程序进行对比。

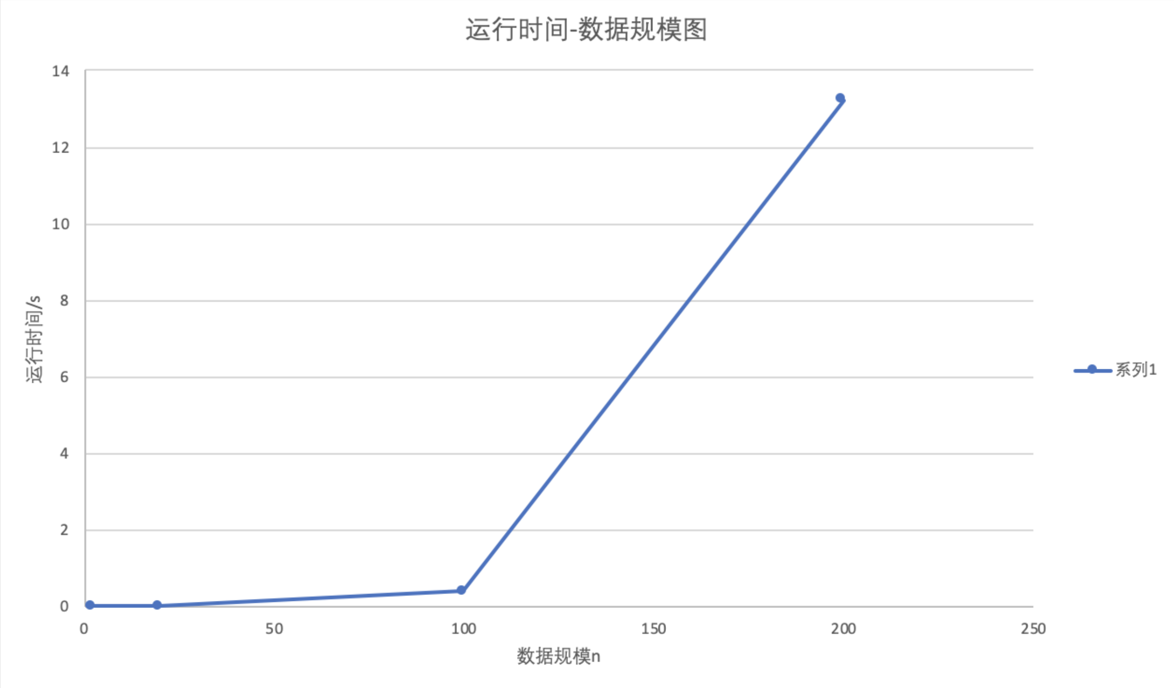
2.2 测试点

进行大数据的测试观察其时间与精度随着数据规模的变化情况。对于每一个数据规模，我都会随机生成100组数据，随后利用100组结果计算其平均值。

- n:m = 2:5，测试一般情况下的运行时间和精度

对于n:m = 2:5的数据，极大多数情况下都是正常解，因此我只统计程序输出正常解的运行时间和精度。（无解和无界的情况发生比例低于1%）

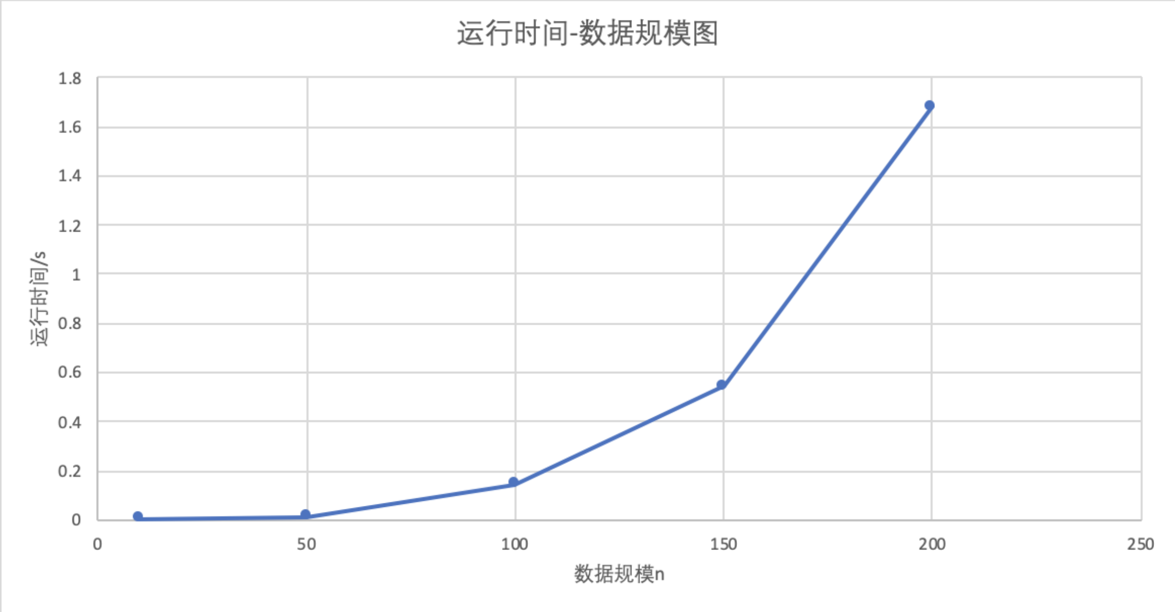
数据规模 (n,m)	运行时间	精度(%)
2,5	0.0058	1.27e-7
20,50	0.0072	1.78e-6
100,250	0.38	7.54e-7
200,500	13.23	3.57e-7

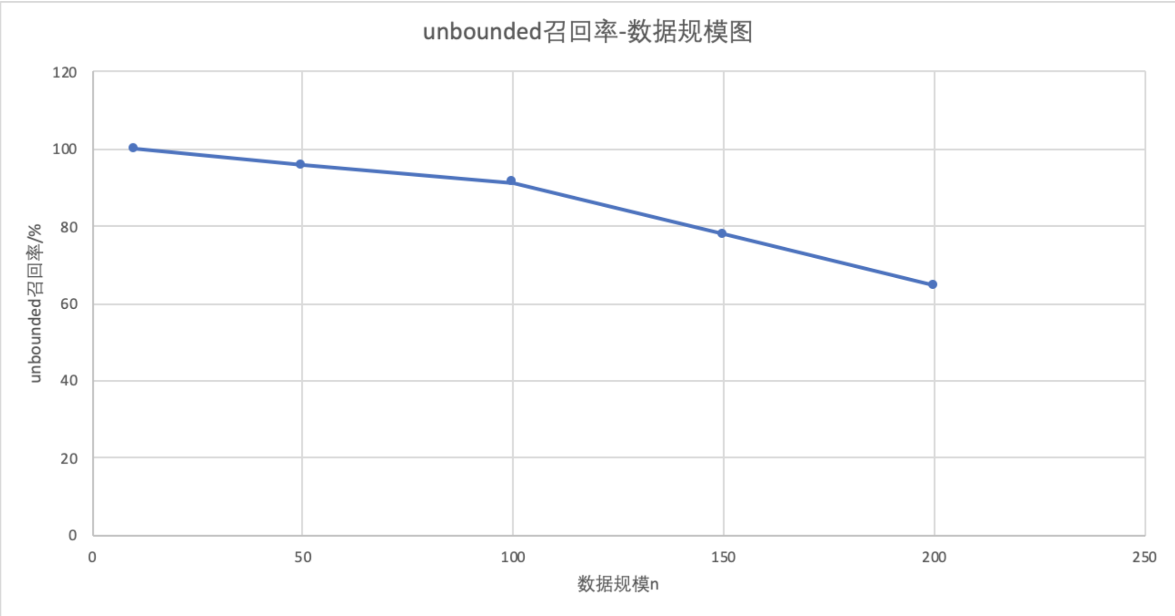


- n:m = 1:1， 测试约束宽松的情况下的运行时间、精度和无界情况召回率

对于n:m=1:1的数据，产生了很多 无界解的情况，因此除了统计计算出正常解的运行时间和精度之外，我还添加了一项无界解的召回率。

数据规模 (n,m)	运行时间	精度(%)	无界情况召回率(%)
10,10	0.0053	1.22e-7	100.0 (44/44)
50,50	0.0132	5.43e-8	95.8 (45/47)
100,100	0.1439	3.42e-8	91.2 (34/37)
150,150	0.5442	1.55e-8	77.8 (35/45)
200,200	1.6829	1.76e-8	64.7 (46/71)





- 测试对于无解测试点的判别和召回率

数据规模 (n,m)	无解召回率（%）
20, 50	100
50, 100	100
100,100	100
150,300	100
200,500	100

2.3 分析

- 时间复杂度分析：单纯形法最坏情况下需要遍历每一组基可行解，对于每一组基可行解，挑出一个出基变量后，都需要遍历修改单纯形表中$m*n$的单元，因此时间复杂度是$O(C_m^n*m*n)$。
- 空间复杂度分析：我们的程序的最大存储单元是单纯形表。因此空间复杂度是$O(m*n)$。

经过测试分析得出，我们实现的单纯形法能够解决较大规模的线性规划，在题目给出的极限情况（n=200, m=500）下仍然取得了很高的精度，和scipy.optimize计算得出的解的误差不超过1e-6%。

同时，经过设置不同的数据规模，我们的程序也能够处理无有限解、无解的情况，成功达到了实验预期的目标。