Optimization for Machine Learning

Stochastic gradient

Clément Royer

CIMPA School "Control, Optimization and Model Reduction in Machine Learning"

February 26, 2025

Sources

Github repository: https://tinyurl.com/3etmd46y

Outline

Outline

Context: Finite sum problems

- Data $\{(x_i, y_i)\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, with an underlying distribution.
- Predictor function/Model h such that $h(\boldsymbol{x}_i) \approx y_i$;
- Model parameterized by $\boldsymbol{w} \in \mathbb{R}^d \Rightarrow h(\boldsymbol{x}_i) = h(\boldsymbol{w}; \boldsymbol{x}_i)$
- Accuracy of model on data measured through a loss ℓ.

Optimization problem

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(h(\boldsymbol{w}; \boldsymbol{x}_i), y_i)}_{f_i(\boldsymbol{w})} = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}).$$

Applying gradient descent

Gradient descent for minimize$_{\boldsymbol{w}\in\mathbb{R}^d}$ $\frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{w})$

Assuming all f_is are differentiable, the gradient descent iteration is:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k) = \boldsymbol{w}_k - \frac{\alpha_k}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{w}_k).$$

Applying gradient descent

Gradient descent for minimize$_{\boldsymbol{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{w})$

Assuming all f_is are differentiable, the gradient descent iteration is:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k) = \boldsymbol{w}_k - \frac{\alpha_k}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{w}_k).$$

- Big data setting: n is very large and $\nabla f(\boldsymbol{w}_k)$ is very expensive to compute;
- One iteration of gradient descent involves looking at the **entire dataset**.

Stochastic gradient for minimize$_{\boldsymbol{w}\in\mathbb{R}^d}\ \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w})$

Iteration

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k),$$

where i_k is drawn randomly in $\{1, \ldots, n\}$.

- Use one (random) data point at a time $\Rightarrow$ **n times cheaper than a full gradient calculation**!

Beyond Stochastic Gradient

Why just sample one data point?

- SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k)$,
  i_k drawn at random;
- **Batch SG:**

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(\boldsymbol{w}_k)$$

  where $S_k \subset \{1, \ldots, n\}$ is drawn at random.

Two batch regimes

- $|S_k| \approx n$: essentially equivalent to full gradient;
- $|S_k| = n_b << n$: mini-batching.

Implementing batch stochastic gradient

Batch SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(\boldsymbol{w}_k)$.

Hyperparameters

- Stepsize/Learning rate α_k.
- Batch size $|S_k|$.

Rule of thumb (from Hardt and Recht '22)

- *"Pick the largest constant value such that the method does not diverge".*
- *"Pick the batch size according to your number of available processors".*

Outline

Analyzing the behavior of Stochastic Gradient

Stochastic Gradient...Descent?

- Commonly called SGD by analogy with GD...
- ...but SG is not a descent method in general!
- It is however a descent method in expectation.

Analyzing the behavior of Stochastic Gradient

Stochastic Gradient...Descent?

- Commonly called SGD by analogy with GD...
- ...but SG is not a descent method in general!
- It is however a descent method in expectation.

Key argument for the analysis when $f \in \mathcal{C}_L^{1,1}$

- For Gradient Descent, the key lemma was

$$f(\boldsymbol{w}_{k+1}) - f(\boldsymbol{w}_k) \leq \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k) + \frac{L}{2}\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|^2.$$

- For Stochastic Gradient, the key lemma is

$$\mathbb{E}_{i_k}\left[f(\boldsymbol{w}_{k+1})\right] - f(\boldsymbol{w}_k) \leq \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}\, \mathbb{E}_{i_k}\left[\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\right] + \frac{L}{2}\,\mathbb{E}_{i_k}\left[\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|^2\right].$$

$\Rightarrow$ Decrease **in expectation**!

Analysis in expectation

Assumptions on stochastic gradient

For every k, i_k is drawn such that:

1. $\mathbb{E}_{i_k}\left[\nabla f_{i_k}(\boldsymbol{w}_k)\right] = \nabla f(\boldsymbol{w}_k)$:
   On average, the stochastic gradient $\nabla f_{i_k}(\boldsymbol{w}_k)$ is close to the true gradient (unbiased estimate).
2. $\mathrm{Var}_{i_k}\left[\|\nabla f_{i_k}(\boldsymbol{w}_k)\|\right] \leq \sigma^2$ with $\sigma^2 > 0$:
   Do not deviate too much from the mean value/the true gradient.

Uniform sampling satisfies those properties.

Under these assumptions, we can establish complexity results/convergence rates for strongly convex/convex/nonconvex problems, that heavily depend on the **step size** α_k.

Choosing the stepsize

- Arguably the biggest issue in ML is **tuning the learning rate**, i.e. choosing the stepsize;
- We illustrate the arguments for constant and decreasing stepsize for strongly convex functions.

Strongly convex

- Assumption: f is μ-strongly convex;
- Unique global minimizer: $\boldsymbol{w}^*$, $f^* = f(\boldsymbol{w}^*)$.

Choosing the stepsize (2)

Constant stepsize in the strongly convex case

If $\alpha_k = \alpha \in (0, \frac{1}{2\mu}) \forall k$, then

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \frac{\alpha L \sigma^2}{4\mu} + (1 - 2\alpha\mu)^k \left[\frac{\alpha L \sigma^2}{2\mu} + f(\boldsymbol{w}_0) - f^*\right].$$

- Convergence at a linear rate.
- Reaches a **neighborhood** of the optimal value $\Rightarrow$ effect of the noise, illustrated by the $\frac{\alpha L \sigma^2}{\mu}$ terms.

- **Pro:** Can take long steps.
- **Con:** Converges to a neighborhood of f^*.

Choosing the stepsize (3)

A practical constant stepsize approach

- In ML, common to run the algorithm with α until it stalls, then use $\alpha/2$ until it starts stalling again, then $\alpha/4$, etc;
- Guaranteed convergence, but at a sublinear rate: $\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \mathcal{O}\left(\frac{1}{k}\right)$

- **Pro:** Adapts the stepsize to reach closer neighborhoods;
- **Con:** Convergence can be slow.

Choosing the stepsize (4)

Decreasing stepsizes for strongly convex problems

- Original SG algorithm: choose $\{\alpha_k\}$ such that

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

- Typical choice: $\alpha_k = \frac{c}{k+1}$, $c > \frac{1}{\mu}$, $\alpha_0 \leq \frac{\mu}{L} \Rightarrow$ leads to

$$\mathbb{E}\left[f(\boldsymbol{w}_k) - f^*\right] \leq \mathcal{O}\left(\frac{1}{k+1}\right).$$

- Pro: Choice less sensitive to parameter values;
- Con: Forced to decrease at every iteration.

Stepsizes in the nonconvex case

In the nonconvex setting, we get guarantees

- On $\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\|\nabla f(\boldsymbol{w}_k)\|^2\right]$ for constant stepsizes;
- On $\mathbb{E}\left[\frac{1}{\sum_{i=1}^{K}\alpha_k}\sum_{i=1}^{K}\alpha_k\|\nabla f(\boldsymbol{w}_k)\|^2\right]$ for decreasing stepsizes.

$\Rightarrow$ Similarly to the strongly convex case, get the usual bound+residual term, resulting in worse rates.

Ex) $\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\|\nabla f(\boldsymbol{w}_k)\|^2\right] \leq \epsilon$ in at most $\mathcal{O}(\epsilon^{-4})$ iterations.

Mini-batch Stochastic Gradient

Property of mini-batch SG

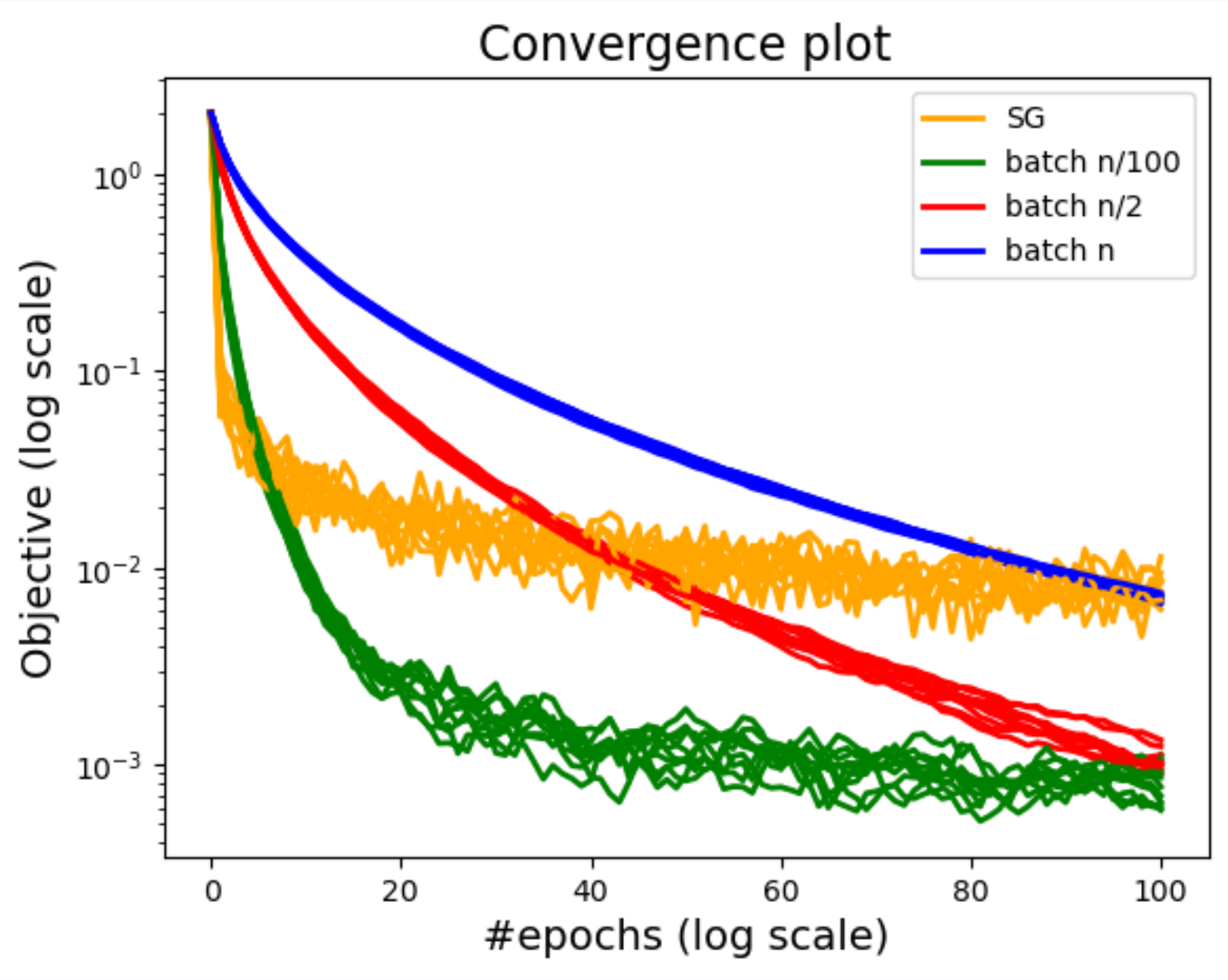
If $|S_k| = n_b \ \forall k$, with the same stepsize, mini-batch SG requires n_b less iterations than SG.

- **Pros:** Parallelization of the n_b stochastic gradients possible, variance improved:

$$\mathrm{Var}_{i_k}\left[\|\nabla f_{i_k}(\boldsymbol{w}_k)\|_2\right] \leq \sigma^2, \qquad \mathrm{Var}_{S_k}\left[\left\|\frac{1}{|S_k|}\sum_{i\in S_k}\nabla f_i(\boldsymbol{w}_k)\right\|_2\right] \leq \frac{\sigma^2}{n_b}.$$

- **Cons:** Still more expensive than SG, more sensitive to redundancies in the data.

Batch stochastic gradient and run variance



Outline

Diagonal scaling

Basic SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{g}_k$.

Scaling idea

- Use one stepsize per coordinate!
- Equivalent to

$$[\boldsymbol{w}_{k+1}]_j = [\boldsymbol{w}_k]_j - \alpha_k \frac{[\boldsymbol{g}_k]_j}{[\boldsymbol{v}_k]_j} \quad \forall j = 1, \ldots, d \qquad \text{for some } \boldsymbol{v}_k \in \mathbb{R}^d.$$

Diagonal scaling

Basic SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{g}_k$.

Scaling idea

- Use one stepsize per coordinate!
- Equivalent to

$$[\boldsymbol{w}_{k+1}]_j = [\boldsymbol{w}_k]_j - \alpha_k \frac{[\boldsymbol{g}_k]_j}{[\boldsymbol{v}_k]_j} \quad \forall j = 1, \ldots, d \qquad \text{for some } \boldsymbol{v}_k \in \mathbb{R}^d.$$

Key variants

$[\boldsymbol{v}_k]_j = \sqrt{[\boldsymbol{r}_k]_j + \epsilon}$ ($\epsilon > 0$ numerical tolerance).

- **Adagrad**: $[\boldsymbol{r}_k]_j = [\boldsymbol{r}_{k-1}]_j + [\boldsymbol{g}_k]_j^2$
- **RMSProp**:

$$[\boldsymbol{v}_k]_j = \beta\,[\boldsymbol{v}_{k-1}]_j^2 + (1-\beta)\,[\boldsymbol{g}_k]_j^2$$

for $\beta \in (0,1)$ (PyTorch: 0.99, JAX: 0.9).

Momentum-based methods

Basic SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{g}_k$.

Momentum techniques

- Augment gradient step with previous/momentum step.
- Written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{m}_k$$

where $\boldsymbol{m}_k$ depends on $\boldsymbol{g}_k$ and $\boldsymbol{m}_{k-1}$ for $k \geq 1$.

Momentum-based methods

Basic SG: $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{g}_k$.

Momentum techniques

- Augment gradient step with previous/momentum step.
- Written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{m}_k$$

where $\boldsymbol{m}_k$ depends on $\boldsymbol{g}_k$ and $\boldsymbol{m}_{k-1}$ for $k \geq 1$.

Important variant: SGD with momentum

$$\boldsymbol{m}_k = \beta \boldsymbol{m}_{k-1} + (1-\beta)\boldsymbol{g}_k,$$

where $\beta \in [0, 1)$ (PyTorch: 0, JAX: 0.9).

- Full batch: Variant of the Heavy-ball method.
- Philosophy: Good directions accumulate, bad directions cancel out.

Adam (Kingma, Ba '15)

$$\forall j = 1, \ldots, d, \qquad [\boldsymbol{w}_{k+1}]_j = [\boldsymbol{w}_k]_j - \alpha_k \frac{[\boldsymbol{m}_k]_j}{[\boldsymbol{v}_k]_j}.$$

Adam (Kingma, Ba '15)

$$\forall j = 1, \ldots, d, \qquad [\boldsymbol{w}_{k+1}]_j = [\boldsymbol{w}_k]_j - \alpha_k \frac{[\boldsymbol{m}_k]_j}{[\boldsymbol{v}_k]_j}.$$

- $\boldsymbol{m}_k = \frac{1-\beta_1}{1-\beta_1^{k+1}} \sum_{i=0}^{k} \beta_1^{k-i} \boldsymbol{g}_i$.
- $[\boldsymbol{v}_k]_j = \sqrt{\frac{1-\beta_2}{1-\beta_2^{k+1}} \sum_{i=0}^{k} \beta_2^{k-i} [\boldsymbol{g}_i]_j^2}$.

Adam (Kingma, Ba '15)

$$\forall j = 1, \ldots, d, \qquad [\boldsymbol{w}_{k+1}]_j = [\boldsymbol{w}_k]_j - \alpha_k \frac{[\boldsymbol{m}_k]_j}{[\boldsymbol{v}_k]_j}.$$

- $\boldsymbol{m}_k = \frac{1-\beta_1}{1-\beta_1^{k+1}} \sum_{i=0}^{k} \beta_1^{k-i} \boldsymbol{g}_i$.
- $[\boldsymbol{v}_k]_j = \sqrt{\frac{1-\beta_2}{1-\beta_2^{k+1}} \sum_{i=0}^{k} \beta_2^{k-i} [\boldsymbol{g}_i]_j^2}$.

- Geometric averages of stochastic gradients/their coordinates.
- PyTorch/JAX/Original paper: $\beta_1 = 0.9$, $\beta_2 = 0.999$.
- **THE** method of choice to train neural networks today.
- Most cited optimization paper (most cited CS paper?).

Conclusions: Stochastic optimization methods

Stochastic gradient

- Motivation: Data!
- Gain because of per-iteration cost.
- Stepsize/Batch size can be tuned (see lab).

Main variants

- Diagonal scaling (Adagrad).
- Momentum (Adam).
- Advice (Hardt & Recht): The β parameters should not be tuned too much.

References

- L. Bottou, F. E. Curtis and J. Nocedal, *Optimization methods for machine learning.* SIAM Review, 2019.
  $\Rightarrow$ Review article with mathematical details and extensions.
- M. Hardt and B. Recht, *Patterns, predictions and actions*. Cambridge University Press, 2022.
  $\Rightarrow$ Reader-friendly book with focus on interpretation.
- D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization*. International Conference on Learning Representations, 2015.