ECE586 MDPs and Reinforcement Learning **University of Illinois at Urbana-Champaign**
Spring 2019 **Acknowledgment: R. Srikant's Notes and Discussions**

Lecture 3: Steepest and Gradient Descent-Part I

Instructor: Dimitrios Katselis *Scribe: Duc Phan, Alireza Moradzadeh*

In this lecture, we introduce the steepest and gradient descent algorithms as numerical schemes for solving the problem

$$\min_{x\in\mathbb{R}^n} f(x),$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is a continuously differentiable (loss) function. Methods for choosing the sequence of stepsizes are discussed and certain convergence results are provided.

3.1 Steepest and Gradient Descent Algorithms

Given a continuously diffentiable (loss) function $f:\mathbb{R}^n\to\mathbb{R}$, **steepest descent** is an iterative procedure to find a local minimum of f by moving in the opposite direction of the gradient of f at every iteration k. Steepest descent is summarized in Algorithm 3.1. ε_k is the stepsize parameter at iteration k. ε_k is sometimes called **learning rate** in the context of machine learning. By our discussion in a previous file, the stepsize ε_k or learning rate specifies how aggressively an iterative algorithm proceeds in each iteration. If the learning schedule $\varepsilon_0,\varepsilon_1,\varepsilon_2,\ldots$ consists of very small values, convergence to a (local) minimum is slow. If the learning rate is high, then the algorithm may be unstable, bouncing around the (local) minimum (i.e., oscillating) or even overshooting the (local) minimum and possibly diverging. Considering constant learning rate algorithms, the steepest descent algorithm with $\varepsilon_k=\varepsilon>0,\forall k\geq 0$ enjoys nice convergence guarantees if the learning rate is tuned according to the scale of smoothness of the gradient function $\nabla f(x)$. In practice, one often uses iteration-dependent learning rate ε_k by employing line search methods, as we will see shortly.

A note about the learning rate in the context of stochastic gradient descent (SGD): Our main target algorithm in this course is SGD, which is the most commonly used optimization algorithm in deep learning, as well as in many other large-scale optimization problems. Mean or deterministic counterparts of this algorithm correspond to steepest and gradient descent. To see this, recall the standard setting of SGD: Instead of observing a full gradient $\nabla f(x_k)$ at iteration k, one observes a stochastic gradient g_k satisfying $E[g_k]=\nabla f(x_k)$ (unbiased gradient estimator) and $E[\|g_k\|^2]\leq C$ (bounded second moment). In practice, learning rate adaptation requires tuning. If certain features associated with the structure of the loss function f, such as Lipschitz and strong convexity parameters are known, learning rate schedules supported by existing theoretical analysis can be applied. However, in practice, such parameters are not known and the loss function of interest is often not convex. These observations motivate **learning the learning rate** in the machine learning literature. The corresponding procedure is often structured as follows: the learning rate is initialized at a high value (being aggressive in the beginning) and is subsequently decreased according to gradient observations (to ensure stability).

Steepest descent is a special case of **gradient descent**, which is described in Algorithm 3.2. In gradient descent, the direction of motion d_k obeys the inequality

$$d_k^T(-\nabla f(x_k))>0 \text{ or } \nabla f(x_k)^T d_k<0,\ \forall k\geq 0. \tag{3.1}$$

In other words, d_k has a positive projection in the direction $-\nabla f(x_k)$. To illuminate a bit more this condition, we note that a natural iterative descent approach to minimize a real-valued loss function f over some set $\mathcal{X}$ relies on

cost improvement: starting at some $x_0 \in \mathcal{X}$, the goal of the iterative scheme is to construct a sequence of iterates $\{x_k\} \subset \mathcal{X}$ such that

$$f(x_{k+1}) < f(x_k), \quad \forall k \geq 0,$$

unless x_k is optimal in which case the algorithm terminates. It is now useful to consider the directional derivative of f at a point x in a direction d, defined as follows:

$$\nabla_d f(x) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon d) - f(x)}{\varepsilon}.$$

For a differentiable f:

$$\nabla_d f(x) = \nabla f(x)^T d.$$

From this formula it follows that if d_k is a descent direction at x_k in the sense that

$$\nabla f(x_k)^T d_k < 0,$$

then we may reduce f by moving from x_k along d_k with a sufficiently small positive stepsize ε. In the unconstrained case where $\mathcal{X} = \mathbb{R}^n$, this leads to the gradient descent scheme summarized in Algorithm 3.2, where d_k is a descent direction at x_k and ε_k is a positive scalar stepsize.

If $d_k = -\nabla f(x_k)$, then the gradient descent algorithm becomes the steepest descent. The previous discussion about the stepsize sequence $\{\varepsilon_k\}$ and its role to the stability of the algorithm applies in this (generalized) setup as well.

Note: In current optimization and machine learning literature, many authors use the term gradient descent to refer to steepest descent.

Algorithm 3.1 Steepest descent

Initialization : $x_0 \in \mathbb{R}^n$
for $k = 0, 1, 2, ...$ **do**
  $x_{k+1} \leftarrow x_k - \varepsilon_k \nabla f(x_k)$
end for

Algorithm 3.2 Gradient descent

Initialization : $x_0 \in \mathbb{R}^n$
for $k = 0, 1, 2, ...$ **do**
  $x_{k+1} \leftarrow x_k + \varepsilon_k d_k$
end for

3.2 Methods for Choosing Stepsizes

In this section, we discuss three popular strategies to select stepsizes for the introduced descent schemes.

3.2.1 Fixed Stepsize

The most natural choice of stepsizes corresponds to

$$\varepsilon_k = \varepsilon, \quad \forall k \geq 0, \tag{3.2}$$

where $\varepsilon > 0$ is a constant. Constant stepsizes lead to stationary steepest and gradient descent schemes. We will see later on that if f satisfies certain conditions and ε is sufficiently small, then the fixed stepsize gradient descent can converge. Note that no computations are required for tuning the stepsizes.

3.2.2 Optimal Line Search

At every iteration k, optimal line search computes the stepsize ε_k as follows:

$$\varepsilon_k = \arg\min_{\varepsilon \geq 0} f(x_k + \varepsilon d_k). \tag{3.3}$$

Clearly, the above problem presumes a chosen direction of motion d_k. The method searches for the value of ε_k over the half line $[0, \infty)$, therefore it is called *optimal line search*. By this description, it is clear that optimal line search corresponds to a sequence of optimal *local* decisions, yielding maximum reduction of f at every iteration k, given that the current state or position is x_k and the descent direction has been chosen to be d_k.

Note: Clearly, optimal line search corresponds to a fast time scale procedure implemented at every iteration of the gradient descent algorithm (slow time scale).

Solving for ε problem (3.3) at every iteration of the gradient or steepest descent algorithms may be difficult and costly. This motivates the *Armijo rule*.

3.2.3 Armijo Rule

As an alternative approach to optimal line search, the Armijo rule, also known as *backtracking line search*, ensures that the (loss) function f decreases sufficiently at every iteration. In return, it reduces complexity as compared to optimal line search. To understand how the Armijo rule works, we may use Taylor's theorem to write:

$$\begin{aligned} f(x_k + \varepsilon d_k) &= f(x_k) + \nabla f(x_k)^T (x_k + \varepsilon d_k - x_k) + o(\|x_k + \varepsilon d_k - x_k\|) \\ &= f(x_k) + \varepsilon \nabla f(x_k)^T d_k + o(\varepsilon). \end{aligned} \tag{3.4}$$

If ε is sufficiently small, then the term $\varepsilon |\nabla f(x_k)^T d_k|$ becomes dominant compared to $o(\varepsilon)$. Also, $\varepsilon \nabla f(x_k)^T d_k < 0$, since d_k is a descent direction at x_k. As a result,

$$f(x_k + \varepsilon d_k) \leq f(x_k). \tag{3.5}$$

Hence, at every iteration, the Armijo rule attempts to pick a sufficiently small ε such that equation (3.5) holds. The rule is summarized as follows:

At iteration k of the gradient descent algorithm (slow time scale):

(i) Start with $\varepsilon = \epsilon_0$ for some pre-selected ϵ_0. Let $0 < \sigma < 1$, $0 < \beta < 1$ be two user-defined parameters.

(ii) If $f(x_k + \varepsilon d_k) \leq f(x_k) + \sigma \varepsilon \nabla f(x_k)^T d_k$, stop and return the current value of ε, else set $\varepsilon \leftarrow \beta \varepsilon$ and repeat step (ii).

Thus, the Armijo rule terminates at the first iteration $l \geq 1$ such that

$$f(x_k + \beta^l \epsilon_0 d_k) \leq f(x_k) + \sigma \beta^l \epsilon_0 \nabla f(x_k)^T d_k. \tag{3.6}$$

Remark: There exist good options for ϵ_0, σ and β in the optimization literature.

Note: Clearly, the Armijo rule corresponds to a fast time scale procedure implemented at every iteration of the gradient descent algorithm (slow time scale).

3.3 The Direction of Largest Increase in f

∇f plays a vital role in gradient and steepest descent algorithms, because $\nabla f(x_k)$ is the direction of largest increase in function f when the current state or position is x_k. As we have mentioned before, the key idea behind steepest descent is to find a local minimum by moving in the opposite direction of $\nabla f(x_k)$, i.e., in the direction of $-\nabla f(x_k)$. In this section, we will prove that $\nabla f(x_k)$ is the direction of the largest increase in f when at x_k.

Proof. Let v be a unit vector, i.e., $||v|| = 1$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function. Considering moving from $x \in \mathbb{R}^n$ in the direction v by a small amount εv, we have:

$$f(x + \varepsilon v) = f(x) + \nabla f(x)^T (x + \varepsilon v - x) + o(\varepsilon) = f(x) + \varepsilon \nabla f(x)^T v + o(\varepsilon),$$

leading to

$$\nabla_v f(x) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon v) - f(x)}{\varepsilon} = \nabla f(x)^T v.$$

Next, by applying Cauchy-Schwarz inequality we have that

$$|\nabla f(x)^T v| \leq \|\nabla f(x)\| \|v\|. \tag{3.7}$$

The equality in Equation 3.7 is achieved when $\nabla f(x)$ and v are collinear. Therefore, v should be in the direction of $\nabla f(x)$ for largest increase in f. □

3.4 Convergence: Fixed Stepsize Steepest Descent

In this section, we focus on the convergence analysis for the steepest descent with constant stepsizes. First, we start by providing the following useful result:

Lemma 3.1 (Descent Lemma). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable with a Lipschitz gradient, i.e, $\exists L > 0$ such that*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Then,

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} L \|x - y\|^2, \quad \forall x, y \in \mathbb{R}^n.$$

Proof. Let $g : \mathbb{R} \to \mathbb{R}$ be defined as follows:

$$g(t) = f(x + t(y - x)).$$

We now have:

$$\begin{aligned}
g(1) &= g(0) + \int_0^1 g'(t)dt = g(0) + \int_0^1 \nabla f(x + t(y-x))^T (y-x) dt && \text{(chain rule)} \\
&= g(0) + \int_0^1 [\nabla f(x + t(y-x)) - \nabla f(x) + \nabla f(x)]^T (y-x) dt \\
&= g(0) + \int_0^1 [\nabla f(x + t(y-x)) - \nabla f(x)]^T (y-x) dt + \int_0^1 \nabla f(x)^T (y-x) dt \\
&= g(0) + \int_0^1 [\nabla f(x + t(y-x)) - \nabla f(x)]^T (y-x) dt + \nabla f(x)^T (y-x) \\
&\leq g(0) + \int_0^1 \|\nabla f(x + t(y-x)) - \nabla f(x)\| \|y-x\| dt + \nabla f(x)^T (y-x) && \text{(Cauchy-Schwarz inequality)} \\
&\leq g(0) + \int_0^1 L\|x + t(y-x) - x\| \|y-x\| dt + \nabla f(x)^T (y-x) && \text{(Lipschitz gradient)} \\
&= g(0) + L \int_0^1 t\|y-x\|^2 dt + \nabla f(x)^T (y-x) \\
&= g(0) + \frac{1}{2} L\|y-x\|^2 + \nabla f(x)^T (y-x).
\end{aligned}$$

Hence, by discarding the intermediate steps we have shown that

$$g(1) \leq g(0) + \frac{1}{2} L\|y-x\|^2 + \nabla f(x)^T (y-x).$$

We now note that $g(1) = f(y)$ and $g(0) = f(x)$. Therefore:

$$f(y) \leq f(x) + \frac{1}{2} L\|y-x\|^2 + \nabla f(x)^T (y-x).$$

□

The following theorem provides conditions for the convergence of the steepest descent algorithm with fixed stepsizes.

Theorem 3.2. *Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable with a Lipschitz gradient and Lipschitz constant L. Suppose that $f_{min} = \min_x f(x) > -\infty$. If $\varepsilon_k = \varepsilon, \ \forall k \geq 0$ and $0 < \varepsilon < \frac{2}{L}$, then the steepest descent converges to a stationary point, i.e.,*

$$\lim_{k \to \infty} \nabla f(x_k) = 0. \tag{3.8}$$

Proof. Theorem 3.2 is proved by first applying the Descent Lemma for $y = x_{k+1}$ and $x = x_k$:

$$\begin{aligned}
f(x_{k+1}) &\leq f(x_k) + \frac{1}{2} L\|x_{k+1} - x_k\|^2 + \nabla f(x_k)^T (x_{k+1} - x_k) \\
&= f(x_k) + \frac{1}{2} L\|x_k - \varepsilon \nabla f(x_k) - x_k\|^2 + \nabla f(x_k)^T (x_k - \varepsilon \nabla f(x_k) - x_k) \\
&= f(x_k) + \frac{1}{2} L\varepsilon^2 \|\nabla f(x_k)\|^2 - \varepsilon \|\nabla f(x_k)\|^2
\end{aligned}$$

or equivalently,

$$\varepsilon \left(1 - \frac{1}{2} L\varepsilon\right) \|\nabla f(x_k)\|^2 \leq f(x_k) - f(x_{k+1}). \tag{3.9}$$

By summing over k from 0 to N we obtain:

$$\begin{aligned}\varepsilon\left(1-\frac{1}{2}L\varepsilon\right)\sum_{k=0}^{N}\|\nabla f(x_k)\|^2 &\leq \sum_{k=0}^{N}\left(f(x_k)-f(x_{k+1})\right)\\ &= f(x_0)-f(x_{N+1})\\ &\leq f(x_0)-f_{min}.\end{aligned}$$

Since $0<\varepsilon<\frac{2}{L}$ by assumption, it follows that $1-\frac{1}{2}L\varepsilon>0$ and therefore,

$$\sum_{k=0}^{N}\|\nabla f(x_k)\|^2 \leq \frac{f(x_0)-f_{min}}{\varepsilon(1-\frac{1}{2}L\varepsilon)},$$

which is valid for any N. Hence,

$$\sum_{k=0}^{\infty}\|\nabla f(x_k)\|^2 \leq \frac{f(x_0)-f_{min}}{\varepsilon(1-\frac{1}{2}L\varepsilon)} < \infty.$$

By the series convergence we conclude that

$$\lim_{k\to\infty}\|\nabla f(x_k)\| = 0$$

or equivalently

$$\lim_{k\to\infty}\nabla f(x_k) = 0.$$

□

Remarks:

1. Theorem 3.2 implies nothing about the convergence of x_k. As an illustration, consider the function $f(x) = \exp(-x^2)$, which is shown in Figure 3.1. If the initial point is $x_0 > 0$, then x_k will diverge to ∞. If $x_0 < 0$, then x_k will diverge to $-\infty$. We also note that initializing at $x_0 = 0$, leads to $x_k = 0$ for any $k \geq 0$, which corresponds to an equilibrium point of the algorithm, although a maximum.
2. The Lipschitz continuity condition on ∇f roughly requires that the curvature of f is no more than L in all directions. It is possible to show that this condition is satisfied for some $L > 0$ and a given loss function f, if f is twice differentiable and the Hessian $\nabla^2 f$ is bounded over $\mathbb{R}^n$. Unfortunately, it is generally difficult of obtain an estimate of L and by extension the range of stepsizes that guarantee convergence. In such cases, experimentation is required to identify an appropriate range of stepsizes.
3. The convergence result in Theorem 3.2 is not too strong because the theorem imposes only few assumptions on the loss function f. If additional assumptions on the structure of f are imposed (besides Lipschitz continuity of ∇f), we can get stronger convergence results and information about the convergence rate. The following theorem provides such a strengthening of the assumptions.

Theorem 3.3. *In addition to the assumptions of Theorem 3.2, assume that:*

(a) *f is convex*

(b) *$\exists x^*$ such that $f(x^*) = \min_x f(x) > -\infty$, i.e, a minimum exists.*

Then, for sufficiently small ε the following hold:

(i) $\lim_{k\to\infty} f(x_k) = f(x^*)$

(ii) $f(x_k) \to f(x^*)$ *at rate* $\frac{1}{k}$.

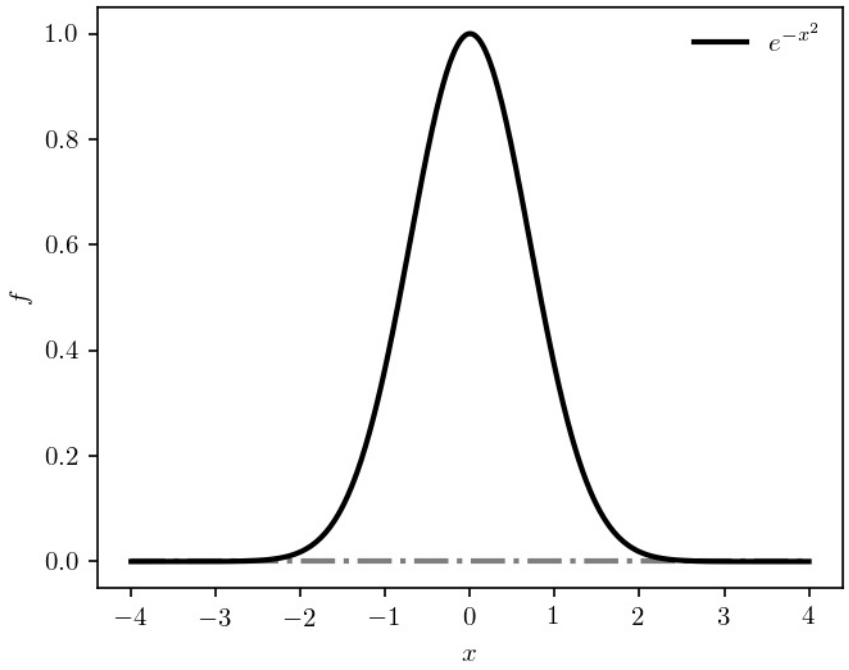

Figure 3.1: Depending on the initial point x_0, $x_k \xrightarrow[k\to\infty]{} \pm\infty$ or $x_k = 0$ for all $k \geq 0$.