

# MACHINE LEARNING BASICS

# MACHINE LEARNING BASICS

## AGENDA

Overfitting — what is it, how do we combat it

Maximization bias — why we need test data sets to evaluate our model

Stopping criteria for gradient descent

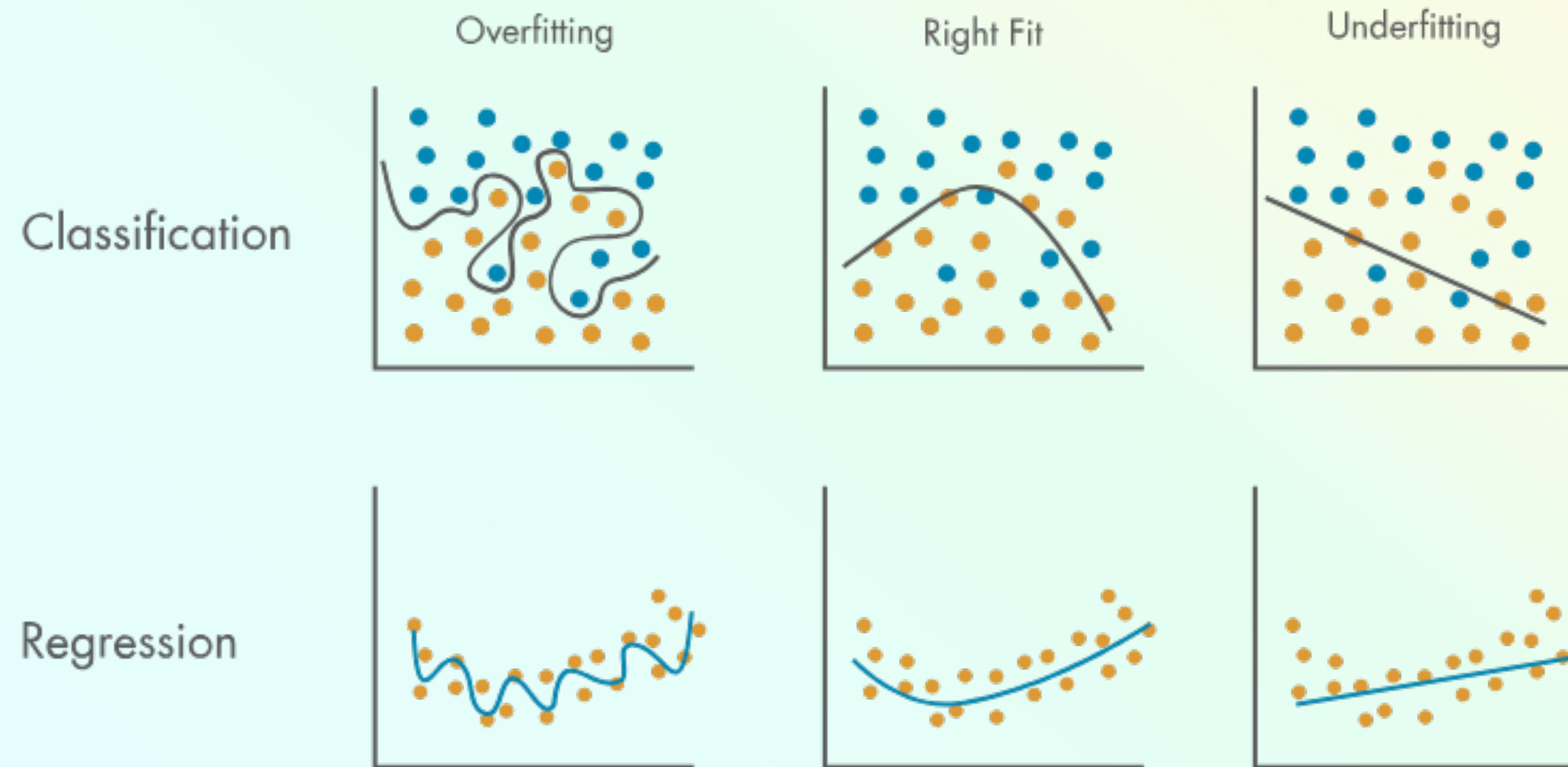
Early stopping method — reduces overfitting

Cross validation:

Select model and hyperparameters without overfitting to the data

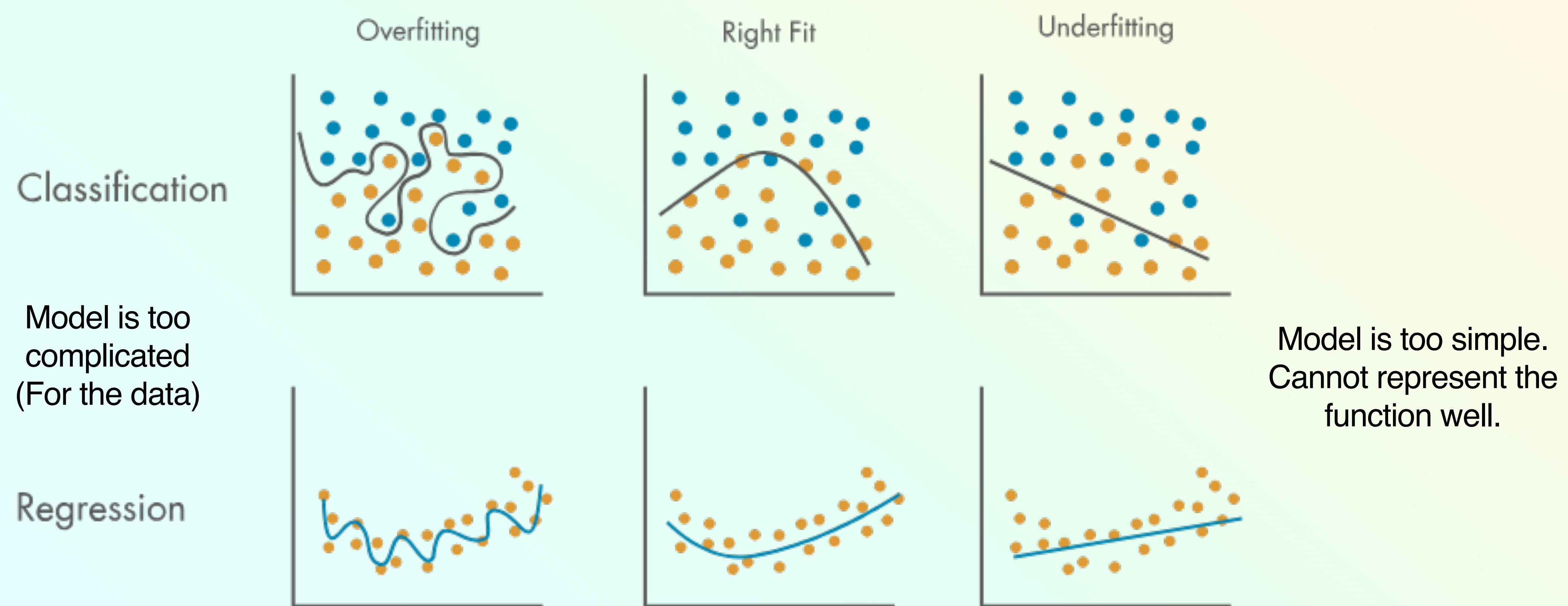
# MACHINE LEARNING BASICS

## OVERFITTING



# MACHINE LEARNING BASICS

## OVERFITTING



# MACHINE LEARNING BASICS

## OVERFITTING

Reasons for overfitting:

1. Model is too complicated
2. Model is being fit to noise in the data
3. Insufficient data to know what is noise vs not noise

# MACHINE LEARNING BASICS

## OVERFITTING

Reasons for overfitting:

1. Model is too complicated
2. Model is being fit to noise in the data
3. **Insufficient data to know what is noise vs not noise**

# MACHINE LEARNING BASICS

## OVERFITTING

$$l(w) = \mathbf{E} \left[ (f(X, w) - Y)^2 \right]$$

# MACHINE LEARNING BASICS

## OVERFITTING

$l(w) = \mathbb{E} \left[ (f(X, w) - Y)^2 \right]$  – we do not know this, nor can we compute it

What we minimize is:

$$l_D(w) = \frac{1}{m} \sum_{i=1}^m (f(x_i, w) - y_i)^2$$

# MACHINE LEARNING BASICS

## OVERFITTING - WHY IS IT BAD

If  $l_D(w) \approx l(w)$  then

$$\arg \min_w l_D(w) \approx \arg \min_w l(w)$$

(We hope, but it depends on  $f$  and  $l$ )

# MACHINE LEARNING BASICS

## OVERFITTING - WHY IS IT BAD

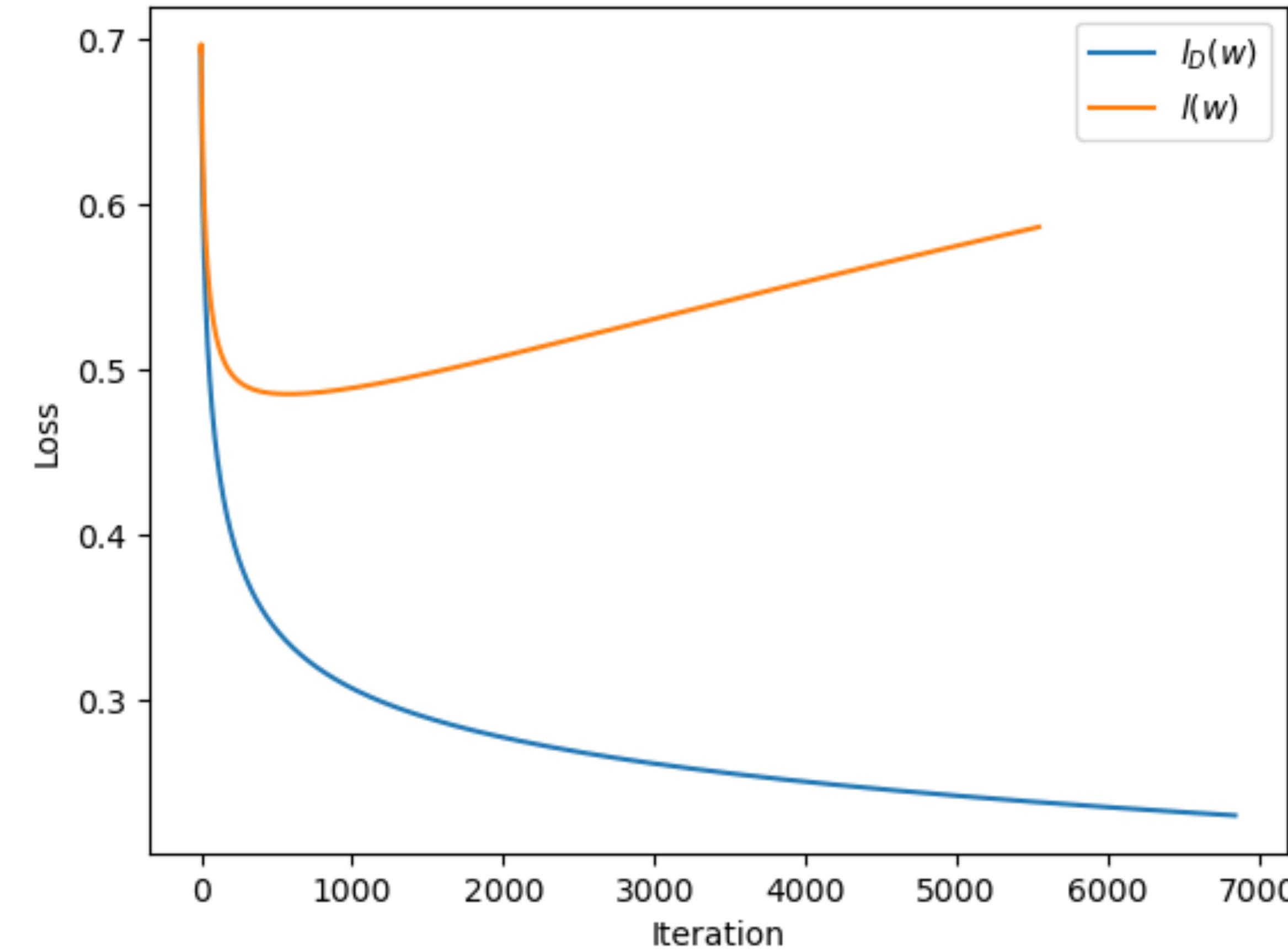
The optimal weights  $w^*$  are different for  $l(w)$  and  $l_D(w)$

Doing well on  $l_D(w)$  usually means doing well on  $l(w)$

But being optimal on  $l_D(w)$  often means being worse on  $l(w)$

# MACHINE LEARNING BASICS

## OVERFITTING - WHY IS IT BAD?



## **Class question**

# BIAS

## DEFINITION

Bias is the difference in the expectation of an estimator and the true expected value we want to estimate

$\mu = \mathbf{E}[X]$  – we want to estimate  $\mu$

$$\hat{\mu}_1 = \frac{1}{n} \sum_{i=1}^n X_i$$

$$\text{Bias of } \hat{\mu}_1: \mathbf{E}[\hat{\mu}_1] - \mu = \mathbf{E} \left[ \frac{1}{n} \sum_{i=1}^n X_i \right] - \mu = \left( \frac{1}{n} \sum_{i=1}^n \mathbf{E}[X_i] \right) - \mu = \left( \frac{1}{n} \sum_{i=1}^n \mu \right) - \mu = \mu - \mu = 0$$

$\hat{\mu}_1$  is an unbiased estimator of  $\mu$

# BIAS

## BIASED ESTIMATOR

$$\hat{\mu}_2 = \frac{1}{n} \sum_{i=1}^n (X_i + C_i)$$

Bias of  $\hat{\mu}_2$ :

$$\mathbf{E}[\hat{\mu}_2] - \mu = \mathbf{E} \left[ \frac{1}{n} \sum_{i=1}^n (X_i + C_i) \right] - \mu = \mu + \frac{1}{n} \sum_{i=1}^n \mathbf{E}[C_i] - \mu = \frac{1}{n} \sum_{i=1}^n \mathbf{E}[C_i]$$

$\hat{\mu}_2$  is a biased estimator if  $\mathbf{E}[C_i] \neq 0$

# MAXIMIZATION BIAS

## BIAS DEFINITION

For stochastic gradient descent we want unbaised estimates of  $l(w)$  and  $\nabla l(w)$

$$\mathbf{E}[l_i(w)] - l(w) = 0$$

$$\mathbf{E}[\nabla l_i(w)] - \nabla l(w) = 0$$

# MAXIMIZATION BIAS

## BIAS DEFINITION

For stochastic gradient descent we want unbaised estimates of  $l(w)$  and  $\nabla l(w)$

$$\mathbf{E}[l_i(w)] - l(w) = 0$$

$$\mathbf{E}[\nabla l_i(w)] - \nabla l(w) = 0$$

Also want to be able to select the best weights.

We need to be able to estimate  $l(w)$  without bias or we might choose the wrong weights

## **Class question**

# MAXIMIZATION BIAS

## EXAMPLE

Consider playing a game where you get to roll dice and want to have the highest roll on average. You own two old dice that have been beat up and may not be unbiased any more. To optimize your advantage in the game you want to roll with your best dice.

You roll each dice  $m$  times and compute the average roll score for each one.

$$\bar{X}_m = \frac{1}{m} \sum_{i=1}^m X_i \quad \bar{Y}_m = \frac{1}{m} \sum_{i=1}^m Y_i$$

You choose  $\arg \max \{\bar{X}_m, \bar{Y}_m\}$  and declare that the advantage is  $\max \{\bar{X}_m - \bar{Y}_m, \bar{Y}_m - \bar{X}_m\}$

# MAXIMIZATION BIAS

## EXAMPLE

Consider playing a game where you get to roll dice and want to have the highest roll on average. You own two old dice that have been beat up and may not be unbiased any more. To optimize your advantage in the game you want to roll with your best dice.

You roll each dice  $m$  times and compute the average roll score for each one.

$$\bar{X}_m = \frac{1}{m} \sum_{i=1}^m X_i \quad \bar{Y}_m = \frac{1}{m} \sum_{i=1}^m Y_i$$

You choose  $\arg \max \{\bar{X}_m, \bar{Y}_m\}$  and declare that the roll average is  $\max \{\bar{X}_m, \bar{Y}_m\}$

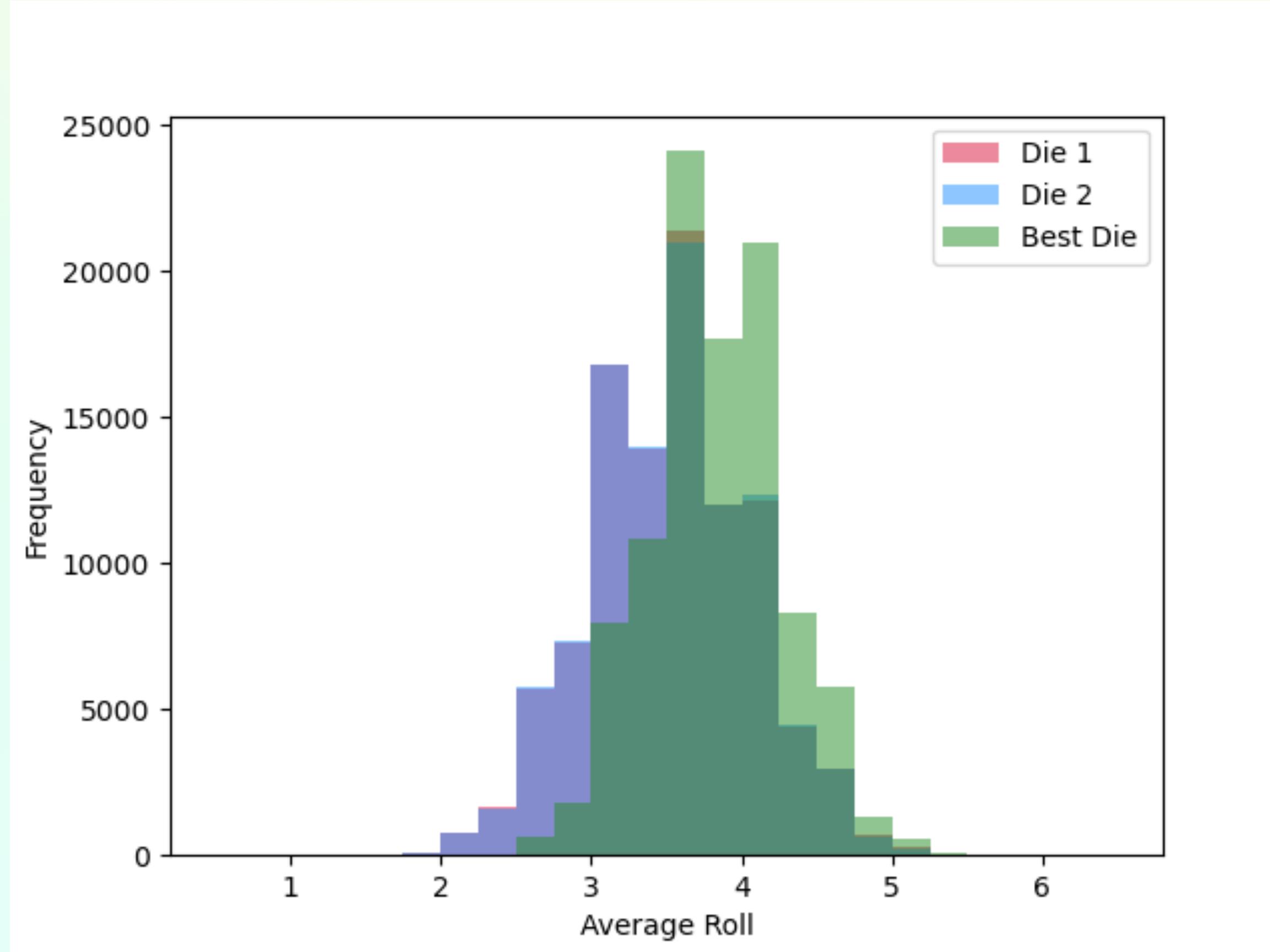
Do we choose the better dice? Is our assessment of the average roll of the chosen dice fair?

# MAXIMIZATION BIAS

## EXAMPLE

Roll each die 10 times

Run this experiment with two identical dies  
100,000 times.



# MAXIMIZATION BIAS

## EXAMPLE

Roll each die 10 times

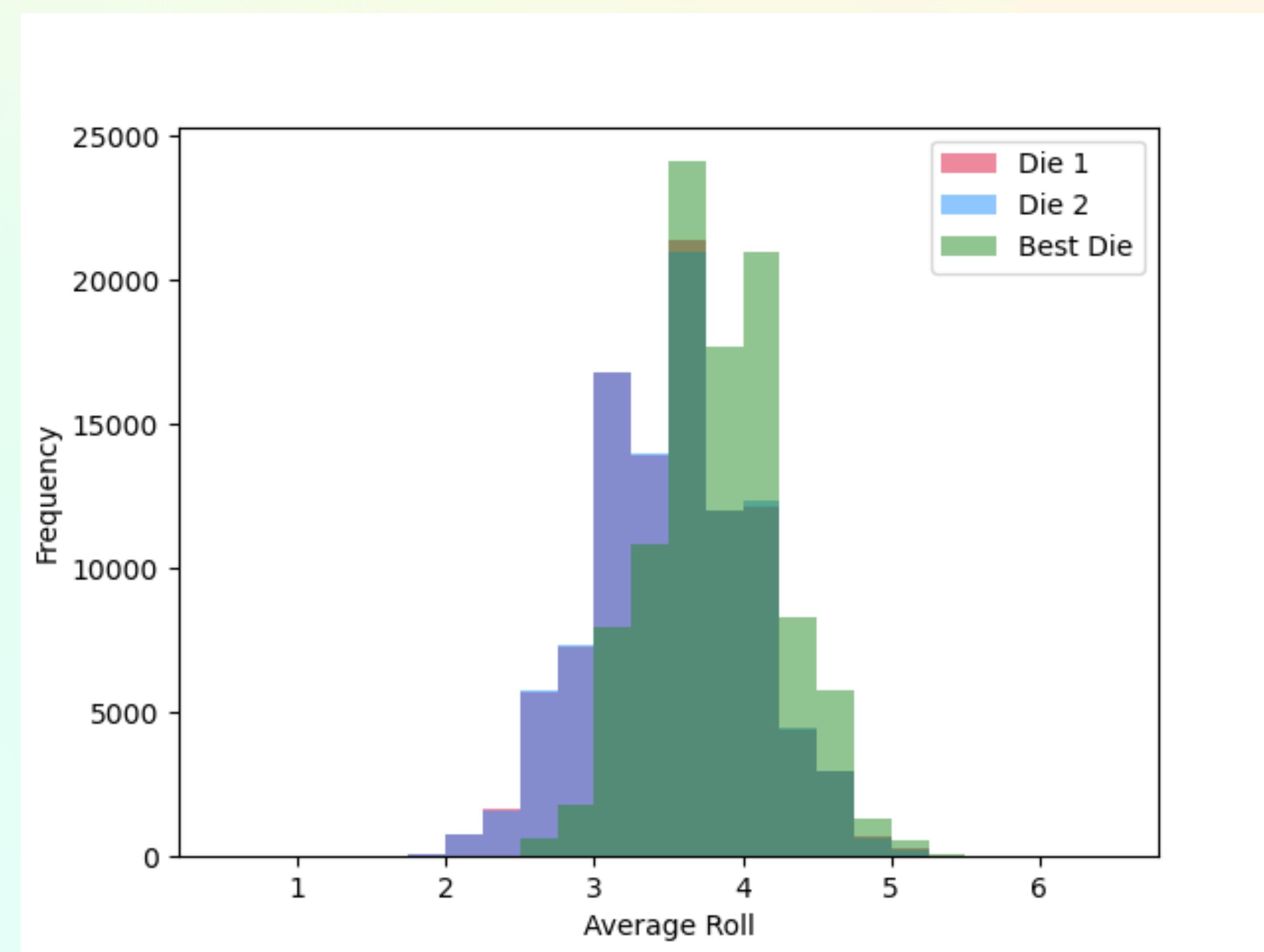
Run this experiment with two identical dies 100,000 times.

Dies 1 and 2 have same distribution

The best die has a different distribution

Its mean is higher. This is bias!

$$E[\bar{X}_m] = E[\bar{Y}_m] \neq E[\max\{\bar{X}_m, \bar{Y}_m\}]$$



# MAXIMIZATION BIAS

IN ML

Before getting the data  $D$  choose some weights  $w^1, w^2$ , etc

We can then evaluate  $w^1, w^2, \dots$  using data  $D$ ,

$l_D(w)$  is the empirical loss using data  $D$

$$\mathbf{E} [l_D(w^1)] = l(w^1), \mathbf{E} [l_D(w^2)] = l(w^2)$$

We can then choose the best  $w$  of this pre chosen set with unbiased evaluations.

However, we cannot say how good these chosen weights are because we already used  $D$ .

# MAXIMIZATION BIAS

IN ML

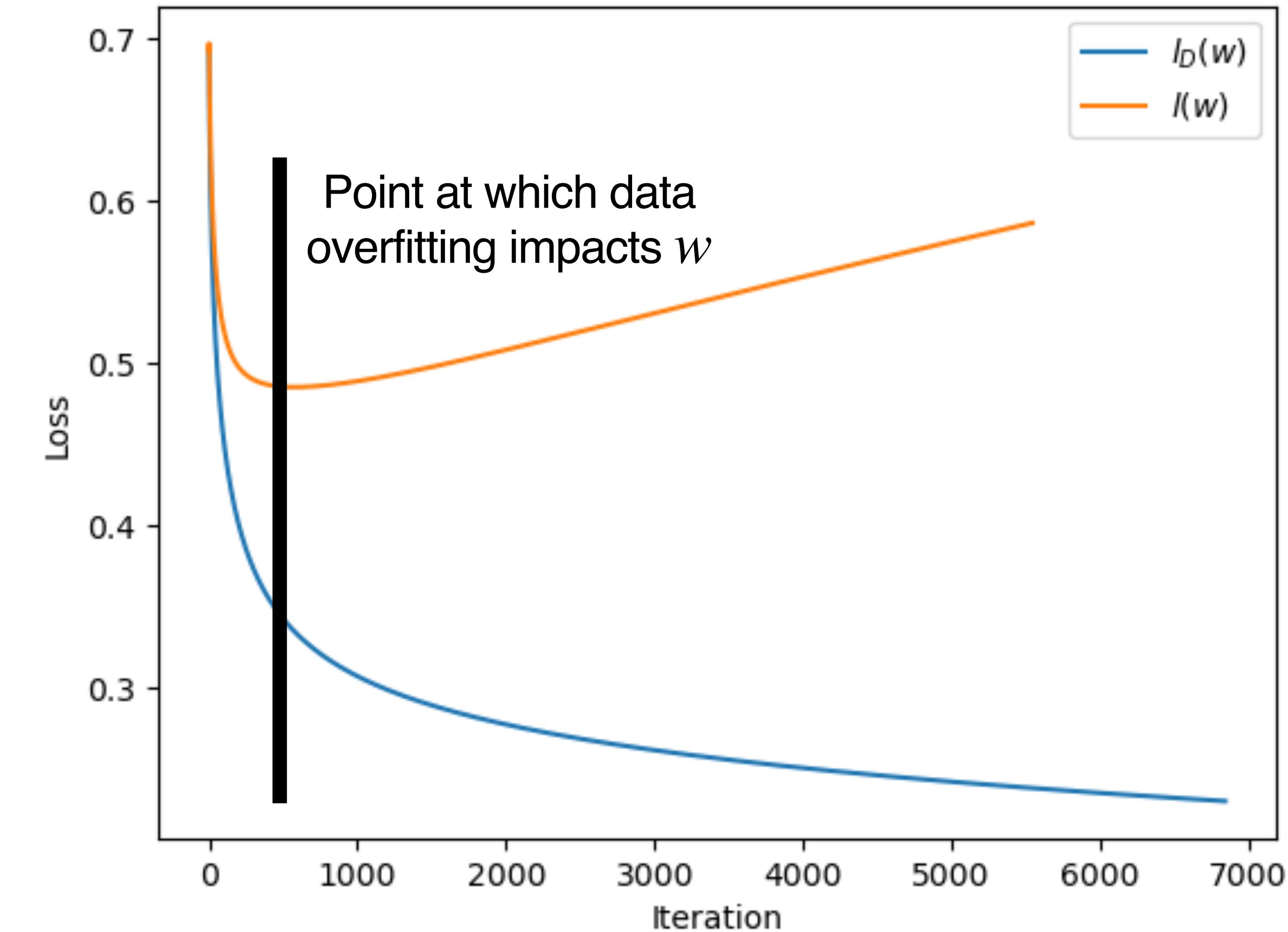
When we optimize  $w$  on  $D$ , we are choosing the  $w$  that does the best on  $D$ .

This biases the  $\min_w l_D(w)$  to be less than  $l(w)$ .

Need new data  $D'$  then we can evaluate  $l_{D'}\left(\arg \min_w l_D(w)\right)$

# MAXIMIZATION BIAS

IN ML



# TEST SETS

## UNBIASED EVALUATIONS

Having both  $D$  to train on and  $D'$  to evaluate the model on is necessary to get an unbaised assement of our model.

Solution: Create  $D_{train}$  and  $D_{test}$  from a single data set  $D$

$D_{train}$  contains some proportion  $p \in (0,1)$  of the data and is used to train the model

$D_{test}$  contains some proportion  $1 - p$  of the data and is used to provide an unbaised evaluation of the model

# TEST SETS

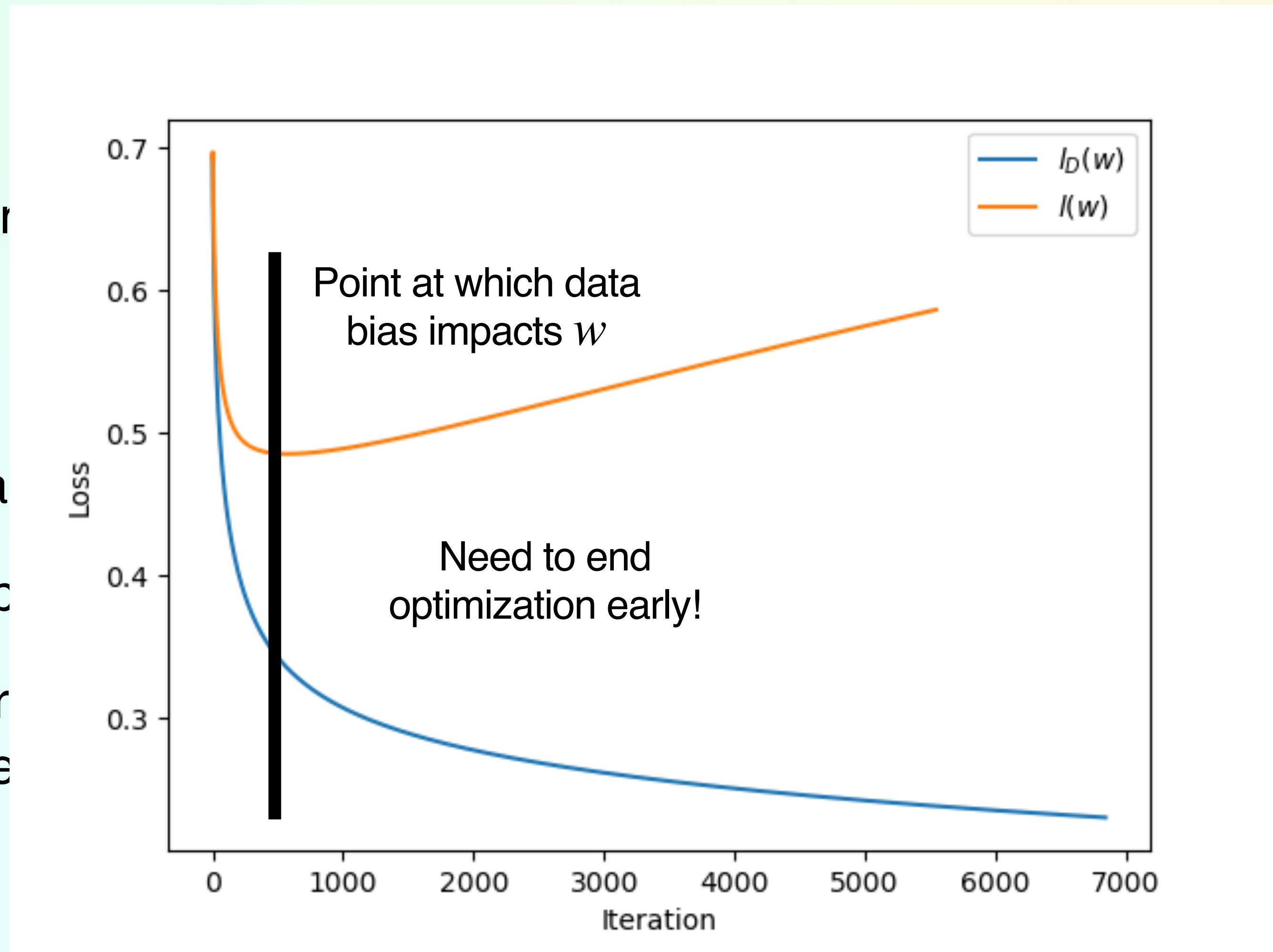
## UNBIASED EVALUATIONS

Having both  $D$  to train on and  $I(w)$  to provide an unbiased assessment of our model.

Solution: Create  $D_{train}$  and  $D_{test}$

$D_{train}$  contains some points from  $D$

$D_{test}$  contains some points from  $D$  not in  $D_{train}$  to provide an unbiased evaluation of the model



unbiased  
model  
aised

# EARLY STOPPING

## USING VALIDATION DATA

Need to decide when to stop — Cannot use  $|l_{D_{train}}(w^{k+1}) - l_{D_{train}}(w^k)| < \epsilon$

Split  $D$  into three data sets  $D_{train}, D_{val}, D_{test}$

$D_{train}$  — search for the best weights (cannot evaluate if overtraining without bias)

$D_{val}$  — Validation data used to evaluate if over training and we should stop.

$D_{test}$  — used to evaluate the final model, i.e., say how accurate it is

# EARLY STOPPING

## USING VALIDATION DATA

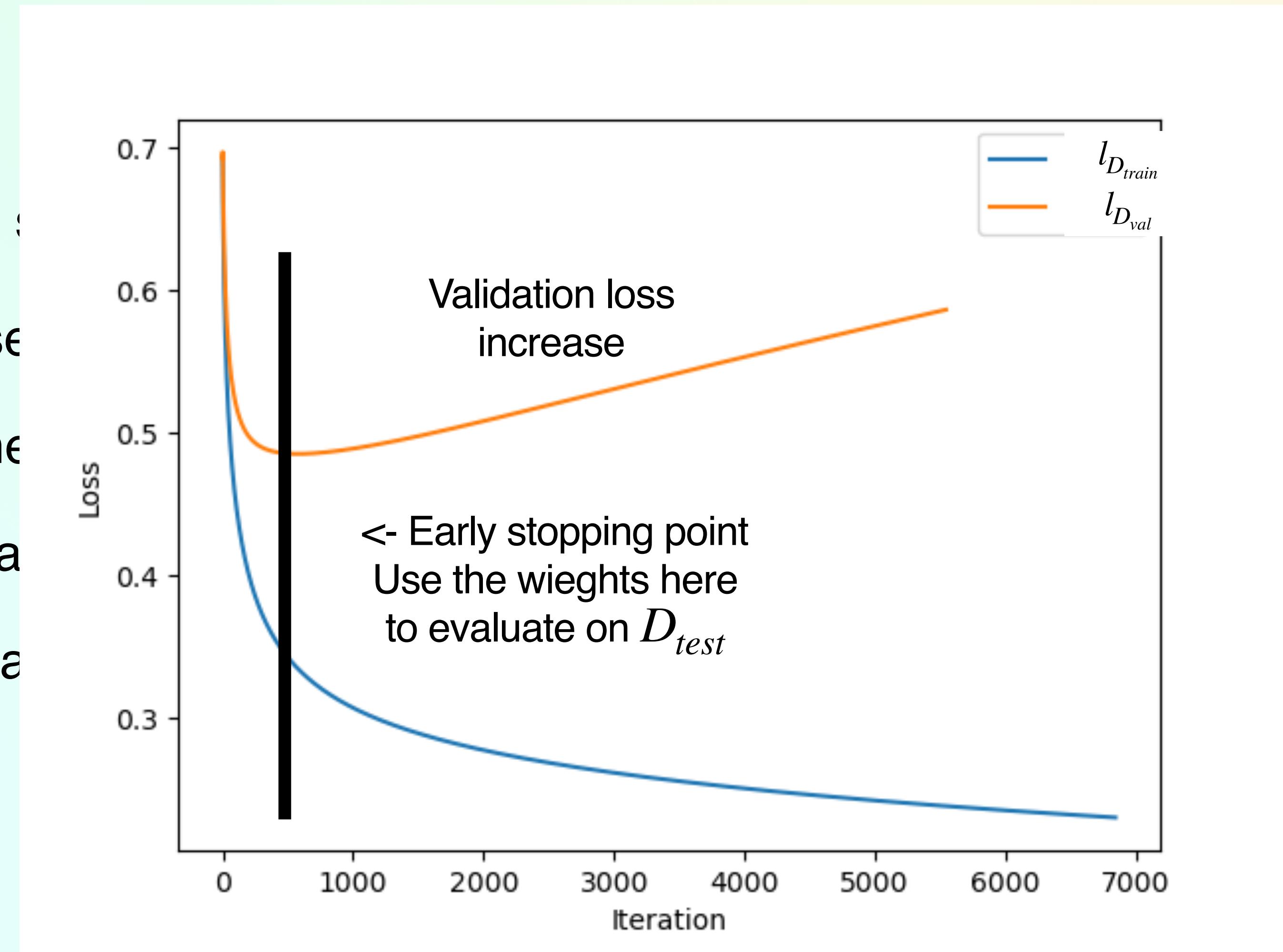
Need to decide when to stop training

Split  $D$  into three data sets

$D_{train}$  — search for the best model

$D_{val}$  — Validation data

$D_{test}$  — used to evaluate the final model



# EARLY STOPPING

## USING VALIDATION DATA

```
gradient_descent_earlystopping( $D, w_0, \rho_{test}, \rho_{val}, \epsilon, \eta$ )
```

```
 $D_{train}, D_{val}, D_{test} \leftarrow \text{split\_data}(D, \rho_{test}, \rho_{val})$ 
```

```
 $w = w_0$ 
```

```
 $w_{best} = w$ 
```

Loop:

```
 $w \leftarrow w - \eta \nabla l_{D_{train}}(w)$ 
```

```
If  $l_{D_{val}}(w) < l_{D_{val}}(w_{best})$ 
```

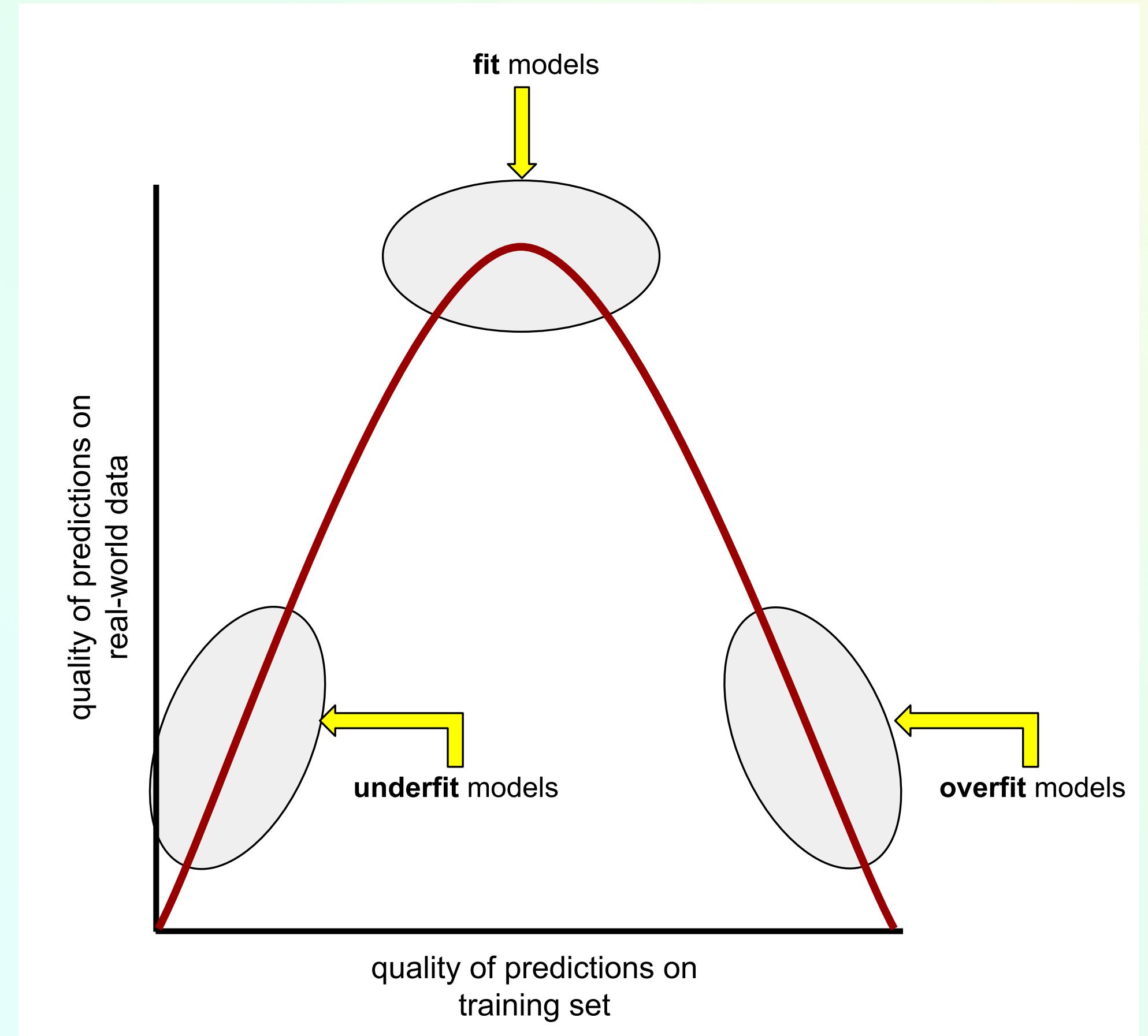
```
 $w_{best} = w$ 
```

```
If  $l_{D_{val}}(w) > l_{D_{val}}(w_{best}) + \epsilon$  // check if validation loss is sufficiently increasing
```

Return  $w_{best}$

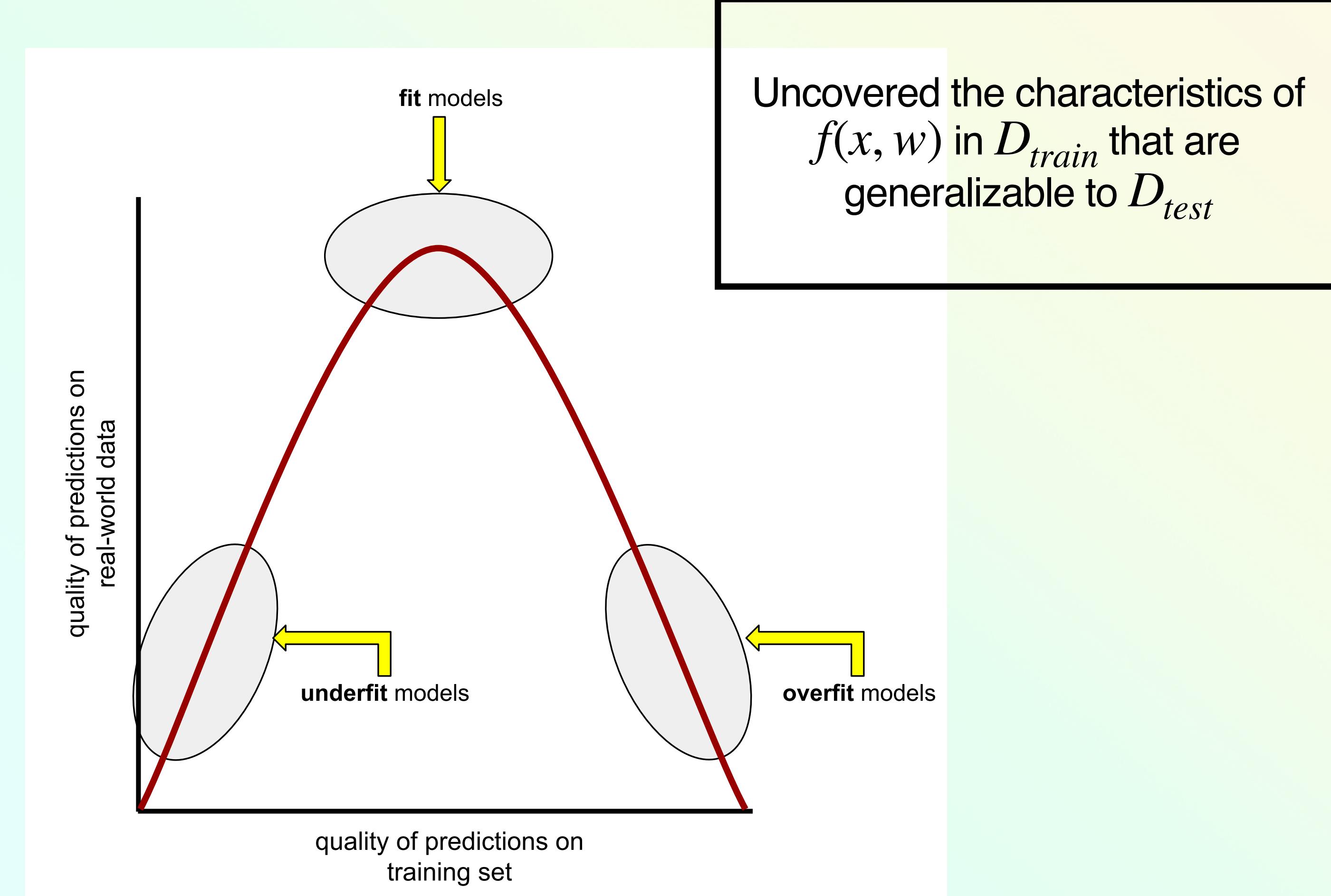
# DEFINING OVERFITTING

BEING PRECISE :(



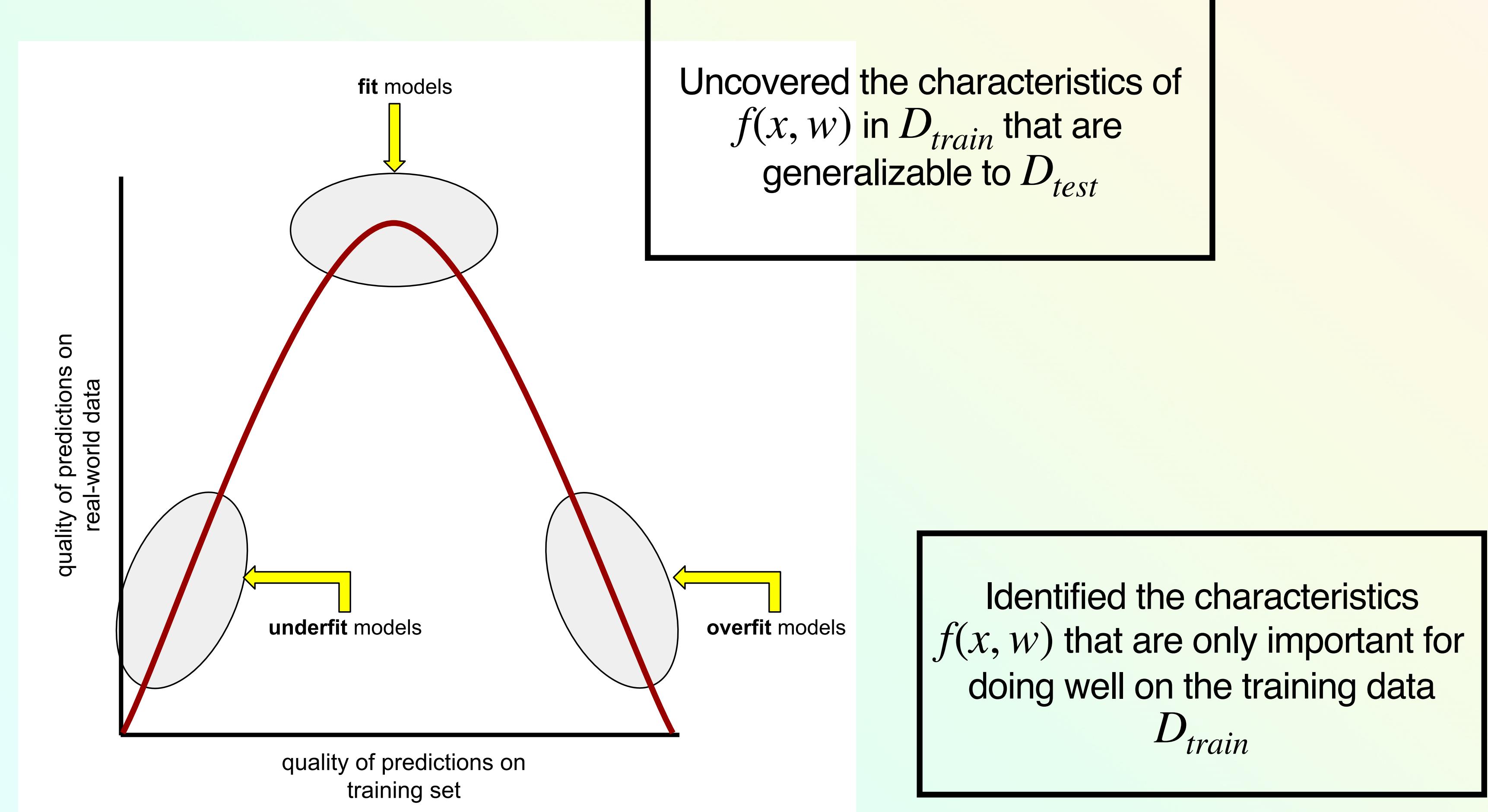
# DEFINING OVERFITTING

BEING PRECISE :(



# DEFINING OVERFITTING

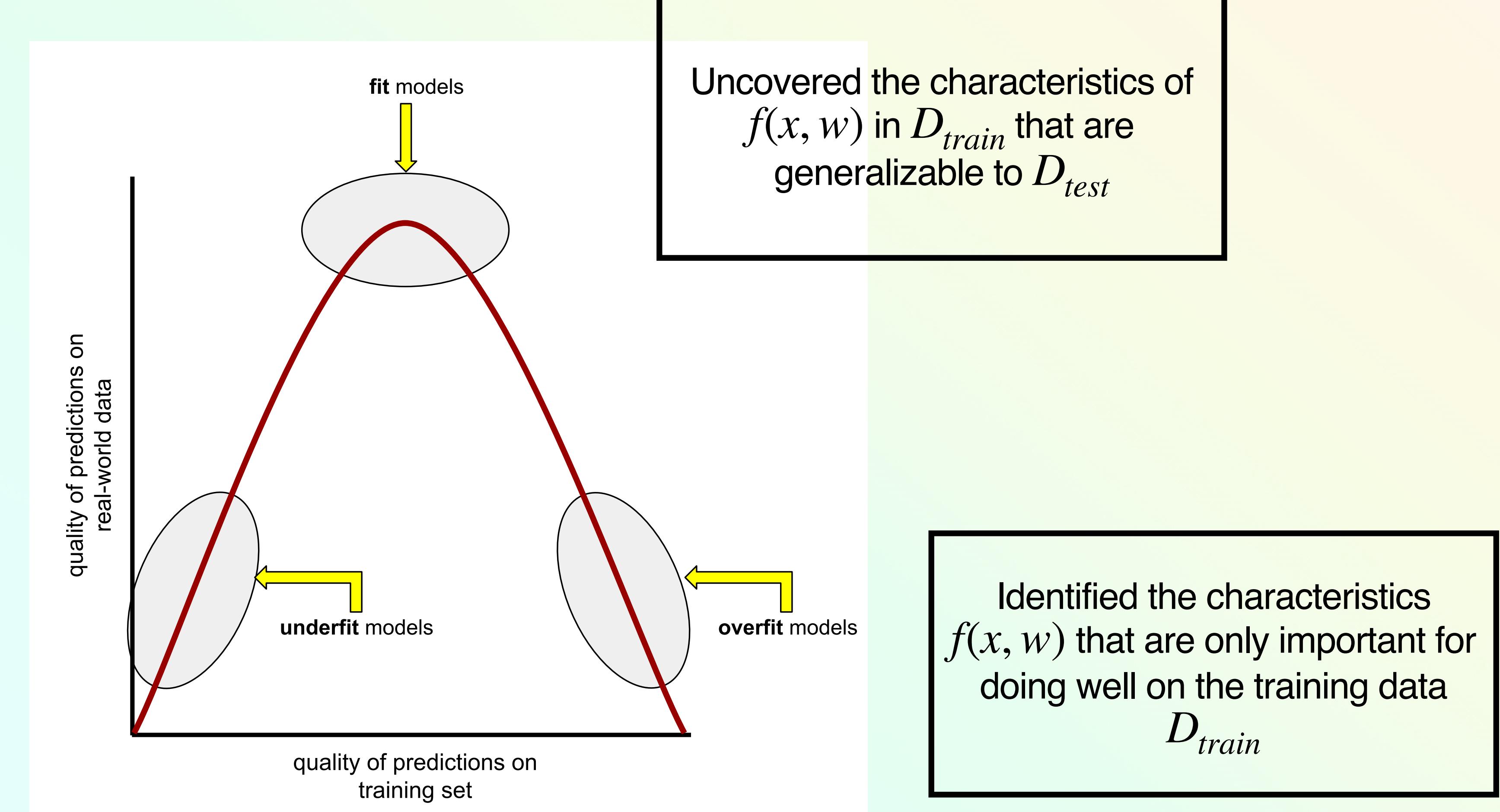
BEING PRECISE :(



# DEFINING OVERFITTING

BEING PRECISE :(

Did not uncover characteristics of  $f(x, w)$  that are present in  $D_{train}$  and did not even able to model  $D_{train}$  well



# OVER/UNDER FITTING

## REASONS

### Overfitting:

- $f$  models noise present in  $D_{train}$
- $f$  is too complex for the data
  - Number of data points < model parameters
- Ran gradient descent too long on  $D_{train}$

$$\arg \min_w l_{D_{train}}(w) \neq \arg \min_w l(w)$$

### Underfitting:

- The model is too simple
  - Ex. Fitting a line to a quadratic
- Can still overfit a line to  $D_{train}$ 
  - (Overfit to data vs. overfit of model class)
- Not extracting enough information out of  $D_{train}$ 
  - Didn't run gradient descent long enough

## **Class question**

# BIAS-VARIANCE

DECOMPOSITION OF ERROR

$$Y = f_*(X) + \xi$$

$$\mathbf{E}[\xi] = 0$$

$f_D$  is approximation produced by an algorithm on a data set  $D$

$$\mathbf{E} \left[ (Y - f_D(x_0))^2 \mid X = x_0 \right] = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$

# BIAS-VARIANCE

## DECOMPOSITION OF ERROR

$$\begin{aligned} (Y - f_D(x_0))^2 &= (Y - f_*(x_0) + f_*(x_0) - f_D(x_0))^2 \\ &= \left( [Y - f_*(x_0)] + [f_*(x_0) - f_D(x_0)] \right)^2 \\ &= (Y - f_*(x_0))^2 + 2 [Y - f_*(x_0)] [f_*(x_0) - f_D(x_0)] + (f_*(x_0) - f_D(x_0))^2 \end{aligned}$$

# BIAS-VARIANCE

DECOMPOSITION OF ERROR

$$\begin{aligned} \mathbf{E} \left[ (Y - f_D(x_0))^2 \mid X = x_0 \right] &= \mathbf{E} \left[ (Y - f_*(x_0))^2 \mid X = x_0 \right] \\ &\quad + 2\mathbf{E} \left[ (Y - f_*(x_0)) \mid X = x_0 \right] \mathbf{E} \left[ (f_*(x_0) - f_D(x_0)) \mid X = x_0 \right] \\ &\quad + \mathbf{E} \left[ (f_*(x_0) - f_D(x_0))^2 \mid X = x_0 \right] \end{aligned}$$

# BIAS-VARIANCE

DECOMPOSITION OF ERROR

$$\mathbb{E} \left[ (Y - f_D(x_0))^2 \mid X = x_0 \right] = \underbrace{\mathbb{E} \left[ (Y - f_*(x_0))^2 \mid X = x_0 \right]}_{\text{Irreducible Error}} + \mathbb{E} \left[ (f_*(x_0) - f_D(x_0))^2 \mid X = x_0 \right]$$

# BIAS-VARIANCE

## DECOMPOSITION OF ERROR

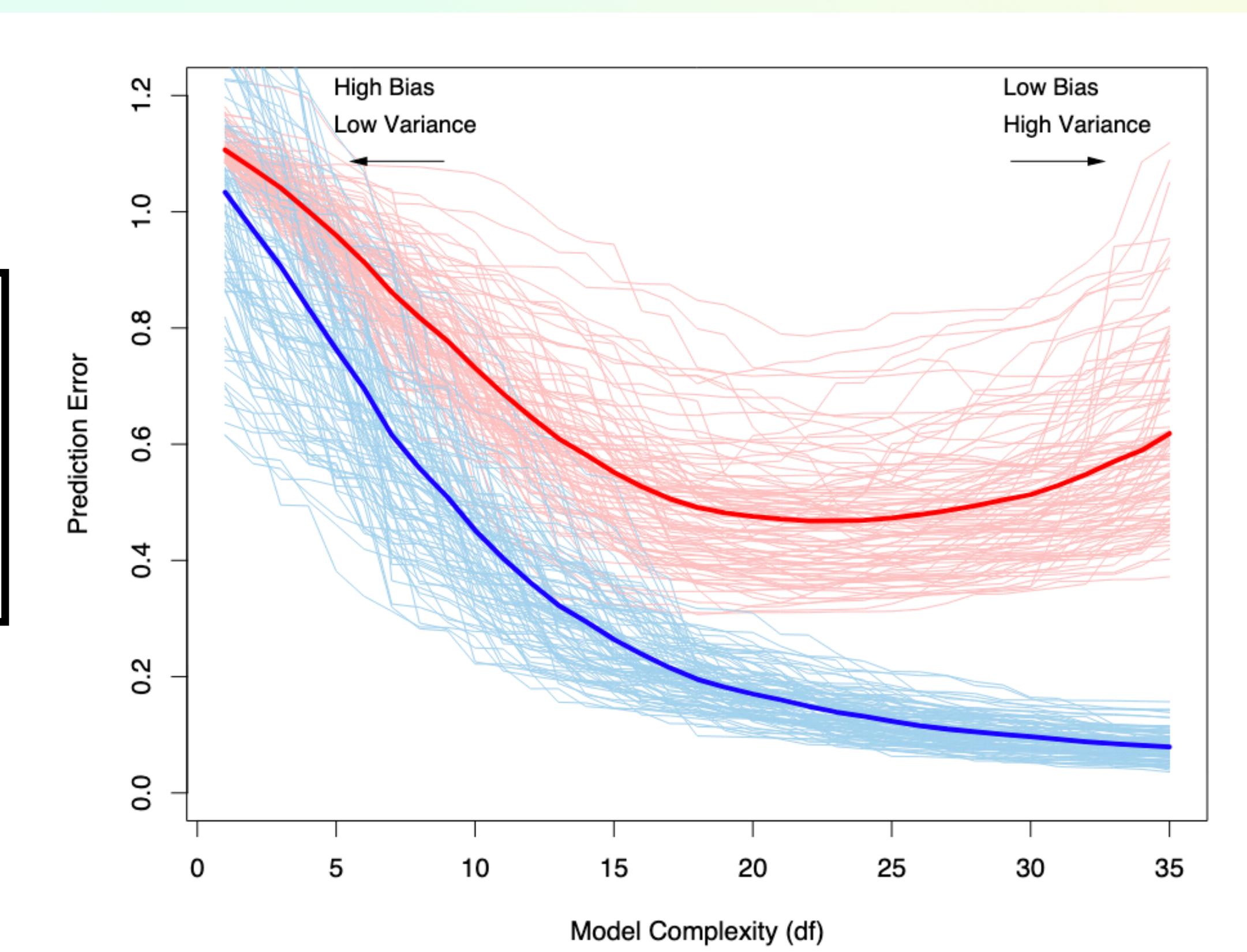
$$\mathbf{E}f \doteq \mathbf{E} [f_D]$$

$$\begin{aligned}\mathbf{E} \left[ (f_*(x_0) - f_D(x_0))^2 \middle| X = x_0 \right] &= \mathbf{E} \left[ (f_*(x_0) - \mathbf{E}f(x_0) + \mathbf{E}f(x_0) - f_D(x_0))^2 \right] \\ &= (f_*(x_0) - \mathbf{E}f(x_0))^2 + 2(f_*(x_0) - \mathbf{E}f(x_0)) \mathbf{E} [\mathbf{E}f(x_0) - f_D(x_0)] + \mathbf{E} [(\mathbf{E}f(x_0) - f_D(x_0))^2] \\ &= \underbrace{(f_*(x_0) - \mathbf{E}f(x_0))^2}_{=\text{Bias}(f)_{x_0}^2} + \underbrace{\mathbf{E} [(\mathbf{E}f(x_0) - f_D(x_0))^2]}_{=\text{Var}(f)_{x_0}}\end{aligned}$$

# BIAS-VARIANCE

## DECOMPOSITION OF ERROR

High bias<sup>2</sup> — the expected model has a large error



High Variance — The model greatly changes its output for different training data sets.

# BIAS

## BREAKING DOWN BIAS FOR LINEAR MODELS

$$w_* \in \arg \min_w \mathbf{E} \left[ (f_*(X) - f(X, w))^2 \right]$$
$$\mathbf{E} [f_*(X) - \mathbf{E} f(X, w)]^2 = \underbrace{\mathbf{E} [f_*(X) - f(X, w_*)]^2}_{\text{Avg}[Model\ Bias]^2} + \underbrace{\mathbf{E} [f(X, w_*) - f(X, w)]^2}_{\text{Avg}[Estimation\ Bias]^2}$$

Model Bias – limits on what the model can represent

Estimation Bias – error that the optimization algorithm has in finding the optimal parameters

## **Class question**

# NEXT CLASS

Next Class —