2. Warm-up

A toy example

Coin Tossing

- Suppose that we have a biased coin
  - The outcome is either Head ($X = 1$) or Tail ($X = 0$)
  - Parametrized by the head probability $\theta := \mathbf{Pr}[X = 1]$
    - which is not known to us

- We toss the coin n times, and get the outcomes:

$$X_{1:n} = (X_1, X_2, \ldots, X_n)$$

  - Assume independence between tosses

Coin Tossing

- **Question.** How would you estimate the head probability θ, as a function of $X_{1:n}$?
  - That is, construct a good estimator $\hat{\theta} = f(X_1, \ldots, X_n)$
  - What "guarantee" do we have, i.e., an upper (and lower) bound on the quantity

$$\mathbf{Pr}[\,|\hat{\theta} - \theta| > \epsilon] = \,?$$

- Note. Why do we care about probability?
  - Sometimes we'll be unlucky, and get the samples $X_{1:n} = (1,1,1,1,\ldots,1)$, even when $\theta = 0.1$
    - However—thankfully—the probability of being unlucky will be very small!

Coin Tossing

- Unless you have a good prior, one would try the **empirical mean** (we'll justify later)

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_n$$

  - That is, the fraction of heads in the dataset.

- **Guarantee.** We can proceed as:

$$\mathbf{Pr}[\,|\hat{\theta} - \theta| > \epsilon] =$$

Coin Tossing

- We have the guarantee

$$\mathbf{Pr}[\,|\hat{\theta} - \theta| > \varepsilon] \leq 2 \cdot \exp(-2n\varepsilon^2)$$

- **Question.** How many samples do we need to guarantee an error less than ϵ with probability $1 - \delta$?
    - called "sample complexity"

Coin Tossing

- We have just analyzed the theoretical property of an estimator.
  - The estimator was the **empirical mean**

- This type of guarantee is called **PAC (probably approximately correct)**
  - Approximately correct, because we care about the event $|\theta - \hat{\theta}| > \varepsilon$
  - Probably, because the error happens with probability no larger than δ
    - Again, the randomness comes from the randomness of data draws

- Sometimes, we take a simpler path to bound something like $\mathbb{E}[|\hat{\theta} - \theta|]$
  - Can be related with PAC bound (e.g., via Markov's inequality)

Coin Tossing

- Note that our guarantee is an **upper bound (achievability)**
  - For some $\hat{\theta}$, we can be successful at least by this amount

- In some cases, we can come up with a **lower bound (converse)**
  - For any $\hat{\theta}$, we cannot achieve the error less than this amount
  - In this course, we won't discuss these much:
    - Requires much more statistical backgrounds:
      - Le Cam's method
      - Fano's method
      - Assouad's method

Coin Tossing

- We have used the empirical mean — **why?**

- **A perspective.** It minimizes some **loss function** w.r.t. the training data.
  - Suppose that we have a family of estimates, called "constant functions"

$$\mathscr{F}_{\text{con}} = \mathbb{R}$$

  - Then, we observe that the population mean (θ) is the MMSE estimate of X under the data-generating distribution P

$$\theta = \arg\min_{z \in \mathscr{F}_{\text{con}}} \mathbb{E}_{X \sim P}[(z - X)^2]$$

  - Likewise, the sample mean ($\hat{\theta}$) is the MMSE estimate of X under the empirical distribution P_n

$$\hat{\theta} = \arg\min_{z \in \mathscr{F}_{\text{con}}} \mathbb{E}_{X \sim P_n}[(z - X)^2]$$

    - As n increases, we know that $P_n \to P$ (in some sense), thus $\hat{\theta} \to \theta$.

Coin Tossing

- In this sense, we have analyzed the behavior of **empirical risk minimization** algorithm
  - Very restricted hypothesis space — $\mathscr{F}_{\text{con}}$
  - Assumed no suboptimality due to poor optimization — no SGD involved

- Let's develop this example into a full-fledged learning problem

Formalisms

Basic setup

- Throughout the course, we focus on the following setup:
  - **Task.** Supervised learning
    - either binary classification or regression
  - **Model.** Multilayer Perceptrons (MLPs)
    - i.e., fully-connected, feedforward networks
  - **Objective.** Empirical risk minimization
  - **Optimizer.** Gradient descent

- Let us be a little more specific…

Task: Supervised Learning

- **Training data.** We have $D = \{(x_i, y_i)\}_{i=1}^n$
  - We assume independence: $(x_i, y_i) \overset{\text{i.i.d.}}{\sim} P$
    - P is not known to the learner
  - Features: $x_i \in \mathcal{X}$
  - Labels: $y_i \in \mathcal{Y}$
    - For classification, we let $\mathcal{Y} = \{-1, +1\}$ or $\{0,1\}$
    - For regression, we let $\mathcal{Y} = \mathbb{R}$

Note: Sometimes, we assume that there exists a measurable function $y^*(\cdot)$ such that

$$y^*(x) = y$$

holds for all (x, y) drawn from P

Task: Supervised Learning

- **Goal.** Find a function such that $f(X) \approx Y$ for all likely data (X, Y)

  - More precisely, minimize the test risk
$$R(f) := \mathbb{E}_{(X,Y)\sim P}[\ell(f(X), Y)]$$

  - Here, $\ell(\cdot\,,\cdot)$ is some pre-defined loss function
    - Zero-one loss: $\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$
    - Logistic loss: $\ell(\hat{y}, y) = \log(1 + \exp(-y\hat{y}))$
    - Squared loss: $\ell(\hat{y}, y) = \|\hat{y} - y\|_2^2$

Model: Multilayer Perceptron

- Basically a repetition of fully-connected layers

- Each FC layer conducts:

$$x \mapsto \sigma(Wx + b)$$

  - Parametrized by: For some input width m_{in} and output width m_{out}
    - Weight matrix. $W \in \mathbb{R}^{m_{\text{out}} \times m_{\text{in}}}$
    - Bias vector. $b \in \mathbb{R}^{m_{\text{out}}}$

  - Further specified by:
    - Activation function. $\sigma : \mathbb{R}^m \to \mathbb{R}^m$
      - e.g., ReLU: $\sigma(x) = [x]_+ = \max\{x, 0\}$ applied entrywise

Model: Multilayer Perceptron

- For example, consider a **two-layer neural net with one-dimensional output**
  - Can be written as:

$$f(x; W, a, b) = \sum_{i=1}^{m} a_i \sigma(w_i^\top x + b_i)$$

  - Here, blue denotes the learnable parameters
  - Given some dataset, we'll want to optimize for the right (W, a, b)

Model: Multilayer Perceptron

- More generally, a **deep network** will be written as:

$$f(x; w) = \sigma_L(W_L \sigma_{L-1}(\cdots \sigma_1(W_1 x + b_1) \cdots + b_L)$$

  - We will simply use the shorthand

$$w = (W_1, b_1, \ldots, W_L, b_L)$$

- Based on this, we can parameterize a family of functions—i.e., a hypothesis space—as:

$$\mathcal{F} = \{f(\,\cdot\,; w) \mid W_i \in \mathbb{R}^{m_i \times m_{i-1}}, b_i \in \mathbb{R}^{m_i}\}$$

  - Later, we'll measure the complexity of this set

Algorithm: Empirical Risk Minimization

Definition (Empirical Risk).

Given some dataset $D = \{(x_i, y_i)\}_{i=1}^n$, the empirical risk is defined as

$$R_n(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

- Can be expressed in terms of the empirical distribution P_n as

$$R_n(f) = \mathbb{E}_{P_n}[\ell(f(X), Y)]$$

  - Convenient, as we know various concentration properties of the empirical distribution to the true distribution

$$P_n \longrightarrow P$$

  - Importantly, we want $R_n(f) \to R(f)$ for "all" $f \in \mathcal{F}$ simultaneously, not for just a single f
    - We'll see why later

Algorithm: Empirical Risk Minimization

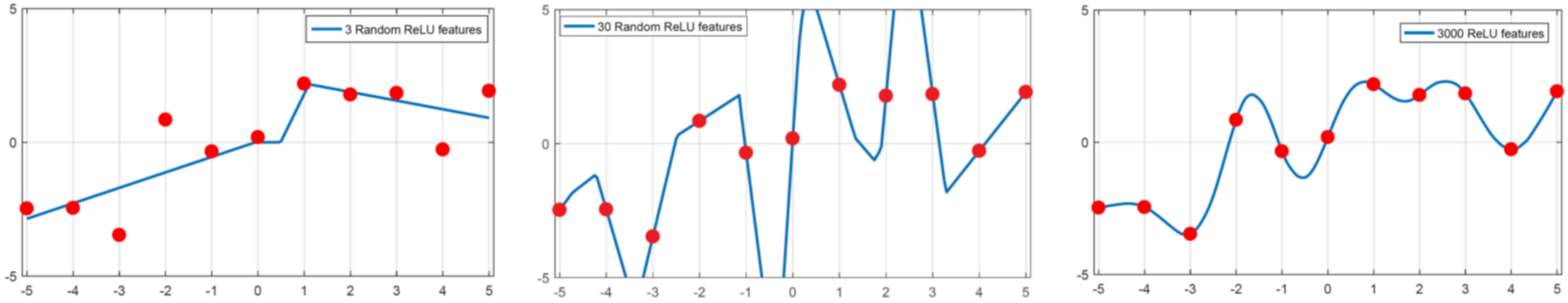
Definition (Empirical Risk Minimization).

Suppose that we have some hypothesis space $\mathcal{F}$.

The empirical risk minimization (ERM) is to solve

$$f_{\mathrm{ERM}} := \arg\min_{f \in \mathcal{F}} R_n(f)$$

- Note. The ERM solution may not be unique!
    - Indeed, this happens a lot in deep learning
    - Spoiler. Some generalize to the unseen data better than others
        - The mystery of deep learning is that somehow it seems to automatically find one that generalizes better than others (when sufficiently overparametrized)



Algorithm: Empirical Risk Minimization

- We now have various functions defined.
  - The empirical risk minimizer: $f_{\text{ERM}} := \arg\min_{f \in \mathcal{F}} R_n(f)$
  - What we actually get by SGD: $\hat{f}$
  - The "true" predictor: $f_{\text{GT}} := \arg\min_{f\text{:any func.}} R(f)$
  - The best we can do, given some $\mathcal{F}$: $f^* := \arg\min_{f \in \mathcal{F}} R(f)$
- **Question.** How do these relate?

First steps: Decomposition

Risk decomposition

- Consider the **excess risk** — the risk added by not knowing the ground truth

$$R(\hat{f}) - R(f_{\mathrm{GT}})$$

Risk decomposition

- By addition-and-subtraction, we can decompose the excess risk into **four terms**

$$R(\hat{f}) - R(f_{\text{GT}}) = \left[R(\hat{f}) - R_n(\hat{f})\right] + \left[R_n(\hat{f}) - R_n(f^*)\right] + \left[R_n(f^*) - R(f^*)\right] + \left[R(f^*) - R(f_{\text{GT}})\right]$$

  - No mysteries — simply added and subtracted empirical risks

- We can adapt a little bit, and show that:

$$R(\hat{f}) - R(f_{\text{GT}}) \leq \left[R(\hat{f}) - R_n(\hat{f})\right] + \left[R_n(\hat{f}) - R_n(f_{\text{ERM}})\right] + \left[R_n(f^*) - R(f^*)\right] + \left[R(f^*) - R(f_{\text{GT}})\right]$$

  - Can do this, because f_{ERM} achieves the smallest $R_n(\cdot)$

Risk decomposition

$$R(\hat{f}) - R(f_{\text{GT}})$$

$$\leq \boxed{\left[R(\hat{f}) - R_n(\hat{f})\right]} + \boxed{\left[R_n(\hat{f}) - R_n(f_{\text{ERM}})\right]} + \boxed{\left[R_n(f^*) - R(f^*)\right]} + \boxed{\left[R(f^*) - R(f_{\text{GT}})\right]}$$

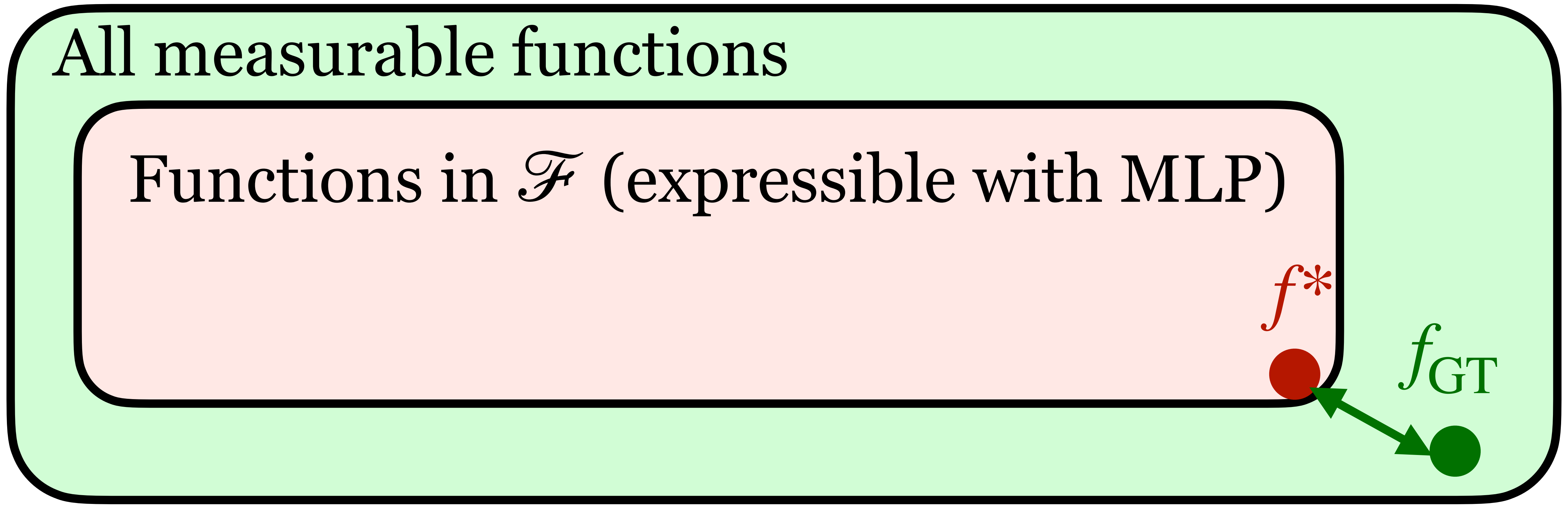
- These terms can be categorized into **3 kinds of penalties**:

Risk decomposition

$$R(\hat{f}) - R(f_{\text{GT}})$$
$$\leq \left[R(\hat{f}) - R_n(\hat{f})\right] + \left[R_n(\hat{f}) - R_n(f_{\text{ERM}})\right] + \left[R_n(f^*) - R(f^*)\right] + \boxed{\left[R(f^*) - R(f_{\text{GT}})\right]}$$

- These terms can be categorized into **3 kinds of penalties**:
  - **(1) Approximation: Penalty from insufficient expressivity**
    - Measures how rich the hypothesis set is

$$R(f^*) - R(f_{\text{GT}}) \quad = \quad \min_{f \in \mathcal{F}} R(f) - \min_{f \text{ meas.}} R(f)$$



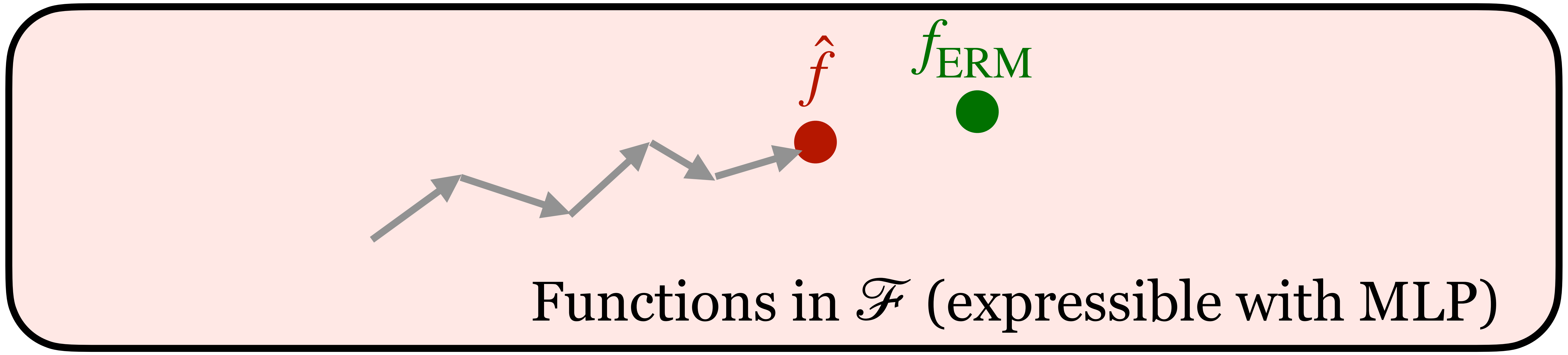
Risk decomposition

$$R(\hat{f}) - R(f_{\mathrm{GT}})$$
$$\leq \left[R(\hat{f}) - R_n(\hat{f})\right] + \boxed{\left[R_n(\hat{f}) - R_n(f_{\mathrm{ERM}})\right]} + \left[R_n(f^*) - R(f^*)\right] + \left[R(f^*) - R(f_{\mathrm{GT}})\right]$$

- These terms can be categorized into **3 kinds of penalties**:
  - **(2) Optimization: Penalty from imperfect fitting**
    - Measures how well we can perform ERM

$$R_n(\hat{f}) - R_n(f_{\mathrm{ERM}}) = R_n(\hat{f}) - \min_{f \in \mathcal{F}} R_n(f)$$

      - Smoothness and convexity matters, for convex optimization
      - For deep learning, there is an SGD magic involved



Risk decomposition

$$R(\hat{f}) - R(f_{\mathrm{GT}})$$
$$\leq \boxed{\left[R(\hat{f}) - R_n(\hat{f})\right]} + \left[R_n(\hat{f}) - R_n(f_{\mathrm{ERM}})\right] + \boxed{\left[R_n(f^*) - R(f^*)\right]} + \left[R(f^*) - R(f_{\mathrm{GT}})\right]$$

- These terms can be categorized into **3 kinds of penalties**:
  - **(3) Generalization: Penalty from scarce data**
    - Measures how well the dataset represents the distribution
    - Classically handled via the uniform deviation

$$R(\hat{f}) - R_n(\hat{f}) + R_n(f^*) - R(f^*) \quad \leq \quad 2\boxed{\sup_{f \in \mathcal{F}} |R(f) - R_n(f)|}$$

      - Note that this is essentially a stochastic quantity
        - Will need some concentration of measures to handle

Risk decomposition

- Throughout the course, we focus on analyzing three components, under the assumption that:
  - $\mathscr{F}$ is a family of functions expressible with MLP

$$f(x; w) = \sigma_L(W_L \sigma_{L-1}(\cdots \sigma_1(W_1 x + b_1) \cdots + b_L)$$

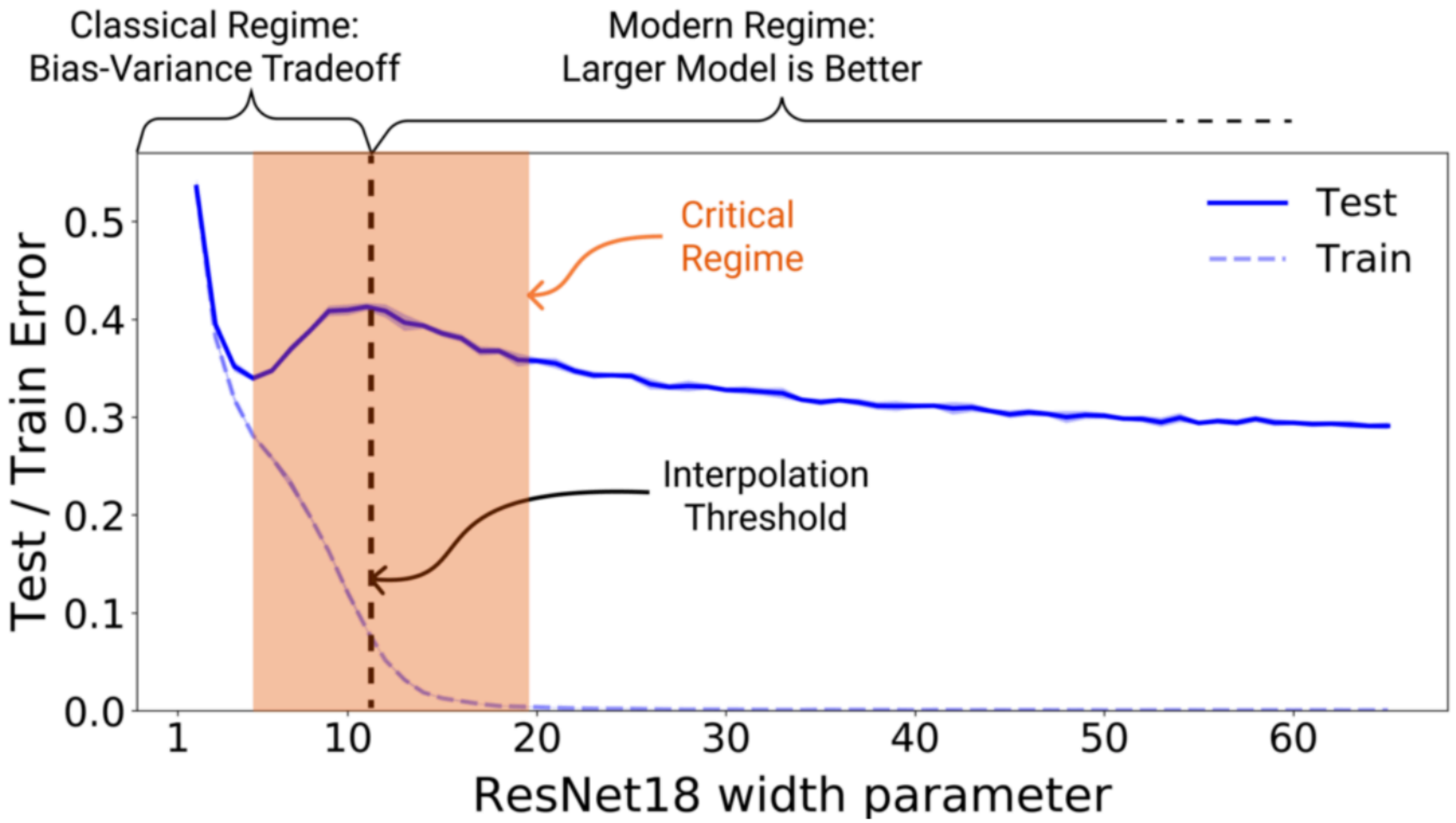
$$\mathscr{F} = \{f(\,\cdot\,; w) \mid W_i \in \mathbb{R}^{m_i \times m_{i-1}}, b_i \in \mathbb{R}^{m_i}\}$$

  - Optimization is done via empirical risk minimization, using gradient descent

$$\hat{f} = \mathrm{SGD}(f_{\mathrm{init}}, D)$$

Caveat

- Importantly, this decoupled approach is far from complete
  - Cannot explain the phenomenon that larger nets generalize better
  - More discussion in the optimization & generalization sections



Next up

- First steps on the **approximation**