Gradient-based Hyperparameter Optimization through Reversible Learning

Dougal Maclaurin, David Duvenaud, Ryan P. Adams

CSC 2541 presentation by
Haoping Xu, Zhihuan Yu, Jingcheng Niu

Overview

- The bilevel hyperparameter optimization of SGD with momentum
- How to calculate the gradient of the inner SGD
- Implementation detail: precision
- Experiments
  - Dataset distillation
  - Learning rate scheduling

Hyperparameter Optimization of SGD with momentum

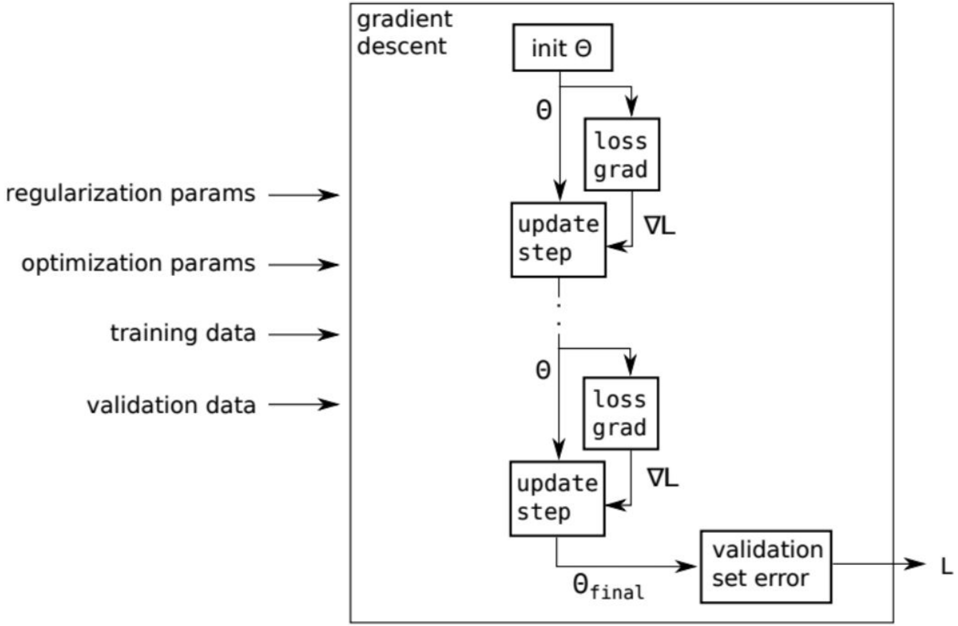
Inner objective:

SGD with momentum

Algorithm 1 Stochastic gradient descent with momentum

input: initial $\mathbf{w}_1$, decays $\boldsymbol{\gamma}$, learning rates $\boldsymbol{\alpha}$, loss function $L(\mathbf{w}, \boldsymbol{\theta}, t)$
initialize $\mathbf{v}_1 = \mathbf{0}$
for $t = 1$ **to** T **do**
  $\mathbf{g}_t = \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)$ ▷ evaluate gradient
  $\mathbf{v}_{t+1} = \gamma_t \mathbf{v}_t - (1 - \gamma_t)\mathbf{g}_t$ ▷ update velocity
  $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{v}_t$ ▷ update position
end for
output trained parameters $\mathbf{w}_T$

Unrolling



Reverse-mode differentiation (RMD) of SGD

Algorithm 2 Reverse-mode differentiation of SGD

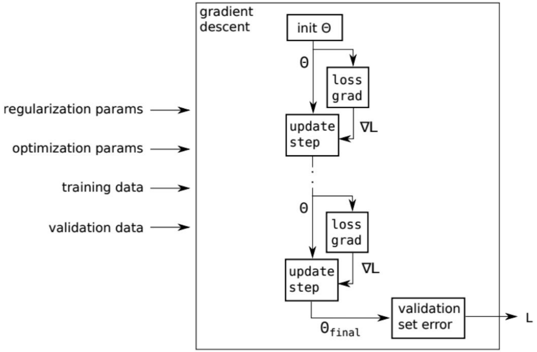
input: $\mathbf{w}_T$, $\mathbf{v}_T$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, train loss $L(\mathbf{w}, \boldsymbol{\theta}, t)$, loss $f(\mathbf{w})$
initialize $d\mathbf{v} = \mathbf{0}$, $d\boldsymbol{\theta} = \mathbf{0}$, $d\alpha_t = \mathbf{0}$, $d\gamma = \mathbf{0}$
initialize $d\mathbf{w} = \nabla_{\mathbf{w}} f(\mathbf{w}_T)$
for $t = T$ **counting down to** 1 **do**
  $d\alpha_t = d\mathbf{w}^{\mathsf{T}}\mathbf{v}_t$

$$\left.\begin{aligned} &\mathbf{w}_{t-1} = \mathbf{w}_t - \alpha_t \mathbf{v}_t \\ &\mathbf{g}_t = \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t) \\ &\mathbf{v}_{t-1} = [\mathbf{v}_t + (1-\gamma_t)\mathbf{g}_t]/\gamma_t \end{aligned}\right\} \text{exactly reverse gradient descent operations}$$

  $d\mathbf{v} = d\mathbf{v} + \alpha_t d\mathbf{w}$
  $d\gamma_t = d\mathbf{v}^{\mathsf{T}}(\mathbf{v}_t + \mathbf{g}_t)$
  $d\mathbf{w} = d\mathbf{w} - (1-\gamma_t) d\mathbf{v} \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)$
  $d\boldsymbol{\theta} = d\boldsymbol{\theta} - (1-\gamma_t) d\mathbf{v} \nabla_{\boldsymbol{\theta}} \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)$
  $d\mathbf{v} = \gamma_t d\mathbf{v}$
end for
output gradient of $f(\mathbf{w}_T)$ w.r.t $\mathbf{w}_1$, $\mathbf{v}_1$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$

Algorithm 1 Stochastic gradient descent with momentum

input: initial $\mathbf{w}_1$, decays $\boldsymbol{\gamma}$, learning rates $\boldsymbol{\alpha}$, loss function $L(\mathbf{w}, \boldsymbol{\theta}, t)$
initialize $\mathbf{v}_1 = \mathbf{0}$
for $t = 1$ **to** T **do**
  $\mathbf{g}_t = \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)$ ▷ evaluate gradient
  $\mathbf{v}_{t+1} = \gamma_t \mathbf{v}_t - (1-\gamma_t)\mathbf{g}_t$ ▷ update velocity
  $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{v}_t$ ▷ update position
end for
output trained parameters $\mathbf{w}_T$



Reverse-mode differentiation (RMD) of SGD

Algorithm 2 Reverse-mode differentiation of SGD

input: $\mathbf{w}_T$, $\mathbf{v}_T$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, train loss $L(\mathbf{w}, \boldsymbol{\theta}, t)$, loss $f(\mathbf{w})$

initialize $d\mathbf{v} = \mathbf{0}$, $d\boldsymbol{\theta} = \mathbf{0}$, $d\alpha_t = 0$, $d\gamma = 0$

initialize $d\mathbf{w} = \boxed{\nabla_{\mathbf{w}} f(\mathbf{w}_T)}$

for $t = T$ **counting down to** 1 **do**

$\quad d\alpha_t = d\mathbf{w}^{\mathsf{T}} \mathbf{v}_t$

$$\left.\begin{aligned} \mathbf{w}_{t-1} &= \mathbf{w}_t - \alpha_t \mathbf{v}_t \\ \mathbf{g}_t &= \boxed{\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)} \\ \mathbf{v}_{t-1} &= [\mathbf{v}_t + (1 - \gamma_t)\mathbf{g}_t]/\gamma_t \end{aligned}\right\} \text{exactly reverse gradient descent operations}$$

$\quad d\mathbf{v} = d\mathbf{v} + \alpha_t d\mathbf{w}$

$\quad d\gamma_t = d\mathbf{v}^{\mathsf{T}} (\mathbf{v}_t + \mathbf{g}_t)$

$\quad d\mathbf{w} = d\mathbf{w} - (1 - \gamma_t) \boxed{d\mathbf{v} \nabla_{\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)}$

$\quad d\boldsymbol{\theta} = d\boldsymbol{\theta} - (1 - \gamma_t) \boxed{d\mathbf{v} \nabla_{\boldsymbol{\theta}} \nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)}$

$\quad d\mathbf{v} = \gamma_t d\mathbf{v}$

end for

output gradient of $f(\mathbf{w}_T)$ w.r.t $\mathbf{w}_1$, $\mathbf{v}_1$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$

Quite a few of gradients!

Reverse-mode differentiation (RMD) of SGD

Algorithm 2 Reverse-mode differentiation of SGD

input: $\mathbf{w}_T$, $\mathbf{v}_T$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, train loss $L(\mathbf{w}, \boldsymbol{\theta}, t)$, loss $f(\mathbf{w})$
initialize $d\mathbf{v} = \mathbf{0}$, $d\boldsymbol{\theta} = \mathbf{0}$, $d\alpha_t = 0$, $d\gamma = 0$
initialize $d\mathbf{w} = \boxed{\nabla_{\mathbf{w}} f(\mathbf{w}_T)}$
for $t = T$ **counting down to** 1 **do**

$$d\alpha_t = d\mathbf{w}^\mathsf{T}\mathbf{v}_t$$

$$\left.\begin{aligned} \mathbf{w}_{t-1} &= \mathbf{w}_t - \alpha_t \mathbf{v}_t \\ \mathbf{g}_t &= \boxed{\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)} \\ \mathbf{v}_{t-1} &= [\mathbf{v}_t + (1-\gamma_t)\mathbf{g}_t]/\gamma_t \end{aligned}\right\} \text{exactly reverse gradient descent operations}$$

$$d\mathbf{v} = d\mathbf{v} + \alpha_t d\mathbf{w}$$

$$d\gamma_t = d\mathbf{v}^\mathsf{T}(\mathbf{v}_t + \mathbf{g}_t)$$

$$d\mathbf{w} = d\mathbf{w} - (1-\gamma_t)\boxed{d\mathbf{v}\nabla_{\mathbf{w}}\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)}$$

$$d\boldsymbol{\theta} = d\boldsymbol{\theta} - (1-\gamma_t)\boxed{d\mathbf{v}\nabla_{\boldsymbol{\theta}}\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\theta}, t)}$$

$$d\mathbf{v} = \gamma_t d\mathbf{v}$$

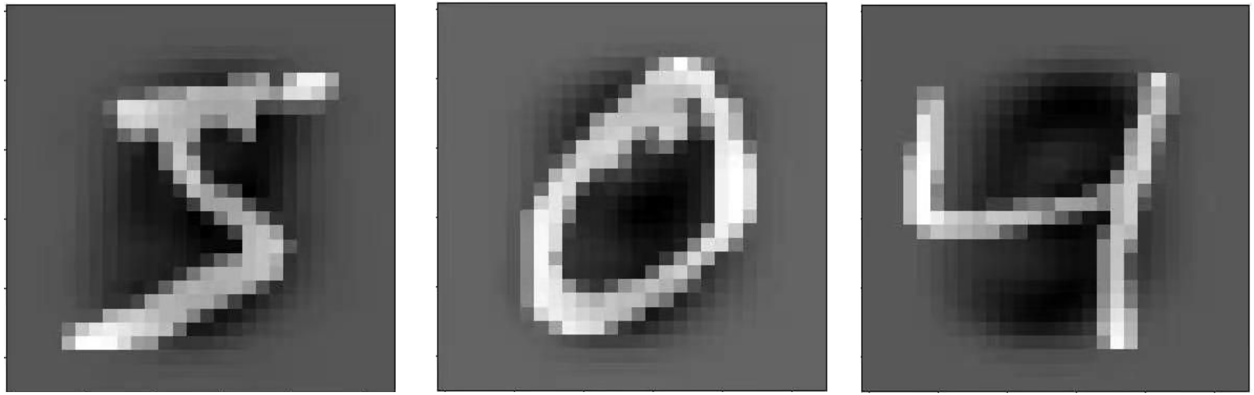
end for
output gradient of $f(\mathbf{w}_T)$ w.r.t $\mathbf{w}_1$, $\mathbf{v}_1$, $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$

Autograd comes in handy!

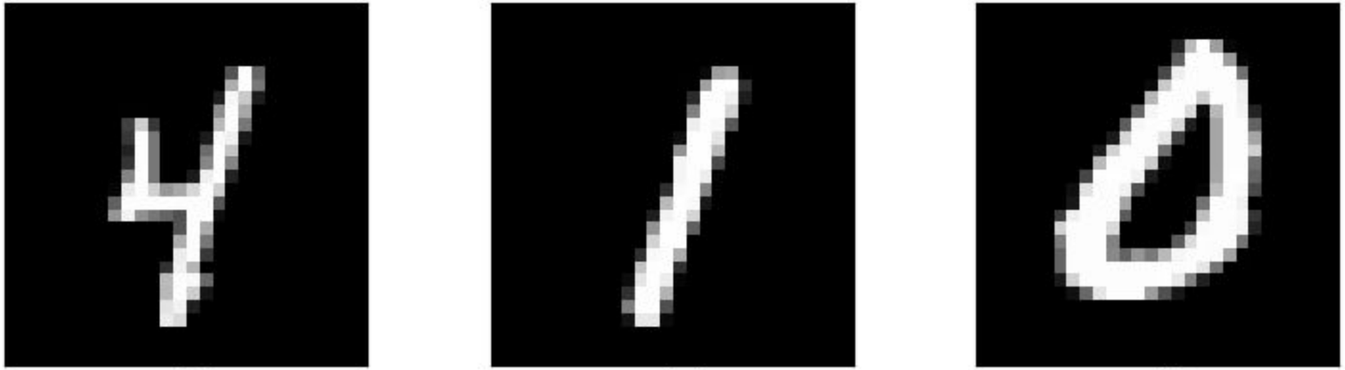
```
loss_grad = grad(loss)

loss_hvp  = grad(loss_grad)

meta_hvp  = grad(loss_grad, argnums=1)
```

Side note: MNIST data

The paper’s
MNIST



Original
MNIST



Numerical Precision

In practice, Algorithm 2 fails due to finite numerical precision.

5: $$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + (1-\gamma)\mathbf{g}_{t-1}$$

8: $$\mathbf{v}_{t-1} = [\mathbf{v}_t + (1-\gamma_t)\mathbf{g}_t]/\gamma_t$$

The reverse process requires repeated multiplication by $1/\gamma$, causing errors accumulate exponentially!

Exactly Reversible Multiplication

So instead, we represent v and w each with a 64-bit integer and a remainder buffer.

Using this method we have can achieve float64 precision and our computations is reversible.

Algorithm 3 Exactly reversible multiplication by a ratio

Input: Information buffer i, value c, ratio n/d
$i = i \times d$ ▷ make room for new digit
$i = i + (c \mod d)$ ▷ store digit lost by division
$c = c \div d$ ▷ divide by denominator
$c = c \times n$ ▷ multiply by numerator
$c = c + (i \mod n)$ ▷ add digit from buffer
$i = i \div n$ ▷ shorten information buffer
return updated buffer i, updated value c

Experiments

- Dataset distillation
- Learning rate scheduling