Acton Documentation

Release 0.3.3

Matthew Alger & Cheng Soon Ong

Aug 30, 2018

Contents

Contents:

CHAPTER 1

acton

acton package

Subpackages

acton.proto package

Submodules

acton.proto.acton_pb2 module

acton.proto.io module

Functions for reading/writing to protobufs.

acton.proto.io.**GeneratedProtocolMessageType**(*name*, **args*, ***kwargs*)

acton.proto.io.**get_ndarray**(*data: list*, *shape: tuple*, *dtype: str*) → <MagicMock id='139842872107416'>

Converts a list of values into an array.

Parameters

- **data** – Raw array data.
- **shape** – Shape of the resulting array.
- **dtype** – Data type of the resulting array.

Returns Array with the given data, shape, and dtype.

Return type numpy.ndarray

acton.proto.io.**read_metadata**(*file: typing.Union[str, typing.BinaryIO]*) → bytes

Reads metadata from a protobufs file.

Parameters **file** – Path to binary file, or file itself.

Returns Metadata.

Return type bytes

acton.proto.io.**read_proto**()

Reads a protobuf from a .proto file.

Parameters

- **path** – Path to the .proto file.
- **Proto** – Protocol message class (from the generated protobuf module).

Returns The parsed protobuf.

Return type *GeneratedProtocolMessageType*

acton.proto.io.**read_protos**()
Reads many protobufs from a file.

Parameters

- **file** – Path to binary file, or file itself.
- **Proto** – Protocol message class (from the generated protobuf module).

Yields *GeneratedProtocolMessageType* – A parsed protobuf.

acton.proto.io.**write_proto**()
Serialises a protobuf to a file.

Parameters

- **path** – Path to binary file. Will be overwritten.
- **proto** – Protobuf to write to file.

acton.proto.io.**write_protos**(*path: str*, *metadata: bytes = b''*)
Serialises many protobufs to a file.

Parameters

- **path** – Path to binary file. Will be overwritten.
- **metadata** – Optional bytestring to prepend to the file.

Notes

Coroutine. Accepts protobufs, or None to terminate and close file.

acton.proto.wrappers module

Classes that wrap protobufs.

class acton.proto.wrappers.**LabelPool**(*proto: typing.Union[str, mock.mock.LabelPool]*)
Bases: object

Wrapper for the LabelPool protobuf.

proto
acton_pb.LabelPool – Protobuf representing the label pool.

db_kwargs
dict – Key-value pairs of keyword arguments for the database constructor.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers. May be None.

DB
Gets a database context manager for the specified database.

Returns Database context manager.

Return type type

classmethod **deserialise**(*proto: bytes*, *json: bool = False*) → acton.proto.wrappers.LabelPool
Deserialises a protobuf into a LabelPool.

Parameters

- **proto** – Serialised protobuf.
- **json** – Whether the serialised protobuf is in JSON format.

Returns

Return type *LabelPool*

ids
Gets a list of IDs.

Returns List of known IDs.

Return type List[int]

labels
Gets labels array specified in input.

Notes

The returned array is cached by this object so future calls will not need to recompile the array.

Returns T x N x F NumPy array of labels.

Return type numpy.ndarray

classmethod **make**(*ids: typing.Iterable[int]*, *db: acton.database.Database*) → acton.proto.wrappers.LabelPool
Constructs a LabelPool.

Parameters

- **ids** – Iterable of instance IDs.
- **db** – Database

Returns

Return type *LabelPool*

class acton.proto.wrappers.**Predictions**(*proto: typing.Union[str, mock.mock.Predictions]*)
Bases: object

Wrapper for the Predictions protobuf.

proto
acton_pb.Predictions – Protobuf representing predictions.

db_kwargs
dict – Dictionary of database keyword arguments.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers. May be None.

DB
Gets a database context manager for the specified database.

Returns Database context manager.

Return type type

classmethod **deserialise**(*proto: bytes*, *json: bool = False*) → acton.proto.wrappers.Predictions
Deserialises a protobuf into Predictions.

Parameters

- **proto** – Serialised protobuf.
- **json** – Whether the serialised protobuf is in JSON format.

Returns

Return type *Predictions*

labelled_ids
Gets a list of IDs the predictor knew the label for.

Returns List of IDs the predictor knew the label for.

Return type List[int]

classmethod **make**(*predicted_ids: typing.Iterable[int], labelled_ids: typing.Iterable[int], predictions: <MagicMock id='139842871968768'>, db: acton.database.Database, predictor: str = ''*) → acton.proto.wrappers.Predictions
Converts NumPy predictions to a Predictions object.

Parameters

- **predicted_ids** – Iterable of instance IDs corresponding to predictions.
- **labelled_ids** – Iterable of instance IDs used to train the predictor.
- **predictions** – T x N x D array of corresponding predictions.
- **predictor** – Name of predictor used to generate predictions.
- **db** – Database.

Returns

Return type *Predictions*

predicted_ids
Gets a list of IDs corresponding to predictions.

Returns List of IDs corresponding to predictions.

Return type List[int]

predictions
Gets predictions array specified in input.

Notes

The returned array is cached by this object so future calls will not need to recompile the array.

Returns T x N x D NumPy array of predictions.

Return type numpy.ndarray

class acton.proto.wrappers.**Recommendations**(*proto: typing.Union[str, mock.mock.Recommendations]*)

Bases: object

Wrapper for the Recommendations protobuf.

proto
acton_pb.Recommendations – Protobuf representing recommendations.

db_kwargs
dict – Key-value pairs of keyword arguments for the database constructor.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers. May be None.

DB
Gets a database context manager for the specified database.

Returns Database context manager.

Return type type

classmethod deserialise(*proto: bytes, json: bool = False*) → acton.proto.wrappers.Recommendations
Deserialises a protobuf into Recommendations.

Parameters

- **proto** – Serialised protobuf.
- **json** – Whether the serialised protobuf is in JSON format.

Returns

Return type *Recommendations*

labelled_ids
Gets a list of labelled IDs.

Returns List of labelled IDs.

Return type List[int]

classmethod make(*recommended_ids: typing.Iterable[int], labelled_ids: typing.Iterable[int], recommender: str, db: acton.database.Database*) → acton.proto.wrappers.Recommendations
Constructs a Recommendations.

Parameters

- **recommended_ids** – Iterable of recommended instance IDs.
- **labelled_ids** – Iterable of labelled instance IDs used to make recommendations.
- **recommender** – Name of the recommender used to make recommendations.
- **db** – Database.

Returns

Return type *Recommendations*

recommendations
Gets a list of recommended IDs.

Returns List of recommended IDs.

Return type List[int]

acton.proto.wrappers.**deserialise_encoder**(*encoder: mock.mock.LabelEncoder*) → <MagicMock name='mock.LabelEncoder' id='139842871924328'>

Deserialises a LabelEncoder protobuf.

Parameters **encoder** – LabelEncoder protobuf.

Returns LabelEncoder (or None if no encodings were specified).

Return type sklearn.preprocessing.LabelEncoder

acton.proto.wrappers.**validate_db**(*db: mock.mock.Database*)

Validates a Database proto.

Parameters **db** – Database to validate.

Raises ValueError

Module contents

Submodules

acton.acton module

Main processing script for Acton.

acton.acton.**draw**(*n: int, lst: typing.List[T], replace: bool = True*) → typing.List[T]

Draws n random elements from a list.

Parameters

- **n** – Number of elements to draw.
- **lst** – List of elements to draw from.
- **replace** – Draw with replacement.

Returns n random elements.

Return type List[T]

acton.acton.**get_DB**(*data_path: str, pandas_key: str = None) -> (<class 'acton.database.Database'>, <class 'dict'>*)

Gets a Database that will handle the given data table.

Parameters

- **data_path** – Path to file.
- **pandas_key** – Key for pandas HDF5. Specify iff using pandas.

Returns

- *Database* – Database that will handle the given data table.
- *dict* – Keyword arguments for the Database constructor.

acton.acton.**label**(*recommendations: acton.proto.wrappers.Recommendations*) → acton.proto.wrappers.LabelPool

Simulates a labelling task.

Parameters

- **data_path** – Path to data file.

- **feature_cols** – List of column names of features. If empty, all columns will be used.
- **label_col** – Column name of the labels.
- **pandas_key** – Key for pandas HDF5. Specify iff using pandas.

Returns

Return type *acton.proto.wrappers.LabelPool*

acton.acton.**main**(*data_path: str, feature_cols: typing.List[str], label_col: str, output_path: str, n_epochs: int = 10, initial_count: int = 10, recommender: str = 'RandomRecommender', predictor: str = 'LogisticRegression', pandas_key: str = '', n_recommendations: int = 1*)

Simulate an active learning experiment.

Parameters

- **data_path** – Path to data file.
- **feature_cols** – List of column names of the features. If empty, all non-label and non-ID columns will be used.
- **label_col** – Column name of the labels.
- **output_path** – Path to output file. Will be overwritten.
- **n_epochs** – Number of epochs to run.
- **initial_count** – Number of random instances to label initially.
- **recommender** – Name of recommender to make recommendations.
- **predictor** – Name of predictor to make predictions.
- **pandas_key** – Key for pandas HDF5. Specify iff using pandas.
- **n_recommendations** – Number of recommendations to make at once.

acton.acton.**predict**(*labels: acton.proto.wrappers.LabelPool, predictor: str*) → acton.proto.wrappers.Predictions

Train a predictor and predict labels.

Parameters

- **labels** – IDs of labelled instances.
- **predictor** – Name of predictor to make predictions.

acton.acton.**recommend**(*predictions: acton.proto.wrappers.Predictions, recommender: str = 'RandomRecommender', n_recommendations: int = 1*) → acton.proto.wrappers.Recommendations

Recommends instances to label based on predictions.

Parameters

- **recommender** – Name of recommender to make recommendations.
- **n_recommendations** – Number of recommendations to make at once. Default 1.

Returns

Return type *acton.proto.wrappers.Recommendations*

acton.acton.**simulate_active_learning**(*ids: typing.Iterable[int], db: acton.database.Database, db_kwargs: dict, output_path: str, n_initial_labels: int = 10, n_epochs: int = 10, test_size: int = 0.2, recommender: str = 'RandomRecommender', predictor: str = 'LogisticRegression', n_recommendations: int = 1*)

Simulates an active learning task.

Parameters

- **ids** – IDs of instances in the unlabelled pool.
- **db** – Database with features and labels.
- **db_kwargs** – Keyword arguments for the database constructor.
- **output_path** – Path to output intermediate predictions to. Will be overwritten.
- **n_initial_labels** – Number of initial labels to draw.
- **n_epochs** – Number of epochs.
- **test_size** – Percentage size of testing set.
- **recommender** – Name of recommender to make recommendations.
- **predictor** – Name of predictor to make predictions.
- **n_recommendations** – Number of recommendations to make at once.

acton.acton.**try_pandas**(*data_path: str*) → bool

Guesses if a file is a pandas file.

Parameters **data_path** – Path to file.

Returns True if the file is pandas.

Return type bool

acton.acton.**validate_predictor**(*predictor: str*)

Raises an exception if the predictor is not valid.

Parameters **predictor** – Name of predictor.

Raises `ValueError`

acton.acton.**validate_recommender**(*recommender: str*)

Raises an exception if the recommender is not valid.

Parameters **recommender** – Name of recommender.

Raises `ValueError`

acton.cli module

Command-line interface for Acton.

acton.cli.**lines_from_stdin**() → typing.Iterable[str]

Yields lines from stdin.

acton.cli.**read_binary**() → bytes

Reads binary data from stdin.

Notes

The first eight bytes are expected to be the length of the input data as an unsigned long long.

Returns Binary data.

Return type bytes

acton.cli.**read_bytes_from_buffer**(*n: int*, *buffer: typing.BinaryIO*) → bytes
Reads n bytes from stdin, blocking until all bytes are received.

Parameters

- **n** – How many bytes to read.
- **buffer** – Which buffer to read from.

Returns Exactly n bytes.

Return type bytes

acton.cli.**write_binary**(*string: bytes*)
Writes binary data to stdout.

Notes

The output will be preceded by the length as an unsigned long long.

acton.database module

Wrapper class for databases.

class acton.database.**ASCIIReader**(*path: str, feature_cols: typing.List[str], label_col: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139842872942664'> = None*)
Bases: *acton.database.Database*

Reads ASCII databases.

feature_cols
List[str] – List of feature columns.

label_col
str – Name of label column.

max_id_length
int – Maximum length of IDs.

n_features
int – Number of features.

n_instances
int – Number of instances.

n_labels
int – Number of labels per instance.

path
str – Path to ASCII file.

encode_labels
bool – Whether to encode labels as integers.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_db
Database – Underlying ManagedHDF5Database.

_db_filepath
str – Path of underlying HDF5 database.

_tempdir
str – Temporary directory where the underlying HDF5 database is stored.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842872889864'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842873868920'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842873006176'>*)

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872911912'>*)

class acton.database.**Database**
Bases: abc.ABC

Base class for database wrappers.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842873068176'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842873081472'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842872582440'>*)
Writes feature vectors to the database.

Parameters

- **ids** – Iterable of IDs.
- **features** – N x D array of feature vectors. The ith row corresponds to the ith ID in *ids*.

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872591640'>*)
Writes label vectors to the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.
- **labels** – T x N x D array of label vectors. The ith row corresponds to the ith labeller ID in *labeller_ids* and the jth column corresponds to the jth instance ID in *instance_ids*.

class acton.database.**FITSReader**(*path: str, feature_cols: typing.List[str], label_col: str, hdu_index: int = 1, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139842872313504'> = None*)
Bases: *acton.database.Database*

Reads FITS databases.

hdu_index
int – Index of HDU in the FITS file.

feature_cols
List[str] – List of feature columns.

label_col
str – Name of label column.

n_features
int – Number of features.

n_instances
int – Number of instances.

n_labels
int – Number of labels per instance.

path
str – Path to FITS file.

encode_labels
bool – Whether to encode labels as integers.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_hdulist
astropy.io.fits.HDUList – FITS HDUList.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842872326408'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842872347896'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x 1 array of label vectors.

Return type numpy.p

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842872360800'>*)

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872378256'>*)

class acton.database.**HDF5Database**(*path: str*)
Bases: *acton.database.Database*

Database wrapping an HDF5 file as a context manager.

path
str – Path to HDF5 file.

_h5_file
h5py.File – HDF5 file object.

class acton.database.**HDF5Reader**(*path: str, feature_cols: typing.List[str], label_col: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139842873044608'> = None*)
Bases: *acton.database.HDF5Database*

Reads HDF5 databases.

feature_cols
List[str] – List of feature datasets.

label_col
str – Name of label dataset.

n_features
int – Number of features.

n_instances
int – Number of instances.

n_labels
int – Number of labels per instance.

path
str – Path to HDF5 file.

encode_labels
bool – Whether to encode labels as integers.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_h5_file
h5py.File – HDF5 file object.

_is_multidimensional
bool – Whether the features are in a multidimensional dataset.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842873024584'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842872995120'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842873016104'>*)

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872937944'>*)

class acton.database.**ManagedHDF5Database**(*path: str*, *label_dtype: str = None*, *feature_dtype: str = None*)
Bases: *acton.database.HDF5Database*

Database using an HDF5 file.

Notes

This database uses an internal schema. For reading files from disk, use another Database.

path
str – Path to HDF5 file.

label_dtype
str – Data type of labels.

feature_dtype
str – Data type of features.

_h5_file
h5py.File – Opened HDF5 file.

_sync_attrs
List[str] – List of instance attributes to sync with the HDF5 file's attributes.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842872639672'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842872674456'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.

Returns T x N x F array of label vectors.

Return type numpy.ndarray

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842872610320'>*)
Writes feature vectors to the database.

Parameters

- **ids** – Iterable of IDs.
- **features** – N x D array of feature vectors. The ith row corresponds to the ith ID in *ids*.

Returns N x D array of feature vectors.

Return type numpy.ndarray

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872652968'>*)
Writes label vectors to the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.
- **instance_ids** – Iterable of instance IDs.
- **labels** – T x N x D array of label vectors. The ith row corresponds to the ith labeller ID in *labeller_ids* and the jth column corresponds to the jth instance ID in *instance_ids*.

class acton.database.**PandasReader**(*path: str, feature_cols: typing.List[str], label_col: str, key: str, encode_labels: bool = True, label_encoder: <MagicMock name='mock.LabelEncoder' id='139842872750544'> = None*)

Bases: *acton.database.Database*

Reads HDF5 databases.

feature_cols
List[str] – List of feature datasets.

label_col
str – Name of label dataset.

n_features
int – Number of features.

n_instances
int – Number of instances.

n_labels
int – Number of labels per instance.

path
str – Path to HDF5 file.

encode_labels
bool – Whether to encode labels as integers.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

_df
pandas.DataFrame – Pandas dataframe.

get_known_instance_ids() → typing.List[int]
Returns a list of known instance IDs.

Returns A list of known instance IDs.

Return type List[str]

get_known_labeller_ids() → typing.List[int]
Returns a list of known labeller IDs.

Returns A list of known labeller IDs.

Return type List[str]

read_features(*ids: typing.Sequence[int]*) → <MagicMock id='139842872763448'>
Reads feature vectors from the database.

Parameters **ids** – Iterable of IDs.

Returns N x D array of feature vectors.

Return type numpy.ndarray

read_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int]*) → <MagicMock id='139842872772648'>
Reads label vectors from the database.

Parameters

- **labeller_ids** – Iterable of labeller IDs.

- **instance_ids** – Iterable of instance IDs.

Returns T x N x 1 array of label vectors.

Return type numpy.ndarray

to_proto() → mock.mock.Database
Serialises this database as a protobuf.

Returns Protobuf representing this database.

Return type DatabasePB

write_features(*ids: typing.Sequence[int], features: <MagicMock id='139842872785552'>*)

write_labels(*labeller_ids: typing.Sequence[int], instance_ids: typing.Sequence[int], labels: <MagicMock id='139842872802944'>*)

acton.database.**product**(*seq: typing.Iterable[int]*)
Finds the product of a list of ints.

Parameters **seq** – List of ints.

Returns Product.

Return type int

acton.database.**serialise_encoder**(*encoder: <MagicMock name='mock.LabelEncoder' id='139842873055216'>*) → mock.mock.LabelEncoder
Serialises a LabelEncoder as a protobuf.

Parameters **encoder** – LabelEncoder.

Returns Protobuf representing the LabelEncoder.

Return type LabelEncoderPB

acton.kde_predictor module

A predictor that uses KDE to classify instances.

class acton.kde_predictor.**KDEClassifier**(*bandwidth=1.0*)
Bases: BaseEstimator, ClassifierMixin

A classifier using kernel density estimation to classify instances.

fit(*X*, *y*)
Fits kernel density models to the data.

Parameters

- **X** (*array_like, shape (n_samples, n_features)*) – List of n_features-dimensional data points. Each row corresponds to a single data point.
- **y** (*array-like, shape (n_samples,)*) – Target vector relative to X.

predict(*X*)
Predicts class labels.

Parameters **X** (*array_like, shape (n_samples, n_features)*) – List of n_features-dimensional data points. Each row corresponds to a single data point.

predict_proba(*X*)
Predicts class probabilities.

Class probabilities are normalised log densities of the kernel density estimates.

Parameters **X** (*array_like, shape (n_samples, n_features)*) – List of n_features-dimensional data points. Each row corresponds to a single data point.

acton.labellers module

Labeller classes.

class acton.labellers.**ASCIITableLabeller**(*path: str*, *id_col: str*, *label_col: str*)
Bases: *acton.labellers.Labeller*

Labeller that obtains labels from an ASCII table.

path
str – Path to table.

id_col
str – Name of the column where IDs are stored.

label_col
str – Name of the column where binary labels are stored.

_table
astropy.table.Table – Table object.

query(*id_: int*) → <MagicMock id='139842872455008'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns 1 x 1 label array.

Return type numpy.ndarray

class acton.labellers.**DatabaseLabeller**(*db: acton.database.Database*)
Bases: *acton.labellers.Labeller*

Labeller that obtains labels from a Database.

_db
acton.database.Database – Database with labels.

query(*id_: int*) → <MagicMock id='139842872480264'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns 1 x 1 label array.

Return type numpy.ndarray

class acton.labellers.**Labeller**
Bases: abc.ABC

Base class for labellers.

query(*id_: int*) → <MagicMock id='139842872457984'>
Queries the labeller.

Parameters **id** – ID of instance to label.

Returns T x F label array.

Return type numpy.ndarray

acton.plot module

Script to plot a dump of predictions.

acton.plot.**plot**(*predictions: typing.Iterable[typing.BinaryIO]*)
Plots predictions from a file.

Parameters **predictions** – Files containing predictions.

acton.predictors module

Predictor classes.

acton.predictors.**AveragePredictions**(*predictor: acton.predictors.Predictor*) → acton.predictors.Predictor
Wrapper for a predictor that averages predicted probabilities.

Notes

This effectively reduces the number of predictors to 1.

Parameters **predictor** – Predictor to wrap.

Returns Predictor with averaged predictions.

Return type *Predictor*

class acton.predictors.**Committee**(*Predictor: type*, *db: acton.database.Database*, *n_classifiers: int = 10*, *subset_size: float = 0.6*, ***kwargs: dict*)
Bases: *acton.predictors.Predictor*

A predictor using a committee of other predictors.

n_classifiers
int – Number of logistic regression classifiers in the committee.

subset_size
float – Percentage of known labels to take subsets of to train the classifier. Lower numbers increase variety.

_db
acton.database.Database – Database storing features and labels.

_committee
List[sklearn.linear_model.LogisticRegression] – Underlying committee of logistic regression classifiers.

_reference_predictor
Predictor – Reference predictor trained on all known labels.

fit(*ids: typing.Iterable[int]*)
Fits the predictor to labelled data.

Parameters **ids** – List of IDs of instances to train from.

predict(*ids: typing.Sequence[int]*) -> (<MagicMock id='139842871835224'>, <MagicMock id='139842871847736'>)
Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x T x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

reference_predict(*ids: typing.Sequence[int]*) -> (<*MagicMock id='139842871856544'*>, <*MagicMock id='139842871864960'*>)

Predicts labels using the best possible method.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x 1 x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

class acton.predictors.**GPClassifier**(*db: acton.database.Database*, *max_iters: int = 50000*, *n_jobs: int = 1*)

Bases: *acton.predictors.Predictor*

Classifier using Gaussian processes.

max_iters
int – Maximum optimisation iterations.

label_encoder
sklearn.preprocessing.LabelEncoder – Encodes labels as integers.

model_
gpy.models.GPClassification – GP model.

_db
acton.database.Database – Database storing features and labels.

fit(*ids: typing.Iterable[int]*)
Fits the predictor to labelled data.

Parameters **ids** – List of IDs of instances to train from.

predict(*ids: typing.Sequence[int]*) -> (<*MagicMock id='139842872252736'*>, <*MagicMock id='139842872270072'*>)

Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x 1 x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

reference_predict (*ids: typing.Sequence[int]) -> (<MagicMock id='139842872194048'>, <MagicMock id='139842872176712'>*)
Predicts labels using the best possible method.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x 1 x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

class acton.predictors.**Predictor**
Bases: abc.ABC

Base class for predictors.

prediction_type
str – What kind of predictions this class generates, e.g. classification.s

fit (*ids: typing.Iterable[int]*)
Fits the predictor to labelled data.

Parameters **ids** – List of IDs of instances to train from.

predict (*ids: typing.Sequence[int]) -> (<MagicMock id='139842872281128'>, <MagicMock id='139842872264968'>*)
Predicts labels of instances.

Notes

Unlike in scikit-learn, predictions are always real-valued. Predicted labels for a classification problem are represented by predicted probabilities of each class.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x T x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

prediction_type = 'classification'

reference_predict (*ids: typing.Sequence[int]) -> (<MagicMock id='139842872298352'>, <MagicMock id='139842871782480'>*)
Predicts labels using the best possible method.

Parameters **ids** – List of IDs of instances to predict labels for.

Returns

- *numpy.ndarray* – An N x 1 x C array of corresponding predictions.
- *numpy.ndarray* – A N array of confidences (or None if not applicable).

acton.predictors.**from_class** (*Predictor: type*, *regression: bool = False*) → type
Converts a scikit-learn predictor class into a Predictor class.

Parameters

- **Predictor** – scikit-learn predictor class.
- **regression** – Whether this predictor does regression (as opposed to classification).

Returns Predictor class wrapping the scikit-learn class.

Return type type

acton.predictors.**from_instance**(*predictor: BaseEstimator*, *db: acton.database.Database*, *regression: bool = False*) → acton.predictors.Predictor

Converts a scikit-learn predictor instance into a Predictor instance.

Parameters

- **predictor** – scikit-learn predictor.
- **db** – Database storing features and labels.
- **regression** – Whether this predictor does regression (as opposed to classification).

Returns Predictor instance wrapping the scikit-learn predictor.

Return type *Predictor*

acton.recommenders module

Recommender classes.

class acton.recommenders.**EntropyRecommender**(*db: acton.database.Database*)

Bases: *acton.recommenders.Recommender*

Recommends instances by confidence-based uncertainty sampling.

recommend(*ids: typing.Sequence[int]*, *predictions: <MagicMock id='139842872142648'>*, *n: int = 1*, *diversity: float = 0.5*) → typing.Sequence[int]

Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x 1 x C array of predictions. The ith row must correspond with the ith ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**MarginRecommender**(*db: acton.database.Database*)

Bases: *acton.recommenders.Recommender*

Recommends instances by margin-based uncertainty sampling.

recommend(*ids: typing.Sequence[int]*, *predictions: <MagicMock id='139842872113360'>*, *n: int = 1*, *diversity: float = 0.5*) → typing.Sequence[int]

Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.

- **predictions** – N x 1 x C array of predictions. The ith row must correspond with the ith ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**QBCRecommender**(*db: acton.database.Database*)
Bases: *acton.recommenders.Recommender*

Recommends instances by committee disagreement.

recommend(*ids: typing.Sequence[int], predictions: <MagicMock id='139842872011968'>, n: int = 1, diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x T x C array of predictions. The ith row must correspond with the ith ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**RandomRecommender**(*db: acton.database.Database*)
Bases: *acton.recommenders.Recommender*

Recommends instances at random.

recommend(*ids: typing.Sequence[int], predictions: <MagicMock id='139842872020048'>, n: int = 1, diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x T x C array of predictions.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**Recommender**
Bases: abc.ABC

Base class for recommenders.

recommend(*ids: typing.Sequence[int], predictions: <MagicMock id='139842871553384'>, n: int = 1, diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x T x C array of predictions.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

class acton.recommenders.**UncertaintyRecommender**(*db: acton.database.Database*)
Bases: *acton.recommenders.Recommender*

Recommends instances by confidence-based uncertainty sampling.

recommend(*ids: typing.Sequence[int], predictions: <MagicMock id='139842871871584'>, n: int = 1, diversity: float = 0.5*) → typing.Sequence[int]
Recommends an instance to label.

Notes

Assumes predictions are probabilities of positive binary label.

Parameters

- **ids** – Sequence of IDs in the unlabelled data pool.
- **predictions** – N x 1 x C array of predictions. The ith row must correspond with the ith ID in the sequence.
- **n** – Number of recommendations to make.
- **diversity** – Recommendation diversity in [0, 1].

Returns IDs of the instances to label.

Return type Sequence[int]

acton.recommenders.**choose_boltzmann**(*features: <MagicMock id='139842871558944'>, scores: <MagicMock id='139842871522696'>, n: int, temperature: float = 1.0*) → typing.Sequence[int]
Chooses n scores using a Boltzmann distribution.

Notes

Scores are chosen from highest to lowest. If there are less scores to choose from than requested, all scores will be returned in order of preference.

Parameters

- **scores** – 1D array of scores.
- **n** – Number of scores to choose.

- **temperature** – Temperature parameter for sampling. Higher temperatures give more diversity.

Returns List of indices of scores chosen.

Return type Sequence[int]

acton.recommenders.**choose_mmr**(*features: <MagicMock id='139842871551872'>*, *scores: <MagicMock id='139842871584416'>*, *n: int*, *l: float = 0.5*) → typing.Sequence[int]

Chooses n scores using maximal marginal relevance.

Notes

Scores are chosen from highest to lowest. If there are less scores to choose from than requested, all scores will be returned in order of preference.

Parameters

- **scores** – 1D array of scores.
- **n** – Number of scores to choose.
- **l** – Lambda parameter for MMR. l = 1 gives a relevance-ranked list and l = 0 gives a maximal diversity ranking.

Returns List of indices of scores chosen.

Return type Sequence[int]

Module contents

CHAPTER 2

Developer Documentation

Contributing

We accept pull requests on GitHub. Contributions must be PEP8 compliant and pass formatting and function tests in the test script `/test`.

Adding a New Predictor

A predictor is a class that implements `acton.predictors.Predictor`. Adding a new predictor amounts to implementing a subclass of `Predictor` and registering it in `acton.predictors.PREDICTORS`.

Predictors must implement:

- `__init__(db: acton.database.Database,*args,**kwargs)`, which stores a reference to the database (and does any other initialisation).
- `fit(ids: Iterable[int])`, which takes an iterable of IDs and fits a model to the associated features and labels,
- `predict(ids: Sequence[int]) -> numpy.ndarray`, which takes a sequence of IDs and predicts the associated labels.
- `reference_predict(ids: Sequence[int]) -> numpy.ndarray`, which behaves the same as `predict` but uses the best possible model.

Predictors should store data-based values such as the model in attributes ending in an underscore, e.g. `self.model_`.

Why Does Acton Use Predictor?

Acton makes use of `Predictor` classes, which are often just wrappers for scikit-learn classes. This raises the question: Why not just use scikit-learn classes?

This design decision was made because Acton must support predictors that do not fit the scikit-learn API, and so using scikit-learn predictors directly would mean that there is no unified API for predictors. An example of where Acton diverges from scikit-learn is that scikit-learn does not support multiple labellers.

Adding a New Recommender

A recommender is a class that implements `acton.recommenders.Recommender`. Adding a new recommender amounts to implementing a subclass of `Recommender` and registering it in `acton.recommenders.RECOMMENDERS`.

Recommenders must implement:

- `__init__(db: acton.database.Database,*args,**kwargs)`, which stores a reference to the database (and does any other initialisation).
- `recommend(ids: Iterable[int],predictions: numpy.ndarray,n: int=1,diversity: float=0.5)` -> `Sequence[int]`, which recommends `n` IDs from the given IDs based on the associated predictions.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

a

Symbols

A

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

W