sLBFGS+PLS Implementation Notes

Chaitanya Rastogi

June 7, 2019

1 Notation, etc.

- f: the learning model we are trying to infer parameters for (objective function); $f(t) = \hat{\mathcal{L}}(x(t))$
- t: the current location; position vector is given by $x(t)$
- y_t, y'_t: noisy function and gradient values of f at t
- σ_f, $\sigma_{f'}$: estimates of noise in function value and gradient
- $\odot$: element-wise operation, i.e. $x^{\odot 2}$ indicates element-wise squaring of the vector x, while $a \odot b$ indicates element-wise multiplication of the two vectors

The sLBFGS used in testing the code here was copied from my original Java code into Matlab. Testing has shown this version is not as efficient as the original Java code, probably due to some minor 0-1 array indexing issue in `twoLoopRecursion` that makes it slightly unstable. However, I still believe the overall trend of the results holds.

2 Scaling Issues

A major component of the probabilistic line search (PLS) paper discussed the elimination of hyperparameters. Many of these parameters were used to set the intrinsic scale of the gaussian process (GP) surrogate function on which the line search optimization was taking place. Upon closer inspection, the implementation of some of these scaling parameters was not in line with the theoretical description in the paper and/or did not make sense in the context of sLBFGS. Correcting these scaling parameters was critical to the stability of the line search when used with sLBFGS and are discussed below.

2.1 Scale factor β

The scaling factor β in the code and pseudocode is used to eliminate the hyperparameter θ, which "scales the prior variance" according to the paper. By setting $\theta = 1$ and rescaling the objective with $|y_0'|$, we get $y(0) = 0$ and $y'(0) = -1$. In other words,

$$y_i = \frac{y_i - y_0}{|y_0'|} \tag{1}$$

$$y_i' = \frac{y_i'}{|y_0'|} \tag{2}$$

Clearly, $|y_0'|$ refers to the 'norm of the gradient of f at the start of the line search,' based off of the definitions used elsewhere in the paper. However, in the pseudocode for `probLineSearch` at line 15, the scaling factor β is defined as

$$\beta \leftarrow |d' \cdot \Sigma_{df_0}|,$$

where d is the search direction and Σ_{df_0} is the sample variances of the gradient. This is clearly wrong, and the code does not reflect this; rather, the Matlab code uses

```
beta = abs(search_direction'*df0);
```

where `df0` is the function gradient at the origin of the line search. While this statement is congruent with the definition in the text of the paper for SGD, it is *not* for BFGS methods. The reason lies in the definition of the search direction d for both methods:

$$\text{SGD: } d = -\nabla f$$

$$\text{BFGS: } d = -H^{-1}\nabla f$$

where H is the pseduo-hessian matrix computed in the BFGS updates. As such, beta in the two cases becomes

$$\text{SGD: } \beta = |(-\nabla f)' \cdot \nabla f| = |y_0'|$$

$$\text{BFGS: } \beta = |(-H^{-1}\nabla f)' \cdot \nabla f| \neq |y_0'|$$

In the case of BFGS updates, it is possible that the above definition of β can be ≈ 0, as the inverse hessian can rotate the gradient vector to be nearly orthogonal to the gradient. The correct code should be

```
beta = norm(df0);
```

Fixing this rescaling greatly enhances the stability of the GP: before this fix, running sLBFGS + PLS with a batchsize of 20 and epoch period of 50 steps resulted in β values approaching 10^{-9}.

2.2 Rescaling by α_0

α_0 is the initial step size in the line search, in non-dimensional units. In the theoretical discussion of the scaling factor β, we see that 1 and 2 only admit the norm of the function gradient. Similarly, we see that the paper discusses rescaling the noise estimates for σ_f and $\sigma_{f'}$ as follows:

$$\sigma_f = \frac{\sigma_f}{|y'(0)|} \tag{3}$$

$$\sigma_{f'} = \frac{\sigma_{f'}}{|y'(0)|} \tag{4}$$

Curiously, the pseudocode for `probLineSearch` at lines 15 and 16 instead show:

$$\sigma_f \leftarrow \sqrt{\Sigma_{f_0}}/(\alpha_0 \cdot \beta)$$

$$\sigma_{df} \leftarrow \sqrt{(d^{\odot 2})' \cdot \sigma_{df_0})}$$

In addition, the pseudocode for `evaluateObjective` at lines 6 and 7 show:

$$y \leftarrow (y - f_0)/(\alpha_0 \cdot \beta)$$

$$dy \leftarrow (dy' \cdot d)/\beta$$

The original Matlab code follows this convention as well. It is unclear why the initial step size is needed to rescale the function values and gradients, especially as it is not motivated in the text. In fact, given that function values y and noise σ_f are scaled by an additional $1/\alpha_0$ term, we can get improperly scaled function value estimates relative to gradient estimates. In the sLBFGS setting, where a fixed $\alpha_0 = .1$ was used, this translates to an order of magnitude variation. As such, the Matlab code was amended as follows:

```
sigmaf = sqrt(var_f0)/beta; (in probLineSearch)
   y = (y - f0)/beta; (in evaluate_function)
       y_tt = y*beta + f0; (in make_outs)
```

3 Noise Estimation

Another major theoretical component of the PLS paper is the usage of noisy estimates to do a somewhat deterministic optimization of an objective function. As such, determining the noise level of these estimates forms a critical component of the PLS. However, I believe there is a major approximation made in the paper that can significantly impact the performance of the line search, especially within the context of stochastic variance-reduced gradients (SVRG).

The original PLS implementation sets the function and gradient noise levels (σ_f and $\sigma_{f'}$) at the *start* of the line search, i.e. at $t = 0$, and maintains it throughout the search. Surprisingly, faster convergence was observed in the SGD setting when the code was modified to change σ_f and $\sigma_{f'}$ to the largest value encountered during the line search. This test highlighted the impact of the approximation in estimating the variance.

3.1 SEM in the Minibatch Setting

The problem in estimating the SEM of function and gradient values in a minibatch can be recast into a toy problem:

> Let us begin with a set of points A, which have mean μ_A, variance σ_A^2, and SEM $\sigma_A/\sqrt{|A|}$, where $|A|$ is the cardinality of the set A. Our goal is to estimate μ_A; however, we only have access to sets A' that are generated by sampling points (without replacement) from A such that $|A'| \leq |A|$. If we can only observe sets A', what is the SEM of the means $\mu_{A'}$ that we observe?

Conceptually, I think understanding of this toy problem is key to understanding the error in the process for estimating the function and gradient SEMs in the current approach. When a minibatch is used to *estimate* function and gradient values, we know the means have some spread (SEM) around the true function and gradient means as computed *exactly* on the entire dataset. As such, in the limit where my minibatch hits the full size of the dataset, there should be *no* variance in the function and gradient means. The current approach employed in the paper and the code uses a naïve algorithm to compute the variance, using Bessel's correction:

$$\sigma_f^2 = \frac{1}{|A'|}\frac{|A'|}{|A'|-1}\left[\frac{1}{|A'|}\sum_{i\in A'}\ell^2(A_i) - \left(\frac{1}{|A'|}\sum_{i\in A'}\ell(A_i)\right)^2\right] \tag{5}$$

$$\sigma_{f'}^2 = \frac{1}{|A'|}\frac{|A'|}{|A'|-1}d^{\odot 2}\cdot\left[\frac{1}{|A'|}\sum_{i\in A'}\nabla\ell(A_i)^{\odot 2} - \left(\frac{1}{|A'|}\sum_{i\in A'}\nabla\ell(A_i)\right)^{\odot 2}\right] \tag{6}$$

3.2 Potential Future Directions

4 PLS in the SVRG Setting

4.1 Scaling to SVRG Space

4.2 Impact of SVRG Updates

5 Determining a Descent Direction

5.1 Metrics for GP Precision and Recall