tf.contrib.learn.SVM

Contents

Class **SVM**

Inherits From: `Estimator`

Defined in `tensorflow/contrib/learn/python/learn/estimators/svm.py`.

Support Vector Machine (SVM) model for binary classification.

Currently, only linear SVMs are supported. For the underlying optimization problem, the `SDCAOptimizer` is used. For performance and convergence tuning, the num_loss_partitions parameter passed to `SDCAOptimizer` (see `__init__()` method), should be set to (#concurrent train ops per worker) x (#workers). If num_loss_partitions is larger or equal to this value, convergence is guaranteed but becomes slower as num_loss_partitions increases. If it is set to a smaller value, the optimizer is more aggressive in reducing the global loss but convergence is not guaranteed. The recommended value in an `Estimator` (where there is one process per worker) is the number of workers running the train steps. It defaults to 1 (single machine).

Example:

```
real_feature_column = real_valued_column(...)
sparse_feature_column = sparse_column_with_hash_bucket(...)

estimator = SVM(
    example_id_column='example_id',
    feature_columns=[real_feature_column, sparse_feature_column],
    l2_regularization=10.0)

# Input builders
def input_fn_train: # returns x, y
  ...
def input_fn_eval: # returns x, y
  ...

estimator.fit(input_fn=input_fn_train)
estimator.evaluate(input_fn=input_fn_eval)
estimator.predict(x=x)
```

Input of `fit` and `evaluate` should have following features, otherwise there will be a `KeyError`: a feature with `key=example_id_column` whose value is a `Tensor` of dtype string. if `weight_column_name` is not `None`, a feature with `key=weight_column_name` whose value is a `Tensor`. for each `column` in `feature_columns`: - if `column` is a `SparseColumn`, a feature with `key=column.name` whose `value` is a `SparseTensor`. - if `column` is a `RealValuedColumn, a feature with` key=column.name `whose` value `is a` Tensor`.

Properties

`config`

`model_dir`

Methods

`__init__`

```
__init__(
    example_id_column,
    feature_columns,
    weight_column_name=None,
    model_dir=None,
    l1_regularization=0.0,
    l2_regularization=0.0,
    num_loss_partitions=1,
    kernels=None,
    config=None,
    feature_engineering_fn=None
)
```

Constructs an `SVM` estimator object.

Args:

- `example_id_column`: A string defining the feature column name representing example ids. Used to initialize the underlying optimizer.
- `feature_columns`: An iterable containing all the feature columns used by the model. All items in the set should be instances of classes derived from `FeatureColumn`.
- `weight_column_name`: A string defining feature column name representing weights. It is used to down weight or boost examples during training. It will be multiplied by the loss of the example.
- `model_dir`: Directory to save model parameters, graph and etc. This can also be used to load checkpoints from the directory into a estimator to continue training a previously saved model.
- `l1_regularization`: L1-regularization parameter. Refers to global L1 regularization (across all examples).
- `l2_regularization`: L2-regularization parameter. Refers to global L2 regularization (across all examples).
- `num_loss_partitions`: number of partitions of the (global) loss function optimized by the underlying optimizer (SDCAOptimizer).
- `kernels`: A list of kernels for the SVM. Currently, no kernels are supported. Reserved for future use for non-linear SVMs.
- `config`: RunConfig object to configure the runtime settings.
- `feature_engineering_fn`: Feature engineering function. Takes features and labels which are the output of `input_fn` and returns features and labels which will be fed into the model.

Raises:

- `ValueError`: if kernels passed is not None.

`evaluate`

```
evaluate(
    x=None,
    y=None,
    input_fn=None,
    feed_fn=None,
    batch_size=None,
    steps=None,
    metrics=None,
    name=None,
    checkpoint_path=None,
    hooks=None,
    log_progress=True
)
```

See `Evaluable`. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-12-01. Instructions for updating: Estimator is decoupled from Scikit Learn interface by moving into separate class SKCompat. Arguments x, y and batch_size are only available in the SKCompat class, Estimator will only accept input_fn. Example conversion: est = Estimator(...) -> est = SKCompat(Estimator(...))

Raises:

- `ValueError`: If at least one of `x` or `y` is provided, and at least one of `input_fn` or `feed_fn` is provided. Or if `metrics` is not `None` or `dict`.

`export`

```
export(
    export_dir,
    signature_fn=None,
    input_fn=None,
    default_batch_size=1,
    exports_to_keep=None
)
```

See BaseEstimator.export. (deprecated)

THIS FUNCTION IS DEPRECATED. It will be removed after 2017-03-25. Instructions for updating: Please use Estimator.export_savedmodel() instead.

`export_savedmodel`

```
export_savedmodel(
    export_dir_base,
    serving_input_fn,
    default_output_alternative_key=None,
    assets_extra=None,
    as_text=False,
    checkpoint_path=None,
    graph_rewrite_specs=(GraphRewriteSpec((tag_constants.SERVING,), ()),)
)
```

Exports inference graph as a SavedModel into given dir.

Args:

- `export_dir_base`: A string containing a directory to write the exported graph and checkpoints.
- `serving_input_fn`: A function that takes no argument and returns an `InputFnOps`.
- `default_output_alternative_key`: the name of the head to serve when none is specified. Not needed for single-headed models.
- `assets_extra`: A dict specifying how to populate the assets.extra directory within the exported SavedModel. Each key should give the destination path (including the filename) relative to the assets.extra directory. The corresponding value gives the full path of the source file to be copied. For example, the simple case of copying a single file without renaming it is specified as `{'my_asset_file.txt': '/path/to/my_asset_file.txt'}`.
- `as_text`: whether to write the SavedModel proto in text format.
- `checkpoint_path`: The checkpoint path to export. If None (the default), the most recent checkpoint found within the model directory is chosen.
- `graph_rewrite_specs`: an iterable of `GraphRewriteSpec`. Each element will produce a separate MetaGraphDef within the exported SavedModel, tagged and rewritten as specified. Defaults to a single entry using the default serving tag ("serve") and no rewriting.

Returns:

The string path to the exported directory.

Raises:

- `ValueError`: if an unrecognized export_type is requested.

`export_with_defaults`

```
export_with_defaults(
    export_dir,
    signature_fn=None,
    input_fn=None,
    default_batch_size=1,
    exports_to_keep=None
)
```

Same as BaseEstimator.export, but uses some defaults. (deprecated)

THIS FUNCTION IS DEPRECATED. It will be removed after 2017-03-25. Instructions for updating: Please use Estimator.export_savedmodel() instead.

`fit`

```
fit(
    x=None,
    y=None,
    input_fn=None,
    steps=None,
    batch_size=None,
    monitors=None,
    max_steps=None
)
```

See `Trainable`. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-12-01. Instructions for updating: Estimator is

decoupled from Scikit Learn interface by moving into separate class SKCompat. Arguments x, y and batch_size are only available in the SKCompat class, Estimator will only accept input_fn. Example conversion: est = Estimator(...) -> est = SKCompat(Estimator(...))

Raises:

- `ValueError`: If `x` or `y` are not `None` while `input_fn` is not `None`.
- `ValueError`: If both `steps` and `max_steps` are not `None`.

`get_params`

```
get_params(deep=True)
```

Get parameters for this estimator.

Args:

- `deep`: boolean, optional

  If `True`, will return the parameters for this estimator and contained subobjects that are estimators.

Returns:

- `params`: mapping of string to any Parameter names mapped to their values.

`get_variable_names`

```
get_variable_names()
```

Returns list of all variable names in this model.

Returns:

List of names.

`get_variable_value`

```
get_variable_value(name)
```

Returns value of the variable given by name.

Args:

- `name`: string, name of the tensor.

Returns:

Numpy array - value of the tensor.

`partial_fit`

```
partial_fit(
    x=None,
    y=None,
    input_fn=None,
    steps=1,
    batch_size=None,
    monitors=None
)
```

Incremental fit on a batch of samples. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-12-01. Instructions for updating: Estimator is decoupled from Scikit Learn interface by moving into separate class SKCompat. Arguments x, y and batch_size are only available in the SKCompat class, Estimator will only accept input_fn. Example conversion: est = Estimator(...) -> est = SKCompat(Estimator(...))

This method is expected to be called several times consecutively on different or the same chunks of the dataset. This either can implement iterative training or out-of-core/online training.

This is especially useful when the whole dataset is too big to fit in memory at the same time. Or when model is taking long time to converge, and you want to split up training into subparts.

Args:

- `x`: Matrix of shape [n_samples, n_features...]. Can be iterator that returns arrays of features. The training input samples for fitting the model. If set, `input_fn` must be `None`.
- `y`: Vector or matrix [n_samples] or [n_samples, n_outputs]. Can be iterator that returns array of labels. The training label values (class labels in classification, real numbers in regression). If set, `input_fn` must be `None`.
- `input_fn`: Input function. If set, `x`, `y`, and `batch_size` must be `None`.
- `steps`: Number of steps for which to train model. If `None`, train forever.
- `batch_size`: minibatch size to use on the input, defaults to first dimension of `x`. Must be `None` if `input_fn` is provided.
- `monitors`: List of `BaseMonitor` subclass instances. Used for callbacks inside the training loop.

Returns:

`self`, for chaining.

Raises:

- `ValueError`: If at least one of `x` and `y` is provided, and `input_fn` is provided.

predict

```
predict(
    x=None,
    input_fn=None,
    batch_size=None,
    outputs=None,
    as_iterable=True
)
```

Returns predictions for given features. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-12-01. Instructions for updating: Estimator is decoupled from Scikit Learn interface by moving into separate class SKCompat. Arguments x, y and batch_size are only available in the SKCompat class, Estimator will only accept input_fn. Example conversion: est = Estimator(...) -> est = SKCompat(Estimator(...))

Args:

- `x`: Matrix of shape [n_samples, n_features...]. Can be iterator that returns arrays of features. The training input samples for fitting the model. If set, `input_fn` must be `None`.
- `input_fn`: Input function. If set, `x` and 'batch_size' must be `None`.
- `batch_size`: Override default batch size. If set, 'input_fn' must be 'None'.
- `outputs`: list of `str`, name of the output to predict. If `None`, returns all.
- `as_iterable`: If True, return an iterable which keeps yielding predictions for each example until inputs are exhausted. Note: The inputs must terminate if you want the iterable to terminate (e.g. be sure to pass num_epochs=1 if you are using something like read_batch_features).

Returns:

A numpy array of predicted classes or regression values if the constructor's `model_fn` returns a `Tensor` for `predictions` or a `dict` of numpy arrays if `model_fn` returns a `dict`. Returns an iterable of predictions if as_iterable is True.

Raises:

- `ValueError`: If x and input_fn are both provided or both `None`.

`predict_classes`

```
predict_classes(
    x=None,
    input_fn=None,
    batch_size=None,
    as_iterable=True
)
```

Runs inference to determine the predicted class. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-09-15. Instructions for updating: The default behavior of predict() is changing. The default value for as_iterable will change to True, and then the flag will be removed altogether. The behavior of this flag is described below.

`predict_proba`

```
predict_proba(
    x=None,
    input_fn=None,
    batch_size=None,
    outputs=None,
    as_iterable=True
)
```

Runs inference to determine the class probability predictions. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed after 2016-09-15. Instructions for updating: The default

behavior of predict() is changing. The default value for as_iterable will change to True, and then the flag will be removed altogether. The behavior of this flag is described below.

set_params

```
set_params(**params)
```

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The former have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Args:

- `**params`: Parameters.

Returns:

self

Raises:

- `ValueError`: If params contain invalid names.

Last updated November 2, 2017.