Tối ưu hoá (Optimization)

Thiện Lê, tham khảo từ Telgarsky

Tham khảo lecture 29/8, 26/9, 3/10, 10/10, 17/10 của Matus Telgarsky CS581 MLT FA18, UIUC

Sơ lược

Sơ lược

Bài toán

- Cho $((x_i, y_i))_{i=1}^n \in \mathbb{R}^d \times \{-1, 1\}$

Bài toán

- Cho $((x_i, y_i))_{i=1}^n \in \mathbb{R}^d \times \{-1, 1\}$
- Gỉa sử tồn tại ít nhất 1 siêu phẳng chia dữ liệu thành 2 phần $y = 1$ và $y = -1$

Bài toán

- Cho $((x_i, y_i))_{i=1}^n \in \mathbb{R}^d \times \{-1, 1\}$
- Gỉa sử tồn tại ít nhất 1 siêu phẳng chia dữ liệu thành 2 phần $y = 1$ và $y = -1$
- Tìm 1 siêu phẳng chia cắt dữ liệu

Thuật toán perceptron

- Chọn $\boldsymbol{w}_0 := 0$
- Lặp lại:
  - Chọn dữ liệu $\boldsymbol{x}_i, y_i$ bất kỳ chưa xem qua
  - $\boldsymbol{w}_i := \boldsymbol{w}_{i-1} + \boldsymbol{x}_i y_i \mathbb{1}_{\langle \boldsymbol{w}_{i-1}, \boldsymbol{x}_i y_i \rangle \leq 0}$

Thuật toán perceptron

- Chọn $\boldsymbol{w}_0 := 0$
- Lặp lại:
  - Chọn dữ liệu $\boldsymbol{x}_i, y_i$ bất kỳ chưa xem qua
  - $\boldsymbol{w}_i := \boldsymbol{w}_{i-1} + \boldsymbol{x}_i y_i \mathbb{1}_{\langle \boldsymbol{w}_{i-1}, \boldsymbol{x}_i y_i \rangle \leq 0}$

Theorem (Perceptron Convergence Theorem)

Giả sử tồn tại $\boldsymbol{u} \in \mathbb{R}^d, \|\boldsymbol{u}\| = 1$, *đặt* $\gamma := \min_i \{\boldsymbol{u}, \boldsymbol{x}_i y_i\} > 0$, *thì Thuật toán perceptron update không qúa* $\frac{1}{\gamma^2}$ *lần.*

Thuật toán perceptron

- Chọn $\boldsymbol{w}_0 := 0$
- Lặp lại:
  - Chọn dữ liệu $\boldsymbol{x}_i, y_i$ bất kỳ chưa xem qua
  - $\boldsymbol{w}_i := \boldsymbol{w}_{i-1} + \boldsymbol{x}_i y_i \mathbb{1}_{\langle \boldsymbol{w}_{i-1}, \boldsymbol{x}_i y_i \rangle \leq 0}$

Theorem (Perceptron Convergence Theorem)

Giả sử tồn tại $\boldsymbol{u} \in \mathbb{R}^d, \|\boldsymbol{u}\| = 1$, *đặt* $\gamma := \min_i \{\boldsymbol{u}, \boldsymbol{x}_i y_i\} > 0$, *thì Thuật toán perceptron update không qúa* $\frac{1}{\gamma^2}$ *lần.*

Chứng minh.

Tóm tắt: kiểm soát $\frac{1}{2}\|\frac{\boldsymbol{w}_i}{\|w_i\|} - u\|^2 = 1 - \frac{1}{\|w_i\|}\langle w_i, u \rangle$ với mọi i □

Sơ lược

Bài toán ML

ML

Cho: gia đình hàm số $\mathcal{H}$, hàm mất mát $L : \mathcal{H} \to \mathbb{R}$
Tìm: $f^* \in \mathcal{H}$ sao cho $L(f^*)$ đạt giá trị nhỏ nhất

Optimization

Cho: tập hợp A, hàm số $f : A \to \mathbb{R}$
Tìm: $\hat{x} \in A$ sao cho giá trị của $f(\hat{x})$ nhỏ nhất

Độ phức tạp của optimization

Bài toán optimzation tổng quát là bài toán khó

Độ phức tạp của optimization

Bài toán optimzation tổng quát là bài toán khó
Nếu có thuật toán "nhanh" "giải" optimization tổng quát thì $P = NP$

Nhưng không phải optimization nào cũng khó...

Optimization dễ

Một số dạng optimization cụ thể thì dễ hơn.
Ví dụ: hồi quy tuyến tính

Chiến thuật

Đưa về một bài toán optimization dễ rồi tìm nghiệm chính xác
hoặc tìm nghiệm xấp xỉ
hoặc dùng các phương pháp đoán nghiệm (heuristics)

Đưa về bài toán dễ hơn

Ví dụ Parameterization

Khi $\mathcal{H} = \{f : x \to ax + b | a, b \in \mathbb{R}\}$ gia đình hàm tuyến tính, ta nói $\mathcal{H}$ được *tham số hoá (parameterized)* bởi a và b.
Bài toán ML trên $\mathcal{H}$ này với loss L bất kỳ là bài toán optimization trên $\mathbb{R}^2$

Parameterization

Parameterization

Thông thường, $\mathcal{H}$ được *tham số hoá (parameterized)* trong một tập hợp con của $\mathbb{R}^n$

$\Rightarrow$ trong nhiều trường hợp, bài toán ML là bài toán optimization trên một tập con của $\mathbb{R}^n$.

$\Rightarrow$ *một cách giải:* dùng công cụ giải tích trên $\mathbb{R}^n$ (đưa về bài toán tìm nghiệm) (có thể thành công có thể không)

Công cụ giải tích

Điểm yếu của công cụ giải tích

- Có thể thành công, có thể không (đưa về một bài toán tìm nghiệm chưa chắc dễ hơn)
- Chắc chắn không đưa về bài toán dễ hơn được cho các bài toán optimization "khó"
- Phải có người giải đạo hàm, hoặc dùng thuật toán giải đạo hàm tự động

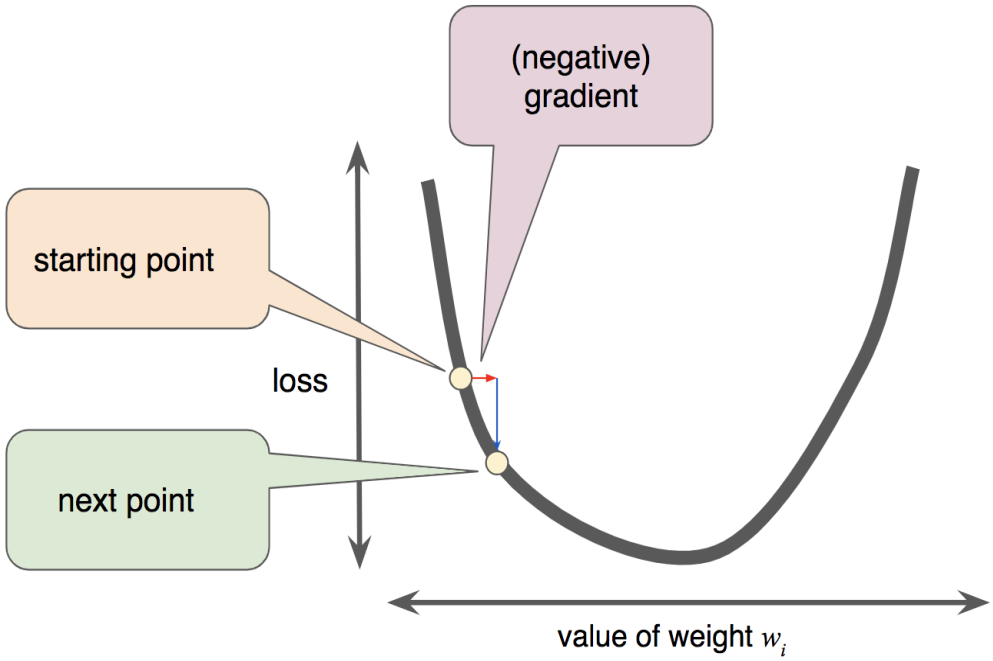
Thuật toán đoán nghiệm (heuristic)

Chốt

Cần những thuật toán nhanh gọn, áp dụng được cho nhiều trường hợp, cho ra đáp án "chấp nhận được" cho dù bài toán optimization có "khó".
$\Rightarrow$ heuristics.

Sơ lược

Ví dụ



(Source: Christian Fei)

Ý tưởng

Ý tưởng

Đạo hàm bậc nhất "chỉ hướng" tới một cực tiểu địa phương (local minimum)
$\Rightarrow$ nhích đi theo hướng chỉ $\Rightarrow$ kiểm tra xem có ở cực tiểu địa phương không (kiểm tra bằng cách nào?)
$\Rightarrow$ nếu phải thì trả về, nếu không thì nhích tiếp

Một bước của Gradient Descent

Cập nhật của gradient descent

Cho một tham số hoá của $\mathcal{H}$ trong $\mathbb{R}$ theo tham số a, hàm $L(x)$ khả vi, hành động nhích đi theo hướng đạo hàm bậc nhất được viết

$$a_n = a_{n-1} - \alpha L'(a_{n-1})$$

Trong đó, $\alpha > 0$ tuỳ chọn, thể hiện nhích lớn hay nhỏ.

Thuật toán đơn biến

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots) > 0$, thực hiện:

1. Chọn a_0 ngẫu nhiên
2. Lặp với $i = 1, 2 \ldots$
   - Nếu $L'(x_i) < \epsilon$ thì trả về a_i
   - Đăt $a_{i+1} = a_i - \alpha_i L'(a_i)$

Đa biến - Một bước

Cho một tham số hoá của $\mathcal{H}$ trong $\mathbb{R}$ theo tham số $\boldsymbol{x}$, hàm mất mát $L(\boldsymbol{x})$ có gradient, hành động nhích đi theo hướng đạo hàm bậc nhất được viêt

$$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} - \alpha \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{i-1})$$

Trong đó, $\alpha > 0$ tuỳ chọn, thể hiện nhích lớn hay nhỏ.

Thuật toán đa biến

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots)$, thực hiện:

1. Chọn $\boldsymbol{x}_0$ ngẫu nhiên
2. Lặp với $i = 1, 2, \ldots$
   - Nếu $L'(\boldsymbol{x}_i) < \epsilon$ thì trả về $\boldsymbol{x}_i$
   - Đặt $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} - \alpha \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{i-1})$

Sơ lược

Định nghĩa lồi

Định nghĩa lồi

Cho một tham số hoá của $\mathcal{H}$ trong $\mathbb{R}$ theo tham số $\boldsymbol{x}$, hàm mất mát $L(\boldsymbol{x})$ khả vi, L lồi nếu

$$\forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d, \forall \lambda \in [0,1], L(\lambda \boldsymbol{a} + (1-\lambda)\boldsymbol{b}) < \lambda L(\boldsymbol{a}) + (1-\lambda)L(\boldsymbol{b})$$

hoặc

$$\forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d, \langle \nabla_{\boldsymbol{x}} f(\boldsymbol{a}) - \nabla_{\boldsymbol{x}} f(\boldsymbol{b}), \boldsymbol{a} - \boldsymbol{b} \rangle \geq 0$$

Convex optimization dễ

Convex optimization dễ

Nhìn chung, nếu L lồi thì bài toán optimization cho L dễ.
Với một số điều kiện của $\mathcal{H}$ (ví dụ, tập hợp lồi), thì mọi local optimum là global optimum.
Nếu đủ giả định của gradient descent thì giải chính xác bài toán ML nếu L và $\mathcal{H}$ lồi.

Định nghĩa Lipschitz

Định nghĩa

Một hàm số $f : \mathbb{R}^d \to \mathbb{R}$ được gọi là β-Lipschitz nếu

$$\forall x, y \in \mathbb{R}^d, |f(x) - f(y)| \leq \beta \|x - y\|$$

Lipschitz gradient

Định nghĩa

Cho một tham số hoá của $\mathcal{H}$ trong $\mathbb{R}$ theo tham số $\boldsymbol{x}$, hàm mất mát $L(\boldsymbol{x})$ khả vi, L có Lipschitz gradient với hằng số β (một số tài liệu gọi là β-smooth) nếu $\nabla_{\boldsymbol{x}} L$ β-Lipschitz.

Ví dụ

Hàm loss trong OLS $L(w) = \|Xw - y\|_2^2$ có Lipschitz gradient với hằng số β trong đó β là singular value lớn nhất của X

β-smooth

Theorem

Cho một tham số hoá của $\mathcal{H}$ trong $\mathbb{R}$ theo tham số $\boldsymbol{x}$, hàm mất mát $L(\boldsymbol{x})$ khả vi, L β-smooth, thì

1. *$L(\boldsymbol{x}_i)$ luôn giảm (hoặc bằng) trong quá trình descent nếu $\alpha \leq 2/\beta$*
2. *Nếu $\alpha = 1/\beta$ thì sau t lần chạy, đã tìm thấy điểm $\boldsymbol{a}$ sao cho $\|\nabla_{\boldsymbol{x}} L(\boldsymbol{a})\| < \sqrt{\frac{2\beta}{t}}$. Điểm a không nhất thiết là điểm cuối cùng (x_t)*
3. *Không có định lý về điểm cuối cùng sẽ về đâu*

Nesterov-Polyak cubic regularization

Theorem

Thêm regularization term bậc 3 có thể đẩy $\|\nabla_{\boldsymbol{x}} L(\boldsymbol{a})\| < \frac{const}{t^{2/3}}$

Smooth và convex

Theorem

Nếu f vừa β-smooth vừa convex thì có định lý về điểm cuối cùng

$$\forall t \in \mathbb{N}, \forall \boldsymbol{x} \in \mathbb{R}^d, |f(\boldsymbol{x}_t) - f(\boldsymbol{x})| < \frac{\beta}{2t}(\|\boldsymbol{x}_0 - \boldsymbol{x}\|_2^2 - \|\boldsymbol{x}_t - \boldsymbol{x}\|_2^2)$$

Saddle point

Theorem

(Vắn tắt) GD khả năng cao (almost surely) không kẹt ở saddle point. (Lee et. al, 2016. Gradient Descent Only Converges to Minimizers.)

Sơ lược

ERM

- Cho
  - Dữ liệu $X = (\boldsymbol{x}_i)_{i=1}^n \in \mathbb{R}^d$
  - Gia đình hàm / mô hình thống kê $\mathcal{H}$
  - Hàm risk $R : \mathcal{H} \times X \to \mathbb{R}$

ERM

- Cho
  - Dữ liệu $X = (\boldsymbol{x}_i)_{i=1}^n \in \mathbb{R}^d$
  - Gia đình hàm / mô hình thống kê $\mathcal{H}$
  - Hàm risk $R : \mathcal{H} \times X \to \mathbb{R}$
- ERM là cách chọn hàm mất mát $L(f) = \frac{1}{n} \sum_{i=1}^n R(f, x_i)$

Thuật toán GD đơn biến

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots) > 0$, thực hiện:

1. Chọn a_0 ngẫu nhiên
2. Lặp với $i = 1, 2 \ldots$
   - Nếu $L'(x_i) < \epsilon$ thì trả về a_i
   - Đăt $a_{i+1} = a_i - \alpha_n L'(a_i)$

Thuật toán đơn biến với ERM

- Cho
  - Dữ liệu $X = (x_i)^n$, gia đình hàm $\mathcal{H}$ với tham số hoá theo $\boldsymbol{x}$
  - Hàm mất mát $L(f) := \frac{1}{n} \sum_{i=1}^{n} R(f, x_i)$

Thuật toán đơn biến với ERM

- Cho
  - Dữ liệu $X = (x_i)^n$, gia đình hàm $\mathcal{H}$ với tham số hoá theo $\boldsymbol{x}$
  - Hàm mất mát $L(f) := \frac{1}{n} \sum_{i=1}^{n} R(f, x_i)$

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots) > 0$, thực hiện:

1. Cho một hàm thí sinh f cố định
2. Chọn a_0 ngẫu nhiên
3. Lặp với $i = 1, 2 \ldots$
   - Tính $L'(f) := \frac{1}{n} \sum_{j=1}^{n} R'(f, x_j)$
   - Nếu $L'(f) < \epsilon$ thì trả về a_i
   - Đặt $a_{i+1} = a_i - \alpha_i L'(f)$

Thuật toán SGD

- Cho
  - Dữ liệu $X = (x_i)_{i=1}^n$, gia đình hàm $\mathcal{H}$
  - Hàm mất mát $L(f) := \frac{1}{n} \sum_{i=1}^n R(f, x_i)$

Thuật toán SGD

- Cho
  - Dữ liệu $X = (x_i)_{i=1}^n$, gia đình hàm $\mathcal{H}$
  - Hàm mất mát $L(f) := \frac{1}{n} \sum_{i=1}^n R(f, x_i)$

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots) > 0$, thực hiện:

1. Cho một hàm thí sinh f cố định
2. Chọn a_0 ngẫu nhiên
3. Lặp với $i = 1, 2 \ldots$
   - Tính $L'(f) := R'(f, x_j)$ với j ngẫu nhiên
   - Nếu $L'(f) < \epsilon$ thì trả về a_i
   - Đặt $a_{i+1} = a_i - \alpha_i L'(f)$

Lý thuyết

- Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In Advances in Neural Information Processing Systems, pages 1067–1077, 2017a.
- Simon S. Du, Xiyu Zhai, Barnabas Poczos, Aarti Singh. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. ICLR 2019
- Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. arXiv preprint arXiv:1804.07795, 2018.

Miền giá trị gò bó

- Bài toán:
  - Cho A compact, convex trong một vector space
  - Cho $f : A \to \mathbb{R}$ convex, khả vi
  - Tìm $\boldsymbol{x}^* \in A$ sao cho $f(\boldsymbol{x}^*)$ nhỏ nhất

Giải với GD

- Khó khăn:
  - f chỉ có giá trị trong A, GD có thể khiến cập nhật vượt ra ngoài A
- Giải quyết:
  - Chiếu kết quả của từng bước GD xuống A

Thuật toán Frank-Wolfe (conditional gradient)

Chọn $\epsilon > 0$, dãy $(\alpha_0, \alpha_1, \alpha_2, \ldots) > 0$, thực hiện:

1. Chọn a_0 ngẫu nhiên
2. Lặp với $i = 1, 2 \ldots$
   - Nếu $\nabla L(a_i) < \epsilon$ thì trả về a_i
   - Tìm $s_i = \text{argmin}_{s \in A} \langle \nabla L(a_i), s \rangle$
   - Đăt $a_{i+1} = a_i \eta_i - s_i(1 - \eta_i)$