

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
4. Stochastic Weight Averaging (SWA) by [Izmailov et al. \(2018\)](#).

3 Stochastic Gradient Descent (SGD) has its roots in stochastic optimization.

4 A stochastic optimization problem looks like

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

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

17 In machine learning, we are interested in solving a slightly different prob-
18 lem 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 ξ 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 ℓ let us denote the samples in the mini-batch
31 by

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

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

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

50 Although we will not discuss this, SGD using mini-batches sampled with re-
 51 placement is faster than with mini-batches sampled without replacement ([Recht
 52 and Ré, 2012](#)).

53 11.1 SGD for least-squares regression

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

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

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

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

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

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

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

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

67 which we will call the “zone of confusion”. Similarly, if weights are initialized
 68 $w^0 \ll w_{\min}$, 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 η is fixed, since the samples ω_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.

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 ω_t . The iterate ω_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

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

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

Lemma 1 (Descent Lemma for SGD).

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

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

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

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

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

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

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

105 **11.2.1 Typical assumptions in the analysis of SGD**

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

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

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

- 114 2. **Second moment of gradient norm does not grow too quickly.** We
 115 will assume that there exist scalars σ_0 and σ such that

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

116 This assumption allows to control the second term in the descent lemma
 117 for SGD. It assumes that the stochastic estimate of the gradient in SGD
 118 $\nabla \ell^{\omega_t}(w)$ is not too different than the full gradient $\ell(w^t)$. In the neighbor-
 119 hood of a critical point (locations where the full gradient $\nabla \ell(w) = 0$),
 120 the stochastic gradient is allowed to grow in a similar fashion as the true
 121 gradient except with a scaling factor $\sigma > 0$ and a constant σ_0 .

122 Let us see how the descent lemma changes with these additional assumptions.
 123

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

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

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

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

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

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

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

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

134 11.2.2 Convergence rate of SGD for strongly-convex func- 135 tions

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

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

137 for m -strongly convex and L -smooth function $\ell(w)$ then the expected optimality gap satisfies
 138

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

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

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

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

- 146 1. The random variable w^{t+1} depends upon all the indices $\omega_1, \omega_2, \dots, \omega_t$
 147 that were sampled during updates of SGD and therefore the expectation
 148 in (11.11) should be over all these random variables.
- 149 2. When the stochastic gradient is noisy, we have a non-zero σ_0 we can no
 150 longer get to the global minimum, there is a first term which does not
 151 decay with time.
- 152 3. If we pick a small η , we get closer to the global minimum but go there
 153 quite slowly. On the other hand, we can pick a large η and get to a
 154 neighborhood of the global minimum quickly but we will then have a
 155 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 η 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

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

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

160 **Theorem 4 (Convergence rate of SGD for decaying step-size).** For a sched-
 161 ule

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

162 then the expected optimality gap satisfies

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

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

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

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

¹⁷⁰ but the second assumption in SGD improves a bit. Since the mini-batch
¹⁷¹ gradient is averaged over ℓ samples we have

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

¹⁷² if σ_0, σ were the constants in (11.9). This changes the convergence rate in Theorem 3 to

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

¹⁷⁴ Note that the maximum learning rate in Theorem 3 is inversely proportional
¹⁷⁵ to σ so we can also pick a larger learning rate $\eta < \frac{\ell}{L\sigma}$. If we do so, the first
¹⁷⁶ and last terms above are not affected by the batch-size but multiplicative term
¹⁷⁷ $(1 - \eta m)^t$ is. Since

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

¹⁷⁸ we see that increasing the learning rate by a factor of ℓ will reduce the number
¹⁷⁹ of iterations required to reach the zone of confusion by a factor of ℓ . Of
¹⁸⁰ course, this comes with the caveat that each iteration also requires $\mathcal{O}(\ell)$ more
¹⁸¹ computation to compute the gradient compared to single-sample SGD.

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

¹⁸³ Theorem 4 indicates that SGD is a very slow algorithm, GD is much faster
¹⁸⁴ than SGD to minimize strongly convex functions. This gap also exists if we do
¹⁸⁵ not have strong convexity: we did not prove this but SGD requires $\mathcal{O}(1/\epsilon^2)$ to
¹⁸⁶ reach an ϵ -neighborhood of the global optimum for convex functions whereas
¹⁸⁷ GD requires a much smaller $\mathcal{O}(1/\epsilon)$. One might wonder why we should use
¹⁸⁸ SGD at all.

¹⁸⁹ It is critical to remember that the objective in machine learning is a sum of
¹⁹⁰ many terms

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

¹⁹¹ One iteration of SGD requires us to compute only $\nabla \ell^{\omega_t}(w)$ whereas one
¹⁹² update of GD requires us to compute the full gradient $\nabla \ell(w)$. One weight
¹⁹³ update of GD is $\mathcal{O}(n)$ more expensive than one weight update using SGD. Let
¹⁹⁴ us do a back-of-the-envelope computation for convex functions. If we want
¹⁹⁵ to reach an ϵ -neighborhood of the global minimum of a convex function, we
¹⁹⁶ need $\mathcal{O}(1/\epsilon)$ iterations of GD, which requires

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

¹⁹⁷ operations. SGD needs $\mathcal{O}(1/\epsilon^2)$ iterations and therefore requires

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

¹⁹⁸ operations to reach the ϵ -neighborhood. This indicates that if our chosen ϵ -ball
¹⁹⁹ is

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

200 GD requires fewer overall operations. But if ϵ -ball is larger than this, we should
 201 use SGD because it is computationally cheaper.

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

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

205 If ϵ^i is constant and does not depend on the number of training samples n
 206 (i.e., say we are happy with the average error over the training dataset being
 207 2% even and do not seek a smaller one even if we collect more data) then we
 208 should use SGD to train our model because it is cheaper. This is not always
 209 the case for other problems, e.g., if you are doing computational tomography
 210 (capturing multiple images from a CT-scan machine and trying to reconstruct
 211 the heart/lung region in the thoracic cavity), we may seek a more and more
 212 accurate answer, i.e., small ϵ if we have more data.

213 11.3 Accelerating SGD using momentum

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

220 **Polyak-Ruppert averaging** Consider the updates

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

221 In a series of papers, Polyak (1990); Polyak and Juditsky (1992); Ruppert
 222 (1988) showed that the quantity

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

223 converges faster than the quantity

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

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

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

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

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

238 11.3.1 Momentum methods do not accelerate SGD

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

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

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

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

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

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

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

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

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



276

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

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

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

288 11.4 Understanding SGD as a Markov Chain

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

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

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

298 11.4.1 Gradient flow

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

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

302 If we again imagine a continuously differentiable curve $W(\tau)$ as a model for
 303 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.

304 we can write a differential equation of the form

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

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

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

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

312 11.4.2 Markov chains

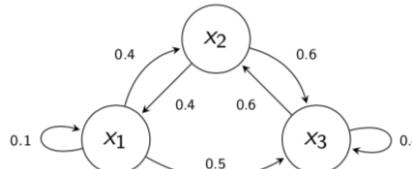
313 Consider the Whack-The-Mole game: a mole has burrowed a network of three
314 holes w^1, w^2, w^3 into the ground. It keeps going in and out of the holes and
315 we are interested in finding which hole it will show up next so that we can give
316 it a nice whack.

- 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

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

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

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

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

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

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

324 The set of states in the Markov chain that *all* communicate with each other
325 are an equivalence class. This means that the Markov chain can visit any state
326 from any other state in this equivalence class with non-zero probability, we
327 just might have to wait for a really long time if $P_{ij}^t \approx 0$ for two states w^i, w^j .
328 If all the states in the Markov chain belong to the same equivalence class, it
329 is called irreducible. A related concept is that of “positive recurrence”, i.e., if
330 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 π^∞ 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 π^∞ always exist? It turns out that if a Markov chain is irreducible and recurrent, then a π^∞ always exists and it is also unique. Because of the above equation, we can also compute the π^∞ 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

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

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

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

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

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

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

370 11.4.3 A Markov chain model of SGD

371 The updates of SGD with mini-batch size ℓ are given by

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

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

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

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

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

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

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

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

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

384 Notice that the variance of the weight updates in SGD is proportional to the
 385 square of the learning rate. We have seen this before, larger the learning rate
 386 more noisy the weight update as compared to the update using the full-gradient
 387 $\eta \nabla \ell(w^t)$. The variance is a large matrix $\in \mathbb{R}^{p \times p}$; this matrix depends on the
 388 current weight w^t .

389 If we are sampling the data inside a mini-batch with replacement the
 390 stochastic gradients are independent for different samples ω^1 and ω^2 in the
 391 mini-batch

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

392 In other words

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

393 You can use this to show that

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

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

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

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

401 where ξ^t is Gaussian noise

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

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

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

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

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

410 **Remark 6.** The above model for SGD is a Markov chain except that the states
 411 in the Markov chain is infinite; the number of states in the Whack-The-Mole
 412 example were finite. It is easy to see that the above model is not exactly SGD:
 413 (i) we assumed the the transition probability was a Gaussian which need not
 414 be the case while training a deep network, (ii) we further assumed that the
 415 Gaussian noise does not depend on w^t and has identity covariance. You can
 416 implement the above model on a computer, first you compute the *full gradient*
 417 $\nabla \ell(w^t)$ and then sample Gaussian noise ξ^t to update the weights to W^{t+1} .
 418 This is obviously not equivalent to SGD which updates weights using the
 419 stochastic gradient $\nabla \ell_\theta(w^t)$.

420 11.4.4 The Gibbs distribution

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

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

426 The quantity

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

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

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

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

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

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

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

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

- 436 2. The probability that the iterates of SGD are found at a location w is
 437 proportional to $e^{-\beta \ell(w)}$. If the training loss $\ell(w)$ is high, this probability
 438 is low and if the training loss is low, the probability is high. The Gibbs
 439 distribution therefore shows that if we let SGD run until it equilibrates
 440 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 η 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 $\beta^{-1} = \frac{\eta}{2\theta}$ fundamentally governs how the Gibbs distribution looks. Higher the temperature, more the noise in the iterates and vice-versa. If the learning rate η is large or the batch-size θ 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

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

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 η until SGD reaches its invariant distribution, i.e., until it equilibrates;
 - Reduce the step-size η and repeat the previous step

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

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.

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.

⁴⁷⁴ **11.4.5 Convergence of a Markov chain to its invariant dis-**
⁴⁷⁵ **tribution**

⁴⁷⁶ For gradient descent and SGD, we had quantities like $\|w^t - w^*\|$ or $\ell(w^t) -$
⁴⁷⁷ $\ell(w^*)$ that let us measure the progress towards the global minimum. For a
⁴⁷⁸ non-convex problem, there may not exist a unique global minimum, or there
⁴⁷⁹ may be multiple local minima in the domain where the gradient vanishes. We
⁴⁸⁰ discussed in the preceding section how the invariant distribution of SGD is
⁴⁸¹ achieved even if the loss $\ell(w)$ is non-convex. In this section, we will see a
⁴⁸² simple tool to measure progress towards this distribution.

⁴⁸³ Let us define a quantity called the Kullback-Leibler (KL) divergence be-
⁴⁸⁴ tween two probability distributions. For two probability distributions $p(w)$ and
⁴⁸⁵ $q(w)$ supported on a discrete set $w \in W$, the KL-divergence is given by

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

⁴⁸⁶ This formula is well-defined only if for all w where $q(w) = 0$, we also have
⁴⁸⁷ $p(w) = 0$. The KL-divergence is a measure of the distance between two
⁴⁸⁸ distances, it is zero if and only if $p(w) = q(w)$ for all $w \in W$. It is always
⁴⁸⁹ positive (you can show this easily using Jensen's inequality). However, the
⁴⁹⁰ KL-divergence is not a metric because it is not symmetric

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

⁴⁹¹ For probability densities, the KL-divergence

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

⁴⁹² is defined analogously and has the same properties.

⁴⁹³ We will now show a very powerful result: the KL-divergence of the state
⁴⁹⁴ distribution of a Markov chain decreases monotonically as the Markov chain
⁴⁹⁵ converges to its invariant distribution. Although, this result is true for SGD
⁴⁹⁶ as well, we will only prove it for a Markov chain with finite states. Let the
⁴⁹⁷ initial distribution of the Markov chain by π^0 , its transition matrix be P and
⁴⁹⁸ its invariant distribution by π^∞ . We will assume that the Markov chain is such
⁴⁹⁹ that the invariant distribution exists (it is irreducible and recurrent).

⁵⁰⁰ Let us also assume that a reverse transition matrix

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

⁵⁰¹ exists; such Markov chains are called reversible. For any distribution $\pi(\cdot)$
⁵⁰² and states w, w' this transition matrix satisfies the definition of conditional
⁵⁰³ probability

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

⁵⁰⁴ 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 π^∞ ,
 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.

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

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

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

□

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 $w^* = \operatorname{argmin}_w \ell(w)$. 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 π^t is conceptually the distribution of weights W^t obtained after many independent run of SGD from different initializations). Although π^t is *not* meaningful across *one* run of SGD, we can use it to understand how SGD also makes monotonic progress as it converges.

⁵¹² Bibliography

- ⁵¹³ Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: 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 preprint arXiv:1710.11029*.
- ⁵²⁰ Izmailov, P., Podoprikin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018).
⁵²¹ Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.
- ⁵²³ Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insufficiency
⁵²⁴ of existing momentum schemes for stochastic optimization. In *2018 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 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 *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. *Avtomatika 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.