STAT 516: Stochastic Modeling of Scientific Data **Autumn 2024**

Lecture 8: EM Algorithm and Gradient Descent

Instructor: Yen-Chi Chen

These notes are partially based on those of Mathias Drton.

8.1 Introduction

In statistics, often we focus a lot on how to design a good estimator with nice theoretical properties (such as consistency, convergence rate, good posterior distribution). However, this is often not enough when we are working with a realistic dataset. When analyzing the data, we need to be able to *numerically compute* the estimator. In many cases, we can propose an estimator that has many good theoretical properties such as uniqueness, fast convergence rate, asymptotic normality, but we are not able to compute it.

For one example, consider a simple linear regression $Y = X^T\beta + \epsilon$, where we observe $n = 1000$ data points and the covariates $X \in \mathbb{R}^{100}$. Assume that half of the parameters β are 0 (i.e., 50 entries of β are 0). Our goal is to find the the ones that are not 0. You can easily see that we can try all combination of β's (there will be totally $\binom{100}{50}$ combinations) and choose the one that best fit to the data. Under suitable assumptions, the estimator is unique, consistent, and has asymptotic normality. However, can we really find all combinations? Mathematically, yes–there are only $\binom{100}{50}$. But practically, very difficult! There are about 10^{29} combinations!

In this lecture, we will talk about numerical methods that help us to find statistical estimators. We will start with the EM algorithm, a famous method in statistics for finding a maximizer, and then talk about a more general approach called gradient descent/ascent method along with a stochastic version of it.

Before we proceed, we start with a simple example about finding the ratio of certain gene. Suppose we collect n individuals from a population and for each individual, his/her blood type $f_i \in \{A, B, AB, O\}$. Recall that blood types are determined by six possible genotypes $g_i \in \{AA, AO, BB, BO, AB, OO\}$ that corresponds to

$$\begin{aligned} g_i &= AA \text{ or } AO \Rightarrow f_i = A \\ g_i &= BB \text{ or } BO \Rightarrow f_i = B \\ g_i &= AB \Rightarrow f_i = AB \\ g_i &= OO \Rightarrow f_i = O. \end{aligned}$$

The data we have is $\{f_1, \cdots, f_n\}$ and our goal is to estimate the ratio of genes A, B, and O (we will denote them by p_A, p_B, p_O). Note that in the language of missing data, $G_n = \{g_1, \cdots, g_n\}$ will be called the complete-data and $F_n = \{f_1, \cdots, f_n\}$ will be called the observed-data. Often G_n are unobserved and we only have access to the observed data F_n so the goal is to see if we can recover p_A, p_B, p_O with the observed data.

Under the Hardy-Weinberg equilibrium, the number of each genotype is drawn from a multinomial distribution with

$$\begin{gathered} P(g_i = AA) = p_A^2, P(g_i = AO) = 2p_Ap_O, P(g_i = BB) = p_B^2, \\ P(g_i = BO) = 2p_Bp_O, P(g_i = AB) = 2p_Ap_B, P(g_i = OO) = p_O^2. \end{gathered}$$

Thus, if we observed the complete-data, the likelihood function is

$$L(p_A, p_B, p_O|G_n) \propto (p_A^2)^{m_{AA}}(2p_Ap_O)^{m_{AO}}(p_B^2)^{m_{BB}}(2p_Bp_O)^{m_{BO}}(2p_Ap_B)^{m_{AB}}(p_O^2)^{m_{OO}},$$

where each $m_{KL} = \sum_{i=1}^{n} I(g_i = KL)$. This likelihood function is often called the *complete data likelihood*. Finding the MLE of (p_A, p_B, p_O) is not very difficult in this case.

However, in reality, we do not observe G_n but instead, we only have F_n. So the likelihood function we are actually working with is

$$L(p_A, p_B, p_O|F_n) \propto (p_A^2 + 2p_A p_O)^{n_A}(p_B^2 + 2p_B p_O)^{n_B}(2p_A p_B)^{n_{AB}}(p_O^2)^{n_O}, \tag{8.1}$$

where $n_K = \sum_{i=1}^{b} I(f_i = K)$ is the number of blood type K. There is no closed-form of the MLE of equation (8.1). Thus, we need to use some numerical methods to find it.

8.2 EM Algorithm

The EM (Expectation Maximization) algorithm offers a simple and elegant way to finding an MLE when the likelihood function is complex. The EM is often applied to the case where the model involves *hidden/latent units*. Latent variable models and missing data are two common scenarios that it can be applied to.

Here we describe the general formulation of the EM algorithm in simple missing data problem. Let $\mathbf{x}$ be the complete data (including the latent variables) and $\mathbf{y}$ be the observed data and let $L(\theta|\mathbf{x}) = p(\mathbf{x};\theta)$ be the likelihood function (on the complete data). Given an initial guess, the EM algorithm keeps iterates the following two steps:

- E-step: evaluate $Q(\theta;\theta^{(n)}|\mathbf{y}) = \mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$,
- M-step: update $\theta^{(n+1)} = \mathsf{argmax}_\theta Q(\theta;\theta^{(n)}|\mathbf{y})$,

until certain criterion is met (e.g., $\|\theta_{n-1} - \theta^{(n)}\|_\infty < \epsilon$). Note that $\mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$ means that $\mathbf{x}$ is from the distribution $p(\cdot;\theta^{(n)})$ conditional on $\mathbf{y}$.

The EM algorithm has a powerful property about *ascending likelihood*.

Proposition 8.1 (Ascending property of EM)

$$\ell(\theta^{(n+1)}|\mathbf{y}) = \log p(\mathbf{y};\theta^{(n+1)}) \geq \log p(\mathbf{y};\theta^{(n)}) = \ell(\theta^{(n)}|\mathbf{y}).$$

Namely, the likelihood value will not decrease after each step of EM.

Although Proposition 8.1 states that the likelihood value is non-decreasing after each iteration, it does not guarantee that it is always increasing and also, it does not guarantee to find the global maximizer (MLE).

Proof:

The key step of the proof is the Jensen's inequality, which regularizes the expectation of a convex function. A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is called *convex* if its domain is a convex set and for any $\alpha \in [0, 1]$ and any $a, b \in \mathbb{R}^d$,

$$\alpha f(a) + (1 - \alpha)f(b) \geq f(\alpha a + (1 - \alpha)b).$$

Jensen's inequality: if g is a convex function, then

$$\mathbb{E}(f(X)) \geq f(\mathbb{E}(X))$$

for any random variable X. If the function f is concave, then

$$\mathbb{E}(f(X)) \leq f(\mathbb{E}(X))$$

Since $\mathbf{x}$ is the complete-data and $\mathbf{y}$ is the observed-data,

$$p(\mathbf{x};\theta) = p(\mathbf{x},\mathbf{y};\theta) = p(\mathbf{x}|\mathbf{y};\theta)p(\mathbf{y};\theta).$$

Therefore,

$$\ell(\theta|\mathbf{y}) = \log p(\mathbf{y};\theta) = \log p(\mathbf{x};\theta) - \log p(\mathbf{x}|\mathbf{y};\theta).$$

Recall that we want to compute the difference of likelihood function under $\theta^{(n+1)}$ versus $\theta^{(n)}$, which is

$$\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) = \log p(\mathbf{x};\theta^{(n+1)}) - \log p(\mathbf{x};\theta^{(n)}) - \left\{\log p(\mathbf{x}|\mathbf{y};\theta^{(n+1)}) - \log p(\mathbf{x}|\mathbf{y};\theta^{(n)})\right\}.$$

Since $\theta^{(n+1)} = \mathsf{argmax}_\theta Q(\theta|\mathbf{y};\theta^{(n)})$, we want to associate this with the function Q. Thus, we take expectation of both sides of the random variable $\mathbf{x}$ conditional on $\mathbf{y}$ and from the distribution $p(\cdot;\theta^{(n)})$, leading to

$$\begin{aligned}\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) &= \mathbb{E}\left\{\log p(\mathbf{x};\theta^{(n+1)}) - \log p(\mathbf{x};\theta^{(n)})|\mathbf{y};\theta^{(n)}\right\}\\ &\quad - \mathbb{E}\left\{\log p(\mathbf{x}|\mathbf{y};\theta^{(n+1)}) - \log p(\mathbf{x}|\mathbf{y};\theta^{(n)})|\mathbf{y};\theta^{(n)}\right\}\\ &= \underbrace{Q(\theta^{(n+1)}|\mathbf{y};\theta^{(n)}) - Q(\theta^{(n)}|\mathbf{y};\theta^{(n)})}_{\geq 0} - \mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\}\\ &\geq -\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\}.\end{aligned}$$

Finally, we apply the Jensen's inequality to the last quantity using the fact that log is a concave function, which implies

$$\begin{aligned}\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\} &\leq \log\mathbb{E}\left\{\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}|\mathbf{y};\theta^{(n)}\right\}\\ &= \log\left(\int\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}p(\mathbf{x}|\mathbf{y};\theta^{(n)})d\mathbf{x}\right)\\ &= \log(1) = 0.\end{aligned}$$

Thus, we have shown that

$$\ell(\theta^{(n+1)}|\mathbf{y}) - \ell(\theta^{(n)}|\mathbf{y}) \geq -\mathbb{E}\left\{\log\left[\frac{p(\mathbf{x}|\mathbf{y};\theta^{(n+1)})}{p(\mathbf{x}|\mathbf{y};\theta^{(n)})}\right]|\mathbf{y};\theta^{(n)}\right\} \geq 0,$$

which completes the proof.

■

Now we return to our blood type problem. We will illustrate how do we do EM algorithm in this case. To determine the E-step, we need the complete data likelihood function, which is

$$L(p_A,p_B,p_O|G_n) \propto (p_A^2)^{m_{AA}}(2p_Ap_O)^{m_{AO}}(p_B^2)^{m_{BB}}(2p_Bp_O)^{m_{BO}}(2p_Ap_B)^{m_{AB}}(p_O^2)^{m_{OO}}.$$

The log-likelihood function is

$$\begin{aligned}\ell(p_A,p_B,p_O|G_n) &= m_{AA}\log(p_A^2) + m_{AO}\log(2p_Ap_O) + m_{BB}\log(p_B^2)\\ &\quad + m_{BO}\log(2p_Bp_O) + m_{AB}\log(2p_Ap_B) + m_{OO}\log(p_O^2) + C_0,\end{aligned}$$

where C_0 is some constant independent of p.

E-step. Let $p = (p_A, p_B, p_O)$. The Q function under t-th iteration is then

$$
\begin{aligned}
Q(p|F_n; p^{(t)}) &= \mathbb{E}(\ell(p|G_n)|F_n; p^{(t)}) \\
&= \mathbb{E}\{m_{AA}|F_n; p^{(t)}\}2\log p_A + \mathbb{E}\{m_{AO}|F_n; p^{(t)}\}\log(p_A p_O) + \mathbb{E}\{m_{BB}|F_n; p^{(t)}\}2\log p_B \\
&\quad + \mathbb{E}\{m_{BO}|F_n; p^{(t)}\}\log(p_B p_O) + \mathbb{E}\{m_{AB}|F_n; p^{(t)}\}\log(p_A p_B) + \mathbb{E}\{m_{OO}|F_n; p^{(t)}\}2\log p_O + C_0.
\end{aligned}
$$

Because $n_A = \sum_{i=1}^n I(f_i = A) = \sum_{i=1}^n I(g_i = AA \text{ or } AO)$. So

$$
m_{AA}|n_A; p^{(t)} \sim \mathsf{Bin}\left(n_A, \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}}\right).
$$

Thus,

$$
\mathbb{E}\{m_{AA}|F_n; p^{(t)}\} = n_A \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}}.
$$

The other conditional expectations can be derived in a similar way:

$$
\begin{aligned}
m_{AA}^{(t)} &= \mathbb{E}\{m_{AA}|F_n; p^{(t)}\} = n_A \frac{p_A^{(t)2}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}} \\
m_{AO}^{(t)} &= \mathbb{E}\{m_{AO}|F_n; p^{(t)}\} = n_A \frac{2p_A^{(t)}p_O^{(t)}}{p_A^{(t)2} + 2p_A^{(t)}p_O^{(t)}} \\
m_{BB}^{(t)} &= \mathbb{E}\{m_{BB}|F_n; p^{(t)}\} = n_B \frac{p_B^{(t)2}}{p_B^{(t)2} + 2p_B^{(t)}p_O^{(t)}} \\
m_{BO}^{(t)} &= \mathbb{E}\{m_{BO}|F_n; p^{(t)}\} = n_B \frac{2p_B^{(t)}p_O^{(t)}}{p_B^{(t)2} + 2p_B^{(t)}p_O^{(t)}} \\
m_{AB}^{(t)} &= \mathbb{E}\{m_{AB}|F_n; p^{(t)}\} = n_{AB} \\
m_{OO}^{(t)} &= \mathbb{E}\{m_{OO}|F_n; p^{(t)}\} = n_{OO}.
\end{aligned}
$$

Using this, we can rewrite the Q function as

$$
Q(p|F_n; p^{(t)}) = (2m_{AA}^{(t)} + m_{AO}^{(t)} + m_{AB}^{(t)})\log p_A + (2m_{BA}^{(t)} + m_{BO}^{(t)} + m_{AB}^{(t)})\log p_B + (2m_{OO}^{(t)} + m_{AO}^{(t)} + m_{BO}^{(t)})\log p_O.
$$

M-step. Based on the Q function, the M-step is the traditional maximization problem with a constraint $p_A + p_B + p_O = 1$. We can use Lagrangian multiplier to solve it, which gives

$$
\begin{aligned}
p_A^{(t+1)} &= \frac{2m_{AA}^{(t)} + m_{AO}^{(t)} + m_{AB}^{(t)}}{2n} \\
p_B^{(t+1)} &= \frac{2m_{BB}^{(t)} + m_{BO}^{(t)} + m_{AB}^{(t)}}{2n} \\
p_O^{(t+1)} &= \frac{2m_{OO}^{(t)} + m_{AO}^{(t)} + m_{BO}^{(t)}}{2n}
\end{aligned}
$$

Starting from an initial guess $p^{(0)}$, we then iterates the E-step and M-step to obtain

$$
p^{(1)}, p^{(2)}, \cdots
$$

until certain stopping rule is satisfied and we use the final parameter value as our *computed MLE.*

In the above ideal case, both E-step and M-step has a closed form. However, in general they may not have a closed form so we need to use other numerical approach to approximate it. The E-step can be approximated by sampling the complete data $\mathbf{x}$ from the conditional PDF/PMF $p(\cdot|\mathbf{y};\theta^{(t)})$ and then use the average as a Monte Carlo estimator of the Q function. This idea is called the *Monte Carlo* EM algorithm[1]. The M-step may not have a closed-form as well. In this case, we need to use numerical optimization method, such as the gradient ascent method (we will discuss it later).

Note that only certain MLEs can be found by the EM algorithm; not all MLE can be found using the EM. Thus many research is about how to design an EM algorithm for finding MLE under certain problems.

Remark.

- A notable issue of the EM-algorithm (and other numerical methods as well) is the critical points problem. Although EM algorithm will not decrease the likelihood value, it may stuck at a critical point (local maxima or saddle points). If the likelihood function is convex, often this will not be a big problem but if the likelihood function is non-convex (which, unfortunately, is often the case), this could be a serious issue. Thus, often people will reinitialize the starting point of the EM multiple times and choose the convergent point that has the highest likelihood value. However, even doing so we may still not getting the true MLE.
- In the theoretical analysis of the EM-algorithm, people often focus on the Q function:

$$Q(\theta;\theta^{(n)}|\mathbf{y}) = \mathbb{E}(\log L(\theta|\mathbf{x})|\mathbf{y};\theta^{(n)})$$

  because essentially, the EM is to update θ by maximizing $Q(\theta;\theta^{(n)}|\mathbf{y})$. This Q function is a sample quantity and it has a population version of it

$$Q(\theta;\theta') = \mathbb{E}\{Q(\theta;\theta'|\mathbf{y})\}.$$

  If we update $\theta^{(n)}$ according to $Q(\theta;\theta^{(n)})$, this is called the *population EM algorithm.* The smoothness of this function and the behavior of $Q(\theta;\theta_{MLE})$ play key role in the convergence of EM[2].
- As you may notice, in the proof of likelihood ascent property, we only need $Q(\theta^{(n+1)}|\mathbf{y};\theta^{(n)}) - Q(\theta^{(n)}|\mathbf{y};\theta^{(n)})$. Thus, even if we do not maximization in the M-step, the likelihood value will be non-decreasing as long as $\theta^{(n+1)}$ has a higher value in the Q function. This idea was further developed into the EM gradient algorithm[3], which only finds a new $\theta^{(n+1)}$ that has a higher Q function value. This is particularly useful when the maximization is intractable. The paper of Kenneth Lange (1995) provides examples where the maximization is hard to compute so EM cannot be applied but the EM gradient works nicely.

8.3 Gradient Descent/Ascent

Gradient descent algorithm is a generic approach of finding a minimum of a smooth function. Note that finding the minimum and finding the maximum are almost equivalent question (just flip f to $-f$ or $1/f$) so here we will focus on finding the minimum. Let f be a smooth function. Out goal is to find $x^* = \mathsf{argmin}_x f(x)$, the location where the minimum of f occurs, and the minimal value $f^* = \min_x f(x)$.

[1] Please see "*implementations of the Monte Carlo EM Algorithm*" by Richard A. Levine and George Casella and https://arxiv.org/abs/1206.4768 for more details.

[2] If you are interested in, I would recommend this paper: https://arxiv.org/abs/1408.2156

[3] "A Gradient Algorithm Locally Equivalent to the EM Algorithm" by Kenneth Lange (1995).

In gradient descent (and almost all other minimization methods), the shape of a function is very important. When the function has a nice shape (like a Gaussian), there is a unique and well-defined minimum and gradient descent is able to find it in a reasonable time.

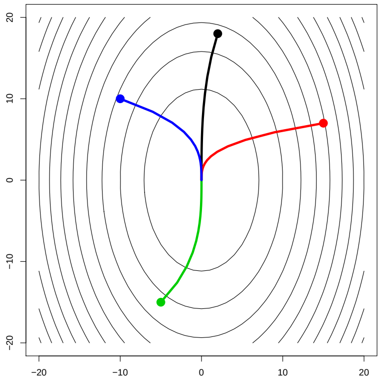
We will start with the simplest case by assuming that f is a smooth and *convex* function. Recall that convexity implies that for any x, y and $\alpha \in [0,1]$,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y).$$

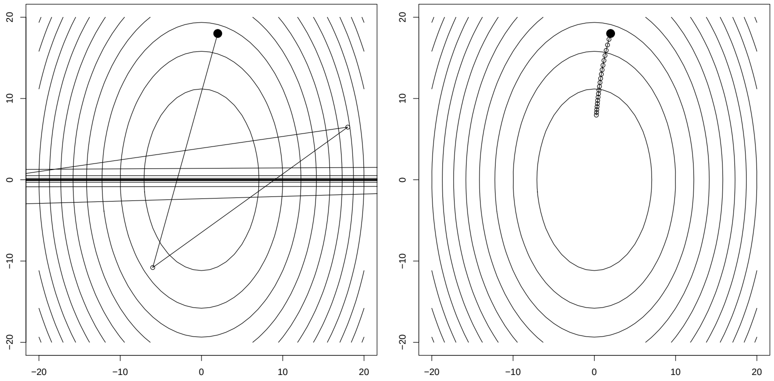
The gradient descent algorithm works as follows.

1. Choose an initial point $x^{(0)}$ and a step size (learning rate) $\gamma > 0$.
2. Iterates the following until some convergence criterion is met:
$$x^{(t+1)} \leftarrow x^{(t)} - \gamma \nabla f(x^{(t)}).$$

Namely, each time we move from the current point $x^{(t)}$ by γ times the value of gradient. The following picture shows a gradient descent path starting from 4 different initial points



The step size γ is very important in gradient descent. It affects the convergence in two senses. If γ is too large, the gradient descent may diverge. If γ is too small, the gradient descent may converge very slowly. Note that one can also choose $\gamma = \gamma_t$ that depends on the number of steps.



In the top left panel, we choose a very large step size and the gradient descent diverges at the end. In the top right panel, we choose a very small step size and the algorithm converges very slowly. Therefore, how to choose a good step size is very important. There are two common strategies for this.

Backtracking line search.

- Choose $0 < \beta < 1$ and $0 < \alpha \leq 1/2$.
- At each iteration, start with $\gamma = \gamma_0$, and while

  $$f(x^{(t)} - \gamma \nabla f(x^{(t)})) > f(x^{(t)}) - \alpha\gamma \|\nabla f(x^{(t)})\|_2^2,$$

  shrink $\gamma = \beta\gamma$. Else, perform gradient descent update $x^{(t+1)} \leftarrow x^{(t)} - \gamma \nabla f(x^{(t)})$.

Note that often we will choose $\alpha = 1/2$.

Exact line search. The exact line search attempts to choose γ based on

$$\gamma = \mathsf{argmin}_{s \geq 0} f(x^{(t)} - s \nabla f(x^{(t)})).$$

It is another optimization problem so it is hard to obtain the minimizer exactly (but we can do approximation to it).

Note that in both backtracking and exact line search, one has to choose the step size at every time step t.

8.3.1 Convergence analysis

Here our convergence analysis is not about the statistical convergence but the algorithmic convergence. We want to analyze how close is our minimizer $x^{(t)}$ to the true minimum x^*. We will analyze the convergence rate in terms of t, the number of steps. Let $f^* = f(x^*) = \min_x f(x)$.

Before stating our main result, we first note the following two important properties of f being L-smooth and convex.

$$(L\text{-smooth}) \ \|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \Rightarrow f(x) \leq f(y) + (x-y)^T \nabla f(y) + \frac{L}{2}\|y - x\|^2 \tag{8.2}$$

$$(\text{convexity}) \ f \text{ is convex } \Rightarrow f(x) \leq f(y) + (x-y)^T \nabla f(x) \tag{8.3}$$

Note that the property

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$$

is also known as L-smooth.

For a function that is L-smooth and convex, we have the following useful lemma.

Lemma 8.2 *Suppose that f is L-smooth and convex and the step size $\gamma \leq 2/L$, then*

$$f(y) - f(x) \leq (y-x)^T \nabla f(y) - \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2 \tag{8.4}$$

$$(y-x)^T(\nabla f(y) - \nabla f(x)) \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2 \tag{8.5}$$

$$f(x^{(t+1)}) \leq f(x^{(t)}) - \frac{\gamma}{2}(2 - \gamma L)\|\nabla f(x^{(t)})\|^2 \tag{8.6}$$

$$\|x^{(t+1)} - x^*\|^2 \leq \|x^{(t)} - x^*\|^2 - \frac{\gamma}{L}(2 - \gamma L)\|\nabla f(x^{(t)})\|^2 \tag{8.7}$$

$$f(x^{(t)}) - f^* \leq \|x^{(0)} - x^*\|\|\nabla f(x^{(t)})\| \tag{8.8}$$

With the above lemma, we can prove the following convergence rate of a convex and L-smooth function.

Theorem 8.3 *Assume that f is convex and differentiable and the domain of f is $\mathbb{R}^d$ and f is L-smooth. Then the gradient descent with a fixed step size $\gamma \leq 2/L$ satisfies*

$$|f(x^{(T)}) - f^*| \leq \frac{2\|x^{(0)} - x^*\|_2^2}{\gamma(2-\gamma L)} \frac{1}{T}.$$

In the case of backtracking, we will replace γ by β/L.

Proof: Define $\delta_t = f(x^{(t)}) - f^*$. Then subtracting f^* in both sides of equation (8.6), we obtain

$$\begin{aligned}
\delta_{t+1} &\leq \delta_t - \frac{\gamma}{2}(2-\gamma L)\|\nabla f(x^{(t)})\|^2 \\
&\leq \delta_t - \frac{\gamma}{2}(2-\gamma L)\frac{|f(x^{(t)}) - f^*|^2}{\|x^{(0)} - x^*\|^2} \qquad \text{by (8.8)} \\
&= \delta_t - \frac{\gamma}{2}(2-\gamma L)\frac{\delta_t^2}{\|x^{(0)} - x^*\|^2} \\
&= \delta_t - \eta\delta_t^2,
\end{aligned}$$

where $\eta = \frac{\gamma(2-\gamma L)}{2\|x^{(0)}-x^*\|^2} > 0$.

To obtain the rate of δ_t, we use the idea of telescoping. Dividing both sides in the above inequality by $\frac{1}{\delta_{t+1}\delta_t}$ leads to

$$\frac{1}{\delta_t} \leq \frac{1}{\delta_{t+1}} - \eta\frac{\delta_t}{\delta_{t+1}}$$

and after rearrangements, this becomes

$$\eta\frac{\delta_t}{\delta_{t+1}} \leq \frac{1}{\delta_{t+1}} - \frac{1}{\delta_t}.$$

Equation (8.7) shows that $\delta_{t+1} < \delta_t$, so $\frac{\delta_t}{\delta_{t+1}} > 1$, which further implies

$$\eta < \eta\frac{\delta_t}{\delta_{t+1}} \leq \frac{1}{\delta_{t+1}} - \frac{1}{\delta_t}.$$

By summing the above inequality for $t = 0$ to $t = T-1$, we obtain

$$\eta T < \frac{1}{\delta_T} - \frac{1}{\delta_0} < \frac{1}{\delta_T}.$$

Thus, we conclude

$$f(x^{(T)}) - f^* = \delta_T < \frac{1}{\eta T} = \frac{2\|x^{(0)} - x^*\|^2}{\gamma(2-\gamma L)} \frac{1}{T} = O(T^{-1}).$$

■

In the case of asymptotic analysis, we will say that the gradient descent gas a convergence rate $O(1/t)$. Or we say that the gradient descent algorithm finds an ϵ-suboptimal point in $O(1/\epsilon)$ iterations.

If we further assume that the function not only has a Lipschitz gradient but is also strongly convex, i.e.,

$$f(x) - \frac{m}{2}\|x\|_2^2$$

is convex for some $m > 0$, then the gradient descent algorithm converges even faster.

Theorem 8.4 *Assume that f is convex and differentiable and the domain of f is $\mathbb{R}^d$ and the gradient $\nabla f(x)$ is Lipschitz continuous with a constant $L > 0$. Moreover, f is strongly convex with $f(x) - \frac{m}{2}\|x - x^*\|_2^2$ for some $m > 0$. Suppose that $f^* = f(x^*) = 0$. Then the gradient descent with a fixed step size $\gamma \leq 1/L$ satisfies*

$$\|x^{(t)} - x^*\|_2^2 \leq (1-\gamma m)^t \|x^{(0)} - x^*\|_2^2, \quad f(x^{(t)}) \leq f(x^{(0)}) \left(1 - \frac{m^2}{L^2} + m^2\left(\gamma - \frac{1}{L}\right)^2\right)^t$$

Note that the fastest choice of γ is $\gamma = 1/L$, leading to

$$\|x^{(t)} - x^*\|_2 \leq (1 - m/L)^{t/2} \|x^{(0)} - x^*\|_2.$$

Proof:

Convergence in distance. The proof relies on the following two facts.

Fact 1: When $f(x)$ is m-strongly convex, we have

$$f(x^*) - f(x^{(t)}) \geq (x^* - x^{(t)})^T \nabla f(x^{(t)}) + \frac{m}{2}\|x^* - x^{(t)}\|^2,$$

which implies

$$(x^{(t)} - x^*)^T \nabla f(x^{(t)}) \geq f(x^{(t)}) - f(x^*) + \frac{m}{2}\|x^* - x^{(t)}\|^2. \tag{8.9}$$

Fact 2: because the gradient is L-Lipschitz and $f(x) \geq f(x^*) = 0$, for any point x we have

$$\begin{aligned} -f(x) &\leq f\left(x - \frac{1}{L}\nabla f(x)\right) - f(x) \\ &\leq -\frac{1}{L}\|\nabla f(x)\|^2 + \frac{1}{L}\|\frac{1}{L}\nabla f(x)\|^2 \\ &= -\frac{1}{2L}\|\nabla f(x)\|^2. \end{aligned}$$

Thus,

$$f(x) \geq \frac{1}{2L}\|\nabla f(x)\|^2. \tag{8.10}$$

Consider the step at $t+1$. Then

$$\begin{aligned} \|x^{(t+1)} - x^*\|^2 &= \|x^{(t)} - x^* - \gamma \nabla f(x^{(t)})\| \\ &= \|x^{(t)} - x^*\|^2 - 2\gamma (x^{(t)} - x^*)^T \nabla f(x^{(t)}) + \gamma^2 \|\nabla f(x^{(t)})\|^2 \\ &\overset{(8.9)}{\leq} \|x^{(t)} - x^*\|^2 - 2\gamma (f(x^{(t)}) - f(x^*) + \frac{m}{2}\|x^* - x^{(t)}\|^2) + \gamma^2 \|\nabla f(x^{(t)})\|^2 \\ &= \|x^{(t)} - x^*\|^2 (1 - \gamma m) - 2\gamma f(x^{(t)}) + \gamma^2 \|\nabla f(x^{(t)})\|^2 \\ &\overset{(8.10)}{\leq} \|x^{(t)} - x^*\|^2 (1 - \gamma m) - 2\gamma f(x^{(t)}) + \gamma^2 \|\nabla f(x^{(t)})\|^2 - 2\gamma f(x^{(t)})(1 - \gamma L) \\ &\leq \|x^{(t)} - x^*\|^2 (1 - \gamma m) \end{aligned}$$

whenever $\gamma < \frac{1}{L}$.

Convergence of objective function. Before proving the convergence of objective funciton, we note another fact about gradient descent.

Fact 3: The m-strongly convexity implies $\|\nabla f(x)\| \geq m\|x - x^*\|$ and the L-Lipscthitz gradient implies $f(x) \leq \frac{L}{2}\|x - x^*\|^2$. Thus,

$$\|\nabla f(x)\|^2 \geq m^2\|x - x^*\|^2 \geq \frac{2m^2}{L} f(x). \tag{8.11}$$

Using the fact that the gradient is L-Lipschitz, we have

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

Thus,

$$\begin{aligned} f(x^{(t+1)}) - f(x^{(t)}) &= f(x^{(t)} - \gamma \nabla f(x^{(t)})) - f(x^{(t)}) \\ &\leq -\gamma \|\nabla f(x^{(t)})\|^2 + \frac{L\gamma^2}{2}\|\nabla f(x^{(t)})\|^2 \\ &= -\gamma\left(1 - \frac{L\gamma}{2}\right)\|\nabla f(x^{(t)})\|^2 \\ &\overset{(8.11)}{\leq} -\gamma\left(1 - \frac{L\gamma}{2}\right)\frac{2m^2}{L} f(x^{(t)}). \end{aligned}$$

Thus, after rearrangements,

$$\begin{aligned} f(x^{(t+1)}) &\leq f(x^{(t)})\left(1 - 2\gamma\frac{m^2}{L}\left(1 - \frac{L\gamma}{2}\right)\right) \\ &= f(x^{(t)})\left(1 - \frac{2m^2}{L}\gamma + m^2\gamma^2\right) \\ &= f(x^{(t)})\left(1 - \frac{m^2}{L^2} + m^2\left(\gamma - \frac{1}{L}\right)^2\right) \end{aligned}$$

By telescoping, we conclude

$$f(x^{(t)}) = f(x^{(0)})\left(1 - \frac{m^2}{L^2} + m^2\left(\gamma - \frac{1}{L}\right)^2\right)^t,$$

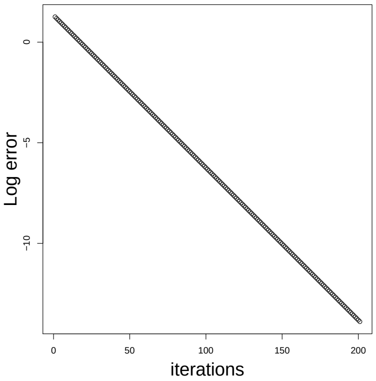
which completes the proof.

■

The key improvement of being strongly convex is the *Fact 3*, which provides a lower bound on the gradient through distance $\|x - x^*\|$. This allows us to establish a quantity like $f(x^{(t+1)}) - f^* = \delta_{t+1} \leq (1-\omega)\delta_t$ (ω is some constant), which leads to a linear (exponential) convergence. If f is only convex, then we can only bound the gradient from below using equation (8.8), which leads to $\delta_{t+1} \leq \delta_t(1 - \eta\delta_t)$, so we can only establish a polynomial convergence.

With the strongly convex assumption, the convergence rate is $O((1 - m/L)^t)$, exponentially fast! Thus, finding an ϵ-suboptimal point only takes $O(\log(1/\epsilon))$ iterations. The quantity L/m is called the *condition number*. A higher condition number implies a slower convergence rate.

When the convergence rate is an exponential function with respect to t, we say that this algorithm has a *linear convergence* rate. This is because it looks like a linear on a semi-log plot (log error versus iterations). The following is the log error versus iterations in the previous example:



Finally, we conclude this by introducing an interesting result about the limitation of convergence rate. The *first-order method* is the collection of iterative methods for updating $x^{(t)}$ using the value of $f(x)$ and its gradients $\nabla f(x)$. The gradient descent is a first-order method.

Theorem 8.5 (Nesterov) *For any $t \leq (d-1)/2$ and any starting point $x^{(0)}$, there exists f , a convex, differentiable function with Lipschitz gradient, such that any first-order method satisfies*

$$f(x^{(t)}) - f^* \geq \frac{3L\|x^{(0)} - x^*\|_2^2}{32(t+1)^2}.$$

Namely, the convergence rate cannot be faster than $O(1/t^2)$. Note that the gradient descent approach achieve the exponential rate due to the strongly convex assumption. This lower bound can be achieved by using the Nesterov accelerated gradient descent algorithm[4], a modification from the gradient descent algorithm.

Remark.

- Similar to the EM algorithm, the gradient descent also suffers from local minima. Note that when we assume f to be a convex function, there is only one unique minimum. If f is non-convex and has multiple local minima, one has to choose the initial point multiple times and apply the gradient descent algorithm with different starting points to increase the chance of obtaining the global minimum.
- If you are interested in the proof or related topics, I would recommend reading the followings:
  - http://www.stat.cmu.edu/~ryantibs/convexopt/
  - https://perso.telecom-paristech.fr/rgower/pdf/M2_statistique_optimisation/grad_conv.pdf
  - *Introductory Lectures on Convex Programming, Volume I: Basic course* by Yu. Nesterov.

8.3.2 A continuous limit view

In analyzing property of gradient descent, often we can consider the continuous limit of the gradient descent. When the step size $\gamma \approx 0$, the gradient descent behaves like the a gradient flow $\eta_x(t)$ with

$$\eta_x(0) = x, \quad \frac{\partial}{\partial t}\eta_x(t) = -\nabla f(\eta_x(t))$$

[4] See http://mitliagkas.github.io/ift6085/ift-6085-lecture-6-notes.pdf

and $x = x^{(0)}$ is the initial point. Thus, analyzing the property of such a gradient flow gives us insight about the property of gradient descent. This idea is particularly useful when the function f is non-convex and may contain multiple critical points[5]. A very important assumption to obtain good gradient flow is that f is a Morse function, meaning that all critical points are well-separated.

The analysis on the continuous-time gradient flow gives us some insight on the gradient descent/ascent algorithm. The study of the continuous-time gradient flow is a common topic in dynamical systems, a branch in applied mathematics. For a gradient system, the collection $\mathcal{C} = \{x : \nabla f(x) = 0\}$ is called *stationary points* or *critical points*. As can be seen easily, the gradient flow will stop when it reaches any point within $\mathcal{C}$. So for any element of $\mathcal{C}$, we can define its *basin of attraction*:

$$D(c) = \{x : \eta_x(\infty)\}$$

Interestingly, this implies that the gradient decent may not reach a local maximum–it may end up being at a saddle point or a local minimum. Here are two interesting properties characterizing the gradient flow and stationary points.

Theorem 8.6 *Let $\eta_x(\infty) = \lim_{t\to\infty} \eta_x(t) \in \mathcal{C}$ be the destination of a gradient descent flow starting at x. Let $(\lambda^+_{\min}(x), v^+_{\min}(x))$ be the smallest positive eigenvalue and its eigenvector of $\nabla\nabla f(x)$.*

- *Terminal flow: If $\lambda^+_{\min}(\eta_x(\infty))$ is unique, then the terminal direction of the gradient flow*

$$\lim_{t\to\infty} \frac{\eta_x(\infty) - \eta_x(t)}{\|\eta_x(\infty) - \eta_x(t)\|} = -v^+_{\min}(\eta_x(\infty)).$$

- *Stable manifold theorem: If f is a Morse function (all critical points are well-separated), the basin of attraction $D(c)$ forms a k-dimensional manifold if there are k positive eigenvalues of $\nabla\nabla f(c)$.*

8.3.3 Connections to common statistical problems

Recall that in the likelihood inference, we are often interested in finding the MLE

$$\widehat{\theta} = \mathsf{argmax}_\theta \ell(\theta|X_1, \cdots, X_n),$$

where $\ell(\theta|X_1, \cdots, X_n) = \frac{1}{n}\sum_{i=1}^n \ell(\theta|X_i)$ is the log-likelihood function.

As you can see, the log-likelihood function behaves like a sample mean. In fact, it can be viewed as an estimator of the *population likelihood function*

$$\ell(\theta) = \mathbb{E}(\ell(\theta|X_i))$$

and the MLE is estimating the *population MLE*

$$\theta^* = \mathsf{argmax}_\theta \ell(\theta).$$

In establishing the asymptotic normality of the MLE, we often assume that the second derivative (known as the Hessian matrix) $H(\theta) = \nabla\nabla\ell(\theta)$ exists and is smooth so that we can apply a Taylor expansion to show that

$$\sqrt{n}(\widehat{\theta} - \theta^*) \approx N(0, \Sigma),$$

[5] see, e.g., https://arxiv.org/pdf/1602.04915.pdf and https://arxiv.org/abs/1807.04431.

where

$$\Sigma = H^{-1}(\theta^*)I(\theta^*)H^{-1}(\theta^*), \quad I(\theta) = \mathbb{E}(\nabla\ell(\theta)\nabla\ell(\theta)^T).$$

We can estimate Σ via a plug-in estimate

$$\widehat{\Sigma} = \widehat{H}^{-1}(\widehat{\theta})\widehat{I}(\widehat{\theta})\widehat{H}^{-1}(\widehat{\theta}), \quad \widehat{H}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\nabla\nabla\ell(\theta|X_i), \quad \widehat{I}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\nabla\ell(\theta|X_i)\nabla\ell(\theta|X_i)^T$$

and use it to construct a confidence interval. The estimator $\widehat{\Sigma}$ is known as the sandwich estimator and it is a consistent estimator of Σ if $H(\theta)$ is smooth. Note that you may find Σ somewhat different from the conventional form. This is because the current form of Σ *does not require the parametric model to be correct.* If the parametric model is correct, Σ will reduce to $I^{-1}(\theta^*)$, the usual Fisher's information matrix.

Here is an interesting relation from the asymptotic normality to the optimization problem we just described. To establish the asymptotic normality, clearly we need the matrix $H(\theta^*)$ to be invertible and it has to be smooth. The invertibility of a matrix is related to its eigenvalues– a matrix is invertible if and only if its eigenvalues are non-zero. In particular, since θ^* is the maximizer, we need all eigenvalues of $H(\theta^*)$ to be negative. This is essentially saying that *we need $H(\theta)$ to be be strongly concave at the neighborhood of θ^** to establish the asymptotic normality and validity of a normal confidence interval.

To sum up, we have the following beautiful theorems connecting statistical validity and optimization feasibility.

Theorem 8.7 *Suppose that the (population) likelihood function $\ell(\theta)$ satisfies the following conditions:*

- *θ^* is the unique maximizer of $\ell(\theta)$ in the interior of the parameter space Θ.*
- *$H(\theta) = \nabla\nabla\ell(\theta)$ is smooth.*
- *all eigenvalues of $H(\theta^*)$ are negative.*

Suppose that we apply a gradient ascent algorithm to find the MLE:

$$\widehat{\theta}^{(t+1)} = \widehat{\theta}^{(t)} + \gamma\nabla\ell(\widehat{\theta}^{(t)}|X_1, \cdots, X_n).$$

Then we have the following statistical validity:

- *Consistency of the MLE: $\widehat{\theta} \xrightarrow{P} \theta^*$.*
- *Asymptotic normality of the MLE: $\sqrt{n}(\widehat{\theta} - \theta^*) \xrightarrow{d} N(0, \Sigma)$.*
- *Validity of sandwich estimator: $\widehat{\Sigma} \xrightarrow{P} \Sigma$.*

as well as the following optimization feasibility: there exists $r_0 > 0$ such that

- *Validity of the gradient ascent: if $\widehat{\theta}^{(0)} \in B(\theta^*, r_0)$, then $\widehat{\theta}^{(\infty)} = \lim_{t\to\infty}\widehat{\theta}^{(t)} = \widehat{\theta}$ when the step size γ is sufficiently small.*
- *Linear convergence of gradient ascent: $\|\widehat{\theta}^{(t)} - \widehat{\theta}\| \leq (1-\omega)^{t/2}\|\widehat{\theta}^{(0)} - \widehat{\theta}\|$ for some $\omega \in (0, 1)$.*

The beauty of the above theorem is that it provides a direction association on statistical stability and algorithmic convergence. Note that in the above theorem, we only require the local strongly concavity of $\ell(\theta)$ around θ^*. If the entire likelihood function is strongly concave, then the requirement $\theta^{(0} \in B(\theta^*, r_0)$ can be relaxed to $\theta^{(0} \in \Theta$–any initial point will work. In fact, a similar theorem can be applied to other statistical estimator as long as it can be written in the form of 'maximizing some sample mean objective function'. A class of this type of statistical problems is called *M-estimator*.

In a sense, Theorem 8.7 establishes the computability of Frequentist inference: we can find the MLE using the gradient ascent method under proper conditions. This, together with what we have learned in the rejection sampling that we draw a proposal from the prior distribution and reject it with the probability $P(\text{Reject}) = \frac{L(\theta|X_1,\cdots,X_n)}{L(\hat{\theta}|X_1,\cdots,X_n)}$, implies computability of a Bayesian inference: we can draw from the posterior distribution by the rejection sampling with the computed MLE. This allows us to handle problems where there is no closed-form MLE (one famous example is the logistic regression).

8.4 Stochastic Gradient Descent/Ascent

The stochastic gradient descent (SGD) is a popular approach to perform minimization of a function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$. There are many concrete example of such case; for instance, to find the MLE, we are trying to minimize the negative log-likelihood function (here we define ℓ_n as the log-likelihood function divided it by n):

$$-\ell_n(\theta) = -\frac{1}{n}\log\prod_{i=1}^n p(X_i;\theta) = -\frac{1}{n}\sum_{i=1}^n \log p(X_i;\theta) = -\frac{1}{n}\sum_{i=1}^n \ell(\theta|X_i).$$

The traditional gradient descent will start with an initial point $\theta^{(0)}$ and update it according to

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \gamma_t \nabla \ell_n(\theta^{(t)}).$$

Note that here we allow the step size $\gamma = \gamma_t$ to be dependent on the iteration.

Let $x^{(0)}$ be the initial point and γ_t be a sequence of step size. To minimizes the function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$, the **stochastic gradient descent (SGD)** uses the following update

$$x^{(t+1)} \leftarrow x^{(t)} + \gamma_t \nabla f_{i_t}(x^{(t)}),$$

where $i_t \in \{1, 2, \cdots, n\}$ is a chosen index at iteration t. Two common rules for choosing i_t are

- Randomized rule: i_t is a uniform distribution over $\{1, 2, \cdots, n\}$.
- Cyclic rule: choose i_t as $1, 2, 3, \cdots, m, 1, 2, 3, \cdots$.

Randomized rule is more popular in practice. The idea of randomized rule is that

$$\mathbb{E}(\nabla f_{i_k}(x)) = \nabla f(x),$$

i.e., each SGD update is an unbiased estimate of the gradient.

The trajectory $\{x^{(0)}, x^{(1)}, \cdots, \}$ forms a Markov chain. Moreover, if we fix the step size $\gamma_t = \gamma > 0$, this Markov chain is homogeneous.

The main appeals of SGD are (i) the iteration cost is independent of n, the number of functions (in statistics, it is often the sample size), and (ii) it may provide a big saving in terms of memory useage (no need to read the entire dataset). Thus, the SGD is powerful when we have a large-scale optimization problem. Specifically, assume that computing the function $f_i(x)$ requires one unit of computational costs. Then, one update of traditional gradient descent algorithm requires $O(n)$ costs whereas one iteration of SGD only requires $O(1)$ costs.

If we keep the step size γ to be the same across iterations, the SGD will not converge (it forms a Markov chain so at the end it will behave like a random point from the stationary distribution). Thus, to ensure SGD converge, a standard approach is to choose diminishing step sizes such as $\gamma_t = 1/t$.

Convergence rate. Recall that when f has a Lipschitz gradient, traditional gradient descent has a convergence rate $O(1/t)$. In the case of SGD, however, the convergence rate is $O(1/\sqrt{t})$[6]. More formally,

$$\mathbb{E}(f(x^{(t)})) - f^* = O(1/\sqrt{t}).$$

Thus, SGD has a slower convergence rate. Moreover, when f is strongly convex with a Lipschitz gradient, the gradient descent is of the rate $O(c^t)$ for some $c < 1$ but under the same condition, SGD gives a rate $O(1/t)$. Although it is cheap to evaluate one iteration of SGD, the convergence rate is a lot slower than the traditional gradient descent.

Does this imply that SGD is not useful? Not at all! Consider the case where the function f has a Lipschitz gradient and our goal is to find a minimizer that is at least 10^{-3} close to the optimal. Assume that f is the average of $n = 10^6$ function (loosely speaking, the sample size is 10^6). How much cost will it take for gradient descent and SGD to achieve this accuracy? For gradient descent, the convergence rate is $O(1/t)$, so $O(1/t) = 10^{-3} \Rightarrow t = O(10^3)$, which implies that we need $O(10^3)$ iterations. Each iteration costs $O(n) = O(10^6)$ computational budget so the total cost is $O(nt) = O(10^9)$. For SGD, the convergence rate is $O(1/\sqrt{t})$. Thus, we need $t = O(10^6)$ iterations. However, each iteration only cost $O(1)$ budget so the total cost is $O(t) = O(10^6)$, a 1000-times saving from traditional gradient descent!

This property has made SGD very popular in large-scale computation especially in the modern big data era. A sample of size $n = 10^6$ is considered as big data set nowadays but as you can see, the larger the dataset, the more benefit SGD has over traditional gradient descent.

Mini-batch approach. A popular variant of SGD is the mini-batch approach. Instead of randomly choose one function, we randomly choose b functions and use the gradient of the average as the update. In more details, let $I_t \subset \{1, 2, \cdots, n\}$ be a set of size $|I_t| = b << m$. We then use the following update:

$$x^{(t+1)} \leftarrow x^{(t)} - \gamma_t \frac{1}{b} \sum_{i \in I_t} f_i(x^{(t)}).$$

Averaged SGD. A new approach in SGD is the averaged SGD approach. The idea is very simple. After running the SGD and observing the trajectory $\{x^0, x^1, \cdots, x^T\}$, instead of using the last point as the output minimizer, we use the average of the last few points as the minimizer. Namely, we choose a number τ and then use

$$\tilde{x}_T = \frac{1}{\tau} \sum_{k=0}^{\tau-1} x^{(T-k)}$$

as the minimizer. This idea has been shown to enjoy both a fast convergence rate and a low computational cost[7].

[6] see, .e.g., Nemirosvki et al. (2009) *Robust stochastic optimization approach to stochastic programming.*

[7] See, e.g., https://www.di.ens.fr/~fbach/colt_2018_pillaud_vivien.pdf and http://auai.org/uai2018/proceedings/papers/71.pdf.

8.4.1 Additive noise approach

There is another set of SGD approach that uses the idea of additive noise. This is called the *perturbed gradient descent*[8]. Let f be the function that we wish to minimize. Starting with an initial point $x^{(0)}$ and a sequence of step sizes γ_t, we use the following update:

$$x^{(t+1)} \leftarrow x^{(t)} - \gamma_t \nabla f(x^{(t)}) + \gamma_t Z_t,$$

where $Z_1, Z_2, \cdots,$ are IID mean 0 random noises (often we choose them to be multivariate normal). This approach does not save computational cost (because we still evaluate a full gradient) but it may escape from (shallow) local minima due to the additive noise Z_t.

Note that a similar idea to the additive noise is the stochastic gradient Langevin dynamics (SGLD)[9], which uses

$$x^{(t+1)} \leftarrow x^{(t)} - \gamma_t \nabla f(x^{(t)}) + \sqrt{\gamma_t} Z_t,$$

where Z_t is a Gaussian vector with identity covariance matrix. When the step size $\gamma_t = \gamma$ is fixed, SGLD forms a Markov chain with a stationary distribution $\propto e^{-\gamma f(x)}$. Thus, SGLD is often used as an alternative to MCMC although it may be used as a method to find the minimizer. Note that one can choose $f(x) = \log \pi(\theta|X_1, \cdots, X_n)$ then SGLD allows us to sample from the posterior distribution[10].

[8] See https://arxiv.org/abs/1703.00887v1 for more details.

[9] For applications in Statistics, see http://www.jmlr.org/papers/volume17/teh16a/teh16a.pdf

[10] see https://www.ics.uci.edu/~welling/publications/papers/stoclangevin_v6.pdf for more details.