Project: More Experiments on Stochastic Gradient Methods

Last updated: March 20, 2021
Goal

- We want to know more about the internal details of simpleNN.
- We want to roughly compare the two stochastic gradient approaches: SG with momentum and Adam.
In our code, stochastic gradient is implemented in a subroutine `gradient_trainer` in `train.py`. You can see a for loop there.

```python
for epoch in range(0, args.epoch):
    ...
    for i in range(num_iters):
        ...
        step, _, batch_loss = sess.run(
            [global_step, optimizer, loss_with_reg],
            feed_dict = {x: batch_input, y: batch_labels, learning_rate: lr})
```

March 20, 2021 3 / 17
The optimizer was specified earlier:

```python
optimizer = tf.compat.v1.train.MomentumOptimizer(
    learning_rate=learning_rate,
    momentum=config.momentum).minimize(
    loss_with_reg,
    global_step=global_step)
```

- It happened that we run the SG steps by ourself, but in Tensorflow there must be a way so that stochastic gradient methods can be directly called in one statement
That is, for a typical user of tensorflow, they would call

\[ \text{train.MomentumOptimizer} \]

once without the for loop

We would like to check if under the same initial model, the two settings give the same results

To check “the same results” you can, for example, compare their models at each iteration or compare their objective values

Therefore, for this part of the project you only need to run very few iterations (e.g., 5)
Further, we should use the simplest setting: SG without momentum

You can print out weight values for the comparison

If you face difficulties, consider to simplify your settings for debugging:

- Use a small set of data (e.g., data/mnist-demo.mat) or even a subset of just 100 instances
- Enlarge --bsize to be the same as the number of data. Then essentially you do gradient descent
Project Contents: First Part V

- We will separately discuss
  - modification of simpleNN, and
  - direct use of Tensorflow in subsequent slides
- The regularization term may be a concern. Need to make sure that the two settings minimize the same objective function
- For this project, you definitely need to trace the subroutine `gradient_trainer` in `train.py`.
One issue is that in the beginning of each update, we randomly select instances as the current batch:

```python
idx = np.random.choice(
    np.arange(0, num_data),
    size=config.bsize, replace=False)
```

Tensorflow doesn’t do that so you can replace the code with

```python
idx = np.arange(i*config.bsize,
    min((i+1)*config.bsize, num_data))
```

The `min` operation handles the situation if number of data is not a multiple of the batch size.
Direct Use of Tensorflow MomentumOptimizer

- The workflow should be like this
  - Specify the network
    ```python
    model = ...
    ```
  - Specify the optimizer
    ```python
    model.compile(optimizer = ...)
    ```
  - Do the training
    ```python
    model.fit = ...
    ```
- To specify the network, CNN cannot be directly used
Instead you can directly do it in the subroutine gradient_trainer

Here we provide the code

```python
model = CNN_model(config.net, config.dim, config.num_cls)
```

You need to change the line

```python
param = tf.compat.v1.trainable_variables()
```

to

```python
param = model.trainable_weights
```
CNN and CNN\_model both use global variables, so we specify which to use to avoid variable conflicts. Note that there are two such places in gradient\_trainer() and you need to change both.

- For calculating the objective value, you need to replace

\[
\text{loss\_with\_reg} = \text{reg\_const}\times\text{reg} + \frac{\text{loss}}{\text{batch\_size}}
\]

with
Direct Use of Tensorflow MomentumOptimizer

```
loss_with_reg = lambda y_true, y_pred:
    reg_const*reg + tf.reduce_mean(tf.reduce_sum(
        tf.square(y_true - y_pred), axis=1))
```

- We no longer have the outputs of the model, so the loss can’t be calculated directly
- Instead we use some Tensorflow functions to calculate the objective value
- For the use of MomentumOptimizer you should check Tensorflow manual in detail
This is what we want you to learn

- There are no restrictions on the data set to be used in this part. Even mnist-demo is fine. You can use any data you want.

- We’ve modified the net.py to make it easier for everyone to do this project. We will also be constantly improving simpleNN. Please constantly git pull the latest version.
We want to check the test accuracy of two stochastic gradient methods: SG with momentum and Adam.

Note that in the first project, what we used is the simplest SG without momentum.

We also hope to roughly check the parameter sensitivity.

Under each parameter setting, we run a large number (e.g., 500) of iterations and use the model at the last iteration.
We do not use a model before the last iteration because a validation process was not conducted.

Please work on the same MNIST and CIFAR10 data sets used in the previous project.

In your report, give your results, observations and thoughts.

In the previous project, we used only default parameters.

You can slightly vary parameters (e.g., learning rate in SGD and Adam) and check the test accuracy.
Due to the lengthy running time, no need to try many parameter settings

Remember we don’t judge you solely by your accuracy
Students selected for presentation please do a 10-minute talk (9-minute the contents and 1-minute Q&A)