Deep Learning

LM Computer Science, Data Science, Cybersecurity

2nd semester - 6 CFU

References:

Deep Learning Book (main book)

Mitchell (machine learining concepts)

Bishop (machine learning)

Practical Methodology (chapter 11), i.e. How to use deep learning

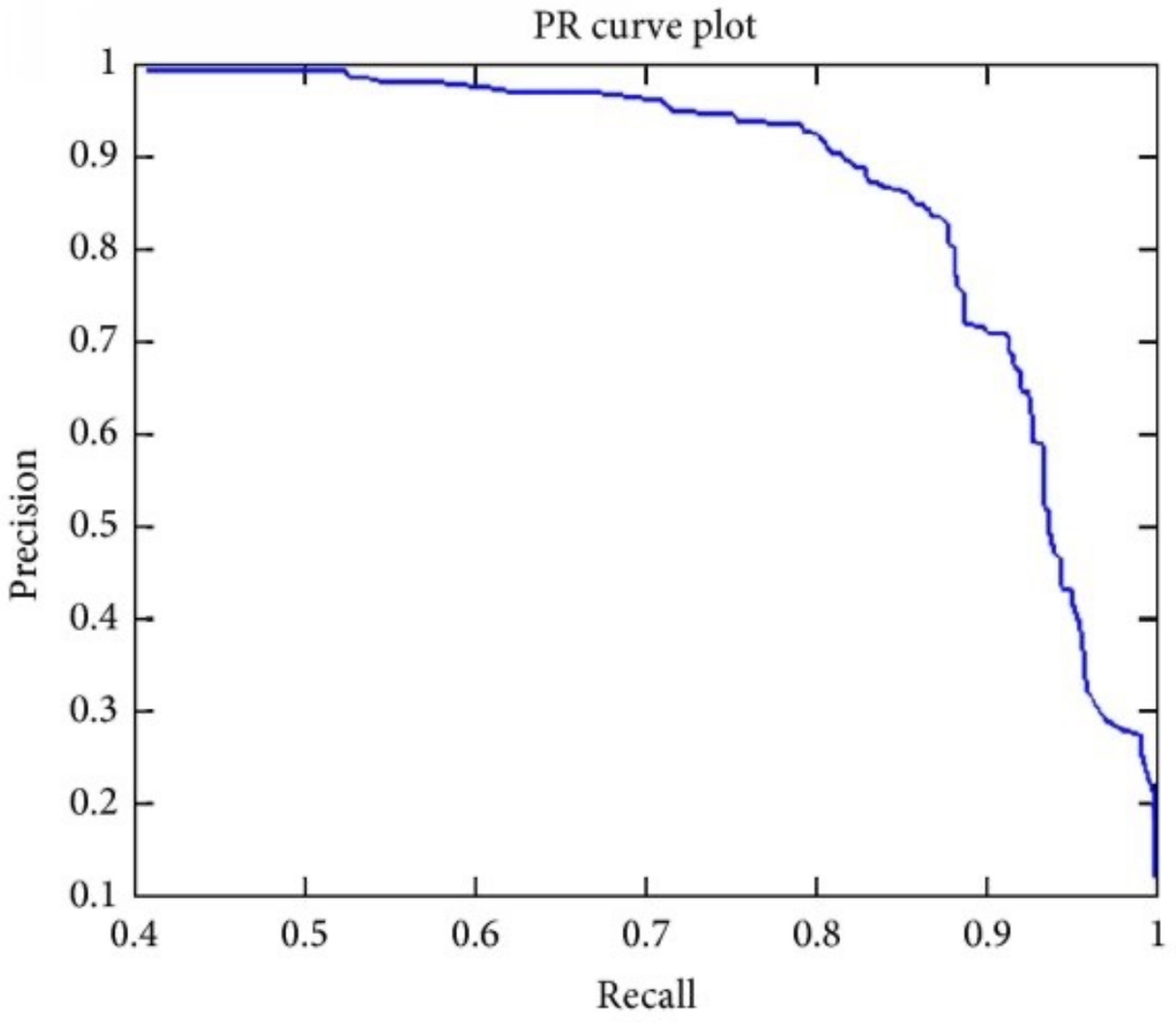
General methodology

- Determine your goals
  - error metrics, target value
- Build an end-to-end pipeline, including:
  - data ingestion, model training
  - and the estimation of the performance metrics
- Be prepared to analyse your system, to understand:
  - which component is performing worse than expected
  - if poor performance is due to underfitting, overfitting or a bug
- Repeatedly make incremental changes (one at a time!)
  - E.g. gathering new data, changing hyperparameters, changing algorithms based on specific observations
  - Understand the effect of each change

Performance metrics

- Decide which error metric to use
  - This metric will drive all future decisions
  - Different from loss: e.g. error rate
  - Often different errors should be weighted differently (e.g. spam classification)
- Detection of rare events:
  - Consider a disease that only one in 1M people have
  - 99.9999% accuracy simply predicting that nobody has the disease
  - **Precision**: fraction of detections that are correct
  - **Recall**: fraction of true events that are detected
  - If the classifier outputs a **score**, metrics depend on a **threshold**

Precision/Recall curve: varying the threshold



Ex. ID	Prediction	Target
5	0.99	1
10	0.97	1
7	0.96	0
1	0.86	1
	…	
42	0.1	1
9	005	0

- Single number summarizing the performance (without fixing the threshold): **Area under Precision/Recall curve**
- If we fix the threshold: $Fscore = \frac{2*precision*recall}{precision+recall}$

Desired level of performance

- Decide a desired level of performance
  - Based on the application (e.g. minimum accuracy for the system to be useful/trusted/safe in practice)
  - Previously published results
  - Zero error is often not possible!
    - Bayes error rate: lowest possible error rate for any classifier (seen in Machine Learning courses, or check it online)
  - May be limited by finite training data

Coverage

- In some applications, it is possible for the machine learning system to refuse to make a decision
- Useful when a wrong decision can be harmful and/or if a human operator is able to occasionally take over
  - Note that the machine learning algorithm should be able to estimate how confident it is about a decision
- Coverage: the fraction of examples for which the machine learning system is able to produce a response.
  - It is possible to trade coverage for accuracy.
  - 100% accuracy by refusing to process any example, but this reduces the coverage to 0%.

Baseline models

- Baselines are simple Neural networks chosen to provide a first estimation of the achievable level of performance
- Depending on the complexity of the problem, you may start with machine learning (not deep learning) algorithms
- Define a baseline NN exploiting your knowledge:
  - if your data has a structure, exploit it (e.g. CNNs on images, RNNs on sequence data)
  - Use piecewise linear units (e.g. ReLU)
  - Chose e relatively simple architecture
  - Choose a standard learning algorithm: SGD with momentum and decaying learning rate, or Adam
  - Include regularization (e.g. Dropout or weight decay)

Search for similar tasks

- If the task is similar to another task that has been studied:
  - Use the same model and algorithms that have been used for the other task
  - Use a pre-trained model
- The pre-trained model can also have been trained with self-supervised learning

Determine whether to gather more data

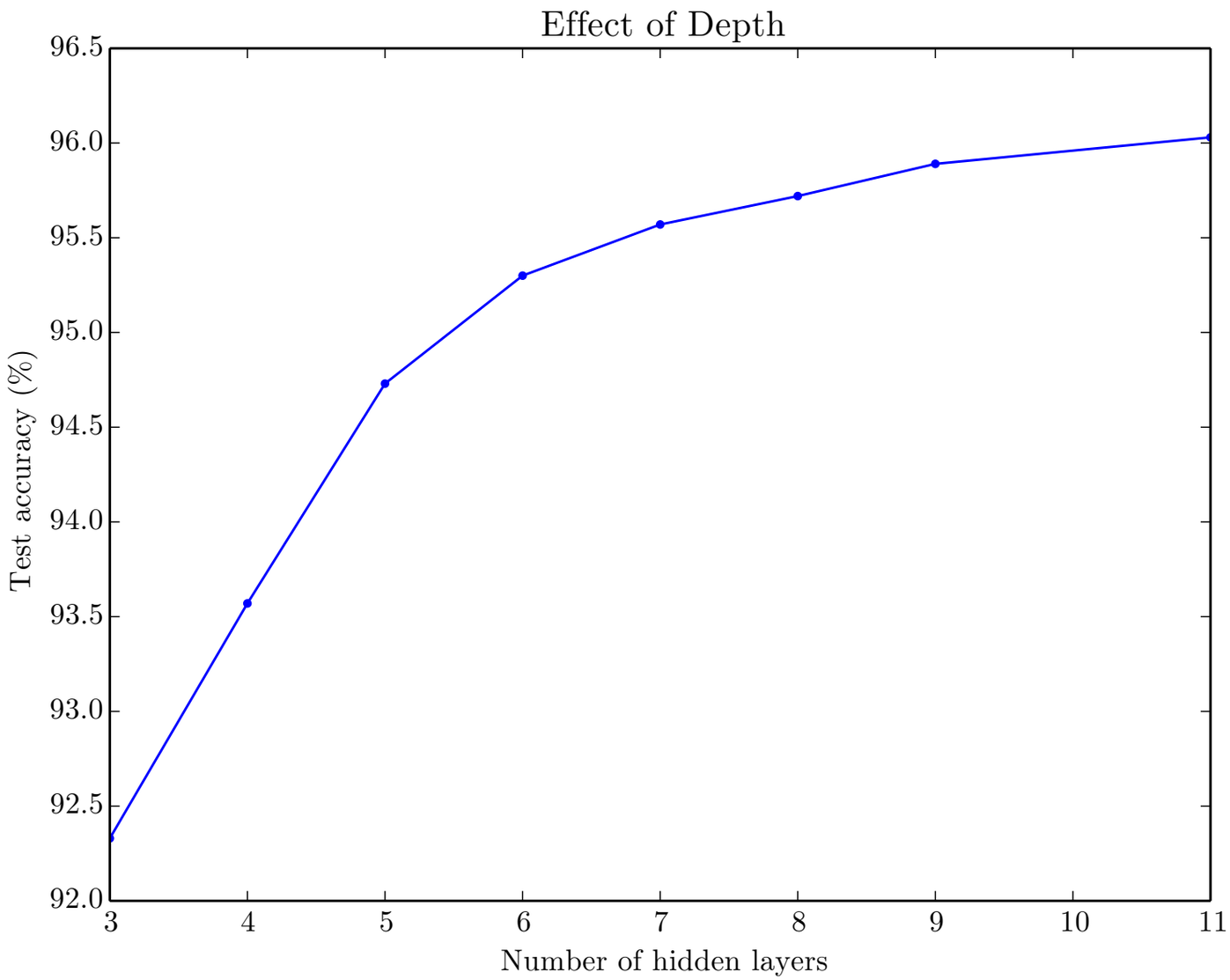
- After training the first models, measure their performance and determine how to improve it
- Find out if the **training** performance is acceptable
  - Low training performance -> You are not exploiting the available data well. Increase model capacity: more layers or more units (**Hyperparameters**)
    - At the extreme, your data may be too noisy or not informative enough (or there are problems in the data: **check** by yourself)
      - Re-collect the data by measuring more variables or fixing the problems
- Measure **test** performance
  - If it is good, you are done
  - If it is not, try to decrease model capacity
    - Reduce the size of the model (**Hyperparameters**)
    - Add regularization
  - If the generalization gap is still high, gather more data if possible

Defects in data

- Can a human process it?

26624

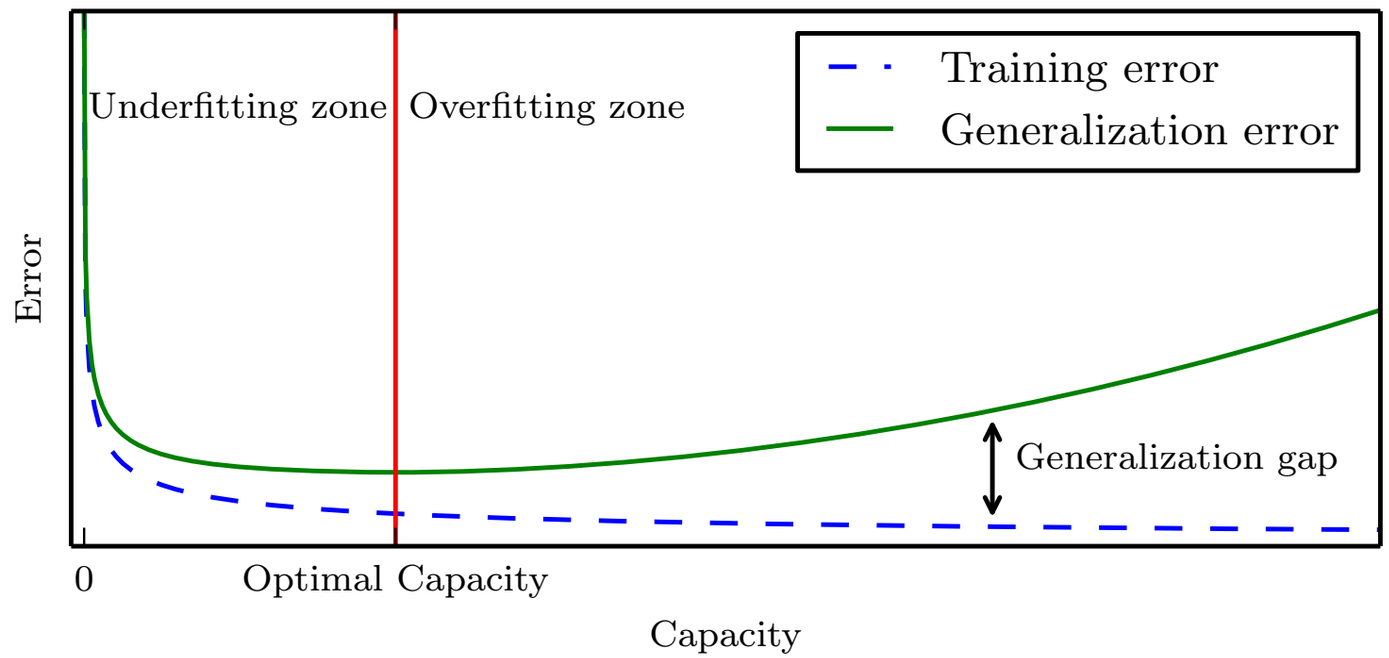
Model capacity



Hyperparameter selection

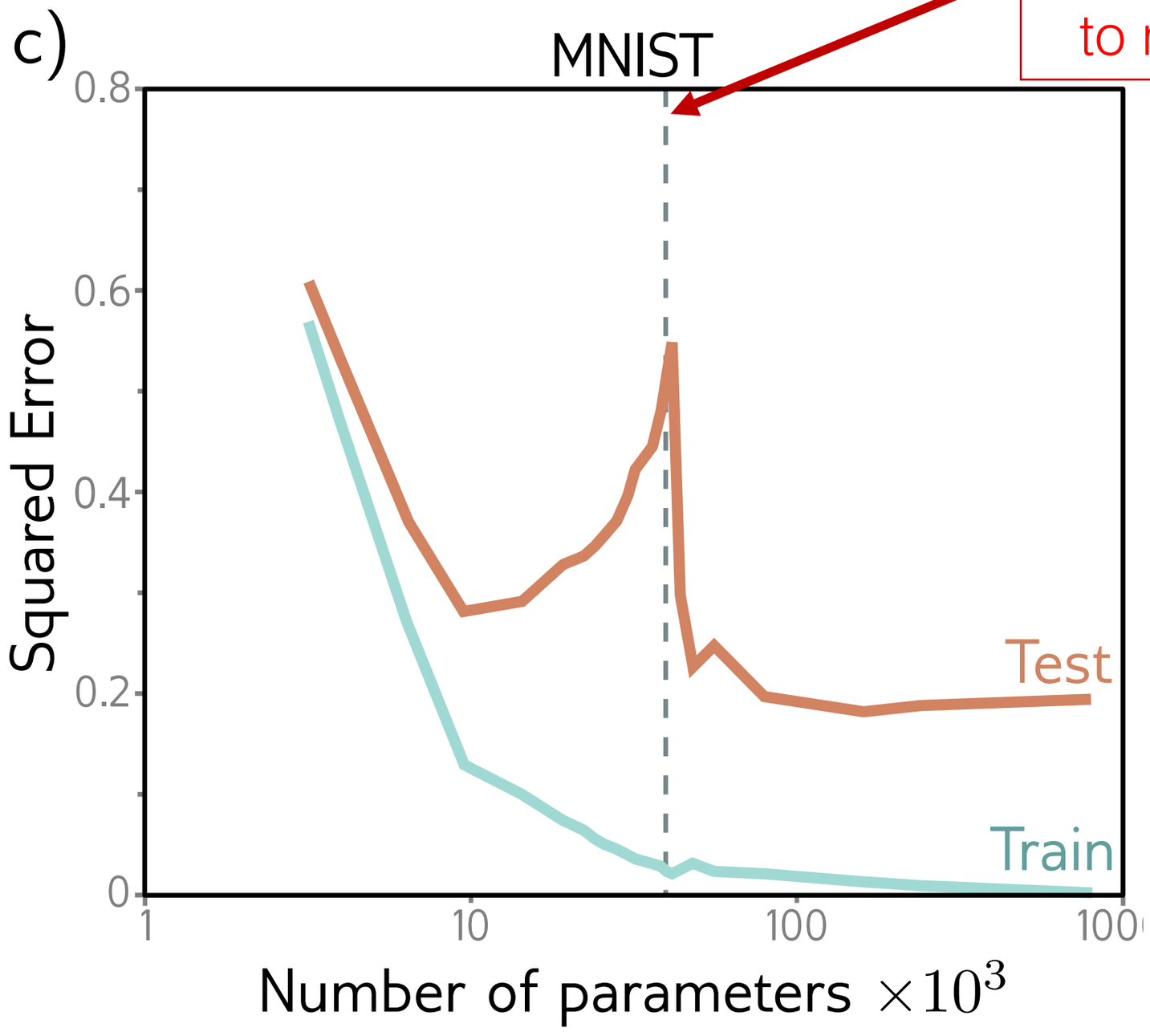
- Deep learning models are sensitive to hyperparameters
- Learning rate can be tuned on the **training** set
  - All the others on the **validation** set
- Usually the performance follows a U-shaped curve for each hyperparameter
  - Optimal model capacity lies between underfitting and overfitting zones
  - Usually, high-capacity models well regularized perform best
- **Intuition** should drive the search for hyperparameters

Actually, not the wole story →



Double Descent

Interpolating threshold: the model has just enough capacity to memorize the training data



Double Descent - intuition

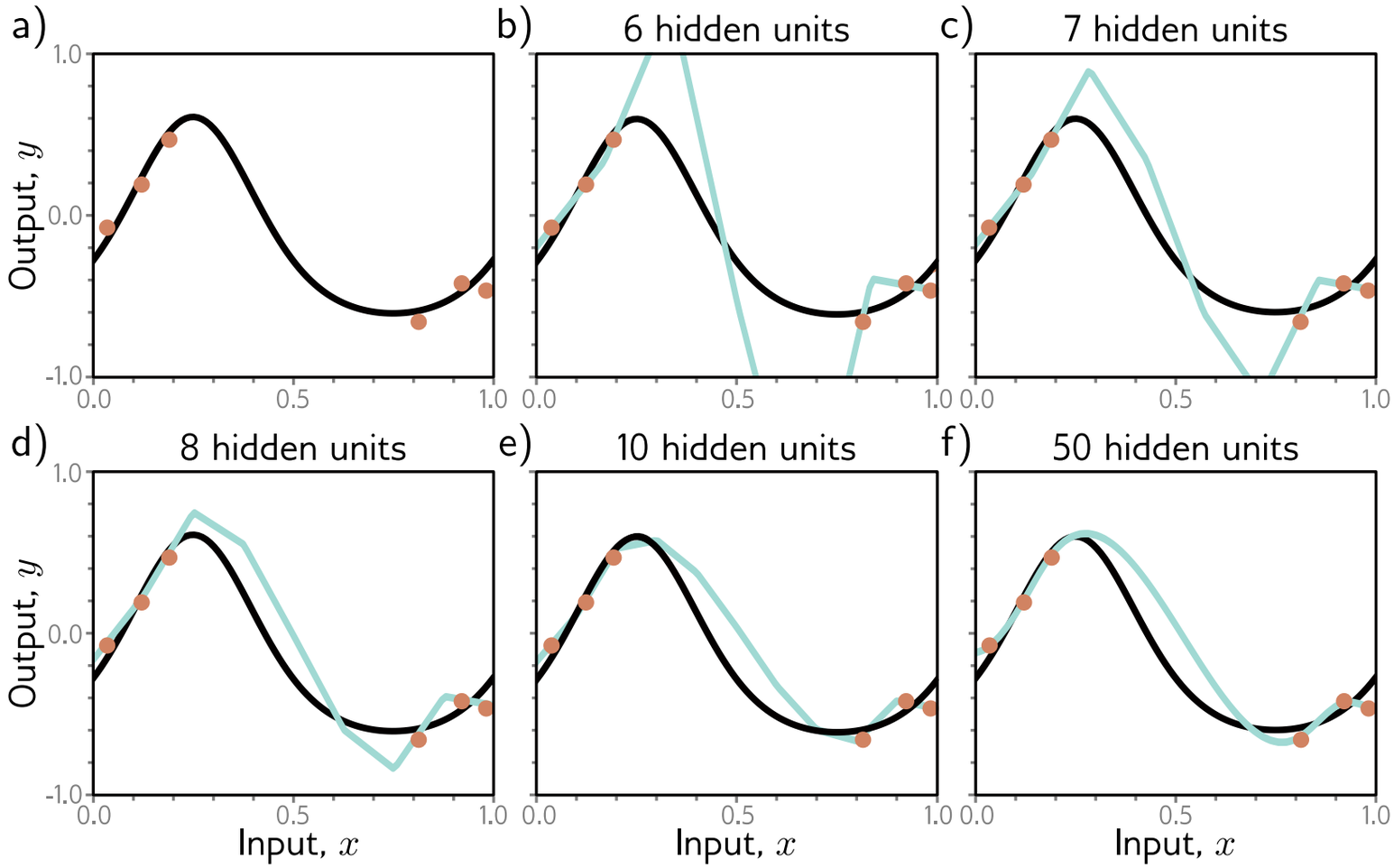
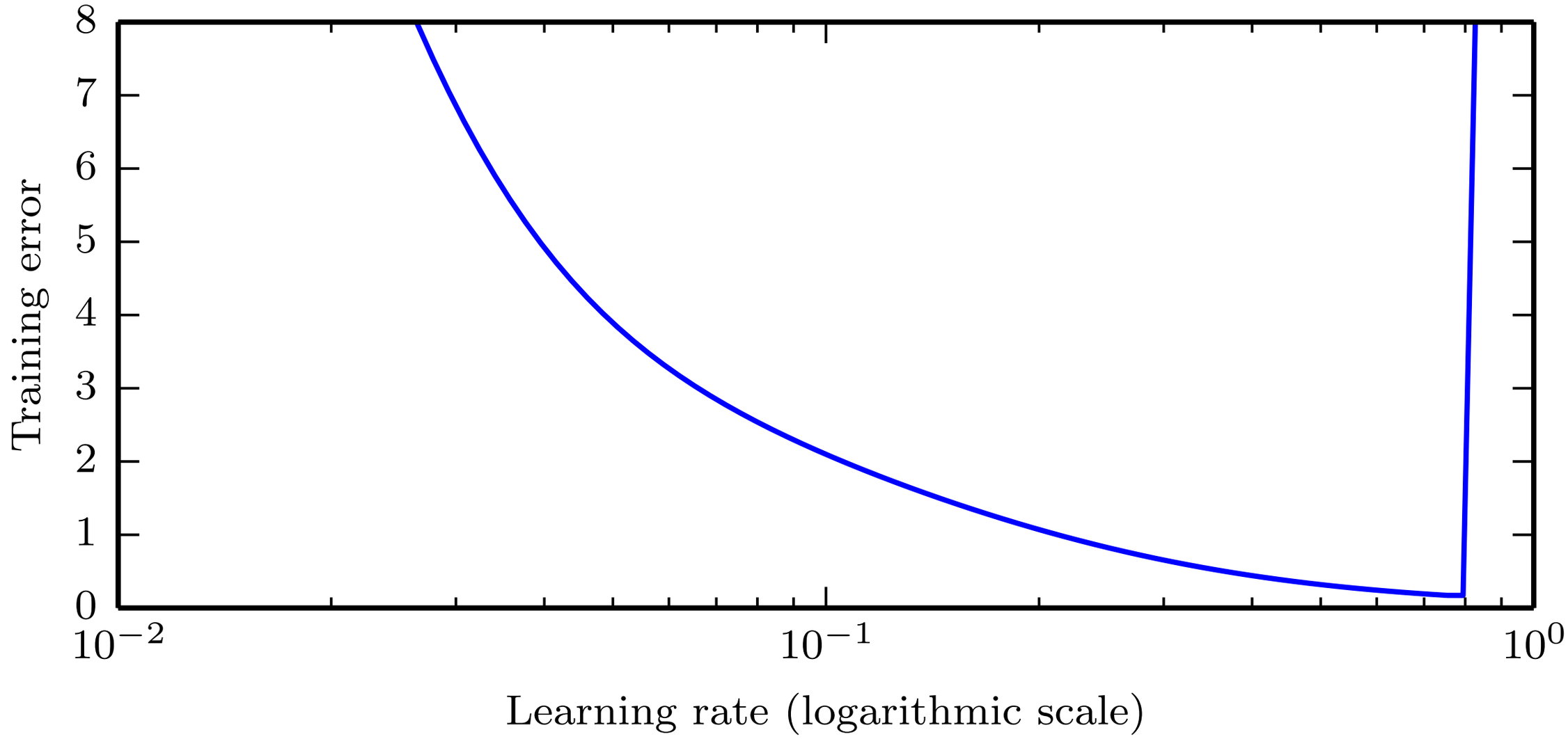


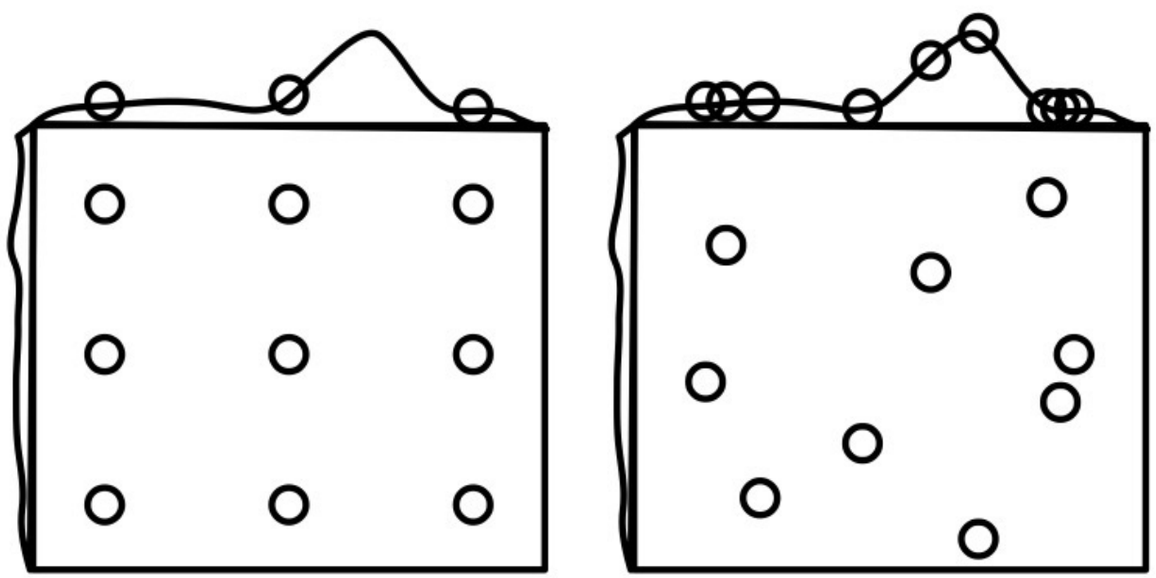
Figure 8.11 Increasing capacity (hidden units) allows smoother interpolation between sparse data points. a) Consider this situation where the training data (orange circles) are sparse; there is a large region in the center with no data examples to constrain the model to mimic the true function (black curve). b) If we fit a model with just enough capacity to fit the training data (cyan curve), then it has to contort itself to pass through the training data, and the output predictions will not be smooth. c–f) However, as we add more hidden units, the model has the *ability* to interpolate between the points more smoothly (smoothest possible curve plotted in each case). However, unlike in this figure, it is not obliged to.

Learning rate on Training set



Automatic hyper-parameter selection

- You just select the **range** for each hyperparameter
- **Grid** search: Test many different pre-defined options
- **Random** search: usually just some of the hyperparameters have a significant impact in performance
  - It can be more efficient than grid search
- Other approaches: e.g. model-based (Bayesian)



Grid Random

Debug your network

- Visualize the outputs of the model
  - E.g. object detection/classification, speech synthesis
- Visualize the worst mistakes
- If training error is high, try to overfit a tiny dataset
- Check the gradient (exploding or vanishing)
  - If you implemented the derivatives, compare them with numerical derivatives
- Check the activations: how many neurons fire?
  - ReLU can induce “dead” neurons

Example

- In the book (chapter 11), an example of multi-digit number recognition (from Google) is described, check it out.