Java Library for Machine Learning (JML)

Mingjie Qian
Department of Computer Science
University of Illinois at Urbana-Champaign
Feb. 2nd, 2014

Features

- Pure Java, cross-platform
- A general framework for users to implement machine learning tools
- Implementations for commonly-used machine learning methods and optimization algorithms
- Well documented source code
- Friendly API, **very** easy to use
- Complete separation between feature engineering and model training

SourceForge: http://sourceforge.net/projects/jlml/ since 03072013
Download statstics: 607 downloads from 55 countries (regions) before 02022014

JML V.S. LAML

- LAML is much faster than JML (more than 3 times faster).
- JML relies on third party linear algebra library, i.e. Apache Commons-math. Sparse matrices and vectors have been deprecated in Commons-math 3.0+, and will be ultimately eliminated. Whereas LAML has its own built-in linear algebra library.
- LAML also provides a lot of commonly used matrix functions in the same signature to Matlab, thus can also be used to manually convert MATLAB code to Java code.
- In short, JML has been replaced by LAML.

Built-in Packages Overview

- jml.clustering
- jml.classification
- jml.topics
- jml.data
- jml.matlab
- jml.optimization
- jml.sequence
- jml.subspace
- ...

Data Interface

- jml.data: IO for dense or sparse matrices
- Load a matrix from a txt file
- Save a matrix to a txt file
- JMatIO can also be used to load and save MATLAB MAT files in Java
- docTermCountArray2Matrix
  - TextProcessor can transform a corpus to a docTermCountArray
- readProblemFromStringArray + features2Matrix

Matlab

- jml.matlab: Commonly used Matlab matrix functions with almost same function input signature
  - ones, zeros, eye, diag, rand, size, sparse, full
  - sort, sum, max, min, kron, vec, cat, vertcat, horzcat, repmat, reshape, find, colon, display
  - svd, eigs, ldivide, rdivide, mldivide, mrdivide, mtimes, times, plus, minus, power, norm, subplus
  - not, and, or, eq, le, ge, lt, gt, isinf, isnan
  - ...

Classification

- jml.classification
  - MCSVM, Logistic Regression, MaxEnt, AdaBoost

```
Classifier multiClassSVM = new MultiClassSVM(C, eps);
multiClassSVM.feedData(trainData);    // double[][] or RealMatrix
multiClassSVM.feedLabels(labels);     // double[][], int[], or RealMatrix
multiClassSVM.train();

multiClassSVM.predict(testData);      // Predicted labels (int[])
```

Clustering

- jml.clustering
  - KMeans, NMFs, Spectral Clustering

```
Clustering spectralClustering = new SpectralClustering(options);
spectralClustering.feedData(data);                // double[][] or RealMatrix
spectralClustering.clustering();

display(spectralClustering.getIndicatorMatrix());
```

Topic Modeling

- jml.topics
  - LDA, LSI // PLSA <=> NMF
  - Read corpora with multiple format:
    - int[][], DocTermCountArray, RealMatrix, LDAInput

```
int[][] documents = { {1, 4, 3, 2, 3, 1, 4, 3, 2, 3, 1, 4, 3, 2, 3, 6},
                      {2, 2, 4, 2, 4, 2, 2, 2, 2, 4, 2, 2},
                      {1, 6, 5, 6, 0, 1, 6, 5, 6, 0, 1, 6, 5, 6, 0, 0} };

LDA LDA = new LDA(LDAOptions);
LDA.readCorpus(documents);          // Multiple corpus input format
LDA.train();

display(LDA.topicMatrix);
display(LDA.indicatorMatrix);
```

Sequence Labeling

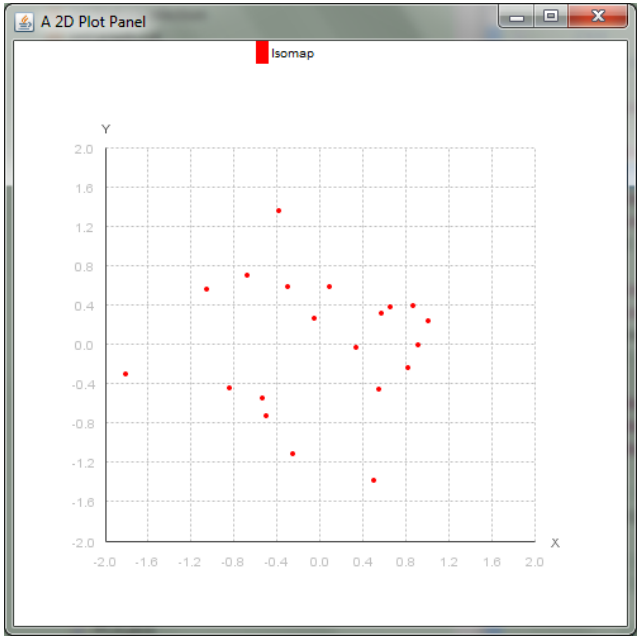
- jml.sequence
  - Conditional Random Field Using L-BFGS
  - Hidden Markov Models

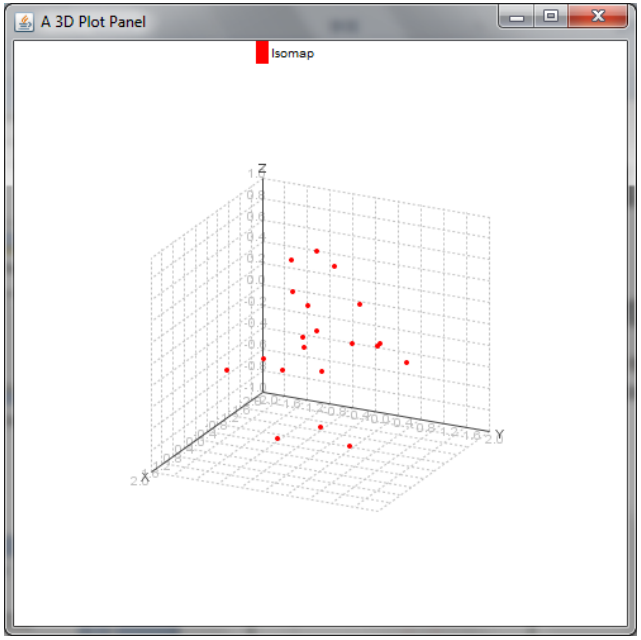
```
HMM HMM = new HMM(numStates, numObservations, epsilon, maxIter);
HMM.feedData(Os);
HMM.feedLabels(Qs); // If not given, random initialization will be used
HMM.train();
HMM.evaluate(O);
HMM.predict(O);

CRF CRF = new CRF(epsilon);
CRF.feedData(Fs);
CRF.feedLabels(Ys);
CRF.train();
CRF.predict(F); // Also compute the probability via Viterbi algorithm
```

Dimensionality Reduction

- jml.subspace
  - PCA, KernelPCA, MDS, Isomap, LLE

```
int n = 20; int p = 10; RealMatrix X = rand(p, n);
int K = 6; // number of nearest neighbors
int r = 3; // reduced dim
RealMatrix R = Isomap.run(X, K, r);
```





Matrix Recovery

- jml.recovery
  - Robust PCA, Matrix Completion

```
RealMatrix D = ... // Observation matrix
double lambda = 1.0;

RobustPCA robustPCA = new RobustPCA(lambda);
robustPCA.feedData(D);
robustPCA.run();

// Low-rank recovery of D
RealMatrix A_hat = robustPCA.GetLowRankEstimation();
// Error matrix between D and A
RealMatrix E_hat = robustPCA.GetErrorMatrix();
```

General-Purpose Optimization

- jml.optimization
  - L-BFGS
  - Proj L-BFGS (Simplex, Box, or Nonnegative)
  - General Quadratic Programming (of course General Linear Programming)
  - Nonlinear Conjugate Gradient
  - Primal-Dual Interior-Point methods
  - Accelerated Proximal Gradient (of course Accelerated Gradient Descent)

Others

- jml.regression
  - LASSO
- jml.kernel
  - 'linear' | 'poly' | 'rbf' | 'cosine'
- jml.manifold: semi-supervised/unsupervised
  - Adjacency graph (directed or undirected)
  - Laplacian regularization
  - Local learning regularization
- jml.random: Probability distributions
  - Multivariate Gaussian Distribution

Combine TextProcessor and JML

- For text mining, our input are text corpora, not sparse matrices.

```
Options options = new Options();
options.workSpacePath = "....";
options.mergedFileName = "";
options.dataDirName = "...";
options.ext = "txt";
TextProcessor textProcessor = new TextProcessor(options);
textProcessor.buildDocStringArray();
textProcessor.processStringArray(textProcessor.docStringArray);
/* Generate the dictionary, DocTermCountArray, GroundTruth, LabelIDMap,
LDA format Input (file or stringArr), LIBSVM format input (file or stringArr) */

RealMatrix X =
Data.docTermCountArray2Matrix(textProcessor.docTermCountArray);
int[] labelIDs = textProcessor.getLabelIDs();
```

Future Work

- Multi-task learning
- Multi-label learning
- Multi-instance learning
- Feature selection
- Deep learning in Java (Parallel)