7. Continuous Optimization

Good set of parameters: by objective function or the probabilistic model

Continuous optimization: unconstrained and constrained optimization

Assume that our objective function is differentiable, hence have access to a gradient at each location in the space to help us find the optimum value

By convention, most objective functions in ML are intended to be minimized

Intuitively finding the best value is like finding the valleys of the objective function, and the gradients point us uphill

Idea to move downhill (opposite to the gradient) and hope to find the deepest point

For unconstrained optimization, this is the only concept we need

For constrained optimization, we need to introduce other concepts to manage the constraints

1. 한 번 미분 -〉 two are minimums and one is a maximum

2. To check whether a stationary point is a minimum or maximum, we need to take the derivative a second time and check whether the second derivative is positive or negative

For convex functions, all local minimums are global minimums

전제: a one-dimensional function

7.1 Optimization using Gradient Descent

min f(x) where f: Rd -〉 R and f is differentiable

-To find a local min of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point(목적 함수의 경사를 따라서 내려감으로써 최소값을 만드는 값을 찾아내는 기법)

Gradient points in the direction of the steepest ascent

-Consider the set of lines where the function is at a certain value which are known as the contour lines(등고선?..)

Gradient points in a direction that is orthogonal to the contour lines of the function we wish to optimize

Multivariate function

$x1 = x0 - \gamma((\nabla f)(x0))^\top$

for a small step-wise $\gamma >= 0$, then, $f(X1) <= f(x0)$

Allow us to define a simple gradient descent algorithm

Local optimum $f(x*)$ of a function $f: Rn -> R$, $x -> f(x)$

Start with an initial guess x0 of the parameters we wish to optimize and iterate

$Xi+1 = xi - \gamma((\nabla f)(xi))^\top$

7.1.1 Step-wise

momentum: Smoothes erratic behavior of gradient updates and dampens oscillations(진동을 감쇠시킴)

Adaptive gradient methods rescale the step-size at each iteration

2 Heuristics: When the function value increases after a gradient step, the step-seize was too large. Undo the step and decrease the step-size.

When the function value decreases the step could have been larger. Try to increase the step-size.

ㄴ Guaranteeing monotonic convergence

Remark: When applied to the solution of linear systems of equations Ax=b, gradient descent may converge slowly

Speed of convergence of gradient descent dependent on the condition number (ratio of the max to the min singular value)

Condition number essentially measures the ratio of the most curved direction versus the least curved direction

Instead of directly solving Ax = b, could instead solve P−1(ax−b)=0

7.1.2 Gradient Descent With Momentum

Convergence of gradient descent may be very slow if the curvature of the optimization surface is such that there are regions that are poorly scaled

Gradient descent with momentum: method that introduces an additional term to remember what happened in the previous iterations

Dampens oscillation and smoothes out the gradient updates

Ball analogy

Idea is to have a gradient update with memory to implement a moving average

The moment-based method remembers the update

$x_{i+1} = x_i - \gamma_i((\nabla f)(x_i))^\top + \alpha \Delta x_i$

$\Delta x_i = x_i - x_{i-1} = \alpha \Delta x_{i-1} - \gamma_{i-1}((\nabla f)(x_{i-1}))^\top$

Momentum term is useful since it averages out different noisy estimates of the gradient.

One particular way, stochastic approximation.

7.1.3 Stochastic Gradient Descent

SGD: a stochastic approximation of the GD method for minimizing an objective function that is written as a sum of differentiable functions

Stochastic: Don't know the gradient precisely, but instead only know a noisy approximation

Standard GD: a batch optimization (performed using the full training set) by updating the vector of parameters according to

- ➔ May require expensive evaluations of the gradients from all individual functions
- ➔ Esp. training set is enormous and no simple formulas exist

SGD 에서는 randomly choose a subset of Ln for mini-batch gradient descent

Randomly select only a single Ln to estimate the gradient

Why take a subset? A. For gradient descent to converge, we only require that the gradient is an unbiased estimate of the true gradient

Any other unbiased empirical estimate of the expected value would suffice for convergence of gradient descent

Why consider using an approximate gradient? Major reason: size of CPU/GPU memory limit. Small mini batches are quick to estimate.

Goal of ML: overall goal is to improve generalization performance. Since the goal does not necessarily need a precise estimate of the minimum of the objective function, approximate gradients using mini-batch approaches have been widely used

Very effective in large-scale problems ex. Training DNN on millions of images, topic models, reinforcement learning, or training of large-scale Gaussian process models

7.2 Constrained Optimization and Lagrange Multipliers

Additional constraints: for real-valued functions, consider the constrained optimization problem

Converting constrained problem->unconstrained

1. Use an indicator function where 1(z) is an infinite step function

2. Lagrange multipliers(라그랑주 승수법, Replace the step function with a linear function)

어떤 문제의 최적점이 아닌, 최적점이 되기 위한 조건을 찾는 방법. 최적해의 필요 조건 찾는 방법.

최적화라는 값에 형식적인 라그랑주 승수(multiplier) 항을 더하여, 제약된 문제를 제약이 없는 문제로 바꾸는 방법.

Lagrangian duality:

라그랑지안	$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\vec{w} \cdot \vec{x} + b) - 1]$
라그랑주 듀얼 함수	$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j$

https://ugong2san.tistory.com/2633

Constrain Lagrange multipliers corresponding to the inequality constraints to be non-negative, and leave the Lagrange multipliers corresponding to the equality constraints unconstrained

7.3 Convex Optimization

Convex optimization problem: f(.) is a convex function, and when the constraints involving g(.) and h(.) are convex sets

Optimal solution of the dual problem is same as the optimal solution of the primal problem

Distinction between convex functions and convex sets are not strictly present in ML

7.3.2 Quadratic Programming

Quadratic program

Consider the case of a convex quadratic objective function, where the constraints are affine, i.e.,

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}\,, \end{aligned} \tag{7.45}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times d}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{c} \in \mathbb{R}^d$. The square symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ is positive definite, and therefore the objective function is convex. This is known as a *quadratic program*. Observe that it has d variables and m linear constraints.

The Lagrangian is given by

$$\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{7.48a}$$

$$= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b}\,, \tag{7.48b}$$

where again we have rearranged the terms. Taking the derivative of $\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ and setting it to zero gives

$$\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) = \boldsymbol{0}\,. \tag{7.49}$$

Assuming that $\boldsymbol{Q}$ is invertible, we get

$$\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})\,. \tag{7.50}$$

Substituting (7.50) into the primal Lagrangian $\mathfrak{L}(\boldsymbol{x},\boldsymbol{\lambda})$, we get the dual Lagrangian

$$\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b}\,. \tag{7.51}$$

Therefore, the dual optimization problem is given by

$$\begin{aligned} \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \quad & -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b} \\ \text{subject to} \quad & \boldsymbol{\lambda} \geqslant \boldsymbol{0}\,. \end{aligned} \tag{7.52}$$

We will see an application of quadratic programming in machine learning in Chapter 12.

7.3.3 Legendre-Fenchel Transform and Convex Conjugate

We can fill up a convex function to obtain the epigraph, which is a convex set

Thus, we can also describe convex functions in terms of their supporting hyperplanes

Summary: Since convex sets can be equivalently described by its supporting hyperplanes, convex functions can be equivalently described by a function of their gradient

Legendre-Fenchel transform: a transformation from a convex differentiable function f(x) to a function that depends on the tangents

Transformation of the function f(.) and not the variable x or the function evaluated at x

Aka convex conjugate

convex conjugate

Definition 7.4. The *convex conjugate* of a function $f : \mathbb{R}^D \to \mathbb{R}$ is a function f^* defined by

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x} \in \mathbb{R}^D} \left(\langle \boldsymbol{s}, \boldsymbol{x} \rangle - f(\boldsymbol{x}) \right) . \tag{7.53}$$

Note that the preceding convex conjugate definition does not need the function f to be convex nor differentiable

The conjugate function has nice properties: For convex functions, applying the Legendre transform again gets us back to the original function

Derived a dual optimization problem using Lagrange multipliers

For convex optimization problems, we have strong duality, that is the solutions of the primal and dual problem match

The Legendre-Fenchel transform described here also can be used to derive a dual optimization problem

+) When convex and differentiable, the supremum is unique