MIT 6.7220/15.084 — Nonlinear Optimization (Spring '25)

Lecture 16
Distributed optimization and ADMM

Instructor: Prof. Gabriele Farina

In this last lecture on first-order methods, we will touch on an important aspect of optimization in machine learning: *distributed* optimization.

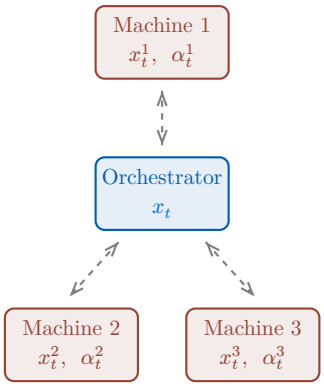
L16.1 Setting

Consider again an empirical risk minimization problem, where we aim to minimize the average loss over a large training set. When the training set is too large to fit on a single machine or privacy concerns require different parts of the data to be stored on different machines, it is common to distribute the optimization problem across multiple machines.

In this setting, we assume that we have m machines, each with access to a local dataset and a local function $f^j : \mathbb{R}^n \to \mathbb{R}$. The goal is to minimize the average of the local functions,

$$f(x) = \frac{1}{m} \sum_{j=1}^{m} f^j(x),$$

where the functions f^j are differentiable.



In our distributed optimization model, each machine can communicate with a central orchestrator, which can send messages to all machines and collect messages from all machines. Furthermore, each machine is powerful enough to solve most optimization problems on their *local* dataset.

Notation: We will denote machine indices as superscripts (these are not powers!) and time indices as subscripts.

L16.1.1 First attempt: independent optimization

One might be tempted to distribute the optimization problem by simply letting each machine compute the optimal solution on their local dataset, and then try to combine the solutions in some way. However, this approach in general has no hope without further assumptions about how the different datasets relate to each other. In the extreme, each solution might

⋆These notes are class material that has not undergone formal peer review. The TAs and I are grateful for any reports of typos.

carry virtually zero information regarding the global optimum; for example, consider a multi-class classification problem where each machine only has access to training data from the same class. Then, a classifier that deterministically outputs the machine's class would be an optimal local solution and yet carry no meaning globally.

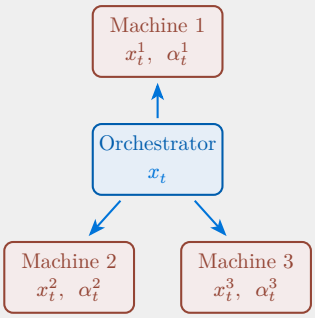
L16.1.2 Second attempt: distributed gradient computation

A more sophisticated approach is to compute the gradient of the loss function at each machine and then combine the gradients at a central location. This approach would work, but has two major drawbacks:

1. The total communication cost is mainly determined by the convergence rate of gradient descent. In particular, when the smoothness or the Lipschitzness of f is not very good, then the convergence rate of gradient descent is not very good. We will discuss today a solution that does not depend on the smoothness of f.
2. Gradient descent does not leverage the computation power of each machine very well. At each iteration, each machine simply computes a gradient. A more cost-effective approach would try to directly use the computational power of each machine to find some sort of optimal solution of the problem (while avoiding the naive approach of independent optimization discussed in Section L16.1.1).

L16.2 The ADMM algorithm

The ADMM algorithm is a distributed optimization algorithm that combines the best of both worlds: it uses the computational power of each machine to find an optimal solution, while also ensuring that the global solution is meaningful. The algorithm can be thought of as a two-step process parameterized by the learning rate $\eta > 0$ and a multiplier $\lambda > 0$, as follows.

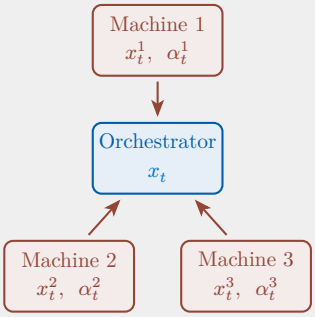


■ **Phase I: Local update.** The orchestrator sends the current x_t to each machine. Each machine computes their updated multiplier

$$\alpha_t^j = \alpha_{t-1}^j - \eta\big(x_t - x_t^j\big)$$

and optimizes on their local training set

$$x_{t+1}^j = \arg\min_{x^j \in \mathbb{R}^n} \Big\{ f^j(x^j) + \lambda \big\| x^j - x_t \big\|_2^2 + \big\langle \alpha_t^j, x^j - x_t \big\rangle \Big\}.$$



■ **Phase II: Synchronization.** Each machine reports their local solution x_{t+1}^j and their local multiplier α_t^j to the orchestrator.

The orchestrator computes

$$x_{t+1} = \frac{1}{m}\left(\sum_{j=1}^m x_{t+1}^j\right) + \frac{1}{2m\lambda}\left(\sum_{j=1}^m \alpha_t^j\right).$$

L16.2.1 ADMM as Lagrangian relaxation

At this point, one might wonder how the update rules of ADMM were picked. The answer is pretty ingenious. ADMM is based on the fundamental idea of rewriting the problem

$$v := \min_x \ \frac{1}{m}\sum_{j=1}^m f^j(x)$$

as the equivalent problem

$$\begin{aligned} \min_{x,x^j} \ & \frac{1}{m}\sum_{j=1}^m \Big(f^j(x^j) + \lambda\|x^j - x\|_2^2\Big) \\ \text{s.t.} \ & x^j = x. \end{aligned} \tag{1}$$

The next step is to relax the equality constraint $x_{t+1}^j = x_{t+1}$ by introducing the *Lagrange relaxation* of the problem, in which the constraint is replaced by a penalization term in the objective function. The following key result shows that the maximum value of the Lagrange relaxation is the same as the value of the original problem when all f^j are convex.

Theorem L16.1. Assuming all f^j are convex, the optimal value of the original problem is equal to the maximum of the Lagrange function $\mathcal{L}$ over all multipliers α^j, that is,

$$v = \max_{\alpha^1,\dots,\alpha^m} \min_{x,x^j} \left\{ \mathcal{L}(x, x^j; \alpha^j) := \frac{1}{m}\sum_{j=1}^m \Big(f^j(x^j) + \lambda\|x^j - x\|_2^2 + \langle \alpha^j, x^j - x\rangle\Big) \right\}.$$

In light of this result, the ADMM algorithm can be seen as a way to maximize the Lagrange function $\mathcal{L}$ over the multipliers α^j by performing alternating optimization steps on different groups of variables:

- Assuming fixed x_t and x_t^j, the multipliers α_{t-1}^j are updated by gradient ascent, performing a step in the direction of the gradient of the objective function:

$$\alpha_t^j = \alpha_{t-1}^j + \eta\big(x_t^j - x_t\big) = \alpha_{t-1}^j - \eta\big(x_t - x_t^j\big).$$

- Assuming now fixed α_t^j and x_t, each machine now *solves* the optimization problem

$$x_{t+1}^j = \arg\min_{x^j \in \mathbb{R}^n} \Big\{ f^j(x^j) + \lambda\|x^j - x_t\|_2^2 + \big\langle \alpha_t^j, x^j - x_t\big\rangle \Big\}.$$

- Assuming now fixed α_t^j and x_{t+1}^j, the orchestrator computes the next iterate x_t by solving the optimization problem

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{m} \sum_{j=1}^m f^j(x^j) + \lambda \|x^j - x\|_2^2 + \langle \alpha_t^j, x^j - x \rangle \right\}$$
$$= \arg\min_{x \in \mathbb{R}^n} \left\{ \frac{\lambda}{m} \sum_{j=1}^m \|x^j - x\|_2^2 - \langle \alpha_t^j, x \rangle \right\}.$$

  By setting the gradient with respect to x to 0 and solving for x, we obtain

$$\frac{1}{m} \sum_{j=1}^m \big(2\lambda(x - x^j) - \alpha_t^j\big) = 0$$

  So, the optimal x_{t+1} is written in closed form as

$$x_{t+1} = \frac{1}{m} \sum_{j=1}^m x_{t+1}^j + \frac{1}{2m\lambda} \sum_{j=1}^m \alpha_t^j.$$

L16.3 Analysis (Optional)

The analysis of the correctness of ADMM can be broken down into several step. Perhaps unsurprisingly at this point, at its core the proof revolves around some appropriate generalization of the Euclidean mirror descent lemma.

L16.3.1 Part I: Orchestrating machine

Theorem L16.2. If the initial multipliers α_0^j are chosen such that $\sum_{j=1}^m \alpha_0^j = 0$, then, at all times t,

$$\sum_{j=1}^m \alpha_t^j = 0, \qquad \text{and} \qquad x_{t+1} = \frac{1}{m} \sum_{j=1}^m x_{t+1}^j.$$

Proof. By definition,

$$\alpha_{t+1}^j = \alpha_t^j - \eta\big(x_{t+1} - x_{t+1}^j\big).$$

So, averaging over $j = 1, ..., m$, we can write

$$\frac{1}{m} \sum_{j=1}^m \alpha_{t+1}^j = \frac{1}{m} \sum_{j=1}^m \alpha_t^j - \eta \left(x_{t+1} - \frac{1}{m} \sum_{j=1}^m x_{t+1}^j \right). \tag{2}$$

Using the definition of x_{t+1}, we also have

$$x_{t+1} = \left(\frac{1}{m} \sum_{j=1}^m x_{t+1}^j \right) + \frac{1}{2m\lambda} \sum_{j=1}^m \alpha_t^j \qquad \Longrightarrow \qquad x_{t+1} - \frac{1}{m} \sum_{j=1}^m x_{t+1}^j = \frac{1}{2m\lambda} \sum_{j=1}^m \alpha_t^j. \tag{3}$$

Substituting the last expression into (2), we get

$$\frac{1}{m}\sum_{j=1}^{m}\alpha_{t+1}^{j}=\frac{1}{m}\sum_{j=1}^{m}\alpha_{t}^{j}-\frac{\eta}{2m\lambda}\sum_{j=1}^{m}\alpha_{t}^{j}=\frac{1}{m}\Big(1-\frac{\eta}{2\lambda}\Big)\sum_{j=1}^{m}\alpha_{t}^{j}.$$

This shows that by induction, $\sum_{j=1}^{m}\alpha_{t}^{j}=0$ at all t as long as the assumption holds for $t=0$.

Plugging the expression $\sum_{j=1}^{m}\alpha_{t}^{j}=0$ into (3), then yields

$$x_{t+1}=\frac{1}{m}\sum_{j=1}^{m}x_{t+1}^{j}$$

at all times, which is the second part of the claim. □

L16.3.2 Part II: Worker machines

Theorem L16.3 (Euclidean mirror descent lemma for ADMM)**.** Let each $f^j:\mathbb{R}^n\to\mathbb{R}$ be convex, and $\eta=2\lambda$. Then, for any $y\in\mathbb{R}^n$ we have

$$\frac{1}{m}\sum_{j=1}^{m}f^j\Big(x_{t+1}^{j}\Big)\le f(y)+\frac{\eta}{2}\Big(\|y-x_t\|_2^2-\|y-x_{t+1}\|_2^2-\|x_t-x_{t+1}\|_2^2\Big)$$
$$+\frac{1}{m}\sum_{j=1}^{m}\frac{1}{2\eta}\Big(\|\alpha_t^j\|_2^2-\|\alpha_{t+1}^j\|_2^2-\|\alpha_{t+1}^j-\alpha_t^j\|_2^2\Big).$$

Proof. Recall that each machine computes the update

$$x_{t+1}^{j}=\operatorname*{arg\,min}_{x^j\in\mathbb{R}^n}\Big\{f^j(x^j)+\lambda\|x^j-x_t\|_2^2+\big\langle\alpha_t^j,x^j-x_t\big\rangle\Big\}.$$

As the domain is open, the first-order necessary optimality conditions imply that

$$\nabla f^j\Big(x_{t+1}^{j}\Big)+2\lambda\Big(x_{t+1}^{j}-x_t\Big)+\alpha_t^j=0. \tag{4}$$

Since f^j is convex, the linear lower bound property of convex functions we can write

$$\begin{aligned}f^j(y)&\ge f^j\Big(x_{t+1}^{j}\Big)+\Big\langle\nabla f^j\Big(x_{t+1}^{j}\Big),y-x_{t+1}^{j}\Big\rangle\\&=f^j\Big(x_{t+1}^{j}\Big)+\Big\langle-2\lambda\Big(x_{t+1}^{j}-x_t\Big)-\alpha_t^j,y-x_{t+1}^{j}\Big\rangle.\qquad\text{(from (4))}\end{aligned}$$

Rearranging the terms and using the hypothesis $\eta=2\lambda$, we get

$$f^j\Big(x_{t+1}^{j}\Big)\le f^j(y)+\eta\Big\langle x_{t+1}^{j}-x_t,y-x_{t+1}^{j}\Big\rangle+\Big\langle\alpha_t^j,y-x_{t+1}^{j}\Big\rangle. \tag{5}$$

At this point, we can use the update rule $\alpha_{t+1}^{j}=\alpha_t^j-\eta\Big(x_{t+1}-x_{t+1}^{j}\Big)$ to write

$$x_{t+1}^{j}=x_{t+1}+\frac{\alpha_{t+1}^{j}-\alpha_t^j}{\eta}.$$

Substituting the above expression into the first inner product of (5) then yields

$$f^j\left(x^j_{t+1}\right) \le f^j(y) + \eta\left\langle x_{t+1} - x_t, y - x^j_{t+1}\right\rangle + \left\langle \alpha^j_{t+1}, y - x^j_{t+1}\right\rangle.$$

(Note that change in time index from α^j_t to α^j_{t+1} in the last term.) Substituting the expression for x^j_{t+1} into the last inner product, we finally obtain

$$f^j\left(x^j_{t+1}\right) \le f^j(y) + \eta\left\langle x_{t+1} - x_t, y - x^j_{t+1}\right\rangle + \left\langle \alpha^j_{t+1}, y - x_{t+1}\right\rangle - \frac{1}{\eta}\left\langle \alpha^j_{t+1}, \alpha^j_{t+1} - \alpha^j_t\right\rangle. \quad (6)$$

From Theorem L16.2, we know that

$$\frac{1}{m}\sum_{j=1}^m \left\langle \alpha^j_{t+1}, y - x_{t+1}\right\rangle = 0, \qquad \text{and} \qquad \frac{1}{m}\sum_{j=1}^m \left\langle x_{t+1} - x_t, y - x^j_{t+1}\right\rangle = \left\langle x_{t+1} - x_t, y - x_{t+1}\right\rangle$$

for all t. Furthermore, we know that $f(y) = \frac{1}{m}\sum_{j=1}^m f^j(y)$. Hence, averaging (6) across all machines $j = 1, ..., m$, we obtain

$$\frac{1}{m}\sum_{j=1}^m f^j\left(x^j_{t+1}\right) \le f(y) + \eta\left\langle x_{t+1} - x_t, y - x_{t+1}\right\rangle - \frac{1}{m\eta}\sum_{j=1}^m \left\langle \alpha^j_{t+1}, \alpha^j_{t+1} - \alpha^j_t\right\rangle.$$

Subtituting the three-point equalities

$$\left\langle x_{t+1} - x_t, y - x_{t+1}\right\rangle = \frac{1}{2}\left(\|x_t - y\|_2^2 - \|x_{t+1} - y\|_2^2 - \|x_t - x_{t+1}\|_2^2\right)$$

$$\text{and} \qquad -\left\langle \alpha^j_{t+1}, \alpha^j_{t+1} - \alpha^j_t\right\rangle = \frac{1}{2}\left(\left\|\alpha^j_t\right\|_2^2 - \left\|\alpha^j_{t+1}\right\|_2^2 - \left\|\alpha^j_{t+1} - \alpha^j_t\right\|_2^2\right)$$

yields the statement. □

L16.3.3 Putting the pieces together: telescoping step

The sum on the right-hand side of Theorem L16.3 telescopes. In particular, since $\alpha^j_0 = 0$ for all j, we have

$$\frac{1}{mT}\sum_{t=0}^{T-1}\sum_{j=1}^m f^j\left(x^j_{t+1}\right) \le f(y) + \frac{\eta}{2T}\|y - x_0\|_2^2 - \frac{1}{2\eta mT}\sum_{t=0}^{T-1}\sum_{j=1}^m \left\|\alpha^j_{t+1} - \alpha^j_t\right\|_2^2.$$

Since by definition $\alpha^j_{t+1} - \alpha^j_t = \eta\left(x^j_{t+1} - x_{t+1}\right)$, the last term satisfies

$$-\frac{1}{2\eta mT}\sum_{t=0}^{T-1}\sum_{j=1}^m \left\|\alpha^j_{t+1} - \alpha^j_t\right\|_2^2 = -\frac{\eta}{2mT}\sum_{t=0}^{T-1}\sum_{j=1}^m \left\|x^j_{t+1} - x_{t+1}\right\|_2^2,$$

leading to

$$\frac{1}{mT}\sum_{t=0}^{T-1}\sum_{j=1}^m f^j\left(x^j_{t+1}\right) \le f(y) + \frac{\eta}{2T}\|y - x_0\|_2^2 - \frac{\eta}{2mT}\sum_{t=0}^{T-1}\sum_{j=1}^m \left\|x^j_{t+1} - x_{t+1}\right\|_2^2.$$

Rearranging the terms and applying $y = x_\star$, we therefore have

$$\frac{1}{T}\sum_{t=0}^{T}\left(\frac{1}{m}\sum_{j=1}^m f^j\left(x^j_{t+1}\right) + \frac{\eta}{2}\left\|x^j_{t+1} - x_{t+1}\right\|_2^2\right) \le f(x_\star) + \frac{\eta}{2T}\|x_\star - x_0\|_2^2.$$

Thus, we have the following guarantee.

Theorem L16.4. Let each $f^j : \mathbb{R}^n \to \mathbb{R}$ be convex and nonnegative, $\lambda = \sqrt{T}$ and $\eta = 2\lambda$. Then, for any $T > 0$, at least one of the iterates $t \in \{0, ..., T-1\}$ satisfies

$$\frac{1}{m}\sum_{j=1}^{m} f^j\left(x_{t+1}^j\right) \leq f(x_\star) + \frac{1}{\sqrt{T}}\|x_\star - x_0\|_2^2,$$

$$\text{and} \qquad \frac{1}{m}\sum_{j=1}^{m} \left\|x_{t+1}^j - x_{t+1}\right\|_2^2 \leq \frac{1}{\sqrt{T}} f(x_\star) + \frac{1}{T}\|x_\star - x_0\|_2^2.$$

So, as time progresses, the iterates x_{t+1}^j concentrate around the average x_{t+1}, and the average of the function values $f^j\left(x_{t+1}^j\right)$, which concentrates around $f(x_{t+1})$, converges to the optimal value at a rate of $\frac{1}{\sqrt{T}}$.

L16.4 Further readings

The review article by Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., & others. [Boy+11] is a modern and approachable introduction to ADMM and its variants and extensions.

[Boy+11] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., & others. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122.

6.7220 Nonlinear Optimization

Spring 2025