Key concepts for STAT 154 / 254

Ryan Giordano

Key concepts

In this lecture, I'll introduce and motivate some of the key concepts for supervised learning:

- Estimation procedures: using data to make predictions
- Function classes and their expressivity
- Loss, risk, and randomness

By the end of the class, you should be able to:

- Recognize a prediction problem and name a few estimation procedures
- Identify a procedure's function class and reason about its expressiveness
- Express the prediction problem in terms of loss, risk, and randomness — and criticise this expression

QSAR Fish Toxicity Dataset

These examples will be based on the UCI QSAR Fish Toxicity Dataset.
The dataset consists of a set of 908 chemicals (one row each), with data for:

- `LC50`: the concentration that causes death in 50% of test fish over a test duration of 96 hours.
- `MLOGP`: (molecular properties)
- `CIC0`: (information indices)
- `GATS1i`: (2D autocorrelations)
- `NdssC`: (atom-type counts)
- `NdsCH`: (atom-type counts)
- `SM1_Dz`: (2D matrix-based descriptors)

Figure 1: Sad fish

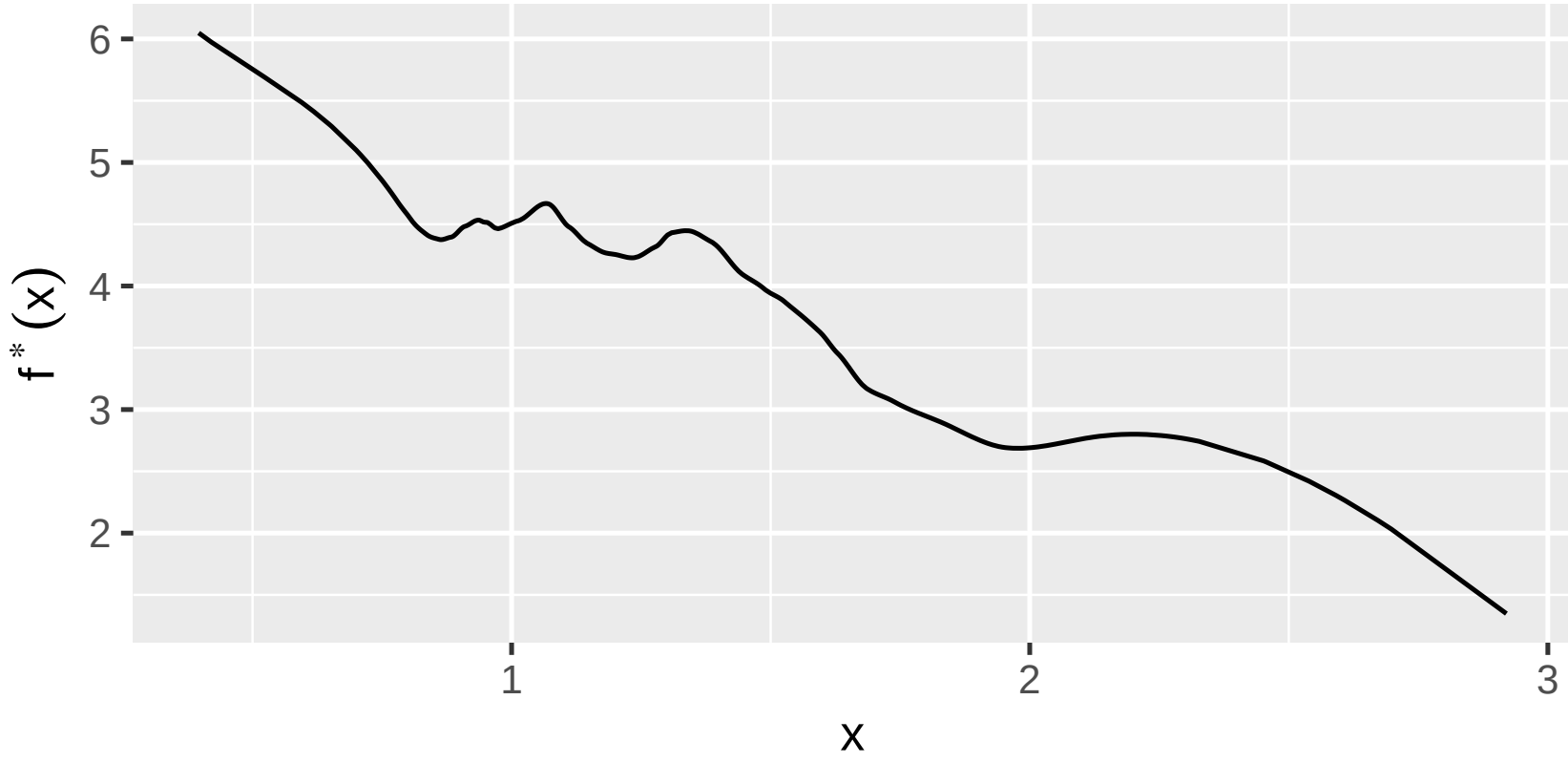
"Quantitative structure–activity relationship" (QSAR) models predict some chemical property (toxicity to the fathead minnow) from structural features of the molecules.

Task: Set $x =$`GATS1i`, $y =$`LC50`, and use ML to predict fish mortality from 2D autocorrelations.

> **Question**
>
> Describe some concrete settings when you would want to use an ML model that could perform this task.

Note: I will be simulating data using a "ground truth" that is loosely based on patterns from the actual data.

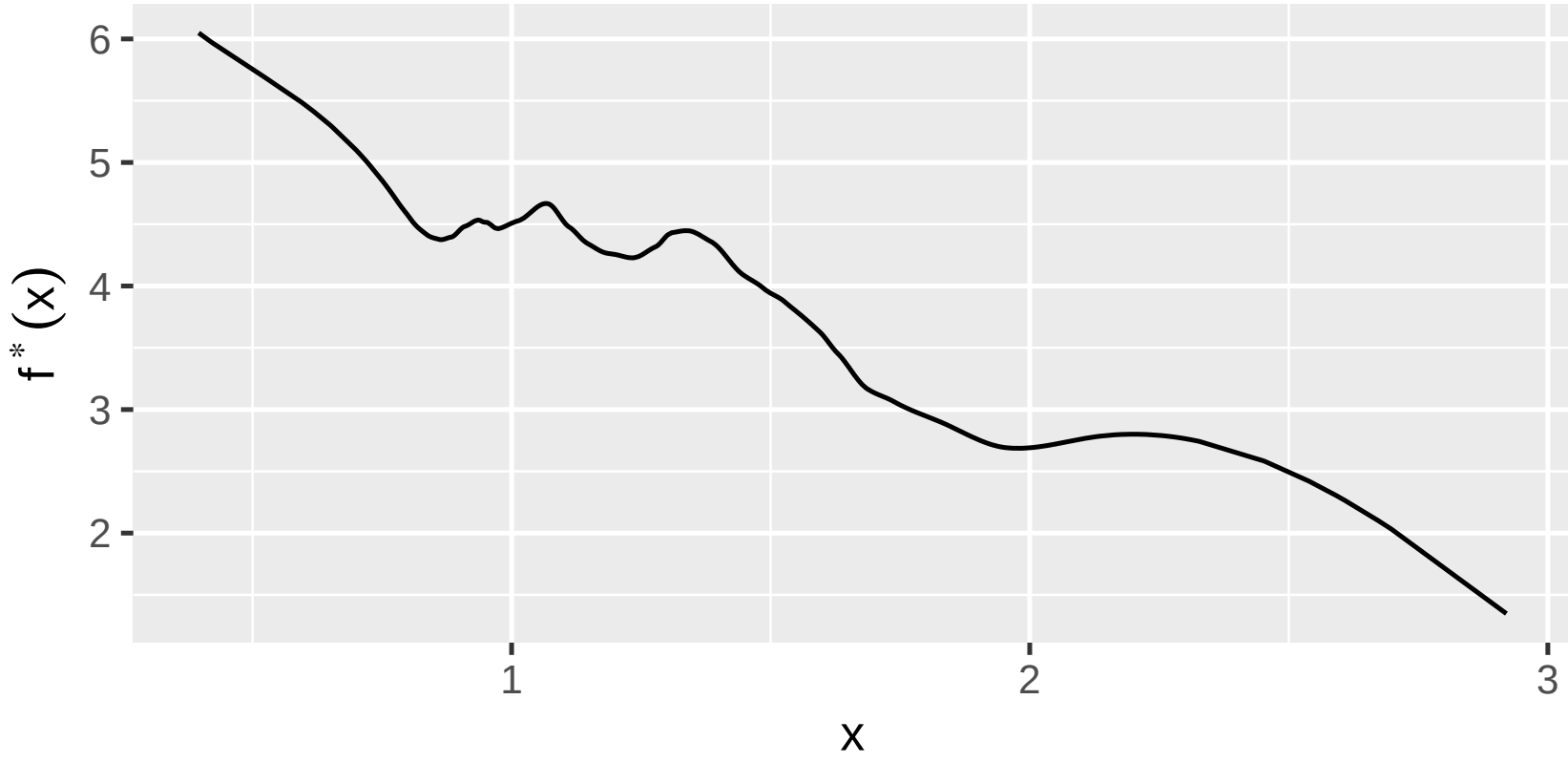
Fitting a function



Let's imagine that y (fish toxicity) is determined exactly by x (2D autocorrelations).

That means $y = f^\star(x)$ for some $f^\star$. (**I will drop this assumption later in the lecture.**)

Fitting a function

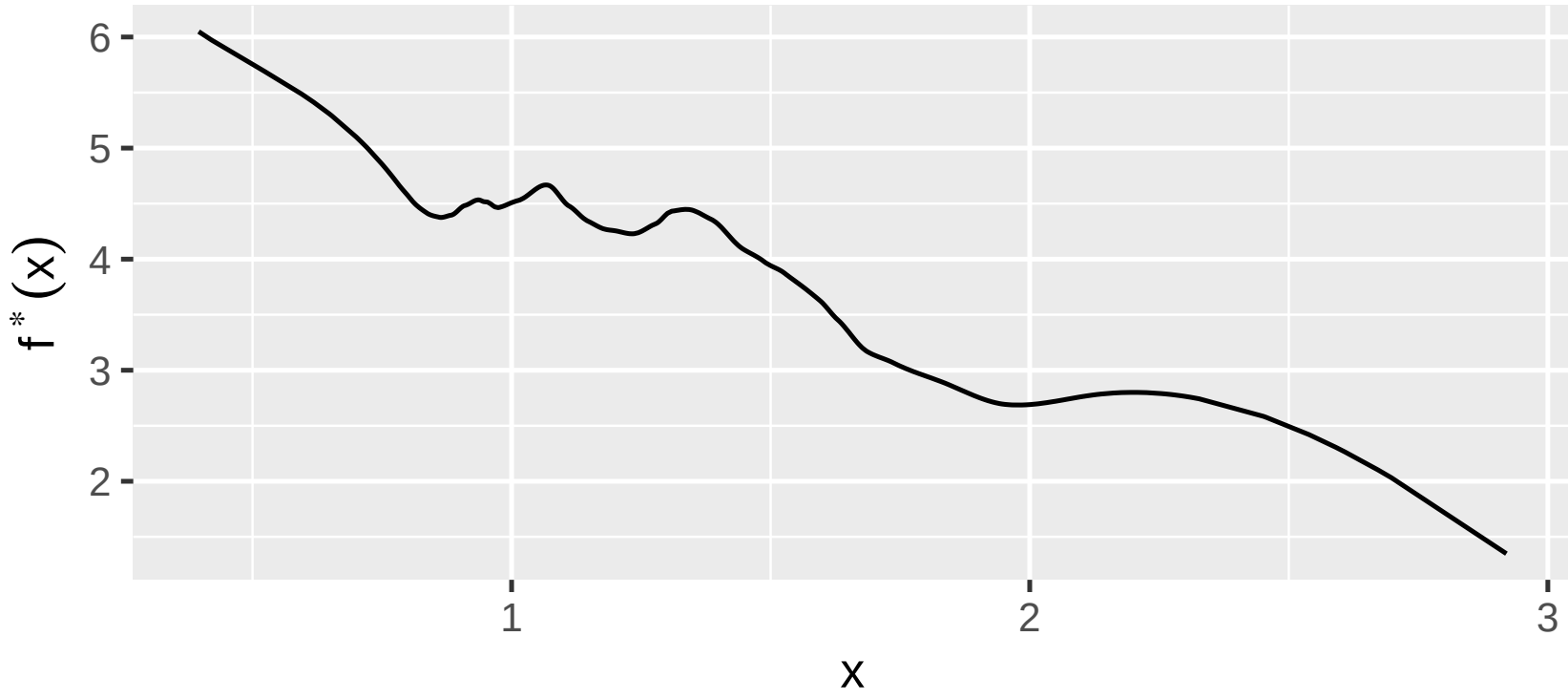


Let's imagine that y (fish toxicity) is determined exactly by x (2D autocorrelations).

That means $y = f^\star(x)$ for some $f^\star$. (**I will drop this assumption later in the lecture.**)

We don't know $f^\star$, but we can *query* it by selecting x (a molecule) and measuring y (seeing what concentration kills minnows)

Fitting a function



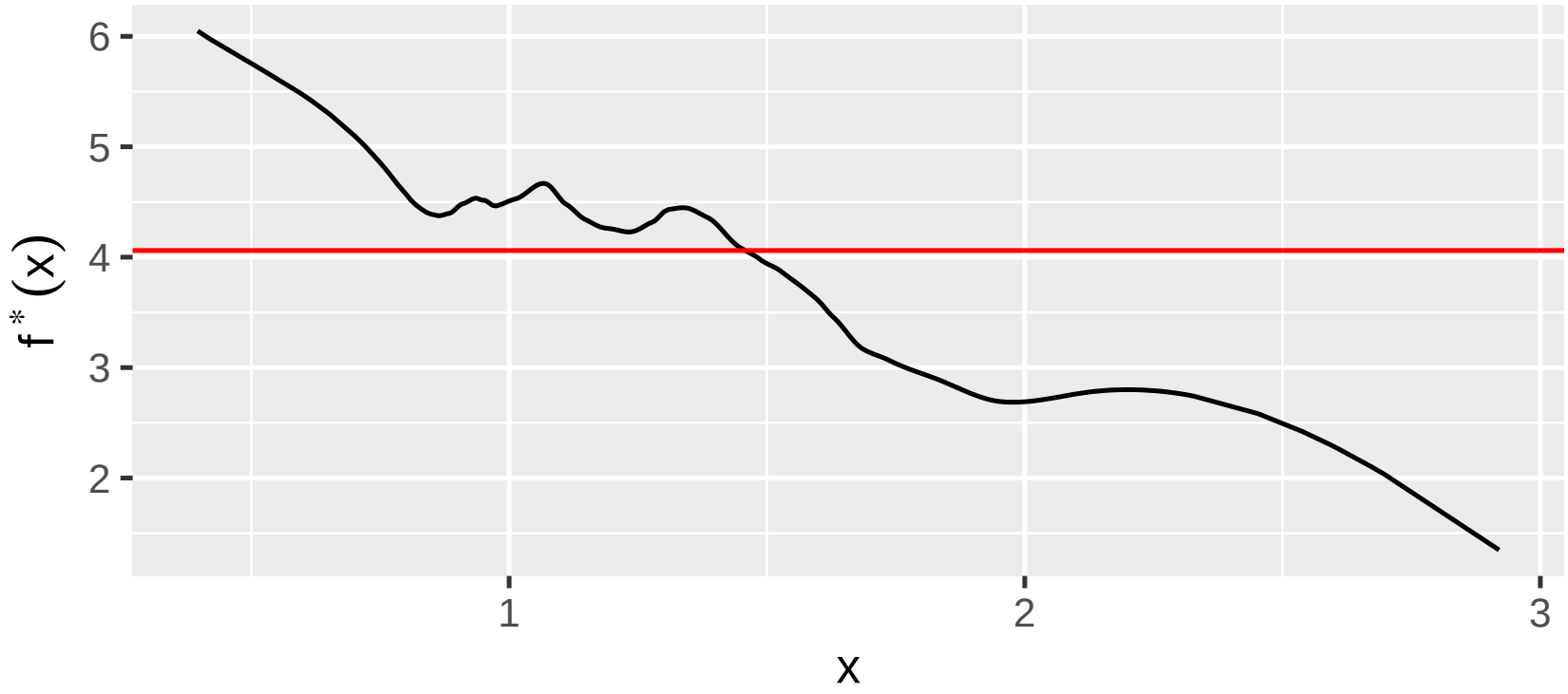
Task: Query (x_n, y_n) pairs and make a *program* that returns $\hat{f}(x) \approx f^\star(x)$.

Question

Suppose you can query a very large number of datapoints.

- Can you ever perfectly approximate $f^\star(x_{\text{new}})$ at an unseen x_{new}?
- Propose the simplest procedure you can think of (even if not very good).
- Propose a more complex procedure that will do a better job.

Constant functions

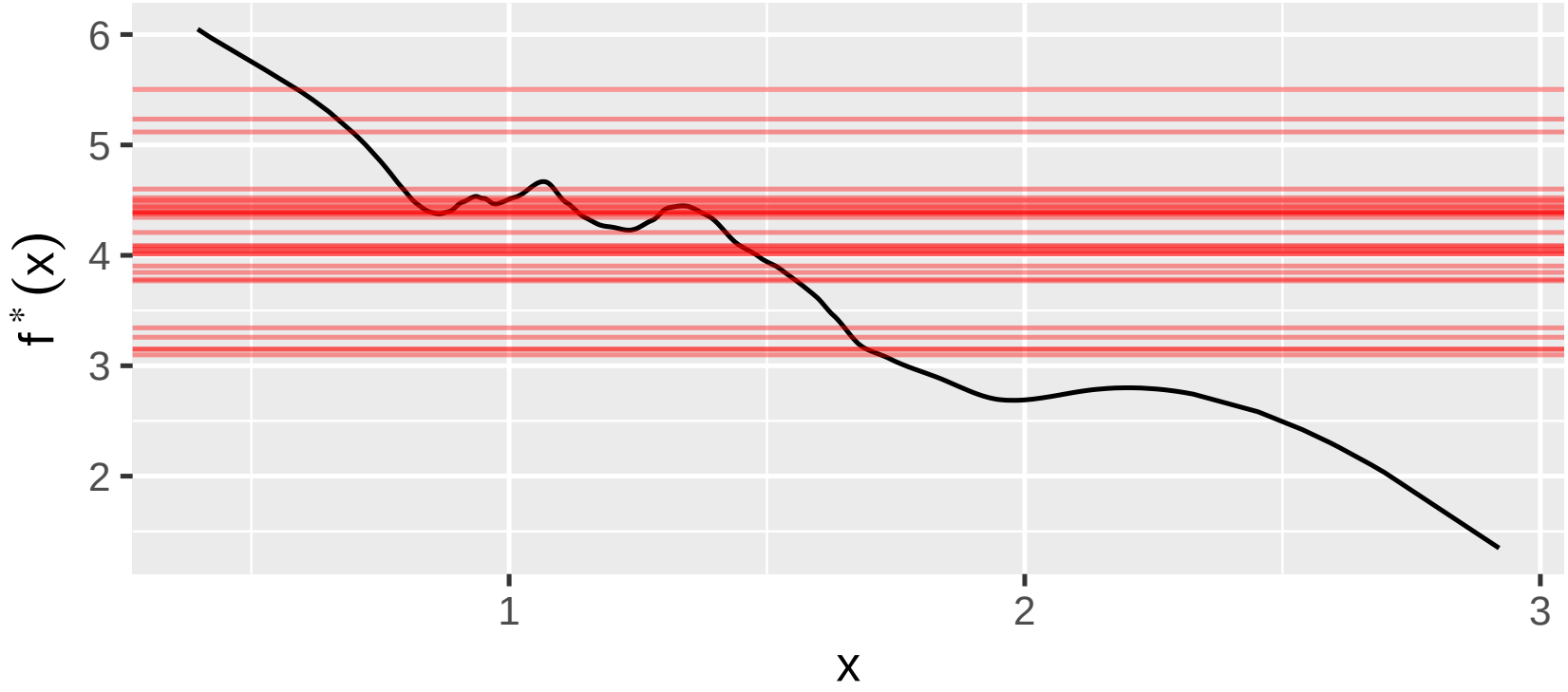


The simplest approximation is arguably the constant $\hat{f}(x) = \frac{1}{N} \sum_{n=1}^{N} f^{\star}(x_n)$.

Question

- Think of an application for which a good constant is enough.
- Think of an application for which a good constant is not enough.
- Will the constant you estimate depend on which x_n you select? How?
- Can you think of other ways to take the data and choose the constant?

Constant functions



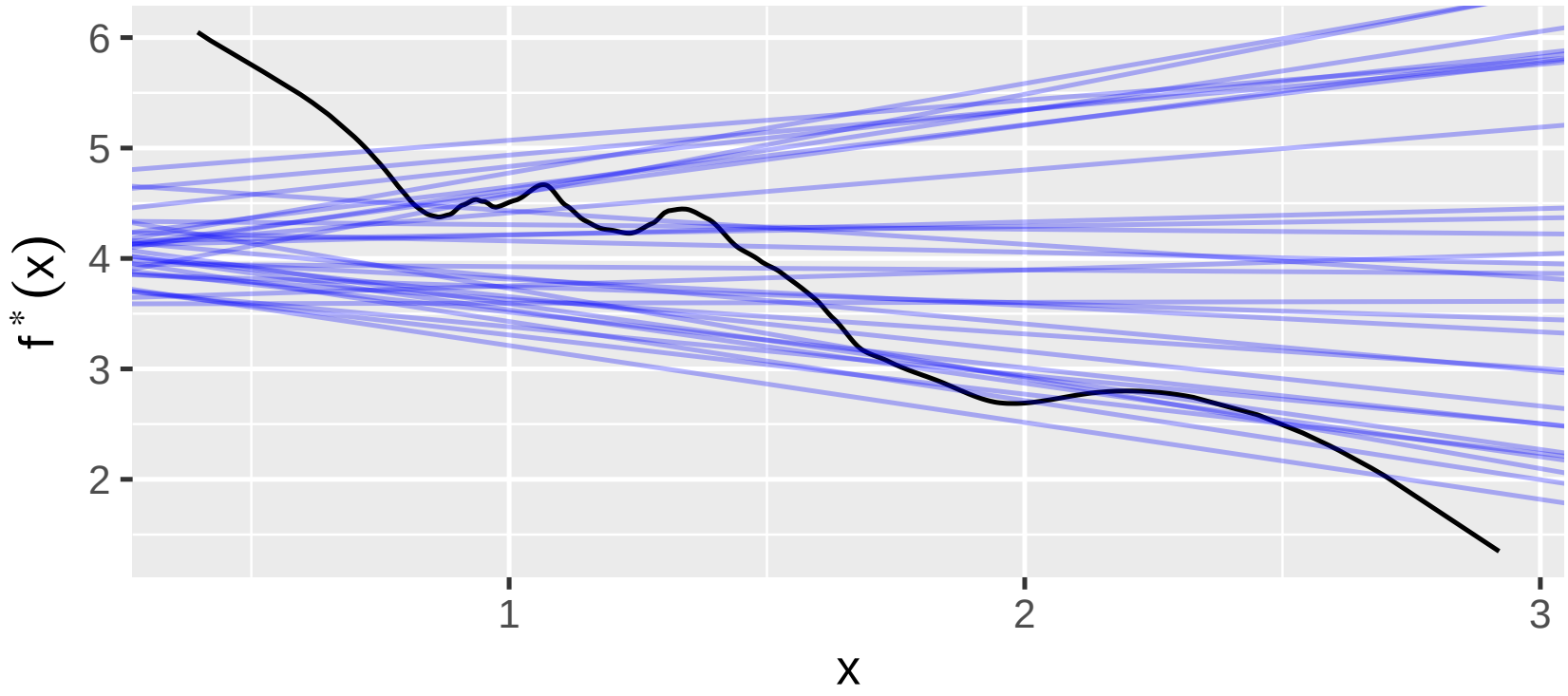
Using a constant function selects $\hat{f}$ from the *function class*:

$$\mathcal{F}_{constant} := \{f : f(x) = c \text{ for some real-valued } c\}$$

Our "estimation procedure" takes in data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and returns a function $\hat{f} \in \mathcal{F}_{constant}$. This is equivalent to choosing a $\hat{c}$ and taking $\hat{f}(x) = \hat{c}$.

For a procedure in general, $\mathcal{F}$ is the set of *all functions that it could represent*.

Linear functions



We can also define the class of linear functions:

$$\mathcal{F}_{constant} := \{f : f(x) = c \text{ for some real-valued } c\}$$
$$\mathcal{F}_{linear} := \{f : f(x) = c + bx \text{ for some real-valued } c, b\}$$

Every constant function is also a linear function, but some linear functions are not constant functions.

We say that $\mathcal{F}_{linear}$ is *a more expressive function class* than $\mathcal{F}_{constant}$.

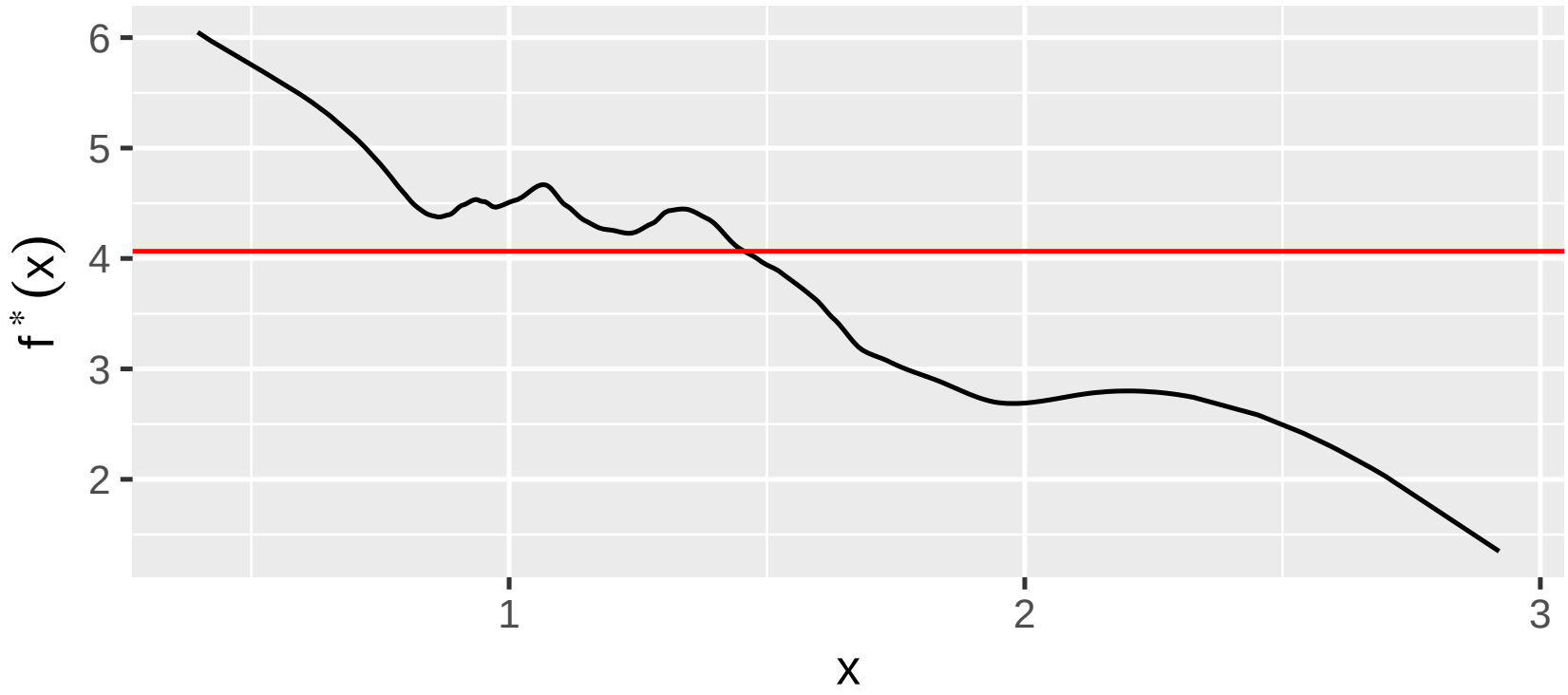
Function classes

A key theme of this class is developing expressive function classes!

i Question

Identify the function classes for some of the examples given earlier. Which are more expressive? Which are the least expressive?

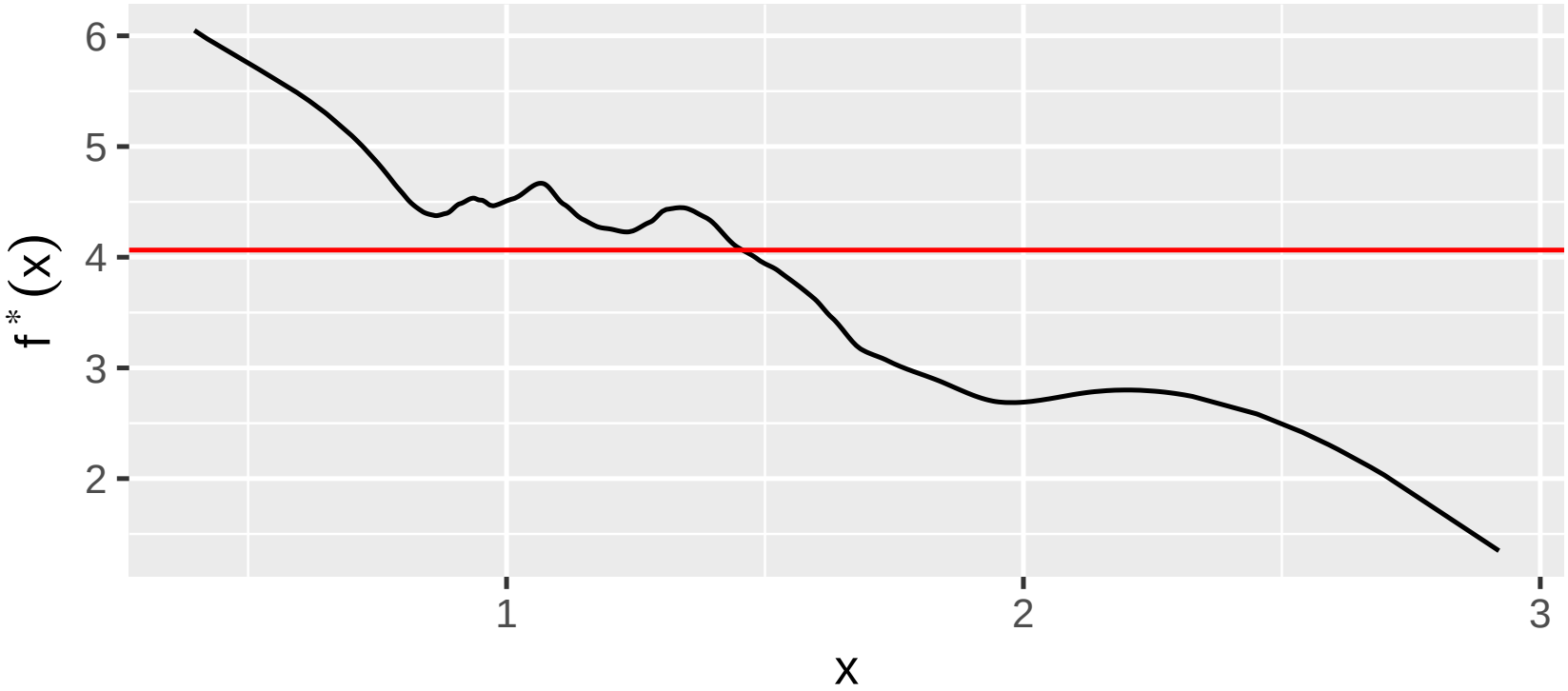
Loss



We want to choose $\hat{f} \in \mathcal{F}$ that is a "good approximation."

Suppose we have some x with associated y. Let the "loss function" $\mathcal{L}(f(x), y)$ measure how "bad it is" to predict $f(x)$ when the truth was y.

Loss



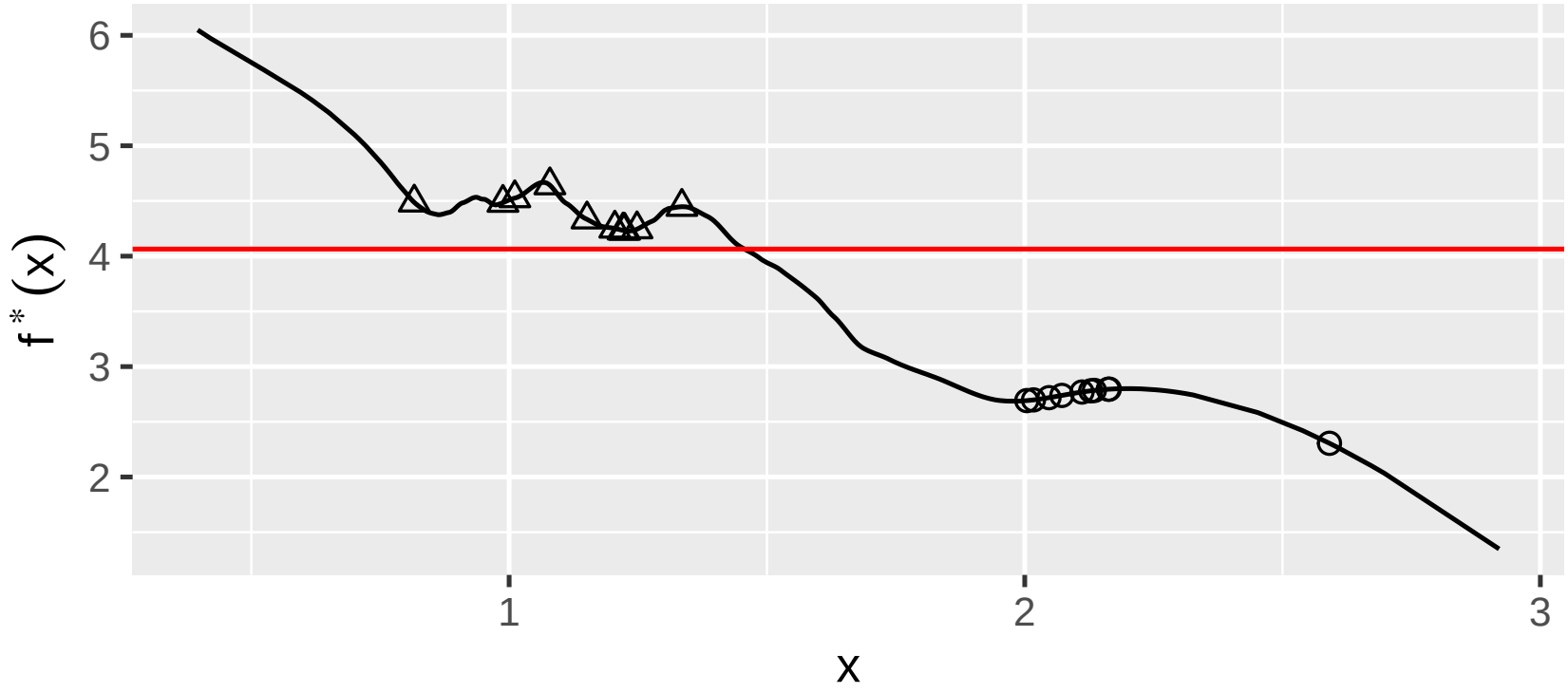
Common examples:

- $\mathcal{L}(f(x), y) = (f(x) - y)^2$
- $\mathcal{L}(f(x), y) = |f(x) - y|$
- $\mathcal{L}(f(x), y) = \mathrm{I}\left(f(x) \neq y\right)$ (Here, $\mathrm{I}\left(\cdot\right)$ is the "indicator function." It evaluates to 1 if its argument is true, and 0 otherwise.)

Question

What are some properties any loss function should satisfy?

Risk

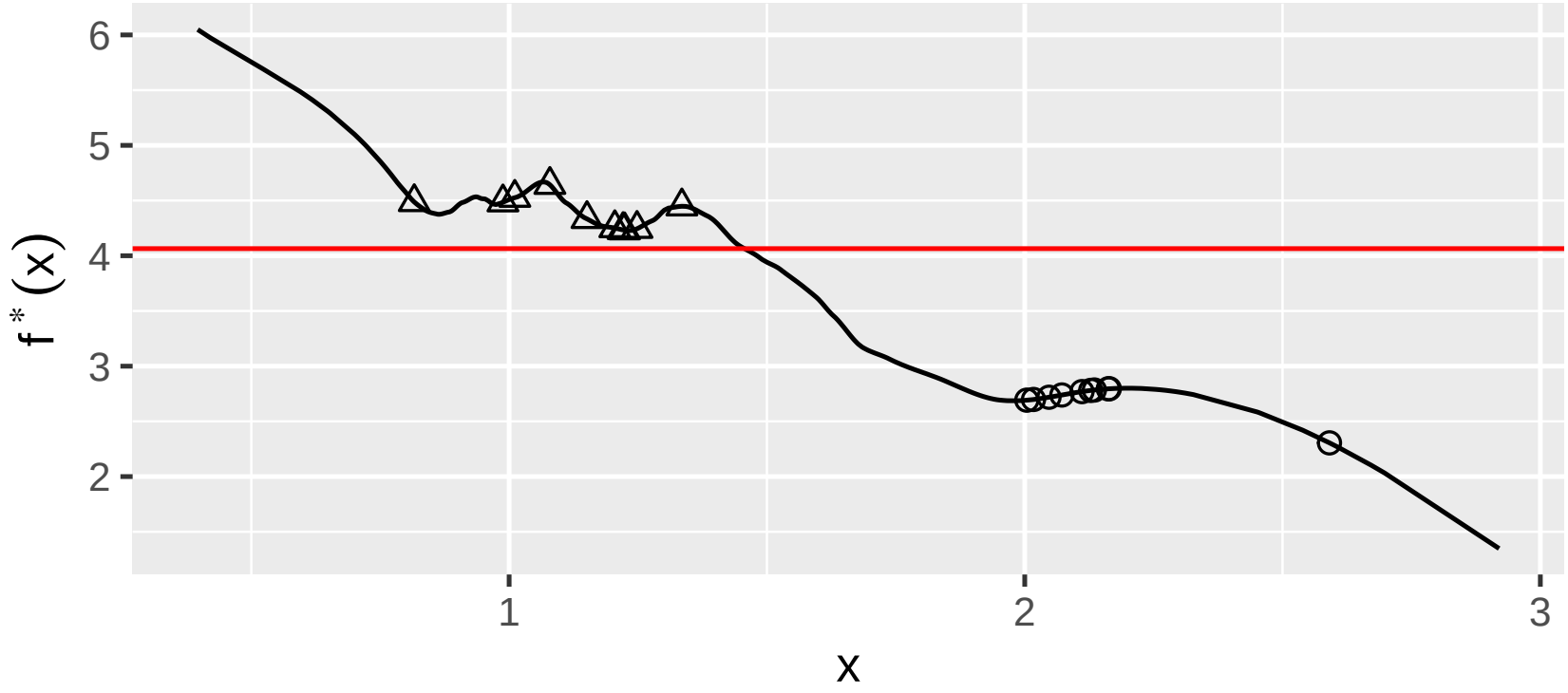


The loss is defined at a single datapoint. Unless we really care about only one prediction, it makes sense to *average* the loss over multiple datapoints:

$$\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(f(x_n), y_n)$$

Note that, for a given f, the average loss will be different depending where the points are!

Risk



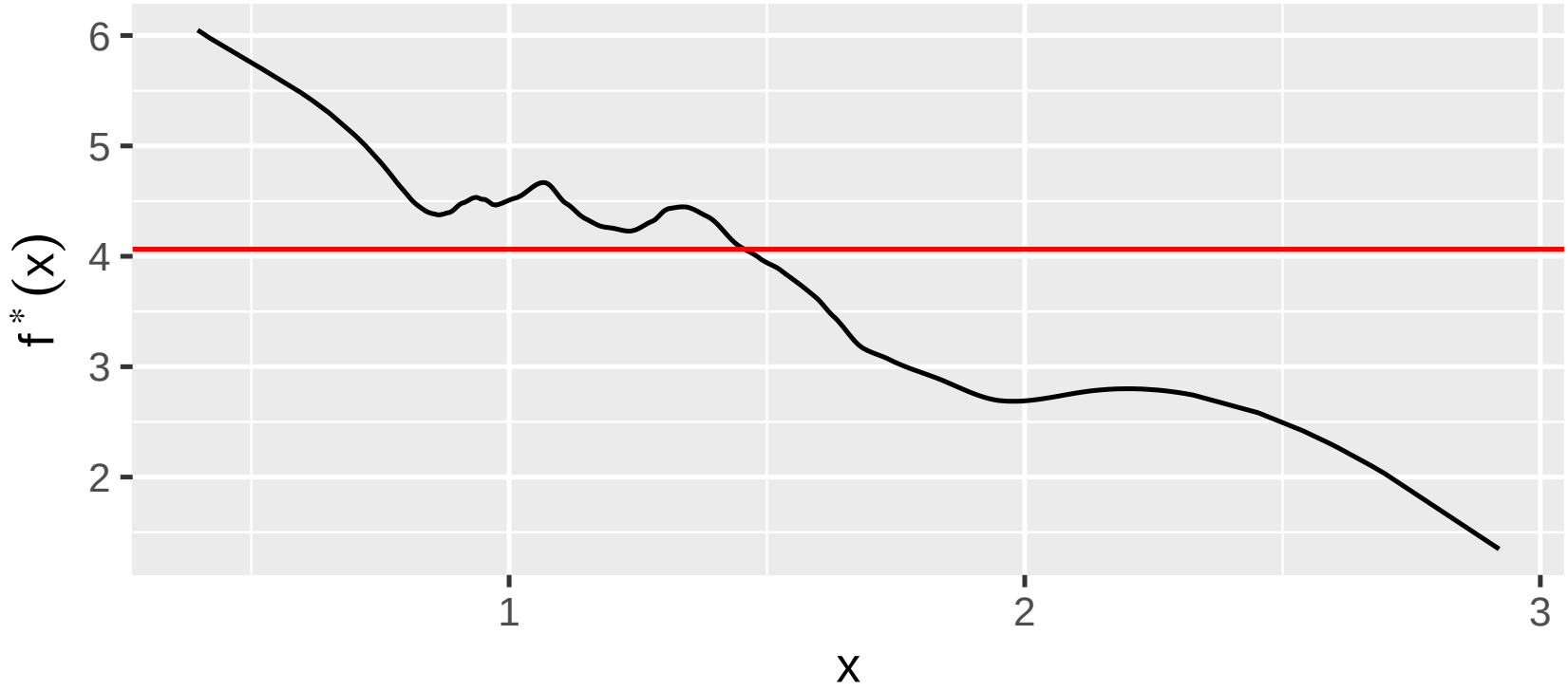
If we believe that future x will be drawn randomly, independently, and identically distributed (IID) from a future distribution $p(x)$, then it makes sense to measure the *average* loss.

This average loss is called "risk".

$$\mathcal{R}(f) := \mathbb{E}_{p(x,y)}\left[\mathcal{L}(f(x), y)\right] \quad \text{"Risk"}$$

Note that risk depends only on the function f — the distribution or data is considered fixed. **Different distributions give different risks.**

Risk

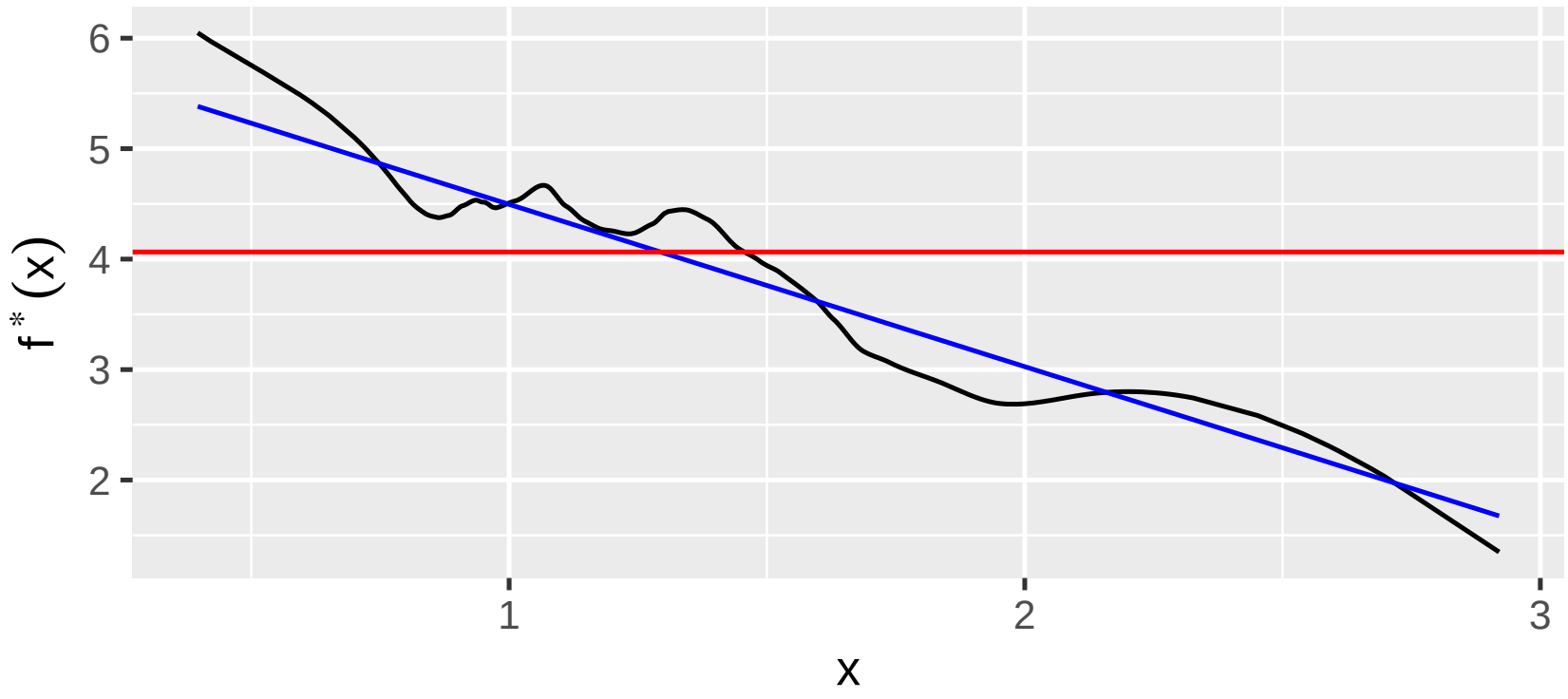


Question

Recall our QSAR application.

- Describe a use case where the IID assumption might be reasonable
- Describe a use case where the IID assumption might not be reasonable
- What does the IID assumption imply about the relationship between the other chemical variables?
- Does minimizing $\mathcal{R}(f)$ guarantee an accurate prediction on any particular x?

Risk



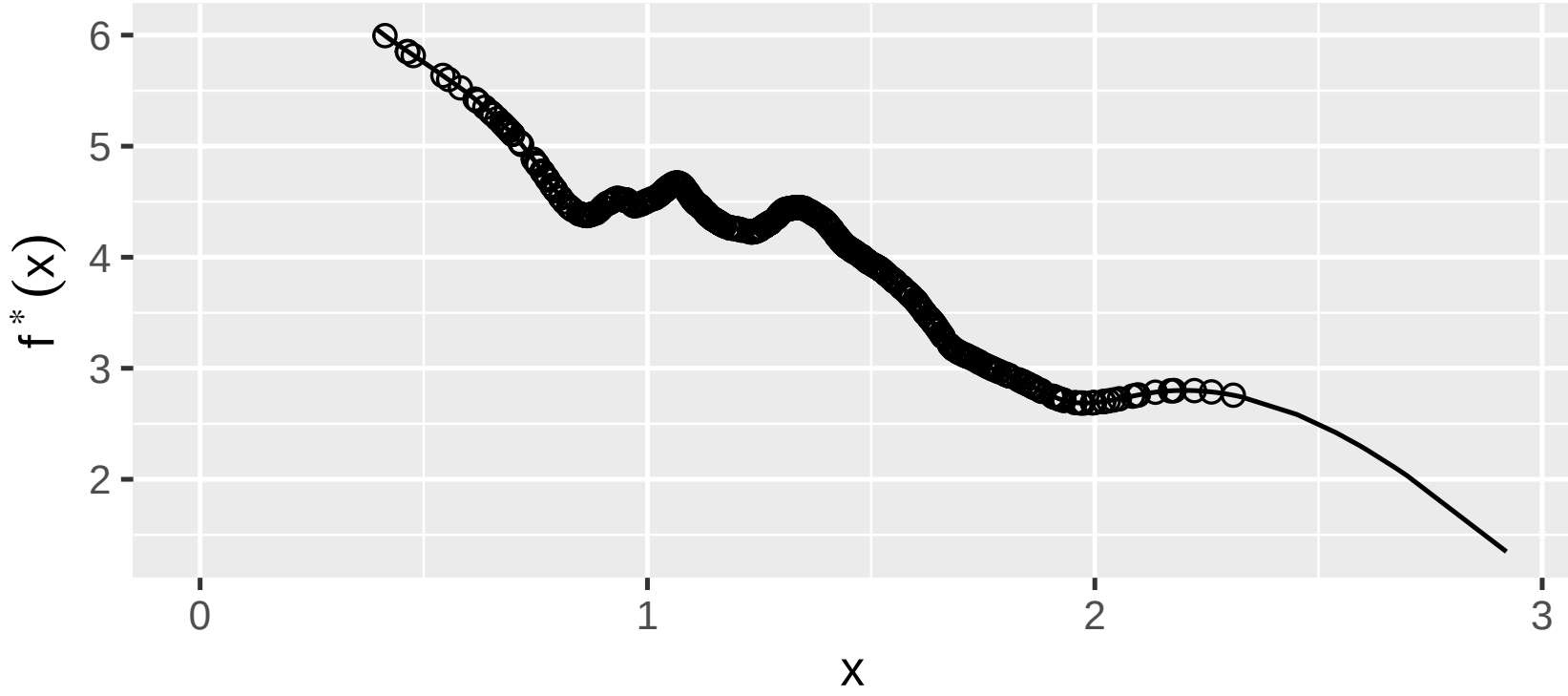
There's usually a gap between the best function $f^\star$ and the best function in our class:

$$\overline{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \mathcal{R}(f) \qquad f^\star := \operatorname*{argmin}_{f} \mathcal{R}(f) \qquad \mathcal{R}(\overline{f}) - \mathcal{R}(f^\star) \geq 0$$

This gap is typically *lower* for *more expressive function classes*.

Why not just make our function classes as expressive as possible?

Risk



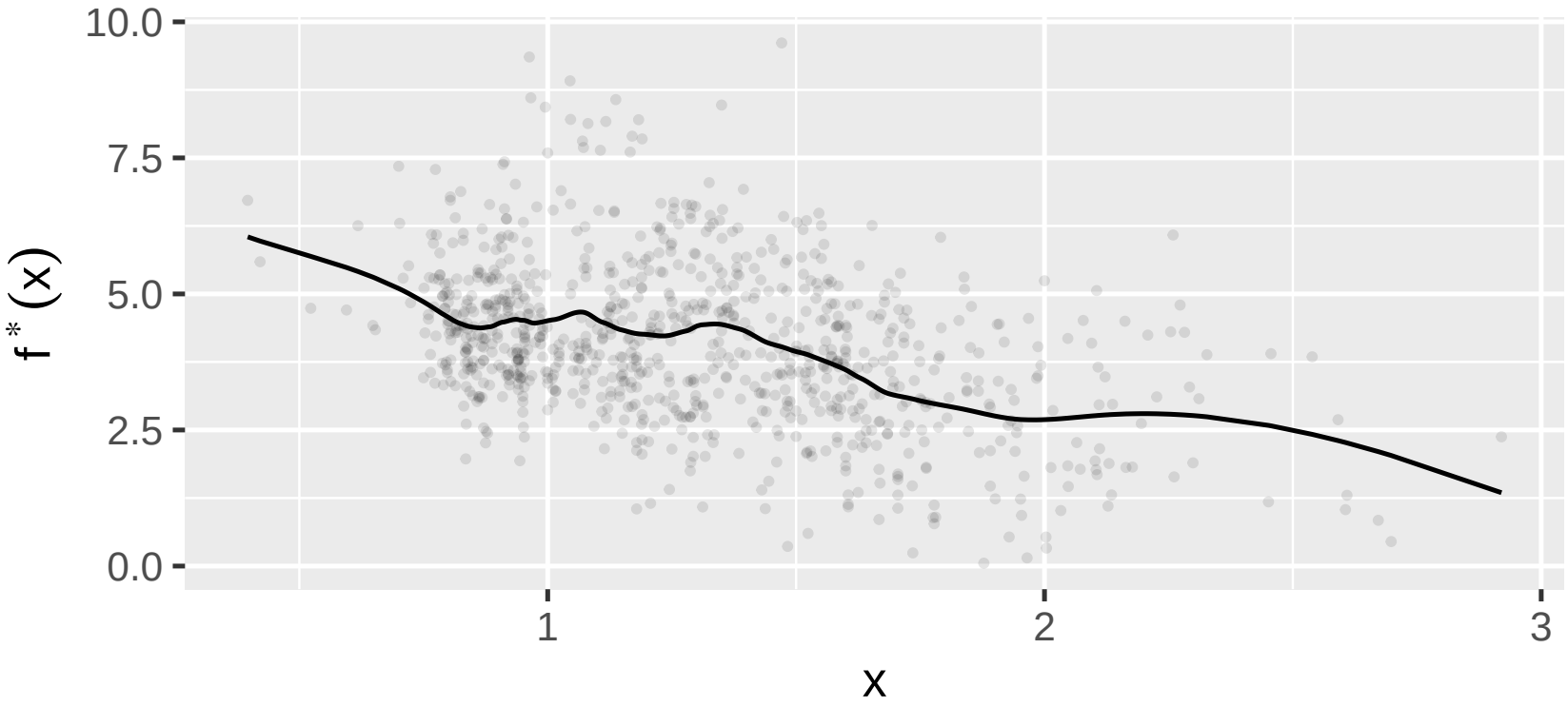
If we want to learn a function that makes $\mathcal{R}(f)$ small, we should draw training data $x_n \overset{\text{IID}}{\sim} p(x)$ and form the "empirical risk:"

$$\mathcal{R}(f) := \mathbb{E}_{p(x,y)}\left[\mathcal{L}(f(x), y)\right] \qquad \text{"Risk"}$$

$$\hat{\mathcal{R}}(f) := \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(f(x_n), y_n) \qquad \text{"Empirical risk".}$$

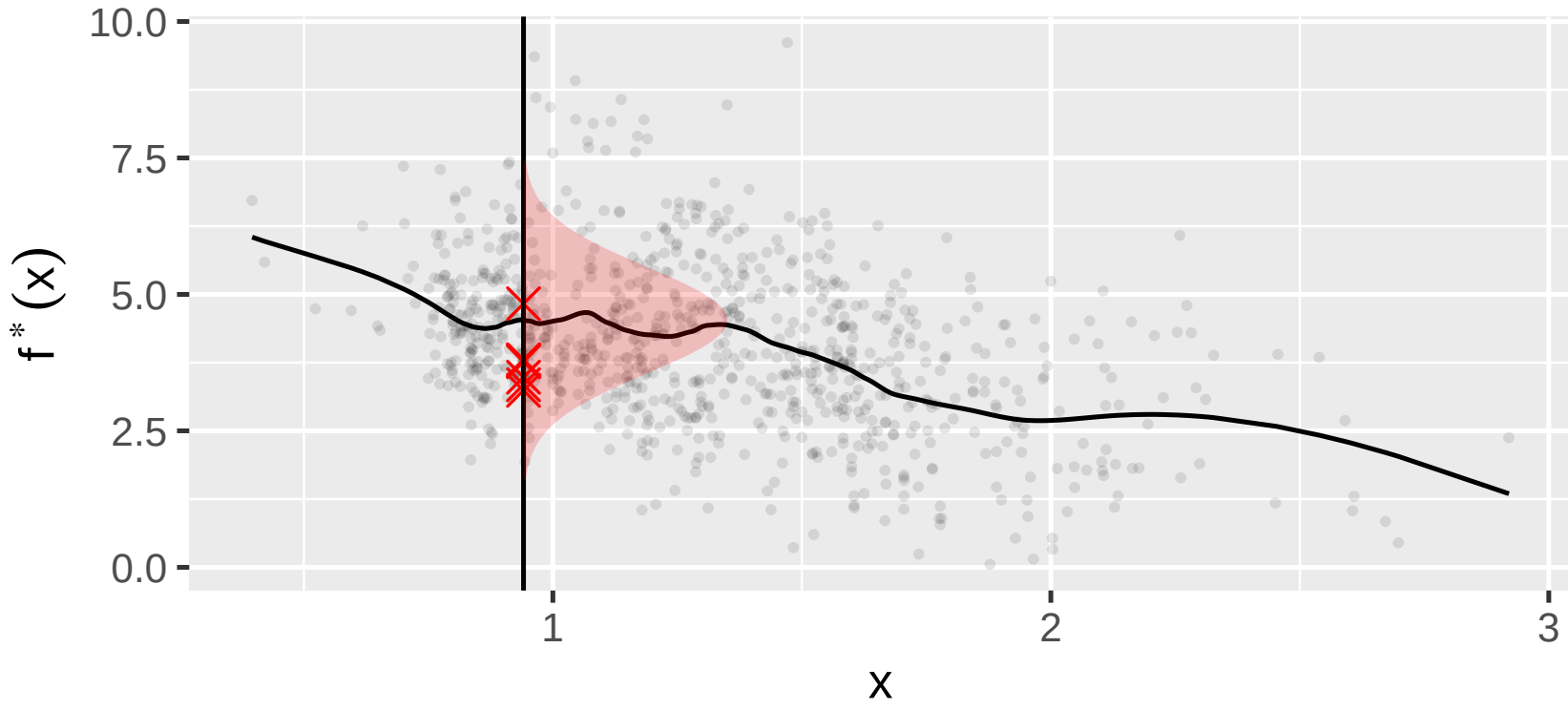
Trying to make $\mathcal{R}(f)$ small by minimizing $\hat{\mathcal{R}}(f)$ is called **empirical risk minimization** (ERM). ERM is the main idea in this course.

Response noise



Up to now, we've been imagining that $y = f^\star(x)$. But usually our data looks like this.

Response noise

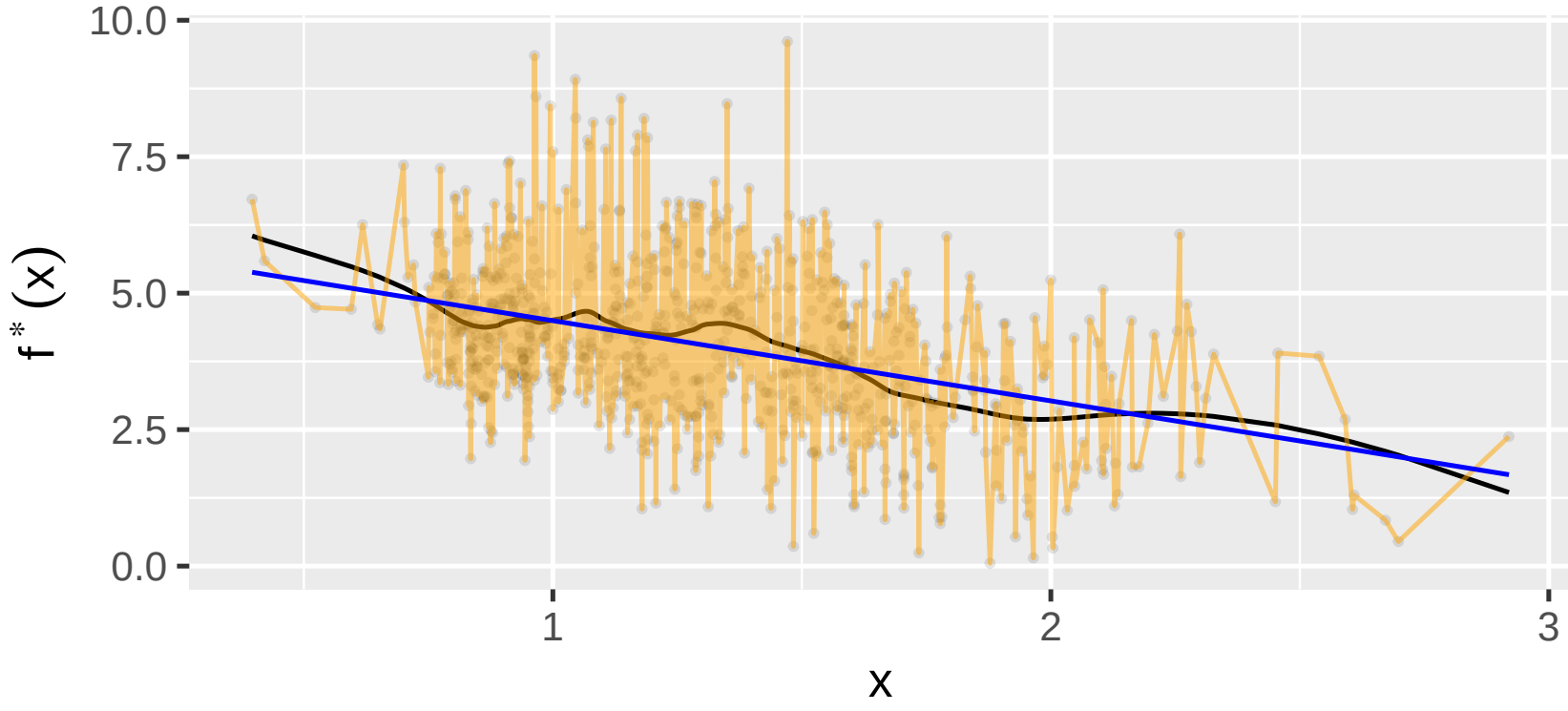


If you query the same x and get different values, you know that there is *randomness in* y. Even if you don't observe repeats, if "nearby" x give very different y, you might think of y as random.

If y is random, then at each x there is a conditional distribution, $p(y|x)$, which is different for each x.

Note that risk is the expectation over both x and y: $\mathcal{R}(f) := \mathbb{E}_{p(x,y)}\left[\mathcal{L}(f(x), y)\right]$.

Response noise



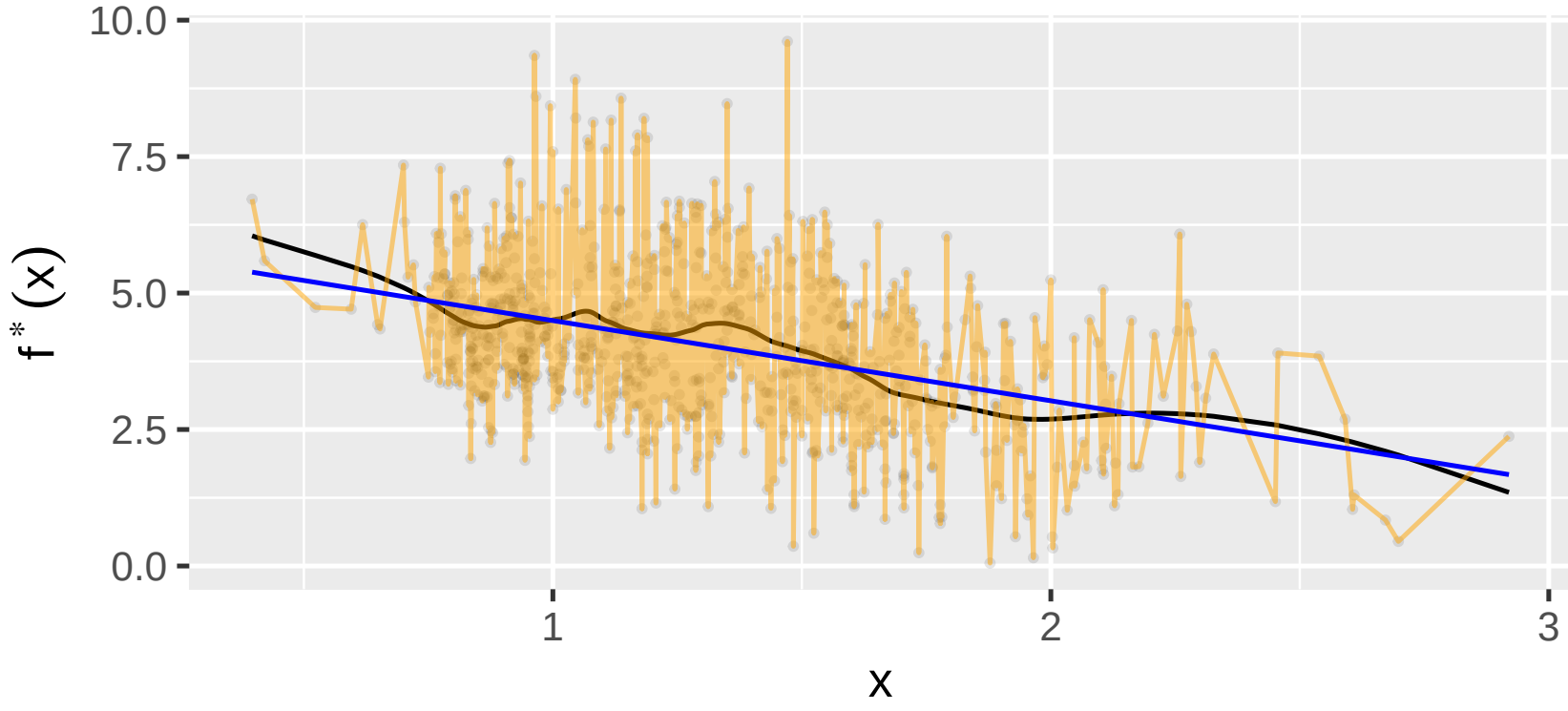
ERM finds the best function in our function class to the *actual data*:

$$\hat{f} := \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \hat{\mathcal{R}}(f)$$

The orange line is your $\hat{f}$ if $\mathcal{F}$ is *very expressive*!

If all you see is the data points, how can you tell which is the better prediction function?

Response noise

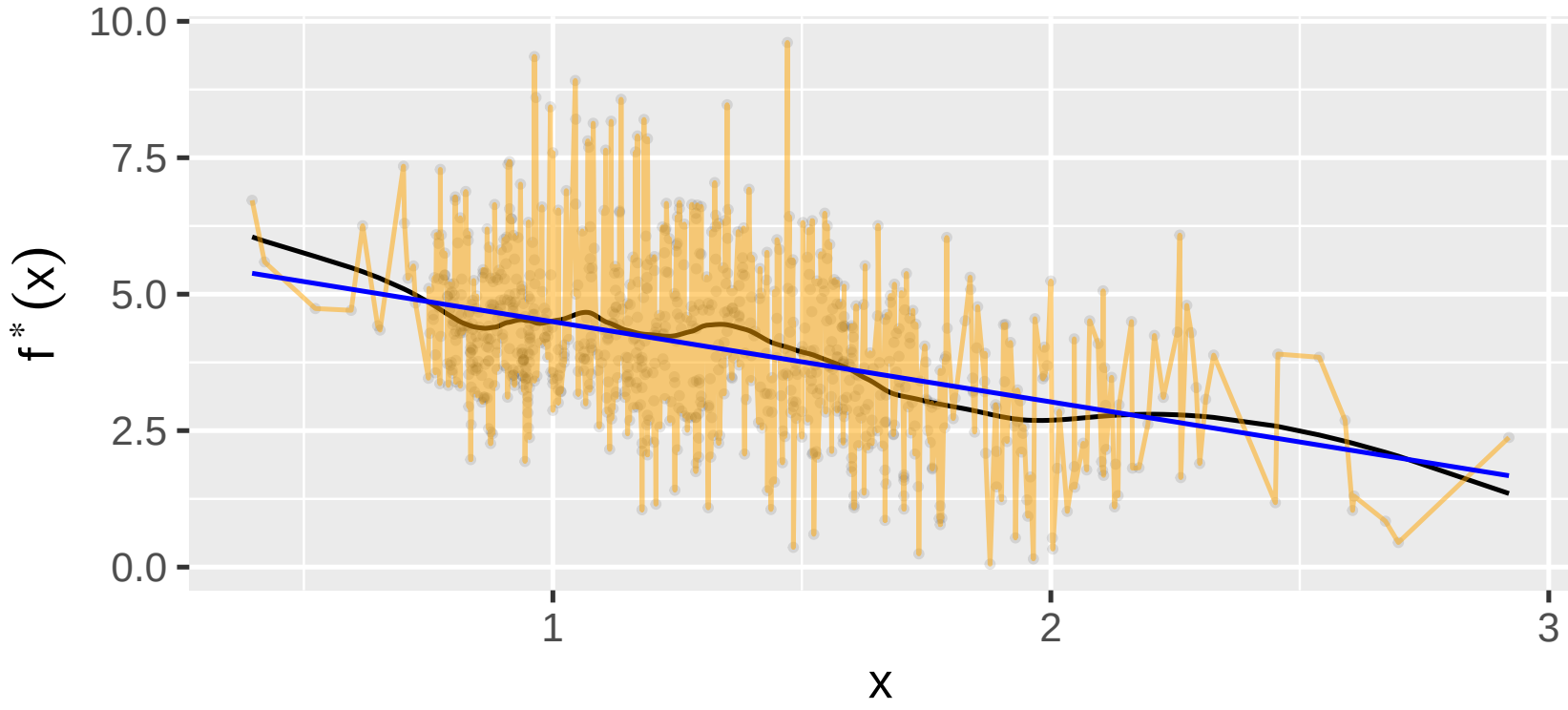


If the function class $\mathcal{F}$ is too expressive, we may fit the noise, not the risk.

$$\hat{f} := \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \hat{\mathcal{R}}(f) \quad \overline{f} := \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathcal{R}(f) \quad \Rightarrow \quad \mathcal{R}(\hat{f}) - \mathcal{R}(\overline{f}) \geq 0$$

This gap tends to be *higher* for *more expressive function classes*.

A fundamental tradeoff



$$\hat{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \hat{\mathcal{R}}(f) \qquad \overline{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \mathcal{R}(f) \qquad f^\star := \operatorname*{argmin}_{f} \mathcal{R}(f)$$

"Regret" is the gap between what we learn and what we might have learned. Its two constituent terms go in opposite directions with the expressivity of the function class:

$$\text{Regret} := \mathcal{R}(\hat{f}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(\hat{f}) - \mathcal{R}(\overline{f})}_{\text{Expressivity} \uparrow} + \underbrace{\mathcal{R}(\overline{f}) - \mathcal{R}(f^\star)}_{\text{Expressivity} \downarrow}$$

STAT 154 / 254 in a nutshell

$$\hat{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \hat{\mathcal{R}}(f) \qquad \overline{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \mathcal{R}(f) \qquad f^\star := \operatorname*{argmin}_{f} \mathcal{R}(f)$$

$$\text{Regret} := \mathcal{R}(\hat{f}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(\hat{f}) - \mathcal{R}(\overline{f})}_{\text{Expressivity} \uparrow} + \underbrace{\mathcal{R}(\overline{f}) - \mathcal{R}(f^\star)}_{\text{Expressivity} \downarrow}$$

In this class we will,

- Learn to make and fit expressive $\mathcal{F}$ so that $\mathcal{R}(\overline{f}) - \mathcal{R}(f^\star)$ is small,
- Learn how to make an $\mathcal{F}$ less expressive so that $\mathcal{R}(\hat{f}) - \mathcal{R}(f^\star)$ is small,
- ... in theory and in practice.

Together with a critical understanding of the IID assumption, the whole class can be understood in these terms.