Review of Machine Learning

Materials from

- Intel Deep Learning https://www.intel.com/content/www/us/en/developer/learn/course-deep-learning.html
- Introduction to Neural Networks https://www.deeplearning.ai/

Legal Notices and Disclaimers

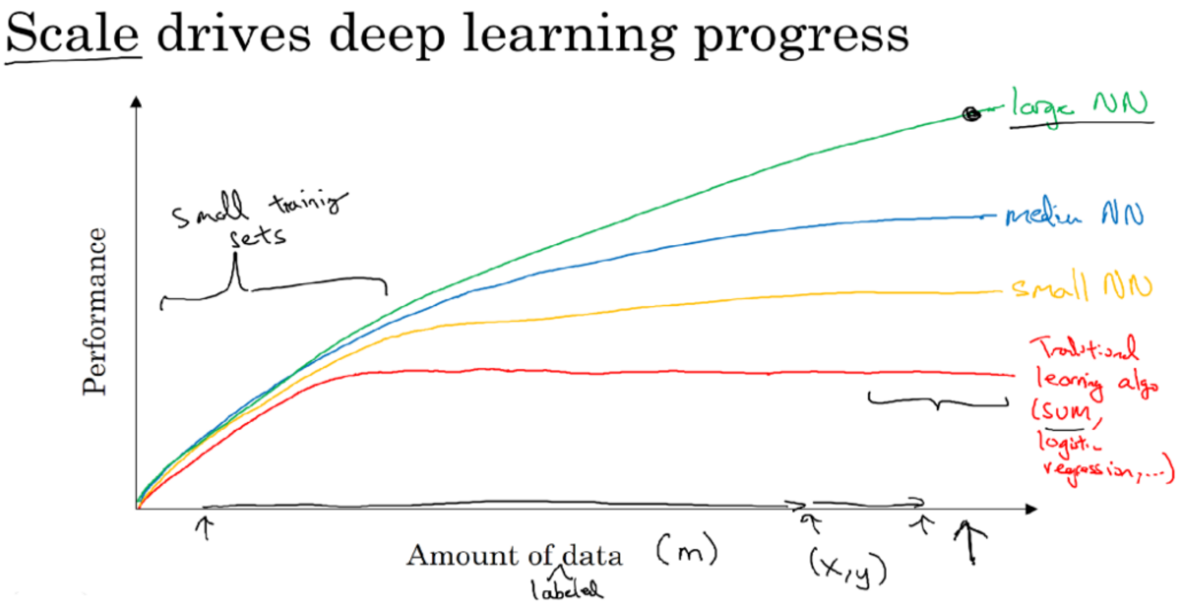
This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

Why is Deep Learning Taking Off?

Deep learning is taking off due to a large amount of data available through the digitization of the society, faster computation and innovation in the development of neural network algorithm.

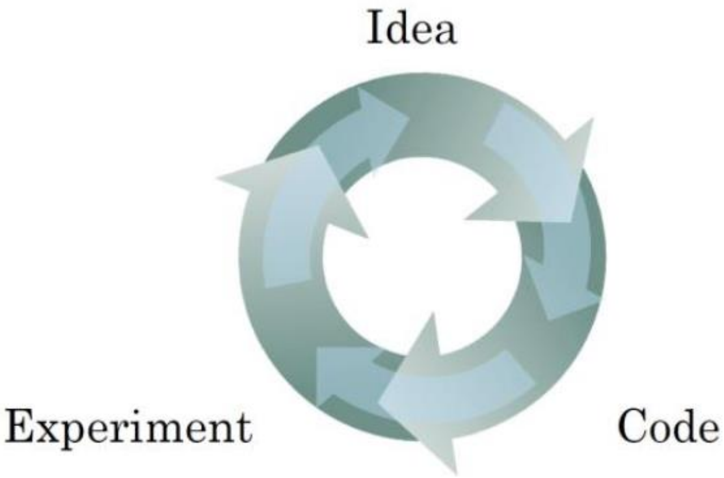


Two things have to be considered to get to the high level of performance:

1. Being able to train a big enough neural network
2. Huge amount of labeled data

Process of Training a Neural Network

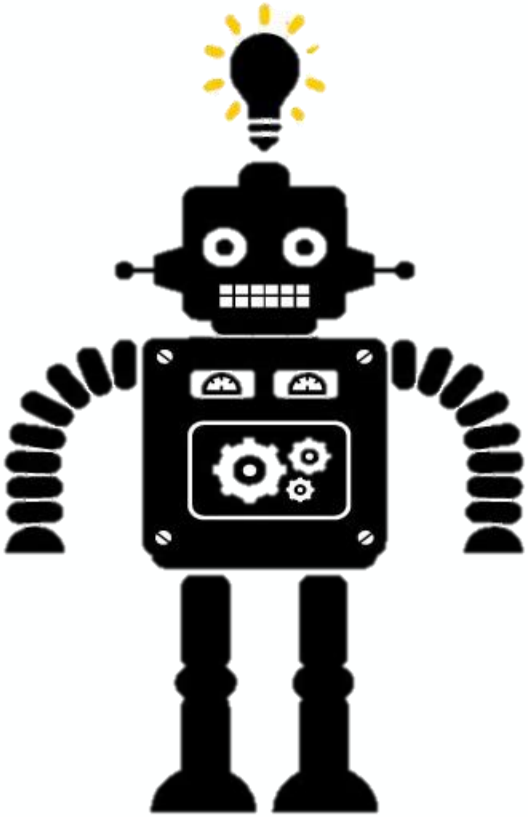
The process of training a neural network is iterative.



It could take a good amount of time to train a neural network, which affects your productivity. Faster computation helps to iterate and improve new algorithm.

What is Machine Learning?

Machine learning allows computers to learn and infer from data.



Classical Programming and Machine Learning

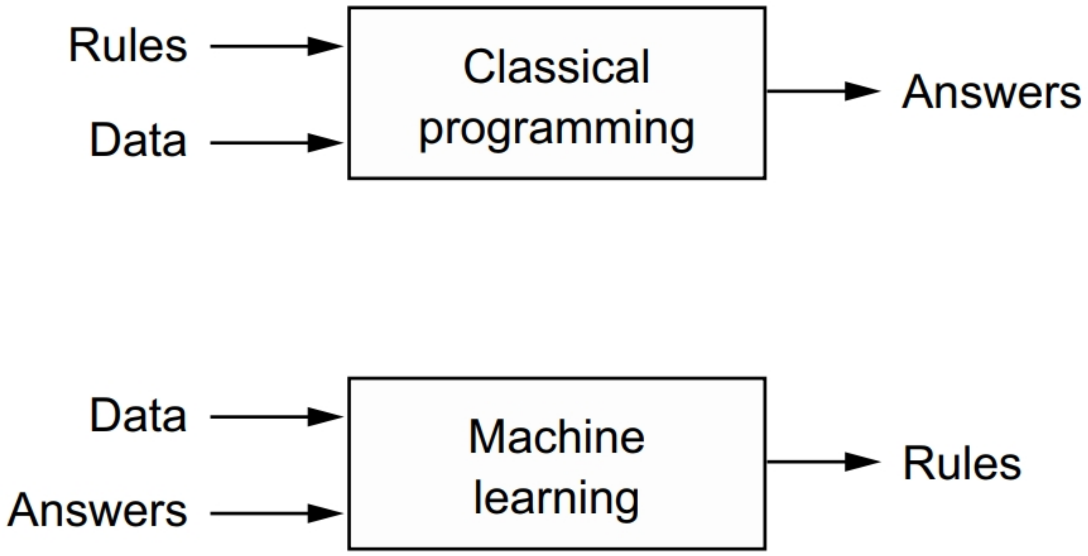


Image Source:
Deep Learning with Python, Second Edition
By Francois Chollet

Artificial Intelligence, Machine Learning, and Deep Learning

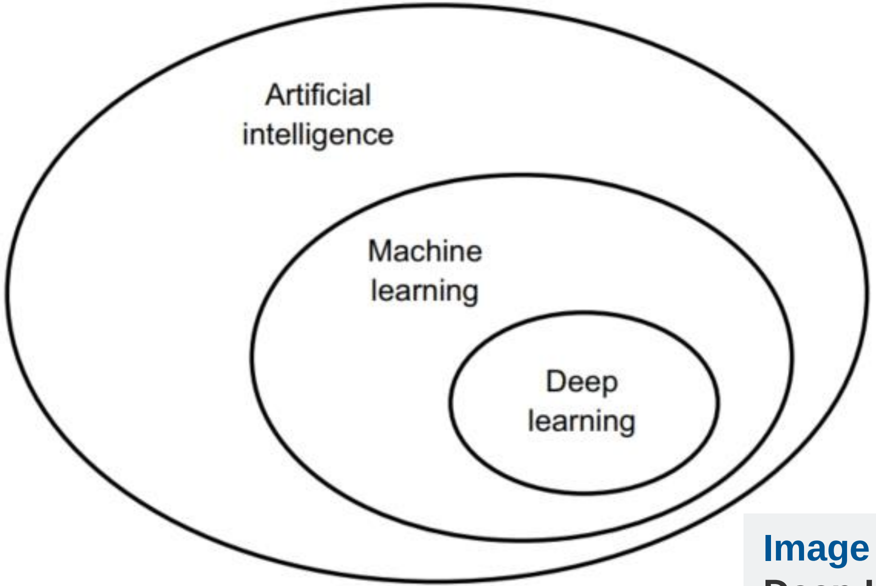


Image Source:
Deep Learning with Python, Second Edition
By Francois Chollet

Types of Machine Learning

Supervised	data points have known outcome
Unsupervised	data points have unknown outcome

Types of Supervised Learning

Regression outcome is continuous (numerical)

Classification outcome is a category

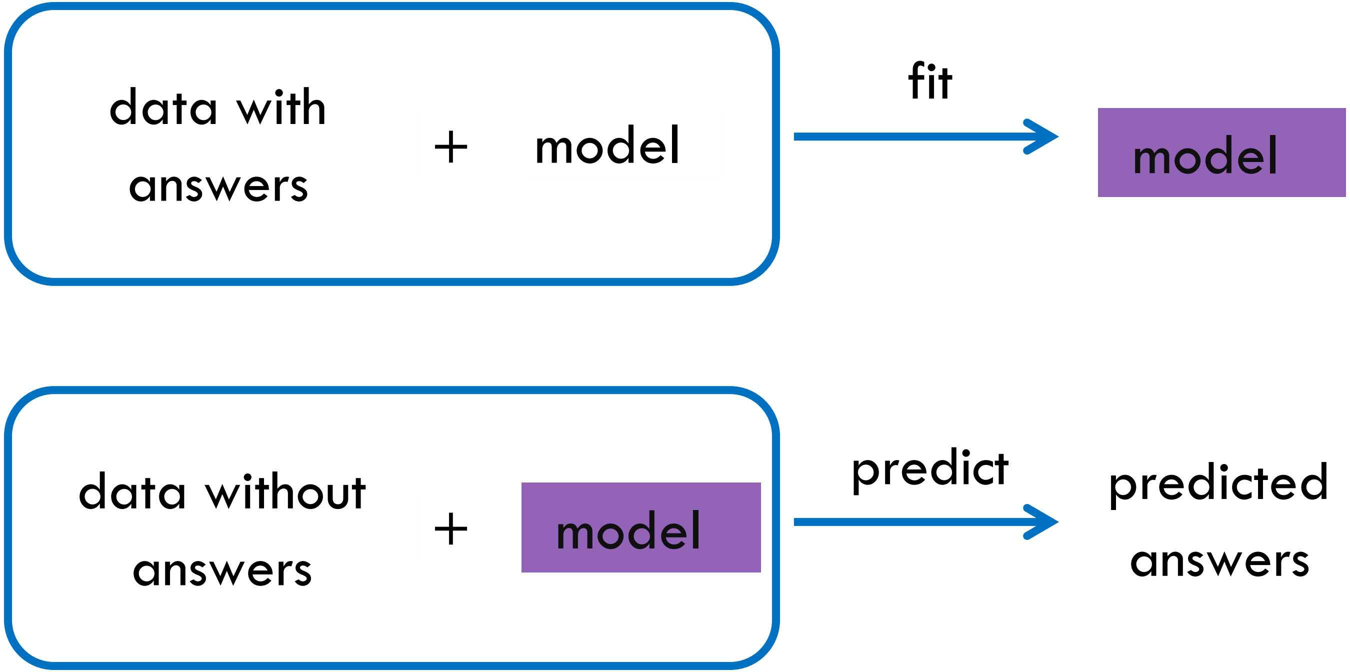
Machine Learning Vocabulary

- **Target:** predicted category or value of the data (column to predict)
- **Features:** properties of the data used for prediction (non-target columns)
- **Example:** a single data point within the data (one row)
- **Label:** the target value for a single data point

Machine Learning Vocabulary (Synonyms)

- **Target:** Response, Output, Dependent Variable, Labels
- **Features:** Predictors, Input, Independent Variables, Attributes
- **Example:** Observation, Record, Instance, Datapoint, Row
- **Label:** Answer, y-value, Category

Supervised Learning Overview

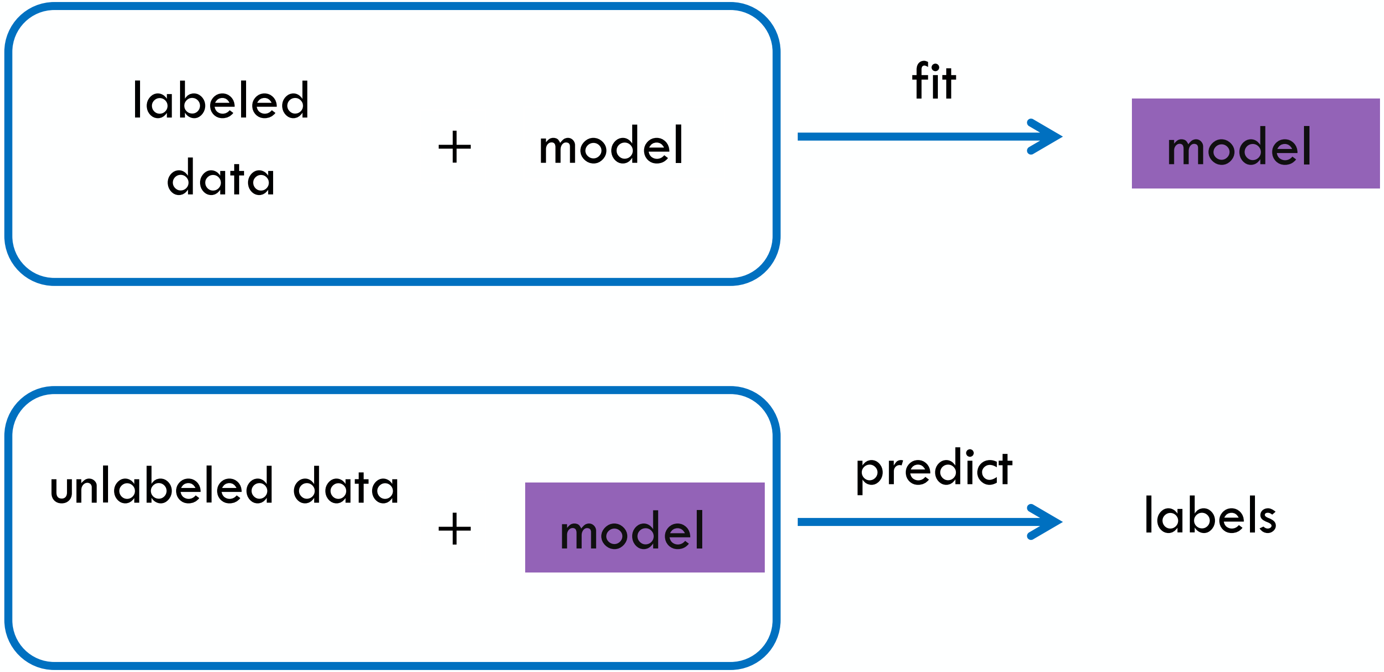


Regression: Numeric Answers

movie data with revenue + model $\xrightarrow{\text{fit}}$ model

movie data (unknown revenue) + model $\xrightarrow{\text{predict}}$ predicted revenue

Classification: Categorical Answers



Classification: Categorical Answers

emails labeled as spam/not spam + model $\xrightarrow{\text{fit}}$ model

unlabeled emails + model $\xrightarrow{\text{predict}}$ spam or not spam

Three Types of Classification Predictions

- **Hard Prediction:** Predict a single category for each instance.
- **Ranking Prediction:** Rank the instances from most likely to least likely. (binary classification)
- **Probability Prediction:** Assign a probability distribution across the classes to each instance.

Metrics for Classification

- **Hard Prediction:** Accuracy, Precision, Recall (Sensitivity), Specificity, F1 Score
- **Ranking Prediction:** AUC (ROC), Precision-Recall Curves
- **Probability Prediction:** Log-loss (aka Cross-Entropy), Brier Score

Metrics for Regression

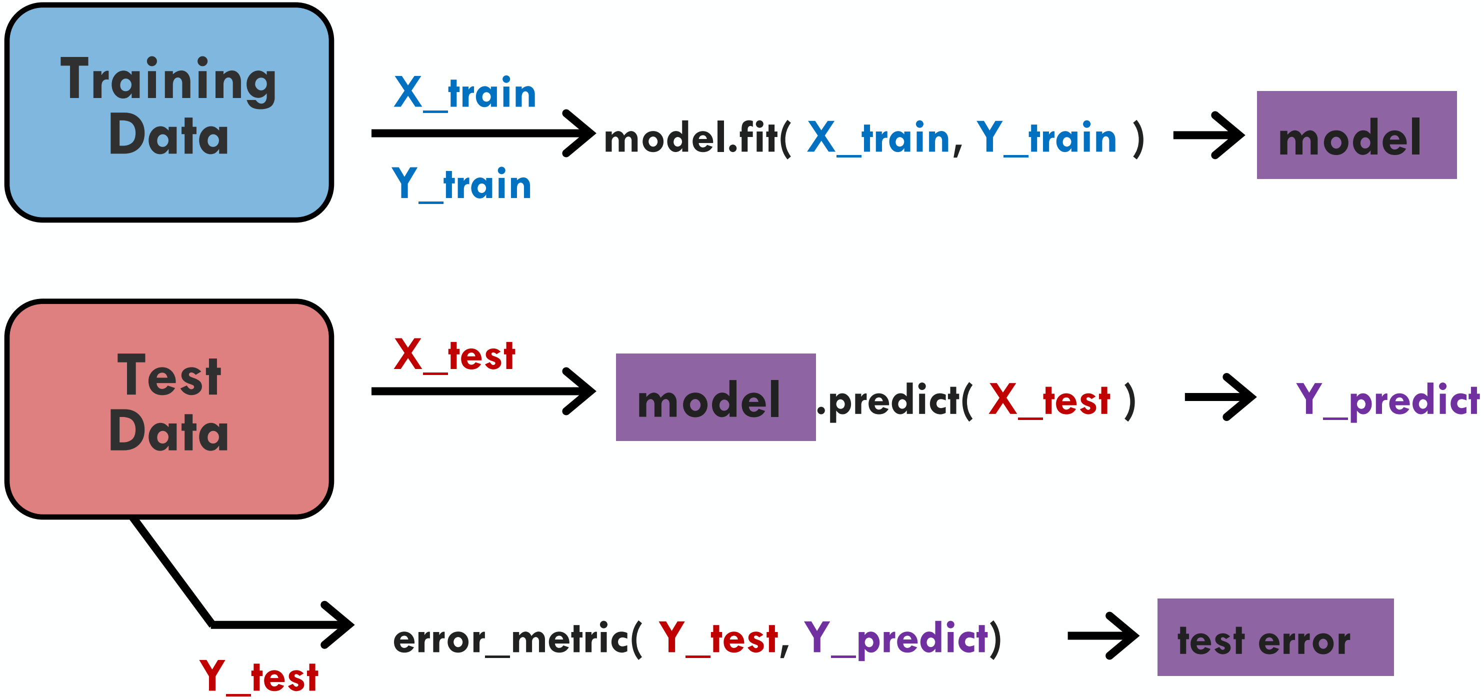
- **Root Mean Square Error (RMSE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Mean Absolute Error (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Fitting Training and Test Data



Using Training and Test Data

Training Data

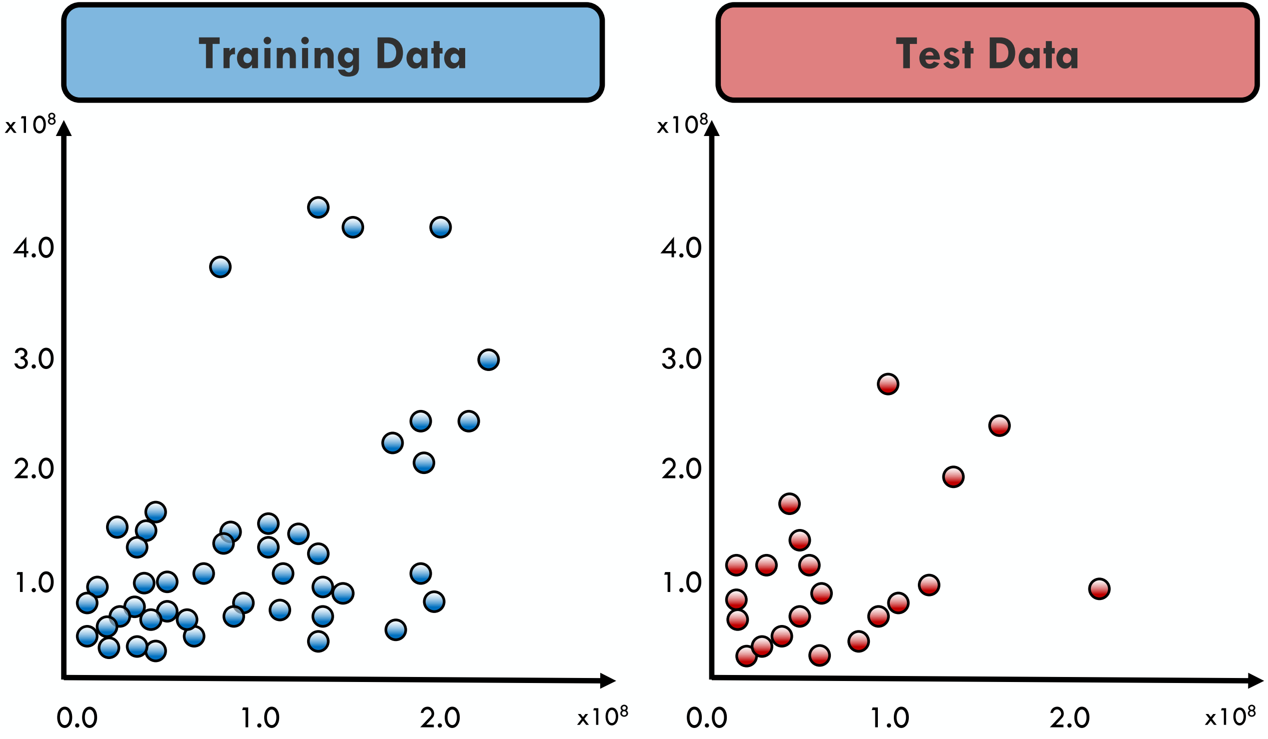
fit the model

Test Data

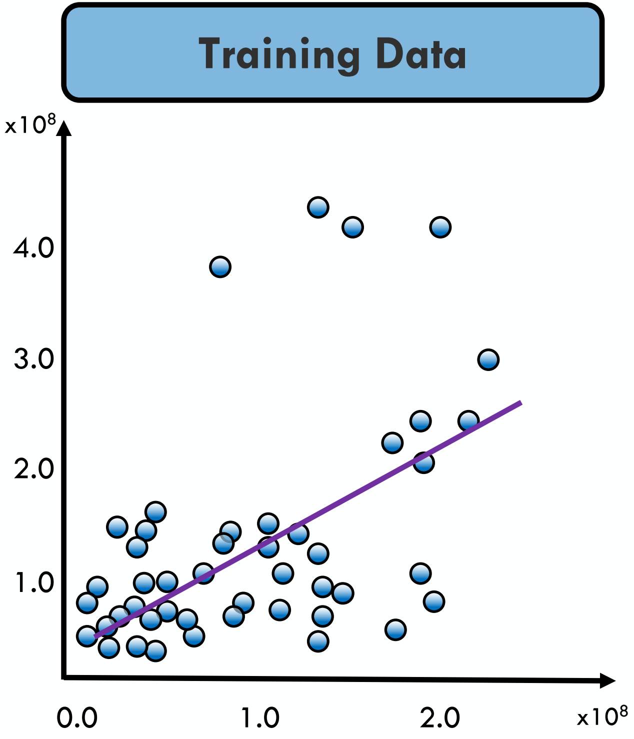
measure performance

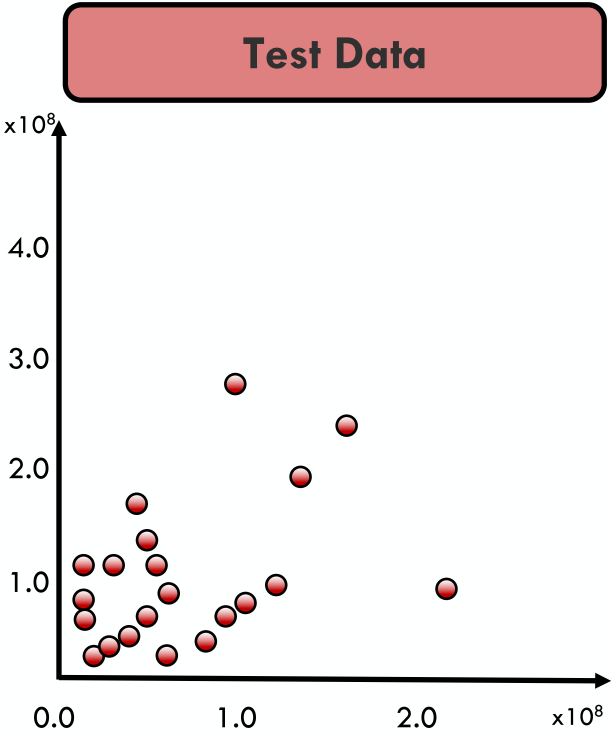
- predict label with model
- compare with actual value
- measure error

Using Training and Test Data



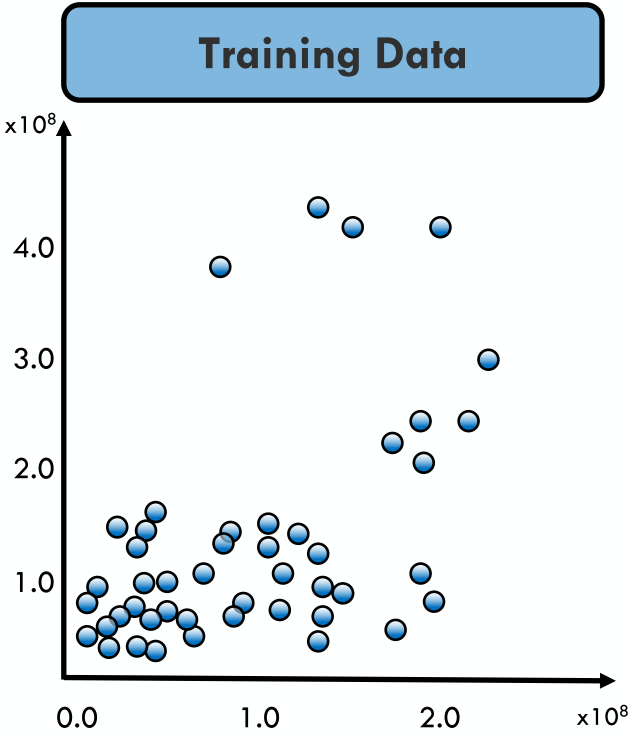
Using Training and Test Data

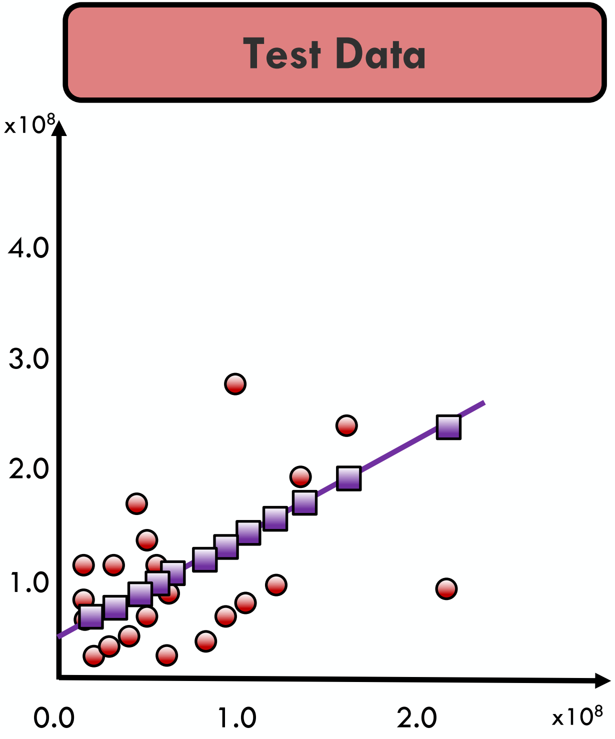




Fit the model

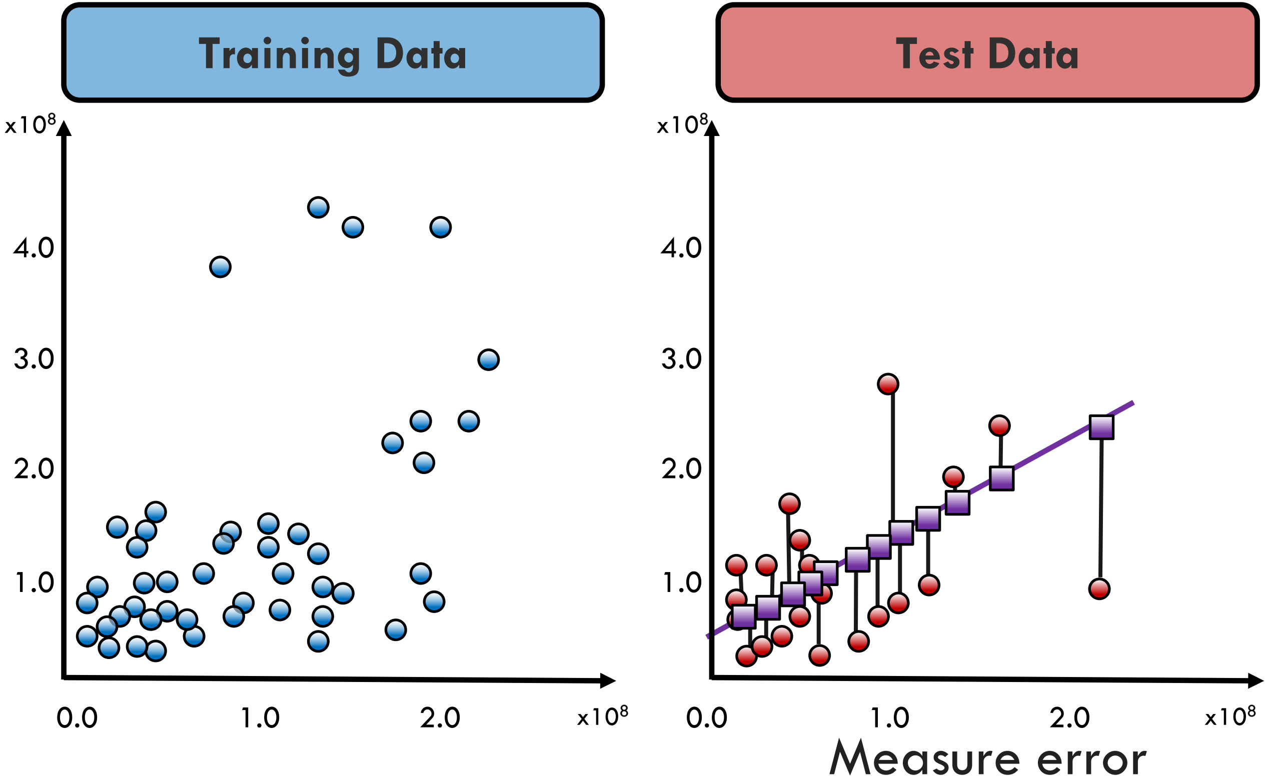
Using Training and Test Data



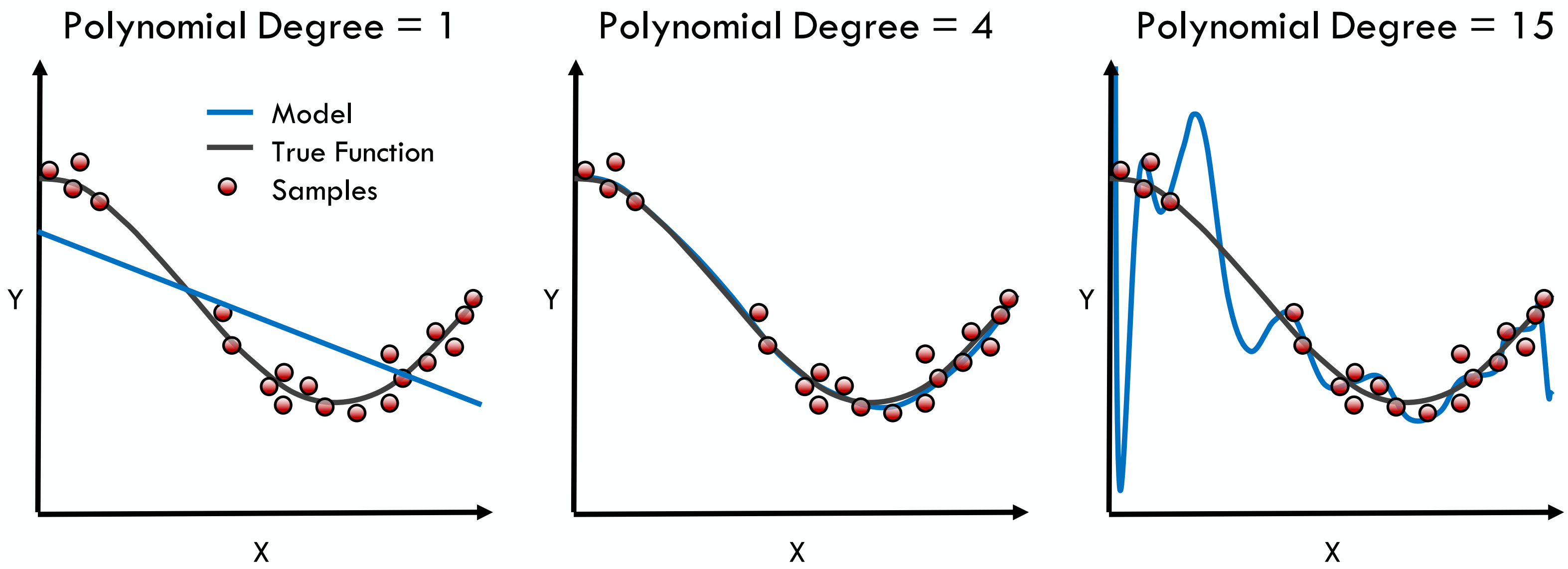


Make predictions

Using Training and Test Data



How Well Does the Model Generalize?

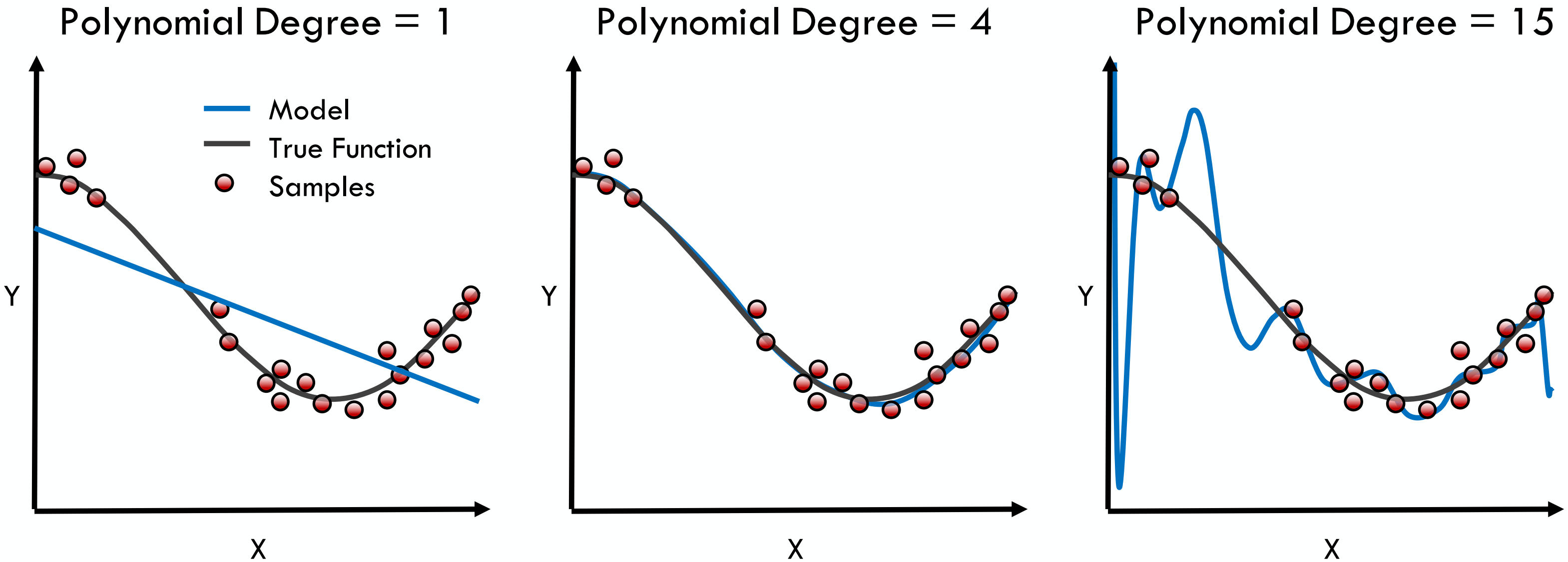


Poor on Training Set
Poor at Predicting

Just Right

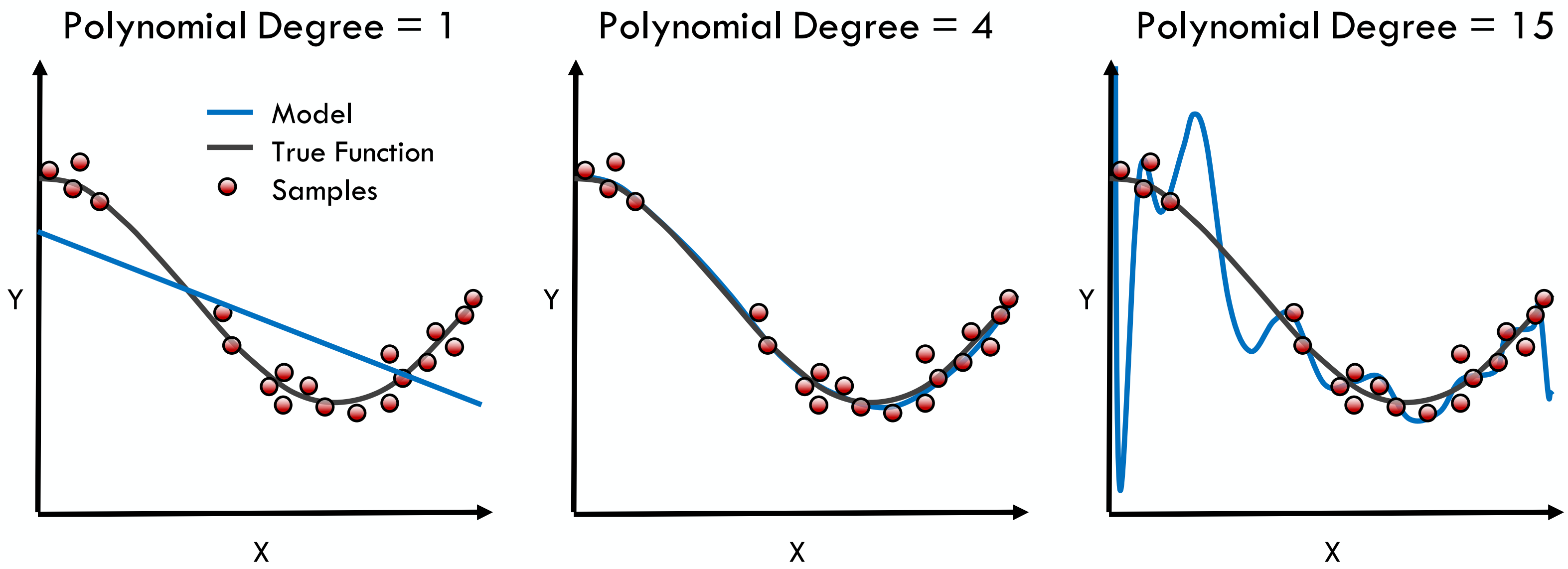
Very Good on Training Set
Poor at Predicting

Underfitting vs Overfitting



Underfitting **Just Right** **Overfitting**

Bias – Variance Tradeoff



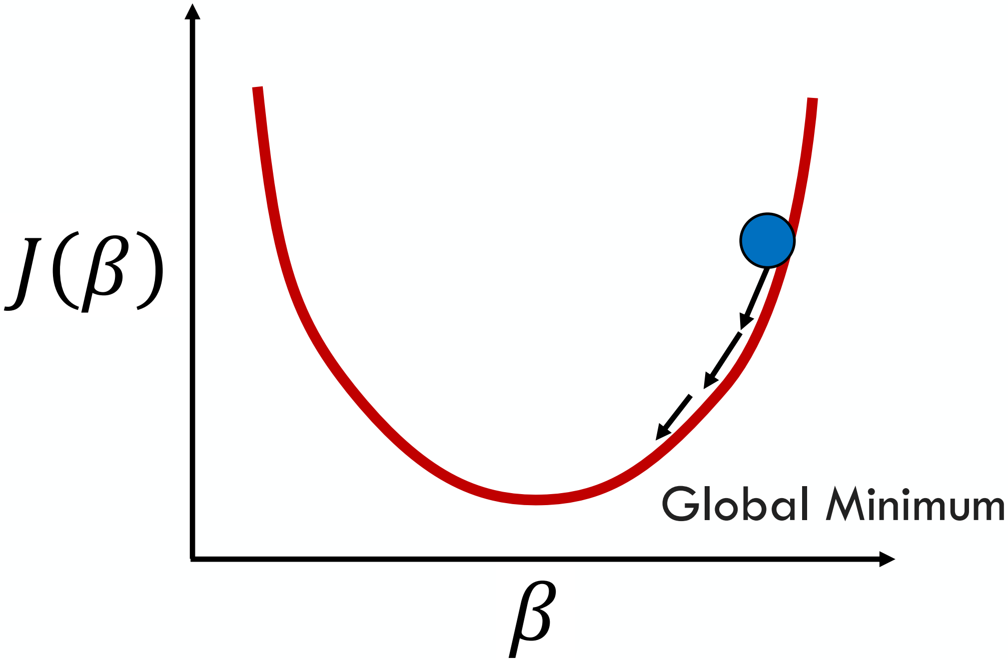
High Bias
Low Variance

Just Right

Low Bias
High Variance

Gradient Descent

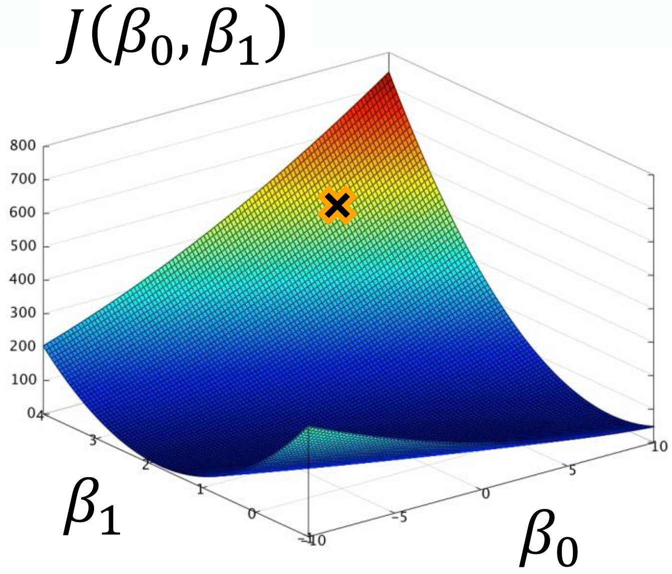
Start with a cost function $J(\beta)$:



Then gradually move towards the minimum.

Gradient Descent with Linear Regression

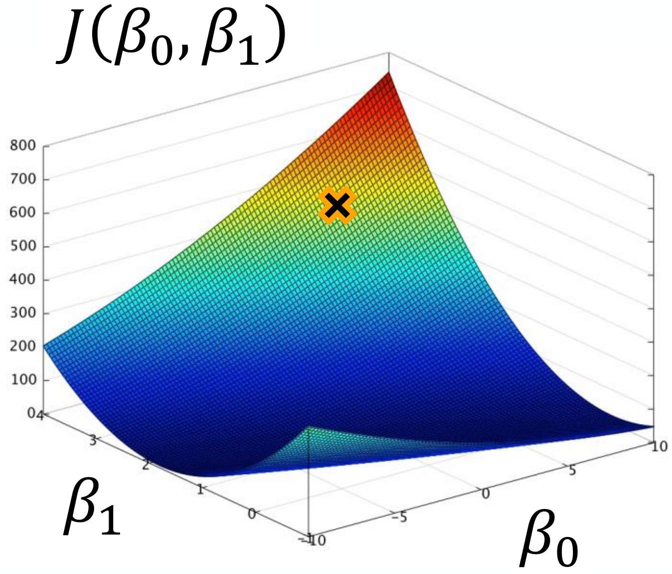
- Now imagine there are two parameters (β_0, β_1)
- This is a more complicated surface on which the minimum must be found
- How can we do this without knowing what $J(\beta_0, \beta_1)$ looks like?



Gradient Descent with Linear Regression

- The gradient is a vector whose coordinates consist of the partial derivatives of the parameters

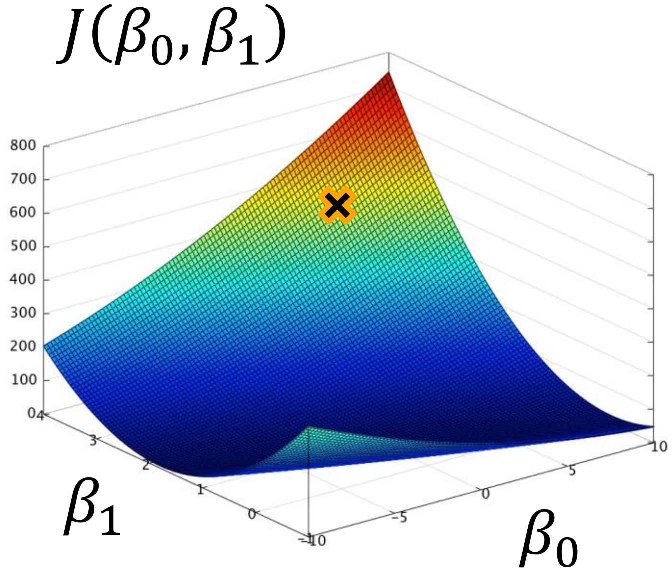
$$\nabla J(\beta_0, \ldots, \beta_n) = < \frac{\partial J}{\partial \beta_0}, \ \ldots, \frac{\partial J}{\partial \beta_n} >$$



Gradient Descent with Linear Regression

- Compute the gradient, $\nabla J(\beta_0, \beta_1)$, which points in the direction of the biggest increase!

- -$\nabla J(\beta_0, \beta_1)$ (negative gradient) points to the biggest decrease at that point!

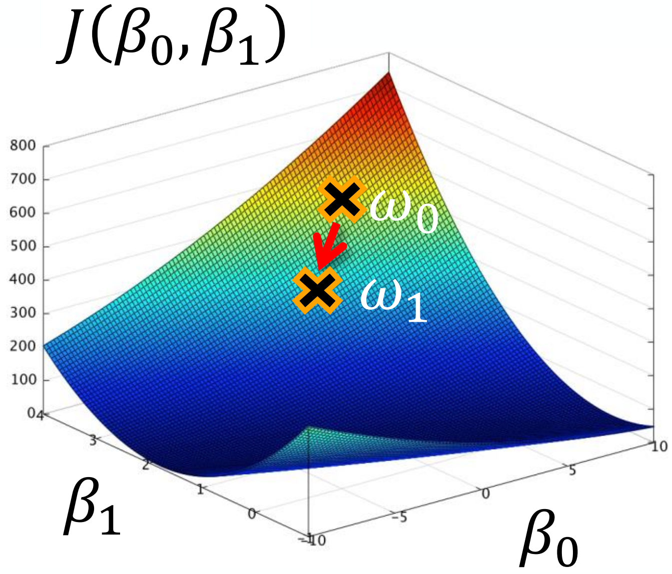


Gradient Descent with Linear Regression

- Then use the gradient (∇) and the cost function to calculate the next point (ω_1) from the current one (ω_0):

$$\omega_1 = \omega_0 - \alpha\nabla\frac{1}{2}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$

- The learning rate (α) is a tunable parameter that determines step size

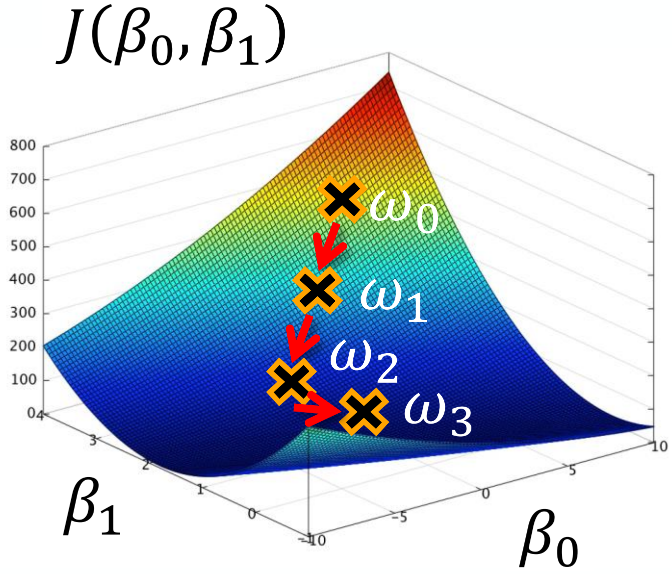


Gradient Descent with Linear Regression

- Each point can be iteratively calculated from the previous one

$$\omega_2 = \omega_1 - \alpha\nabla \frac{1}{2}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$

$$\omega_3 = \omega_2 - \alpha\nabla \frac{1}{2}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$



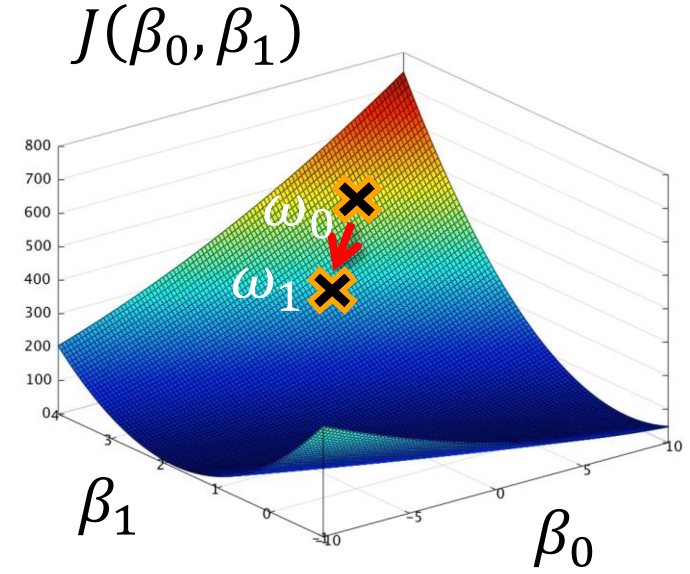
Stochastic Gradient Descent

- Use a single data point to determine the gradient and cost function instead of all the data

$$\omega_1 = \omega_0 - \alpha\nabla\frac{1}{2}\cancel{\sum_{i=1}^{m}}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$

$$\Downarrow$$

$$\boxed{\omega_1 = \omega_0 - \alpha\nabla\frac{1}{2}\left(\left(\beta_0 + \beta_1 x_{obs}^{(0)}\right) - y_{obs}^{(0)}\right)^2}$$



Stochastic Gradient Descent

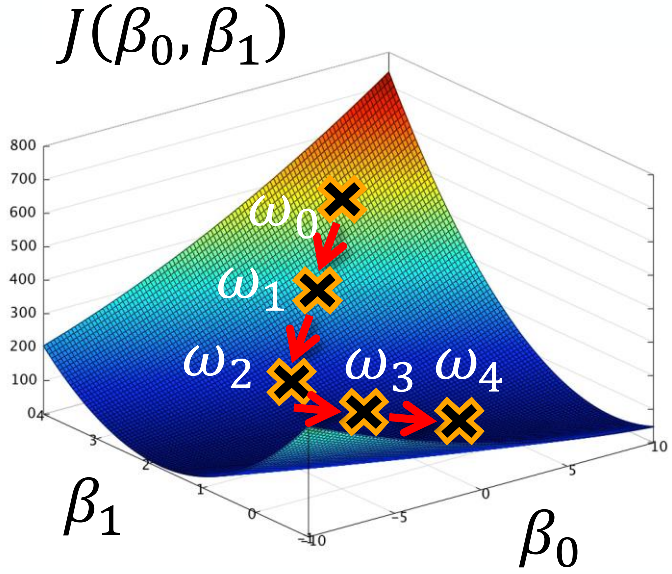
- Use a single data point to determine the gradient and cost function instead of all the data

$$\omega_1 = \omega_0 - \alpha\nabla\frac{1}{2}\left(\left(\beta_0 + \beta_1 x_{obs}^{(0)}\right) - y_{obs}^{(0)}\right)^2$$

$$\cdots$$

$$\omega_4 = \omega_3 - \alpha\nabla\frac{1}{2}\left(\left(\beta_0 + \beta_1 x_{obs}^{(3)}\right) - y_{obs}^{(3)}\right)^2$$

- Path is less direct due to noise in single data point—"stochastic"



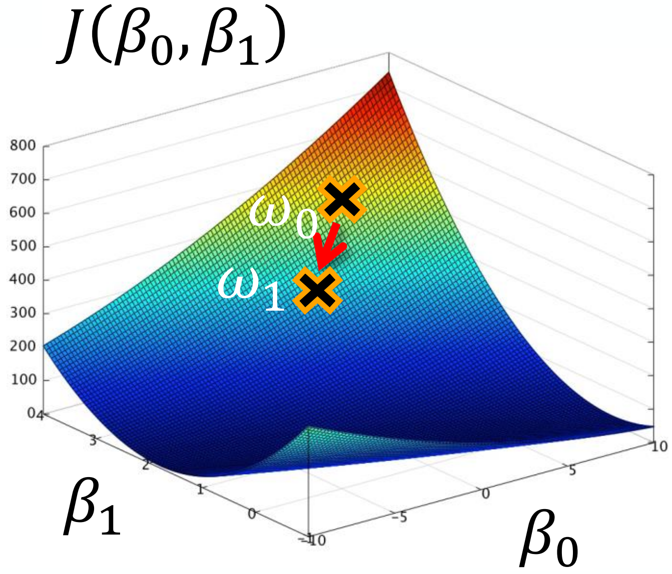
Mini Batch Gradient Descent

- Perform an update for every n training examples

$$\omega_1 = \omega_0 - \alpha\nabla\frac{1}{2}\sum_{i=1}^{n}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$

Best of both worlds:

- Reduced memory relative to "vanilla" gradient descent
- Less noisy than stochastic gradient descent



Software