GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 7: REGRESSION & REGULARIZATION

LAST TIME

0. DATA EXPLORATION PRESENTATIONS

I. LINEAR REGREGSSION

II. MATH BEHIND THE SCENES

5. INTRO TO MACHINE LEARNING & KNN
6. LINEAR REGRESSION
7. REGRESSION & REGULARIZATION (TODAY)
8. STATISTICS & BAYES
9. DECISION TREES
10. RECAP SUPERVISED LEARNING

I. POLYNOMIAL REGRESSION
II. OVERFITTING
III. REGULARIZATION
IV. LINEAR ALGEBRA & NUMPY
V. EXERCISES (NUMPY, LINEAR ALGEBRA, LINEAR REGRESSION)

I: POLYNOMIAL REGRESSION

	continuous	*categorical*
supervised	*regression*	*classification*
unsupervised	*dimension reduction*	*clustering*

$$y = \alpha + \beta x + \varepsilon$$

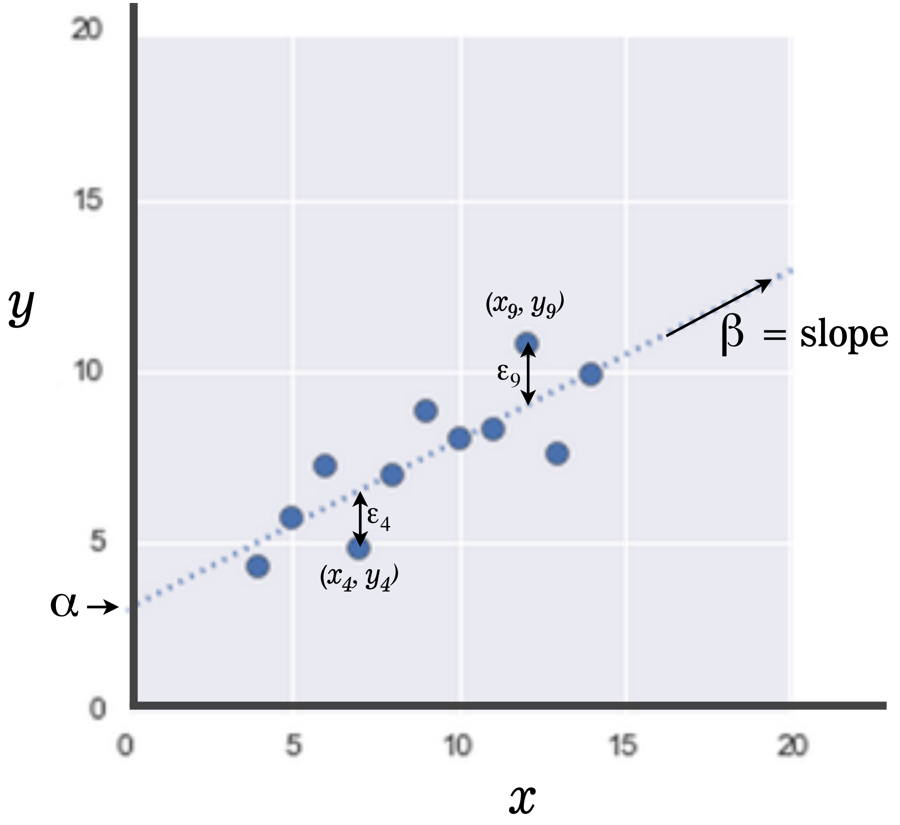
y = **response variable**

x = **input variable**

α = **intercept**

β = **regression coefficient**

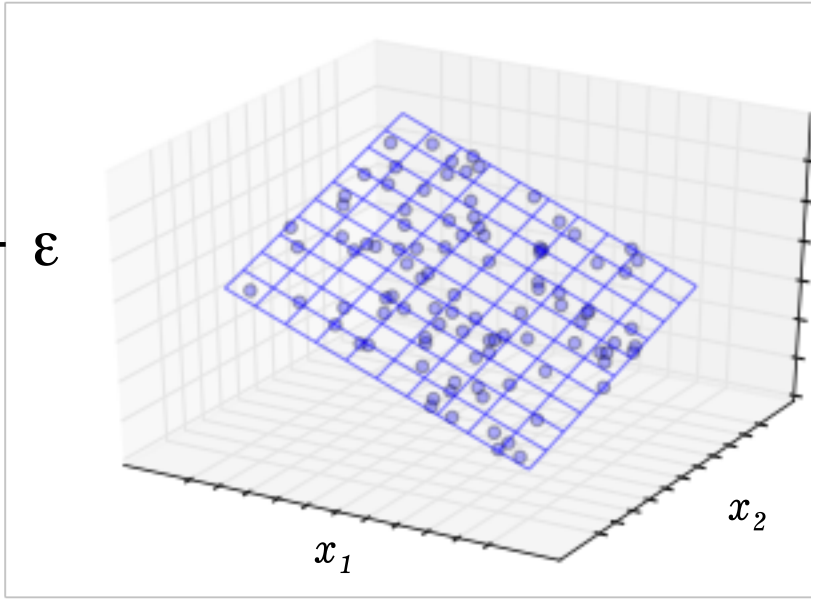
ε = **residual** *(the error)*



Source: Anscombe's Quartet

We can extend this model to several input variables, giving us the **multiple linear regression** *model:*

$$y = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n + \varepsilon$$



Consider the following **polynomial regression** *model:*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Consider the following **polynomial regression** *model:*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Q: This represents a nonlinear relationship. Is it still a linear model?

Consider the following **polynomial regression** *model:*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Q: This represents a nonlinear relationship. Is it still a linear model?

A: Yes, because it's linear in the β*'s!*

Consider the following **polynomial regression** *model:*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

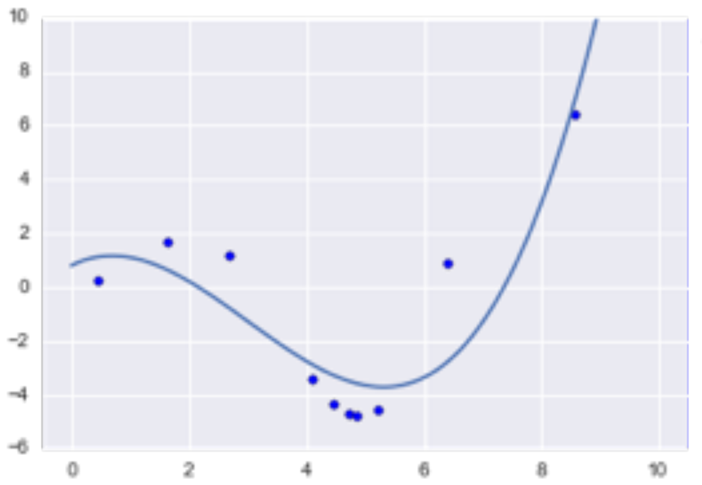
Q: This represents a nonlinear relationship. Is it still a linear model?

A: Yes, because it's linear in the β*'s!*

"Although polynomial regression fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y|x) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression." -- Wikipedia

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$



Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

But there is one problem with the model we've written down so far.

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

But there is one problem with the model we've written down so far.

Q: Does anyone know what it is?

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

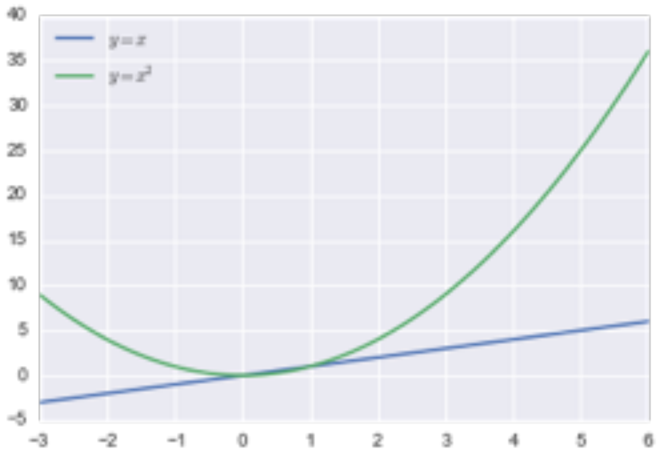
But there is one problem with the model we've written down so far.

Q: Does anyone know what it is?

A: This model violates one of the assumptions of linear regression!

This model displays **multicollinearity**, *which means the predictor variables are highly correlated with each other.*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$



x and x^2 have an $R^2 > 0.9$ on the interval [0, 1]

This model displays **multicollinearity**, *which means the predictor variables are highly correlated with each other.*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

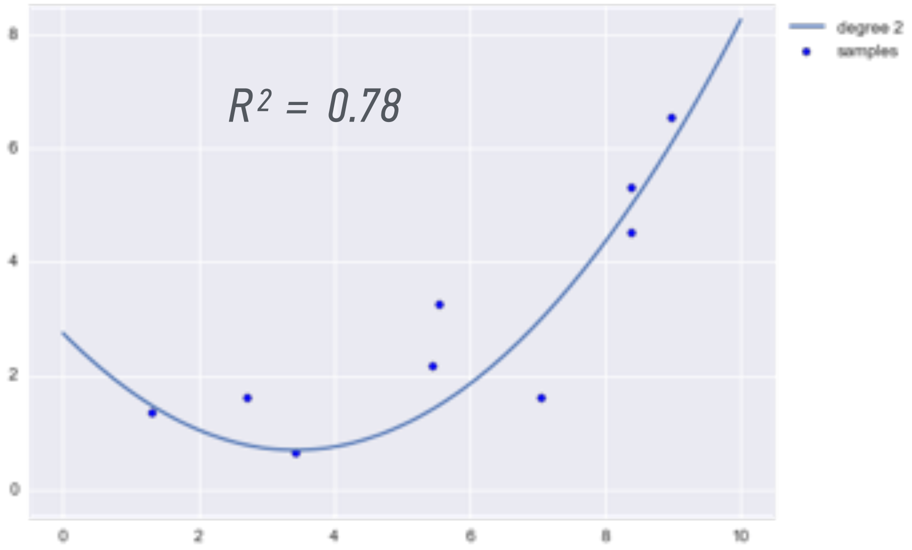
Multicollinearity causes the linear regression model to break down, because it can't tell the predictor variables apart.

This model displays **multicollinearity***, which means the predictor variables are highly correlated with each other.*

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

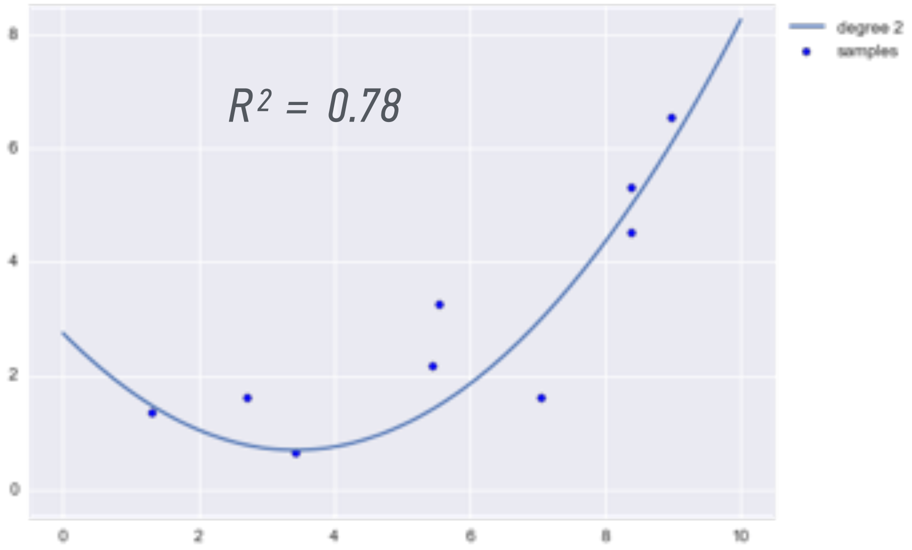
For now, let's keep this in the back of our minds.

So far, we've seen how polynomial regression allows us to fit complex nonlinear relationships



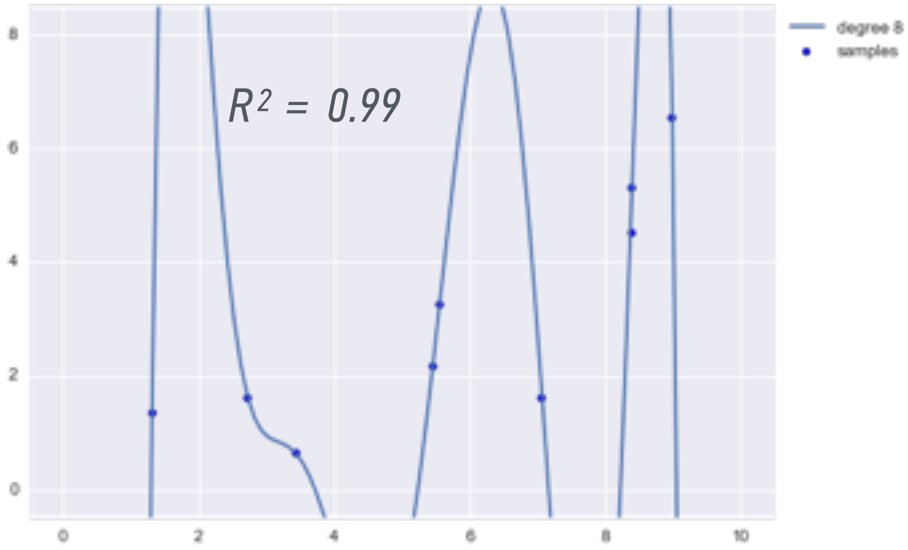
So far, we've seen how polynomial regression allows us to fit complex nonlinear relationships

Q: Can a regression model be too complex?



So far, we've seen how polynomial regression allows us to fit complex nonlinear relationships

Q: Can a regression model be too complex?



II. OVERFITTING

	continuous	categorical
supervised	regression	classification
unsupervised	dimension reduction	clustering

Q: What does “supervised” mean?

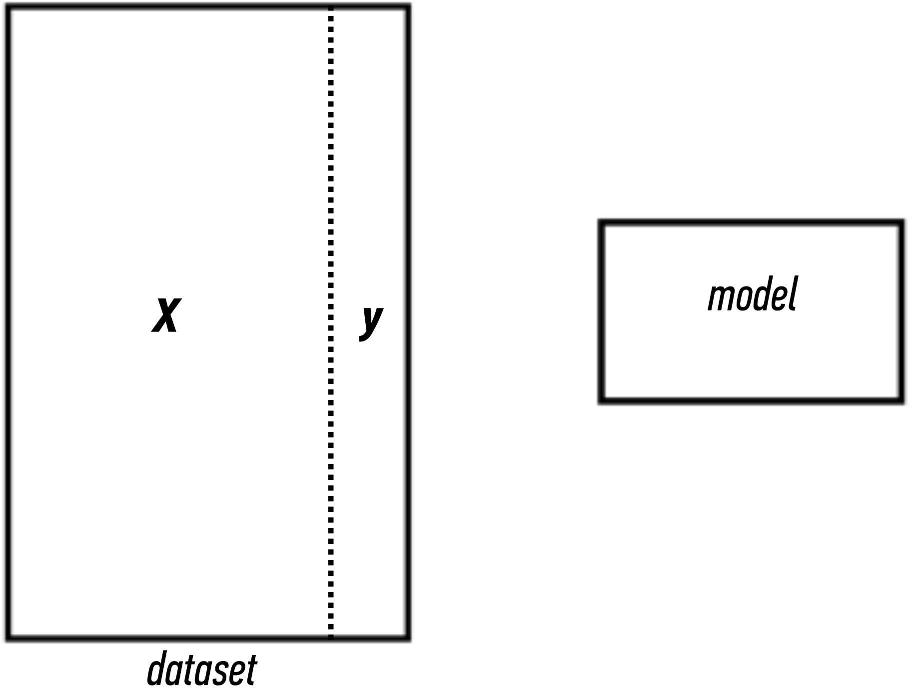
Q: What does "supervised" mean?

A: We know the labels.

sex	role	yrs	degree	yrs w/deg	salary
male	full	25	doctorate	35	36350
male	full	13	doctorate	22	35350
male	full	10	doctorate	23	28200
female	full	7	doctorate	27	26775
male	full	19	masters	30	33696

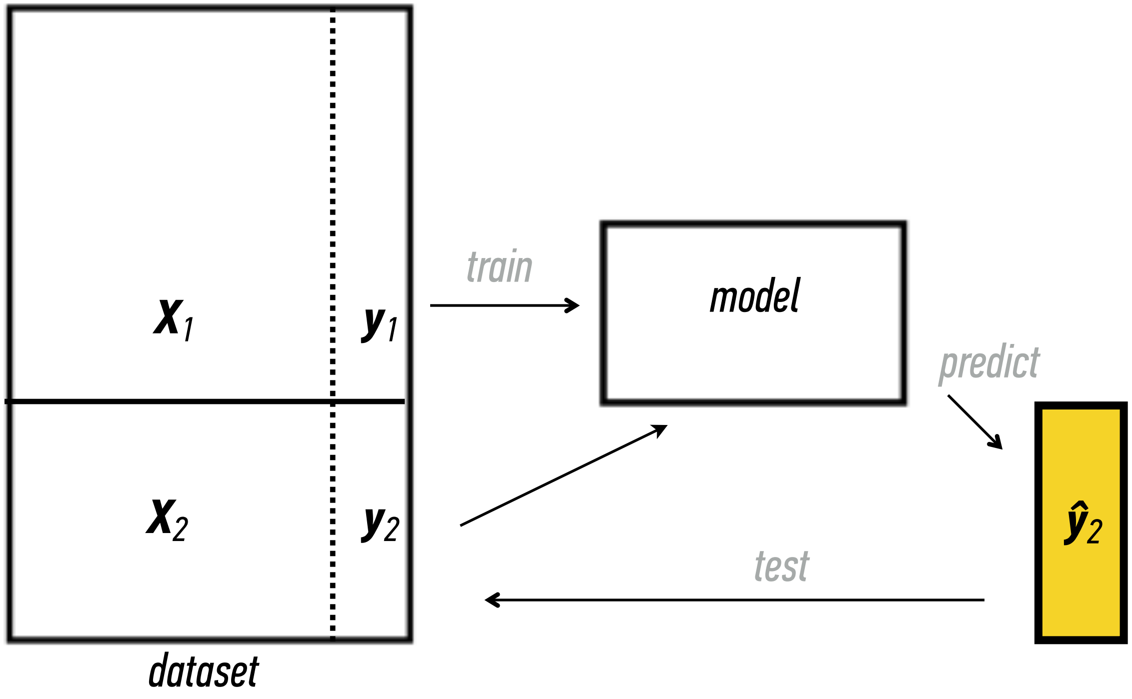
sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica

Q: How do we test the model's predictions?



Q: How do we test the model's predictions?

Train model on a part of $\boldsymbol{X}$*, and test the results on the rest of the data*



Q: Why should we use training & test sets?

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

Q: Why should we use training & test sets?

Thought experiment:
Suppose instead, we train our model using the entire dataset.
Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

- We can make the model arbitrarily complex (effectively "memorizing" the entire training set).

A: Down to zero!

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

A: Down to zero!

NOTE

This phenomenon is called *overfitting*.

Q: Why should we use training & test sets?

Thought experiment:
Suppose instead, we train our model using the entire dataset.
Q: How low can we push the training error?
- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

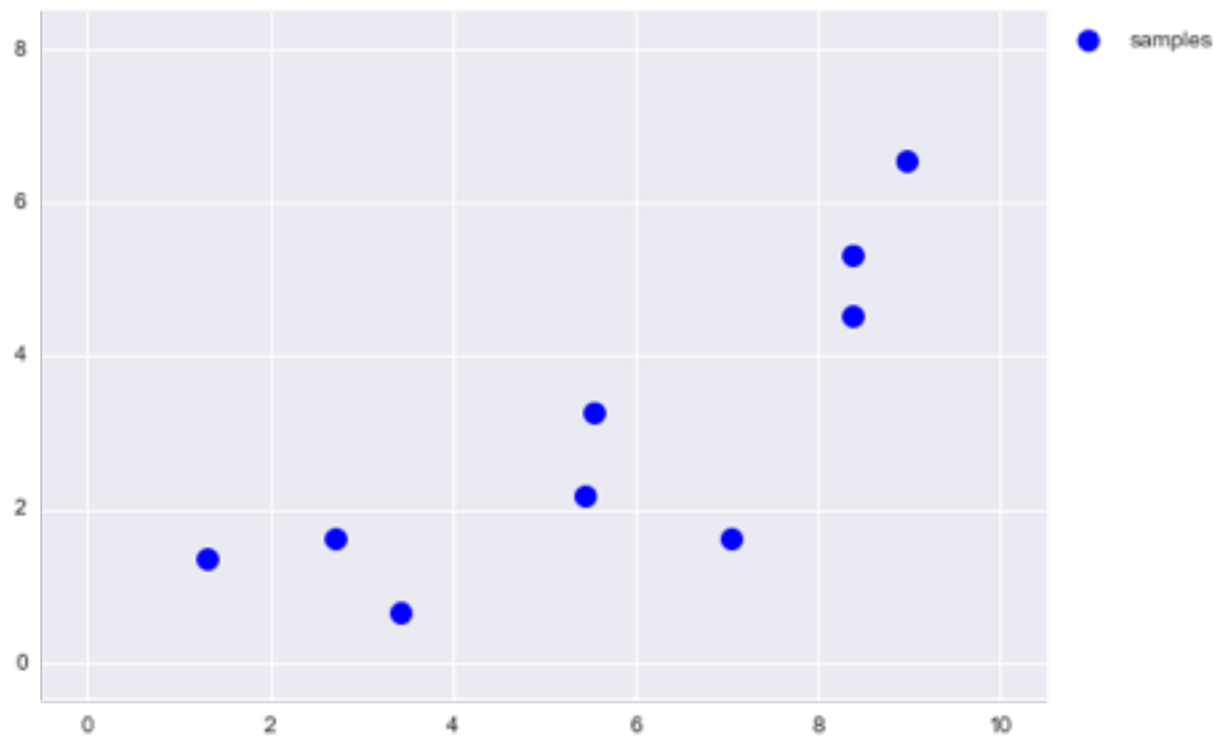
A: Down to zero!

A: Training error is not a good estimate of out-of-sample accuracy.

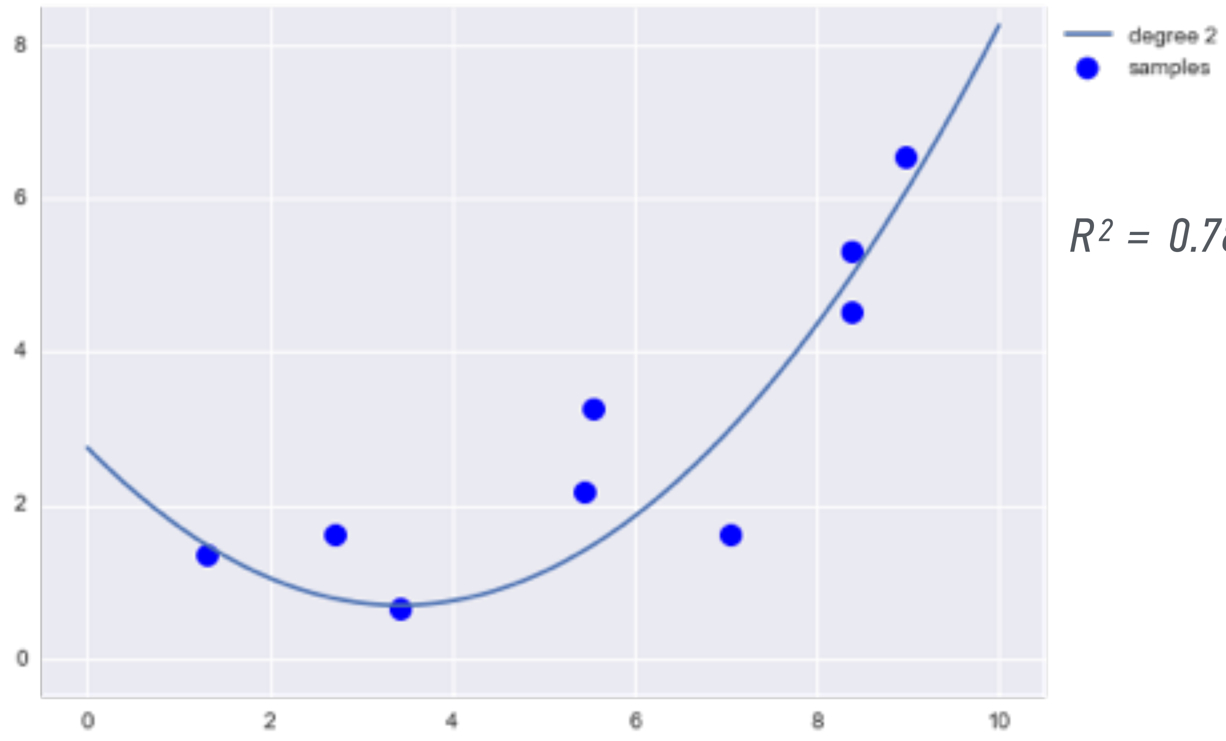
NOTE

This phenomenon is called *overfitting*.

OVERFITTING – REGRESSION EXAMPLE

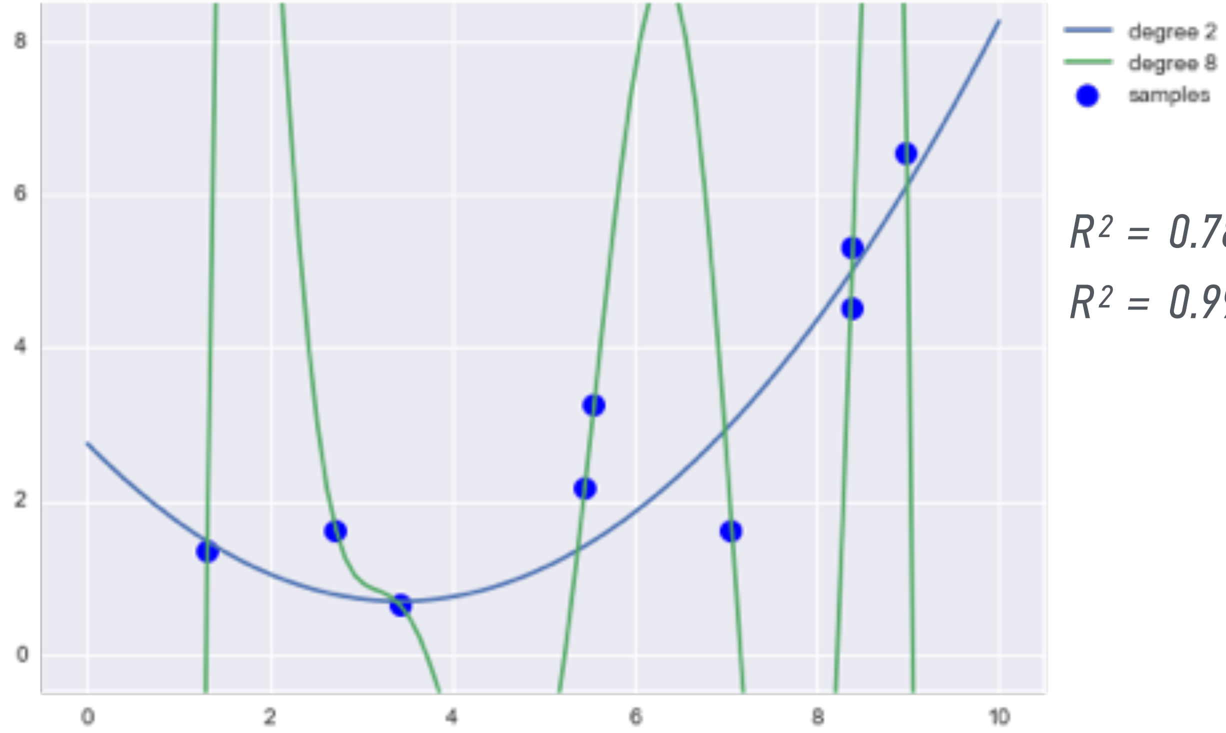


OVERFITTING – REGRESSION EXAMPLE



$R^2 = 0.78$

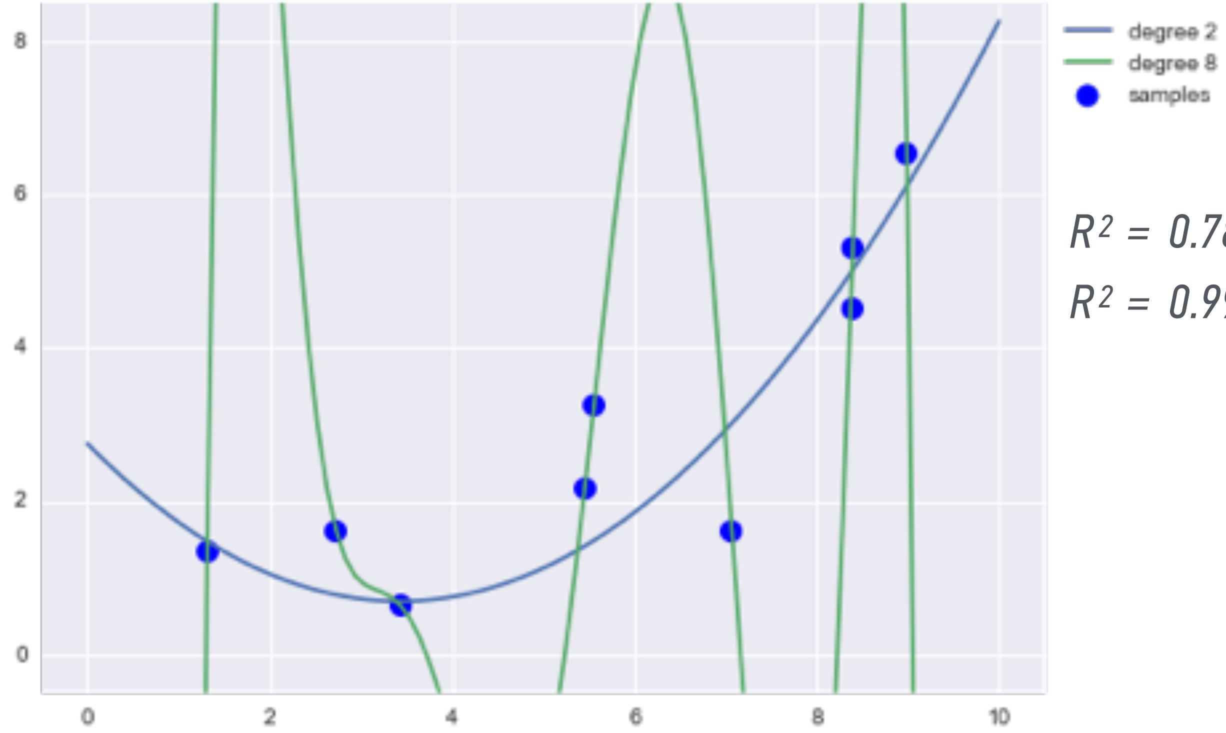
OVERFITTING – REGRESSION EXAMPLE



$R^2 = 0.78$

$R^2 = 0.99$

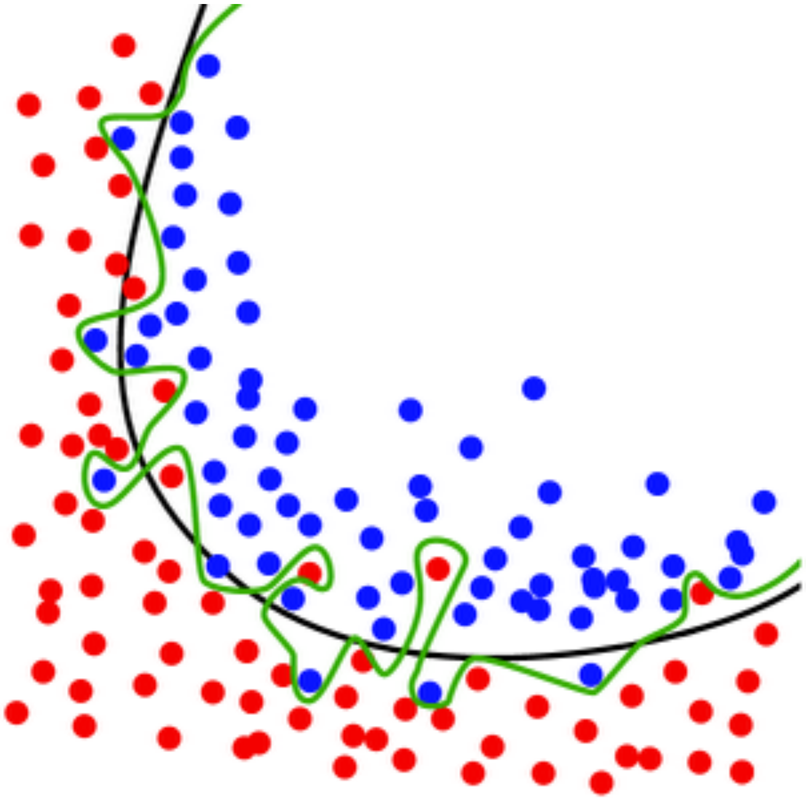
Which model is better?



$R^2 = 0.78$

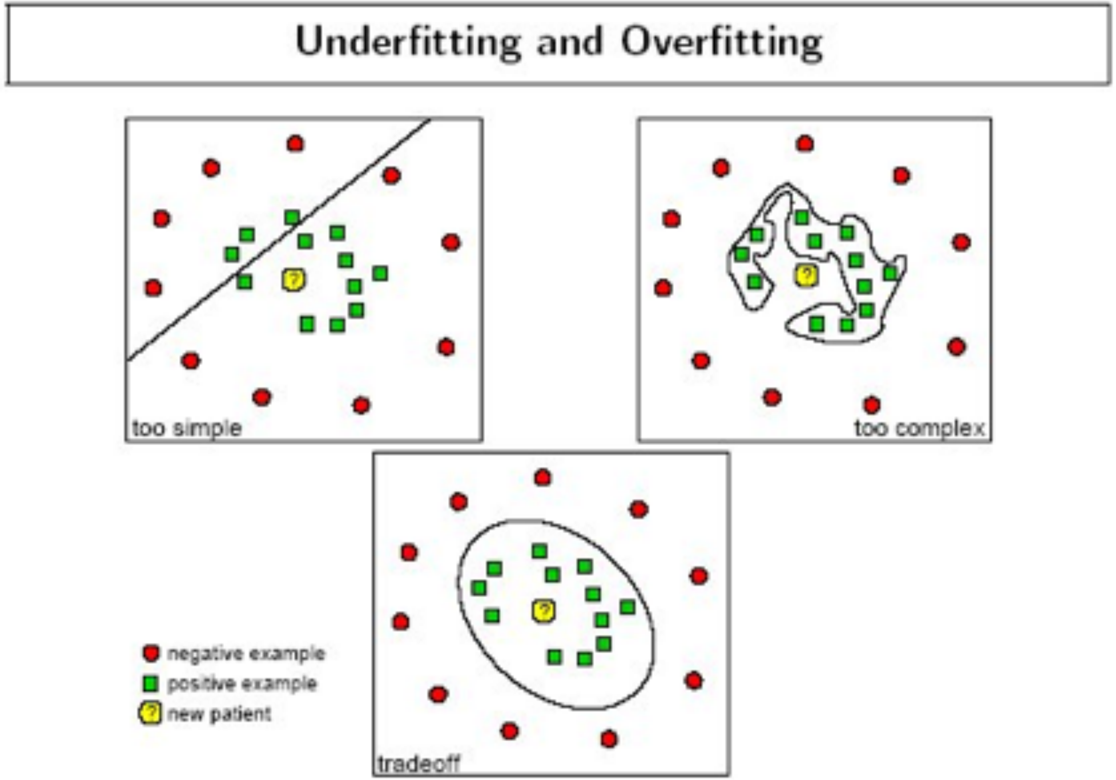
$R^2 = 0.99$

OVERFITTING – CLASSIFICATION EXAMPLE



source: http://www.dtreg.com

OVERFITTING VS UNDERFITTING – CLASSIFICATION EXAMPLE



source: http://www.dtreg.com

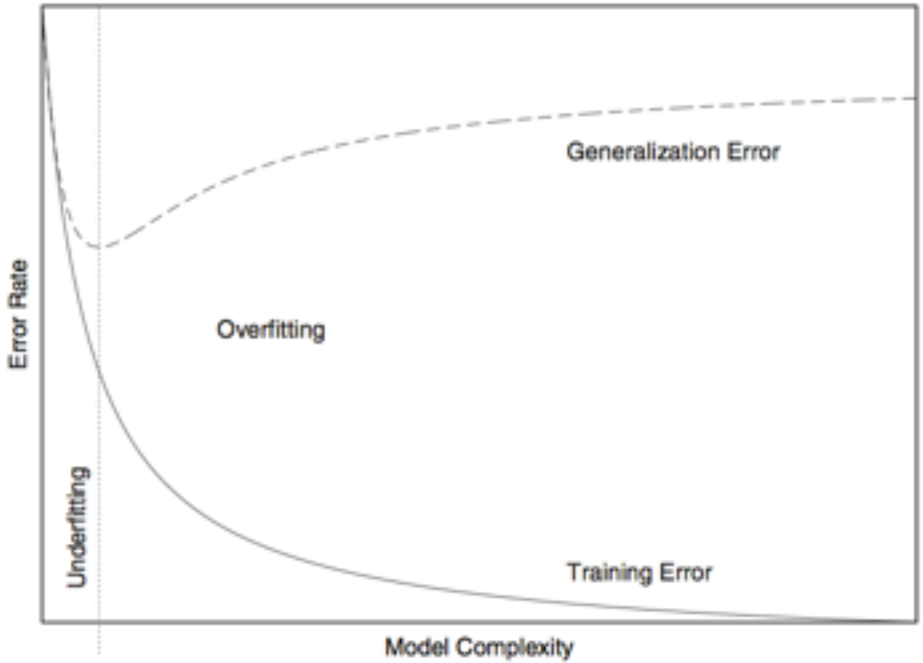


FIGURE 18-1. Overfitting: as a model becomes more complex, it becomes increasingly able to represent the training data. However, such a model is overfitted and will not generalize well to data that was not used during training.

source: Data Analysis with Open Source Tools, by Philipp K. Janert. O'Reilly Media, 2011.

Overfitting can happen in classification and regression problems.

It is a result of matching the training set too closely:
the model matches the **noise** *in the dataset instead of the* **signal***.*

Overfitting can happen in classification and regression problems.

It is a result of matching the training set too closely:
the model matches the **noise** *in the dataset instead of the* **signal**.

This happens when the model becomes too complex for the data to support.

Overfitting can happen in classification and regression problems.

It is a result of matching the training set too closely:
the model matches the **noise** *in the dataset instead of the* **signal***.*

This happens when the model becomes too complex for the data to support: **too many features** *(columns), or* **too few samples** *(rows).*

Suppose we do the train/test split.

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

OOS = out-of-sample

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

A: Of course not!

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

A: Of course not!

A: On its own, not very well.

Something is still missing!

Q: How can we do better?

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

A: Now you're talking!

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

A: Now you're talking!

A: Cross-validation.

Steps for n-fold cross-validation:

Steps for n-fold cross-validation:

1) *Randomly split the dataset into n equal partitions.*

Steps for n-fold cross-validation:

1) *Randomly split the dataset into n equal partitions.*
2) *Use partition 1 as test set & union of other partitions as training set.*

Steps for n-fold cross-validation:

1) *Randomly split the dataset into n equal partitions.*
2) *Use partition 1 as test set & union of other partitions as training set.*
3) *Find generalization error.*

Steps for n-fold cross-validation:

1) *Randomly split the dataset into n equal partitions.*
2) *Use partition 1 as test set & union of other partitions as training set.*
3) *Find generalization error.*
4) *Repeat steps 2-3 using a different partition as the test set at each iteration.*

Steps for n-fold cross-validation:

1) *Randomly split the dataset into n equal partitions.*
2) *Use partition 1 as test set & union of other partitions as training set.*
3) *Find generalization error.*
4) *Repeat steps 2-3 using a different partition as the test set at each iteration.*
5) *Take the average generalization error as the estimate of OOS accuracy.*

Features of n-fold cross-validation:

Features of n-fold cross-validation:

1) *More accurate estimate of OOS prediction error.*

Features of n-fold cross-validation:

1) *More accurate estimate of OOS prediction error.*
2) *More efficient use of data than single train/test split.*
    - *Each record in our dataset is used for both training and testing.*

Features of n-fold cross-validation:

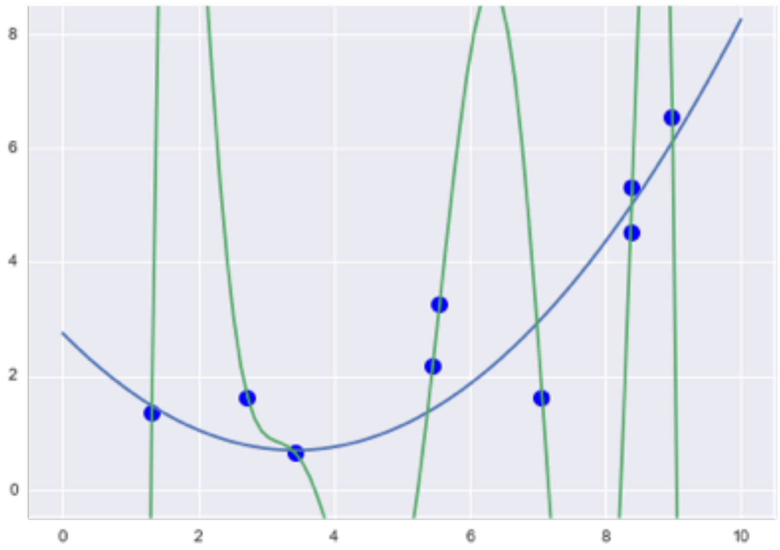
1) *More accurate estimate of OOS prediction error.*
2) *More efficient use of data than single train/test split.*
   - *Each record in our dataset is used for both training and testing.*
3) *Presents tradeoff between efficiency and computational expense.*
   - *10-fold CV is 10x more expensive than a single train/test split*

Features of n-fold cross-validation:

1) *More accurate estimate of OOS prediction error.*
2) *More efficient use of data than single train/test split.*
   - *Each record in our dataset is used for both training and testing.*
3) *Presents tradeoff between efficiency and computational expense.*
   - *10-fold CV is 10x more expensive than a single train/test split*
4) *Can be used for model selection.*

OK, so now we know how to properly test our models.

But how do we prevent overfitting from happening?



III. REGULARIZATION

Q: How do we define the **complexity** *of a regression model?*

$$\boxed{y = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n + \varepsilon}$$

Q: How do we define the **complexity** *of a regression model?*

A: One method is to define complexity as a function of the size of the coefficients.

$$\boxed{y = \alpha + \beta_1 x_1 + \ldots + \beta_n x_n + \varepsilon}$$

Q: How do we define the **complexity** *of a regression model?*

A: One method is to define complexity as a function of the size of the coefficients.

Ex 1: $\Sigma|\beta_i|$

Ex 2: $\Sigma\,\beta_i^2$

Q: How do we define the **complexity** *of a regression model?*

A: One method is to define complexity as a function of the size of the coefficients.

Ex 1: $\Sigma|\beta_i|$ *this is called the* **L1-norm**

Ex 2: $\Sigma\,\beta_i^2$ *this is called the* **L2-norm**

These measures lead to the following **regularization** *techniques:*

These measures lead to the following **regularization** *techniques:*

L1 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\| \right)$

penalize each coefficient

These measures lead to the following **regularization** *techniques:*

L1 regularization: $\min (\|y - x\beta\|^2 + \lambda\|\beta\|)$

L2 regularization: $\min (\|y - x\beta\|^2 + \lambda\|\beta\|^2)$

penalize each coefficient

These measures lead to the following **regularization** *techniques:*

OLS: $\min(\ \|y - x\beta\|^2\ )$

L1 regularization: $\min(\ \|y - x\beta\|^2 + \lambda\|\beta\|\ )$

L2 regularization: $\min(\ \|y - x\beta\|^2 + \lambda\|\beta\|^2\ )$

We are no longer just minimizing error but also an additional term.

These measures lead to the following **regularization** *techniques:*

OLS: $\min\left(\|y - x\beta\|^2 \right)$

L1 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\| \right)$

L2 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\|^2 \right)$

Regularization *refers to the method of preventing* **overfitting** *by explicitly controlling model* **complexity**.

These measures lead to the following **regularization** *techniques:*

OLS: $\min(\|y - x\beta\|^2)$

***Lasso* regularization**: $\min(\|y - x\beta\|^2 + \lambda\|\beta\|)$

***Ridge* regularization**: $\min(\|y - x\beta\|^2 + \lambda\|\beta\|^2)$

Regularization *refers to the method of preventing* **overfitting** *by explicitly controlling model* **complexity**.

Q: What are bias and variance?

Q: What are bias and variance?

A: **Bias** *refers to predictions that are systematically inaccurate.*

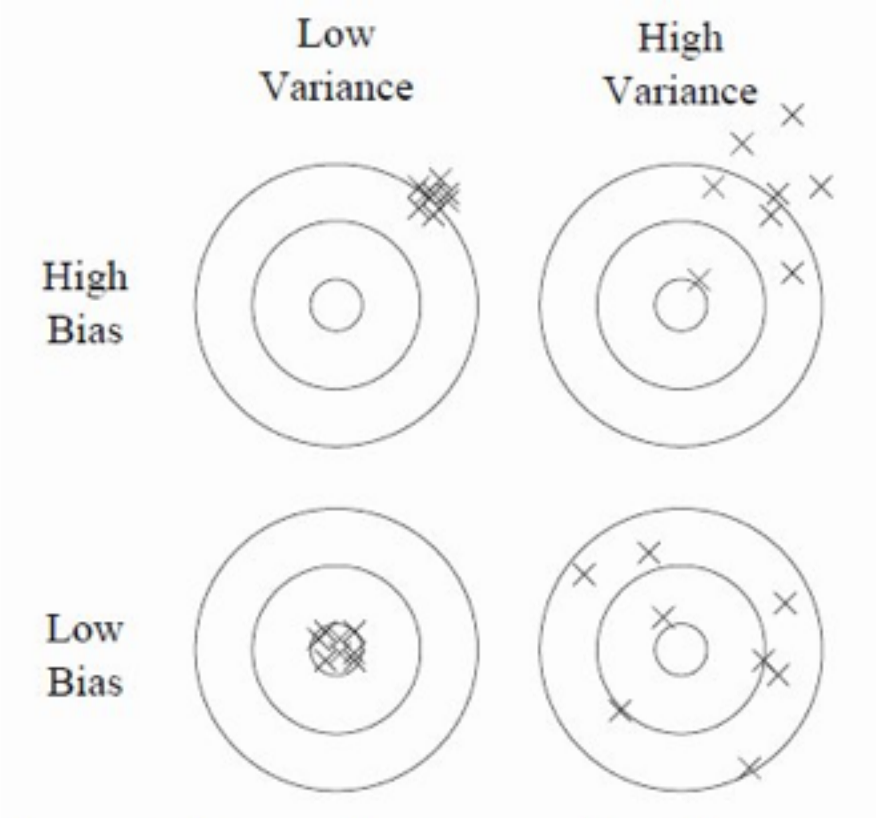
Q: What are bias and variance?

A: **Bias** *refers to predictions that are systematically inaccurate.*

Variance *refers to predictions that are generally inaccurate.*

Q: What are bias and variance?

Bias = *systematic error*
Variance = *general error*



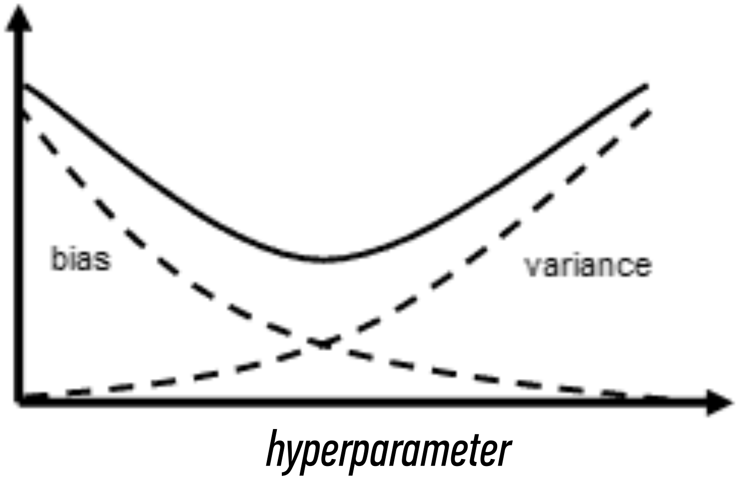
source: http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Q: What are bias and variance?

Bias = *systematic error*
Variance = *general error*

It turns out (after some math) that the generalization error in our model can be decomposed into a bias component and variance component.

Q: What are bias and variance?

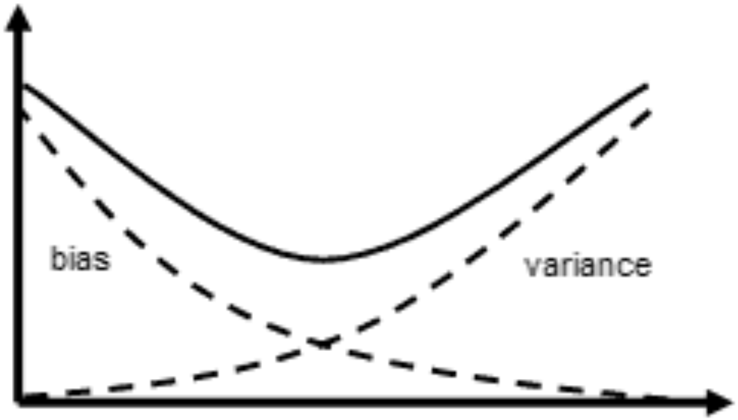


Bias = *systematic error*

Variance = *general error*

This is another example of the **bias-variance tradeoff***.*

Q: What are bias and variance?



hyperparameter

Bias = *systematic error*

Variance = *general error*

NOTE

The *hyper-parameter* here is the lambda we saw above.

This is another example of the **bias-variance tradeoff**.

This tradeoff is regulated by a **hyperparameter** λ, *which we've seen:*

OLS: $\min\left(\|y - x\beta\|^2 \right)$

L1 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\| \right)$

L2 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\|^2 \right)$

This tradeoff is regulated by a **hyperparameter** λ*, which we've seen:*

OLS: $\min\left(\|y - x\beta\|^2 \right)$

L1 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\| \right)$

L2 regularization: $\min\left(\|y - x\beta\|^2 + \lambda\|\beta\|^2 \right)$

So **regularization** *represents a method to trade away some* ***variance for a little bias*** *in our model, thus achieving a better overall fit.*

DISCUSSION