Tutorial 3: Stochastic gradient

Optimization for data science, M2 MIAGE ID/ID Apprentissage

December 8, 2025

Dauphine | PSL
UNIVERSITÉ PARIS

Exercise 1: Huber loss

We consider a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $n \geq 1$, $\boldsymbol{x}_i \in \mathbb{R}^d$ with $d \geq 1$ and $y_i \in \mathbb{R}$. We seek a linear model that best predicts every y_i given the corresponding $\boldsymbol{x}_i$. To this end, we consider a family of models parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ of the form

$$\begin{array}{rccl} h_{\boldsymbol{w}}: & \mathbb{R}^d & \to & \mathbb{R} \\ & \boldsymbol{x} & \mapsto & \boldsymbol{x}^{\mathrm{T}}\boldsymbol{w} = \sum_{i=1}^d [\boldsymbol{x}]_i [\boldsymbol{w}]_i. \end{array}$$

Given a model $h_{\boldsymbol{w}}$, we consider that this model perfectly predicts y_i given $\boldsymbol{x}_i$ if $\ell\left(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right) = \ell\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i\right) = 0$, where $\ell : \mathbb{R} \to \mathbb{R}$ is the Huber loss given by

$$\ell(t) = \begin{cases} \frac{1}{2}t^2 & \text{if } |t| < 1 \\ |t| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{1}$$

This function behaves like $t \mapsto \frac{t^2}{2}$ for $|t| < 1$ and like $t \mapsto |t|$ when $|t|$ is large enough. Unlike what its expression could suggest, the function ℓ is $\mathcal{C}^1$.

The term $\ell\left(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right)$ represents the error corresponding to the data point $(\boldsymbol{x}_i, y_i)$, and we seek a model (i.e. a vector $\boldsymbol{w} \in \mathbb{R}^d$) that yields the minimum sum of these errors. As a result, we consider the problem:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i). \tag{2}$$

a) Justify that 0 is a lower bound of the objective f of problem (2). Is it necessarily its minimum value?

b) The gradient of f at $\boldsymbol{w} \in \mathbb{R}^d$ is given by

$$\nabla f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^n \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)\boldsymbol{x}_i, \tag{3}$$

with

$$\ell'(t) = \begin{cases} 1 & \text{if} \quad t > 1 \\ t & \text{if} \quad |t| \leq 1 \\ -1 & \text{if} \quad t < -1. \end{cases}$$

Write down the gradient descent iteration with a constant stepsize α and using the formula (3) for the gradient. If the current point is a local minimum, what happens to this iteration?

c) The gradient ∇f is L-Lipschitz continuous with $L = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i\|^2$. How can this constant be used to define the stepsize? Give two other strategies for choosing the stepsize that do not require knowledge of L.

d) The function f has the form $f = \frac{1}{n}\sum_{i=1}^{n} f_i$, where $f_i(w) = \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)$. The gradient of f_i at $\boldsymbol{w}$ is

$$\nabla f_i(\boldsymbol{w}) = \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i)\boldsymbol{x}_i.$$

Write the iteration of stochastic gradient for problem (2), using a generic choice for the stepsize.

e) *For the rest of the exercise, we consider that our unit of cost is one access to a single $\boldsymbol{x}_i$.* Using the unit, what is the cost of a gradient descent iteration? What is the cost of a stochastic gradient iteration?

f) Discuss the interest of stochastic gradient in the following two cases:

   i) $n \gg 1$ and there are redundancies in the dataset $\{(\boldsymbol{x}_i, y_i)\}$ in the form of duplicate elements;

   ii) $n = d$ and the $\boldsymbol{x}_i$ are the coordinate vectors in $\mathbb{R}^n$.

g) Suppose that we run stochastic gradient with a constant stepsize on our problem, and that we observe that the method generates iterates with increasingly large norm, leading to a memory overflow. Provide a justification for this behavior.

h) We consider a batch variant of stochastic gradient where we draw n_b elements of $\{(\boldsymbol{x}_i, y_i)\}$ at every iteration.

   i) Write the corresponding iteration.

   ii) If n_b corresponds to the number of processors available for parallel calculations, what can be the interest of choosing n_b as batch size?

   iii) What is the statistical advantage of batch methods over vanilla stochastic gradient?

   iv) Suppose that we compare several batch sizes. We observe that the practical convergence rate of the method improves as n_b increases from 1 to $\frac{n}{10}$, but that it deteriorates as n_b increases from $n/10$ to n. How can you explain these observations?

Exercise 2: Random reshuffling

In this exercise, we consider a stochastic optimization problem of the form

$$\text{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}), \tag{4}$$

where every f_i is $\mathcal{C}^1$ and depends solely on the ith data point in a dataset of size n. We also assume that the objective function f is convex with Lipschitz continuous gradient and we let f^* denote its minimal value.

We assume that n is so large that using the entire dataset at once is not possible in practice, and we thus perform stochastic gradient iterations of the form

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k), \tag{5}$$

where $\alpha_k > 0$ and $i_k \in \{1,\dots,n\}$. The goal of the exercise is to explore *random reshuffling stochastic gradient* techniques, where the indices $\{i_k\}$ are drawn according to a random permutation of $\{1,\dots,n\}$ that changes every n iterations. At iteration 0, a random permutation of $\{1,\dots,n\}$ is computed, and it defines $\{i_0,\dots,i_{n-1}\}$. At iteration n, another random permutation is computed, that defines $\{i_n,\dots,i_{2n-1}\}$, and so on.

a) Recall the definition of an epoch. With random reshuffling, what guarantee do we have on the data points that have been used over the course of the first epoch?

b) Consider an iteration index k that corresponds to the first iteration of an epoch (i.e. $k = \ell n$ for some $\ell \in \mathbb{N}$).

   i) Show that
   $$\mathbb{E}_{i_k}\left[\nabla f_{i_k}(x_k)\right] = \nabla f(x_k).$$

   ii) Since the indices $i_k,\dots,i_{k+n-1}$ are not independent, justify that this property no longer holds for other indices (we say that the stochastic gradient estimates for random reshuffling are biased).

c) Despite the bias, it is possible to show convergence results for appropriate choices of step sizes. Such results are based on analyzing the behavior of the average iterate sequence $\{\bar{x}_K\}_K$, where

$$\bar{x}_K = \frac{1}{K+1}\sum_{k=0}^{K} x_k \quad \forall K \in \mathbb{N}.$$

   What interesting property does this sequence possess in general for stochastic gradient methods?

d) A typical convergence analysis for (5) shows that, for any $K \geq 1$, after nK iterations, we have

$$\mathbb{E}\left[f\left(\bar{x}_{nK}\right)\right] - f^* \leq \mathcal{O}\left(\frac{1}{\sqrt{nK}}\right),$$

   Compare this rate with that typically obtained by gradient descent.

e) Consider a batch variant of stochastic gradient where the indices are drawn following a random reshuffling approach. With a batch size of n, what algorithm do we recover?

f) Given that the random reshuffling strategy does not draw the same index twice during an epoch, one may be tempted to use information from the past iterations to incorporate more (stochastic) gradient information into the current iteration. What advanced stochastic gradient technique would you use for that purpose? Justify your answer.

Solutions

Solutions for Exercise 1

a) The function ℓ is nonnegative on $\mathbb{R}$. For any $\boldsymbol{w} \in \mathbb{R}^d$, we thus have

$$f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i) \geq \frac{1}{n}\sum_{i=1}^{n} 0 = 0.$$

Therefore, the value 0 is a lower bound for the objective of problem (2). This value is reached only when there exists a point $\boldsymbol{w}$ such that $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} - y_i = 0$ for every i. This is not always possible (take for instance $n = 2, d = 1, \boldsymbol{x}_1 = 1, \boldsymbol{x}_2 = -1, y_1 = y_2 = 1$), hence 0 is not necessarily the minimum value for the problem.

b) At $\boldsymbol{w}_k \in \mathbb{R}^d$, the gradient descent iteration with a constant stepsize α on this specific problem is

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{\alpha}{n}\sum_{i=1}^{n} \ell'(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}_k - y_i)\boldsymbol{x}_i.$$

If $\boldsymbol{w}_k$ is a local minimum, then $\nabla f(\boldsymbol{w}_k) = 0$, and the iteration becomes $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k$.

c) If the Lispchitz constant L is known, then choosing $\alpha = \frac{1}{L}$ is a good value.
If this value is unknown, we can instead use a decreasing stepsize sequence (such as $\alpha_k = \frac{1}{k+1}$) or use a line search to compute a stepsize tailored to the given iteration.

d) An iteration of stochastic gradient at $\boldsymbol{w}_k \in \mathbb{R}^d$ using stepsize α_k first draws an index i_k in $\{1, \ldots, n\}$ at random. Then, the new iterate $\boldsymbol{w}_{k+1}$ is given by

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k) = \boldsymbol{w}_k - \alpha_k \ell'(\boldsymbol{x}_{i_k}^{\mathrm{T}}\boldsymbol{w}_k - y_{i_k})\boldsymbol{x}_{i_k}.$$

e) Every gradient descent iteration must access all data points in order to compute the full gradient. Since our cost unit corresponds to an access to one point $\boldsymbol{x}_i$, the cost of one gradient descent iteration according to this metric is n. As for an iteration of stochastic gradient, its cost is 1 because it only requires one data point (namely $\boldsymbol{x}_{i_k}$ at iteration k, where i_k is the random index drawn at that iteration).

i) When $n \gg 1$ and there are redundancies in the data, it is not necessary to "see" all data points in order to perform optimization. As a result, stochastic gradient can be more efficient than gradient descent, in that it will perform more optimization steps given the same amount of accesses to data points. This is a situation in which stochastic gradient is relevant. *N.B. More broadly, when the data points are correlated, but not necessarily identical, we expect a similar argument to hold in favor of stochastic gradient.*

ii) When $n = d$ and $\boldsymbol{x}_i = \boldsymbol{e}_i$ (where $\boldsymbol{e}_i$ is the ith coordinate vector in $\mathbb{R}^n$ defined by $[\boldsymbol{e}_i]_i = 1$ and $[\boldsymbol{e}_i]_j = 0$ for $i \neq j$), the problem can be rewritten as

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{w} - y_i) = \frac{1}{n}\sum_{i=1}^{n} \ell([\boldsymbol{w}]_i - y_i).$$

It can then be seen that the objective function is a sum of n terms, each involving a different coordinate of $\boldsymbol{w}$. An iteration of gradient descent will then update all coordinates at once, whereas an iteration of stochastic gradient will only modify one (random) coordinate at a time. In that context, gradient descent is more interesting than stochastic gradient. *N.B. Here all terms in the finite sum must be considered to compute the solution of the optimization problem. The data points are independent, and not correlated.*

f) Stochastic gradient is a randomized method, implying that the result of a particular run depends on a random draw of a sequence of indices. As a result, it is possible that a particular run does not converge (even though the theory guarantees convergence in expectation), and this is an explanation for the observed behavior.

i) The kth iteration of a batch stochastic gradient with batch size n_b, starting from a point $\boldsymbol{w}_k \in \mathbb{R}^d$ proceeds as follows. First, a random subset of indices of cardinality n_b is drawn such that $S_k \subset \{1, \dots, n\}^{n_b}$. Then, the next iterate is computed through the formula

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

where $\alpha_k > 0$ is a stepsize.

ii) If n_b are available and the gradients of the f_is can be computed in parallel, then the evaluation of the batch stochastic gradient can be distributed over these n_b processors.

iii) Batch stochastic gradient methods rely on a gradient estimate of the form $\frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(\boldsymbol{w}_k)$. The variance of this estimator (as defined in the lectures) is smaller than that of a standard stochastic gradient estimate, of the form $\nabla f_{i_k}(\boldsymbol{w}_k)$.

iv) If we observe that the convergence improves while increasing the batch size, it means that considering more than one data point is beneficial (typically because of the variance reduction effect, but also because more information is captured by those gradient estimators). However, increasing the batch size too much leads to a drop in performance, as the method then gets more expensive (with a per-iteration cost being significantly higher than stochastic gradient) while being more sensitive to redundancies in the data. This explains that the performance worsens as n_b gets above $n/10$.

Solutions for Exercise 2

a) An epoch is a unit of cost corresponding to n accesses of an example in the data set. Running random reshuffling for one epoch represents a pass through the entire dataset.

b) Consider an iteration index k that corresponds to the first iteration of an epoch (i.e. $k = \ell n$ for some $\ell \in \mathbb{N}$).

i) Since i_k is the first index corresponding to a random permutation, it has equal probability to be equal to any value in $\{1, \dots, n\}$, and it is independent of x_k. As a result, the expected value can be analyzed similarly to the uniform distribution case, and we get

$$\mathbb{E}_{i_k}\left[\nabla f_{i_k}(x_k)\right] = \sum_{i=1}^{n} \frac{1}{n} \times \nabla f_i(x_k) = \nabla f(x_k).$$

ii) Consider the iteration $k+1$. The value of i_{k+1} depends on that of i_k, and therefore the reasoning above no longer holds. The situation complicates even further when considering later indices.

c) The sequence of averaged iterates has less variance than the sequence of iterates.

d) After nK iterations of gradient descent, we would obtain a point x_{nK}^G such that

$$f(x_{nK}^G) - f^* \leq \mathcal{O}\left(\frac{1}{nK}\right),$$

This is a better convergence rate than random reshuffling, in the sense that the quantity to which the rate applies is deterministic and relates to the last iterate, and that the rate converges to zero faster than that of random reshuffling when $K \to \infty$.

On the other hand, one iteration of gradient descent is more expensive than an iteration of random reshuffling. For a fixed number of epochs N_E, random reshuffling performs nN_E iterations while gradient descent performs only N_E iterations. Provided n is sufficiently large, we will have $\sqrt{nN_E} \gg N_E$, and thus the convergence rate will be better for random reshuffling.

e) With a batch size of n, we recover gradient descent since the indices will be drawn without replacement.

f) Advanced stochastic gradient techniques such as `SGD with momentum` and `Adam` are based on combining the current stochastic gradient with past directions chosen in previous iterations, thus they would be good choices for this purpose.