Explaining Complex Machine Learning Models with LIME

Dr. Shirin Glander

shirin.glander@codecentric.de

2017-12-11
Talk #7 Carlos Guestrin - Explaining the Predictions of Machine Learning models.


Human interpretable explanations:
- For neural networks, random forests, boosted decision trees, etc.

lime - Python
- Created for R: github.com/thomasgb85/LIME

LIME:
- Stands for "Local Interpretable Model-Agnostic Explanations"
- Recreated for R: github.com/marcotcr/lime

LIME:
- Fit simple explanations to cause approximations to the underlying decision
- Find most important variables for specific predictions
- Explanations are found for individual cases by comparing them with similar cases
- Simple models won't be able to solve complex problems!

Trust Transparency
- Improvement through feedback
- Public perception
- Making informed decisions

Trade-off
- Accuracy vs. simplicity

KL = Black Box
- Model-agnostic
- Make use of human intuition
- We typically overestimate the accuracy of our ML models

Local Set Approximation
- Mere weight is being put on data that is similar to the original data
- Generate "new data" by permutation of the input data (test set)
- Predictions on this new data with simple model
LIME can explain any classifier

- image recognition

Figure 4: Explaining an image classification prediction made by Google’s Inception neural network. The top 3 classes predicted are “Electric Guitar” ($p = 0.32$), “Acoustic guitar” ($p = 0.24$) and “Labrador” ($p = 0.21$)

Ribeiro, Singh, and Guestrin (2016)
LIME can explain any classifier

- text classification

Ribeiro, Singh, and Guestrin (2016)
How LIME works

1. Permutation of each test case to explain
2. Complex model predicts all permuted test cases
3. Distance between permutations and original text case is calculated and converted to similarity scores
4. Subsetting features with highest importance in complex model for each permuted test case
5. Fitting a linear model with the subsetted features to the permuted data (weights represent similarity score)
6. Using simple model to explain test case prediction
An example in R

- Data: Chronic Kidney Disease  
  (http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease)
- Nonparametric Missing Value Imputation using Random Forest  
  (library(missForest))
- Categorical features converted to dummy variables  
  (library(dummies))
- Scaled and centered

Predictor: ckd or notckd (class)

- Random Forest model with library(caret) (5x10 repeated CV)
## Random Forest

## 360 samples
## 48 predictor
## 2 classes: 'ckd', 'notckd'

## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 324, 324, 324, 324, 325, 324, ...

## Resampling results across tuning parameters:

<table>
<thead>
<tr>
<th>mtry</th>
<th>Accuracy</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9922647</td>
<td>0.9838466</td>
</tr>
<tr>
<td>25</td>
<td>0.9917392</td>
<td>0.9826070</td>
</tr>
<tr>
<td>48</td>
<td>0.9872930</td>
<td>0.9729881</td>
</tr>
</tbody>
</table>

## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
## Confusion Matrix and Statistics

<table>
<thead>
<tr>
<th></th>
<th>ckd</th>
<th>notckd</th>
</tr>
</thead>
<tbody>
<tr>
<td>ckd</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>notckd</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>

## Accuracy : 0.95
95% CI : (0.8308, 0.9939)
No Information Rate : 0.575
P-Value [Acc > NIR] : 1.113e-07
Kappa : 0.8961
Mcnemar's Test P-Value : 0.4795
Sensitivity : 1.0000
Specificity : 0.8824
Pos Pred Value : 0.9200
Neg Pred Value : 1.0000
Explaining the predictions

Explanation function:

- train_x is the training data
- model_rf is the complex model
- n_bins = 10 groups continuous variables into 10 bins
- quantile_bins = TRUE bases bins on quantiles (bins are not evenly spread across data range)
- dist_fun = “euclidean” sets distance function to calculate weights

library(lime)
explainer <- lime(train_x,
    model_rf,
    n_bins = 10,
    quantile_bins = TRUE,
    dist_fun = "euclidean")
Explaining the predictions

- \( n_{\text{labels}} = 1 \) because we want to explain the most likely class predicted
- \( n_{\text{features}} = 8 \) returns top 8 most important features for each test case
- \( n_{\text{permutations}} = 1000 \) permutes test case 1000x
- feature_select = "highest_weights": fits a ridge regression and selects the top features with highest absolute weight

```r
explanation_df <- lime::explain(test_x,
    explainer,
    n_labels = 1,
    n_features = 8,
    n_permutations = 1000,
    feature_select = "highest_weights")
```
Explanation quality

- model $r^2$
Plotting the explanations

```
plot_features(explanation_df[1:8, ])
```

Case: 4  
Label: ckd  
Probability: 0.98

<table>
<thead>
<tr>
<th>Feature</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcv</td>
<td>0.444</td>
</tr>
<tr>
<td>hemo</td>
<td>0.466</td>
</tr>
<tr>
<td>rbcc</td>
<td>0.238</td>
</tr>
<tr>
<td>sc</td>
<td>0.08026</td>
</tr>
<tr>
<td>sod</td>
<td>0.782</td>
</tr>
<tr>
<td>bu</td>
<td>0.2008</td>
</tr>
<tr>
<td>bgr</td>
<td>0.208</td>
</tr>
<tr>
<td>bp</td>
<td>0.1111</td>
</tr>
</tbody>
</table>

Supports
Contradicts
Plotting the predictions

```python
plot_features(explanation_df[9:16, ])
```

Case: 7  
Label: ckd  
Probability: 0.81

<table>
<thead>
<tr>
<th>Feature</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>bgr &lt;= 0.17</td>
<td>Supports</td>
</tr>
<tr>
<td>sod &lt;= 0.78</td>
<td>Contradicts</td>
</tr>
<tr>
<td>pcv &lt;= 0.53</td>
<td>Supports</td>
</tr>
<tr>
<td>bp &lt;= 0.11</td>
<td>Supports</td>
</tr>
<tr>
<td>hemo &lt;= 0.57</td>
<td>Supports</td>
</tr>
<tr>
<td>sc &lt;= 0.78</td>
<td>Supports</td>
</tr>
<tr>
<td>bgr &lt;= 0.17</td>
<td>Supports</td>
</tr>
<tr>
<td>pot &lt;= 0.03</td>
<td>Supports</td>
</tr>
<tr>
<td>bu &lt;= 0.14</td>
<td>Supports</td>
</tr>
</tbody>
</table>
Plotting the predictions

```python
plot_features(explanation_df[17:24, ])
```

Case: 23  
Label: ckd  
Probability: 0.94  

0.320 < hemo <= 0.388
0.08026 < sc
0.801 < sod <= 0.810
0.404 < pcv <= 0.444
0.135 < bgr <= 0.155
0.170 < wbcc <= 0.189
0.156 < rbcc <= 0.201
0.1111 < bp <= 0.1667

**Feature**

**Weight**

- **Supports**
- **Contradicts**
LIME in action

- Explaining fraud predictions:

https://shiring.shinyapps.io/fraud_example_dashboard/
More about LIME

Publication

- Ribeiro, Singh, and Guestrin (2016)

Contribute

- https://github.com/marcotcr/lime
- https://github.com/thomasp85/lime
Thank you!

...and stay connected...

You can find me on

- my blog: www.shirin-glander.de
- Twitter: https://twitter.com/ShirinGlander
- Github: https://github.com/ShirinG

Code and slides will go up on my blog!

MünsterR User group

- https://www.meetup.com/Munster-R-Users-Group