MS&E 317/CS 263: Algorithms for Modern Data Models, Spring 2014
`http://msande317.stanford.edu`.
Instructors: Ashish Goel and Reza Zadeh, Stanford University.

Lecture 12, 5/12/2014. Scribed by Sam Steyer.

12.1 Alternating Direction Method of Multipliers (ADMM)

Suppose we start with an objective function

$$f(x) = \sum_{i=1}^{N} f_i(x)$$

and we refer to $f_i()$ as the ith block of the objective function. This family of objective functions can help solve a lot of different machine learning problems.

Now we define b, the number of machines we have access to, which is $O(10,000)$ (if we are lucky). Then we have the number of features, M which is $O(10^8)$.

In order to minimize the amount of communication between the machines, we reformulate the problem as

$$\min \sum_{i=1}^{N} f_i(x)$$

s.t. $\forall i x_i - z = 0$.

In this case, z is a global variable of dimension M. The x_i are specific to the machine, so there are b distinct x_i.

We then run the following recursive algorithm:

$$\begin{aligned} x_i^{(k+1)} &= \text{argmin}_{x_i} \left(f(x_i) + y_i^{(k)T}(x)i - \bar{x}^{(k)} + \frac{p}{2}\|x_i - \bar{x}^{(k)}\|_2^2 \right) && (1) \\ y_i^{(k+1)} &= y_i^{(k)} + p\left(x_i^{(k+1)} - \bar{x}^{(k+1)} \right) && (2) \\ z_i^{(k+1)} &= \bar{x}^{(k+1)} && (3) \end{aligned}$$

NOTE: parenthesis are included to indicate that the superscripts are iteration markers, not exponents.

12.2 Components of ADMM

This solution requires $\bar{x}$ to be re-computed every round. Reza will post a proof that it converges to the website. The first term of the top equation:

$$f(x_i) + y_i^{(k)T}(x)i - \bar{x}^{(k)}$$

is the Lagrangian term for a local minimization. Any of the standard optimization methods will work.

The second term,

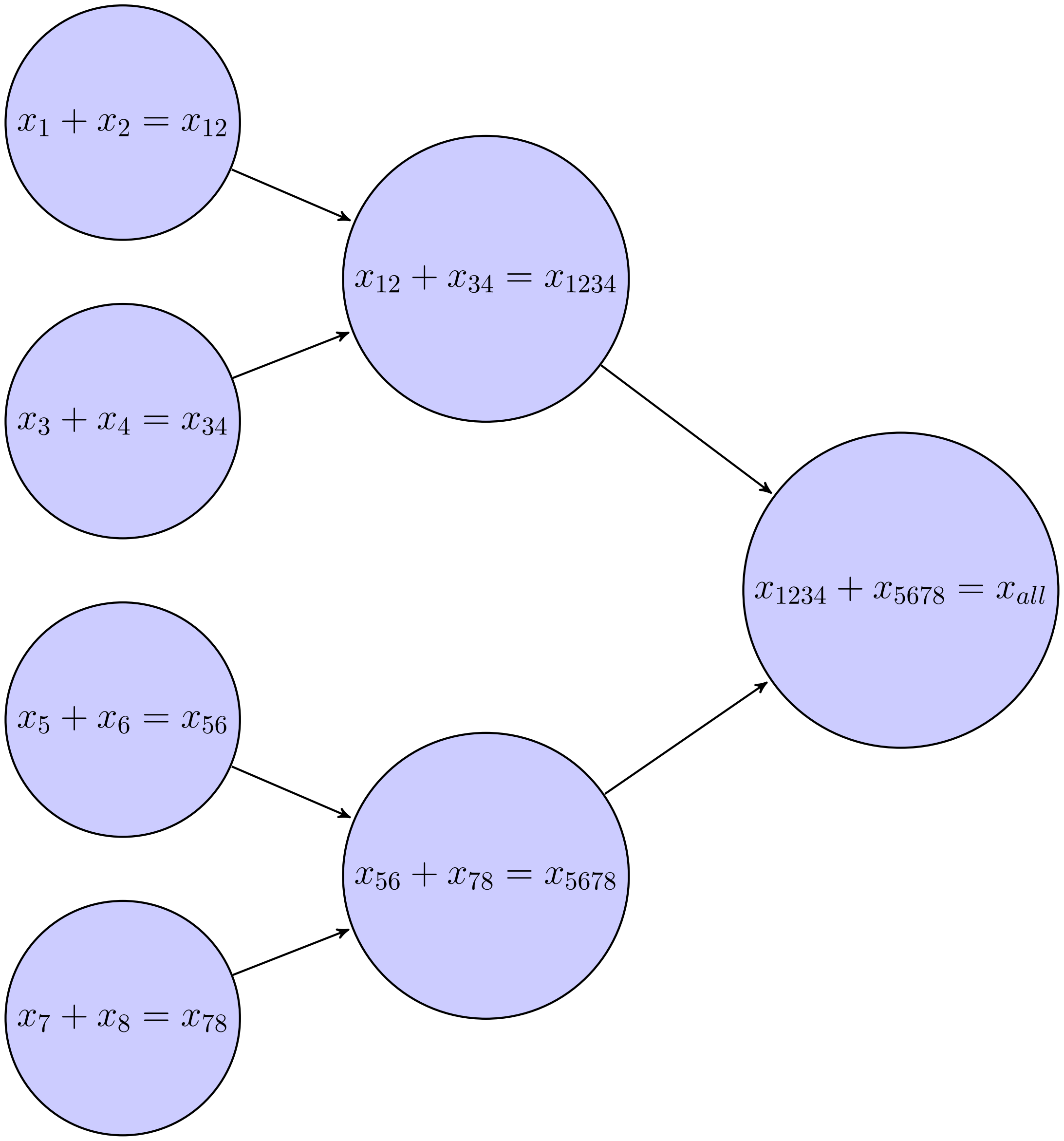
$$\frac{p}{2}\|x_i - \bar{x}^{(k)}\|_2^2$$

is a penalty function to enforce the constraint that $\forall i x_i - z = 0$. If the x_i are far from $\bar{x}$, it adds to the sum and thus hurts the minimization. To guarantee convergence, the weight on the penalty function, p, should be "pretty large," but does not need to go to ∞.

12.3 Calculating $\bar{x}$ Using All Reduce

How do we calculate $\bar{x}^{(k)}$? It requires all of the x_is, so it can't be done on one machine. In order to do this, we use an *all reduce*, an important function for getting averages of large variables.

Note that the computationally difficult part of calculating an average is the sum. to get the sum in an all reduce, we add 2 variables on one machine and then pass our result to another machine to add to another sum of two variables. We repeat this process until we have the sum of all the variables. For example, if we have 8 variables, we can complete the addition in 3 rounds:


$x_1 + x_2 = x_{12}$
$x_3 + x_4 = x_{34}$
$x_5 + x_6 = x_{56}$
$x_7 + x_8 = x_{78}$
$x_{12} + x_{34} = x_{1234}$
$x_{56} + x_{78} = x_{5678}$
$x_{1234} + x_{5678} = x_{all}$

In this way, an all reduce requires $\log_2(b)$ rounds to compute $\bar{x}$.

12.4 Data Classification with ADMM

ADMM can be used for classification of data (ie. sorting emails as spam and not spam). In order to use it, we define our data as a data point $a_i \in \mathbb{R}^n$ and a classification $b_i \in [-1, 1]$. We then calculate

$$\min \sum_{i=1}^{n} Loss(b_i(a_i^T w + v)) + r(w))$$

where $r(w)$ is a regularization term in either L_1 or L_2. We then use ADMM making each x_i a training block of data.

12.5 Statistical Interpretation of ADMM

Maximum Likelihood Estimator: The maximum likelihood estimator, or *MLE*, given an observed outcome x (or group of outcomes $x_1, x_2, ..., x_n$) is the value of the set of unknown parameters of the random variable that would make the outcome most likely (examples of parameters: the bounds of a uniform random variable, or the mean and variance of a Gaussian distribution.).

If we have a prior for the parameters, ie. a distribution that gives us a sense of the probability that the parameters hold certain values, then the MLE is called a **Maximum Aposteriori Estimation** (MAP).

An example that demonstrates both concepts: Suppose we have a coin with an unknown probability of landing heads, p, and we have observed it flipped $1,000$ times. Without a prior,

$$mle(x_i) = \frac{\#heads}{n}$$

But if we have a prior about the value of p that estimation changes. As a trivial example, if I tell you $p = .85$, then we will be able to know p from the prior (frankly, something of this example must have been lost in my notetaking).

ADMM can be used to perform MAP with a prior

$$\mathcal{N}(\bar{x}^k + \frac{1}{p} y_i^k, pI)$$

12.6 Loose Ends

1. How do you do stochastic gradient descent in a distributed way?

   Shuffle: take a random uniform draw for each item, send the first $\frac{1}{b}$ to the machine 1...

   Perform local optimization

   Take the average

2. Note that the average of averages taken from sub-samples of the same size is equal to the average of the whole sample.