Introduction to Deep Learning (CS474)

Lecture 9

Outline

- Mechanics of Learning-PART IV
    - Training, validation, and overfitting

Introduction

- A highly adaptable model will tend to use its **many** parameters to make sure the loss is minimal at the data points, but we'll have no guarantee that the model behaves well.

- After all, that's what we're **asking the optimizer** to do: **minimize** the loss at the data points.

- Sure enough, if we had independent data points that we didn't use to evaluate our loss or descend along its negative gradient, we would soon find out that evaluating the loss at those independent data points would yield higher-than-expected loss.

- We have already mentioned this phenomenon, called **overfitting**.

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Introduction

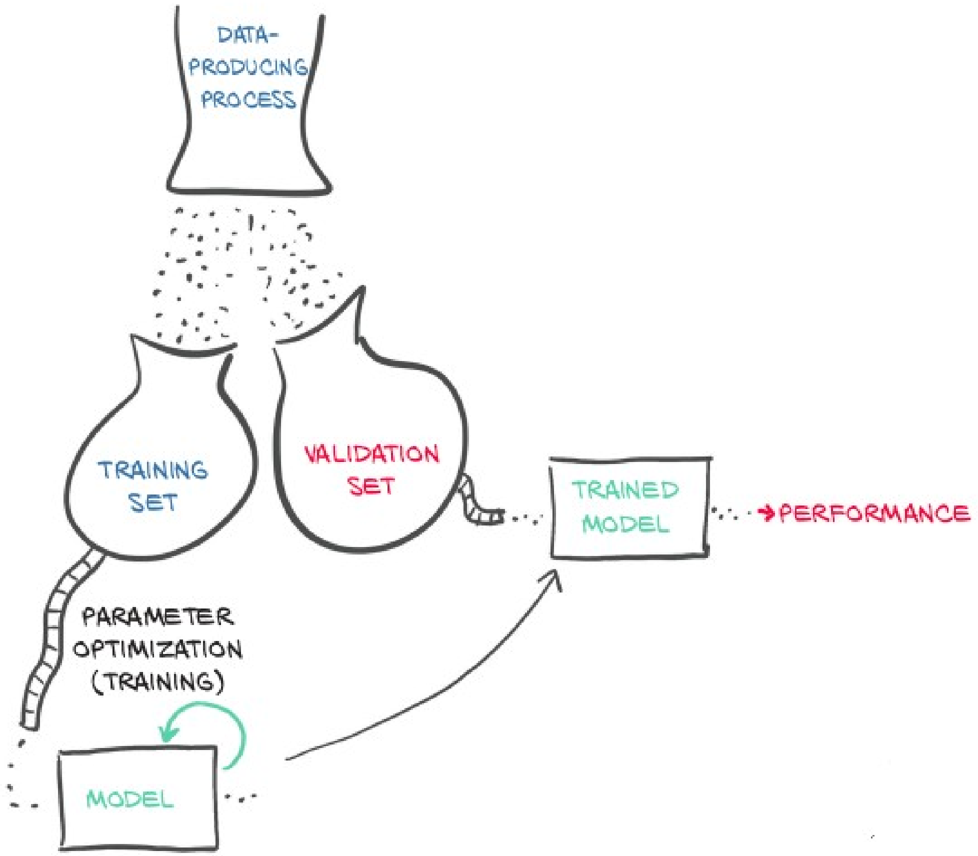


Image credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Evaluating the training loss

- The training loss will tell us if our model can fit the training set at all—in other words, if our model has enough capacity to process the relevant information in the data.
- A deep neural network can potentially approximate complicated functions, provided that the number of neurons, and therefore parameters, is high enough.
- The fewer the number of parameters, the simpler the shape of the function our network will be able to approximate.
- If the training loss is not decreasing, chances are the model is too simple for the data.
- The other possibility is that our data just doesn't contain meaningful information.

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Generalizing To The Validation Set

- Well, if the loss evaluated in the validation set doesn’t decrease along with the training set, it means our model is improving its fit of the samples it is seeing during training, but it is not ***generalizing*** to samples outside this precise set.
- As soon as we evaluate the model at new, previously unseen points, the values of the loss function are poor.
- So, if the training loss and the validation loss diverge, we’re overfitting.

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Example of Overfitting

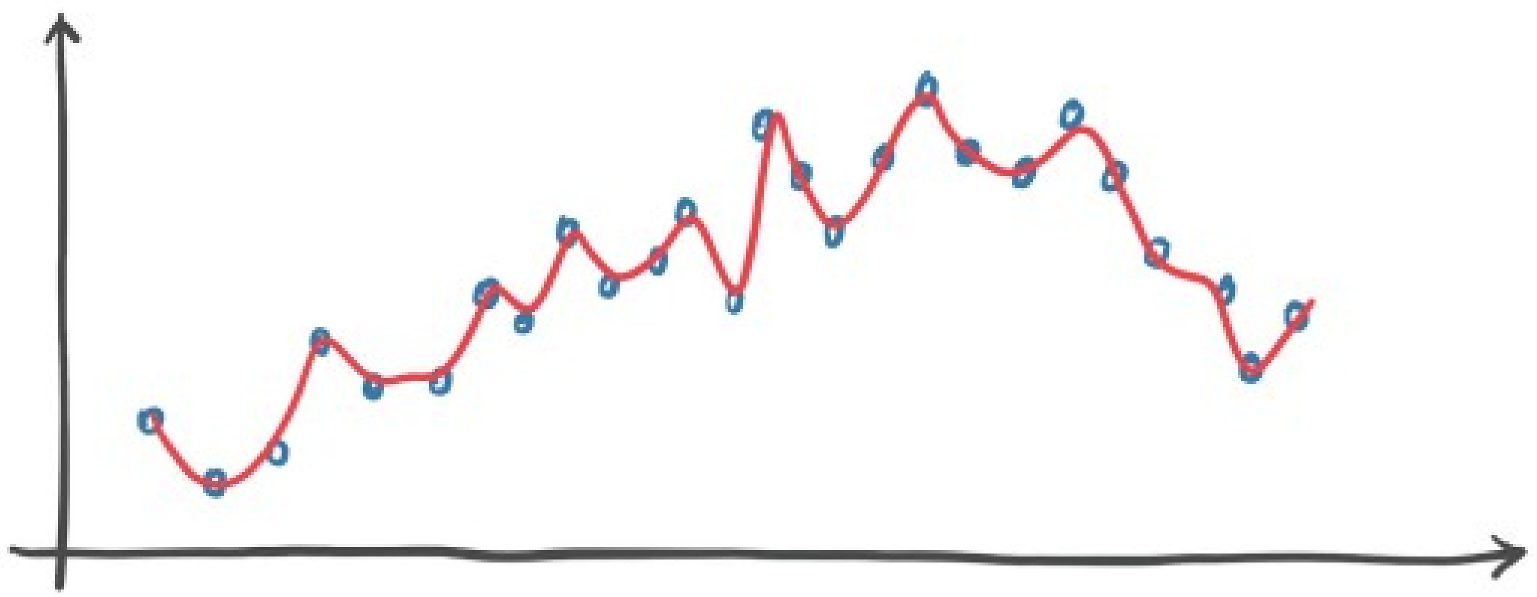
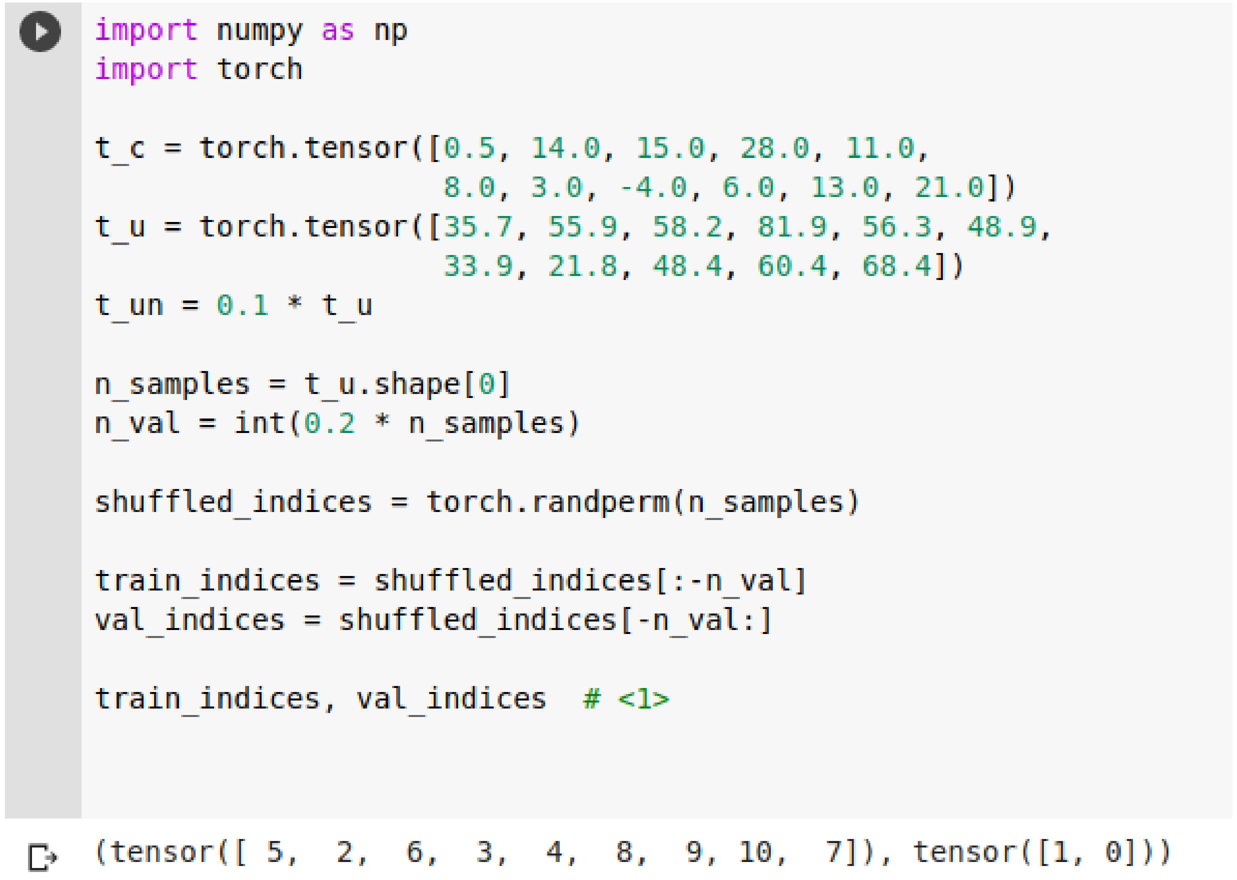


Image credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Example in PyTorch

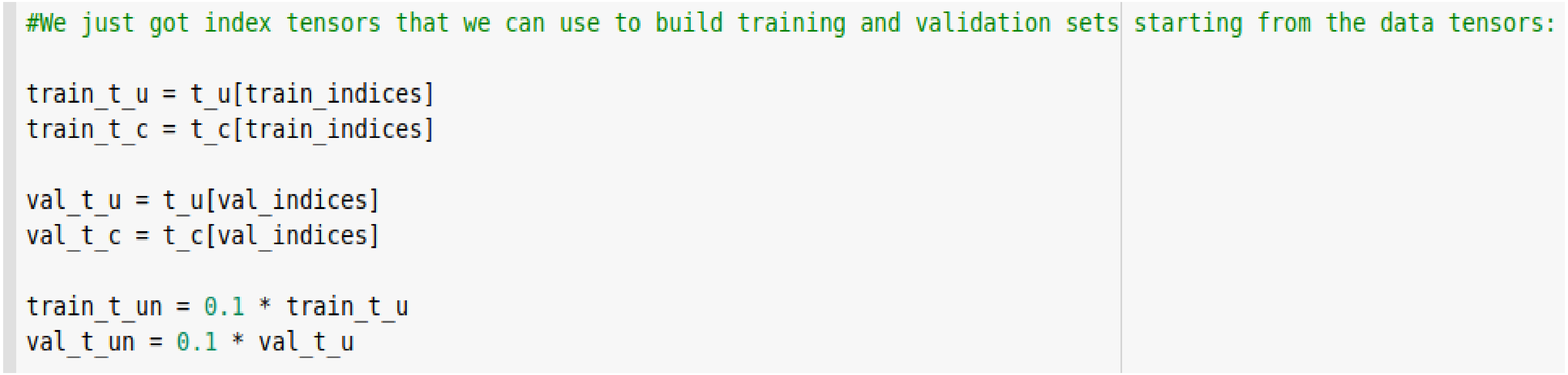


```
import numpy as np
import torch

t_c = torch.tensor([0.5, 14.0, 15.0, 28.0, 11.0,
                    8.0, 3.0, -4.0, 6.0, 13.0, 21.0])
t_u = torch.tensor([35.7, 55.9, 58.2, 81.9, 56.3, 48.9,
                    33.9, 21.8, 48.4, 60.4, 68.4])
t_un = 0.1 * t_u

n_samples = t_u.shape[0]
n_val = int(0.2 * n_samples)

shuffled_indices = torch.randperm(n_samples)

train_indices = shuffled_indices[:-n_val]
val_indices = shuffled_indices[-n_val:]

train_indices, val_indices  # <1>
```

```
(tensor([ 5,  2,  6,  3,  4,  8,  9, 10,  7]), tensor([1, 0]))
```

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

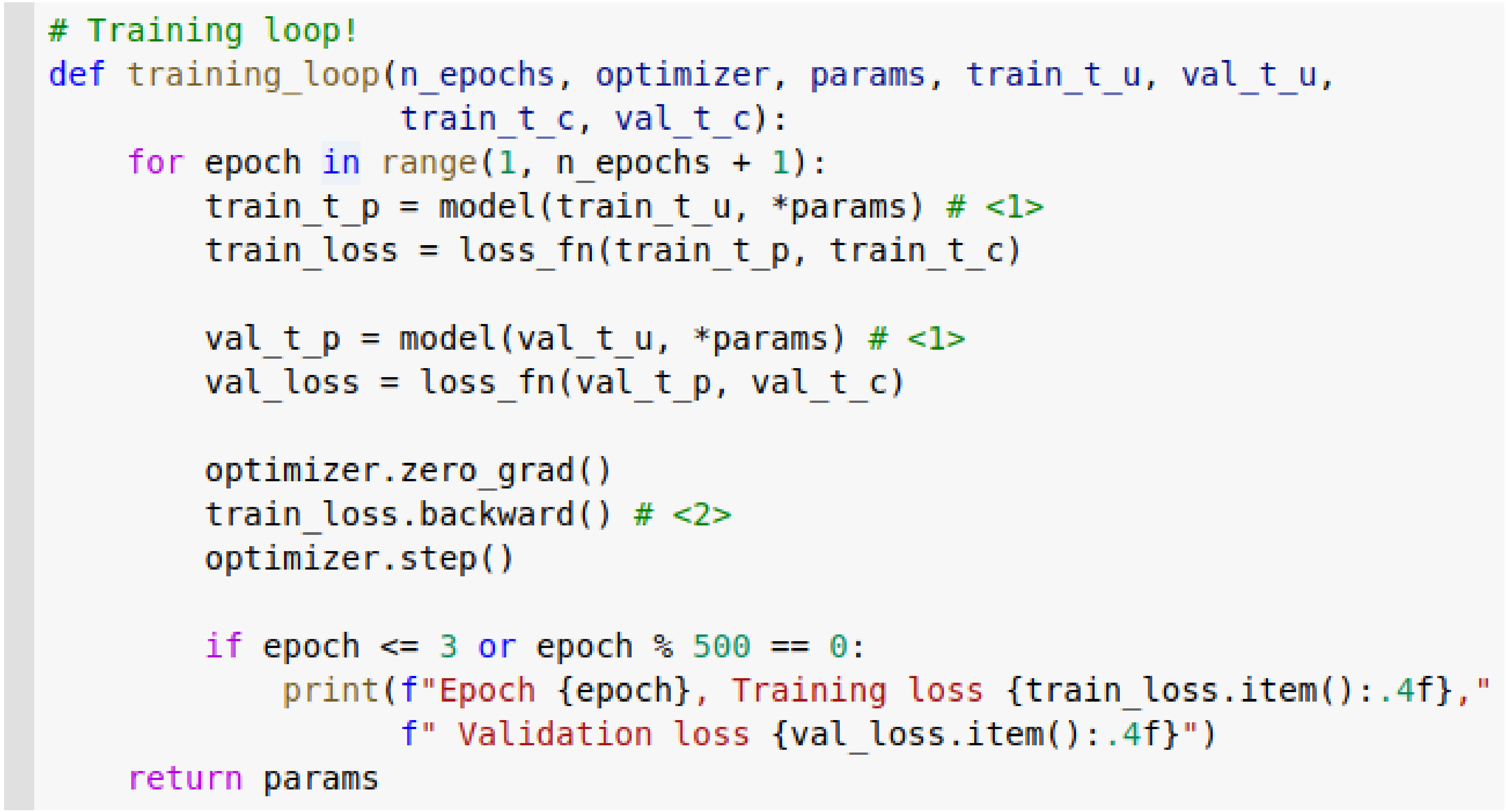
Example in PyTorch



```
#We just got index tensors that we can use to build training and validation sets starting from the data tensors:

train_t_u = t_u[train_indices]
train_t_c = t_c[train_indices]

val_t_u = t_u[val_indices]
val_t_c = t_c[val_indices]

train_t_un = 0.1 * train_t_u
val_t_un = 0.1 * val_t_u
```

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

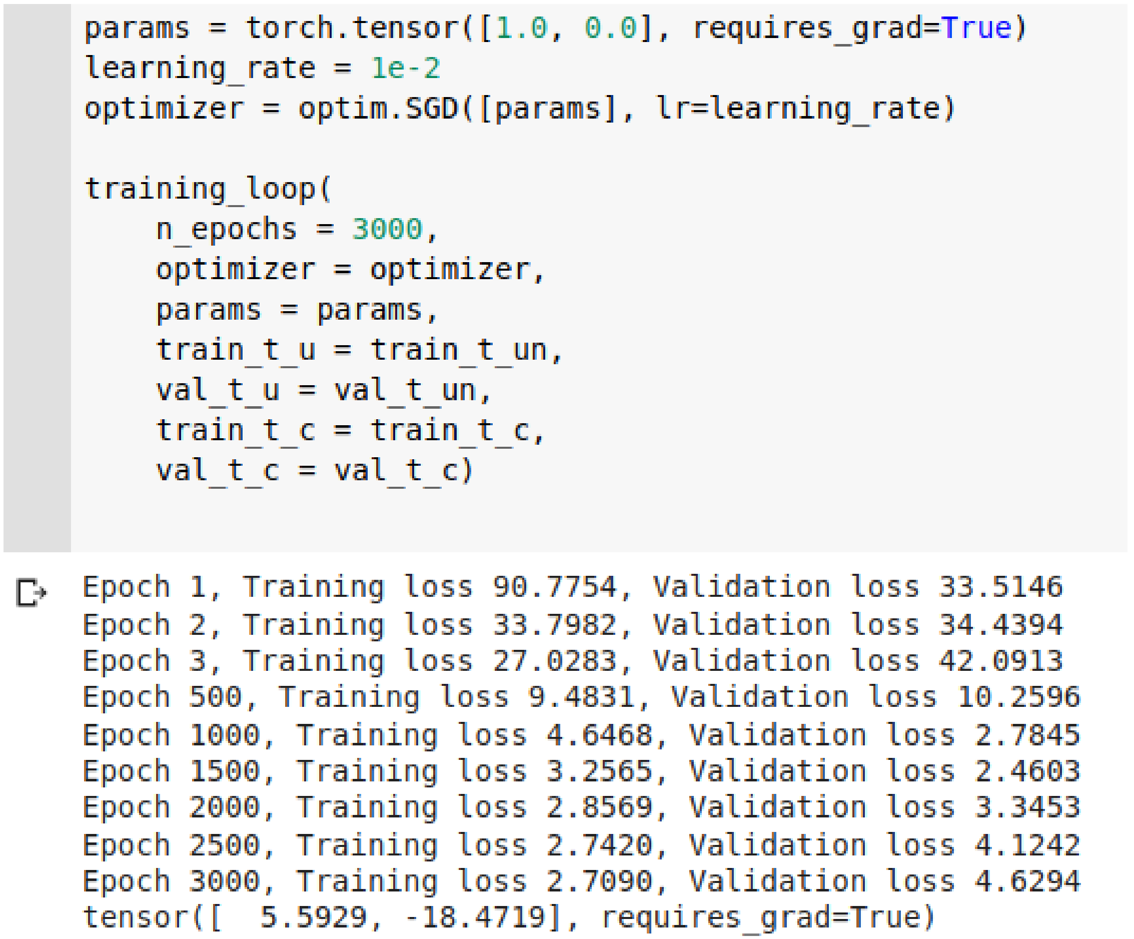
Example in PyTorch



```python
# Training loop!
def training_loop(n_epochs, optimizer, params, train_t_u, val_t_u,
                  train_t_c, val_t_c):
    for epoch in range(1, n_epochs + 1):
        train_t_p = model(train_t_u, *params) # <1>
        train_loss = loss_fn(train_t_p, train_t_c)

        val_t_p = model(val_t_u, *params) # <1>
        val_loss = loss_fn(val_t_p, val_t_c)

        optimizer.zero_grad()
        train_loss.backward() # <2>
        optimizer.step()

        if epoch <= 3 or epoch % 500 == 0:
            print(f"Epoch {epoch}, Training loss {train_loss.item():.4f},"
                  f" Validation loss {val_loss.item():.4f}")
    return params
```

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Example in PyTorch



```
params = torch.tensor([1.0, 0.0], requires_grad=True)
learning_rate = 1e-2
optimizer = optim.SGD([params], lr=learning_rate)

training_loop(
    n_epochs = 3000,
    optimizer = optimizer,
    params = params,
    train_t_u = train_t_un,
    val_t_u = val_t_un,
    train_t_c = train_t_c,
    val_t_c = val_t_c)
```

```
Epoch 1, Training loss 90.7754, Validation loss 33.5146
Epoch 2, Training loss 33.7982, Validation loss 34.4394
Epoch 3, Training loss 27.0283, Validation loss 42.0913
Epoch 500, Training loss 9.4831, Validation loss 10.2596
Epoch 1000, Training loss 4.6468, Validation loss 2.7845
Epoch 1500, Training loss 3.2565, Validation loss 2.4603
Epoch 2000, Training loss 2.8569, Validation loss 3.3453
Epoch 2500, Training loss 2.7420, Validation loss 4.1242
Epoch 3000, Training loss 2.7090, Validation loss 4.6294
tensor([  5.5929, -18.4719], requires_grad=True)
```

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

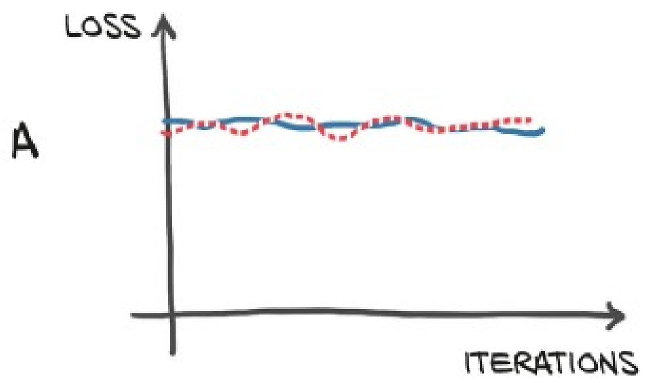
Training Loss and Validation Loss

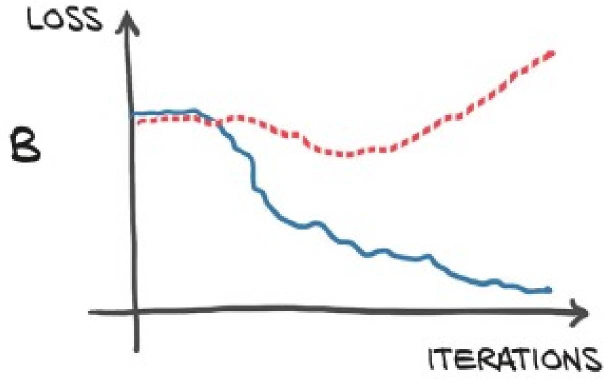
- Our main goal is to also see both the training loss and the validation loss decreasing.

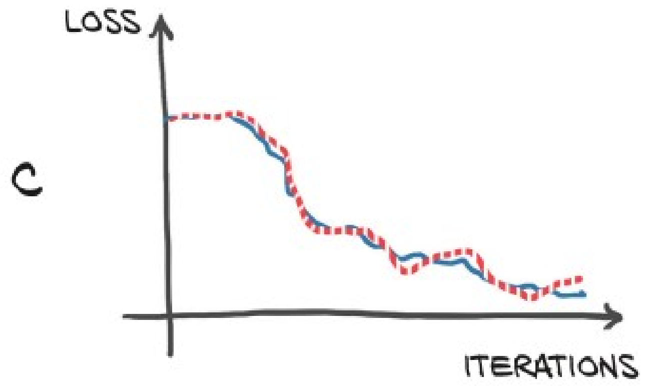
- While ideally both losses would be roughly the same value, as long as the validation loss stays reasonably close to the training loss, we know that our model is continuing to learn generalized things about our data.

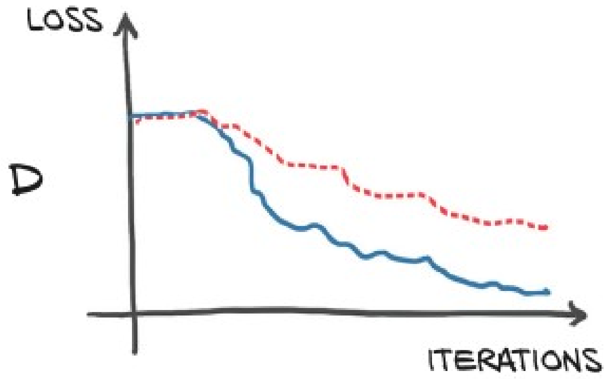
Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Training Loss and Validation Loss









Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

Training Loss and Validation Loss

Look into the figure given in the previous slide!

(A) Training and validation losses do not decrease; the model is not learning due to no
information in the data or insufficient capacity of the model.

(B) Training loss decreases while validation loss increases: overfitting.

(C) Training and validation losses decrease exactly in tandem. Performance may be improved further as the model is not at the limit of overfitting.

(D) Training and validation losses have different absolute values but similar trends: overfitting is under control.

Slide credit: E. STEVENS, L. ANTIGA, and T. VIEHMANN

References

- All the contents present in the slides are taken from various online resources. Due credit is given in the respective slides.These slides are used for *academic* purposes only.