Convex Optimization 10-725, Lecture 9: Distributed Optimization

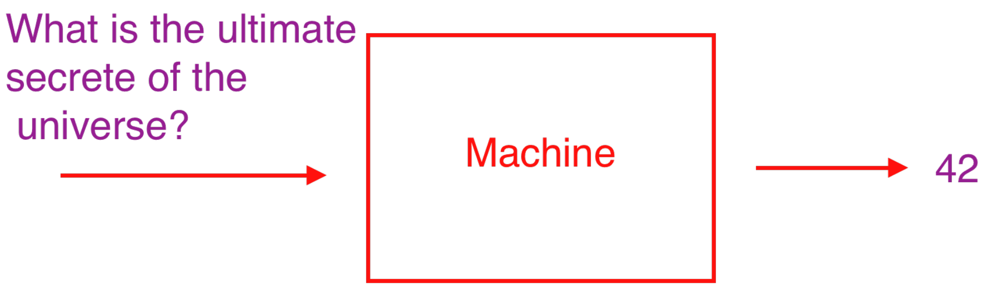
Yuanzhi Li

Assistant Professor, Carnegie Mellon University

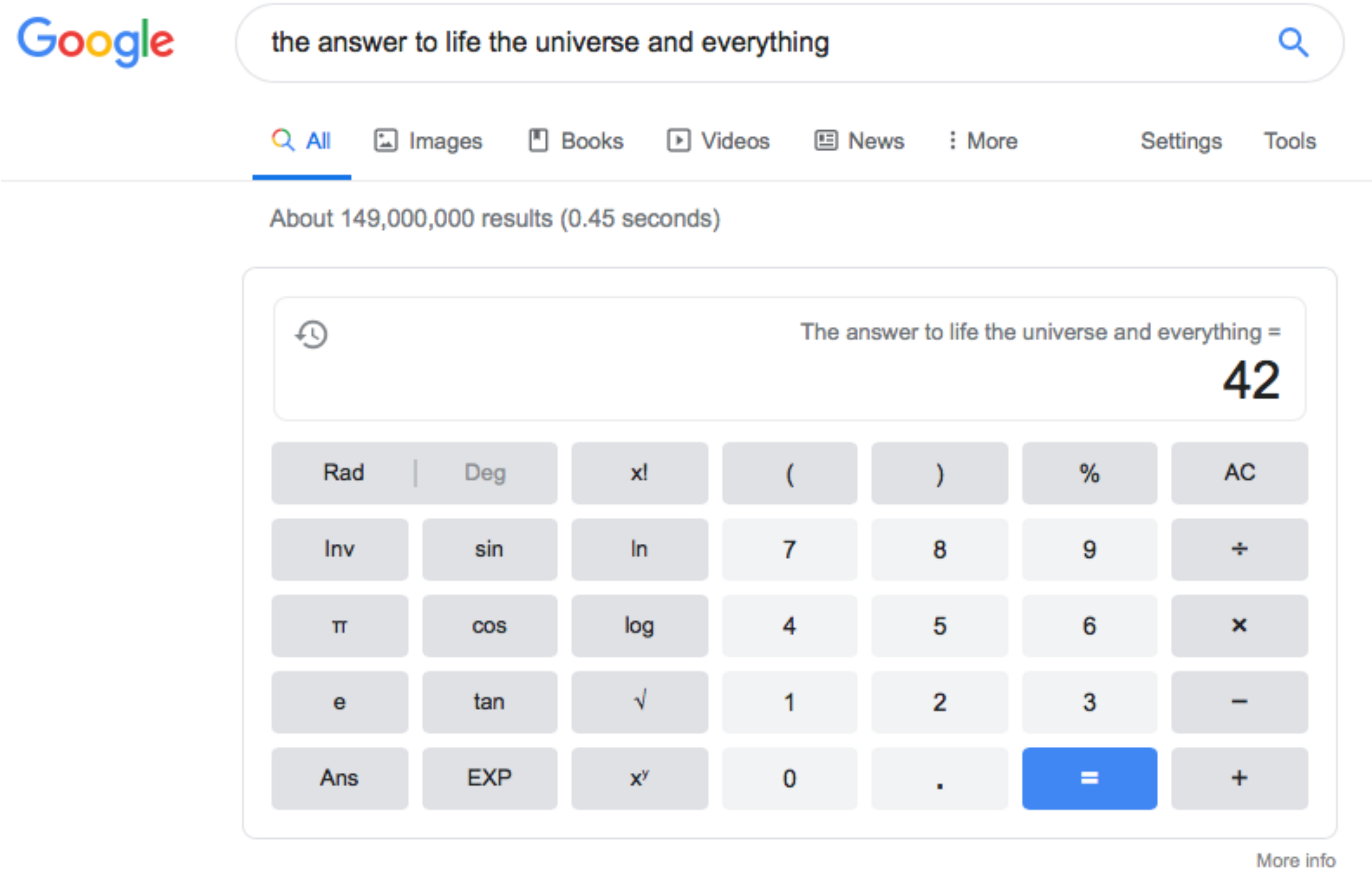
Today

Last lecture

- We learnt duality and min-max optimization. Today, we are going to see a real application.
- Before that, we learnt the Stochastic Gradient Descent Algorithm.
- It is the most important algorithm in machine learning.
- Reason: the Machine has limited computation power, so evaluating the full gradient for ERM-type problems at each iteration is too expensive.
- Question: What if the Machine has infinite computational power, then what optimization algorithm do we use? (Or do we even need to use any optimization algorithms?)



Machine Learning with infinite computation power



Machine Learning with infinite computation power

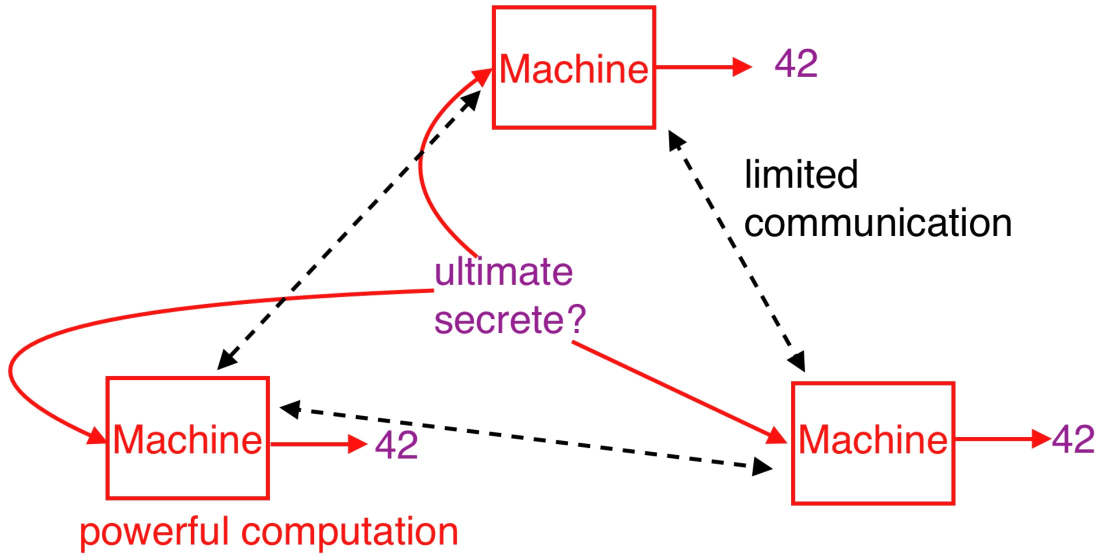
- We are going to study this case, in the situation of distributed optimization.
- Where we do assume that each machine has close to infinite computation power and can optimize any function very fast, using some hard-coded algorithm.
- Why do we still need another optimization algorithm?

Machine Learning with infinite computation power

- Recall the ERM type of problem:

$$\min_{W} \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, W), y_i) + R(W)$$

- Key challenge: In many cases, when N is extremely large, the data $\{x_i, y_i\}_{i=1}^{N}$ can only be stored on different machines.
- Each individual machine has really strong computation power, but the communication between the machines are very limited. (In particular, the data set S_j can not be transmitted from one machine to another).



-

Machine Learning with infinite computation power

- Recall the ERM type of problem:

$$\min_W \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, W), y_i) + R(W)$$

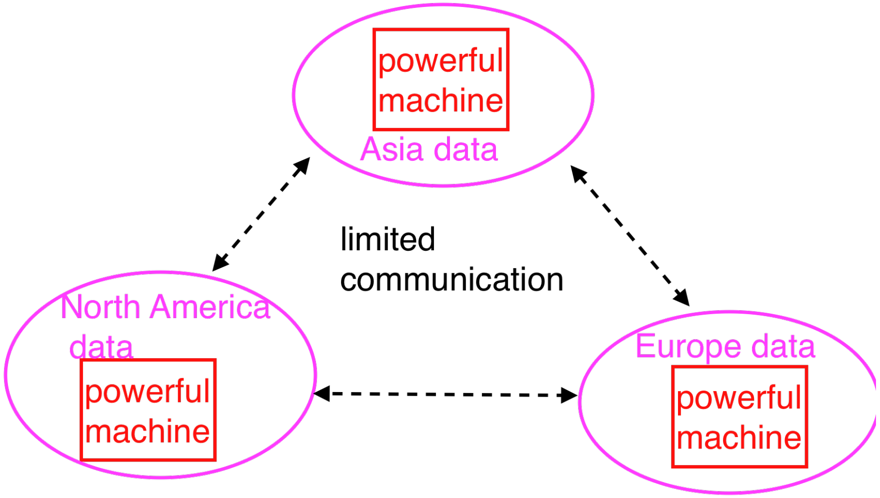
- Besides memory issue, there is another reason that data $\{x_i, y_i\}_{i=1}^N$ should be stored on different machines and should not be put in the same machine.
- Alice, Bob and Charlie each collects $N/3$ data points, and their data might contain their private information (location, time, style, homework grade etc.) they do not want to share with the others.
- Can they still find the minimizer $\min_W \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, W), y_i) + R(W)$ over all the N data points?

Distributed optimization

- Recall the ERM type of problem:

$$\min_{W} \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, W), y_i) + R(W)$$

- The data $\{x_i, y_i\}_{i=1}^{N}$ are stored on m different machines, the machines each having disjoint data set $S_1, S_2, \cdots, S_m$ where $\cup_{j \in [m]} S_j = [N]$.
- Each individual machine has really strong computation power (assuming to be infinite), but the communication between the machines are very limited.



-
- What can we do in this case?

Distributed optimization

- Recall the ERM type of problem:

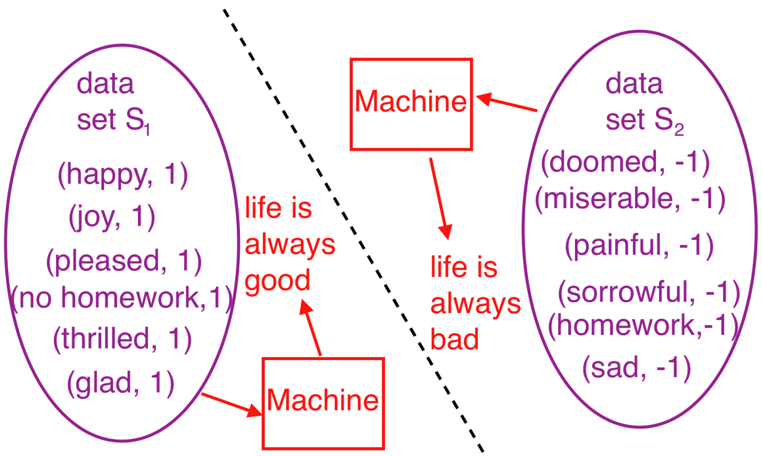
$$\min_{W} \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, W), y_i) + R(W)$$

- The data $\{x_i, y_i\}_{i=1}^{N}$ are stored on m different machines, the machines each having disjoint data set $S_1, S_2, \cdots, S_m$ where $\cup_{j \in [m]} S_j = [N]$.
- This is called distributed optimization, where each individual machine is assumed to have infinite computation power, but the communication between the machines are limited.
- The goal is to find the minimizer of the problem and minimize the communication between the machines.

Distributed optimization

- First try: Each machine minimizes

$$\min_W \frac{1}{|S_j|} \sum_{i \in S_j} \ell(h(x_i, W), y_i) + R(W)$$

  to obtain W_j^*, and combine them in some way.
- Not going to work at all – The data's are not distributed randomly.
- Consider classification problem, with $m = 2$ and $y \in \{-1, 1\}$, now S_1 can contain all the data with label 1, and S_2 contains all the data with label -1.
- 

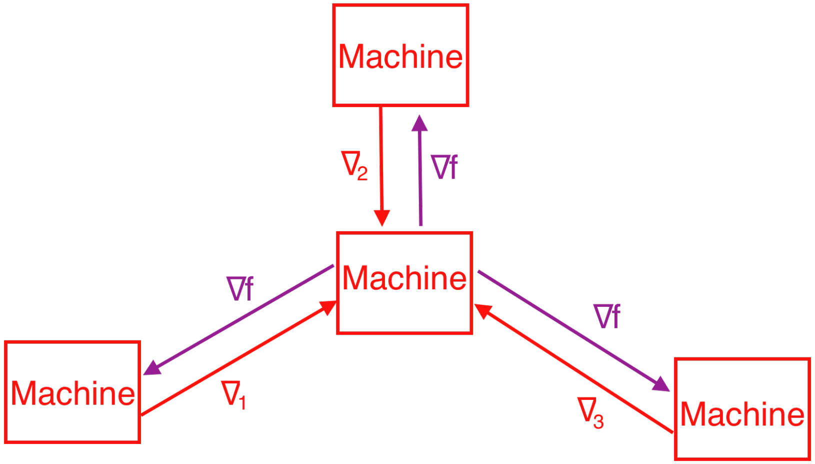
- Minimizing $\min_W \frac{1}{|S_j|} \sum_{i \in S_j} \ell(h(x_i, W), y_i) + R(W)$ for each S_j can get meaningless results.

Distributed optimization

- Second try: At each iteration, each machine computes

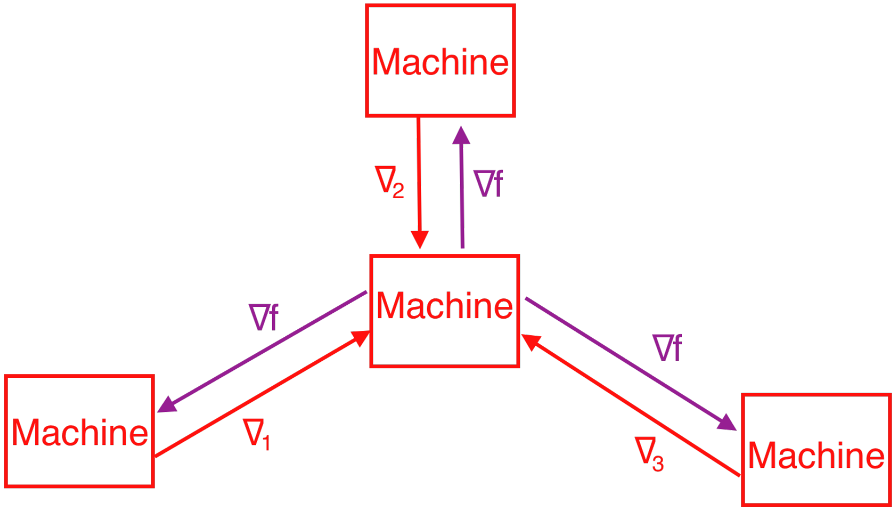
$$\nabla_j = \frac{1}{N} \sum_{i \in S_j} \nabla \ell(h(x_i, W_t), y_i)$$

  and send to a central server (or just one leading machine).
- The central server computes $\nabla f(W_t) = \sum_{j \in [m]} \nabla_j + \nabla R(W_t)$ and send it back to each machine, each machine can then update using Gradient Descent: $W_{t+1} = W_t - \eta \nabla f(W_t)$.



-
- Per iteration communication cost: $O(md)$, where $W \in \mathbb{R}^d$. Total number of iterations: The convergence rate of gradient descent.

Distributed optimization using gradient descent



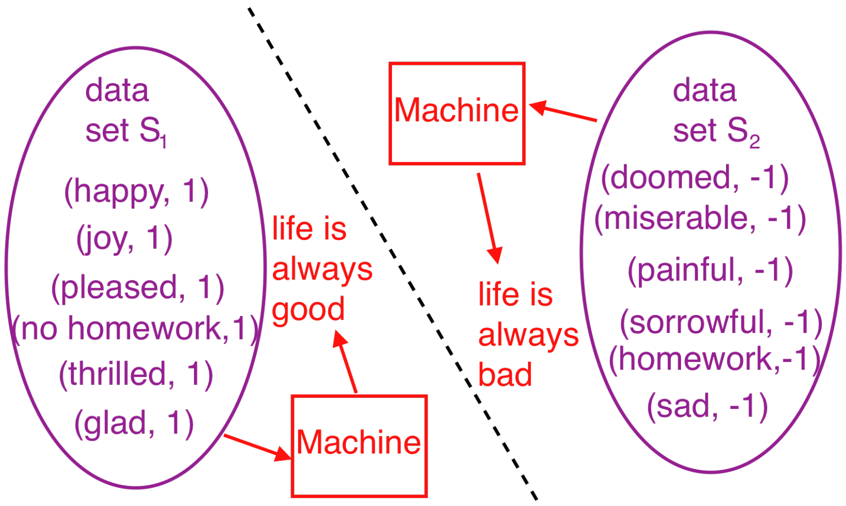
-
- Using gradient descent, the per iteration communication cost: $O(md)$, where $W \in \mathbb{R}^d$. Total number of iterations: The convergence rate of gradient descent.
- The per iteration communication cost is very good, the total communication cost is mainly determined by the convergence rate of 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.

Improve gradient descent for distributed optimization

- Key observation: Gradient descent does not use the infinite computation power of each individual machine very well.
- At each iteration, each machine simply computes

$$\nabla_j = \frac{1}{N}\sum_{i \in S_j} \nabla \ell(h(x_i, W_t), y_i)$$

- In principle, at each iteration, we should directly use the computation power of each machine to find some sort of *minimizer* of $\min_W \frac{1}{|S_j|}\sum_{i \in S_j} \ell(h(x_i, W), y_i) + R(W)$ directly.
- But recall, it won't work either...



Distributed optimization: The real approach

- Now we are going to see some approach that actually works, and the convergence rate does not depend on the smoothness or the Lipschitzness of f. Let us denote for each $j \in [m]$:

$$f_j(W) = \frac{m}{N} \sum_{i \in S_j} \ell(h(x_i, W), y_i) + R(W)$$

- Recall the m machines each having disjoint data set $S_1, S_2, \cdots, S_m$ where $\cup_{j \in [m]} S_j = [N]$, so we have: $f(W) = \frac{1}{m} \sum_{j \in [m]} f_j(W)$
- Key observation: minimizing $f(W)$ is equivalent to minimizing:

$$\frac{1}{m} \sum_{j \in [m]} f_j(W_j), s.t. \forall j \in [m] : W_j = W$$

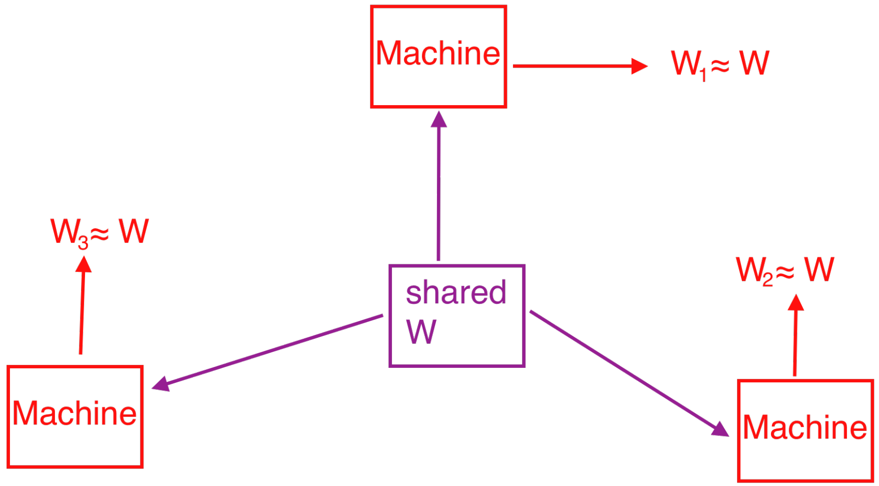
- Which is equivalent to minimizing over $\{W_j\}_{j \in [m]}, W$ (for any $\lambda \geq 0$):

$$\frac{1}{m} \sum_{j \in [m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 \right), s.t. \forall j \in [N] : W_j = W$$

Distributed optimization: The real approach

- Now we reduce the problem to minimizing over $\{W_j\}_{j\in[m]}, W$

$$\frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j) + \lambda\|W_j - W\|_2^2\right), s.t. \forall j \in [N] : W_j = W$$

- How do individual machine minimize $f_j(W_j) + \lambda\|W_j - W\|_2^2$ and also keep all W_j equal to W?
- How does the algorithm really work? What λ to pick?
- 

Distributed optimization: The dual approach

- Now we want to minimize

$$\min_{\{W_j\}_{j\in[m]},W} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2\right), s.t. \forall j\in[N]: W_j = W$$

- By KKT condition/theorem, this can be done by

$$(*) \max_{\{\alpha_j\}_{j\in[m]}} \min_{\{W_j\}_{j\in[m]},W} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2+\langle\alpha_j, W_j-W\rangle\right)$$

- Let us further define (for $\alpha = (\alpha_1, \cdots, \alpha_j)$):

$$G(\alpha) = -\min_{\{W_j\}_{j\in[m]},W} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2+\langle\alpha_j, W_j-W\rangle\right)$$

- So we just want to find the minimizer of $G(\alpha)$.
- We will do it using gradient descent.

Distributed optimization: The dual approach

- Now we want to minimize

$$\min_{\{W_j\}_{j\in[m]},W} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2\right), s.t. \forall j\in[N]: W_j = W$$

- By KKT condition/theorem, this can be done by

$$(*) \max_{\{\alpha_j\}_{j\in[m]}} \min_{\{W_j\}_{j\in[m]},W} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2+\langle\alpha_j, W_j-W\rangle\right)$$

- Reason of adding $\lambda\|W_j-W\|_2^2$: Making sure that the minimizer $\operatorname{argmin}_{\{W_j\}_{j\in[m]},W}\frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2+\langle\alpha_j, W_j-W\rangle\right)$ is a continuous function of α, so we can apply KKT condition/theorem.

Distributed optimization: The dual approach

- How do we compute the gradient of $G(\alpha)$?
- Key fact: For a (differentiable) function h and a matrix A, if

$$G(\alpha) = -\min_X \{h(X) + \langle \alpha, AX \rangle\}$$

- Then we have (fenchel duality): for $X^* = \operatorname{argmin}\{h(X) + \langle \alpha, AX \rangle\}$.

$$\nabla G(\alpha) = -AX^*$$

- Proof: Let X_α be a minimizer of $h(X) + \langle \alpha, AX \rangle$, then we have:

$$\nabla_X h(X_\alpha) + A^\top \alpha = 0$$

- On the other hand, $G(\alpha) = -h(X_\alpha) - \langle \alpha, AX_\alpha \rangle$, so we have:

$$\begin{aligned}\nabla G(\alpha) &= -\nabla_\alpha X_\alpha^\top \nabla_X h(X_\alpha) - AX_\alpha - \nabla_\alpha X_\alpha^\top A^\top \alpha \\ &= -\nabla_\alpha X_\alpha^\top \left(\nabla_X h(X_\alpha) + A^\top \alpha\right) - AX_\alpha = -AX_\alpha\end{aligned}$$

- Computing the gradient of $G(\alpha)$ = Finding the minimizer of $h(X) + \langle \alpha, AX \rangle$.

Distributed optimization: The dual approach

- Key fact: For a (differentiable) function h and a matrix A, if

$$G(\alpha) = -\min_X h(X) + \langle \alpha, AX \rangle$$

- Then we have: for $X^* = \operatorname{argmin}\{h(X) + \langle \alpha, AX \rangle\}$

$$\nabla G(\alpha) = -AX^*$$

- Recall in our application:

$$G(\alpha) = -\min_{\{W_j\}_{j\in[m]}, W} \frac{1}{m} \sum_{j\in[m]} \left(f_j(W_j) + \lambda \|W_j - W\|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- Apply the fact with $X = (W_1, W_2, \cdots, W_m, W)$, $AX = (W_1 - W, \cdots, W_m - W)$, $h(X) = \frac{1}{m} \sum_{j\in[m]} \left(f_j(W_j) + \lambda \|W_j - W\|_2^2 \right)$, we have: for $\{W_j^*\}_{j\in[m]}, W^*$ being the minimizer of

$$\frac{1}{m} \sum_{j\in[m]} \left(f_j(W_j) + \lambda \|W_j - W\|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- It holds that $\nabla_{\alpha_j} G(\alpha) = W^* - W_j^*$.

Distributed optimization: The dual approach

- Recall in our application:

$$G(\alpha) = -\min_{\{W_j\}_{j\in[m]}, W} \frac{1}{m} \sum_{j\in[m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- To optimize G, we just need to solve

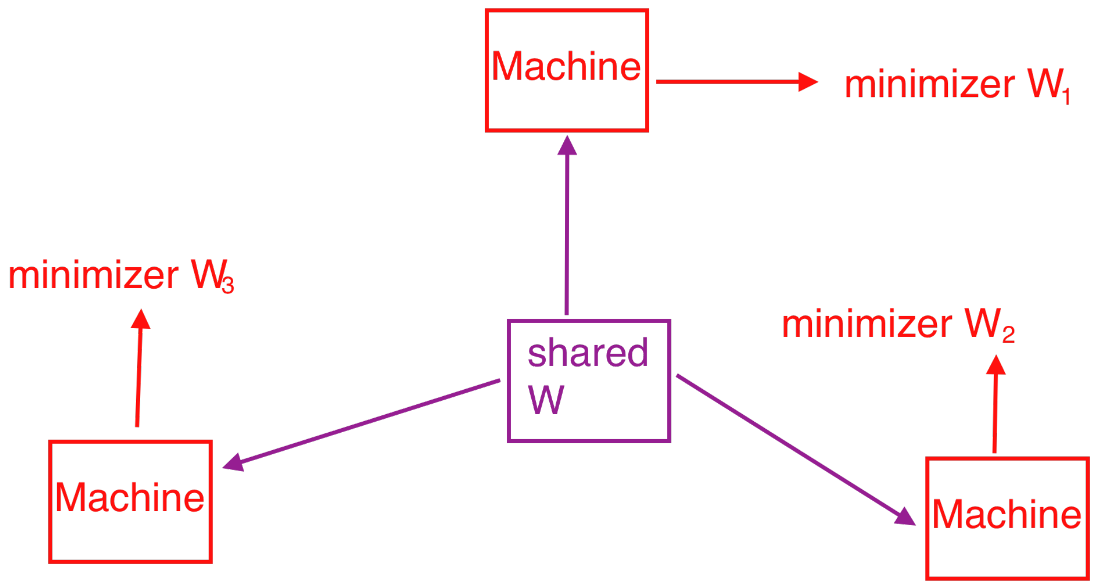
$$\min_{\{W_j\}_{j\in[m]}, W} \frac{1}{m} \sum_{j\in[m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- How fast can we do in the distributed setting?

Distributed optimization: The dual approach

- Key observation: fixing W,

$$\min_{\{W_j\}_{j\in[m]}} \frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j)+\lambda\|W_j-W\|_2^2+\langle\alpha_j, W_j-W\rangle\right)$$

- Only depends on f_j and W_j.
- Can be done locally in each machine.
- 

Distributed optimization: The dual approach

- Key observation: fixing $W_1, \cdots, W_j$,

$$\min_W \frac{1}{m} \sum_{j \in [m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- Also has a simple solution: the minimizer W^* satisfies:

$$\sum_{j \in [m]} (2\lambda(W^* - W_j) - \alpha_j) = 0$$

- Which is

$$W^* = \frac{\sum_{j \in [m]} \alpha_j}{2m\lambda} + \frac{1}{m} \sum_{j \in [m]} W_j$$

- We just do alternative minimization over $\{W_j\}_{j \in [m]}$ and W.

Distributed optimization: ADMM

- This gives us the algorithm (Basic) ADMM (Alternating Direction Method of Multipliers):
- At each iteration t, maintain a shared $W^{(t)}$ on across different machines, and local $\{W_j^{(t)}, \alpha_j^{(t)}\}$ on each machine.
- At each iteration t, each machine compute locally compute:

$$W_j^{(t+1)} = \mathrm{argmin}_{W_j} \left(f_j(W_j) + \lambda \| W_j - W^{(t)} \|_2^2 + \langle \alpha_j^{(t)}, W_j - W^{(t)} \rangle \right)$$

- All the machine together compute

$$W^{(t+1)} = \frac{\sum_{j \in [m]} \alpha_j^{(t)}}{2m\lambda} + \frac{1}{m} \sum_{j \in [m]} W_j^{(t+1)}$$

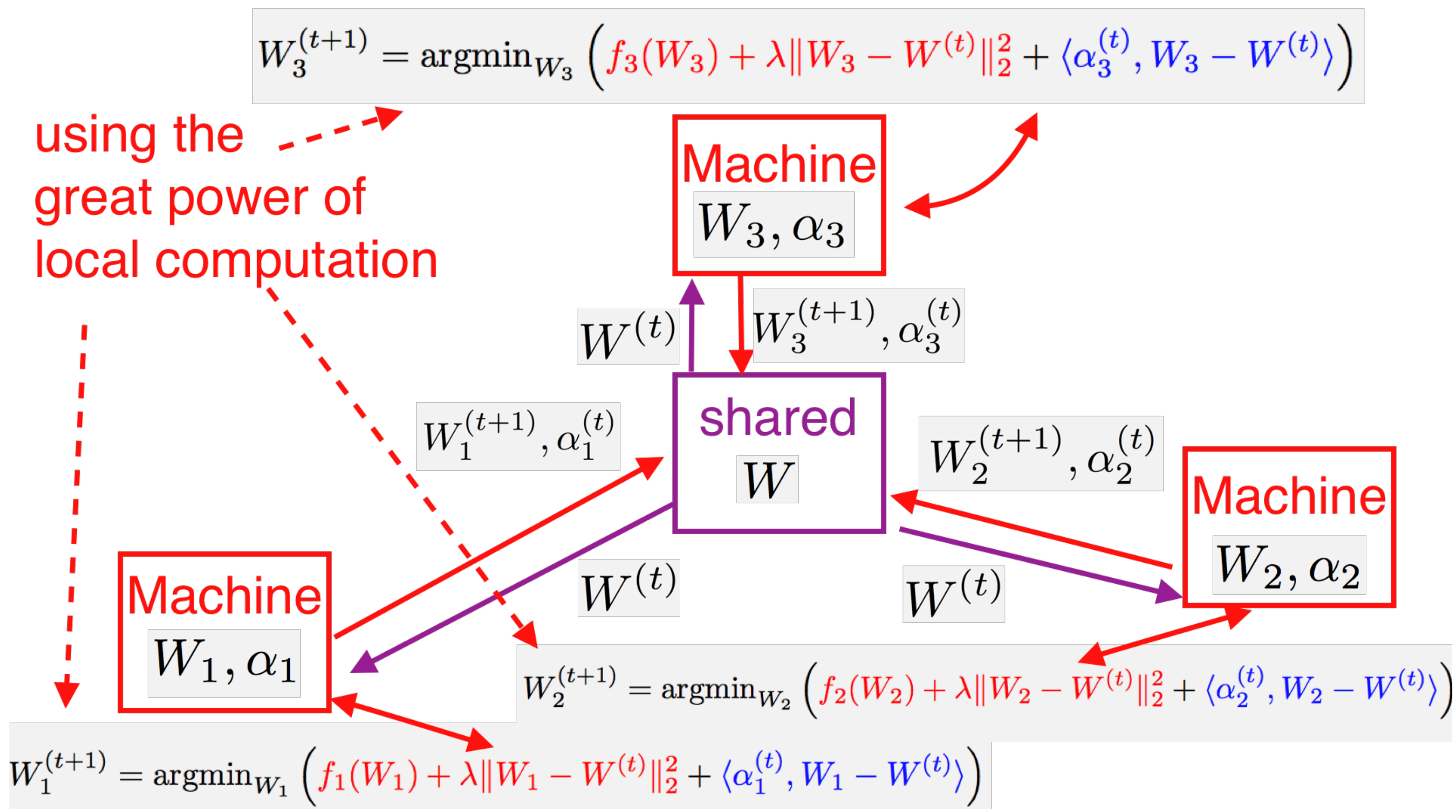
- Each machine update

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} - \eta \left(W^{(t+1)} - W_j^{(t+1)} \right)$$

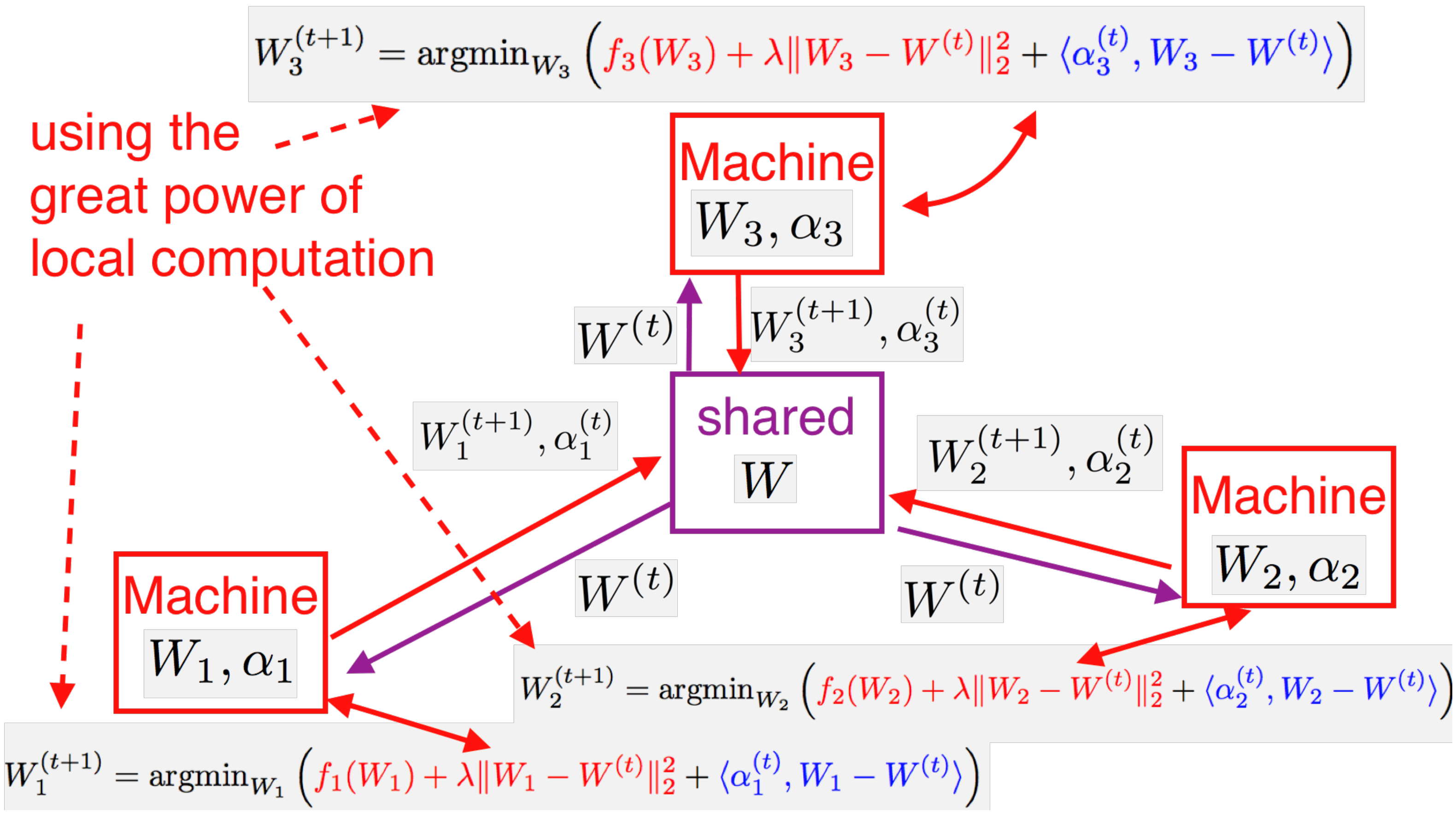
- Individual minimizer $\rightarrow$ shared minimizer $\rightarrow$ gradient ascent.

Distributed optimization: ADMM

- At each iteration t, each machine compute locally compute:
  $W_j^{(t+1)} = \mathrm{argmin}_{W_j} \left(f_j(W_j) + \lambda \| W_j - W^{(t)} \|_2^2 + \langle \alpha_j^{(t)}, W_j - W^{(t)} \rangle \right)$.
- Machines together compute $W^{(t+1)} = \frac{\sum_{j \in [m]} \alpha_j^{(t)}}{2m\lambda} + \frac{1}{m} \sum_{j \in [m]} W_j^{(t+1)}$.
- Each machine update $\alpha_j^{(t+1)} = \alpha_j^{(t)} - \eta \left(W^{(t+1)} - W_j^{(t+1)} \right)$.



Distributed optimization: ADMM



- Each iteration, machines only needs to communicate $O(md)$ bits of message.
- And we are going to see how the convergence rate of the ADMM does not depend on the Lipschitzness/Smoothness of f.

Distributed optimization: ADMM

- Basically, the algorithm first minimize on each machine:

$$\min_{W_j} \sum_{j \in [m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- Then all machines together minimizes (using a closed form formula):

$$\min_{W} \frac{1}{m} \sum_{j \in [m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

- And then does a gradient descent step on α with respect to:

$$G(\alpha) = -\min_{\{W_j\}_{j \in [m]}, W} \frac{1}{m} \sum_{j \in [m]} \left(f_j(W_j) + \lambda \| W_j - W \|_2^2 + \langle \alpha_j, W_j - W \rangle \right)$$

  The gradient is computed using the *minimizer* W_j, W found in the previous steps.
- But why does it work? How to chose the best η, λ?
- Warning: The following calculations should not be performed at home (a.k.a. it will not appear in the Homework).

Distributed optimization: ADMM

- Critical observation about $\alpha_j^{(t)}$: by definition, we have

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} - \eta\left(W^{(t+1)} - W_j^{(t+1)}\right)$$

- Average over $j = 1, 2, \cdots, m$, we have:

$$\frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t+1)} = \frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t)} - \eta\left(W^{(t+1)} - \frac{1}{m}\sum_{j\in[m]}W_j^{(t+1)}\right)$$

- By definition again, we also have

$$W^{(t+1)} = \frac{\sum_{j\in[m]}\alpha_j^{(t)}}{2m\lambda} + \frac{1}{m}\sum_{j\in[m]}W_j^{(t+1)}$$

- Therefore,

$$\frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t+1)} = \frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t)} - \eta\frac{\sum_{j\in[m]}\alpha_j^{(t)}}{2m\lambda}$$

Distributed optimization: ADMM

- Now we have:

$$\frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t+1)} = \frac{1}{m}\sum_{j\in[m]}\alpha_j^{(t)} - \eta\frac{\sum_{j\in[m]}\alpha_j^{(t)}}{2m\lambda}$$

- Thus, if we initially pick $\alpha_j^{(0)} = 0$ for all j, then

$$\sum_{j\in[m]}\alpha_j^{(t)} = 0$$

holds for all $t \geq 0$, thus, we are actually updating W using an average of W_j:

$$W^{(t+1)} = \frac{\sum_{j\in[m]}\alpha_j^{(t)}}{2m\lambda} + \frac{1}{m}\sum_{j\in[m]}W_j^{(t+1)} = \frac{1}{m}\sum_{j\in[m]}W_j^{(t+1)}$$

- Now we move on to look at individual $W_j^{(t)}$:

Distributed optimization: ADMM

- Recall that

$$W_j^{(t+1)} = \text{argmin}_{W_j} \left(f_j(W_j) + \lambda \| W_j - W^{(t)} \|_2^2 + \langle \alpha_j^{(t)}, W_j - W^{(t)} \rangle \right)$$

- This implies that (calculating the gradient):

$$\nabla f_j(W_j^{(t+1)}) + 2\lambda(W_j^{(t+1)} - W^{(t)}) + \alpha_j^{(t)} = 0$$

- Therefore,

$$\nabla f_j(W_j^{(t+1)}) = -2\lambda(W_j^{(t+1)} - W^{(t)}) - \alpha_j^{(t)}$$

- We will them use Mirror Descent Analysis to show the convergence rate of ADMM on *convex functions*.

Distributed optimization: ADMM

- Now we have

$$\nabla f_j(W_j^{(t+1)}) = -2\lambda(W_j^{(t+1)} - W^{(t)}) - \alpha_j^{(t)}$$

- When each f_j is a *convex* function, by the lower linear bound, for every W:

$$f_j(W) \geq f_j(W_j^{(t+1)}) + \langle \nabla f_j(W_j^{(t+1)}), W - W_j^{(t+1)} \rangle$$

- Therefore,

$$f_j(W) \geq f_j(W_j^{(t+1)}) + \left\langle \left(-2\lambda(W_j^{(t+1)} - W^{(t)}) - \alpha_j^{(t)}\right), W - W_j^{(t+1)} \right\rangle$$

Distributed optimization: ADMM

- Now we have

$$f_j(W) \geq f_j(W_j^{(t+1)}) + \left\langle \left(-2\lambda(W_j^{(t+1)} - W^{(t)}) - \alpha_j^{(t)}\right), W - W_j^{(t+1)} \right\rangle$$

- Which says that

$$f_j(W_j^{(t+1)}) \leq f_j(W) + 2\lambda\langle W_j^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle + \langle \alpha_j^{(t)}, W - W_j^{(t+1)}\rangle$$

- By our update rule,

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} - \eta\left(W^{(t+1)} - W_j^{(t+1)}\right)$$

- Which implies that

$$W_j^{(t+1)} = W^{(t+1)} + \frac{\alpha_j^{(t+1)} - \alpha_j^{(t)}}{\eta}$$

Distributed optimization: ADMM

- Now we have

$$f_j(W_j^{(t+1)}) \le f_j(W) + 2\lambda\langle W_j^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle + \langle \alpha_j^{(t)}, W - W_j^{(t+1)}\rangle$$

$$W_j^{(t+1)} = W^{(t+1)} + \frac{\alpha_j^{(t+1)} - \alpha_j^{(t)}}{\eta}$$

- Taking $\eta = 2\lambda$, and replacing the blue terms:

$$f_j(W_j^{(t+1)}) \le f_j(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle + \langle \alpha_j^{(t+1)}, W - W_j^{(t+1)}\rangle$$

Distributed optimization: ADMM

- Now we have

$$f_j(W_j^{(t+1)}) \leq f_j(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle + \langle \alpha_j^{(t+1)}, W - W_j^{(t+1)}\rangle$$

$$W_j^{(t+1)} = W^{(t+1)} + \frac{\alpha_j^{(t+1)} - \alpha_j^{(t)}}{\eta}$$

- Replacing the blue terms again, we have

$$f_j(W_j^{(t+1)}) \leq f_j(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle + \langle \alpha_j^{(t+1)}, W - W^{(t+1)}\rangle$$

$$-\frac{1}{\eta}\langle \alpha_j^{(t+1)}, \alpha_j^{(t+1)} - \alpha_j^{(t)}\rangle$$

Distributed optimization: ADMM

- Now we have:

$$f_j(W_j^{(t+1)}) \leq f_j(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W_j^{(t+1)}\rangle$$

$$+\langle \alpha_j^{(t+1)}, W - W^{(t+1)}\rangle - \frac{1}{\eta}\langle \alpha_j^{(t+1)}, \alpha_j^{(t+1)} - \alpha_j^{(t)}\rangle$$

- Average over j, using the critical observation that $\sum_{j\in[m]} \alpha_j^{(t+1)} = 0$ and $\frac{1}{m}\sum_{j\in[m]} W_j^{(t+1)} = W^{(t+1)}$, we have:

$$\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)}) \leq f(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W^{(t+1)}\rangle$$

$$-\frac{1}{m}\sum_{j\in[m]} \frac{1}{\eta}\langle \alpha_j^{(t+1)}, \alpha_j^{(t+1)} - \alpha_j^{(t)}\rangle$$

Distributed optimization: ADMM

- Now we have:

$$\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)}) \leq f(W) + \eta\langle W^{(t+1)} - W^{(t)}, W - W^{(t+1)}\rangle$$

$$-\frac{1}{m}\sum_{j\in[m]}\frac{1}{\eta}\langle \alpha_j^{(t+1)}, \alpha_j^{(t+1)} - \alpha_j^{(t)}\rangle$$

- This gives us the (six terms) Mirror Descent Lemma for ADMM: For every W and η, if $\eta = 2\lambda$:

$$\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)}) \leq f(W)$$

$$+\frac{\eta}{2}\left(\|W - W^{(t)}\|_2^2 - \|W - W^{(t+1)}\|_2^2 - \|W^{(t)} - W^{(t+1)}\|_2^2\right)$$

$$+\frac{1}{m}\sum_{j\in[m]}\frac{1}{2\eta}\left(\|\alpha_j^{(t)}\|_2^2 - \|\alpha_j^{(t+1)}\|_2^2 - \|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2\right)$$

Distributed optimization: ADMM

- Now we have the Mirror Descent Lemma for ADMM: For every W and η, if $\eta = 2\lambda$:

$$\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)}) \le f(W)$$

$$+\frac{\eta}{2}\left(\|W - W^{(t)}\|_2^2 - \|W - W^{(t+1)}\|_2^2 - \|W^{(t)} - W^{(t+1)}\|_2^2\right)$$

$$+\frac{1}{m}\sum_{j\in[m]}\frac{1}{2\eta}\left(\|\alpha_j^{(t)}\|_2^2 - \|\alpha_j^{(t+1)}\|_2^2 - \|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2\right)$$

- This is a "minus $\|W^{(t)} - W^{(t+1)}\|_2^2$" instead of plus because we are minimizing W_j on each local machine, and together to obtain $W^{(t)}$, instead of doing a gradient step on $W^{(t)}$.

Distributed optimization: ADMM

- Now we have:

$$\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)}) \le f(W)$$

$$+\frac{\eta}{2}\left(\|W - W^{(t)}\|_2^2 - \|W - W^{(t+1)}\|_2^2 - \|W^{(t)} - W^{(t+1)}\|_2^2\right)$$

$$+\frac{1}{m}\sum_{j\in[m]}\frac{1}{2\eta}\left(\|\alpha_j^{(t)}\|_2^2 - \|\alpha_j^{(t+1)}\|_2^2 - \|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2\right)$$

- This is a telescoping sum, so we have (since $\alpha_j^{(0)} = 0$):

$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)})\right) \le f(W)$$

$$+\frac{\eta}{2T}\left(\|W - W^{(0)}\|_2^2\right) - \frac{1}{2\eta m T}\sum_{t=0}^{T-1}\|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2$$

Distributed optimization: ADMM

- Now we have:

$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)})\right) \le f(W)$$

$$+\frac{\eta}{2T}\left(\|W - W^{(0)}\|_2^2\right) - \frac{1}{2\eta m T}\sum_{t=0}^{T-1}\|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2$$

- By definition, $\alpha_j^{(t+1)} - \alpha_j^{(t)} = \eta(W_j^{(t+1)} - W^{(t+1)})$, so we have:

$$-\frac{1}{2\eta m T}\sum_{t=0}^{T-1}\|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2 = -\frac{\eta}{2mT}\sum_{t=0}^{T-1}\|W_j^{(t+1)} - W^{(t+1)}\|_2^2$$

Distributed optimization: ADMM

- Eventually, we obtain:

$$\frac{1}{T}\sum_{t=0}^{T-1}\left(\frac{1}{m}\sum_{j\in[m]} f_j(W_j^{(t+1)})\right) \le f(W) + \frac{\eta}{2T}\left(\|W - W^{(0)}\|_2^2\right)$$

$$-\frac{\eta}{2mT}\sum_{t=0}^{T-1}\|W_j^{(t+1)} - W^{(t+1)}\|_2^2$$

- Thus, there must be an iteration $t \le T-1$ where

$$\frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j^{(t+1)}) + \frac{\eta}{2}\|W_j^{(t+1)} - W^{(t+1)}\|_2^2\right)$$

$$\le f(W) + \frac{\eta}{2T}\left(\|W - W^{(0)}\|_2^2\right)$$

Distributed optimization: ADMM

- Now we conclude there must be an iteration $t \le T - 1$ where

$$\frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j^{(t+1)}) + \frac{\eta}{2}\|W_j^{(t+1)} - W^{(t+1)}\|_2^2\right)$$

$$\le f(W) + \frac{\eta}{2T}\left(\|W - W^{(0)}\|_2^2\right)$$

- Suppose each f_j is non-negative, taking $\eta = 2\sqrt{T}$, we know that

$$\frac{1}{m}\sum_{j\in[m]}\|W_j^{(t+1)} - W^{(t+1)}\|_2^2 \le \frac{1}{\sqrt{T}}f(W) + \frac{1}{T}\left(\|W - W^{(0)}\|_2^2\right)$$

$$\frac{1}{m}\sum_{j\in[m]}\left(f_j(W_j^{(t+1)})\right) \le f(W) + \frac{1}{\sqrt{T}}\|W - W^{(0)}\|_2^2$$

- The convergence rate does not depend on the smoothness/ Lipschitzness of the function f. This is because we have already find the *minimizer* of f_j locally.

Summary

- Today we learn the distributed optimization, and the algorithm (Basic) ADMM, which utilize the powerful local machines to minimize the communications.
- Distributed optimization is a active research field now.
- Other problems in distributed optimization: Failure, Asynchronous machines, or the local machines are not infinitely powerful, so we want to trade local computation times with communications.