

# Chapter 11

## Stochastic Gradient Descent

### Reading

1. “Stochastic gradient descent tricks” by Bottou (2012). Great paper with lots of little tricks of how to use SGD in practice.
2. Up to Section 4.2 of “Optimization methods for large-scale machine learning” by Bottou et al. (2018). This is advanced material, you do not need to be able to follow it completely.
3. [http://fa.bianp.net/teaching/2018/eecs227at/stochastic\\_gradient.html](http://fa.bianp.net/teaching/2018/eecs227at/stochastic_gradient.html)
4. Stochastic Weight Averaging (SWA) by Izmailov et al. (2018).

Stochastic Gradient Descent (SGD) has its roots in stochastic optimization. A stochastic optimization problem looks like

$$w^* = \underset{w}{\operatorname{argmin}} \mathbb{E}_{\xi}[\ell(w, \xi)] \quad (11.1)$$

where  $\xi$  is a random variable. This is a very old and rich area, there was lots of action in it already in the 1950s, e.g., (Kushner and Yin, 2003; Robbins and Monro, 1951). It is also a highly relevant problem: for instance, when a plane goes from Los Angeles to Philadelphia, the route that the plane takes depends on the local weather conditions along its path and airlines will optimize this route using a stochastic optimization problem of the above form. The variable  $w$  will be the trajectory of the plane and  $\xi$  are the weather conditions which we do not know exactly but may perhaps have estimated a distribution for them. Such problems are very common in other fields like operations research, e.g., optimizing the time at which an Amazon package arrives with various disturbances such as delays in shipping, missing inventory in the warehouse etc.

In machine learning, we are interested in solving a slightly different problem called the finite-sum problem. Given a finite dataset  $D = \{(x^i, y^i)\}_{i=1, \dots, n}$

19 we minimize

$$\ell(w) := \frac{1}{n} \sum_{i=1}^n \ell^i(w) \quad (11.2)$$

20 where we will use the shorthand

$$\ell^i(w) := \ell(w; x^i, y^i)$$

21 to denote the loss on the datum  $(x^i, y^i)$  with weights  $w$ . Essentially, the  
 22 random variable  $\xi$  in (11.1) represents the samples in the training dataset;  
 23 with important differences being that neither do we know anything about the  
 24 distribution of the input data, nor do we have an infinite number of samples.

25 It is difficult to do gradient descent if the number of samples  $n$  is large  
 26 because the gradient is a summation of a large number of terms

$$\nabla \ell(w) = \frac{1}{n} \sum_{i=1}^n \nabla \ell^i(w).$$

27 If the mini-batch size is 1, i.e., at each iteration we sample one of the training  
 28 sample denoted by

$$\omega_t \in \{1, \dots, n\}$$

29 we update the weights using

$$w^{t+1} = w^t - \eta \nabla \ell^{\omega_t}(w^t). \quad (11.3)$$

30 For a larger mini-batch for size  $\ell$  let us denote the samples in the mini-batch  
 31 by

$$\{(x^{\omega_t^1}, y^{\omega_t^1}), \dots, (x^{\omega_t^\ell}, y^{\omega_t^\ell})\}$$

32 where each  $\omega_t^k \in \{1, \dots, n\}$  is the index chosen uniformly randomly from  
 33 the training dataset. We will choose these indices with replacement (analyzing  
 34 SGD for sampling without replacement is quite hard). The gradient on this  
 35 sampled mini-batch is denoted by

$$\nabla \ell_\ell(w) := \frac{1}{\ell} \sum_{i=1}^{\ell} \nabla \ell^{\omega_t^i}(w) \quad (11.4)$$

36 and we update the weights as usual using

$$w^{t+1} = w^t - \eta \nabla \ell_\ell(w^t).$$

37 If  $\ell = 1$ , we will denote the gradient by  $\nabla \ell_\omega$  to keep the notation clear.

38 **What is an epoch in PyTorch?** We will not think of epochs when we de-  
 39 velop the theory for SGD. Epochs is a construct introduced in the deep learning  
 40 libraries for bookkeeping purposes. It also ensures that if Algorithm A obtains  
 41 so and so training/validation error after 100 epochs, it can be compared directly  
 42 with Algorithm B which obtains the same training/validation error after, say,  
 43 120 epochs, e.g., one can say Algorithm A is faster than Algorithm B at training  
 44 a network. Instead of sampling a mini-batch of data uniformly randomly with  
 45 replacement, PyTorch shuffles the entire training set at the beginning of each  
 46 epoch and samples the mini-batch *with replacement* during each epoch. This

is reasonable but there will be some discrepancies in the performance of SGD as predicted by theory and obtained by PyTorch on deep networks, especially if the mini-batch size is large.

Although we will not discuss this, SGD using mini-batches sampled with replacement is faster than with mini-batches sampled without replacement (Recht and Ré, 2012).

## 11.1 SGD for least-squares regression

Let us understand SGD for one dimensional least-squares, our data and targets are  $x^i, y^i \in \mathbb{R}$  and the objective is

$$\ell(w) = \frac{1}{2n} \sum_{i=1}^n (x^i w - y^i)^2 \quad (11.5)$$

for the weights  $w \in \mathbb{R}$ . Notice that the objective is a sum of  $n$  different quadratics, each quadratic is minimized by *different* weights

$$w^i := \frac{y^i}{x^i};$$

in other words, each sample in the training dataset would like the weight to be  $y^i/x^i$  to minimize its residual and the least-squares objective which sums up their individual residuals forces them to make trade-offs. Focus on two quantities

$$w_{\min} = \min_i \{w^i\}, \quad w_{\max} = \max_i \{w^i\}.$$

Notice that the interval  $(-\infty, w_{\max})$  is the region where the descent direction on any sample in the dataset moves the weights  $w^t$  to the right. The interval  $(w_{\max}, \infty)$  is the region where the descent direction on any sample moves the weights to the left. If weights are initialized in the latter region,  $w^0 \gg \max_i w^i$ , successive iterations of SGD will quickly bring the weights to

$$w^t \in (w_{\min}, w_{\max}) \quad (11.6)$$

which we will call the “zone of confusion”. Similarly, if weights are initialized  $w^0 \ll w_{\min}$ , they will move right until iterates reach the zone of confusion.

After  $w^t \in (w_{\min}, w_{\max})$ , there is no real convergence of the weights,

**▲** Draw the objective here for different values of  $w^i$  and understand how SGD works for this problem.

if the learning rate  $\eta$  is fixed, since the samples  $\omega_t$  are sampled uniformly randomly, depending upon which sample was chosen to compute the gradient the weights move to the right or the left and therefore keep shuttling back and forth in this region.

Notice that the objective in (11.5) is convex because it is the sum of convex functions so there is a unique global minimum namely

$$w^* = \frac{\sum_{i=1}^n x^i y^i}{\sum_{i=1}^n (x^i)^2}.$$

If one were to execute gradient descent on this same problem  $w^{t+1} = w^t - \eta \nabla \ell(w^t)$ , we will converge to this value. But since SGD samples a different sample at each iteration, SGD never converges, it remains in this large zone  $(w_{\min}, w_{\max})$ .

## 11.2 Convergence of SGD

If the learning rate is large, SGD makes quick progress outside the zone of confusion but bounces around a lot inside the zone of confusion. If the learning rate is too small, SGD is slow outside the zone of confusion but does not bounce around too much inside the zone. You can explore how the learning rate changes the dynamics of SGD at

[http://fa.bianp.net/teaching/2018/eecs227at/stochastic\\_gradient.html](http://fa.bianp.net/teaching/2018/eecs227at/stochastic_gradient.html).

In this section, we will study under what conditions SGD converges to the global minimum and how the learning rate of SGD should be reduced to make it converge quickly. We will first analyze SGD with mini-batch size of 1.

**Strongly convex functions** The proofs for convex functions are tedious, so we will only consider strongly convex functions in this section. As usual the strong convexity parameter is  $m$  and smoothness parameter is  $L$ . One key thing to notice that these constants  $L, m$  refer to the full objective, i.e.,

$$\|\nabla \ell(w) - \nabla \ell(w')\| \leq L\|w - w'\|$$

and

$$\ell(w) - \frac{m}{2}\|w\|^2 \text{ is convex.}$$

Here  $\ell(w)$  is the *full objective*

$$\ell(w) = \frac{1}{n} \sum_{i=1}^n \ell^i(w).$$

**What is the appropriate notion of convergence?** The key difference between updates of SGD and those of GD is that SGD updates also depend on the random variable  $\omega_t$ . The iterate  $w_t$  is a *random variable* and therefore instead of simply bounding the gap  $\ell(w^t) - \ell(w^*)$  we will have to obtain an upper bound for

$$\mathbb{E}_{w^t} [\ell(w^t)] - \ell(w^*).$$

Similar to the case of SGD, let us construct a descent lemma for one iteration of SGD update.

**Lemma 1 (Descent Lemma for SGD).**

$$\begin{aligned} \mathbb{E}_{\omega_t} [\ell(w^{t+1}) - \ell(w^t) \mid w^t] &\leq -\eta \left\langle \nabla \ell(w^t), \mathbb{E}_{\omega_t} [\nabla \ell^{\omega_t}(w^t)] \right\rangle \\ &\quad + \frac{L\eta^2}{2} \mathbb{E}_{\omega_t} [\|\nabla \ell^{\omega_t}(w^t)\|^2]. \end{aligned} \quad (11.7)$$

**Proof.** First, compare this with the descent lemma for gradient descent (if we substitute  $w^{t+1} - w^t = -\eta \nabla \ell(w^t)$  from Chapter 9)

$$\ell(w^{t+1}) - \ell(w^t) \leq -\eta \langle \nabla \ell(w^t), \nabla \ell(w^t) \rangle + \frac{L\eta^2}{2} \|\nabla \ell(w^t)\|^2$$

The only difference now is that in the case of SGD we have

$$w^{t+1} - w^t = -\eta \nabla \ell^{\omega_t}(w^t).$$

The most important difference however is that the descent term, namely the left-hand side in (11.7) is conditioned on the random variable  $w^t$ . The proof of this lemma is easy, we simply substitute the expression for the weight updates of SGD and take an expectation over the index of datum sampled by SGD  $\omega_t$  on both sides of the inequality.  $\square$

The implication of the above lemma is that SGD updates need more refined conditions under which we can claim monotonic progress towards the global minimum. Effectively, we need to make sure that the right-hand side is negative, *always* irrespective of what the value of the random variable  $w^t$  is. We would like to upper bound the right-hand side by a deterministic quantity ideally.

### 11.2.1 Typical assumptions in the analysis of SGD

1. **Stochastic gradients are unbiased.** Assume that the stochastic gradient is unbiased

$$\nabla \ell(w) = \mathbb{E}_{\omega} [\nabla \ell^{\omega}(w)] \quad (11.8)$$

for all  $w$  in the domain. This is akin to assuming that the way we sample images in the mini-batch is such that the average is always pointing towards the true gradient with a similar magnitude. This is a natural condition and will only change if the sampling distribution is not uniform. This assumption allows to control the first term in the descent lemma.

2. **Second moment of gradient norm does not grow too quickly.** We will assume that there exist scalars  $\sigma_0$  and  $\sigma$  such that

$$\mathbb{E}_{\omega_t} [\|\nabla \ell^{\omega}(w)\|^2] \leq \sigma_0 + \sigma \|\nabla \ell(w)\|^2. \quad (11.9)$$

This assumption allows to control the second term in the descent lemma for SGD. It assumes that the stochastic estimate of the gradient in SGD  $\nabla \ell^{\omega_t}(w)$  is not too different than the full gradient  $\nabla \ell(w^t)$ . In the neighborhood of a critical point (locations where the full gradient  $\nabla \ell(w) = 0$ ), the stochastic gradient is allowed to grow in a similar fashion as the true gradient except with a scaling factor  $\sigma > 0$  and a constant  $\sigma_0$ .

Let us see how the descent lemma changes with these additional assumptions.

**Lemma 2 (Descent Lemma for SGD with additional assumptions).** If SGD gradients are unbiased and the second moment of the stochastic gradients can be bounded, we have

$$\begin{aligned}
& \mathbb{E}_{\omega_t} [\ell(w^{t+1}) - \ell(w^t) \mid w^t] \\
& \leq -\eta \left\langle \nabla \ell(w^t), \mathbb{E}_{\omega_t} [\nabla \ell^{\omega_t}(w^t)] \right\rangle + \frac{L\eta^2}{2} \mathbb{E}_{\omega_t} [\|\nabla \ell^{\omega_t}(w^t)\|^2] \\
& \leq -\eta \|\nabla \ell(w^t)\|^2 + \frac{L\eta^2}{2} \mathbb{E}_{\omega_t} [\|\nabla \ell^{\omega_t}(w^t)\|^2] \\
& = -\eta \left(1 - \frac{\eta L \sigma}{2}\right) \|\nabla \ell(w^t)\|^2 + \frac{\eta^2 L \sigma_0}{2}.
\end{aligned} \tag{11.10}$$

The proof is given in (11.10) itself. Compare this to the corresponding result we have derived for gradient descent in Chapter 9

$$\ell(w^{t+1}) - \ell(w^t) \leq -\frac{\eta}{2} \|\nabla \ell(w^t)\|^2.$$

In addition to the negative term  $-\frac{\eta}{2} \|\nabla \ell(w^t)\|^2$ , we have two additional positive terms

$$\frac{\eta^2 L \sigma}{2} \|\nabla \ell(w^t)\|^2 + \frac{\eta^2 L \sigma_0}{2};$$

this indicates that depending upon the magnitude of these terms we may not get monotonic improvement of the objective for SGD. There is no such concern for gradient descent, we get monotonic progress at all parts of the domain.

We need to pick the learning rate  $\eta$  in such a way that balances the the right-hand side of (11.10) and makes it negative.

## 11.2.2 Convergence rate of SGD for strongly-convex functions

**Theorem 3 (Optimality gap for SGD).** If we pick a step-size

$$\eta \leq \frac{1}{L\sigma}$$

for  $m$ -strongly convex and  $L$ -smooth function  $\ell(w)$  then the expected optimality gap satisfies

$$\begin{aligned}
& \mathbb{E}_{\omega_1, \omega_2, \dots, \omega_t} [\ell(w^{t+1})] - \ell(w^*) \\
& \leq \frac{\eta L \sigma_0}{2m} + (1 - \eta m)^t \left( \ell(w^0) - \ell(w^*) - \frac{\eta L \sigma_0}{2m} \right).
\end{aligned} \tag{11.11}$$

We will not cover the proof of this theorem, it is a direct application of the descent lemma. See Bottou et al. (2018, Theorem 4.6) for an elaborate proof.

This theorem beautiful demonstrates the interplay between the step-size and the variance of SGD gradients. If there is no stochasticity, i.e.,  $\sigma_0 = 0$

and  $\sigma = 1$ , we get the same result as that of gradient descent, namely, the function value  $\ell(w^{t+1})$  converges at a *linear rate*  $(1 - \alpha m)^t$ . Some points to notice

1. The random variable  $w^{t+1}$  depends upon all the indices  $\omega_1, \omega_2, \dots, \omega_t$  that were sampled during updates of SGD and therefore the expectation in (11.11) should be over all these random variables.
2. When the stochastic gradient is noisy, we have a non-zero  $\sigma_0$  we can no longer get to the global minimum, there is a first term which does not decay with time.
3. If we pick a small  $\eta$ , we get closer to the global minimum but go there quite slowly. On the other hand, we can pick a large  $\eta$  and get to a neighborhood of the global minimum quickly but we will then have a large error leftover at the end.

How can we make SGD converge and drive down the first term in (11.11) to zero? A simple trick is to reduce the learning rate  $\eta$  with time. We do not want to decay the learning rate too quickly however because the second term in (11.11) is small, i.e., optimality gap is reduced quickly by its multiplicative nature, for a large value of the learning rate. A good schedule to pick is

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \text{and} \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty. \quad (11.12)$$

**Heuristic for training neural networks** The two terms in the convergence rate of SGD explain the widely used heuristic of “divide the learning rate by some constant” if the training error seems plateaued. We are reducing the size of the ball in which SGD iterates bounce around by doing so.

**Theorem 4 (Convergence rate of SGD for decaying step-size).** For a schedule

$$\eta_t = \frac{\beta}{t + t_0} \text{ where } \beta > \frac{1}{m} \text{ and } t_0 \text{ is such that } \eta_1 < \frac{1}{L\sigma}$$

then the expected optimality gap satisfies

$$\mathbb{E}_{\omega_1, \omega_2, \dots, \omega_t} [\ell(w^{t+1})] - \ell(w^*) = \mathcal{O}\left(\frac{1}{t + t_0}\right). \quad (11.13)$$

We will not do the proof. If you are interested, see Bottou et al. (2018, Theorem 4.7). Compare this to the convergence rate of  $\mathcal{O}(\kappa \log(1/\epsilon))$  for gradient descent for strongly-convex functions. Notice that we converge only at a sub-linear rate for SGD even for strongly convex loss functions. SGD is a much slower algorithm than GD.

**Convergence rate for mini-batch SGD** The mini-batch gradient  $\nabla \ell_b(w)$  is still an unbiased estimate of the full-gradient

$$\mathbb{E}_b [\nabla \ell_b(w)] = \nabla \ell(w)$$

but the second assumption in SGD improves a bit. Since the mini-batch gradient is averaged over  $\ell$  samples we have

$$\mathbb{E}_{\ell} [\|\nabla \ell_{\ell}(w)\|^2] \leq \frac{\sigma_0}{\ell} + \frac{\sigma}{\ell} \|\nabla \ell(w)\|^2$$

if  $\sigma_0, \sigma$  were the constants in (11.9). This changes the convergence rate in Theorem 3 to

$$\begin{aligned} & \mathbb{E}_{\omega_1, \omega_2, \dots, \omega_t} [\ell(w^{t+1})] - \ell(w^*) \\ & \leq \frac{\eta L \sigma_0}{2m\ell} + (1 - \eta m)^t \left( \ell(w^0) - \ell(w^*) - \frac{\eta L \sigma_0}{2m\ell} \right). \end{aligned} \quad (11.14)$$

Note that the maximum learning rate in Theorem 3 is inversely proportional to  $\sigma$  so we can also pick a larger learning rate  $\eta < \frac{\ell}{L\sigma}$ . If we do so, the first and last terms above are not affected by the batch-size but multiplicative term  $(1 - \eta m)^t$  is. Since

$$(1 - \eta m)^t \leq e^{-tm\eta}$$

we see that increasing the learning rate by a factor of  $\ell$  will reduce the number of iterations required to reach the zone of confusion by a factor of  $\ell$ . Of course, this comes with the caveat that each iteration also requires  $\mathcal{O}(\ell)$  more computation to compute the gradient compared to single-sample SGD.

### 11.2.3 When should one use SGD in place of GD?

Theorem 4 indicates that SGD is a very slow algorithm, GD is much faster than SGD to minimize strongly convex functions. This gap also exists if we do not have strong convexity: we did not prove this but SGD requires  $\mathcal{O}(1/\epsilon^2)$  to reach an  $\epsilon$ -neighborhood of the global optimum for convex functions whereas GD requires a much smaller  $\mathcal{O}(1/\epsilon)$ . One might wonder why we should use SGD at all.

It is critical to remember that the objective in machine learning is a sum of many terms

$$\ell(w) = \frac{1}{n} \sum_{i=1}^n \ell^i(w)$$

One iteration of SGD requires us to compute only  $\nabla \ell^{\omega_t}(w)$  whereas one update of GD requires us to compute the full gradient  $\nabla \ell(w)$ . One weight update of GD is  $\mathcal{O}(n)$  more expensive than one weight update using SGD. Let us do a back-of-the-envelope computation for convex functions. If we want to reach an  $\epsilon$ -neighborhood of the global minimum of a convex function, we need  $\mathcal{O}(1/\epsilon)$  iterations of GD, which requires

$$\mathcal{O}\left(\frac{n}{\epsilon}\right)$$

operations. SGD needs  $\mathcal{O}(1/\epsilon^2)$  iterations and therefore requires

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$$

operations to reach the  $\epsilon$ -neighborhood. This indicates that if our chosen  $\epsilon$ -ball is

$$\epsilon \lesssim \frac{1}{n}$$

GD requires fewer overall operations. But if  $\epsilon$ -ball is larger than this, we should use SGD because it is computationally cheaper.

SGD is particularly suited to machine learning compared to GD for the following reason. Let  $\epsilon^i = \ell^i(w^t) - \ell^i(w^*)$  be the residual on the  $i^{\text{th}}$  datum in the training dataset. Observe that our  $\epsilon$ -neighborhood is

$$\epsilon = \ell(w^t) - \ell(w^*) = \frac{1}{n} \sum_{i=1}^n \epsilon^i.$$

If  $\epsilon^i$  is constant and does not depend on the number of training samples  $n$  (i.e., say we are happy with the average error over the training dataset being 2% even and do not seek a smaller one even if we collect more data) then we should use SGD to train our model because it is cheaper. This is not always the case for other problems, e.g., if you are doing computational tomography (capturing multiple images from a CT-scan machine and trying to reconstruct the heart/lung region in the thoracic cavity), we may seek a more and more accurate answer, i.e., small  $\epsilon$  if we have more data.

## 11.3 Accelerating SGD using momentum

The convergence rate of SGD is quite bad, it is sub-linear. Roughly speaking, the successive iterates of SGD are computed using different mini-batches; the gradient on each such mini-batch is a noisy approximation of the full-gradient on the training dataset (that of GD). This makes the SGD iterates noisy and one may improve the convergence rate of SGD by simply averaging the weights. This leads to a simple technique to accelerate SGD which we discuss next.

**Polyak-Ruppert averaging** Consider the updates

$$\begin{aligned} w^{t+1} &= w^t - \eta_t \nabla \ell_t(w^t) \\ u^t &= \frac{w^0 + w^1 + \dots + w^t}{t}. \end{aligned} \tag{11.15}$$

In a series of papers, [Polyak \(1990\)](#); [Polyak and Juditsky \(1992\)](#); [Ruppert \(1988\)](#) showed that the quantity

$$\mathbb{E}_{\omega_1, \dots, \omega_{t-1}} [\ell(u^t)] - \ell(w^*)$$

converges faster than the quantity

$$\mathbb{E}_{\omega_1, \dots, \omega_{t-1}} [\ell(w^t)] - \ell(w^*);$$

both of these still converge at rate  $\mathcal{O}(1/\epsilon^2)$  but the former has a smaller constant. This is quite surprising and useful: essentially we are still performing mini-batch updates for the weights  $w^t$  but instead of thinking of  $w^t$  as the answer, we think of  $u^t$  as the output of SGD. This averaging of iterates does not change the SGD algorithm. Computing this output requires us to remember all the past iterations  $w^0, \dots, w^t$  but we can easily approximate that step by exponential averaging of the *weights*

$$u^t = \rho u^{t-1} + (1 - \rho) w^t;$$

exponential averaging is likely to achieve the same purpose with a much smaller memory requirement.

Further, this idea of using averaged iterates to speed up stochastic optimization algorithms is quite general and also works for algorithms other than SGD. The paper on Stochastic Weight Averaging by [Izmailov et al. \(2018\)](#) performs weight averaging (with quite different motivations) and works very well in practice.

### 11.3.1 Momentum methods do not accelerate SGD

We saw that momentum is very useful to accelerate the convergence of gradient descent. The power of momentum lies in making faster progress using the inertia of the particle: if the velocity and the current gradient are aligned with each other (as is the case at the beginning of training when the iterates are far from the global optimum) momentum speeds up things. Towards the end of training when gradients are typically mis-aligned with the velocity, we need friction (as in Nesterov's updates) to reduce this effect.

Observe that in SGD, the gradient is *always* incorrect; it is after all only a noisy stochastic approximation of the full gradient on the dataset. Since the velocity  $w^t - w^{t-1}$  was computed using the previous stochastic gradient, there is no reason to believe that this velocity is accurate and will speed up SGD. Here is a very important point ([Kidambi et al., 2018](#); [Liu and Belkin, 2018](#)) that you should remember.

Momentum methods (Polyak's or Nesterov's) do not significantly accelerate SGD.

To be more precise, we saw that for Nesterov's updates in GD for strongly-convex functions we have a result of the form

$$\|w^t - w^*\| \leq e^{-t/\sqrt{\kappa}} \|w^0 - w^*\|$$

while the constant without momentum is larger, it is  $e^{-t/\kappa}$ . This term is directly related to the second term in Theorem 4. The above authors come up with counterexamples to show that Nesterov's updates with SGD only improve this multiplicative term to something like  $e^{-ct/\kappa}$  for some  $c$ ; in other words using Nesterov's updates with SGD only lead to a constant factor improvements in the convergence rate.

Accelerating stochastic optimization algorithms is done via the use of control variates ([Le Roux et al., 2012](#)). Broadly speaking these methods work by using the previous gradients in SGD  $\{\nabla \ell^{\omega_1}(w^1), \dots, \nabla \ell^{\omega_t}(w^t)\}$  to compute some surrogate for the current full gradient  $\nabla \ell(w^t)$  and compute the descent direction using both this surrogate full gradient and the standard SGD gradient.

**Why do we use Nesterov's method to train deep networks?** It is worthwhile to think why we use Nesterov's momentum to train deep networks: (i) we know that momentum does not help speed up training, and (ii) momentum is simply a faster way to minimize the same objective  $\ell$  so it does not have any regularization properties that help generalization either. We do not have a definitive answer to this question yet but here is what we know.

Datasets that we use in deep learning represent quite narrow distributions (natural images of animals, household objects etc.), e.g., for instance the two images below are essentially the same with minor differences in the input distribution



Most weights of a deep network will have a similar gradient for these images as input, the weights for which the gradient will differ are likely to be the weights at the top few layers of the network. This entails that even if the stochastic gradients are computed on different mini-batches, they are essentially quite similar to each other, and thereby to the full-gradient. More precisely, the covariance of mini-batch gradients

$$\text{Cov}(\nabla \ell_{\delta}(w), \nabla \ell_{\delta'}(w)) = \mathbb{E}_{\delta, \delta'} \left[ (\nabla \ell_{\delta}(w) - \nabla \ell(w)) (\nabla \ell_{\delta'}(w) - \nabla \ell(w))^{\top} \right]$$

is a matrix with very few non-zero eigenvalues; only about 0.5% of the eigenvalues are non-zero (Chaudhari and Soatto, 2017) even for large networks. This means that the SGD gradient while training deep networks is essentially the full gradient and we should expect momentum to accelerate convergence in practice.

## 11.4 Understanding SGD as a Markov Chain

The preceding development tells us how SGD works and how many iterations of SGD we need to get within an  $\epsilon$ -neighborhood of the global minimum for convex functions. Things are not this easy to understand for non-convex functions; essentially if we have two minima  $u^*, v^*$

$$\nabla \ell(u^*) = \nabla \ell(v^*) = 0$$

depending upon where GD/SGD are initialized they can converge to different places. In this section, we will look at an alternative way of understanding how SGD works for non-convex functions. The development here will be much more abstract than the preceding section because we want to capture the overall properties of SGD.

### 11.4.1 Gradient flow

Let us first talk about gradient descent. Just like we constructed a model for Nesterov's updates using a differential equation, we will first construct a model for gradient descent using a differential equation. The updates are given by

$$w^{t+1} - w^t = -\eta \nabla \ell(w^t).$$

If we again imagine a continuously differentiable curve  $W(\tau)$  as a model for these discrete-time updates and time

$$d\tau := \eta$$

▲ A non-convex function with two local minima. The one on the left is the global minimum but gradient descent may not always reach here.



we can write a differential equation of the form

$$\frac{dW}{d\tau} = \dot{W}(\tau) = -\nabla \ell(W(\tau)); \quad W(0) = w^0. \quad (11.16)$$

This is called gradient flow. If we wanted to run execute gradient flow on a computer, we can do so using Euler discretization

$$\dot{W}(\tau) \approx \frac{W(\tau + \Delta\tau) - W(\tau)}{\Delta\tau} = -\nabla \ell(W(\tau)).$$

for any value of the time-step  $\Delta\tau$ . If the time-step  $\Delta\tau = \eta$  we get exactly gradient descent. More precisely, gradient flow is the limit of gradient descent as the learning rate  $\eta \rightarrow 0$ . It is important to always remember that gradient flow is a model for GD, not GD itself. Our goal in the remainder of the section is to develop a similar model for SGD.

## 11.4.2 Markov chains

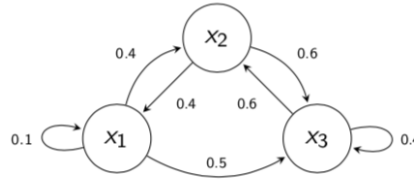
Consider the Whack-The-Mole game: a mole has burrowed a network of three holes  $w^1, w^2, w^3$  into the ground. It keeps going in and out of the holes and we are interested in finding which hole it will show up next so that we can give it a nice whack.

- Three holes:

$$X = \{x_1, x_2, x_3\}.$$

- Transition probabilities:

$$T = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.4 & 0 & 0.6 \\ 0 & 0.6 & 0.4 \end{bmatrix}$$



317

This is an example of a Markov chain. There is a transition matrix  $P$  which determines the probability  $P_{ij}$  of the mole resurfacing on a given hole  $w^j$  given that it resurfaced at hole  $w^i$  the last time. The matrix  $P^t$  is the  $t$ -step transition matrix

$$P_{ij}^t = \mathbb{P}(w^t = w^j \mid w^{(0)} = w^i).$$

If there exist times  $t, t'$  such the both the probabilities

$$\mathbb{P}(w^t = w^j \mid w^{(0)} = w^i) \quad \mathbb{P}(w^{t'} = w^i \mid w^{(0)} = w^j)$$

are non-zero the two states  $w^i$  and  $w^j$  are said to “communicate”

$$w^i \leftrightarrow w^j$$

The set of states in the Markov chain that *all* communicate with each other are an equivalence class. This means that the Markov chain can visit any state from any other state in this equivalence class with non-zero probability, we just might have to wait for a really long time if  $P_{ij}^t \approx 0$  for two states  $w^i, w^j$ . If all the states in the Markov chain belong to the same equivalence class, it is called irreducible. A related concept is that of “positive recurrence”, i.e., if the Markov chain was at a state  $w$  at some time, it comes back to the same

state after some finite time. Since the process is Markov it forgets that is just came back to the same state and therefore positive recurrence also means that if we consider an infinitely long trajectory of a Markov chain, the chain visits the same state infinitely many times along this trajectory. You can see the animations at <https://setosa.io/ev/markov-chains> to build more intuition.

**Invariant distribution of a Markov chain** The probability of being in a state  $w^i$  at time  $t + 1$  can be written as

$$\mathbb{P}(w^{t+1} = w^i) = \sum_{j=1}^N \mathbb{P}(w^{t+1} = w^i \mid w^t = w^j) \mathbb{P}(w^t = w^j).$$

This equation governs how the probabilities  $\mathbb{P}(w^t = w^i)$  change with time  $t$ . Let's do the calculations for the Whack-The-Mole example. Say the mole was at hole  $w^1$  at the beginning. So the probability distribution of its presence

$$\pi^{(t)} = \begin{bmatrix} \mathbb{P}(w^t = w^1) \\ \mathbb{P}(w^t = w^2) \\ \mathbb{P}(w^t = w^3) \end{bmatrix}$$

is such that

$$\pi^1 = [1, 0, 0]^\top.$$

We can now write the above formula as

$$\pi^{(t+1)} = P^\top \pi^{(t)}$$

and compute the distribution  $\pi^{(t)}$  for all times

$$\begin{aligned} \pi^2 &= P^\top \pi^1 = [0.1, 0.4, 0.5]^\top; \\ \pi^3 &= P^\top \pi^2 = [0.17, 0.34, 0.49]^\top; \\ \pi^4 &= P^\top \pi^3 = [0.153, 0.362, 0.485]^\top; \\ &\vdots \\ \pi^\infty &= \lim_{t \rightarrow \infty} P^t \pi^1 \\ &= [0.158, 0.355, 0.487]^\top. \end{aligned}$$

If such a distribution  $\pi^\infty$  exists, the Markov chain is said to have “equilibrated” or reached an invariant distribution. The numbers  $\mathbb{P}(w^{t+1} = w^i)$  stop changing with time. We can compute this invariant distribution by writing

$$\pi^\infty = P^\top \pi^\infty.$$

Does such a limiting invariant distribution  $\pi^\infty$  always exist? It turns out that if a Markov chain is irreducible and recurrent, then a  $\pi^\infty$  always exists and it is also unique. Because of the above equation, we can also compute the  $\pi^\infty$  given a transition matrix  $P$ : the invariant distribution is the (right-)eigenvector of the matrix  $P^\top$  corresponding to the eigenvalue 1.

**Example 5.** Consider a Markov chain on two states where the transition matrix is given by

$$P = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \end{bmatrix}.$$

This is an irreducible Markov chain because you can hop between any two states with non-zero probability within one step. It is also recurrent: this is intuitive because say the Markov chain was in state 1, it is easy for it to come back to this state after a few hops. After the chain comes back to state 1, the Markov property means the chain forgets all the past steps and will again come back to state 1. The expected number of times the Markov chain comes back to state 1 is infinite. We are therefore guaranteed that an invariant distribution exists. In this case it is

$$\begin{aligned}\pi^1 &= 0.5\pi^1 + 0.4\pi^2 \\ \pi^2 &= 0.5\pi^1 + 0.6\pi^2.\end{aligned}$$

Note that the constraint for  $\pi$  being a probability distribution, i.e.,  $\pi^1 + \pi^2 = 1$  is automatically satisfied by the two equations. We can solve for  $\pi^1, \pi^2$  to get

$$\pi^1 = 4/9 \quad \pi^2 = 5/9.$$

**Time spent at a particular state by the Markov chain** We can observe a long trajectory of a Markov chain and compute the number of times the chain is in a particular state  $w^i$ . This is inversely proportional to  $\pi^\infty(w^i)$ . In other words, if the invariant distribution gives small probability to a particular state, if we stop the Markov chain at an arbitrary time during its trajectory, we are very unlikely to find the Markov chain at this state.

### 11.4.3 A Markov chain model of SGD

The updates of SGD with mini-batch size  $\ell$  are given by

$$w^{t+1} - w^t = -\eta \nabla \ell_\ell(w^t).$$

Notice that conditional on the iterate  $w^t$ , the next iterate  $w^{t+1}$  is independent of  $w^{t-1}$ , all these three quantities are random variables because they depend on the input data  $\omega_0, \dots, \omega_t$  sampled by SGD in the previous time-steps. You should never make the mistake of saying that gradient descent is a Markov chain; there is no randomness in the iterates of GD.

**Transition probability of SGD** What is the transition probability

$$\mathbb{P}(w^{t+1} \mid w^t)$$

for SGD? If we take the conditional expectation on both sides

$$\mathbb{E}_\ell [w^{t+1} - w^t \mid w^t] = -\eta \mathbb{E}_\ell [\nabla \ell(w^t)] = -\eta \nabla \ell(w^t);$$

in other words, on-average the change in weights at  $w^t$  is proportional to the full gradient  $\nabla \ell(w^t)$ . Notice that the change in weights exactly the same for GD; this should not be surprising after all, if the gradient of SGD is unbiased then SGD is GD “on-average”.

**Variance of SGD weight updates** The variance is computed as follows

$$\begin{aligned}\text{Var}_\ell (w^{t+1} - w^t \mid w^t) &= \eta^2 \text{Var}_\ell (\nabla \ell_\ell(w^t) \mid w^t) \\ &= \eta^2 \mathbb{E}_\ell \left[ (\nabla \ell_\ell(w^t) - \nabla \ell(w^t)) (\nabla \ell_\ell(w^t) - \nabla \ell(w^t))^T \right]\end{aligned}$$

Notice that the variance of the weight updates in SGD is proportional to the square of the learning rate. We have seen this before, larger the learning rate more noisy the weight update as compared to the update using the full-gradient  $\eta \nabla \ell(w^t)$ . The variance is a large matrix  $\in \mathbb{R}^{p \times p}$ ; this matrix depends on the current weight  $w^t$ .

If we are sampling the data inside a mini-batch with replacement the stochastic gradients are independent for different samples  $\omega^1$  and  $\omega^2$  in the mini-batch

$$\nabla \ell^{\omega^1}(w) \perp \nabla \ell^{\omega^2}(w).$$

In other words

$$\mathbb{E}_{\omega^1, \omega^2} \left[ (\nabla \ell^{\omega^1}(w^t) - \nabla \ell(w^t)) (\nabla \ell^{\omega^2}(w^t) - \nabla \ell(w^t))^T \right] = 0.$$

You can use this to show that

$$\begin{aligned} \text{Var}_{\omega^1, \dots, \omega^{\ell}}(w^{t+1} - w^t \mid w^t) &= \eta^2 \text{Var}_{\omega^1, \dots, \omega^{\ell}} \left( \frac{1}{\ell} \sum_{i=1}^{\ell} \nabla \ell^{\omega^i}(w^t) \right) \\ &= \frac{\eta^2}{\ell^2} \sum_{i=1}^{\ell} \text{Var}_{\omega^i}(\nabla \ell^{\omega^i}(w^t)) \\ &= \frac{\eta^2}{\ell} \text{Var}_{\omega}(\nabla \ell^{\omega}(w^t)). \end{aligned} \quad (11.17)$$

The last step follows because we are sampling inputs  $\omega^i$  uniformly randomly and therefore gradients  $\nabla \ell^{\omega^i}(w^t)$  are not just independent but also identically distributed. In other words, a mini-batch size of  $\ell$  reduces the variance by a factor of  $\ell$ .

**SGD is like GD with Gaussian noise** We now *model* the transition probability  $\mathbb{P}(w^{t+1} \mid w^t)$  as a Gaussian distribution. Let us denote by  $W^t, W^{t+1}$  etc. the updates of this model. We now have

$$W^{t+1} = W^t + \xi^t$$

where  $\xi^t$  is Gaussian noise

$$\xi^t \sim N \left( -\eta \nabla \ell(w^t), \frac{\eta^2}{\ell} \text{Var}_{\omega}(\nabla \ell^{\omega}(w^t)) \right).$$

In other words, on-average SGD updates weights like gradient descent, by a term  $-\eta \nabla \ell(w^t)$  but SGD's updates also have a variance.

Such equations are called stochastic difference equations and they are quite difficult to understand compared to non-stochastic difference equations (what we see in gradient descent). So we will make a drastic simplification in our model. We will say that the variance of the mini-batch gradients is identity. Our model for SGD is

$$W^{t+1} = W^t - \eta \nabla \ell(W^t) + \frac{\eta}{\sqrt{\ell}} \xi^t \quad (11.18)$$

where we have zero-mean unit-variance Gaussian noise  $\xi^t \sim N(0, I_{p \times p})$ .

**Remark 6.** The above model for SGD is a Markov chain except that the states in the Markov chain is infinite; the number of states in the Whack-The-Mole example were finite. It is easy to see that the above model is not exactly SGD: (i) we assumed the the transition probability was a Gaussian which need not be the case while training a deep network, (ii) we further assumed that the Gaussian noise does not depend on  $w^t$  and has identity covariance. You can implement the above model on a computer, first you compute the *full gradient*  $\nabla \ell(w^t)$  and then sample Gaussian noise  $\xi^t$  to update the weights to  $W^{t+1}$ . This is obviously not equivalent to SGD which updates weights using the stochastic gradient  $\nabla \ell_{\beta}(w^t)$ .

#### 11.4.4 The Gibbs distribution

In a Markov chain we were interested in the invariant distribution because that gives us a way to understand where the chain spends most of its time. We can compute the invariant distribution for our model of SGD. It is a very powerful result (which we will not do) and leads to the so-called Gibbs distribution. The probability density of the invariant distribution is given by

$$\rho^{\infty}(w) = \frac{1}{Z(\beta)} e^{-\beta \ell(w)}. \quad (11.19)$$

The quantity

$$\beta = \frac{2\beta}{\eta} \quad (11.20)$$

and  $Z(\beta)$  is a normalizing constant for probability density

$$Z(\beta) = \int_{\mathbb{R}^p} e^{-\beta \ell(w)} dw.$$

Let us list a few properties of the Gibbs distribution that are apparent simply by looking at the above expression.

1. The invariant distribution is reached asymptotically and is the limiting distribution of the weights. For instance the sum of the weights along an infinitely long trajectory converges to the mean of the Gibbs distribution

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T W^t = \int_w w \rho^{\infty}(w) dw. \quad (11.21)$$

Similarly, the second moment of the weights along a long trajectory of SGD converges to the second moment of the Gibbs distribution; and same for the variance.

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t'=1}^T \sum_{t=1}^T (W^{t'}) (W^t)^{\top} = \int_{w, w'} w w'^{\top} \rho^{\infty}(w) \rho^{\infty}(w') dw dw'. \quad (11.22)$$

2. The probability that the iterates of SGD are found at a location  $w$  is proportional to  $e^{-\beta \ell(w)}$ . If the training loss  $\ell(w)$  is high, this probability is low and if the training loss is low, the probability is high. The Gibbs distribution therefore shows that if we let SGD run until it equilibrates we have a high chance of finding the iterates that have a small training

loss. This observation is powerful because it does not require us to assume that  $\ell(w)$  is convex. However this statement does require the assumption that the steps-size  $\eta$  of SGD does not go to zero; after all SGD iterates *stop* if  $\eta = 0$ .

3. The quantity  $1/\beta$  is quite common in physics where it is called the “temperature”. This temperature  $\beta^{-1} = \frac{\eta}{2\ell}$  fundamentally governs how the Gibbs distribution looks. Higher the temperature, more the noise in the iterates and vice-versa. If the learning rate  $\eta$  is large or the batch-size  $\ell$  is small, it is easy for *our model of SGD* to jump over hills. This is the reason why the Gibbs distribution will be spread around the entire domain at high temperature. On the other hand, if temperature is very small, the Gibbs distribution puts a large probability mass in places where the training loss is small and the probability of finding iterates at other places in the domain diminishes. In particular, if  $\beta \rightarrow \infty$ , the Gibbs distribution only puts non-zero probability on the global minima of the loss function  $\ell(w)$ .

4. Written in another way, if we want the Gibbs distribution to remain the same we should ensure that

$$\beta^{-1} = \frac{\eta}{2\ell} \text{ is a constant.}$$

If you increased the batch-size by two times, you should also double the learning rate if you desire that the solutions of SGD are qualitatively similar.

5. We have achieved something remarkable by looking at the Gibbs distribution. We have discovered an algorithm to find the global minimum of a non-convex loss function.

- Start from some initial condition  $w^0$ ;
- Take lots of steps of SGD with learning rate  $\eta$  until SGD reaches its invariant distribution, i.e., until it equilibrates;
- Reduce the step-size  $\eta$  and repeat the previous step

This is only a formal algorithm but in theory it will converge to the global minimum of a non-convex function  $\ell(w)$  if the number of steps is very large. The catch of course is that at each step we have to wait until SGD equilibrates. For many problems, it may take an inordinately long amount of time to SGD to equilibrate.

❓ How much time does it take SGD to equilibrate for a convex loss function?

It is very important to remember that when we train a deep network we are executing one run of SGD. The invariant distribution is an abstract concept that does not really exist on your computer. We have constructed this model to help us understand how updates of SGD behave.

## 11.4.5 Convergence of a Markov chain to its invariant distribution

For gradient descent and SGD, we had quantities like  $\|w^t - w^*\|$  or  $\ell(w^t) - \ell(w^*)$  that let us measure the progress towards the global minimum. For a non-convex problem, there may not exist a unique global minimum, or there may be multiple local minima in the domain where the gradient vanishes. We discussed in the preceding section how the invariant distribution of SGD is achieved even if the loss  $\ell(w)$  is non-convex. In this section, we will see a simple tool to measure progress towards this distribution.

Let us define a quantity called the Kullback-Leibler (KL) divergence between two probability distributions. For two probability distributions  $p(w)$  and  $q(w)$  supported on a discrete set  $w \in W$ , the KL-divergence is given by

$$\text{KL}(p \parallel q) = \sum_{w \in W} p(w) \log \frac{p(w)}{q(w)}. \quad (11.23)$$

This formula is well-defined only if for all  $w$  where  $q(w) = 0$ , we also have  $p(w) = 0$ . The KL-divergence is a measure of the distance between two distances, it is zero if and only if  $p(w) = q(w)$  for all  $w \in W$ . It is always positive (you can show this easily using Jensen's inequality). However, the KL-divergence is not a metric because it is not symmetric

$$\text{KL}(p \parallel q) \neq \text{KL}(q \parallel p) = \sum_{w \in W} q(w) \log \frac{q(w)}{p(w)}.$$

For probability densities, the KL-divergence

$$\text{KL}(p \parallel q) = \int_w p(w) \log \frac{p(w)}{q(w)} dw \quad (11.24)$$

is defined analogously and has the same properties.

We will now show a very powerful result: the KL-divergence of the state distribution of a Markov chain decreases monotonically as the Markov chain converges to its invariant distribution. Although, this result is true for SGD as well, we will only prove it for a Markov chain with finite states. Let the initial distribution of the Markov chain be  $\pi^0$ , its transition matrix be  $P$  and its invariant distribution by  $\pi^\infty$ . We will assume that the Markov chain is such that the invariant distribution exists (it is irreducible and recurrent).

Let us also assume that a reverse transition matrix

$$P_{ij}^{\text{rev}} = \mathbb{P}(w^t = w^i | w^{t+1} = w^j).$$

exists; such Markov chains are called reversible. For any distribution  $\pi(\cdot)$  and states  $w, w'$  this transition matrix satisfies the definition of conditional probability

$$\mathbb{P}(w^{t+1} = w' | w^t = w) \mathbb{P}(w^t = w) = \mathbb{P}(w^t = w | w^{t+1} = w') \mathbb{P}(w^{t+1} = w').$$

In our notation, this becomes

$$P_{ww'}^{\text{rev}} = \frac{P_{w'w} \pi(w')}{\pi(w)} = \frac{P_{w'w} \pi(w')}{\sum_{w'} P_{w'w} \pi(w')}.$$

505 **Lemma 7.** For a reversible Markov chain with an invariant distribution  $\pi^\infty$ ,  
 506  $\text{KL}(\pi^\infty \parallel \pi^t)$  decreases monotonically:

$$\text{KL}(\pi^\infty \parallel \pi^{t+1}) \leq \text{KL}(\pi^\infty \parallel \pi^t). \quad (11.25)$$

507 **Proof.** The proof is a simple calculation as follows.

$$\begin{aligned} \text{KL}(\pi^\infty \parallel \pi^{t+1}) &= \sum_w \pi^\infty(w) \log \frac{\pi^\infty(w)}{\pi^{t+1}(w)} \\ &= \sum_w \pi^\infty(w) \log \frac{\pi^\infty(w)}{\sum_{w'} P_{w'w} \pi^t(w')} \\ &= - \sum_w \pi^\infty(w) \log \frac{\sum_{w'} P_{w'w} \pi^t(w')}{\pi^\infty(w)} \\ &= - \sum_w \pi^\infty(w) \log \left( \sum_{w'} P_{ww'}^{\text{rev}} \frac{\pi^t(w')}{\pi^\infty(w')} \right) \quad (\text{substitute definition of } P^{\text{rev}} \text{ for distribution } \pi^\infty) \\ &\leq - \sum_w \pi^\infty(w) \sum_{w'} P_{ww'}^{\text{rev}} \log \frac{\pi^t(w')}{\pi^\infty(w')} \quad (\text{Jensen's inequality}) \\ &= \sum_{w'} \sum_x P_{ww'}^{\text{rev}} \pi^\infty(w) \log \frac{\pi^\infty(w')}{\pi^t(w')} \quad (\text{flip the negative sign, exchange sum}) \\ &= \sum_{w'} \pi^\infty(w') \log \frac{\pi^\infty(w')}{\pi^t(w')} \\ &= \text{KL}(\pi^\infty \parallel \pi^t). \end{aligned}$$

508 The distance to the invariant distribution  $\pi^\infty$  decreases at each step of the  
 509 Markov chain. A similar computation is true for the reverse KL divergence as  
 510 well:

$$\text{KL}(\pi^{t+1} \parallel \pi^\infty) \leq \text{KL}(\pi^t \parallel \pi^\infty).$$

511

□

The above result is also true for SGD which, as we discussed, can be modeled as a Markov chain with infinite states. It gives us some very important intuition. Just like gradient descent makes monotonic progress towards the global minimum  $w^*$ , a Markov chain (or SGD) makes monotonic progress towards its invariant distribution. The big difference between them is that while we required that the loss function  $\ell(w)$  is convex for gradient descent to guarantee this monotonic progress, the loss need not be convex for the case of the Markov chain model of SGD.

This result *does not* mean that SGD makes monotonic progress towards the global minimum  $w^* = \text{argmin}_w \ell(w)$ . We choose to look at SGD not as one particle undergoing (stochastic) gradient descent updates but rather as a Markov chain. The probability distribution of states of this Markov chain is then a legitimate object (the distribution  $\pi^t$  is conceptually the distribution of weights  $W^t$  obtained after many independent run of SGD from different initializations). Although  $\pi^t$  is *not* meaningful across *one* run of SGD, we can use it to understand how SGD also makes monotonic progress as it converges.

# Bibliography

- 513 Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks:*  
514 *Tricks of the trade*, pages 421–436. Springer.
- 515 Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for  
516 large-scale machine learning. *Siam Review*, 60(2):223–311.
- 517 Chaudhari, P. and Soatto, S. (2017). Stochastic gradient descent performs  
518 variational inference, converges to limit cycles for deep networks. *arXiv*  
519 *preprint arXiv:1710.11029*.
- 520 Izmailov, P., Podoprikin, D., Gariyov, T., Vetrov, D., and Wilson, A. G. (2018).  
521 Averaging weights leads to wider optima and better generalization. *arXiv*  
522 *preprint arXiv:1803.05407*.
- 523 Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insuffi-  
524 ciency of existing momentum schemes for stochastic optimization. In *2018*  
525 *Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE.
- 526 Kushner, H. and Yin, G. G. (2003). *Stochastic approximation and recursive*  
527 *algorithms and applications*, volume 35. Springer Science & Business  
528 Media.
- 529 Le Roux, N., Schmidt, M. W., Bach, F. R., et al. (2012). A stochastic gradient  
530 method with an exponential convergence rate for finite training sets. In  
531 *NIPS*, pages 2672–2680.
- 532 Liu, C. and Belkin, M. (2018). Mass: an accelerated stochastic method for  
533 over-parametrized learning. *arXiv preprint arXiv:1810.13395*.
- 534 Polyak, B. T. (1990). A new method of stochastic approximation type. *Av-*  
535 *tomatika i telemekhanika*, (7):98–107.
- 536 Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic ap-  
537 proximation by averaging. *SIAM Journal on Control and Optimization*,  
538 30(4):838–855.
- 539 Recht, B. and Ré, C. (2012). Beneath the valley of the noncommutative  
540 arithmetic-geometric mean inequality: conjectures, case-studies, and conse-  
541 quences. *arXiv preprint arXiv:1202.4184*.
- 542 Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The*  
543 *annals of mathematical statistics*, pages 400–407.
- 544 Ruppert, D. (1988). Efficient estimations from a slowly convergent robbins-  
545 monro process. Technical report, Cornell University Operations Research  
546 and Industrial Engineering.