RjpWiki アーカイブス

【最適化関数一覧 (06.06.27) 】

1　多変数目的関数の最適化

1.1　汎用的最適化関数

Nelder-Mead 法、準ニュートン法と共役勾配法アルゴリズムに基づく汎用的最適化プログラム。オプションとして矩形型制約下での最適化とシミュレーティッドアニーリング法も行う。

```
optim(par, fn, gr = NULL,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"),
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE, ...)
```

1.2　線形制約下で多変数関数の最大、最小値を求める

適応的バリヤアルゴリズム (要するに境界に近付くに従い増加するペナルティ関数を適応的に加える) を用いて線形制約下での関数最小化を行う。

```
constrOptim(theta, f, grad, ui, ci, mu = 1e-04, control = list(),
            method = if(is.null(grad)) "Nelder-Mead" else "BFGS",
            outer.iterations = 100, outer.eps = 1e-05, ...)
```

1.3　一変数関数の最大、最小値を求める

関数 f の第一引数に関する最大・最小値を区間 lower から upper の間で求める。optimise() は optimize() の別名である。

```
optimize(f = , interval = , lower = min(interval),
        upper = max(interval), maximum = FALSE,
        tol = .Machine$double.eps^0.25, ...)
optimise(f = , interval = , lower = min(interval),
        upper = max(interval), maximum = FALSE,
        tol = .Machine$double.eps^0.25, ...)
```

2　一変数目的関数の最適化

2.1　一変数関数の零点を見付ける

関数 uniroot() は関数 f の第一引数に付いて区間 [lower, upper] 内で根 (零点) を探す。

```
uniroot(f, interval, lower = min(interval), upper = max(interval),
        tol = .Machine$double.eps^0.25, maxiter = 1000, ...)
```

2.2　実・複素多項式の零点を見付ける

実・複素多項式の零点を見付ける。

```
polyroot(z)
```

3　関連関数

3.1　数式微分関数

簡単な表現式の数式微分を計算する。

```
D(expr, name)
deriv(expr, namevec, function.arg, tag = ".expr", hessian = FALSE)
deriv3(expr, namevec, function.arg, tag = ".expr", hessian = TRUE)
```