Introduction to Generalised Low-Rank Model and Missing Values

Jo-fai (Joe) Chow
Data Scientist
joe@h2o.ai
@matlabulus

Based on work by Anqi Fu, Madeleine Udell, Corinne Horn, Reza Zadeh & Stephen Boyd.

About H2O.ai

- H2O in an open-source, distributed machine learning library written in Java with APIs in R, Python, Scala and REST/JSON.
- Produced by H2O.ai in Mountain View, CA.
- H2O.ai advisers are Trevor Hastie, Rob Tibshirani and Stephen Boyd from Stanford.

About Me

- 2005 - 2015
- Water Engineer
  - Consultant for Utilities
  - EngD Research
- 2015 - Present
- Data Scientist
  - Virgin Media
  - Domino Data Lab
  - H2O.ai

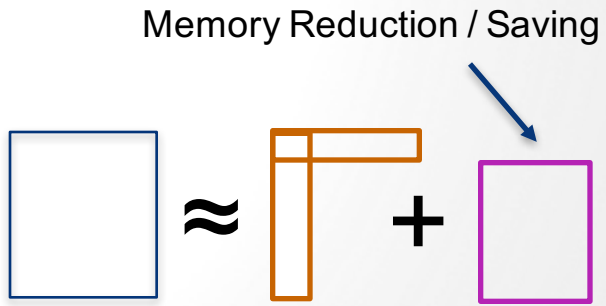
About This Talk

- Overview of generalised low-rank model (GLRM).
- Four application examples:
  - Basics.
  - How to accelerate machine learning.
  - How to visualise clusters.
  - How to impute missing values.
- Q & A.

GLRM Overview

- GLRM is an extension of well-known matrix factorisation 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

$$\underset{m}{\left\{\right.}\overbrace{\left[\quad A \quad\right]}^{n} \approx \underset{m}{\left\{\right.}\overbrace{\left[\quad X \quad\right]}^{k}\ \overbrace{\left[\quad Y \quad\right]}^{n}\left.\right\}k$$



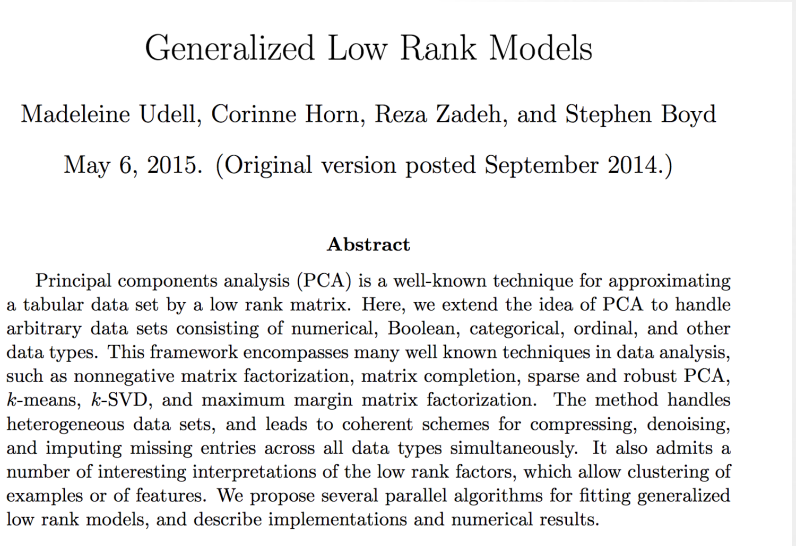
- *Y* = archetypal features created from columns of *A*
- *X* = row of A in reduced feature space
- GLRM can approximately reconstruct *A* from product *XY*

GLRM Key Features

- **Memory**
  - Compressing large data set with minimal loss in accuracy
- **Speed**
  - Reduced dimensionality = short model training time
- **Feature Engineering**
  - Condensed features can be analysed visually
- **Missing Data Imputation**
  - Reconstructing data set will automatically impute missing values

GLRM Technical References

- Paper
  - arxiv.org/abs/1410.0342
- Other Resources
  - H2O World Video
  - Tutorials



Generalized Low Rank Models

Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd

May 6, 2015. (Original version posted September 2014.)

Abstract

Principal components analysis (PCA) is a well-known technique for approximating a tabular data set by a low rank matrix. Here, we extend the idea of PCA to handle arbitrary data sets consisting of numerical, Boolean, categorical, ordinal, and other data types. This framework encompasses many well known techniques in data analysis, such as nonnegative matrix factorization, matrix completion, sparse and robust PCA, k-means, k-SVD, and maximum margin matrix factorization. The method handles heterogeneous data sets, and leads to coherent schemes for compressing, denoising, and imputing missing entries across all data types simultaneously. It also admits a number of interesting interpretations of the low rank factors, which allow clustering of examples or of features. We propose several parallel algorithms for fitting generalized low rank models, and describe implementations and numerical results.

Example 1: Motor Trend Car Road Tests

"mtcars" dataset in R

n = 11

m = 32

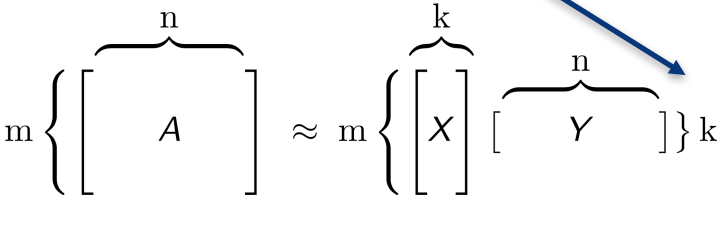
```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

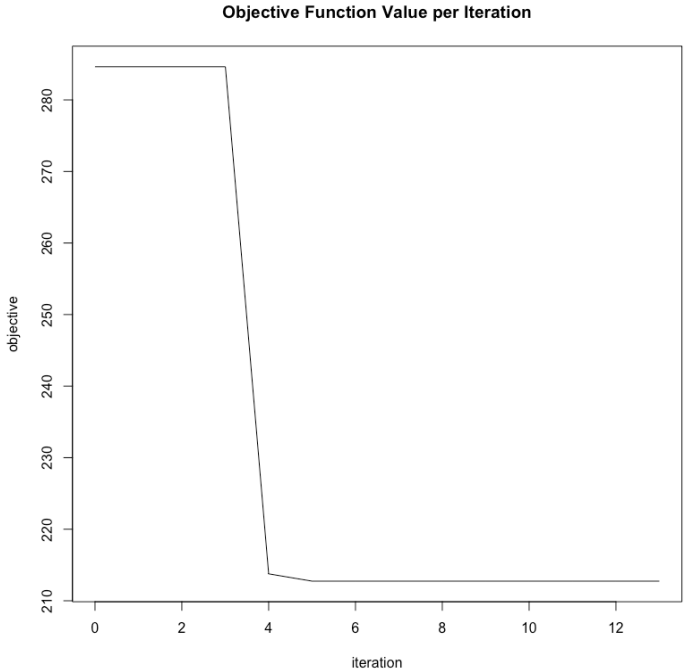
Original Data Table

$$m\left\{\left[\overbrace{\quad A \quad}^{n}\right]\right. \approx m\left\{\left[\overbrace{X}^{k}\right]\overbrace{\left[\quad Y \quad\right]}^{n}\right\}k$$

Example 1: Training a GLRM

```r
# Initialise H2O Cluster
library(h2o)
h2o.init(nthreads = -1)

# Load and import mtcars data
data(mtcars)
hex_mtcars ← as.h2o(mtcars)

# Build a GLRM and compress data into a 3-column matrix
glrm_mtcars ← h2o.glrm(training_frame = hex_mtcars,
                       k = 3, seed = 1234)

# Check convergence
plot(glrm_mtcars)
```





Example 1: X and Y from GLRM

3

```
> X <- h2o.getFrame(glrm_mtcars@model$representation_name)
> print(as.data.frame(X))
```

	Arch1	Arch2	Arch3
1	0.70215790	0.07243783	0.15437558
2	0.71247885	0.06503881	0.15991126
3	1.02435370	0.06261124	-0.13989618
4	0.03202082	-0.57575008	1.60055217
5	-0.26444539	-0.09051645	1.42864559
6	0.15816143	-0.38871398	1.19466072
7	0.21369543	1.12529204	-0.35398127
8	0.52474104	-0.72363776	1.22207822
9	0.87206152	-0.16622314	0.33217725
10	0.74588811	0.24471634	-0.06286804
11	0.71814005	0.26654037	-0.07981878
12	0.35528158	0.55637[illegible]	0.04944270
13	0.38342268	0.53547[illegible]	0.06506703
14	0.33281056	0.[illegible]582	0.03518461
15	-1.05315545	-0.[illegible]1939	2.21453898
16	-0.87636717	0.[illegible]884	1.79458983
17	-0.48050470	0.35631[illegible]2	1.20280683
18	1.28455006	0.39612[illegible]	0.25412546
19	1.10982039	-0.54983319	0.51755064
20	1.38192559	-0.40140627	0.19558614
21	0.95562884	0.06507248	-0.07475759
22	-0.26325328	-0.15489231	1.37226578
23	-0.15762926	-0.06792138	1.16876121
24	0.25501405	1.21174521	-0.51697903
25	-0.55072956	-0.35159377	2.02066047
26	1.13122594	-0.28350109	0.17124408
27	0.96905862	-0.09463050	0.12999972
28	1.47816899	0.31969412	-0.70490833
29	0.47512505	1.45761448	-0.92465792
30	1.33969769	1.23270517	-1.67116221
31	1.45542785	2.90965358	-3.38764080
32	1.02969787	0.26193304	-0.35805575

32

X

11

```
> Y <- glrm_mtcars@model$archetypes
> Y
Archetypes:
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Arch1	24.546607	4.828778	122.307646	96.888321	3.882959	2.449971	18.490718	0.729608	0.638545	4.012193	2.201411
Arch2	13.298687	8.431597	390.575251	248.418136	3.273191	4.289782	16.375532	-0.056826	0.198540	3.480427	4.408219
Arch3	17.561549	6.874096	299.136288	156.114641	3.277853	3.687826	17.714195	0.301478	0.144801	3.205061	2.510986

3

Y

$$\underset{m}{\left\{\left[\overbrace{\quad A \quad}^{n}\right]\right.} \approx \underset{m}{\left\{\left[\overbrace{\quad X \quad}^{k}\right]\right.} \overbrace{\left[\quad Y \quad\right]}^{n} \left.\right\} k$$

Example 1: Summary

```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

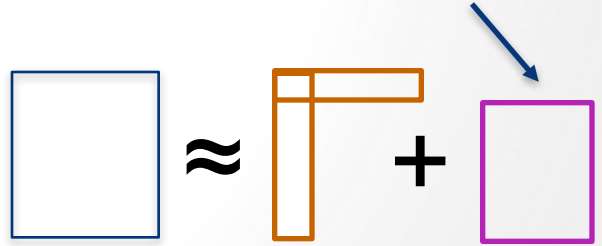
A ≈ X

```
> X <- h2o.getFrame(glrm_mtcars@model$representation_name)
> print(as.data.frame(X))
         Arch1       Arch2       Arch3
1   0.70215790  0.07243783  0.15437558
2   0.71247885  0.06503881  0.15991126
3   1.02435370  0.06261124 -0.13989618
4   0.03202082 -0.57575008  1.60055217
5  -0.26444539 -0.09051645  1.42864559
6   0.15816143 -0.38871398  1.19466072
7   0.21369543  1.12529204 -0.35398127
8   0.52474104 -0.72363776  1.22207822
9   0.87206152 -0.16622314  0.33217725
10  0.74588811  0.24471634 -0.06286804
11  0.71814005  0.26654037 -0.07981878
12  0.35528158  0.55637140  0.04944270
13  0.38342268  0.[illegible]547[illegible]  0.06506703
14  0.33281056  0.5[illegible]32  0.03518461
15 -1.05315545 -0.15[illegible]939  2.21453898
16 -0.87636717  0.[illegible]4  1.79458983
17 -0.48050470  0.[illegible]6329  1.20280683
18  1.28455006  0.[illegible]3961230  0.25412546
19  1.10982039 -0.54983319  0.51755064
20  1.38192559 -0.40140627  0.19558614
21  0.95562884  0.06507248 -0.07475759
22 -0.26325328 -0.15489231  1.37226578
23 -0.15762926 -0.06792138  1.16876121
24  0.25501405  1.21174521 -0.51697903
25 -0.55072956 -0.35159377  2.02066047
26  1.13122594 -0.28350109  0.17124408
27  0.96905862 -0.09463050  0.12999972
28  1.47816899  0.31969412 -0.70490833
29  0.47512505  1.45761448 -0.92465792
30  1.33969769  1.23270517 -1.67116221
31  1.45542785  2.90965358 -3.38764080
32  1.02969787  0.26193304 -0.35805575
```

```
> Y <- glrm_mtcars@model$archetypes
> Y
Archetypes:
            mpg      cyl       disp         hp     drat       wt      qsec       vs       am     gear     carb
Arch1 24.546607 4.828778 122.307646  96.888321 3.882959 2.449971 18.490718 0.729608 0.638545 4.012193 2.201411
Arch2 13.298687 8.431597 390.575251 248.418136 3.273191 4.289782 16.375532 -0.056826 0.198540 3.480427 4.408219
Arch3 17.561549 6.874096 299.136288 156.114641 3.277853 3.687826 17.714195  0.301478 0.144801 3.205061 2.510986
```

Y

Memory Reduction / Saving



$$m\left\{\left[\overbrace{\quad A \quad}^{n}\right]\right. \approx m\left\{\left[\overbrace{\quad X \quad}^{k}\right]\left[\overbrace{\quad Y \quad}^{n}\right]\right\}k$$

Example 2: ML Acceleration

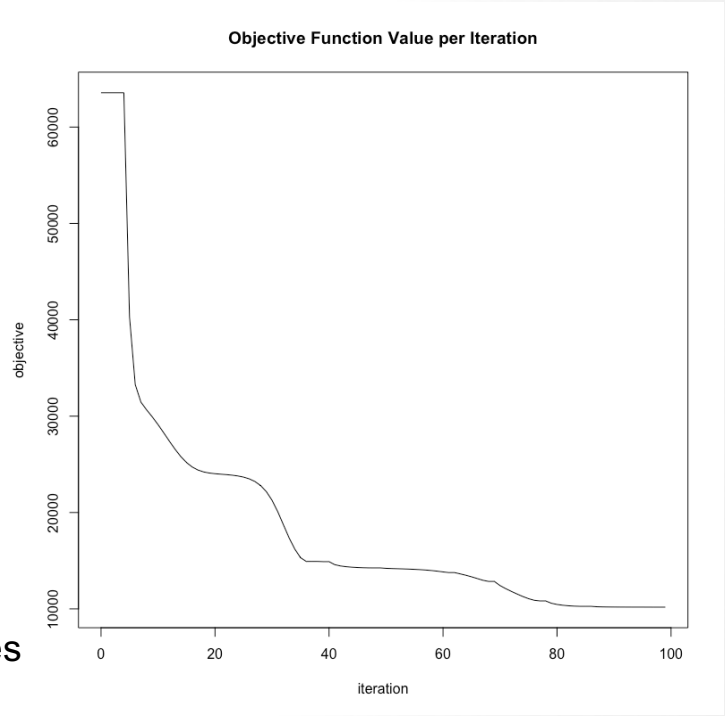
- About the dataset
  - R package “mlbench”
  - Multi-spectral scanner image data
  - x1 to x36: predictors
  - Classes:
    - 6 levels
    - Different type of soil
  - Use GLRM to compress predictors

```
> head(Satellite[sample(1:nrow(Satellite), 100),], 10)
     x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16 x.17 x.18 x.19 x.20 x.21
2474  64 112 128 103  64 116 122  99  64  121  122   96   67  111  123  100   67  116  123  100   71
4800  68  71  75  59  60  57  60  45  53   54   53   38   63   61   63   42   55   51   50   29   55
4165  63  96 108  89  66  96 112  89  66  100  112   92   67   99  110   94   67  103  114   94   71
5924  67  92 101  76  63 102 114  90  67  102  114   94   64   85   98   76   64   89  106   83   64
1327  59  56  80  70  59  60  80  63  66   63   76   66   63   57   86   72   59   57   82   68   59
6393  50  73  94  79  53  81 102  83  53   77   98   79   46   79   96   78   50   79   96   81   53
1838  72  81  82  65  76  81  82  65  72   85   86   68   71   83   83   63   76   79   79   67   71
2826  71 107 118  96  67 103 113  96  67  107  118   96   63  109  117  100   63  104  117   96   63
3397  55  79  93  75  51  75  93  75  51   79   96   79   56   83   96   85   56   83  108   85   56
1175  84  95 100  78  80  91  96  81  71   87   91   74   79   95   96   79   79   91   93   75   79
     x.22 x.23 x.24 x.25 x.26 x.27 x.28 x.29 x.30 x.31 x.32 x.33 x.34 x.35 x.36              classes
2474  111  128  100   66  113  122  100   66  113  127  100   66  113  122  100            red soil
4800   54   57   37   63   67   69   52   59   56   62   48   56   53   66   48 vegetation stubble
4165  103  114   98   67  106  114   94   70  106  119   94   70  106  119   94            red soil
5924  102  115   91   68   87   96   78   68   87  100   78   64   95  104   81            red soil
1327   60   82   68   57   56   82   73   57   53   85   76   57   56   82   65 vegetation stubble
6393   79   96   81   48   68   89   75   48   75   89   79   51   75   96   79            red soil
1838   83   87   63   71   79   85   62   71   79   85   62   71   79   85   67 very damp grey soil
2826  109  112   92   63  103  119   90   63  103  119   94   67  103  119   94            red soil
3397   83  100   81   56   81   93   83   56   88  101   83   56   88  105   83            red soil
1175   91   96   75   82   96  100   81   78   91   96   78   78   91   96   78      damp grey soil
```

Example 2: Use GLRM to Speed Up ML

```r
# Initialise H2O Cluster
library(h2o)
h2o.init(nthreads = -1)

# Use Satellite data from "mlbench"
library(mlbench)
library(data.table)
data("Satellite")

hex_sat ← as.h2o(Satellite)

# Build a GLRM and compress data into a 10-column matrix
glrm_sat ← h2o.glrm(training_frame = hex_sat[, -37], # exclude response
                    k = 6, seed = 1234,
                    transform = "STANDARDIZE",
                    regularization_x = "Quadratic",
                    regularization_y = "L1",
                    max_iterations = 100)

# Check convergence
plot(glrm_sat)
```

k = 6
Reduce to 6 features



Example 2: Random Forest

- Train a vanilla H2O Random Forest model with …
  - Full data set (36 predictors)
  - Compressed data set (6 predictors)

```
# Train DRF with Full Dataset
time_start ← proc.time()
model_drf_full ← h2o.randomForest(training_frame = hex_sat,
                                  x = features_full,
                                  y = "classes",
                                  ntrees = 500,
                                  nfolds = 10)
time_end_full ← timetaken(time_start)
model_drf_full

# Train DRF with Compressed Dataset
time_start ← proc.time()
model_drf_glrm ← h2o.randomForest(training_frame = hex_sat_glrm,
                                  x = features_glrm,
                                  y = "classes",
                                  ntrees = 500,
                                  nfolds = 10)
time_end_glrm ← timetaken(time_start)
model_drf_glrm
```

Example 2: Results Comparison

Data	Time	10-fold Cross Validation	
		Log Loss	Accuracy
Raw data (36 Predictors)	4 mins 26 sec	0.24553	91.80%
Data compressed with GLRM (6 Predictors)	1 min 24 sec	0.25792	90.59%

- Benefits of GLRM
  - Shorter training time
  - Quick insight before running models on full data set

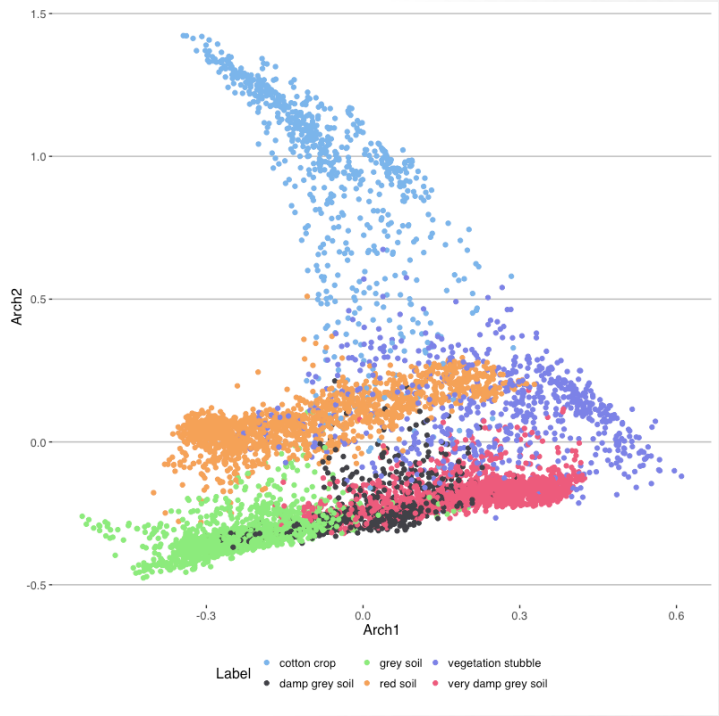
Example 3: Clusters Visualisation

- About the dataset
  - Multi-spectral scanner image data
  - Same as example 2
  - x1 to x36: predictors
  - Use GLRM to compress predictors to 2D representation
  - Use 6 classes to colour clusters

```
> head(Satellite[sample(1:nrow(Satellite), 100),], 10)
     x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16 x.17 x.18 x.19 x.20 x.21
2474  64 112 128 103  64 116 122  99  64  121  122   96   67  111  123  100   67  116  123  100   71
4800  68  71  75  59  60  57  60  45  53   54   53   38   63   61   63   42   55   51   50   29   55
4165  63  96 108  89  66  96 112  89  66  100  112   92   67   99  110   94   67  103  114   94   71
5924  67  92 101  76  63 102 114  90  67  102  114   94   64   85   98   76   64   89  106   83   64
1327  59  56  80  70  59  60  80  63  66   63   76   66   63   57   86   72   59   57   82   68   59
6393  50  73  94  79  53  81 102  83  53   77   98   79   46   79   96   78   50   79   96   81   53
1838  72  81  82  65  76  81  82  65  72   85   86   68   71   83   83   63   76   79   79   67   71
2826  71 107 118  96  67 103 113  96  67  107  118   96   63  109  117  100   63  104  117   96   63
3397  55  79  93  75  51  75  93  75  51   79   96   79   56   83   96   85   56   83  108   85   56
1175  84  95 100  78  80  91  96  81  71   87   91   74   79   95   96   79   79   91   93   75   79
     x.22 x.23 x.24 x.25 x.26 x.27 x.28 x.29 x.30 x.31 x.32 x.33 x.34 x.35 x.36              classes
2474  111  128  100   66  113  122  100   66  113  127  100   66  113  122  100            red soil
4800   54   57   37   63   67   69   52   59   56   62   48   56   53   66   48  vegetation stubble
4165  103  114   98   67  106  114   94   70  106  119   94   70  106  119   94            red soil
5924  102  115   91   68   87   96   78   68   87  100   78   64   95  104   81            red soil
1327   60   82   68   57   56   82   73   57   53   85   76   57   56   82   65  vegetation stubble
6393   79   96   81   48   68   89   75   48   75   89   79   51   75   96   79            red soil
1838   83   87   63   71   79   85   62   71   79   85   62   71   79   85   67 very damp grey soil
2826  109  112   92   63  103  119   90   63  103  119   94   67  103  119   94            red soil
3397   83  100   81   56   81   93   83   56   88  101   83   56   88  105   83            red soil
1175   91   96   75   82   96  100   81   78   91   96   78   78   91   96   78      damp grey soil
```

Example 3: Clusters Visualisation

```r
# Initialise H2O Cluster
library(h2o)
h2o.init(nthreads = -1)

# Use Satellite data from "mlbench"
library(mlbench)
data("Satellite")
hex_sat ← as.h2o(Satellite)

# Build a GLRM and compress data into a 10-column matrix
glrm_sat ← h2o.glrm(training_frame = hex_sat[, -37], # exclude response
                    k = 2, seed = 1234,
                    transform = "STANDARDIZE",
                    regularization_x = "Quadratic",
                    regularization_y = "L1",
                    max_iterations = 100)

# Extract X from GLRM
X ← as.data.frame(h2o.getFrame(glrm_sat@model$representation_name))

# Create data frame for data visualisation
d_sat ← data.frame(X, as.data.frame(hex_sat[, 37]))
colnames(d_sat) ← c("Arch1", "Arch2", "Label")
d_sat$Label ← as.factor(d_sat$Label)

# Visualise
library(ggplot2)
library(ggthemes)
ggplot(d_sat, aes(Arch1, Arch2, colour = Label)) +
  geom_point() + theme_hc() + scale_colour_hc()
  ggtitle("Clusters in Satellite Dataset")
```



Example 4: Imputation

”mtcars” – same dataset for example 1

```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Randomly introduce 50% missing values

```
> mtcars_with_na
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6    NA 110 3.90    NA 16.46  0 NA    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875    NA NA NA    4    4
Datsun 710          22.8   4 108.0  NA 3.85    NA    NA NA  1   NA    1
Hornet 4 Drive      21.4  NA 258.0  NA 3.08    NA    NA  1 NA   NA    1
Hornet Sportabout   18.7  NA    NA  NA   NA    NA 17.02  0  0   NA    2
Valiant               NA   6    NA 105 2.76 3.460 20.22  1 NA   NA    1
Duster 360          14.3   8    NA 245   NA 3.570 15.84 NA NA   NA   NA
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00 NA NA    4   NA
Merc 230              NA   4    NA  NA 3.92    NA 22.90  1  0    4    2
Merc 280              NA   6    NA 123   NA    NA    NA  1 NA   NA   NA
Merc 280C             NA   6 167.6  NA   NA    NA 18.90 NA NA    4    4
Merc 450SE            NA   8 275.8  NA   NA 4.070    NA  0 NA   NA   NA
Merc 450SL            NA  NA 275.8  NA 3.07 3.730    NA  0 NA    3    3
Merc 450SLC           NA   8    NA 180   NA 3.780    NA  0  0    3    3
Cadillac Fleetwood  10.4   8    NA  NA 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0  NA   NA 5.424 17.82  0 NA   NA   NA
Chrysler Imperial   14.7   8 440.0 230 3.23    NA 17.42  0 NA    3   NA
Fiat 128            32.4   4  78.7  66   NA 2.200    NA NA NA    4    1
Honda Civic         30.4   4    NA  NA 4.93    NA 18.52  1  1    4   NA
Toyota Corolla        NA   4    NA  65 4.22 1.835    NA  1 NA    4   NA
Toyota Corona       21.5   4 120.1  NA   NA 2.465 20.01 NA  0   NA   NA
Dodge Challenger    15.5   8 318.0 150 2.76    NA 16.87  0  0    3   NA
AMC Javelin         15.2  NA 304.0 150   NA    NA 17.30  0  0   NA    2
Camaro Z28            NA   8 350.0  NA 3.73 3.840    NA  0  0   NA   NA
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845    NA  0  0    3    2
Fiat X1-9           27.3   4  79.0  NA 4.08 1.935    NA  1  1   NA    1
Porsche 914-2         NA   4 120.3  91 4.43    NA    NA  0 NA    5    2
Lotus Europa          NA   4  95.1 113 3.77 1.513 16.90 NA  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino          NA   6 145.0 175   NA    NA    NA NA NA    5    6
Maserati Bora         NA  NA    NA  NA   NA 3.570    NA  0 NA    5    8
Volvo 142E          21.4   4 121.0 109 4.11    NA    NA NA NA    4   NA
```

Example 4: GLRM with NAs

```
# Convert to H2O Data Frame
hex_mtcars_na ← as.h2o(mtcars_with_na)

# Build a GLRM using data table with missing values
glrm_mtcars ← h2o.glrm(training_frame = hex_mtcars_na,
                       k = 10, seed = 1234,
                       init = "SVD",
                       regularization_x = "Quadratic",
                       regularization_y = "Quadratic",
                       max_iterations = 100)

# Use GLRM to impute missing values
mtcars_recon ← h2o.predict(glrm_mtcars, hex_mtcars_na)
```

When we reconstruct the table using GLRM, missing values are automatically imputed.

```
> mtcars_recon
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	222.2	110.0	3.9	2.7	16.4	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110.0	3.9	2.9	17.7	0	1	4	4
Datsun 710	22.8	4	108.0	149.3	3.8	2.4	18.6	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110.0	3.1	3.2	17.7	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175.0	3.2	3.2	17.0	0	0	3	2
Valiant	19.7	6	222.2	105.0	2.7	3.5	20.2	1	0	4	1
Duster 360	14.3	8	222.2	245.0	3.2	3.6	15.8	0	0	3	4
Merc 240D	24.4	4	146.7	62.0	3.7	3.1	20.0	0	1	4	2
Merc 230	22.8	4	140.8	95.0	3.9	3.1	22.9	1	0	4	2
Merc 280	19.2	6	222.2	123.0	3.9	3.1	17.7	1	0	4	3
Merc 280C	19.7	6	167.6	149.3	3.9	3.1	18.9	1	0	4	4
Merc 450SE	16.4	8	275.8	180.0	3.1	4.1	17.4	0	0	3	3
Merc 450SL	17.3	8	275.8	180.0	3.1	3.7	17.6	0	0	3	3
Merc 450SLC	15.2	8	275.8	180.0	3.1	3.8	18.0	0	0	3	3
Cadillac Fleetwood	10.4	8	222.2	205.0	2.9	5.2	18.0	0	0	3	4
Lincoln Continental	10.4	8	460.0	215.0	3.0	5.5	17.8	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230.0	3.2	5.3	17.4	0	0	3	3
Fiat 128	32.4	4	78.7	66.0	4.1	2.2	19.5	0	1	4	1
Honda Civic	30.4	4	75.7	52.0	4.9	1.7	18.5	1	1	4	2
Toyota Corolla	33.9	4	71.1	65.0	4.2	1.8	17.7	1	1	4	1
Toyota Corona	21.5	4	120.1	97.0	3.7	2.5	20.0	1	0	3	1
Dodge Challenger	15.5	8	318.0	150.0	2.8	3.2	16.9	0	0	3	2
AMC Javelin	15.2	8	304.0	150.0	3.1	3.2	17.3	0	0	3	2
Camaro Z28	13.3	8	350.0	245.0	3.7	3.8	15.4	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175.0	3.1	3.8	17.1	0	0	3	2
Fiat X1-9	27.3	4	79.0	66.0	4.1	2.0	18.9	1	1	4	1
Porsche 914-2	19.7	4	120.3	91.0	4.5	2.1	16.7	0	1	5	2
Lotus Europa	19.6	4	95.1	113.0	3.8	1.5	16.9	0	1	5	2
Ford Pantera L	15.8	8	351.0	264.0	4.2	3.2	14.5	0	1	5	4
Ferrari Dino	19.7	6	145.0	175.0	3.6	2.7	15.5	0	1	5	6
Maserati Bora	15.0	8	301.0	335.0	3.5	3.6	14.6	0	1	5	8
Volvo 142E	21.4	4	121.0	109.0	4.1	2.8	18.6	1	1	4	2

Example 4: Results Comparison

- We are asking GLRM to do a difficult job
  - 50% missing values
  - Imputation results look reasonable

Absolute difference between original and imputed values.

```
> round(mtcars_recon - mtcars, 1)
                   mpg cyl   disp hp drat   wt qsec vs am gear carb
Mazda RX4          0.0   0   62.2  0    0  0.0  0.0  0  0    0    0
Mazda RX4 Wag      0.0   0    0.0  0    0  0.0  0.7  0  0    0    0
Datsun 710         0.0   0    0.0 56    0  0.1  0.0  0  0    0    0
Hornet 4 Drive     0.0   0    0.0  0    0  0.0 -1.7  0  0    0    0
Hornet Sportabout  0.0   0    0.0  0    0 -0.3  0.0  0  0    0    0
Valiant            1.6   0   -2.8  0    0  0.0  0.0  0  0    1    0
Duster 360         0.0   0 -137.8  0    0  0.0  0.0  0  0    0    0
Merc 240D          0.0   0    0.0  0    0  0.0  0.0 -1  1    0    0
Merc 230           0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Merc 280           0.0   0   54.6  0    0 -0.3 -0.6  0  0    0   -1
Merc 280C          1.9   0    0.0 26    0 -0.3  0.0  0  0    0    0
Merc 450SE         0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Merc 450SL         0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Merc 450SLC        0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Cadillac Fleetwood 0.0   0 -249.8  0    0  0.0  0.0  0  0    0    0
Lincoln Continental 0.0  0    0.0  0    0  0.0  0.0  0  0    0    0
Chrysler Imperial  0.0   0    0.0  0    0  0.0  0.0  0  0    0   -1
Fiat 128           0.0   0    0.0  0    0  0.0  0.0 -1  0    0    0
Honda Civic        0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Toyota Corolla     0.0   0    0.0  0    0  0.0 -2.2  0  0    0    0
Toyota Corona      0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Dodge Challenger   0.0   0    0.0  0    0 -0.4  0.0  0  0    0    0
AMC Javelin        0.0   0    0.0  0    0 -0.3  0.0  0  0    0    0
Camaro Z28         0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Pontiac Firebird   0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Fiat X1-9          0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Porsche 914-2     -6.3   0    0.0  0    0  0.0  0.0  0  0    0    0
Lotus Europa     -10.8   0    0.0  0    0  0.0  0.0 -1  0    0    0
Ford Pantera L     0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Ferrari Dino       0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Maserati Bora      0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
Volvo 142E         0.0   0    0.0  0    0  0.0  0.0  0  0    0    0
```

Conclusions

- Use GLRM to
  - Save memory
  - Speed up machine learning
  - Visualise clusters
  - Impute missing values
- A great tool for data pre-processing
  - Include it in your data pipeline

Any Questions?

- Contact
  - joe@h2o.ai
  - @matlabulous
  - github.com/woobe
- Slides & Code
  - github.com/h2oai/h2o-meetups
- H2O in London
  - Meetups / Office (soon)
  - www.h2o.ai/careers
- H2O Help Docs & Tutorials
  - www.h2o.ai/docs
  - university.h2o.ai