Lineáris belsőpontos Newton-iteráció

Implementáció Haskellben

Dr. Érdi Gergő
http://gergo.erdi.hu/

Az alábbiakban összeállítunk egy Haskell modult, amely a belsőpontos Newton-iteráció algoritmusával old meg lineáris programozási feladatokat.

```
module InteriorNewton where
```

1. Reprezentáció

A lineáris algebrai műveletek megvalósítását a *HMatrix* csomagból vesszük át:

```
import Foreign.Storable
import Data.Packed.Vector
import Data.Packed.Matrix
import Numeric.LinearAlgebra.Algorithms
import Numeric.LinearAlgebra.Linear
import Numeric.LinearAlgebra.Interface
import Numeric.LinearAlgebra.Instances
```

A megoldási folyamat naplózásához a *Writer* monádot használjuk:

```
import qualified Control.Monad.Writer as W
import Control.Monad.Loops (dropWhileM)
import Control.Monad (liftM)
```

Kétszeres pontosságú, lebegőpontos számábrázolást fogunk használni, ezért minden vektorunk és mátrixunk *Double* elemeket tárol. Az egyszerűség kedvéért bevezetjük az alábbi típusszinonímákat. A *toVec* segédfüggvényre azért van szükségünk, mert a *HMatrix* csomag megkülönbözteti a vektorokat és az egyoszlopos mátrixokat.

```
type Vec = Vector Double
type Mtx = Matrix Double

toVec :: Mtx → Vec
toVec = head ∘ toColumns
```

A megoldandó lineáris programozási feladatot az alábbi formában várjuk el:

$$\begin{aligned} \min\ & c^T x \\ A\,x &= b \\ x &\geq 0 \end{aligned}$$

Egy ilyen feladatot az alábbi algebrai adattípussal reprezentálunk:

```
data Problem = Problem { prob_A :: Mtx,
                         prob_b :: Vec,
                         prob_c :: Vec}
  deriving Show
```

2. ε-egyenlőség

A lebegőpontos számábrázolás korlátai, illetve a numerikus hibák feltorlódása miatt számok, illetve vektorok egyenlőségét nem érdemes szigorúan néznünk, ezért bevezetjük az alábbi $\approx$ relációt:

```
infix 4 ≈
class Approx a where
  (≈) :: a → a → Bool

ε = 10^-6

instance Approx Double where
  x ≈ y = 2 * |x - y| / (1 + |x| + |y|) < ε

instance (Storable a, Approx a) ⇒ Approx (Vector a) where
  x ≈ y = and $ zipWith (≈) (toList x) (toList y)
```

3. Naplózás

```
data OptimizationStep = Infeasible (Vec, Vec, Vec) Double
                      | Feasible (Vec, Vec, Vec)
                        deriving Show

type Logging a = W.Writer [OptimizationStep] a

writeLog :: OptimizationStep → Logging ()
writeLog x = W.tell [x]

iterateUntil :: (a → Bool) → (a → OptimizationStep) → (a → a) → a → Logging a
iterateUntil isFinished logger step initial = liftM head $ dropWhileM notFinished (iterate step initial)
  where notFinished x = do writeLog (logger x)
    return (¬ (isFinished x))
```

4. A feladat perturbálása

Ahhoz, hogy kezdeti belső megoldásunk legyen, a feladatot az alábbi formában perturbáljuk, azzal a céllal, hogy $\nu = 1, \zeta = 1$ esetén választhassuk az $(e, 0, e)$ vektor-hármast kezdeti megoldásnak, és $\nu \to 0$ esetén az eredeti feladathoz tartsunk:

$$\begin{array}{rrcl} & A\,x & = & b - \nu\, r_b \\ A^T\,y + & s & = & c - \nu\, r_c \end{array}$$

ahol

$$\begin{aligned} r_b &:= b - \zeta A\, e \\ r_c &:= c - \zeta e \end{aligned}$$

A megvalósítás során a ζ paraméter értékét fixen 1-en tartjuk.

$perturbation :: Problem \to (Vec, Vec)$
$perturbation\ (Problem\ A\ b\ c) = (r_b, r_c)$
  where $r_b = b - (\zeta \cdot Ae)$
        $r_c = c - \zeta \cdot e$

        $\zeta = 1$
        $(n, m) = (cols\ A, rows\ A)$
        $e = constant\ 1\ n$

5. Centrális út

A centrális utat az alábbi, μ-vel paraméterezett egyenletrendszer (x, y, s) megoldásaival definiáljuk:

$$\begin{array}{rrcl} & A\,x & = & b \\ A^T\,y + & s & = & c \\ & xs & = & \mu\, e \end{array}$$

6. Haladás az eredeti feladatba, illetve a centrális úton

Adott ν és (x, y, s) belső, perturbált megoldás esetén olyan (x', y', s') belső megoldát keresünk, amelyre teljesülnek a feltételek valamilyen ν' és μ paraméterek esetén:

$$\begin{array}{rrcl} & A\,x' & = & b - \nu'\, r_b \\ A^T\,y' + & s' & = & c - \nu'\, r_c \\ & x's' & = & \mu\, e \end{array}$$

Ehhez az alábbi formában állítjuk elő őket:

$$\begin{aligned} x' &:= x + \alpha\, \Delta x \\ y' &:= y + \alpha\, \Delta y \\ s' &:= s + \alpha\, \Delta s \end{aligned}$$

Ahol $\alpha \in (0, 1]$-et úgy választjuk, hogy a megengedettség és a pozitivitás megmaradjon, $(\Delta x, \Delta y, \Delta s)$-et pedig (x, y, s) megoldás-voltát kihasználva, az előjel-kötöttséget elhagyva kapjuk:

$$\begin{array}{rrcl} A\,x & & = & b - \nu\, r_b \\ A\,(x + \Delta x) & & = & b - \nu'\, r_b \\ A^T\,y + & s & = & c - \nu\, r_c \\ A^T\,(y + \Delta y) + & (s + \Delta s) & = & c - \nu'\, r_c \\ & (x + \Delta x)(s + \Delta s) & = & \mu\, e \end{array}$$

Az egyenletrendszert átalakítva, a kvadratikus tagot elhagyva a következő lineáris egyenletrendszer alapján számítható ki $(\Delta x, \Delta y, \Delta s)$:

$$\begin{array}{rrcl} & A\,\Delta x & = & (\nu - \nu')\, r_b \\ A^T\,\Delta y + & \Delta s & = & (\nu - \nu')\, r_c \\ s\,\Delta x + & x\,\Delta s & = & \mu\, e - x\, s \end{array}$$

Ennek az iterációs lépésnek az irányát számítja ki az alábbi függvény:

$$
\begin{aligned}
&dirNewton :: Problem \to Double \to Double \to (Vec, Vec, Vec) \to (Vec, Vec, Vec)\\
&dirNewton\ prob@(Problem\ A\ b\ c)\ \mu\ \Delta\nu\ (x, y, s) = (\Delta x, \Delta y, \Delta s)\\
&\quad \textbf{where } \Delta x = subVector\ 0\ n\ \xi\\
&\qquad\quad \Delta y = subVector\ n\ m\ \xi\\
&\qquad\quad \Delta s = subVector\ (n + m)\ n\ \xi\\
&\qquad\quad \xi = toVec\ (linearSolve\ mtx\ (asColumn\ \lambda))\\
&\qquad\quad mtx = fromBlocks\ [[A,\ \ 0,\ \ 0],\\
&\qquad\qquad\qquad\qquad [0,\ \ A^T, I],\\
&\qquad\qquad\qquad\qquad [mS, 0,\ \ mX]]\\
&\qquad\qquad \textbf{where } mS = diag\ s\\
&\qquad\qquad\qquad\quad mX = diag\ x\\
&\qquad\qquad\qquad\quad I = ident\ n\\
&\qquad\quad \lambda = join\ [\lambda_b, \lambda_c, \lambda_\mu]\\
&\qquad\quad \lambda_b = \Delta\nu \cdot r_b\\
&\qquad\quad \lambda_c = \Delta\nu \cdot r_c\\
&\qquad\quad \lambda_\mu = \mu \cdot e - xs\\
&\qquad\quad (r_b, r_c) = perturbation\ prob\\
&\qquad\quad (n, m) = (cols\ A, rows\ A)\\
&\qquad\quad e = mapVector\ (const\ 1)\ x
\end{aligned}
$$

Az irány ismeretében már csak az α lépéshossz meghatározása van hátra. Ehhez hányadostesztet végzünk az irány alapján, az eredményt eltolva ε-nal a szigorú pozitivitás érdekében.

$$
\begin{aligned}
&\alpha MaxFromRatios :: (Vec, Vec, Vec) \to (Vec, Vec, Vec) \to Double\\
&\alpha MaxFromRatios\ (x, y, s)\ (\Delta x, \Delta y, \Delta s) = min\ 1\ ((minimum\ ratios) - \varepsilon)\\
&\quad \textbf{where } ratios = ratiosX +\!\!+ ratiosS\\
&\qquad\quad ratiosX = filter\ (>0)\ \$\ zipWith\ ratio\ (toList\ x)\ (toList\ \Delta x)\\
&\qquad\quad ratiosS = filter\ (>0)\ \$\ zipWith\ ratio\ (toList\ s)\ (toList\ \Delta s)\\
&\qquad\quad ratio\ z\ dz = \frac{-z}{dz}
\end{aligned}
$$

7. Kezdő belső pont keresése

Az optimalizálás első fázisában ν-t 1-ről 0-ra lenyomva keresünk olyan pontot, amely az eredeti feladatnak belső pontja. Ehhez a perturbált feladat $(e, 0, e)$ belső kezdőpontjából indítjuk a fent definiált Newton-iterációt, úgy, hogy minden lépésben 0-ra próbáljuk csökkenteni ν'-t (azaz $\Delta\nu = \nu$). Az alábbi függvény az iteráció egy lépését számítja ki úgy, hogy a tényleges ν'-t a Newton-lépés megtétele után, egyszerű behelyettesítéssel végzi.

$$
\begin{aligned}
&stepToFeasible :: Problem \to Double \to ((Vec, Vec, Vec), Double) \to ((Vec, Vec, Vec), Double)\\
&stepToFeasible\ prob@(Problem\ A\ b\ c)\ \mu\ ((x, y, s), \nu) = ((x', y', s'), \nu')\\
&\quad \textbf{where } (\Delta x, \Delta y, \Delta s) = dirNewton\ prob\ \mu\ \nu\ (x, y, s)\\
&\qquad\quad \alpha = \alpha MaxFromRatios\ (x, y, s)\ (\Delta x, \Delta y, \Delta s)\\
&\qquad\quad x' = x + \alpha \cdot \Delta x\\
&\qquad\quad y' = y + \alpha \cdot \Delta y\\
&\qquad\quad s' = s + \alpha \cdot \Delta s
\end{aligned}
$$

$$b' = A(x + \alpha \cdot \Delta x)$$
$$(r_b, \_) = \mathit{perturbation}\ \mathit{prob}$$
$$\nu' = \frac{(b - b')_0}{r_{b0}}$$

Minden adott tehát, hogy elkészíthessük azt a kezdőpont-kereső függvényt, amelyik addig iterál, amíg az eredeti feladatnak egy (ε-)megengedett belső pontját kapja. A *Writer* monádot használjuk a számítási sorozat naplózásához.

```
initialInfeasible :: Problem → (Vec, Vec, Vec)
initialInfeasible (Problem A b c) = (x, y, s)
  where x = constant 1 n
        y = constant 0 m
        s = constant 1 n
        (n, m) = (cols A, rows A)

solveToFeasible :: Problem → Logging (Vec, Vec, Vec)
solveToFeasible prob@(Problem A b c) = liftM fst $ iterateUntil isFeasible toLog (stepToFeasible prob μ) ((x, y,
  where (x, y, s) = initialInfeasible prob
        μ = 1
        ν = 1
        isFeasible ((x', y', s'), ν') = Ax' ≈ b
        toLog ((x, y, s), ν) = Infeasible (x, y, s) ν
```

8. Optimalizálás

Egy (x, y, s) belső pont birtokában végre elkezdhetjük a tényleges belsőpontos optimalizálást, vagyis a centrális úton haladást μ csökkentésével. Lépésenként kiszámítjuk a javító irányt és lépéshosszt, és ha $\mathcal{C}(\mu)$ megfelelő közelébe kerültünk, akkor μ-t is csökkentjük. A centrális úttól való távolságot az alábbi függvénnyel értelmezzük:

$$\delta(\cdot,\cdot,\cdot) :: \mathit{Vec} \to \mathit{Vec} \to \mathit{Double} \to \mathit{Double}$$
$$\delta(x, s, \mu) = 0.5 * \|u - \mathit{recip}\ u\|_2$$
$$\textbf{where}\ u = \mathit{zipVector}\ f\ x\ s$$
$$f\ \xi\ s_i = \sqrt{\frac{\xi * s_i}{\mu}}$$

A lépéshosszt pedig úgy választjuk meg, hogy amellett, hogy az $x, s > 0$ feltétel teljesüljön, ne „ugorjunk ki” az optimális megoldáson átlépve a megengedett pontokból, vagyis a dualitási rés nemnegatív maradjon.

$$\mathit{target}\ (\mathit{Problem}\ A\ b\ c)\ (x, y, s) = x^T c$$
$$\mathit{shadow}\ (\mathit{Problem}\ A\ b\ c)\ (x, y, s) = y^T b$$
$$\mathit{gap}\ \mathit{prob}\ (x, y, s) = \mathit{target}\ \mathit{prob}\ (x, y, s) - \mathit{shadow}\ \mathit{prob}\ (x, y, s)$$

```
stepNewton :: Problem → ((Vec, Vec, Vec), Double) → ((Vec, Vec, Vec), Double)
stepNewton prob ((x, y, s), μ) = ((x', y', s'), μ')
  where (Δx, Δy, Δs) = dirNewton prob μ 0 (x, y, s)

        x' = x + α · Δx
        y' = y + α · Δy
        s' = s + α · Δs
```

$$
\begin{aligned}
&\alpha Max = \alpha MaxFromRatios\ (x, y, s)\ (\Delta x, \Delta y, \Delta s)\\
&\alpha = until\ isPosGap\ (*0.95)\ \alpha Max\\
&\quad \textbf{where}\ isPosGap\ \alpha = gap\ prob\ (x', y', s') \geqslant 0\\
&\qquad \textbf{where}\ x' = x + \alpha \cdot \Delta x\\
&\qquad\qquad\ \ y' = y + \alpha \cdot \Delta y\\
&\qquad\qquad\ \ s' = s + \alpha \cdot \Delta s
\end{aligned}
$$

$$
\mu' = \textbf{if}\ \delta(x', s', \mu) < \tau\ \textbf{then}\ (1 - \theta) * \mu\ \textbf{else}\ \mu
$$

$$
\begin{aligned}
&\tau = 0.1\\
&\theta = 0.8
\end{aligned}
$$

Ezekután nincs más dolgunk, mint addig iterálni, amig optimális, azaz ε-nál kiesebb dualitási résű megoldást nem találunk:

$$
\begin{aligned}
&solveNewton :: Problem \to (Vec, Vec, Vec) \to Logging\ (Vec, Vec, Vec)\\
&solveNewton\ prob\ (x, y, s) = liftM\ fst\ \$\ iterateUntil\ optimal\ toLog\ (stepNewton\ prob)\ ((x, y, s), \mu)\\
&\quad \textbf{where}\ \mu = \frac{x^T s}{n}\\
&\qquad\qquad\ \ n = fromIntegral\ (dim\ x)\\
&\qquad\qquad\ \ optimal\ ((x', y', s'), \mu') = gap\ prob\ (x', y', s') \approx 0\\
&\qquad\qquad\ \ toLog\ ((x, y, s), \mu) = Feasible\ (x, y, s)
\end{aligned}
$$

$$
\begin{aligned}
&optimize :: Problem \to Logging\ (Vec, Vec, Vec)\\
&optimize\ prob = \textbf{do}\ (x, y, s) \leftarrow solveToFeasible\ prob\\
&\quad solveNewton\ prob\ (x, y, s)
\end{aligned}
$$