MovieLens 20M Rating Prediction Using Factorization Machine

Data Preparation

Download ml-20m.zip and unzip it. Then, create a database and import the raw ratings data into Treasure Data from the downloaded CSV. The `--time-value` is used to add a dummy time column. Because Treasure Data prefers that each row have a timestsamp.

```
$ td db:create movielens20m
$ td table:create movielens20m ratings
$ td import:auto --format csv --column-header --time-value `date +%s` --auto-create movielens20m.ratings .
/ratings.csv
```

The first step is to split the original data for training and testing.

```
$ td table:create movielens20m ratings_fm

$ td query -w --type hive -d movielens20m "
    INSERT OVERWRITE TABLE ratings_fm
    select
        rowid() as rowid,
        categorical_features(array('userid','movieid'), userid, movieid)
          as features,
        rating,
        rand(31) as rnd
    from
        ratings;
  "
```

Now, split the data 80% for training and 20% for testing.

```
$ td table:create movielens20m training_fm

$ td query -x --type hive -d movielens20m "
  INSERT OVERWRITE TABLE training_fm
  SELECT rowid, features, rating, rnd
  FROM ratings_fm
  ORDER BY rnd DESC
  LIMIT 16000000
"
```

Caution: If you got “java.lang.RuntimeException: INSERT into ‘v’ field is not supported”, then disable V-column of the table through TD Console. Or, avoid using ‘*’ in the query.

```
$ td table:create movielens20m testing_fm

$ td query -w --type hive -d movielens20m "
  INSERT OVERWRITE TABLE testing_fm
  SELECT rowid, features, rating, rnd
  FROM ratings_fm
  ORDER BY rnd ASC
  LIMIT 4000263
"

$ td table:create movielens20m testing_fm_exploded

$ td query -x --type hive -d movielens20m "
INSERT OVERWRITE TABLE testing_fm_exploded
select
  rowid,
  extract_feature(fv) as feature,
  extract_weight(fv) as Xi,
  rating
from
  testing_fm t1 LATERAL VIEW explode(add_bias(features)) t2 as fv
"
```

Caution: Don't forget to call "add_bias" in the above query. No need to call "add_bias" for preparing training data in Factorization Machines because it always considers it.

Training

```
$ td table:create movielens20m fm_model

$ td query -w -x --type hive -d movielens20m "
INSERT OVERWRITE TABLE fm_model
select
  feature,
  avg(Wi) as Wi,
  array_avg(Vif) as Vif
from (
  select
    train_fm(features, rating, '-factors 10 -iters 50 -min 1 -max 5')
        as (feature, Wi, Vif)
  from
    training_fm
) t
group by feature;
"
```

Training Options

You can get information about hyperparameter for training using `-help` option as follows:

```
$ td query -w --type hive -d movielens20m "
  select
    train_fm(features, rating, '-help')
        as (feature, Wi, Vif)
  from
    training_fm
"

usage: train_fm(array<string> x, double y [, const string options]) -
     Returns a prediction value [-adareg] [-c] [-cv_rate <arg>]
     [-disable_cv] [-eta <arg>] [-eta0 <arg>] [-f <arg>] [-help]
     [-init_v <arg>] [-int_feature] [-iters <arg>] [-lambda <arg>] [-max
     <arg>] [-maxval <arg>] [-min <arg>] [-min_init_stddev <arg>] [-p
     <arg>] [-power_t <arg>] [-seed <arg>] [-sigma <arg>] [-t <arg>]
     [-va_ratio <arg>] [-va_threshold <arg>]
-adareg,--adaptive_regularizaion             Whether to enable adaptive
                                            regularization [default:
                                            OFF]
-c,--classification                          Act as classification
-cv_rate,--convergence_rate <arg>            Threshold to determine
                                            convergence [default: 0.005]
-disable_cv,--disable_cvtest                 Whether to disable
                                            convergence check [default:
                                            OFF]
-eta <arg>                                   The initial learning rate
-eta0 <arg>                                  The initial learning rate
                                            [default 0.1]
-f,--factor <arg>                            The number of the latent
                                            variables [default: 10]
-help                                        Show function help
-init_v <arg>                                Initialization strategy of
                                            matrix V [random, gaussian]
                                            (default: random)
-int_feature,--feature_as_integer            Parse a feature as integer
                                            [default: OFF, ON if -p
                                            option is specified]
-iters,--iterations <arg>                    The number of iterations
                                            [default: 1]
-lambda,--lambda0 <arg>                      The initial lambda value for
                                            regularization [default:
                                            0.01]
-max,--max_target <arg>                      The maximum value of target
                                            variable
-maxval,--max_init_value <arg>               The maximum initial value in
                                            the matrix V [default: 1.0]
-min,--min_target <arg>                      The minimum value of target
                                            variable
-min_init_stddev <arg>                       The minimum standard
                                            deviation of initial matrix
                                            V [default: 0.1]
-p,--size_x <arg>                            The size of x
-power_t <arg>                               The exponent for inverse
                                            scaling learning rate
                                            [default 0.1]
-seed <arg>                                  Seed value [default: -1
                                            (random)]
-sigma <arg>                                 The standard deviation for
                                            initializing V [default:
                                            0.1]
-t,--total_steps <arg>                       The total number of training
                                            examples
-va_ratio,--validation_ratio <arg>           Ratio of training data used
                                            for validation [default:
                                            0.05f]
-va_threshold,--validation_threshold <arg>   Threshold to start
                                            validation. At least N
                                            training examples are used
                                            before validation [default:
                                            1000]
```

Prediction

```
$ td table:create movielens20m fm_predict

$ td query -w -x --type hive -d movielens20m "
INSERT OVERWRITE TABLE fm_predict
select
  t1.rowid,
  fm_predict(p1.Wi, p1.Vif, t1.Xi) as predicted
from
  testing_fm_exploded t1
  LEFT OUTER JOIN fm_model p1 ON (t1.feature = p1.feature)
group by
  t1.rowid
"
```

Evaluation

```
$ td query -w --type hive -d movielens20m "
  select
    mae(p.predicted, rating) as mae,
    rmse(p.predicted, rating) as rmse
  from
    testing_fm as t
    JOIN fm_predict as p on (t.rowid = p.rowid);
  "
```

mae	rmse
0.6056794023148888	0.7964665312607863

Fast Training Using Feature Hashing

Training of Factorization Machines (FM) can be done more efficiently, in terms of speed and memory consumption, by using INT features. In this section, you see how to run FM training by using int features, more specifically by using feature hashing.

Training Using Feature Hashing

Caution: Hivemall uses a dense array internally when both `-int_feature` and `-num_features` are specified, otherwise uses a sparse map. Dense array is more CPU efficient than sparse map. The default number of features created by feature_hashing is `2^24 = 16777216` and thus `-num_features 16777216` is the case for using `feature_hashing`.

```
$ td query -w -x --type hive -d movielens20m "
INSERT OVERWRITE TABLE fm_model
select
  feature,
  avg(Wi) as Wi,
  array_avg(Vif) as Vif
from (
  select
    train_fm(feature_hashing(features), rating,
      "-factor ${factor} -iters ${iters} -eta 0.01 -int_feature -num_features 16777216"
    ) as (feature, Wi, Vif)
  from
    training_fm
) t
group by feature;
"
```

Prediction and Evaluation Using Feature Hashing

Caution: DOT NOT forget to apply `feature_hashing` for test data.

```
$ td query -w --type hive -d movielens20m "
WITH predicted as (
  select
    t1.rowid,
    fm_predict(p1.Wi, p1.Vif, t1.Xi) as predicted
  from
    testing_fm_exploded t1
    LEFT OUTER JOIN fm_model p1 ON (feature_hashing(t1.feature) = p1.feature)
  group by
    t1.rowid
)
select
  mae(p.predicted, rating) as mae,
  rmse(p.predicted, rating) as rmse
from
  testing_fm as t
  JOIN predicted as p on (t.rowid = p.rowid);
"
```