# Introduction to Neural Networks

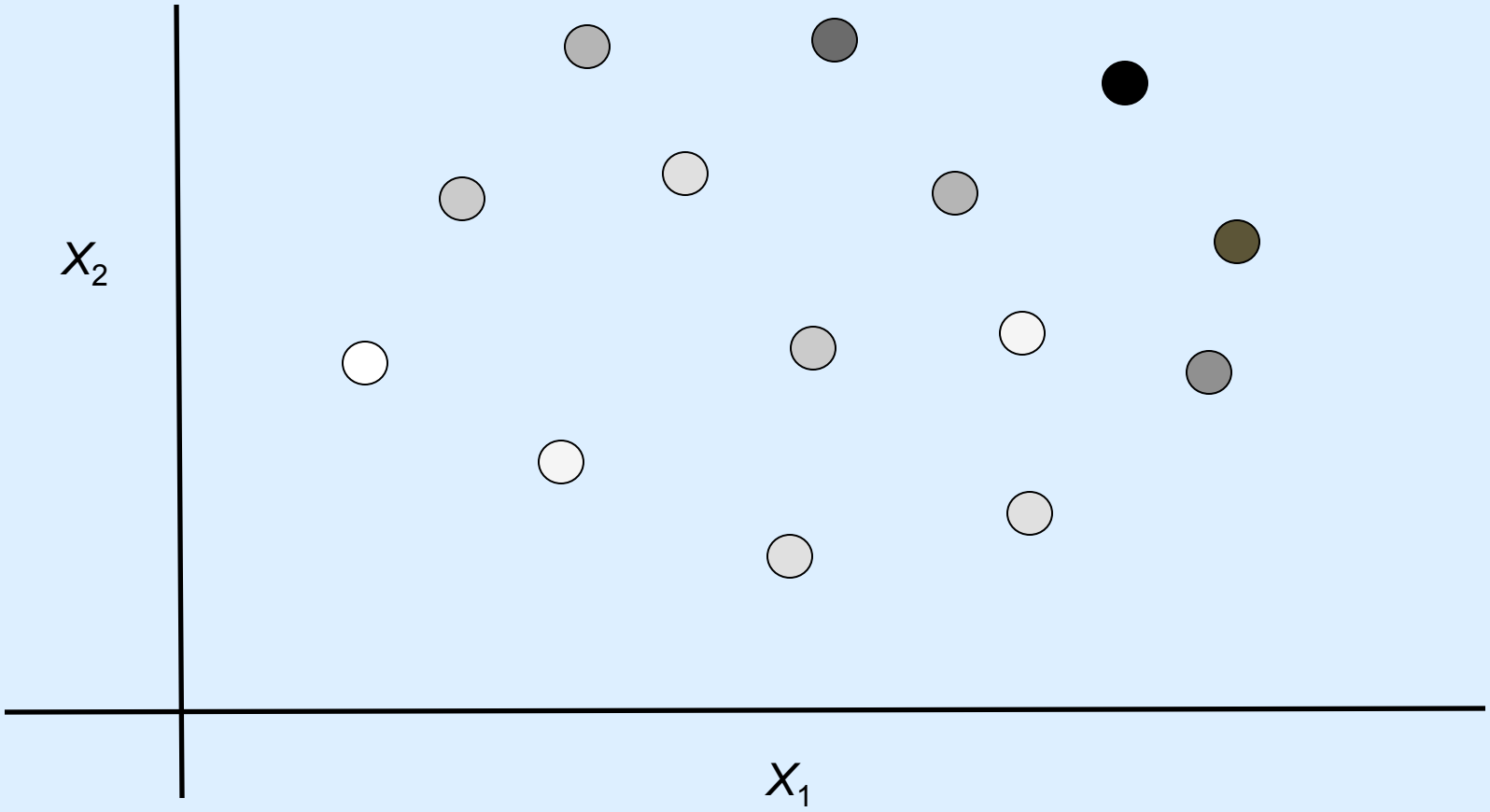
Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer


Module 7.1: The Training Process

# This Sub-Module Covers …

- What a typical training scenario looks like.
- Issues that arise during the training process.
  - Overtraining and efficiency
  - Lay the groundwork for describing procedures to handle multiple input/output data pairs and related issues
  - Motivate important performance measures covered in subsequent sub-modules

# A Classification Problem



# Can A Neural Network Model a Polynomial?

- From the manner of training a Neural Network, they can
  - Interpolate data given the training data.
  - Can they model a polynomial?
- How can we mathematically define a polynomial given specific data?

# Collocation Method of Polynomials

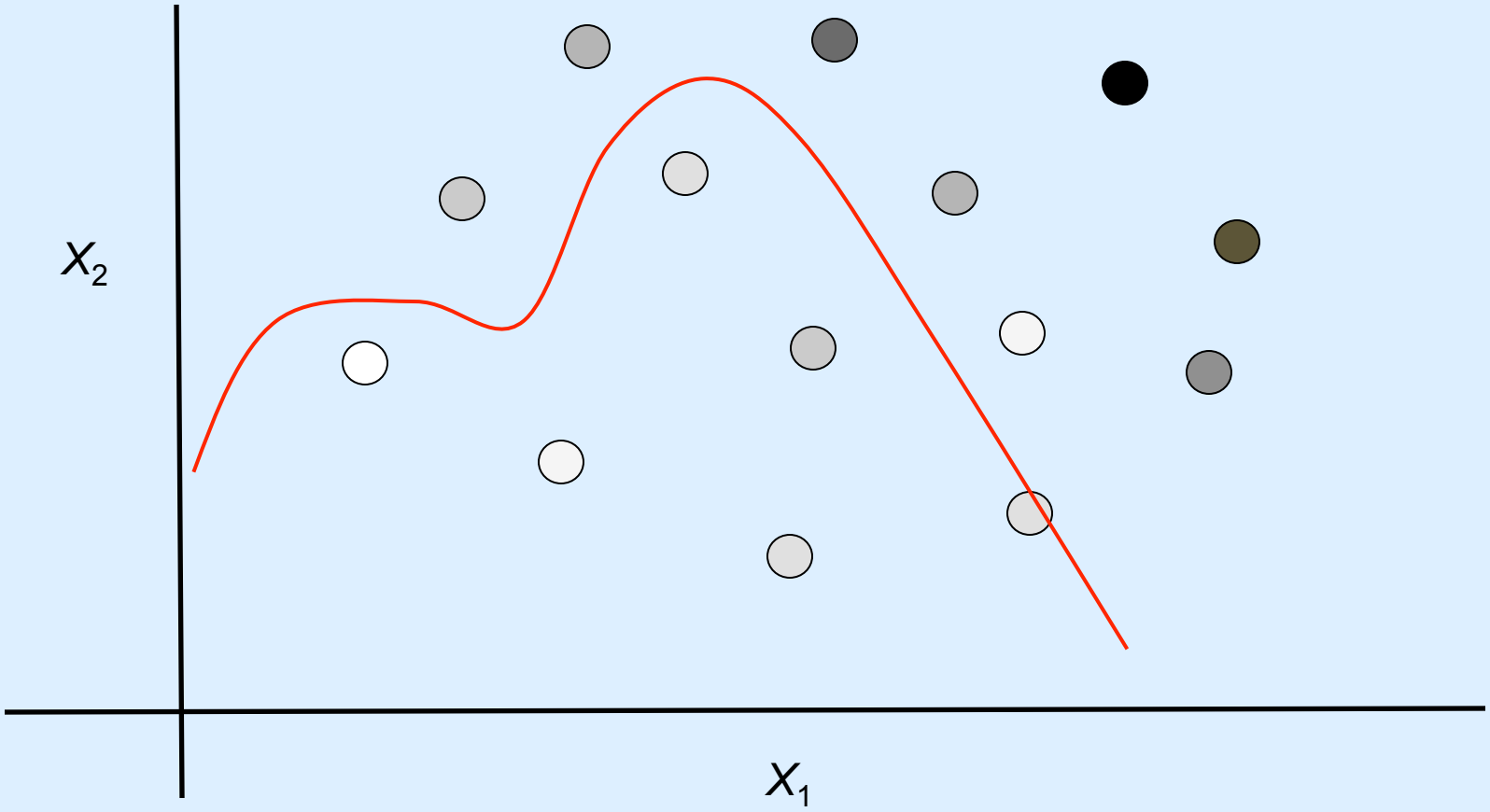
To define an *n*-degree polynomial function $P_n(x)$ that goes "through" a specified set of points $(x,y)$, define

$$P_n(x) = \sum_{i=0}^{n} f_i L_i(x)$$

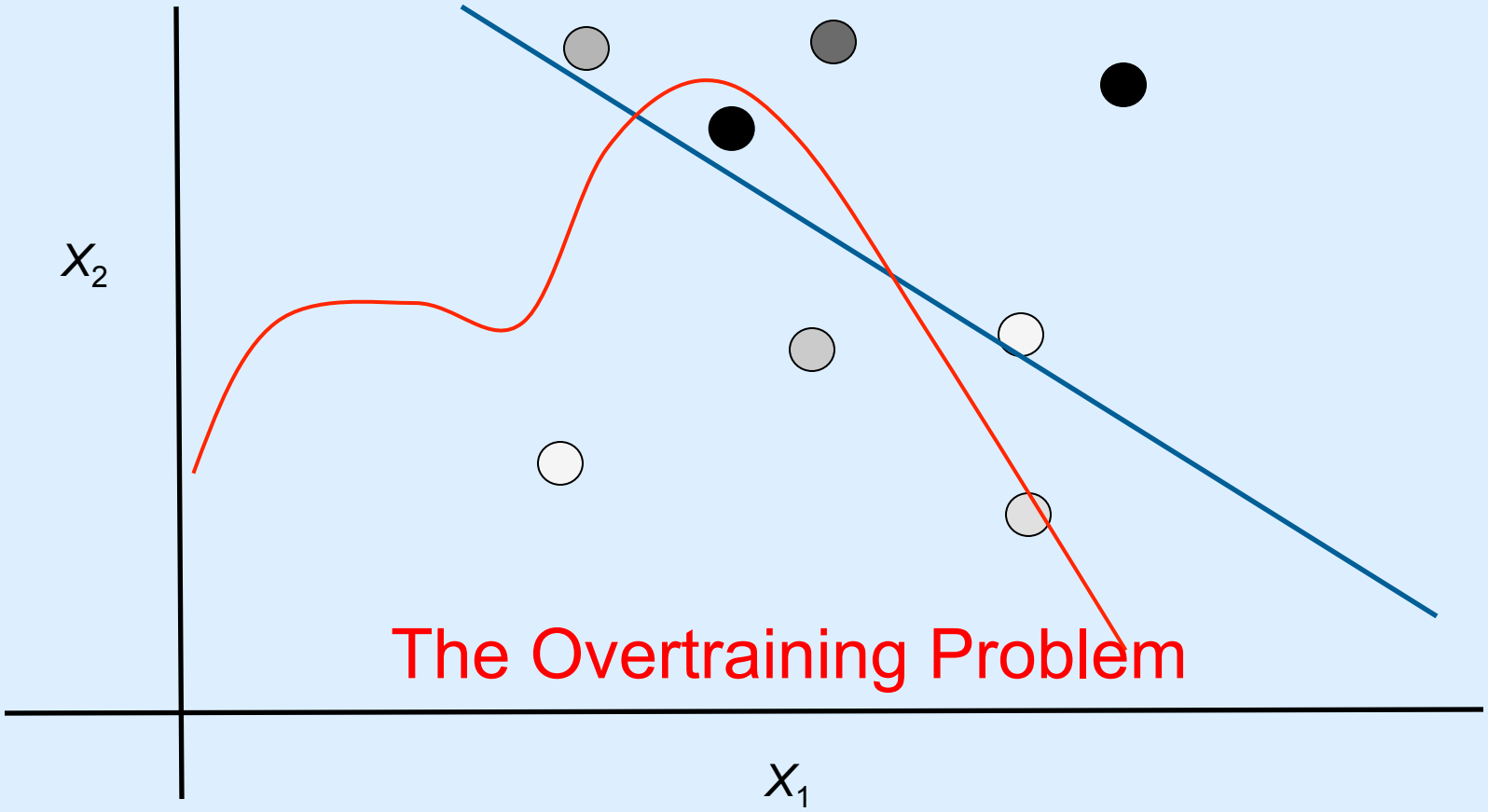
where

$$L_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{(x - x_k)}{(x_i - x_k)}$$

# A Classification Problem



# A Classification Problem



# Training: Supervised Learning

- Want to use a neural network to handle data in some future scenario.
- Need to train the network with data we already have to do this.
- But we also need to assess how well the network works.
- Obviously, we can't use the same data we trained the network with to assess its performance. Why?

# Training: Supervised Learning

- A common approach is to minimize errors in a FFBP setup.
- Obtain raw data:
  - parts corresponding to inputs,
  - parts corresponding to associated outputs.
- Divide the data into two parts: Training Data and Testing data.
- Take the training data and train and train and train, (but not too much) until errors are as small as possible.
- Then test the trained network using the testing data set to see how well it works using various performance criteria.
- Then make adjustments as necessary.

# Partitioning Data Sets and Statistical Analysis

- Many ways to partition data
  - Just once as in the following example, or
  - Many times using different 'partitions' and then average the results (the weights) from each partition.
- Many statistical methods do this type of partitioning. Popular methods include:
  - Bootstrap Method
  - Jacknife Method
- Basically idea is to use the same data but create many training/testing sets. Randomly select out the training set and the testing set. E.g. for 20 I/O pairs, using 10 I/O pairs for training, 10 I/O pairs for testing. But

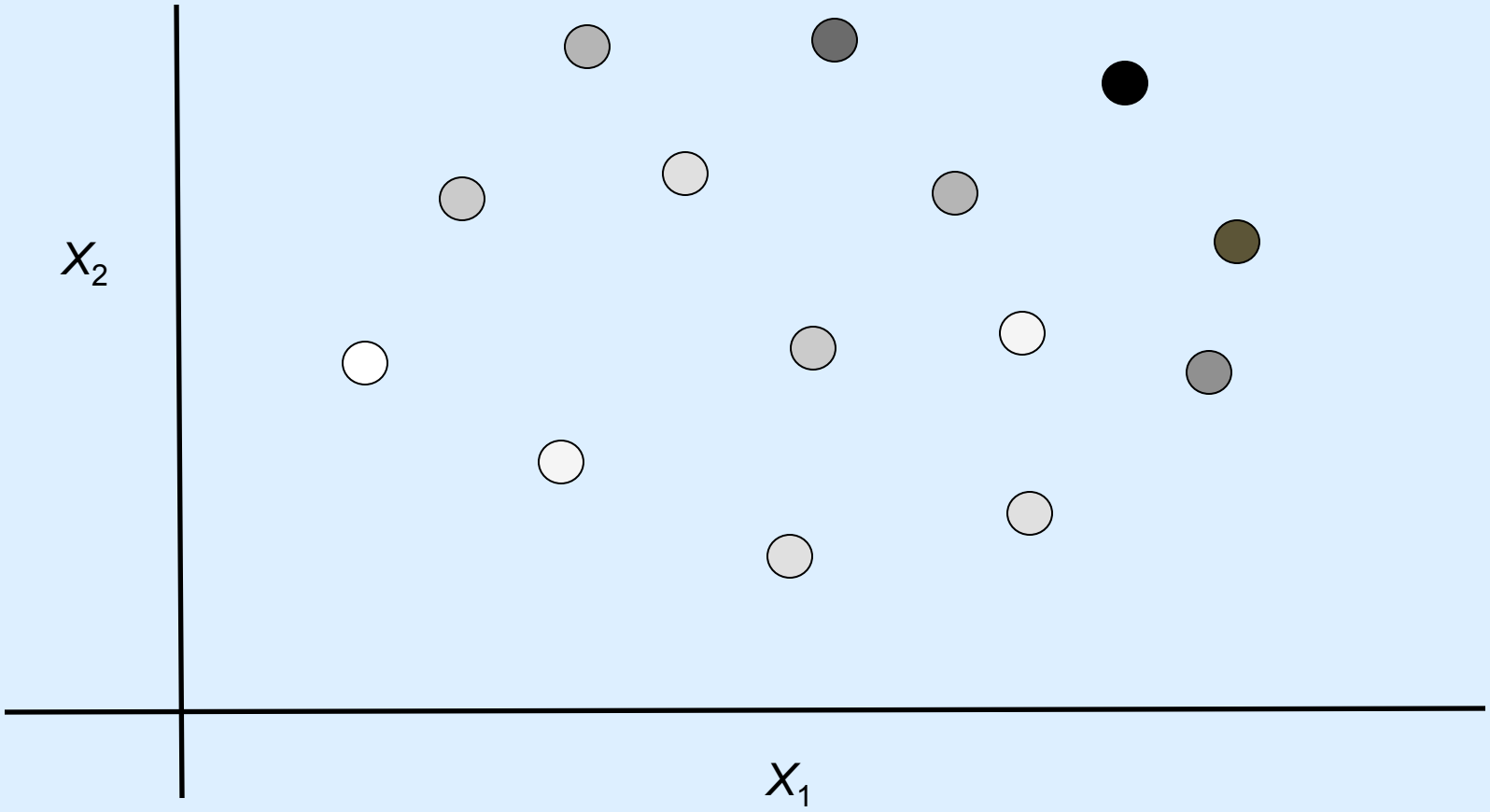
$$\binom{20}{10} = \frac{20!}{10!10!} = 184,756$$

# This Scenario Requires

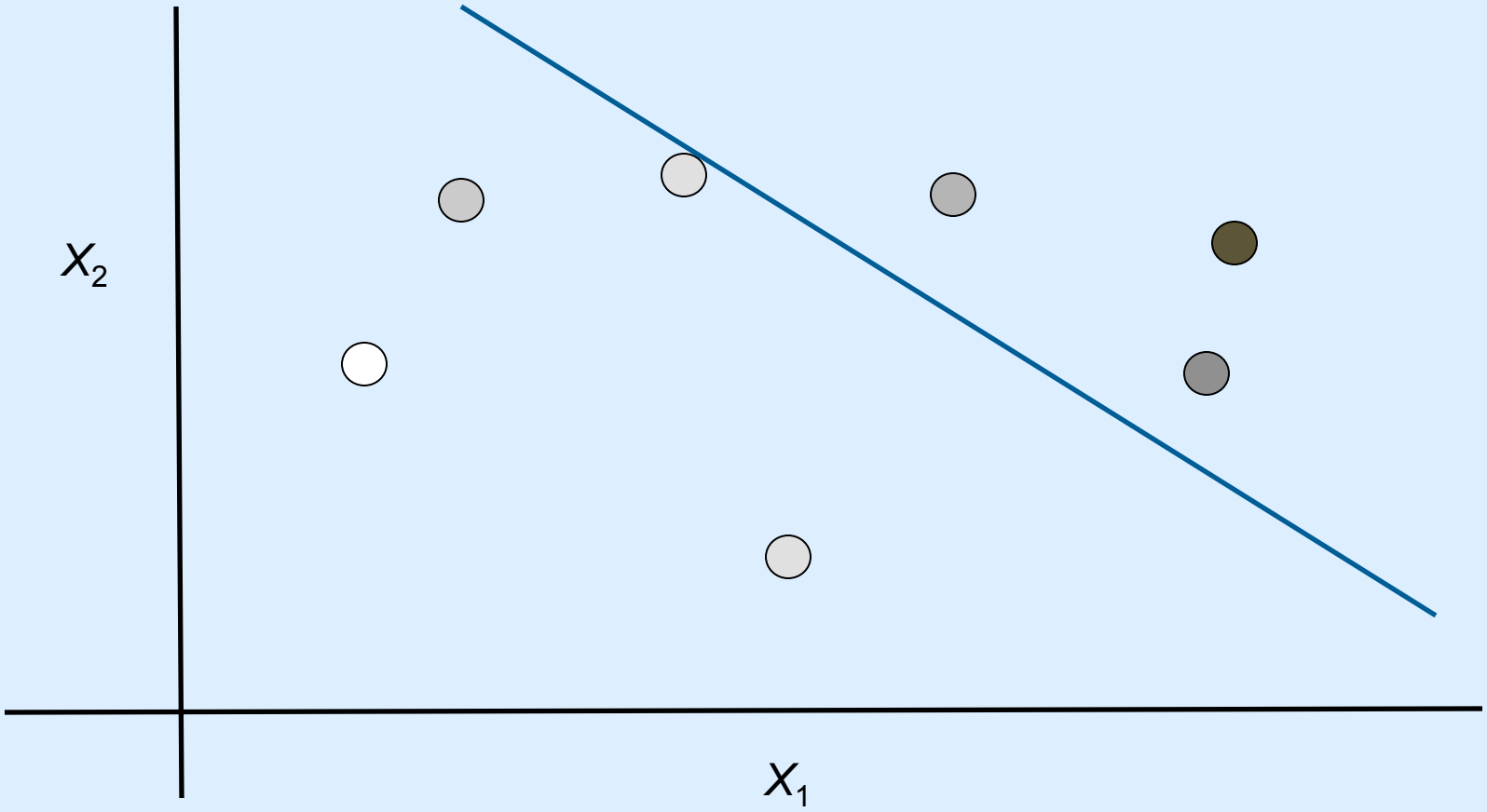
- A means for evaluating network performance (obviously).
  - Performance refers to how well the neural network performs the desired function for which it is being trained.
  - We will, for now, assume some method of measuring performance and get into the details in later sub-modules.
- Also means being able to use the neural network and associated data effectively and efficiently!
  - Computational efficiency is important for training AND assessing performance.

**Efficiency**

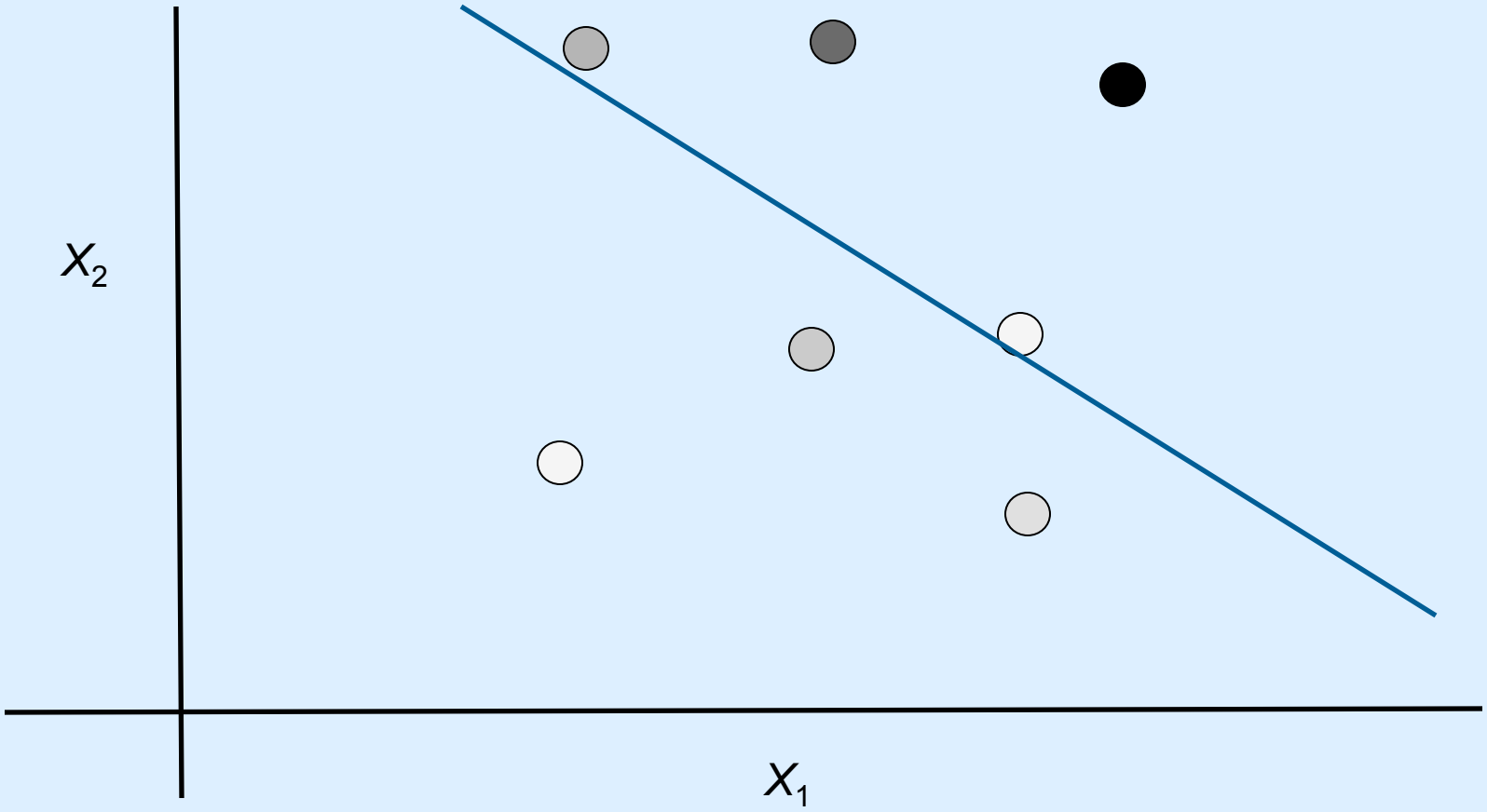
# A Classification Problem



# A Classification Problem

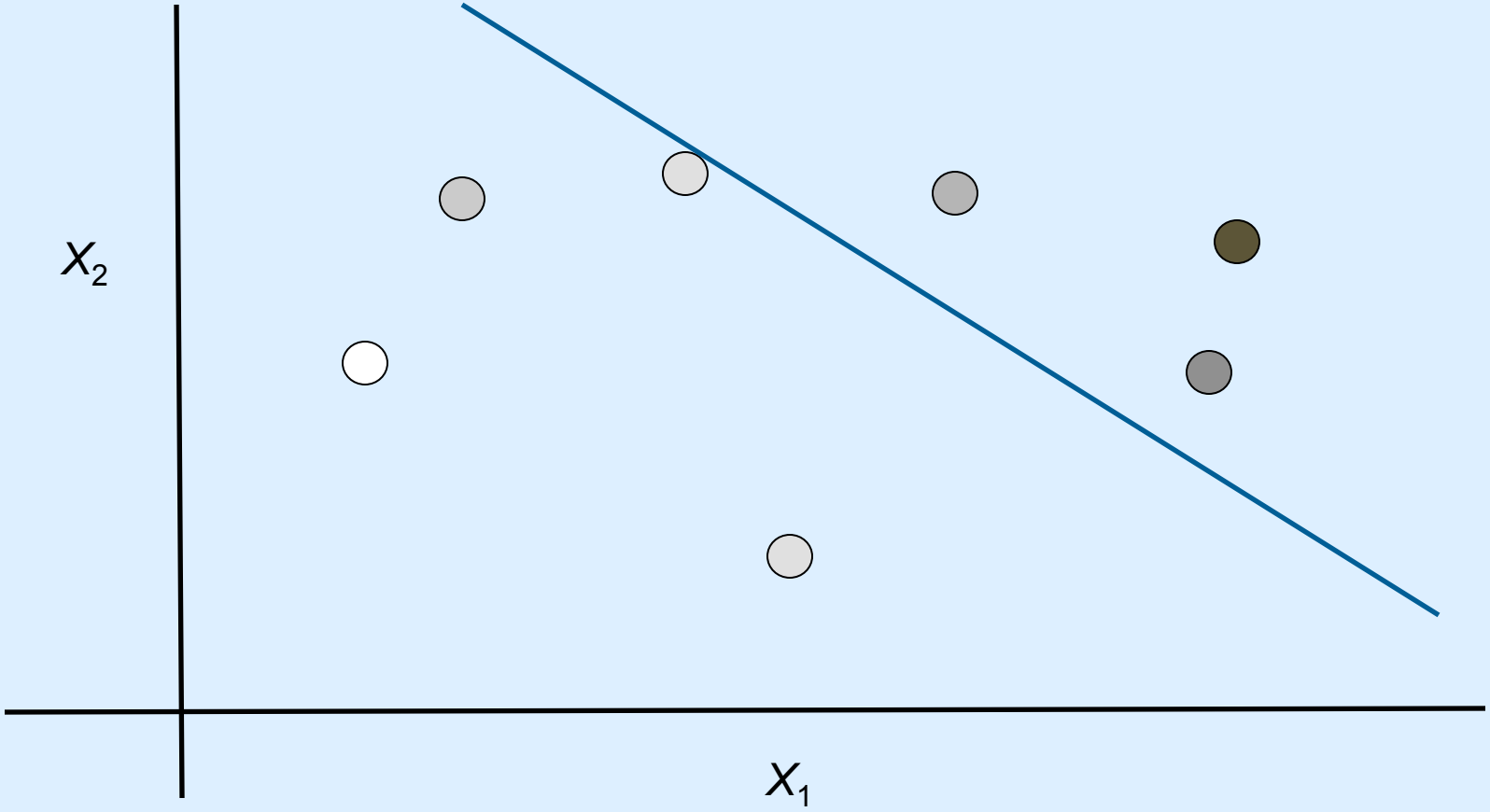


# A Classification Problem

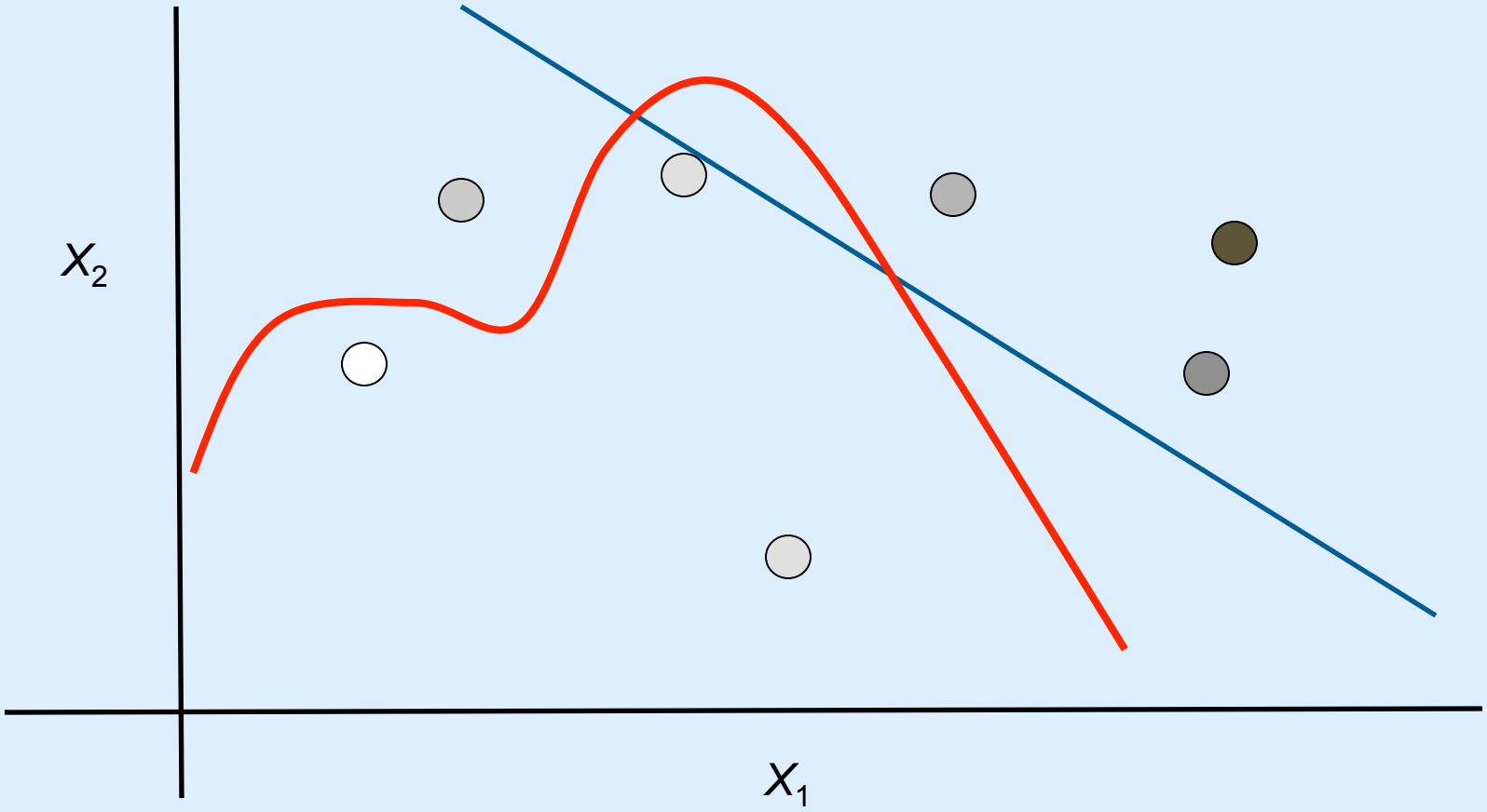


# A Classification Problem

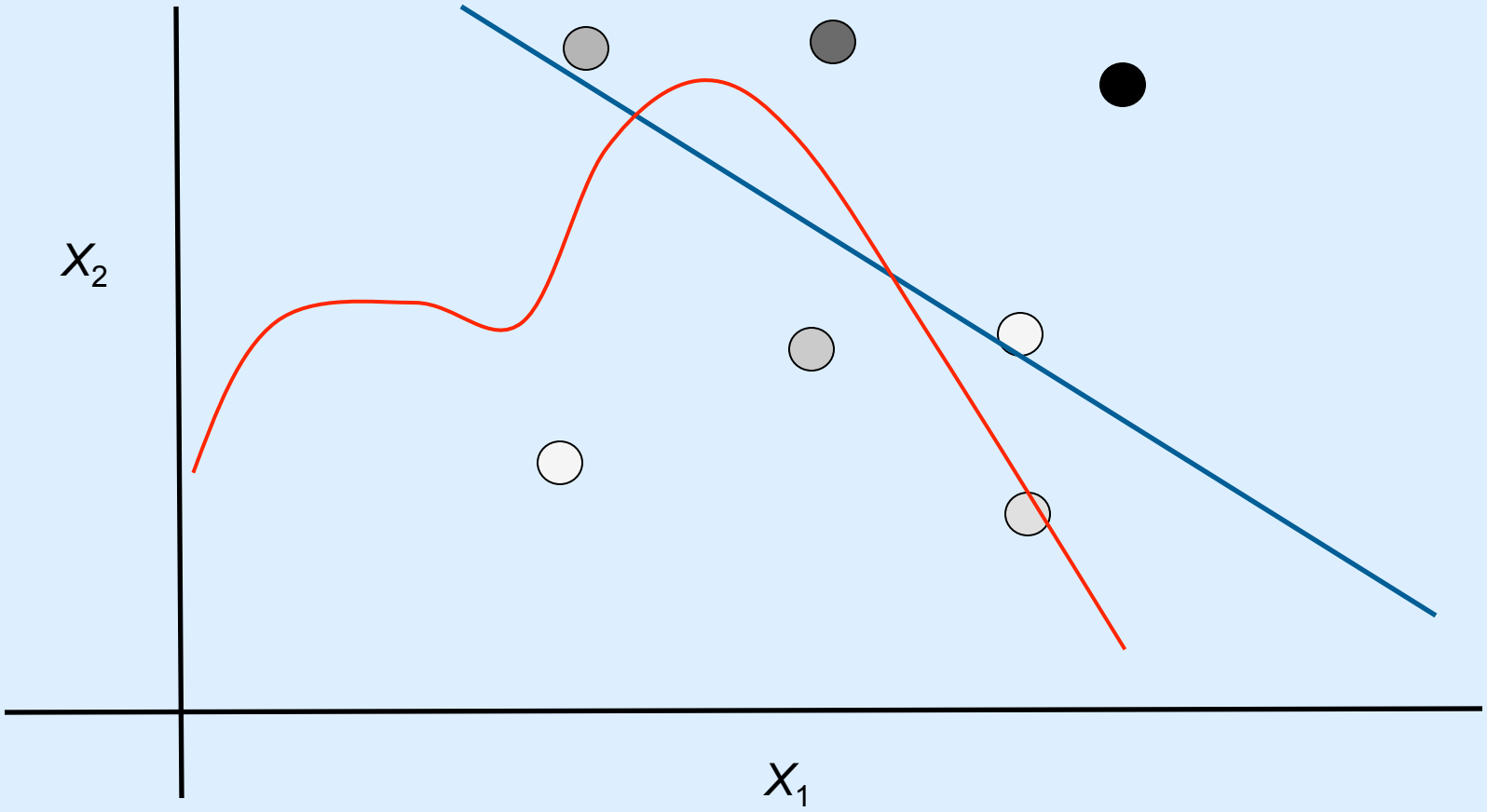
## Back to Training Set



# A Classification Problem



# The Overtraining Problem



# The Overtraining Problem

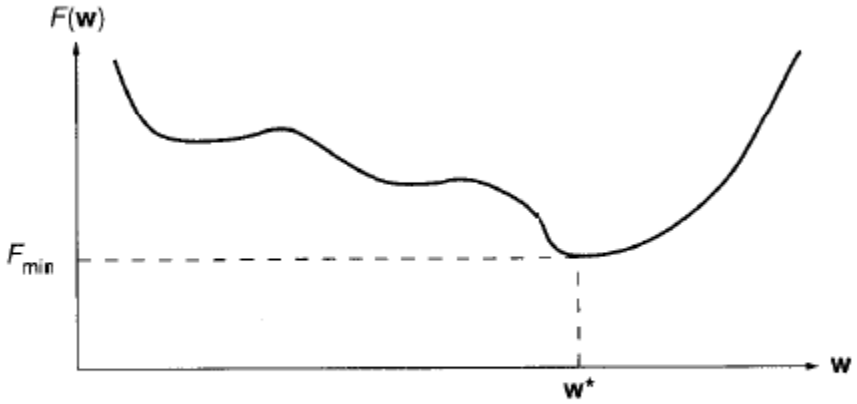



***Fig. 5.1.*** • A typical error surface. A goal of neural network training is to find a network weight vector $\mathbf{w}^*$ that minimize the error $F$.

From *Neurocomputing*
by Hecht-Nielsen

# The Overtraining Problem

- Random variations may be part of some underlying process.
- A Regression Line estimates the deterministic part of the process.
- Training attempts to capture this process.
- Training too much may end up including the effects of random variation in the deterministic part of the modeling of a process.

We end up training the network on the randomness instead of the underlying deterministic process!

Training on the errors.

# Training Reprise

- During training, we use the error to compute the gradient in steepest descent methods, to guide us towards minimizing the error.
- During testing/evaluation, we use the error to minimize the testing error.
- Must do a lot of experimentation: training, testing, training, testing…
- A lot of computational effort is involved to get it right!

# Training Efficiency

- How can we maximize use of our data?
- How do we decide how many times to train and test? How many different partitions to use?
- Efficient and effective training and testing methods must be employed.