APS360 Fundamentals of AI

Lisa Zhang

Lecture 4; Jan 17, 2019

Agenda

- Training Terminology
- Training Curve
- Hyperparameters
- Validation Set
- Assignment 2

Agenda

- Training Terminology
- Training Curve
- Hyperparameters
- Validation Set
- Assignment 2

Reminder: Assignment 1 is due Sunday 9pm

Neural Network Training

Training

Terms from last time:

- Loss function $L(actual, predicted)$
- Optimizer
- Training set
- Test set

Code from last week

```
for (image, label) in mnist_train[:1000]:
    # actual ground truth: is the digit less than 3?
    actual = (label < 3).reshape([1,1]) \
                        .type(torch.FloatTensor)
    # prediction
    out = pigeon(img_to_tensor(image))
    # update the parameters based on the loss
    loss = criterion(out, actual) # compute loss
    loss.backward()        # compute param updates
    optimizer.step()       # make param updates
    optimizer.zero_grad()  # clean up
```

Summary of Code

1. use our network to make the predictions for **one image**
2. compute the loss for that **one image**
3. take a “step” to optimize the loss of the **one image**

Batching

1. use our network to make the predictions for *n* **images**
2. compute the *average* loss for those *n* **image**
3. take a “step” to optimize the *average* loss of those *n* **image**

Averaging Loss

- Average loss across multiple training inputs is less “noisy”
- Less likely to provide “bad information” because of a single “bad” input

Batching Code

```
train_loader = torch.utils.data.DataLoader(mnist_train, bat
for n, (imgs, labels) in enumerate(train_loader):
    if n >= 10: break
    actual = (label < 3).reshape([1,1]).type(torch.FloatTen
    out = pigeon(img_to_tensor(image))
    loss = criterion(out, actual)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

Batching Code

```
train_loader = torch.utils.data.DataLoader(mnist_train, bat
for n, (imgs, labels) in enumerate(train_loader):
    if n >= 10: break
    actual = (label < 3).reshape([1,1]).type(torch.FloatTen
    out = pigeon(img_to_tensor(image))
    loss = criterion(out, actual)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

Exactly the same!

Batch Size

The **batch size** is the number of training examples used per optimization “step”.

Each optimization “step” is known as an **iteration**.

The parameters are updated once in an iteration.

Q: What happens if the batch size is too small? Too large?

Ineffective Batch Size

- **Too small**:
  - We optimize a (possily very) different function L at each iteration
  - Noisy
- **Too large**:
  - Expensive
  - Average loss might not change very much as batch size grows

Epoch

An **epoch** is a measure of the number of times all training data are used once to update the parameters.

Example:

- There are 1000 images we use for training
- If `batch_size = 10` then 100 iterations $=$ 1 epoch

Optimizer Settings

- The optimizer settings can also affect the speed of neural network training.

```
optimizer = optim.SGD(pigeon.parameters()
                      lr=0.005,
                      momentum=0.9)
```

Learning Rate

The **learning rate** determines the size of the “step” that an optimizer takes during each *iteration*.

Larger step size = make a bigger change in the parameters in each iteration.

Q: What happens if the learning rate is small? Large?

Learning Rate Size

- **Too small**:
  - Parameters don't change very much in each iteration
  - Takes a long time to train the network
- **Too large**:
  - "Noisy"
  - Average loss might not change very much as batch size grows
  - Very large can be detrimental to neural network training

Appropriate Learning Rate

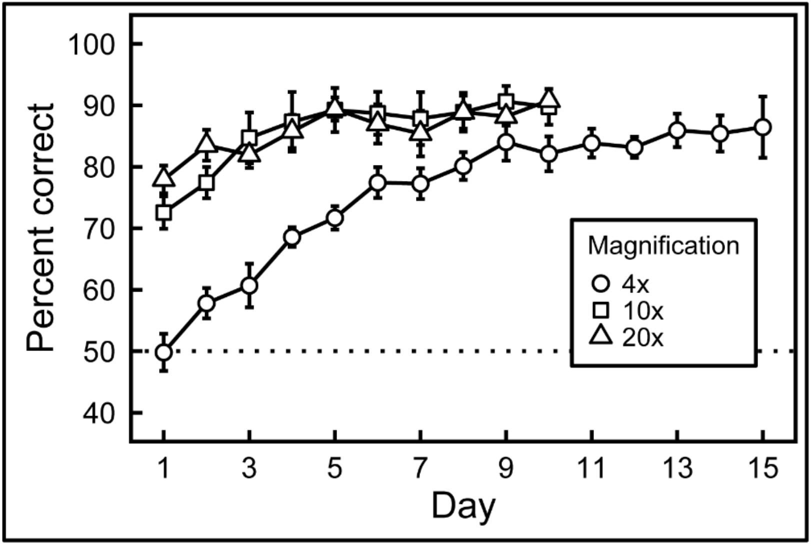
Depends on:

- The learning problem
- The optimizer
- The batch size
  - Smaller learning rate for larger batch size
  - Larger learning rate for smaller batch size
- The stage of training
  - *Reduce* learning rate as training progresses

Tracking Training

- How do we know when to stop training?
- Is training going well?
- Do we have a good batch size?
- Do we have a good learning rate?

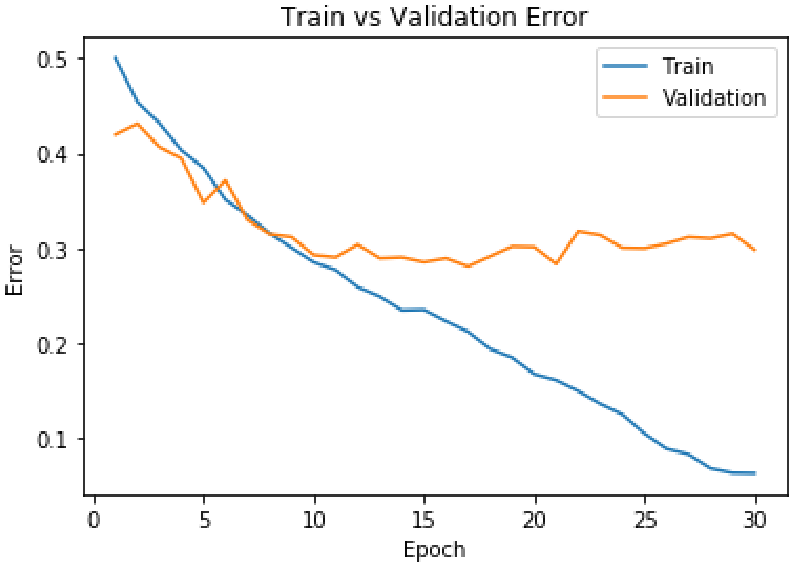
Training Curve for Biological Pigeon



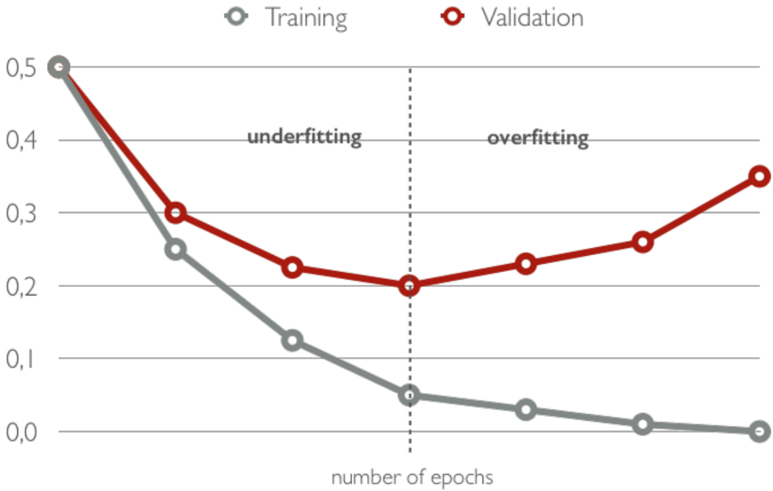
Training Curve

- **x-axis**: epochs or iterations
- **y-axis**: loss, error, or accuracy

Typical Training Curve



Assessing the Fit



Hyperparameters

- Size of network
  - Number of layers
  - Number of neurons in each layer
- Choice of Activation Function
- Learning Rate
- Batch Size

Q: How do we tune hyperparameters?

Assignment 2

- Distinguishing cats and dogs
- You have pretty much everything you need to begin assignment 2!