|  |  |  |
| --- | --- | --- |
| **Module/framework/package** | **Name and brief description of algorithm** | **An example of a situation where using the provided GLM implementation provides superior performance compared to that of base R or its equivalent in Python (identify the equivalent in Python)** |
| Base R | Iteratively Reweighted Least Squares (IWLS). The Fisher scoring method applies repeated weighting of response data until parameter convergence is reached. | Baseline implementation. The algorithm works efficiently with datasets of small to medium size that fit within system memory while maintaining strong stability during typical Generalized Linear Model applications. |
| Big Data version of R | The system offers multiple processing options alongside its implementations. GLMs can execute in parallel through Rmpi and snowfall packages while the bigmemory and biglm packages support out-of-memory processing of large datasets. | When used in combination with biglm packages users can conduct incremental calculations on datasets too large for memory storage. The rxode2 package enables ODE solving across multiple computing units. SparkR functions along with distributed computing frameworks enable GLMs to distribute computations across multiple clusters. |
| Dask ML | The distributed computing platform with iterative algorithms allows spark.glm to distribute its computations across multiple cluster nodes. Supports various family distributions (Gaussian, binomial, Poisson, Gamma, Tweedie). | The software works optimally with immense datasets which are spread across multiple clusters. The system performs large-scale data handling that base R cannot support because of insufficient memory capacity. The system delivers optimal performance when working with extensive datasets containing numerous rows that can be divided between nodes. |
| Spark R | Gradient Descent joins forces with Stochastic Gradient Descent (SGD) and L-BFGS and ADMM (Alternating Direction Method of Multipliers) as the implemented algorithms. Implements both first-order methods (SGD variants) and quasi-Newton methods (L-BFGS). | The L-BFGS method achieves faster convergence of GLMs compared to SGD when using fewer iteration cycles. Through distributed implementation the program can handle datasets which exceed the capacity of scikit-learn. Particularly effective for high-dimensional sparse datasets common in text analysis and recommendation systems. |
| Spark optimization | The available solver options for this model include 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga'. Each solver from the available options has distinct areas of expertise between L1, L2 and ElasticNet penalties and problem types. | The 'sag' and 'saga' solvers process large datasets more quickly while the 'liblinear' solver works fastest on high-dimensional sparse data when L1 penalties are applied. The vectorized operations of Scikit-learn optimize medium-sized datasets by utilizing modern CPU structures efficiently. |
| Scikit-Learn | The implementation provides users with five algorithm choices which comprise ADMM and gradient descent and L-BFGS and Newton's method and proximal gradient method. Supports various regularizers (L1, L2, ElasticNet) | Offers superior performance for datasets larger than memory on a single machine. The system adapts to grow from one machine operation to distributed cluster operations without issues. The system supports scikit-learn API functionality by processing distributed dask arrays and dataframes. This system demonstrates particular success when used for processing data in small continuous increments. |