

# Preventing Overfitting with Ridge and Lasso Regression

Scott O'Hara

Metrowest Developers Machine  
Learning Group

03/28/2018

# References

## **Applied Machine Learning in Python**

University of Michigan, Prof. Kevin Collins Thompson

<https://www.coursera.org/learn/python-machine-learning/home/welcome>

## **Machine Learning: Regression**

University of Washington, Profs. Emily Fox & Carlos Guestrin

<https://www.coursera.org/learn/ml-regression/home/welcome>

## **CS181 Intelligent Machines: Perception, Learning and Uncertainty**

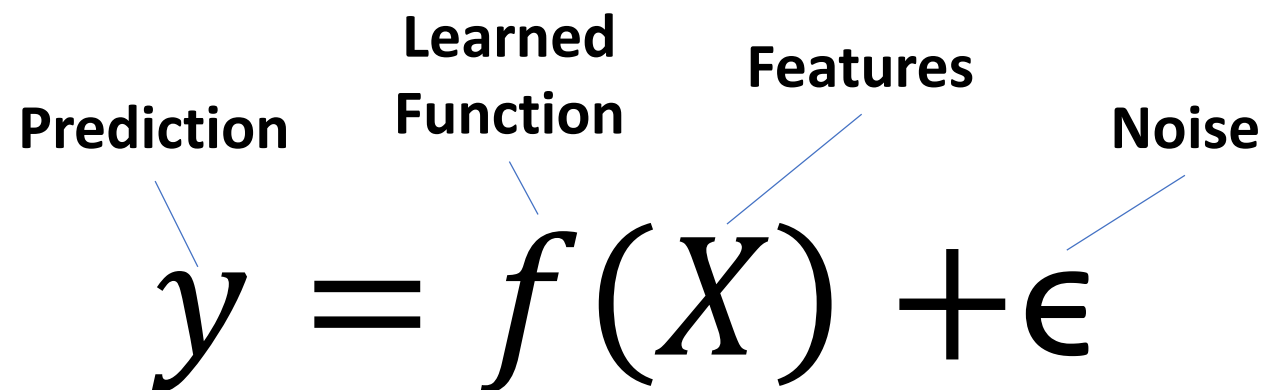
Harvard University, Prof. David Parkes, Spring 2011.

# The Machine Learning Problem

One way of looking at machine learning is that we are trying to approximate (learn) a hidden function in the domain we are interested in so we can make predictions about that domain.

**AGENT**

Prediction      Learned Function      Features      Noise

$$y = f(X) + \epsilon$$


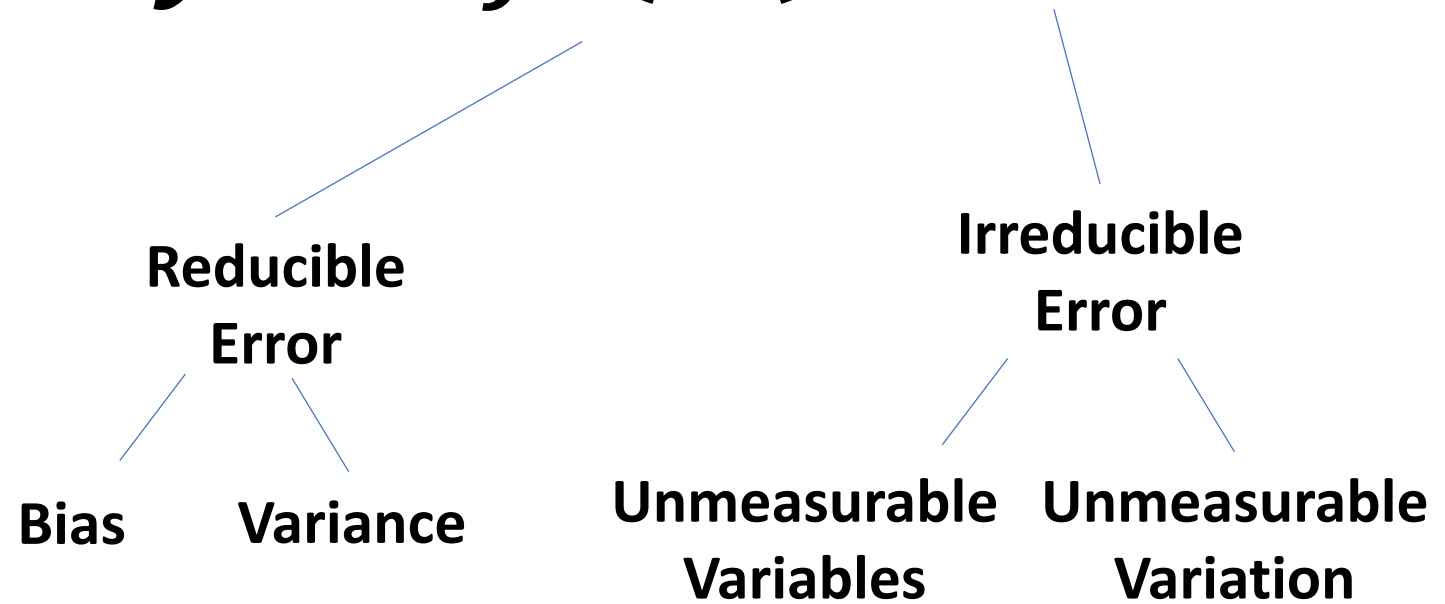
---

$$y_W = F_W(X_W)$$

**WORLD**

# Sources of Error in Machine Learning Predictions

$$y = f(X) + \epsilon$$



# Bias

---

Bias is the systematic error in a learning algorithm.

---

Bias is introduced by approximating a complex domain with a simpler model.

---

Bias is introduced when the model structure does not “fit” the domain.

---

For example, if your domain is non-linear and you try to represent it with a linear model, then there will be systematic errors.

---

No amount of data can fix the bias in a learning algorithm.

# Kinds of Biases: Restriction Bias

A learning algorithm does not consider all possible classifiers / regressors but restricts the learning to task to a particular kind of solution.

For example:

- Linear Classifier classifies items by partitioning the feature space with a hyperplane.
- Decision Tree classifies items by projecting a tree hierarchy onto feature space.
- Neural Network classifies items through the ability of features to activate neurons in a hierarchical neural structure.

# Kinds of Biases: Preference Bias

Among possible hypotheses that a learning algorithm may consider, prefer some hypotheses over others.

For example:

- Linear Classifier/Regressor: prefer weights on linear equations to be small. (Ridge and Lasso)
- Decision Tree: prefer shallow trees; prefer fewer nodes; prefer higher information gain; etc.
- Neural Network prefer fewer hidden units.

# Variance

---

Variance is the error in prediction that can be attributed to the training set.

---

Classification and regression algorithms require a training set to optimize the parameters of the model it creates.

---

Different training sets can result in different models and different prediction algorithms.

---

The more susceptible a machine learning algorithm is to differences in the training set, the more variance it has.



# Variance in a High Complexity Model

Assume we fit  
a high-order  
polynomial

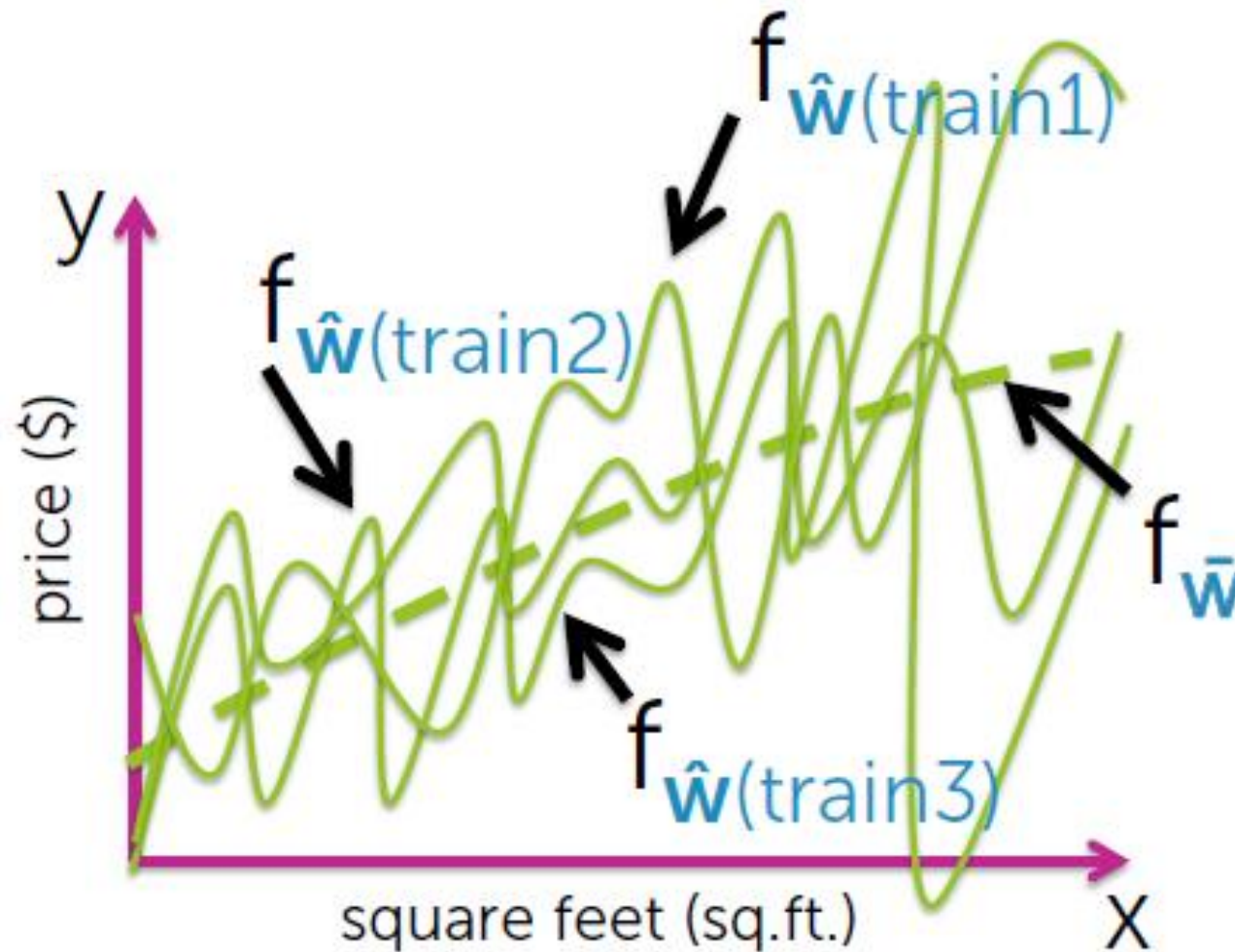


Figure Credit: Emily Fox  
& Carlos Guestrin, University  
of Washington

# Bias and Variance

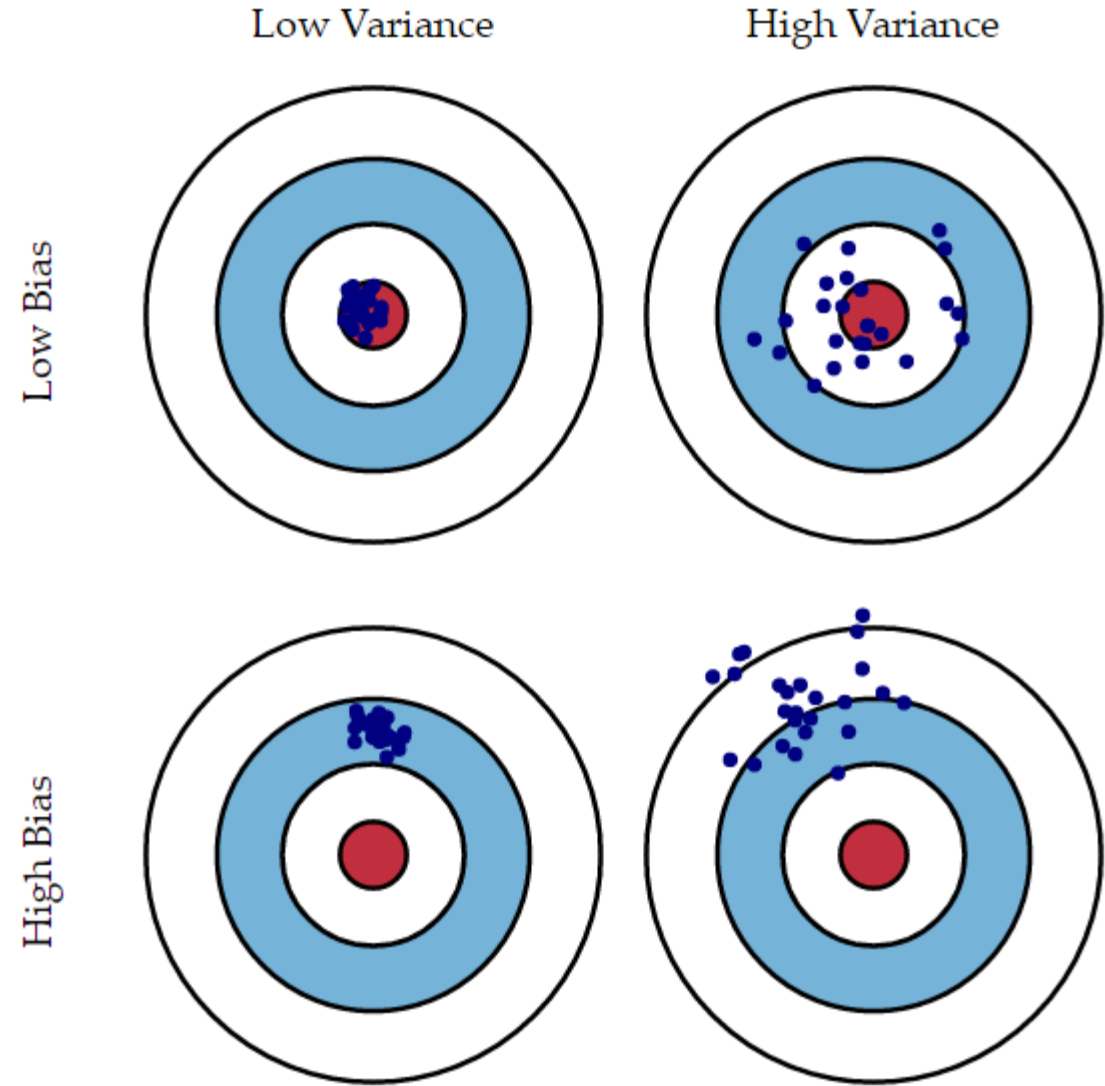
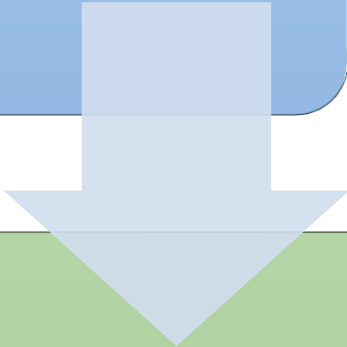


Figure Credit : [An Introduction to Statistical Learning](#) by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

# The Bias-Variance Tradeoff

Ideally, to improve a machine learning algorithm, we would like to decrease bias AND decrease variance.



Unfortunately, a decrease in bias often results in an increase in variance and vice-versa.

# The Problem of Overfitting

---

Overfitting is said to occur when the test set error is much greater than the training set error.

---

The greater the complexity of a model, the more likely it is to fit noise or spurious patterns in the training data.

---

A simpler model is more immune to noise, but it is unable to capture more complex relationships.

---

So the game becomes finding an optimal balance between bias and variance.

# The Bias-Variance Tradeoff

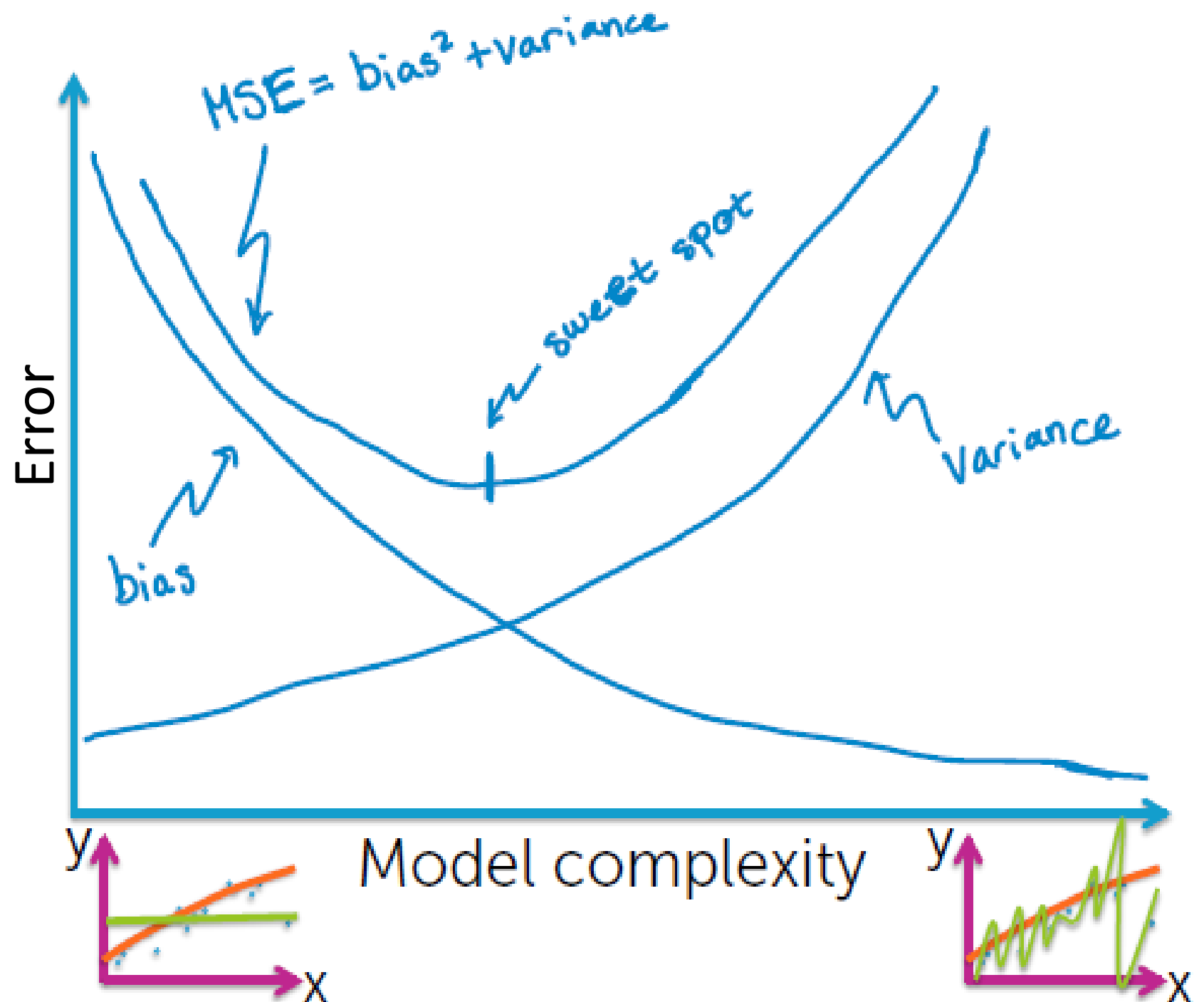


Figure Credit: Emily Fox  
& Carlos Guestrin, University  
of Washington, Coursera  
Machine Learning Specialization.

# Ways to Address Overfitting: **Avoid It**



1

**Use More Data:** Need at least 5-10 x data for each additional parameter.



2

**Use Cross-Validation:** Make better use of existing data by using cross-validation. This requires greater computation.



3

**Use Simpler Models:** *Occam's Razer*: prefer simpler explanations, restrict the classes of models to consider.

## Ways to Address Overfitting: Regularization

- Explicitly penalize model complexity.
- Example: Decision Trees: limit tree depth.
- Example: Ridge Regression -  $\min_w (RMSE) + \lambda w \cdot w$  for suitable a  $\lambda$ .

# Simple Linear Regression

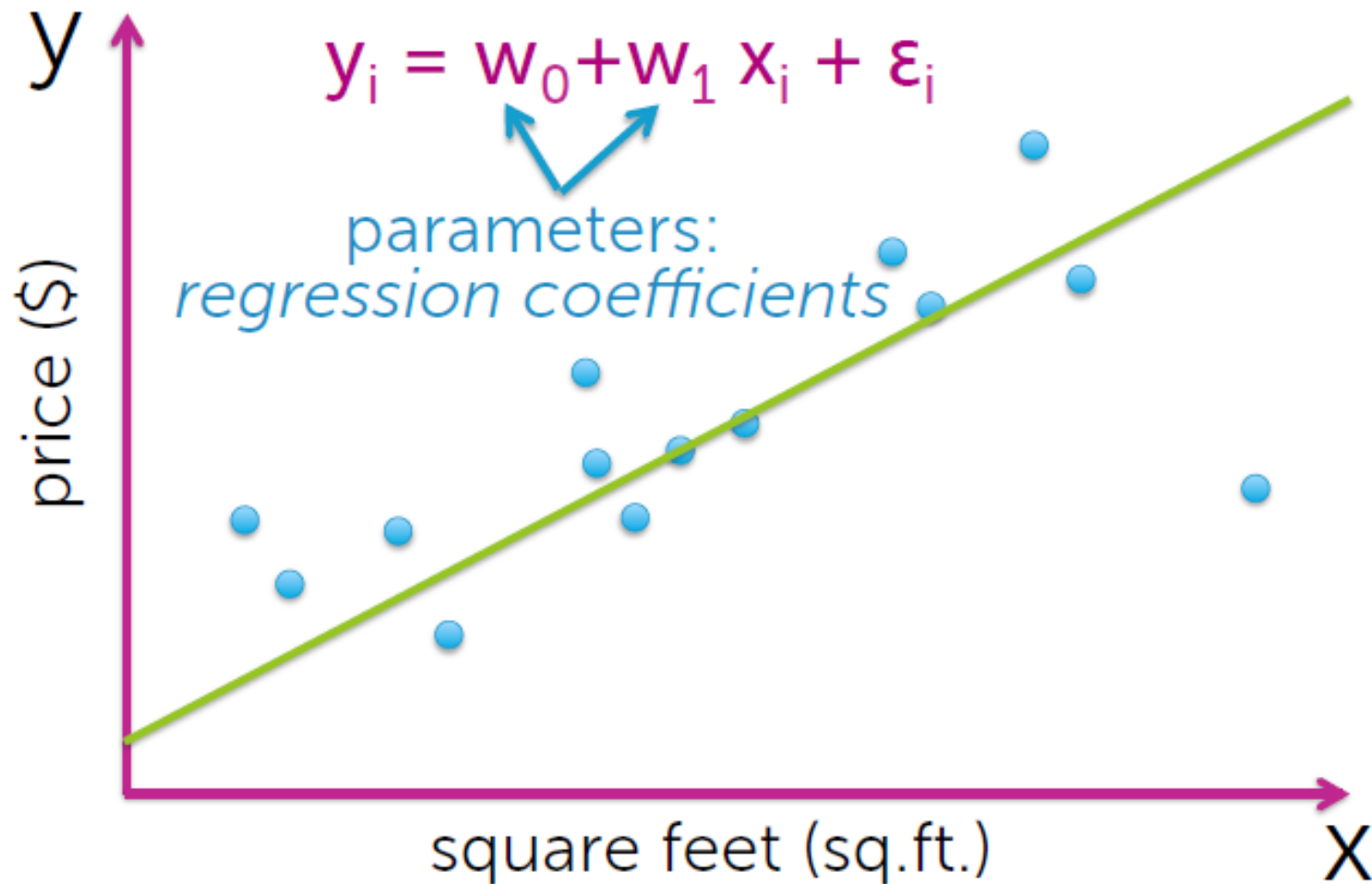


Figure Credit: Emily Fox  
& Carlos Guestrin, University  
of Washington



# Multiple Linear Regression

Add more inputs

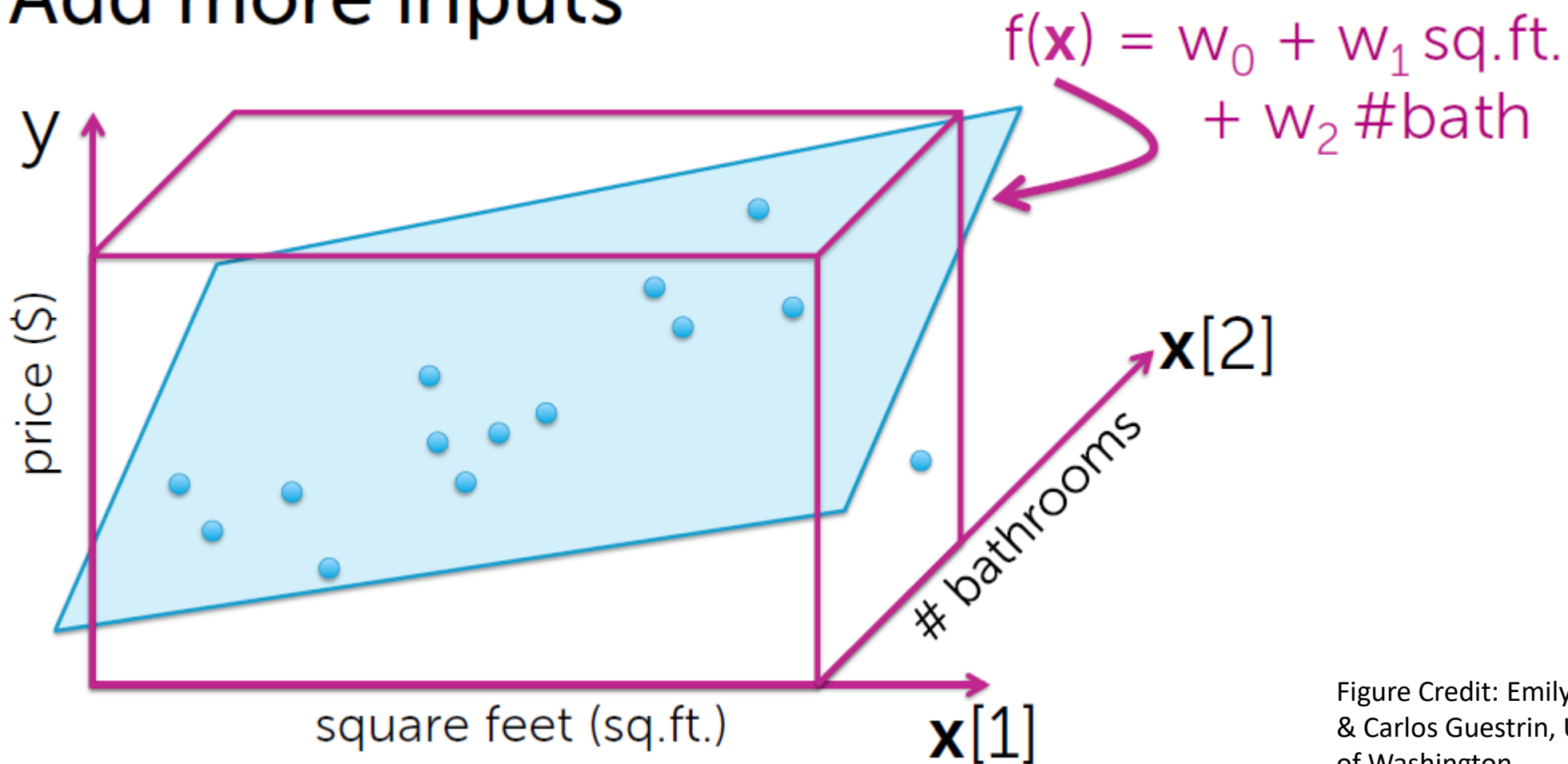


Figure Credit: Emily Fox  
& Carlos Guestrin, University  
of Washington

# Polynomial Regression

Model:

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p + \varepsilon_i$$

treat as different **features**

feature 1 = 1 (constant)    parameter 1 =  $w_0$

feature 2 =  $x$     parameter 2 =  $w_1$

feature 3 =  $x^2$     parameter 3 =  $w_2$

...

...

feature  $p+1$  =  $x^p$     parameter  $p+1$  =  $w_p$

Figure Credit: Emily Fox  
& Carlos Guestrin, University  
of Washington

## Error Sum of Squares

$$SSE = \sum_{i=1}^n \left( y_i - \sum_{j=1}^p x_{ij} w_j \right)^2$$

# LASSO and Ridge Regression

---

A tuning parameter is added which lets you change the complexity or smoothness of the model.

---

The regularization value imposes a special penalty on complex models.

# L2 Regularization – Ridge Regression

- The L2 penalty is the sum of the square of the weights.
- Adds “squared magnitude” of coefficient as penalty term to the loss function.
- Penalizes large coefficients, which are associated with overfitting in linear regression.

$$\sum_{i=1}^n \left( y_i - \sum_{j=1}^p x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^p w_j^2$$

# Change in Weights as Lambda Increase (L2)

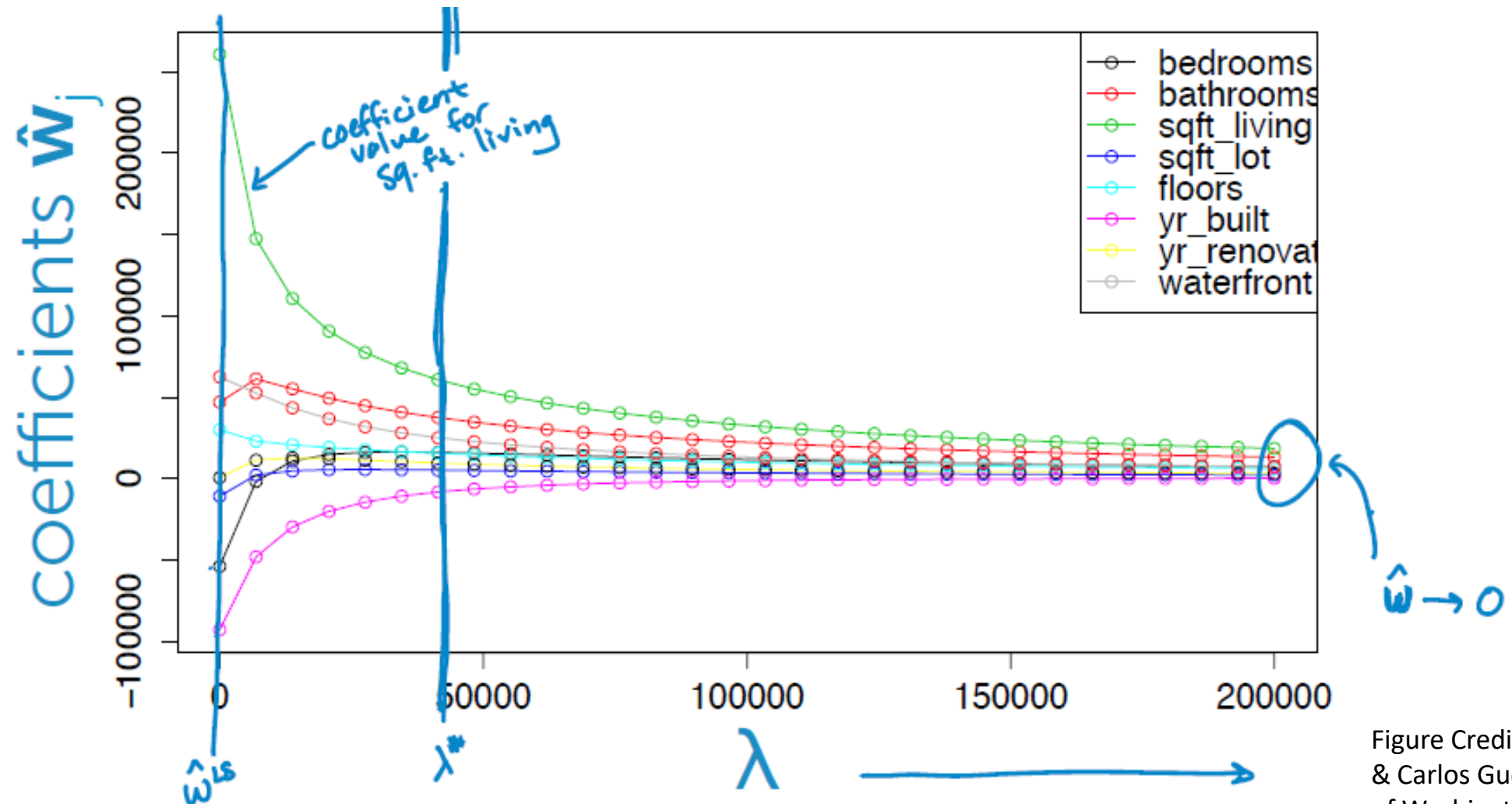


Figure Credit: Emily Fox & Carlos Guestrin, University of Washington

# L1 Regularization – LASSO Regression

- The L1 penalty is the sum of the weight magnitudes.
- Lasso shrinks the less important feature's coefficient to zero removing some feature altogether. This works well for feature selection in cases having a huge number of features.
- As lambda increases, the number of non-zero weights decreases.

$$\sum_{i=1}^n \left( y_i - \sum_{j=1}^p x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^p |w_j|$$

## Change in Weights as Lambda Increase (L2)

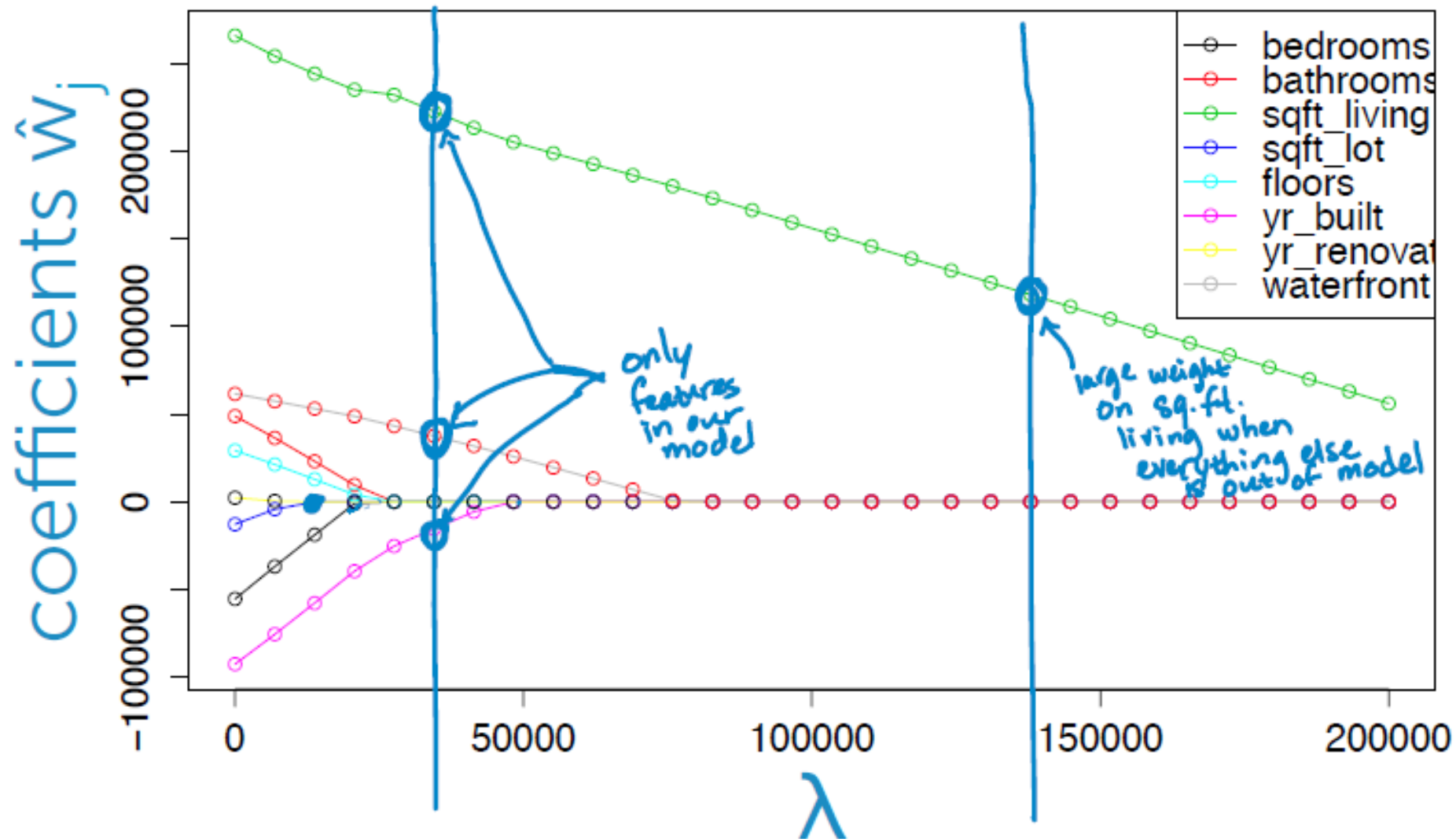
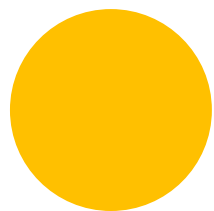
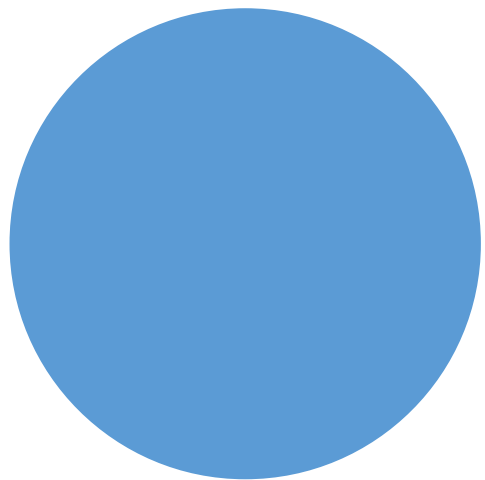


Figure Credit: Emily Fox & Carlos Guestrin, University of Washington



## L1 and L2 Summary

<b>L2 regularization</b>	<b>L1 regularization</b>
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection










# Preventing Overfitting with Ridge and Lasso Regression

Scott O'Hara

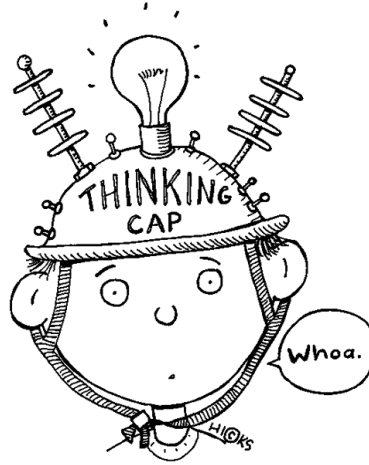
# Other Slides

# A Highly Biased Classification Algorithm ()



Labels:	Not Dog	Dog	Dog	Dog	Dog	Dog	Not Dog
Training Set:							

# A Highly Biased Classification Algorithm (2)



Predictions:

**Dog**

**Dog**

**Dog**

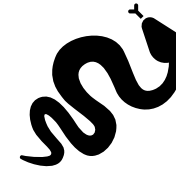
**Dog**

**Dog**

**Dog**

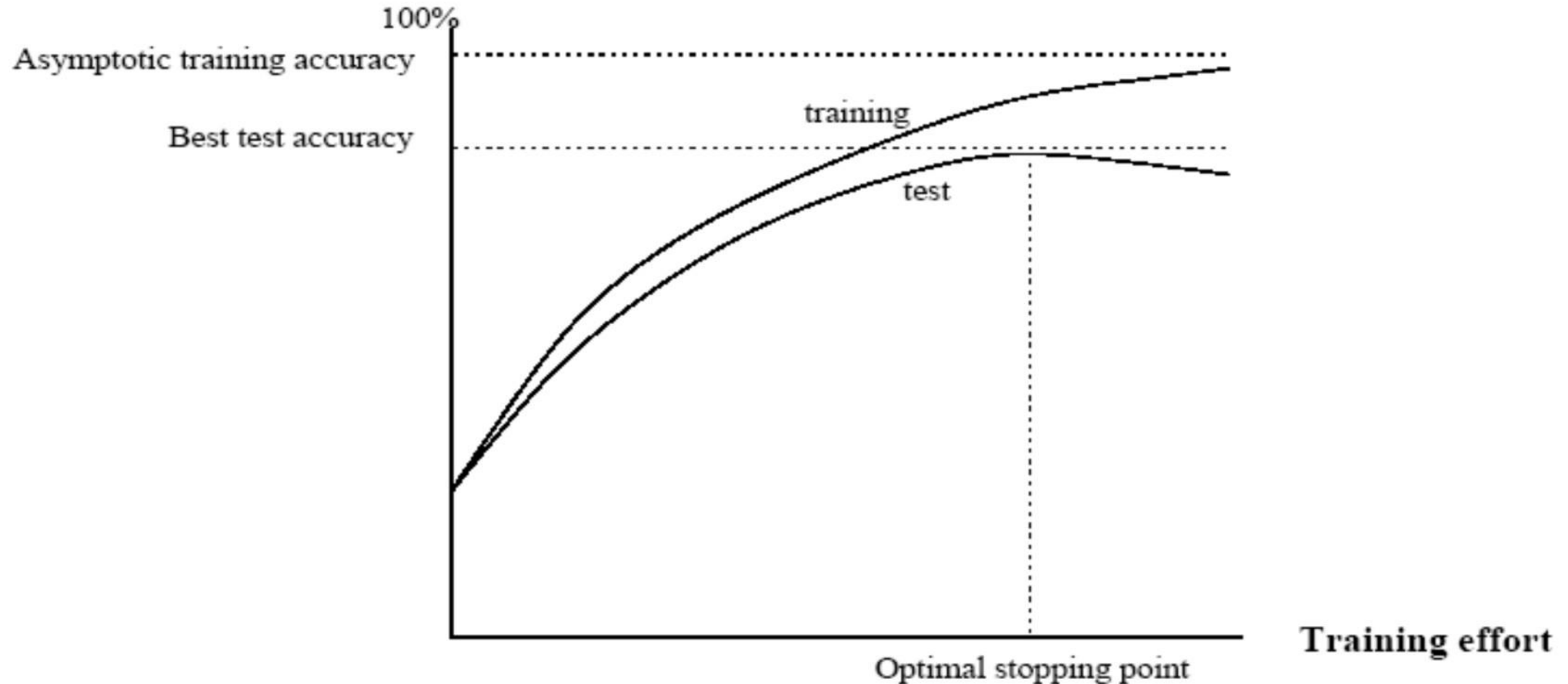
**Dog**

Test Set:



Accuracy: 43%

# More Data Reduces Over-Fitting



# Inductive Bias

- In order for learning to be possible, one needs to make assumptions about what it is one wants to learn. This implies that some true concepts are considered more likely than others.
- Consider Nelson Goodman's Grue concept:
  - "All emeralds are green"
  - "All emeralds are grue" (grue = green before Jan 2100, blue afterwards)
  - Clearly, we are biased toward the first concept.

# Inductive Bias $\neq$ Bias

- **Inductive Bias** refers to the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered. E.g., Occam's razor, minimum description length, minimum cross-validation error etc.
- **Bias** refers to the systematic error of a prediction algorithm, as we have seen.