tf.contrib.learn.MetricSpec

Contents

Class MetricSpec
Properties
label_key
metric_fn

Class `MetricSpec`

Defined in `tensorflow/contrib/learn/python/learn/metric_spec.py`.

See the guide: Learn (contrib) > Estimators

MetricSpec connects a model to metric functions.

The MetricSpec class contains all information necessary to connect the output of a `model_fn` to the metrics (usually, streaming metrics) that are used in evaluation.

It is passed in the `metrics` argument of `Estimator.evaluate`. The `Estimator` then knows which predictions, labels, and weight to use to call a given metric function.

When building the ops to run in evaluation, an `Estimator` will call `create_metric_ops`, which will connect the given `metric_fn` to the model as detailed in the docstring for `create_metric_ops`, and return the metric.

Example:

Assuming a model has an input function which returns inputs containing (among other things) a tensor with key "input_key", and a labels dictionary containing "label_key". Let's assume that the `model_fn` for this model returns a prediction with key "prediction_key".

In order to compute the accuracy of the "prediction_key" prediction, we would add

```
"prediction accuracy": MetricSpec(metric_fn=prediction_accuracy_fn,
                                  prediction_key="prediction_key",
                                  label_key="label_key")
```

to the metrics argument to `evaluate`. `prediction_accuracy_fn` can be either a predefined function in metric_ops (e.g., `streaming_accuracy`) or a custom function you define.

If we would like the accuracy to be weighted by "input_key", we can add that as the `weight_key` argument.

```
"prediction accuracy": MetricSpec(metric_fn=prediction_accuracy_fn,
                                  prediction_key="prediction_key",
                                  label_key="label_key",
                                  weight_key="input_key")
```

An end-to-end example is as follows:

```
estimator = tf.contrib.learn.Estimator(...)
estimator.fit(...)
_ = estimator.evaluate(
    input_fn=input_fn,
    steps=1,
    metrics={
        'prediction accuracy':
            metric_spec.MetricSpec(
                metric_fn=prediction_accuracy_fn,
                prediction_key="prediction_key",
                label_key="label_key")
    })
```

Properties

`label_key`

`metric_fn`

Metric function.

This function accepts named args: `predictions`, `labels`, `weights`. It returns a single `Tensor` or `(value_op, update_op)` pair. See `metric_fn` constructor argument for more details.

Returns:

Function, see `metric_fn` constructor argument for more details.

`prediction_key`

`weight_key`

Methods

`__init__`

```
__init__(
    metric_fn,
    prediction_key=None,
    label_key=None,
    weight_key=None
)
```

Constructor.

Creates a MetricSpec.

Args:

- `metric_fn`: A function to use as a metric. See `_adapt_metric_fn` for rules on how `predictions`, `labels`, and `weights` are passed to this function. This must return either a single `Tensor`, which is interpreted as a value of this metric, or a pair `(value_op, update_op)`, where `value_op` is the op to call to obtain the value of the metric, and `update_op` should be run for each batch to update internal state.
- `prediction_key`: The key for a tensor in the `predictions` dict (output from the `model_fn`) to use as the

`predictions` input to the `metric_fn`. Optional. If `None`, the `model_fn` must return a single tensor or a dict with only a single entry as `predictions`.

- `label_key`: The key for a tensor in the `labels` dict (output from the `input_fn`) to use as the `labels` input to the `metric_fn`. Optional. If `None`, the `input_fn` must return a single tensor or a dict with only a single entry as `labels`.
- `weight_key`: The key for a tensor in the `inputs` dict (output from the `input_fn`) to use as the `weights` input to the `metric_fn`. Optional. If `None`, no weights will be passed to the `metric_fn`.

`create_metric_ops`

```
create_metric_ops(
    inputs,
    labels,
    predictions
)
```

Connect our `metric_fn` to the specified members of the given dicts.

This function will call the `metric_fn` given in our constructor as follows:

```
  metric_fn(predictions[self.prediction_key],
            labels[self.label_key],
            weights=weights[self.weight_key])
```

And returns the result. The `weights` argument is only passed if `self.weight_key` is not `None`.

`predictions` and `labels` may be single tensors as well as dicts. If `predictions` is a single tensor, `self.prediction_key` must be `None`. If `predictions` is a single element dict, `self.prediction_key` is allowed to be `None`. Conversely, if `labels` is a single tensor, `self.label_key` must be `None`. If `labels` is a single element dict, `self.label_key` is allowed to be `None`.

Args:

- `inputs`: A dict of inputs produced by the `input_fn`
- `labels`: A dict of labels or a single label tensor produced by the `input_fn`.
- `predictions`: A dict of predictions or a single tensor produced by the `model_fn`.

Returns:

The result of calling `metric_fn`.

Raises:

- `ValueError`: If `predictions` or `labels` is a single `Tensor` and `self.prediction_key` or `self.label_key` is not `None`; or if `self.label_key` is `None` but `labels` is a dict with more than one element, or if `self.prediction_key` is `None` but `predictions` is a dict with more than one element.
