CS690 Optimization Paper Notes 3

Due: March 12, 2021, 11:59 pm

Curvature-Exploiting Acceleration of Elastic Net Computations
Vien V. Mai and Mikael Johansson

1 Elastic Net Regression

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{2n}||Ax-b||_2^2+\frac{\lambda_2}{2}||x||_2^2+\lambda_1||x||_1$$

- when $\lambda_2=0$, it is lasso regression.
- when $\lambda_1=0$, it is ridge regression.

In the real word, there are often a few dominant features, and some other features are strongly correlated with these dominant features.

2 Related work

2.1 First-order method

Proximal Gradient Decent. Stochastic Proximal Gradient Decent is often used in large-scale problem, and the time complexity is improved than using vanilla proximal method. In both cases, adding Nesterov momentum to the optimization process could improve the time complexity.

2.2 Second-order method

Such as Newtons method that need to compute second-order derivatives during each iteration. It is robust than first-order method, especially in non-linear and ill-conditioned cases. But this method require high computational cost per iteration. There are some work that use an approximation approach, where they estimating the Hessian matrix, but the time complexity is raised a lot.

3 Algorithm

Proximal gradient decent is the main type of method. In Elastic Net problem, the smooth part is the objective function plus the ridge loss

$$\frac{1}{2n}||Ax-b||_2^2+\frac{\lambda_2}{2}||x||_2^2$$

and the lasso loss as the undifferentiable part. In the approach of this paper, the optimizing process is a method called ProxSVRG but with a second order method updating rule, i.e. a Newton-like method that will normalize the gradients by the Hessian of the function before the proximal mapping step. The following is the pseudo-code for the second-order ProxSVRG, which is an proximal stochastic gradient descent method, that update the parameters with gradient at one data point. The global gradients are computed periodically, and will be used to adjust stochastic gradients. Denote the learning rate as α, and a momentum-like coefficient τ.

```
initialize($x_0$);
for $s$ = 0, 1, ..., $S$ do
    $\nabla f(x_s) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_s)$;
    $x_0 = x_s$;
    $z_0 = x_0$;
    for $k$=0, 1, ..., $T$ do
        randomly select $j \in \{0, 1, ..., n\}$;
        $y_k = \frac{1}{1+\tau} x_k + \frac{\tau}{1+\tau} z_k$;
        $v_k = \nabla f_j(y_k) - \nabla f_j(x_s) + \nabla f(x_s)$;
        $x_{k+1} = \text{prox}_{\alpha h}^{H}(y_k - \alpha H^{-1} v_k)$;
        $g_{k+1} = \frac{1}{\alpha}(y_k - x_{k+1})$;
        $z_{k+1} = z_k + \tau(y_k - z_k) - \frac{\tau}{\mu} g_{k+1}$;
    end
    $x_{s+1} = x_T$
end
return $x_S$
```

In order to reduce the noise and effect of damping, a mini-batch with size $O(\sqrt{\kappa})$ is often used for computing gradient at each iteration (With Nesterov momentum, such size is under toleration), where κ is the condition number of the problem.

3.1 Approximation of Hessian

In the above algorithm, computing the true Hessian have time complexity $O(nd^2)$. The author proposed a method that only require $O(rd)$ time complexity, where r is the rank of the low-rank approximation of the matrix A in the original objective function. By SVD, we have the best rank r approximation of matrix $A_r = U_r \Sigma_r V_r^T$. Then the Hessian is derived as

$$H = V_r(\Sigma_r^2 + \lambda_2 I)V_r^T + (\sigma_r^2 + \lambda_2)(I - V_r V_r^T)$$

where the authors state that the second term is used to capture information in the subspace orthogonal to the column space of V_r. (σ is the singular value) Then the inverse of Hessian have the closed form

$$H^{-1} = V_r(\Sigma_r^2 + \lambda_2 I)^{-1}V_r^T + \frac{1}{\sigma_r^2 + \lambda_2}(I - V_r V_r^T)$$

3.2 Condition Number and Time Complexity

The condition number is given as

$$Pr(\kappa_H \leq \min(\sum_{i=1}^{d} \frac{\sigma_i}{\lambda_2(\sigma_i + \lambda_2)}, \frac{r\sigma_r + \sum_{i>r}\sigma_i}{\lambda_2} + d)) \geq \frac{9}{10}$$

Here σ_i are the singular value of the correlation matrix (AA^T), and d is the rank of correlation matrix.

The time complexity is given as

$$O(d(n + \kappa_H)\log\frac{1}{\epsilon})$$

where the d here is the feature dimension, and ϵ is the tolerant error to the optima.