Big Data Analytics - Systems and Methods

Portfolioprüfung Part 2: Implementierung

Prof. Dr. Benjamin Buchwitz

—

Gegenstand der Implementierungsaufgabe ist das Verfassen einer wissenschaftlichen Arbeit in Form einer Hausarbeit, die mindestens aus den folgenden Kernbestandteilen besteht: 1) Theoretische Grundlagen der implementierten Methode(n), 2) Architektur des konfigurierten Systems, 3) programmatischer Ansatz der Implementierung mit Erläuterung der wesentlichen Codebestandteile, 4) Benchmarking der Implementierung entlang der vorgegebenen Dimension.

Der Hauptteil der Arbeit diskutiert dabei ausgewählte Teile des Programmcodes im Kontext der zugehörigen Anwendung und die erziehlten Ergebnisse. Die Arbeit schließt mit einer ausführlichen Darstellung der erarbeiteten Benchmarkingergebnisse und einer kritischen Würdigung des gewählten vorgehens. Der *vollständige* und *lauffähige* Code für das Projekt wird als technischer Anhang zur Ausarbeitung eingereicht. Die verwendete Programmiersprache kann R oder Pyhton oder eine Mischung aus beiden Sprachen sein. Sollten Sie eine andere Sprache verwenden wollen halten Sie bitte kurz Rücksprache. Da es sich bei der Arbeit um eine wissenschaftliche Arbeit handelt, sind insbesondere die Grundsätze des wissenschaftlichen Arbeitens (präzise Formulierung, hohe Inhaltsdichte, Zitation, Formatierung, Qualität von Grafiken, etc.) zu beachten. Es wird ausdrücklich empfohlen die Arbeit **in Englisch** zu verfassen.

Die Kooperation als Gruppe pro Thema ist ausdrücklich erlaubt. Jede Gruppe reicht dabei nur **eine** Ausarbeitung ein und ergänzt die Arbeit um einem Anhang mit einer Übersicht der beigetragenen Eigenleistungen der Autoren.

Regression in Apache Spark

Apache Spark stellt per MLlib unter anderem Routinen zur Schätzung von linearen Regressionsmodellen zur Verfügung. Während Apache Spark auch komplexe lineare Modelle (z.B. Generalized Linear Models) unterstützt, beschränkt sich der Methodenteil der Aufgabe auf das klassische lineare Regressionsmodell (OLS) mit dem Mean Square Error (MSE) als Loss-Funktion. Entsprechend werden nur Modelle ohne Regularisierung oder weiterführende methodische Ausgestaltungen betrachtet, sodass komplexere Schätzmethoden (ála Stochastic Gradient Descent) von der Betrachtung zunächst ausgeschlossen werden können. Die relevanten in Apache Spark verfügbaren Methoden, einige Erläuterungen und der Hinweis auf die Optimierung per Normalengleichungen (Normal Equations) via WeightedLeastSquares finden sind in der Dokumentation an folgenden Stellen:

- https://spark.apache.org/docs/latest/mllib-linear-methods.html#regression
- https://spark.apache.org/docs/latest/ml-advanced.html#optimization-of-linear-methods-developer
- https://spark.apache.org/docs/latest/api/scala/org/apache/spark/mllib/regression/index.html

Ordinary Least Squares Estimation (OLS)

Wachsende Datenmengen führen dazu, dass sich Regressionsmodelle nicht auf Single-Node Systemen lokal schätzen lassen, sondern auf Systemen mit mehreren Nodes verteilt werden müssen. Das multiple Regressionsproblem

$$y = X\beta + \epsilon$$

mit den Bestandteilen

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} x_{10} & x_{11} & \dots & x_{1p} \\ x_{20} & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n0} & x_{n1} & \dots & x_{np} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

zeigt deutlich, dass die Komplexität des Problems in den Dimensionsn n (Anzahl der Beobachtungen) und p (Anzahl der Regressoren) wachsen kann. Elementaren Operationen zur Handhabung großer Datenmengen und zur Anwendung mathematischer Operatoren stellt Apache Spark zur Verfügung. Aus Gründen numerischer Stabilität und ggf. algorithmischer Präzision werden Regressionsschätzer in der implementierungstechnischen Praxis jedoch nicht direkt über die in einführenden Lehrbüchern hergeleitete analytische Lösung $\hat{\beta} = (X^T X)^{-1} X^T y$ berechnet, sondern über Dekompositionsverfahren. Die verwendeten Verfahren sind insbesondere die LU Dekomposition, die QR Dekomposition und die Berechnung der OLS-Schätzer über die Moore-Penrose-Pseudoinverse und damit mittelbar über die Singularwertzerlegung (SVD).

Eine Einführung in das Thema liefern Hansen et al. und Higham. Zur Effizienz der Algorithmen führen Golub und Van Loan Argumente an. Druinsky und Toledo äußern sich zur Präzision bei der Lösung von Gleichungssystemen mittels Matrizeninversion.

- Hansen, Per Christian; Pereyra, Victor; Scherer, Godela (2013): Least Squares Data Fitting with Applications, Johns Hopkins University Press.
- Higham, Nicholas J. (2002): Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM.
- Golub, Gene H.; Van Loan, Charles F. (2013): Matrix Computations, Fourth Edition, Johns Hopkins University Press.
- Druinsky, Alex; Toledo, Sivan (2012): How Accurate is inv(A)*b?, arXiv Preprint, https://arxiv.org/abs/1201.6035.

Implementierungsaufgabe: Distributed OLS

Ziel dieser Implementierungsaufgabe ist die Beurteilung der Performance von konkurrierenden Verfahrensvarianten der mulitplen OLS-Regression in einer verteilten Umgebung. Die Performance wird dabei für eine feststehende Clusterkonfiguration und einem feststehenden Datensatz mittels der Rechenzeit gemessen. Mit zunehmender Komplexität der Probleminstanz (Hinzunahme von Beobachtungen n bzw. Regressoren p) wird die Entwicklung der Rechenzeit untersucht. Entsprechend der Komplexitätsdimensionen n und p ergeben sich in Abhängigkeit der Dekompositionsverfahren QR-Dekomposition und Singularwertzerlegung (SVD) insgesamt vier mögliche Themenkonstellationen:

a) QR-Dekomposition und wachsendes n
b) QR-Dekomposition und wachsendes p
c) SVD und wachsendes n
d) SVD und wachsendes p

Da es sich bei der Implementierung um eine verteilte Verfahrensvariante handelt, die für die Verarbeitung großer Datenmengen konzipiert ist, sind die Grenzen der Parameter n und p entsprechend groß und damit realistisch zu dimensionieren (z.B. $n \gg 10^9$ und $p \gg 10^5$). Da die Handhabung solcher Datenstäze ggf. mühsam ist, reicht es aus einen Datensatz zu simulieren. Dieser muss für die Regression geeignet sein, d.h. die Zusammenhangsstruktur in den Daten muss einem definierten, benanntem und erläutertem Regressionsmodell folgen. Zudem ist darauf zu achten, dass sich die β-Koeffizienten auch hinsichtlich der Größenordnungen und hinsichtlich ihrer Varianzen unterscheiden und entsprechend Streuung im Datensatz vorhanden ist (z.B. $R^2 \approx 0.7$) sowie die Gauss-Markov Annahmen der Regression gelten.

Als wesentliche Schritte bei der Bearbeitung der Aufgabe, ergeben sich die folgenden Punkte, die ebenfalls in Hauptgliederungskapiteln der Abgabe repräsentiert werden sollten:

1) Zugewiesene Verfahrensvariante erläutern (QR bzw. SVD)
2) Clusterkonfiguration und Aufsetzen via Docker
3) Verteilte Implementierung der zugewiesenen Verfahrensvariante in Apache Spark
4) Simulation eines geeigneten Datensatzes für die Regressionsschätzung
5) Bechmarking der Implementierung bei wachsendem n bzw. p

Hinweise

- Bitte stellen Sie Ihre Ergebnisse in jedem Fall grafisch dar.
- Simulieren Sie zunächst den ganzen Datensatz, also für maximales n bzw p und betrachten Sie die Performance anhand von zufällig gezogenen Subsamples dieses Datensatzes.
- Um aussagekräftige Ergebnisse zu erhalten, führen Sie die Analyse für ein gewähltes n bzw p mehrfach durch. Nur so können Sie über die verschiedenen Größen der Probleminstanz (Werte für n bzw. p) bestimmen ob nur die Rechenzeit oder auch deren Streuung ansteigt.
- Wählen Sie mindestens 10 Werte für n bzw p.
- Ihre Individuelle Implementierung der Regressionsfunktion muss einen Output haben, der mit dem Konsolenoutput von `summary(lm(...))` oder `statsmodels` (Python) identisch ist. Sie können diesen Output um die Rechenzeit zur Erzeugung der Ergebnisse ergänzen.