Chapter 8

Gradient Descent and Convexity

In the previous chapter, we introduced the concept of empirical risk minimization. The key idea being that we define predictive models by minimizing loss functions on samples of training data. In this chapter we will introduce the most fundamental and popular approach for performing this optimization: the gradient descent algorithm. We saw a variant of this idea in Chapter 6, where we discussed a gradient ascent approach to maximize a logistic regression model. This chapter will generalize that approach and discuss some of its theoretical properties.

8.1 Gradient Descent as a General Approach

We introduced gradient ascent in Chapter 6 as a way to find the parameters that maximized the log-likelihood of the data, in a case where an exact closed-form solution was intractable. In fact, gradient ascent/descent is a general strategy that can be used to maximize/minimize an arbitrary function. Without loss of generality, we will use the term gradient descent in what follows and discuss the minimization problem, since this fits with the context of empirical risk minimization. Note also that we will focus on the use of gradient descent in empirical risk minimization, but the gradient descent technique is appropriate for a wide-range of optimization problems, including many outside the scope of machine learning.

Suppose we want to minimize some empirical risk function $R(\mathbf{w}) : \mathbb{R}^m \to \mathbb{R}$ with respect to some parameter vector $\mathbf{w} \in \mathbb{R}^m$.[1] The gradient descent approach to minimization involves starting with an initial guess for the parameter—e.g.,

[1] Note that for simplicity we assume that the parameter set Θ is defined by a single parameter vector.

$\mathbf{w}^{(0)} = \mathbf{0}$—and iteratively refines the estimate

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} - \alpha^{(k)} \nabla_{\mathbf{w}} R(\mathbf{w}^{(k-1)}) \tag{8.1}$$

until we hit a stopping criterion

$$\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\| < \epsilon. \tag{8.2}$$

Note that we add a superscript to the learning rate α to indicate that it can in principle change over time.

8.2 Convergence, Smoothness, and Convexity

Two natural questions for the gradient descent approach are:

1. When will the algorithm converge?

2. When can we guarantee that the model will converge to a "good" solution?

A trivial answer to the first question is that we can guarantee convergence as long as the learning rate $\alpha^{(k)} \to 0$ decays to zero in a finite number of steps. However, as we will see, we can guarantee convergence in a more meaningful way using the notions of *smoothness* and *convexity*.

8.2.1 Smoothness and convergence

In general, we can guarantee that gradient descent will converge to a point where the gradient is zero in a finite number of iterations as long as the function R that we are optimizing is *smooth*. In other words, we can guarantee that gradient descent will converge to a *critical point* of the function R.

Differentiablity

First and foremost, we require that the function we are optimizing is smooth enough so that it is twice differentiable. Otherwise, we cannot guarantee that we can even run gradient descent (which requires first derivatives) or test whether we are at an maximum or minimum (which requires second derivatives). In formal terms, for a function $R : \mathbb{R}^m \to \mathbb{R}$, we must be able to compute

$$\nabla R(\mathbf{w}) = \left[\frac{\partial R(\mathbf{w})}{\mathbf{w}[0]}, \frac{\partial R(\mathbf{w})}{\mathbf{w}[1]}, ..., \frac{\partial R(\mathbf{w})}{\mathbf{w}[m-1]} \right]^\top, \forall \mathbf{w} \in \mathbb{R}, \tag{8.3}$$

where $\nabla R(\mathbf{w}) \in \mathbb{R}^m$ is known as the *Jacobian* and gives the gradient for each input dimension at a point $\mathbf{w}$. In addition, we must be able to compute the

Hessian

$$\nabla^2 R(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[0]^2} & \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[0]\,\partial \mathbf{w}[1]} & \cdots & \frac{\partial^2 f}{\partial \mathbf{w}[0]\,\partial \mathbf{w}[m-1]} \\ \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[1]\,\partial \mathbf{w}[0]} & \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[1]^2} & \cdots & \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[1]\,\partial \mathbf{w}[m-1]} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[m-1]\,\partial \mathbf{w}[0]} & \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[m-1]\,\partial \mathbf{w}[1]} & \cdots & \frac{\partial^2 R(\mathbf{w})}{\partial \mathbf{w}[m-1]^2} \end{bmatrix}, \forall \mathbf{w} \in \mathbb{R}, \tag{8.4}$$

which contains the second derivatives for all pairs of input dimensions. Being able to define the Jacobian and Hessian at all input points is—in some sense—a basic prerequisite for doing any kind of meaningful optimization. We have to be able to differentiate our input function and test for optima. However, we also need a stronger form of smoothness—in addition to differentiability—to ensure that gradient descent converges.

Lipschitz continuity

We can formalize the additional smoothness we require via the notion of *Lipschitz continuity.*

Definition 2. *A function $R : \mathbb{R}^m \to \mathbb{R}^n$ is said to be Lipschitz continuous with constant L if and only if*

$$\|R(\mathbf{x}) - R(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m. \tag{8.5}$$

Informally, Lipschitz continuity bounds how much a function can vary between two nearby inputs, which provides a notion of smoothness. Visually, Lipschtiz continuity requires that one can place a double cone—with slopes defined by L—at any point on the function and guarantee that no other point on the function lies outside this cone (Figure 8.1). Lipschitz continuity is impor-

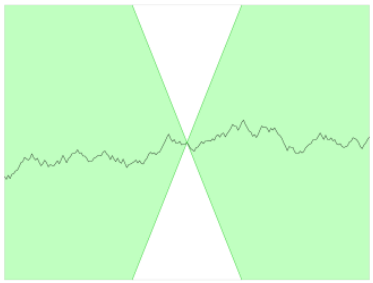

Figure 8.1: Illustration of the double-cone defined by the Lipschitz constant L for a function.

tant because the constant L tells us how smooth a function is (i.e., how fast it can vary between nearby points). If a function is not Lipschitz continuous at all (i.e., no finite L exists), then the change in function value between two nearby points can be unbounded (i.e., infinite). It is difficult—if not impossible—to optimize such non-smooth functions.

Gradient descent converges for smooth functions

Using the notion of Lipschitz continuity, we can characterize the cases in which gradient descent will converge.

Theorem 2. *Assume that the gradient $\nabla R : \mathbb{R}^m \to \mathbb{R}$ is Lipschitz continuous with constant L (which implies that R is twice differentiable), and assume that there exists at least one point $\mathbf{w}^* \in \mathbb{R}^m$ such that $\nabla R(\mathbf{w}^*) = 0$. Then the gradient descent algorithm is guaranteed to converge in a finite number of iterations K to some point $\mathbf{w}^{(K)}$ with $\nabla R(\mathbf{w}^{(K)}) = 0$ if we use a constant step size of $\alpha = \frac{1}{L}$.*

Proof. **Note that this proof is included for completeness and interest only; it is not examinable material.** We have that the Lipschitz continuity of the gradient is equivalent to the requirement that

$$\nabla^2 R(\mathbf{w}) \preccurlyeq L\mathbf{I}, \forall \mathbf{w} \in \mathbb{R}^m, \tag{8.6}$$

where $\preccurlyeq$ denotes an element-wise inequality relation and $\mathbf{I}$ is the identity matrix. In formal terms, Lipschitz continuity of the gradient implies that the Hessian matrix (which contains the second derivatives) has eigenvalues bounded above by L. This further implies that

$$\mathbf{u}^\top \nabla^2 R((w))\mathbf{u} \leq L\|\mathbf{u}\|^2, \forall \mathbf{w}, \mathbf{u} \in \mathbb{R}^m. \tag{8.7}$$

Now, since f is twice differentiable, we can use a multivariate variation of Taylor's Theorem to show that

$$R(\mathbf{u}) = R(\mathbf{w}) + \nabla R(\mathbf{w})^\top (\mathbf{u} - \mathbf{w}) + \frac{1}{2}(\mathbf{u} - \mathbf{w})^\top \nabla^2 R((v))(\mathbf{u} - \mathbf{w}) \tag{8.8}$$

for some $\mathbf{v} = \beta\mathbf{w} + (1 - \beta)\mathbf{u}, \beta \in (0, 1)$. And combining this with Equation 8.7 we have that

$$R(\mathbf{u}) \leq R(\mathbf{w}) + \nabla R(\mathbf{w})^\top (\mathbf{u} - \mathbf{w}) + \frac{L}{2}\|\mathbf{u} - \mathbf{w}\|^2 \tag{8.9}$$

Equation 8.9 is sometimes known as the *descent lemma*.

Let us know consider a step of gradient descent with a step-size of $\alpha = \frac{1}{L}$. We have that

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \frac{1}{L}\nabla R(\mathbf{w}^{(k)}) \tag{8.10}$$

$$\Leftrightarrow$$

$$\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)} = -\frac{1}{L}\nabla R(\mathbf{w}^{(k)}). \tag{8.11}$$

Taking this fact and plugging $\mathbf{w}^{(k)}$ and $\mathbf{w}^{(k+1)}$ into Equation 8.9, we have that

$$R(\mathbf{w}^{(k+1)}) \leq R(\mathbf{w}^{(k)}) + \nabla R(\mathbf{w}^{(k)})^\top (\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}) + \frac{L}{2}\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\|^2 \tag{8.12}$$

$$= R(\mathbf{w}^{(k)}) - \frac{1}{L}\nabla R(\mathbf{w}^{(k)})^\top \nabla R(\mathbf{w}^{(k)}) + \frac{L}{2}\|\frac{1}{L}\nabla R(\mathbf{w}^{(k)})\|^2 \tag{8.13}$$

$$= R(\mathbf{w}^{(k)}) - \frac{L}{2}\|\nabla R(\mathbf{w}^{(k)})\|^2. \tag{8.14}$$

The proof concludes by noting that $\|\nabla R(\mathbf{w}^{(k)})\|^2 > 0$ unless $\nabla R(\mathbf{w}^{(k)}) = 0$. This implies that we will always make finite progress unless we are at a critical point, which in turn gives a guarantee on convergence to a critical point in a finite number of iterations. □

Practical learning rates We can guarantee convergence with $\alpha = \frac{1}{L}$, but this convergence may be very slow and estimating L can be expensive (or impossible). In practice, researchers often just try a few reasonably large constant step-sizes (e.g., $\alpha = \{0.01, 0.1, 0.5\}$) and simply use the result that converges to the lowest objective value fastest.

There are also more practical but theoretically grounded approaches, where we try to find step sizes at each iteration that are large enough, while still guaranteeing some progression. For example, one classic approach is called *backtracking line search with the Armijo condition.* In this approach, we start every iteration k of gradient descent by guessing a reasonable and large step-size $\tilde{\alpha}^{(k)} = \alpha_0$ (with $\alpha_0 \in (0, 1]$). Next, we check whether or not the *Armijo* condition is satisfied for this candidate step size:

$$R(\mathbf{w}^{(k)} - \tilde{\alpha}^{(k)}\nabla R(\mathbf{w}^{(k)})) \leq R(\mathbf{w}^{(k)}) - \tilde{\alpha}^{(k)}\gamma\|\nabla R(\mathbf{w}^{(k)})\|^2, \tag{8.15}$$

where $\gamma \in (0, 1)$ is a hyperparameter. If this condition is satisfied, then we accept the candidate step-size $\alpha^{(k)} = \tilde{\alpha}^{(k)}$ and do the gradient update. Otherwise, we decrease our guess by a fixed rate by setting

$$\tilde{\alpha}^{(k)} = \beta\tilde{\alpha}^{(k)} \tag{8.16}$$

for some hyperparameter $\beta \in (0, 1)$ and check the Armijo condition (Equation 8.15) again. The motivation behind the Armijo rule is that it ensures that our step-size decreases the objective by a large enough amount. The γ parameter controls the tolerance for our search. For instance, we set γ very small, then we will likely find an acceptable step-size faster, at the cost of not decreasing the objective as much. Armijo line search is one simple example of a backtracking line search approach, but there are many more complex variants of this idea, as well as other more advanced step-size selection techniques.

8.2.2 Convexity and uniqueness

So, we can guarantee that gradient descent will converge to a point $\nabla R(\mathbf{w}^{(k)}) = 0$ where the gradient of the empirical risk is zero. However, how can we guarantee that this a good solution (e.g., that it is a global minimum)? Generally, we can only guarantee convergence to a unique, global minimum in cases where a function is *convex*. Convexity formalizes the mathematical requirement that a function only has one minimum. Figure 8.2 illustrates the intuition behind convexity for a one-dimensional function. Convex functions have a single minimizer.

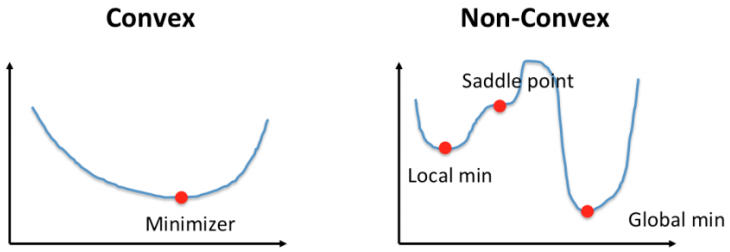

Figure 8.2: Illustration of convex and non-convex functions.

On the other hand, non-convex functions may have multiple *local minima* as well as *saddle points* that are not local minima but where the gradient is still zero. Figure 8.3 illustrates a convex and non-convex functions in a higher dimension.

Geometrically, a function is convex if and only if a line segment joining two points on the function always lies above the two points (Figure 8.4). In terms of proving whether a function is convex, a practical methods relies on the second derivative, i.e., the Hessian matrix. The Hessian matrix of a convex function will be positive semi-definite at all points, i.e.,

$$\mathbf{x}^\top \nabla^2 R(\mathbf{w})\mathbf{x} \geq 0, \forall \mathbf{x}, \mathbf{w} \in \mathbb{R}^m. \tag{8.17}$$

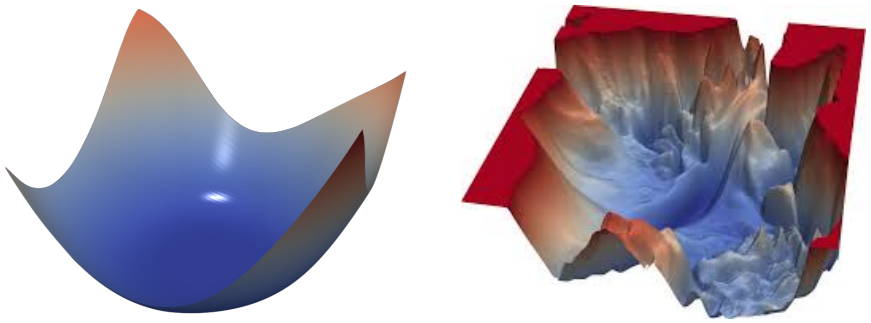

Figure 8.3: Illustration of convex (left) and non-convex (right) functions in two dimension.

Thus, a common way to prove convexity is by demonstrating that the Hessian is positive semi-definite at all points. Note that in the case of a univariate function this reduces to the requirement that the second derivative is non-negative everywhere.

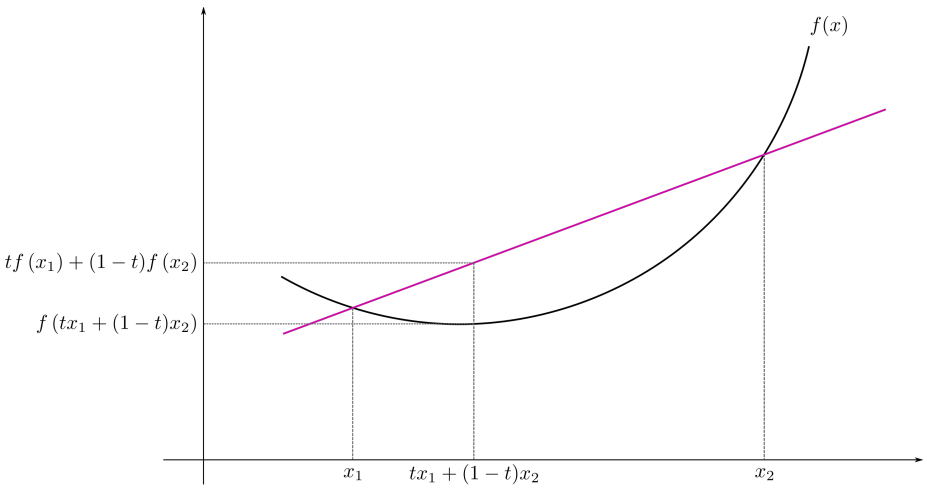

Figure 8.4: Definition of convexity based on an intersecting line segment.[2]

Convexity, risk, models, and loss functions

So far, we have been focusing on using gradient descent to minimize the empirical risk

$$R(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{trn}}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{trn}}} L(y_i, f_{\mathbf{w}}(\mathbf{x}_i)), \tag{8.18}$$

where L is a loss function and $f_{\mathbf{w}}$ is our prediction function (i.e., our model). An important point to note is that the smoothness and convexity of the empirical risk function R entirely depends on the definitions of the model and the loss. In fact, it is easy to prove that the sum of a set of convex functions is convex. Thus, as long as the composition of the model and loss function, i.e., $L(y_i, f_{\mathbf{w}}(\mathbf{x}_i))$, on a single example is convex and smooth, then we can guarantee that the empirical risk will also be convex and smooth. For this reason, you will often hear researchers discuss whether models or loss functions are convex, rather than discussing whether the empirical risk is convex. In this course—for the sake of clarity—when we refer to a model being convex, we will mean the composition of the prediction model and a loss function (e.g., the logistic regression model). Note that a loss function must be convex

[2]Image credit: Eli Osherovich - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10764763

Examples covered so far

So far in the course, we have encountered convex and smooth functions, convex and non-smooth functions, and even functions that are neither convex nor smooth. The logistic regression model—for example—is both convex and smooth, which justifies our use of gradient descent to optimize it. The hinge-loss discussed in Chapter 7 is also convex when combined with a linear model (e.g., as in the perceptron), but it is not smooth (since it relies on the sign function).[3] Lastly, the 0-1 loss—also discussed in Chapter 7—is neither convex nor smooth, which explains why it is difficult to optimize and not used in practice.

8.3 Variants of Gradient Descent

Recall that the empirical risk function R averages the loss over the entire training set. As a consequence, one limitation of naively running gradient descent on the risk function R is that we must process the entire dataset at each iteration. This approach is often called *full-batch gradient descent* and it can be quite expensive in practice. As an alternative, it is common to use *stochastic mini-batch gradient descent*, where we compute the gradient using only a sample of the training set at each iteration. The stochastic mini-batch algorithm is given below:

Algorithm 2: Stochastic Minibatch Gradient Descent

input : training set $\mathcal{D}_{\text{trn}}$; model $f_{\mathbf{w}}$; loss L; max iterations K; batch-size B; tolerance ϵ; learning rates $\alpha^{(k)}, k = 0, ..., K-1$

output: Model parameters $\Theta = \{\mathbf{w}\}$

$\mathbf{w}^{(0)}$ = Random initialization

for $k = 0$ **to** $K-1$ **do**

  $\mathcal{D}_{\text{batch}}$ = Random sample of B points from $\mathcal{D}_{\text{trn}}$

  $R_{\text{batch}}(\mathbf{w}^{(k)}) = \frac{1}{|\mathcal{D}_{\text{batch}}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{batch}}} L(y_i, f_{\mathbf{w}}(\mathbf{x}_i))$

  $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} R_{\text{batch}}(\mathbf{w}^{(k)})$

  if $\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\| \leq \epsilon$ **then**

    break;

  end

end

Return $\mathbf{w}^{(k)}$

This approach is called stochastic due to the fact that we randomly subsample a batch of data at each iteration. In the special case where the batch-size is equal to one, then this approach is usually referred to simply as stochastic gradient descent.

From a theoretical perspective, stochastic (mini-batch) gradient descent—often abbreviated SGD—optimizes a stochastic estimate of the risk at each iteration. SGD is chosen over full-batch gradient descent in practice because it is less memory intensive and often results in much faster convergence. SGD is

[3] Interestingly, despite the non-smooth nature of the hinge-loss, it can still be optimized via a variant of gradient descent—called subgradient descent—but that approach is outside the scope of this course.

guaranteed to converge for smooth and convex models as long as learning rates satisfy the *Robbins-Munroe* conditions:

$$\sum_{k=0}^{\infty} \alpha^{(k)} = \infty \qquad \sum_{k=0}^{\infty} (\alpha^{(k)})^2 < \infty. \tag{8.19}$$

The proof of this is outside the scope of this course but can be found in most textbooks on optimization. In recent years, researchers have also found that SGD tends to find good solutions when optimizing non-convex models in the context of *deep learning*—a topic that will be discussed in Chapter 21.