Sampling Strategies for Empirical Risk Minimization
by Dominik Csiba

First supervisor: Peter Richtárik

Machine learning research has received a lot of attention in the past few years. One of the main problems in machine learning is the so-called *supervised learning*. Given an observation $x \in \mathcal{X}$, one wishes to predict its label $y \in \mathcal{Y}$. An example of supervised learning is the task of *object recognition*, where the goal is to classify an object (e.g., cat, flatfish, chihuahua) on an image given a pixel-wise representation of the image. In this case, $\mathcal{X}$ is the set of all images and $\mathcal{Y}$ is the set of all considered object labels.

Formally, our goal is to find a predictor $h : \mathcal{X} \to \mathcal{Y}$, such that $h(x) = y$ for all $x \in \mathcal{X}$. In general, this is a very difficult problem. Many different approaches were proposed to learn a good predictor h. We will focus here on the approach called **E**mpirical **R**isk **M**inimization (ERM). ERM consists of four steps:

(1) Pick a flexible set of predictors $\mathcal{H}$ (e.g., linear predictors, deep learning).
(2) Pick a risk function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ (e.g., squared loss, logistic loss).
(3) Observe a sequence of object-label pairs $\{(x_i, y_i)\}_{i=1}^n$ (e.g., labeled images)
(4) Find the predictor h^* which solves the optimization problem

$$h^* \quad = \quad \arg\min_{h \in \mathcal{H}} \quad \frac{1}{n} \sum_{i=1}^n \ell(h(x_i), y_i) \tag{1}$$

The optimization problem (1) can be interpreted as finding the function h with the most accurate predictions on the *training set* $\{(x_i, y_i)\}_{i=1}^n$. Under some assumptions on the set of predictors $\mathcal{H}$ and the risk function ℓ, one can show [1] that for a large enough training set size, the predictor h^* will perform reasonably well for a random instance $x \in \mathcal{X}$. These assumptions often result in (1) being an optimization task over a high-dimensional domain (millions) with a large amount of summands n (also millions).

For a better grip on (1), the set of functions $\mathcal{H}$ is often considered parametrized, i.e., there exists a set $\mathcal{C} \subset \mathbb{R}^d$ and a function $f : \mathcal{X} \times \mathcal{C} \to \mathcal{Y}$, such that $\mathcal{H} = \{f(\cdot, \mathbf{w}), \mathbf{w} \in \mathcal{C}\}$. This results in the problem of finding the optimal vector $\mathbf{w}^*$ using

$$\mathbf{w}^* \quad = \quad \arg\min_{\mathbf{w} \in \mathcal{C}} \quad \left[P(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \ell(f(x_i, \mathbf{w}), y_i) \right]. \tag{2}$$

Problem (2) does not have a closed-form solution in general, and hence iterative methods are usually needed to find a solution close to $\mathbf{w}^*$. Standard iterative methods, such as *gradient descent* or *Newton's method*, may look like a natural choice, but they suffer from a very high iteration cost. Indeed, consider the computation of a single gradient $\nabla P(\mathbf{w})$. We have to compute d partial derivatives $\frac{\partial}{\partial w_j} P(\mathbf{w})$, each consisting of a sum over n functions of the form $\frac{\partial}{\partial w_j} \ell(f(x_i, \mathbf{w}), y_i)$. Usually, d or n or both are large in practice, which renders the computation of the full gradient $\nabla P(\mathbf{w})$ impractical.

Due to the above considerations the current state-of-the-art methods are performing stochastic updates. There are two main approaches:

- **R**andomized **C**oordinate **D**escent: Randomly sample a single dimension $j \in \{1, \ldots, d\}$ and perform the update $w_j \leftarrow w_j - \eta \frac{\partial}{\partial w_j} P(\mathbf{w})$ for a given stepsize $\eta > 0$, where w_j is the j-th entry of $\mathbf{w}$. The update cost is independent of d.
- **S**tochastic **G**radient **D**escent: Randomly sample a single data-point (x_i, y_i) and perform the update $\mathbf{w} \leftarrow \mathbf{w} - G(\nabla \ell(f(x_i, \mathbf{w}), y_i))$, where $G : \mathbb{R}^d \to \mathbb{R}^d$ is some function of the single sampled gradient. The update cost is independent of n.

The particular random sampling strategy used is a key ingredient in both RCD and SGD. The simplest sampling used in practice is the *uniform sampling*, i.e., choosing each coordinate/data-point with the same probability. While uniform sampling is very popular for its simplicity, it was observed both theoretically and empirically that non-uniform sampling distributions offer better convergence rates. A thorough study of various non-uniform sampling methods and their associated algorithms for ERM is the main topic of my research. I describe our main contributions in the following text.

The sampling which minimizes the number of iterations needed to obtain a certain precision is a data-dependent non-uniform sampling, usually called *importance sampling* in the literature. However, the iteration cost of performing an update can vary over the coordinates/data-points, because of sparsity in the data or other considerations. Therefore, the total time needed to reach any desired accuracy is better modeled by the quantity: number of iterations $\times$ average cost per iteration. In [2], we showed that, surprisingly, for training linear predictors the standard importance sampling also minimizes this new quantity. Also, we performed a joint theoretical study to compare the performance of RCD and SGD for ERM. We showed, that the community belief that RCD outperforms SGD when $d \gg n$ and vice-versa does not necessarily hold for sparse data, and offered a more precise way to differentiate between these cases.

The results described above hold for a fixed sampling distribution over the whole iterative process. In [3], we showed both theoretically and empirically, that by adapting the sampling scheme over the iterative process, one gets potentially a huge speed-up. This was the first work to analyze a non-stationary sampling distribution in the literature. Also, the work has been awarded the 2$^{\text{nd}}$ prize for the best contribution at Optimization and Big Data 2015 workshop in Edinburgh.

Both RCD and SGD can be extended to sample multiple coordinates/data-points at each iteration, often called *minibatches* in the literature. The main advantage of such an approach is that the update for each coordinate/data-point can be computed in parallel or distributed fashion. In [4], we propose such an extension to an SGD-type of method. Also, our analysis allows for what we call "arbitrary" sampling schemes (all subsets of examples can have their dedicated probabilities with which they are chosen). We showcase the advantage of this approach by proposing one such sampling scheme for better load-balancing over parallel cores.

The importance sampling is a standard trick to obtain a faster convergence when sampling a single coordinate/data-point at each iteration. Until recently, there was no importance sampling scheme available for minibatches. In [5], we proposed the first flexible non-uniform sampling scheme for minibatches, capable of gaining a similar speed-up as observed for standard importance sampling.

This sums up my work for the first two years of my PhD. studies.

References

[1] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[2] Dominik Csiba and Peter Richtárik. Coordinate descent face-off: Primal or dual? *arXiv:1605.08982*, 2016.

[3] Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinte ascent with adaptive probabilities. *In proceedings of International Conference on Machine Learning*, 2015.

[4] Dominik Csiba and Peter Richtárik. Primal method for erm with flexible mini-batching schemes and non-convex losses. *arXiv:1506.02227*, 2015.

[5] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *arXiv:1602.02283*, 2016.