Ben Tankus

HW 3

05-15-21

1 Understanding the Evaluation Metric

This kaggle contest uses “Root Mean Squared Log Error” (RMSLE) rather than the standard “Root Mean Squared Error” (RMSE).

1. What exactly is this RMSLE error? (write the mathematical definition).

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where n is the total number of observations in the (public/private) data set, pi is your prediction of target, and ai is the actual target for i . log(x) is the natural logarithm of x ($\log_e(x)$) .

This is a regression methodology that is used when values have a large range, or exponential pattern.

2. What’s the difference between RMSLE and RMSE?

RMSLE has a lower likelihood of underestimation than RMSE as the algorythim incurs a larger penalty for underestimation than overestimation. RMSE also has only linear error, whereas RMSLE can have non-linear error. Mathematically, they are very similar and only differ by RMSLE taking the log of both RMSE terms.

3. Why does this contest adopt RMSLE rather than RMSE?

A log is used for housing price data as there is a very wide spread in housing price that is difficult to model. It is much easier to model a simplified linear model, and taking the log of the data allows us to do this.

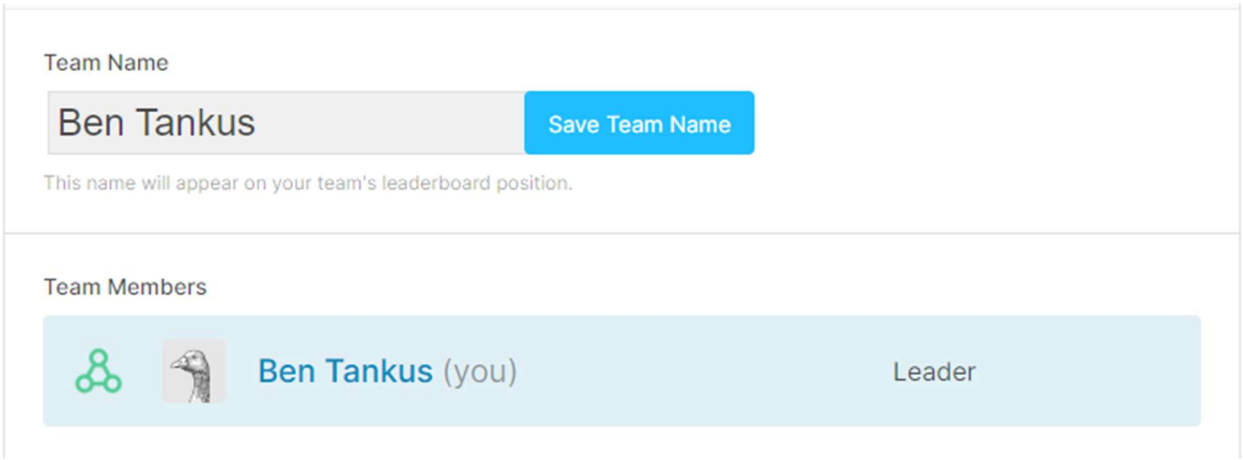
4. One of our TAs got an RMSLE score of 0.11 and was ranked 28 in Spring 2018. What does this 0.11 mean intuitively, in terms of housing price prediction error?

This 0.11 means their housing price prediction model averaged e^0.11 times (1.1163) the actual price of the house.

5. What are your RMSLE error and ranking if you just submit sample_submission.csv?

I get a score of 0.406 if I just submit sample_submission.csv. This puts me in 9384th place where it looks like many others have submitted just the sample_submission.

6. What is your "Team name" on kaggle (note this HW should be done individually)?



For the rest of this HW, it is highly recommended that you take the logarithm for the y field (SalePrice) before doing any experiment, so that you reduce the problem back to RMSE. However, do not forget to exponentiate it back before submitting to kaggle. In other words, you train your regression systems to predict the logarithm of housing prices, log(y), but you submit your predictions in the original prices, not the ones after taking log.

2 Naive data processing: binarizing all fields

Take a look at the data. Each training example contains 81 fields in total: the unique Id field, 79 input fields, and one output field SalePrice. Like in HW1 data, there are two types of input fields: categorical, such as SaleCondition and GarageType, and numerical, such as LotArea and YrSold. Note that some fields might be mixed categorical/numerical, such as LotFrontage and GarageYrBlt: as you know, not all homes have lot frontages (the length of the front side of the lot facing a street – some lots are not facing any street), and not all homes have garages, thus both fields could have occasional NA values (for "N/A" or "not applicable"). Following HW1, the simplest thing to do is to binarize all fields. You can use your own Python implementation, or pandas.get_dummies(), but I think it is best to use your own implementation for its flexibility and for the rest of this HW.

1. How many features do you get? (Hint: around 7230). You can use this command to see the total number of binary features: for i in `seq 2 80`; do cat my_train.csv | cut -f $i -d ',' | sort | uniq | wc -l; done | \ awk '{s+=$1-1} END {print s}'
   Let me explain a little bit: the first line loops over each column (except the first which is Id and the last which is the output), and print the values for that column (cut -f $i), sort and make unique, and count how many unique values there are for that column (wc -l). The second line uses awk to sum it up; here $1 denotes the first column, and -1 for excluding the header row, and finally print the sum.

Doing a blind binarization of all fields (excluding sales) I get 7227 features.

2. How many features are there for each field?

```
MSSubClass has 15 unique features
MSZoning has 5 unique features
LotFrontage has 369 unique features
LotArea has 1073 unique features
Street has 2 unique features
Alley has 3 unique features
LotShape has 4 unique features
LandContour has 4 unique features
Utilities has 2 unique features
LotConfig has 5 unique features
LandSlope has 3 unique features
Neighborhood has 25 unique features
Condition1 has 9 unique features
Condition2 has 8 unique features
BldgType has 5 unique features
HouseStyle has 8 unique features
OverallQual has 10 unique features
OverallCond has 9 unique features
YearBuilt has 112 unique features
YearRemodAdd has 61 unique features
RoofStyle has 6 unique features
RoofMatl has 8 unique features
Exterior1st has 15 unique features
Exterior2nd has 16 unique features
```

MasVnrType has 5 unique features
MasVnrArea has 335 unique features
ExterQual has 4 unique features
ExterCond has 5 unique features
Foundation has 6 unique features
BsmtQual has 5 unique features
BsmtCond has 5 unique features
BsmtExposure has 5 unique features
BsmtFinType1 has 7 unique features
BsmtFinSF1 has 637 unique features
BsmtFinType2 has 7 unique features
BsmtFinSF2 has 144 unique features
BsmtUnfSF has 780 unique features
TotalBsmtSF has 721 unique features
Heating has 6 unique features
HeatingQC has 5 unique features
CentralAir has 2 unique features
Electrical has 6 unique features
1stFlrSF has 753 unique features
2ndFlrSF has 417 unique features
LowQualFinSF has 24 unique features
GrLivArea has 861 unique features
BsmtFullBath has 4 unique features
BsmtHalfBath has 3 unique features
FullBath has 4 unique features
HalfBath has 3 unique features
BedroomAbvGr has 8 unique features
KitchenAbvGr has 4 unique features
KitchenQual has 4 unique features
TotRmsAbvGrd has 12 unique features
Functional has 7 unique features
Fireplaces has 4 unique features
FireplaceQu has 6 unique features
GarageType has 7 unique features
GarageYrBlt has 178 unique features
GarageFinish has 4 unique features
GarageCars has 5 unique features
GarageArea has 441 unique features
GarageQual has 6 unique features
GarageCond has 6 unique features
PavedDrive has 3 unique features
WoodDeckSF has 274 unique features
OpenPorchSF has 202 unique features
EnclosedPorch has 120 unique features
3SsnPorch has 20 unique features
ScreenPorch has 76 unique features
PoolArea has 8 unique features
PoolQC has 4 unique features
Fence has 5 unique features
MiscFeature has 5 unique features
MiscVal has 21 unique features
MoSold has 12 unique features
YrSold has 5 unique features
SaleType has 9 unique features

SaleCondition has 6 unique features

3. Do you need to augment the space (add bias dimension) like in HW1, or does your regression tool automatically does it for you?

Regression does this automatically using the intercept of the model.

4. Train linear regression using sklearn.linear_model.LinearRegression or np.polyfit on my_train.csv and test on my_dev.csv. What's your root mean squared log error (RMSLE)? (Hint: should be around 0.15 or 0.16).

My mean square log error is 0.15630755195257634.

5. What are your top 10 most positive and top 10 most negative features? Do they make sense?

```
Positive Index: 45 Positive Weight: 0.1502200765778792 Positive Feature: (45, '1710') - GRLivArea
Neg Index: 551 Neg Weight: -0.195700683334053 Neg Feature: (1, 'C (all)') - MSZoning

Positive Index: 299 Positive Weight: 0.1447142007193873 Positive Feature: (16, '9') - OverallQual
Neg Index: 35 Neg Weight: -0.18695065472452088 Neg Feature: (35, '0') - BsmtFinSF2

Positive Index: 8 Positive Weight: 0.128341444476475076 Positive Feature: (8, 'AllPub') - Utilitie
s
Neg Index: 0 Neg Weight: -0.16406225528777732 Neg Feature: (0, '60') - MSSubClass

Positive Index: 311 Positive Weight: 0.12749464883943112 Positive Feature: (48, '3') - BsmtFullBa
th
Neg Index: 37 Neg Weight: -0.14604398612052277 Neg Feature: (37, '856') - TotalBsmtSF

Positive Index: 168 Positive Weight: 0.12378709421335773 Positive Feature: (16, '8') - OverallQua
l
Neg Index: 1803 Neg Weight: -0.12858627347290982 Neg Feature: (45, '968') - GrLivArea

Positive Index: 813 Positive Weight: 0.1205229242608985 Positive Feature: (11, 'StoneBr') - Neigh
borhood
Neg Index: 3469 Neg Weight: -0.12769029207771548 Neg Feature: (67, '236') - EnclosedPorch

Positive Index: 10 Positive Weight: 0.11893708405244831 Positive Feature: (10, 'Gtl') - LandSlope
Neg Index: 3468 Neg Weight: -0.1133367009291251 Neg Feature: (35, '311') - BsmtFinSF2

Positive Index: 43 Positive Weight: 0.11753997962046643 Positive Feature: (43, '854') - 2ndflrSF
Neg Index: 3466 Neg Weight: -0.1133367009291251 Neg Feature: (3, '8281') - LotArea

Positive Index: 34 Positive Weight: 0.11071464385833127 Positive Feature: (34, 'Unf')  - BsmtFinT
ype2
Neg Index: 789 Neg Weight: -0.10530603522565173 Neg Feature: (0, '160') - MSSubClass

Positive Index: 4575 Positive Weight: 0.10799250253959562 Positive Feature: (43, '472') - 2ndflrS
F
Neg Index: 974 Neg Weight: -0.10360596910231745 Neg Feature: (16, '3') - OverallQual
```

Most of the positive features make sense, and many of the negative ones do as well. Obviously, the *OverallQual* would be a large positive indicator, taking the 2nd and 5th most positive spots with "good" values of 9 and 8 respectively. *GrLivArea* was also an important factor as the most positive, and 5th most negative value, as sales for living area above ground would be much more valuable than below ground.

There are some factors that make less sense, for instance the MSSubClass being considered fairly negative for seemingly good estates. The third most negative value is for houses "2-STORY 1946 & NEWER" which I would think would be a *positive* indicator. Maybe more intelligent models will fix this.

6. What's your feature weight for the bias dimension? Does it make sense?

The bias dimension is positive, 11.7. This makes sense as all of the log sales prices are positive.

7. Now predict on test.csv, and submit your predictions to the kaggle server. What's your score (RMSLE) and ranking?

My score is 0.1638 with a placing of 7286. Great improvement from my last score, but still really bad in comparison to the 10000 some participants.

3 Smarter binarization: Only binarizing categorical features

You might have observed that most numerical features shouldn't have been binarized: for example, features like LotArea are always positively correlated with the sale price.

1. What are the drawbacks of naive binarization? (Hint: data sparseness)

Naïve binarization is by definition, naïve. It will include all factors regardless of importance, which may skew the model in an incorrect way. Including columns like Alley with mostly NA values will add unnecessary complexity to the model, and likely contribute to nonsense values and predictions.

2. Now binarize only the categorical features, and keep the numerical features as is. What about the mixed features such as LotFrontage and GarageYrBlt?

I will drop the features with more than 50% NA. Alley, MiscFeature, Fence, PoolQC, and FireplaceQu. All remaining NAs will be replaced with 0.0001 (to prevent false zeros and division by zero errors)

3. Redo all the questions asked in the naive binarization section. (Hint: the new dev error should be around 0.14, which is much better than naive binarization). You will see that even if you just do everything up to this point, you should be ranked reasonably in this contest.

Size: 280 features, the same number of unique features as above for categorical, only one each for numerical (obviously)

RMSLE: 0.1242

Intercept: 10.659

Top and bottom 10:

```
Positive Index: 241 Positive Weight: 0.6118120187965786 Positive Feature: (14, 'CBlock')
Neg Index: 275 Neg Weight: -2.338997683125946 Neg Feature: (14, 'CBlock')

Positive Index: 215 Positive Weight: 0.46863921346666954 Positive Feature: (13, 'Tar&Grv')
Neg Index: 257 Neg Weight: -0.7289277992951565 Neg Feature: (13, 'Tar&Grv')

Positive Index: 260 Positive Weight: 0.40100490430077973 Positive Feature: (13, 'Tar&Grv')
Neg Index: 271 Neg Weight: -0.3939452474385067 Neg Feature: (13, 'Tar&Grv')

Positive Index: 272 Positive Weight: 0.35431883553260773 Positive Feature: (13, 'Tar&Grv')
Neg Index: 168 Neg Weight: -0.32776894758913616 Neg Feature: (20, 'Fa')

Positive Index: 228 Positive Weight: 0.32509770428142837 Positive Feature: (36, 'Oth')
Neg Index: 263 Neg Weight: -0.2259928954964688 Neg Feature: (36, 'Oth')

Positive Index: 274 Positive Weight: 0.30501096416332896 Positive Feature: (36, 'Oth')
Neg Index: 240 Neg Weight: -0.22251002280815738 Neg Feature: (25, 'Floor')

Positive Index: 188 Positive Weight: 0.2744963122231323 Positive Feature: (33, 'Gd')
Neg Index: 253 Neg Weight: -0.20661054336342874 Neg Feature: (33, 'Gd')

Positive Index: 251 Positive Weight: 0.2709432473867676 Positive Feature: (33, 'Gd')
Neg Index: 255 Neg Weight: -0.20042937738382388 Neg Feature: (33, 'Gd')
```

```
Positive Index: 49 Positive Weight: 0.229319393285911O6 Positive Feature: (29, 'TA')
Neg Index: 254 Neg Weight: -0.198841328133962 Neg Feature: (29, 'TA')

Positive Index: 197 Positive Weight: 0.21134440425989734 Positive Feature: (8, 'RRNe')
Neg Index: 179 Neg Weight: -0.1686568879562334 Neg Feature: (13, 'Metal')
```

For some reason the positive and negative values for the top and bottom 10 were identical. The scores are good, so I believe it is just the feature values that are pulling incorrectly.

4 Experimentation

Try the following to improve your score.

1. Try regularized linear regression (sklearn.linear_model.Ridge). Tune α on dev. Should improve both naive and smart binarization by a little bit.

The best error rate for the naive binarization is 0.139 and occurs when alpha is between 4.8 and 6.0.

For the smart binarization the ridge method seems to have minimal effect on the error value, remaining around 0.1275.

2. Try non-linear features: what if the sale price is quadratically correlated with some of the most important numerical features such as OverallArea (also known as square footage) and LotArea?

This feature makes sense, but will likely not improve the model significantly as OverallArea is not an important feature on its own. Possibly transforming a high-impact feature such as neighborhood would potentially cause a greater impact to the model.

3. Try feature combinations. An obvious candidate is adding a new feature Years_since_remodeled = YearRemodeled - YearBuilt. But is a feature like this really useful?

A feature such as this is not as useful because this additional feature will have a high degree of correlation with both the YearRemodeled and YearBuilt features. Because these are already in the model, it would be redundant to also add this derived field.

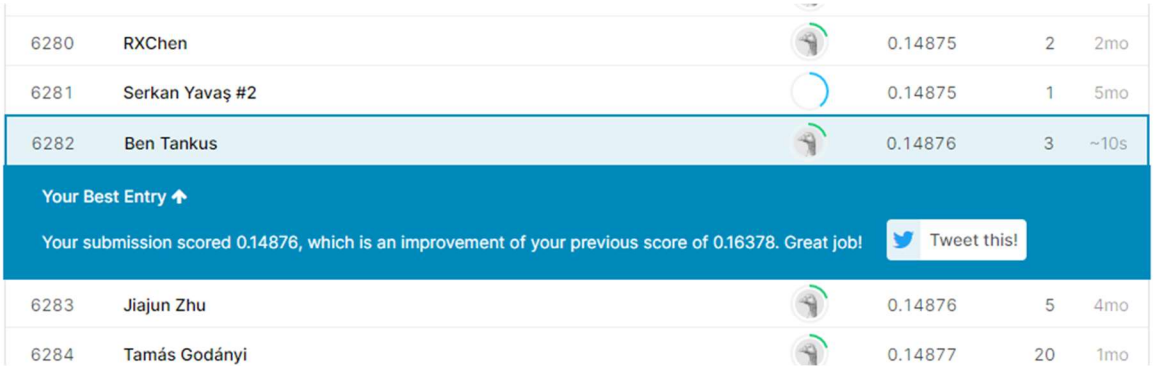
4. BTW, how are these non-linear features (including feature combinations) relate to non-linear features in the perceptron? (think of XOR)

   Perceptron has difficulties separating XOR data whereas non-linear features aid regression in fitting these types of data. This cannot be done in the same way using perceptron.

5. Try anything else that you can think of. You can also find inspirations online, but you have to implement everything yourself (you are not allowed to copy other people's code). What's your best dev error, and what's your best test error and ranking? Take a screen shot of your best test error and ranking, and include your best submission file.

I tried lasso regression but didn't have much luck with naïve jumping up to 0.365 and smart to 0.1593. My best error rate comes from simple smart binarization with 0.12424584328249617

According to Kaggle my error rate is 0.14876 (rank 6282), so that sucks. Anyways, that's what I've got.



Debriefing (required):

1. Approximately how many hours did you spend on this assignment?

About 15 hours. I started late because of mothers day travel, and missed most of the office hours.

2. Would you rate it as easy, moderate, or difficult?

Easier than HW1 and 2, likely because Data Analytics students are heavily exposed to regression in ST 516-518 so I had already learned that part of it.

3. Did you work on it mostly alone, or mostly with other people?

90% alone, 10% in office hours

4. How deeply do you feel you understand the material it covers (0%–100%)? 5. Any other comments?

90%, not to terribly conceptually difficult.