Advanced topics

Optimization of linear methods (developer)

Limited-memory BFGS (L-BFGS)

L-BFGS is an optimization algorithm in the family of quasi-Newton methods to solve the optimization problems of the form $\min_{\mathbf{w} \in \mathbb{R}^d} \; f(\mathbf{w})$. The L-BFGS method approximates the objective function locally as a quadratic without evaluating the second partial derivatives of the objective function to construct the Hessian matrix. The Hessian matrix is approximated by previous gradient evaluations, so there is no vertical scalability issue (the number of training features) unlike computing the Hessian matrix explicitly in Newton's method. As a result, L-BFGS often achieves faster convergence compared with other first-order optimizations.

Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) is an extension of L-BFGS that can effectively handle L1 and elastic net regularization.

L-BFGS is used as a solver for LinearRegression, LogisticRegression, AFTSurvivalRegression and MultilayerPerceptronClassifier.

MLlib L-BFGS solver calls the corresponding implementation in breeze.

Normal equation solver for weighted least squares

MLlib implements normal equation solver for weighted least squares by WeightedLeastSquares.

Given n weighted observations (w_i, a_i, b_i):

- w_i the weight of i-th observation
- a_i the features vector of i-th observation
- b_i the label of i-th observation

The number of features for each observation is m. We use the following weighted least squares formulation:

$$\min_{\mathbf{x}} \frac{1}{2} \sum_{i=1}^{n} \frac{w_i(\mathbf{a}_i^T \mathbf{x} - b_i)^2}{\sum_{k=1}^{n} w_k} + \frac{\lambda}{\delta} \left[\frac{1}{2}(1-\alpha) \sum_{j=1}^{m} (\sigma_j x_j)^2 + \alpha \sum_{j=1}^{m} |\sigma_j x_j| \right]$$

where λ is the regularization parameter, α is the elastic-net mixing parameter, δ is the population standard deviation of the label and σ_j is the population standard deviation of the j-th feature column.

This objective function requires only one pass over the data to collect the statistics necessary to solve it. For an $n \times m$ data matrix, these statistics require only $O(m^2)$ storage and so can be stored on a single machine when m (the number

of features) is relatively small. We can then solve the normal equations on a single machine using local methods like direct Cholesky factorization or iterative optimization programs.

Spark MLlib currently supports two types of solvers for the normal equations: Cholesky factorization and Quasi-Newton methods (L-BFGS/OWL-QN). Cholesky factorization depends on a positive definite covariance matrix (i.e. columns of the data matrix must be linearly independent) and will fail if this condition is violated. Quasi-Newton methods are still capable of providing a reasonable solution even when the covariance matrix is not positive definite, so the normal equation solver can also fall back to Quasi-Newton methods in this case. This fallback is currently always enabled for the `LinearRegression` and `GeneralizedLinearRegression` estimators.

`WeightedLeastSquares` supports L1, L2, and elastic-net regularization and provides options to enable or disable regularization and standardization. In the case where no L1 regularization is applied (i.e. $\alpha = 0$), there exists an analytical solution and either Cholesky or Quasi-Newton solver may be used. When $\alpha > 0$ no analytical solution exists and we instead use the Quasi-Newton solver to find the coefficients iteratively.

In order to make the normal equation approach efficient, `WeightedLeastSquares` requires that the number of features be no more than 4096. For larger problems, use L-BFGS instead.

Iteratively reweighted least squares (IRLS)

MLlib implements iteratively reweighted least squares (IRLS) by IterativelyReweightedLeastSquares. It can be used to find the maximum likelihood estimates of a generalized linear model (GLM), find M-estimator in robust regression and other optimization problems. Refer to Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and some Robust and Resistant Alternatives for more information.

It solves certain optimization problems iteratively through the following procedure:

- linearize the objective at current solution and update corresponding weight.
- solve a weighted least squares (WLS) problem by WeightedLeastSquares.
- repeat above steps until convergence.

Since it involves solving a weighted least squares (WLS) problem by `WeightedLeastSquares` in each iteration, it also requires the number of features to be no more than 4096. Currently IRLS is used as the default solver of GeneralizedLinearRegression.