Introduction to Machine Learning (IML)

LECTURE #2: CLASSIFICATION - PRELIMINARIES

236756 – 2024 WINTER – TECHNION

LECTURER: YONATAN BELINKOV

Classification

Today:

- Fundamentals of classification via simple example
- But don’t confuse *simple* with *easy*
- Sets ground for the first $\sim 2/3$ of the entire course
- We’ll revisit many of the issues we’ll discuss today
- So make sure you understand these fundamentals <u>now</u>

Classification

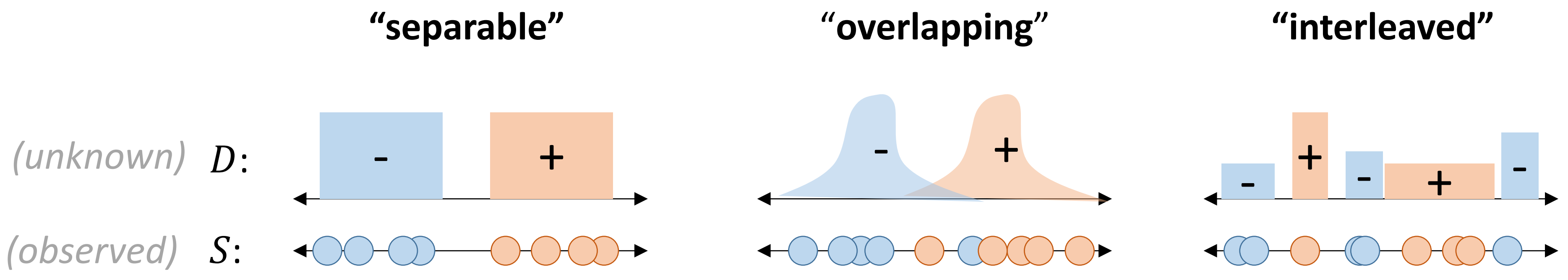
Recall:

- **Data generating process**: $(x, y) \overset{iid}{\sim} D,\ y \in \{\pm 1\}$ (classification)
- **Sample set**: $S = \{(x_i, y_i)\}_{i=1}^{m} \sim D^m$ (data)
- **Model class:** $H = \{h : h: \mathcal{X} \to \mathcal{Y}\}$
- **Expected error**: $L_D(h) = \mathbb{P}_D[y \neq h(x)] = \mathbb{E}_D[\mathbb{1}\{y \neq h(x)\}]$
- **Empirical error**: $L_S(h) = \mathbb{P}_S[y \neq h(x)] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$
- **Goal**: learn $h \in H$ with low $L_D(h)$
- **Means**: return $\hat{h} = \operatorname*{argmin}_{h \in H} L_S(h)$ (ERM)

Classification

Today:

- Learning on the real line, $\mathcal{X} = \mathbb{R}$
- Consider three "types" of distributions of increasing difficulty:



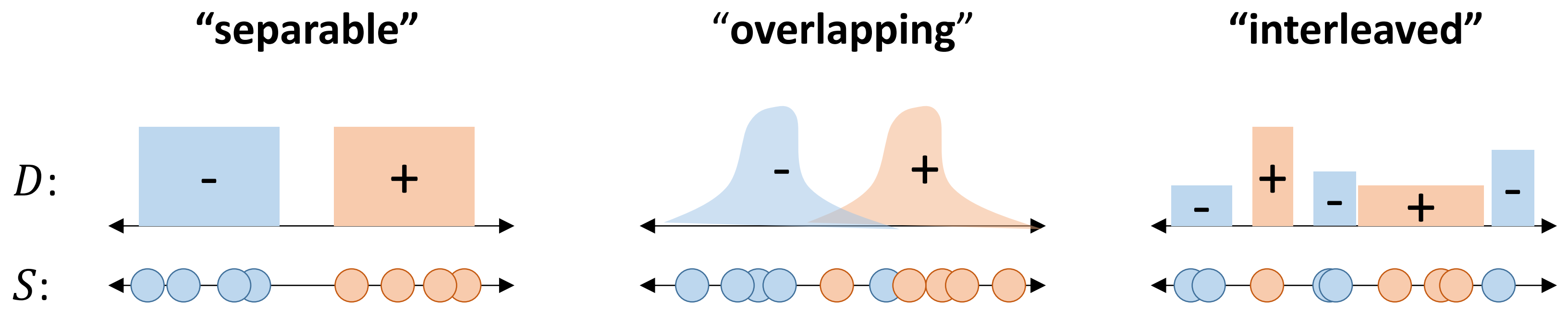
- For each, we will discuss the **three aspects of learning** we care about:

Modeling, Optimization, Statistics

Classification

Today:

- As we progress, we will make less and less assumptions
- Holy grail: having <u>no</u> assumptions on D
- Our goal for today: develop intuition for *distribution-independent* learning algorithms
- **Head's up**: we will ask many questions (but will answer relatively few)



Case I: Separable Data

Separable data

- **Definition (interim)***:

D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.

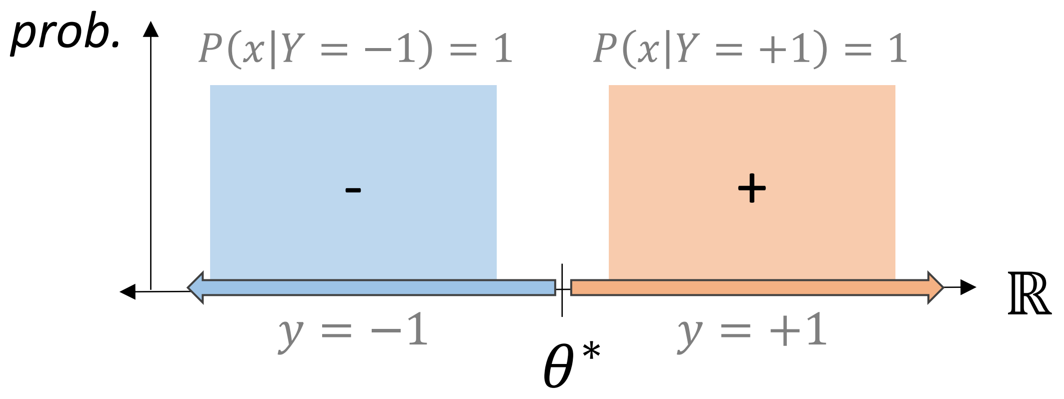
* This definition is not precise, and is in fact class-dependent. This will later be refined.

Separable data

- **Definition (interim)***:

D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.
- Example distribution:



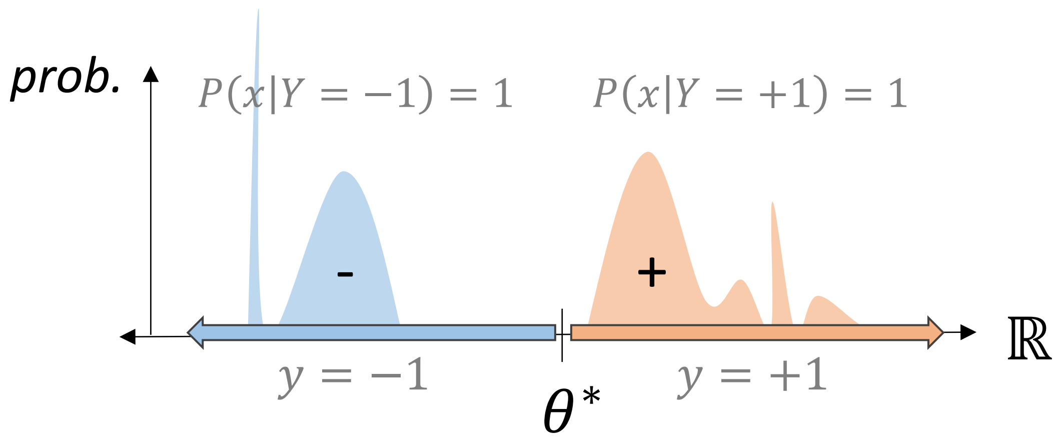
* This definition is not precise, and will later be refined.

Separable data

- **Definition (interim)***:

D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.
- Example distribution:



* This definition is not precise, and will later be refined.

Modeling

- Time to choose a model class!
- **Think** – what are good criteria for choosing?
- Let's try *threshold functions*:

$$\boxed{H_{\text{thresh}} = \{h_\theta(x) = \text{sign}(x \geq \theta) : \theta \in \mathbb{R}\}}$$

- **Q**: Why is this a good class?
- (**Remember**: we're assuming separability, but not any specific distribution)
- **A**: Because it's "just right" – intuitively:
  1. For any separable D there is some $h \in H$ with $L_S(h) = 0$
  2. And, it's not too large – no "redundant" h-s
- **Note:** $|H| = \infty$, suggesting that "large" is not the right measure (we'll return to this)

separable

Optimization

- Recall the ERM template
- To actually learn, need to implement for H_{thresh}
- Plug $h_\theta(x) = \text{sign}(x \geq \theta)$ into formulas

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective**:
  classifier with lowest expected error
  $$h^* = \operatorname*{argmin}_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Optimization

- Recall the ERM template
- To actually learn, need to implement for H_{thresh}
- Plug $h_\theta(x) = \text{sign}(x \geq \theta)$ into formulas
- We will call θ the **parameter**(s) of h_θ
- Optimizing over $h \equiv$ optimizing over θ
- This is an example of model class *structure*, which we will see is helpful (and necessary)
- Can we solve ERM now?

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error

$$\theta^* = \underset{\theta \in \mathbb{R}}{\text{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq \text{sign}(x \geq \theta)\}]$$

- **Return**:
  classifier with lowest empirical error

$$\hat{\theta} = \underset{\theta \in \mathbb{R}}{\text{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq \text{sign}(x_i \geq \theta)\}$$

separable

Optimization

Simple algorithm:

1. Sort x_i-s

1b. Validate separability

2. Find largest x_i with $y_i = -1$; call it x_{i^*}
3. Return $\hat{\theta} = x_{i^*}$

- **Congrats**! We just implemented our first ERM
- What do we need to prove?
- (Remember the algorithm is designed for *threshold classifiers* on *separable data*)

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^{m}$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $$\theta^* = \underset{\theta \in \mathbb{R}}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq \operatorname{sign}(x \geq \theta)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{\theta} = \underset{\theta \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq \operatorname{sign}(x_i \geq \theta)\}$$

Optimization

Simple algorithm:

1. Sort x_i-s $O(m \log m)$

1b. Validate separability $O(m)$

2. Find largest x_i with $y_i = -1$; call it x_{i^*} $O(m)$
3. Return $\hat{\theta} = x_{i^*}$

- **Congrats**! We just implemented our first ERM
- What do we need to prove?
- (Remember the algorithm is designed for *threshold classifiers* on *separable data)*

1. **Runtime**: $O(m \log m)$ – great!

- Recall $m = |S|$
- But H is also "input", and $|H| = \infty$! *How can this be?*

2. **Correctness**: $L_S(h_{\hat{\theta}}) = 0$

- Easy to show.
- Not only solution (what are others?)

3. ***Generalization***: $L_D(h_{\hat{\theta}}) = ?$

- How can we relate L_S and L_D?
- Enter *statistics.*

Statistics I

- Algorithm returns $\hat{\theta}$ with zero *empirical* error, $L_S(h_{\hat{\theta}}) = 0$
- But we actually care about its *expected* error, $L_D(h_{\hat{\theta}})$
- **Q**: What does $L_D(h_{\hat{\theta}})$ depend on? What can cause it to vary?
- **A**: m

- We expect L_D to go down with m – but at what rate?
- **Let's simulate!**
- Empirically, we can see that $L_D \approx \frac{1}{m}$
- Can we run this simulation on real problems? (or – where did we cheat?)
- This is where theory comes in: bound rate *analytically*

Statistics II

- Our algorithm takes in S and returns $\hat{\theta}$
- But different $S \Rightarrow$ different $\hat{\theta} \Rightarrow$ different $L_D(\hat{h})$! (even for fixed m)
- Output of ERM is a **random variable** (because it's a function of the random variable S)
- How much variance in performance should we expect?
- **Let's simulate!**

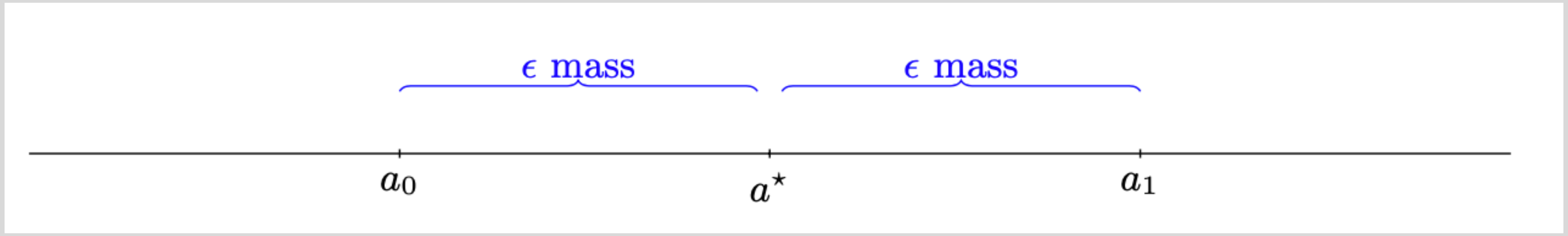
- **Q**: So how can we tell if our algorithm is "good"?
- **A**: Use **probabilistic criteria** (w.r.t. $S \sim D^m$)
- **Corollary:**

Performance on a single dataset has less meaning than we'd like to believe

Statistics III

- Let’s see this in action.
- **Claim**: $P_{S\sim D^m}\left(L_D(\hat{h}) > \epsilon\right) \leq 2e^{-\epsilon m}$
- **Proof:** [on board; UML book Sec. 6.1, p67]
- **Corollary**: $\epsilon \geq \frac{1}{m}\log(\frac{2}{\delta})$

Statistics III



Let a* be a threshold such that the hypothesis $h^*(x)=\text{sign}(x \geq a^*)$ achieves $L_D(h^*) = 0$. Let $a_0 < a^* < a_1$ be two points as in the figure. Formally, $P_{Dx}(x \in (a_0,a^*)) = P_{Dx}(x \in (a^*, a_1)) = \varepsilon$.

Given a set S, define $b_0 = \max\{x_i : y_i = -1\}$ and $b_1 = \min\{x_i : y_i = 1\}$
If b_S is a threshold for ERM hypothesis h_S, then $b_S \in (b_0, b_1)$
A sufficient condition for $L_D(h_S) \leq \varepsilon$ is: $b_0 \geq a_0$ and $b_1 \leq a_1$. Or:
$P_S{\sim}D^m (L_D(h_S) > \varepsilon) \leq P_S{\sim}D^m (b_0 < a_0 \text{ or } b_1 > a_1) \leq P_S{\sim}D^m (b_0 < a_0) + P_S{\sim}D^m (b_1 > a_1)$, where the last inequality is by union bound.
Now, $b_0 < a_0$ iff all samples in S are not in (a_0, a^*), whose prob mass is ε, so:
$P_S{\sim}D^m (b_0 < a_0) = P_S{\sim}D^m (\forall (x,y) \text{ in } S, x \notin (a_0, a^*)) = (1-\varepsilon)^m \leq e^{-\varepsilon m}$.
In the same way, $P_S{\sim}D^m (b_1 > a_1) \leq e^{-\varepsilon m}$. Adding both, we get the desired bound.

Discussion

- **Recall**: our algorithm sets $\hat{\theta}$ to be the largest negative example
- **Observation**: just as good – any point between *largest negative* and *smallest positive*
- **Conclusion**: there can be multiple possible ERM rules!

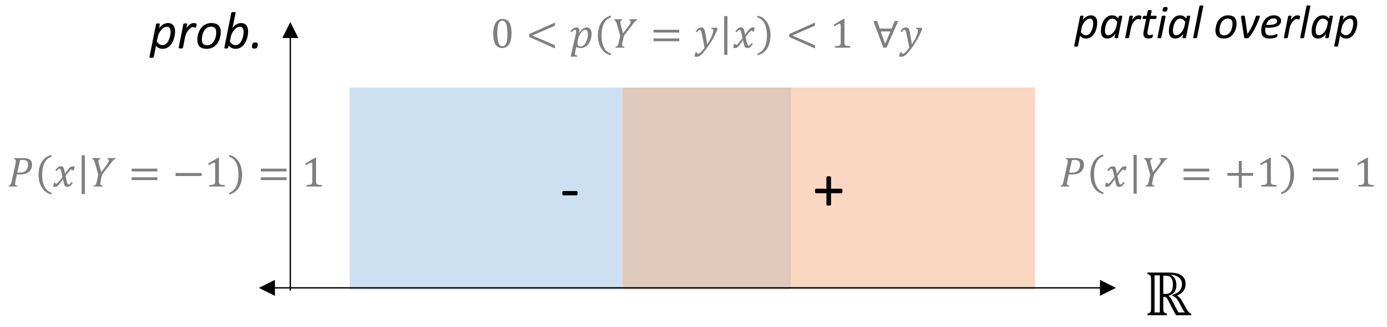
  largest negative, smallest positive, mean, median, convex combination, random in interval, …

- **Ask yourself:**
  - Are some ERM rules better than others for some *D*?
  - Are some ERM rules better than others for all *D?*
  - Do your answers implicitly rely on additional assumptions?
  - Are your claims statistical (*observed*), distributional (*latent*), or both?
  - How would you validate your claims?
  - Actually – what does "better" even mean? Be precise!

Case II: Overlapping Classes

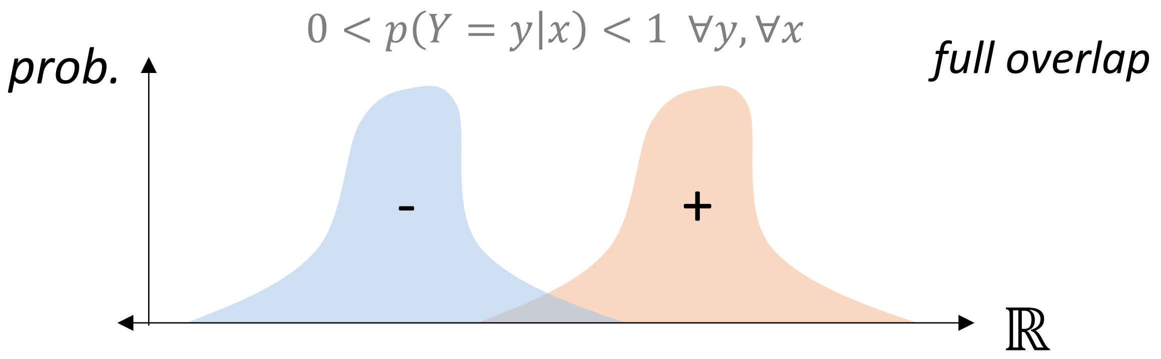
Overlapping classes

- **Hand-wavy**: continuous "middle" region where labels are uncertain
- Some example distributions:



Overlapping classes

- **Hand-wavy**: continuous "middle" region where labels are uncertain
- Some example distributions:



- Separable is special case (=no overlap)
- (Our new algorithm should be "backward-compatible")
- But in general, no θ^*-like object
- (**Remember:** we only observe samples with binary y)

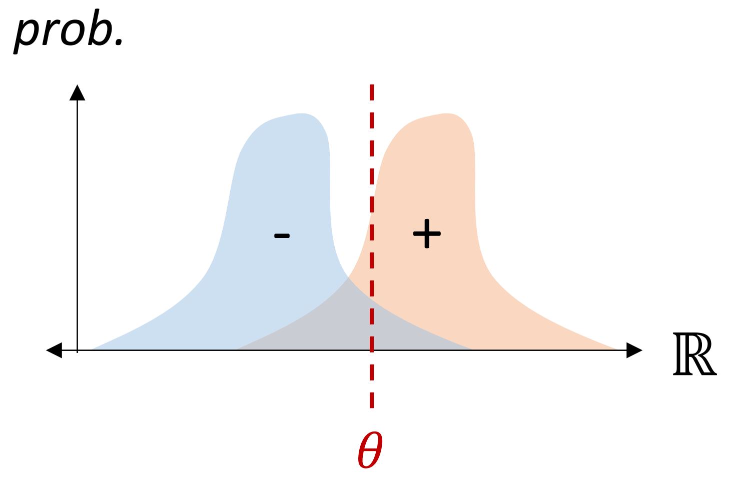
Modeling

- What would be a good model class now?
- How about… thresholds?
- **Recall**:

$$H_{\text{thresh}} = \{h_\theta(x) = \text{sign}(x \geq \theta) : \theta \in \mathbb{R}\}$$

- **(Sneak-peak)**: will work out well
- But with *one big difference*:

minimal error can be strictly positive



overlapping

Optimization

Recall our previous (separable) ERM algorithm:

Simple algorithm:

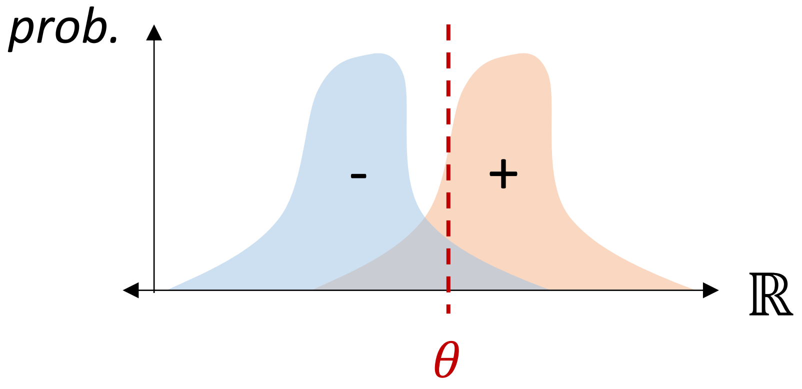
1. Sort x_i-s

1b. Validate separability

2. Find largest x_i with $y_i = -1$; call it x_{i^*}
3. Return $\hat{\theta} = x_{i^*}$

- **Q**: Will it still work?
- **A**: no – it breaks! (consider mixture of Gaussians)

➢ **Take away**: be careful in what you assume



Optimization

Recall our previous (separable) ERM algorithm:

Simple algorithm:

1. Sort x_i-s

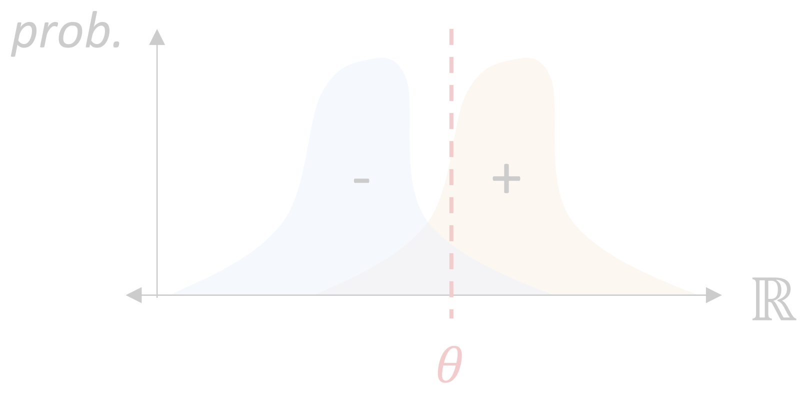
1b. Validate separability

2. Find largest x_i with $y_i = -1$; call it x_{i^*}
3. Return $\hat{\theta} = x_{i^*}$

- **Q**: Will it still work?
- **A**: no – it breaks! (consider mixture of Gaussians)

➢ **Take away**: be careful in what you assume

- Nonetheless, is there an underlying principal we can salvage?
- Recall: $\hat{\theta} = x_{i^*}$
- Claim: x_{i^*} was useful because it was an **informative summary** of the data
- Let's apply this principle to our current problem



Optimization

- Let’s try again:

Revised algorithm:

```
1. Sort $x_i$-s
2. For each $x_i$, compute empirical error
   “as if” $x_i$ was the threshold
3. Find example with lowest error; call it $x_{i^*}$
4. Return $\hat{\theta} = x_{i^*}$
```

- **Runtime**: $O(m \log m)$ (backward and forward counting passes)
- **Correctness**: $\hat{h} = h_{\hat{\theta}} = \operatorname{argmin}_{h \in H} L_S(h)$ (easy to show)

- Nonetheless, is there an underlying principal we can salvage?
- Recall: $\hat{\theta} = x_{i^*}$
- Claim: x_{i^*} was useful because it was an **informative summary** of the data
- Let’s apply this principle to our current problem

overlapping

Optimization

- Let's try again:

Revised algorithm:

1. Sort x_i-s
2. For each x_i, compute empirical error "as if" x_i was the threshold
3. Find example with lowest error; call it x_{i^*}
4. Return $\hat{\theta} = x_{i^*}$

- **Runtime**: $O(m \log m)$
- **Correctness**: $\hat{h} = h_{\hat{\theta}} = \text{argmin}_{h \in H} L_S(h)$

- **Congrats**! We just implemented a more general ERM algorithm.
- Note that the learned $\hat{h}$ is a very simple function of the data.
- **Hand-wavy**: this in effect "removes" the runtime dependence on H.
- This principle appears in many learning algorithms.
- Challenge is often in finding (and appropriately combining) the "important" elements of the data.

overlapping

Statistics

- **Observation**: minimal empirical error can be *strictly positive* (e.g., when data is not separable)
- Imagine we ran our ERM and got an (empirical) error of $L_S(\hat{h}) = 0.1$.
- **Q**: Should we be happy?
- **A**: It’s complicated.

- Our principle concern is **generalization**.
- But we can’t compute it…

generalization:

$$\begin{tikzcd} L_D(\hat{h}) \arrow[loop above] \end{tikzcd}$$

Statistics

- **Observation**: minimal empirical error can be *strictly positive* (e.g., when data is not separable)
- Imagine we ran our ERM and got an (empirical) error of $L_S(\hat{h}) = 0.1$.
- **Q**: Should we be happy?
- **A**: It’s complicated.

generalization:

$$\begin{tikzcd} L_D & (\hat{h}) \arrow[l, bend right] & \overset{?}{\gtrless} & L_S(\hat{h}) & \geq 0 \end{tikzcd}$$

- Our principle concern is **generalization**.
- But we can’t compute it…
- **Observations**:

1. We can’t just assume empirical error ≈ expected error

Statistics

- **Observation**: minimal empirical error can be *strictly positive* (e.g., when data is not separable)
- Imagine we ran our ERM and got an (empirical) error of $L_S(\hat{h}) = 0.1$.
- **Q**: Should we be happy?
- **A**: It's complicated.

generalization:

separable:

$$0 = \quad L_D(h^*) \leq L_D(\hat{h}) \overset{?}{\gtrless} L_S(\hat{h}) \geq 0$$

- Our principle concern is **generalization**.
- But we can't compute it… nor a bound.
- **Observations**:

1. We can't just assume empirical error $\approx$ expected error
2. "Good" is *relative* (to optimal classifier $h^* = \operatorname{argmin}_{h \in H} L_D(h)$)

Statistics

- Trivial (but helpful!) *error decomposition*:

$$\underbrace{L_D(\hat{h})}_{\textit{generalization}} = \underbrace{L_D(h^*)}_{\textit{approximation error}} + \underbrace{L_D(\hat{h}) - L_D(h^*)}_{\textit{estimation error}}$$

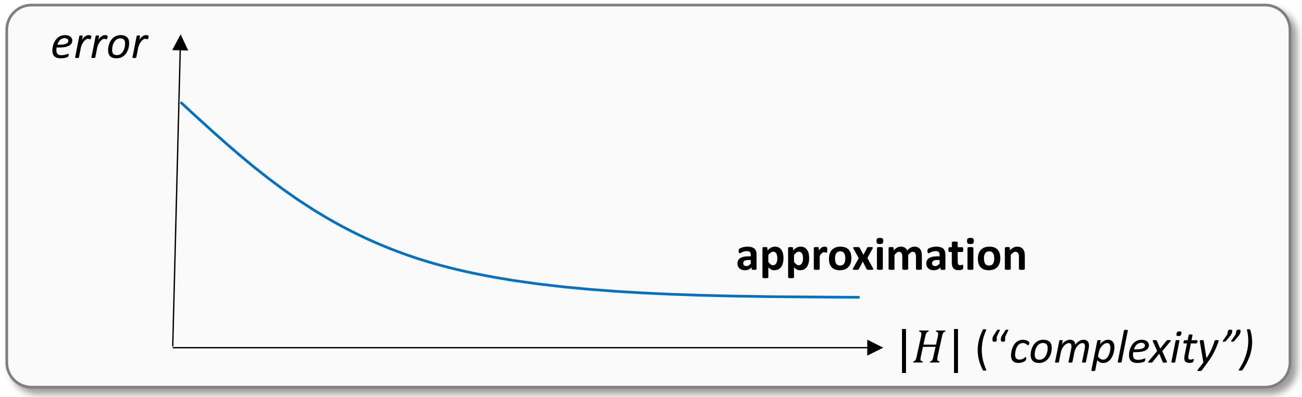
- **Approximation** = "how good H is for D" (independent of S)
- **Estimation** = "given H, how good is S for D" (empirical error as estimate of expected)

- **Conclusion**: model class H is useful if:
  - We believe it can approximate the data well *[modeling; approximation]*
  - We can associate empirical and expected errors *[statistics; estimation]*
  - We know we can compute ERM *[optimization; estimation]*
- **Remember**: The only thing we can compute (exactly) is $L_S(\hat{h})$! **But where is it?**

Statistics

$$\underbrace{L_D(\hat{h})}_{\textit{generalization}} = \underbrace{L_D(h^*)}_{\textit{approximation error}} + \underbrace{L_D(\hat{h}) - L_D(h^*)}_{\textit{estimation error}}$$

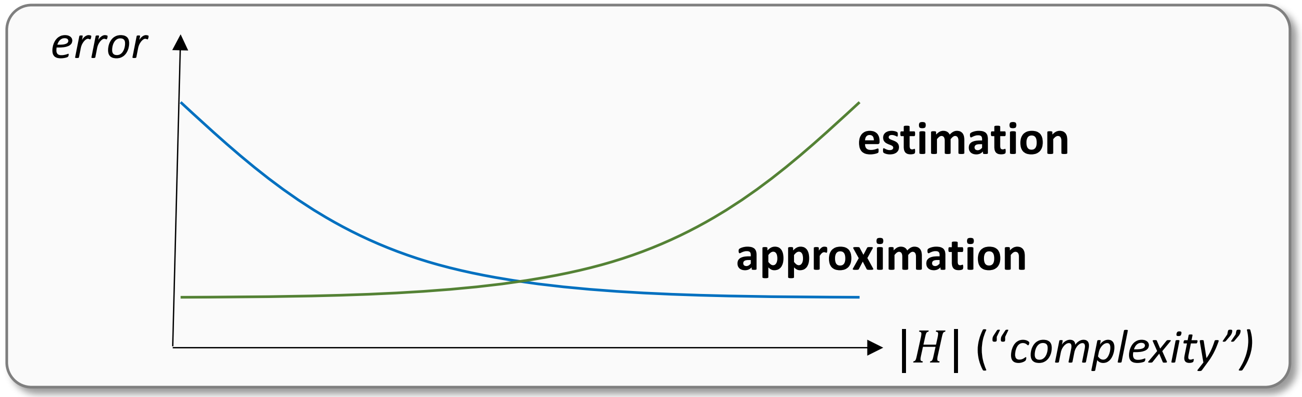
- **Intuition**: as we increase the "size" of H (keeping m fixed):
  - larger H -> less bias ("assumptions") -> lower approx. error -> better generalization



Statistics

$$\underbrace{L_D(\hat{h})}_{\textbf{generalization}} = \underbrace{L_D(h^*)}_{\textbf{approximation error}} + \underbrace{L_D(\hat{h}) - L_D(h^*)}_{\textbf{estimation error}}$$

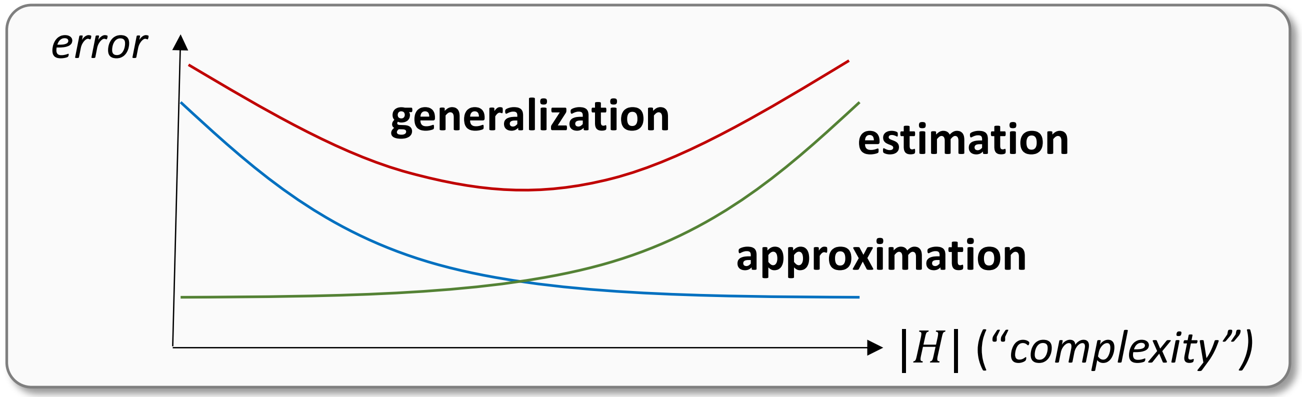
- **Intuition**: as we increase the "size" of H (keeping m fixed):
  - larger H -> less bias ("assumptions") -> lower approx. error -> better generalization
  - larger H -> worse estimation -> higher error -> worse generalization



Statistics

$$\underbrace{L_D(\hat{h})}_{\textit{generalization}} = \underbrace{L_D(h^*)}_{\textit{approximation error}} + \underbrace{L_D(\hat{h}) - L_D(h^*)}_{\textit{estimation error}}$$

- **Intuition**: as we increase the "size" of H (keeping m fixed):
  - larger H -> less bias ("assumptions") -> lower approx. error -> better generalization
  - larger H -> worse estimation -> higher error -> worse generalization



- This is called the **bias-complexity tradeoff**, we will later see why.

overlapping

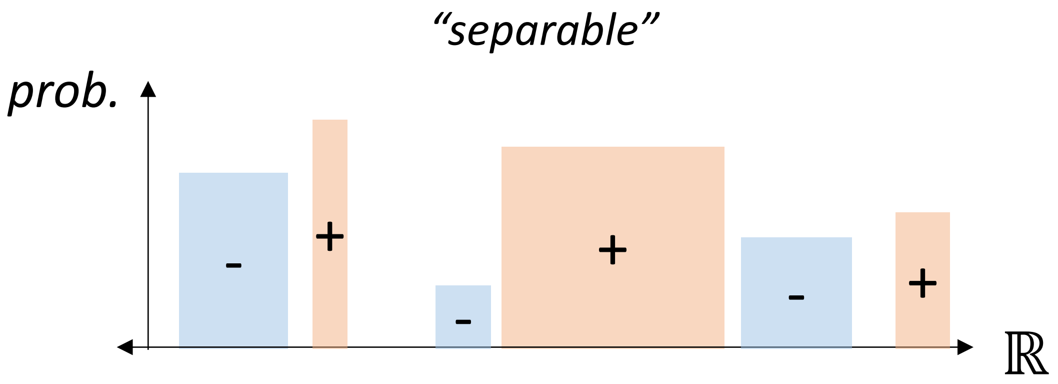
Discussion

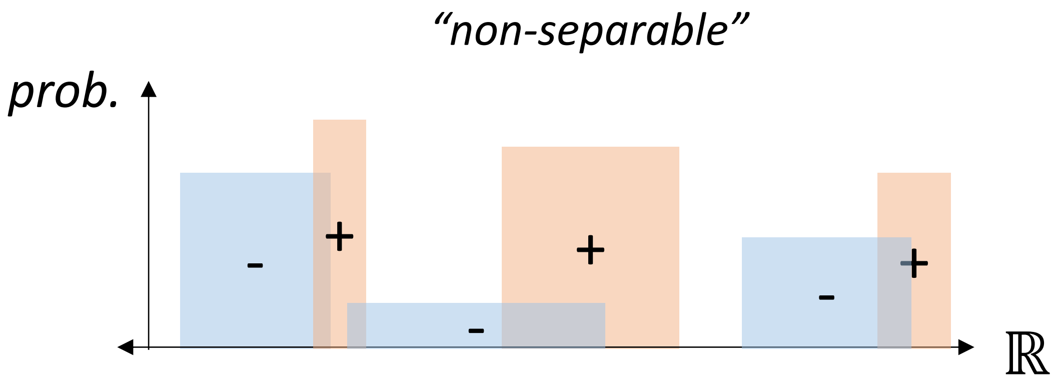
- **Recall:** we assume data is *jointly* sampled $(x, y) \sim D_{XY}$
- Classification focuses on $p(y|x)$ (where $p(x, y) = p(x)p(y|x)$)
- But sometimes it helps to consider $p(x|y)$, such as when we:
  - Have assumptions on data distribution
  - Directly model the distribution
- (Think of the distribution “types” we discussed today)
- Above quantities are related through *Bayes rule*:
$$p(y|x) = p(x|y)p(y)/p(x)$$
- Learning $p(x|y)$ is called ***generative learning*** – we will return to this later in the course.

Case III: Interleaved Classes

Interleaved classes

- **Hand-wavy**: more than one $D_{X|Y=y}$ for each class y
- Example distributions:





- **Note**: effectively, unbounded "chunks" + non-separable = no assumptions
- This is where we're aiming!

Modeling

$$L_D(\hat{h}) \quad = \quad L_D(h^*) \; + \; L_D(\hat{h}) - L_D(h^*)$$

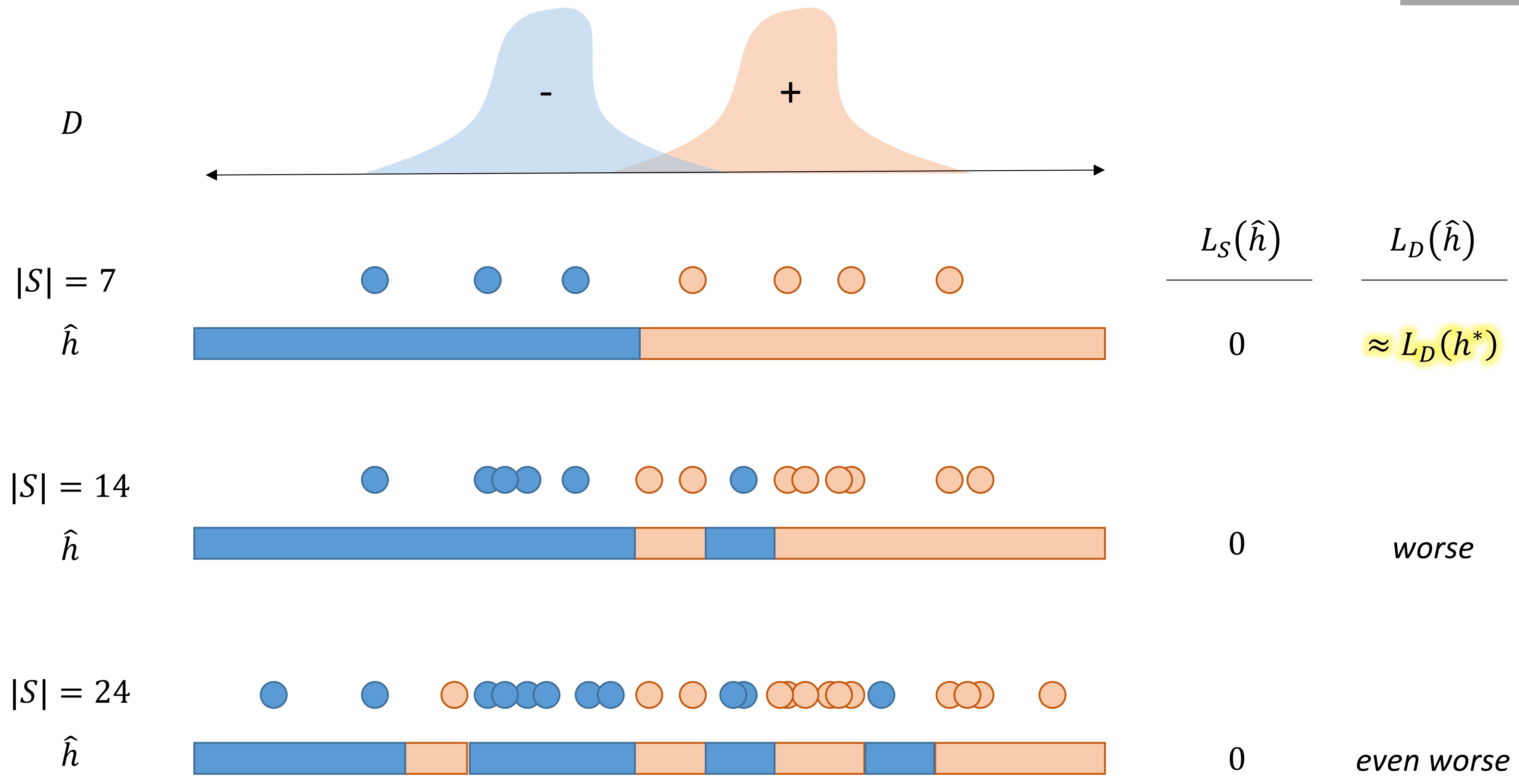
- **Q**: Are thresholds still a good choice?
- **A**: Yes! Because they are statistically well-behaved (low *estimation error*)
- **A**: No! Because may not be best alternative (high *approximation error*)

- Ideas for better class?
- **Union of (half) intervals:**

$$h_I(x) = \mathbb{1}\{x \in \cup_j \, (a_j, b_j)\}, \qquad I = \{(a_j, b_j)\}_{j=1}^{d}$$

- **On separable data**: approximation error = 0 (easy to show)
- But…

interleaved

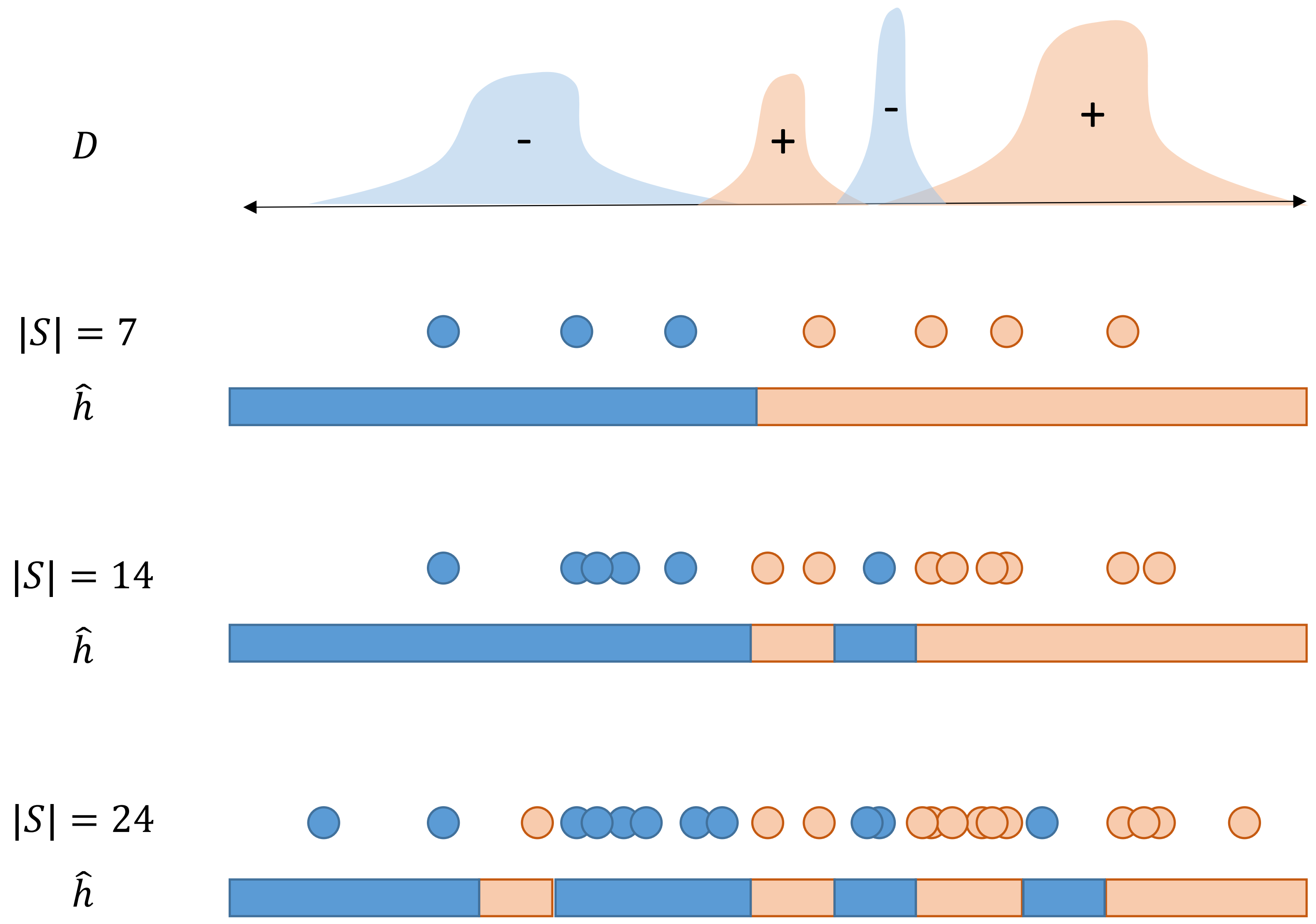


Statistics

- **Conclusion**: more data is… *worse*? What's going on here?
- **Observation**: with enough intervals (d) – can always get zero empirical error!
- Models with larger d are more *expressive*
- We will think of *expressivity* as controlled by model class *complexity*
- For intervals, complexity = number of intervals

- **Notice:** as we added more data, we also increased model complexity
- Our example shows: increasing complexity can hurt expected error
- This is a fundamental concept in learning called ***overfitting***
- (Insufficient complexity is called *underfitting)*

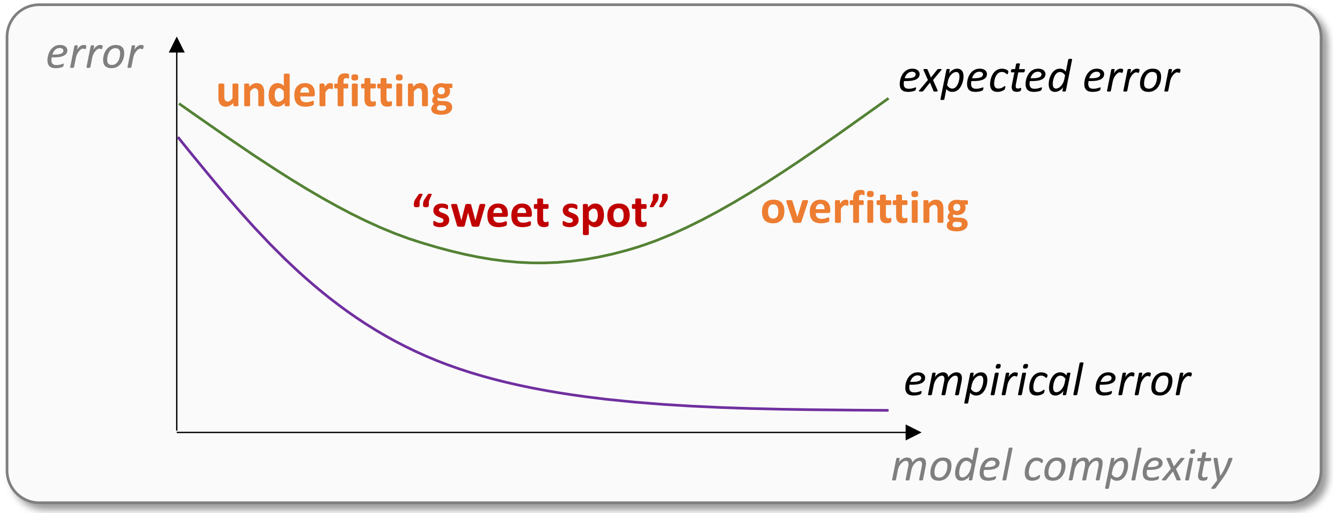
interleaved



$L_S(\hat{h})$	$L_D(\hat{h})$
0	*worse*
0	$\approx L_D(h^*)$
0	*worse*

Statistics

- Let's generalize this intuition. For fixed data size m:



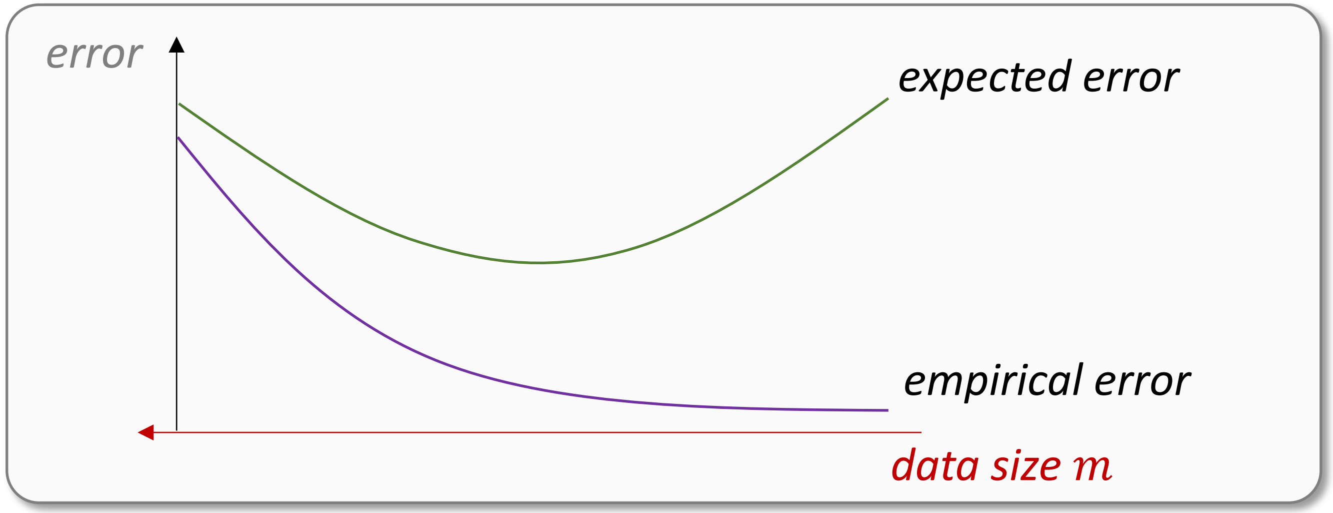
- Recall that we would like to minimize *expected* error.
- Plot suggests that we should **control complexity**. But how? Think:
  - What can we compute?
  - What *can't* we compute?

Statistics

- In our example, model complexity increased when *more examples* were observed.
- What role does data size m play in over/under-fitting?

Statistics

- In our example, model complexity increased when *more examples* were observed.
- What role does data size m play in over/under-fitting?



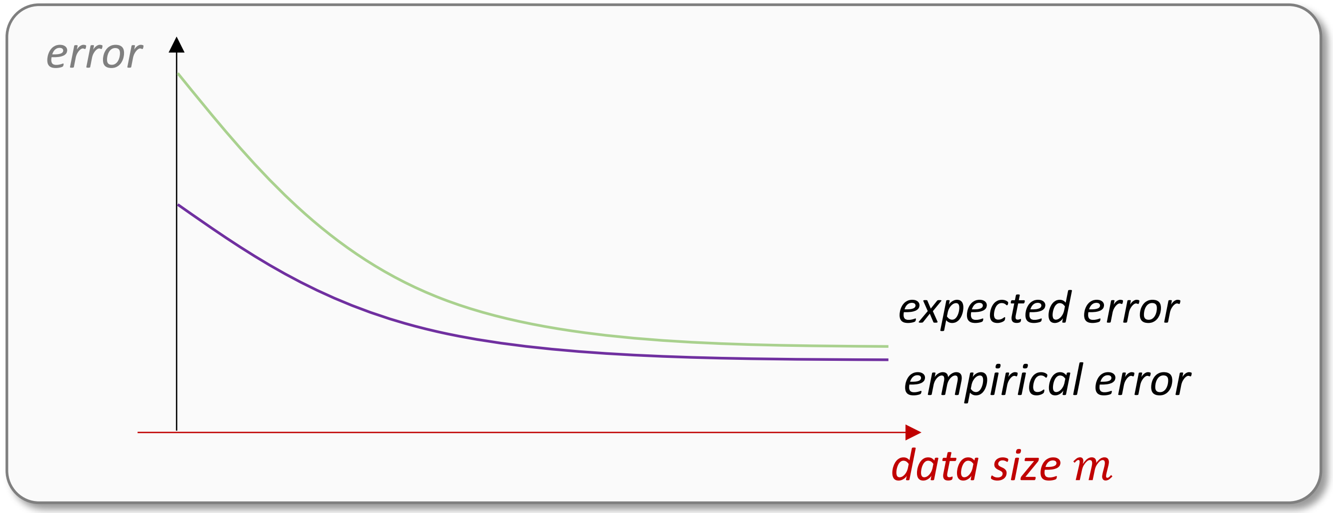
- Data and complexity are **complementary** in their effect on performance.
- **Q**: What happens if we restrict model complexity?

Statistics

- Assume we restrict model complexity (e.g., use at most $\mathrm{d} = 5$ intervals)
- How does additional data effect performance now?

Statistics

- Assume we restrict model complexity (e.g., use at most $\mathrm{d} = 5$ intervals)
- How does additional data effect performance now?



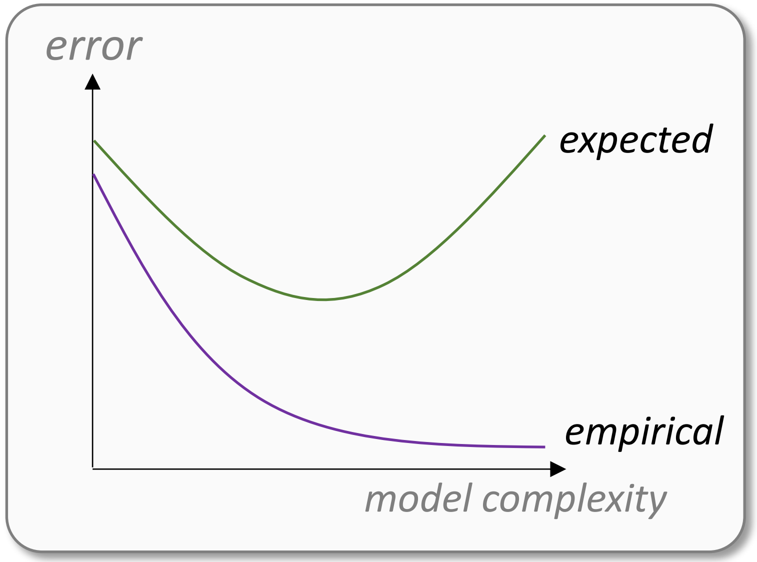
- **Q**: Will they always converge?
- **A**: Technically – yes (as $m \to \infty$), but helpful only if this happens **polynomially-fast**.
- **Important**: not all model classes are *learnable* (≈polynomial convergence)

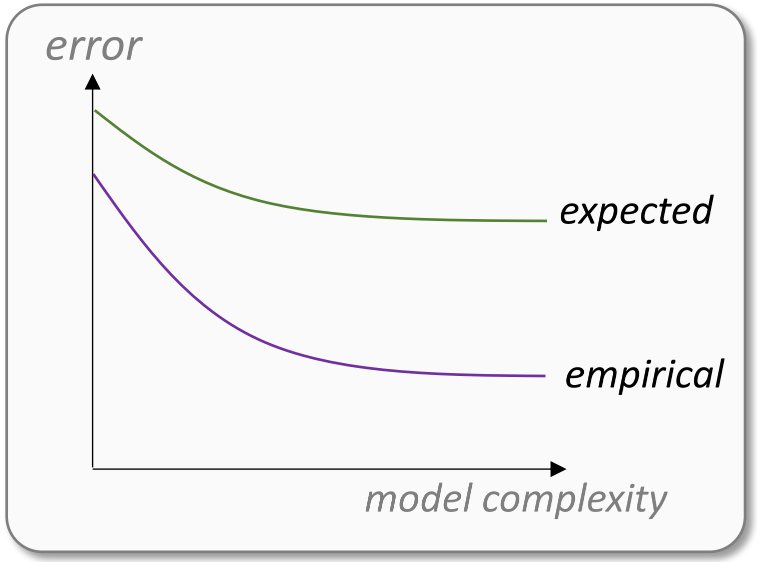
Statistics

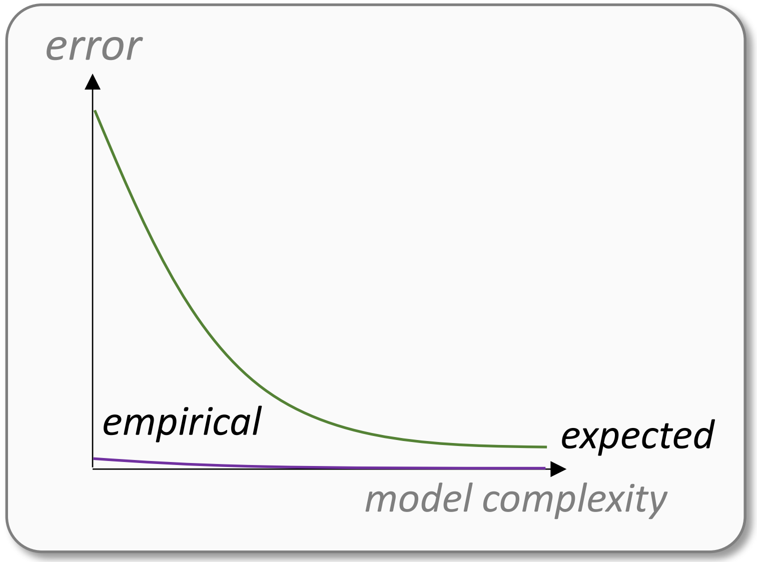
- This is all in **theory** (and on average!) – what happens in **practice**?
- Typically, we have a finite data set (fixed m), and can choose the model class (complexity).
- Three possible scenarios:

Statistics

- This is all in **theory** (and on average!) – what happens in **practice**?
- Typically, we have a finite data set (fixed m), and can choose the model class (complexity).
- Three possible scenarios:







- **Remember:** expect smooth behavior only when averaging – single datasets are *noisy*

Optimization

- **Separable**: easy (won't go into details)
- **Non-separable**: hard
- **Intuition** –ERM as discrete optimization: (fix d)
  - Our previous approaches constructed $\hat{\theta}$ from data points
  - Can think of d intervals as partition $\{x_i\}_{i=1}^{m}$ into d (consecutive) subsets
  - This means we now need to find best *subset*
  - This (plus a discrete objective) results in a *hard combinatorial problem*
- This is typical of many learning problems
- We will therefore mostly deal with *approximate** learning algorithms
- **Remember this** when you reason about the performance of $\hat{h}$

* approximate in the algorithmic sense – don't confuse this with approximation error

Modeling, revisited

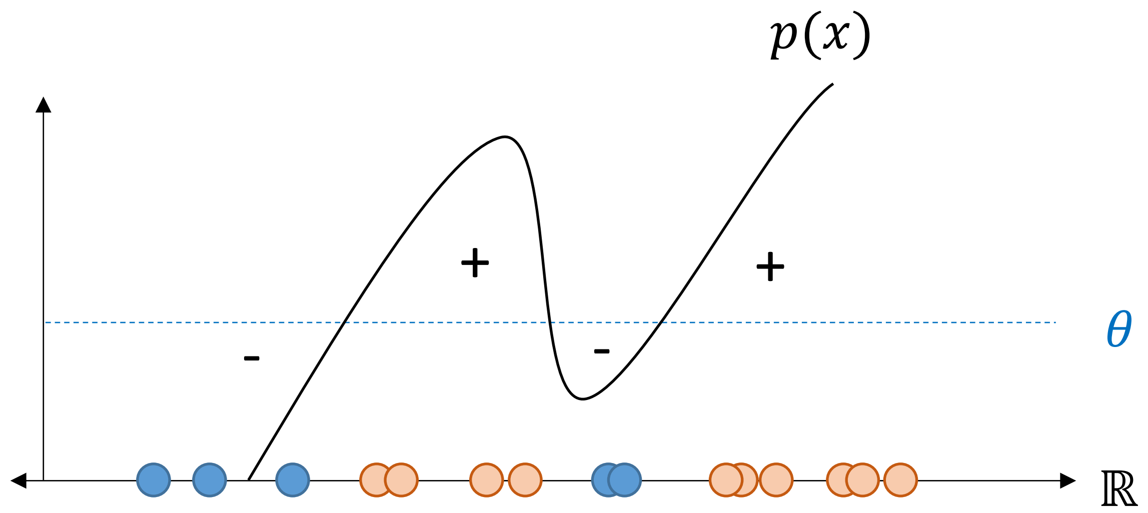
- **Recall**: ERM implementations are typically class-specific
- Models in the class need to have “structure” that can be utilized (think parameterization)
- Unions of intervals are not ideal in that sense
- **Q**: Can we perhaps tweak thresholds to “behave” like intervals?
- **A**: Yes - using… **polynomials!** (?)

Modeling, revisited

- **Recall**: ERM implementations are typically class-specific
- Models in the class need to have "structure" that can be utilized (think parameterization)
- Unions of intervals are not ideal in that sense
- **Q**: Can we perhaps tweak thresholds to "behave" like intervals?
- **A**: Yes - using… **polynomials!** (?)

Thresh. polynomial classifier:

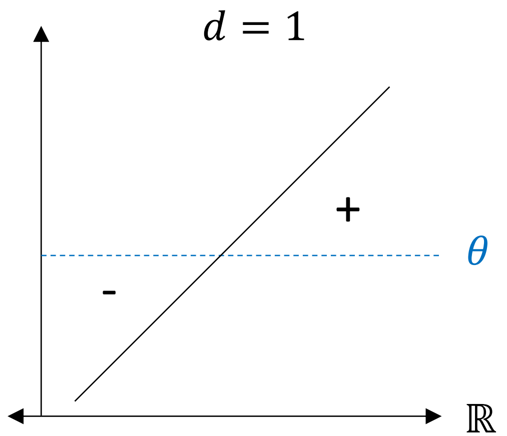
$$h_{p,\theta}(x) = \mathrm{sign}(p(x) > \theta)$$

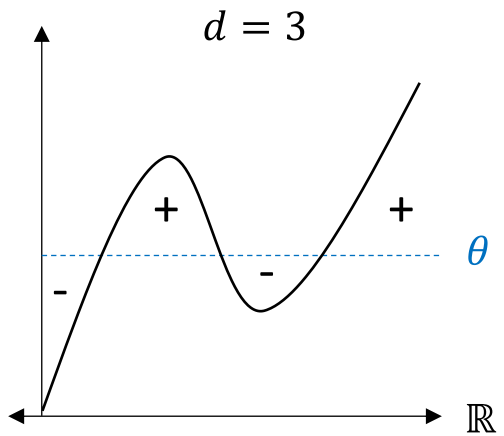


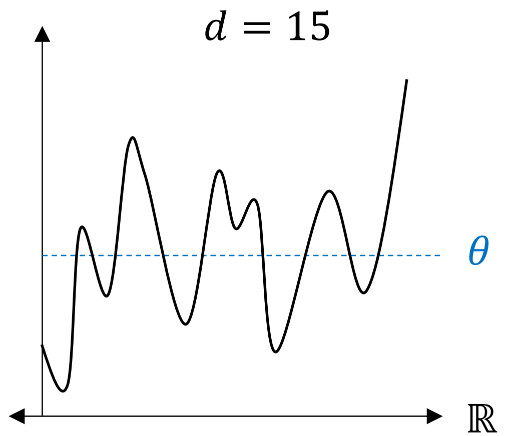
Modeling, revisited

- **Parameterization:** $p(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots w_d x^d$

- **Complexity:** d = number of intervals <-> degree of polynomial



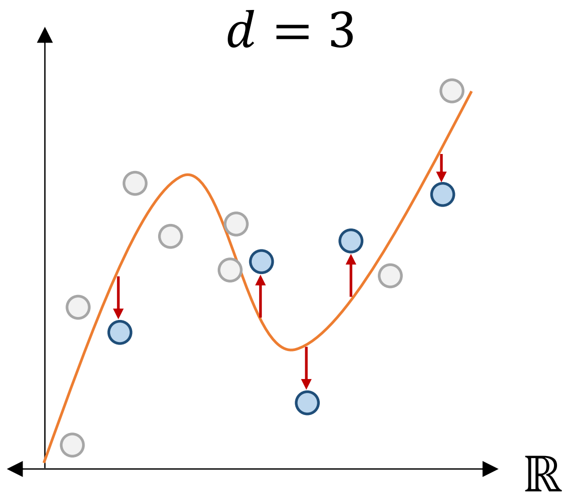




- **Polynomial thresholds are great!**
  - Generalize 1D thresholds – *intuitions and conclusions carry over*
  - **Algorithmically**: can solve (tractable) continuous optimization problem
  - **Statistically**: easy to analyze (well-studied objects)

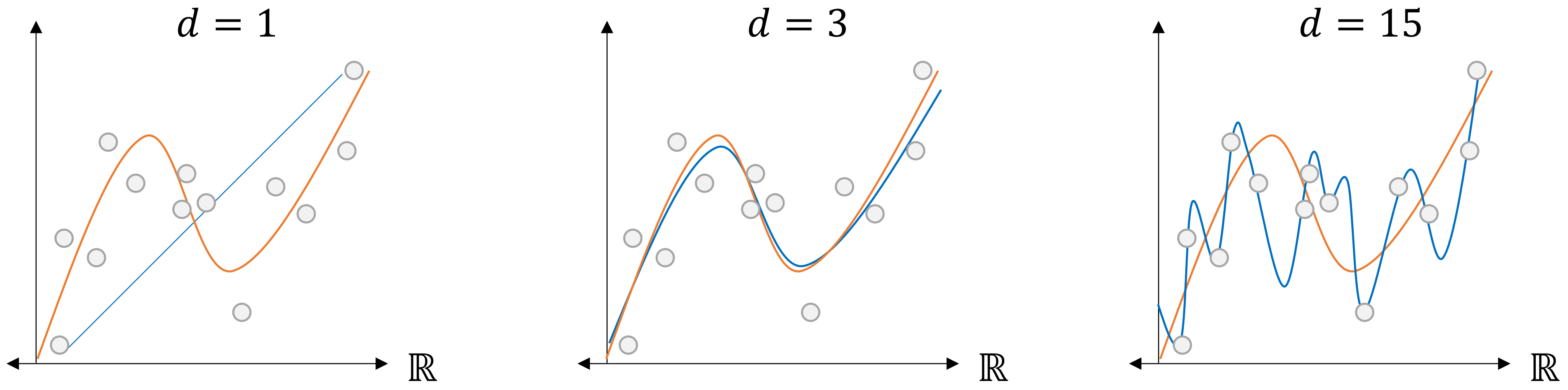
Statistics, revisited

- **Intuition**: *overfitting* happens when overly-complex models fit *noise*
- Easy to visualize for *polynomial regression* ($\mathcal{Y} = \mathbb{R}$; e.g., *least squares*)



Statistics, revisited

- **Intuition**: *overfitting* happens when overly-complex models fit *noise*
- Easy to visualize for *polynomial regression* ($\mathcal{Y} = \mathbb{R}$; e.g., *least squares*)



- (For binary classification – imagine "bit-flip" noise)

Discussion

Recap

- ERM aims to minimize *empirical error*, but we care about *expected error*
- Our key quantity of interest is **generalization**
- Seems straightforward, but in practice – many, many subtleties to consider:
    - Approximation error and estimation error
    - Overfitting and underfitting
    - The relation between H and m
    - The computational hardness of learning
    - What we can compute and what we cannot
    - How to define “good”
    - The randomness in… everything
    - …
- **Beware of potential pitfalls!**

Discussion

- Today we discussed several “types” of distributions
- In practice, we won’t know our type
- Can make assumptions, but this should be done with care
- (Remember: we *always* make *some* assumptions)
- **Alternative**: learning algorithms that are **distribution-free**
- **Pro**: guaranteed to work well for any D
- **Con**: guarantees are always relative (e.g., vs. h^*)
- (In effect, choosing H + sufficing with relative measures = making an assumption)

Next: beyond 1D data

- Consider a degree-d polynomial over reals ($z \in \mathbb{R}$):
$$p(z) = w_0 + w_1 z + w_2 z^2 + w_3 z^3 + \cdots w_d z^d$$
- Now define a *vector representation* of powers of z:
$$\boldsymbol{x} = (x_1, x_2, \dots, x_d) = (z, z^2, \dots, z^d)$$
- Notice that $p(z)$ is linear in $\boldsymbol{x}$:
$$\boxed{p(z) = b + \boldsymbol{w}^\top \boldsymbol{x}, \qquad b = w_0, \boldsymbol{w} = (w_1, \dots, w_d)}$$
- **Transformed problem**: univariate inputs / complex model class => multivariate inputs / simple model class
- **Next step**: generalize to arbitrary high-dimensional vector inputs, $\boldsymbol{x} \in \mathbb{R}^d$
- This is called ***linear classification*** – a problem we will devote much of our time to.

Next week

- **Classification** – three methods:

1. Similarity-based
2. Rule-based
3. Linear classifiers