SCALTRA

Leonardo Di Nino

May 2024

Consider the batch optimization problem of a neural network parametrized by a vector of weights $f(w, x)$ over a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ (we'll add back the regularization term later on):

$$\min_{w} \sum_{i=1}^{N} \mathcal{L}(y_i, f(w, x_i) \tag{1}$$

In order to deploy a Successive Convex Approximation (SCA) procedure, we need to define a proper surrogate.

If we consider a second order expansion as a surrogate for the loss (namely $\mathcal{L}_i(w) = \mathcal{L}(y_i, f(x_i, w))$:

$$\widetilde{\mathcal{L}}_i(w|w^t) = \mathcal{L}(w^t) + \nabla\mathcal{L}(w^t)^T(w - w^t) + \frac{1}{2}(w - w^t)^T(bb^T + \tau\mathbf{I})(w - w^t) \tag{2}$$

it's easy to verify that all the requirements of a proper strongly convex surrogate are satisfied.

In particular, the quadratic form is positive definite by design, yielding strong convexity, and the first term expansion ensures that $\nabla_w\widetilde{\mathcal{L}}_i(w|w^t) = \nabla_w\mathcal{L}_i(w)$ at every iteration point.

Our SCA framework is just the following recursion:

$$\hat{w}(w^t) = \operatorname*{argmin}_{w} \sum_{i=1}^{N} \widetilde{\mathcal{L}}_i(w|w^t) \tag{3}$$

$$w^{t+1} = w^t + \gamma^t[\hat{w}(w^t) - w^t] \tag{4}$$

Moreover, we know that the SCA framework subsumes many well known routine in large scale optimization. In particular, focusing on equation (3) we can find back a batch quasi-Newton method:

$$\nabla_w \sum_{i=1}^{N} \widetilde{\mathcal{L}}_i(w|w^t) = 0 \tag{5}$$

$$\nabla_w \sum_{i=1}^{N} [\mathcal{L}_i(w^t) + \nabla \mathcal{L}_i(w^t)^T (w - w^t) + \frac{1}{2}(w - w^t)^T (bb^T + \tau \mathbb{I})(w - w^t)] = 0 \tag{6}$$

$$\sum_{i=1}^{N} [(\nabla \mathcal{L}_i(w^t) + (bb^T + \tau \mathbb{I})(w - w^t))] = 0 \tag{7}$$

$$N(bb^T + \tau \mathbf{I})w = N(bb^T + \tau \mathbb{I})w^t - \sum_{i=1}^{N} \nabla \mathcal{L}_i(w^t) \tag{8}$$

$$\hat{w}(w^t) = w^t - (bb^T + \tau \mathbb{I})^{-1} (\frac{1}{N} \sum_{i=1}^{N} \nabla \mathcal{L}_i(w^t)) \tag{9}$$

From this equation is actually very easy to retrieve also mini-batch or online update equations.

Now the interesting twist: we want to learn the best possible parametrization in order to achieve subquadratic performance in optimization.

Similarly to what is done in [1], the idea is to firstly deploy a learn to optimize (L2O) version of this optimization routine. Two main issues arise:

- How do we design the unrolling for the optimization routine to result in a neural optimizer?
- How do we train this neural optimizer in order to achieve the most efficient optimization routine?

These two points are actually crucial, since they will be the foundations for all our developments. The idea is to deploy a generalized L20 through a deep unrolling approach, resulting in a deep recurrent neural network for our optimizer: this is going to be framed in a reinforce approach, where the whole optimization procedure is recasted as Markov Decision Process. SCA well studied properties are going to be robust to any kind of convex regularization, as well as to distributed or federated implementation.

To begin with, we'll consider a single unrolled layer whose induced truncation bias is solved injecting momentum in the equation. The training of the parametrization of the expansion is doing on the fly, using the backpropagation of the loss up to the variables of the optimizer, using a meta-optimizer like ADAM or ADAGrad to step-wise optimize them.

Finally, let's have a look to our Successive Convex Approximation for neuraL network TRAining (SCALTRA) algorithm in its very first formulation as a batch optimization routine:

[1]Learning to Optimize Quasi-Newton Methods, Isaac C. Liao, Rumen R. Dangovski, Jakob N. Foerster, Marin Soljačić soljacic, 2023

Algorithm 1 SCALTRA (Batch routine)

```
Require: $f(w, x)$: Parametrization of a neural network to be trained
Require: $\mathcal{D}\{(x_i, y_i\}_{i=1}^{N}$: Training set
Require: $\mathcal{L}$: Loss function
Require: $w^0$: Weight initialization
Require: $m^0$: Momentum initialization
Require: $\{\theta^0 = \{b^0, \tau^0\}\}_{w \in \mathcal{W}}$: Optimizer parameters initialization
Require: $T$: Fixed number of epochs
Require: $\{\gamma_t\}_{t=1}^{T}$: Decreasing sequence of smoothing parameters
Require: $0 \leq \beta < 1$: Momentum parameter
Require: $\alpha$: Meta-learning rate
Require: $G$: Meta-optimizer
Require: $\alpha_0$: Initial learning rate

  for $t =, ..., T$ do
      for $w \in \mathcal{W}$ do
          $\hat{w}(w^t) = w^t - (bb^T + \tau\mathbb{I})^{-1} m^t$
          $w^{t+1} = w^t + \gamma^t[\hat{w}(w^t) - w^t]$
      end for
      $l^{t+1} = \sum_{i=1}^{N} \mathcal{L}(y_i, f(w^{t+1}, x_i)$
      $\theta^{t+1} = \theta^t + G(\nabla_{\theta^t} l^{t+1})$
      for $w \in \mathcal{W}$ do
          $m^{t+1} = \beta m^t + (1 - \beta)\nabla_{w^{t+1}} l^{t+1}$
      end for
  end for
  return $w^t$
```