On consensum

Giulio Galvan

30 settembre 2015

In a *consesum* inspired way, we can rewrite the RNNs training problem as:

$$\begin{aligned} \min_{W, W_i} \quad & f(W_1, W_2, \dots, W_L) \\ \text{subject to} \quad & W_i = W, \quad i = 1, \dots, L \end{aligned}, \tag{1}$$

considering only W^{rec} for easiness. Lhe Lagrangian can then be written as:

$$\mathcal{L}(W, W_1, \dots, W_L, \lambda_1, \dots, \lambda_L) = f(W_1, W_2, \dots, W_L) + \sum_{i=1}^{L} \lambda_i^T (W - W_i). \tag{2}$$

So problem 1 is equivalent to:

$$\min_{W, W_i, \lambda_i} \mathcal{L}(W, W_1, \dots, W_L, \lambda_1, \dots, \lambda_L). \tag{3}$$

Algorithm 1 shows a Gauss-Seidel like decomposition method which optimizes a function $f(\cdot)$ w.r.t a block of variables i in a iteratively fashion. It can be shown that the algorithm converges if $\mathcal{L}_0$ is compact.

We could apply algorithm 1 to problem 3, using as blocks of variables $W^i, i = 1, ..., L$ where L is the length of the sequence (together with the lambdas), and the master variable W. However we cannot have L of such matrix in memory so we have to devise a modification of the algorithm like I tried to do in Algorithm 2. Note that, because of the peculiar structure of the network, we can compute $\nabla_i f(z)$ even without storing z which is what we cannot do, if we loop in a bottom-up style and the upper matrices share the same value. (it is easy to see, I will write something about it). So the only real problem remain line 10: essentially we must ensure that at the end of one inner loop (the i one) all the matrices share the same value as the master variable.

Now, of course, we have to specify a meaningful way to update the master variable, one simple example could be averaging, somehow similarly to what done in ADMM, over W_i:

$$W = \frac{1}{L} \sum_{i=1}^{L} W_i, \tag{4}$$

or, maybe, using lambdas as weights (because higher lambdas means...):

$$W = \frac{1}{L \sum_{i=i}^{L} \frac{1}{\|\lambda_i\|}} \sum_{i=1}^{L} W_i \cdot \frac{1}{\|\lambda_i\|}. \tag{5}$$

Algorithm 1: Gauss-Seidel like decomposition method

Data:
$x^0 \in \mathbb{R}^m$: candidate solution
$k \leftarrow 0$
while *stop criterion* **do**
    $z \leftarrow x^k$
    for $i = 1, ..., m$ **do**
        $d_i^k \leftarrow -\nabla_i f(z)$ (j components are fixed)
        compute α_i^k with line search
        $x_i^{k+1} \leftarrow x_i^k + \alpha_i^k d_i^k$
        $z_i \leftarrow x_i^{k+1}$
    end
    $x^{k+1} \leftarrow z$
    $k \leftarrow k + 1$
end

Algorithm 2: RNN consensum-decomposition method

Data:
$W = W^0$ candidate solution
$k \leftarrow 0$
while *stop criterion* **do**
    $z \leftarrow (W^k, W_1^k, W_2^k, \ldots W_L^k)$ (virtual assignment)
    $l \leftarrow (\lambda_1^k, \lambda_2^k, \ldots, \lambda_L^k)$ (virtual assignment)
    for $i = 1, ..., L$ **do**
        $d_i^k \leftarrow -\nabla_i \mathcal{L}(z, l))$
        compute α_i^k with line search
        $W_i^{k+1}, \lambda_i^{k+1} = W_i^k, \lambda_i^k + \alpha_i^k d_i^k$
        $z_i = W_i^{k+1}$ (virtual assignment)
        $l_i = \lambda_i^{k+1}$ (virtual assignment)
        update W and l, and store information used to compute $\nabla_{i+1} \mathcal{L}(z, l)$
    end
    $k \leftarrow k + 1$
end