CIS3990-002: Mathematics of Machine Learning Fall 2023

Lecture: Continuous Optimization

Date: November 15th, 2023 *Author: Eric Wong*

1 Continuous optimization

The idea of continuous optimization is that to minimize a function, we can look at a small window and minimize our function within that small window. We assume that our function is defined over a continuous space, and hence the notion of a small window exists. In contrast, discrete optimization searches over discrete values, and there is no such thing as a "small window".

- Consider $\ell(x) = (x-1)(x-2)(x-3) = x^3 - 6x^2 + 11x - 6$. We can compute the gradient as

$$\frac{d\ell(x)}{dx} = 3x^2 - 12x + 11$$

- The classic way to minimize this function would be to check where the derivatives are zero, and the extreme points, and then whether these are local minima or local maxima.
- In continuous optimization, we instead run an iterative procedure that should converge to a local minima. This is useful when we can't analytically find a solution.
- In the most general from, we are interested in solving for $\min_x f(x)$. This can be viewed as a generalization of the risk minimization problem,

$$\min_\theta R_{\text{emp}}(f_\theta) = \min_\theta \sum_i \ell(f_\theta(x_i), y_i)$$

  but for this section we'll just abstract it away as $\min_x f(x)$. $f(x)$ is often considered the objective.

1.1 Gradient descent and step sizes

- Even though we are abstracting away the objective, in order to minimize this function, we have to make some kind of assumption. For gradient descent, we assume f is differentiable. Then, gradient descent uses some initial guess x_0 and iterates for all i:

$$x_{t+1} = x_t - \gamma_i (\nabla f(x_i))^T$$

  for a small step size γ_i.
- The intuition is that gradient descent exploits the fact that $f(x_i)$ decreases fastest if one moves in the direction of the negative gradient. One can imagine that gradient descent releases a ball on the surface of the function, and moves downhill in the direction of steepest descent.

- Another intuition is to consider what is called the contour lines, $f(x) = c$ for some value c. The gradient points in the direction that is orthogonal to the contour line of what we want to optimize.
- For a suitable step-size γ_i, then $f(x_0) \geq f(x_1) \geq \dots$ converges to a local minimum.

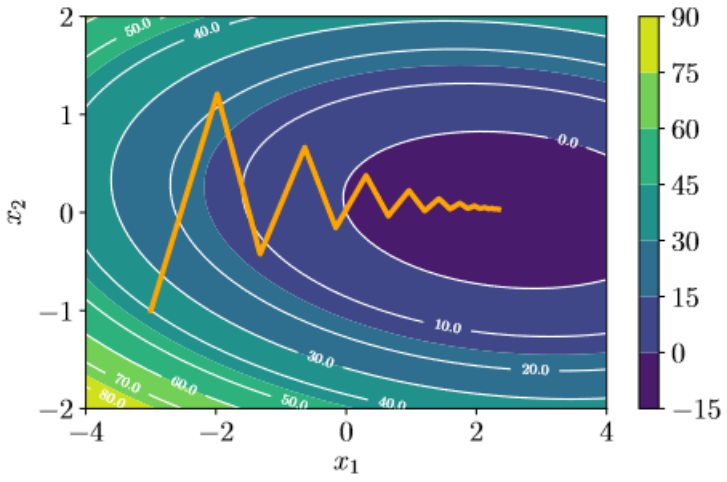

Figure 7.3 Gradient descent on a two-dimensional quadratic surface (shown as a heatmap). See Example 7.1 for a description.

Figure 1: Figure 7.3 from the textbook

- While very general (we only assumed differentiability of f, gradient descent can be quite slow in practice: its asymptotic rate of convergence is inferior to many other methods. In particular, if the ball is rolling down a hill, if the slope is a very long and thin valley then it will zig zag and make slow progress (the problem is then called *poorly conditioned*).
- The step size is critical: if too small, gradient descent is too slow. If too large, gradient descent will overshoot, fail to converge, or even diverge.
- Heuristically: if $f(x_i) > f(x_{i+1}$ increases, then it was too large: undo and decrease the step size. If $f(x_i) < f(x_{i+1})$ but λ_i could have been larger, try to increase the step size. Redo the step if the larger step size still decreases the objective.
- This guarantees monotonic convergence: $f(x_i) < f(x_{i+1})$ for all i.
- Example: $\min_x \|Ax-b\|^2$ for some matrix A and vector b (this is a linear system of equations).

$$\|Ax - b\|^2 = (Ax - b)^\top (Ax - b) \Rightarrow \nabla f(x) = 2(Ax - b)^\top A$$

- The condition number of this matrix is $\kappa = \frac{\sigma(A)_{\max}}{\sigma(A)_{\min}}$ where $\sigma(A)$ are the eigenvalues of $A^\top A$. If this is very large, the problem is poorly-conditioned–this is the ratio of the most curved vs. least curved direction. Think long flat valleys—very curved in one direction, but very flat in another.
- A preconditioner P tries to flatten out these values, and instead solve $P^{-1}(Ax - b) = 0$ in the hopts that $P^{-1}A$ has a better condition number.

1.2 Momentum, minibatching

While this is the classic version of gradient descent, a number of bells and whistles have been developed for this approach.

- The valley problem occurs because gradient descent hops between the walls of a valley and therefore makes slow progress. One way to tweak the algorithm is to maintain a memory so that the algorithm can "remember" the direction of the valley.
- Momentum: the intuitive version is that a ball rolling down a slope has inertia, and is reluctant to change direction. This inertia is the memory.
- Formally:
$$x_{i+1} = x_i - \gamma_i(\nabla f(x_i)^\top + \alpha \Delta x_i)$$
where
$$\Delta x_i = x_i - x_{i-1} = \alpha \Delta x_{i-1} - \gamma_{i-1} \nabla f(x_{i-1})^\top$$
In other words, we will take a step that combines the memory and the current gradient, then update the memory.
- In addition to speeding up progress along valleys, this also helps to average out noisy estimates of the gradient, which we will see next.
- SGD (stochastic gradient descent): Computing the full gradient may be expensive (think empirical risk with billions of examples). Approximiating a gradient is still useful if it points in roughly the same direction.
- In SGD, we assume the loss is of the form
$$L(\theta) = \sum_{n=1}^{N} L_n(\theta)$$
where θ is the parameter of interest and L_n are the losses incurred by each example n. For example, if your model predicts a probability of a class, then
$$L(\theta) = -\sum_{n=1}^{N} \log p(y_n | x_n, \theta)$$
where x_n are the training inputs, y_n are the training targets, and θ are the model parameters.
- Gradient descent can be viewed as a batched method which does the following update:
$$\theta_{i+1} = \theta_i - \gamma_i \sum_{n=1}^{N} \nabla L_n(\theta)^T$$
- In SGD, we take a sum over a smaller set of L_n. Instead of using all n from $1 \dots N$, we can randomly select a single L_n to estimate the gradient:
$$\theta_{i+1} = \theta_i - \gamma_i \nabla L_n(\theta)^T, \quad n \sim \text{Uniform}[1, \dots, N]$$

- for gradient descent to converge, we only need the estimate of the gradient to be unbiased, i.e.
$$\mathbb{E}_{n,x,y}\nabla L_n(\theta; x, y)^T] = \mathbb{E}_{x,y}[\nabla L_n(\theta; x, y)^T]$$
- Instead of just one, we can also use any subsample of the data which is also unbiased.
- Larger subsamples have lower variance (remember the average goes down with $O(1/\sqrt{N})$), but smaller subsamples are quick to calculate. If your problem doesn't need an exact gradient, SGD can be faster.
- Why SGD? Minibatches are easier to fit in memory.

2 Constrained Optimization

Up to this point, we've considered optimization problems of the form $\min_x f(x)$. This is called unconstrained: the variable x is unrestricted. We can consider constrained problems, i.e.

$$\min_x f(x), \quad \text{subject to } g_i(x) \leq 0 \text{ for all } i = 1, \ldots, m$$

An obvious way to convert this to an unconstrained problem is to write

$$\min_x J(x) = \min_x f(x) + \sum_i 1(g_i(x))$$

where $1(g_i(x)) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$ is the infinite step function. This gives an infinite penalty of the constraint is not satisfied. THis is technically not constrained but is equally difficult to optimize (it is not differentiable). We will instead overcome this difficulty with a method call Lagrange multipliers.

- The idea is to introduce more variables to the problem called Lagrange multipliers $\lambda_i \geq 0$ for each inequality constraint to construct
$$\mathfrak{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) = f(x) + \lambda^T g(x)$$
which is called the lagrangian. This is also called the *primal* problem.
- The dual problem is
$$\max_\lambda \mathfrak{D}(\lambda) \quad \text{subject to} \quad \lambda \geq 0$$
where $\mathfrak{D}(\lambda) = \min_x \mathfrak{L}(x, \lambda)$.
- λ are the dual variables and x are the primal variables.
- We can sometimes solve for $\mathfrak{D}(\lambda)$ analytically by setting the gradient to zero, i.e.
$$\frac{\partial}{\partial x}\mathfrak{L}(x, \lambda) = 0$$

- Minimax inequality:

$$\max_y \min_x \varphi(x,y) \leq \min_x \max_y \varphi(x,y)$$

which can be proven by considering that the following holds for all x, y:

$$\min_x \varphi(x,y) \leq \max_y \varphi(x,y)$$

and so in particular, it holds for the maximum over y on the left and minimum over x on the right.

- The other concept is weak duality: When $\lambda \geq 0$, i.e. the dual variable is feasible, then the Langrangian is a lower bound on $J(x)$, i.e.

$$J(x) = \max_{\lambda \geq 0} \mathfrak{L}(x,\lambda)$$

but since the original problem was minimizing $J(x)$, we have

$$\min_x \max_{\lambda \geq 0} \mathfrak{L}(x,\lambda)$$

and by the minimax inequality, we can swap the order to get

$$\min_x \max_{\lambda \geq 0} \mathfrak{L}(x,\lambda) \geq \max_{\lambda \geq 0} \min_x \mathfrak{L}(x,\lambda)$$

This concept of dual feasible points always being upper bounded by primal feasible points is called weak duality. This equation can be re-expressed as

$$\min_x J(x) \geq \max_{\lambda \geq 0} \mathfrak{D}(\lambda)$$

- A problem satisfies strong duality if this is an equality instead:

$$\min_x J(x) = \max_{\lambda \geq 0} \mathfrak{D}(\lambda)$$

- What about equality constraints?

$$\min_x f(x), \quad \text{subject to } g_i(x) \leq 0 \text{ for all } i = 1, \ldots, m \quad h_j(x) = 0 \text{ for all } j = 1, \ldots, n$$

We can replace each equality constraint with two inequality constraints,

$$h_j(x) = 0 \Leftrightarrow h_j(x) \leq 0 \wedge h_j(x) \geq 0$$

Then, we have that

$$\mathfrak{L}(x, \lambda, \psi_-, \psi_+) = f(x) + \lambda^T g(x) + \psi_+^T h(x) - \psi_-^T h(x)$$

But the dual problem is

$$\max_{\lambda \geq 0, \psi_- \geq 0, \psi_+ \geq 0} \mathfrak{L}(x, \lambda, \psi_-, \psi_+) = \max_{\lambda \geq 0, \psi_- \geq 0, \psi_+ \geq 0} f(x) + \lambda^T g(x) + \psi_+^T h(x) - \psi_-^T h(x)$$

which can be re-written as

$$\max_{\lambda \geq 0, \psi} \mathfrak{L}(x, \lambda, \psi_-, \psi_+) = \max_{\lambda \geq 0, \psi} f(x) + \lambda^T g(x) + \psi^T h(x)$$

and so therefore, to handle inequality constraints we simply leave the dual variable unconstrained and our Lagrangian is

$$\mathfrak{L}(x, \lambda, \psi) = f(x) + \lambda^T g(x) + \psi^T h(x)$$

3 Convex Optimization

We'll now return to a special case of constrained optimization problems that have a nice property that we've seen before: convexity. Convex problems have a global minima and good convergence properties!

- A set $\mathcal{C}$ is convex if for any $x, y \in \mathcal{C}$, and any θ we have
$$\theta x(1-\theta)y \in \mathcal{C}$$
- And as a reminder, a convex function is f such that for all x, y in the domain of f, and any $0 \leq \theta \leq 1$, we have
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$
And for a multivariate function $f : \mathbb{R}^N \to \mathbb{R}$, f is convex if
$$f(y) \geq f(x) + \nabla_x f(x)^\top (y - x)$$
for all x, y or if $H = \nabla^2 f(x)$ is positive semidefinite, i.e. $z^T H z \geq 0 \forall z$.
- Then, a constrained optimization problem is called convex if all f, g_i, h_j are convex:
$$\min_x f(x), \quad \text{subject to } g_i(x) \leq 0 \text{ for all } i = 1, \ldots, m \quad h_j(x) = 0 \text{ for all } j = 1, \ldots, n$$
- Linear program: Take $f(x) = c^\top x$, $g(x) = Ax - b$. Then, we have
$$\min_x c^\top x \quad \text{subject to } Ax \leq b$$
- To get the dual, we can write down the Lagrangian and minimize over the primal variables:
$$\mathfrak{L}(x, \lambda) = c^\top x + \lambda^\top (Ax - b) = (c + A^\top \lambda)^\top x - \lambda^\top b$$
Setting the derivative to zero implies that
$$c + A^\top \lambda = 0$$
This leaves us with
$$\mathfrak{D}(\lambda) = -\lambda^\top b, \quad \text{subject to} \quad c + A^\top \lambda = 0, \lambda \geq 0$$
- The dual can be easier to solve, i.e. if the number of constraints is less than the number of variables in the primal.
- Quadratic program: take $f(x) = \frac{1}{2} x^\top Q x + c^\top x$, $g(x) = Ax - b$. Then, we have
$$\min_x \frac{1}{2} x^\top Q x \quad \text{subject to } Ax \leq b$$
- See textbook for the dual of the quadratic program.
- Convex problems satisfy strong duality, so one can solve the primal or the dual prpoblem to get the same solution:
$$\min_x J(x) = \max_{\lambda \geq 0} \mathfrak{D}(\lambda)$$

3.1 Legendre Transforms and the Convex Conjugate

Convex functions can be equivalently described by their supporting hyperplanes. It can be helpful to re-express convex functions as their supporting hyperplanes, a concept called the Legendre-Fenchel transform or the convex conjugate.

- The convex conjugate of f is
$$f^*(s) = \sup_x \langle s, x\rangle - f(x).$$
In general, we'll use the standard dot product between finite-dimensional vectors, $\langle s, x\rangle = s^\top x$
- Intutiively, consider any hyperplane $y = sx + c$. Let's try to use this function as a supporting hyperplane to describe f. In particular, consider the smallest c such that $y = sx + c$ still touches f.
For any x_0, the line going through $(x_0, f(x_0))$ with slope s is
$$y - f(x_0) = s(x - x_0)$$
This has a y intercept of $c = -sx_0 + f(x_0)$ (can see this by plugging in $x = 0$). Therefore, the smallest c is $\min_{x_0} -sx_0 + f(x_0) = -f^*(s)$
- The conjugate by convention is the negative of this minimum intercept.
- The conjugate function has nice properties:
  - The Legendre transform twice gets the orginal function, i.e. $(f^*)^* = f$
  - When the slope of $f(x)$ is s, the slope of $f^*(s)$ is x.
  - Example 7.8: Let $L(t) = \sum_i \ell_i(t_i)$. Then,
$$L^*(z) = \sup_t z^\top t - \sum_i \ell_i(t_i) = \sup_t \sum_i z_i t_i - \ell_i(t_i) = \sum_i \ell_i^*(z_i)$$
- Conjugates occur frequently in Lagrange multipliers.
- For example consider a linearly constrained sum of two convex functions:
$$\min_x f(Ax) + g(x) = \min_{Ax=y} f(y) + g(x)$$
$$\min_{Ax=y} f(y) + g(x) = \min_{x,y} \max_u f(y) + g(x) + (Ax - y)^\top u$$
But since f, g are convex, we have strong duality so
$$\min_{Ax=y} f(y) + g(x) = \max_u \min_{x,y} f(y) + g(x) + (Ax - y)^\top u$$
Splitting up x and y, we get
$$\max_u [\min_y (f(y) - y^\top u) + \min_x (g(x) + x^\top A^\top u)] = \max_u -f^*(u) - g^*(-A^\top u)$$
Thus, if we know the convex conjugate of f and g, then we can easily derive the equivalent dual problem
- Why do we use the dual problem? Sometimes it is easier to solve. For example, in the last example, we could now solve it with gradient ascent (there are no constraints in the dual problem).