리뷰 - Learning to Learn by Gradient Descent by Gradient Descent

Created by 임 도형, last modified on Apr 22, 2018

개요

DL에 사용되는 optimizer 자체를 RNN으로 학습하자.

방법 설명

일반적인 DL의 optimizer는 고정되어 있다.

그러지 말고 optimizer자체도 학습시키자.

이를 위해 optimizer를 RNN으로 한다.

기존의 방식은 단순 gradient를 그냥 업데이트 했는데

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

대신 gradient를 입력으로 하는 g()를 학습시키고, 그 결과로 업데이트 하자.

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$$

일반화된 gradient 학습 방법

기존의 NN의 학습방법은 다음과 같이 일반화된 방법으로 기술할 수 있다.

임의의 optimizer m이 있다.

optimizer는 3개를 입력으로 한다.

- gradient (∇)
- optimizer 내부 상태(h_t)
- ???

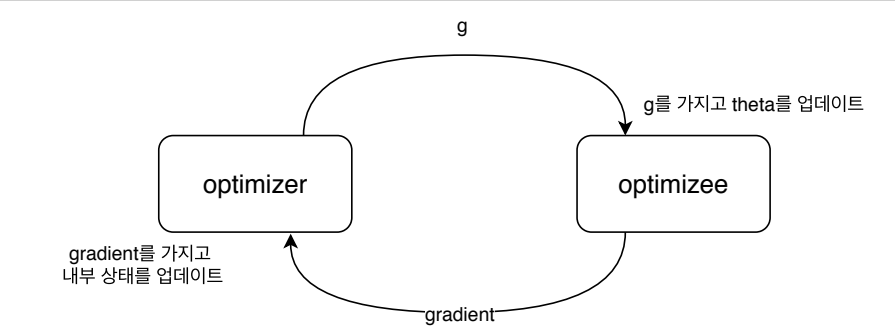
그리고 2개를 출력한다.

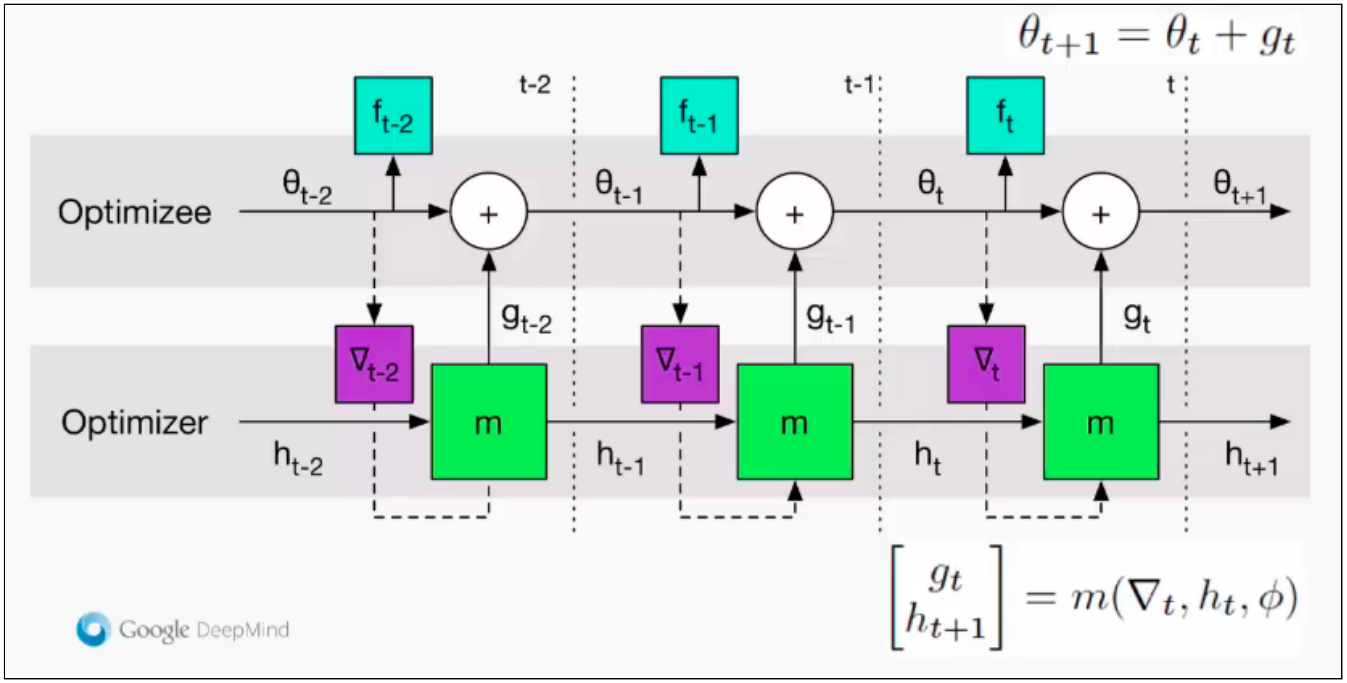
- optimizer의 다음 내부상태(h_{t+1})
- 문제(optimizee)를 업데이트 할 양(g_t)

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

optimizer와 optimizee는 다음을 반복하며 optimizee의 theta를 업데이트 한다.

- optimizer update : optimizee에 의한 gradient를 가지고 optimizer의 상태를 업데이트. 그리고 g를 구한다.
- opimizee update : optimizer에 의한 g를 가지고 optimizee의 상태를 업데이트. 그리고 gradient를 구한다.





f

입력으로 출력을 내보내는 NN자체. 함수를 의미.

y = f(x)로 표현된다.

f는 theta로 구성된다.

theta(θ)

theta는 우리가 풀어야 할 NN의 웨이트.

기존 NN을 학습시킨다는 것은 손실함수를 최소로 하는 theta를 찾아내는 것.

학습의 목표 혹은 문제 해결은 theta를 찾아내는 것.

g

optimizer에 의해 계산된 theta의 업데이트 할 양.

$$\theta_{t+1} = \theta_t + g_t$$

gradient(∇)

현재 theta일 때에 해결할 문제(optimizee)의 경사.

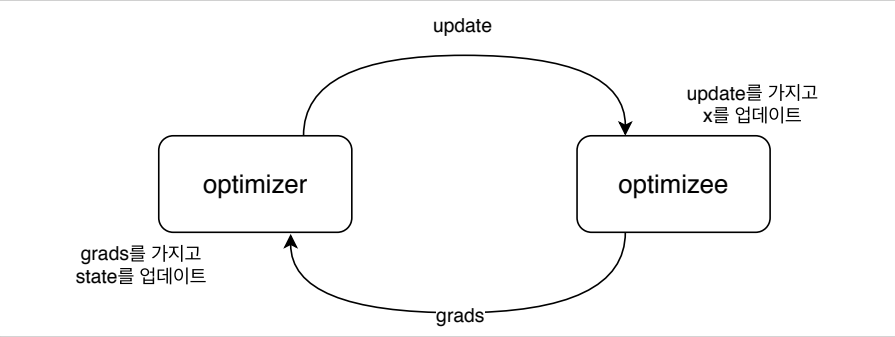
모든 가능한 x에 대하여 값을 구하는 듯... 한데.

코드를 사용한 예시

https://github.com/LlionJ/blog/blob/master/blog/Meta-Learning.ipynb에서 발췌함.

학습 루프.

```
def learn(optimizer):
    losses = []
    state = None
    for _ in range(TRAINING_STEPS):
        loss = f(x)
        grads, = tf.gradients(loss, x)
        update, state = optimizer(grads, state)
        x += update
    return losses
```



f(x)는 일반적인 문제를 의미.

구체적으로 코스트 펑션이라 보면되고, 그 출력이 최소가 되는 x를 찾는 것이 목적이다.

코드에서는 다음과 같은 10차원 포물선을 예로 하였다. 원점에서 최소값 0을 갖는다.

```
DIMS = 10  # Dimensions of the parabola
scale = tf.random_uniform([DIMS], 0.5, 1.5)
# This represents the network we are trying to optimize,
# the `optimizee' as it's called in the paper.
# Actually, it's more accurate to think of this as the error
# landscape.
def f(x):
    x = scale*x
    return tf.reduce_sum(x*x)
```

learn()의 파라매터 optimizer는 기존의 다양한 optimizer이거나 논문에서 제안하는 optimizer이다.

코드에서는 기존의 SGD, RMSProp 2개와 제안한는 optimizer, 모두 3개를 사용하였다.

SGD optimizer

```
def g_sgd(gradients, state, learning_rate=0.1):
    return -learning_rate*gradients, state
```

RMSProp optimizer

```
def g_rms(gradients, state, learning_rate=0.1, decay_rate=0.99):
    if state is None:
        state = tf.zeros(DIMS)
    state = decay_rate*state + (1-decay_rate)*tf.pow(gradients, 2)
    update = -learning_rate*gradients / (tf.sqrt(state)+1e-5)
    return update, state
```

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$

제안하는 RNN optimizer

```
LAYERS = 2
STATE_SIZE = 20
```

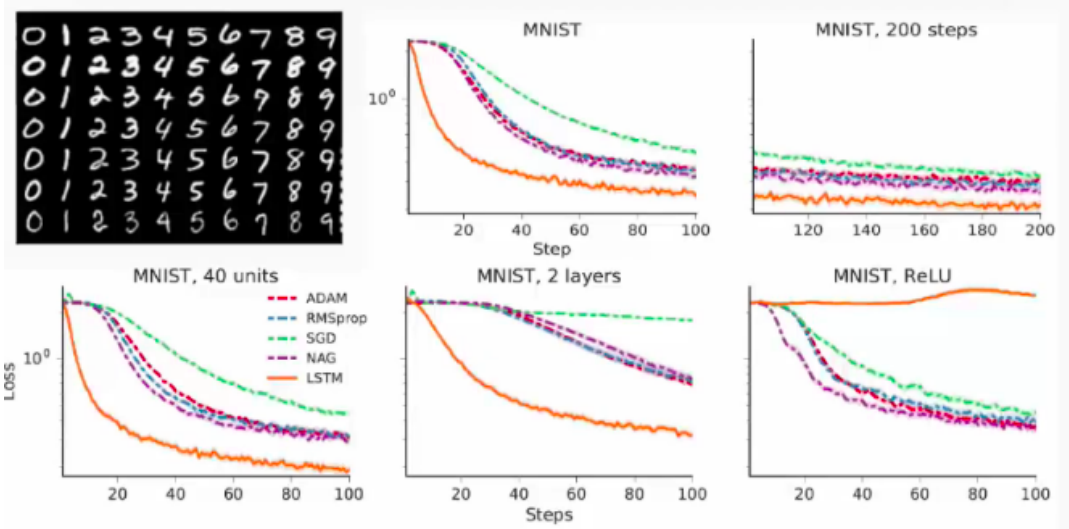
```
cell = tf.contrib.rnn.MultiRNNCell(
    [tf.contrib.rnn.LSTMCell(STATE_SIZE) for _ in range(LAYERS)])
cell = tf.contrib.rnn.InputProjectionWrapper(cell, STATE_SIZE)
cell = tf.contrib.rnn.OutputProjectionWrapper(cell, 1)
cell = tf.make_template('cell', cell)

def g_rnn(gradients, state):
    # Make a `batch' of single gradients to create a
    # "coordinate-wise" RNN as the paper describes.
    gradients = tf.expand_dims(gradients, axis=1)

    if state is None:
        state = [[tf.zeros([DIMS, STATE_SIZE])] * 2] * LAYERS
    update, state = cell(gradients, state)
    # Squeeze to make it a single batch again.
    return tf.squeeze(update, axis=[1]), state
```

실제 할습을 실행하는 코드는 다음과 같다.

```
sgd_losses = learn(g_sgd)
rms_losses = learn(g_rms)
rnn_losses = learn(g_rnn)
```

당연하지만 3개의 optimizer를 호출하는 방법은 동일하다.

결과



LSTM이 논문의 방법. 학습이 빠르게 진행 된다.

Reference

- paper : https://arxiv.org/abs/1611.03824

- paper : https://arxiv.org/abs/1606.04474
  - 논문 아이디어로 코드 구현 : https://hackernoon.com/learning-to-learn-by-gradient-descent-by-gradient-descent-4da2273d64f2
    - jupyter notebook : https://github.com/LlionJ/blog/blob/master/blog/Meta-Learning.ipynb
- 5 minute talk : https://www.youtube.com/watch?v=yxGyNv0Kjcs
- RNN Symposium talk : https://www.youtube.com/watch?v=5yNirTp92Uk

- 공식 code : https://github.com/deepmind/learning-to-learn

No labels