Stochastic Gradient Descent

MokkeMeguru[1]

2020-02-12 Wed

[1]meguru.mokke@gmail.com

Outline

Presentaion agenda

Stochastic Gradient Descent

Stochastic Gradient Descent (確率的勾配降下法)

パラメータ更新法の一つ
* おそらく最もシンプルな手法
⇒ 本当に最適なパラメータ (最適解) に収束するかは…

$$w := w - \eta \nabla Q(w)$$

, *where*

$$w \quad \text{is} \quad \text{weights}$$

$$Q(w) \quad \text{is} \quad \text{objective function}$$

パラメータを、目的関数 $Q(w)$ で求められる勾配 $\nabla Q(w)$ を用いて、更新率 η で更新する。

パラメータ更新法色々

- Adam, AdaGrad, RMSProp, etc.
  Wikipedia を皮切りに探せる
- Tensorflow.train.XXXOptimizer
  公式のページ, 公式のページ (tf2.x)
- 最近のホットな更新法
  AdaBound / AMSBound (tf2.x には今度実装します)
- 基本的には Adam を使っておくと良い

SGD でパラメータを求める I

```
1 def teacher(x: np.float32):
2     y = 5.0 * x + 8.0
3     return y
```

求める式, パラメータ

$$y = ax + b$$

, where

$$a = 5.0$$
$$b = 8.0$$

SGD でパラメータを求める II

モデルの定義

```
1  def simple_model(x: tf.Tensor, scope: str = 'simple_model'):
2      weights_scope = scope + '/weights'
3      with tf.variable_scope(weights_scope):
4          a = tf.get_variable(
5              name=weights_scope + '_a',
6              shape=[], dtype=tf.float32,
7              initializer=tf.initializers.constant(
8                  value=0.0, dtype=tf.float32),
9              trainable=True)
10         b = tf.get_variable(
11             name=weights_scope + '_b',
12             shape=[], dtype=tf.float32,
13             initializer=tf.initializers.constant(
14                 value=0.0, dtype=tf.float32),
15             trainable=True)
16     with tf.name_scope(scope + '/formula'):
17         y = a * x + b
18     with tf.name_scope(scope + '/log'):
19         log_op = tf.strings.format('a - {} / b - {}', [a, b])
20     return y, log_op
```

SGD でパラメータを求める III

目的関数と勾配

`loss_op` は損失関数 $\in$ 目的関数

```
1  y_hat, log_op = simple_model(x)
2  loss_op = tf.math.squared_difference(y, y_hat)
3
4  # 更新率
5  learning_rate = 1e-3
6
7  # Stochastic Gradient Descent
8  optimizer = tf.train.GradientDescentOptimizer(
9      learning_rate=learning_rate)
10
11 # 勾配を用いてパラメータを更新する
12 train_step = optimizer.minimize(loss_op)
```

Presentaion agenda

なぜ スコープ がいるか

我々の要望

- パラメータを管理したい
- 同じパラメータを再利用したい
- 同じようなコードを何度も書きたくない。

スコープを用いる例: 重み

先程のコードから引用

a とは simple_model/weights_a という場所にある値
scope = 'simple_model2' のとき、
simpel_model2/weights_a という場所 ⇒ 別の値

```
1  scope = 'simple_model'
2  weights_scope = scope + '/weights'
3  with tf.variable_scope(weights_scope):
4      a = tf.get_variable(
5          name=weights_scope + '_a',
6          shape=[], dtype=tf.float32,
7          initializer=tf.initializers.constant(
8              value=0.0, dtype=tf.float32),
9          trainable=True)
10     b = tf.get_variable(
11         name=weights_scope + '_b',
12         shape=[], dtype=tf.float32,
13         initializer=tf.initializers.constant(
14             value=0.0, dtype=tf.float32),
15         trainable=True)
```

スコープを用いる例: 演算など

重みではないけど管理したいものを囲っておく
→ 計算グラフの可視化

```
1 with tf.name_scope(scope + '/formula'):
2     y = a * x + b
3 with tf.name_scope(scope + '/log'):
4     log_op = tf.strings.format('a - {} / b - {}', [a, b])
```

Presentaion agenda

計算グラフの可視化

複雑なコードは動作を追うのが難しい

- Define by Run → すこしずつ実行して見れば良い
- Define and Run → 計算グラフの構築と実行が別

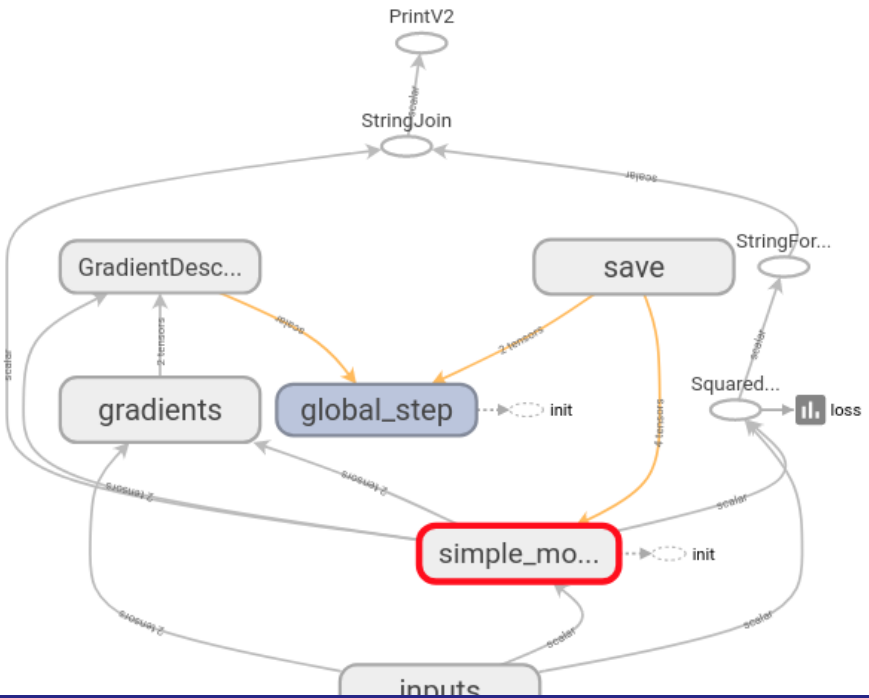
何からの手段で計算グラフをわかりやすく確認する必要がある
⇒ 視覚化

計算グラフの可視化: コード

```
1  path = './tmp/sgd'
2  # ここまでに定義したすべての変数を初期化するための演算
3  init_op = tf.global_variables_initializer()
4
5  with tf.Session() as sess:
6      # 実際に全部初期化する
7      sess.run(init_op)
8      # path の位置にログを残すためのソケット writer を作る
9      # + 計算グラフを描く
10     writer = tf.summary.FileWriter(path, sess.graph)
```

計算グラフの可視化: 可視化結果

`pipenv run tensorboard --logdir path/to/tmp/sgd` をターミナルで実行 `https://localhost:6006` をブラウザで確認する。



Presentaion agenda

課題

1. sgd_function.py を 実行し、標準出力から a, b, loss の遷移を確認せよ。
2. sgd_function.py から path を見つけ、計算グラフの可視化を行ってみよ
3. ブラウザから計算グラフを確認し、 `formula` を探し出せ