Cross-Validation

Matt Golub
Hunter Schafer

UNIVERSITY of WASHINGTON

How… How… How???????

- How do we pick the number of basis functions…

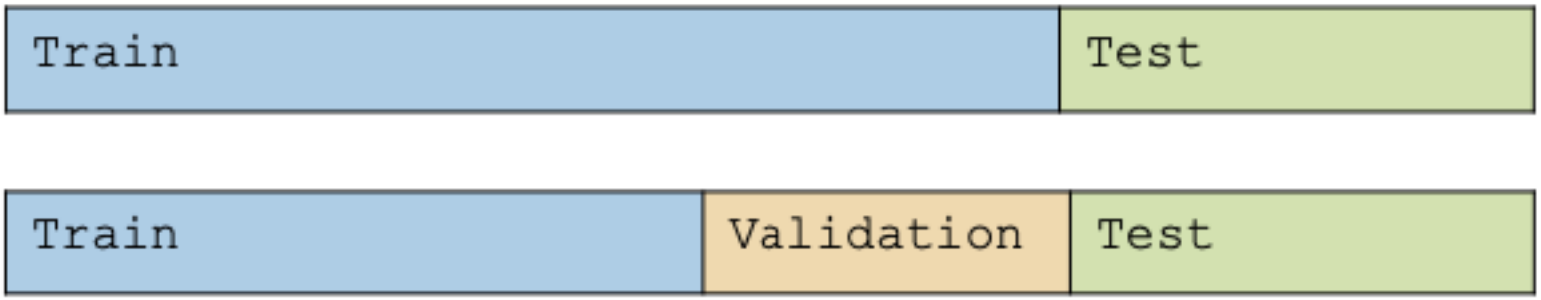
- We could use the test data, but…

How… How… How???????

- How do we pick the number of basis functions…
- We could use the test data, but…
  - Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever train on the test data

Validation Set

- Simplest approach is to add another dataset partition called the **validation set**



- Each set has its own role
  - Train: Used to train a model of each model complexity
  - Validation: Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  - Test: Once the model is chosen, can get an estimate of future error by test

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data**
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn classifier $f_{D\setminus j}$ with $D\setminus j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!**

(LOO) Leave-one-out cross validation

- **Consider a validation set with 1 example:**
  - **D – training data**
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**
- **Learn classifier $f_{D\setminus j}$ with D\j dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of $\text{error}_{true}(f_{D\setminus j})$!**

- **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new classifier $f_{D\setminus j}$**
  - **Estimate error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate!

- When computing LOOCV error, we only use *N-1* data points
  - So it's not estimate of true error of learning with *N* data points
  - Usually pessimistic, though – learning with less data typically gives worse answer

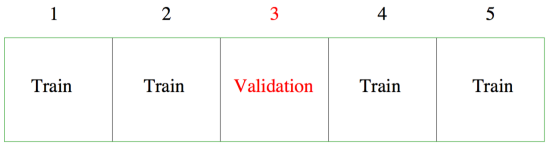
- LOO is almost unbiased! Use LOO error for model selection!!!
  - E.g., picking degree

Computational cost of LOO

- Suppose you have 100,000 data points
- You implemented a great version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day!!!
  -

Use *k*-fold cross validation

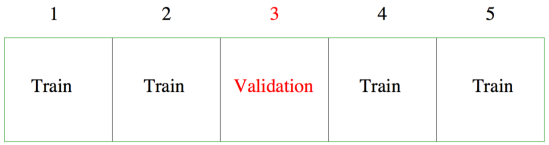
- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn classifier $f_{D\backslash Di}$ using data point not in D_i
  - Estimate error of $f_{D\backslash Di}$ on validation set D_i:



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\backslash\mathcal{D}_i}(x_j))^2$$

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn classifier $f_{D\setminus Di}$ using data point not in D_i
  - Estimate error of $f_{D\setminus Di}$ on validation set D_i:



$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\setminus\mathcal{D}_i}(x_j))^2$$

- k-fold cross validation error is average over data splits:

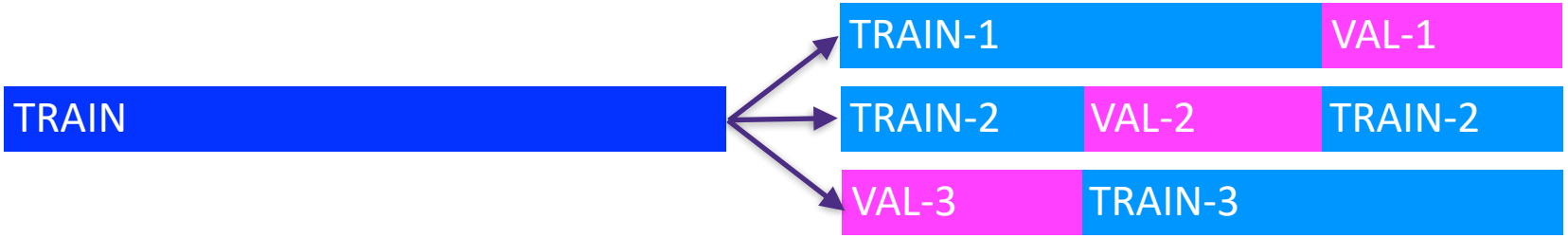
$$error_{k-fold} = \frac{1}{k}\sum_{i=1}^{k} error_{\mathcal{D}_i}$$

- k-fold cross validation properties:
  - Much faster to compute than LOO
  - More (pessimistically) biased – using much less data, only *n(k-1)/k*
  - Usually, k = 10

Recap

- Given a dataset, begin by splitting into

- Model selection: Use k-fold cross-validation on **TRAIN** to train predictor and choose magic parameters such as degree



- Model assessment: Use **TEST** to assess the accuracy of the model you output
  - **Never ever ever ever ever train or choose parameters based on the test data**

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST**
- **What's wrong with this procedure?**

Example 2

- **Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the entire dataset:**

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- **After picking our 50 features, we then break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., least squares on TRAIN
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features
  - Justifying the accuracy of the estimate
    - E.g., report TEST error

Bias-Variance Tradeoff

Matt Golub
Hunter Schafer

UNIVERSITY of WASHINGTON

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 $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

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 $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

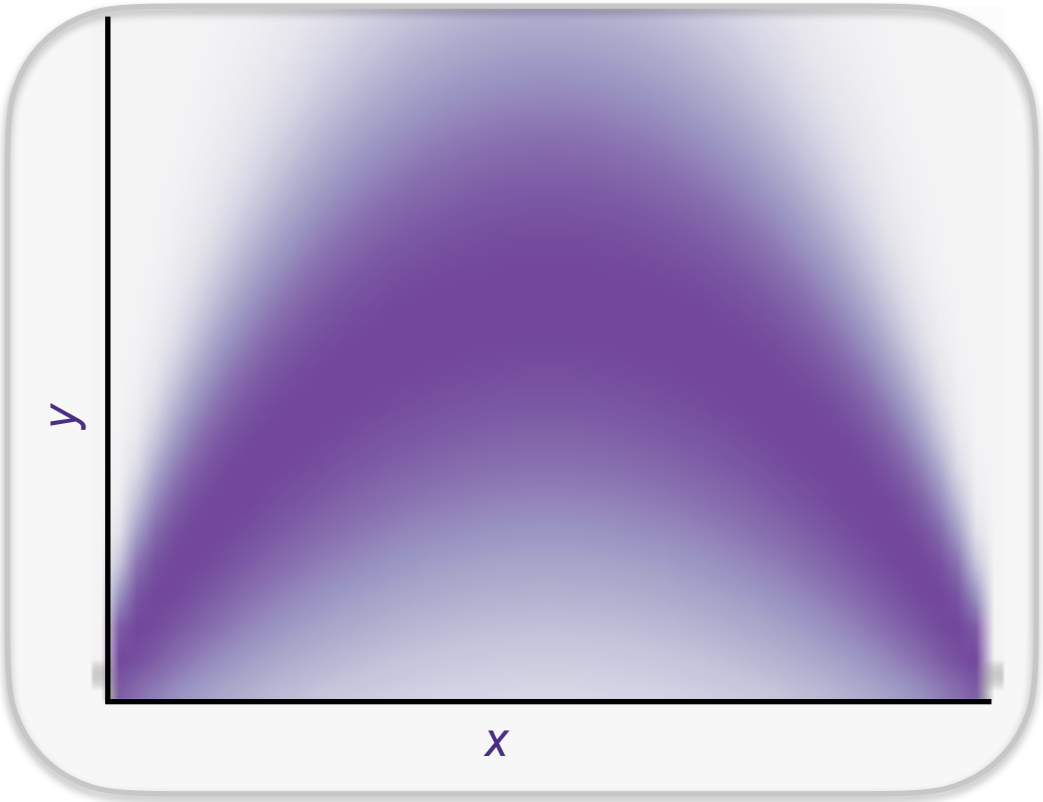
$$\begin{aligned} 0 &= \frac{d}{dc_x}\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[\frac{d}{dc_x}(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[-2(Y - c_x) | X = x] = -2\mathbb{E}_{Y|X}[Y | X = x] + 2c_x \end{aligned}$$

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

Statistical Learning

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

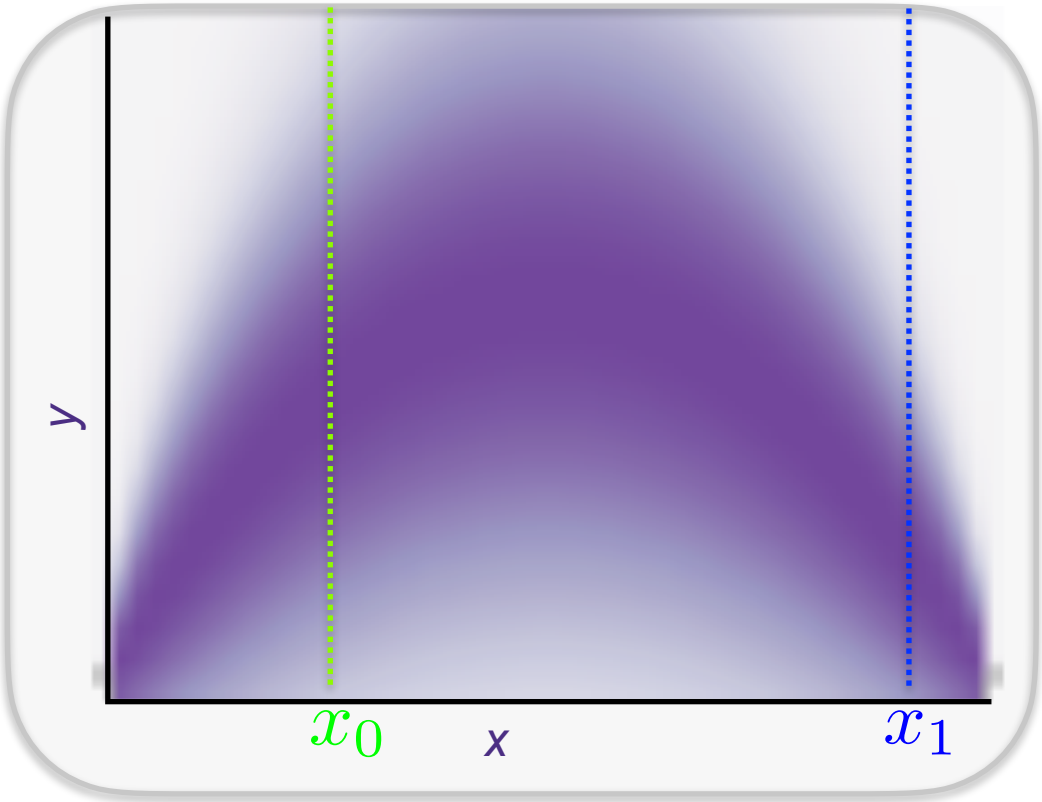
$$P_{XY}(X = x, Y = y)$$



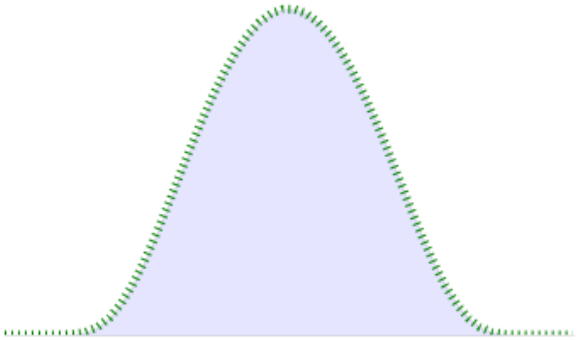
Statistical Learning

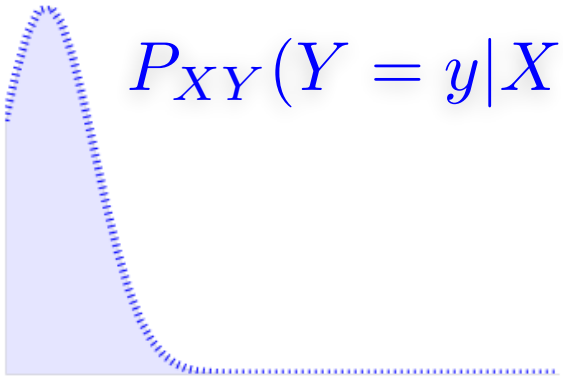
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

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



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



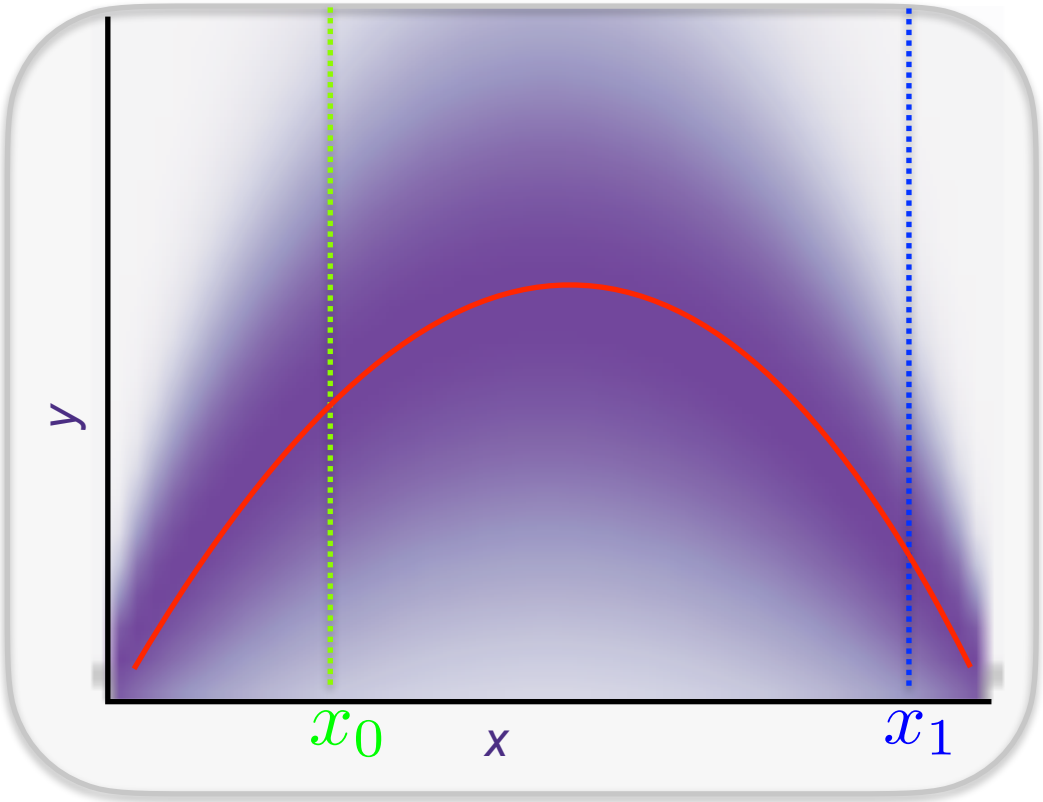


$P_{XY}(Y = y|X = x_1)$

Statistical Learning

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

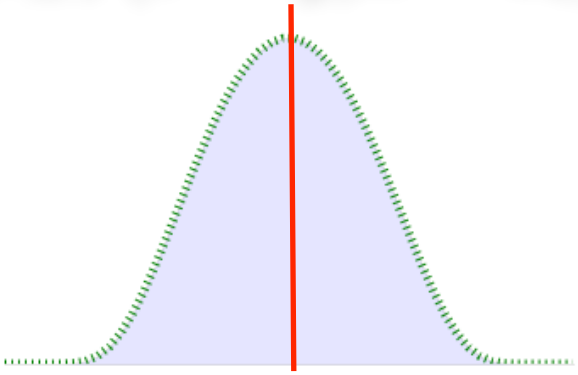
$P_{XY}(X = x, Y = y)$

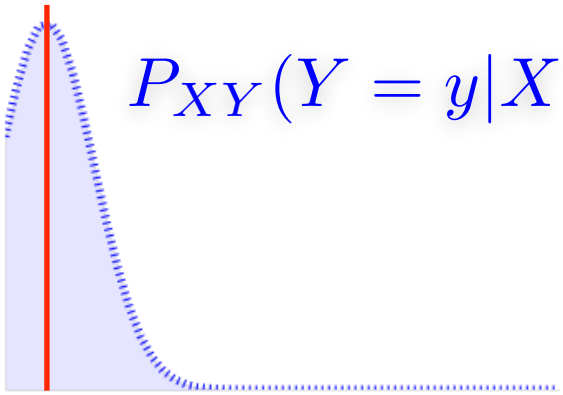


Ideally, we want to find:

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

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

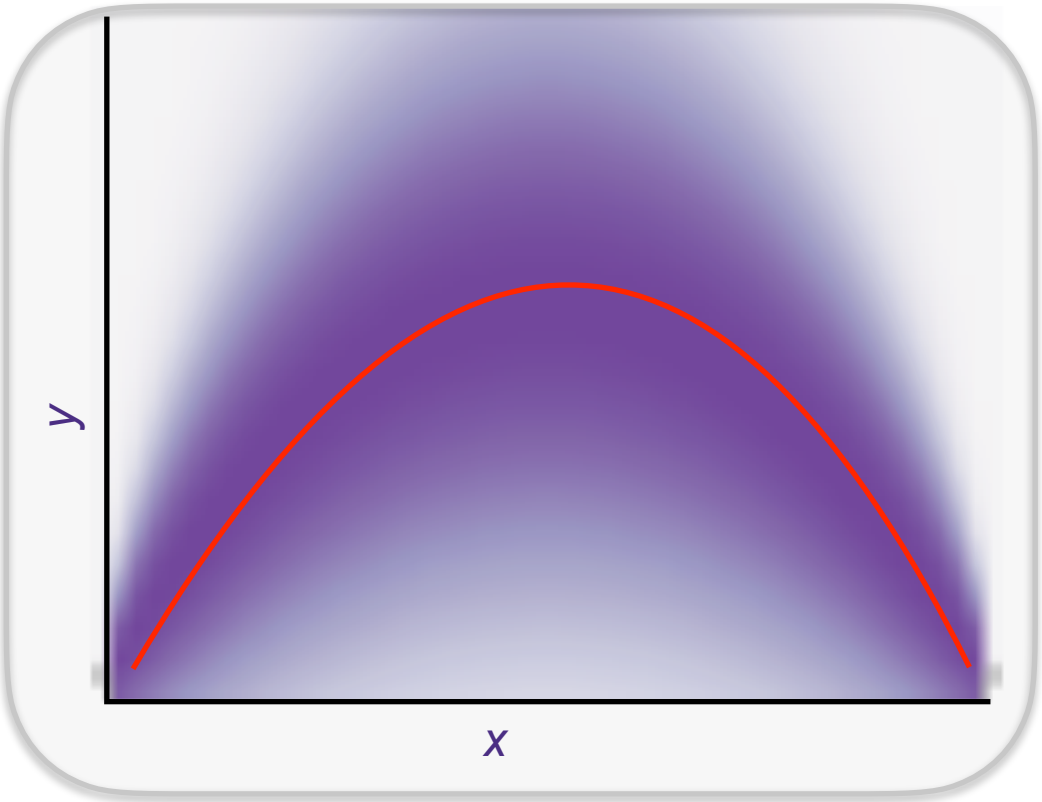




$P_{XY}(Y = y|X = x_1)$

Statistical Learning

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

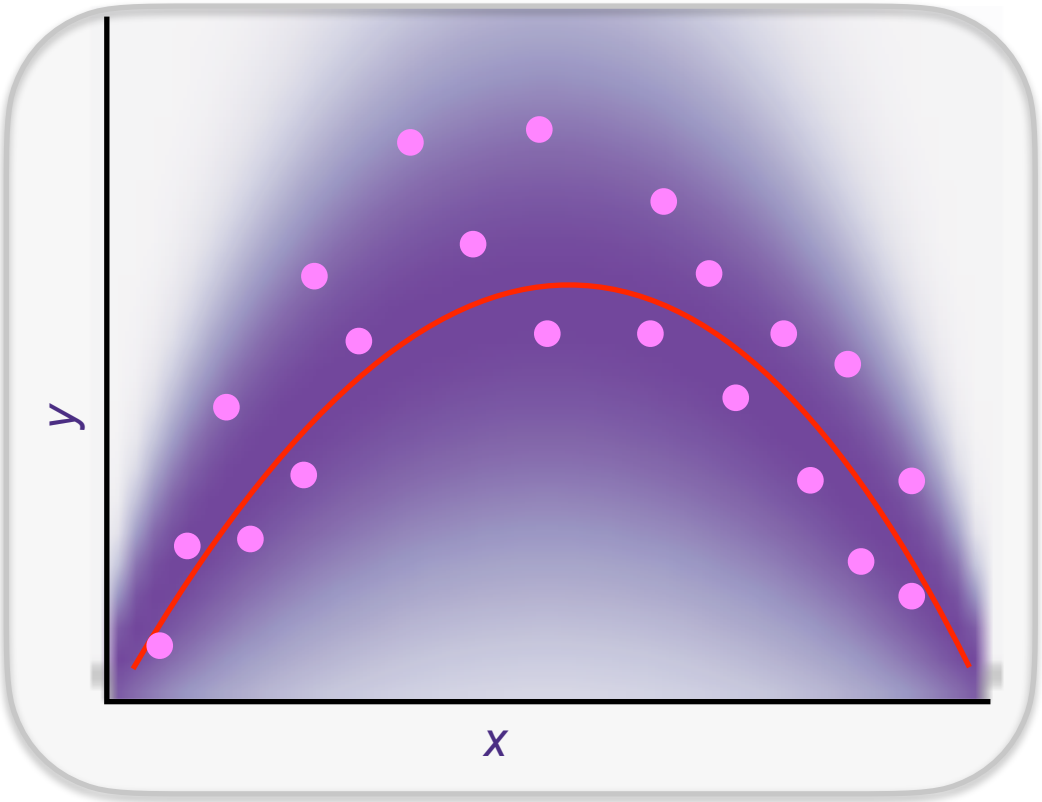


Ideally, we want to find:

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

Statistical Learning

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



Ideally, we want to find:

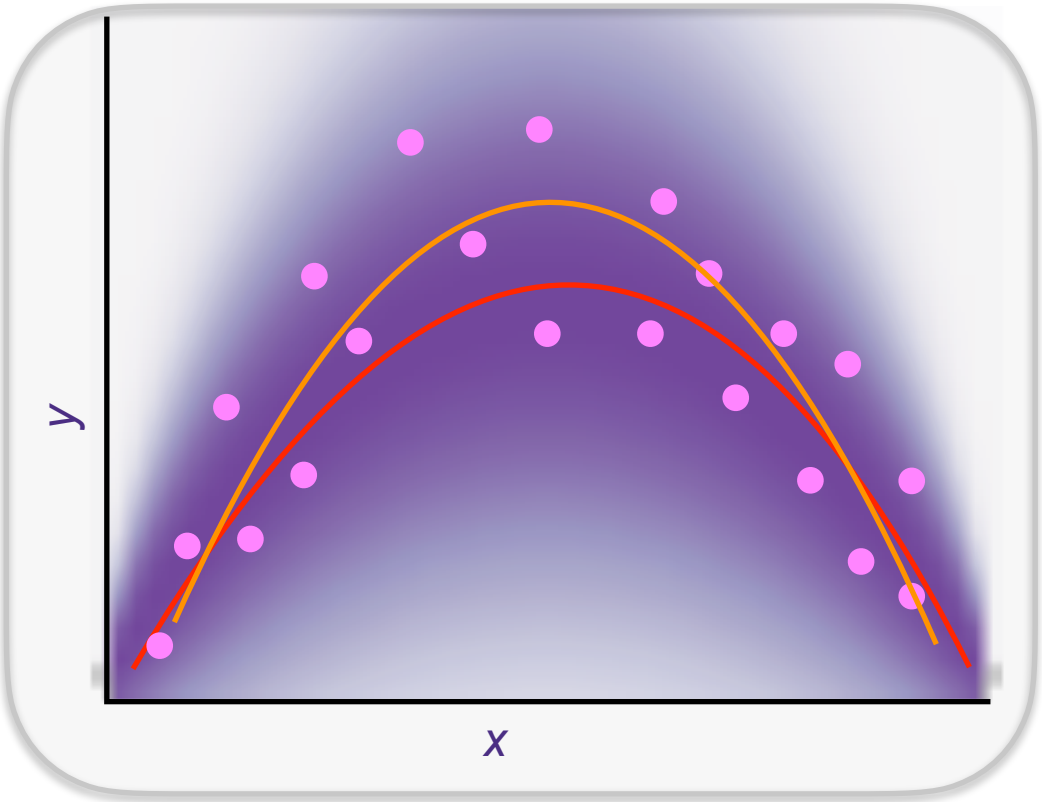
$$\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$$

But we only have samples:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

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:

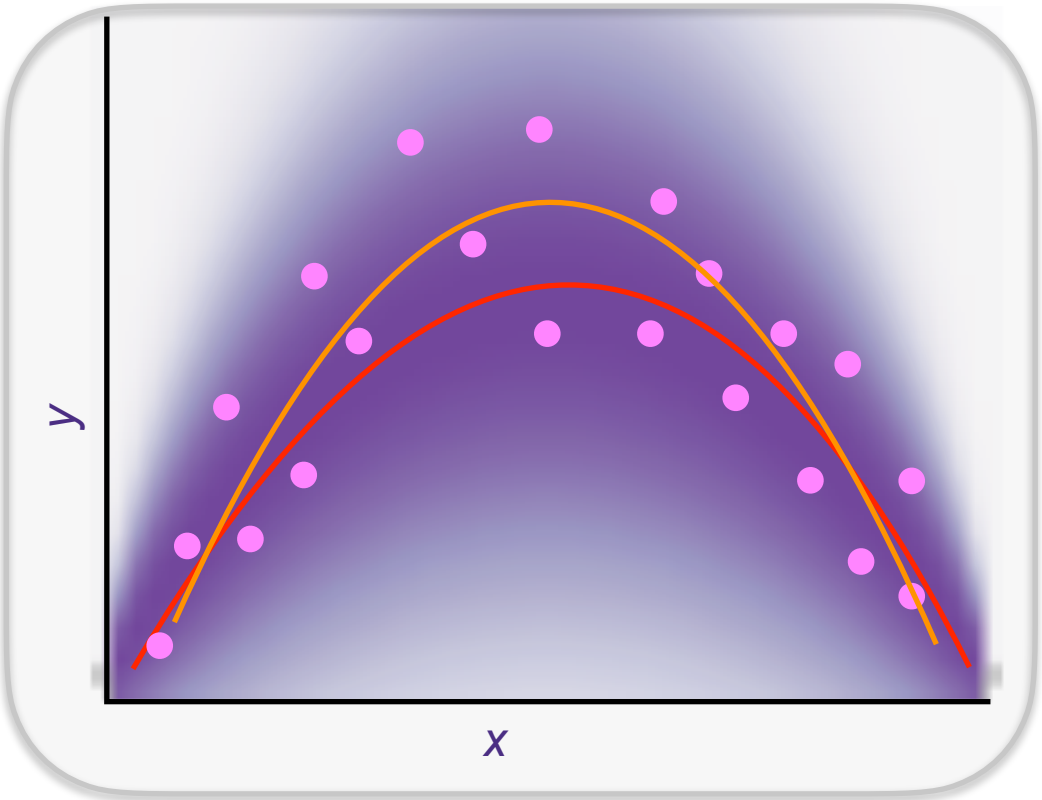
$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

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

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

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:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

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

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

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

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:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

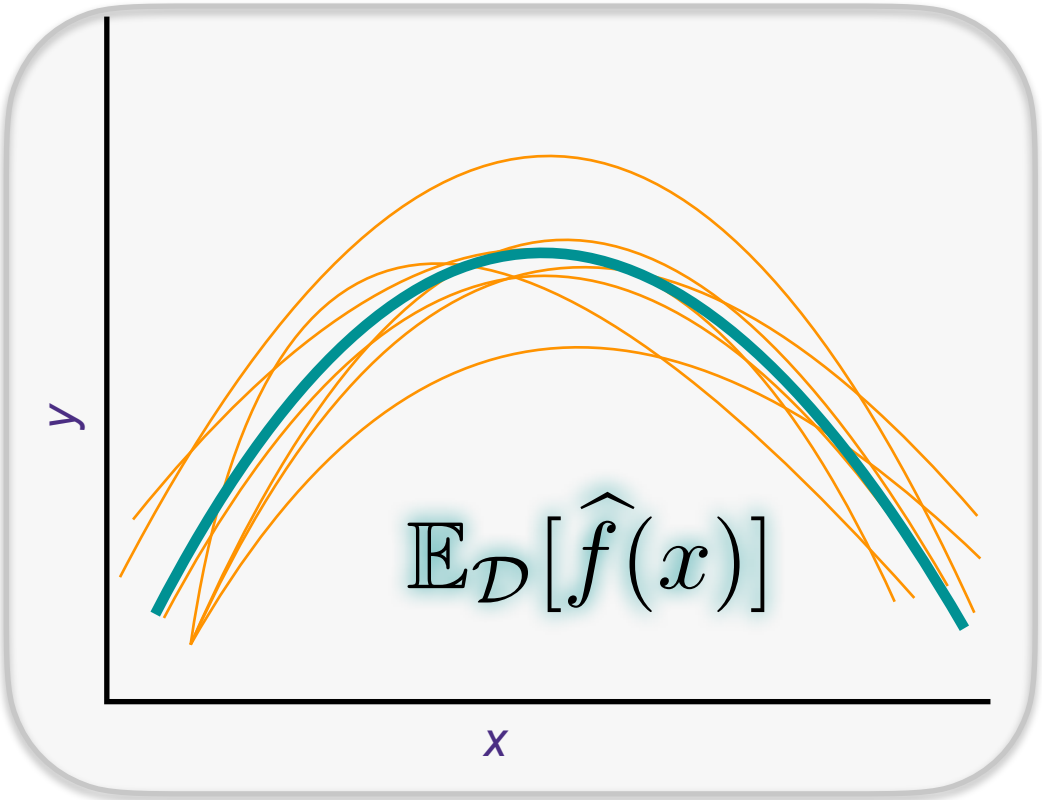
and are restricted to a function class (e.g., linear) so we compute:

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

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

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:

$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

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

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

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

Bias-Variance Tradeoff

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

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

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

Bias-Variance Tradeoff

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

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

Bias-Variance Tradeoff

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

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

$$= \mathbb{E}_{Y|X}\Big[\mathbb{E}_{\mathcal{D}}[(Y - \eta(x))^2 + 2(Y - \eta(x))(\eta(x) - \widehat{f}_{\mathcal{D}}(x)) + (\eta(x) - \widehat{f}_{\mathcal{D}}(x))^2]\big|X = x\Big]$$

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

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

Bias-Variance Tradeoff

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

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

Bias-Variance Tradeoff

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

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

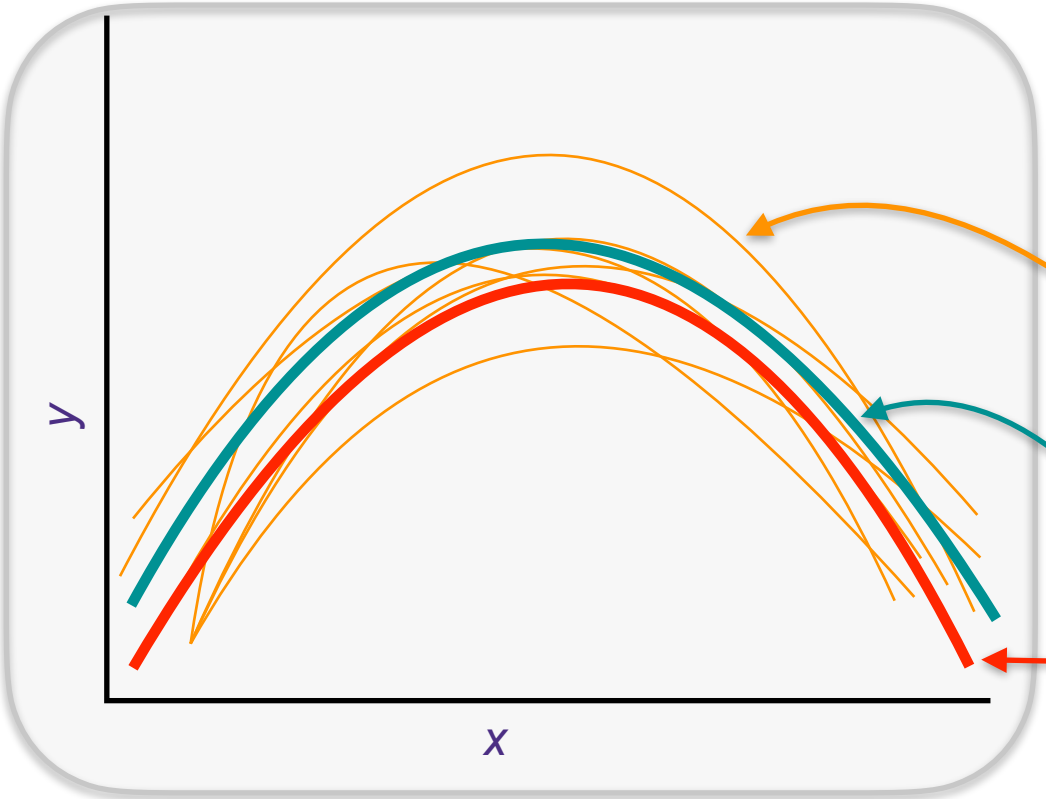
$$= \mathbb{E}_{\mathcal{D}}[(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2 + 2(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))$$
$$+ (\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)]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

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

biased squared **variance**

Statistical Learning

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



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

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

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

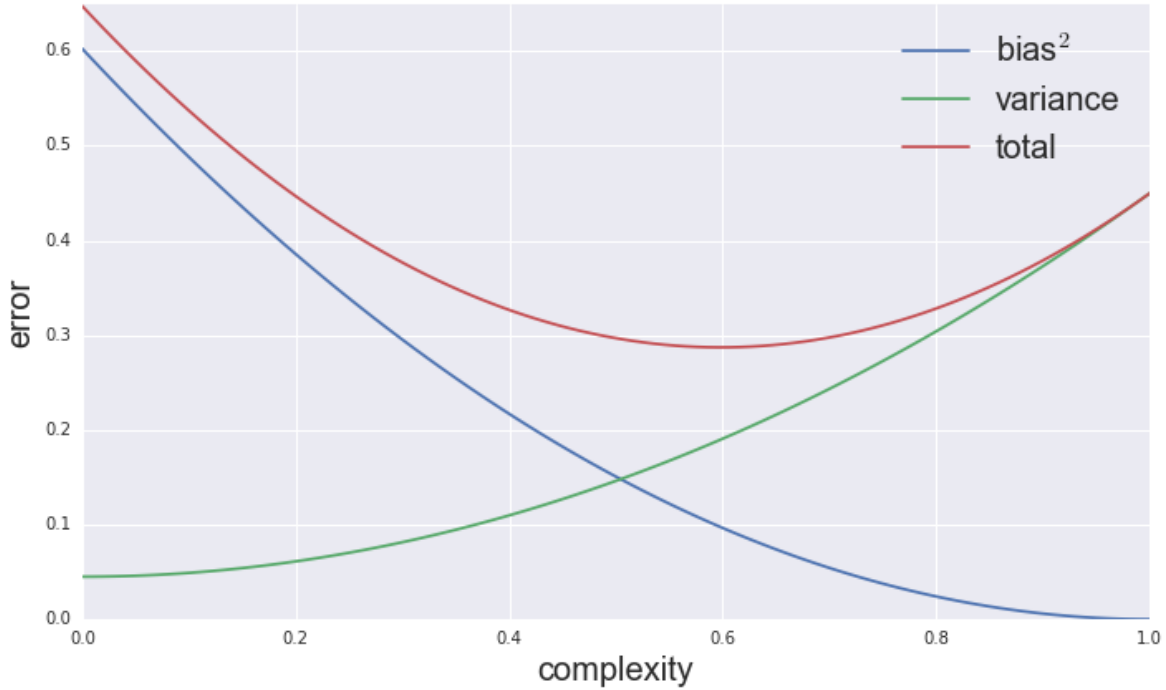
Bias-Variance Tradeoff

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

irreducible error

$$+(\eta(x) - \mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)])^2 + \mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - \widehat{f}_{\mathcal{D}}(x))^2]$$

biased squared **variance**



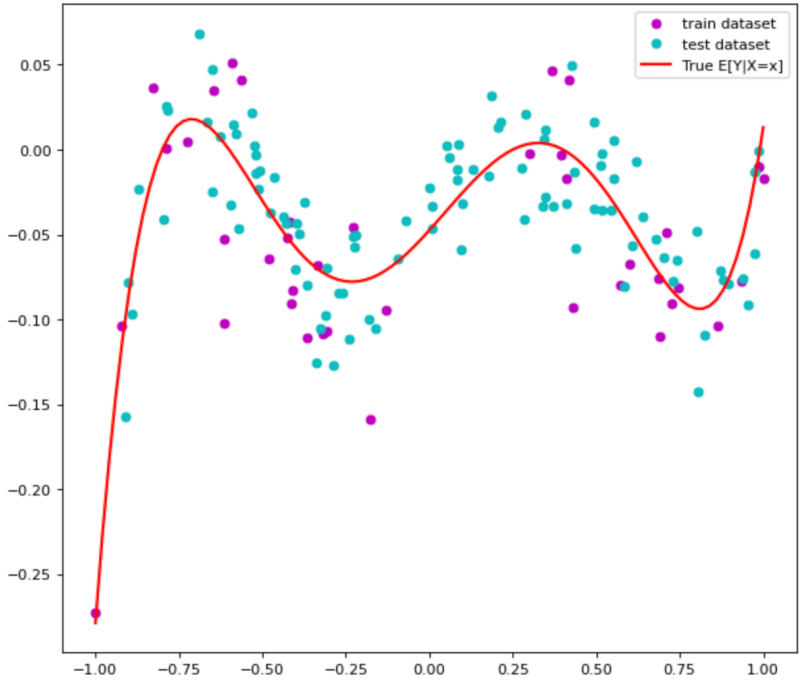
Bias-Variance Demo

demo_tradeoff.ipynb

Bias-variance tradeoff

we consider degree-5 polynomial model (for ground truth) with additive Gaussian noise

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

# ground truth is 5-th order polynomial and we add Gaussian noise to it
eta = lambda x: (x-.99)*(x-.4)*(x-.25)*(x+.6)*(x+.8)
noise_sigma = 0.03

# generate training data
n = 40 # sample size
x = np.random.uniform(-1,1,n) #sample input data
x = np.sort(x)
x[0]=-1
x[n-1]=1
y = eta(x) + noise_sigma*np.random.randn(n)

# generate test data
n_ = 100
x_ = np.random.uniform(-1,1,n_)
y_ = eta(x_) + noise_sigma*np.random.randn(n_)

# plot the samples and grounstruth
t = np.linspace(-1,1,100)
y0 = eta(t)
fig=plt.figure(figsize=(9, 8), dpi= 80, facecolor='w', edgecolor='k')
plt.plot(x,y,'mo',label='train dataset')
plt.plot(x_,y_,'co',label='test dataset')
plt.plot(t,y0,'r-', linewidth=2, label='True E[Y|X=x]')
plt.legend()
plt.show()
```



In typical scenarios, we only have one set of samples, which we separate into $\mathcal{S}_{\text{test}}$ and $\mathcal{S}_{\text{train}}$

- 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 frmo the same distribution as the new samples that will come in the future

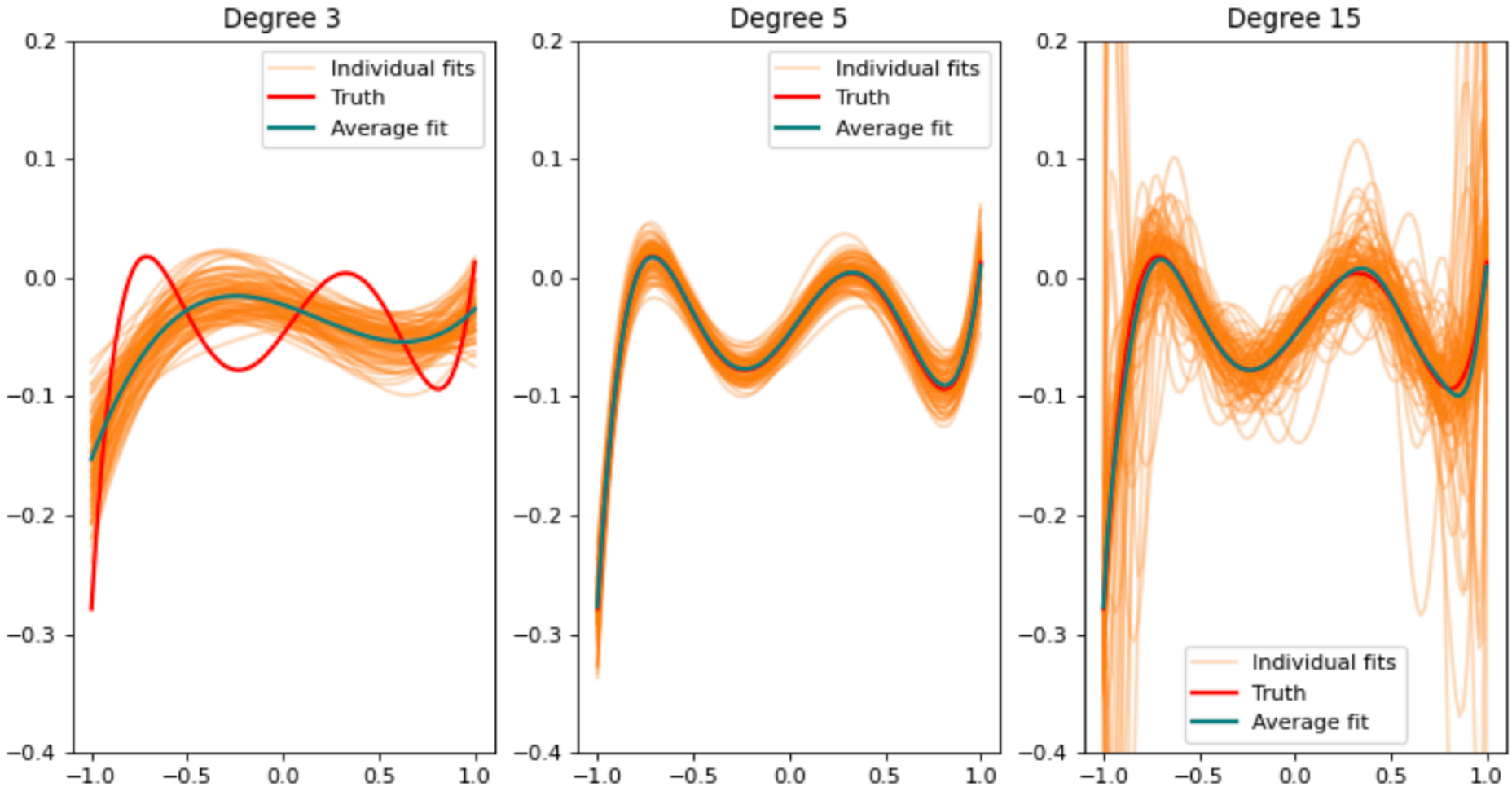
However, in order to understand how the test error behaves (theoretically), we consider the expected test error, and call it true error: i.e. $\mathcal{L}_{\text{true}} = \mathbb{E}[\mathcal{L}_{\text{test}}]$

- test error is an **unbiased** estimate of the true error
- true error is unobservable (we canot copmute 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 $f_{\mathcal{S}_{\text{train}}}(\cdot)$ is a random function, where the randomness coems from the fresh random training data set $\mathcal{S}_{\text{train}}$. We will draw many such random functions, and plot them AND see how test, train, true errors behave.

In [2]:

```
num_runs = 100
yhat_simple = np.zeros((num_runs,len(t)))
yhat_complex = np.zeros((num_runs,len(t)))
yhat_justright = np.zeros((num_runs,len(t)))

run=0
while run < num_runs:
    n = 40 # sample size
    x = np.random.uniform(-1,1,n) #sample input data
    x[0]=-1
    x[n-1]=1
    y = eta(x) + .03*np.random.randn(n)

    # degree-3 polynomial linear regression
    p=3
    X = np.vstack([np.ones(len(x)),x,x**2,x**3]).T
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = np.vstack([np.ones(len(t)),t,t**2,t**3]).T
    yhat_simple[int(run)] = np.matmul(T_,w)
    yh = np.matmul(X,w)
    X_ = np.vstack([np.ones(len(x_)),x_,x_**2,x_**3]).T
    y_h= np.matmul(X_,w)
    w_ = w

    # degree-5 polynomial linear regression
    p=5
    X = np.vstack([np.ones(len(x)),x,x**2,x**3,x**4,x**5]).T
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = np.vstack([np.ones(len(t)),t,t**2,t**3,t**4,t**5]).T
    yhat_justright[int(run)] = np.matmul(T_,w)
    yh = np.matmul(X,w)
    X_ = np.vstack([np.ones(len(x_)),x_,x_**2,x_**3,x_**4,x_**5]).T
    y_h= np.matmul(X_,w)
    w_ = w

    # degree-15 polynomial linear regression
    p=15
    X = np.vstack([np.ones(len(x)),x,x**2,x**3,x**4,x**5,x**6,x**7,x**8,x**9,x**10,x**11,x**12,x**13,x**14,x**15]).T
    w = np.array(p+1)
    w = np.matmul(np.matmul(np.linalg.inv(np.matmul(X.T,X)),X.T),y)
    T_ = np.vstack([np.ones(len(t)),t,t**2,t**3,t**4,t**5,t**6,t**7,t**8,t**9,t**10,t**11,t**12,t**13,t**14,t**15]).
    yhat_complex[int(run)] = np.matmul(T_,w)
    yh1 = np.matmul(X,w)
    X_ = np.vstack([np.ones(len(x_)),x_,x_**2,x_**3,x_**4,x_**5,x_**6,x_**7,x_**8,x_**9,x_**10,x_**11,x_**12,x_**13,
    y_h1= np.matmul(X_,w)

    run=run+1

def build_plot(yhat, title):
    ave_fit = np.zeros(np.size(t))
    for irun in range(1, num_runs):
        plt.plot(t,yhat[int(irun)], color='tab:orange',alpha=0.3)
        ave_fit = ave_fit + yhat[int(irun)]
    plt.plot(t,yhat[0],'tab:orange',alpha=0.3,label='Individual fits')
    plt.plot(t,y0,'r-', linewidth=2, label='Truth')
    plt.plot(t,(1/float(num_runs))*ave_fit, color='teal', linewidth=2,label='Average fit')
    plt.legend()
    plt.title(title)
    axes = plt.gca()
    axes.set_ylim([-0.4,0.2])

fig=plt.figure(figsize=(12, 6), dpi= 80, facecolor='w', edgecolor='k')
plt.subplot(1,3,1)
build_plot(yhat_simple, 'Degree 3')
plt.subplot(1,3,2)
build_plot(yhat_justright, 'Degree 5')
plt.subplot(1,3,3)
build_plot(yhat_complex, 'Degree 15')
plt.show()
```



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 (bias + variance) of each individual fit is high.
- Degree 15 has very low bias because the teal line is almost equal to the red (because 15 > 5). But each individual fit varies a lot about its average teal line indicating high variance. The overall error (bias + 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 (bias + variance) of each individual fit is low.