
H2O.ai

H2O Generalized Low-Rank Models

```
h2o.glrm(training_frame, cols = NULL, model_id = NULL, validation_frame = NULL,
        ignore_const_cols = TRUE, score_each_iteration = FALSE, loading_name = NULL,
        transform = c("NONE", "STANDARDIZE", "NORMALIZE", "DEMEAN", "DESCALE"), k = 1,
        loss = c("Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic", "Periodic"),
        loss_by_col = c("Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic",
        "Periodic", "Categorical", "Ordinal"), loss_by_col_idx = NULL,
        multi_loss = c("Categorical", "Ordinal"), period = 1,
        regularization_x = c("None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse",
        "UnitOneSparse", "Simplex"), regularization_y = c("None", "Quadratic", "L2", "L1",
        "NonNegative", "OneSparse", "UnitOneSparse", "Simplex"), gamma_x = 0, gamma_y = 0,
        max_iterations = 1000, max_updates = 2000, init_step_size = 1,
        min_step_size = 1e-04, seed = -1, init = c("Random", "SVD", "PlusPlus", "User"),
        svd_method = c("GramSVD", "Power", "Randomized"), user_y = NULL, user_x = NULL,
        expand_user_y = TRUE, impute_original = FALSE, recover_svd = FALSE, max_runtime_secs = 0)
```

```
from h2o.estimators.glrm import H2OGeneralizedLowRankEstimator
model = H2OGeneralizedLowRankEstimator(…)
model.train(x = x, training_frame = data)
```

Irrelevant Data

Real Data	• Not all features are related to the target.
Not all data have value	• Noise can be mistaken as signal by machine learning algorithms.
Solution	• Once identified, remove from the analysis. Do not rely on algorithms to remove irrelevant features. Have doubts? Simulate random numeric and categorical features and find how many of them appear to be important.

H2O Generalized Low-Rank Models

```
h2o.glrm(training_frame, cols = NULL, model_id = NULL, validation_frame = NULL,
        ignore_const_cols = TRUE, score_each_iteration = FALSE, loading_name = NULL,
        transform = c("NONE", "STANDARDIZE", "NORMALIZE", "DEMEAN", "DESCALE"), k = 1,
        loss = c("Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic", "Periodic"),
        loss_by_col = c("Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic",
        "Periodic", "Categorical", "Ordinal"), loss_by_col_idx = NULL,
        multi_loss = c("Categorical", "Ordinal"), period = 1,
        regularization_x = c("None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse",
        "UnitOneSparse", "Simplex"), regularization_y = c("None", "Quadratic", "L2", "L1",
        "NonNegative", "OneSparse", "UnitOneSparse", "Simplex"), gamma_x = 0, gamma_y = 0,
        max_iterations = 1000, max_updates = 2000, init_step_size = 1,
        min_step_size = 1e-04, seed = -1, init = c("Random", "SVD", "PlusPlus", "User"),
        svd_method = c("GramSVD", "Power", "Randomized"), user_y = NULL, user_x = NULL,
        expand_user_y = TRUE, impute_original = FALSE, recover_svd = FALSE, max_runtime_secs = 0)
```

```
from h2o.estimators.glrm import H2OGeneralizedLowRankEstimator
model = H2OGeneralizedLowRankEstimator(…)
model.train(x = x, training_frame = data)
```