Quantitative Finance

Benders Decomposition

Patrick Hénaff

Version: 09 Mar 2025

Contents

```r
library(foreach)
library(doParallel)
library(kableExtra)
library(linprog)
library(gtools)
library(pracma)
library(kernlab)
library(optiSolve)
library(piqp)
library(latex2exp)
library(formatdown)
library(lpSolveAPI)
library(xts)
library(PortfolioOptim)
```

The algorithm due to Benders provides a practical strategy for solving two-stage stochastic optimization problems. The principle of the method is to observe that in an optimization problem, there are sometimes "complicating variables", in the sense that, if these variables were known, the remaining problem would be simple to solve. In a mixed-integer programming problem, the "complicating variables" are the integer variables. In a two-stage problem with recourse, they are the variables pertaining to the first stage decision.

We first present the method with a simple model, then explore how it may be used to solve stochastic optimization problems with recourse.

1 The Benders decomposition algorithm

Consider the problem

$$\begin{aligned} V^* = \min\ & c^T x + f^T y \\ \text{s.t.} & \\ & Ax = b \\ & Bx + Dy = d \\ & x \geq 0; y \geq 0 \end{aligned}$$

In the context of stochastic programming with recourse, x would be the first stage decision variables. The problem is reformulated as follows:

$$\begin{aligned} V^* = \min\ & c^T x + z(x) \\ \text{s.t.} & \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

with

$$\begin{aligned} P2 : z(x) = \min\ & f^T y \\ \text{s.t.} & \\ & Dy = d - Bx \\ & y \geq 0 \end{aligned}$$

Let $D2$ be the dual of $P2$:

$$\begin{aligned} D2 : z(x) = \max_p\ & p^T (d - Bx) \\ \text{s.t.} & \\ & D^T p \leq f \end{aligned}$$

The domain $\{p | D^T p \leq f\}$ is defined by a set of extreme points $p^i, i = 1, \ldots, I$ and of extreme rays $r^j, j = 1, \ldots, J$.

If $D2$ is unbounded, there exists an extreme ray r^j such that $(r^j)^T (d - Bx) > 0$. If the solution of $D2$ is finite, that solution identifies an extreme point p^i of the domain such that the optimal value $z(x) = (p^i)^T (d - Bx)$ is the optimum over all the extreme points.

We can in theory, restate $D2$ in terms of the extreme points and extreme rays of the feasible domain; incorporating this formulation in the original problem, we obtain the full master problem:

$$\begin{aligned} FMP : V = \min_{x,z}\ & c^T x + z \\ \text{s.t.} & \\ & Ax = b \\ & x \geq 0 \\ & (p^i)^T(d - Bx) \leq z, i = 1, \dots, I \\ & (r^j)^T(d - Bx) \leq 0, j = 1, \dots, J \end{aligned}$$

The set of extreme points and extreme rays may be very large, and we expect that only a few constraints will be active at the optimum. The idea behind Benders' partitioning is therefore to progressively build the set of constraints, until we reach an optimum of FMP which is feasible for the original problem.

The above formulation, with only a subset of extreme rays and extreme points constraints is called the restricted master problem (RMP).

Assume that we have just solved the RMP with l constraints:

$$\begin{aligned} RMP^k : V^k = \min_{x,z}\ & c^T x + z \\ \text{s.t.} & \\ & Ax = b \\ & x \geq 0 \\ & (p^i)^T(d - Bx) \leq z, i = 1, \dots, k - l \\ & (r^j)^T(d - Bx) \leq 0, j = 1, \dots, l \end{aligned}$$

and obtained the solution $(\bar{x}, \bar{z})$. Since this solution may not be feasible for the original problem, we have $V^k <= V^*$. Does this solution violate any constraint not included thus far? To answer that question, we solve

$$\begin{aligned} Q(\bar{x}) : \max_p\ & p^T(d - B\bar{x}) \\ \text{s.t.} & \\ & D^T p < f \end{aligned}$$

or the corresponding primal, which involves the original variables:

$$\begin{aligned} Q(\bar{x}) : \min_y\ & f^T y \\ \text{s.t.} & \\ & Dy = d - B\bar{x} \\ & y \geq 0 \end{aligned}$$

If $Q(\bar{x})$ is unbounded, the linear program reveals a new extreme ray r^j such that $(r^j)^T(d - B\bar{x}) > 0$. Therefore, r^j violates the constraint $(r^j)^T(d - B\bar{x}) \leq 0$, and that constraint must be added to the RMP.

If $Q(\bar{x})$ is finite, the linear program identifies an extreme point p^i and a feasible solution $(\bar{x}, \bar{y})$ to the original problem. The objective value $c^T\bar{x} + f^T\bar{y}$ is therefore an upper bound to the solution of the FMP. If $(p^i)^T(d - B\bar{x}) > \bar{z}$, the extreme point violates the constraints of the RMP, and that constraint must added to the RMP. Otherwise, we conclude that the RMP provides a solution $(\bar{x}, \bar{z})$ that is feasible and optimal for the FMP.

2 Example 1

Let's consider a small problem that will both illustrate the generation of both types of cuts.

$$\begin{aligned} \min\ & c^T x + f^T y \\ \text{s.t.}\ & \\ & Ax = b \\ & Bx + Dy = d \\ & x \geq 0; y \geq 0 \end{aligned}$$

with

$$c = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \\ 1 \end{bmatrix} \qquad f = \begin{bmatrix} 4 \\ 5 \\ 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix} \qquad b = \begin{bmatrix} 6 \\ 15 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 \\ 3 & 1 & 2 & 3 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 6 & 5 \end{bmatrix} \qquad d = \begin{bmatrix} 18 \\ 31 \end{bmatrix}$$

We shall see in a second example that Benders's algorithm is particularly adapted when the D matrix has a special block-diagonal structure, but we postpone that discussion in order to focus of the principles of the algorithm.

The setup and solution of the restricted master problem are implemented as follows. The initial version of the RMP only involves the constraint $Ax = b$.

```r
make.rmp <- function() {
  obj <- c(cost, 1)
  rmp <- make.lp(ncol=length(obj))
  set.objfn(rmp, obj)
  for(i in seq(nrow(A))) {
    add.constraint(rmp, c(A[i,], 0), "=", b[i])
  }
  rmp
}
```

At each iteration, an optimality or feasibility constraint is added to the RMP:

```r
solve.rmp <- function(rmp, p=NULL, r=NULL) {

  # optimality constraint
  if(!is.null(p)) {
    new.const <- c(-p %*% B, -1)
    new.rhs <- -p %*% d
    add.constraint(rmp, new.const, "<=", new.rhs)
  }

  # feasibility constraint
  if(!is.null(r)) {
    new.const <- c(-r %*% B, 0)
    new.rhs <- -r %*% d
    add.constraint(rmp, new.const, "<=", new.rhs)
  }

  status <- solve(rmp)
  if(status != 0) {
    print(rmp)
    stop()
  }
  rmp
}
```

We solve the dual of the sub-problem, rather than the primal, because the constraint $D^T p \leq f$ does not change from one iteration to the next. This means that the previous solution to the subproblem is a feasible starting point for the next iteration.

```r
make.subproblem <- function() {
  Amat <- t(D)
  dim.p <- ncol(Amat)
  obj <- rep(0, dim.p)
  sub.p <- make.lp(ncol=dim.p)
  lp.control(sub.p, sense="max")
  for(i in seq(nrow(Amat))) {
    add.constraint(sub.p, Amat[i,], "<=", f[i])
  }
  set.bounds(sub.p, lower=rep(-BigM, dim.p), upper=rep(BigM, dim.p))
  sub.p
}

solve.subproblem <- function(sub.p, x.bar) {
  obj <- d - B %*% x.bar
  set.objfn(sub.p, obj)
  status <- solve(sub.p)

  if(status != 0) {
    print(sub)
    stop()
  }
  sub.p
}
```

Let's follow the algorithm one step at a time. We first solve the initial RMP

$$
\begin{aligned}
&\min\ c^T x \\
&\quad \text{s.t.} \\
&\qquad Ax = b \\
&\qquad x \geq 0
\end{aligned}
$$

```
cost <- c(1,2,3,2,1)
f <- c(4,5,4)
A <- matrix(seq(10), nrow=2, byrow=T)
b <- c(6,15)
B <- matrix(c(1,2,3,1,2,3,1,2,3,1), nrow=2, byrow=T)
D <- matrix(c(4,5,6,7,6,5), nrow=2, byrow=T)
d <- c(18, 31)

LB = -Inf
UB = Inf
BigM <- 1.e+4

rmp <- make.rmp()
rmp <- solve.rmp(rmp)
sol <- get.variables(rmp)
x.bar <- sol[1:length(cost)]
z.bar <- sol[length(sol)]
opt <- get.objective(rmp)
```

The solution is $\bar{x} = (\ 0.7500, 0, 0, 0, 1.050\ )$.

Next, we solve the dual sub-problem with the current value of $\bar{x}$.

$$
\begin{aligned}
&Q(\bar{x}) : \max_p\ \ p^T(d - B\bar{x}) \\
&\qquad\quad \text{s.t.} \\
&\qquad\qquad D^T p < f
\end{aligned}
$$

```r
sub <- make.subproblem()
sub <- solve.subproblem(sub, x.bar)
p <- get.variables(sub)
```

The solution is $p = (\ -10000, 5715\ )$. The dual is unbounded (le primal is infeasible). The solution is an extreme ray r^1 that provides a feasibility cut for the restricted master program.

At iteration 2, we solve:

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.} & \\ & Ax = b \\ & (r^1)^T(d - Bx) \leq 0 \\ & x \geq 0 \end{aligned}$$

The solution is $\bar{x} = (\ 0.6442, 0, 0, 0.4232, 0.7326\ )$. The objective value provides a lower bound to the solution: $LB = 2.223$. We go back to the sub-problem with the new value of $\bar{x}$.

This time, the solution is an extreme point $p^2 = (\ 0.3636, 0.3636\ )$. The corresponding solution to the primal program is $\bar{y} = (\ 3.866, 0, 0.0004921\ )$. The solution $(\bar{x}, \bar{y})$ is feasible for the original problem, and provides an upper bound to the solution: $UB = c^T\bar{x} + f^T\bar{y} = 17.69$.

We find that $Q(\bar{x}) = 15.47$ while $\bar{z} = 0$. Therefore, $Q(\bar{x}) > \bar{z}$ and the RMP solution $(\bar{x}, \bar{z})$ violates the constraint

$$(p^2)^T(d - Bx) \leq z$$

We have identified a optimality cut to add to the restricted master program, which becomes

$$\begin{aligned} \min\ & c^T x + z \\ \text{s.t.} & \\ & Ax = b \\ & (r^1)^T(d - Bx) \leq 0 \\ & (p^2)^T(d - Bx) \leq z \\ & x \geq 0 \end{aligned}$$

The solution is $\bar{x} = (\ 0.6442, 0, 0, 0.4232, 0.7326\ )$. The objective value provides an undated lower bound to the solution: $LB = 17.69$. The upper and lower bounds are now equal, the optimum to the original problem has been found.

3 Example 2

An alternative to mean-variance portfolio optimization is to optimize a portfolio while only penalizing the downside risk. Consider for example the Lower Semi-Absolute Deviation (LSAD) risk measure, which only penalizes the shortfall below a mean or target return.

We have a portfolio of N assets and consider M scenarios for the investment horizon. Let r_{ij} be the return of asset i under scenario j. Let τ^* be the target portfolio return, and we would like to minimize the expected shortfall below this target return. The scenarios have equal probabiities.

The corresponding program can be stated as follows:

$$
\begin{aligned}
\min \quad & \sum_j z_j \\
\text{s.t.} & \\
& \sum_i x_i = 1 \\
& \sum_i r_{ij} x_i + z_j \geq \tau^* \quad j = 1, \ldots M \\
& x \geq 0
\end{aligned}
$$

We recognize the structure of the two-stage problem. Benders's algorithm is particularly adapted when the number of scenarios is large. Let's compare this formulation to the classical mean-variance model.

Daily returns for 11 assets

```
library(Rsymphony)
library(Rglpk)
```

```
## Loading required package: slam

## Using the GLPK callable library version 5.0
```

```
daily.ret.file <- file.path(get.data.folder(), "daily.ret.rda")
load(daily.ret.file)
```

We use monthly of data. The optimal weights are shown in Table 1.

```
monthly.ret <- period.apply(daily.ret, endpoints(daily.ret, "months"), mean)*30
N <- ncol(monthly.ret)
nb.obs <- nrow(monthly.ret)

dat <- cbind(as.matrix(monthly.ret), matrix(1/nb.obs, nb.obs, 1))
tau <- .03
eps <- 1.e-5
a0 <- rep(1, N)
A.const <- rbind(a0, -a0)
b.const <- c(1+eps, -1+eps)
LB <- rep(0, N)
```

```r
UB <- rep(1, N)

res <- BDportfolio_optim(dat, tau, "CVAR", .95, Aconst=A.const,
                          bconst=b.const, LB=LB, UB=UB, maxiter=200, tol=eps)
```

Table 1: LSAD allocation

	weights (%)
AAPL	18.4
AMZN	31.7
MSFT	0.0
F	0.0
SPY	0.0
QQQ	0.0
XOM	0.0
MMM	0.0
HD	26.7
PG	0.5
KO	22.7