tf.contrib.factorization.KMeans

Contents

Class **KMeans**

Defined in `tensorflow/contrib/factorization/python/ops/clustering_ops.py`.

Creates the graph for k-means clustering.

Methods

`__init__`

```
__init__(
    inputs,
    num_clusters,
    initial_clusters=RANDOM_INIT,
    distance_metric=SQUARED_EUCLIDEAN_DISTANCE,
    use_mini_batch=False,
    mini_batch_steps_per_iteration=1,
    random_seed=0,
    kmeans_plus_plus_num_retries=2
)
```

Creates an object for generating KMeans clustering graph.

This class implements the following variants of K-means algorithm:

If use_mini_batch is False, it runs standard full batch K-means. Each step runs a single iteration of K-Means. This step can be run sharded across multiple workers by passing a list of sharded inputs to this class. Note however that a single step needs to process the full input at once.

If use_mini_batch is True, it runs a generalization of the mini-batch K-means algorithm. It runs multiple iterations, where each iteration is composed of mini_batch_steps_per_iteration steps. Two copies of cluster centers are maintained: one that is updated at the end of each iteration, and one that is updated every step. The first copy is used to compute cluster allocations for each step, and for inference, while the second copy is the one updated each step using the mini-batch update rule. After each iteration is complete, this second copy is copied back the first copy.

Note that for use_mini_batch=True, when mini_batch_steps_per_iteration=1, the algorithm reduces to the standard mini-batch algorithm. Also by setting mini_batch_steps_per_iteration = num_inputs / batch_size, the algorithm becomes an asynchronous version of the full-batch algorithm. Note however that there is no guarantee by this implementation that each input is seen exactly once per iteration. Also, different updates are applied asynchronously without locking. So this asynchronous version may not behave exactly like a full-batch version.

Args:

- `inputs`: An input tensor or list of input tensors
- `num_clusters`: An integer tensor specifying the number of clusters. This argument is ignored if initial_clusters is a tensor or numpy array.
- `initial_clusters`: Specifies the clusters used during initialization. One of the following:
  - a tensor or numpy array with the initial cluster centers.
  - a function f(inputs, k) that returns up to k centers from `inputs`.
  - "random": Choose centers randomly from `inputs`.
  - "kmeans_plus_plus": Use kmeans++ to choose centers from `inputs`. In the last three cases, one batch of `inputs` may not yield `num_clusters` centers, in which case initialization will require multiple batches until enough centers are chosen. In the case of "random" or "kmeans_plus_plus", if the input size is <= `num_clusters` then the entire batch is chosen to be cluster centers.
- `distance_metric`: Distance metric used for clustering. Supported options: "squared_euclidean", "cosine".
- `use_mini_batch`: If true, use the mini-batch k-means algorithm. Else assume full batch.
- `mini_batch_steps_per_iteration`: Number of steps after which the updated cluster centers are synced back to a master copy.
- `random_seed`: Seed for PRNG used to initialize seeds.
- `kmeans_plus_plus_num_retries`: For each point that is sampled during kmeans++ initialization, this parameter specifies the number of additional points to draw from the current distribution before selecting the best. If a negative value is specified, a heuristic is used to sample O(log(num_to_sample)) additional points.

Raises:

- `ValueError`: An invalid argument was passed to initial_clusters or distance_metric.

`training_graph`

```
training_graph()
```

Generate a training graph for kmeans algorithm.

This returns, among other things, an op that chooses initial centers (init_op), a boolean variable that is set to True when the initial centers are chosen (cluster_centers_initialized), and an op to perform either an entire Lloyd iteration or a mini-batch of a Lloyd iteration (training_op). The caller should use these components as follows. A single worker should execute init_op multiple times until cluster_centers_initialized becomes True. Then multiple workers may execute training_op any number of times.

Returns:

A tuple consisting of: `all_scores`*: A matrix (or list of matrices) of dimensions (num_input, num_clusters) where the value is the distance of an input vector and a cluster center.* `cluster_idx`: A vector (or list of vectors). Each element in the vector corresponds to an input row in 'inp' and specifies the cluster id corresponding to the input. `scores`*: Similar to cluster_idx but specifies the distance to the assigned cluster instead.* `cluster_centers_initialized`: scalar indicating whether clusters have been initialized. `cluster_centers_var`*: a Variable holding the cluster centers.* `init_op`: an op to initialize the clusters. * `training_op`: an op that runs an iteration of training.

Last updated November 2, 2017.