

Introduction to Machine Learning

Day 12: Optimizers, Learning Rate Schedules, and Batch Normalization

Joanna Bieri DATA301

Get out your handwritten notes!

Mini Review - Where We Left Off

REVIEW

10 hidden layers, ReLU, Kaiming, plain SGD, learning rate 0.001.

epoch 1	valid RMSE: 0.8198
epoch 5	valid RMSE: 0.6951
epoch 10	valid RMSE: 0.6473
epoch 15	valid RMSE: 0.6342
epoch 20	valid RMSE: 0.6076

The deep network learns, but slowly. This is the number to beat today.

Mini Review - Momentum

REVIEW

$$v = \beta \cdot v + \text{gradient}$$

$$w = w - \eta \cdot v$$

If the gradient is 1 at every step, the velocity goes 1, 1.9, 2.71, ... and levels off at about **10**.

```
torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

Mini Review - Adam

REVIEW

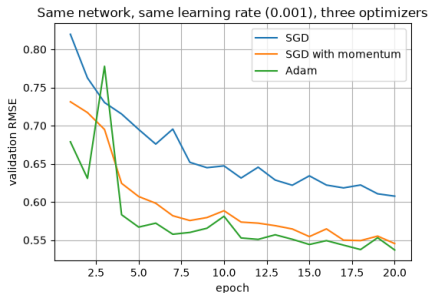
Adam gives every weight its own step size.

- 1 It keeps a velocity for each weight, like momentum.
- 2 It keeps track of how big each weight's gradients have been lately, and divides the step by that size.

```
adam_optimizer = torch.optim.Adam(  
    adam_model.parameters(),  
    lr=0.001)          # the usual starting point for Adam
```

Mini Review - Three Optimizers

REVIEW



After 20 epochs: SGD 0.6076, momentum 0.5454, Adam 0.537.

Mini Review - Learning Rate Schedules

REVIEW

```
scheduler = torch.optim.lr_scheduler.StepLR(  
    schedule_optimizer,  
    step_size=10,      # every 10 epochs...  
    gamma=0.1)         # ...multiply by 0.1
```

Epochs 1 to 10 use 0.001, epochs 11 to 20 use 0.0001, epochs 21 to 30 use 0.00001.

Call `scheduler.step()` **once per epoch**, after that epoch's batches are done.

Mini Review - Batch Normalization

REVIEW

It is `StandardScaler`, moved inside the network. For every neuron in the layer before it:

- 1 Take that neuron's outputs for all the rows in the current **batch**.
- 2 Subtract their mean and divide by their standard deviation.
- 3 Multiply by a scale and add a shift, two new numbers per neuron that the network learns.

```
nn.Linear(8, 50),  
nn.BatchNorm1d(50),  
nn.ReLU(),
```

You will see it again inside CNNs.

Mini Review - Training Mode and Prediction Mode

REVIEW

- **Training mode** (`model.train()`): the layer uses the mean and standard deviation of the current batch.
- **Prediction mode** (`model.eval()`): it uses the running averages it saved during training.

`model.train()` before training. `model.eval()` before predicting.

Momentum

PREDICT

step	1	2	3
gradient	1	1	-1
velocity	?	?	?

The velocity starts at 0 and $v = 0.9 \cdot v + \text{gradient}$. What is the velocity after each step?

Hint. Keep 90 percent of the old velocity, then add the new gradient.

Momentum

ANSWER

The gradients are 1, 1, -1. What is the velocity after each step?

$$\text{step 1: } 0.9 * 0 + 1 = 1$$

$$\text{step 2: } 0.9 * 1 + 1 = 1.9$$

$$\text{step 3: } 0.9 * 1.9 - 1 = 0.71$$

The velocity builds while the gradients agree. When the gradient changes sign the velocity shrinks, and it is still positive after step 3.

Adam

ASK

In your own words,
what does "adaptive" mean
in the name Adam?

Adam

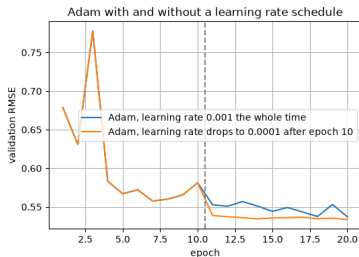
ANSWER

In your own words, what does "adaptive" mean in the name Adam?

- Every weight gets its own step size.
- Adam keeps a velocity for each weight, like momentum, and keeps track of how big that weight's gradients have been lately.
- Weights with big gradients get their steps scaled down, weights with tiny gradients get them scaled up.

Learning Rate Schedules

ASK



Adam, learning rate 0.001, 20 epochs. One run uses StepLR with `step_size=10` and `gamma=0.1`. What changed at the dashed line? Why is that curve steadier after it?

Learning Rate Schedules

ANSWER

What changed at the dashed line? Why is one curve so much steadier after it?

- The learning rate dropped from 0.001 to 0.0001.
- Smaller steps stop the model from jumping back and forth.
- The last five epochs moved from 0.5361 to 0.5336. Without the schedule they bounced between 0.537 and 0.5531.

Learning Rate Schedules

ASK

Why do we want big steps
early in training
and small steps late?

Hint. How far are we from a good set of weights at the start? At the end?

Learning Rate Schedules

ANSWER

Why do we want big steps early in training and small steps late?

- Early on we are far from a good set of weights. Big steps get us close quickly.
- Late in training we are close. Big steps jump around it.
- One learning rate can not do both, so a schedule changes it.

Batch Normalization

ASK

Why does a network with batch normalization
need `model.train()` and `model.eval()`?

Batch Normalization

ANSWER

Why does batch normalization need `model.train()` and `model.eval()`?

- In training mode the layer standardizes with the current batch, so a prediction depends on the other rows in the batch.
- In prediction mode it uses the running averages saved in training, so a prediction is always the same.
- Forget `model.eval()` and the validation score is wrong, with no error message.

Comparing Everything

ASK

The Adam model: training RMSE 0.4974,
validation RMSE 0.537.

On Day 10 they were 0.600 and 0.6067.

What is starting to happen?

Hint. Which rows does the model fit better now?

Comparing Everything

ANSWER

Adam: training RMSE 0.4974, validation RMSE 0.537. On Day 10 they were 0.600 and 0.6067. What is starting to happen?

- The model fits the training rows better than rows it has never seen.
- That is **overfitting**. The gap is small so far.
- Dealing with overfitting in neural networks is next class.

Questions?

Questions?