Quick Start Guide for Users

This page explains how to quickly get started with MADlib® using a sample problem.

1. Install MADlib
2. Sample problem using logistic regression
3. Next steps

One you have MADlib installed, you can use the available Jupyter notebooks for many MADlib algorithms.

Install MADlib

Please refer to the Installation Guide for MADlib on how to install from binaries, as well as step-by-step descriptions on how to compile from source.

Please note that a Greenplum database sandbox VM with MADlib pre-installed is also available to get started quickly, as an alternative to installing MADlib yourself.

Sample Problem Using Logistic Regression

1. The sample data set and an introduction to logistic regression are described here.

   The MADlib function used in this example is described in the MADlib logistic regression documentation.

   Suppose that we are working with doctors on a project related to heart failure. The dependent variable in the data set is whether the patient has had a second heart attack within 1 year (yes=1). We have two independent variables: one is whether the patient completed a treatment on anger control (yes=1), and the other is a score on a trait anxiety scale (higher score means more anxious).

   The idea is to train a model using labeled data, then use this model to predict second heart attack occurrence for other patients.

2. To interact with the data using MADlib, use the standard `psql` terminal provided by the database. You could also use a tool like `pgAdmin`.

```
DROP TABLE IF EXISTS patients, patients_logregr, patients_logregr_summary;

CREATE TABLE patients( id INTEGER NOT NULL,
                                            second_attack INTEGER,
                                            treatment INTEGER,
                                            trait_anxiety INTEGER);

INSERT INTO patients VALUES
(1,                1,                1,                70),
(3,                1,                1,                50),
(5,                1,                0,                40),
(7,                1,                0,                75),
(9,                1,                0,                70),
(11,        0,                1,                65),
(13,        0,                1,                45),
(15,        0,                1,                40),
(17,        0,                0,                55),
(19,        0,                0,                50),
(2,                1,                1,                80),
(4,                1,                0,                60),
(6,                1,                0,                65),
(8,                1,                0,                80),
(10,        1,                0,                60),
(12,        0,                1,                50),
(14,        0,                1,                35),
(16,        0,                1,                50),
(18,        0,                0,                45),
(20,        0,                0,                60);
```

3. Call MADlib built-in function to train a classification model using the training data table as input. Note that the `1` entry in the `ARRAY` denotes an additional bias term in the model in the standard way, to allow for a non-zero intercept value.

```
SELECT madlib.logregr_train(
    'patients',                                 -- source table
    'patients_logregr',                         -- output table
    'second_attack',                            -- labels
    'ARRAY[1, treatment, trait_anxiety]',       -- features
    NULL,                                       -- grouping columns
    20,                                         -- max number of iteration
    'irls'                                      -- optimizer
    );
```

4. View the model that has just been trained:

```
   -- Set extended display on for easier reading of output (\x is for psql only)
    \x on
    SELECT * from patients_logregr;

    -- ************ --
    --    Result    --
    -- ************ --
    coef                     | [-6.36346994178187, -1.02410605239327, 0.119044916668606]
    log_likelihood           | -9.41018298389
    std_err                  | [3.21389766375094, 1.17107844860319, 0.0549790458269309]
    z_stats                  | [-1.97998524145759, -0.874498248699549, 2.16527796868918]
    p_values                 | [0.0477051870698128, 0.38184697353045, 0.0303664045046168]
    odds_ratios              | [0.0017233763092323, 0.359117354054954, 1.12642051220895]
    condition_no             | 326.081922792
    num_rows_processed       | 20
    num_missing_rows_skipped | 0
    num_iterations           | 5
    variance_covariance      | [[10.3291381930637, -0.47430466519573, -0.171995901260052],
[-0.47430466519573, 1.37142473278285, -0.00119520703381598], [-0.171995901260052, -0.00119520703381598,
0.00302269548003977]]

    -- Alternatively, unnest the arrays in the results for easier reading of output (\x is for psql only)
    \x off
    SELECT unnest(array['intercept', 'treatment', 'trait_anxiety']) as attribute,
           unnest(coef) as coefficient,
           unnest(std_err) as standard_error,
           unnest(z_stats) as z_stat,
           unnest(p_values) as pvalue,
           unnest(odds_ratios) as odds_ratio
    FROM patients_logregr;

    -- ************ --
    --    Result    --
    -- ************ --
    +---------------+---------------+------------------+-----------+-----------+--------------+
    | attribute     |   coefficient |   standard_error |    z_stat |    pvalue |   odds_ratio |
    |---------------+---------------+------------------+-----------+-----------+--------------|
    | intercept     |     -6.36347  |         3.2139   | -1.97999  | 0.0477052 |   0.00172338 |
    | treatment     |     -1.02411  |         1.17108  | -0.874498 | 0.381847  |   0.359117   |
    | trait_anxiety |      0.119045 |         0.054979 |  2.16528  | 0.0303664 |   1.12642    |
    +---------------+---------------+------------------+-----------+-----------+--------------+
```

5. Now use the model to predict the dependent variable (second heart attack within 1 year) using the logistic regression model. For the purpose of demonstration, we will use the original data table to perform the prediction. Typically a different test dataset with the same features as the original training dataset would be used for prediction.

```
    -- Display prediction value along with the original value
SELECT p.id, madlib.logregr_predict(m.coef, ARRAY[1, p.treatment, p.trait_anxiety]),
       p.second_attack
FROM patients p, patients_logregr m
ORDER BY p.id;

-- ************ --
--    Result    --
-- ************ --
+------+-------------------+-----------------+
|   id | logregr_predict   |   second_attack |
|------+-------------------+-----------------|
|    1 | True              |               1 |
|    2 | True              |               1 |
|    3 | False             |               1 |
|    4 | True              |               1 |
|    5 | False             |               1 |
|    6 | True              |               1 |
|    7 | True              |               1 |
|    8 | True              |               1 |
|    9 | True              |               1 |
|   10 | True              |               1 |
|   11 | True              |               0 |
|   12 | False             |               0 |
|   13 | False             |               0 |
|   14 | False             |               0 |
|   15 | False             |               0 |
|   16 | False             |               0 |
|   17 | True              |               0 |
|   18 | False             |               0 |
|   19 | False             |               0 |
|   20 | True              |               0 |
+------+-------------------+-----------------+

-- Predicting the probability of the dependent variable being TRUE.
-- Display prediction value along with the original value
SELECT p.id, madlib.logregr_predict_prob(coef, ARRAY[1, treatment, trait_anxiety])
FROM patients p, patients_logregr m
ORDER BY p.id;

-- ************ --
--    Result    --
-- ************ --
+------+------------------------+
|   id |   logregr_predict_prob |
|------+------------------------|
|    1 |              0.720223  |
|    2 |              0.894355  |
|    3 |              0.19227   |
|    4 |              0.685513  |
|    5 |              0.167748  |
|    6 |              0.798098  |
|    7 |              0.928568  |
|    8 |              0.959306  |
|    9 |              0.877576  |
|   10 |              0.685513  |
|   11 |              0.586701  |
|   12 |              0.19227   |
|   13 |              0.116032  |
|   14 |              0.0383829 |
|   15 |              0.0674976 |
|   16 |              0.19227   |
|   17 |              0.545871  |
|   18 |              0.267675  |
|   19 |              0.398619  |
|   20 |              0.685513  |
+------+------------------------+
```

If the probability is greater than 0.5, the prediction is given as `True`. Otherwise it is given as `False`.

Next Steps

- For details on all of the machine learning functions provided by MADlib, please refer to User Documentation
- Try out the available Jupyter notebooks for many MADlib algorithms
- To contribute new modules to MADlib, please refer to the Quick Start Guide for Developers