Stochastic Knapsack Problem with random weights

Stefanie Kosuch and Abdel Lisser

Université Paris XI - Sud
LRI - GraphComb

May 25, 2009

The Deterministic Knapsack Problem

The Deterministic Knapsack Problem

- $c > 0$: Knapsack weight capacity

The Deterministic Knapsack Problem

- $c > 0$: Knapsack weight capacity
- n items

The Deterministic Knapsack Problem

- $c > 0$: Knapsack weight capacity
- n items
- $r_i > 0$: reward per weight unit of item i

The Deterministic Knapsack Problem

- $c > 0$: Knapsack weight capacity
- n items
- $r_i > 0$: reward per weight unit of item i
- w_i: weight of item i

The Stochastic Knapsack Problem with Random Weights

- $c > 0$: Knapsack weight capacity
- n items
- $r_i > 0$: reward per weight unit of item i
- χ_i: independently normally distributed weight of item i

The Stochastic Knapsack Problem with Random Weights

- $c > 0$: Knapsack weight capacity
- n items
- $r_i > 0$: reward per weight unit of item i
- χ_i: independently normally distributed weight of item i
- μ_i,σ_i: mean and standard deviation of χ_i

The Stochastic Knapsack Problem with Random Weights

- $c > 0$: Knapsack weight capacity
- n items
- $r_i > 0$: reward per weight unit of item i
- χ_i: independently normally distributed weight of item i
- μ_i,σ_i: mean and standard deviation of χ_i
- $x \in \{0,1\}^n$: decision vector

Outline

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x \in \{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x\in\{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

- $[x]^+ := \max(0, x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x)$ $(x \in \mathbb{R})$

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x\in\{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

- $[x]^+ := \max(0, x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x) \; (x \in \mathbb{R})$
- $\mathbb{1}_{\mathbb{R}^+}$: indicator function of the positive real interval

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x \in \{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

- $[x]^+ := \max(0, x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x)$ $(x \in \mathbb{R})$
- $\mathbb{1}_{\mathbb{R}^+}$: indicator function of the positive real interval
- $d > 0$

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x \in \{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

- $[x]^+ := \max(0, x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x)$ $(x \in \mathbb{R})$
- $\mathbb{1}_{\mathbb{R}^+}$: indicator function of the positive real interval
- $d > 0$

Stochastic Knapsack Problem with simple recourse (SRKP)

$$\max_{x\in\{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i\chi_i x_i] - d\cdot\mathbb{E}[[\sum_{i=1}^{n}\chi_i x_i - c]^+]$$

- $[x]^+ := \max(0,x) = x\cdot\mathbb{1}_{\mathbb{R}^+}(x)$ $(x\in\mathbb{R})$
- $\mathbb{1}_{\mathbb{R}^+}$: indicator function of the positive real interval
- $d>0$: penalty factor per weight unit

Outline

Analytic description

Analytic description

- f, F: density / cumulative distribution function of the standard normal distribution

Analytic description

- f, F: density / cumulative distribution function of the standard normal distribution
- $\hat{\mu} := \sum_{i=1}^{n} \mu_i x_i$, $\hat{\sigma} := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$

Analytic description

- f, F: density / cumulative distribution function of the standard normal distribution
- $\hat{\mu} := \sum_{i=1}^{n} \mu_i x_i$, $\hat{\sigma} := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$

Analytic description

- f, F: density / cumulative distribution function of the standard normal distribution
- $\hat{\mu} := \sum_{i=1}^{n} \mu_i x_i$, $\hat{\sigma} := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$

$$J_{det}(x) =$$

$$\sum_j r_j \mu_j x_j - d \cdot \left[\hat{\sigma} \cdot f\left(\frac{c - \hat{\mu}}{\hat{\sigma}}\right) - (c - \hat{\mu}) \cdot \left[1 - F\left(\frac{c - \hat{\mu}}{\hat{\sigma}}\right)\right]\right]$$

Problem Solving Method - Scheme

Problem Solving Method - Scheme

- Branch-and-Bound algorithm

Problem Solving Method - Scheme

- Branch-and-Bound algorithm
- Solve linear relaxation to compute upper bounds

Problem Solving Method - Scheme

- Branch-and-Bound algorithm
- Solve linear relaxation to compute upper bounds
- Use stochastic gradient algorithm to solve linear relaxation

Problem Solving Method - Scheme

- Branch-and-Bound algorithm
- Solve linear relaxation to compute upper bounds
- Use stochastic gradient algorithm to solve linear relaxation
- Apply "Approximation by convolution" - method to approximate the gradient of the objective function

Outline

Definition

Relaxed Stochastic Knapsack Problem (with simple recourse):

$$\max_{x\in[0,1]^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

Definition

Relaxed Stochastic Knapsack Problem (with simple recourse):

$$\max_{x \in [0,1]^n} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+]$$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in X_{ad}

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$
Update x^k as follows:

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$
Update x^k as follows:

$$x^{k+1} = x^k + \epsilon^k r^k$$

where $r^k = \nabla_x j(x,\chi)$ and $(\epsilon^k)_{k \in \mathbb{N}}$ is a σ-sequence

where $j(x,\chi) = \sum_j r_j \chi_j x_j - d \cdot [\sum_{i=1}^n \chi_i x_i - c]^+$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$
Update x^k as follows:

$$x^{k+1} = x^k + \epsilon^k r^k$$

where $r^k = \nabla_x j(x,\chi)$ and $(\epsilon^k)_{k\in\mathbb{N}}$ is a σ-sequence
For all $i = 1, ..., n$:
If $x_i^{k+1} > 1$ set $x_i^{k+1} = 1$ / If $x_i^{k+1} < 0$ set $x_i^{k+1} = 0$

where $j(x,\chi) = \sum_j r_j \chi_j x_j - d \cdot [\sum_{i=1}^n \chi_i x_i - c]^+$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$
Update x^k as follows:

$$x^{k+1} = x^k + \epsilon^k r^k$$

where $r^k = \nabla_x j(x, \chi)$ and $(\epsilon^k)_{k \in \mathbb{N}}$ is a σ-sequence
For all $i = 1, ..., n$:
If $x_i^{k+1} > 1$ set $x_i^{k+1} = 1$ / If $x_i^{k+1} < 0$ set $x_i^{k+1} = 0$

where $j(x, \chi) = \sum_j r_j \chi_j x_j - d \cdot [\sum_{i=1}^n \chi_i x_i - c]^+$

The Stochastic Gradient Algorithm

Stochastic Gradient Algorithm

Choose x^0 in $X_{ad} = [0,1]^n$
At step k, draw $\chi = (\chi_1, ..., \chi_n)$
Update x^k as follows:

$$x^{k+1} = x^k + \epsilon^k r^k$$

where $r^k = \nabla_x j(x, \chi)$ and $(\epsilon^k)_{k \in \mathbb{N}}$ is a σ-sequence
For all $i = 1, ..., n$:
If $x_i^{k+1} > 1$ set $x_i^{k+1} = 1$ / If $x_i^{k+1} < 0$ set $x_i^{k+1} = 0$

where $j(x, \chi) = \sum_j r_j \chi_j x_j - d \cdot \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^n \chi_i x_i - c)(\sum_{i=1}^n \chi_i x_i - c)$

Outline

Definition

The *convolution* of two real-valued functions is defined as follows:

Definition

The *convolution* of two real-valued functions is defined as follows:

$$(f * h)(x) := \int_{-\infty}^{\infty} f(y)h(x - y)\, \mathrm{d}y$$

Approximation by convolution

Approximation by convolution

Let h be a pair, continuous and non-negative function such that:

- $\int_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$
- $\arg\max h(x) = 0$

Approximation by convolution

Let h be a pair, continuous and non-negative function such that:

- $\int_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$
- $\arg\max h(x) = 0$

$$h_r(x) := \frac{1}{r} h\left(\frac{x}{r}\right)$$

Approximation by convolution

Let h be a pair, continuous and non-negative function such that:

- $\int\limits_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$
- $\arg\max h(x) = 0$

$$h_r(x) := \frac{1}{r} h\left(\frac{x}{r}\right)$$

Then, for small values of $r > 0$, we get the following approximation of a locally integrable real valued function f:

$$f_r(x) := (f * h_r)(x) = \frac{1}{r} \int\limits_{-\infty}^{\infty} f(y) h\left(\frac{y-x}{r}\right)\,\mathrm{d}y$$

Approximation by convolution

Let h be a pair, continuous and non-negative function such that:

- $\int_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$
- $\arg\max h(x) = 0$

$$h_r(x) := \frac{1}{r} h\left(\frac{x}{r}\right)$$

Then, for small values of $r > 0$, we get the following approximation of a locally integrable real valued function f:

$$f_r(x) := (f * h_r)(x) = \frac{1}{r} \int_{-\infty}^{\infty} f(y) h\left(\frac{y-x}{r}\right)\,\mathrm{d}y$$

$$h := \frac{3}{4}(1 - x^2)\mathbb{1}_1(x)$$

($\mathbb{1}_1$: indicator function of the interval $]-1, 1[$)

$$h := \frac{3}{4}(1 - x^2)\mathbb{1}_1(x)$$

Approximated gradient

$$\nabla_x j(x, \chi) = \begin{pmatrix} r_1\chi_1 \\ \cdot \\ \cdot \\ \cdot \\ r_n\chi_n \end{pmatrix} +$$

$$d \cdot \left(\frac{3}{4r}\left(1 - \left(\frac{g(x,\chi)}{r}\right)^2\right)\mathbb{1}_1\left(\frac{g(x,\chi)}{r}\right)\chi \cdot g(x,\chi) - \mathbb{1}_{\mathbb{R}^+}(g(x,\chi)) \cdot \chi \right)$$

$$h := \frac{3}{4}(1 - x^2)\mathbb{1}_1(x)$$

Approximated gradient

$$\nabla_x j(x,\chi) = \begin{pmatrix} r_1\chi_1 \\ \cdot \\ \cdot \\ \cdot \\ r_n\chi_n \end{pmatrix} +$$

$$d \cdot \left(\frac{3}{4r} \left(1 - \left(\frac{g(x,\chi)}{r} \right)^2 \right) \mathbb{1}_1 \left(\frac{g(x,\chi)}{r} \right) \chi \cdot g(x,\chi) - \mathbb{1}_{\mathbb{R}^+}(g(x,\chi)) \cdot \chi \right)$$

where $g(x,\chi) := \sum_{i=1}^n \chi_i x_i$

Outline

Dominance relationships

Item i dominates item j if one of the following holds:

Dominance relationships

Item i dominates item j if one of the following holds:

- $\mu_i = \mu_j$, $r_i \geq r_j$, $\sigma_i \leq \sigma_j$

Dominance relationships

Item i dominates item j if one of the following holds:

- $\mu_i = \mu_j$, $r_i \geq r_j$, $\sigma_i \leq \sigma_j$
- $\mu_i \leq \mu_j$, $\sigma_i \leq \sigma_j$, $r_i \cdot \mu_i \geq r_j \cdot \mu_j$

Dominance relationships

Item i dominates item j if one of the following holds:

- $\mu_i = \mu_j$, $r_i \geq r_j$, $\sigma_i \leq \sigma_j$
- $\mu_i \leq \mu_j$, $\sigma_i \leq \sigma_j$, $r_i \cdot \mu_i \geq r_j \cdot \mu_j$

Dominance relationships

Item i dominates item j if one of the following holds:

- $\mu_i = \mu_j,\ r_i \geq r_j,\ \sigma_i \leq \sigma_j$
- $\mu_i \leq \mu_j,\ \sigma_i \leq \sigma_j,\ r_i \cdot \mu_i \geq r_j \cdot \mu_j$

Motivation

$$\frac{\partial J_{det}}{\partial \hat{\sigma}}(x) =$$

$$\hat{\sigma} := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$$

Dominance relationships

Item i dominates item j if one of the following holds:

- $\mu_i = \mu_j,\ r_i \geq r_j,\ \sigma_i \leq \sigma_j$
- $\mu_i \leq \mu_j,\ \sigma_i \leq \sigma_j,\ r_i \cdot \mu_i \geq r_j \cdot \mu_j$

Motivation

$$\frac{\partial J_{det}}{\partial \hat{\sigma}}(x) = -d \cdot f\left(\frac{c-\mu}{\sigma}\right) < 0$$

$$\hat{\sigma} := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$$

The Branch-and-Bound Algorithm

Ranking

The Branch-and-Bound Algorithm

Ranking

- Number of objects dominated
- value of $\frac{r_i^2}{\sigma_i}$

The Branch-and-Bound Algorithm

Ranking

Plunging

The Branch-and-Bound Algorithm

Ranking

Plunging

- Beginning at the root, add current item iff objective function increases
- INF $\leftarrow$ maximum value of the objective function found
- Add branch found to list of waiting branches;
  set assigned value SUP to ∞

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

- **If** no branch left on list of branches $\longrightarrow$ step G.
- **Else** take branch having maximum objective function value. $\longrightarrow$ step D.

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

- **If** $SUP > INF \longrightarrow$ step E.
- **Else** delete branch. $\longrightarrow$ step C.

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

Calculating upper bounds

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

Calculating upper bounds

- **If** no accepted item left in selected branch that does not already have a plunged or rejected subtree, delete branch from list. $\longrightarrow$ step C.
- **Else** choose first accepted item that does not already have a plunged or rejected subtree. Calculate upper bound SUP for the subtree defined by rejecting this item. $\longrightarrow$ step F.

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

Calculating upper bounds

Plunging

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

Calculating upper bounds

Plunging

- **If** SUP $\leq$ INF, reject subtree, $\rightarrow$ E.
- **Else** plunge subtree as described in B, add the found branch with value SUP to list. Update INF. $\rightarrow$ step C.

The Branch-and-Bound Algorithm

Ranking

Plunging

Branch choosing

Delete bad branches

Calculating upper bounds

Plunging

The current value INF is the optimal solution.

Thank you!

Merci!

Danke!

n	Approximation by convolution					Cohn/Barnhard: "The Stochastic Knapsack Problem with Random Weights"				
	Upper Bound	CPU-time (msec) continuous	Optimum	considered nods	CPU-time (sec) B-and-B	Upper Bound	CPU-time (msec) continuous	Optimum	considered nods	CPU-time (sec) B-and-B
C./B.	4676.208	4	4618.025	100	0.342	4759.000	0	4618.025	144	0.000
15	4934.583	4	4889.781	41	0.139	5146.927	0	4889.781	65	0.002
20	6690.744	6	6650.513	80	0.348	6936.017	0	6650.517	280	0.003
30	10279.908	9	10264.683	455	2.808	10529.541	0	10264.756	2525	0.037
50	16954.343	12	16950.579	13173	131.171	17224.803	0	16950.757	364960	779.325
75	25519.688	16	25513.555	63972	934.550	25811.775	0	*	*	*
100	33846.095	22	*	*	*	34131.754	0	*	*	*
150	50607.008	31	*	*	*	50932.104	0	*	*	*
250	85098.136	52	*	*	*	85459.649	1	*	*	*
500	170110.459	104	*	*	*	170503.708	3	*	*	*
1000	340922.966	240	*	*	*	340822.740	5	*	*	*
5000	1703811.095	1110	*	*	*	1704935.949	107	*	*	*

—Mean over 50 randomly created instances—

*** average CPU time exceeds 1h**