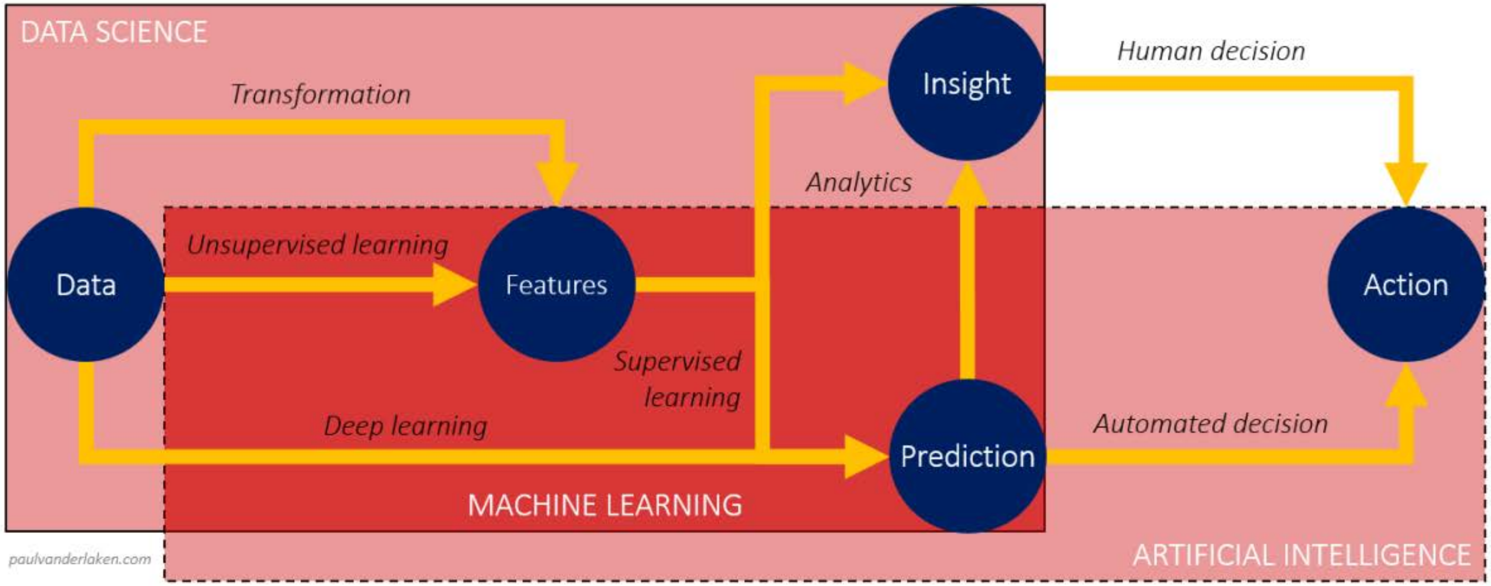
Introduction to Machine Learning



ML Boot Camp

Course Overview

- This course provides a survey of machine learning (ML) techniques, teaching you to implement ML models with open-source software for applications in artificial intelligence (AI).

- The course will enable you to explore and learn from data, finding underlying patterns useful for data reduction, feature analysis, prediction, and classification.

Course Goals

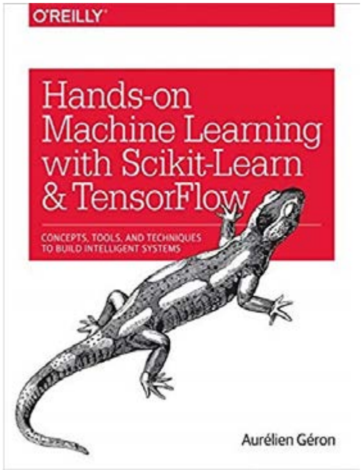
1. Describe ML methods for a variety of applications in AI.
2. Compare traditional statistical methods and ML methods.
3. Distinguish between supervised, unsupervised and reinforcement learning methods.
4. Design experiments for training and testing ML models.
5. Evaluate ML models for regression and classification.
6. Construct trees, random forests, gradient boosted models, and artificial neural networks.
7. Explore deep learning models for vision and natural language processing.

Course Outline

Module	Topic	Date
1	Introduction to Machine Learning	13 SEP 19
2	Supervised Learning for Classification	20 SEP 19
3	Supervised Learning for Regression	27 SEP 19
4	Trees, Random Forests, and Gradient Boosting	04 OCT 19
5	Unsupervised Learning	11 OCT 19
6	Artificial Neural Networks	01 NOV 19
7	Deep Learning for Computer Vision	08 NOV 19
8	Deep Learning for Natural Language Processing	15 NOV 19
9	Neural Network Autoencoders	22 NOV 19
10	Reinforcement Learning	06 DEC 19

Course Materials

Referenced Textbooks

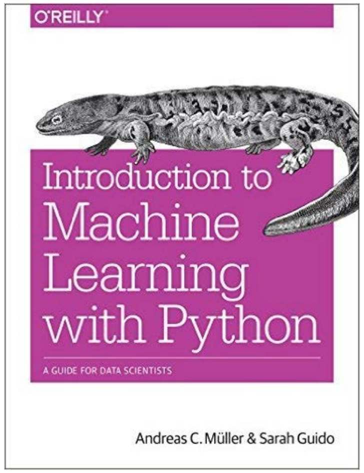


O'REILLY

Hands-on Machine Learning with Scikit-Learn & TensorFlow

CONCEPTS, TOOLS, AND TECHNIQUES TO BUILD INTELLIGENT SYSTEMS

Aurélien Géron

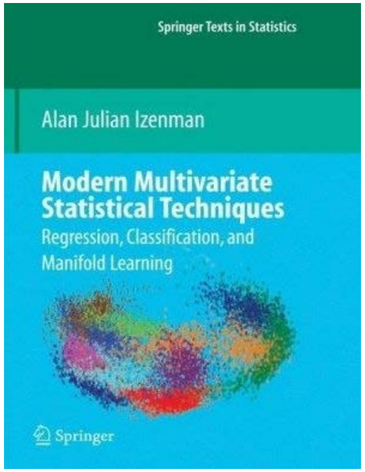


O'REILLY

Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

Andreas C. Müller & Sarah Guido



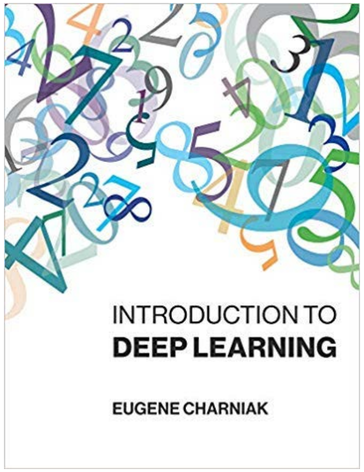
Springer Texts in Statistics

Alan Julian Izenman

Modern Multivariate Statistical Techniques

Regression, Classification, and Manifold Learning

Springer



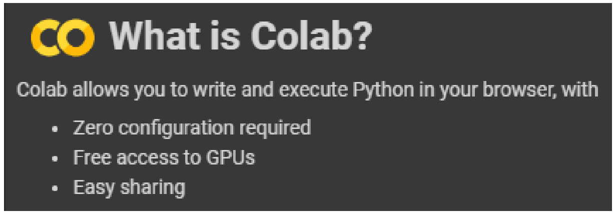
INTRODUCTION TO DEEP LEARNING

EUGENE CHARNIAK

Required Software

python

scikit learn + TensorFlow



What is Colab?

Colab allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

QUESTIONS????

Big Data is Everywhere

- We are in the era of **big data**!
  - 40 billion indexed web pages
  - 100 hours of video are uploaded to YouTube every minute

- The rapid advancement of advanced computational methods provides unprecedented opportunities for **understanding large, complex datasets.**

- **Bottom line:** The deluge of data calls for automated methods of data analysis, which is what **machine learning** provides!

What is Machine Learning?

- **Machine learning** is a set of methods that can *automatically* detect patterns in data.
- These uncovered patterns are then used to predict future data, or to perform other kinds of decision-making under uncertainty.
- The key premise is ***learning*** from data!!

What is Machine Learning?

- Addresses the problem of analyzing huge bodies of data so that they can be **understood**.
- Providing techniques to **automate** the analysis and exploration of large, complex data sets.
- Tools, methodologies, and theories from statistics, computer science, etc. for revealing **patterns** in data – critical step in knowledge discovery.

What is Machine Learning?

- **Driving Forces:**
  - **Explosive growth** of data in a great variety of fields
    - Cheaper storage devices with higher capacity
    - Faster communication
    - Better database management systems
  - Rapidly increasing computing power

- We want to **make the data work for us!!**

The Learning Problem

- Learning from data is used in situations where we don’t have any analytic solution, but we do have data that we can use to construct an empirical solution.

- The basic premise of learning from data is the use of a set of observations to uncover an underlying process.

The Learning Problem (cont.)

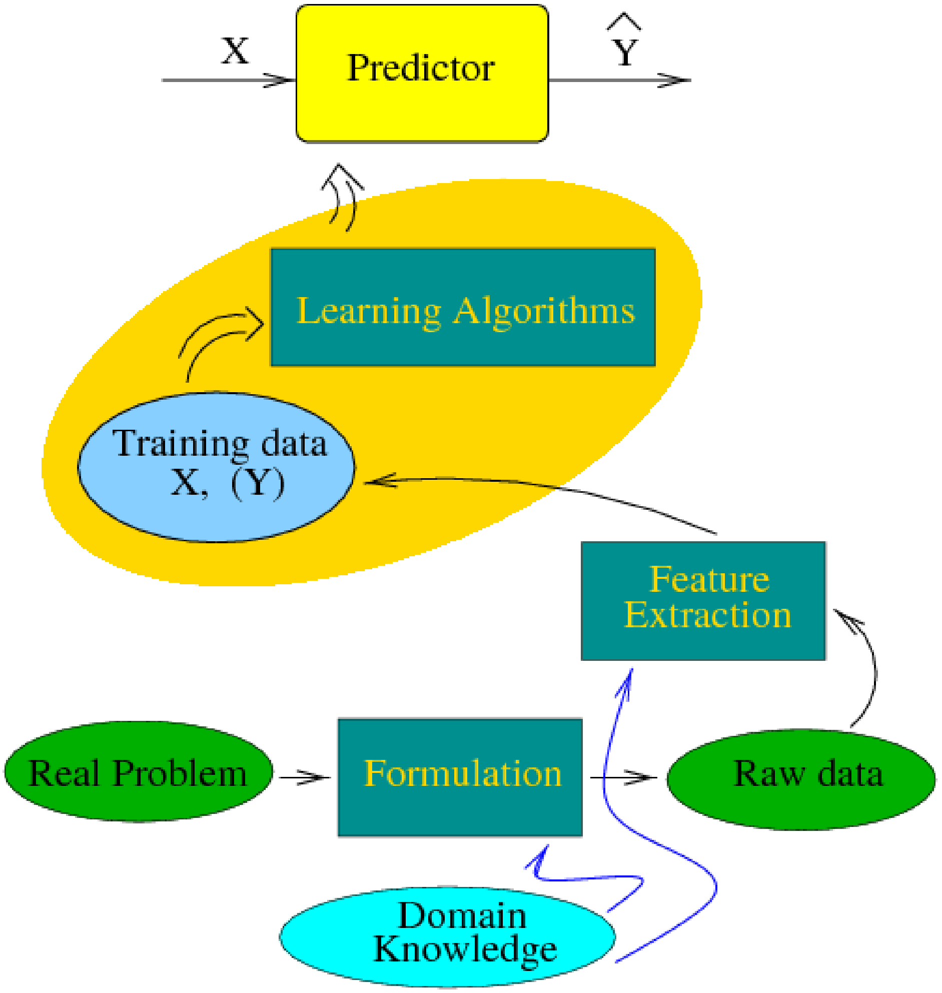
- Suppose we observe the output space Y_i and the input space $X_i = \left(X_{i1}, \dots, X_{ip}\right) \forall\, i = 1, \dots, n$

- We believe that there is a *relationship* between *Y* and at least one of the *X*’s.

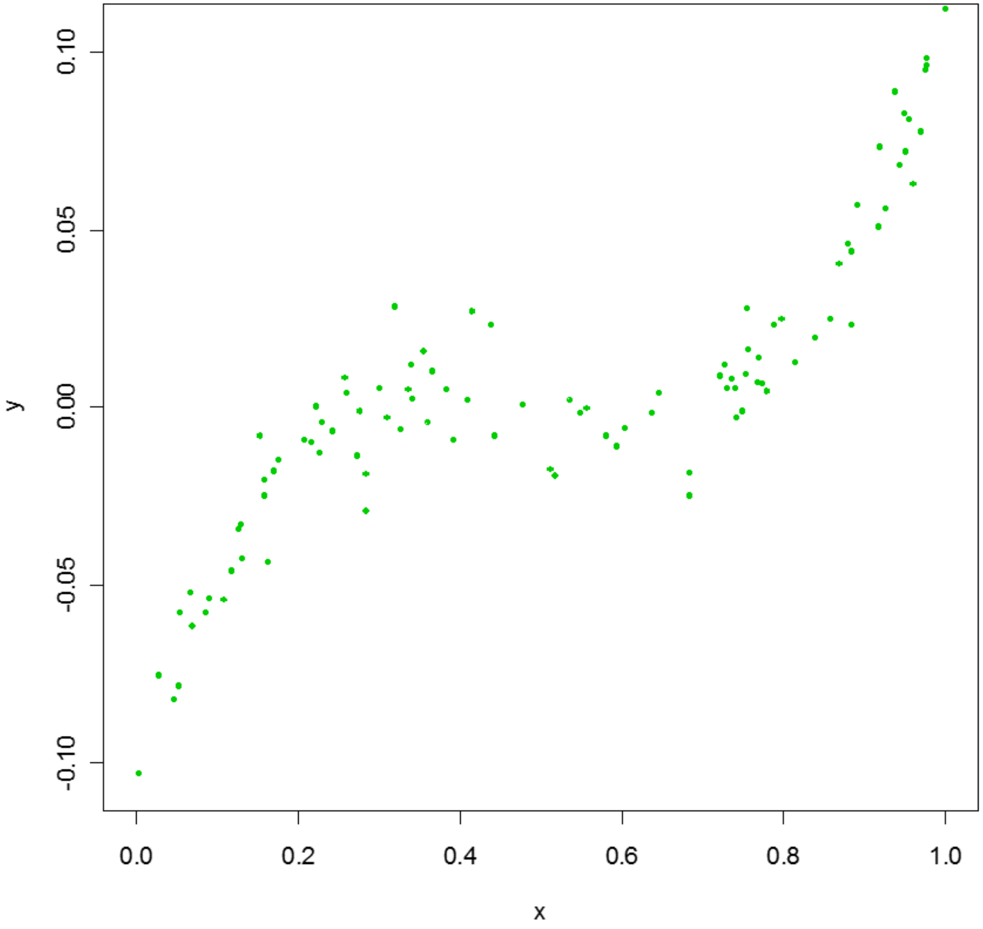
- We can model the relationship as: $Y_i = f(\boldsymbol{X_i}) + \varepsilon_i$

where *f* is an unknown function and *ε* is a random error (noise) term, independent of ***X*** with mean zero.

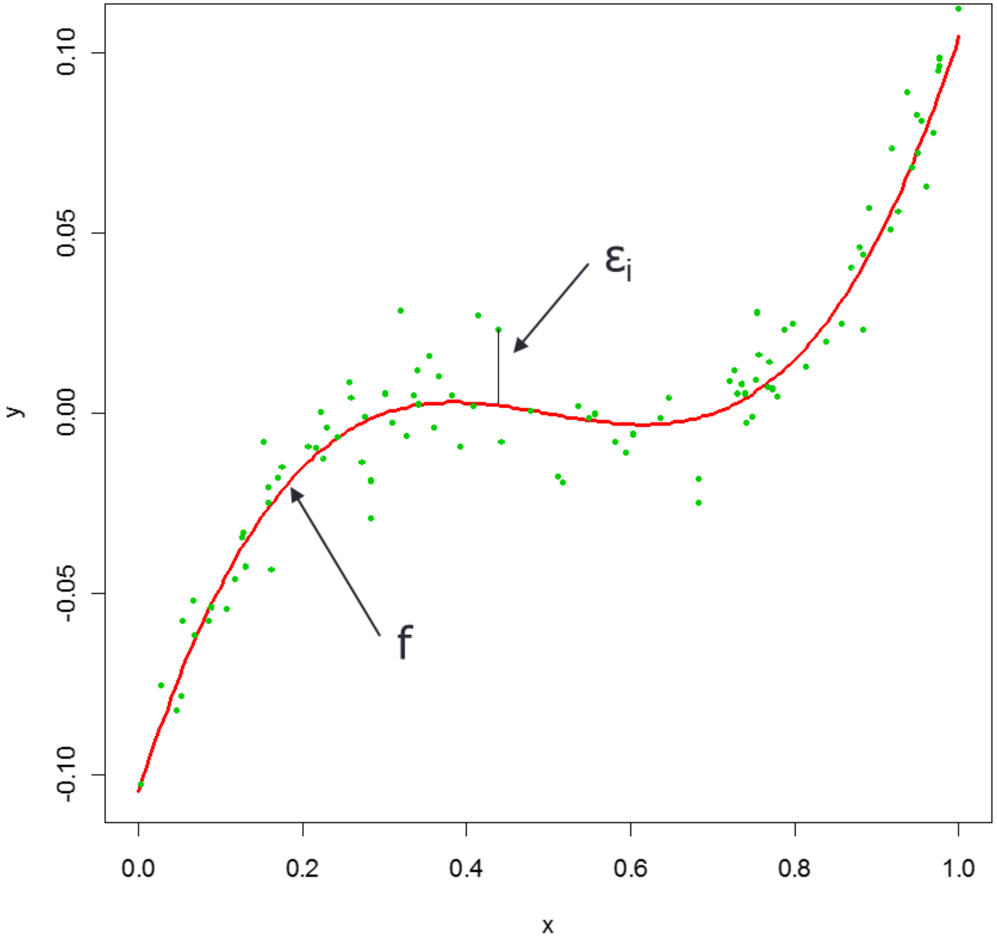
The Learning Problem (cont.)



The Learning Problem: Example

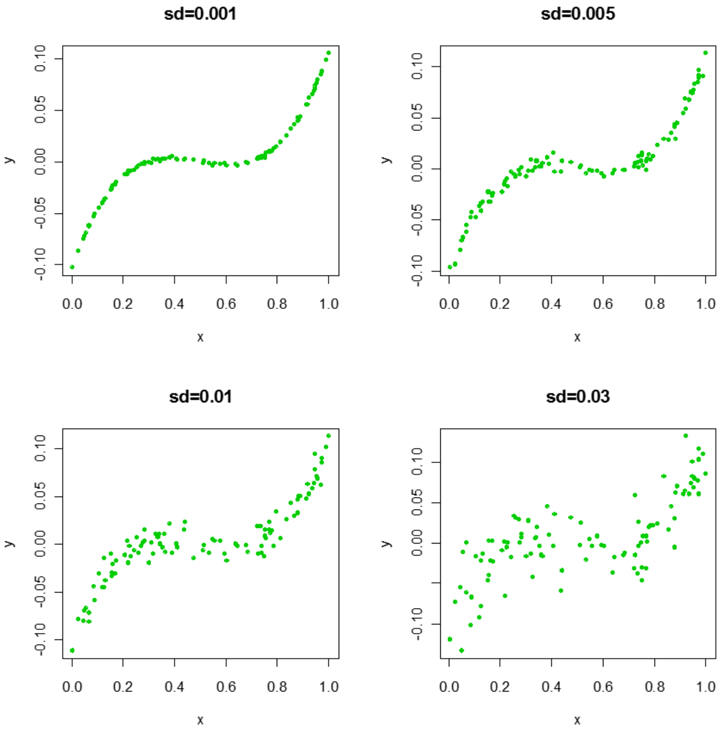


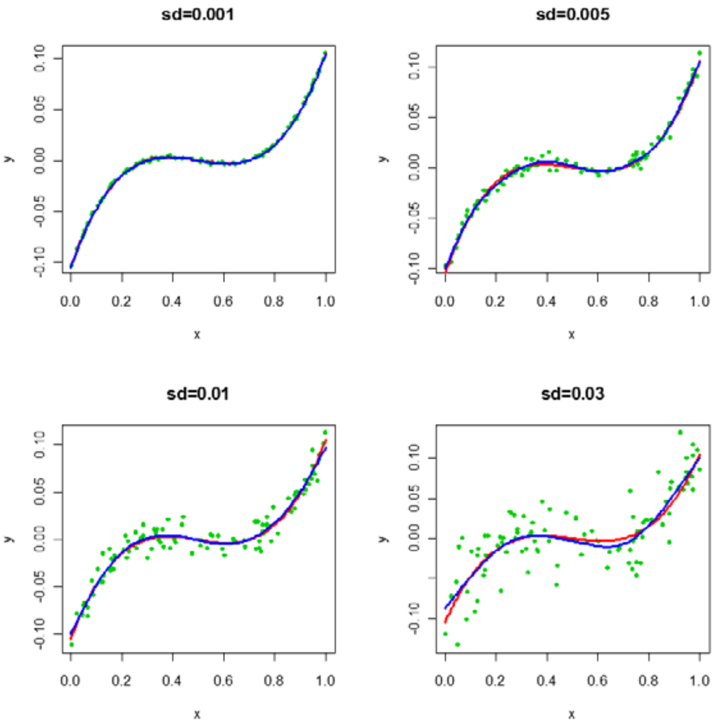
The Learning Problem: Example (cont.)



The Learning Problem: Example (cont.)

- Different estimates for the target function *f* that depend on the standard deviation of the *ε*'s





Why do we estimate *f*?

- We use modern machine learning methods to estimate *f* by *learning* from the data.

- The target function *f* is unknown.

- We estimate *f* for two key purposes:
  - Prediction
  - Inference

Prediction

- By producing a good estimate for *f* where the variance of *ε* is not too large, then we can make accurate predictions for the response variable, *Y*, based on a new value of ***X***.

- We can predict *Y* using $\hat{Y} = \hat{f}$(***X***)

where $\hat{f}$ represents our estimate for *f*, and $\hat{Y}$ represents the resulting prediction for *Y*.

Prediction (cont.)

- The accuracy of $\hat{Y}$ as a prediction for Y depends on:
  - Reducible error
  - Irreducible error

- Note that $\hat{f}$ will not be a perfect estimate for f; this inaccuracy introduces error.

Prediction (cont.)

- This error is ***reducible*** because we can potentially improve the accuracy of the estimated (i.e., hypothesis) function $\hat{f}$ by using the most appropriate learning technique to estimate the target function *f*.

- Even if we could perfectly estimate *f*, there is still variability associated with *ε* that affects the accuracy of predictions = ***irreducible*** error.

Prediction (cont.)

- Average of the squared difference between the predicted and actual value of *Y*.
- Var(*ε*) represents the *variance* associated with *ε*.

$$E[(Y - \hat{f}(X))^2 | X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{Reducible} + \underbrace{\mathrm{Var}(\epsilon)}_{Irreducible}$$

- Our aim is to minimize the reducible error!!

Example: Direct Mailing Prediction

- We are interested in predicting how much money an individual will donate based on observations from 90,000 people on which we have recorded over 400 different characteristics.

- We don't care too much about each individual characteristic.

- Learning Problem:
  - For a given individual, should I send out a mailing?

Inference

- Instead of prediction, we may also be interested in the type of relationship between *Y* and the *X*’s.

- Key questions:
  - Which predictors actually affect the response?
  - Is the relationship positive or negative?
  - Is the relationship a simple linear one or is it more complicated?

Example: Housing Inference

- We wish to predict median house price based on numerous variables.

- We want to *learn* which variables have the largest effect on the response and how big the effect is.

- For example, how much impact does the number of bedrooms have on the house value?

How do we estimate *f*?

- First, we assume that we have observed a set of **training data**.

$$\{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_n, Y_n)\}$$

- Second, we use the training data and a **machine learning method** to estimate *f*.
  - Parametric or non-parametric methods

Parametric Methods

- This reduces the *learning problem* of estimating the target function *f* down to a problem of estimating a set of **parameters**.

- This involves a two-step approach…

Parametric Methods (cont.)

- **Step 1:**
  - Make some assumptions about the functional form of *f*. The most common example is a linear model:

$$f(\mathbf{X}_i) = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}$$

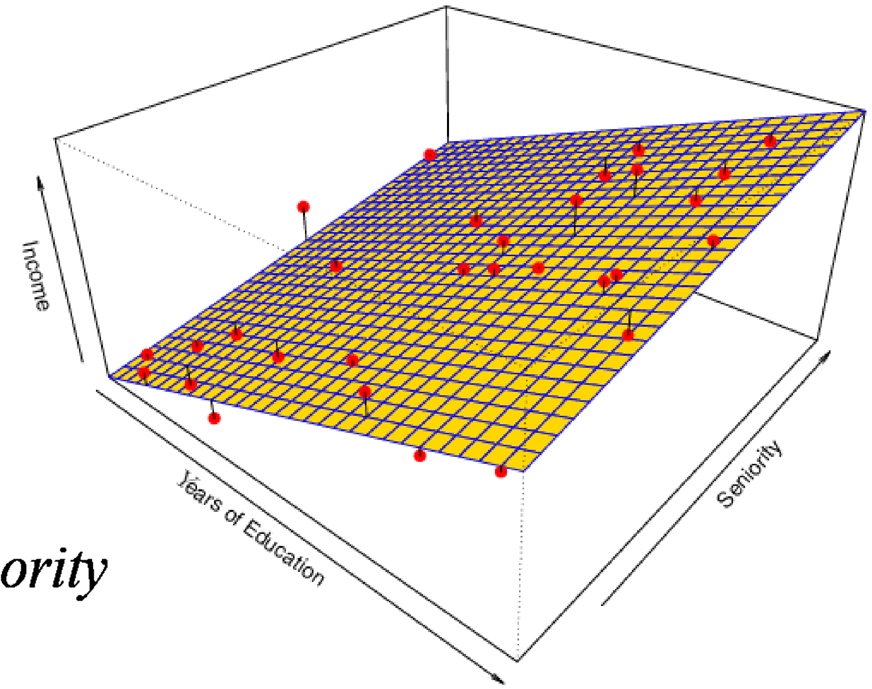
  - In this course, we will examine far more complicated and flexible models for *f*.

Parametric Methods (cont.)

- **Step 2:**
  - We use the *training data* to fit the model (i.e., estimate *f*….the unknown parameters).
  - The most common approach for estimating the parameters in a linear model is via ordinary least squares (OLS) linear regression.
  - However, there are superior approaches, as we will see in this course.

Example: OLS Regression Estimate

- Even if the standard deviation is low, we will still get a bad answer if we use the incorrect model.



$$f = \beta_0 + \beta_1 \times Education + \beta_2 \times Seniority$$

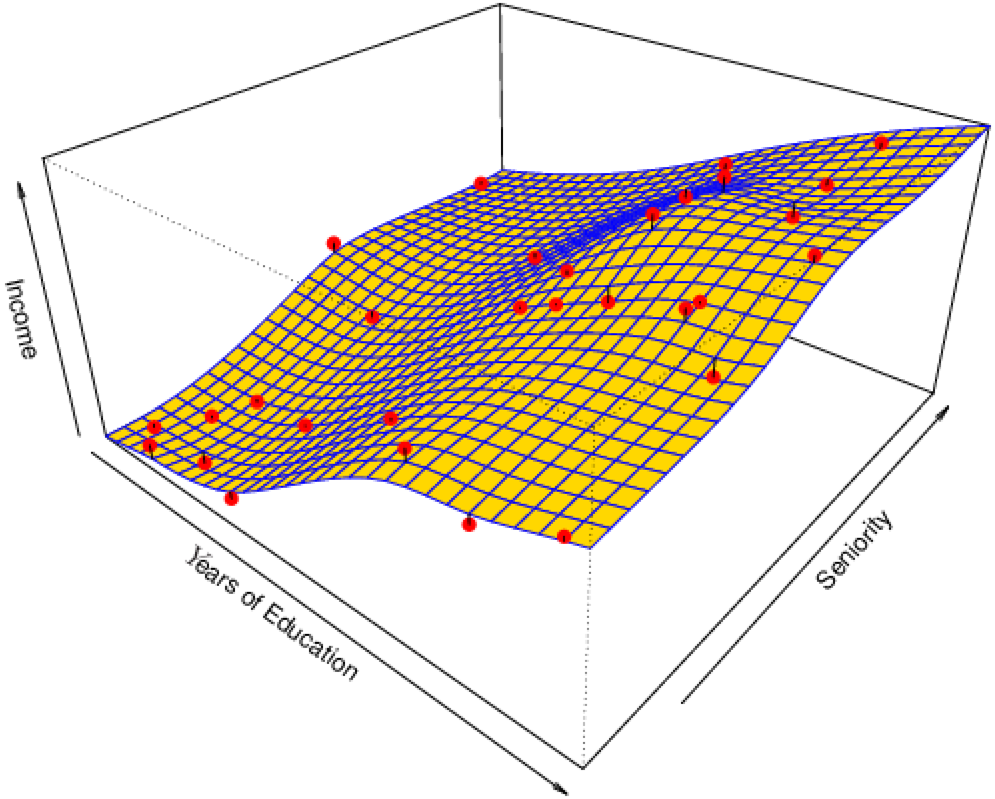
Non-Parametric Methods

- As opposed to parametric methods, these do not make explicit assumptions about the functional form of *f*.

- Advantages:
  - Accurately fit a wider range of possible shapes of *f*.

- Disadvantages:
  - Requires a very large number of observations to acquire an accurate estimate of *f*.

Example: Thin-Plate Spline Estimate



- Non-linear regression methods are more flexible and can potentially provide more accurate estimates.

- However, these methods can run the risk of over-fitting the data (i.e., follow the errors, or noise, too closely), so too much flexibility can produce poor estimates for *f*.

Learning Algorithm Trade-off

- There are always two aspects to consider when designing a machine learning algorithm:
  - Try to fit the data well
  - Be as robust as possible

- The predictor that you have generated using your training data must also work well on new data.

Learning Algorithm Trade-off (cont.)

- When we create predictors, usually the simpler the predictor is, the more robust it tends to be in the sense of being able to be estimated reliably.

- On the other hand, the simple models do not fit the training data aggressively.

Learning Algorithm Trade-off (cont.)

- Training Error vs. Testing Error:
  - Training error → reflects whether the data fits well
  - Testing error → reflects whether the predictor actually works on new data

- Bias vs. Variance:
  - Bias → how good the predictor is, on average; tends to be smaller with more complicated models
  - Variance → tends to be higher for more complex models

Learning Algorithm Trade-off (cont.)

- Fitting vs. Over-fitting:
  - If you try to fit the data too aggressively, then you may over-fit the training data. This means that the predictors works very well on the training data, but is substantially worse on the unseen test data.

- Empirical Risk vs. Model Complexity:
  - Empirical risk → error rate based on the training data
  - Increase model complexity = decrease empirical risk but less robust (higher variance)

Supervised vs. Unsupervised Learning

- Supervised Learning:
  - All the predictors, $\boldsymbol{X_i}$, and the response, Y_i, are observed.
    - Many regression and classification methods

- Unsupervised Learning:
  - Here, only the $\boldsymbol{X_i}$'s are observed (not Y_i*'s).*
  - We need to use the $\boldsymbol{X_i}$'s to guess what *Y* would have been, and then build a model form there.
    - Clustering and principal components analysis

Terminology

- **Notation**
  - Input *X*: *feature, predictor,* or *independent variable*
  - Output *Y*: *response, dependent variable*

- **Categorization**
  - Supervised learning vs. unsupervised learning
    - *Key question*: Is *Y* available in the training data?
  - Regression vs. Classification
    - *Key question*: Is *Y* quantitative or qualitative?

Terminology (cont.)

- **Quantitative:**
  - Measurements or counts, recorded as numerical values (height, temperature, etc.)

- **Qualitative:** group or categories
  - Ordinal: possesses a natural ordering (e.g., shirt sizes)
  - Nominal: just name the categories (marital status, gender, etc.)

Terminology (cont.)

Feature X Label Y

Training Samples

x_1	x_2	x_3	…	x_p	Y
3	5	2	…	1	A
4	2	3	…	2	B
…	…	…	…	…	…
4	2	3	…	3	A

Model

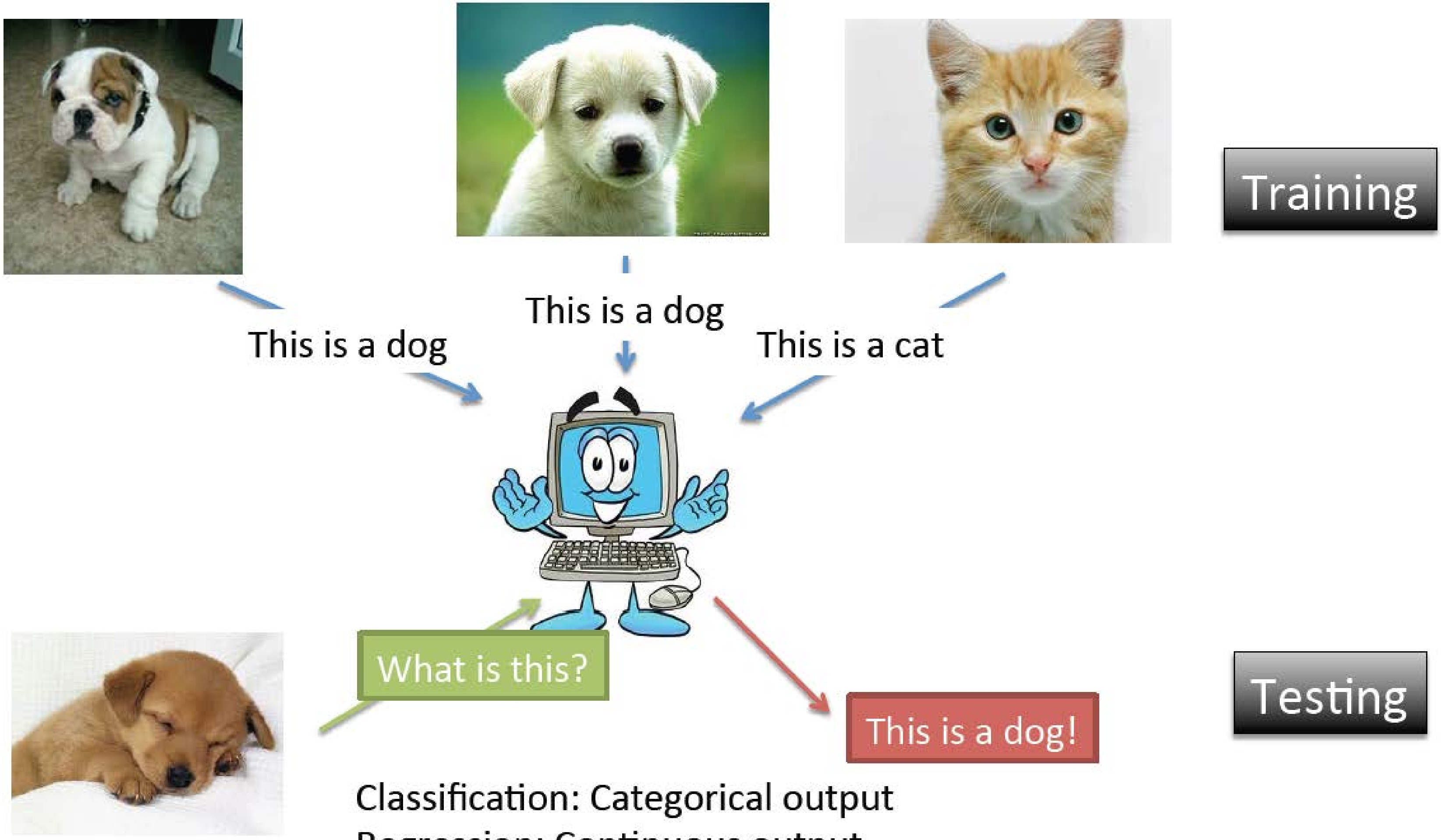
Feature X

Label Y (unknown)

Testing

x_1	x_2	x_3	…	x_p	Y
5	5	2	…	1	?
2	2	1	…	2	?
…	…	…	…	…	…
1	3	2	…	4	?

Supervised Learning



This is a dog

This is a dog

This is a cat

Training

What is this?

This is a dog!

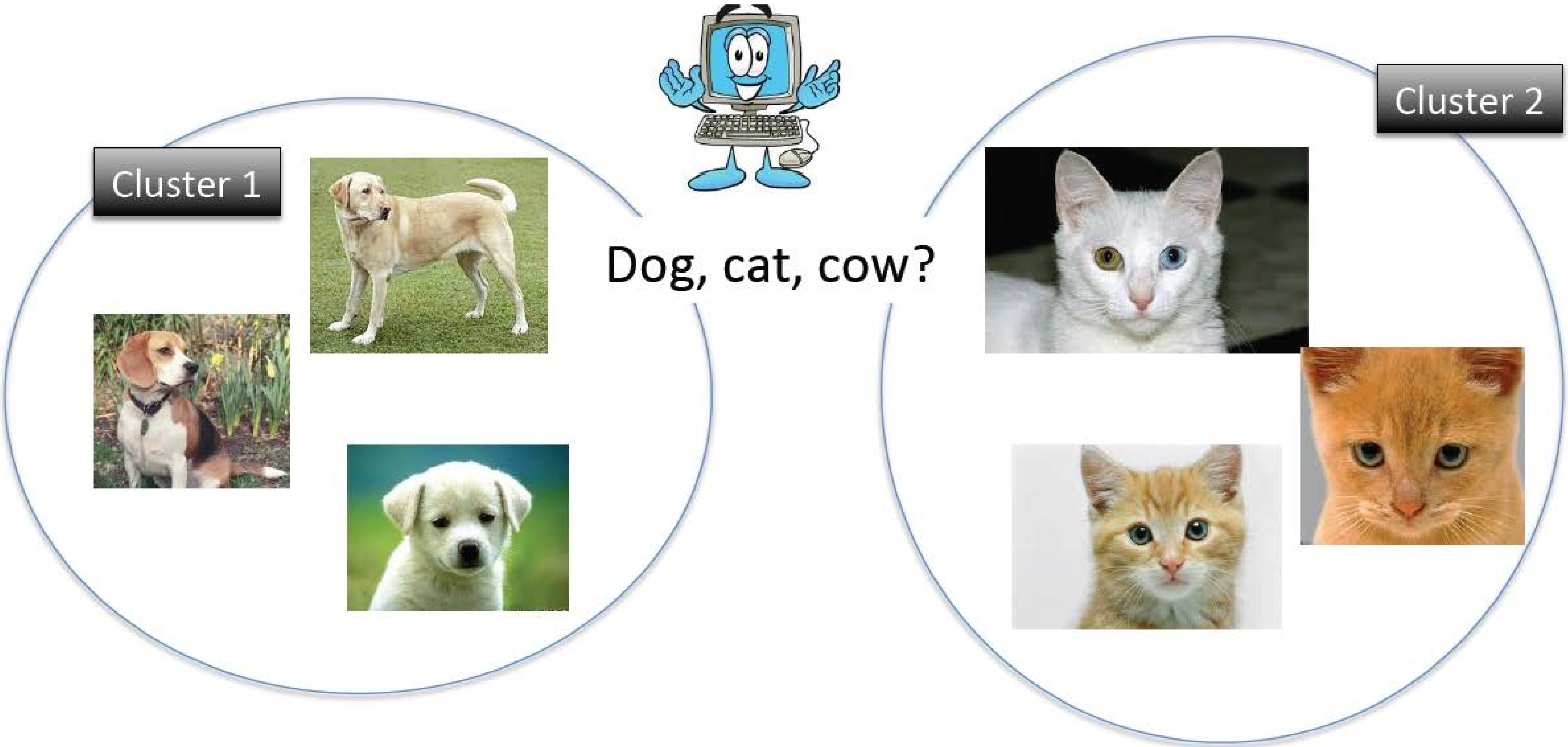
Testing

Classification: Categorical output
Regression: Continuous output

Supervised Learning: Regression vs. Classification

- Regression
  - Covers situations where *Y* is continuous (quantitative)
  - E.g. predicting the value of the Dow in 6 months, predicting the value of a given house based on various inputs, etc.
- Classification
  - Covers situations where *Y* is categorical (qualitative)
  - E.g. Will the Dow be up or down in 6 months? Is this email spam or not?

Unsupervised Learning



Unsupervised: semantic meanings of clusters are not clear

Unsupervised Learning (cont.)

- The training data does not contain any output information at all (i.e., unlabeled data).

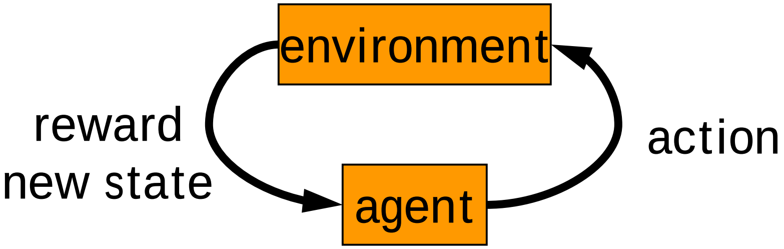
- Viewed as the task of spontaneously finding patterns and structure in input data.

- Viewed as a way to create a higher-level representation of the data and dimension reduction.

Reinforcement Learning

- Generalization of supervised learning

- Learn from interaction with environment to achieve a goal



- Markov decision processes (states, actions, transition, value function, optimal policy)

Assessing Model Accuracy

- For a given set of data, we need to decide which machine learning method produces the **best** results.

- We need some way to measure the quality of fit (i.e., how well its predictions actually match the observed data).

- In regression, we typically use mean squared error (MSE).

Assessing Model Accuracy (cont.)

Suppose we fit a model $\hat{f}(x)$ to some training data $\mathsf{Tr} = \{x_i, y_i\}_1^N$, and we wish to see how well it performs.

- We could compute the average squared prediction error over Tr:

$$\text{MSE}_{\mathsf{Tr}} = \text{Ave}_{i \in \mathsf{Tr}}[y_i - \hat{f}(x_i)]^2$$

This may be biased toward more overfit models.

- Instead we should, if possible, compute it using fresh *test* data $\mathsf{Te} = \{x_i, y_i\}_1^M$:

$$\text{MSE}_{\mathsf{Te}} = \text{Ave}_{i \in \mathsf{Te}}[y_i - \hat{f}(x_i)]^2$$

Assessing Model Accuracy (cont.)

- Thus, we really care about how well the method works on new, unseen test data.

- There is no guarantee that the method with the smallest *training MSE* will have the smallest *test MSE*.

Training vs. Test MSEs

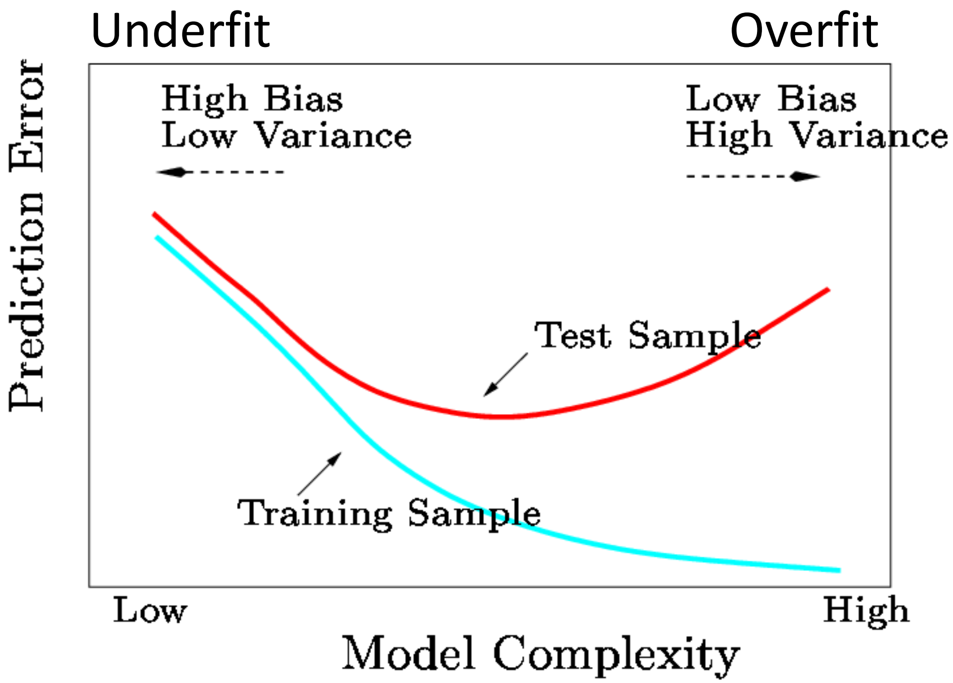
- In general, the more flexible a method is the lower its training MSE will be.

- However, the test MSE may in fact be higher for a more flexible method than for a simple approach like linear regression.

- More flexible methods can generate a wider range of possible shapes to estimate *f* as compared to less flexible and more restrictive methods.

- The less flexible the method, the easier to interpret the model. There is a trade-off between flexibility and model interpretability

Bias-Variance Trade-Off



When selecting a machine learning method, remember that more flexible/complex is not necessarily better!!

- In general, training errors will always decline.
- However, test errors will decline at first (as reductions in bias dominate) but will then start to increase again (as increases in variance dominate).

Bias-Variance Trade-off (cont.)

- The previous graph of test versus training MSEs illustrates a very important trade-off that governs the choice of machine learning methods.
- There are always two competing forces that govern the choice of learning method:
  - bias and variance

Bias of Learning Methods

- Bias refers to the error that is introduced by modeling a real life problem (that is usually extremely complicated) by a much simpler problem.

- Generally, the more flexible/complex a machine learning method is, the **less bias** it will generally have.

Variance of Learning Methods

- Variance refers to how much your estimate for *f* would change by if you had a different training data set.

- Generally, the more flexible/complex a machine learning method is the **more variance** it has.

The Trade-Off: Expected Test MSE

Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $\text{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$.

Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off.*

Test MSE, Bias and Variance

- Thus, in order to minimize the expected test MSE, we must select a machine learning method that simultaneously achieves *low variance* and *low bias*.

- Note that the expected test MSE can never lie below the irreducible error - Var(ε).

QUESTIONS????