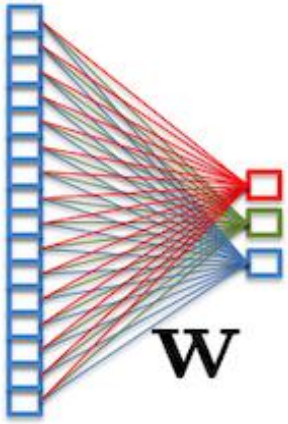
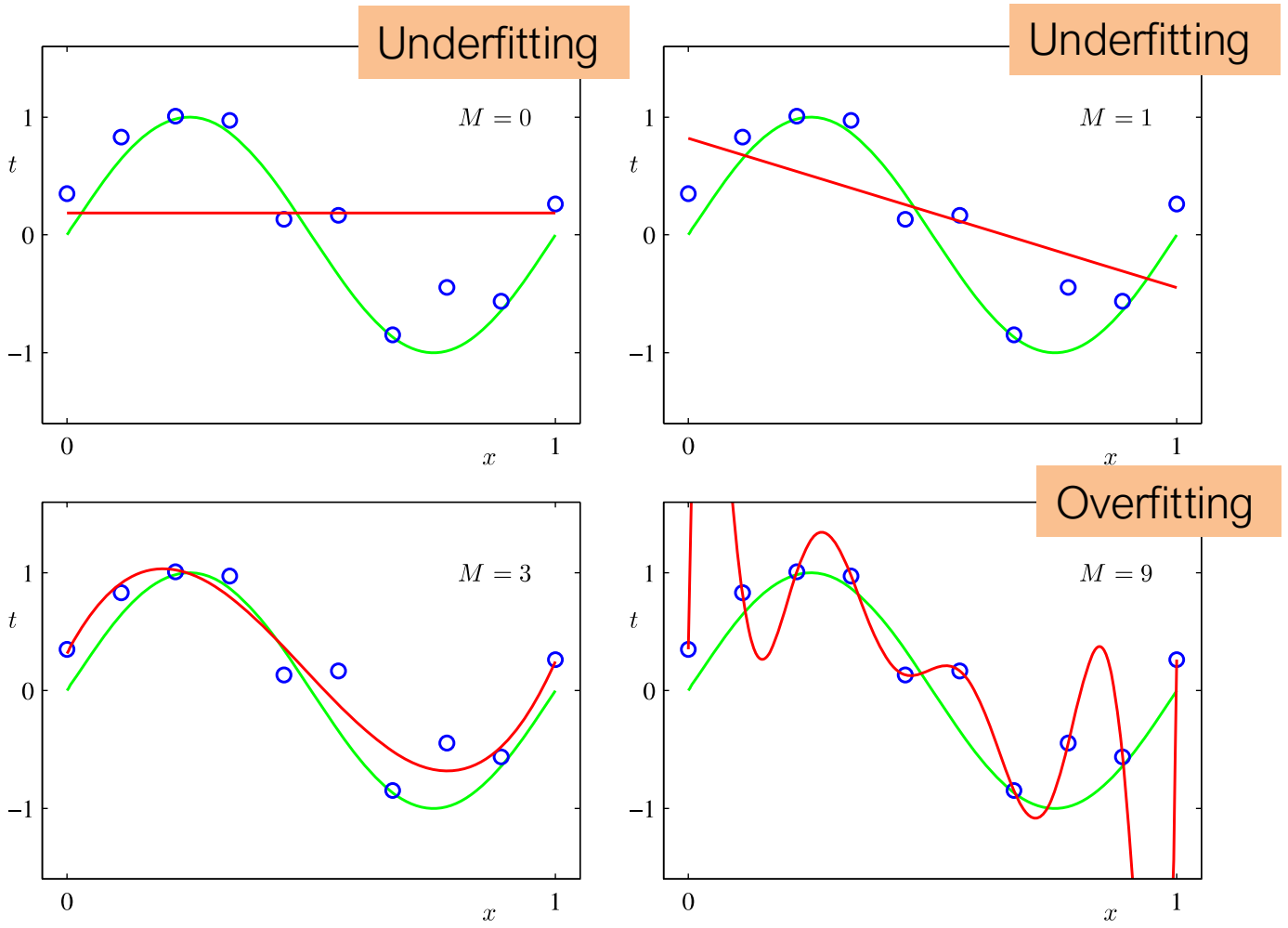
AI and Data Analysis

2.4 Train/Validation/Test split - Tensorboard



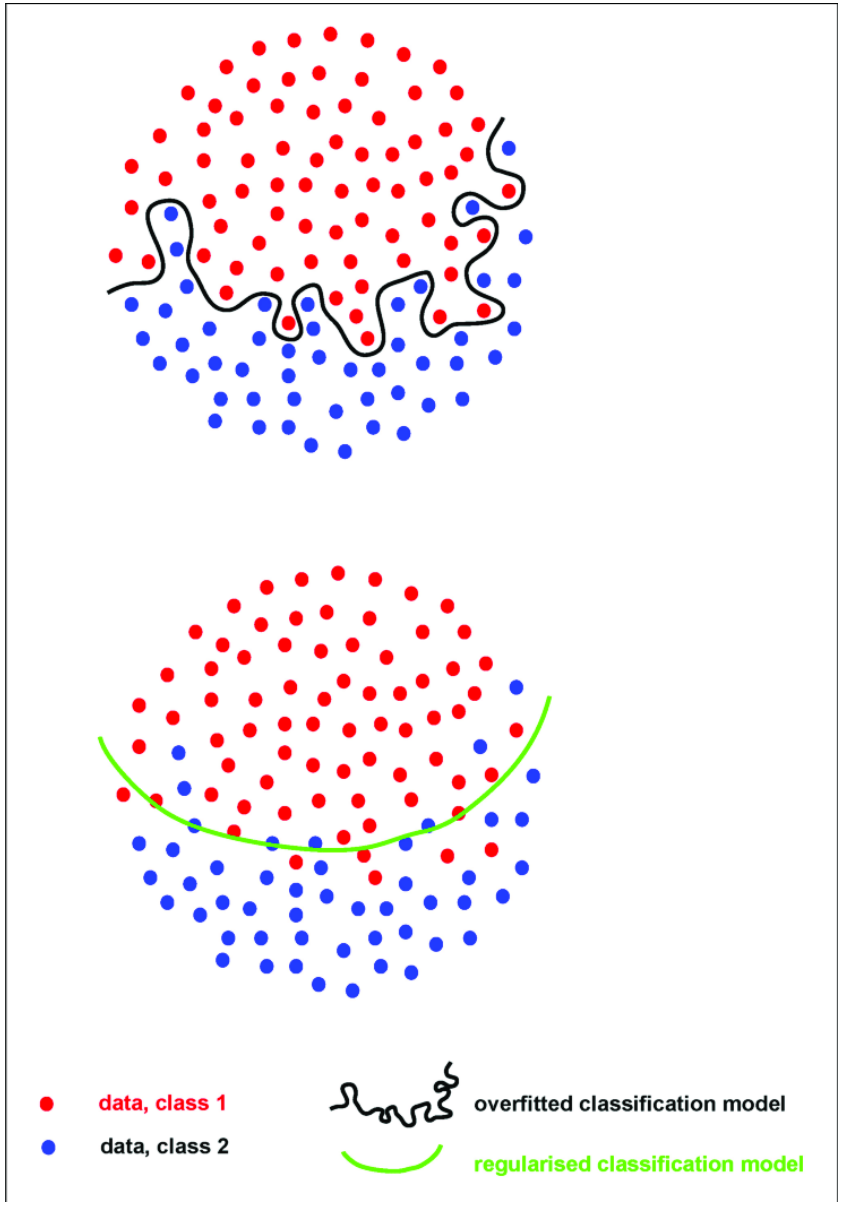
Christian Wolf

Under- and Overfitting in Regression



[C. Bishop, Pattern recognition and Machine learning, 2006]

Overfitting in classification



[Figure: Tim Hulsen]

We can overfit on what exactly?

We would like to learn to predict a value y from observed input x

$$y = h(x, \theta)$$

Handcrafted from domain knowledge (e.g. type of model, number of layers, number of units: **architecture)**

Learned from training data

Optimization over the training data

We optimize a loss function $\mathcal{L}$ taking ground-truth targets t and predictions y of a model h over its parameters θ:

$$\theta = \arg\min_{\theta} \mathcal{L}(h(x, \theta), t)$$

Fitting is thus the result of the optimization process ... and every optimization on data process can lead to potential overfitting.
We overfit on the data on which we optimize (here: training data).
The validation data provides a better estimate of the error of the model.

Validation (hold-out) set

We remove parts of the data from training, and use it for evaluation, to check whether we overfit.

Training data

Validation data

Optimization over the validation data

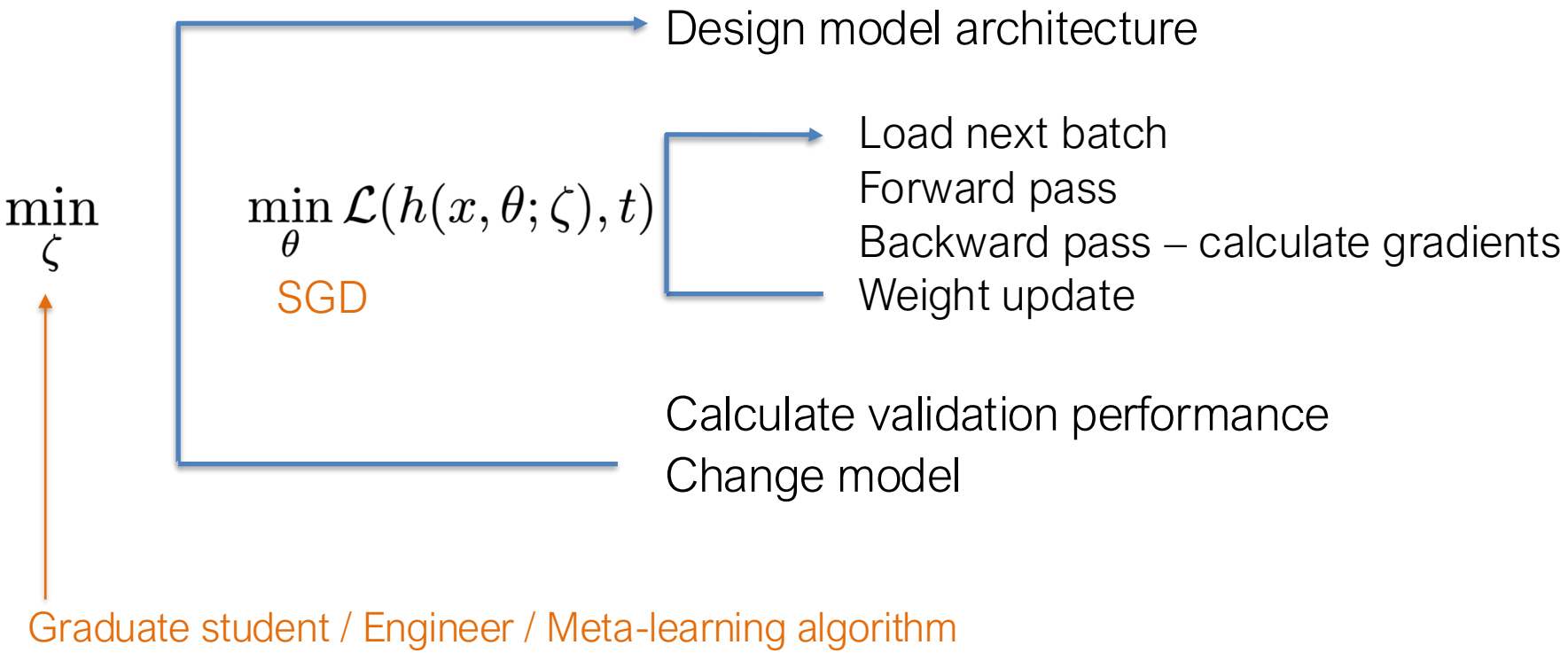
Problem: while working on the problem, we select and optimize network architectures, so a better notation for the function h would be:

$$h(x, \theta; \zeta)$$

where ζ are the hyper-parameters of the network.
In practice, we really solve the following problem:

$$\min_{\zeta} \min_{\theta} \mathcal{L}(h(x, \theta; \zeta), t)$$

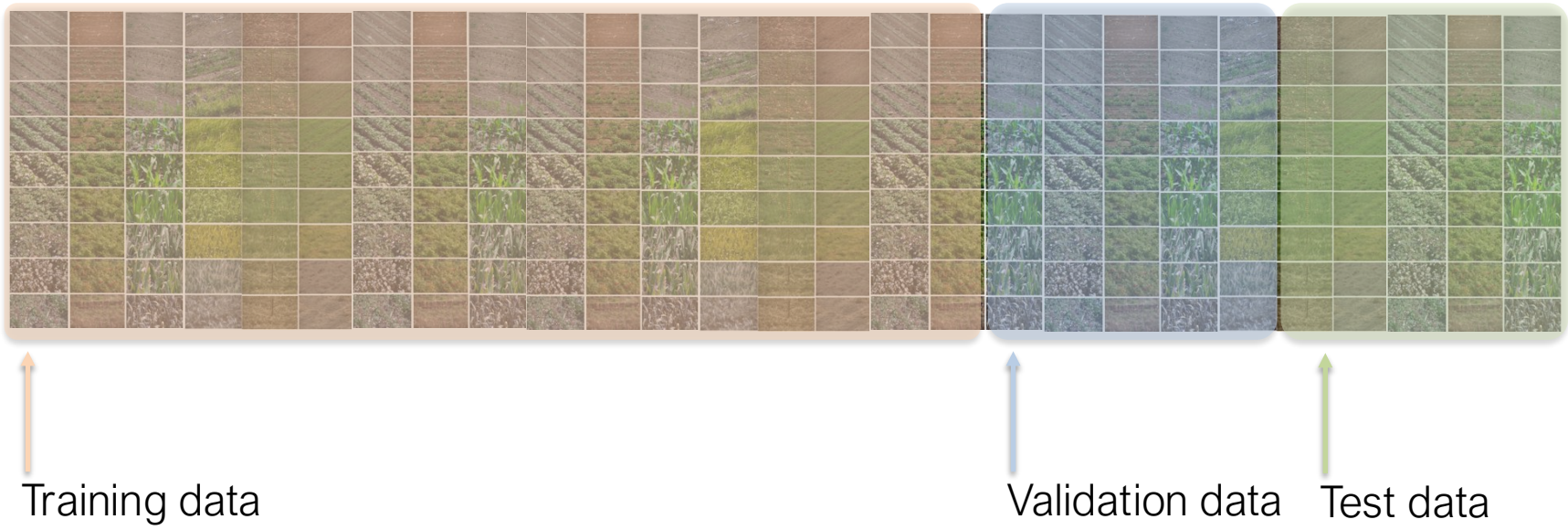
Optimization loops



Validation performance is NOT representative of the performance of the model in the wild! We overfit on the validation data!

Validation (hold-out) set

We remove parts of the data from training, and use it for evaluation, to check whether we overfit.



Visualization of the training process

Running Tensorboard

Install:

```
pip3 install tensorboard
```

PyTorch code:

```
from torch.utils.tensorboard import SummaryWriter

# Setup a writer and configure a log file directory
writer = SummaryWriter('runs/mnist_mlp_r0.01')
```

Start the tensorboard exectuable

```
tensorboard --logdir=runs
```

The executable sets up a web server on port 6006:

```
TensorBoard 2.0.0 at http://localhost:6006/ (Press CTRL+C to
    quit)
```

Running Tensorboard

We load two different datasets, a training and a validation set:

```
# Training set and loader
valid_dataset = MNISTDataset ("MNIST-png/testing",
    transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))]))
valid_loader = torch.utils.data.DataLoader(
    valid_dataset,
    batch_size=BATCHSIZE, shuffle=True)

# Validation set and loader ("hold out" set)
train_dataset = MNISTDataset ("MNIST-png/training",
    transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))]))
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=BATCHSIZE, shuffle=True)
```

Writing into Tensorboard

```
for epoch in range(100):
  for batch_idx, (data, labels) in enumerate(train_loader):

        (...)

        # Print statistics
        if (batch_idx % STATS_INTERVAL) == 0:
          train_err=100.0*(1.0-running_correct/count)

          # Call a second loop which iterates over
          # validation batches, calculating error
          vloss, verr = calcError (model, valid_loader)

          # Write statistics to the log file
          writer.add_scalars ('Loss', {
            'training:': running_loss / STATS_INTERVAL,
            'validation:': vloss },
            epoch * len(train_loader) + batch_idx)

          writer.add_scalars ('Error', {
            'training:': train_err,
            'validation:': verr },
            epoch * len(train_loader) + batch_idx)
```

TensorBoard SCALARS GRAPHS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0,474

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

- mnist_mlp_r0.001/Error_validation:
- mnist_mlp_r0.01
- mnist_mlp_r0.01/Loss_training:
- mnist_mlp_r0.01/Loss_validation:
- mnist_mlp_r0.01/Error_training:
- mnist_mlp_r0.01/Error_validation:
- mnist_mlp_r0.01_h600
- mnist_mlp_r0.01_h600/Loss_training:
- mnist_mlp_r0.01_h600/Loss_validation:
- mnist_mlp_r0.01_h600/Error_training:
- mnist_mlp_r0.01_h600/Error_validation:

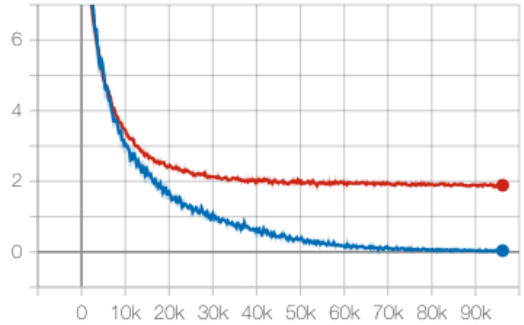
TOGGLE ALL RUNS

runs

Filter tags (regular expressions supported)

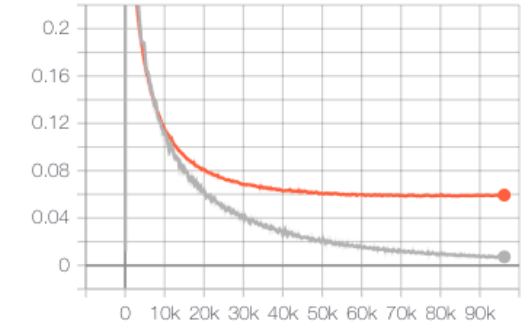
Error

Error



Loss

Loss



Logistic Regression underfits