Better Missing Value Handling

21..03.2016

Maniteja Nandana
India

Feedback on Github or mailing list welcome!

Proposed projects

I have looked at various stalled PRs that were widely discussed upon and perceived as a useful and viable addition to the library. I would sincerely appreciate if anyone has other issues and PRs to add related to the above topics or which I missed that are of prime utility to the users

It is used a meta estimator to train BernoulliNB or MultinomialNB by semi supervised learning

Matrix factorization with missing values **IMP**

Issues and PRs: #4237 #2387

Goal

The PR adds Matrix Factorization support. The problem is to factorize a matrix with missing values into product of 2 matrices of lower rank in order to restore missing values (perform imputation).using ALS Alternating Least squares and ALS1 algorithms

Implementation

new transformers **sklearn.decomposition.MatrixFactorization** and **sklearn.preprocessing.FactorizationImputer.**

- *FactorizationImputer* is implemented similar to Imputer in case of row-statistics
- The algorithms used are :
    - `sgd` and `sgd_adagrad` are variants of stochastic gradient descent, the later one uses AdaGrad learning rate adjustment.
    - `ALS` and `ALS1` are variants of Alternating Least Squares

Discussion

The following is the abstract of the paper :

- Alternating least squares (ALS) is a powerful matrix factorization (MF) algorithm for both explicit and implicit feedback based recommender systems.
- Increasing the number of latent factors (denoted by K) boosts the prediction accuracy of MF based recommender systems, including ALS as well
- The price of the better accuracy is paid by the increased running time: the running time of the original version of ALS is proportional to K^3.
- The paper presents novel and fast ALS variants both for the implicit and explicit feedback datasets, which offers better trade-off between running time and accuracy. Due to the significantly lower computational complexity of the algorithm - linear in terms of K - the model being generated under the same amount of time is more accurate, since the faster training enables to build model with more latent factors.

Add KNN strategy for imputation **IMP**

Issues and PRs: #4844

Goal

According to Tian Wang , this algorithm is to find all rows with full features as complete set, and impute the missing features in a row by taking the mean of those features among its K-nearest neighbors in complete set, based on Hastie

Implementation

It is currently using the sum of squared differences, and the goal is to implement the Hastie's approach. Now it can impute with setting 'strategy' to be KNN, but KNN would not be a 'strategy', but as a neighbor selection which other imputation methods (eg. mean, median,..) would use.

Similar package for R: Impute Similar function for Matlab:Knnimpute

Discussion

There seems to be a need for rework of API here but does look like a good addition if there can be some consensus reached there.

- The option of non-euclidean metrics is provided by other implementations. Weighted means should be possible, etc. selecting a feature subspace in which objects are compared, rather than calculating distance over all non-missing features per sample.
- We find the k nearest neighbors using a Euclidean metric, confined to the columns for which that gene is NOT missing. Each candidate neighbor might be missing some of the coordinates used to calculate the distance. In this case we average the distance from the non-missing coordinates.
- Further, one might consider strategy, in terms of mean/median/custom, as orthogonal to knn vs whole dataset. The MatLab package allows median to be used (which may not make a whole lot of sense with euclidean distance, but is easy to justify otherwise). I suspect that if we're not doing the above, we should not be adding strategy="knn" but allowing the user to choose n_neighbors (defaulting to None meaning all) as orthogonal to strategy.

Adding input from Joel Nothman here :

I think the brief summary is something along the lines of: this ideally requires a little research into how others have done KNN imputation (particularly: how are missing values in other columns dealt with? which options of NN need to be configurable?), and whether therefore this deserves its own estimator, or should rather be integrated into the current Imputer.

References: FancyImpute (Thanks to Alex for the permission to reuse ideas) MATLAB KNN

OneHotEncoder for imputed features in Imputer

Issues and PRs: #6556 #6607

Goal

output dummy one-hot encoder features for imputer to specify if the feature value is imputed or not

Added points by Giorgio Patrini

The motivation is the following: imputation implicitly discards the information that certain examples (rows) had missing values originally. Therefore, before imputation, we add a column for each feature that has at least one missing a value. The additional column is 0 everywhere, except that is it 1 when the corresponding value is missing.

.

Implementation

If axis=0 , shape of output is **(n_samples x n_features_with_no_all_missing + n_features_with_partial_missing)**

If axis=1, shape of output is **(n_samples x n_features + n_features_with_missing)**

Added metrics support for multiclass-multioutput classification **MOD**

Issues and PRs: #3681

At present, no metric supports the multioutput-multiclass classification task. This PR enables multiclass multioutput support for accuracy_score and zero_one loss. This can be completed and also the possibility of adding it to the other classification metrics can be tried. I have done some initial changes at here I have faced this need while working on MultiOutputClassifier at #6127

Time line

- Community Bonding Period (April 22, 2016 - May 22, 2016)
  - Continue my current pull requests, help in the contribution process to scikit learn by playing my part in reviewing the pull requests
  - Binary indicator for imputed features in Imputer
- Week 1-3(May 23, 2016 - June 13, 2016)
  - Work on the stalled Matrix Factorisation code in #4237 #2387
  - Write tests for the presently implemented ALS and ALS1 algorithms
  - Narrative documentation and examples for the Matrix factorisation
- Week 4-6(June 14, 2016 - July 4, 2016)
  - KNN strategy for imputation
  - Decide on the API design regarding the new imputer methods
  - The tests for KNN Imputer and possibly extend for different weights for neighbours and multiple metrics like mean, median for various types of distances
- Week 7-9(July 5, 2016 - July 25, 2016)

    - Compare the imputation methods and matrix factorisation methods for handling missing values. Something on the lines of this example
- Week 10-12(July 26, 2016 - August 15, 2016)
    - Work on multi class - multi output support for some metrics in classification and regression
- Week 13(August 16, 2016 - August 23, 2016)
    - Finish up tests and documentation for the work done and get it ready for
    - final review and merging later.

The above timeline is tentative and tried not to overreach the scope of the contributions. Depending on the progress of the PRs, the pace can be accordingly adjusted. I shall devote at least the minimum of 40 hours per week.

Commitments

During the period till August, I have no other commitments during which I will spend extra time for completing the major parts of the project. The schedule of my job is set to begin from around mid August, during which I will still work to the fullest but might not be available for the whole of the day.

Contribution

Merged

#6127[MRG+2] MultiOutputClassifier Thanks to HugoBowne and others for letting me work on this and also to all the reviewers for their patience

#6611 [MRG] DOC: Correct some errors in chebyshev distance

#6481 DOC: Clarify the scoring argument in Logistic Regression

#6232 MAINT: Print info message for fetch_20newsgroups

#6104 [MRG+1] Enable pandas input to log_loss

Open

#6607 [WIP] Add add_missing_indicator option to show missing values in the output

#6445 [MRG] Add get_feature_names to PCA

#6217 [MRG] ENH: Add sample_weight to median_absolute_error

#6141 MAINT: Return self for fit in Spectral Biclustering and CoClustering

#6516 DOC: Link functions to equivalent classes #5984 DOC: Link to LDA in example

Closed

#6337 MAINT: Fix errors in tests for Unvariate Selection

#6216 ENH: Add feature_names_ property to PolynomialFeatures

#6199 [MRG+1] DOC: Add Raises Section #6199

Student Information:

Name: Maniteja Nandana

Email: maniteja.modesty067@gmail.com

Telephone: +91 9912304125

Time Zone: GMT/UTC +05:30 IST

Blog: http://www.inspiremaniteja.wordpress.com ;
https://sites.google.com/site/inspiremaniteja/

LinkedIn : https://www.linkedin.com/in/modestymaniteja

University: Birla Institute of Technology and Sciences, India

Degree: Bachelors of Engineering

Major: Computer Science

Current Year: Senior (4th year)

Expected Graduation date: 2016

Degree: Bachelor

I am Maniteja, a final year undergraduate studying Computer Science at BITS Pilani, India. I have started Machine Learning from college curriculum and from online courses.

In addition to my research experience, I have also interned at Bhabha Atomic Research Center, India; Amazon, India and currently am a project trainee at Center for Artificial Intelligence, India. I have implemented basic algorithms like decision tree, vector space retrieval model and candidate elimination, which can be found in my github repositories.

I started contributing to open source since last year and to scikit-learn from December, 2015. I have used the library for some of my college projects mentioned below.

Scikit-learn has been awesome community to work with and have learnt a lot about machine learning that ever before from contributing to this community. I would be grateful if provided the opportunity to contribute my skills to the project and help in the development of the library. Having worked with the open source, I have familiarity with the contribution guidelines, coding styles and review process. I would strive to deliver to the best of my abilities and will take feedback from the experienced people to ensure the implementations done would be in the best interests of the users.

Experience

1. Summer Internship at Bhabha Atomic Research Centre - 2014

   Worked on network security model using Django, RESTful API and PostGRESQL

2. Summer Internship at Amazon India - 2015

   Worked on online interviewing system to automate the process of automating the test cases generated for different data structures for competitive programming questions

3. Semester internship at Center for Artificial Intelligence and Robotics.

   Worked on Multi agent robotics using JADE and Semantic service oriented architecture in JENA

4. Projects in college:
   a. Prediction of defect occurrence patterns in open source software using bug prediction models
   b. Classification of open source contributors based on the recurring patterns in their commit history
   c. Development of a pipelined multiprocessor based on the x8086 model in SystemC, a high level description language in Cpp.
5. I have been contributing to Scipy since 2015 and recently been contributing to scikit learn.
6. I have also completed many courses related to ML, AI and NLP on Coursera, Udacity and EdX, including Andrew Ng course on Machine learning, Columbia university course on NLP, Sebastian Thrun course on Introduction to AI robotics and University of Washington course on Regression.