tf.contrib.linear_optimizer.SdcaModel

Class **`SdcaModel`**

Defined in `tensorflow/contrib/linear_optimizer/python/ops/sdca_ops.py`.

Stochastic dual coordinate ascent solver for linear models.

This class currently only supports a single machine (multi-threaded) implementation. We expect the weights and duals to fit in a single machine.

Loss functions supported:

- Binary logistic loss
- Squared loss
- Hinge loss
- Smooth hinge loss

This class defines an optimizer API to train a linear model.

Usage

```
# Create a solver with the desired parameters.
lr = tf.contrib.linear_optimizer.SdcaModel(examples, variables, options)
min_op = lr.minimize()
opt_op = lr.update_weights(min_op)

predictions = lr.predictions(examples)
# Primal loss + L1 loss + L2 loss.
regularized_loss = lr.regularized_loss(examples)
# Primal loss only
unregularized_loss = lr.unregularized_loss(examples)

examples: {
  sparse_features: list of SparseFeatureColumn.
  dense_features: list of dense tensors of type float32.
  example_labels: a tensor of type float32 and shape [Num examples]
  example_weights: a tensor of type float32 and shape [Num examples]
  example_ids: a tensor of type string and shape [Num examples]
}
variables: {
  sparse_features_weights: list of tensors of shape [vocab size]
  dense_features_weights: list of tensors of shape [dense_feature_dimension]
}
options: {
  symmetric_l1_regularization: 0.0
  symmetric_l2_regularization: 1.0
  loss_type: "logistic_loss"
  num_loss_partitions: 1 (Optional, with default value of 1. Number of
  partitions of the global loss function, 1 means single machine solver,
  and >1 when we have more than one optimizer working concurrently.)
  num_table_shards: 1 (Optional, with default value of 1. Number of shards
  of the internal state table, typically set to match the number of
  parameter servers for large data sets.
}
```

In the training program you will just have to run the returned Op from minimize().

```
# Execute opt_op and train for num_steps.
for _ in range(num_steps):
  opt_op.run()

# You can also check for convergence by calling
lr.approximate_duality_gap()
```

Methods

__init__

```
__init__(
    examples,
    variables,
    options
)
```

Create a new sdca optimizer.

approximate_duality_gap

```
approximate_duality_gap()
```

Add operations to compute the approximate duality gap.

Returns:

An Operation that computes the approximate duality gap over all examples.

`minimize`

```
minimize(
    global_step=None,
    name=None
)
```

Add operations to train a linear model by minimizing the loss function.

Args:

- `global_step`: Optional `Variable` to increment by one after the variables have been updated.
- `name`: Optional name for the returned operation.

Returns:

An Operation that updates the variables passed in the constructor.

`predictions`

```
predictions(examples)
```

Add operations to compute predictions by the model.

If logistic_loss is being used, predicted probabilities are returned. Otherwise, (raw) linear predictions (w*x) are returned.

Args:

- `examples`: Examples to compute predictions on.

Returns:

An Operation that computes the predictions for examples.

Raises:

- `ValueError`: if examples are not well defined.

`regularized_loss`

```
regularized_loss(examples)
```

Add operations to compute the loss with regularization loss included.

Args:

- `examples`: Examples to compute loss on.

Returns:

An Operation that computes mean (regularized) loss for given set of examples.

Raises:

- `ValueError`: if examples are not well defined.

`unregularized_loss`

```
unregularized_loss(examples)
```

Add operations to compute the loss (without the regularization loss).

Args:

- `examples`: Examples to compute unregularized loss on.

Returns:

An Operation that computes mean (unregularized) loss for given set of examples.

Raises:

- `ValueError`: if examples are not well defined.

`update_weights`

```
update_weights(train_op)
```

Updates the model weights.

This function must be called on at least one worker after **`minimize`**. In distributed training this call can be omitted on non-chief workers to speed up training.

Args:

- `train_op`: The operation returned by the **`minimize`** call.

Returns:

An Operation that updates the model weights.