lime Documentation

Release 0.1

Marco Tulio Ribeiro

Aug 10, 2017

Contents

1 lime package **3**
- 1.1 Submodules . . . 3
- 1.2 lime.lime_tabular module . . . 3
- 1.3 lime.lime_text module . . . 7
- 1.4 lime.discretize module . . . 9
- 1.5 lime.explanation module . . . 10
- 1.6 lime.lime_base module . . . 12

2 Indices and tables **15**

Python Module Index **17**

In this page, you can find the Python API reference for the lime package (local interpretable model-agnostic explanations). For tutorials and more information, visit the github page.

CHAPTER 1

lime package

Submodules

lime.lime_tabular module

Functions for explaining classifiers that use tabular data (matrices).

class lime.lime_tabular.**LimeTabularExplainer**(*training_data*, *mode='classification'*, *training_labels=None*, *feature_names=None*, *categorical_features=None*, *categorical_names=None*, *kernel_width=None*, *verbose=False*, *class_names=None*, *feature_selection='auto'*, *discretize_continuous=True*, *discretizer='quartile'*)

Bases: object

Explains predictions on tabular (i.e. matrix) data. For numerical features, perturb them by sampling from a Normal(0,1) and doing the inverse operation of mean-centering and scaling, according to the means and stds in the training data. For categorical features, perturb by sampling according to the training distribution, and making a binary feature that is 1 when the value is the same as the instance being explained.

Init function.

Parameters

- **training_data** – numpy 2d array
- **mode** – "classification" or "regression"
- **training_labels** – labels for training data. Not required, but may be used by discretizer.
- **feature_names** – list of names (strings) corresponding to the columns in the training data.

- **categorical_features** – list of indices (ints) corresponding to the categorical columns. Everything else will be considered continuous. Values in these columns MUST be integers.
- **categorical_names** – map from int to list of names, where categorical_names[x][y] represents the name of the yth value of column x.
- **kernel_width** – kernel width for the exponential kernel.
- **None, defaults to sqrt** (*If*) –
- **verbose** – if true, print local prediction values from linear model
- **class_names** – list of class names, ordered according to whatever the classifier is using. If not present, class names will be '0', '1', ...
- **feature_selection** – feature selection method. can be 'forward_selection', 'lasso_path', 'none' or 'auto'. See function 'explain_instance_with_data' in lime_base.py for details on what each of the options does.
- **discretize_continuous** – if True, all non-categorical features will be discretized into quartiles.
- **discretizer** – only matters if discretize_continuous is True. Options are 'quartile', 'decile' or 'entropy'

static convert_and_round(*values*)

explain_instance(*data_row*, *predict_fn*, *labels=(1,)*, *top_labels=None*, *num_features=10*, *num_samples=5000*, *distance_metric='euclidean'*, *model_regressor=None*)

Generates explanations for a prediction.

First, we generate neighborhood data by randomly perturbing features from the instance (see __data_inverse). We then learn locally weighted linear models on this neighborhood data to explain each of the classes in an interpretable way (see lime_base.py).

Parameters

- **data_row** – 1d numpy array, corresponding to a row
- **predict_fn** – prediction function. For classifiers, this should be a function that takes a numpy array and outputs prediction probabilities. For regressors, this takes a numpy array and returns the predictions. For ScikitClassifiers, this is

  classifier.predict_proba(). For ScikitRegressors, this is *regressor.predict()*.
- **labels** – iterable with labels to be explained.
- **top_labels** – if not None, ignore labels and produce explanations for the K labels with highest prediction probabilities, where K is this parameter.
- **num_features** – maximum number of features present in explanation
- **num_samples** – size of the neighborhood to learn the linear model
- **distance_metric** – the distance metric to use for weights.
- **model_regressor** – sklearn regressor to use in explanation. Defaults
- **Ridge regression in LimeBase. Must have model_regressor.coef** (*to*) –
- **'sample_weight' as a parameter to model_regressor.fit()** (*and*) –

Returns An Explanation object (see explanation.py) with the corresponding explanations.

class lime.lime_tabular.**RecurrentTabularExplainer**(*training_data*, *training_labels=None*, *feature_names=None*, *categorical_features=None*, *categorical_names=None*, *kernel_width=None*, *verbose=False*, *class_names=None*, *feature_selection='auto'*, *discretize_continuous=True*, *discretizer='quartile'*)

Bases: *lime.lime_tabular.LimeTabularExplainer*

An explainer for keras-style recurrent neural networks, where the input shape is (n_samples, n_timesteps, n_features). This class just extends the LimeTabularExplainer class and reshapes the training data and feature names such that they become something like

(val1_t1, val1_t2, val1_t3, ..., val2_t1, ..., valn_tn)

Each of the methods that take data reshape it appropriately, so you can pass in the training/testing data exactly as you would to the recurrent neural network.

Parameters

- **training_data** – numpy 3d array with shape (n_samples, n_timesteps, n_features)
- **training_labels** – labels for training data. Not required, but may be used by discretizer.
- **feature_names** – list of names (strings) corresponding to the columns in the training data.
- **categorical_features** – list of indices (ints) corresponding to the categorical columns. Everything else will be considered continuous. Values in these columns MUST be integers.
- **categorical_names** – map from int to list of names, where categorical_names[x][y] represents the name of the yth value of column x.
- **kernel_width** – kernel width for the exponential kernel.
- **None, defaults to sqrt** (*If*) –
- **verbose** – if true, print local prediction values from linear model
- **class_names** – list of class names, ordered according to whatever the classifier is using. If not present, class names will be '0', '1', ...
- **feature_selection** – feature selection method. can be 'forward_selection', 'lasso_path', 'none' or 'auto'. See function 'explain_instance_with_data' in lime_base.py for details on what each of the options does.
- **discretize_continuous** – if True, all non-categorical features will be discretized into quartiles.
- **discretizer** – only matters if discretize_continuous is True. Options are 'quartile', 'decile' or 'entropy'

explain_instance(*data_row*, *classifier_fn*, *labels=(1,)*, *top_labels=None*, *num_features=10*, *num_samples=5000*, *distance_metric='euclidean'*, *model_regressor=None*)

Generates explanations for a prediction.

First, we generate neighborhood data by randomly perturbing features from the instance (see __data_inverse). We then learn locally weighted linear models on this neighborhood data to explain each of the classes in an interpretable way (see lime_base.py).

Parameters

- **data_row** – 2d numpy array, corresponding to a row
- **classifier_fn** – classifier prediction probability function, which takes a numpy array and outputs prediction probabilities. For ScikitClassifiers , this is classifier.predict_proba.
- **labels** – iterable with labels to be explained.
- **top_labels** – if not None, ignore labels and produce explanations for the K labels with highest prediction probabilities, where K is this parameter.
- **num_features** – maximum number of features present in explanation
- **num_samples** – size of the neighborhood to learn the linear model
- **distance_metric** – the distance metric to use for weights.
- **model_regressor** – sklearn regressor to use in explanation. Defaults to Ridge regression in LimeBase. Must have **model_regressor.coef_** and 'sample_weight' as a parameter to model_regressor.fit()

Returns An Explanation object (see explanation.py) with the corresponding explanations.

class lime.lime_tabular.**TableDomainMapper**(*feature_names*, *feature_values*, *scaled_row*, *categorical_features*, *discretized_feature_names=None*)

Bases: *lime.explanation.DomainMapper*

Maps feature ids to names, generates table views, etc

Init.

Parameters

- **feature_names** – list of feature names, in order
- **feature_values** – list of strings with the values of the original row
- **scaled_row** – scaled row
- **categorical_features** – list of categorical features ids (ints)

map_exp_ids(*exp*)

Maps ids to feature names.

Parameters **exp** – list of tuples [(id, weight), (id,weight)]

Returns list of tuples (feature_name, weight)

visualize_instance_html(*exp*, *label*, *div_name*, *exp_object_name*, *show_table=True*, *show_all=False*)

Shows the current example in a table format.

Parameters

- **exp** – list of tuples [(id, weight), (id,weight)]
- **label** – label id (integer)
- **div_name** – name of div object to be used for rendering(in js)
- **exp_object_name** – name of js explanation object
- **show_table** – if False, don't show table visualization.
- **show_all** – if True, show zero-weighted features in the table.

lime.lime_text module

Functions for explaining text classifiers.

class lime.lime_text.**IndexedString**(*raw_string*, *split_expression=u'\W+'*, *bow=True*)
Bases: object

String with various indexes.

Initializer.

Parameters

- **raw_string** – string with raw text in it
- **split_expression** – string will be split by this.
- **bow** – if True, a word is the same everywhere in the text - i.e. we will index multiple ocurrences of the same word. If False, order matters, so that the same word will have different ids according to position.

inverse_removing(*words_to_remove*)
Returns a string after removing the appropriate words.

If self.bow is false, replaces word with UNKWORDZ instead of removing it.

Parameters **words_to_remove** – list of ids (ints) to remove

Returns original raw string with appropriate words removed.

num_words()
Returns the number of tokens in the vocabulary for this document.

raw_string()
Returns the original raw string

string_position(*id_*)
Returns a np array with indices to **id_** (int) ocurrences

word(*id_*)
Returns the word that corresponds to **id_** (int)

class lime.lime_text.**LimeTextExplainer**(*kernel_width=25*, *verbose=False*, *class_names=None*, *feature_selection=u'auto'*, *split_expression=u'\W+'*, *bow=True*)
Bases: object

Explains text classifiers. Currently, we are using an exponential kernel on cosine distance, and restricting explanations to words that are present in documents.

Init function.

Parameters

- **kernel_width** – kernel width for the exponential kernel
- **verbose** – if true, print local prediction values from linear model
- **class_names** – list of class names, ordered according to whatever the classifier is using. If not present, class names will be '0', '1', ...
- **feature_selection** – feature selection method. can be 'forward_selection', 'lasso_path', 'none' or 'auto'. See function 'explain_instance_with_data' in lime_base.py for details on what each of the options does.
- **split_expression** – strings will be split by this.

- **bow** – if True (bag of words), will perturb input data by removing all ocurrences of individual words. Explanations will be in terms of these words. Otherwise, will explain in terms of word-positions, so that a word may be important the first time it appears and uninportant the second. Only set to false if the classifier uses word order in some way (bigrams, etc).

explain_instance(*text_instance*, *classifier_fn*, *labels=(1,)*, *top_labels=None*, *num_features=10*, *num_samples=5000*, *distance_metric=u'cosine'*, *model_regressor=None*)

Generates explanations for a prediction.

First, we generate neighborhood data by randomly hiding features from the instance (see __data_labels_distance_mapping). We then learn locally weighted linear models on this neighborhood data to explain each of the classes in an interpretable way (see lime_base.py).

Parameters

- **text_instance** – raw text string to be explained.
- **classifier_fn** – classifier prediction probability function, which takes a list of d strings and outputs a (d, k) numpy array with prediction probabilities, where k is the number of classes. For ScikitClassifiers , this is classifier.predict_proba.
- **labels** – iterable with labels to be explained.
- **top_labels** – if not None, ignore labels and produce explanations for the K labels with highest prediction probabilities, where K is this parameter.
- **num_features** – maximum number of features present in explanation
- **num_samples** – size of the neighborhood to learn the linear model
- **distance_metric** – the distance metric to use for sample weighting, defaults to cosine similarity
- **model_regressor** – sklearn regressor to use in explanation. Defaults
- **Ridge regression in LimeBase. Must have model_regressor.coef**(*to*) –
- **'sample_weight' as a parameter to model_regressor.fit()**(*and*) –

Returns An Explanation object (see explanation.py) with the corresponding explanations.

class lime.lime_text.**TextDomainMapper**(*indexed_string*)

Bases: *lime.explanation.DomainMapper*

Maps feature ids to words or word-positions

Initializer.

Parameters **indexed_string** – lime_text.IndexedString, original string

map_exp_ids(*exp*, *positions=False*)

Maps ids to words or word-position strings.

Parameters

- **exp** – list of tuples [(id, weight), (id,weight)]
- **positions** – if True, also return word positions

Returns list of tuples (word, weight), or (word_positions, weight) if examples: ('bad', 1) or ('bad_3-6-12', 1)

visualize_instance_html(*exp*, *label*, *div_name*, *exp_object_name*, *text=True*, *opacity=True*)

Adds text with highlighted words to visualization.

Parameters

- **exp** – list of tuples [(id, weight), (id,weight)]
- **label** – label id (integer)
- **div_name** – name of div object to be used for rendering(in js)
- **exp_object_name** – name of js explanation object
- **text** – if False, return empty
- **opacity** – if True, fade colors according to weight

lime.discretize module

Discretizers classes, to be used in lime_tabular

class lime.discretize.**BaseDiscretizer**(*data*, *categorical_features*, *feature_names*, *labels=None*)
Bases: object

Abstract class - Build a class that inherits from this class to implement a custom discretizer. Method bins() is to be redefined in the child class, as it is the actual custom part of the discretizer.

Initializer :param data: numpy 2d array :param categorical_features: list of indices (ints) corresponding to the

categorical columns. These features will not be discretized. Everything else will be considered continuous, and will be discretized.

Parameters

- **categorical_names** – map from int to list of names, where categorical_names[x][y] represents the name of the yth value of column x.
- **feature_names** – list of names (strings) corresponding to the columns in the training data.

bins(*data*, *labels*)
To be overridden Returns for each feature to discretize the boundaries that form each bin of the discretizer

discretize(*data*)
Discretizes the data. :param data: numpy 2d or 1d array

Returns numpy array of same dimension, discretized.

undiscretize(*data*)

class lime.discretize.**DecileDiscretizer**(*data*, *categorical_features*, *feature_names*, *labels=None*)
Bases: *lime.discretize.BaseDiscretizer*

bins(*data*, *labels*)

class lime.discretize.**EntropyDiscretizer**(*data*, *categorical_features*, *feature_names*, *labels=None*)
Bases: *lime.discretize.BaseDiscretizer*

bins(*data*, *labels*)

class lime.discretize.**QuartileDiscretizer**(*data*, *categorical_features*, *feature_names*, *labels=None*)
Bases: *lime.discretize.BaseDiscretizer*

bins(*data*, *labels*)

lime.explanation module

Explanation class, with visualization functions.

class lime.explanation.**DomainMapper**
Bases: object

Class for mapping features to the specific domain.

The idea is that there would be a subclass for each domain (text, tables, images, etc), so that we can have a general Explanation class, and separate out the specifics of visualizing features in here.

map_exp_ids(*exp*, ***kwargs*)
Maps the feature ids to concrete names.

Default behaviour is the identity function. Subclasses can implement this as they see fit.

Parameters

- **exp** – list of tuples [(id, weight), (id,weight)]
- **kwargs** – optional keyword arguments

Returns list of tuples [(name, weight), (name, weight)...]

Return type exp

visualize_instance_html(*exp*, *label*, *div_name*, *exp_object_name*, ***kwargs*)
Produces html for visualizing the instance.

Default behaviour does nothing. Subclasses can implement this as they see fit.

Parameters

- **exp** – list of tuples [(id, weight), (id,weight)]
- **label** – label id (integer)
- **div_name** – name of div object to be used for rendering(in js)
- **exp_object_name** – name of js explanation object
- **kwargs** – optional keyword arguments

Returns js code for visualizing the instance

class lime.explanation.**Explanation**(*domain_mapper*, *mode=u'classification'*, *class_names=None*)
Bases: object

Object returned by explainers.

Initializer.

Parameters

- **domain_mapper** – must inherit from DomainMapper class
- **type** – "classification" or "regression"
- **class_names** – list of class names (only used for classification)

as_html(*labels=None*, *predict_proba=True*, *show_predicted_value=True*, ***kwargs*)
Returns the explanation as an html page.

Parameters

- **labels** – desired labels to show explanations for (as barcharts). If you ask for a label for which an explanation wasn't computed, will throw an exception. If None, will show explanations for all available labels. (only used for classification)
- **predict_proba** – if true, add barchart with prediction probabilities for the top classes. (only used for classification)
- **show_predicted_value** – if true, add barchart with expected value (only used for regression)
- **kwargs** – keyword arguments, passed to domain_mapper

Returns code for an html page, including javascript includes.

as_list(*label=1*, ***kwargs*)
Returns the explanation as a list.

Parameters

- **label** – desired label. If you ask for a label for which an explanation wasn't computed, will throw an exception. Will be ignored for regression explanations.
- **kwargs** – keyword arguments, passed to domain_mapper

Returns list of tuples (representation, weight), where representation is given by domain_mapper. Weight is a float.

as_map()
Returns the map of explanations.

Returns Map from label to list of tuples (feature_id, weight).

as_pyplot_figure(*label=1*, ***kwargs*)
Returns the explanation as a pyplot figure.

Will throw an error if you don't have matplotlib installed :param label: desired label. If you ask for a label for which an

explanation wasn't computed, will throw an exception. Will be ignored for regression explanations.

Parameters **kwargs** – keyword arguments, passed to domain_mapper

Returns pyplot figure (barchart).

available_labels()
Returns the list of classification labels for which we have any explanations.

save_to_file(*file_path*, *labels=None*, *predict_proba=True*, *show_predicted_value=True*, ***kwargs*)
Saves html explanation to file. .

Params: file_path: file to save explanations to

See as_html() for additional parameters.

show_in_notebook(*labels=None*, *predict_proba=True*, *show_predicted_value=True*, ***kwargs*)
Shows html explanation in ipython notebook.

See as_html() for parameters. This will throw an error if you don't have IPython installed

lime.explanation.**id_generator**(*size=15*)
Helper function to generate random div ids. This is useful for embedding HTML into ipython notebooks.

lime.lime_base module

Contains abstract functionality for learning locally linear sparse model.

class `lime.lime_base.`**LimeBase**(*kernel_fn*, *verbose=False*)
Bases: `object`

Class for learning a locally linear sparse model from perturbed data

Init function

Parameters

- **kernel_fn** – function that transforms an array of distances into an array of proximity values (floats).
- **verbose** – if true, print local prediction values from linear model.

explain_instance_with_data(*neighborhood_data*, *neighborhood_labels*, *distances*, *label*, *num_features*, *feature_selection='auto'*, *model_regressor=None*)
Takes perturbed data, labels and distances, returns explanation.

Parameters

- **neighborhood_data** – perturbed data, 2d array. first element is assumed to be the original data point.
- **neighborhood_labels** – corresponding perturbed labels. should have as many columns as the number of possible labels.
- **distances** – distances to original data point.
- **label** – label for which we want an explanation
- **num_features** – maximum number of features in explanation
- **feature_selection** – how to select num_features. options are: 'forward_selection': iteratively add features to the model.

  This is costly when num_features is high

  'highest_weights': selects the features that have the highest product of absolute weight * original data point when learning with all the features

  'lasso_path': chooses features based on the lasso regularization path

  'none': uses all features, ignores num_features 'auto': uses forward_selection if num_features <= 6, and

  'highest_weights' otherwise.
- **model_regressor** – sklearn regressor to use in explanation. Defaults to Ridge regression if None. Must have **model_regressor.coef_** and 'sample_weight' as a parameter to model_regressor.fit()

Returns intercept is a float. exp is a sorted list of tuples, where each tuple (x,y) corresponds to the feature id (x) and the local weight (y). The list is sorted by decreasing absolute value of y. score is the R^2 value of the returned explanation

Return type (intercept, exp, score)

feature_selection(*data*, *labels*, *weights*, *num_features*, *method*)
Selects features for the model. see explain_instance_with_data to understand the parameters.

static **forward_selection**(*data*, *labels*, *weights*, *num_features*)
Iteratively adds features to the model

static **generate_lars_path**(*weighted_data*, *weighted_labels*)
Generates the lars path for weighted data.

Parameters

- **weighted_data** – data that has been weighted by kernel
- **weighted_label** – labels, weighted by kernel

Returns (alphas, coefs), both are arrays corresponding to the regularization parameter and coefficients, respectively

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

lime.discretize, 9
lime.explanation, 10
lime.lime_base, 12
lime.lime_tabular, 3
lime.lime_text, 7

Index

A

as_html() (lime.explanation.Explanation method), 10
as_list() (lime.explanation.Explanation method), 11
as_map() (lime.explanation.Explanation method), 11
as_pyplot_figure() (lime.explanation.Explanation method), 11
available_labels() (lime.explanation.Explanation method), 11

B

BaseDiscretizer (class in lime.discretize), 9
bins() (lime.discretize.BaseDiscretizer method), 9
bins() (lime.discretize.DecileDiscretizer method), 9
bins() (lime.discretize.EntropyDiscretizer method), 9
bins() (lime.discretize.QuartileDiscretizer method), 9

C

convert_and_round() (lime.lime_tabular.LimeTabularExplainer static method), 4

D

DecileDiscretizer (class in lime.discretize), 9
discretize() (lime.discretize.BaseDiscretizer method), 9
DomainMapper (class in lime.explanation), 10

E

EntropyDiscretizer (class in lime.discretize), 9
explain_instance() (lime.lime_tabular.LimeTabularExplainer method), 4
explain_instance() (lime.lime_tabular.RecurrentTabularExplainer method), 5
explain_instance() (lime.lime_text.LimeTextExplainer method), 8
explain_instance_with_data() (lime.lime_base.LimeBase method), 12
Explanation (class in lime.explanation), 10

F

feature_selection() (lime.lime_base.LimeBase method), 12
forward_selection() (lime.lime_base.LimeBase static method), 12

G

generate_lars_path() (lime.lime_base.LimeBase static method), 13

I

id_generator() (in module lime.explanation), 11
IndexedString (class in lime.lime_text), 7
inverse_removing() (lime.lime_text.IndexedString method), 7

L

lime.discretize (module), 9
lime.explanation (module), 10
lime.lime_base (module), 12
lime.lime_tabular (module), 3
lime.lime_text (module), 7
LimeBase (class in lime.lime_base), 12
LimeTabularExplainer (class in lime.lime_tabular), 3
LimeTextExplainer (class in lime.lime_text), 7

M

map_exp_ids() (lime.explanation.DomainMapper method), 10
map_exp_ids() (lime.lime_tabular.TableDomainMapper method), 6
map_exp_ids() (lime.lime_text.TextDomainMapper method), 8

N

num_words() (lime.lime_text.IndexedString method), 7

Q

QuartileDiscretizer (class in lime.discretize), 9

R

raw_string() (lime.lime_text.IndexedString method), 7

RecurrentTabularExplainer (class in lime.lime_tabular), 4

S

save_to_file() (lime.explanation.Explanation method), 11
show_in_notebook() (lime.explanation.Explanation method), 11
string_position() (lime.lime_text.IndexedString method), 7

T

TableDomainMapper (class in lime.lime_tabular), 6
TextDomainMapper (class in lime.lime_text), 8

U

undiscretize() (lime.discretize.BaseDiscretizer method), 9

V

visualize_instance_html() (lime.explanation.DomainMapper method), 10
visualize_instance_html() (lime.lime_tabular.TableDomainMapper method), 6
visualize_instance_html() (lime.lime_text.TextDomainMapper method), 8

W

word() (lime.lime_text.IndexedString method), 7