DDA5001 Machine Learning

Gradient-based Optimization Algorithm (Part II)

Xiao Li

School of Data Science
The Chinese University of Hong Kong, Shenzhen

Recap: Convex Instances in Machine Learning

We have the following functions are convex:

- Least squares:

$$\mathcal{L}(\boldsymbol{\theta}) = \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2.$$

- Robust linear regression (HW1).
- Logistic regression:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{x}_i\right)\right).$$

- Multi-class logistic regression:

$$\mathcal{L}(\boldsymbol{\Theta}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{K} 1_{\{y_i=\ell\}} \log\left(\frac{\exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x}_i)}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \boldsymbol{x}_i)}\right).$$

- SVM learning problem (later).

Recap: Gradient Descent

Gradient descent (GD)

$$\boxed{\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)}$$

- μ_k is the learning rate / stepsize.

GD is equivalent to

$$\boxed{\boldsymbol{\theta}_{k+1} = \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^d} \; l_k(\boldsymbol{\theta}) := \mathcal{L}(\boldsymbol{\theta}_k) + \nabla \mathcal{L}(\boldsymbol{\theta}_k)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2\mu_k} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_2^2}$$

- $\mathcal{L}(\boldsymbol{\theta}_k) + \nabla \mathcal{L}(\boldsymbol{\theta}_k)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k)$ is linear approximation of $\mathcal{L}$ at $\boldsymbol{\theta}_k$.
- $\frac{1}{2\mu_k} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_2^2$ is proximal term related to learning rate μ_k.
- $l_k(\boldsymbol{\theta})$ is a quadratic function to be minimized at each iteration.

More on Gradient Descent

Gradient Descent with Acceleration

A Useful Algorithm Design Framework

Suppose the task is $\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$, we can design an algorithm as

$$\boxed{\boldsymbol{\theta}_{k+1} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ q_k(\boldsymbol{\theta}) + \frac{1}{2\mu_k} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_2^2 \right\}}$$

μ_k is learning rate-like quantity.

- When $q_k(\boldsymbol{\theta})$ is linear approximation of $\mathcal{L} \Longrightarrow$ **gradient descent**
- When $q_k(\boldsymbol{\theta})$ is second-order approximation of $\mathcal{L} \Longrightarrow$ **Newton's method**
- When $q_k(\boldsymbol{\theta})$ is $\mathcal{L}$ itself $\Longrightarrow$ **proximal point method**
- When $q_k(\boldsymbol{\theta})$ is single component linear approximation of $\mathcal{L} \Longrightarrow$ **stochastic gradient descent (SGD)**
- ...

Many optimization algorithms follow this designing framework.

Convergence Issue

Convergence of Iterative Algorithm

- To solve $\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$, we cannot obtain the solution $\widehat{\boldsymbol{\theta}}$ analytically.
- Design an iterative algorithm, start with $\boldsymbol{\theta}_0$, it will generate

$$\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k, \ldots\}.$$

Convergence analysis of an algorithm concerns:

- Will $\boldsymbol{\theta}_k$ converge to the solution $\widehat{\boldsymbol{\theta}}$? That is

$$\lim_{k \to \infty} \boldsymbol{\theta}_k \stackrel{?}{=} \widehat{\boldsymbol{\theta}}.$$

- If yes, what is the speed of this convergence?

Convergence of GD

- Suppose that $\mathcal{L}$ is convex and differentiable and has Lipschitz continuous gradient with parameter L,

$$\|\nabla\mathcal{L}(\boldsymbol{w}) - \nabla\mathcal{L}(\boldsymbol{u})\|_2 \leq L\|\boldsymbol{w} - \boldsymbol{u}\|_2, \quad \forall \boldsymbol{w}, \boldsymbol{u}$$

$\rightsquigarrow$ Both convexity and Lipschitz gradient are satisfied in LR.

Theorem: Convergence and Convergence rate of GD

Gradient descent with constant learning rate $\mu_k = \mu = 1/L$ satisfies

$$\boxed{\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}(\widehat{\boldsymbol{\theta}}) \leq \frac{L\|\boldsymbol{\theta}_0 - \widehat{\boldsymbol{\theta}}\|_2^2}{2k}}$$

- $\mathcal{L}(\boldsymbol{\theta}_k)$ converges to $\mathcal{L}(\widehat{\boldsymbol{\theta}})$ at the rate of $\mathcal{O}(1/k)$.
- It does not mean $\{\boldsymbol{\theta}_k\}$ converges to $\widehat{\boldsymbol{\theta}}$ at a certain rate.
- $\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}(\widehat{\boldsymbol{\theta}})$ is called sub-optimality gap.
- Proof is put in the supplementary material.

Learning Rate, Stopping Criterion, and Applying GD to LR

The Choice of Learning Rate

Gradient descent:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$

- In practice, the simplest choice for the learning rate is to use a constant learning rate, i.e., $\mu_k = \mu$ for $k = 0, 1, \ldots$ for some relatively small μ. Some typical guess: $0.05, 0.01, 0.005...$
  $\rightsquigarrow$ Typically, a larger model often needs a smaller constant learning rate.
- In optimization, one can also use certain kind of line-search method for choosing μ_k in an adaptive manner.
  $\rightsquigarrow$ However, in machine learning, line-search is not used as it wastes too many computations.
- We will introduce decaying learning rate schedules when we study stochastic gradient descent ($\rightsquigarrow$ later).

Stopping Criterion for GD

A practical question: Run GD, when to stop?

- We cannot let k go to infinity in practice since the algorithm will never stop in this way.
- We need a practical stopping criterion.

Typical stopping criteria:

- Fix the total number of iterations as K.
- Stop when $\|\nabla \mathcal{L}(\boldsymbol{\theta}_k)\|_2$ is small, say $\|\nabla \mathcal{L}(\boldsymbol{\theta}_k)\|_2 \leq \varepsilon$.
  - since $\|\nabla \mathcal{L}(\widehat{\boldsymbol{\theta}})\|_2 = 0$ at minimum $\widehat{\boldsymbol{\theta}}$.
- Stop when $\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_2$ is small, say $\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_2 \leq \varepsilon$.
- In machine learning, stop when validation error is going to increase. ($\rightsquigarrow$Later)

Applying GD to LR

Given a set of training data points: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, the learning problem of binary LR is:

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathbb{R}^d} \left\{ \mathcal{L}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + \exp \left(-y_i \cdot \boldsymbol{\theta}^\top \boldsymbol{x}_i \right) \right) \right\}.$$

- $\mathcal{L}$ is convex and has Lipschitz gradient.
- We can apply GD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$

  and it has convergence guarantee of $\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}(\widehat{\boldsymbol{\theta}}) \leq \mathcal{O}(1/k)$ if the learning rate is chosen properly.
- The main elements are: 1. Compute the gradient and form the search direction. 2. Determine learning rate (usually a small constant). 3. Stop when the stopping criterion is satisfied.
- Need to know how to compute the gradient by chain rule.

Some Pros and Cons for GD

Pros:

- GD has simple implementation.
- It works well for almost all differentiable convex problems.

Cons:

- The convergence speed of GD is relatively slow.

Is there an method that simultaneously has simple implementation and fast convergence speed? $\rightsquigarrow$ Acceleration technique.

More on Gradient Descent

Gradient Descent with Acceleration

Gradient Descent with Momentum

Gradient Descent with Momentum

GD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$

- GD forces suffcient decrease at each iteration, leaving the possibility of exploring more efficient search direction.

A quite popular technique for accelerating gradient descent method is the momentum technique:

$$\boxed{\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k) + \beta_k \underbrace{(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})}_{\text{momemtum}}}$$

An equivalent form:

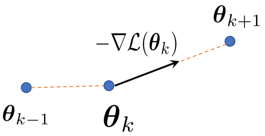
$$\boxed{\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \boldsymbol{m}_k \\ \boldsymbol{m}_k &= \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k) + \beta_k \boldsymbol{m}_{k-1} \end{aligned}}$$

- Due to Boris T. Polyak.
- Each iteration takes nearly the same time cost as GD.
- Widely used in practice, notably in Adam algorithm.

Momentum Acceleration: Geometric Interpretation

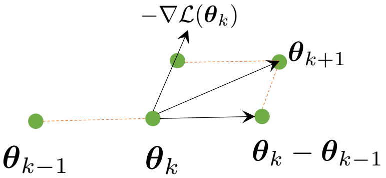
GD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$



GD with momentum:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k) + \beta_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$$



Nesterov's Accelerated Gradient Descent

Nesterov's Acceleration

GD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$

Another very useful technique to accelerate GD is Nesterov's accelerated gradient descent (AGD):

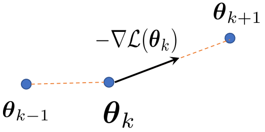
$$\boxed{\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{w}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{w}_k) \\ \boldsymbol{w}_k &= \boldsymbol{\theta}_k + \frac{k-1}{k+2}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \end{aligned}}$$

- Nesterov's acceleration is motivated by momentum but with gradient evaluated at the extrapolated point and a specific choice of β_k.
- Each iteration takes nearly the same time cost as GD.
- Widely used in practice.

Nesterov's Acceleration: Geometric Interpretation

GD:

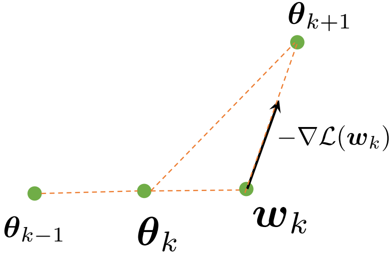
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$$



AGD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{w}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{w}_k)$$

$$\boldsymbol{w}_k = \boldsymbol{\theta}_k + \frac{k-1}{k+2}(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$$



Convergence of AGD

- Suppose that $\mathcal{L}$ is convex and differentiable and has Lipschitz continuous gradient with parameter L (For example, LR problem).

Theorem: Convergence of AGD

Accelerated gradient descent with constant learning rate $\mu_k = \mu = 1/L$ satisfies

$$\boxed{\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}(\widehat{\boldsymbol{\theta}}) \leq \frac{2L\|\boldsymbol{\theta}_0 - \widehat{\boldsymbol{\theta}}\|_2^2}{(k+1)^2}}$$

- $\mathcal{L}(\boldsymbol{\theta}_k)$ converges to $\mathcal{L}(\widehat{\boldsymbol{\theta}})$ at the rate of $\mathcal{O}(1/k^2)$.
- This rate is much faster than GD, which only has $\mathcal{O}(1/k)$ convergence rate.

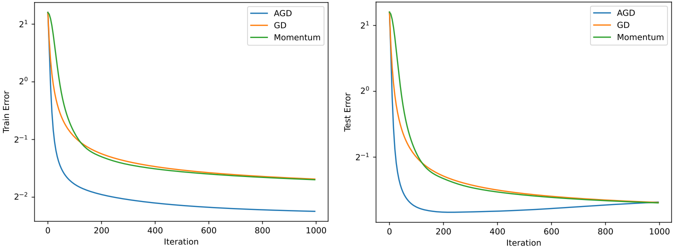
Solving Logistic Regression

Learning problem: Multi-class logistic regression

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta} \in \mathbb{R}^{(d+1)\times K}}{\text{argmin}} \quad -\frac{1}{n} \sum_{i=1}^{n} \sum_{\ell=1}^{K} 1_{\{y_i = \ell\}} \log \left(\frac{\exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x}_i)}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^\top \boldsymbol{x}_i)} \right)$$

Setup:

- MNIST classification. $n = 60000, d = 784, K = 10.$
- $\beta_k = 0.98$ in GD with momentum.
- Learning rate $\mu_k = 2 \times 10^{-2}$.



⇝ We will implement in HW2.

⇝ Next lectures: Overfitting.