A declarative language for machine learning

Abstract

Machine learning has matured to the point where much of it can be codified. Currently, procedural languages like Python are the primary tool for machine learning. But they are used as a tool to call libraries in the correct sequence. A cleaner solution is presented using a declarative language modeled after the SQL language. To make this language operational, a model of components is also defined.

Introduction

Example

Here are several examples to show how this can be used.

A simple example

Assume all the servers and components are defined. Data for housing sales has been loaded. Create a model to predict housing sales prices, review the measures, infer the prices for a list of houses.

Create Model Housing using ...

Train Model Housing using data ...

Test Model Housing using data ...

Infer from data ... using model housing

A moderate example

A complex example

Functionality

Architecture

Components

The components are defined in the following table. All components can be installed on one computer or each can be on a separate computer. There must be at least one instance of each component. Meta Data is only updated in the meta data store. But the meta data is replicated to every other component as read only. An update to meta data is only complete when all components a=have accepted the update.

Component	Use	Max Instances	Notes
API	Allows access through APIs	N	
Data Get	This can get data using FTP, API, etc.	N	
Data Store	A location for data to be dropped off	N	
Meta Data Store	Information about all the other components. For example, has the data that the Data Get components needs to get.	N	
Model Store	Stores the parameters for each model as well as versions of a model.	N	
Processor	Can perform data movement, training, or inference. There must be at least three processors, one for each.	N	
User Interface	Uses the API to access the components for management, operations, and monitoring.	N	

Data Store

The data store is a data lake with a controlled architecture. On the left any data can be provided using any mechanism available. One mechanism is Data Get. Once data arrives, it is reviewed and cataloged and placed in the proper location in the middle. If there are rules defined, any data in the middle can be processed and moved to the righthand side. Data in the right hand side is available for training or inference.

Levels

The meta data supports the definition of levels of components and a promotion mechanism between them.

Language Reference

This section is an alphabetical list of language elements and their semantics.

Alter

Alter Data

Alter Model

Delete

Delete Data

Delete Model

Create

Create Component

Used to create components on specific servers or server groups.

Create Data

Create Data Get

Creates an outline of a way to get data. It can be one time or repetitive.

Create Level

Creates levels that can be used to promote components through.

Create Model

Create Server

Create Server Group

Infer

Uses a model and data to make an inference. The data can be provided on the inference statement or come from the data store component.

Infer from {data} using {model}

Infers from the data provided using the model provided. The model can specify a version.

Promote

Used to promote components from one level to the next.

Promote Component

Implementation

The initial implementation could be done by converting this language to Python and executing Python. If successful and useful, more direct implementation can be done.

Language Syntax

Syntax diagrams of the language.