Machine Learning Final Project Report

Team Name: Zero
Kuan-Ting Yu (r99922070) and Yu-Chun Huang (d98922047)
January 13, 2012

Abstract

The final project of this machine learning class is a challenging multi-label prediction problem with missing data. We use polynomial surface regression for pairwise-feature fitting, and then use the features with least fitting error to predict missing data. Then we did 5-fold cross-validation against 4 learners and applied uniform blending on the learners to achieve better performance. A brief comparison of the learners and recommendations for different application requirements are also discussed in this survey report.

1. Dealing with missing data

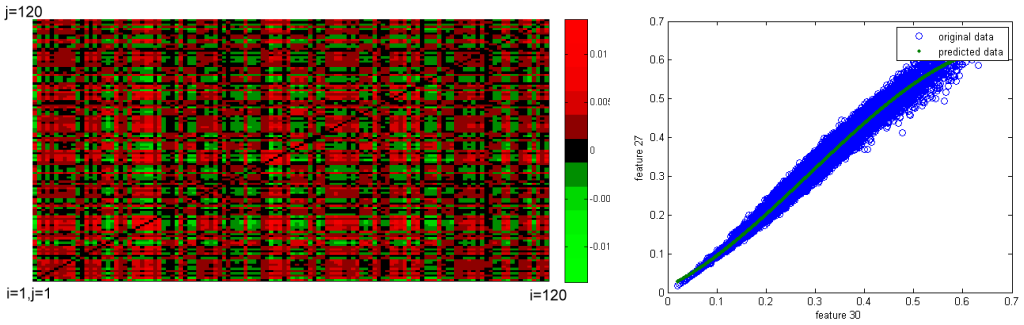
We designed three algorithms for recovering missing data, 1) sorting by pairwise correlation then use polynomial fitting for interpolation, 2) sorting by pairwise fitting error then use polynomial fitting for interpolation, 3) sorting by fitting error then use a polynomial surface to fit the missing data from two existing features.

1.1. Sorting by correlation - polynomial fitting

Step1. Calculate the correlation among every pairs of feature as shown in Fig 1 (a).

Step2. For each missing feature *i*, we rank the candidate feature *j* for recovering by the absolute correlation value because higher absolute correlation implies the pair have stronger relation rather than being independent.

Step3. Given the missing feature *i* and feature *j* for recovering, we use second order polynomial fitting to interpolate the missing value. The fitted curve is shown in Fig 1 (b). The reason to pick second order polynomial is that most distribution when we visualize the data are not linear and monotonically increasing or decreasing. The second reason is that higher order polynomial may is more possibly incur overfitting and from experiments we observe no much improvement as the order becomes greater than two.



(a) (b)

Fig 1. (a) The correlation value presented using heat map, the origin is at the bottom-left corner. We intentionally ignore the value of self-correlation because we need another feature to recover the missing feature. (b) Using feature 30 to fit feature 27.

1.2. Sorting by fitting error - polynomial fitting

Although correlation is one heuristic that estimate which two pair has better fitting result, a more direct measurement would be the fitting error. The fitting error correspond to the training error taught in class. We can further use cross validation to further estimate the testing error. As a result we gain a significant improvement in testing performance of the entire multilabel problem. The error we used is root mean squared error (RMSE).

1.3. Sorting by fitting error - surface fitting

The method above is good enough for certain feature but not for all features. For example, the best curve to fit feature 60 is shown in Fig 2. We can observe a great error on the right hand side. Therefore, we propose to use two features to fit the target missing feature in order to lower the fitting error. We can achieve a better fitting error of 0.052 against the error of 0.08 obtained from section 1.2. We choose the two features for each target by fitting error.

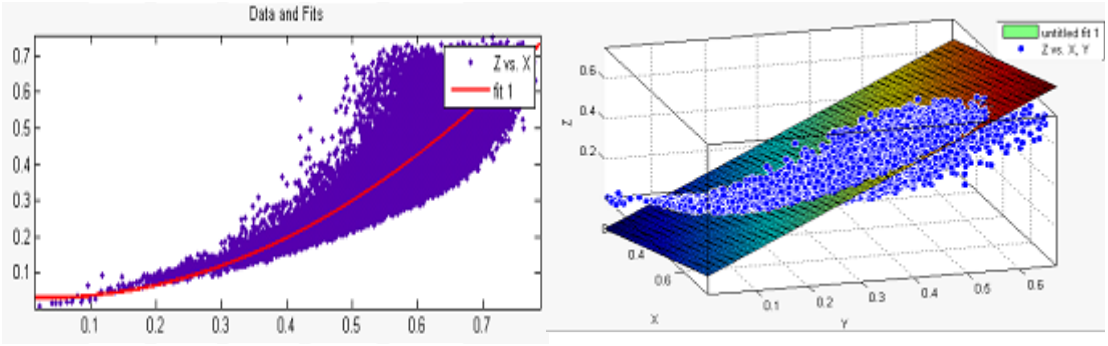


Fig 2. (a) Curve fitting result of feature 60 against feature 56. (b) Surface fitting result.

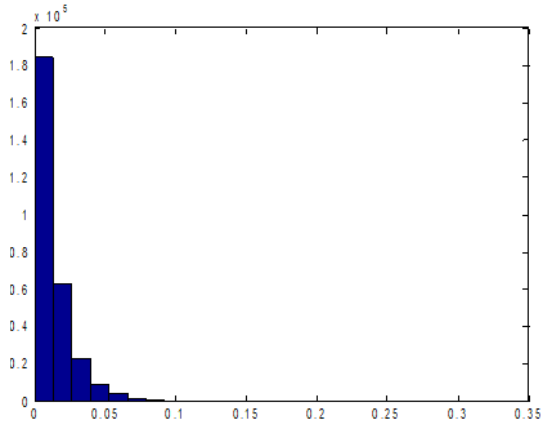


Fig 3. (a) We added some missing value to the testing data to test our recovering mechanism. This shows the distribution of L1 error. The error is mostly bounded by 0.1.

2. Choosing learners

In order to find a good multi-label predictor for this particular data set, we applied 5-fold cross-validation against 4 learners (MLkNN, J48, SVM, and Random Forest[1]) with various parameter configurations. Note that, all the parameters not listed here are left using implementation defaults. After the evaluation is done, uniform blending is applied to further improve the performance.

[1] We applied two meta-algorithms (RAkEL and Power Labelset[3]) to J48 and Random Forest.

The learner implementations we use for this project are Mulan[1] and LIBSVM[2]. The source code of this project can be found at http://code.google.com/p/ml2011-final-project/.

2.1. MLkNN 5-fold cross-validation

Table 1. MLkNN 5-fold cross-validation result

K	Smoothness	Hamming loss	F1
8	0.7	0.0328±0.0003	0.4844±0.0018
8	1.0	0.0328±0.0003	0.4844±0.0018
8	1.3	0.0328±0.0003	0.4844±0.0018
12	0.7	0.0326±0.0003	0.4829±0.0023
12	1.0	0.0326±0.0003	0.4828±0.0023
12	1.3	0.0326±0.0003	0.4828±0.0023
16	0.7	0.0325±0.0003	0.4849±0.0039
16	1.0	0.0325±0.0003	0.4849±0.0039
16	1.3	0.0325±0.0003	0.4849±0.0039
22	1.0	0.0324±0.0003	0.4904±0.0046
26	1.0	0.0323±0.0003	0.4909±0.0030
30	1.0	0.0323±0.0003	0.4895±0.0025
34*	1.0	0.0323±0.0003	0.4922±0.0034

* Considered the best choice and used by our team.

2.2. J48 5-fold cross-validation

Table 2. J48 5-fold cross-validation result

Confidence interval	Hamming loss	F1
0.25	0.0321±0.0002	0.5572±0.0036
0.30*	0.0322±0.0002	0.5569±0.0036
0.40	0.0322±0.0002	0.5569±0.0037
0.50	0.0323±0.0002	0.5569±0.0038

* Considered the best choice and used by our team.

2.3. Random Forest 5-fold cross-validation

Table 3. Random Forest 5-fold cross-validation result

Number of trees	Number of features	Hamming loss	F1
10	3	0.0287±0.0002	0.5659±0.0029
10	5	0.0285±0.0003	0.5716±0.0017
10	8	0.0261±0.0001	0.6192±0.0021
10	12	0.0262±0.0001	0.6194±0.0040
10	16	0.0261±0.0002	0.6201±0.0023
10	20	0.0261±0.0002	0.6216±0.0037
20	8	0.0255±0.0002	0.6264±0.0034
20	16	0.0254±0.0001	0.6296±0.0028
20	24	0.0253±0.0002	0.6317±0.0029
20*	32	0.0253±0.0001	0.6320±0.0024
20	40	0.0254±0.0002	0.6316±0.0029
30	8	0.0255±0.0002	0.6259±0.0036
30	16	0.0253±0.0002	0.6292±0.0043
30	24	0.0253±0.0002	0.6304±0.0031
30	32	0.0253±0.0002	0.6318±0.0033

* Considered the best choice and used by our team.

2.4. SVM 5-fold cross-validation

We tried two meta-algorithms to transform the multilabel problem into single label classification problem, 1) label combination 2) binary transformation. For label combination method, we obtain 3436 classes from all possible label combination in the training dataset. However, the best result we obtained is (Hamming loss: 3.7092 F1: 0.5284515637) using surface fitting data, which is very limited. The reason we think is because after dividing all the data into 3436 classes, there are not much examples left for each class, and the number of examples of each class is unbalanced.

For binary transformation, we obtain generally better result than using label combination but the training time is rather long, because every label requires both training and testing. On the other hand, the default of libsvm outputs accuracy, which is correspond to hamming loss. To optimize the parameter for F1 score, we modify the output to be F1 score so that the “grid.py” program for choosing SVM parameters that will maximize the F1 score.

Table 4. SVM 5-fold cross-validation result

	Recover method	Hamming loss	F1
SVM Binary	Surface fitting	3.047	0.5734046426
SVM Binary	ErrSort CurveFitting	3.2604	0.5163759345
SVM Binary	CorrSort CurveFitting	3.598	0.4434131551
SVM Label Combine	Surface fitting	3.7092	0.5284515637
SVM Label Combine	ErrSort CurveFitting	3.7886	0.5387293934
SVM Label Combine	CorrSort CurveFitting	4.2394	0.4077892325

2.5 Blending

We tried 3 methods for blending, 1) Pocket PLA, 2) Linear Regression, and 3) Uniform voting. After several experiments, we found that uniform blending generates best result and is much simpler. Hence we use uniform blending to produce the final prediction for the test data set.

We also tried several prediction combinations of the 4 learners to do uniform blending. Interestingly, the result shows that mixing MLkNN and J48 altogether with Hamming-optimized SVM, F1-optimized SVM and Random Forest does not help. We believe the reason why MLkNN and J48 does not make the prediction better is that Random Forest and the 2 SVMs are already too powerful, and the votes from MLkNN and J48 are not accurate. Therefore, blending Hamming-optimized SVM, F1-optimized SVM, and Random Forest is our final choice, and it does help us win the 3rd place in turns of Hamming loss in the final competition (Hamming loss: 2.9942; F1: 0.5802797948).

3. Discussion

In this section, a brief comparison of the 4 learners, and recommendation for different application requirements are discussed.

Because our team's best Hamming loss and F1 are predicted by our blending solution, **we recommend this blending solution for both Hamming loss and F1**. The pros and cons are also stated in Table 6 (application requirement: "Hamming loss is important" and "F1 is important").

Table 5. Comparison between the 4 learners

	Efficiency	Scalability	Popularity	Interpret-ability
MLkNN	High	High	Medium	Medium
J48	Medium	High	Medium	High
Random Forest	Medium	Medium	High	High
SVM	Low	Medium	High	Low

Table 6. Recommendations for different application requirements

Application Requirement	Recommendation*	Pros and cons**
Computation time is critical	MLkNN	**Pros:** - Fast (train: 16.74 minutes; predict: 14.64 minutes). **Cons:** - Comparing to other learners evaluated by us, the out-of-sample error is relatively high for both Hamming loss and F1.
Out-of-sample error is critical (Hamming loss & F1)	Uniform blending of Random Forest and SVM	**Pros:** - Out-of-sample error is low, especially for Hamming loss measure (Hamming loss: 0.029942; F1: 0.5802797948). **Cons:** - Computation intensive (training time is more than 24 hours). - Hard to interpret how the prediction is done. Decision maker should have "faith" on the learner.
Hamming loss is critical	Same as above	Same as above
F1 is critical	Same as above	Same as above

* Learner configurations are based on the cross-validation results discussed in section 2.

** Computation time is machine-dependent.

4. Credits

Kuan-ting Yu (r99922070)
- Missing data pre-processing;
- SVM prediction;
- Survey report.

Yu-chun Huang (d98922047)
- MLkNN, J48, and Random Forest prediction;
- Prediction blending;
- Survey report.

References

[1] Mulan: A Java library for multi-label learning, http://mulan.sourceforge.net/.

[2] LIBSVM -- A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[3] Multi-Label Classification: An Overview, International Journal of Data Warehousing & Mining, 3(3), 1-13, July-September 2007.