

TENSOR.BY

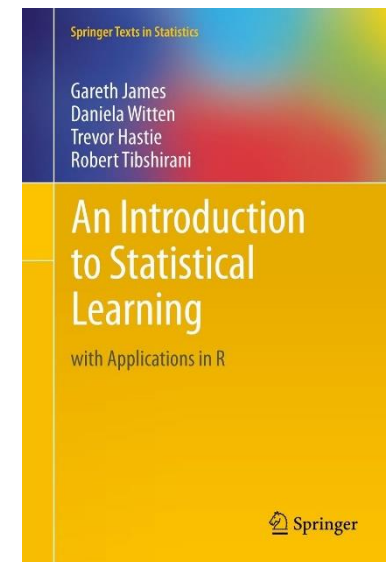
ML-course

## 2. Regression in Python Scikit-learn

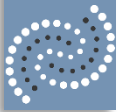
Kate Miniukovich (Data Scientist),  
miniukovich@rocketscience.ai

# Reference

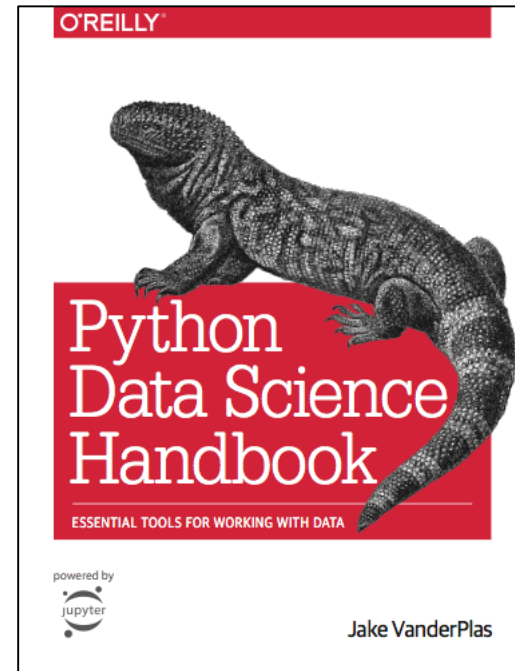
*An Introduction to Statistical Learning* by  
Gareth James, Daniela Witten, Trevor Hastie,  
and Robert Tibshirani, [http://www-  
bcf.usc.edu/~gareth/ISL/](http://www-bcf.usc.edu/~gareth/ISL/)  
(available online for free)



# Reference



*Jake VanderPlas*

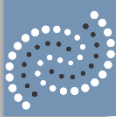


*Python Data Science Handbook*

<https://jakevdp.github.io/PythonDataScienceHandbook/>

*Video*

<https://www.youtube.com/watch?v=L7R4HUQ-eQ0&t=6033s>



# Reference

The screenshot shows the scikit-learn documentation website for version 0.19.1. The header includes the scikit-learn logo, navigation links for Home, Installation, Documentation, and Examples, a search bar, and a 'Fork me on GitHub' button. The main heading is 'Documentation of scikit-learn 0.19.1'. Below this, there are six sections arranged in a 2x3 grid: Quick Start, User Guide, Other Versions, Tutorials, API, and Additional Resources. Each section has a brief description of its content.

**Quick Start**  
A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

**User Guide**  
The main documentation. This contains an in-depth description of all algorithms and how to apply them.

**Other Versions**

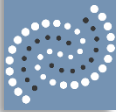
- [Development version](#)
- [All available versions](#)
- [PDF documentation](#)

**Tutorials**  
Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

**API**  
The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

**Additional Resources**  
Talks given, slide-sets and other information relevant to scikit-learn.

<http://scikit-learn.org>

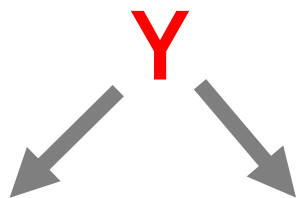


# Supervised vs. Unsupervised Learning

## Supervised

Data:

- 1)  $n$  observations;
- 2)  $p$  variables  $X_1, X_2, \dots, X_p$ , measured on each observation;
- 3) **response  $Y$**  measured on same  $n$  observations



Continuous  
**Regression**

Discrete  
**Classification**

## Unsupervised

Data:

- 1)  $n$  observations;
- 2)  $p$  variables  $X_1, X_2, \dots, X_p$ , measured on each observation

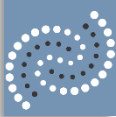
Clustering...

# Regression / Classification Problem



## Steps to solve

- *Working with data*
- *Modeling*



# Representation of data in Scikit-learn

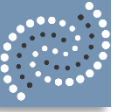
- **X**  
*two-dimensional numpy array*  
*shape - (n\_samples, m\_features)*
- **Y**  
*one-dimensional numpy array*  
*shape - (n\_samples, )*



# Modeling

- Choose a class of model
- Fit the model to data
- Validate the model and optimize hyperparameters
- Predict for unknown data





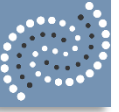
# Mathematical model

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

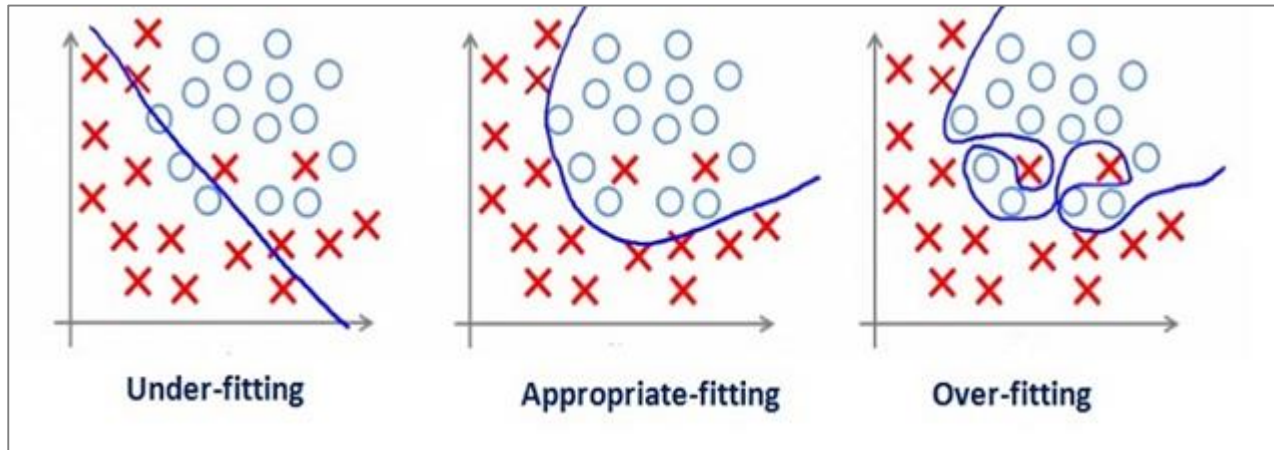
$f$  is some fixed but unknown function of  $X_1, \dots, X_p$ , and  $e$  is a random *error term*, which is independent of  $X$  and has mean zero. In this formulation,  $f$  represents the *systematic* information that  $X$  provides about  $Y$ .

We can predict  $Y$  using our estimate for  $f$

$$\hat{Y} = \hat{f}(X)$$

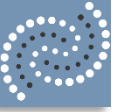


# Bias-Variance Trade-Off



***Underfitting*** (*high bias*) - algorithm is missing the relevant relations between features and target outputs

***Overfitting*** (*high variance*) - modeling the random noise in the training data, rather than the intended outputs.



# Model validation

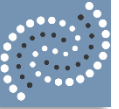
- **Data**

- train + test (e.g. 75% + 25%)
- train + valid + test (e.g. 60% + 20% + 20%)
- train with cross-validation + test (e.g. 80% + 20% )

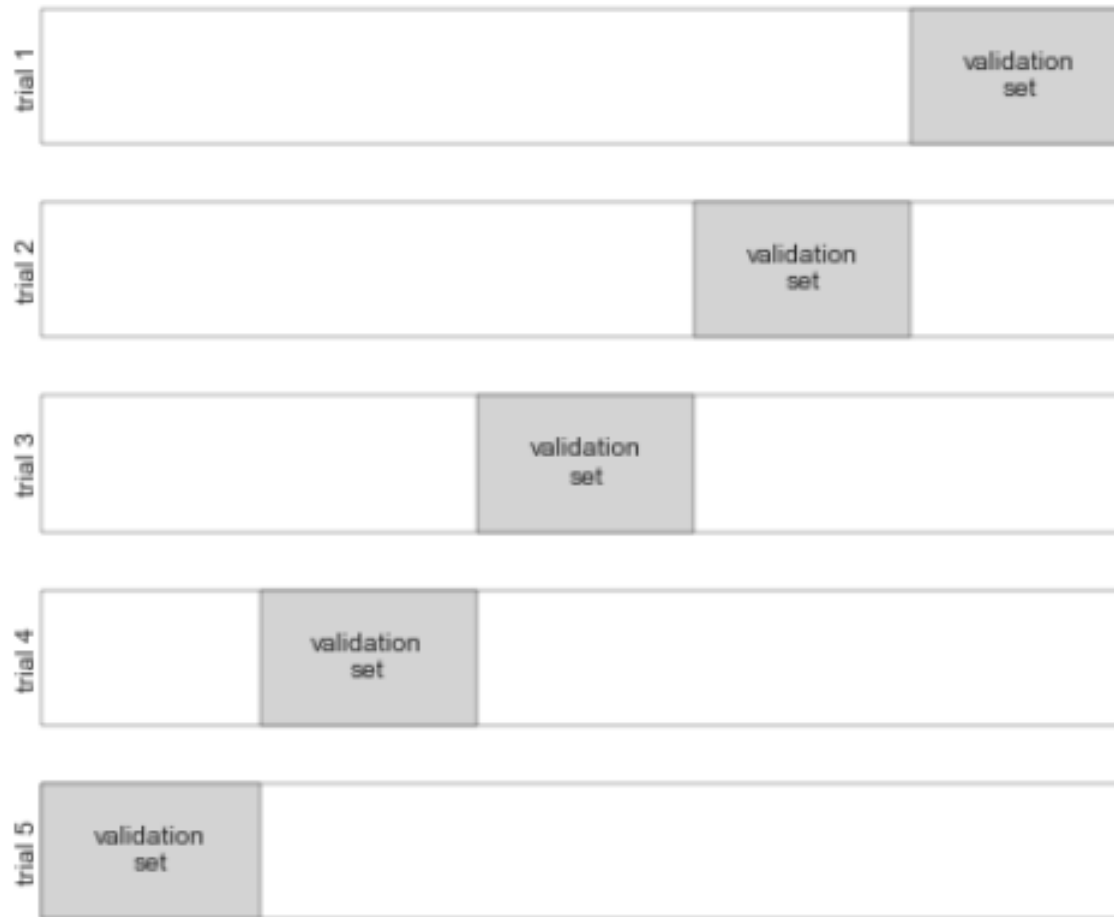
- **Metrics**

- Regression:  $R^2$ , MSE, MAE,...

[http://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metricsmodel.score\(\)](http://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metricsmodel.score())



# Model validation via cross-validation





# Some models for Regression in Python scikit-learn

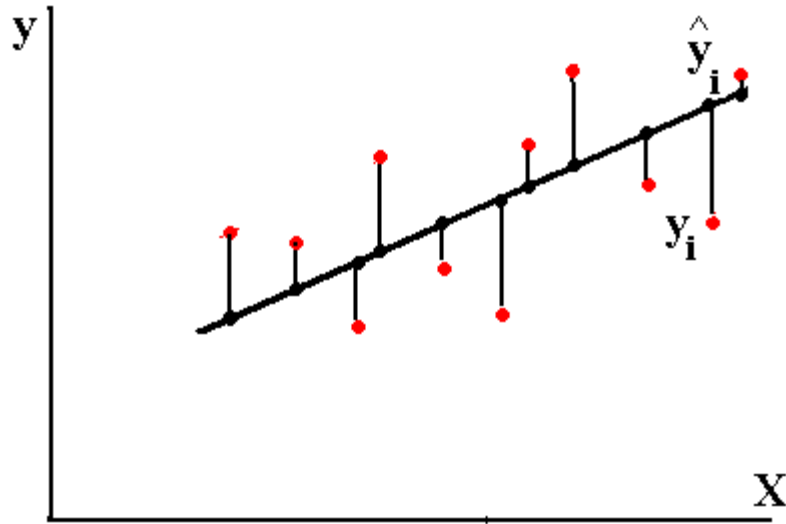
- **Generalized Linear Models**
  - Linear Regression
  - Ridge Regression
  - Lasso Regression

*example of Linear, Ridge and Lasso Regressions in regression.ipynb*

- **Ensemble methods**
  - Random Forests
  - Gradient Tree Boosting



# Linear Regression with one variable



$(x_i, y_i)$ ,  $i=1, n$  - number of observations (red points)

$$\hat{y} = ax + b$$

$$\hat{y} = \theta_0 + \theta_1 x_1 = \theta_0 x_0 + \theta_1 x_1, \quad x_0 = 1$$

$\theta_0$  - intercept,  $\theta_1$  - slope

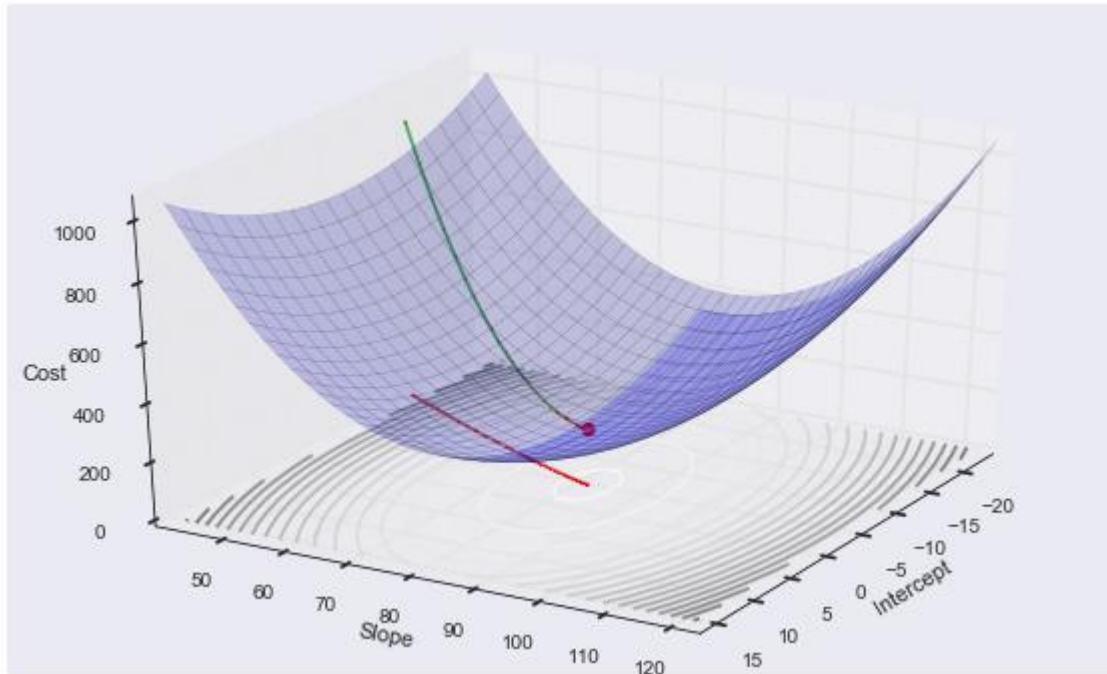
**The method of least squares**

$$Cost = J(\theta_0, \theta_1) = \sum_{i=1}^n (\hat{y}^i - y^i)^2 = \sum_{i=1}^n (\theta_0 x_0^i + \theta_1 x_1^i - y^i)^2$$

**Our aim** -  $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$



# Gradient descent to find $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$



*Need to choose*

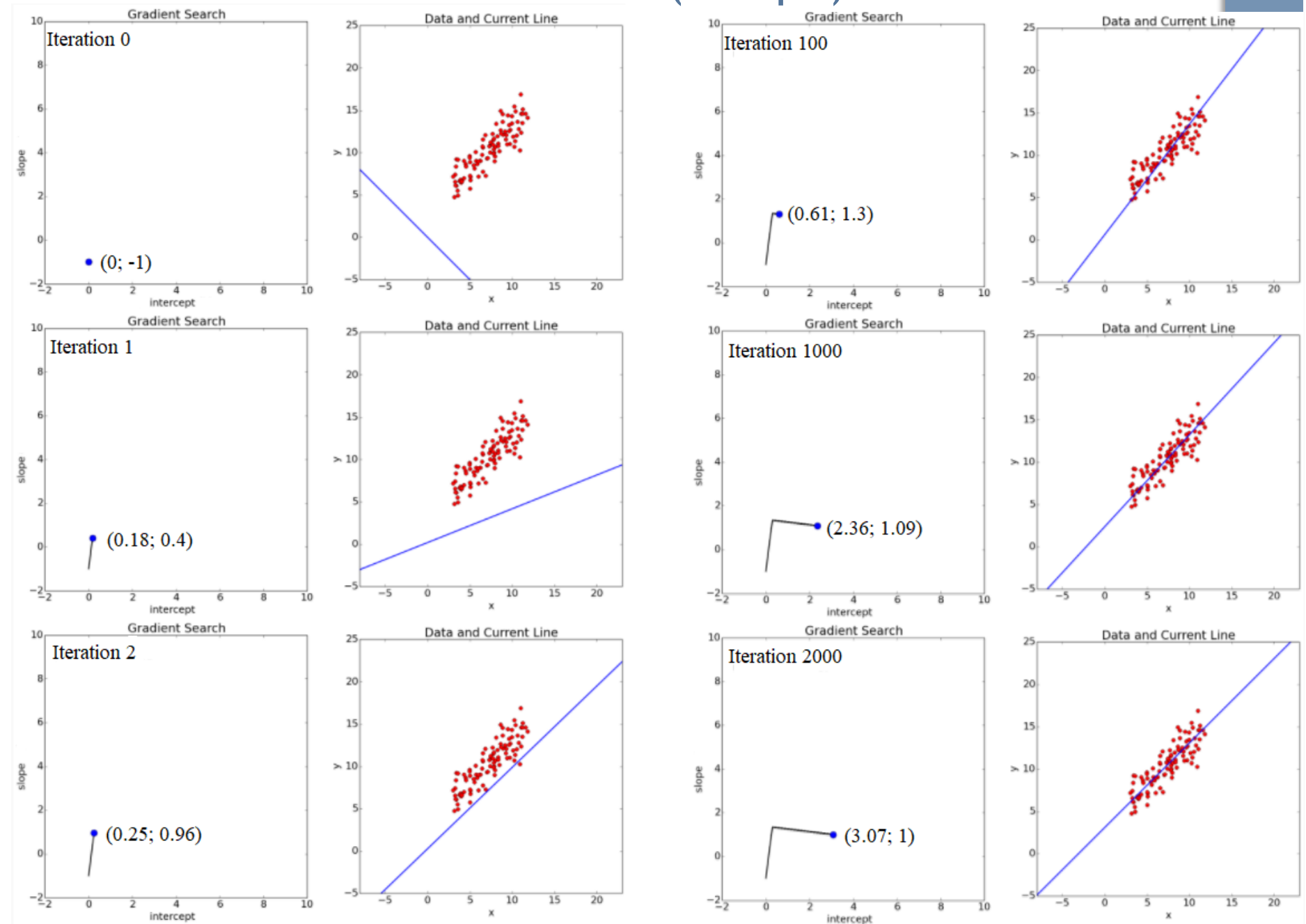
$\alpha$  – learning rate (step size)  
 $(\theta_0, \theta_1)$  – start point

**Repeat until convergence**

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - 2\alpha \sum_{i=1}^n (\theta_0 x_0^i + \theta_1 x_1^i - y^i) x_0^i$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - 2\alpha \sum_{i=1}^n (\theta_0 x_0^i + \theta_1 x_1^i - y^i) x_1^i$$

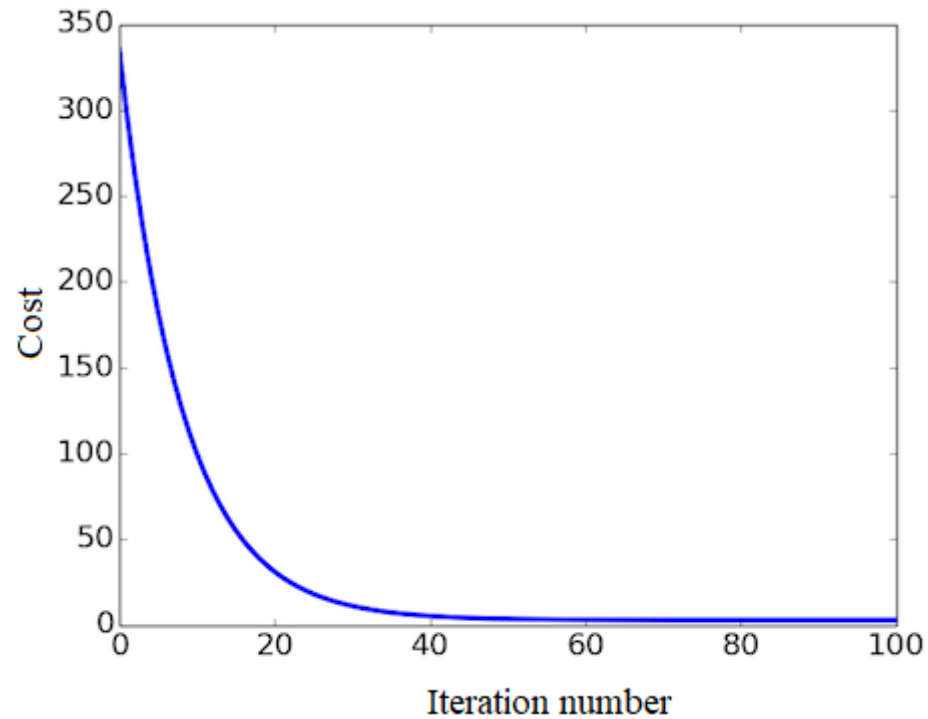
# Gradient descent (example)







# Gradient descent (example)





# Linear Regression with multiple variables

*m variables, n observations*

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_m x_m, \quad x_0 = 1$$

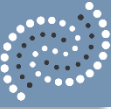
$$X = [1, x_1, \dots, x_m] \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_m \end{bmatrix} \quad \hat{y} = \mathbf{h}_\theta(X) = X\theta$$

*Dataset for training:*  $X^{(i)} = [1, x_1^{(i)}, \dots, x_m^{(i)}], y^{(i)}, i = 1, \dots, n$

$$\text{Cost} = J(\theta) = \sum_{i=1}^n (\mathbf{h}_\theta(X^{(i)}) - y^{(i)})^2 \quad \text{Our aim} - \min_{\theta} J(\theta)$$

*Repeat until convergence:*  
for  $j=0\dots m$

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) = \theta_j - 2\alpha \sum_{i=1}^n (\mathbf{h}_\theta(X^{(i)}) - y^{(i)}) x_j^{(i)}$$



# Cost functions

## for Linear Regression, Ridge and Lasso in scikit-learn

$m$  variables,  $n$  observations

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_m x_m, \quad x_0 = 1$$

$$X = [1, x_1, \dots, x_m] \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \quad \hat{y} = \mathbf{h}_\theta(X) = X\theta$$

**Linear Regression**

$$\text{Cost} = J(\theta) = \sum_{i=1}^n (X^{(i)}\theta - y^{(i)})^2$$

**Ridge (regularization **I2**)**  $\text{Cost} = J(\theta) = \sum_{i=1}^n (X^{(i)}\theta - y^{(i)})^2 + \alpha \sum_{j=0}^m \theta_j^2$

**Lasso (regularization **I1**)**  $\text{Cost} = J(\theta) = \sum_{i=1}^n (X^{(i)}\theta - y^{(i)})^2 + \alpha \sum_{j=0}^m |\theta_j|$



# Regression metrics

**$R^2$  score, the coefficient of determination**

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^n (y^{(i)} - \bar{y})^2}, \quad \text{where } \bar{y} = \frac{\sum_{i=1}^n y^{(i)}}{n}$$

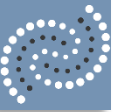
**Mean squared error**

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

**Mean absolute error**

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}|$$

[http://scikit-learn.org/stable/modules/model\\_evaluation.html#regression-metrics](http://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics)



# Bias-Variance Trade-Off

High-bias model: Underfits the data



High-variance model: Overfits the data



- For **high-bias** models, the **performance** of the model on the **validation** set is **similar** to the performance on the **training set** (*but the performance is worse than for appropriate fitting*).
- For **high-variance** models, the **performance** of the model on the **validation** set is **far worse** than the performance on the **training** set.



# Bias-Variance Trade-Off

**What to do in case of high-bias or high variance?**

**Change**

- Model complexity (e.g. via regularization)
- Quantity of training samples
- Set of features

**Reading**

*Jake VanderPlas Python Data Science Handbook (05.03-Hyperparameters-and-Model-Validation)*

*Andrew Ng ML: Advice for Applying Machine Learning*



# Ways to fix high bias/variance in linear models

High bias (underfitting)	High variance (overfitting)
<ul style="list-style-type: none"><li>• Add more features</li><li>• Add polynomial features</li></ul>	<ul style="list-style-type: none"><li>• More training examples</li><li>• Smaller set of features</li><li>• Use regularization</li><li>• Increase regularization strength (coefficient)</li></ul>



# Choose the best model

Models	R <sup>2</sup>	
	train	test
LinearRegression()		
LinearRegression(normalize = True)		
LinearRegression with PolynomialFeatures		
n=2		
n=3		
Ridge with Polynomial Features, n=2		
a=0.1		
a=1		
a=10		
a=100		
Lasso with Polynomial Features, n=2		
a=0.1		
a=1		
a=10		
a=100		



# Q & A

# Thank you!