Note 9: Optimization of Non-Convex Functions

Math 198: Math for Machine Learning

1 Taylor's Theorem

Taylor's theorem is a result from one-dimensional calculus which states that k-times continuously differentiable functions can be approximated around a point a by a polynomial h_a of degree k. While we will not approximate functions by Taylor polynomials in this class, we will use this theorem today to prove important results about minima which will help us solve non-convex optimization problems. Suppose f is continuously differentiable, and consider some $\mathbf{h} \in \mathbb{R}^n$. Then there exists some $t \in (0, 1)$ such that

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \nabla f(\mathbf{x} + t\mathbf{h})^\top \mathbf{h}$$

Additionally, if f is twice continuously differentiable, then

$$\nabla f(\mathbf{x} + \mathbf{h}) = \nabla f(\mathbf{x}) + \int_0^1 \nabla^2 f(\mathbf{x} + t\mathbf{h})\mathbf{h} \mathrm{d}t$$

and there exists a (possibly different) $t \in (0, 1)$ such that

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \nabla^2 f(\mathbf{x} + t\mathbf{h})\mathbf{h}$$

Further approximations can be made using higher derivatives of f; we will only use the first- and second-order approximations in this class.

2 Finding Critical Points with Taylor's Theorem

We can use Taylor's theorem to show that $\nabla f(\mathbf{x}) = \mathbf{0}$ if $\mathbf{x}$ is a local minimum of f, which we stated without proof in note 7. Suppose $\mathbf{x}^*$ is such a local minimum, but $\nabla f(\mathbf{x}) \neq \mathbf{0}$. Let $\mathbf{h} = -\nabla f(\mathbf{x}^*)$. Then since f is continuously differentiable, we have

$$\lim_{n \to 0} -\nabla f(\mathbf{x}^* + n\mathbf{h}) = -\nabla f(\mathbf{x}^*) = \mathbf{h}$$

and so

$$\lim_{n \to 0} \mathbf{h}^\top \nabla f(\mathbf{x}^* + n\mathbf{h}) = \mathbf{h}^\top \nabla f(\mathbf{x}^*) = -\mathbf{h}^\top \mathbf{h} < 0$$

There must be some point $N > 0$ at which this function first becomes negative, such that $\mathbf{h}^\top \nabla f(\mathbf{x}^* + n\mathbf{h}) < 0$ for all $n \in [0, N]$. By Taylor's theorem, we can then say that for any $n \in (0, N]$, there exists $t \in (0, 1)$ such that

$$f(\mathbf{x}^* + n\mathbf{h}) = f(\mathbf{x}^*) + n\mathbf{h}^\top \nabla f(\mathbf{x}^* + nt\mathbf{h})$$

But $nt < N$ and $n > 0$, so $n\mathbf{h}^\top \nabla f(\mathbf{x}^* + nt\mathbf{h}) < 0$ and thus $f(\mathbf{x}^* + n\mathbf{h}) < f(\mathbf{x}^*)$, a contradiction as we assumed $\mathbf{x}^*$ was a local minimum.

From this proof, we see that as long as $\nabla f(\mathbf{x})$ is non-zero, there always exists a small step $\alpha > 0$ which we can take such that $f(\mathbf{x} - \alpha \nabla f(\mathbf{x}) < f(\mathbf{x})$. Therefore, $-\nabla f(\mathbf{x})$ points in the direction of steepest descent, a property we used when considering the gradient descent algorithm.

3 Finding Minima with Taylor's Theorem

If $\mathbf{x}^*$ is a local minimum of f, then $f(\mathbf{x}^*) = \mathbf{0}$, but the reverse is not always true; the gradient will be 0 at any critical point of f, including saddle points and local maxima. However, using the Hessian, we can establish that a critical point is a local minimum. Given some critical point $\mathbf{x}^*$ and a function f which is twice continuously differentiable in a neighborhood of $\mathbf{x}^*$, then $\mathbf{x}^*$ is a local minimum of f if and only if $\nabla^2 f$ is positive semi-definite in a neighborhood of $\mathbf{x}^*$.

We start by proving this condition is necessary for $\mathbf{x}^*$ to be a minimum. Suppose it is, and that $\nabla^2 f(\mathbf{x}^*)$ is not positive semi-definite. Therefore there exists some $\mathbf{h}$ such that $\mathbf{h}^\top \nabla^2 f(\mathbf{x}^*)\mathbf{h} < 0$, and so by the continuity of $\nabla^2 f$, we have

$$\lim_{t \to 0} \mathbf{h}^\top \nabla^2 f(\mathbf{x}^* + t\mathbf{h}) = \nabla^2 f(\mathbf{x}^*)$$

which implies

$$\lim_{t \to 0} \mathbf{h}^\top \nabla^2 f(\mathbf{x}^* + t\mathbf{h})\mathbf{h} = \mathbf{h}^\top \nabla^2 f(\mathbf{x}^*)\mathbf{h} < 0$$

Therefore there exists $T > 0$ such that $\mathbf{h}^\top \nabla^2 f(\mathbf{x}^* + t\mathbf{h})\mathbf{h} < 0$ for all $t \in [0, T]$. Then for any $t \in (0, T]$, by Taylor's theorem there exists $t' \in (0, t)$ such that

$$f(\mathbf{x}^* + t\mathbf{h}) = f(\mathbf{x}^*) + t\mathbf{h}^\top \nabla f(\mathbf{x}^*) + \frac{1}{2} t^2 \mathbf{h}^\top \nabla^2 f(\mathbf{x}^* + t'\mathbf{h})\mathbf{h}$$

Since $\nabla f(\mathbf{x}^*) = 0$, this simplifies and we have

$$f(\mathbf{x}^* + t\mathbf{h}) = f(\mathbf{x}^*) + \frac{1}{2} \mathbf{h}^\top \nabla^2 f(\mathbf{x}^* + t'\mathbf{h})\mathbf{h} < f(\mathbf{x}^*)$$

which contradicts our premise that $\mathbf{x}^*$ is a local minimum.
Proving that this condition is sufficient is more complicated. Let B be an open ball of radius $r > 0$ centered at $\mathbf{x}^*$ which is contained in the neighborhood of $\mathbf{x}^*$. Then for any $\mathbf{h}$ with $||\mathbf{h}||_2 < r$, by Taylor's theorem there exists $t \in (0, 1)$ such that

$$f(\mathbf{x}^* + \mathbf{h}) = f(\mathbf{x}^*) + \mathbf{h}^\top f(\mathbf{x}^*) + \frac{1}{2} \nabla^2 f(\mathbf{x}^* + t\mathbf{h})\mathbf{h} = f(\mathbf{x}^*) + \frac{1}{2} \nabla^2 f(\mathbf{x}^* + t\mathbf{h})\mathbf{h}$$

Since $\nabla^2 f$ is positive semi-definite in the neighborhood of $\mathbf{x}^*$, and since B is contained in this neighborhood, $\nabla^2 f$ is positive semi-definite for all points $\mathbf{x}^* + t\mathbf{h}$. Therefore $f(\mathbf{x}^* + \mathbf{h}) \geq f(\mathbf{x}^*)$ for all $\mathbf{h}$, and so $\mathbf{x}^*$ is a local minimum. It follows that if $\nabla^2 f$ is positive definite, $\mathbf{x}^*$ is a unique local minimum.

Why do we require $\nabla^2 f$ to be positive semi-definite in the neighborhood of $\mathbf{x}^*$? Consider the function $f(x) = x^3$. 0 is not a minimum of f, but $f'(0) = f''(0) = 0$. $f''(x) = 6x$, which is negative for all $x < 0$, but non-negative at $x = 0$. So instead of getting a minimum, we get a saddle point.

Application: Newton's Method

Newton's method is usually first encountered in calculus as a method for finding the roots of functions (the points x for which, given a function f, $f(x) = 0$). However, we can also use Newton's method in an optimization context for a twice-differentiable f, searching instead for the roots of f', which correspond to minima of f. We start with an initial guess x_0, from which Newton's method will produce a series $x_1, \ldots, x_n$ which converges towards a minimum x^*.

Newton's method differs from gradient descent in that it also uses information about the function's curvature from the second derivative (or in the multivariate case, the Hessian). Therefore, it generally requires fewer steps to converge than gradient descent. That said, for some functions (particularly those with many parameters. such as in the case of fitting neural networks) the Hessian can be expensive to calculate, and so Newton's method is not necessarily always preferred. Additionally, Newton's method requires the Hessian to be positive-definite and invertible, conditions which cannot always be satisfied in practice.

We will first discuss the single-variable case; the multivariate case follows directly. At each iteration, Newton's method proceeds by constructing a second-order Taylor approximation of f around x_k:

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f'(x_k)t^2$$

If $f''(x_k)$ is positive, this second-order approximation is a convex function in t; we seek to minimize it and set $x_{k+1} = x_k + t^*$. Let $p(t) = f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2$; then $p'(t) = f'(x_k) + f''(x_k)t$ and so

$$t^* = -\frac{f'(x_k)}{f''(x_k)}$$

From this we derive our iterative step

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

It follows that in the multivariate case, the equivalent iterative step is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

Again, this requires the Hessian to be invertible and positive-definite (or else the second-order Taylor approximation will not be convex). Newton's method is therefore heavily dependent on the initial guess x_0; picking a point near a maximum or saddle point will cause the algorithm to not converge.

Application: Gauss-Newton Algorithm

We have previously encountered ordinary least squares, in which, given n observations of an m-dimensional vector of labels $\mathbf{x}_i$ and an output value y_i, we sought an m-dimensional vector $\mathbf{w}$ which minimized the loss function

$$L(\mathbf{w}) = \sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x_i})^2$$

We can generalize this method to optimize models which are not merely linear combinations of their inputs. Suppose we have some function $f(\mathbf{x}; \beta)$ which is parameterized by an m-dimensional parameter vector β. In particular, we require $m \leq n$, that is, there are at least as many observations as parameters. Then we can define the loss function as

$$L(\beta) = \sum_{i=1}^{n}(y_i - f(\mathbf{x_i}; \beta))^2 = \sum_{i=1}^{n} r_i(\beta)^2$$

where r_i is referred to as the i-th *residual* given β. Let $\mathbf{r}(\beta)$ be the vector of these residuals; then we redefine

$$L(\beta) = ||\mathbf{r}(\beta)||_2^2$$

Note that $\mathbf{r}(\cdot)$ is a function which maps $\mathbb{R}^m \to \mathbb{R}^n$, and so its Jacobian is an $n \times m$ matrix

$$\mathbf{J_r}(\beta) = \begin{bmatrix} \nabla r_1(\beta)^\top \\ \vdots \\ \nabla r_n(\beta)^\top \end{bmatrix}$$

which we can use to rewrite the gradient and Hessian of L as[1]

$$\nabla L(\beta) = 2\mathbf{J_r}^\top(\beta)\mathbf{r}(\beta)$$

$$\nabla^2 L(\beta) = 2(\mathbf{J_r}(\beta)^\top \mathbf{J_r}(\beta) + \sum_{i=1}^{n} r_i(\beta)\nabla^2 r_i(\beta))$$

Why is any of this useful? Recall that the Hessian, $\nabla^2 L(\beta)$, is often expensive to compute. However, as we get close to an optimal point β^*, the second term in the Hessian of L becomes much smaller than the first term,[2] and so we can approximate $\nabla^2 L(\beta) \approx \mathbf{J_r}(\beta)^\top \mathbf{J_r}(\beta)$. (Note that this approximation is a characteristic of nonlinear least-squares regression specifically, and not of nonlinear optimization generally.) We can use this approximation to enable the Gauss-Newton algorithm, which uses Newton's method to optimize β with update rule

$$\beta_{k+1} = \beta_k - (\mathbf{J_r}(\beta_k)^\top \mathbf{J_r}(\beta_k))^{-1}\mathbf{J_r}^\top(\beta_k)\mathbf{r}(\beta_k)$$

As with Newton's method, this algorithm is heavily dependent on a good starting value β_0.

[1] Proof omitted.

[2] Either because the loss function "smooths out", or because the residuals become small, or both.