Outline

1. Review: Non-linear Basis Functions
2. Overfitting and Regularization
3. Hyperparameter Tuning and Cross-Validation
4. Bias-Variance Trade-off

Review: Non-linear Basis Functions

General Nonlinear Basis Functions

We can use a nonlinear mapping to a new feature vector:

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{z} \in \mathbb{R}^M$$

- M is dimensionality of new features $\boldsymbol{z}$ (or $\boldsymbol{\phi}(\boldsymbol{x})$)
- M could be greater than, less than, or equal to D

We can apply existing learning methods on the transformed data:

- linear methods: prediction is based on $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$

Regression with Nonlinear Basis

Residual sum of squares

$$\sum_n [\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n]^2$$

where $\boldsymbol{w} \in \mathbb{R}^M$, the same dimensionality as the transformed features $\boldsymbol{\phi}(\boldsymbol{x})$.

The LMS solution can be formulated with the new design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^\top \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^\top \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \boldsymbol{w}^{\text{LMS}} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$$

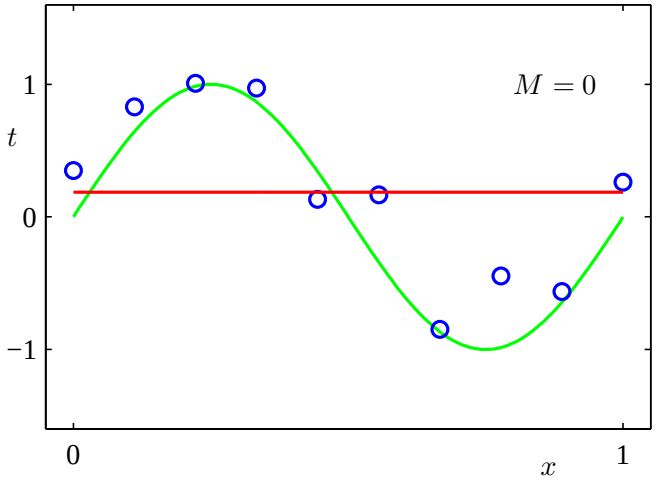
Non-linear Basis Functions: Polynomial Regression

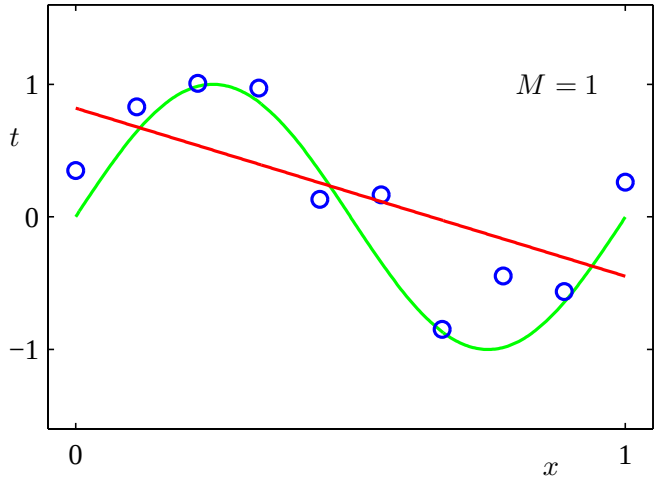
Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m$$

Fitting samples from a sine function:

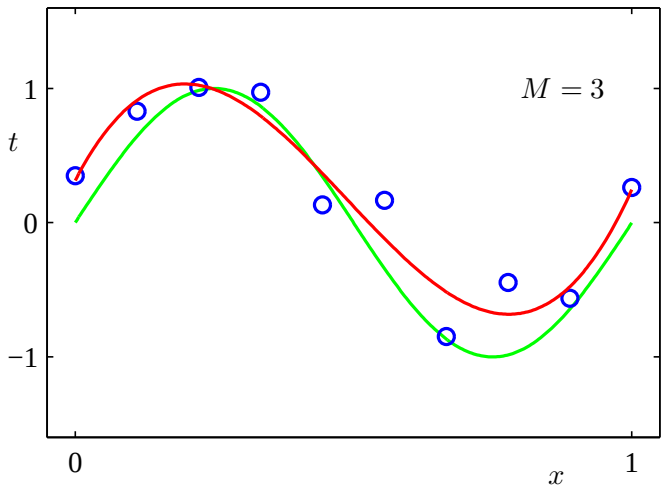
underfitting since $f(x)$ is too simple



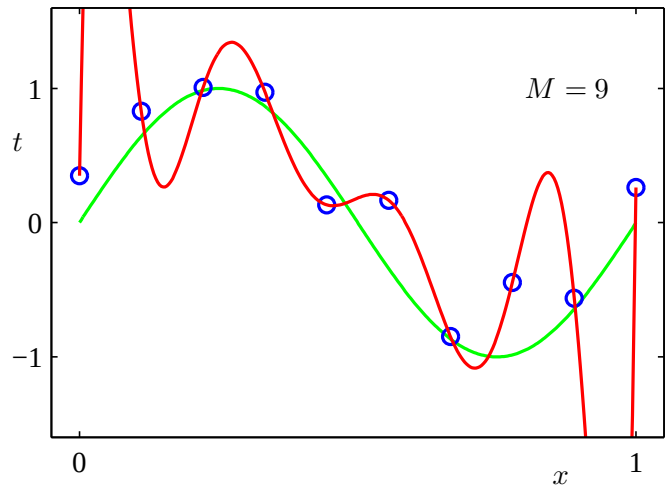


Adding Higher-order Terms

M=3



M=9: overfitting

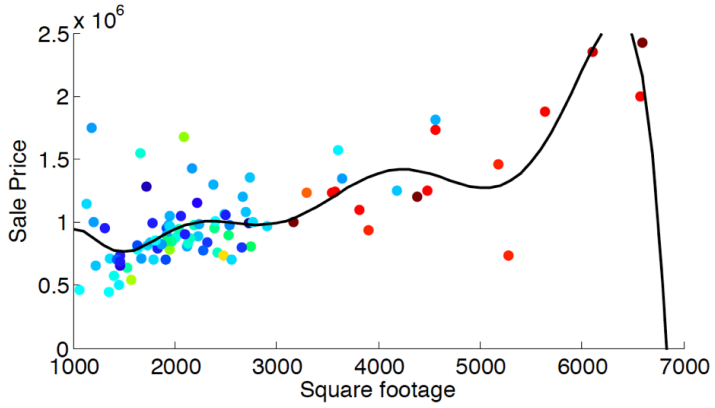


More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!

Overfitting and Regularization

Overfitting Can Be Quite Disastrous

Fitting the housing price data with large M:



Predicted price goes to zero (and is ultimately negative) if you buy a big enough house!

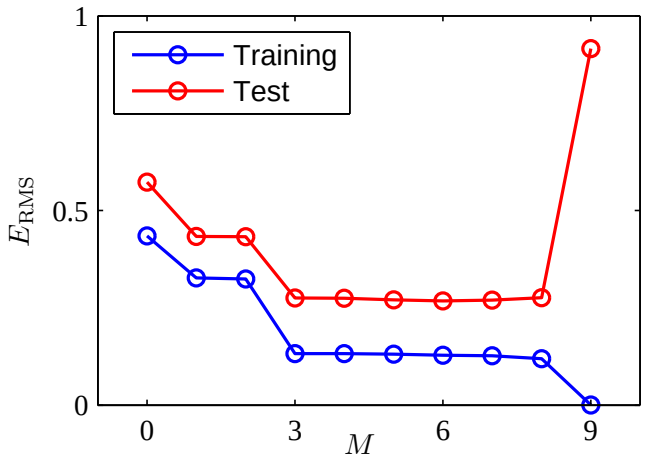
This is called poor generalization/overfitting.

Detecting Overfitting

Plot model complexity versus objective function:

- X axis: model complexity, e.g., M
- Y axis: error, e.g., RSS, RMS (square root of RSS), 0-1 loss

Compute the objective on a training (used to train the model) and test (used to evaluate the model) dataset.

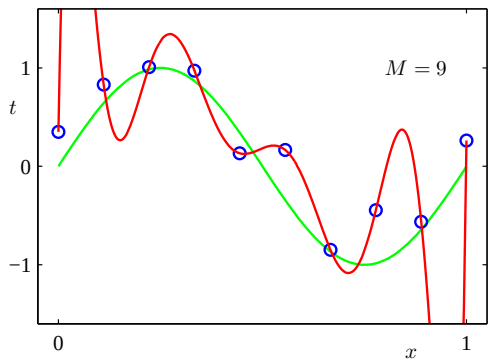


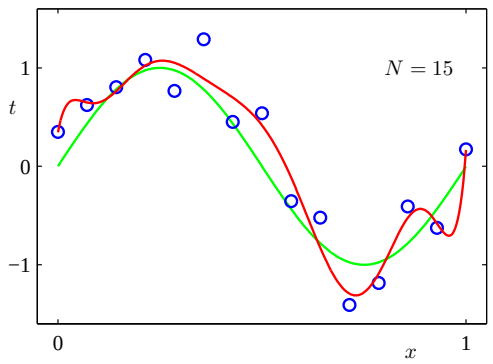
As a model increases in complexity:

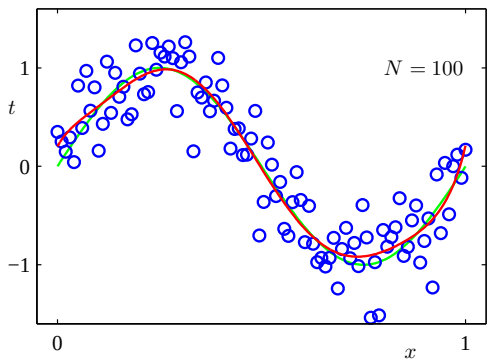
- Training error keeps reducing
- Test error may first reduce but eventually increase

Preventing Overfitting: Option 1

Try to use more training data



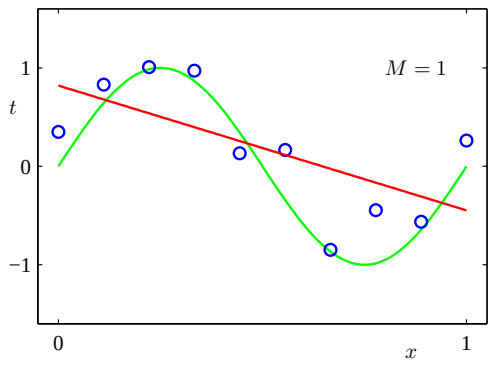


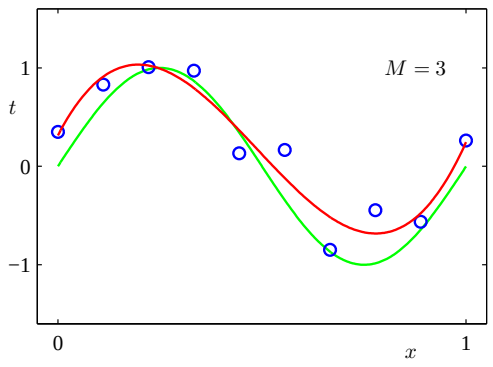


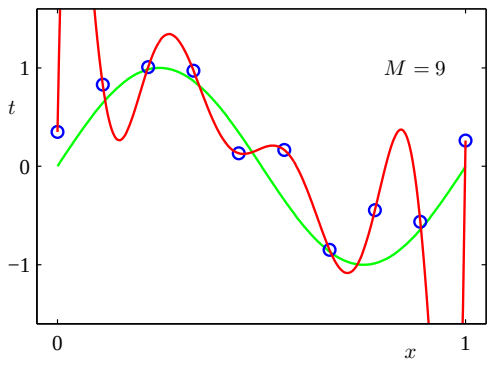
But getting a lot of data can be expensive and time-consuming

Preventing Overfitting: Option 2

Reduce the Number of Features







Q: how to select the right M? - will talk about that later.

Preventing Overfitting: Option 3

Regularization Methods: Give preference to 'simpler' models

- How do we define a simple linear regression model — $\boldsymbol{w}^\top \boldsymbol{x}$?

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

- Intuitively, the weights corresponding to higher order terms should not be "too large"

Regularization Methods

Add a term to the objective function.
Choose the parameters to not just minimize risk, but avoid being large.

$$\frac{1}{2}\|\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}\|_2^2+\frac{1}{2}\lambda\|w\|_2^2$$

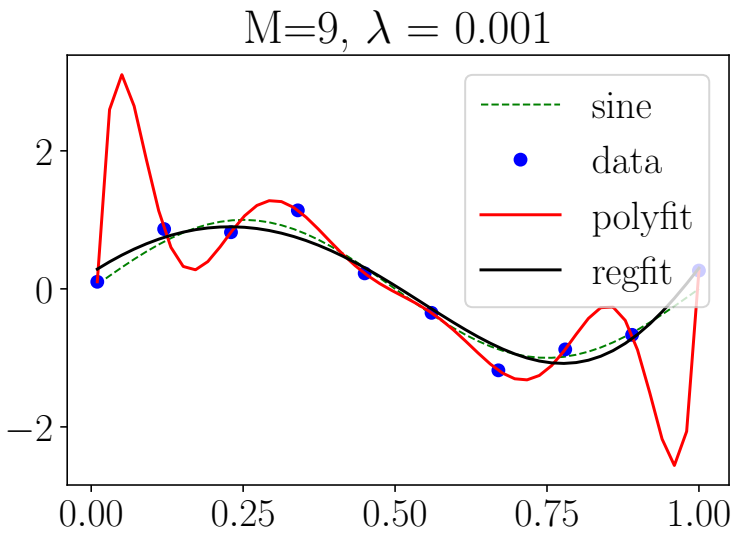
Ridge regression is just regularized linear regression.

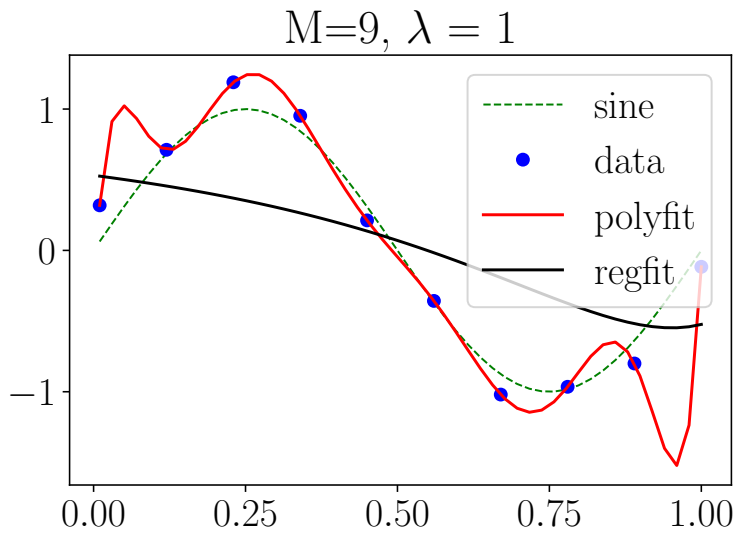
Advantages

- Forces the magnitude of $\boldsymbol{w}$ to be small
- Alleviates overfitting and generalizes well to new data points

Example: Effect of Regularization

As λ increases, the overfitting issue is alleviated.





How to choose λ? Up-next: hyperparameter tuning

Hyperparameter Tuning and Cross-Validation

How Much Regularization Do We Need?

Can we tune λ on the training dataset?
i.e. try different λ and choose the one with the smallest training error.

No: In the overfitting example, we would pick $\lambda = 0$ as it already has ZERO training error. This defeats our intention of alleviating overfitting.

To tune λ,

- We can use a validation set or do cross validation.
- Pick the value of λ that yields lowest error on the validation dataset.

Similar idea applies to tuning degree of polynomial M, learning rate η (or any other hyperparameter) as well.

Tuning with a Validation Dataset

Training data are used to learn $f(\cdot)$ (our model).
N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$

Test data are used to assess the prediction error.

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Validation data are used to optimize hyperparameter(s).
L samples/instances: $\mathcal{D}^{\text{VAL}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{L}}, y_{\text{L}})\}$

Training, validation and test data should not overlap!

Recipe

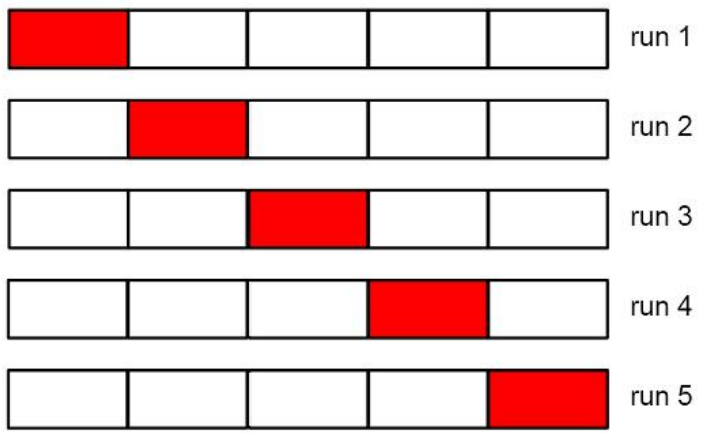
- For each possible value of the hyperparameter (say $\lambda = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{VAL}}$
- Choose the λ with the best performance on $\mathcal{D}^{\text{VAL}}$
- Evaluate this model on $\mathcal{D}^{\text{TEST}}$ to get the final prediction error

Cross-validation

What if we do not have validation data?

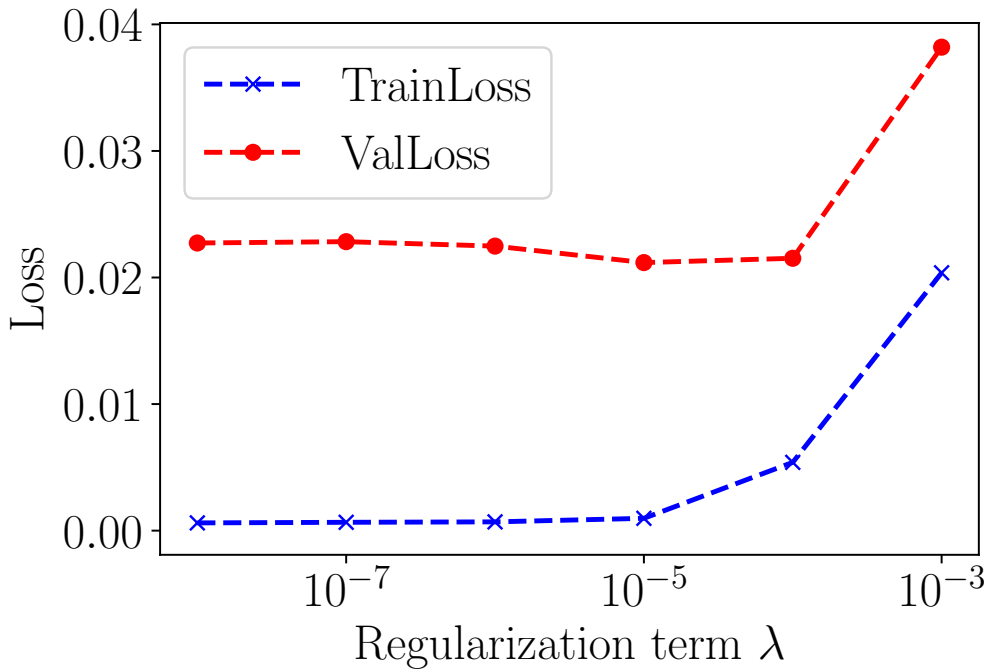
- We split the training data into S equal parts.
- We use each part in turn as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that the model performs the best over S trials (based on average, variance, etc.)
- Re-train with this hyperparameter on the entire training dataset.

Figure 1: S = 5: 5-fold cross validation



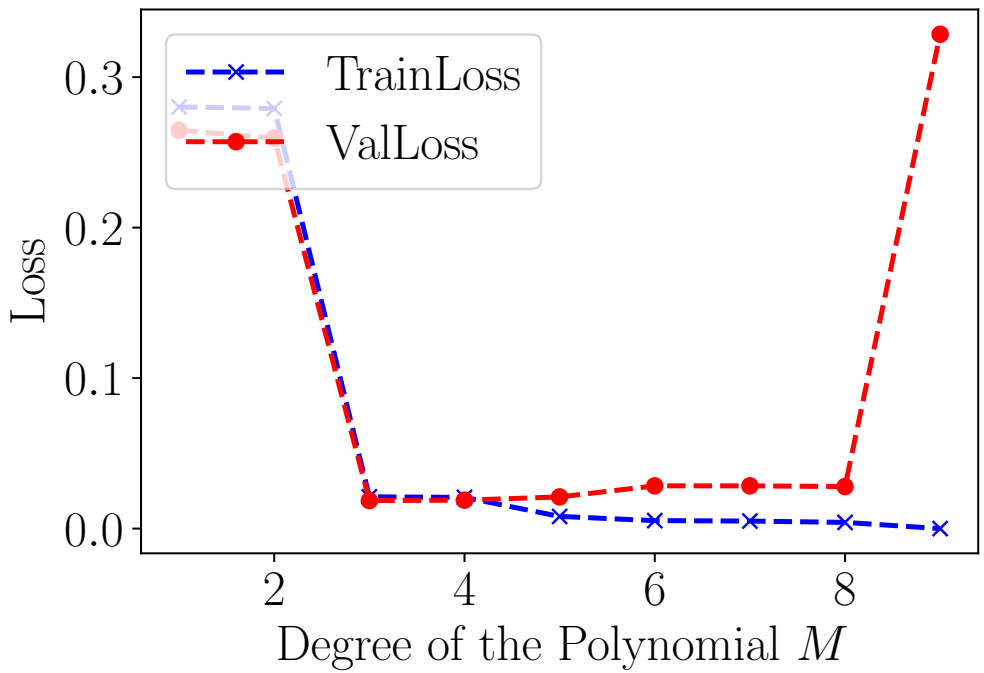
Example: Hyper-parameter Tuning λ

- $\lambda = 10^{-4}$ gives the smallest validation loss
- Strikes a balance between over- and under-fitting



Example: Hyper-parameter Tuning M

- Considering polynomial regression without regularization
- $M = 3$ or $M = 4$ gives the smallest validation loss
- Strikes a balance between over- and under-fitting



Bias-Variance Trade-off

Empirical Risk Minimization

Supervised learning

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a loss

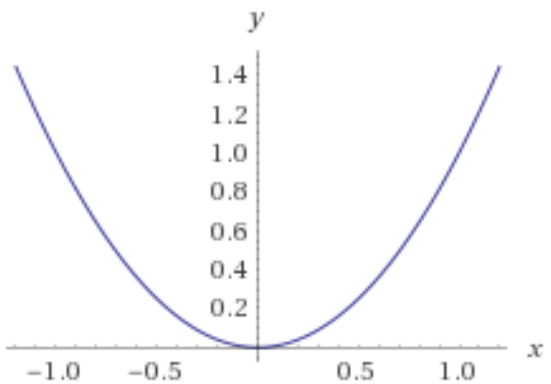
$$\ell(h(\boldsymbol{x}), y)$$

Example:

Quadratic loss function for regression when y is continuous:

$$\ell(h(\boldsymbol{x}), y) = [h(\boldsymbol{x}) - y]^2$$

Ex: linear regression



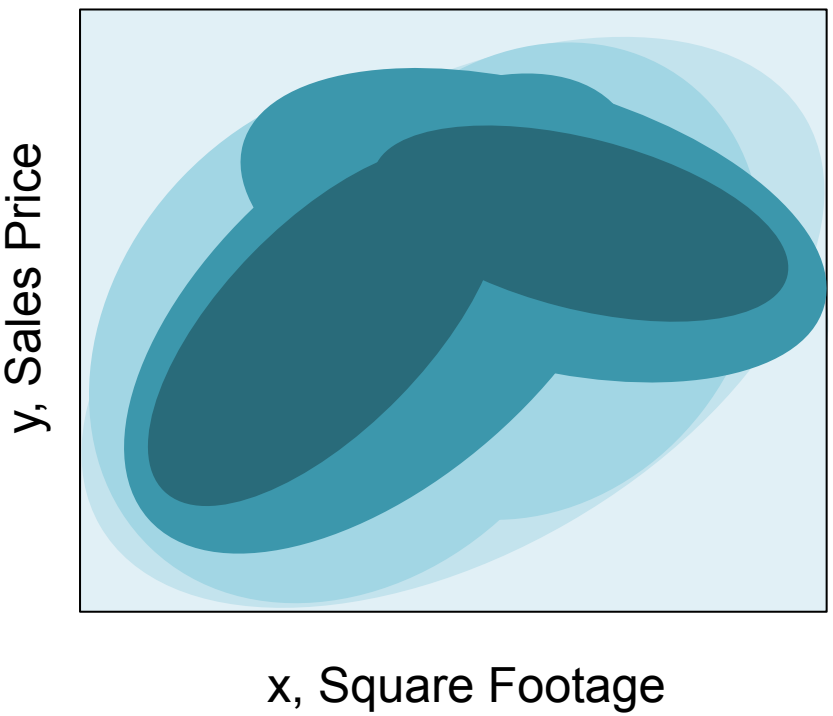
Bias-Variance Trade-off: Illustration

How to evaluate $\ell(h(\boldsymbol{x}), y)$ "generally" over $(\boldsymbol{x}, y)$?

Using "true" data joint distribution $p(\boldsymbol{x}, y)$.

Example:

- Joint distribution of square footage x and house sales price y
- Darker color indicates higher probability regions



How Good Is Our Predictor?

Risk:
Given the true distribution of data $p(\boldsymbol{x}, y)$, the risk of a given predictor $h(\boldsymbol{x})$ is its expected loss ℓ:

$$R[h(\boldsymbol{x})] = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} d\, y$$

However, we cannot compute $R[h(\boldsymbol{x})]$ (we do not know p), so we use the empirical risk, given a training dataset $\mathcal{D}$:

$$R^{\mathrm{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_{n} \ell(h(\boldsymbol{x}_n), y_n)$$

Intuitively, as $N \to +\infty$,

$$R^{\mathrm{EMP}}[h(\boldsymbol{x})] \to R[h(\boldsymbol{x})]$$

Connection to Underfitting/Overfitting

Error decomposes into 3 terms

$$\text{Expected Risk} = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

And underfitting/overfitting can be interpreted as **bias/variance** tradeoff.

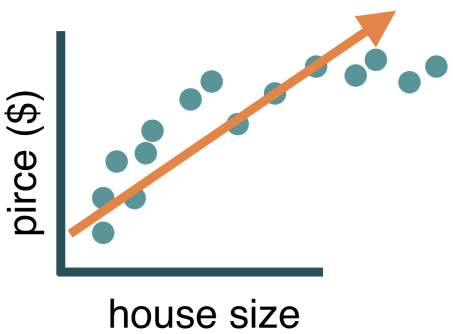


Figure 2: High Bias

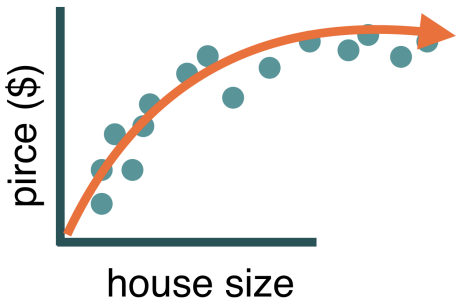


Figure 3: Just Right

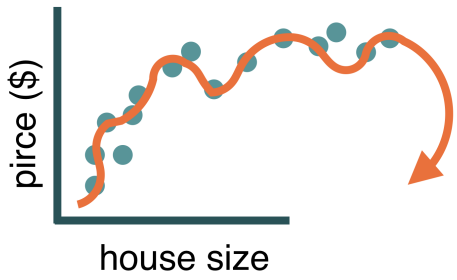


Figure 4: High Variance

Next: Set up the notations needed to state and derive the decomposition.

Bias-Variance Tradeoff for Regression

- $\mathcal{D}$: our training data
- $\ell(h(\boldsymbol{x}), y)$: our square loss function for regression

$$\ell(h(\boldsymbol{x}), y) = [h(\boldsymbol{x}) - y]^2$$

- $h_{\mathcal{D}}(\boldsymbol{x})$: our prediction function learned from training data $\mathcal{D}$
  - We are using the subscript $\mathcal{D}$ to indicate that the prediction function is learned on the specific set of training data $\mathcal{D}$
  - $h_{\mathcal{D}}(\boldsymbol{x})$ is the model within a model class (e.g. linear, degree M polynomial) that minimizes the training loss, e.g. the (regularized) empirical risk based on $\mathcal{D}$.
- Unknown joint distribution $p(\boldsymbol{x}, y)$
- Risk of $h_{\mathcal{D}}$: $R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$

The Effect of Finite Training Samples

Every training sample $\mathcal{D}$ is a sample from the following joint distribution of all possible training datasets

$$\mathcal{D} \sim P(\mathcal{D}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n)$$

Thus, the prediction function $h_{\mathcal{D}}(\boldsymbol{x})$ is a random function with respect to this distribution of possible training datasets. So is also its risk

$$R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

We will now evaluate the expected risk $\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})]$: the average risk over the distribution of possible training datasets, $P(\mathcal{D})$.

Bias-Variance Trade-off: Intuition

Error decomposes into 3 terms

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

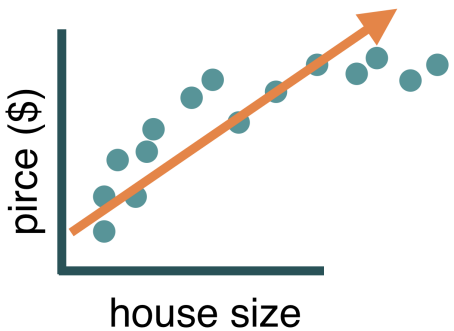


Figure 5: High Bias

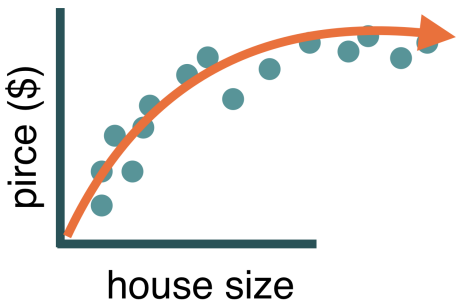


Figure 6: Just Right

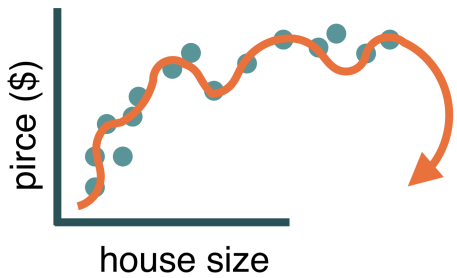


Figure 7: High Variance

We will prove this result, and interpret what it means...

Warning: The next few slides are somewhat mathematical – please review them carefully after class.

Average over the Distribution of the Training Data

Expected risk

$$\mathbb{E}_{\mathcal{D}}\left[R[h_{\mathcal{D}}(\boldsymbol{x})]\right] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Namely, the randomness with respect to $\mathcal{D}$ is marginalized out.

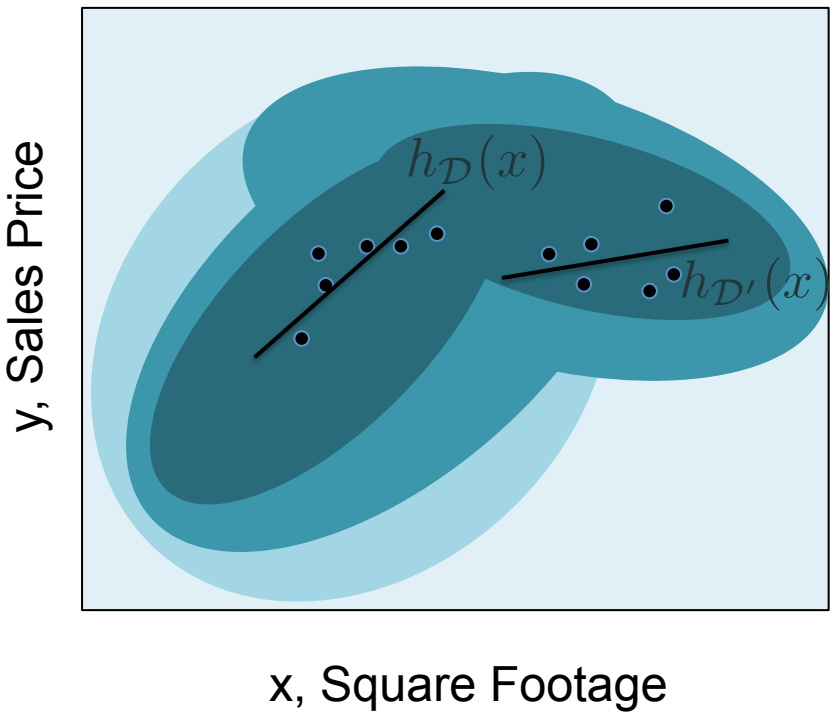
Averaged prediction

$$\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\boldsymbol{x})\right] = \int_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) P(\mathcal{D}) d\mathcal{D}$$

Namely, if we have seen many training datasets, we predict with the average of the prediction functions learned on each training dataset.

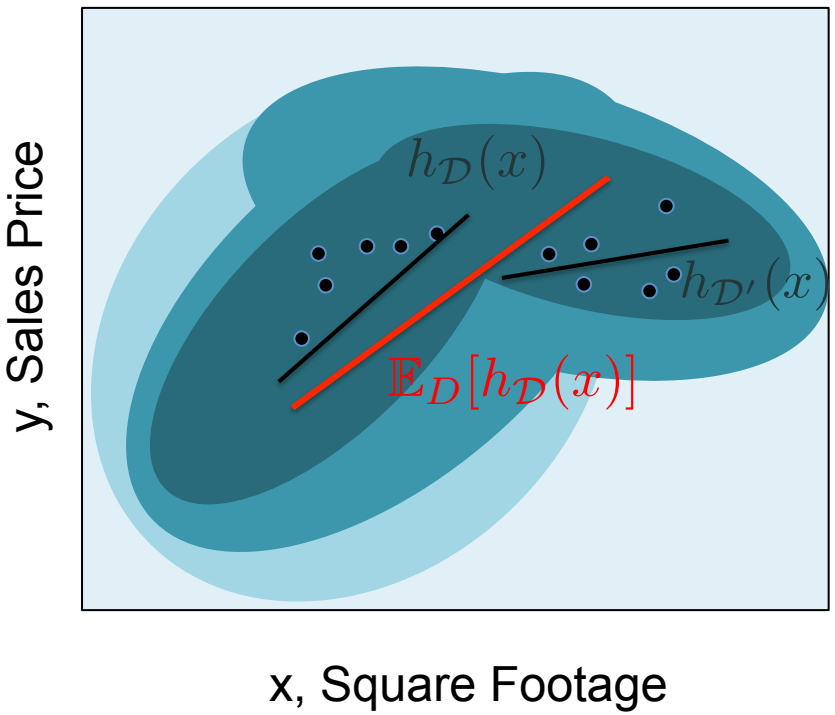
Bias-Variance Trade-off: Illustration

- Learning the model for a different dataset $\mathcal{D}'$ yields a new linear model $h_{\mathcal{D}'}(x)$
- Average of such models over infinitely many datasets sampled from the joint distribution is $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$



Bias-Variance Trade-off: Illustration

- Learning the model for a different dataset $\mathcal{D}'$ yields a new linear model $h_{\mathcal{D}'}(x)$
- Average of such models over infinitely many datasets sampled from the joint distribution is $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$



Variance

We will subtract the averaged prediction from the averaged risk

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] &= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D} \\
&= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \\
&\qquad + \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D} \\
&= \underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE}} \\
&\qquad + \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}
\end{aligned}$$

Where Does the Cross-term Go?

It is zero

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})][\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\boldsymbol{x}} \int_{y} \left\{ \int_{\mathcal{D}} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})] P(\mathcal{D}) d\mathcal{D} \right\} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \int_{\boldsymbol{x}} \int_{y} \left\{ \int_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) P(\mathcal{D}) d\mathcal{D} - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \right\} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$= 0 \leftarrow$ (the term within the braces vanishes, by definition)

Variance

We will subtract the averaged prediction from the averaged risk

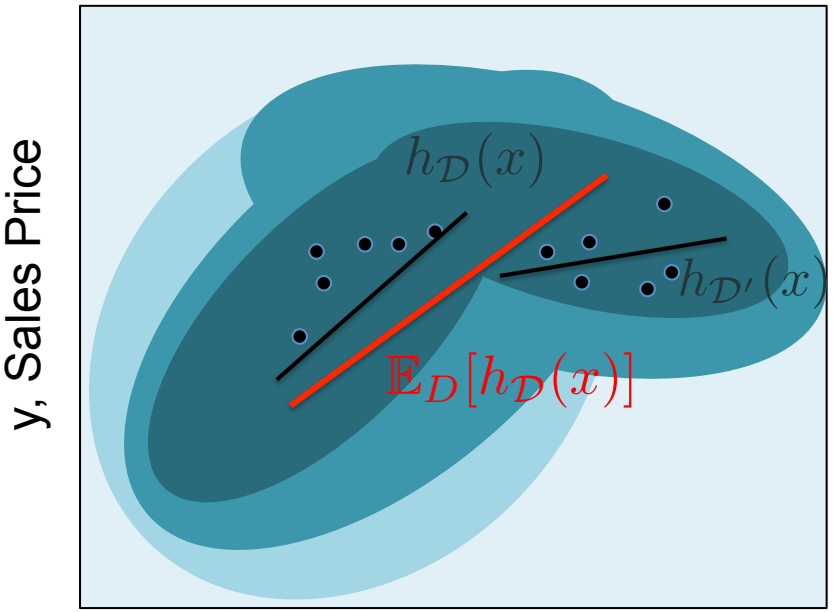
$$\begin{aligned}
\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] &= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D} \\
&= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \\
&\qquad + \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D} \\
&= \underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE}} \\
&\qquad + \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}
\end{aligned}$$

Now let's analyze the Variance term!

Bias-Variance Trade-off: Illustration

$$\underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE: error due to training dataset}}$$

- $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$: Average of such models over infinitely many datasets sampled from the joint distribution.
- Variance term captures how much individual models differ from the average



Analyzing the Variance

$$\underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE: error due to training dataset}}$$

- For each $(\boldsymbol{x}, y)$ pair, we compute the squared difference of $h_{\mathcal{D}}(\boldsymbol{x})$ (the prediction with training dataset $\mathcal{D}$) and the averaged prediction $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$.
- High variances correspond to **Overfitting!**

How can we reduce the variance?

- Use a lot of data (ie, increase the size of $\mathcal{D}$)
- Use a simple $h(\cdot)$ so that $h_{\mathcal{D}}(\boldsymbol{x})$ does not vary much across different training datasets. An extreme example is $h(\boldsymbol{x}) = 0$ (not sensitive to $\mathcal{D}$ at all).

The Remaining Item

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$
$$+ \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

The integrand has no dependency on $\mathcal{D}$ anymore and simplifies to

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

We will apply a similar add-and-subtract trick, by using an averaged target y (what we want to predict from $\boldsymbol{x}$):

$$\mathbb{E}_y[y|\boldsymbol{x}] = \int_y y p(y|\boldsymbol{x}) dy$$

Bias and Noise

Decompose again

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}] + \mathbb{E}_{y}[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2}$$

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{y}[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE}}$$

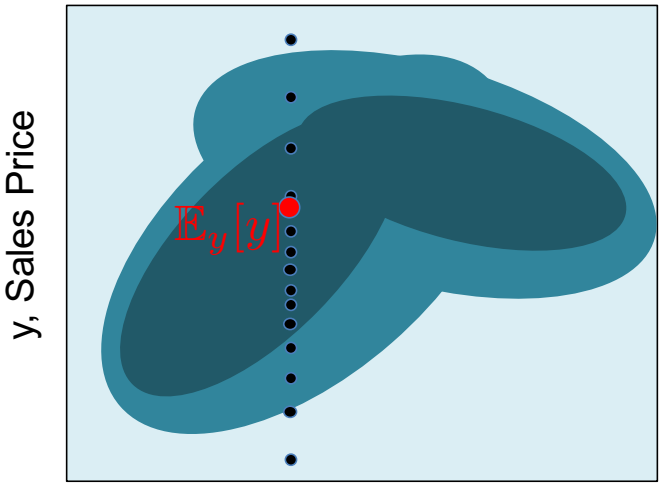
Where is the cross-term?

Take-home exercise: Show that it is zero

Bias-Variance Trade-off: Illustration

$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE: error due to randomness of } y}$$

- For a given $\boldsymbol{x}$, we have a conditional distribution $p(y|\boldsymbol{x})$: the Bayesian optimal prediction of the label value for a given $\boldsymbol{x}$ is $\mathbb{E}_y[y|\boldsymbol{x}]$;
- The noise term measures the inherent variance in labels y
- This term has nothing to do with our prediction $h_{\mathcal{D}}(\boldsymbol{x})$



Analyzing the Noise

$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE: error due to randomness of } y}$$

How can we reduce noise?

There is nothing we can do. This quantity depends on $p(\boldsymbol{x}, y)$ only; choosing $h(\cdot)$ or the training dataset $\mathcal{D}$ will not affect it.

Analyzing the Bias Term

$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2\text{: error due to the model approximation}}$$

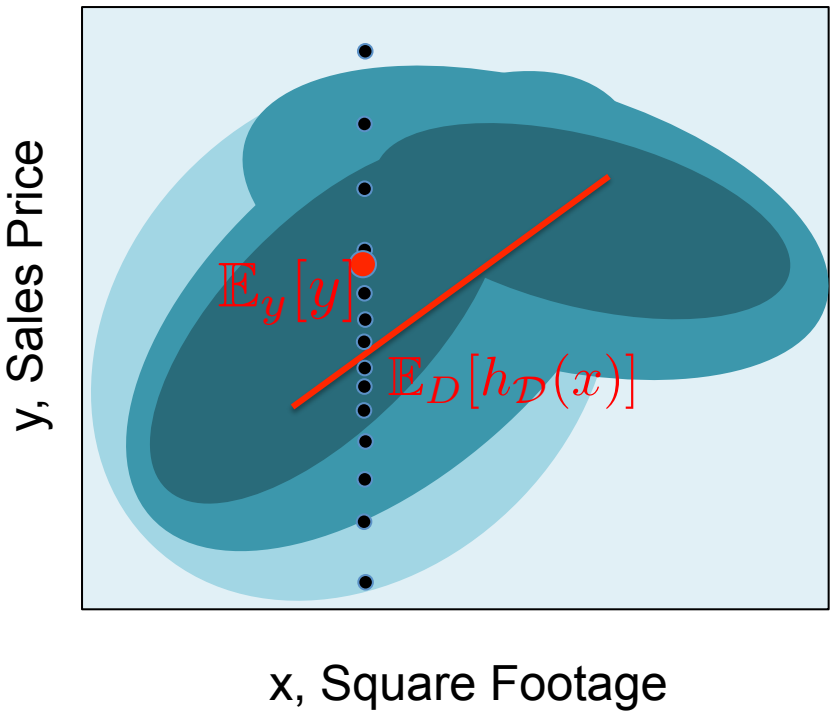
Understanding the bias

- For each $(\boldsymbol{x}, y)$ pair, we compute the loss of our averaged prediction $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ compared to the $\mathbb{E}[y|x]$, the optimal prediction.
- If our model class was rich enough (eg. a high-degree polynomial), then $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ should perfectly match $\mathbb{E}_{y}[y|\boldsymbol{x}]$
- Restricting to simpler models (eg. linear) results in a large bias, aka **underfitting**!

Analyzing the Bias Term

$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2\text{: error due to the model approximation}}$$

Restricting to simpler models (eg. linear) results in a large bias



Analyzing the Bias Term

How can we reduce the bias?

- It can be reduced by using more complex models. We shall choose $h(\cdot)$ to be as flexible as possible: the better $h(\cdot)$ approximates $\mathbb{E}_y[y|\boldsymbol{x}]$, the smaller the bias.
- However, this will increase the VARIANCE term.

Summary of Risk Components

The average risk (with quadratic loss) can be decomposed as:

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE: error due to training dataset}}$$

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2\text{: error due to the model approximation}}$$

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{y}[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE: error due to randomness of } y}$$

Here we define: $h_{\mathcal{D}}(\boldsymbol{x})$ as the output of the model trained on $\mathcal{D}$, $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ as the expectation of the model over all datasets $\mathcal{D}$, and $\mathbb{E}_{y}[y|\boldsymbol{x}]$ as the expected value of y.

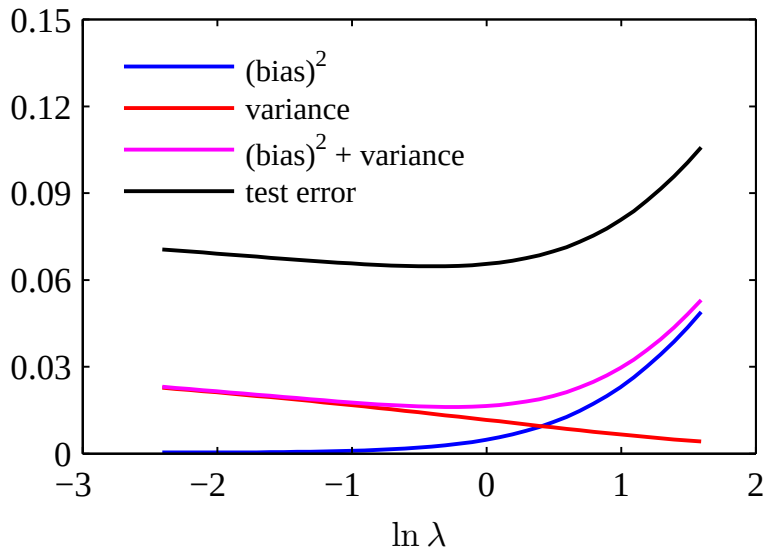
Bias-Variance Tradeoff

Written compactly, the risk decomposition is

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

where the first and the second term are inherently in conflict in terms of choosing what kind of $h(\boldsymbol{x})$ we should use (unless we have an infinite amount of data).

If we can compute all terms analytically, they will look like this



Summary for Today

- Overfitting and how to cope with it (more data, simpler model, regularization)
- Validation datasets (or cross-validation) are used to tune model hyperparameters.
- "Risk" framework for analyzing machine learning, and error decomposition into bias, variance, and noise terms.
  - Variance: Due to only optimizing over an empirical sample of the complete $(\boldsymbol{x}, y)$ distribution. High variance is responsible for **overfitting**.
  - Bias: Due to our choosing a model that does not fit the exact $(\boldsymbol{x}, y)$ relationship. High bias is responsible for **underfitting**.
  - Noise: Due to the output y's randomness with respect to the input $\boldsymbol{x}$.
- Choosing a more complex model improves the bias, but increases the variance (and vice versa for less complex models).
- The noise is independent of the model that we choose.