vowpalwabbit.pyvw

Python binding for pylibvw class

class vowpalwabbit.pyvw.**SearchTask**(*vw*, *sch*, *num_actions*)

Bases: object

Search task class

Methods

example(self[, initStringOrDict, labelType])	Create an example initStringOrDict can specify example as VW formatted string, or a dictionary labelType can specify the desire label type
learn(self, data_iterator)	Train search task by providing an iterator of examples.
predict(self, my_example[, useOracle])	Predict on the example

__init__(*self*, *vw*, *sch*, *num_actions*)

Parameters:
vw : *vw object*

sch : *search object*

num_actions : *integer*
The number of actions with the object can be initialized with

Returns:
self : *SearchTask*

example(*self*, *initStringOrDict=None*, *labelType=0*)

Create an example initStringOrDict can specify example as VW formatted string, or a dictionary labelType can specify the desire label type

Parameters:
initStringOrDict : *str/dict*
Example in either string or dictionary form

labelType : *integer*
labelType of the example, by default is 0(lDefault)

Returns: **out** : *Example*

learn(*self*, *data_iterator*)

Train search task by providing an iterator of examples.

Parameters: **data_iterator: iterable objects**
Consists of examples to be learned

Returns: **self** : *SearchTask*

predict(*self*, *my_example*, *useOracle=False*)

Predict on the example

Parameters: **my_example** : *Example*
example used for prediction

useOracle : *bool*

Returns: **out** : *integer*
Prediction on the example

class vowpalwabbit.pyvw.**abstract_label**

Bases: **object**

An abstract class for a VW label.

Methods

from_example(self, ex)	grab a label from a given VW example

__init__(*self*)

Initialize self. See help(type(self)) for accurate signature.

from_example(*self*, *ex*)

grab a label from a given VW example

class vowpalwabbit.pyvw.**cbandits_label**(*costs=[]*, *prediction=0*)

Bases: **vowpalwabbit.pyvw.abstract_label**

Class for contextual bandits VW label

Methods

`from_example`(self, ex)	grab a label from a given VW example

`__init__`(*self*, *costs=[]*, *prediction=0*)

Initialize self. See help(type(self)) for accurate signature.

`from_example`(*self*, *ex*)

grab a label from a given VW example

class `vowpalwabbit.pyvw.`**`cost_sensitive_label`**(*costs=[]*, *prediction=0*)

Bases: `vowpalwabbit.pyvw.abstract_label`

Class for cost sensative VW label

Methods

`from_example`(self, ex)	grab a label from a given VW example

`__init__`(*self*, *costs=[]*, *prediction=0*)

Initialize self. See help(type(self)) for accurate signature.

`from_example`(*self*, *ex*)

grab a label from a given VW example

class `vowpalwabbit.pyvw.`**`example`**(*vw*, *initStringOrDictOrRawExample=None*, *labelType=0*)

Bases: **`pylibvw.example`**

The example class is a (non-trivial) wrapper around pylibvw.example. Most of the wrapping is to make the interface easier to use (by making the types safer via namespace_id) and also with added python-specific functionality.

Methods

`ensure_namespace_exists`(self, ns)	Check to see if a namespace already exists.
`erase_namespace`()	Remove all the features from a given namespace
`feature`(self, ns, i)	Get the i-th hashed feature id in a given namespace

feature_weight(self, ns, i)	Get the value(weight) associated with a given feature id
get_action_scores()	Get action scores from example prediction
get_cbandits_class(arg1, arg2)	Assuming a contextual_bandits label type, get the label for a given pair (i=0..
get_cbandits_cost(arg1, arg2)	Assuming a contextual_bandits label type, get the cost for a given pair (i=0..
get_cbandits_num_costs()	Assuming a contextual_bandits label type, get the total number of label/cost pairs
get_cbandits_partial_prediction(arg1, arg2)	Assuming a contextual_bandits label type, get the partial prediction for a given pair (i=0..
get_cbandits_prediction()	Assuming a contextual_bandits label type, get the prediction
get_cbandits_probability(arg1, arg2)	Assuming a contextual_bandits label type, get the bandits probability for a given pair (i=0..
get_costsensitive_class(arg1, arg2)	Assuming a cost_sensitive label type, get the label for a given pair (i=0..
get_costsensitive_cost(arg1, arg2)	Assuming a cost_sensitive label type, get the cost for a given pair (i=0..
get_costsensitive_num_costs()	Assuming a cost_sensitive label type, get the total number of label/cost pairs

get_costsensitive_partial_prediction(arg1, arg2)	Assuming a cost_sensitive label type, get the partial prediction for a given pair (i=0..
get_costsensitive_prediction()	Assuming a cost_sensitive label type, get the prediction
get_costsensitive_wap_value(arg1, arg2)	Assuming a cost_sensitive label type, get the weighted-all-pairs recomputed cost for a given pair (i=0..
get_decision_scores()	Get decision scores from example prediction
get_example_counter()	Returns the counter of total number of examples seen up to and including this one
get_feature_id(self, ns, feature[, ns_hash])	Get the hashed feature id for a given feature in a given namespace.
get_feature_number()	Returns the total number of features for this example
get_ft_offset(arg1)	Returns the feature offset for this example (used, eg, by multiclass classification to bulk offset all features)
get_label(self[, label_class])	Given a known label class (default is simple_label), get the corresponding label structure for this example.
get_loss()	Returns the loss associated with this example
get_multiclass_label()	Assuming a multiclass label type, get the true label
get_multiclass_prediction()	Assuming a multiclass label type, get the prediction

get_multiclass_weight()	Assuming a multiclass label type, get the importance weight
get_multilabel_predictions()	Get multilabel predictions from example prediction
get_ns(self, id)	Construct a namespace_id
get_partial_prediction()	Returns the partial prediction associated with this example
get_prob()	Get probability from example prediction
get_scalars()	Get scalar values from example prediction
get_simplelabel_initial()	Assuming a simple_label label type, return the initial (baseline) prediction
get_simplelabel_label(arg1)	Assuming a simple_label label type, return the corresponding label (class/regression target/etc.)
get_simplelabel_prediction()	Assuming a simple_label label type, return the final prediction
get_simplelabel_weight()	Assuming a simple_label label type, return the importance weight
get_tag()	Returns the tag associated with this example
get_topic_prediction()	For LDA models, returns the topic prediction for the topic id given
get_total_sum_feat_sq()	The total sum of feature-value squared for this example
get_updated_prediction()	Returns the partial prediction as if we had updated it after learning

iter_features(self)	Iterate over all feature/value pairs in this example (all namespace included).
learn(self)	Learn on this example (and before learning, automatically call setup_example if the example hasn't yet been setup).
namespace()	Get the namespace id for namespace i (for i = 0..
num_features_in(self, ns)	Get the total number of features in a given namespace
num_namespaces()	The total number of namespaces associated with this example
pop_feature(self, ns)	Remove the top feature from a given namespace
pop_namespace(self)	Remove the top namespace from an example
push_feature(self, ns, feature[, v, ns_hash])	Add an unhashed feature to a given namespace
push_feature_dict()	Add a (Python) dictionary of namespace/feature-list pairs
push_feature_list()	Add a (Python) list of features to a given namespace
push_features(self, ns, featureList)	Push a list of features to a given namespace.
push_hashed_feature(self, ns, f[, v])	Add a hashed feature to a given namespace.
push_namespace(self, ns)	Push a new namespace onto this example.
set_label_string(self, string)	Give this example a new label
set_test_only()	Change the test-only bit on an example

`setup_example`(self)	If this example hasn't already been setup (ie, quadratic features constructed, etc.), do so.
`sum_feat_sq`(self, ns)	Get the total sum feature-value squared for a given namespace
`unsetup_example`(self)	If this example has been setup, reverse that process so you can continue editing the examples.

`__init__`(*self*, *vw*, *initStringOrDictOrRawExample=None*, *labelType=0*)

Construct a new example from vw.

Parameters:

vw : *vw*

vw model

initStringOrDictOrRawExample : *dict/string/None*

If initString is None, you get an "empty" example which you can construct by hand (see, eg, example.push_features). If initString is a string, then this string is parsed as it would be from a VW data file into an example (and "setup_example" is run). if it is a dict, then we add all features in that dictionary. finally, if it's a function, we (repeatedly) execute it fn() until it's not a function any more(for lazy feature computation). By default is None

labelType : *integer*

The labelType of example, by default is 0(lDefault)

Returns:

self : *Example*

`ensure_namespace_exists`(*self*, *ns*)

Check to see if a namespace already exists.

Parameters:

ns : *namespace*

If namespace exists does, do nothing. If it doesn't, add it.

`feature`(*self*, *ns*, *i*)

Get the i-th hashed feature id in a given namespace

Parameters: **ns** : *namespace*
namespace used to get the feature

i : *integer*
to get i-th hashed feature id in a given ns. It must range from 0 to self.num_features_in(ns)-1

Returns: **f** : *integer*
i-th hashed feature-id in a given ns

feature_weight(*self*, *ns*, *i*)

Get the value(weight) associated with a given feature id

Parameters: **ns** : *namespace*
namespace used to get the feature id

i : *integer*
to get the weight of i-th feature in the given ns. It must range from 0 to self.num_features_in(ns)-1

Returns: **out** : *float*
weight(value) of the i-th feature of given ns

get_feature_id(*self*, *ns*, *feature*, *ns_hash=None*)

Get the hashed feature id for a given feature in a given namespace. feature can either be an integer (already a feature id) or a string, in which case it is hashed.

Parameters: **ns** : *namespace*
namespace used to get the feature

feature : *integer/string*
If integer the already a feature else will be hashed

ns_hash : *Optional, by default is None*
The hash of the namespace

Returns: **out** : *integer*
Hashed feature id

Note:

If –hash all is on, then get_feature_id(ns,”5”) !=

get_feature_id(ns, 5). If you’ve already hashed the namespace,

you can optionally provide that value to avoid re-hashing it.

get_label(*self*, *label_class=<class 'vowpalwabbit.pyvw.simple_label'>*)

Given a known label class (default is simple_label), get the corresponding label structure for this example.

Parameters: **label_class** : *label classes*
Get the label of the example of label_class type, by default is simple_label

get_ns(*self*, *id*)

Construct a namespace_id

Parameters: **id** : *namespace_id/str/integer*
id used to create namespace

Returns: **out** : *namespace_id*
namespace_id created using parameter passed(if id was namespace_id, just return it directly)

iter_features(*self*)

Iterate over all feature/value pairs in this example (all namespace included).

learn(*self*)

Learn on this example (and before learning, automatically call setup_example if the example hasn’t yet been setup).

num_features_in(*self*, *ns*)

Get the total number of features in a given namespace

Parameters: **ns** : *namespace*
Get the total features of this namespace

Returns: **num_features** : *integer*
Total number of features in the given ns

pop_feature(*self*, *ns*)

Remove the top feature from a given namespace

Parameters: **ns** : *namespace*
namespace from which feature is popped

Returns: **out** : *bool*
True if feature was removed else False as no feature was there to pop

pop_namespace(*self*)

Remove the top namespace from an example

Returns: **out** : *bool*
True if namespace was removed else False as no namespace was there to pop

push_feature(*self*, *ns*, *feature*, *v=1.0*, *ns_hash=None*)

Add an unhashed feature to a given namespace

Parameters: **ns** : *namespace*
namespace in which the feature is to be pushed

f : *integer*
feature

v : *float*
The value of the feature, be default is 1.0

ns_hash : *Optional, by default is None*
The hash of the namespace

push_features(*self*, *ns*, *featureList*)

Push a list of features to a given namespace.

Parameters: **ns** : *namespace*
namespace in which the features are pushed

featureList : *list*
Each feature in the list can either be an integer (already hashed) or a string (to be hashed) and may be paired with a value or not (if not, the value is assumed to be 1.0

Examples

```
>>> from vowpalwabbit import pyvw
>>> vw = pyvw.vw(quiet=True)
>>> ex = vw.example('1 |a two features |b more features here')
>>> ex.push_features('x', ['a', 'b'])
>>> ex.push_features('y', [('c', 1.), 'd'])
>>> space_hash = vw.hash_space('x')
>>> feat_hash  = vw.hash_feature('a', space_hash)
>>> ex.push_features('x', [feat_hash]) #'x' should match the space_hash!
>>> ex.num_features_in('x')
3
>>> ex.num_features_in('y')
2
```

push_hashed_feature(*self*, *ns*, *f*, *v=1.0*)

Add a hashed feature to a given namespace.

Parameters: **ns** : *namespace*
namespace in which the feature is to be pushed

f : *integer*
feature

v : *float*
The value of the feature, be default is 1.0

push_namespace(*self*, *ns*)

Push a new namespace onto this example. You should only do this if you're sure that this example doesn't already have the given namespace

Parameters: **ns** : *namespace*
namespace which is to be pushed onto example

set_label_string(*self*, *string*)

Give this example a new label

Parameters: **string** : *str*
a new label to this example, formatted as a string (ala the VW data file format)

setup_example(*self*)
If this example hasn't already been setup (ie, quadratic features constructed, etc.), do so.

sum_feat_sq(*self*, *ns*)
Get the total sum feature-value squared for a given namespace

Parameters: **ns** : *namespace*
Get the total sum feature-value squared of this namespace

Returns: **sum_sq** : *float*
Total sum feature-value squared of the given ns

unsetup_example(*self*)
If this example has been setup, reverse that process so you can continue editing the examples.

class vowpalwabbit.pyvw.**example_namespace**(*ex*, *ns*, *ns_hash=None*)
Bases: object

The example_namespace class is a helper class that allows you to extract namespaces from examples and operate at a namespace level rather than an example level. Mainly this is done to enable indexing like ex['x'][0] to get the 0th feature in namespace 'x' in example ex.

Methods

iter_features(self)	iterate over all feature/value pairs in this namespace.
num_features_in(self)	Return the total number of features in this namespace.
pop_feature(self)	Remove the top feature from the current namespace; returns True if a feature was removed, returns False if there were no features to pop.
push_feature(self, feature[, v])	Add an unhashed feature to the current namespace (fails if setup has already run on this example).

`push_features`(self, ns, featureList)	Push a list of features to a given namespace.

`__init__`(*self*, *ex*, *ns*, *ns_hash=None*)

Construct an example_namespace

Parameters:
- **ex** : *Example*
  examples from which namespace is to be extracted
- **ns** : *namespace_id*
  Target namespace
- **ns_hash** : *Optional, by default is None*
  The hash of the namespace

Returns: **self** : *example_namespace*

`iter_features`(*self*)

iterate over all feature/value pairs in this namespace.

`num_features_in`(*self*)

Return the total number of features in this namespace.

`pop_feature`(*self*)

Remove the top feature from the current namespace; returns True if a feature was removed, returns False if there were no features to pop.

`push_feature`(*self*, *feature*, *v=1.0*)

Add an unhashed feature to the current namespace (fails if setup has already run on this example).

Parameters:
- **feature** : *integer/str*
  Feature to be pushed to current namespace
- **v** : *float*
  Feature value, by default is 1.0

`push_features`(*self*, *ns*, *featureList*)

Push a list of features to a given namespace.

Parameters: **ns** : *namespace*

namespace to which feature list is to be pushed

featureList : *list*

Each feature in the list can either be an integer (already hashed) or a string (to be hashed) and may be paired with a value or not (if not, the value is assumed to be 1.0).

See example.push_features for examples.

vowpalwabbit.pyvw.**get_prediction**(*ec*, *prediction_type*)

Get specified type of prediction from example

Parameters: **ec** : *Example*

prediction_type

- pSCALAR : Scalar prediction type
- pSCALARS : Multiple scalar-valued prediction type
- pACTION_SCORES : Multiple action scores prediction type
- pACTION_PROBS : Multiple action probabilities prediction type
- pMULTICLASS : Multiclass prediction type
- pMULTILABELS : Multilabel prediction type
- pPROB : Probability prediction type
- pMULTICLASSPROBS : Multiclass probabilities prediction type
- pDECISION_SCORES : Decision scores prediction type

Returns: **out** : *integer/list*

Prediction according to parameter prediction_type

Examples

```
>>> from vowpalwabbit import pyvw
>>> import pylibvw
>>> vw = pyvw.vw(quiet=True)
>>> ex = vw.example('1 |a two features |b more features here')
>>> pyvw.get_prediction(ex, pylibvw.vw.pSCALAR)
0.0
>>> pyvw.get_prediction(ex, pylibvw.vw.pPROB)
0.0
```

class vowpalwabbit.pyvw.**multiclass_label**(*label=1*, *weight=1.0*, *prediction=1*)

Bases: `vowpalwabbit.pyvw.abstract_label`

Class for multiclass VW label with prediction

Methods

`from_example`(self, ex)	grab a label from a given VW example

__init__(*self*, *label=1*, *weight=1.0*, *prediction=1*)

Initialize self. See help(type(self)) for accurate signature.

from_example(*self*, *ex*)

grab a label from a given VW example

class vowpalwabbit.pyvw.**multiclass_probabilities_label**(*label*, *prediction=None*)

Bases: `vowpalwabbit.pyvw.abstract_label`

Class for multiclass VW label with probabilities

Methods

`from_example`(self, ex)	grab a label from a given VW example

__init__(*self*, *label*, *prediction=None*)

Initialize self. See help(type(self)) for accurate signature.

from_example(*self*, *ex*)

grab a label from a given VW example

class vowpalwabbit.pyvw.**namespace_id**(*ex*, *id*)

Bases: `object`

The namespace_id class is simply a wrapper to convert between hash spaces referred to by character (eg ‘x’) versus their index in a particular example. Mostly used internally, you shouldn’t really need to touch this.

__init__(*self*, *ex*, *id*)

Given an example and an id, construct a namespace_id.

Parameters: **ex** : *Example*
example used to create a namespace id

id : *integer/str*
The id can either be an integer (in which case we take it to be an index into ex.indices[]) or a string (in which case we take the first character as the namespace id).

Returns: **self** : *namespace_id*

class vowpalwabbit.pyvw.**simple_label**(*label=0.0*, *weight=1.0*, *initial=0.0*, *prediction=0.0*)

Bases: vowpalwabbit.pyvw.abstract_label

Class for simple VW label

Methods

from_example(self, ex)	grab a label from a given VW example

__init__(*self*, *label=0.0*, *weight=1.0*, *initial=0.0*, *prediction=0.0*)
Initialize self. See help(type(self)) for accurate signature.

from_example(*self*, *ex*)
grab a label from a given VW example

class vowpalwabbit.pyvw.**vw**(*arg_str=None*, ***kw*)

Bases: **pylibvw.vw**

The pyvw.vw object is a (trivial) wrapper around the pylibvw.vw object; you're probably best off using this directly and ignoring the pylibvw.vw structure entirely.

Methods

audit_example()	print example audit information
example(self[, stringOrDict, labelType])	Create an example initStringOrDict can specify example as VW formatted string, or a dictionary labelType can specify the desire label type

finish(self)	stop VW by calling finish (and, eg, write weights to disk)
finish_example(self, ex)	Should only be used in conjunction with the parse method
get_arguments()	return the arguments after resolving all dependencies
get_id()	return the model id
get_label_type()	return parse label type
get_prediction_type()	return prediction type
get_search_ptr()	return a pointer to the search data structure
get_stride()	return the internal stride
get_sum_loss()	return the total cumulative loss suffered so far
get_weight(self, index[, offset])	Get the weight at a particular position in the (learned) weight vector.
get_weighted_examples()	return the total weight of examples so far
hash_feature()	given a feature string (arg2) and a hashed namespace (arg3), hash that feature
hash_space()	given a namespace (as a string), compute the hash of that namespace
learn(self, ec)	Perform an online update
learn_multi()	given a list pyvw examples, learn (and predict) on those examples
num_weights(self)	Get length of weight vector.
parse(self, str_ex[, labelType])	Returns a collection of examples for a multiline example learner or a single example for a single example learner.
predict(self, ec[, prediction_type])	Just make a prediction on the example
predict_multi()	given a list of pyvw examples, predict on that example
run_parser()	parse external data file

save(self, filename)	save model to disk
set_weight()	set the weight for a particular index
setup_example(arg1, arg2)	given an example that you've created by hand, prepare it for learning (eg, compute quadratic feature)
unsetup_example()	reverse the process of setup, so that you can go back and modify this example

init_search_task

__init__(*self*, *arg_str=None*, ***kw*)

Initialize the vw object.

Parameters: **arg_str** : *str*

The arg_str is the same as the command line arguments, by default is None. You'd use to run vw (eg,"–audit").

****kw** : *Using key/value pairs for different options available*

Returns: **self** : *vw*

Examples

```
>>> from vowpalwabbit import vw
>>> vw1 = pyvw.vw('--audit')
>>> vw2 = pyvw.vw(audit=True, b=24, k=True, c=True, l2=0.001)
>>> vw3 = pyvw.vw("--audit", b=26)
>>> vw4 = pyvw.vw("-q", ["ab", "ac"])
```

example(*self*, *stringOrDict=None*, *labelType=0*)

Create an example initStringOrDict can specify example as VW formatted string, or a dictionary labelType can specify the desire label type

Parameters: **initStringOrDict** : *str/dict*

Example in either string or dictionary form

labelType : *integer*

labelType of the example, by default is 0(lDefault)

Returns: **out** : *Example*

finish(*self*)

stop VW by calling finish (and, eg, write weights to disk)

finish_example(*self*, *ex*)

Should only be used in conjunction with the parse method

Parameters: **ex** : *Example*
example to be finished

get_weight(*self*, *index*, *offset=0*)

Get the weight at a particular position in the (learned) weight vector.

Parameters: **index** : *integer*
position in the learned weight vector

offset : *integer*
By default is 0

Returns: **weight** : *float*
Weight at the given index

init_search_task(*self*, *search_task*, *task_data=None*)

learn(*self*, *ec*)

Perform an online update

Parameters: **ec** : *example/str/list*
examples on which the model gets updated

num_weights(*self*)

Get length of weight vector.

parse(*self*, *str_ex*, *labelType=0*)

Returns a collection of examples for a multiline example learner or a single example for a single example learner.

Parameters: **str_ex** : *str/list of str*
string representing examples

labelType : *integer*
labelType of the example, by default is 0(lDefault)

Returns: **ec** : *list*
list of examples parsed

Examples

```
>>> from vowpalwabbit import pyvw
>>> model = pyvw.vw(quiet=True)
>>> ex = model.parse("0:0.1:0.75 | a:0.5 b:1 c:2")
>>> len(ex)
1
>>> model = vw(quiet=True, cb_adf=True)
>>> ex = model.parse(["| a:1 b:0.5", "0:0.1:0.75 | a:0.5 b:1 c:2"])
>>> len(ex) # Shows the multiline example is parsed
2
```

predict(*self*, *ec*, *prediction_type=None*)

Just make a prediction on the example

Parameters: **ec** : *Example/list/str*
examples to be predicted

prediction_type : *optional, if provided then the matching return type is*
used otherwise the the learner's prediction type will determine the output, by default is None

Returns: **prediction** : *Prediction made on each examples*

save(*self*, *filename*)

save model to disk