Optimization for Machine Learning

Lecture 13: EM, CCCP, and friends

6.881: MIT

Suvrit Sra
Massachusetts Institute of Technology

06 Apr, 2021

Motivation

(example task)

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

- SVD yields dense B and C
- B and C contain negative entries, even if $A \geq 0$

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

- SVD yields dense B and C
- B and C contain negative entries, even if $A \geq 0$

$$\boxed{\textbf{NMF} \text{ imposes } B \geq 0, C \geq 0}$$

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_{\mathrm{F}}^2 \quad \text{s.t. } B, C \geq 0.$$

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_F^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_{\mathrm{F}}^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

♣ NP-Hard (Vavasis 2007) – no surprise

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_F^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

- ♣ NP-Hard (Vavasis 2007) – no surprise
- ♣ Arora, Ge, Kanna, Moitra (2011) showed that if the matrix A has a special "separable" structure, then actually globally optimal NMF is approximately solvable. More recent progress too

Algorithms

$$A \approx BC \quad \text{s.t. } B, C \geq 0$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_{\mathrm{F}}^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

- ♣ NP-Hard (Vavasis 2007) – no surprise
- ♣ Arora, Ge, Kanna, Moitra (2011) showed that if the matrix A has a special "separable" structure, then actually globally optimal NMF is approximately solvable. More recent progress too

We'll look at simple (local) methods

Background on NMF Algorithms

- Hack: Compute TSVD; "zero-out" negative entries
- Alternating minimization (AM)
- Majorize-Minimize based (MM)
- Global optimization (not covered)
- "Online" algorithms (not covered)

AltMin / AltDesc

$$\min \quad F(B, C)$$

Alternating Descent

1. Initialize B^0, $k \leftarrow 0$

AltMin / AltDesc

$$\min \quad F(B, C)$$

Alternating Descent

1. Initialize $B^0, k \leftarrow 0$
2. Compute C^{k+1} s.t. $F(A, B^k C^{k+1}) \leq F(A, B^k C^k)$

AltMin / AltDesc

$$\min \quad F(B, C)$$

Alternating Descent

1. Initialize B^0, $k \leftarrow 0$
2. Compute C^{k+1} s.t. $F(A, B^k C^{k+1}) \leq F(A, B^k C^k)$
3. Compute B^{k+1} s.t. $F(A, B^{k+1} C^{k+1}) \leq F(A, B^k C^{k+1})$
4. $k \leftarrow k+1$, and repeat until stopping criteria met.

AltMin / AltDesc

$$\min \quad F(B, C)$$

Alternating Descent

1. Initialize B^0, $k \leftarrow 0$
2. Compute C^{k+1} s.t. $F(A, B^k C^{k+1}) \leq F(A, B^k C^k)$
3. Compute B^{k+1} s.t. $F(A, B^{k+1} C^{k+1}) \leq F(A, B^k C^{k+1})$
4. $k \leftarrow k+1$, and repeat until stopping criteria met.

$$\text{(Observe:)} \quad F(B^{k+1}, C^{k+1}) \leq F(B^k, C^{k+1}) \leq F(B^k, C^k)$$

AltMin for NMF: naive version

Alternating Least Squares (ALS)

$$C = \underset{C}{\operatorname{argmin}} \quad \|A - B^k C\|_F^2;$$

AltMin for NMF: naive version

Alternating Least Squares (ALS)

$$C = \operatorname*{argmin}_C \quad \|A - B^k C\|_F^2; \qquad\qquad C^{k+1} \leftarrow \max(0, C)$$

AltMin for NMF: naive version

Alternating Least Squares (ALS)

$$C = \underset{C}{\operatorname{argmin}} \quad \|A - B^k C\|_F^2; \qquad\qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \underset{B}{\operatorname{argmin}} \quad \|A - B C^{k+1}\|_F^2; \qquad\qquad B^{k+1} \leftarrow \max(0, B)$$

AltMin for NMF: naive version

Alternating Least Squares (ALS)

$$C = \operatorname*{argmin}_{C} \quad \|A - B^k C\|_{\mathrm{F}}^2; \qquad\qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \operatorname*{argmin}_{B} \quad \|A - BC^{k+1}\|_{\mathrm{F}}^2; \qquad\qquad B^{k+1} \leftarrow \max(0, B)$$

ALS is *fast, simple, often effective*, but ...

AltMin for NMF: naive version

Alternating Least Squares (ALS)

$$C = \underset{C}{\mathrm{argmin}} \quad \|A - B^k C\|_F^2; \qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \underset{B}{\mathrm{argmin}} \quad \|A - BC^{k+1}\|_F^2; \qquad B^{k+1} \leftarrow \max(0, B)$$

ALS is *fast, simple, often effective*, but ...

$$\|A - B^{k+1}C^{k+1}\|_F^2 \leq \|A - B^k C^{k+1}\|_F^2 \leq \|A - B^k C^k\|_F^2$$

AltMin for NMF: naive version

Alternating Least Squares (ALS)

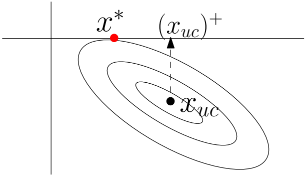
$$C = \operatorname*{argmin}_C \quad \|A - B^k C\|_F^2; \qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \operatorname*{argmin}_B \quad \|A - BC^{k+1}\|_F^2; \qquad B^{k+1} \leftarrow \max(0, B)$$

ALS is *fast, simple, often effective*, but ...

$$\|A - B^{k+1}C^{k+1}\|_F^2 \leq \|A - B^kC^{k+1}\|_F^2 \leq \|A - B^kC^k\|_F^2$$

descent **can fail to hold!**



NMF AltMin: correct way

Use alternating **nonnegative least-squares**

$$C^{k+1} = \operatorname*{argmin}_{C} \quad \|A - B^k C\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad C \geq 0$$

$$B^{k+1} = \operatorname*{argmin}_{B} \quad \|A - BC^{k+1}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad B \geq 0$$

NMF AltMin: correct way

Use alternating **nonnegative least-squares**

$$C^{k+1} = \operatorname*{argmin}_{C} \quad \|A - B^k C\|_F^2 \quad \text{s.t.} \quad C \geq 0$$
$$B^{k+1} = \operatorname*{argmin}_{B} \quad \|A - BC^{k+1}\|_F^2 \quad \text{s.t.} \quad B \geq 0$$

Advantages: Guaranteed descent. Theory of two-block BCD guarantees convergence to a *stationary point*.

Disadvantages: more complex; slower than ALS

NMF AltMin: correct way

Use alternating **nonnegative least-squares**

$$
\begin{aligned}
C^{k+1} &= \underset{C}{\operatorname{argmin}} \quad \|A - B^k C\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad C \geq 0 \\
B^{k+1} &= \underset{B}{\operatorname{argmin}} \quad \|A - BC^{k+1}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad B \geq 0
\end{aligned}
$$

Advantages: Guaranteed descent. Theory of two-block BCD guarantees convergence to a *stationary point*.

Disadvantages: more complex; slower than ALS

Explore. Faster methods; e.g., an SGD-style method for NMF?

NMF AltMin: correct way

Use alternating **nonnegative least-squares**

$$\begin{aligned} C^{k+1} &= \operatorname*{argmin}_{C} \quad \|A - B^k C\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad C \geq 0 \\ B^{k+1} &= \operatorname*{argmin}_{B} \quad \|A - BC^{k+1}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad B \geq 0 \end{aligned}$$

Advantages: Guaranteed descent. Theory of two-block BCD guarantees convergence to a *stationary point*.

Disadvantages: more complex; slower than ALS

Explore. Faster methods; e.g., an SGD-style method for NMF?

Ref. Mairal, Bach, Ponce, Sapiro. *Online Learning for Matrix Factorization and Sparse Coding*. JMLR 11(2):19–60, 2010.

Just Descend

(EM, CCCP, MM methods!)

Revisiting NMF

Consider $F(B, C) = \frac{1}{2}\|A - BC\|_F^2$: convex separately in B and C

Revisiting NMF

Consider $F(B,C) = \frac{1}{2}\|A - BC\|_{\mathrm{F}}^2$: convex separately in B and C

We use $F(C)$ to denote function restricted to C.

Revisiting NMF

Consider $F(B, C) = \frac{1}{2}\|A - BC\|_F^2$: convex separately in B and C

We use $F(C)$ to denote function restricted to C.
Aim: Find C_{k+1} such that $F(B_k, C_{k+1}) \leq F(B_k, C_k)$

Revisiting NMF

Consider $F(B, C) = \frac{1}{2}\|A - BC\|_F^2$: convex separately in B and C

We use $F(C)$ to denote function restricted to C.
Aim: Find C_{k+1} such that $F(B_k, C_{k+1}) \leq F(B_k, C_k)$

Since $F(C)$ *separable* (over cols of C), we just illustrate

$$\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

Remark. This is the well-known NNLS problem.

Revisiting NMF

Consider $F(B, C) = \frac{1}{2}\|A - BC\|_F^2$: convex separately in B and C

We use $F(C)$ to denote function restricted to C.
Aim: Find C_{k+1} such that $F(B_k, C_{k+1}) \leq F(B_k, C_k)$

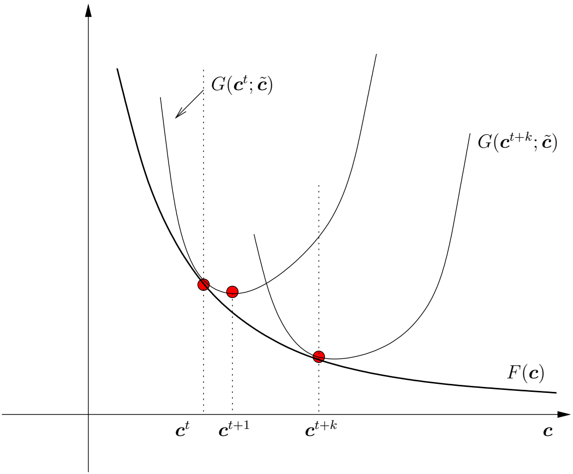
Since $F(C)$ *separable* (over cols of C), we just illustrate

$$\min_{c \geq 0} \quad f(c) = \frac{1}{2}\|a - Bc\|_2^2$$

Remark. This is the well-known NNLS problem.

$$\boxed{\text{Doing descent (not necc minimization) over } f!}$$

The Majorize-Minimize (MM) idea



(**Majorize:** get upper bound; Minorize: minimize this bound)

Descent technique

$$\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1 Find a function $g(c,\tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), && \text{for all} \quad c,\\ g(c,\tilde{c}) &\geq f(c), && \text{for all} \quad c,\tilde{c}. \end{aligned}$$

Descent technique

$$\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), && \text{for all} \quad c, \\ g(c,\tilde{c}) &\geq f(c), && \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c\geq 0} g(c, c^t)$

Descent technique

$$\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

Descent technique

$$\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1})$$

Descent technique

$$\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), \quad \text{for all} \quad c, \\ g(c,\tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c\geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \overset{\text{def}}{\leq} g(c^{t+1}, c^t)$$

Descent technique

$$\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), && \text{for all} \quad c,\\ g(c,\tilde{c}) &\geq f(c), && \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c\geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \overset{\text{def}}{\leq} g(c^{t+1}, c^t) \overset{\text{argmin}}{\leq} g(c^t, c^t)$$

Descent technique

$$\boxed{\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), \quad \text{for all} \quad c, \\ g(c,\tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c\geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \overset{\text{def}}{\leq} g(c^{t+1}, c^t) \overset{\text{argmin}}{\leq} g(c^t, c^t) \overset{\text{def}}{=} f(c^t).$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$f(c) = \frac{1}{2}\sum_i (a_i - b_i^T c)^2 =$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$f(c) = \frac{1}{2}\sum_i (a_i - b_i^T c)^2 = \frac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned} f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\ &= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j b_{ij} c_j\Big)^2 \end{aligned}$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \left(\sum_j b_{ij} c_j\right)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c
\end{aligned}$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned} f(c) &= \tfrac{1}{2}\sum\nolimits_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\ &= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j b_{ij} c_j\Big)^2 \\ &= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j \lambda_{ij} b_{ij} c_j / \lambda_{ij}\Big)^2 \end{aligned}$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum\nolimits_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j b_{ij} c_j\Big)^2 \\
&= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j \lambda_{ij} b_{ij} c_j / \lambda_{ij}\Big)^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij} \big(b_{ij} c_j / \lambda_{ij}\big)^2
\end{aligned}$$

Constructing g for $f(c) = \|a - Bc\|^2$

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0,\ \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum\nolimits_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j b_{ij} c_j\Big)^2 \\
&= \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_i \Big(\sum\nolimits_j \lambda_{ij} b_{ij} c_j / \lambda_{ij}\Big)^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\sum\nolimits_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij} \Big(b_{ij} c_j / \lambda_{ij}\Big)^2 \\
&=: g(c, \tilde{c}), \quad \text{where} \quad \lambda_{ij} \quad \text{are convex coeffts}
\end{aligned}$$

Constructing $g(c, \tilde{c})$

$$\begin{aligned} f(c) &= \tfrac{1}{2}\|a - Bc\|_2^2 \\ g(c, \tilde{c}) &= \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\big(b_{ij}c_j/\lambda_{ij}\big)^2. \end{aligned}$$

Only remains to **pick** λ_{ij} as functions of $\tilde{c}$

Constructing $g(c, \tilde{c})$

$$f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

$$g(c, \tilde{c}) = \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2.$$

Only remains to **pick** λ_{ij} as functions of $\tilde{c}$

$$\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{b_i^T\tilde{c}}$$

Constructing $g(c, \tilde{c})$

$$f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$
$$g(c, \tilde{c}) = \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\big(b_{ij}c_j/\lambda_{ij}\big)^2.$$

Only remains to **pick** λ_{ij} as functions of $\tilde{c}$

$$\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{b_i^T\tilde{c}}$$

Exercise: Verify that $g(c,c) = f(c)$;

Exercise: Let $f(c) = \sum_i a_i \log(a_i/(Bc)_i) - a_i + (Bc)_i$. Derive an auxiliary function $g(c,\tilde{c})$ for this $f(c)$.

Reapting the benefits of g

Key step

$$
\begin{aligned}
g(c, \tilde{c}) &= \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2 \\
c^{t+1} &= \underset{c \geq 0}{\operatorname{argmin}}\, g(c, c^t)
\end{aligned}
$$

Reapting the benefits of g

Key step

$$\begin{aligned} g(c,\tilde{c}) &= \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2 \\ c^{t+1} &= \operatorname*{argmin}_{c\geq 0} g(c, c^t) \end{aligned}$$

Exercise: Solve $\partial g(c,c^t)/\partial c_p = 0$ to obtain ***closed form***

$$c_p = c_p^t \frac{[B^T a]_p}{[B^T B c^t]_p}$$

Reapting the benefits of g

Key step

$$\begin{aligned} g(c,\tilde{c}) &= \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2 \\ c^{t+1} &= \operatorname*{argmin}_{c\geq 0} g(c, c^t) \end{aligned}$$

Exercise: Solve $\partial g(c,c^t)/\partial c_p = 0$ to obtain ***closed form***

$$c_p = c_p^t \frac{[B^T a]_p}{[B^T B c^t]_p}$$

This yields the famous "multiplicative update" algorithm of Lee/Seung (1999) – the paper that popularized NMF.

Broader view of what we just did

- We exploited convexity of x^2

Broader view of what we just did

- We exploited convexity of x^2
- Our technique one instance of more general *Majorization-Minimization* (MM) idea

Broader view of what we just did

- We exploited convexity of x^2
- Our technique one instance of more general ***Majorization-Minimization*** (MM) idea
- Gradient-descent also an MM algorithm (**Why?**)
  Hint: Assume L-smooth function, and then argue

Broader view of what we just did

- We exploited convexity of x^2
- Our technique one instance of more general ***Majorization-Minimization*** (MM) idea
- Gradient-descent also an MM algorithm (**Why?**)
  Hint: Assume L-smooth function, and then argue

Exercise: View few other optim methods via MM lens

Explore: Various other ways of doing MM!

Some key MM methods

- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Convex-Concave Procedure (CCCP)
- Variational Methods
- **Explore:** More broadly, ***d.c. programming***

Example: Variational Methods

Examples

$$
\begin{aligned}
-\log x &= \min_{\lambda} \lambda x - \log \lambda - 1 \\
|w| &= \min_{\lambda \geq 0} \tfrac{1}{2} \frac{w^2}{\lambda} + \tfrac{1}{2}\lambda.
\end{aligned}
$$

Example: Variational Methods

Examples

$$
\begin{aligned}
-\log x &= \min_{\lambda} \lambda x - \log \lambda - 1 \\
|w| &= \min_{\lambda \geq 0} \tfrac{1}{2}\frac{w^2}{\lambda} + \tfrac{1}{2}\lambda.
\end{aligned}
$$

An Introduction to Variational Methods for Graphical Models

MICHAEL I. JORDAN jordan@cs.berkeley.edu
Department of Electrical Engineering and Computer Sciences and Department of Statistics, University of California, Berkeley, CA 94720, USA

ZOUBIN GHAHRAMANI zoubin@gatsby.ucl.ac.uk
Gatsby Computational Neuroscience Unit, University College London WC1N 3AR, UK

TOMMI S. JAAKKOLA tommi@ai.mit.edu
Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA

LAWRENCE K. SAUL lsaul@research.att.edu
AT&T Labs–Research, Florham Park, NJ 07932, USA

Example: Variational Methods

Examples

$$
\begin{aligned}
-\log x &= \min_{\lambda} \lambda x - \log \lambda - 1 \\
|w| &= \min_{\lambda \geq 0} \tfrac{1}{2} \frac{w^2}{\lambda} + \tfrac{1}{2}\lambda.
\end{aligned}
$$

An Introduction to Variational Methods for Graphical Models

MICHAEL I. JORDAN jordan@cs.berkeley.edu
Department of Electrical Engineering and Computer Sciences and Department of Statistics, University of California, Berkeley, CA 94720, USA

ZOUBIN GHAHRAMANI zoubin@gatsby.ucl.ac.uk
Gatsby Computational Neuroscience Unit, University College London WC1N 3AR, UK

TOMMI S. JAAKKOLA tommi@ai.mit.edu
Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA

LAWRENCE K. SAUL lsaul@research.att.edu
AT&T Labs–Research, Florham Park, NJ 07932, USA

See also: *Francis Bach's blog, Posts Jul 1 & Aug 5, 2019.*

Blei, Kucukelbir, McAuliffe. *Variational Inference: A Review for Statisticians*

The EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is a mixture density.

The EM algorithm

Assume $p(x) = \sum_{j=1}^K \pi_j p(x; \theta_j)$ is a mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^n \log\Big(\sum_{j=1}^K \pi_j p(x_i; \theta_j)\Big).$$

The EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is a mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^{n} \log\Big(\sum_{j=1}^{K} \pi_j p(x_i; \theta_j)\Big).$$

Use concavity of $\log t$ to compute lower-bound

$$\ell(\mathcal{X}; \Theta) \geq \sum_{ij} \beta_{ij} \log\left(\pi_j p(x_i; \theta_j)/\beta_{ij}\right).$$

The EM algorithm

Assume $p(x) = \sum_{j=1}^K \pi_j p(x;\theta_j)$ is a mixture density.

$$\ell(\mathcal{X};\Theta) := \sum_{i=1}^n \log\Big(\sum_{j=1}^K \pi_j p(x_i;\theta_j)\Big).$$

Use concavity of $\log t$ to compute lower-bound

$$\ell(\mathcal{X};\Theta) \geq \sum_{ij} \beta_{ij} \log\left(\pi_j p(x_i;\theta_j)/\beta_{ij}\right).$$

E-Step: Optimize over β_{ij}, to set them to *posterior* probabilities:

$$\beta_{ij} := \frac{\pi_j p(x_i;\theta_j)}{\sum_l \pi_l p(x_i;\theta_l)}.$$

The EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is a mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^{n} \log\Big(\sum_{j=1}^{K} \pi_j p(x_i; \theta_j)\Big).$$

Use concavity of $\log t$ to compute lower-bound

$$\ell(\mathcal{X}; \Theta) \geq \sum_{ij} \beta_{ij} \log\left(\pi_j p(x_i; \theta_j)/\beta_{ij}\right).$$

E-Step: Optimize over β_{ij}, to set them to *posterior* probabilities:

$$\beta_{ij} := \frac{\pi_j p(x_i; \theta_j)}{\sum_l \pi_l p(x_i; \theta_l)}.$$

M-Step: optimize the bound over Θ, using above β values

The EM algorithm

Assume $p(x) = \sum_{j=1}^K \pi_j p(x;\theta_j)$ is a mixture density.

$$\ell(\mathcal{X};\Theta) := \sum_{i=1}^n \log\Bigl(\sum_{j=1}^K \pi_j p(x_i;\theta_j)\Bigr).$$

Use concavity of $\log t$ to compute lower-bound

$$\ell(\mathcal{X};\Theta) \geq \sum_{ij} \beta_{ij} \log\left(\pi_j p(x_i;\theta_j)/\beta_{ij}\right).$$

E-Step: Optimize over β_{ij}, to set them to *posterior* probabilities:

$$\beta_{ij} := \frac{\pi_j p(x_i;\theta_j)}{\sum_l \pi_l p(x_i;\theta_l)}.$$

M-Step: optimize the bound over Θ, using above β values

Exercise: Derive a "stochastic" version of EM.

Convex-Concave Procedure

$$\boxed{\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

Convex-Concave Procedure

$$\boxed{\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

CCCP is an MM method

$h(x) \geq h(y) + \langle \nabla h(y),\ x - y \rangle$. Thus,

Convex-Concave Procedure

$$\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

CCCP is an MM method

$$h(x) \geq h(y) + \langle \nabla h(y), x - y \rangle. \text{ Thus,}$$
$$F(x) \leq f(x) - h(y) - \langle \nabla h(y), x - y \rangle =: G(x, y)$$

Convex-Concave Procedure

$$\boxed{\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

CCCP is an MM method

$$h(x) \geq h(y) + \langle \nabla h(y),\ x - y \rangle. \text{ Thus,}$$
$$F(x) \leq f(x) - h(y) - \langle \nabla h(y),\ x - y \rangle =: G(x, y)$$

Observe: $F(x) = G(x, x)$ and $F(x) \leq G(x, y)$. CCCP algo is

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \; G(x, x_k) \\ \nabla f(x_{k+1}) &= \nabla h(x_k) \end{aligned}$$

Convex-Concave Procedure

$$\boxed{\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

CCCP is an MM method

$$h(x) \geq h(y) + \langle \nabla h(y),\ x - y \rangle. \text{ Thus,}$$
$$F(x) \leq f(x) - h(y) - \langle \nabla h(y),\ x - y \rangle =: G(x, y)$$

Observe: $F(x) = G(x, x)$ and $F(x) \leq G(x, y)$. CCCP algo is

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \ G(x, x_k) \\ \nabla f(x_{k+1}) &= \nabla h(x_k) \end{aligned}$$

Exercise: Show that the EM algorithm is a special case of CCCP.

Convex-Concave Procedure

$$\boxed{\min_x F(x) := f(x) - h(x), \text{ where } f, h \text{ are both convex.}}$$

Difference of convex (DC) functions widely studied in d.c. programming. They have many nice properties, including: set of dc functions is a vector space; dc functions are locally Lipschitz on the interior of their domain, etc.

CCCP is an MM method

$$h(x) \geq h(y) + \langle \nabla h(y),\, x - y \rangle. \text{ Thus,}$$
$$F(x) \leq f(x) - h(y) - \langle \nabla h(y),\, x - y \rangle =: G(x, y)$$

Observe: $F(x) = G(x, x)$ and $F(x) \leq G(x, y)$. CCCP algo is

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \; G(x, x_k) \\ \nabla f(x_{k+1}) &= \nabla h(x_k) \end{aligned}$$

Exercise: Show that the EM algorithm is a special case of CCCP.

CCCP often quite useful: always try as a baseline!

Example 1 – Sinkhorn's method

Theorem. (Sinkhorn, 1964). Let A be a strictly positive matrix. There exists a unique doubly stochastic matrix $M = EAD$, where E and D are strictly positive diagonal matrices. Moreover, the iterative procedure of alternatingly normalizing the rows and columns of A to sum to 1 converges to M.

Example 1 – Sinkhorn's method

Theorem. (Sinkhorn, 1964). Let A be a strictly positive matrix. There exists a unique doubly stochasic matrix $M = EAD$, where E and D are strictly positive diagonal matrices. Moreover, the iterative procedure of alternatingly normalizing the rows and columns of A to sum to 1 converges to M.

Theorem. (Yuille, Rangarajan, 2002). Sinkhorn's algorithms is CCCP with cost function: $\phi(r) = -\sum_i \log r_i + \sum_i \log\left(\sum_j r_j A_{ij}\right)$ where $\{r_i\}$ are the diagonal elements of E and the diagonal elements of D are given by $\left(\sum_j r_j A_{ij}\right)^{-1}$.

Exercise: Verify the above claim.

Explore CCCP applied to the so-called *operator scaling problem*

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \ \phi(L) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \; \phi(L) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

$$\nabla\phi(L) = 0 : \quad \sum\nolimits_{i=1}^{n} U_i (U_i^* L U_i)^{-1} U_i^* - n(I+L)^{-1} = 0$$

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \; \phi(L) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

$$\nabla \phi(L) = 0 : \quad \sum\nolimits_{i=1}^{n} U_i (U_i^* L U_i)^{-1} U_i^* - n(I + L)^{-1} = 0$$

Now a simple but crucial trick: write

$$\Delta \;:=\; \tfrac{1}{n} \sum\nolimits_i U_i (U_i^* L U_i)^{-1} U_i^* - (I + L)^{-1}$$

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \ \phi(L) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

$$\nabla\phi(L) = 0 : \quad \sum\nolimits_{i=1}^{n} U_i (U_i^* L U_i)^{-1} U_i^* - n(I+L)^{-1} = 0$$

Now a simple but crucial trick: write

$$\begin{aligned} \Delta &:= \tfrac{1}{n} \sum\nolimits_i U_i (U_i^* L U_i)^{-1} U_i^* - (I+L)^{-1} \\ {\color{red}\Delta + L^{-1}} &= {\color{red}L^{-1}} \quad (\nabla\phi(L) = 0). \end{aligned}$$

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \ \phi(L) := \frac{1}{n} \sum_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

$$\nabla\phi(L) = 0 : \quad \sum_{i=1}^{n} U_i (U_i^* L U_i)^{-1} U_i^* - n(I + L)^{-1} = 0$$

Now a simple but crucial trick: write

$$\begin{aligned} \Delta \quad &:= \quad \tfrac{1}{n} \sum_i U_i (U_i^* L U_i)^{-1} U_i^* - (I + L)^{-1} \\ \Delta + L^{-1} \quad &= \quad L^{-1} \quad (\nabla\phi(L) = 0). \end{aligned}$$

Fixed-point iteration of Mariet-Sra (2015)

$$L_{k+1} \leftarrow L_k + L_k \Delta_k L_k$$

Example 2 – Learning DPP Kernels

$$\max_{L \succ 0} \ \phi(L) := \frac{1}{n} \sum\nolimits_{i=1}^{n} \log\det(U_i^* L U_i) - \log\det(I + L)$$

MLE for learning DPP kernel L; U_i: compression matrices

$$\nabla\phi(L) = 0 : \quad \sum\nolimits_{i=1}^{n} U_i (U_i^* L U_i)^{-1} U_i^* - n(I + L)^{-1} = 0$$

Now a simple but crucial trick: write

$$\begin{aligned} \Delta \quad &:= \quad \tfrac{1}{n} \sum\nolimits_i U_i (U_i^* L U_i)^{-1} U_i^* - (I + L)^{-1} \\ \color{red}{\Delta + L^{-1}} \quad &= \quad \color{red}{L^{-1}} \quad (\nabla\phi(L) = 0). \end{aligned}$$

Fixed-point iteration of Mariet-Sra (2015)

$$L_{k+1} \leftarrow L_k + L_k \Delta_k L_k$$

Remarkably, this generates monotonic $\uparrow$ sequence $\{\phi(L_k)\}_{k \geq 1}$.

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

$$\psi(S) \quad = \quad \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S^{-1})$$

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

$$
\begin{aligned}
\psi(S) &= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S^{-1}) \\
&= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S) + \log\det(S)
\end{aligned}
$$

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

$$\begin{aligned}
\psi(S) &= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S^{-1}) \\
&= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S) + \log\det(S) \\
&= f(S) + h(S),
\end{aligned}$$

where h is concave and f is convex.

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

$$\begin{aligned}
\psi(S) &= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S^{-1}) \\
&= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S) + \log\det(S) \\
&= f(S) + h(S),
\end{aligned}$$

where h is concave and f is convex. Now invoke CCCP (remember we are maximizing). ∎

Demystifying the iteration

Key idea: Write $\psi(S) = \phi(L^{-1})$. Then

$$\begin{aligned}
\psi(S) &= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S^{-1}) \\
&= \frac{1}{n}\sum_i \log\det(U_i^* S^{-1} U_i) - \log\det(I + S) + \log\det(S) \\
&= f(S) + h(S),
\end{aligned}$$

where h is concave and f is convex. Now invoke CCCP (remember we are maximizing). ■

Conjecture. For every "step-size" $\alpha \in (0, \gamma)$, the iteration $L_{k+1} = L_k + \alpha L_k \Delta_k L_k$ generates monotonic $\phi(L_k)$ values.

DC Programming

$$f(x) - g(x)$$

DC programming

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 103, No. 1, pp. 1-43, OCTOBER 1999

DC Programming: Overview

R. HORST[1] AND N. V. THOAI[2]

DC programming

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 103, No. 1, pp. 1–43, OCTOBER 1999

DC Programming: Overview

R. HORST[1] AND N. V. THOAI[2]

Math. Program., Ser. B (2018) 169:5–68
https://doi.org/10.1007/s10107-018-1235-y

FULL LENGTH PAPER

DC programming and DCA: thirty years of developments

Hoai An Le Thi[1] · Tao Pham Dinh[2]

DC programming

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 103, No. 1, pp. 1–43, OCTOBER 1999

DC Programming: Overview

R. HORST[1] AND N. V. THOAI[2]

Math. Program., Ser. B (2018) 169:5–68
https://doi.org/10.1007/s10107-018-1235-y

FULL LENGTH PAPER

DC programming and DCA: thirty years of developments

Hoai An Le Thi[1] · Tao Pham Dinh[2]

Example: The k-th largest singular value: $\sigma_k(X) = \|X\|_k - \|X\|_{k-1}$. This shows that $\sigma_k(\cdot)$ is locally Lipschitz (d.c. functions are known to be LL), which is otherwise a challenging result to establish directly.

DC programming

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 103, No. 1, pp. 1–43, OCTOBER 1999

DC Programming: Overview

R. HORST[1] AND N. V. THOAI[2]

Math. Program., Ser. B (2018) 169:5–68
https://doi.org/10.1007/s10107-018-1235-y

FULL LENGTH PAPER

DC programming and DCA: thirty years of developments

Hoai An Le Thi[1] · Tao Pham Dinh[2]

Example: The k-th largest singular value: $\sigma_k(X) = \|X\|_k - \|X\|_{k-1}$. This shows that $\sigma_k(\cdot)$ is locally Lipschitz (d.c. functions are known to be LL), which is otherwise a challenging result to establish directly.

Note: DC programming does not assume differentiability

DC programming

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 103, No. 1, pp. 1–43, OCTOBER 1999

DC Programming: Overview

R. HORST[1] AND N. V. THOAI[2]

Math. Program., Ser. B (2018) 169:5–68
https://doi.org/10.1007/s10107-018-1235-y

FULL LENGTH PAPER

DC programming and DCA: thirty years of developments

Hoai An Le Thi[1] · Tao Pham Dinh[2]

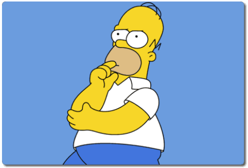
Example: The k-th largest singular value: $\sigma_k(X) = \|X\|_k - \|X\|_{k-1}$. This shows that $\sigma_k(\cdot)$ is locally Lipschitz (d.c. functions are known to be LL), which is otherwise a challenging result to establish directly.

Note: DC programming does not assume differentiability

Explore: DC programming theory, algos, applications.

Amusement

$$I(p) := \sqrt{p} \int_0^\infty \left|\frac{\sin x}{x}\right|^p dx$$



$\boxed{\text{Is } I(p) = f(p) - h(p) \text{ for convex } f, h \text{ for } p \geq 1?}$