Metamorphic Testing of Deep Learning based Forecaster

Anurag Dwarakanath*, **Manish Ahuja***‡, Sanjay Podder*, Silja Vinu†, Arijit Naskar†, Koushik MV†

*Accenture Labs

†Accenture ‡Speaker

Accenture Labs, 2019

Introduction

- It is becoming extremely common to use Machine Learning in Business Applications.
- However, testing ML applications is a challenge
  - Most ML Applications are non-testable.
  - Presence of bug, especially if application has decent accuracy on validation set.

Current State of Practice

- **Train-Test Split:** Current Testing Methods Splits the data in two parts one for training and other for testing

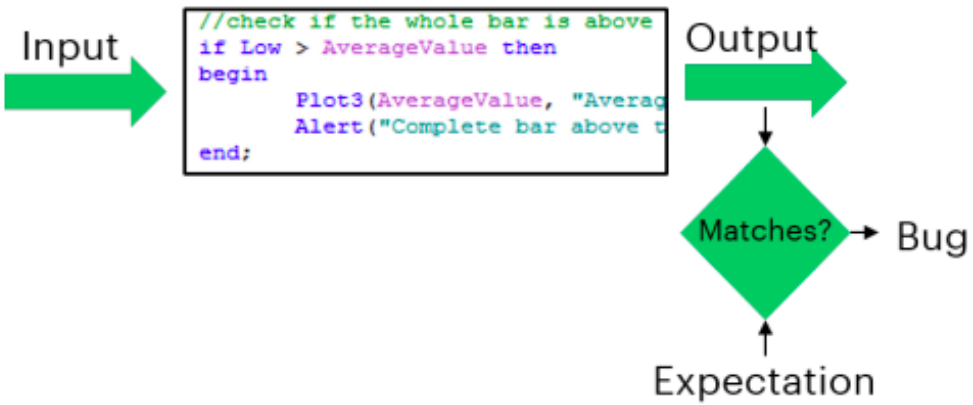
Metamorphic Testing Concept

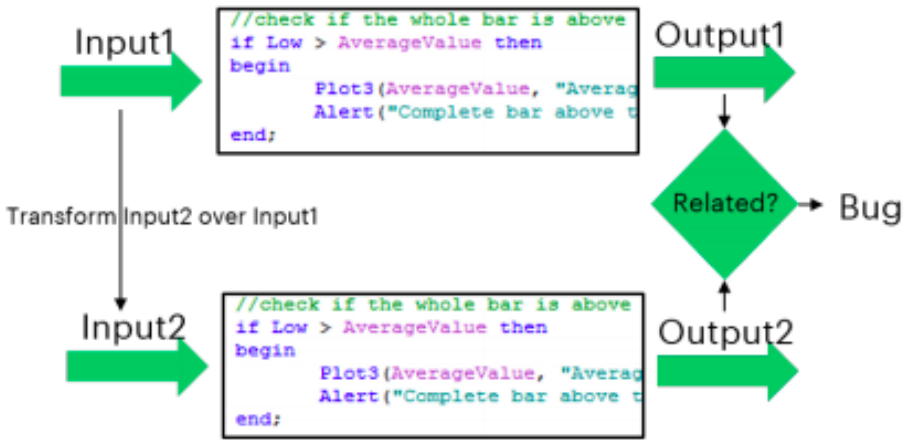
Conventional Testing

We give an input to the Application Under Test and check the output matches the expectation.

Metamorphic Testing

We check for a relation by comparing multiple outputs from the Application Under Test.





Problem Definition

Given Outage Predictor(OP) code and model, is the implementation correct?

- Application trains a model on past system characteristics(features).
- Predict future characteristic values.
- Outage is predicted if prediction is beyond threshold.

We apply Metamorphic Testing on two components of Outage Predictor(OP) separately.

- Predicting the characteristics relevant to target characteristic using Pearson's correlation coefficient.
- Forecasting value of target characteristic using LSTM.

Learning correlated features

- We use Pearson's Coefficient to compute how correlated two features are.
- Negative, Zero and Positive value implies negatively correlated, uncorrelated and positively correlated.

$$P_{XY} = \frac{Cov(X, Y)}{\sigma_X \sigma_Y}$$

$$Cov(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

- Higher the value of $|P_{XY}|$, more correlated X and Y are.

MR's on Pearson's coefficient

Invariant properties of Pearson's coefficient P_{XY}

- **MR1 bounds:** $-1 \leq P_{XY} \leq 1$
- **MR6 Linear Scaling:** P_{XY} is invariant to linear scaling.
  - For $Z = aY + b \implies P_{XY} = P_{XZ}$

Proof.

$Cov(X, aY + b) = a \times Cov(X, Y)$

$\sigma_{aY+b} = a\sigma_Y$

$P_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$

$P_{XY} = P_{X,aY+b}$ □

MR's on Pearson's coefficient

Shuffling data

- **MR2 Interchanging feature columns:** $P_{XY} = P_{YX}$
- **MR3 Shuffling data-points paired order:** Pearson's coefficient depends on paired inputs (x_i, y_i)

MR's on Pearson's coefficient

Perfect Correlation

- **MR4 Duplicate a feature X:** The pearson's coefficient between a feature X and $cX + d, c > 0$ is 1

Proof.

$$\begin{aligned} Cov(X, cX + d) &= \mathbb{E}[(X - \mathbb{E}[X])(cX + d - \mathbb{E}[cX + d])] \\ &= c\mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= c\sigma_X^2 \end{aligned}$$

$$P_{X,cX+d} = \frac{Cov(X,cX+d)}{\sigma_X \sigma_{cX+d}}$$

$$P_{X,cX+d} = \frac{c\sigma_X^2}{c\sigma_X \sigma_X} = 1$$

□

- **MR5 Duplicate a feature $-X$:** The pearson's coefficient between a feature X and $cX + d, c < 0$ is -1

MR's on Pearson's coefficient

Appending new features/values

- **MR7 Add uncorrelated Features:** Add set of features that have 0 correlation coefficient.
- **MR8 Add 0 variance feature:** Application handles this case or not
  - correlation coefficient computation involves dividing by variance.
- **MR9 Introduce outliers**
  - correlation coefficient is impacted hugely by outliers
- **MR10 Introduce Missing Values:** see if the case is handled properly.

Our LSTM Application components

Our LSTM Forecaster takes the 'sales' values of $TIME\_STEPS = 10$ days and predict the forecast for subsequent $NUM\_DAYS\_FORECAST = 1$ days. The forecast application has below 5 components in sequence

1. Formatting Data
2. Preprocess data to $[0, 1]$
3. Train LSTM based RNN on Normalized data
4. Predict normalized data using trained model
5. Convert prediction to Original Scale

Formatting data

- Training and test data consists of 'sales' values for 750 and 187 days respectively.
- Format the data such that input is sequences of $TIME\_STEPS = 10$ days 'sales' values and target as sequences of $NUM\_DAYS\_FORECAST = 1$ days 'sales' values.
- Below is an example sequence for TIME-STEPS= 10 and $NUM\_DAYS\_FORECAST = 2$ days.

	A	B	C
1	date	sales	price
2	1/1/2014	0	1.29
3	1/2/2014	70	1.29
4	1/3/2014	59	1.29
5	1/4/2014	93	1.29
6	1/5/2014	96	1.29
7	1/6/2014	145	1.29
8	1/7/2014	179	1.29
9	1/8/2014	321	1.29
10	1/9/2014	125	1.09
11	1/10/2014	88	1.09
12	1/11/2014	188	1.09
13	1/12/2014	121	1.09
14	1/13/2014	134	1.09

	A	B	C
1	date	sales	price
2	1/27/2016	129	1.46
3	1/28/2016	144	1.46
4	1/29/2016	173	1.79
5	1/30/2016	293	1.79
6	1/31/2016	219	1.79
7	2/1/2016	131	1.79
8	2/2/2016	184	1.79
9	2/3/2016	266	1.79
10	2/4/2016	98	2.18
11	2/5/2016	67	2.29
12	2/6/2016	85	2.29
13	2/7/2016	78	2.29
14	2/8/2016	71	2.19

```
In [32]: # Print the first few sequences of the sales that have been created
         [trainingDataSequence_sales[i, :, 0] for i in range(3)]

Out[32]: [array([  0.,  70.,  59.,  93.,  96., 145., 179., 321., 125.,  88.]),
          array([ 70.,  59.,  93.,  96., 145., 179., 321., 125.,  88., 188.]),
          array([ 59.,  93.,  96., 145., 179., 321., 125.,  88., 188., 121.])]

In [33]: #Print the first few target sales
         [targetDataSequence_sales[i] for i in range(3)]

Out[33]: [array([188., 121.]), array([121., 134.]), array([134.,  80.])]
```

LSTM forecast application

Preprocess data to $[0, 1]$

- We Preprocess data to range $[0, 1]$ by subtracting minimum and dividing by range
- every point $x \in X$ gets normalized to $\frac{x - min(X)}{max(X) - min(X)}$

Train LSTM

- We train a LSTM based RNN with hidden unit size of 250 and 10 *TIME_STEPS*
- We normalize last cell output 'lastCellOutput' with a learned weight matrix W to get predictions
- $normalized\_prediction = W^T(lastCellOutput) + b$

Convert forecasted value to original scale

- $x^{forecast} = (max(X) - min(X)) \times x^{forecast}_{normalized} + min(X)$

MR's for LSTM Forecaster

Linear scaling

- **MR1 Linear scaling of training and test data**
  - Linear scaling of training and test data will not change training and validation loss and forecast will be scaled accordingly.

Proof.

Training data remains same, i.e.

$$
\begin{aligned}
x_{normalized} &= \frac{ax + b - min(aX + b)}{max(aX + b) - min(aX + b)} \\
&= \frac{x - min(X)}{max(X) - min(X)} \\
x^{forecast} &= (max(aX + b) - min(aX + b)) \times x^{forecast}_{normalized} + min(aX + b) \\
&= a((max(X) - min(X)) \times x^{forecast}_{normalized} + min(X)) + b \\
&= a * x^{forecast}_{prev} + b
\end{aligned}
$$

MR's for LSTM forecaster

MR2 Linearly scaling only test data

- Linear Scaling only test data should significantly alter results
- The MR also checks that normalization is not done by mistake on test data

MR3 Checking for missed training data

Check for following training data sizes

1. training data size $= TIME\_STEPS + NUM\_DAYS\_FORECAST$
2. training data size $= TIME\_STEPS + NUM\_DAYS\_FORECAST - 1$
3. training data size $= TIME\_STEPS + BATCH\_SIZE - 1$
4. training data size $= TIME\_STEPS + BATCH\_SIZE$

MR4 is exactly same as MR3, except that it is validated on test data

MR8: Optimal *TIME_STEPS*

- *TIME_STEPS* denotes the amount of past information we need to predict the future outcome.
- Too small value, may lead to poor predictions.
- Too large will be unnecessary ,overhead and will lead to bad performance speed.

We use fourier transform to reconstruct the data after removing high frequencies.

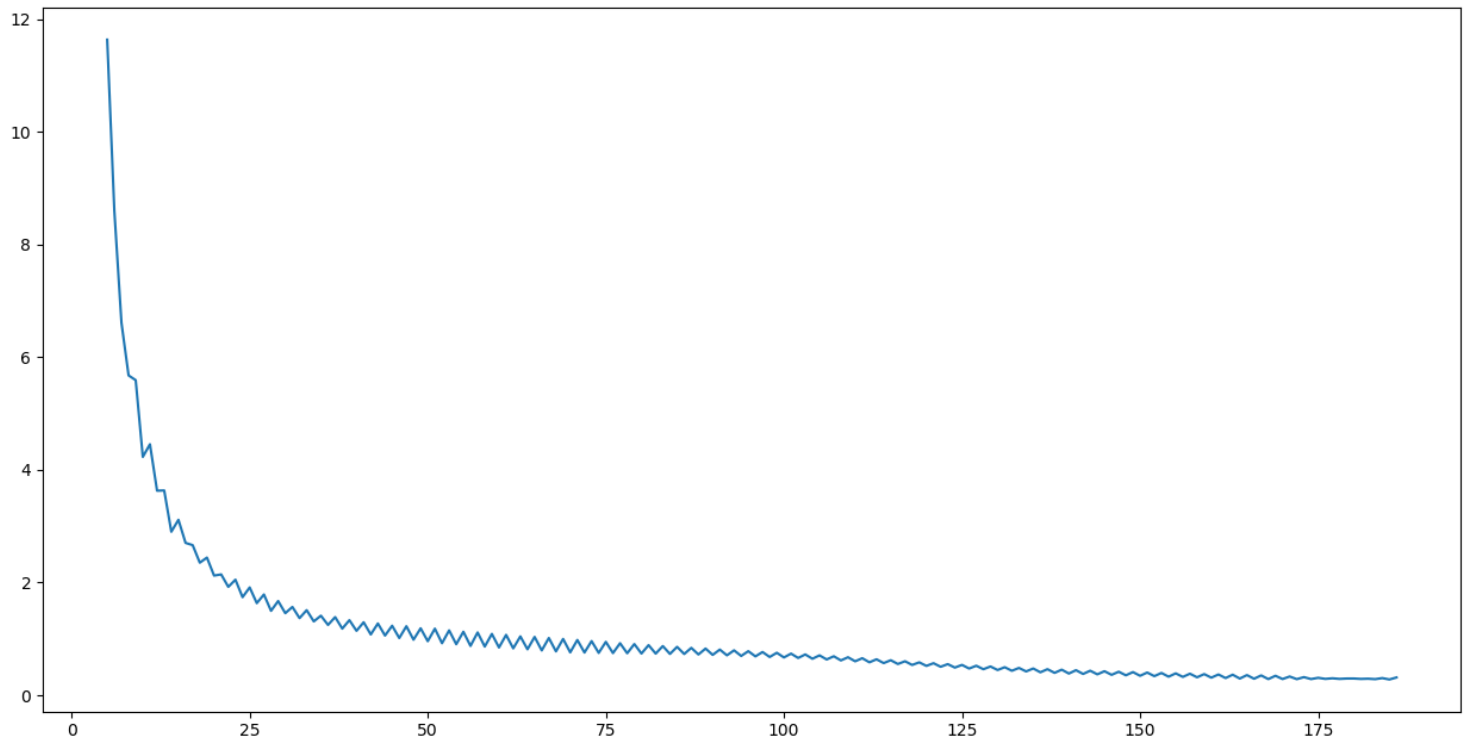
1. Convert the time series data to frequency domain
2. Remove the two highest frequency(lowest information) components.
3. Reconstruct the time series data by inverse fourier transform.
4. Compute the l2 distance between reconstructed and original data.

MR8 example

1. 1 2 3 2 1 2 3 2 1
2. 0.032-0.18j, 0.5 -0.86j, -3.38+2.83j
   -1.15+0.42j, 17. +0.j , -1.15-0.42j
   -3.38-2.83j, 0.5 +0.86j, 0.032+0.18j
3. 0.0+0.0j, 0.5 -0.86j, -3.38+2.83j
   -1.15+0.42j, 17. +0.j , -1.15-0.42j
   -3.38-2.83j, 0.5 +0.86j, 0.0+0.0j
4. 0.99 2.02 2.97 2.03 0.96 2.03 2.97 2.02 0.99

MR8: Reconstruction loss vs *TIME_STEPS*

Following graph shows the reconstruction error vs *TIME_STEPS*, it is obvious from the graph that *TIME_STEPS* of 20-25 is optimal for our data.



MR9: Adversarials on sequence data

- Given a sample input and forecast X_s, y_s
- Find $X_p \approx X_s$, s.t its forecast $y_p = 2y_s$
- Large number of such examples show trained model is not Robust.
- We use optimization method similar to Carlini Wagner for generating adversarials.
- Minimize the loss: $\lambda||X_s - X_p||_2^2 + (2y_s - y_p)^2$

MR9 Example

- Original sequence: 78 122 164 288 138 189 321 209 161 157
- Original forecast: 73
- Perturbed Sequence: 77.98 121.97 164.01 288.03 138.03 188.99 320.96 209.05 161.01 157.58
- Perturbed forecast: 146

Results OP Application

Module	MRs that failed	Num Issues
1) Correlation Coefficient	MR4 MR9 MR10	4
2) LSTM	MR5 MR8 MR9	4

Table: Results: OP Application

Results LSTM forecaster

- We injected hypothetical bugs in reference implementation using Mutation Testing tool 'Mutpy'.
- Mutation Testing generates bugs by modifying lines of code, for example $<$ changed to $>$.
- These modified files are called mutants
- 403 mutants were generated, out of which 44 were valid implementation bugs.

Source File	Mutants	MRs that failed	Mutants caught
Training	30	MR1 MR3	18
Test	14	MR1 MR2 MR3	11

Table: Results: LSTM Forecaster

Conclusion

- We presented metamorphic relations to identify implementation bugs in LSTM based forecaster.
- We defined 10 MR's to check correctness of module designed to choose relevant features using pearson's correlation coefficient.
- We defined 9 MR's to check correctness of LSTM forecaster.
- We used mutation testing to introduce bugs in the application.
- Our MR's caught a total of 29 mutation bugs out of 44.