NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Health data analytics using scalable logistic regression with stochastic gradient descent

Research Studied By:

- Shozab Mehdi (22k-4522)
- Taha Sharif (22k-4145)
- Talha Durrani (22k-4144)

Under The Supervision Of **Sir Usama Antuley**

Hello!

Warm greetings to all who are present. As we gather here today, we are excited to introduce our study on the research proposal of the **Stochastic Gradient Descent Method** of Numerical Optimization, keeping the main focus on **health data analytics using scalable logistic regression**

Agenda Overview

01 Code

02 Abstract of the study

03 Intro/ Problem Statement

04 Related Work

05 Proposed Framework

- Wearable sensor data collection unit
- Big data analytics and cloud computing
- Big data analytics using Apache Mahout

06 Results and discussions

07 Conclusions

Code

Part 1	Part 2	Part 3

x = np.random.rand(10, 1)
y = 2 * x + np.random.randn(10, 1)
theta = np.zeros(2)
L_rate = 0.01
iters = 1007
	#Definiton of hθ(x) (Hypothesis Function)
	def hypothesis_func(theta, x):
	return theta[0] + theta[1] * x
		#Definition of J(θ0,θ1) (Cost Function)
		def cost_func(theta, x, y):
		m = len(x)
		sum = 0
		for i in range (m):
		sum += (hypothesis_func(theta, x[i]) - y[i])**2
		
		return (1/(2*m))*(sum)

Code: Part 6 & 7 + Gradient Descent

Part 6	Part 7	Part 3

#(a)A plot of J(θ0,θ1) with initial point
plt.plot(range(1, iters + 1), costs, marker='o', linestyle='-', color='g', label='Cost')
plt.scatter(0, costs[0], color='red', label='Initial Cost', zorder=5)
plt.xlabel('N')
plt.ylabel('Cost')
plt.title('Cost Function during Optimization')
plt.legend()
plt.grid(True)
plt.show()
 	
#(b)Table of Gradient Descent over Iterations
print(tb(data, headers=['n', 'J', 'THETA0', 'THETA1'], tablefmt="github"))
	#(c)Scatter Plot of Linear Regression
	plt.scatter(x, y, color='blue', label='Data')
	plt.plot(x, hypothesis_func(theta, x), label=f'Regression Line: y = {theta[0]:.2f} + {theta[1]:.2f}x', color='red')
	plt.xlabel('X Data')
	plt.ylabel('Y Data')
	plt.title('Linear Regression on Data')
	plt.legend()
	plt.grid(True)
	plt.show()
	return
		#Definition of Gradient Descent to find the minimum possible value of J
		def grad_descent(theta, x, y, iters, L_rate):
		data = []
		costs = []
		m = len(x)
		for i in range(iters):
		dtheta0 = 0
		dtheta1 = 0
		for j in range(m):
		dtheta0 += (hypothesis_func(theta, x[j]) - y[j])
		dtheta1 += (hypothesis_func(theta, x[j]) - y[j]) * x[j, 0]
		
		theta[0] -= L_rate * dtheta0
		theta[1] -= L_rate * dtheta1
		costs.append(cost_func(theta,x,y))
		data.append([i+1, cost_func(theta, x, y), theta[0], theta[1]])

Abstract of the Study

The paper focuses on creating a **scalable architecture** for processing wearable medical sensor data in cloud computing for healthcare applications. It utilizes big data technologies like **Apache Flume, Apache Pig, and Apache HBase on Amazon Web Services** to handle large amounts of sensor data.

The implementation includes **Apache Mahout** for MapReduce-based online stochastic gradient descent in logistic regression, trained using the Cleveland Heart Disease Database. Wearable sensors capture patient data like blood pressure, blood sugar, and heart rate to predict heart disease status. The proposed model achieves an **accuracy of 81.99% in training and 81.52% in validation samples** for heart disease classification.

Introduction/ Problem Statement

Problem Statement

Sensors are critical in recent computing, specifically in healthcare to capture environmental data. Wearable medical devices collect extensive health data. **The volume and complexity of sensor data pose challenges for traditional database systems.**

Possible Solution Overview

Advanced tools are needed to store, process, and extract insights from large sensor data sets. Wearable sensors aid in the continuous monitoring of fitness, enabling better diagnosis and treatment recommendations. **Scalable big data architectures in cloud computing** are essential for handling the growing volume and speed of sensor data.

Related Work

Nowadays many health monitoring frameworks follow a three-tier architecture that includes the following layers:

1. Data Communication and Networking:

- Open geo-spatial consortium (OGC)-based systems and wearable sensors are used to measure and capture physiological parameters such as blood pressure, body temperature, heart rate, and respiration rate.
- Communication between devices in the system is facilitated through technologies like Bluetooth, Zigbee wireless networks, UMTS, WiMax, and the internet.

2. Acquisition Unit:

- The acquisition unit is responsible for collecting biomedical signals using fabric sensing elements, including electrocardiogram, respiration, and activity.
- These sensors are fixed on the patient's body and integrated with sensor servers to continuously send data.

3. Service Layer:

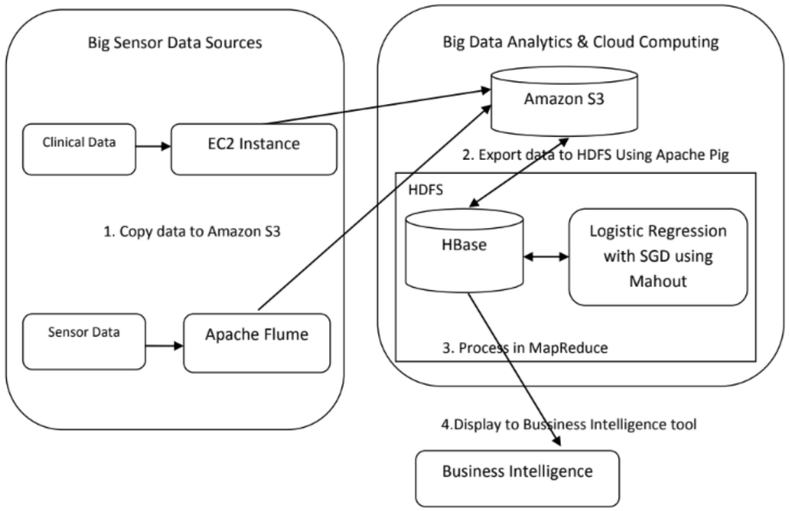
- Technologies like Apache Flume, Amazon S3, Apache HBase, Apache Pig, and Apache Mahout are used in the service layer for data communication, storage, transfer, and processing.
- Machine learning algorithms such as stochastic gradient descent (SGD) are employed to develop prediction models using both sensor data and electronic health records (EHR).

Proposed Framework

Objective: Develop a scalable framework for storing and processing body sensor data for healthcare applications.
Challenges: Increasing the size and structure of healthcare data necessitate solutions beyond traditional RDBMS.
Solution: Utilize NoSQL databases for their advantages in auto-scaling, performance, and high availability in distributed healthcare systems.
Proposed Architecture:

- Electronic medical records collected through clinical tests are stored in cloud storage (Amazon S3).
- MapReduce implementation of online SGD algorithm is employed in logistic regression for prediction model development.
- Logistic regression model trained on prior electronic medical records predicts heart disease status using current sensor data (blood pressure, blood sugar level, heart rate).

Scalable sensor data processing architecture in the cloud



Wearable Sensor Data Collection with Apache Flume

Challenge: Moving data efficiently within the Hadoop framework, crucial in cloud computing environments where data generation is rapid.

Solution: Utilize Apache Flume for reliable and distributed data collection, aggregation, and movement.

Key Features of Apache Flume:

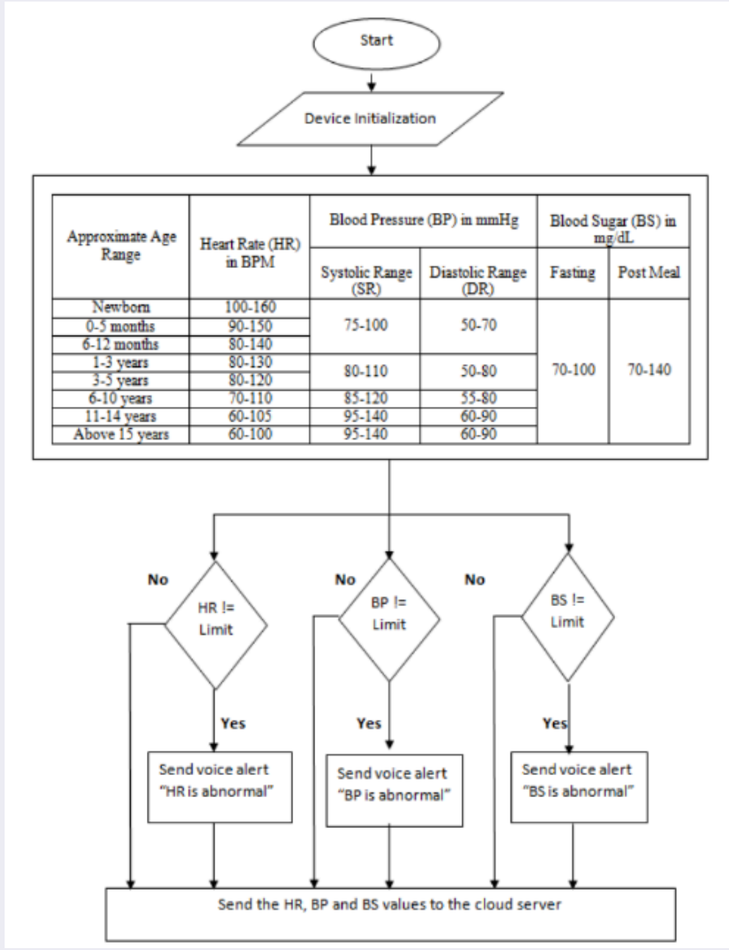
- Reliable and fault-tolerant for handling large volumes of data.
- Utilizes streaming data flows for real-time processing.
- Consists of agents, collectors, and a master component.

Workflow:

- Agents collect data from various sources like API logs, web logs, etc.
- Collectors store data into HDFS or Amazon S3.
- Simple and hierarchical model allows scalability with fewer collectors needed.

Implementation:

- Apache Flume deployed to transfer body sensor data to Amazon S3.
- Sensor devices send alerts with clinical values to the sensor server when vital signs exceed threshold values.
- Flume installed on the sensor server facilitates seamless transfer of body sensor measurements to Amazon S3.



Approximate Age Range	Heart Rate (HR) in BPM	Blood Pressure (BP) in mmHg		Blood Sugar (BS) in mg/dL	
		Systolic Range (SR)	Diastolic Range (DR)	Fasting	Post Meal
Newborn	100-160	75-100	50-70	70-100	70-140
0-5 months	90-150				
6-12 months	80-140				
1-3 years	80-130	80-110	50-80		
3-5 years	80-120				
6-10 years	70-110	85-120	55-80		
11-14 years	60-105	95-140	60-90		
Above 15 years	60-100	95-140	60-90		

Big data analytics and cloud computing block

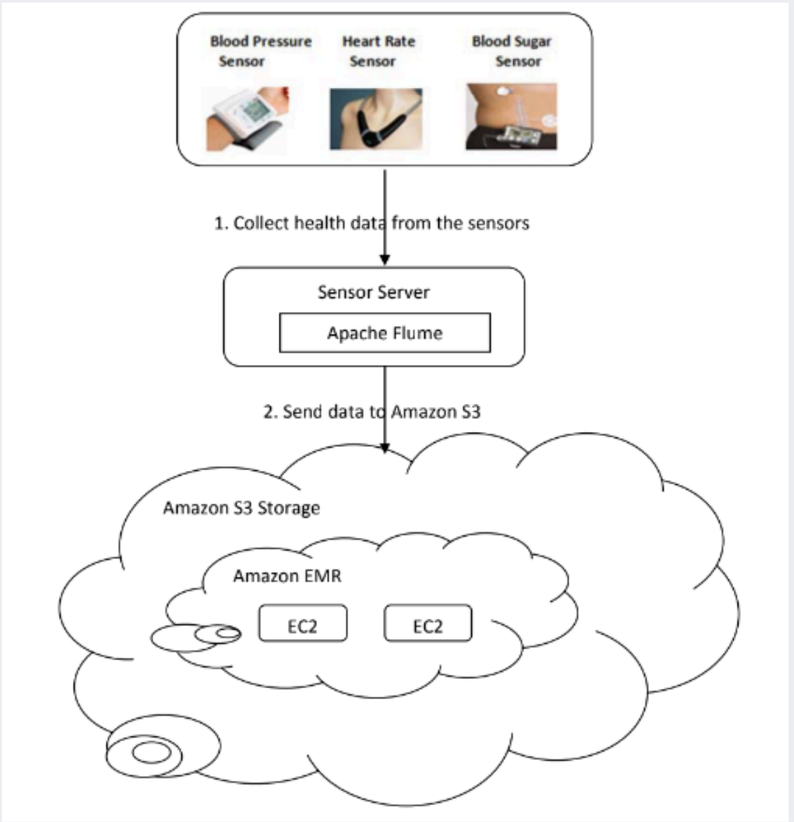
Amazon Elastic MapReduce (EMR):
Overview:
AWS service for processing vast data using Hadoop framework.
Features:

- Utilizes Hadoop MapReduce for distributed data processing.
- Automatically handles node failure, task monitoring, load balancing, and scheduling.

Moving Clinical Data to Amazon S3:
Overview:
Enable access to clinical data stored on EC2 instances within Amazon EMR.

Data Transfer from Amazon S3 to HBase
Overview:
Transfer data from Amazon S3 to HBase cluster using Apache Pig, the platform for analysing large datasets using PigLatin scripting language.
Features:

- Load clinical data from Amazon S3.
- Join incremental data with previous clinical data.



Big Data Analytics with Apache Mahout

Logistic Regression

Overview:
Statistical method for analyzing datasets with dichotomous dependent variables.

Process:
Relationship between dependent binary variable and independent variables.
Weighted sum passed through sigmoid function to yield result between 0 and 1.

Objective:
Find best weights/regression coefficients using optimization techniques.

Equations:
Define probability of dependent variable becoming 1 or -1 based on weights.

Stochastic Gradient Descent

Optimisation Functions: Used to update regression weights iteratively.

Comparison:

- Traditional gradient descent vs. SGD.
- SGD more suitable for big data analytics due to incremental updates.

Update Process: Parameters updated using gradient computed from single or few training examples.

Mahout Implementation of SGD for Logistic Regression

Overview: Apache Mahout - library of scalable machine-learning algorithms.

Features:
Implemented on Apache Hadoop using MapReduce paradigm.
Supports execution of machine learning algorithms to extract meaningful patterns from big data.

-**Commands:**

- Training the model.
- Testing the model.

Dataset:

- Utilizes Cleveland Heart Disease Database (CHDD) for training prediction model.
- Contains subset of 14 variables widely used in heart disease research.

Result & Discussions

Objective: Develop prediction model for heart disease status using logistic regression with SGD.

Training Data: Prior clinical records and sensor data of patients.

Prediction Model: Utilizes current sensor data (blood pressure, blood sugar level, heart rate).

Performance Analysis (Table 4):

Training Accuracy: 72.51%

Validation Accuracy: 72.82%

Performance Evaluation Metrics:

Accuracy: Correctly classified instances divided by total instances.

Sensitivity: Proportion of actual positives correctly identified.

Specificity: Proportion of actual negatives correctly identified.

Precision: Proportion of predicted positives that are actually positive.

F-Measure: Harmonic mean of precision and sensitivity.

Table 4 Performance analysis of the prediction model

Accuracy	*Sensitivity*	*Specificity*	*Precision*	*F-measure*
		Training sample		
81.99%	86.84%	76.28%	81.14%	83.89%
		Validation sample		
81.52%	80%	83.33%	85.1%	82.47%

Conclusions

Objective: Develop architecture for storing and processing body sensor data in healthcare applications.

Components Used:

- Apache Flume, Apache Pig, and Apache HBase for data collection and storage in AWS.
- Online SGD algorithm with logistic regression implemented using Apache Mahout for scalable diagnosis model.

Evaluation Results:

Dataset Used: Cleveland Heart Disease Database (CHDD) and wearable body sensors.

Accuracy of Prediction Model:

- Training Sample: 81.99%
- Testing Sample: 81.52%

Thank You

27th April, 2024