Bias-Variance Tradeoff

UNIVERSITY of WASHINGTON

Optimal Prediction

Goal: Predict $Y \in \mathbb{R}^d$ **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 $Y \in \mathbb{R}^d$ **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 $\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x]$)

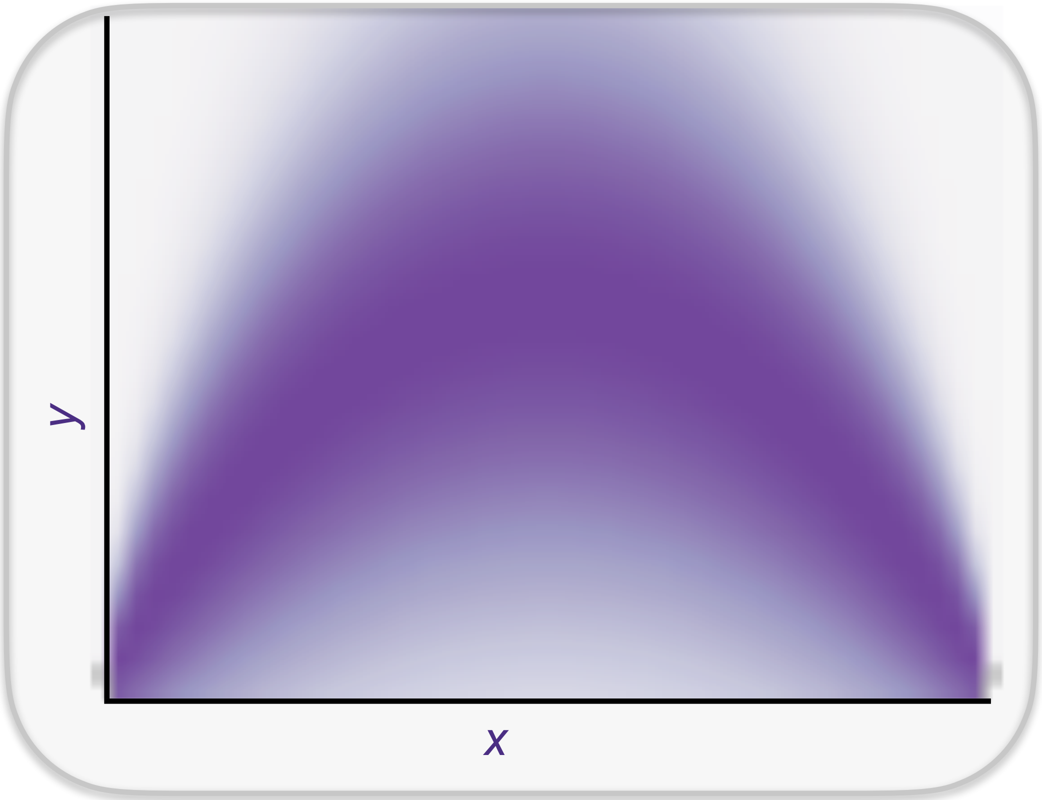
$$\begin{aligned} 0 &= \frac{d}{dc_x}\mathbb{E}_{Y|X}[(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[\frac{d}{dc_x}(Y - c_x)^2 | X = x] \\ &= \mathbb{E}_{Y|X}[-2(Y - c_x) | X = x] = -2\mathbb{E}_{Y|X}[Y | X = x] + 2c_x \end{aligned}$$

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

Statistical Learning

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

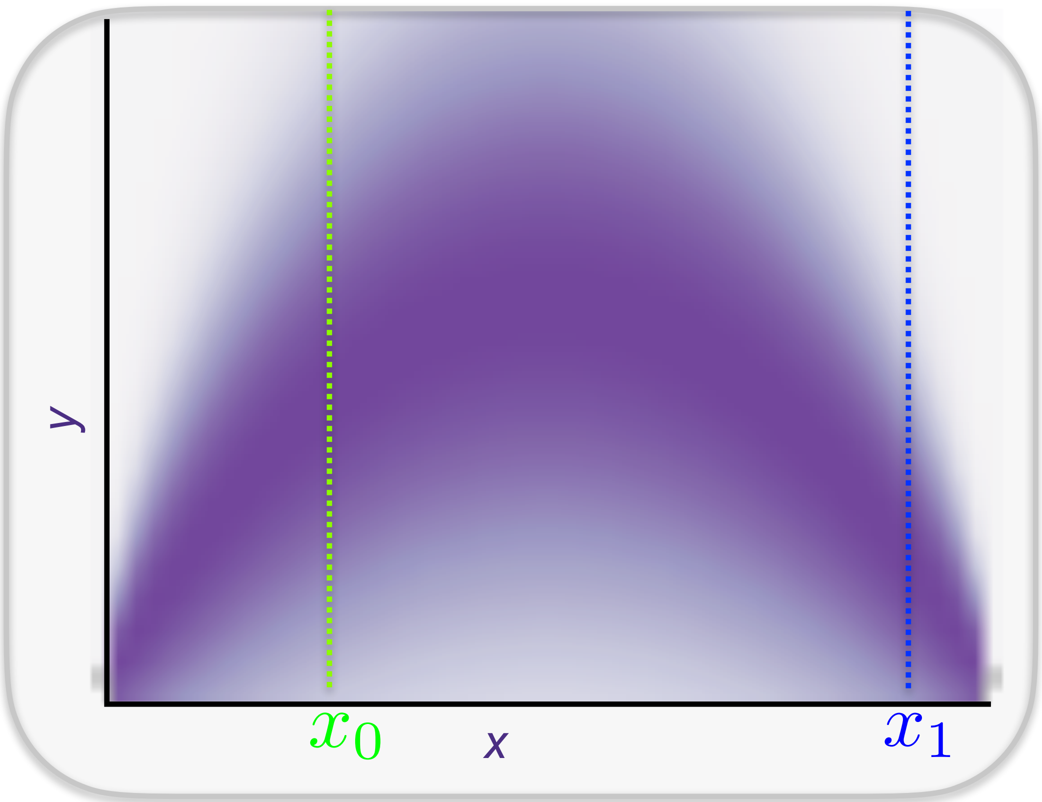
$$P_{XY}(X = x, Y = y)$$



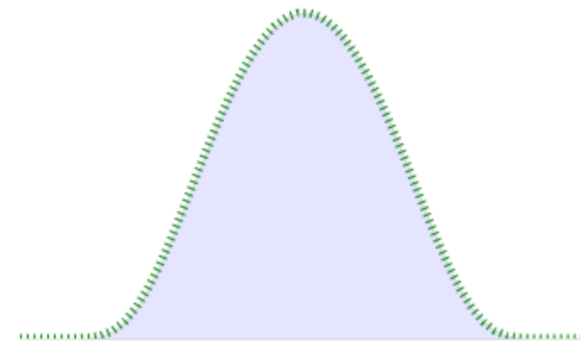
Statistical Learning

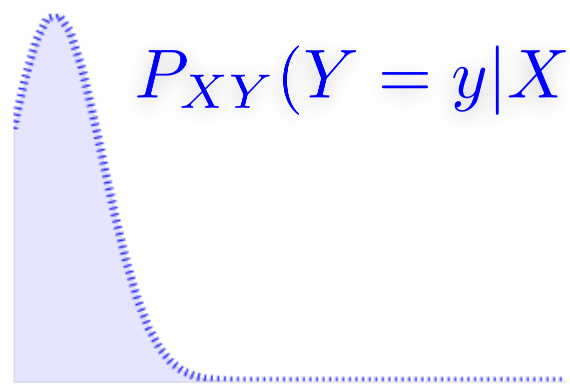
$$\mathbb{E}_{XY}[(Y - \eta(X))^2]$$

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



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



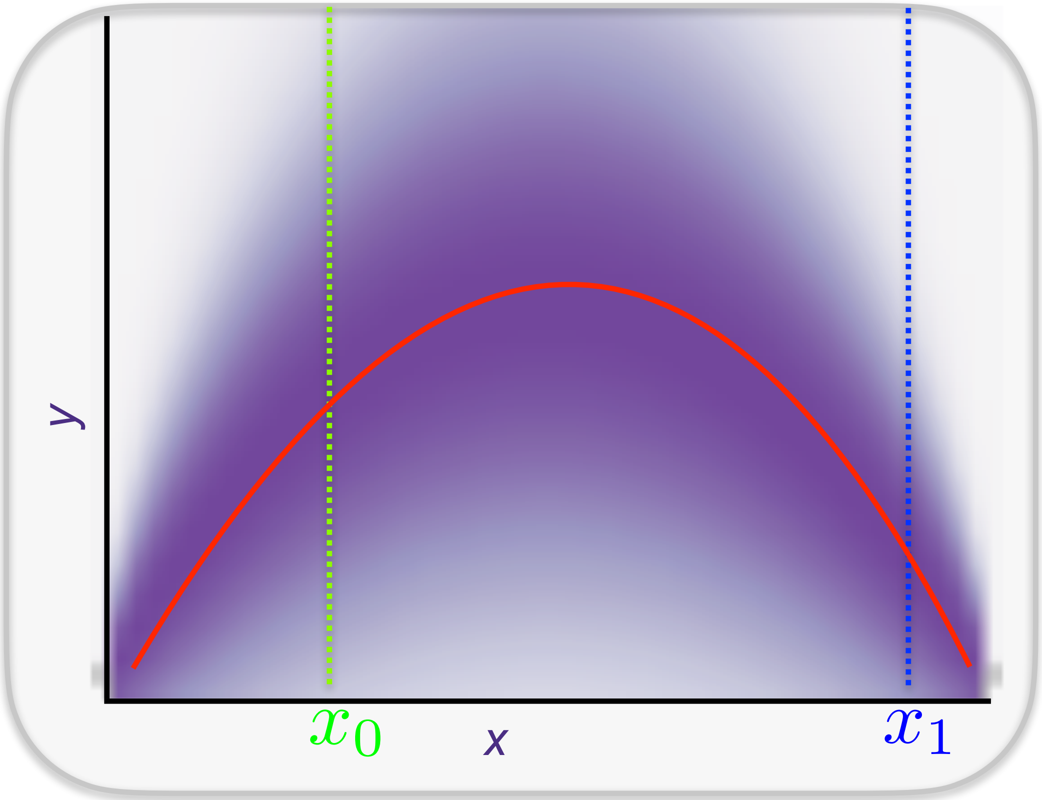


$$P_{XY}(Y = y | X = x_1)$$

Statistical Learning

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

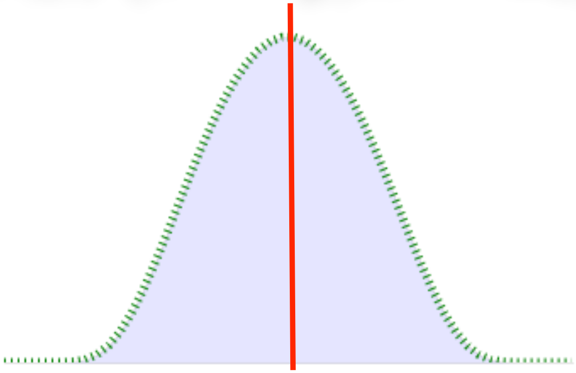
$P_{XY}(X = x, Y = y)$



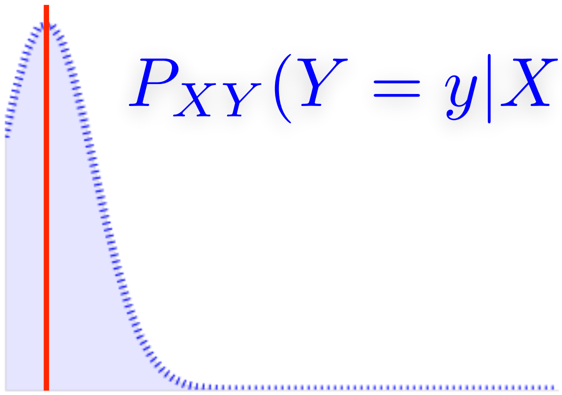
Ideally, we want to find:

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

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

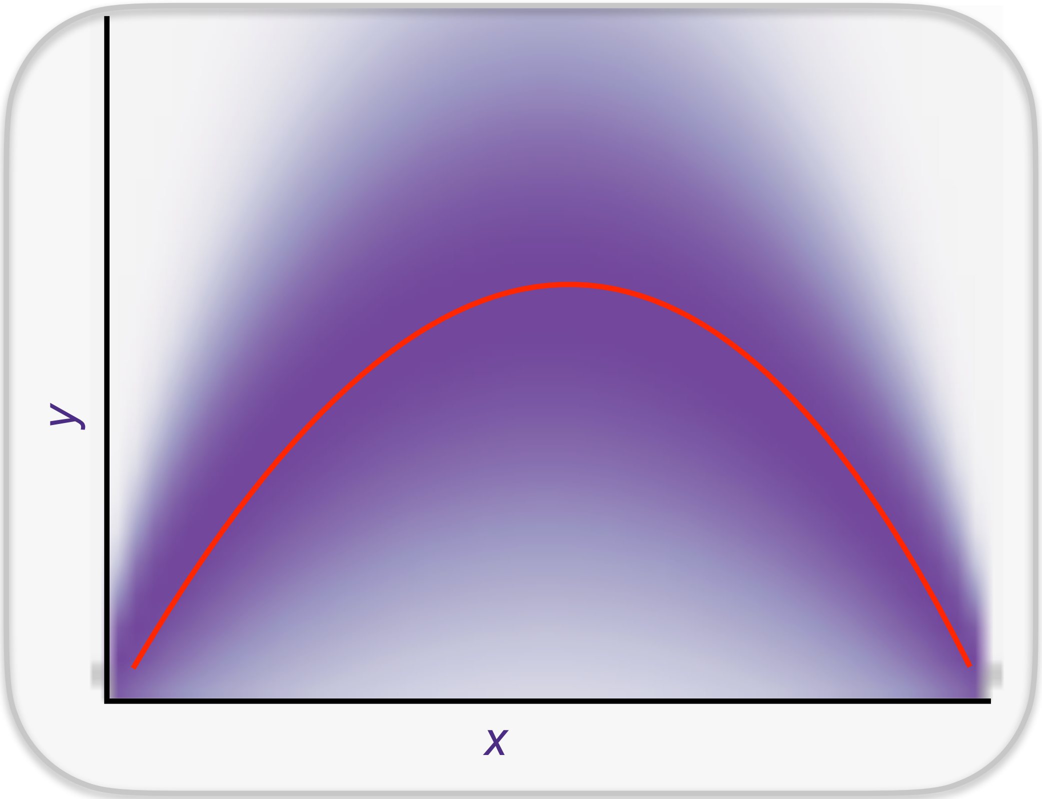


$P_{XY}(Y = y|X = x_1)$



Statistical Learning

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

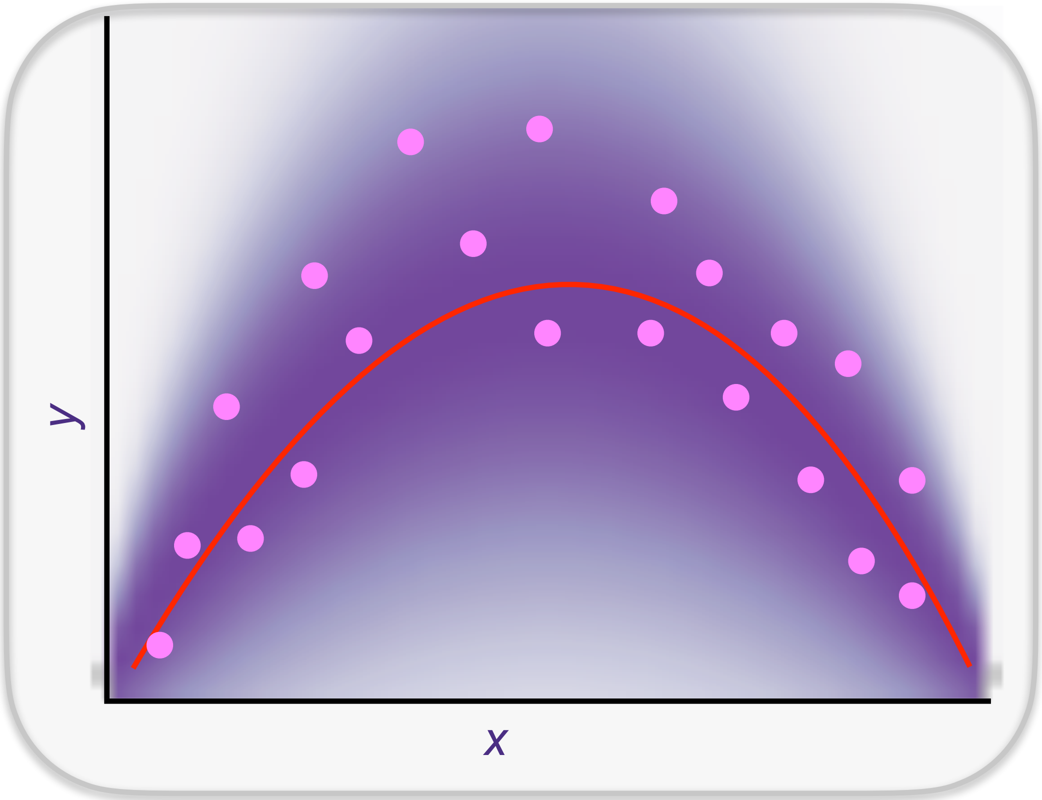


Ideally, we want to find:

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

Statistical Learning

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



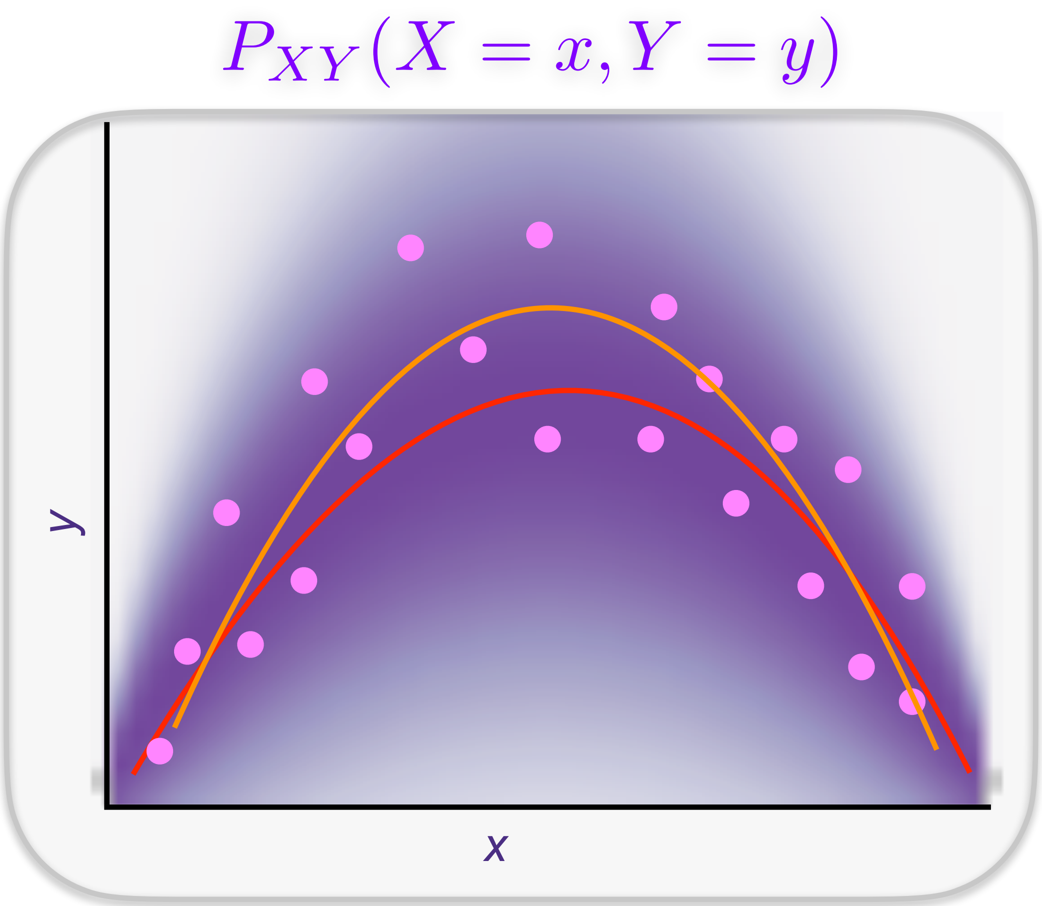
Ideally, we want to find:

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

But we only have samples:

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

Statistical Learning



Ideally, we want to find:

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

But we only have samples:

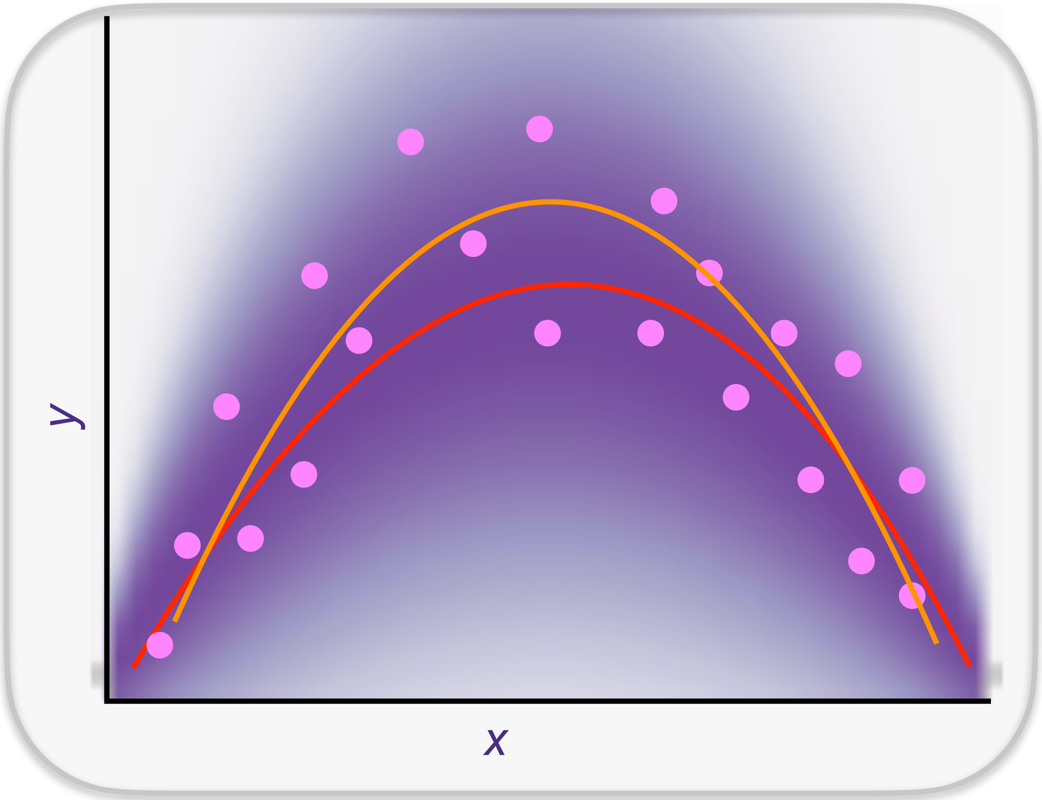
$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY}$ for $i = 1, \ldots, n$

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 $i = 1, \dots, n$

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 $i = 1, \ldots, n$

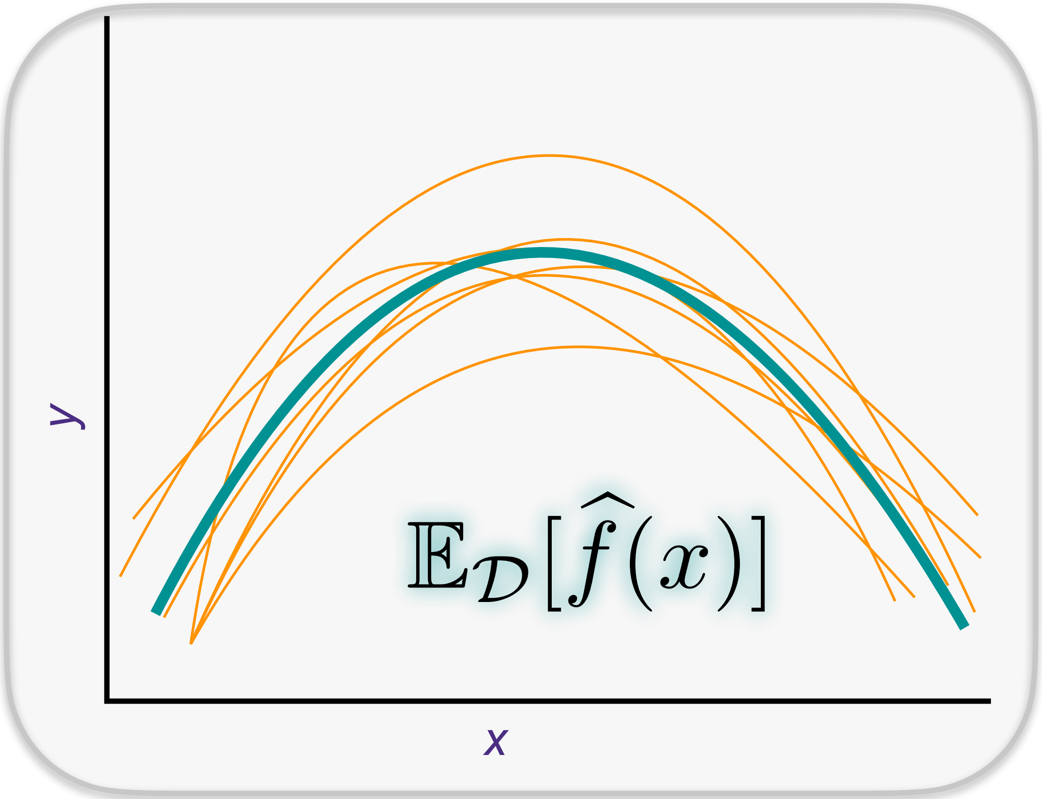
and are restricted to a function class (e.g., linear) so we compute:

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

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 $\widehat{f}$

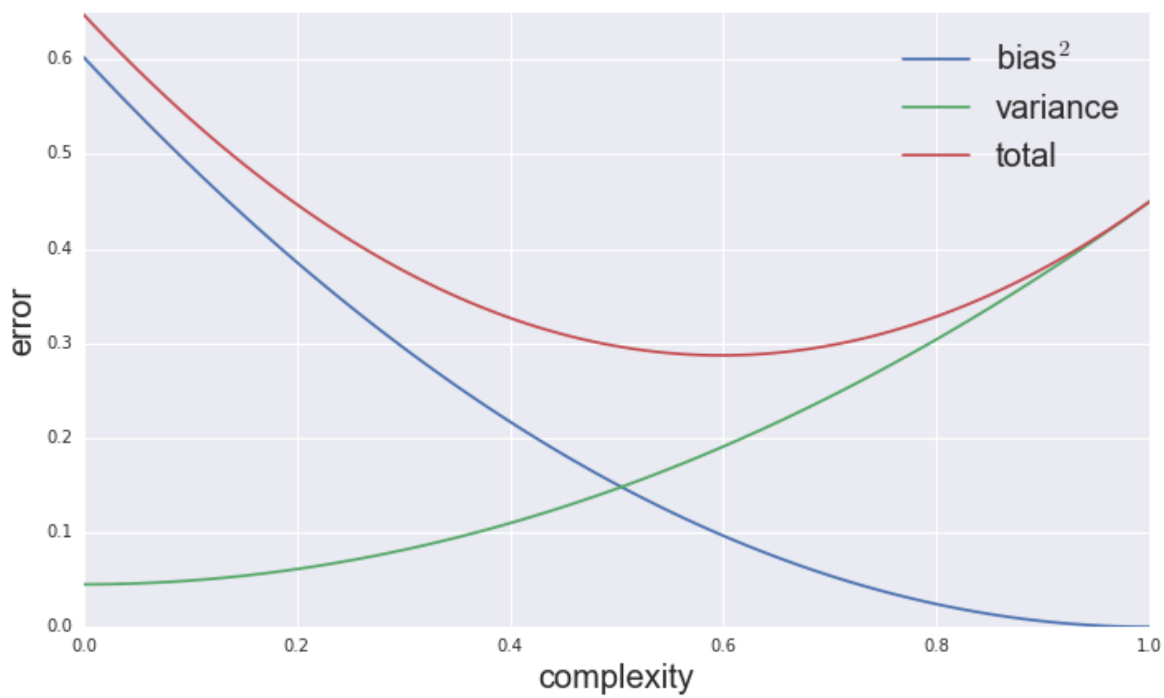
Bias-Variance Tradeoff

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

irreducible error

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

biased squared **variance**



Cross-Validation

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 train on the test data

(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 $f_{D\backslash j}$ with *D*\j dataset**
- **Estimate true error** as squared error on predicting **y_j**:
  - **Unbiased estimate of $error_{true}(f_{D\backslash j})$!**

(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 $f_{D\backslash j}$ with *D*\j dataset**
- **Estimate true error** as squared error on predicting **y_j**:
  - **Unbiased estimate of $\text{error}_{\text{true}}(f_{D\backslash j})$!**

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

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

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

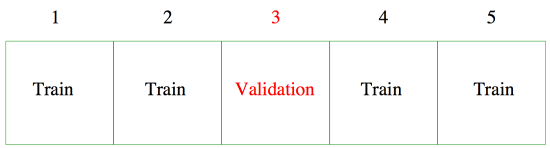
- LOO is almost unbiased! Use LOO error for model selection!!!
  - E.g., picking degree

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 $f_{D \backslash D_i}$ using data point not in D_i
  - Estimate error of $f_{D \backslash D_i}$ 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**
  - **$D_1,\ldots,D_k$**

1	2	3	4	5
Train	Train	Validation	Train	Train

- **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:**

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

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

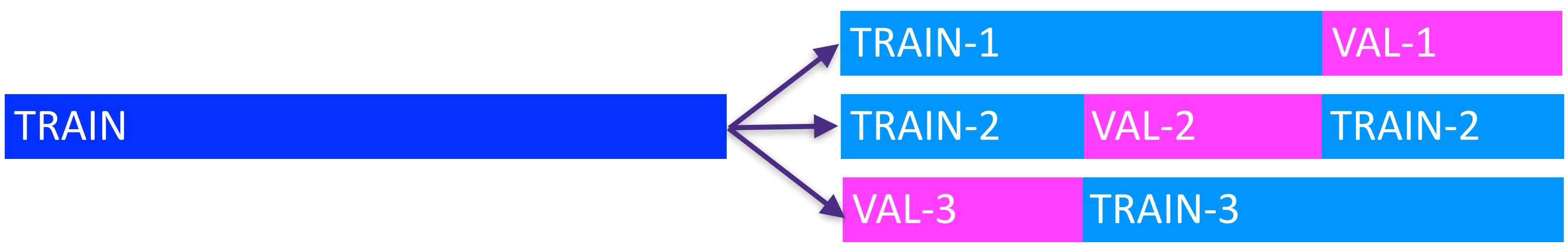
$$error_{k-fold} = \frac{1}{k} \sum_{i=1}^{k} error_{\mathcal{D}_i}$$

- **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