Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

Topic 2 Model Selection

PROF. LINDA SELLIE

Thanks to:

- Some of the material is from Prof. Sundeep Rangan
  - This includes some slides and the motivating examples
- Some slides (the slides with the green background) are from Yaser Abu-Mostafa

Learning objectives

- Understand how to create a more complex model using feature transformation
- Visually identity overfitting and undercutting of a model from a scatterplot
- Understand how overfitting and underfitting affect the in-sample and out of sample errors
- Understand the effect of bias/variance/noise in out of sample error
- Know how to compute generalization bound for classification
- Choose a model based on validation set
- Know how to use training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
- Explain the difference between (1) training error, (2) validation error, (3) cross-validation error, (4) test error, and (5) out of sample error
- Know the effect of L1 and L2 regularization and how to modify the objective function to use L1 or L2 regularization

Finding Parameters via Optimization
A general ML recipe

General ML problem

- Get data
- Pick a model with parameters
- Pick a loss function
  - Measures goodness of fit model to data
  - Function of the parameters
- Find parameters that minimizes loss

Multiple linear regression

1) Finding a way to have a more complex hypothesis class
2) If we have more than one hypothesis class to choose from - how do we select which one to use?

Loss function: $\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$

Select $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_d]^T$ to minimize $\text{RSS}(\mathbf{w})$

- In learning, our goal is to find a hypothesis that minimizes $E_{\text{out}}(g(\mathbf{x}))$ (not just $E_{\text{in}}(g(\mathbf{x}))$).
- In this lecture, we observe that choosing the model with the smallest training error doesn't work.
- Next, we explore the different types of errors we make.
- We have to find a way to compare models.

Outline

- ❑Motivating example
- ❑ Feature transformation
- ❑Underfitting and overfitting
- ❑Understanding error: Bias and variance and noise
- ❑Learning curves
- ❑validation and model selection
- ❑Model selection (with limited data)
- ❑ K-fold cross-validation
- ❑Regularization

How to create a more complex hypothesis

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

Outline

- ❑Motivating example
- ❑ Feature transformation
- ❑Underfitting and overfitting
- ❑Understanding error: Bias and varianc
- ❑Learning curves
- ❑validation and model selection
- ❑Model selection (with limit
- ❑ K-fold cross-validation
- ❑Regularization

Yea!

Uh oh….

How to create a more complex hypothesis

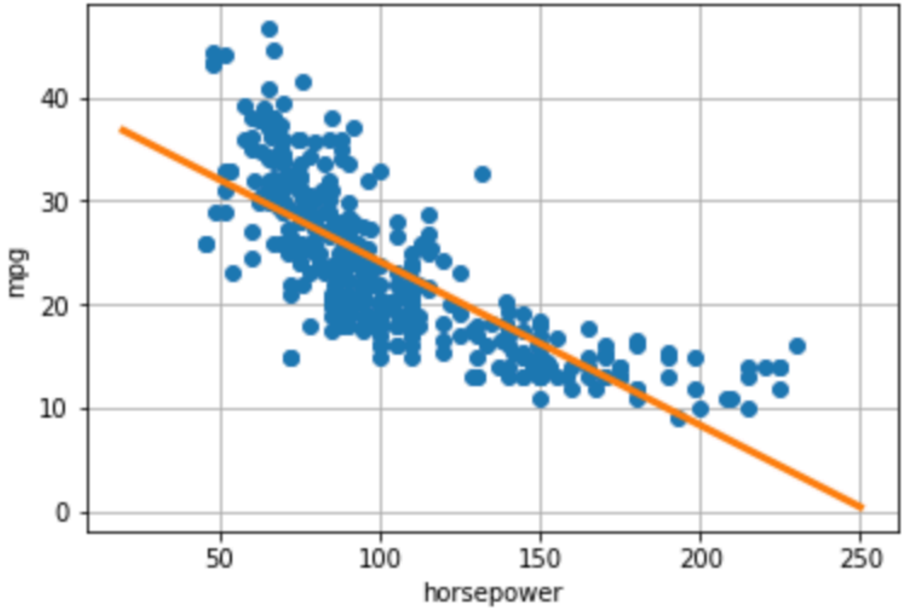
Understanding what went wrong

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

Our strategy

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

Estimating Automobile MPG



- Found best line/hyperplane

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

- Shape appear to be nonlinear…

- To reduce E_{in} (RSS) we need something non-linear…

How can we get a non-linear hypothesis ***easily***?

Outline

- Motivating example:
- Feature transformation
- Underfitting and overfitting
- Understanding error: Bias and variance and noise
- Learning curves
- Validation
- Validation and model selection
- Model selection (with limited data)

How to create a more complex hypothesis

Understanding where the error comes from and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

$g(\mathbf{x})$ is our hypothesis

If we have many different hypothesis classes to choose from - how can we choose wisely? And what is the error of the hypothesis we chose?

- Regularization

Slide added after lecture

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$$ Feature transformation $$\begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix}$$

Let $$\begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix} = \mathbf{z} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix}$$

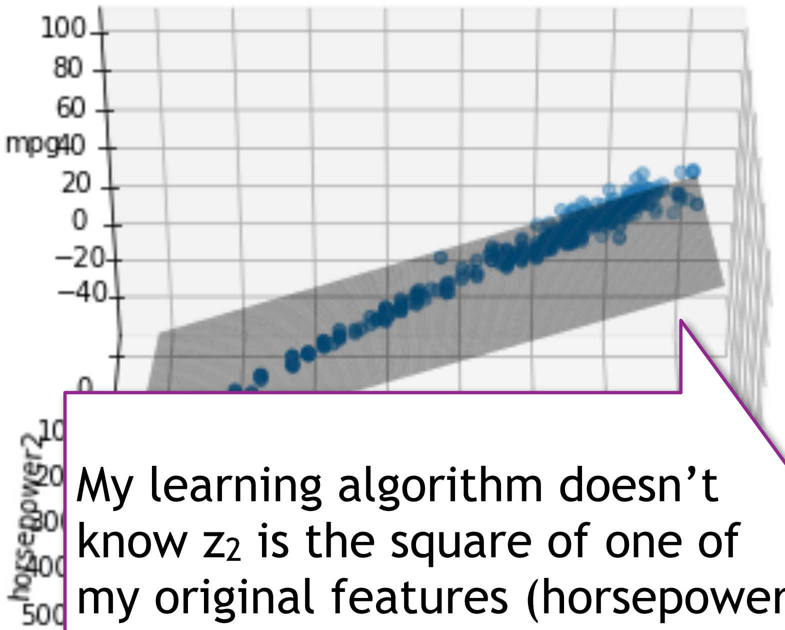
$$\tilde{\mathbf{w}}^T = [\tilde{w}_0, \tilde{w}_1, \tilde{w}_2]$$

$$\hat{y} = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T\mathbf{z} = \tilde{\mathbf{w}}^T\Phi(\mathbf{x})$$

A better hypothesis:

z_1 = horsepower
z_2 = horsepower^2

The R^2 value is 0.69 which is better our previous R^2 value 0.53



My learning algorithm doesn't know z_2 is the square of one of my original features (horsepower^2). The learning algorithm only sees feature z_2

Trained my linear model on these features: z_1 and z_2
(aka horsepower and horsepower^2)

$$\mathbf{x} \rightarrow \mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix}$$

Learn in **z** space with $\tilde{\mathbf{w}} = [\tilde{w}_0, \tilde{w}_1, \tilde{w}_2]$

Predict in **z** space $\hat{y} = g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})) = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T\mathbf{z}$

$$\hat{y} = 56.9 \cdot 1 + (-0.466) \cdot z_1 + 0.00123 \cdot z_2$$

$$\hat{y} = \underbrace{56.9}_{\tilde{w}_0} \cdot \underbrace{1}_{z_0} + \underbrace{(-0.466)}_{\tilde{w}_1} \cdot \underbrace{x}_{z_1} + \underbrace{0.00123}_{\tilde{w}_2} \cdot \underbrace{x^2}_{z_2}$$

$$= 56.9 \cdot 1 + (-0.466) \cdot \phi_1(\mathbf{x}) + 0.00123 \cdot \phi_2(\mathbf{x})$$

What is the feature vector in z-space of a car whose horsepower is 170 ?

$[170]$	(A)
$[1, 170]^T$	(B)
$[1, 170, 170^2]^T$	(C)
None of the above	(D)

$$X = \begin{array}{c} \begin{array}{cc} & \text{Horsepower} \end{array} \\ \begin{bmatrix} 1 & 130.0 \\ 1 & 165.0 \\ 1 & 150.0 \\ 1 & 150.0 \\ 1 & 140.0 \\ 1 & 198.0 \\ 1 & 220.0 \\ 1 & 215.0 \\ \cdots & \cdots \end{bmatrix} \end{array} \quad \mathbf{y} = \begin{bmatrix} 18.0 \\ 15.0 \\ 18.0 \\ 16.0 \\ 17.0 \\ 15.0 \\ 14.0 \\ 14.0 \\ \cdots \end{bmatrix}$$

$$X = \begin{array}{c} \begin{array}{ccc} & \text{Horsepower} & \text{Horsepower}^2 \end{array} \\ \begin{bmatrix} 1 & 130.0 & 16900.0 \\ 1 & 165.0 & 27225.0 \\ 1 & 150.0 & 22500.0 \\ 1 & 150.0 & 22500.0 \\ 1 & 140.0 & 19600.0 \\ 1 & 198.0 & 39204.0 \\ 1 & 220.0 & 48400.0 \\ 1 & 215.0 & 46225.0 \\ \cdots & \cdots & \cdots \end{bmatrix} \end{array} \quad \mathbf{y} = \begin{bmatrix} 18.0 \\ 15.0 \\ 18.0 \\ 16.0 \\ 17.0 \\ 15.0 \\ 14.0 \\ 14.0 \\ \cdots \end{bmatrix}$$

Using our closed form solution we calculate $\tilde{\mathbf{w}} = \begin{bmatrix} 56.9 \\ -0.466 \\ 0.00123 \end{bmatrix}$

To predict a new **x**

- transform **x** to $\Phi(\mathbf{x})=\mathbf{z}$
- predict with $\tilde{\mathbf{w}}$ in **z**-space

$$\hat{y} = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T\mathbf{z} = \tilde{\mathbf{w}}^T\Phi(\mathbf{x})$$

Estimated value of a car with horsepower = 170?

$$\tilde{\mathbf{w}}^T\Phi(\mathbf{x}) = \tilde{\mathbf{w}}^T \begin{bmatrix} 1 \\ 170 \\ 28900 \end{bmatrix} = [56.9 \quad -0.466 \quad 0.00123] \begin{bmatrix} 1 \\ 170 \\ 28900 \end{bmatrix}$$

$$X = \begin{array}{cc} & \text{Horsepower} \\ \left[\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \cdots \end{array}\right. & \left.\begin{array}{c} 130.0 \\ 165.0 \\ 150.0 \\ 150.0 \\ 140.0 \\ 198.0 \\ 220.0 \\ 215.0 \\ \cdots \end{array}\right] \end{array} \qquad \mathbf{y} = \begin{bmatrix} 18.0 \\ \vdots \\ 14.0 \\ \cdots \end{bmatrix}$$

My learning algorithm doesn't know the features are a transformation of my original features

$$X = \begin{array}{ccc} & \text{Horsepower} & \text{Horsepower}^2 \\ \left[\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \cdots \end{array}\right. & \begin{array}{c} 130.0 \\ 165.0 \\ 150.0 \\ 150.0 \\ 140.0 \\ 198.0 \\ 220.0 \\ 215.0 \\ \cdots \end{array} & \left.\begin{array}{c} 16900.0 \\ 27225.0 \\ 22500.0 \\ 22500.0 \\ 19600.0 \\ 39204.0 \\ 48400.0 \\ 46225.0 \\ \cdots \end{array}\right] \end{array} \qquad \mathbf{y} = \begin{bmatrix} 18.0 \\ 15.0 \\ 18.0 \\ 16.0 \\ 17.0 \\ 15.0 \\ 14.0 \\ 14.0 \\ \cdots \end{bmatrix}$$

Using our closed form solution we calculate $\tilde{\mathbf{w}} = \begin{bmatrix} 56.9 \\ -0.466 \\ 0.00123 \end{bmatrix}$

To predict a new **x**

- transform **x** to $\Phi(\mathbf{x})=\mathbf{z}$
- predict with $\tilde{\mathbf{w}}$ in **z**-space

$$\hat{y} = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T\mathbf{z} = \tilde{\mathbf{w}}^T\Phi(\mathbf{x})$$

Estimated value of a car with horsepower = 170?

$$\tilde{\mathbf{w}}^T\Phi(\mathbf{x}) = \tilde{\mathbf{w}}^T \begin{bmatrix} 1 \\ 170 \\ 28900 \end{bmatrix} = [56.9 \quad -0.466 \quad 0.00123] \begin{bmatrix} 1 \\ 170 \\ 28900 \end{bmatrix}$$

General Feature Transform

$\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^{\tilde{d}}$ is also called a **feature map**

Any function of the original features could be used

$\mathcal{X}$ – space is $\mathbb{R}^d$

$$\mathbf{x}^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \cdots \\ x_d^{(i)} \end{bmatrix}$$

Z – space is $\mathbb{R}^{\tilde{d}}$

$$\Phi(\mathbf{x}^{(i)}) = \mathbf{z}^{(i)} = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}^{(i)}) \\ \phi_2(\mathbf{x}^{(i)}) \\ \vdots \\ \phi_{\tilde{d}}(\mathbf{x}^{(i)}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1^{(i)} \\ z_2^{(i)} \\ \vdots \\ z_{\tilde{d}}^{(i)} \end{bmatrix}$$

Training data : $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$

$(\mathbf{z}^{(1)}, y^{(1)}), (\mathbf{z}^{(2)}, y^{(2)}), \ldots, (\mathbf{z}^{(N)}, y^{(N)})$

No weights in original space

$$\hat{y} = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T \mathbf{z} \;= \tilde{\mathbf{w}}^T \Phi(\mathbf{x})$$

$$\tilde{\mathbf{w}} = \begin{bmatrix} \tilde{w}_0 \\ \tilde{w}_1 \\ \vdots \\ \tilde{w}_{\tilde{d}} \end{bmatrix}$$

We form a linear combination of the ϕ_j thus they are called **basis functions**

Replacement for previous slide

General Feature Transform

$\Phi : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ is also called a **feature map**

Any function of the original features could be used

$\mathcal{X}$ – space is $\mathbb{R}^d$

$$\mathbf{x}^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix}$$

Z – space is $\mathbb{R}^{\tilde{d}}$

$$\Phi(\mathbf{x}^{(i)}) = \mathbf{z}^{(i)} = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}^{(i)}) \\ \phi_2(\mathbf{x}^{(i)}) \\ \vdots \\ \phi_{\tilde{d}}(\mathbf{x}^{(i)}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1^{(i)} \\ z_2^{(i)} \\ \vdots \\ z_{\tilde{d}}^{(i)} \end{bmatrix}$$

We perform the transformation for each training example:

Training data : $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$ $\Rightarrow$ $(\mathbf{z}^{(1)}, y^{(1)}), (\mathbf{z}^{(2)}, y^{(2)}), \ldots, (\mathbf{z}^{(N)}, y^{(N)})$

No weights in original space

$$\tilde{\mathbf{w}} = \begin{bmatrix} \tilde{w}_0 \\ \tilde{w}_1 \\ \vdots \\ \tilde{w}_{\tilde{d}} \end{bmatrix}$$

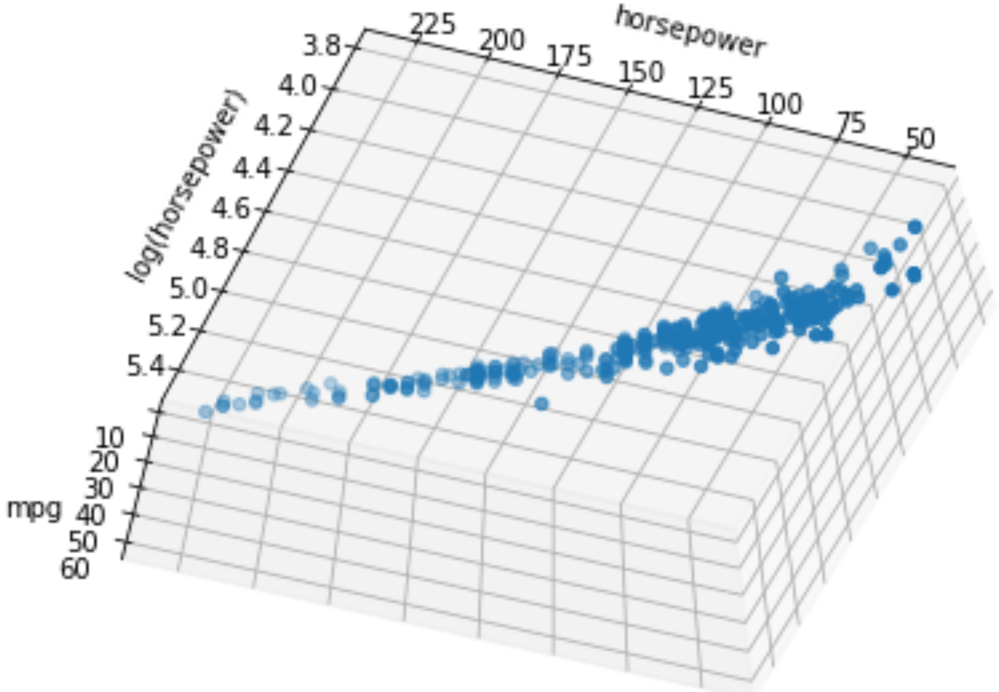
$$\hat{y} = \tilde{g}(\mathbf{z}) = \tilde{\mathbf{w}}^T\mathbf{z} = \tilde{\mathbf{w}}^T\Phi(\mathbf{x})$$

We form a linear combination of the ϕ_j thus they are called **basis functions**

Slide inspired by (modified from) Malik Magdon-Ismail's slide

Many nonlinear features may work

$$\mathbf{x}^{(i)} \to \mathbf{z}^{(i)} = \Phi(\mathbf{x}^{(i)}) = \begin{bmatrix} 1 \\ \phi_1(\mathbf{x}^{(i)}) \\ \phi_2(\mathbf{x}^{(i)}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1^{(i)} = x_1^{(i)} \\ z_2^{(i)} = \log(x_1^{(i)}) \end{bmatrix}$$



$$\tilde{\mathbf{w}} = [156.04, 0.12, -31.60]^T$$

The R^2 value is 0.68

The General Polynomial Transform Φ_k

Polynomial basis function
polynomial features

Example: The degree-k polynomial transform over two features

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

k is a hyperparameter (i.e. not one of the decision variables being optimized when fitting the data)

$$\mathbf{z} = \Phi_1(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix} \quad \Phi_2(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} \quad \Phi_3(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \\ x_1^3 \\ x_1^2x_2 \\ x_1x_2^2 \\ x_2^3 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \end{bmatrix} \quad \Phi_4(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \\ x_1^3 \\ x_1^2x_2 \\ x_1x_2^2 \\ x_2^3 \\ x_1^4 \\ x_1^3x_2 \\ x_1^2x_2^2 \\ x_1x_2^3 \\ x_2^4 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \end{bmatrix} \quad \dots$$

And so on …

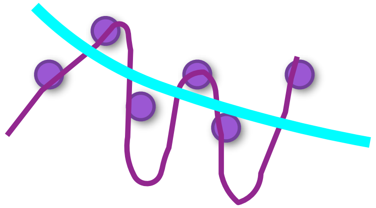
Dimensionality of the features space increases rapidly

Polynomial Regression

- Models the relationship between the response and features as an d^th^ order polynomial $y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + \cdots + w_d x_1^d + \epsilon$

  Example is using only one feature (monomial)

- Observation: the higher the order of the polynomial, the more shapes you can fit!
  - costs:
    - computational complexity grows as the number of coefficients grows
    - Increases how much you *model the noise. i.e.* increases overfitting → lose generalization



- Warning! It is always possible to perfectly fit N points with a polynomial of order d=(N-1). It is unlikely that such a model will provide knowledge of the unknown function or be able to predict as well on unseen data as a lower-order polynomial

How can we choose which (if any) transformation to use?

Typo fixed on this slide after lecture

Outline

- Motivating example:
- Feature transformation
- Underfitting and overfitting
- Understanding error: Bias and varianc
- Learning curves
- validation and model selection
- Model selection (with limit
- K-fold cross validation

- Regularization

Yea!

Uh oh....

How to create a more complex hypothesis

Understanding what went wrong

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

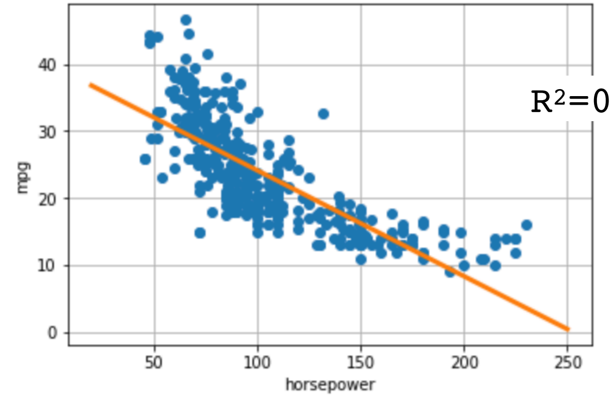
Our strategy

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

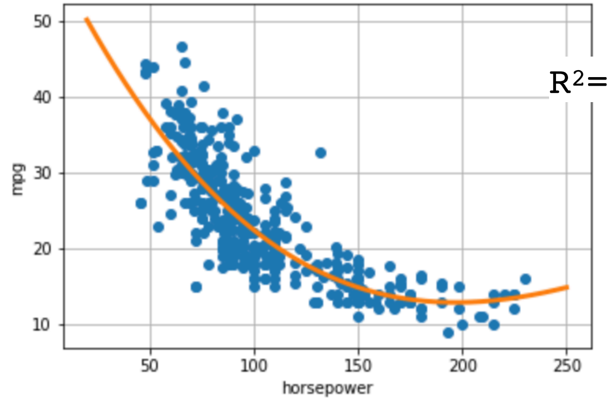
Automobile MPG

As we increase the degree of the polynomial, do we improve the fit of the model to the data?

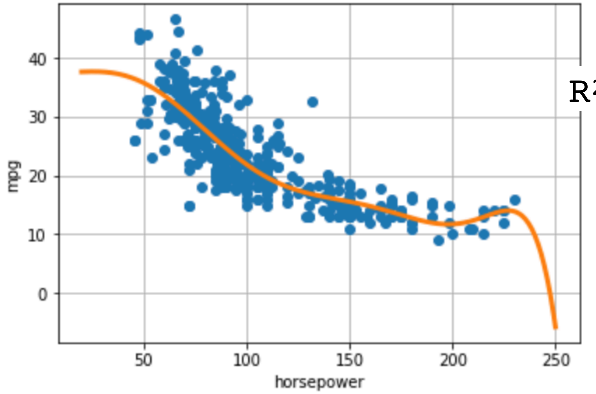
$$\hat{y} = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + \cdots + w_d x_1^d$$



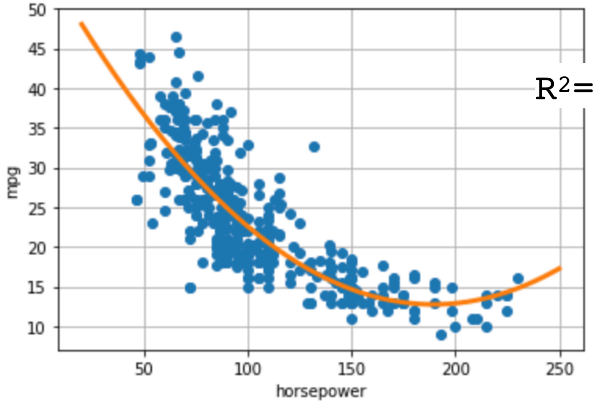
R^2=0.53



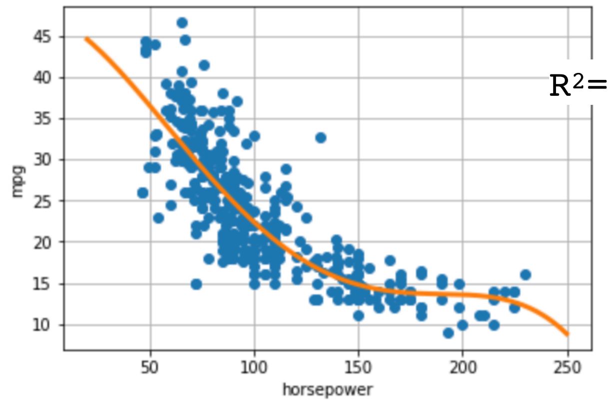
R^2=0.688214



R^2=0.694398



R^2=0.687559



R^2=0.689343

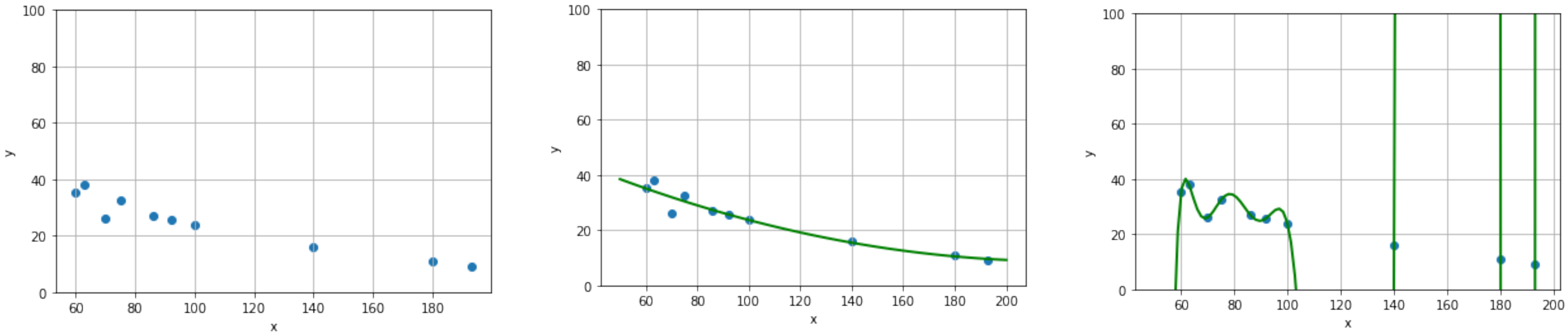
We can keep improving our wrt our training data - but does that mean we would do better on examples not in the training data?

Typo fixed on this slide after lecture

Example

Pair share/poll:
Should we choose the model based on the:
(A) R^2 score
(B) Training MSE
(C) Either (A) or (B)
(D) Neither (A) nor (B)

mean = 23.4710191083



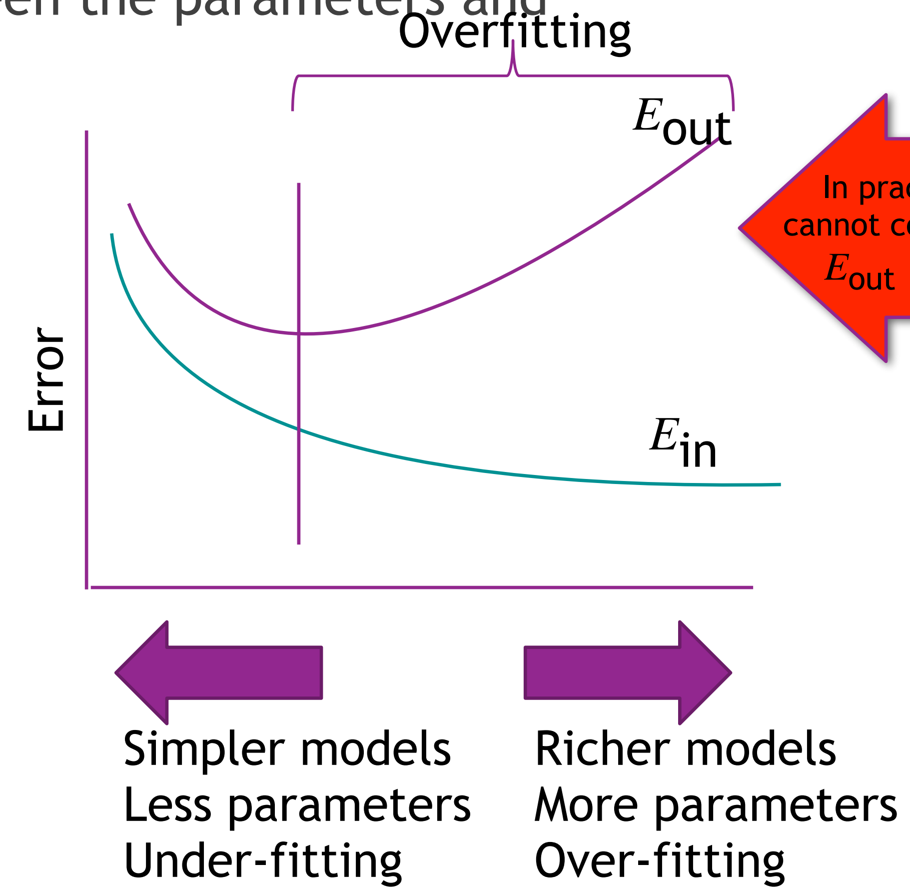
$E_{\text{in}} \approx 0; \quad E_{\text{out}} \gg 0$

Overfitting: Complex hypothesis that fits the training data too well.
It predicts well on patterns found in training data that won't be found in the the future data

What can go wrong with choosing the hypothesis which has the smallest lost/cost?

Overfitting

- If we allow a very flexible model by using complicated features or model, our model ***can*** be *too* complicated. The regression model can become tailored to fit the noise of the training data and does not generalize well (i.e., predicts well on the training set, and does not accurately describe the relationship between the parameters and the outcome)
- A too complicated model will not generally do well.
- ***overfitting:*** The model performs worse on unseen data than a different model from the same class despite performing better on the training data
- Example: Using a degree d=N-1 polynomial transformation
- Training RSS (or MSE), R^2 is not a good indicator of future performance



In practice we cannot compute E_{out}

Typo fixed after lecture

Overfitting

Complicated features

↓ Creates

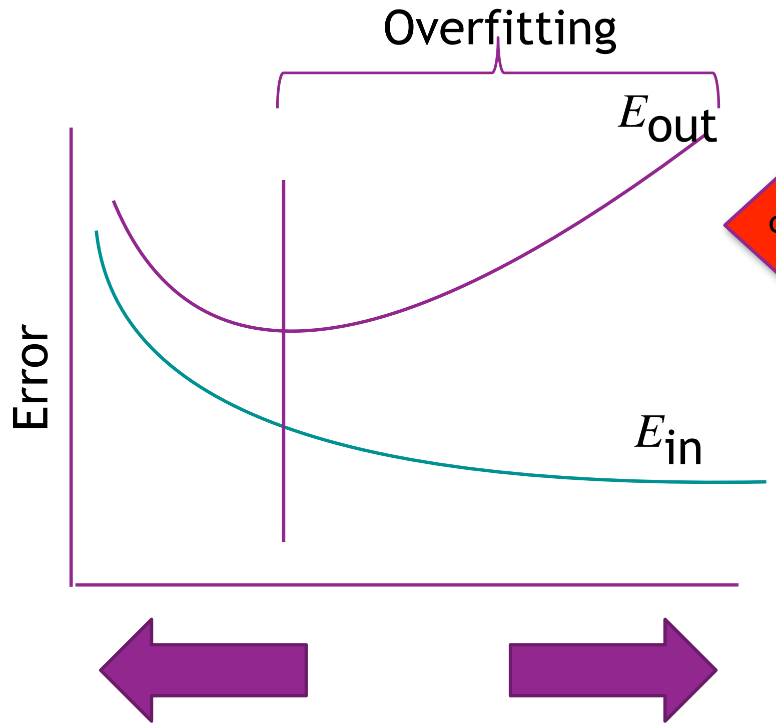
Complicated models

↓ Sometimes

Overfitting (modeled noise)

↓

Doesn't generalize well



Simpler models
Less parameters
Under-fitting

Richer models
More parameters
Over-fitting

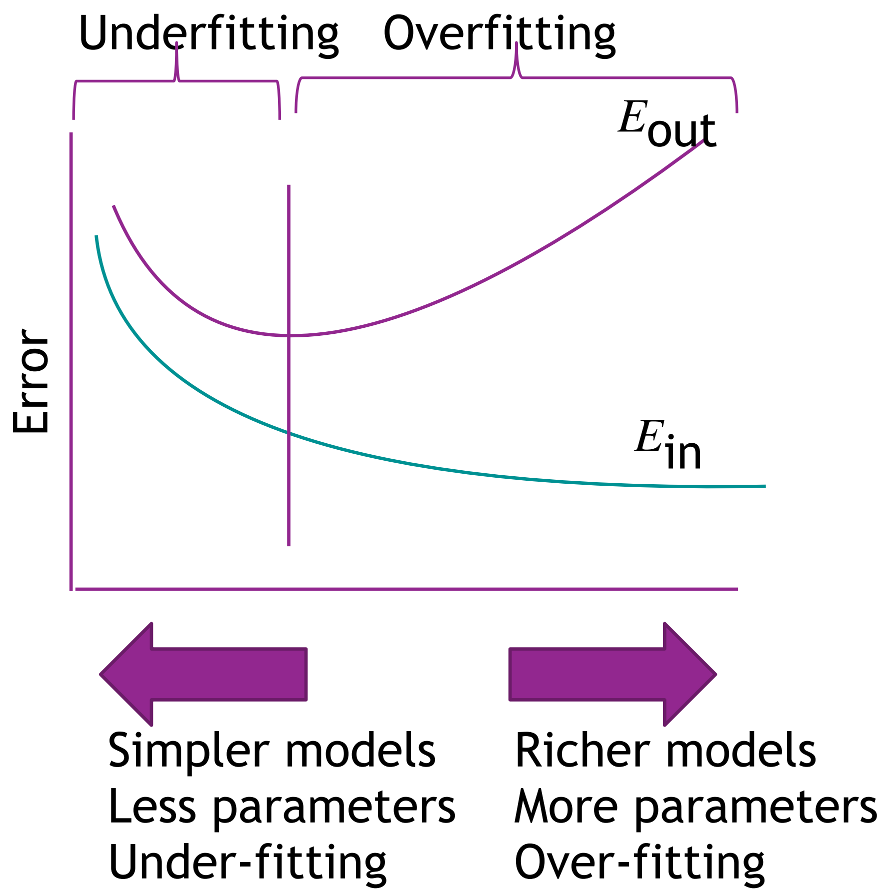
In practice we cannot compute E_{out}

- Example: Using a degree d=N-1 polynomial transformation
- Training RSS (or MSE), R^2 is not a good indicator of future performance

Slide added after lecture

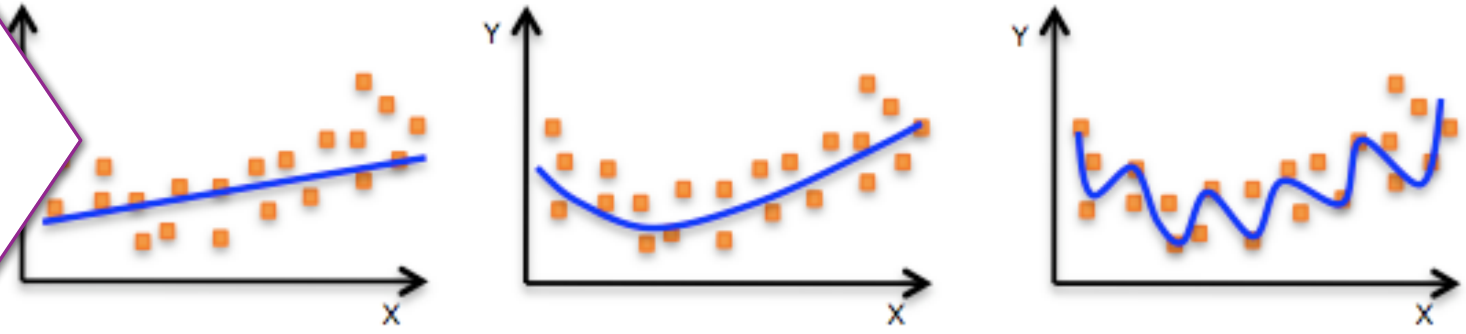
Underfitting

- The model learned does not do well on the training data and does not do well on unseen examples
- A too simple model is called ***underfitting***
- Example: predicting mean of target



How Can You Tell from Data?

Model is too inflexible, it is not fitting the pattern



Model is too flexible, it is fitting the noise

- Is there a way to tell what is the correct model order to use?
- Must use the data. Do not have access to the true d?
- What happens if we guess:
  - d too big?
  - d too small?

Question

For the examples below, might we encounter a problem with the model we chose?

- Examples:
  - True function $f(x) = 2 + 3x + \epsilon$ model class $w_0 + w_1 x + w_2 x^2$
  - True function $f(x) = 2 + 3x + 4x^2$ model class $w_0 + w_1 x$

What can go wrong with choosing the hypothesis which has the smallest lost/cost?

1. Limited Hypothesis class (model class). No function in our hypothesis class can model the data well - biased solution
2. Limited Data. We might model the noise and not the true pattern. Small changes to the data causes the hypothesis (model) to change - high variance solution

Outline

- ❑Motivating example:
- ❑ Feature transformation
- ❑Underfitting and overfitting
- ❑Understanding error: Bias and varianc
- ❑Learning curves
- ❑validation and model selection
- ❑Model selection (with limit
- ❑K-fold cross validation
- ❑Regularization

Yea!

Uh oh....

How to create a more complex hypothesis

Understanding what went wrong

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

Our strategy

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

How do we evaluate our model? Or choose among models (e.g. the which polynomial transformation should we choose?)

Open discussion

- We can evaluate how well it works by looking at its errors
- We would like the error to be zero on all future data. However:
  - The unseen variables means the true model has non-zero error (i.e. the world is a messy place)
  - Our hypothesis probably doesn't contain the underlying true model
  - We don't get enough data to perfectly estimate our model. We only get a finite sample of the data. The more data we receive, the more our sample is representative of underlying data and our estimates should converge

Noise/irreducible error

Bias

Variance

Where did the prediction error in our hypothesis come from?

- Regression example: $y = f(\mathbf{x}) + \epsilon$

Deterministic

Noise $\sim N(0,\sigma)$
We are assuming the noise has mean 0 and variance σ^2

This means $E_{\mathbf{x},y}[f(\mathbf{x}) - y] = 0$ and $E_{\mathbf{x},y}[(f(\mathbf{x}) - y)^2] = E_{\mathbf{x}}(\epsilon^2) = \sigma^2$

Best estimate for y given $\mathbf{x}$ is $f(\mathbf{x})$

- Goal is to understand why our *expected* hypothesis (model) does not have zero error

$$E_D\left[E_{\text{out}}(g^{(D)})\right] = E_D\left[E_{\mathbf{x},y}[(g^{(D)}(\mathbf{x}) - y)^2]\right] \neq 0$$

$E_{\text{out}}(g^{(D)})$

$E_{\mathbf{x},y}[(g^{(D)}(\mathbf{x}) - y)^2]$ expected error for they hypothesis $g^{(D)}(\mathbf{x})$

The expected error of the hypothesis on any future example. The hypothesis was fit using the data set D

We focus on Algorithm Bias

Understanding Error
Bias-Variance-Noise Decomposition

$$E_{\text{out}}(g^{(D)}(\mathbf{x})) = E_{\mathbf{x},y}[(y - g^{(D)}(\mathbf{x}))^2]$$

Our definitions will be for the squared loss function
You can think of how to substitute other loss functions

$$E_{\text{out}}(g) = \text{bias} + \text{variance} + \text{noise}$$

This cannot be computed in practice because we do not have access to the target function or the probability distribution

In predictions there are three sources of error.

1. noise - irreducible error
2. bias - error of average hypothesis (estimated from N examples) from the true function
3. variance - how much would the prediction for an example change if the hypothesis was fit on a different set of N points

High Bias $\leftrightarrow$ underfitting

High Variance $\leftrightarrow$ overfitting