

Bias-Variance Tradeoff

Natasha Jaques



Optimal Prediction

True distribution
don't have access

Goal: Predict $Y \in \mathbb{R}$ given $X \in \mathbb{R}^d$ if $(X, Y) \sim P_{XY}$

Find function η that minimizes

// power rule

$$\mathbb{E}_{XY} [(Y - \eta(X))^2] = \mathbb{E}_X \left[\mathbb{E}_{Y|X} [(Y - \eta(x))^2 | X = x] \right]$$

(Hint: for any x , $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X} [(Y - c_x)^2 | X = x]$)

$$\begin{aligned} & \frac{d}{dc} \mathbb{E}_{Y|X} [(Y - c)^2 | X = x] \\ &= \mathbb{E}_{Y|X} [2(Y - c)(-1) | X = x] \\ &= -2 \left[\mathbb{E}_{Y|X} [Y | X = x] - c \right] = 0 \\ & \underline{c_x} = \mathbb{E}_{Y|X} [Y | X = x] = \eta(x) \end{aligned}$$

Optimal Prediction

Goal: Predict $Y \in \mathbb{R}$ **given** $X \in \mathbb{R}^d$ **if** $(X, Y) \sim P_{XY}$

Find function η that minimizes

$$\mathbb{E}_{XY} [(Y - \eta(X))^2] = \mathbb{E}_X \left[\mathbb{E}_{Y|X} [(Y - \eta(x))^2 | X = x] \right]$$

(Hint: for any x , $\eta(x) = c_x$ where c_x minimizes $\mathbb{E}_{Y|X} [(Y - c_x)^2 | X = x]$)

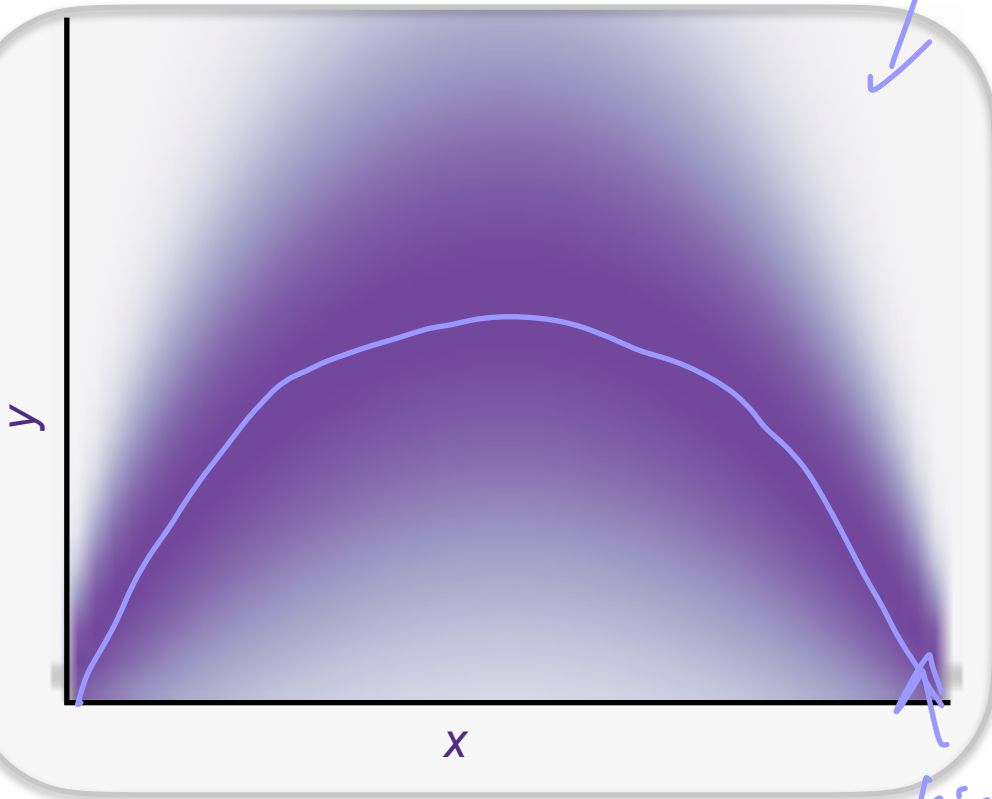
$$\begin{aligned} 0 &= \frac{d}{dc_x} \mathbb{E}_{Y|X} [(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X} \left[\frac{d}{dc_x} (Y - c_x)^2 | X = x \right] \\ &= \mathbb{E}_{Y|X} [-2(Y - c_x) | X = x] = -2\mathbb{E}_{Y|X} [Y | X = x] + 2c_x \end{aligned}$$

Squared Error Optimal Predictor: $\eta(x) = \mathbb{E}_{Y|X} [Y | X = x]$

Statistical Learning

$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

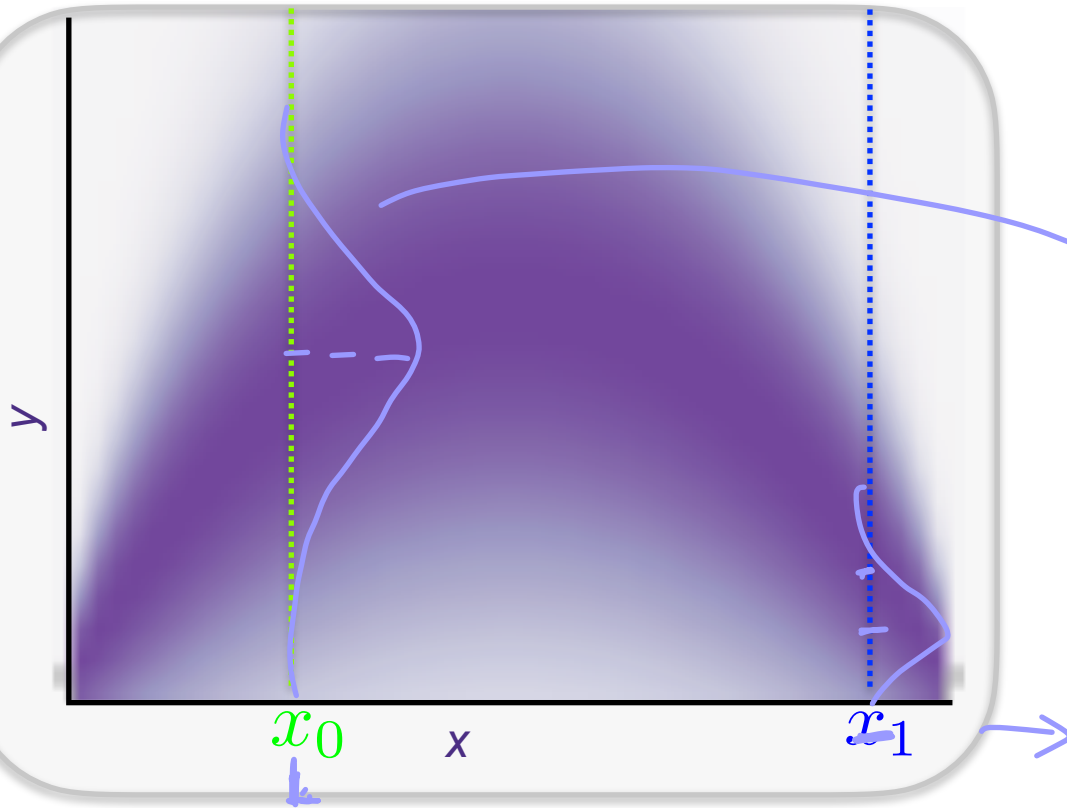
$$P_{XY}(X = x, Y = y)$$



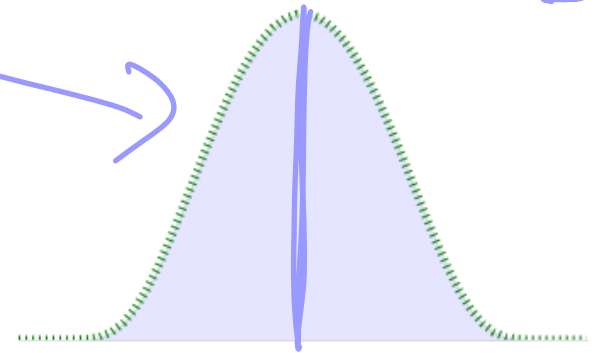
Statistical Learning

$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

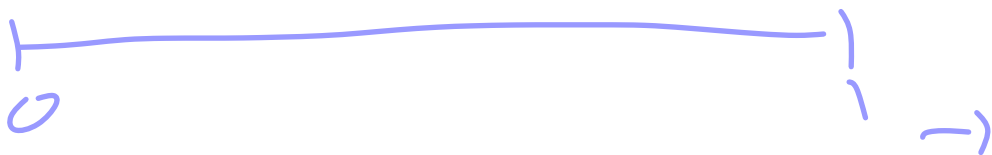
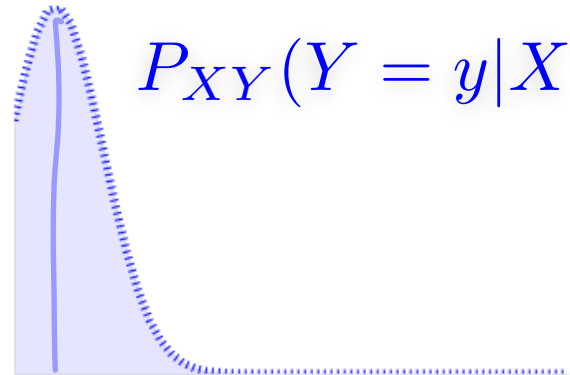
$$P_{XY}(X = x, Y = y)$$



$$P_{XY}(Y = y | X = x_0)$$



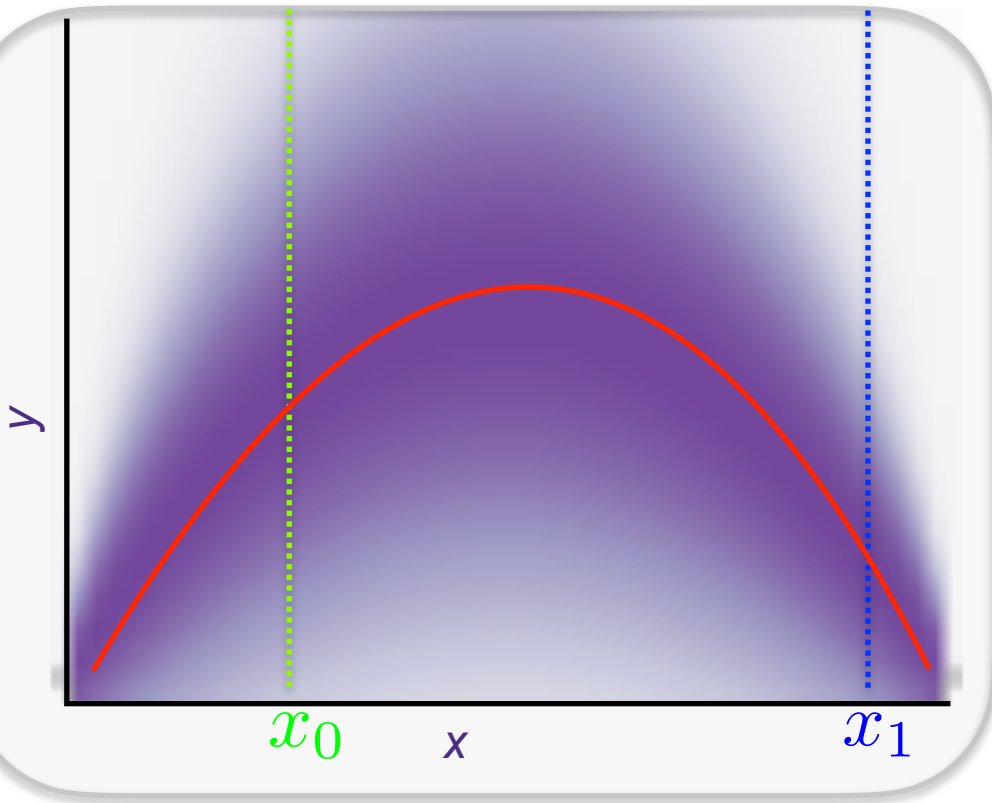
$$P_{XY}(Y = y | X = x_1)$$



Statistical Learning

$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

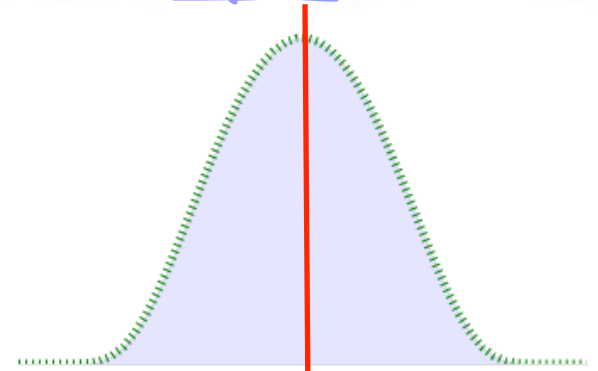
$$P_{XY}(X = x, Y = y)$$



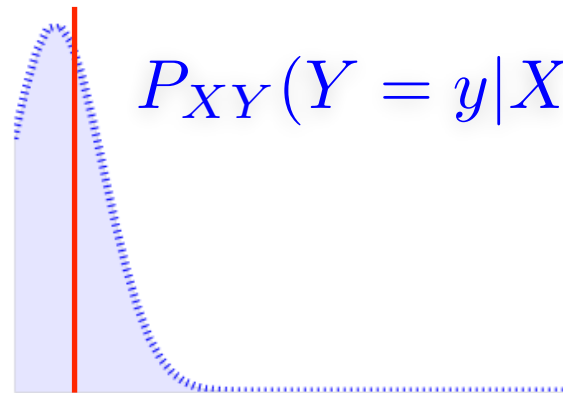
Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$$P_{XY}(Y = y|X = x_0)$$



$$P_{XY}(Y = y|X = x_1)$$

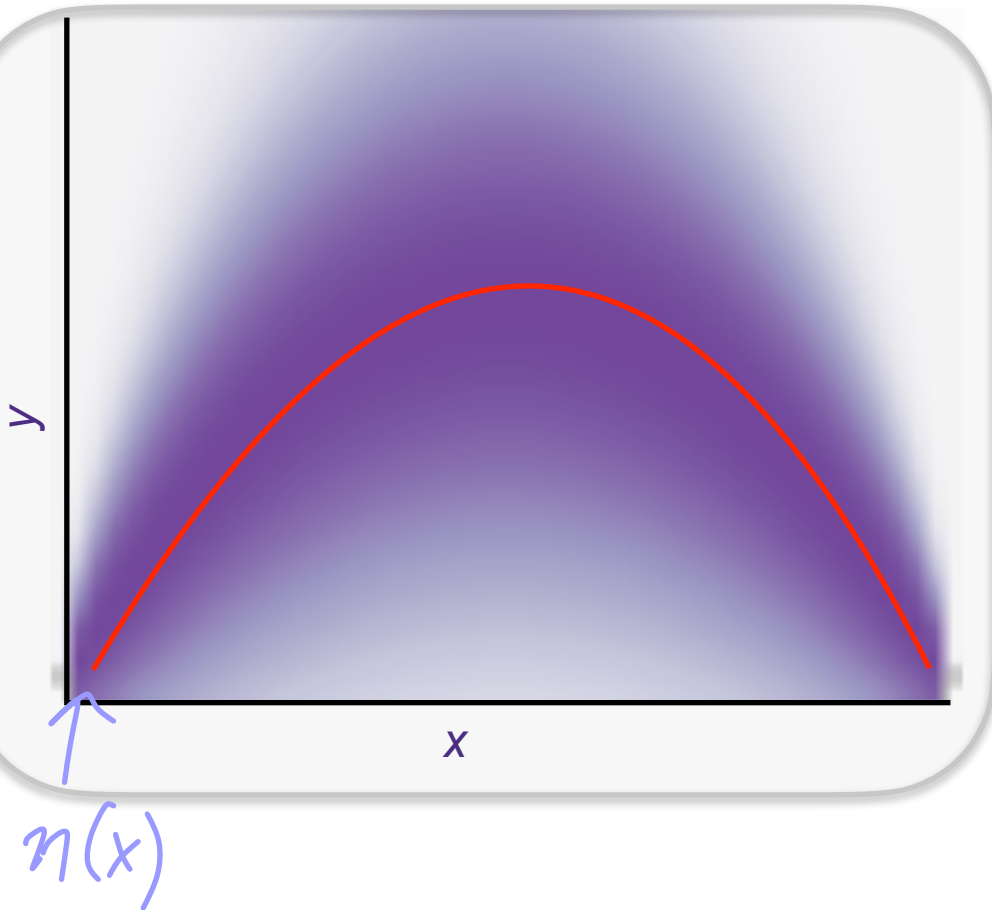


Statistical Learning

$$P_{XY}(X = x, Y = y)$$

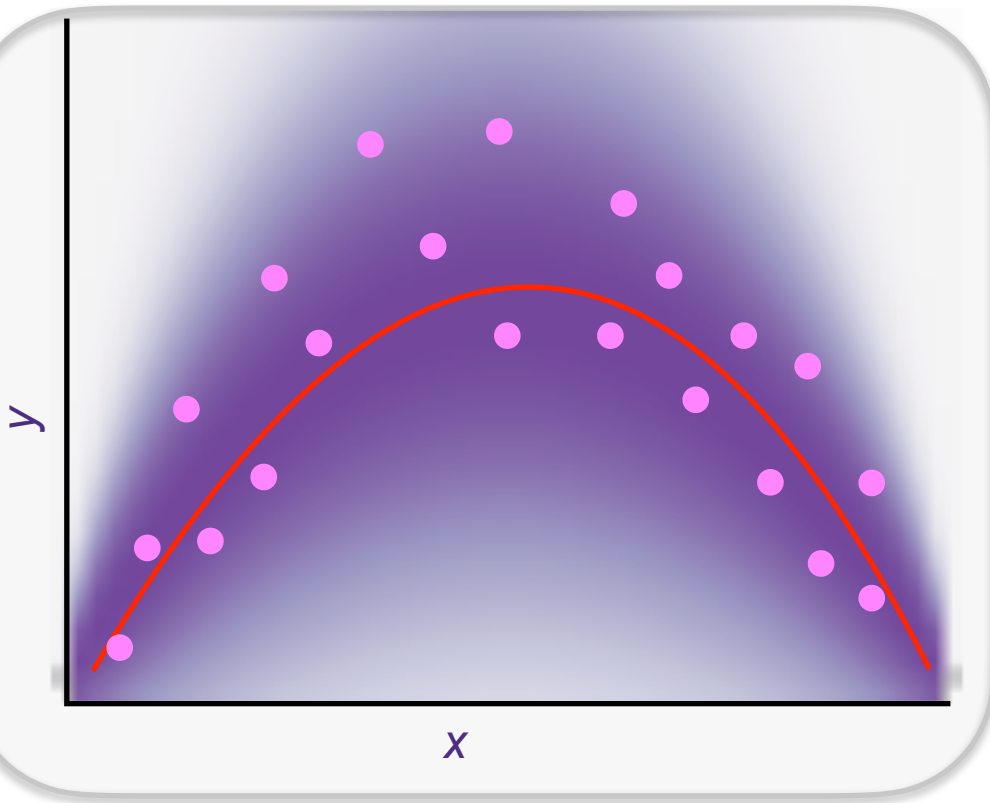
Ideally, we want to find:

$$\underline{\eta(x)} = \mathbb{E}_{Y|X}[Y|X = x]$$



Statistical Learning

$$P_{XY}(X = x, Y = y)$$



Ideally, we want to find:

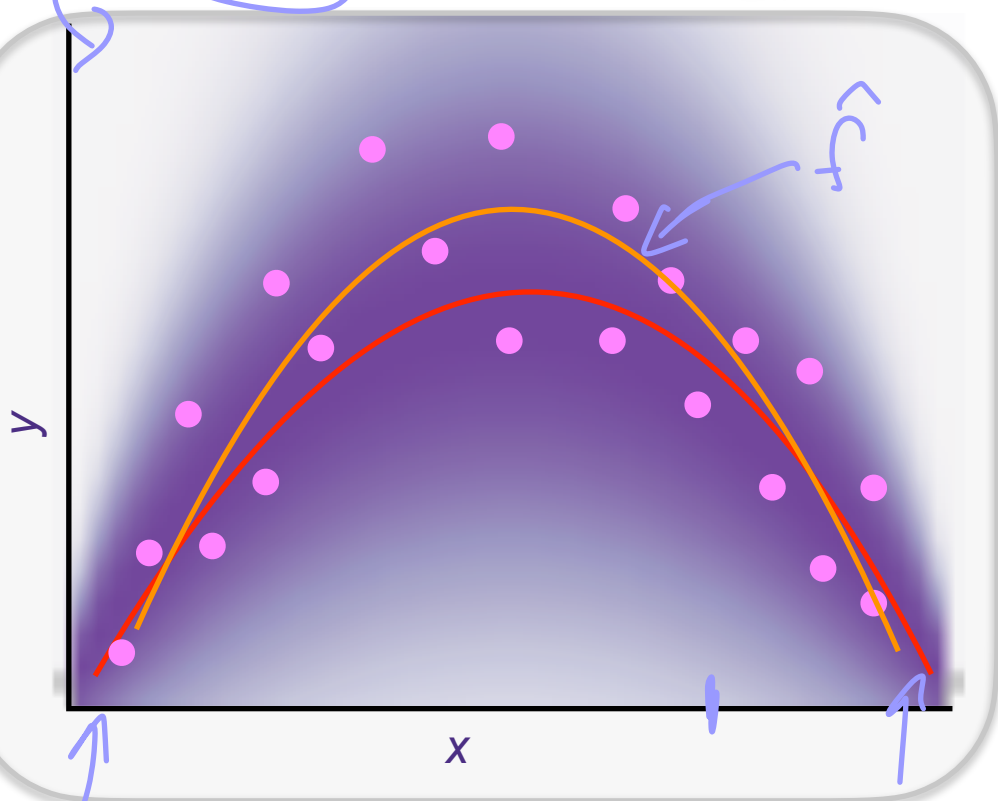
$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$$\underbrace{(x_i, y_i)}_{i.i.d.} \sim \underbrace{P_{XY}} \quad \text{for } i = 1, \dots, n$$

Statistical Learning

$$P_{XY}(X = x, Y = y)$$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$$(x_i, y_i) \stackrel{i.i.d.}{\sim} P_{XY} \quad \text{for } i = 1, \dots, n$$

and are restricted to a function class (e.g., linear)

so we compute:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$\hat{f} \neq \eta(x)$
↪ n is limited
↪ chosen the wrong model class

Empirical reconstruction error (ERE)

$$\mathbb{E}_{x,y} \left[\frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2 \right]$$

$$= \frac{1}{n} \sum_{i=1}^n \mathbb{E} [(y_i - f(x_i))^2]$$

by IID $\mathbb{E}[x_i]$ for an $i = \mathbb{E}[x] \forall x$

$$= \frac{1}{n} \sum_{i=1}^n \mathbb{E} [(y - f(x))^2]$$

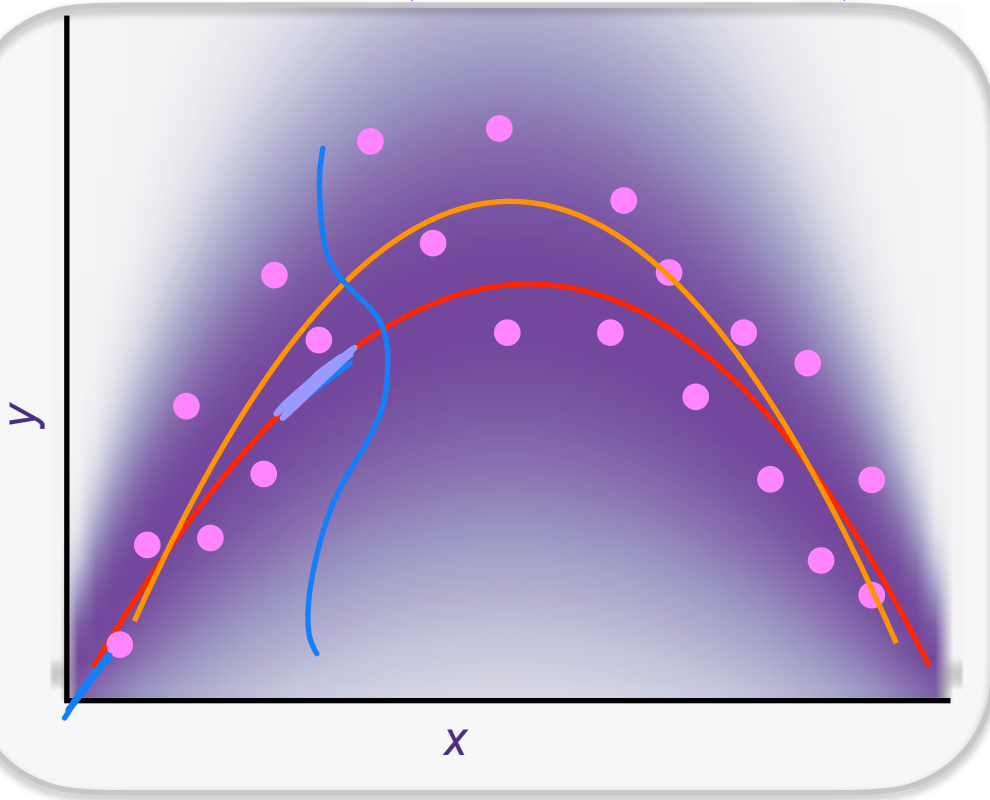
$$= \mathbb{E} [(y - f(x))^2]$$

Minimizing ERE \rightarrow minimizing loss in general squared error



Statistical Learning

$$P_{XY}(X = x, Y = y)$$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:
 $(x_i, y_i) \stackrel{i.i.d.}{\sim} P_{XY}$ for $i = 1, \dots, n$

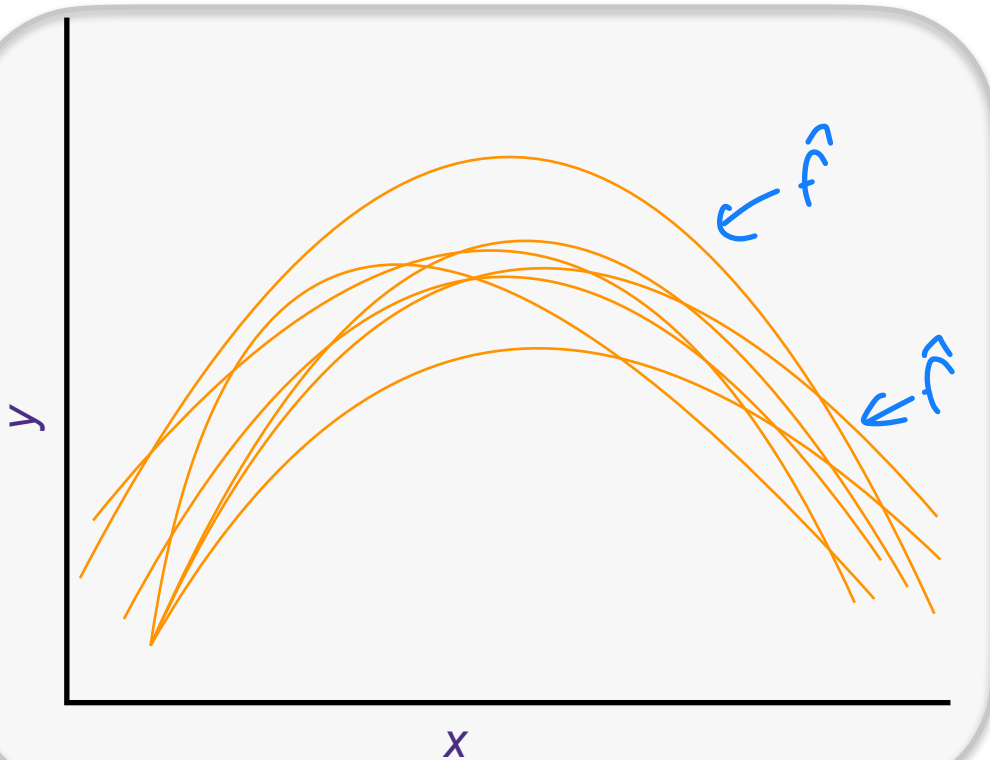
and are restricted to a function class (e.g., linear) so we compute:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

We care about future predictions: $\mathbb{E}_{XY}[(Y - \hat{f}(X))^2]$

Statistical Learning

$$P_{XY}(X = x, Y = y)$$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:
 $(x_i, y_i) \stackrel{i.i.d.}{\sim} P_{XY}$ for $i = 1, \dots, n$

and are restricted to a function class (e.g., linear) so we compute:

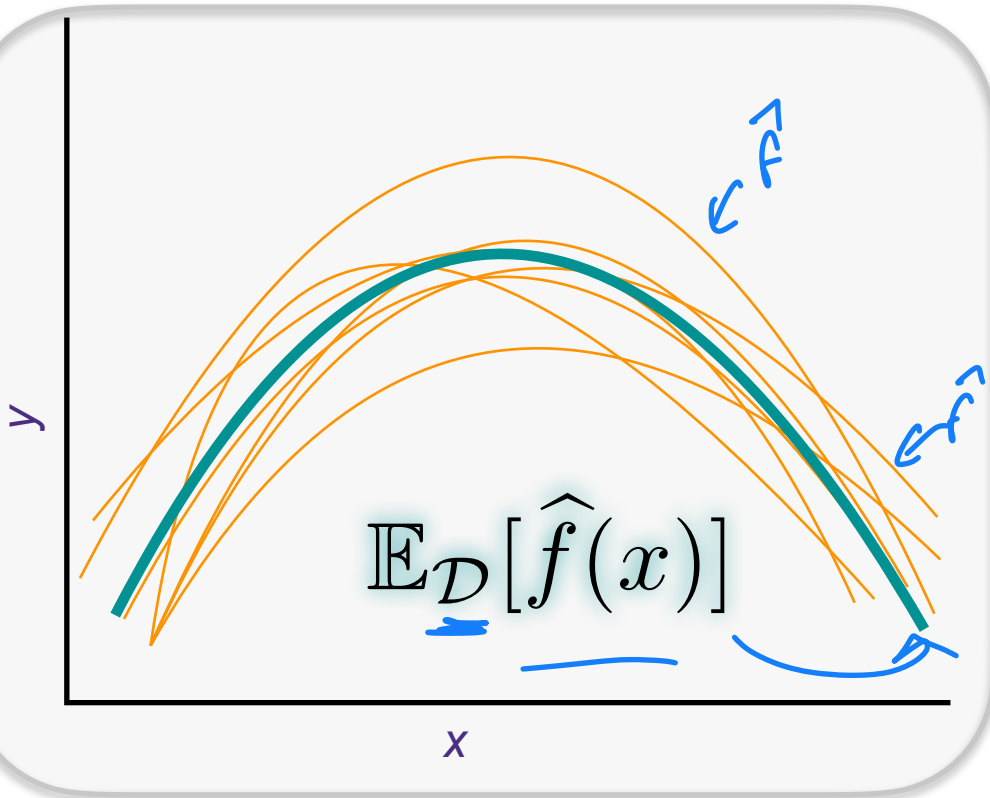
$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different \hat{f}

Statistical Learning

$$\bar{E} \quad \mathcal{D}_n \sim P_{XY}$$

$$P_{XY}(X = x, Y = y)$$



Ideally, we want to find:

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:
 $(x_i, y_i) \stackrel{i.i.d.}{\sim} P_{XY}$ for $i = 1, \dots, n$

and are restricted to a function class (e.g., linear) so we compute:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ results in different \hat{f}

Bias-Variance Tradeoff

magical func that gives true $\mathbb{E}[Y|X]$

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \hat{f}_{\mathcal{D}}(x))^2] | X = x]$$

→ generalization error we want to minimize

model class → finite data

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x] \quad \hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \hat{f}_{\mathcal{D}}(x))^2]|X = x] = \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[\underbrace{(Y - \eta(x))}_a + \underbrace{\eta(x) - \hat{f}_{\mathcal{D}}(x)}_b]^2]|X = x]$$

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$$\begin{aligned} \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \hat{f}_{\mathcal{D}}(x))^2] | X = x] &= \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x) + \eta(x) - \hat{f}_{\mathcal{D}}(x))^2] | X = x] \\ &= \mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x))^2 + 2(Y - \eta(x))(\eta(x) - \hat{f}_{\mathcal{D}}(x)) \\ &\quad + (\eta(x) - \hat{f}_{\mathcal{D}}(x))^2] | X = x] \\ &= \mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x] + \mathbb{E}_{\mathcal{D}}[(\eta(x) - \hat{f}_{\mathcal{D}}(x))^2] \end{aligned}$$

→ **irreducible error**
Caused by stochastic label noise

learning error
Caused by either using too "simple" of a model or not enough data to learn the model accurately

this is what we want to minimize in our control

$$\sum ab \rightarrow 0$$

$$E_{Y|X} [E_D \left[\underbrace{\sum (Y - \eta(x))}_{no D} \frac{(\eta(x) - \hat{f}_D(x))}{no Y} \right]]$$

$$E_{Y|X} [\cancel{E_D} [\sum (Y - \eta(x))]]$$

$$E_{Y|X} [Y - E[Y|X=x] | X=x] \leftarrow \text{by } \eta(x) = E_{Y|X}[Y|X=x]$$

$$= E_{Y|X} [Y|X=x] - E_{Y|X} [Y|X=x]$$

$$= 0$$

Y & D are independent
of each other w.r.t
the sources of randomness,
we're considering

Bias-Variance Tradeoff

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

learning error -

$$\mathbb{E}_{\mathcal{D}}[(\eta(x) - \hat{f}_{\mathcal{D}}(x))^2] = \mathbb{E}_{\mathcal{D}}[\underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])}_{a} + \underbrace{(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))}_{b}]^2]$$

Bias-Variance Tradeoff

$$y = mx + b$$

↑
offset

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}[(\eta(x) - \hat{f}_{\mathcal{D}}(x))^2] &= \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))^2] \\ &= \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + 2(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x)) \\ &\quad + (\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))^2 \end{aligned}$$

$$= \underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2}_{\text{biased squared}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))^2]}_{\text{variance}}$$

biased squared

variance

learning error = bias² + variance

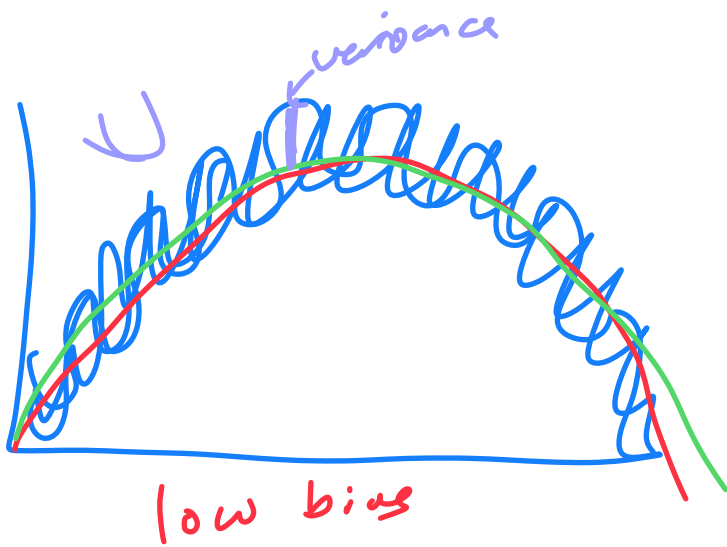
bias → if \hat{f} function class can't describe data

variance → for a particular dataset $\mathcal{D} \sim p(x, y)$, how diff
is \hat{f} from average \bar{f} - varying wildly → error

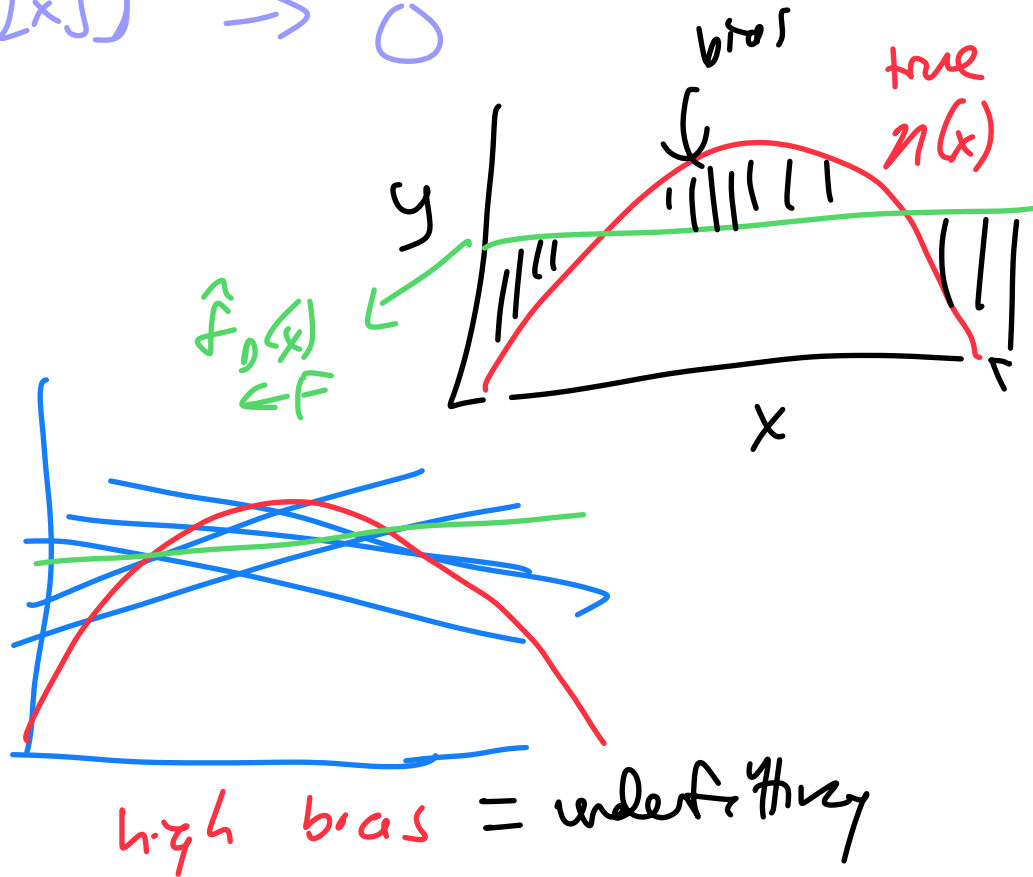
$Z ab \rightarrow 0$ b.c. $b \rightarrow 0$

$$E_D \left[Z \frac{(n(x) - E_D[\hat{f}_D(x)])}{a} \frac{(E_D[\hat{f}_D(x)] - \hat{f}_D(x))}{b \rightarrow 0} \right]$$

$$E_D [f_D(x)] - E_D [\hat{f}_D(x)] \rightarrow 0$$



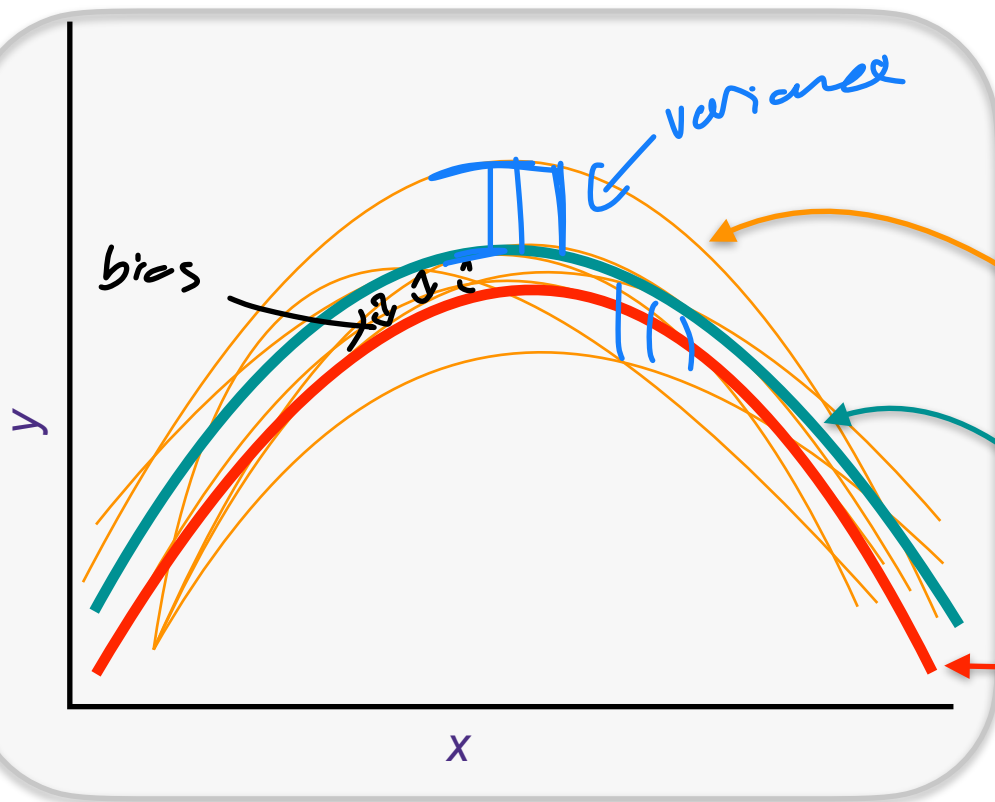
$$E_D [\hat{f}] \quad n(x)$$



high variance = overfitting

Statistical Learning

$$\underbrace{\mathbb{E}_{\mathcal{D}}[(\eta(x) - \hat{f}_{\mathcal{D}}(x))^2]}_{\text{learning error}} = \underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2}_{\text{biased squared}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))^2]}_{\text{variance}}$$



$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

$$\mathbb{E}_{\mathcal{D}}[\hat{f}(x)]$$

$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

Bias-Variance Tradeoff

the generalization error

$$\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y - \hat{f}_{\mathcal{D}}(x))^2] | X = x] = \underbrace{\mathbb{E}_{Y|X}[(Y - \eta(x))^2 | X = x]}_{\text{irreducible error}}$$

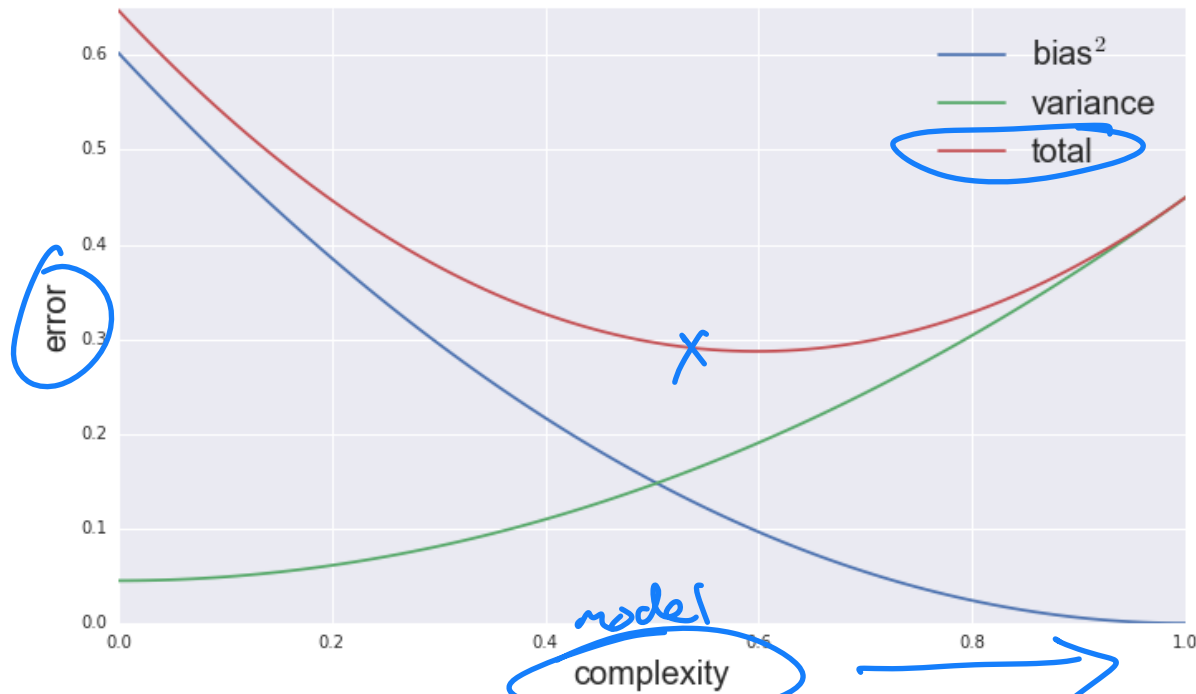
irreducible error ←

$$+ \underbrace{(\eta(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2}_{\text{biased squared}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \hat{f}_{\mathcal{D}}(x))^2]}_{\text{variance}}$$

biased squared

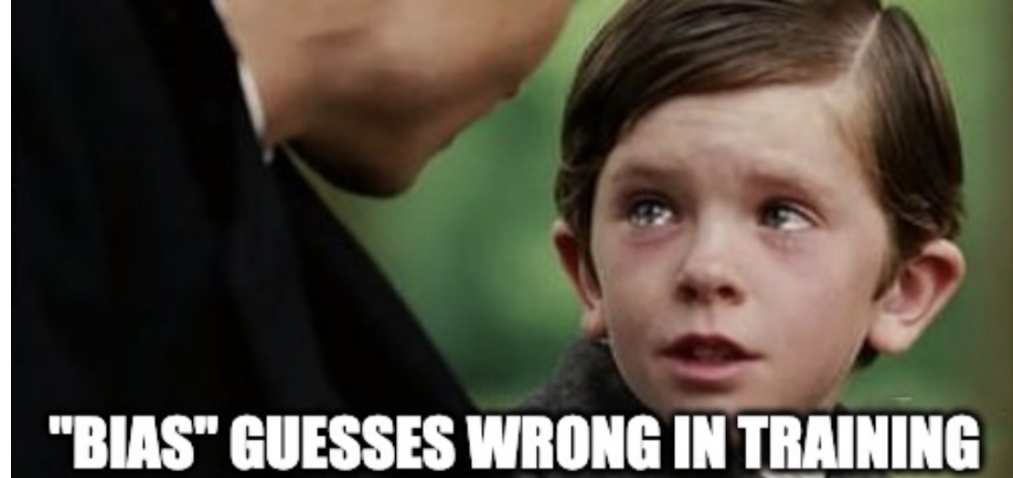
variance

learning error



That was a lot of
math

That was a lot of
math



Bias-Variance Demo

See [colab notebook](#)

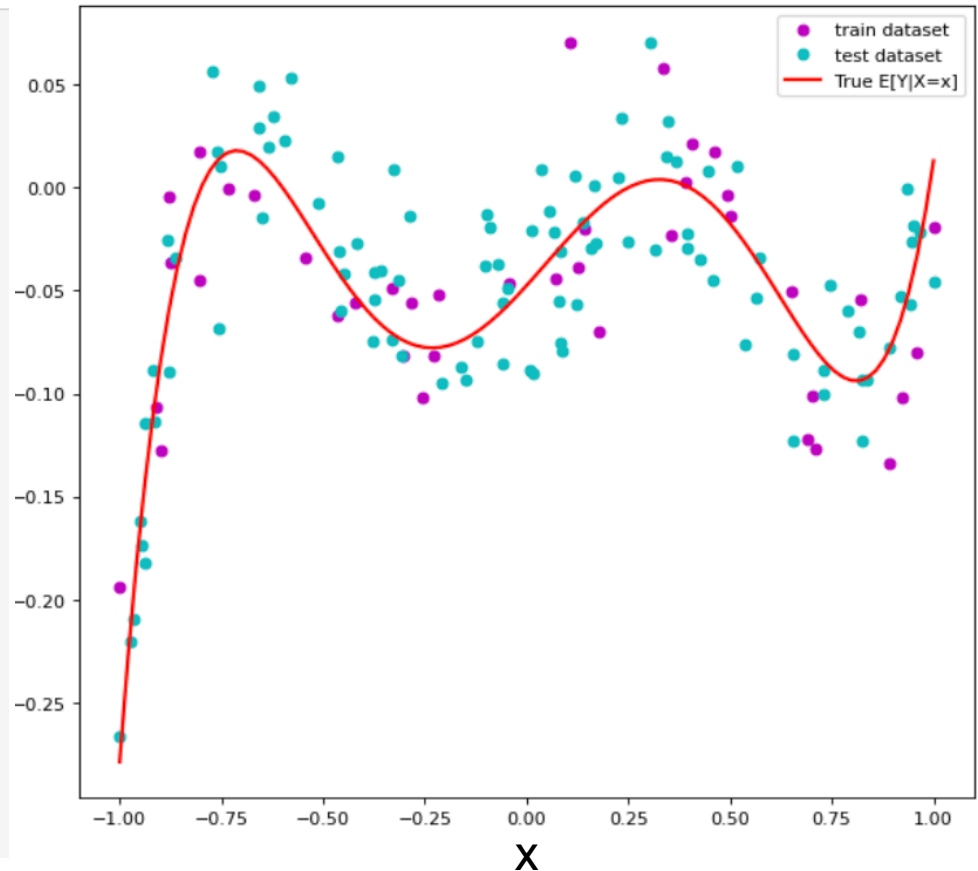
Bias-variance tradeoff

Let's define a ground truth $\eta(x) = E[Y|X = x]$ to be a degree-5 polynomial.

Then, define $P(Y|X = x) = \mathcal{N}(x, \sigma^2)$

```
In [1]: 1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Define \eta = E[Y|X] as a 5-th order polynomial
5 # (this is the same function we used in demo_polynomial.ipynb)
6 eta = lambda x: (x-.99)*(x-.4)*(x-.25)*(x+.6)*(x+.8)
7
8 # Generate samples from P(X,Y)
9 def sample_Pxy(n, noise_sigma=0.03):
10
11     # Sample inputs x from P(x).
12     x = np.random.uniform(-1,1,n)
13     x = np.sort(x)
14     x[0]=-1
15     x[-1]=1
16
17     # Draw samples from P(Y|X)
18     y = eta(x) + noise_sigma*np.random.randn(n)
19
20     return x,y
21
22 # Generate training data.
23 n_train = 40 # sample size
24 x, y = sample_Pxy(n=n_train)
25
26 # Generate test data.
27 n_test = 100
28 x_, y_ = sample_Pxy(n=n_test)
29
30 # Plot the samples and ground truth.
31 t = np.linspace(-1,1,100)
32 y0 = eta(t)
33 fig=plt.figure(figsize=(9, 8), dpi= 80, facecolor='w', edgecolor='k')
34 plt.plot(x,y,'mo',label='train dataset')
35 plt.plot(x_,y_,'co',label='test dataset')
36 plt.plot(t,y0,'r-', linewidth=2, label='True E[Y|X=x]')
37 plt.legend()
38 plt.show()
39
```

y



In typical scenarios, we only have one set of samples, which we separate into S_{test} and S_{train}

- it is critical that those two sets do not overlap and the sets are chosen randomly to ensure that the test error is independent of the training error, and also the test samples are coming from the same distribution as the new samples that will come in the future

However, in order to understand how the test error behaves (theoretically), we consider the expected test error, and call it true error: i.e. $\mathcal{L}_{\text{true}} = \mathbb{E}[\mathcal{L}_{\text{test}}]$

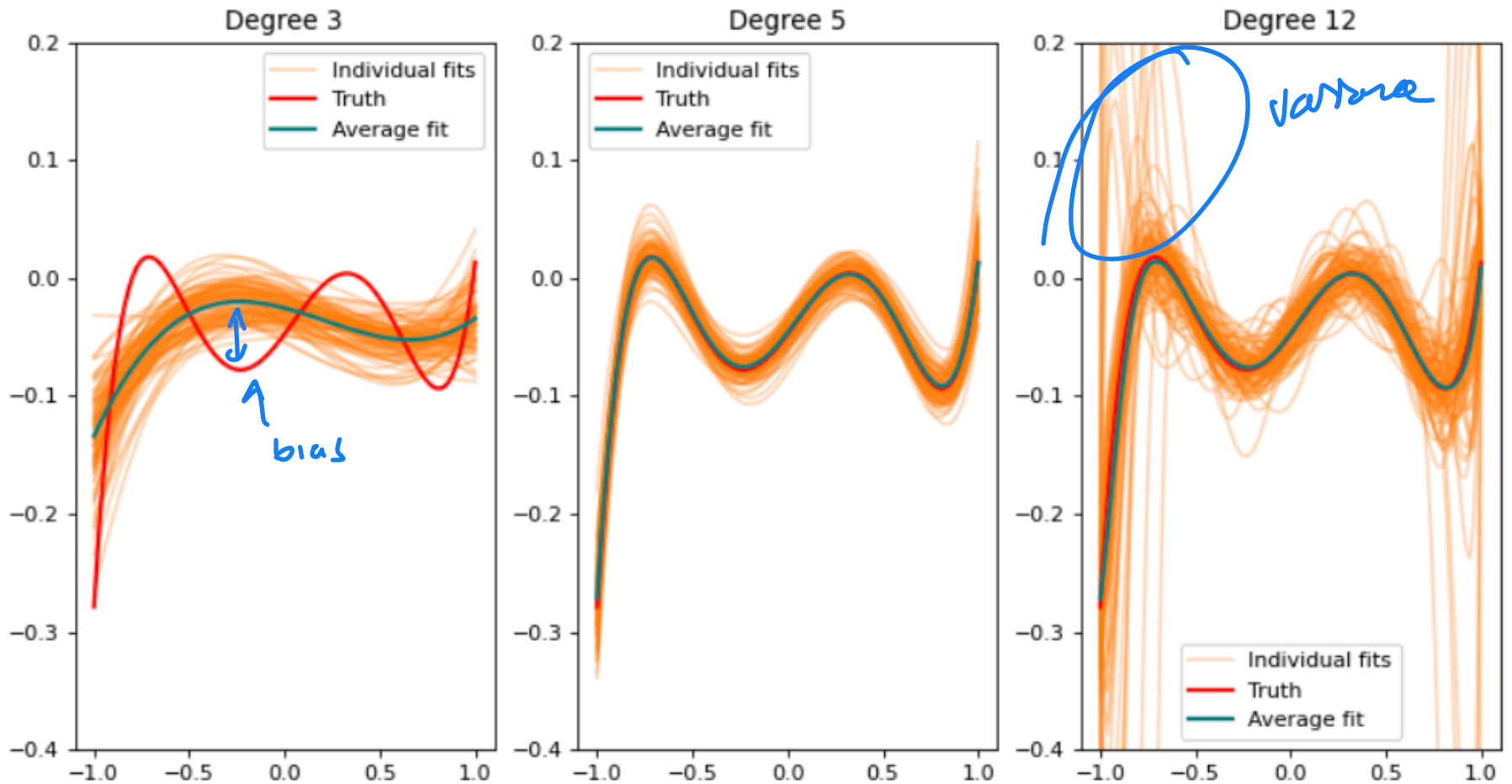
- test error is an **unbiased** estimate of the true error
- true error is unobservable (we cannot compute it, given finite samples)
- but we care the most about the true error
- so we use test error as a surrogate or an approximation

In order to compute the true error, we simulate a process where we get many fresh samples, and train new predictor each time with the fresh set of samples. It is important to understand that the resulting predictor $f_{S_{\text{train}}}(\cdot)$ is a random function, where the randomness comes from the fresh random training data set S_{train} . We will draw many such random functions, plot them, and see how test, train, true errors behave.

```

In [2]: 1 num_runs = 100
2 yhat_simple = np.zeros((num_runs,len(t)))
3 yhat_complex = np.zeros((num_runs,len(t)))
4 yhat_justright = np.zeros((num_runs,len(t)))
5
6 def poly_features(x, degree):
7     """
8     Generates a matrix where each column corresponds to x raised to a power from 0 to 'degree'.
9
10    Parameters:
11    x (numpy array): The input data.
12    degree (int): The maximum degree of the polynomial features.
13
14    Returns:
15    numpy array: A matrix where the columns are [1, x, x^2, ..., x^degree].
16    """
17    return np.vstack([x**i for i in range(degree + 1)]).T
18
19 run=0
20 while run < num_runs:
21     n = 40 # sample size
22     x,y = sample_Pxy(n)
23
24     # degree-3 polynomial linear regression
25     p=3
26     X = poly_features(x, degree=p)
27     w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
28     T_ = poly_features(t, degree=p)
29     yhat_simple[int(run)] = np.matmul(T_,w)
30     yh = np.matmul(X,w)
31     X_ = poly_features(x_, degree=p)
32     y_h = np.matmul(X_,w)
33     w_ = w
34
35     # degree-5 polynomial linear regression
36     p=5
37     X = poly_features(x, degree=p)
38     w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
39     T_ = poly_features(t, degree=p)
40     yhat_justright[int(run)] = np.matmul(T_,w)
41     yh = np.matmul(X,w)
42     X_ = poly_features(x_, degree=p)
43     y_h = np.matmul(X_,w)
44     w_ = w
45
46     # degree-12 polynomial linear regression
47     p=12
48     X = poly_features(x, degree=p)
49     w = np.array(p+1)
50     w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
51     T_ = poly_features(t, degree=p)
52     yhat_complex[int(run)] = np.matmul(T_,w)
53     yh1 = np.matmul(X,w)
54     X_ = poly_features(x_, degree=p)
55     y_h1 = np.matmul(X_,w)
56
57     run=run+1
58
59 def build_plot(yhat, title):
60     ave_fit = np.zeros(np.size(t))
61     for irun in range(1, num_runs):
62         plt.plot(t,yhat[int(irun)], color='tab:orange',alpha=0.3)
63         ave_fit = ave_fit + yhat[int(irun)]
64     plt.plot(t,yhat[0], 'tab:orange',alpha=0.3,label='Individual fits')
65     plt.plot(t,y0, 'r-', linewidth=2, label='Truth')
66     plt.plot(t,(1/float(num_runs))*ave_fit, color='teal', linewidth=2,label='Average fit')
67     plt.legend()
68     plt.title(title)
69     axes = plt.gca()
70     axes.set_ylim([-0.4,0.2])
71
72 fig=plt.figure(figsize=(12, 6), dpi= 80, facecolor='w', edgecolor='k')
73 plt.subplot(1,3,1)
74 build_plot(yhat_simple, 'Degree 3')
75 plt.subplot(1,3,2)
76 build_plot(yhat_justright, 'Degree 5')
77 plt.subplot(1,3,3)
78 build_plot(yhat_complex, 'Degree 12')
79 plt.show()
80

```



Takeaways

- True function has degree 5, with additive noise
- Degree 3 fit has very high bias because the teal line, which is the average of the fits, is very different from true red line (because $3 < 5$). Each individual fit (orange) varies about the average fit (teal) moderately indicating moderate variance. The overall error ($\text{bias}^2 + \text{variance}$) of each individual fit is high.
- Degree 12 has very low bias because the teal line is almost equal to the red (because $12 > 5$). But each individual fit varies a lot about its average teal line indicating high variance. The overall error ($\text{bias}^2 + \text{variance}$) of each individual fit is high.
- Degree 5 has very low bias because the teal line is almost equal to the red (because $5 = 5$). And each individual fit varies only a little about its average teal line indicating small variance. The overall error ($\text{bias}^2 + \text{variance}$) of each individual fit is low.