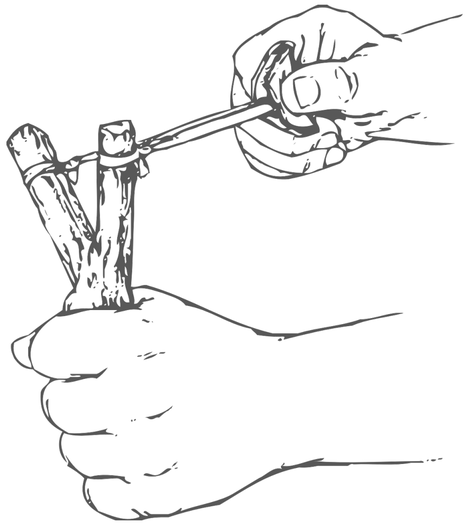
Neural·Pragmatic
Natural
Language
Generation

N·P
NLG

Learning goals



1. Understand what PyTorch is good for.
2. Ability to create, access and manipulate tensors.
3. Understand parameterized model predictions.
4. Understand of what ‘parameter optimization’ is.
5. Ability to use stochastic gradient descent to optimize parameters in PyTorch.

Key features

PyTorch

high-level framework for ML

- specifically artificial neural networks

efficient tensor algebra

- ability to run on GPUs etc.

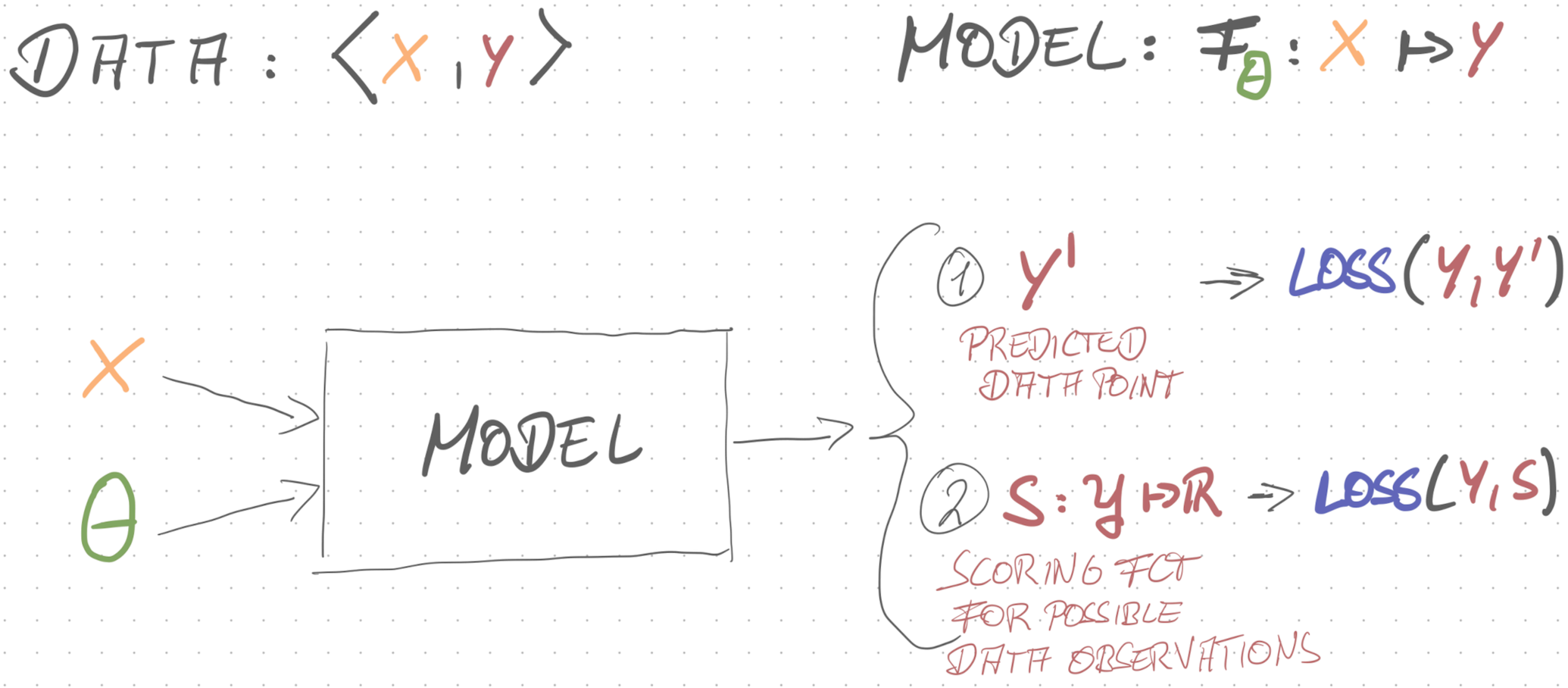
pre-defined building blocks for ANNs

- standard layers, data handling etc.

automatic differentiation

- enables efficient optimization

Models, parameters, predictions & loss



Anatomy of a training step

1. compute predictions

   what do we predict in the current state?

2. compute the loss

   how good is this prediction (for the training data)?

3. backpropagate the error

   in which direction would we need to change the relevant parameters to make the prediction better?

4. update the parameters

   change the parameters (to a certain degree, the so-called learning rate) in the direction that should make them better

5. zero the gradients

   reset the information about “which direction to tune” for the next training step

```
nTrainingSteps= 10000
for i in range(nTrainingSteps):
    pred =  torch.distributions.Normal(loc=location,
                                       scale=1.0)
    loss = -torch.sum(prediction.log_prob(trainData))
    loss.backward()
    if (i+1) % 500 == 0:
        opt.step()
        opt.zero_grad()
```