

LAML: Linear Algebra and Machine Learning

- A stand-alone Java library for linear algebra and machine learning

Mingjie Qian

Department of Computer Science

University of Illinois at Urbana-Champaign

July 13th, 2015

Features

- Standalone, pure Java, and cross-platform
- Built-in Linear Algebra (LA) library
- Fast 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/lamal/> since 12/07/2013

Statistics: 1400+ downloads from 59 countries (regions)

LAML V.S. Liblinear

- heart_scale with $C = 1.0$ and $\text{eps} = 1\text{e-}2$
- LAML: 0.04s
- Liblinear: 0.06s
- Intel(R) Core(TM) i7 CPU M620 @ 2.67GHz with 4.00GB memory and 64-bit Windows 7

LAML V.S. JML

- 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- In short, JML has been replaced by LAML.

LAML V.S. LibLAML

- libLAML is a stand-alone pure C++ static library for linear algebra and machine learning
- libLAML is at least 4 times faster than LAML
- libLAML is not cross-platform

Built-in Linear Algebra Packages

- `la.decomposition`
 - EVD, LU, QR, and SVD
- `la.io`
 - IO functions
- `la.matrix`
 - `DenseMatrix` and `SparseMatrix`
- `la.vector`
 - `DenseVector` and `SparseVector`

Built-in Machine Learning Packages

- ml.clustering
- ml.regression
- ml.classification
- ml.topics
- ml.optimization
- ml.sequence
- ml.subspace
- ml.recommendation
- ...

Data Interface

- la.io: 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

Decomposition

Matrix A = hilb(m, n);

Matrix[] VD = EigenValueDecomposition.decompose(A);

Matrix[] LUP = LUDecomposition.decompose(A);

Matrix[] QRP = QRDecomposition.decompose(A);

Matrix[] USV = SingularValueDecomposition.decompose(A);

Matrix

- Matrix interface:
 - mtimes, times, plus, minus, transpose, operate, getEntry, setEntry, clear, copy, ...
- DenseMatrix implements Matrix, Serializable
- SparseMatrix implements Matrix, Serializable
 - Standard compressed sparse column (CSC) and compressed sparse row (CSR)
 - full control of the interior arrays in sparse matrices

Vector

- Vector interface:
 - times, plus, minus, operate, get, set, getDim, clear, and copy
- DenseVector implements Vector, Serializable
- SparseVector implements Vector, Serializable
 - full control of the interior arrays in sparse vectors

Matlab

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

Classification

- ml.classification
 - MCSVM, Regularized Logistic Regression, MaxEnt, AdaBoost

```
Classifier linearMCSVM = new LinearMCSVM(C, eps);  
linearMCSVM.feedData(trainData); // double[][] or Matrix  
linearMCSVM.feedLabels(labels); // double[][], int[], or Matrix  
linearMCSVM.train();  
linearMCSVM.predict(testData); // Predicted labels (int[])
```

```
Classifier logReg = new LogisticRegression(regularizationType, lambda);  
logReg.feedData(data);  
logReg.feedLabels(labels);  
logReg.train();  
logReg.predict(testData);
```

Regression

- ml.regression
 - LASSO and Linear Regression

Regression LASSO = new LASSO(options);
LASSO.feedData(data);
LASSO.feedDependentVariables(depVars);
LASSO.train();
Matrix Yt = LASSO.predict(testData);

Regression LR = new LinearRegression(options);
LR.feedData(data);
LR.feedDependentVariables(depVars);
LR.train();
Matrix Yt = LR.predict(testData);

Clustering

- ml.clustering
 - KMeans, NMFs, Spectral Clustering

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

Topic Modeling

- ml.topics
 - LDA // PLSA \Leftrightarrow NMF
 - Read corpora with multiple format:
 - int[][], DocTermCountArray, Matrix, 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

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

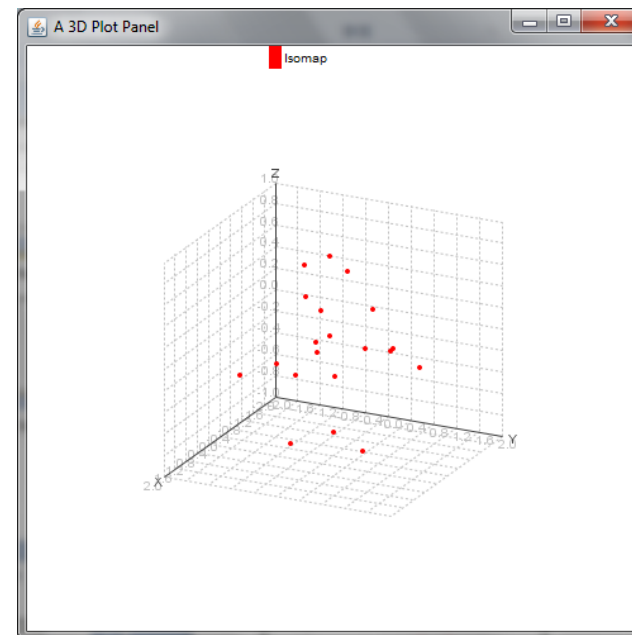
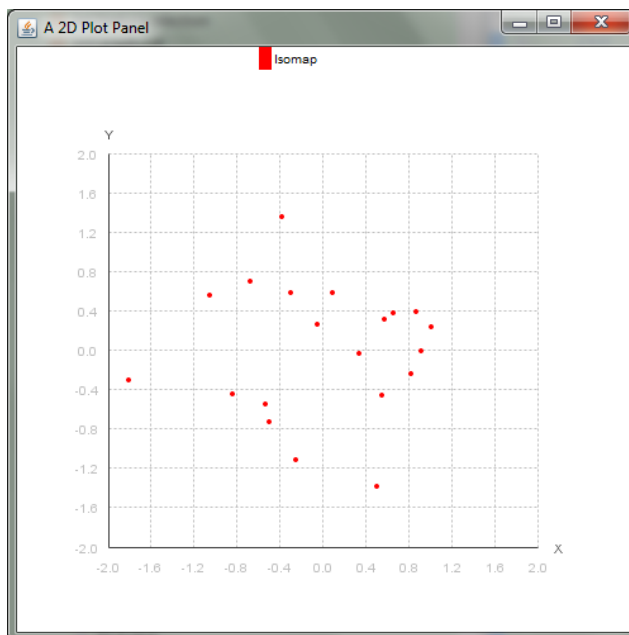
```
HMM HMM = new HMM(numStates, numObservations, epsilon, maxIter);  
HMM.feedData(Os);  
HMM.feedLabels(Qs); // If not given, random initialization will be used  
HMM.train();  
HMM.evaluate(O);  
HMM.predict(O);
```

```
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

- ml.subspace
 - PCA, KernelPCA, MDS, Isomap, and LLE

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



Matrix Recovery

- ml.recovery
 - Robust PCA and Matrix Completion

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

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

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

Recommendation

- ml.recommendation
 - Factorization Machines (FM), STructured Sparse Regression (STSR)

```
// Training
```

```
int idxStart = 0;
```

```
FM.feedTrainingData(trainFilePath, idxStart);
```

```
FM.allocateResource(k);
```

```
FM.feedParams(maxIter, lambda);
```

```
FM.initialize();
```

```
FM.train();
```

```
// Prediction
```

```
DataSet testData = FM.loadData(testFilePath, idxStart);
```

```
double[] Y_pred = FM.predict(testData.X);
```

General-Purpose Optimization

- ml.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
 - Accelerated Gradient Descent

Utils

- `ml.utils`
 - `ArrayOperator`: commonly used double array operations
 - `InPlaceOperator`: a set of in-place functions without memory allocation
 - `Printer`: print matrices, vectors, arrays, and formatted strings
 - `Times`: `tic` and `toc`
 - ...

Others

- `ml.graph`
 - Minimum Spanning Tree and Shortest Path
- `ml.kernel`
 - 'linear' | 'poly' | 'rbf' | 'cosine'
- `ml.manifold`: semi-supervised/unsupervised
 - Adjacency graph (directed or undirected)
 - Laplacian regularization
 - Local learning regularization
- `ml.random`: Probability distributions
 - Multivariate Gaussian Distribution

DataSet and Data

- `la.io.DataSet` and `la.io.Data`
 - `readDataSet(ArrayList<String> feaArray)`
 - `readDataSetFromFile(String filePath)`
 - `writeDataSet(Matrix X, int[] Y, String filePath)`

// Read a data set from a LIBSVM formatted string array or file

`DataSet.IdxStart = 0;`

`trainDataSet = Data.readDataSetFromFile(featureFilePath);`

`Matrix X = trainDataSet.X; // n x p`

`int[] labels = trainDataSet.Y; // Used for classification or recommendation`

`Data.IdxStart = 0;`

`trainDataSet = Data.readDataSetFromFile(featureFilePath);`

`Matrix X = trainDataSet.X; // n x p`

`double[] labels = trainDataSet.Y; // Used for regression`

Combine TextProcessor and LAML

- 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) */

Matrix X = IO.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)