H_2O.ai

H2O Principal Components Analysis

```r
h2o.prcomp(training_frame, x, model_id = NULL, validation_frame = NULL,
           ignore_const_cols = TRUE, score_each_iteration = FALSE,
           transform = c("NONE", "STANDARDIZE", "NORMALIZE", "DEMEAN",
            "DESCALE"), pca_method = c("GramSVD", "Power", "Randomized",
            "GLRM"), k = 1, max_iterations = 1000,
            use_all_factor_levels = FALSE, compute_metrics = TRUE,
            impute_missing = FALSE, seed = -1, max_runtime_secs = 0)
```

```python
from h2o.estimators.pca import H2OPrincipalComponentAnalysisEstimator
pca = H2OPrincipalComponentAnalysisEstimator(…)
pca.train(x = x, training_frame = data)
```

Generalized Low-Rank Models

- GLRM is an extension of well-known matrix factorization methods such as Principal Component Analysis (PCA).
- Unlike PCA which is limited to numerical data, GLRM can also handle categorical, ordinal and Boolean data.
- **Given**: Data table *A* with *m* rows and *n* columns
- **Find**: Compressed representation as numeric tables *X* and *Y* where *k* is a small user-specified number

$$m\left\{\overbrace{\begin{bmatrix} \quad A \quad \end{bmatrix}}^{n}\right. \approx m\left\{\overbrace{\begin{bmatrix} X \end{bmatrix}}^{k}\right. \overbrace{\begin{bmatrix} \quad Y \quad \end{bmatrix}}^{n}\Big\}k$$

- *Y* = archetypal features created from columns of *A*
- *X* = row of A in reduced feature space
- GLRM can approximately reconstruct *A* from product *XY*

H2O Principal Components Analysis

```r
h2o.prcomp(training_frame, x, model_id = NULL, validation_frame = NULL,
           ignore_const_cols = TRUE, score_each_iteration = FALSE,
           transform = c("NONE", "STANDARDIZE", "NORMALIZE", "DEMEAN",
           "DESCALE"), pca_method = c("GramSVD", "Power", "Randomized",
           "GLRM"), k = 1, max_iterations = 1000,
           use_all_factor_levels = FALSE, compute_metrics = TRUE,
           impute_missing = FALSE, seed = -1, max_runtime_secs = 0)
```

```python
from h2o.estimators.pca import H2OPrincipalComponentAnalysisEstimator
pca = H2OPrincipalComponentAnalysisEstimator(…)
pca.train(x = x, training_frame = data)
```