# tf.contrib.timeseries.ARModel

Contents



## Class **ARModel**

Defined in `tensorflow/contrib/timeseries/python/timeseries/ar_model.py`.

Auto-regressive model, both linear and non-linear.

Features to the model include time and values of input_window_size timesteps, and times for output_window_size timesteps. These are passed through zero or more hidden layers, and then fed to a loss function (e.g. squared loss).

Note that this class can also be used to regress against time only by setting the input_window_size to zero.

## Methods

### `__init__`

```
__init__(
    periodicities,
    input_window_size,
    output_window_size,
    num_features,
    num_time_buckets=10,
    loss=NORMAL_LIKELIHOOD_LOSS,
    hidden_layer_sizes=None
)
```

Constructs an auto-regressive model.

#### Args:

- `periodicities`: periodicities of the input data, in the same units as the time feature. Note this can be a single value or a list of values for multiple periodicities.
- `input_window_size`: Number of past time steps of data to look at when doing the regression.
- `output_window_size`: Number of future time steps to predict. Note that setting it to > 1 empiricaly seems to give a better fit.
- `num_features`: number of input features per time step.
- `num_time_buckets`: Number of buckets into which to divide (time % periodicity) for generating time based features.
- `loss`: Loss function to use for training. Currently supported values are SQUARED_LOSS and

NORMAL_LIKELIHOOD_LOSS. Note that for NORMAL_LIKELIHOOD_LOSS, we train the covariance term as well. For SQUARED_LOSS, the evaluation loss is reported based on un-scaled observations and predictions, while the training loss is computed on normalized data (if input statistics are available).

- `hidden_layer_sizes`: list of sizes of hidden layers.

### `define_loss`

```
define_loss(
    features,
    mode
)
```

Default loss definition with state replicated across a batch.

Time series passed to this model have a batch dimension, and each series in a batch can be operated on in parallel. This loss definition assumes that each element of the batch represents an independent sample conditioned on the same initial state (i.e. it is simply replicated across the batch). A batch size of one provides sequential operations on a single time series.

More complex processing may operate instead on get_start_state() and get_batch_loss() directly.

Args:

- `features`: A dictionary (such as is produced by a chunker) with at minimum the following key/value pairs (others corresponding to the `exogenous_feature_columns` argument to `__init__` may be included representing exogenous regressors):
- `TrainEvalFeatures.TIMES`: A [batch size x window size] integer Tensor with times for each observation. If there is no artificial chunking, the window size is simply the length of the time series.
- `TrainEvalFeatures.VALUES`: A [batch size x window size x num features] Tensor with values for each observation.
- `mode`: The tf.estimator.ModeKeys mode to use (TRAIN, EVAL). For INFER, see predict().

Returns:

A ModelOutputs object.

### `generate`

```
generate(
    number_of_series,
    series_length,
    model_parameters=None,
    seed=None
)
```

### `get_batch_loss`

```
get_batch_loss(
    features,
    mode,
    state
)
```

Computes predictions and a loss.

Args:

- `features`: A dictionary (such as is produced by a chunker) with the following key/value pairs (shapes are given as required for training): TrainEvalFeatures.TIMES: A [batch size, self.window_size] integer Tensor with times for each observation. To train on longer sequences, the data should first be chunked. TrainEvalFeatures.VALUES: A [batch size, self.window_size, self.num_features] Tensor with values for each observation. When evaluating, `TIMES` and `VALUES` must have a window size of at least self.window_size, but it may be longer, in which case the last window_size - self.input_window_size times (or fewer if this is not divisible by self.output_window_size) will be evaluated on with non-overlapping output windows (and will have associated predictions). This is primarily to support qualitative evaluation/plotting, and is not a recommended way to compute evaluation losses (since there is no overlap in the output windows, which for window-based models is an undesirable bias).
- `mode`: The tf.estimator.ModeKeys mode to use (TRAIN or EVAL).
- `state`: Unused

Returns:

A model.ModelOutputs object.

Raises:

- `ValueError`: If `mode` is not TRAIN or EVAL, or if static shape information is incorrect.

### `get_start_state`

```
get_start_state()
```

### `initialize_graph`

```
initialize_graph(input_statistics=None)
```

Define ops for the model, not depending on any previously defined ops.

Args:

- `input_statistics`: A math_utils.InputStatistics object containing input statistics. If None, data-independent defaults are used, which may result in longer or unstable training.

### `loss_op`

```
loss_op(
    targets,
    prediction_ops
)
```

Create loss_op.

### `predict`

```
predict(features)
```

Computes predictions multiple steps into the future.

Args:

- `features`: A dictionary with the following key/value pairs:
- `PredictionFeatures.TIMES`: A [batch size, predict window size] integer Tensor of times, after the window of data indicated by `STATE_TUPLE`, to make predictions for.
- `PredictionFeatures.STATE_TUPLE`: A tuple of (times, values), times with shape [batch size, self.input_window_size], values with shape [batch size, self.input_window_size, self.num_features] representing a segment of the time series before `TIMES`. This data is used to start of the autoregressive computation. This should have data for at least self.input_window_size timesteps.

Returns:

A dictionary with keys, "mean", "covariance". The values are Tensors of shape [batch_size, predict window size, num_features] and correspond to the values passed in `TIMES`.

### `prediction_ops`

```
prediction_ops(
    times,
    values
)
```

Compute model predictions given input data.

Args:

- `times`: A [batch size, self.window_size] integer Tensor, the first self.input_window_size times in each part of the batch indicating input features, and the last self.output_window_size times indicating prediction times.
- `values`: A [batch size, self.input_window_size, self.num_features] Tensor with input features.

Returns:

Tuple (predicted_mean, predicted_covariance), where each element is a Tensor with shape [batch size, self.output_window_size, self.num_features].

### `random_model_parameters`

```
random_model_parameters(seed=None)
```

## Class Members

### `NORMAL_LIKELIHOOD_LOSS`

### `SQUARED_LOSS`

**Stay Connected**

Blog

GitHub

Twitter

**Support**

Issue Tracker

Release Notes

Stack Overflow

English

**Terms** | **Privacy**