EE6367: Topics in Data Storage and Communications **2023**

Lecture 11: 4 October 2023

Instructor: Shashank Vatedka *Scribe: Gautam Singh*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

11.1 Applications of Quantization Problems

We consider some applications of quantization in real-world problems.

11.1.1 Mean Estimation by a Server

Suppose there are m users U_i, each containing a data point $\mathbf{x_i} \in \mathbb{R}^d$ where $\|\mathbf{x_i}\| \leqslant r$. Consider a separate server that wants to compute the mean estimate $\bar{\mathbf{X}}$ from the $\mathbf{x_i}$, where we define

$$\bar{\mathbf{X}} \triangleq \frac{1}{m}\sum_{i=1}^{m} \mathbf{x_i}. \tag{11.1}$$

The goal is to minimise the *mean squared error*, defined as

$$\text{MSE}\left(\mathbf{x_1}, \ldots, \mathbf{x_n}\right) \triangleq \mathbb{E}\left[\left\|\hat{\mathbf{X}} - \bar{\mathbf{X}}\right\|^2\right]. \tag{11.2}$$

One possible scheme is

1. Each user independently quantizes their $\mathbf{x_i}$ to form $\mathbf{y_i}$.
2. The $\mathbf{y_i}$ is transmitted to the server.
3. Server decodes the $\mathbf{y_i}$ and reconstructs
$$\hat{\mathbf{X}} = \frac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{x}}_{\mathbf{i}}. \tag{11.3}$$

For this scheme,

$$\text{MSE} = \mathbb{E}\left[\left\|\sum_{i=1}^{m}\frac{\hat{\mathbf{X}}_{\mathbf{i}}}{m} - \sum_{i=1}^{m}\frac{\mathbf{x_i}}{m}\right\|^2\right] \tag{11.4}$$

$$= \frac{1}{m^2}\mathbb{E}\left[\left\|\sum_{i=1}^{m}\left(\hat{\mathbf{X}}_{\mathbf{i}} - \mathbf{x_i}\right)\right\|^2\right] \tag{11.5}$$

$$= \frac{1}{m^2} \left(\sum_{i=1}^{m} \mathbb{E} \left[\left\| \hat{\mathbf{X}}_{\mathbf{i}} - \mathbf{x}_{\mathbf{i}} \right\|^2 \right] + \sum i = 1^m \sum_{j=i+1}^{m} \left(\hat{\mathbf{X}}_{\mathbf{i}} - \mathbf{x}_{\mathbf{i}} \right)^\top \left(\hat{\mathbf{X}}_{\mathbf{j}} - \mathbf{x}_{\mathbf{j}} \right) \right). \tag{11.6}$$

Considering an unbiased scheme like DRIVE for each user, (11.6) becomes

$$\mathrm{MSE} = \frac{1}{m^2} \sum_{i=1}^{m} \mathrm{MSE} \left(\mathbf{x}_{\mathbf{i}} \right) \tag{11.7}$$

$$= \frac{1}{m} \Theta \left(r^2 \right). \tag{11.8}$$

A better metric is to normalize (11.8) with the squared 2-norm of the true mean. It is possible to achieve lower MSE with shared randomness between users, etc.

11.1.2 Stochastic Gradient Descent

In many machine learning problems, we are given iid samples (x_i, y_i) according to some unknown distribuion p_{XY} where the y_i are observables and x_i is the quantity to be estimated. The goal is to construct an estimator that minimizes average error. Mathematically, if this estimator be parametrized as $g_{\boldsymbol{\beta}}(y)$, we need to find

$$\boldsymbol{\beta}^* \triangleq \arg\min_{\boldsymbol{\beta}} \mathbb{E} \left[l \left(X, g_{\boldsymbol{\beta}} (Y) \right) \right]. \tag{11.9}$$

In *empirical risk minimization*, we restate the problem as

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} l \left(X_i, g_{\boldsymbol{\beta}} (Y_i) \right)}_{f(\boldsymbol{\beta})}. \tag{11.10}$$

It is computationally expensive to calculate $\boldsymbol{\nabla} f(\boldsymbol{\beta})$. Hence, at each step we compute a *stochastic gradient* $\mathbf{h}(\boldsymbol{\beta})$ satisfying

$$\mathbb{E} \left[\mathbf{h} (\boldsymbol{\beta}) \right] = \left(\boldsymbol{\nabla} f \right) (\boldsymbol{\beta}) \ \ \forall \boldsymbol{\beta} \in \mathbf{x}. \tag{11.11}$$

An example of a stochastic gradient can be (where $I \sim \mathrm{Unif}\{1, 2, \ldots, n\}$)

$$\mathbf{h} (\boldsymbol{\beta}) \triangleq \boldsymbol{\nabla}_{\boldsymbol{\beta}} l \left(X_I, g_{\boldsymbol{\beta}} (Y_I) \right). \tag{11.12}$$

If k iid samples are taken as above, the average of the individual stochastic gradients is also a stochastic gradient. This is a widely used technique known as *minibatching*.

11.1.3 Improving Speed of Minibatch SGD

Just like the server mean estimation problem, we assume that there are k distributed GPUs, each with its own dataset D_i. The following scheme is adopted in this problem for iteration $1 \leqslant t \leqslant T$.

1. The server sends $\boldsymbol{\beta_{t-1}}$ to all GPUs.
2. Each GPU computes $\boldsymbol{\nabla}_{\boldsymbol{\beta}} l\left(X_j, g_{\boldsymbol{\beta}}\left(Y_j\right)\right)$ for random samples evaluated at $\boldsymbol{\beta}_{t-1}$.
3. Server updates

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \eta_t \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{\nabla}_{\boldsymbol{\beta}} l\left(X_j, g_{\boldsymbol{\beta}}\left(Y_j\right)\right). \tag{11.13}$$

In this case, a possible bottleneck is in sending the gradients to the server. Quantization can be a workaround to this bottleneck.