The CREATE MODEL statement for Matrix Factorization

CREATE MODEL statement for Matrix Factorization

Matrix factorization models are only available to flat-rate customers or customers with reservations uery/docs/reservations-intro). On-demand customers are encouraged to use flex slots uery/pricing#flex-slots-pricing) to use matrix factorization.

To create a matrix factorization model in BigQuery, use the BigQuery ML `CREATE MODEL` statement and specify `MODEL_TYPE` to be `'MATRIX_FACTORIZATION'`.

CREATE MODEL syntax

```
TE MODEL (#create_model) | CREATE MODEL IF NOT EXISTS (#create_model_if_not_exists) | CREATE (
_name (#model_name)
NS(MODEL_TYPE = 'MATRIX_FACTORIZATION'
FEEDBACK_TYPE (#feedback_type) = {'EXPLICIT' | 'IMPLICIT'} ]
NUM_FACTORS (#num_factors) = int64_value ]
USER_COL (#user_col) = string_value ]
ITEM_COL (#item_col) = string_value ]
RATING_COL (#rating_col) = string_value ]
WALS_ALPHA (#wals_alpha) = float64_value ]
L2_REG (#l2_reg) = float64_value ]
MAX_ITERATIONS (#max_iterations) = int64_value ]
EARLY_STOP (#early_stop) = { TRUE | FALSE } ]
MIN_REL_PROGRESS (#min_rel_progress) = float64_value ]
DATA_SPLIT_METHOD (#data_split_method) = { 'AUTO_SPLIT' | 'RANDOM' | 'CUSTOM' | 'SEQ' |
DATA_SPLIT_EVAL_FRACTION (#data_split_eval_fraction) = float64_value ]
DATA_SPLIT_COL (#data_split_col) = string_value ])
ery_statement (#query_statement)
```

CREATE MODEL

Loading [MathJax]/jax/output/SVG/jax.js

Creates a new BigQuery ML model in the specified dataset. If the model name exists, `CREATE MODEL` returns an error.

CREATE MODEL IF NOT EXISTS

Creates a new BigQuery ML model only if the model does not currently exist in the specified dataset.

CREATE OR REPLACE MODEL

Creates a new BigQuery ML model and replaces any existing model with the same name in the specified dataset.

model_name

`model_name` is the name of the BigQuery ML model you're creating or replacing. The model name must be unique per dataset: no other model or table can have the same name. The model name must follow the same naming rules as a BigQuery table. A model name can contain the following:

- Up to 1,024 characters
- Letters of either case, numbers, and underscores

`model_name` is not case-sensitive.

If you do not have a default project configured, prepend the project ID to the model name in following format, including backticks:

`[PROJECT_ID].[DATASET].[MODEL]`

For example:

`myproject.mydataset.mymodel`

`CREATE MODEL` supports the following options:

MODEL_TYPE

Loading [MathJax]/jax/output/SVG/jax.js

Syntax

```
_TYPE = 'MATRIX_FACTORIZATION'
```

Description

Specifies the model type. To create a matrix factorization model, set `model_type` to `'MATRIX_FACTORIZATION'`.

model_option_list

In the `model_option_list`, the `model_type` option is required. All others are optional.

Matrix factorization models support the following options:

FEEDBACK_TYPE

Syntax

```
ACK_TYPE = { 'EXPLICIT' | 'IMPLICIT' }
```

Description

Specifies the feedback type for matrix factorization models. The feedback type determines the algorithm that is used during training.

There are two types of ratings (user feedback): `'EXPLICIT'` and `'IMPLICIT'`. Use the desired feedback type in model creation options depending on your use case.

- If the user has explicitly provided a rating (for example, 1-5) to an item such as movie recommendations, then specify `FEEDBACK_TYPE='EXPLICIT'`. This will train a model using the Alternating Least Squares (https://en.wikipedia.org/wiki/Matrix_completion#Alternating_least_squares_minimization) algorithm.
- Most product recommendation problems do not have explicit user feedback. Instead, the rating value must be artificially constructed based on user's interaction with the item (for example, clicks, pageviews, and purchases). In this situation, specify

Loading [MathJax]/jax/output/SVG/jax.js

`FEEDBACK_TYPE='IMPLICIT'`. This will train a model using the Weighted-Alternating Least Squares (http://yifanhu.net/PUB/cf.pdf) algorithm.

For more information about the differences between the two feedback types and when to use which type, see Additional information about feedback types (#feedback-info).

Arguments

The default value is `'EXPLICIT'`.

NUM_FACTORS

Syntax

`NUM_FACTORS = int64_value`

Description

Specifies the number of latent factors to use for matrix factorization models.

Arguments

`int64_value` is an `'INT64'`. Allowed values are 2-200. The default value is $\log_2(n)$, where `n` is the number of training examples.

USER_COL

Syntax

`USER_COL = string_value`

Description

The user column name for matrix factorization models.

Arguments *`string_value`* is a `'STRING'`. The default value is `'user'`.

ITEM_COL

Syntax

`ITEM_COL = string_value`G/jax.js

Description

The item column name for matrix factorization models.

Arguments ***`string_value`*** is a `'STRING'`. The default value is `'item'`.

RATING_COL

Syntax

`RATING_COL =` ***`string_value`***

Description

The rating column name for matrix factorization models.

Arguments ***`string_value`*** is a `'STRING'`. The default value is `'rating'`.

WALS_ALPHA

Syntax

`WALS_ALPHA =` ***`float64_value`***

Description

A hyperparameter for `'IMPLICIT'` matrix factorization model.

For more information, see Additional information about feedback types (#feedback-info)

Arguments ***`float64_value`*** is a `'FLOAT64'`. The default value is 40.

L2_REG

Syntax

`L2_REG =` ***`float64_value`***

Description

The amount of L2 regularization (https://developers.google.com/machine-learning/glossary/#L2_regularization) applied.

Loading [MathJax]/jax/output/SVG/jax.js

Arguments

float64_value is a `FLOAT64`. The default value is 1.0.

MAX_ITERATIONS

Syntax

`MAX_ITERATIONS = int64_value`

Description

The maximum number of training iterations or steps.

Arguments

int64_value is an `INT64`. The default value is 20.

EARLY_STOP

Syntax

```
_STOP = { TRUE | FALSE }
```

Description

Whether training should stop after the first iteration in which the relative loss improvement is less than the value specified for MIN_REL_PROGRESS (#min_rel_progress).

Arguments

The value is a `BOOL`. The default value is `TRUE`.

MIN_REL_PROGRESS

Syntax

`MIN_REL_PROGRESS = float64_value`

Description

Loading [MathJax]/jax/output/SVG/jax.js

The minimum relative loss improvement necessary to continue training when `EARLY_STOP` is set to true. For example, a value of 0.01 specifies that each iteration must reduce the loss by 1% for training to continue.

Arguments

`float64_value` is a `FLOAT64`. The default value is 0.01.

DATA_SPLIT_METHOD

Syntax

```
SPLIT_METHOD = { 'AUTO_SPLIT' | 'RANDOM' | 'CUSTOM' | 'SEQ' | 'NO_SPLIT' }
```

Description

The method to split input data into training and evaluation sets. Training data is used to train the model. Evaluation data is used to avoid overfitting (https://developers.google.com/machine-learning/glossary/#overfitting) via early stopping.

Arguments

Accepts the following values:

`'AUTO_SPLIT'` The automatic split strategy is as follows:

- When there are fewer than 500 rows in the input data, all rows are used as training data.
- When there are between 500 and 50,000 rows in the input data, 20% of the data is used as evaluation data in a `RANDOM` split.
- When there are more than 50,000 rows in the input data, only 10,000 of them are used as evaluation data in a `RANDOM` split.

`'RANDOM'` Split data randomly. A random split is deterministic: different training runs produce the same split results if the underlying training data remains the same.

`'CUSTOM'` Split data using a customer provided column of type `BOOL`. The rows with a value of `TRUE` are used as evaluation data. The rows with a value of `FALSE` are used as training data.

Loading [MathJax]/jax/output/SVG/jax.js

`'SEQ'` Split data sequentially using a customer-provided column. The column can have any orderable data type: `NUMERIC`, `STRING`, or `TIMESTAMP`. All rows with split values smaller than the threshold are used as training data. The remaining rows including `NULL`s are used as evaluation data.

`'NO_SPLIT'` Use all data as training data.

The default value for matrix factorization models is `'NO_SPLIT'`. While the other methods are supported, use
aution. Due to the nature of the matrix factorization algorithm, if a split eliminates all of the ratings for a user a
factor weight vector is not generated for the user and/or item.

DATA_SPLIT_EVAL_FRACTION

Syntax

`DATA_SPLIT_EVAL_FRACTION = `***`float64_value`***

Description

This option is used with `'RANDOM'` and `'SEQ'` splits. It specifies the fraction of the data used for evaluation, accurate to two decimal places.

Arguments

`float64_value` is a `FLOAT64`. The default value is 0.2.

DATA_SPLIT_COL

Syntax

`DATA_SPLIT_COL = `***`string_value`***

Description

Identifies the column used to split the data. This column cannot be used as a feature or label, and will be excluded from features automatically.

- When the value of `DATA_SPLIT_METHOD` is `'CUSTOM'`, the corresponding column should be of type `BOOL`. The rows with `TRUE` or `NULL` values are used as evaluation data. Rows with `FALSE` values are used as training data.

Loadi thJax]/jax/output/SVG/jax.js

- When the value of DATA_SPLIT_METHOD is 'SEQ', the last ***n*** rows from smallest to largest in the corresponding column are used as evaluation data, where ***n*** is the value specified for DATA_SPLIT_EVAL_FRACTION (#data_split_eval_fraction). The first rows are used as training data.

For information on supported input types, see Supported input types for DATA_SPLIT_COL (#split-inputs).

Arguments

string_value is a STRING.

query_statement

The AS query_statement clause specifies the standard SQL query that is used to generate the training data. For information about the supported SQL syntax of the query_statement clause, see Standard SQL query syntax (/bigquery/docs/reference/standard-sql/query-syntax#sql-syntax).

For matrix factorization models, the query_statement is expected to contain exactly 3 columns (user, item, and rating) unless the user specifies a DATA_SPLIT_METHOD that requires use of a DATA_SPLIT_COL.

Supported inputs

The CREATE MODEL statement supports the following data types for the user, item, and rating columns.

Supported data types for matrix factorization model inputs

BigQuery ML supports different standard SQL data types for the input columns for matrix factorization. Supported data types for each respective column include:

Matrix factorization input column	Supported types
user	Any groupable (/bigquery/docs/reference/standard-sql/data-types#data-type-properties) data type

Loading [MathJax]/jax/output/SVG/jax.js

`item`	Any groupable (/bigquery/docs/reference/standard-sql/data-types#data-type-properties) data type
`rating`	`INT64` (/bigquery/docs/reference/standard-sql/data-types#integer-type) `NUMERIC` (/bigquery/docs/reference/standard-sql/data-types#numeric-type) `FLOAT64` (/bigquery/docs/reference/standard-sql/data-types#floating-point-type)

Additional information about feedback types

An important part of creating a good matrix factorization model for recommendations is to make sure that data is trained on the algorithm that is best suited for it. For matrix factorization models, there are two different ways to get a rating for a user-item pair.

Ratings that the user had to input and set are considered to be explicit feedback. A low explicit rating tends to imply the user felt very negatively about an item while a high explicit rating tends to imply that the use liked the item. Movie streaming sites where users give ratings are examples of explicitly labeled datasets. For explicit feedback problems, we use the alternating least squares algorithm, commonly referred to as ALS. ALS seeks to minimize the following loss function:

$$Loss = \sum_{u,i \in \text{observed ratings}} (r_{ui} - x_u^T y_i)^2 + \lambda\left(\sum_u ||x_u||^2 + \sum_i ||y_i||^2\right)$$

Where

$r_{ui} =$ rating that user u gave to item i
$x_u =$ latent factor weights vector for user u. Is length `NUM_FACTORS` (#num_factors).
$y_i =$ latent factor weights vector for item i. Is length `NUM_FACTORS` (#num_factors).
$\lambda =$ `L2_REG` (#l2_reg)

However, most of the time data, is rarely labeled by users. Often, the only metrics that a company has as to whether a user liked an item or movie is by the click rate or engagement time. This can often be used as a proxy rating, but it is not necessarily a definitive indication as

Loading [MathJax]/jax/output/SVG/jax.js

to whether a user likes or dislikes something. The data in these datasets are considered to be implicit feedback. For implicit feedback problems, we use a variant of this algorithm called weighted-alternating least squares, or WALS, which is described in http://yifanhu.net/PUB/cf.pdf (http://yifanhu.net/PUB/cf.pdf). This approach uses these proxy ratings and treats them as a confidence for an observation that a user gives for an item. WALS seeks to minimize the following loss function:

$$Loss = \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2)$$

Where, in addition to the variables defined above, the function also introduces the following variables:

$p_{ui} = 1$ when $r_{ui} > 0$ and $p_{ui} = 0$ when $r_{ui} < 0$
$c_{ui} = 1 + \alpha r_{ui}$
$\alpha =$ `WALS_ALPHA` (#wals_alpha)

For explicit matrix factorization, the input is typically integers within a known fixed range. For implicit matrix factorization, the input ratings can be doubles or integers that span a wider range. We recommend that you make sure there aren't any outliers in the input ratings, and that you scale the input ratings if the model is performing poorly.

Known limitations

`CREATE MODEL` statements for matrix factorization models must comply with the following rules:

- If "Model is too large (>100 MB)" error is thrown, check the input data. This is caused by having too many ratings for a single user or single item. Hashing the user or item columns into an `INT64` value or reducing the data size can help. A general formula to determine whether this will occur is the following:

  `max(num_rated_user, num_rated_item) < 100 million`

  Where num_rated_user is the maximum item ratings that a single user has entered and num_rated_items is the maximum user ratings for a given item.

`CREATE MODEL` examples

Loading [MathJax]/jax/output/CommonHTML/jax.js

The following example creates models named `mymodel` in `mydataset` in your default project.

Training a matrix factorization model with explicit feedback

This example creates an explicit feedback matrix factorization model.

```
E MODEL `project_id.mydataset.mymodel`
ONS(MODEL_TYPE='MATRIX_FACTORIZATION') AS
T
r,
m,
ing

dataset.mytable`
```

Training a matrix factorization model with implicit feedback

This example creates an implicit feedback matrix factorization model.

```
E MODEL `project_id.mydataset.mymodel`
ONS(MODEL_TYPE='MATRIX_FACTORIZATION',
    FEEDBACK_TYPE='IMPLICIT') AS
T
r,
m,
ing

dataset.mytable`
```

What's next

- Walk through our tutorials that use the matrix factorization model in BigQuery ML:
  - Using BigQuery ML to make recommendations from Google analytics data (/bigquery-ml/docs/bigqueryml-mf-implicit-tutorial) (implicit feedback)
  - Using BigQuery ML to make recommendations from movie ratings (/bigquery-ml/docs/bigqueryml-mf-explicit-tutorial) (explicit feedback)

Loading [MathJax]/jax/output/SVG/jax.js

Loading [MathJax]/jax/output/SVG/jax.js