DDA5001 Machine Learning

Convex Optimization & Gradient-based Optimization Algorithm

Xiao Li

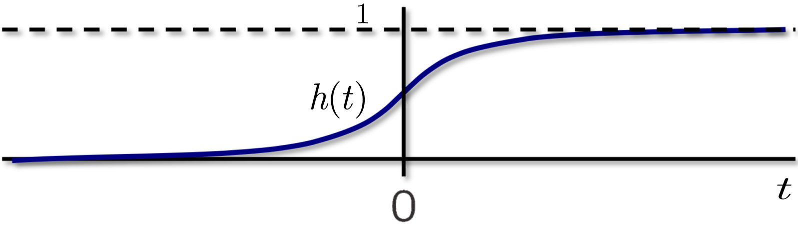
School of Data Science
The Chinese University of Hong Kong, Shenzhen

Recap: Logistic Function

The function

$$h(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

is called the logistic function or sigmoid.



Sigmoid: 'S'-like function.

Some other 'S'-like function: Hyperbolic tangent: $\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$.

Recap: Logistic Regression for Binary Classification

Logistic Regression (LR) Model:

$$\boxed{\Pr_{\boldsymbol{\theta}}[y|\boldsymbol{x}] = \frac{1}{1+\exp\left(-y\cdot\boldsymbol{\theta}^\top\boldsymbol{x}\right)}}$$

Through MLE principle, the learning problem of LR is given by

$$\boxed{\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\text{argmin}} \ \frac{1}{n}\sum_{i=1}^{n}\log\left(1+\exp\left(-y_i\cdot\boldsymbol{\theta}^\top\boldsymbol{x}_i\right)\right)}$$

- LR is for classification.
- LR is a linear classifier.

Recap: Softmax and Multi-class Logistic Regression

- Consider K classes. Assign each class $k = 1, \ldots, K$ a parameter / weight vector $\boldsymbol{\theta}_k$.
- Let $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\right] \in \mathbb{R}^{(d+1)\times K}$ and $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be the training data.
- Softmax:

$$\boxed{\Pr_{\boldsymbol{\Theta}}\left[y_i = k | \boldsymbol{x}_i\right] = \frac{\exp(\boldsymbol{\theta}_k^\top \boldsymbol{x}_i)}{\sum_{j=1}^K \exp(\boldsymbol{\theta}_j^\top \boldsymbol{x}_i)}}$$

- Multi-class logistic regression learning problem:

$$\boxed{\widehat{\boldsymbol{\Theta}} = \operatorname*{argmin}_{\boldsymbol{\Theta}\in\mathbb{R}^{d\times K}} \; -\frac{1}{n}\sum_{i=1}^n \sum_{k=1}^K 1_{\{y_i = k\}} \log\left(\frac{\exp(\boldsymbol{\theta}_k^\top \boldsymbol{x}_i)}{\sum_{j=1}^K \exp(\boldsymbol{\theta}_j^\top \boldsymbol{x}_i)}\right),}$$

How to Learn $\widehat{\boldsymbol{\theta}}$?

The objective function (using binary logistic regression as an example)

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

The learning problem (from MLE principle and how to make $\mathrm{Er}_{\mathrm{out}}$ small)

$$\boxed{\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\text{argmin}} \ \mathcal{L}(\boldsymbol{\theta})}$$

- Bad news ✗: No closed-form solution.
- Good news ✓: The objective function $\mathcal{L}(\boldsymbol{\theta})$ is convex in $\boldsymbol{\theta}$.

  $\rightsquigarrow$ Convex optimization and gradient-based learning algorithm.

Convex Optimization

Gradient-based Optimization Algorithms

What is Convex Optimization?

Consider the optimization problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$$

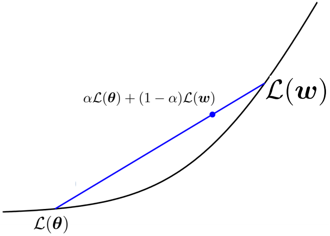
- The optimization problem is said to be convex optimization if $\mathcal{L}(\boldsymbol{\theta})$ is a convex function.
- Otherwise, it is called nonconvex optimization.

Definition of Convex Function

Definition: Convex function

A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $\boldsymbol{\theta}, \boldsymbol{w} \in \mathbb{R}^d$ and any $\alpha \in [0, 1]$,

$$\mathcal{L}(\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{w}) \leq \alpha\mathcal{L}(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}(\boldsymbol{w})$$



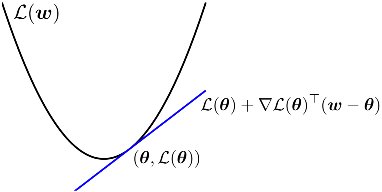
- Geometric intuition: Uniform upward curvature.
- Simple examples: $\mathcal{L}(\theta) = \theta, \mathcal{L}(\theta) = \theta^2, \mathcal{L}(\theta) = |\theta|, \mathcal{L}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|^2$.

First-order Characterization of Convexity

Theorem: First order convexity characterization

Suppose $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is differentiable. $\mathcal{L}$ is convex if and only if for all $\boldsymbol{\theta}, \boldsymbol{w} \in \mathbb{R}^d$

$$\mathcal{L}(\boldsymbol{w}) \geq \mathcal{L}(\boldsymbol{\theta}) + \nabla \mathcal{L}(\boldsymbol{\theta})^\top (\boldsymbol{w} - \boldsymbol{\theta}).$$



- This theorem is often used for analysis.
- Implication:

$$\nabla \mathcal{L}(\boldsymbol{\theta}^\star) = \mathbf{0} \text{ if and only if } \boldsymbol{\theta}^\star \text{ is global minima.}$$

- This is how we find the optimal parameters for Least squares.

Second-order Characterization of Convexity

Theorem: Convexity via Hessian

Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be twice continuously differentiable. Then $\mathcal{L}$ is convex if and only if its Hessian matrix is positive semidefinite (PSD), i.e.,

$$\boldsymbol{d}^\top \nabla^2 \mathcal{L}(\boldsymbol{\theta}) \boldsymbol{d} \geq 0 \quad \forall\, \boldsymbol{d} \in \mathbb{R}^d, \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^d.$$

- A way to test convexity if the objective function is twice cont. differentiable.

Examples: 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:

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

- SVM learning problem (later).

The Advantage of Convex Optimization

- No local minimum. Zero gradient means global optimal solution, corresponding to $\widehat{\boldsymbol{\theta}}$.
- Though we usually do not have closed-form solution, but we have reliable and efficient algorithms to find the global minimum, i.e., points provide zero gradient.
- There are a set of fully developed algorithmic tools for convex optimization.

Algorithms:

- Gradient-based method.
- Subgradient method (HW2).
- ...

The 'Easy' and 'Difficult' Optimization Problems

- Linear v.s. nonlinear?
- Differentiable v.s. nondifferentiable?

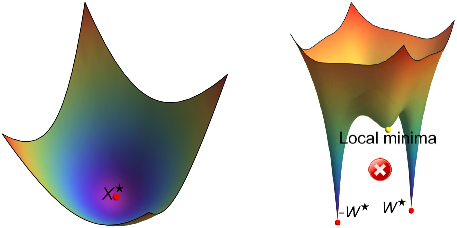


Figure: Convex geometry and nonconvex geometry.

Classify whether a problem is hard or easy: Convex (easy) v.s. nonconvex (hard).

- Convex optimization: Reasonable algorithms can almost always find the global minimizer, i.e., $\widehat{\boldsymbol{\theta}}$.
- Nonconvex optimization: It is very hard to find a global minimizer.

Algorithms for Learning $\widehat{\boldsymbol{\theta}}$

What we have so far?

- Logistic regression does not have a closed-from solution.
- Logistic regression is a convex optimization problem.
- Convex optimization problems are easy to solve.

$\rightsquigarrow$ Algorithmic tool:

Gradient-based optimization algorithms.

Convex Optimization

Gradient-based Optimization Algorithms

Iterative Algorithm

Iterative algorithm

Start with an initial point $\boldsymbol{\theta}_0$, an iterative algorithm $\mathcal{A}$ will generate a sequence of iterates

$$\boxed{\boldsymbol{\theta}_{k+1} = \mathcal{A}(\boldsymbol{\theta}_k)}$$

for $k = 0, 1, 2, \ldots$

- k represents iteration, an indexing number.
- $\boldsymbol{\theta}_k$ represents iterate at k-th iteration.

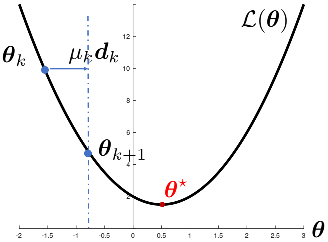
What form $\mathcal{A}$ usually has in practice?

$$\boxed{\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mu_k \boldsymbol{d}_k}$$

- $\mathbb{R} \ni \mu_k > 0$ is stepsize / learning rate.
- $\boldsymbol{d}_k \in \mathbb{R}^d$ is the search direction, typically depends on $\boldsymbol{\theta}_k$.
- The key is to choose a proper direction $\boldsymbol{d}_k$ at each iteration.

Illustration and Important Elements

Iterative algorithm: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mu_k \boldsymbol{d}_k$.



- The new iterate $\boldsymbol{\theta}_{k+1}$ is expected to be closer to $\boldsymbol{\theta}^\star$ than $\boldsymbol{\theta}_k$

Things to determine:

- Initial point $\boldsymbol{\theta}_0$ (fine for convex optimization).
- Search direction $\boldsymbol{d}_k$.
- Learning rate μ_k.
- Stopping criterion.

Search Direction $\boldsymbol{d}_k$

Goal:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}).$$

The least:

$\boldsymbol{d}_k$ should point to a direction that decreases the function value.

Propsition: Descent direction

Suppose $\mathcal{L}$ is continuously differentiable, if there exists a $\boldsymbol{d}$ such that

$$\nabla \mathcal{L}(\boldsymbol{\theta})^\top \boldsymbol{d} < 0$$

then, there exists a $\tilde{\mu} > 0$ such that

$$\mathcal{L}(\boldsymbol{\theta} + \mu \boldsymbol{d}) < \mathcal{L}(\boldsymbol{\theta})$$

for all $\mu \in (0, \tilde{\mu})$. Thus, $\boldsymbol{d}$ is a descent direction at $\boldsymbol{\theta}$.

- The proposition can be proved by Taylor Theorem.

Gradient Descent

- This proposition tells us: At k-th iteration, find a $\boldsymbol{d}_k$ satisfying

$$\nabla \mathcal{L}(\boldsymbol{\theta}_k)^\top \boldsymbol{d}_k < 0.$$

Then, $\boldsymbol{d}_k$ must be a descent direction at the current iterate $\boldsymbol{\theta}_k$.

Thus, one possible choice is

$$\boxed{\boldsymbol{d}_k = -\nabla \mathcal{L}(\boldsymbol{\theta}_k)}$$

The resultant algorithm

Gradient descent (GD)

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

Gradient Descent: Interpretation

The gradient descent

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

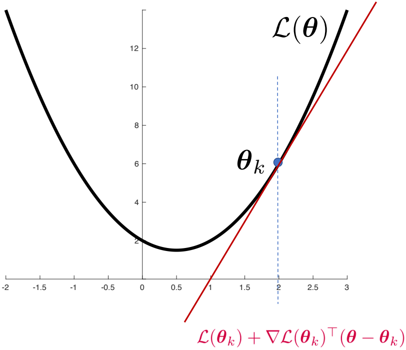
can be equivalently written as

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

- $\mathcal{L}(\boldsymbol{\theta}_k) + \nabla \mathcal{L}(\boldsymbol{\theta}_k)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k)$ 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.

Gradient Descent: Interpretation

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



- Cannot directly minimize the linear approximation.
- The linear approximation is accurate only around θ_k.
- Thus, we need the proximal term.

- The proximal term is used to control how far the algorithm goes.

Gradient Descent: Interpretation

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

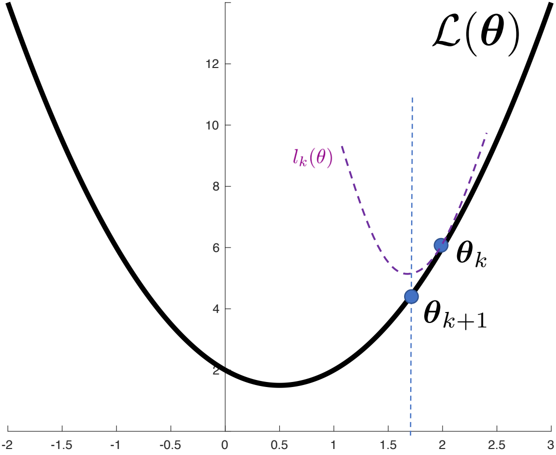
- $l_k(\boldsymbol{\theta})$ is a quadratic function in $\boldsymbol{\theta}$
- At each iteration, we have closed-form update, i.e., the gradient descent algorithm

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

Almost universal algorithmic design strategy

Solving the original problem by solving a sequence of simpler subproblems.

Gradient Descent: Interpretation



- Iteratively construct $l_k(\boldsymbol{\theta})$ to get the next $\boldsymbol{\theta}_{k+1}$

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

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

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.

$\rightsquigarrow$ Next lecture: More on GD and the starting of overfitting.