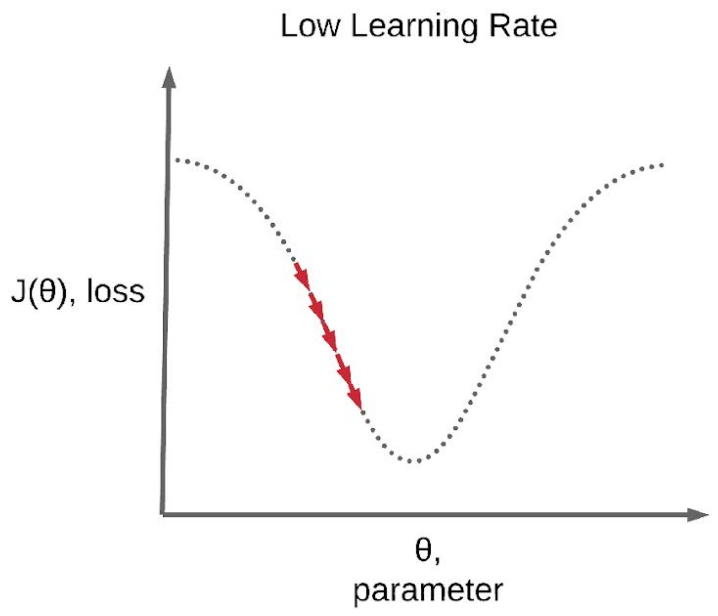
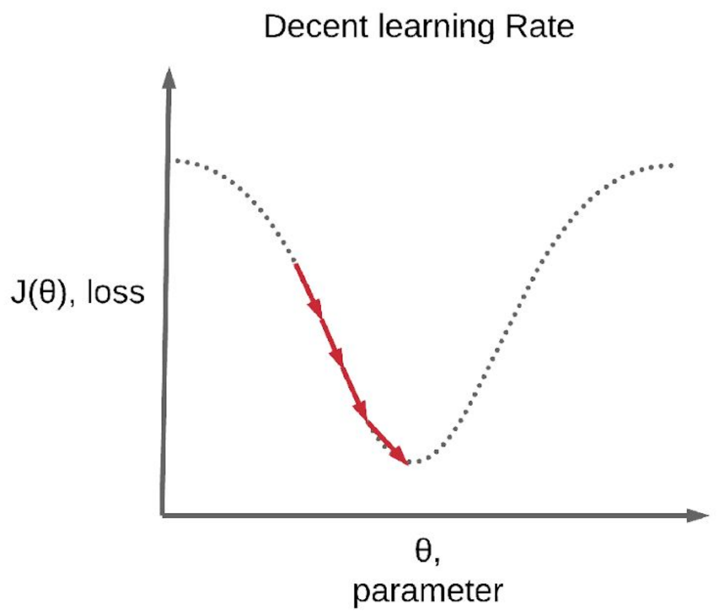
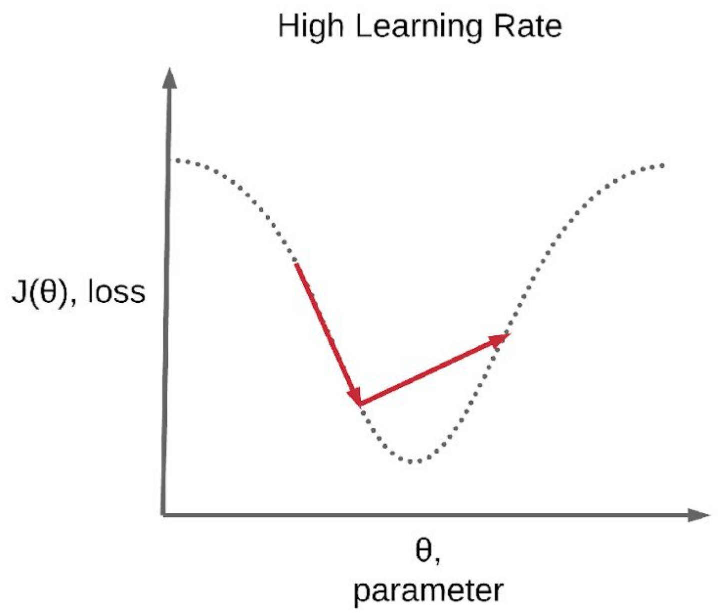
Learning Rate

The Effect of the Learning Rate

- Benefits:
  - Faster convergence?
  - Higher accuracy?





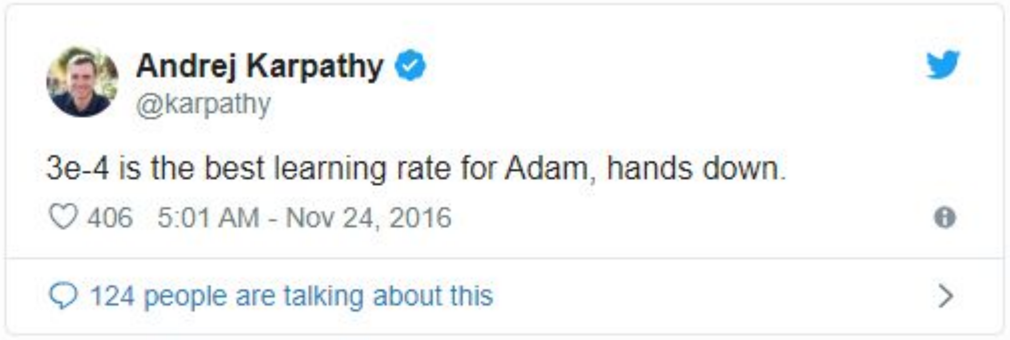


How to choose a Learning Rate?

- *“The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate.”*

  – Andrew Ng

- But the choice of a good learning rate seems arbitrary…



Andrej Karpathy
@karpathy

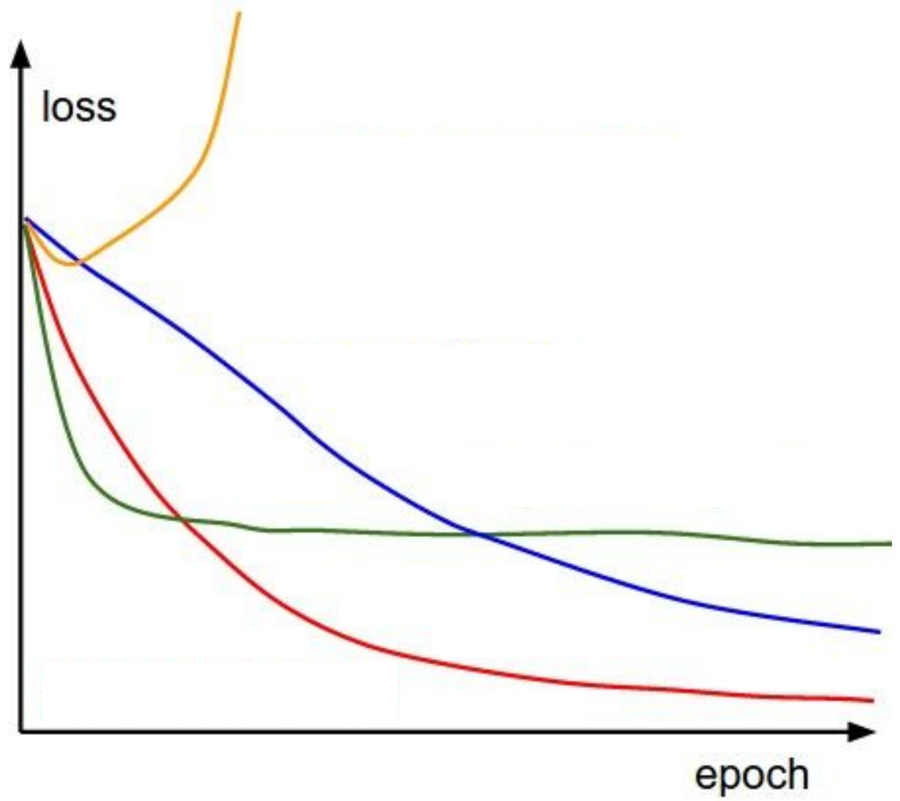
3e-4 is the best learning rate for Adam, hands down.

406 5:01 AM - Nov 24, 2016

124 people are talking about this

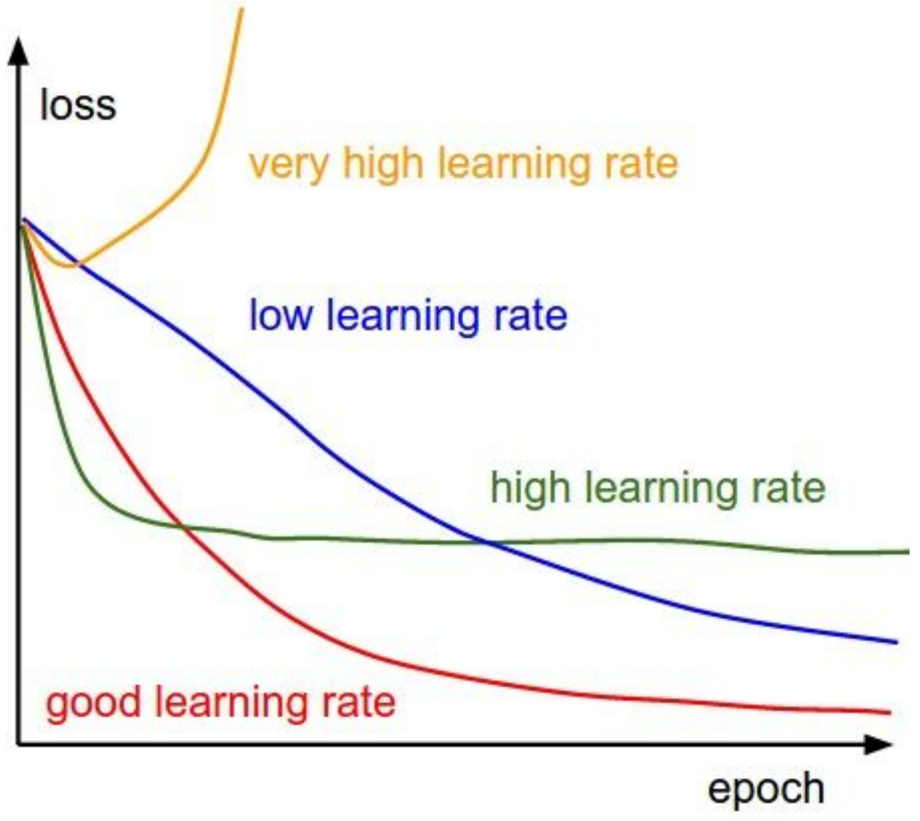
Learning Rate

- What is the effect of the learning rate on convergence?
- Which of the graphs corresponds to a Low / Good / High LR?



Learning Rate

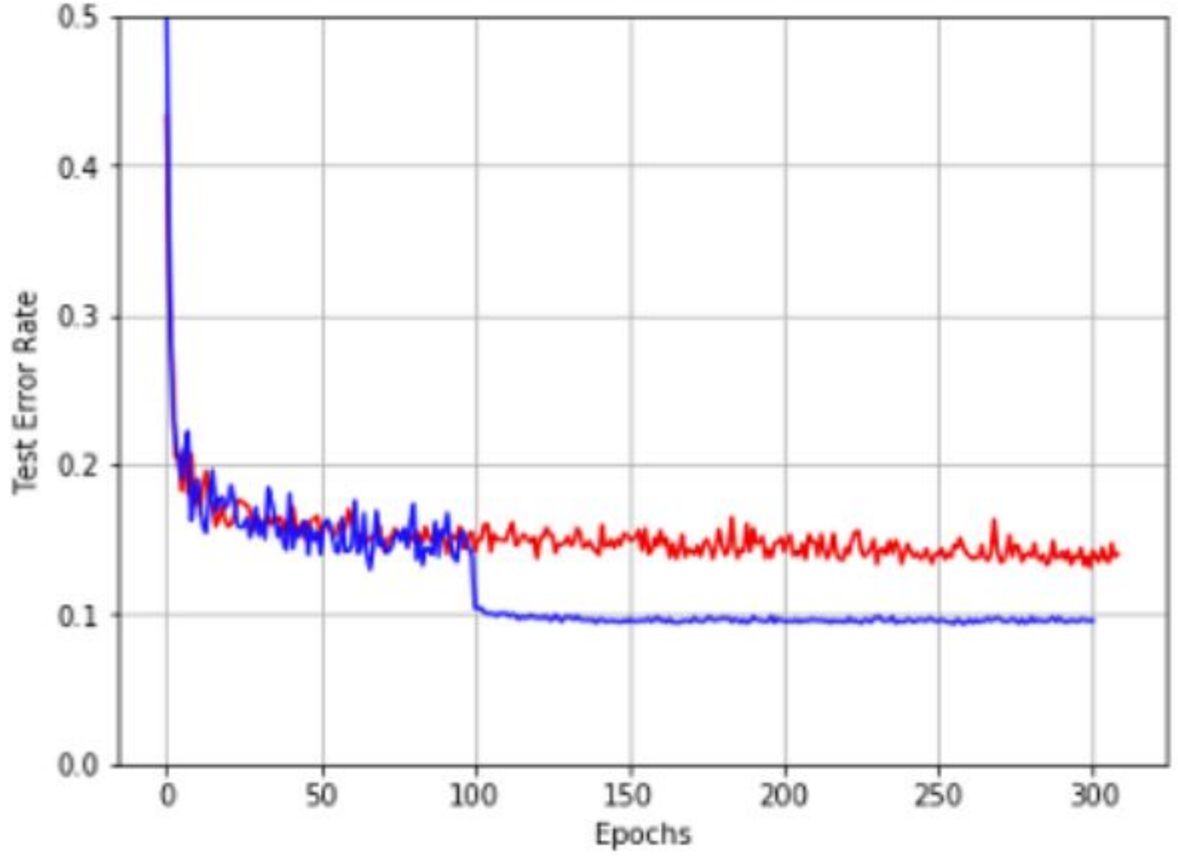
- What is the effect of the learning rate on convergence?



Learning Rate Scheduling

- It is often useful to lower the learning rate as learning progresses
- Common practices –
  - Reduce by some factor every X epochs
  - Reduce by some factor every time the validation error increases
  - 1/t decay

    $\alpha = \alpha_0/(1+kt)$ where a_0, k are hyperparameters and t is the iteration number.
  - If in doubt, prefer a slower decay



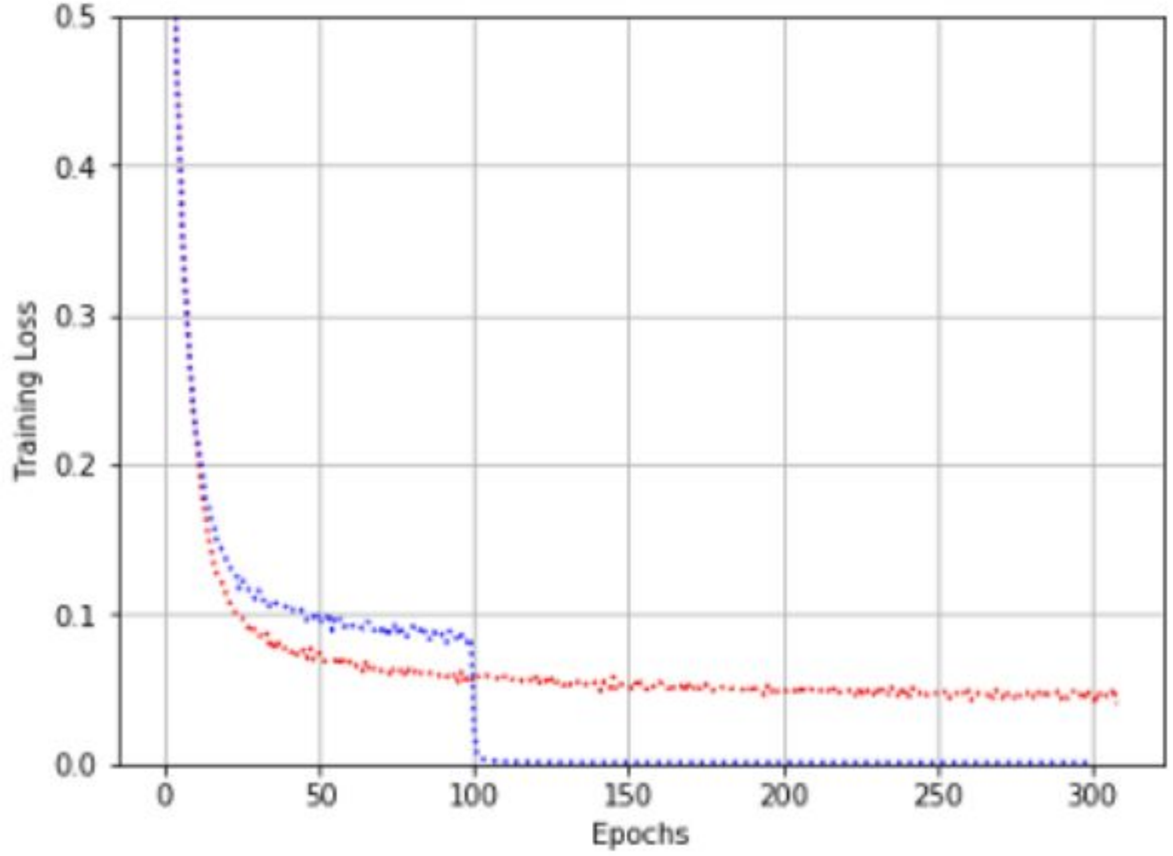
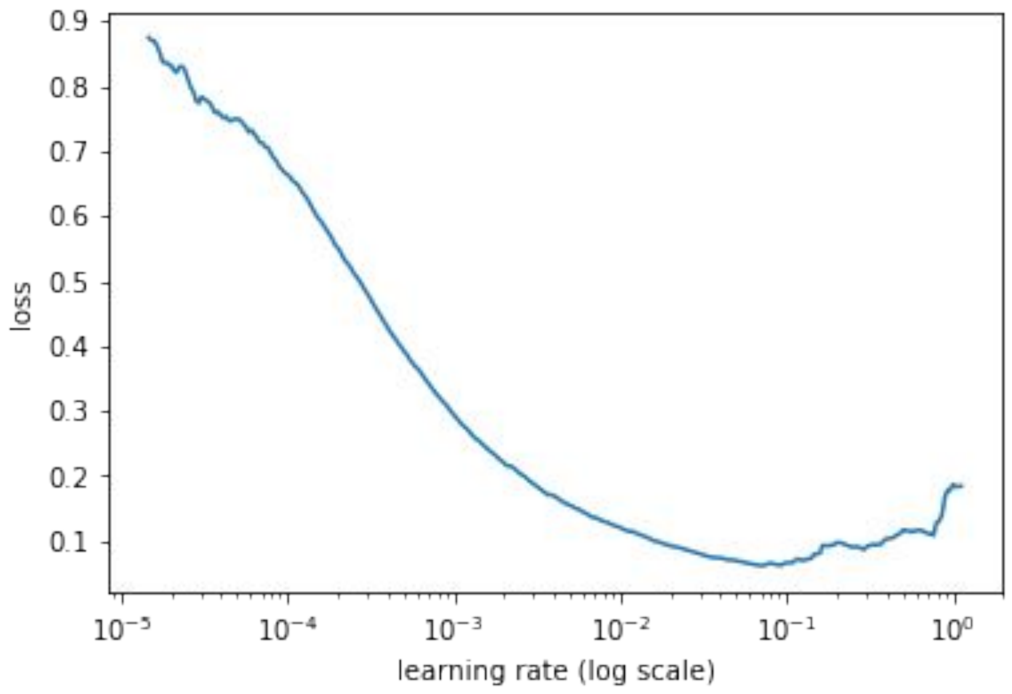


Fig. 5: Comparison of the results obtained by tuning and not tuning the learning rate. **Red lines**: the results without tuning; **Blue lines**: the results with tuning.

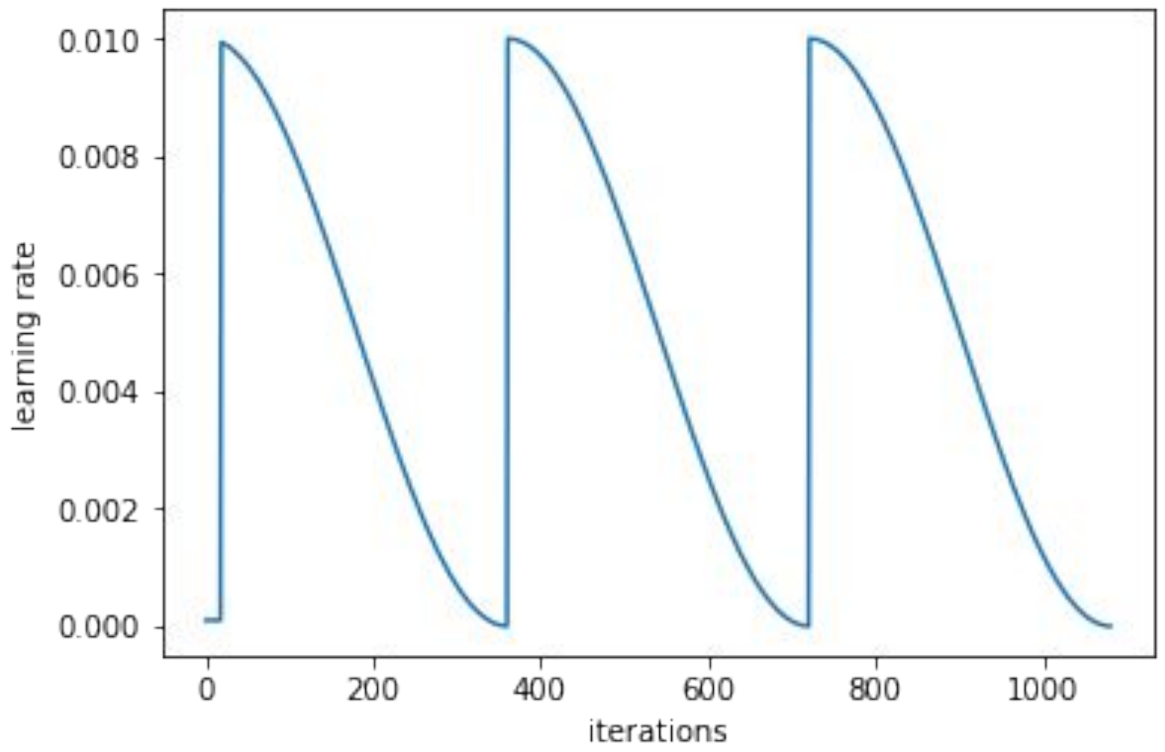
The LR Range Test

- Pick a very small learning rate (much smaller than you would ever likely use). As you train, exponentially increase the learning rate. Keep track of the loss function for each value of the learning rate. If you're in the right range, the loss should drop, then increase as the learning rate gets too high



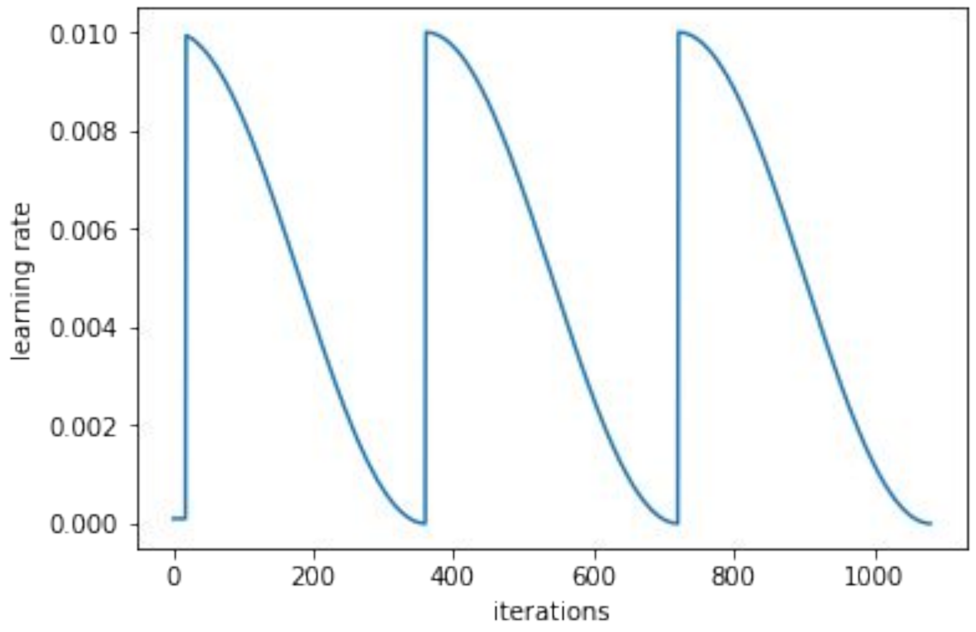
Cyclical Learning Rates

- No need to design a LR decay scheme
- Higher chances of hitting it right
- Why is the learning rate increasing sharply instead of gradually?

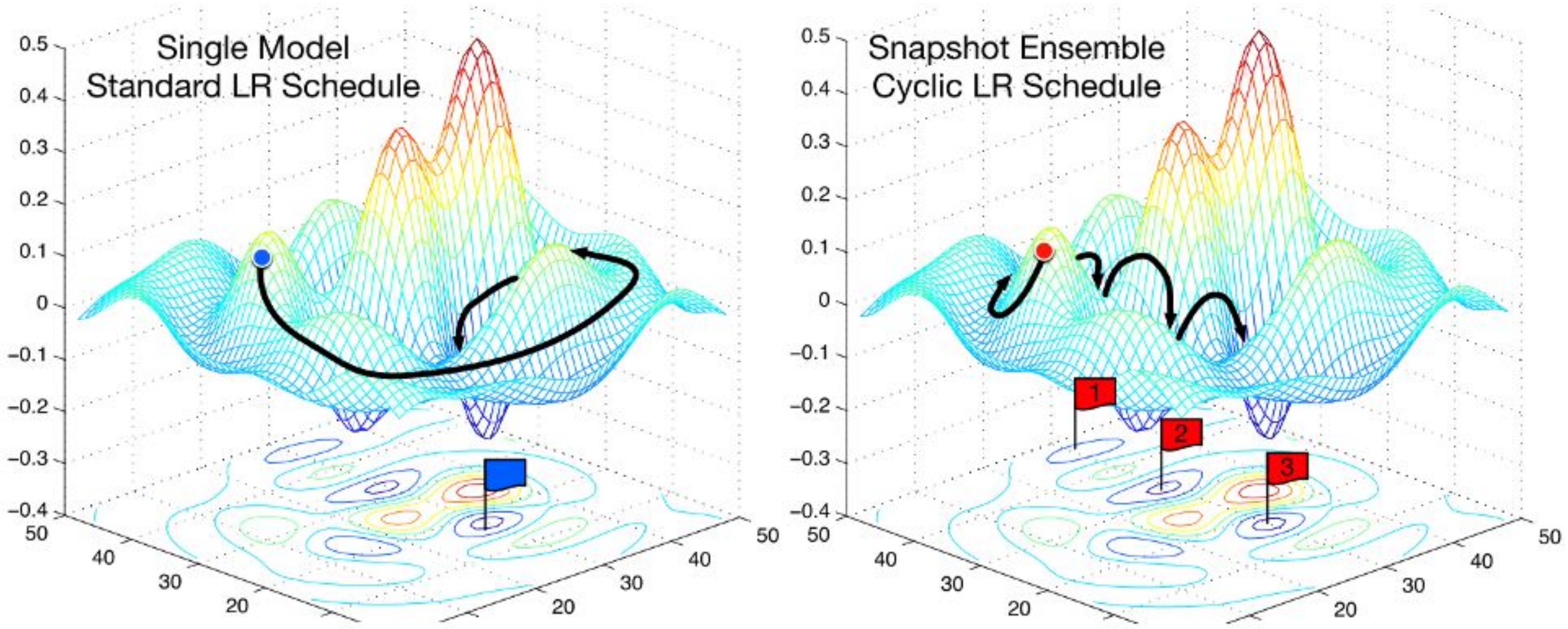


Cyclical Learning Rates

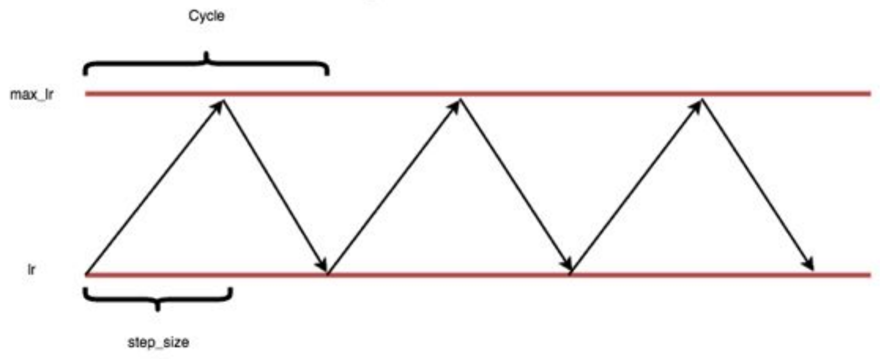
- We want the network to converge to a:
  - Low minimum
  - Wide minimum (more likely to generalize well)
- The length of a cycle is often an epoch
- It can increase with time
- Shuffling is important
- We still need to find a good *LR range* – use the LR range test



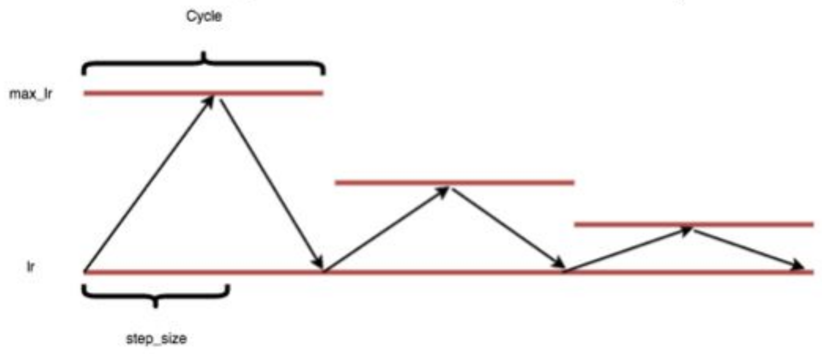
Cyclic Learning Rate



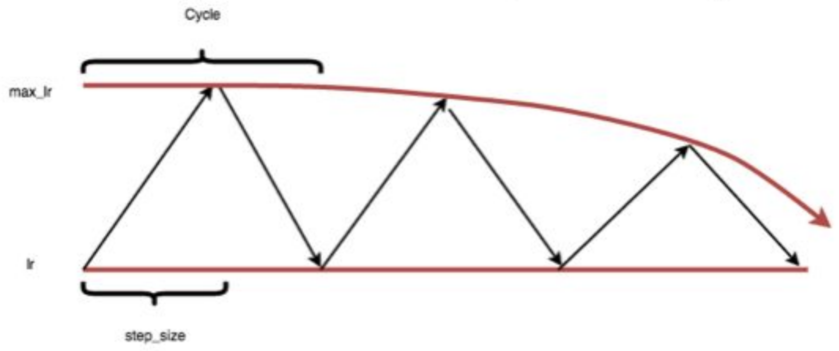
Triangular schedule



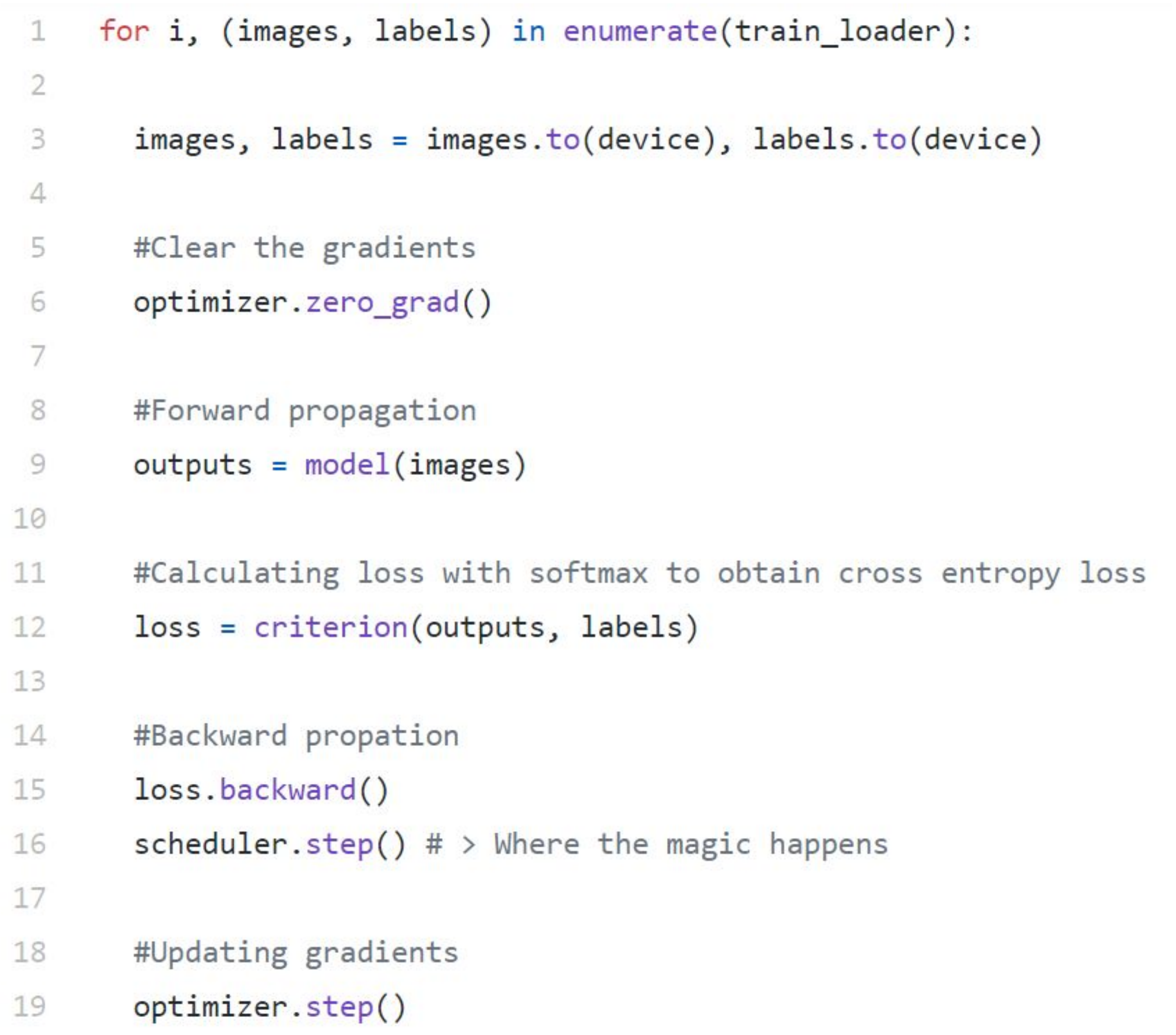
Triangular schedule with fixed decay



Triangular schedule with exponential decay



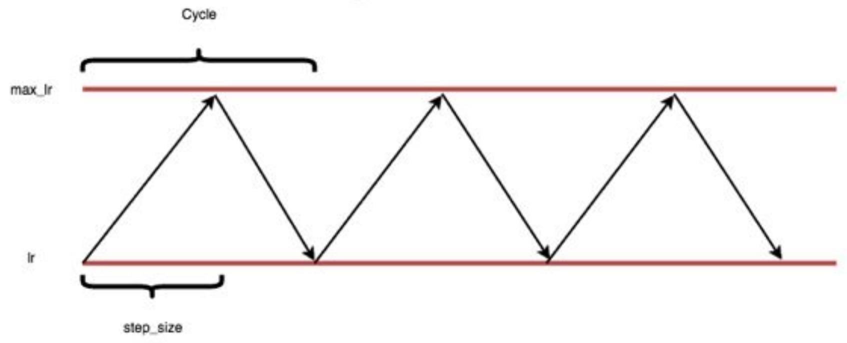
The Training Loop



```python
for i, (images, labels) in enumerate(train_loader):

  images, labels = images.to(device), labels.to(device)

  #Clear the gradients
  optimizer.zero_grad()

  #Forward propagation
  outputs = model(images)

  #Calculating loss with softmax to obtain cross entropy loss
  loss = criterion(outputs, labels)

  #Backward propation
  loss.backward()
  scheduler.step() # > Where the magic happens

  #Updating gradients
  optimizer.step()
```

```
model = CNN().to(device)
criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.SGD(model.parameters(), lr=1.)
step_size = 4*len(train_loader)
clr = cyclical_lr(step_size, min_lr=end_lr/factor, max_lr=end_lr)
scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, [clr])
```

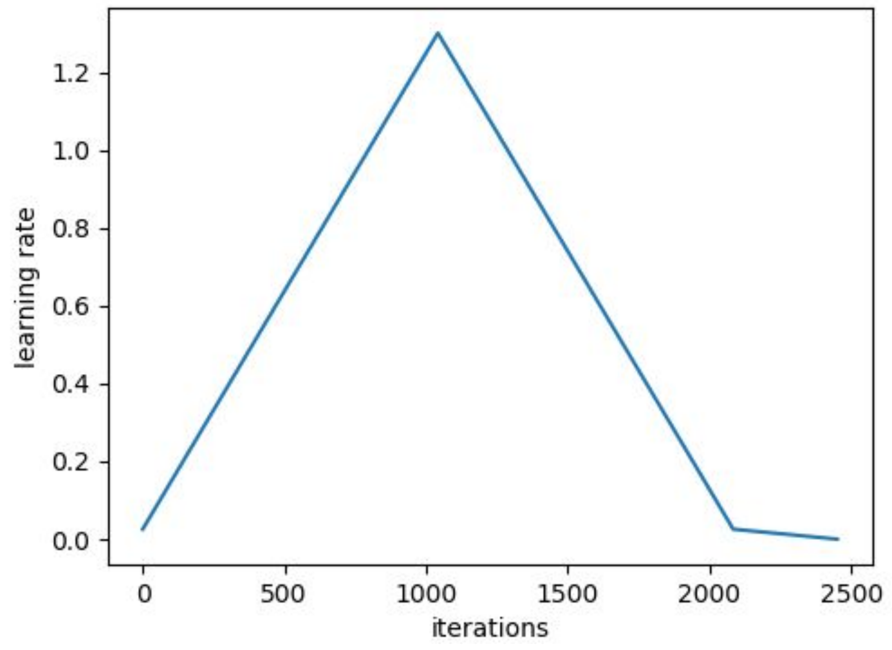
Triangular schedule

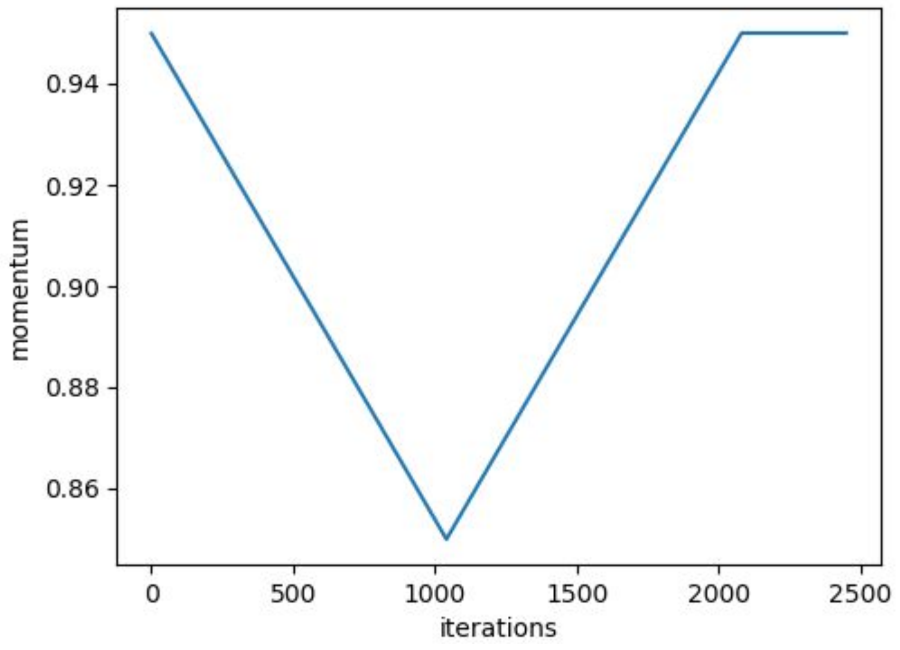


```
def cyclical_lr(stepsize, min_lr=3e-4, max_lr=3e-3):

    # Scaler: we can adapt this if we do not want the triangular CLR
    scaler = lambda x: 1.

    # Lambda function to calculate the LR
    lr_lambda = lambda it: min_lr + (max_lr - min_lr) * relative(it, stepsize)

    # Additional function to see where on the cycle we are
    def relative(it, stepsize):
        cycle = math.floor(1 + it / (2 * stepsize))
        x = abs(it / stepsize - 2 * cycle + 1)
        return max(0, (1 - x)) * scaler(cycle)

    return lr_lambda
```

“Super-Convergence” - The 1cycle Policy

- Have a single cycle of change – and then lower the LR and increase the momentum
- Supposed to make training much faster





Learning Rate in PyTorch

```
optim = torch.optim.SGD(model.parameters(), lr=0.01)


for g in optim.param_groups:
    g['lr'] = 0.001
```

Suggested Reading

- https://www.mygreatlearning.com/blog/understanding-learning-rate-in-machine-learning/
- https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
- https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

Recommended Resources

- https://techburst.io/improving-the-way-we-work-with-learning-rate-5e99554f163b
- https://towardsdatascience.com/adaptive-and-cyclical-learning-rates-using-pytorch-2bf904d18dee
- https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1
- https://www.jeremyjordan.me/nn-learning-rate/
- https://pytorch.org/docs/master/optim.html
- https://www.freecodecamp.org/news/how-to-pick-the-best-learning-rate-for-your-machine-learning-project-9c28865039a8/

Home Experiments

- Write code to do a Learning Rate check for the Israeli Politicians task we’ve done in the first lesson.
- Compare convergence with the default LR we used and the LR you found.