Introduction to Machine Learning
Day 1

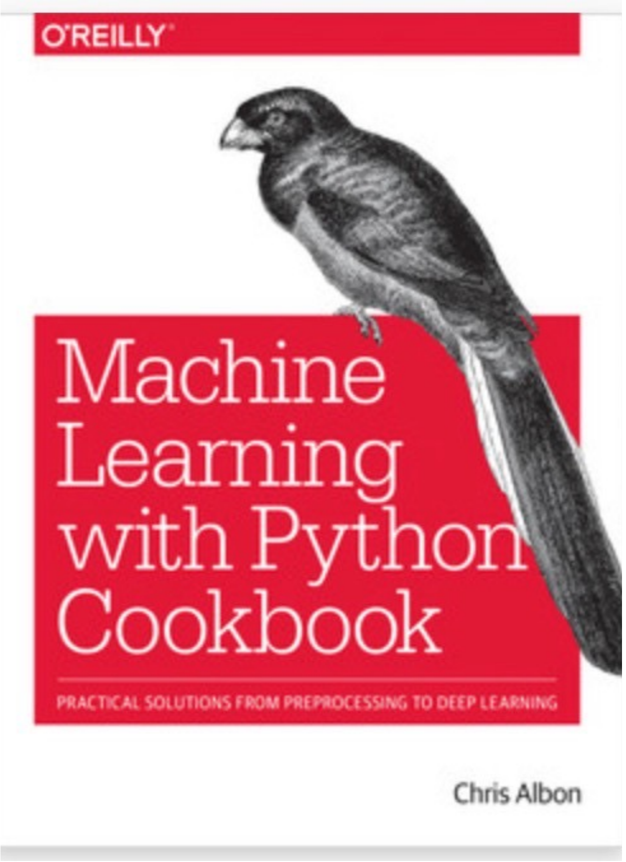
Machine Learning Course
Sammie Omranian
6/10/2024

About instructor

- Sammie Omranian
- PhD student in Biomedical and Health Informatics
- Research area: Applications of Natural Language Processing in Healthcare
- Research Trainee at Brigham and Women's Hospital, Harvard Medical School

Book

- Book description from O'REILY website:
- This practical guide provides nearly 200 self-contained recipes to help you solve machine learning challenges you may encounter in your daily work.
- Vectors, matrices, and arrays
- Handling numerical and categorical data, text, images, and dates and times
- Dimensionality reduction using feature extraction or feature selection
- Model evaluation and selection
- Linear and logical regression, trees and forests, and k-nearest neighbors
- Support vector machines (SVM), naïve Bayes, clustering, and neural networks
- Saving and loading trained models



Terminology Used in This Book

- Observation
  - A single unit in our level of observation—for example, a person, a sale, or a record.
- Learning algorithms
  - An algorithm used to learn the best parameters of a model—for example, linear regression, naive Bayes, or decision trees.
- Models
  - An output of a learning algorithm's training. Learning algorithms train models, which we then use to make predictions.
- Parameters
  - The weights or coefficients of a model learned through training.

- Hyperparameters
  - The settings of a learning algorithm that need to be set before training.

Terminology Used in This Book

- Performance
  - A metric used to evaluate a model.
- Loss
  - A metric to maximize or minimize through training.
- Train
  - Applying a learning algorithm to data using numerical approaches like gradient descent.
- Fit
  - Applying a learning algorithm to data using analytical approaches.
- Data
  - A collection of observations.

Introduction to Machine Learning

Arthur Samuel (1959), an American pioneer in the field of artificial intelligence, defined machine learning as:

The field of study that gives computers the ability to learn without being explicitly programmed.

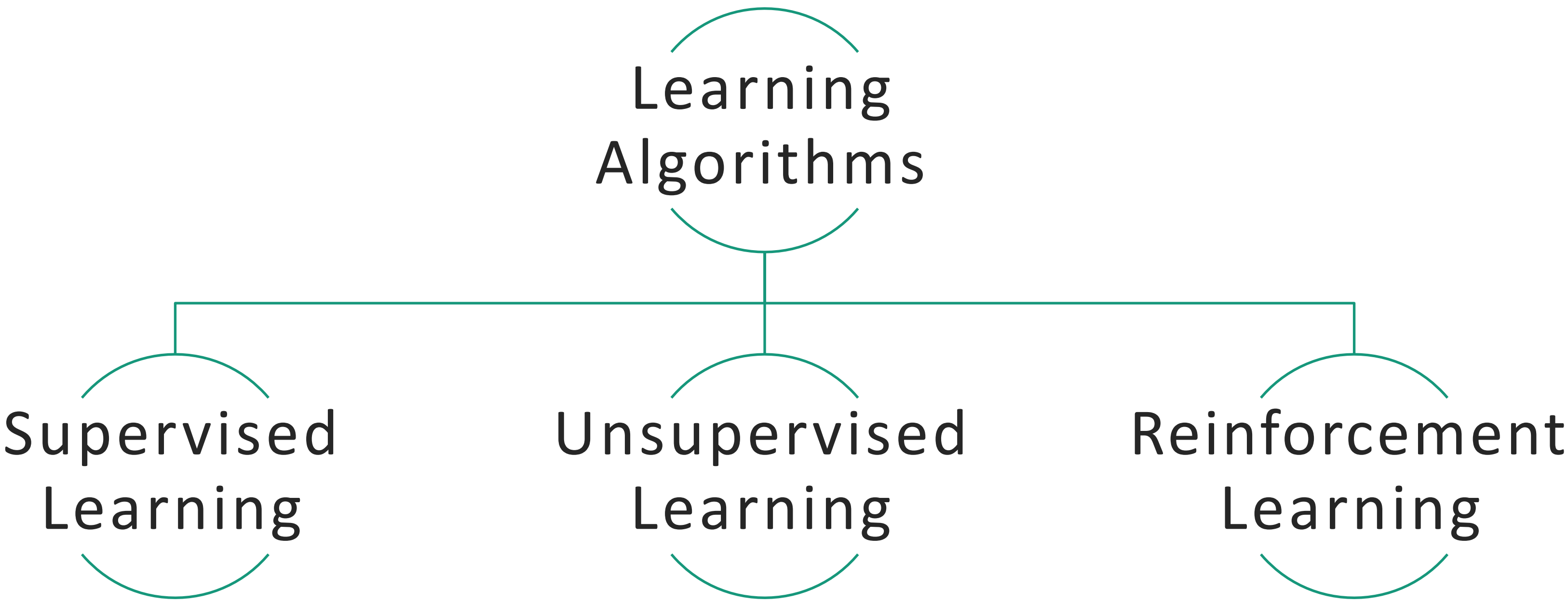
He developed the checkers-playing program, in the late 1950s, one of his most notable achievements in the field of artificial intelligence and machine learning.

Introduction to Machine Learning

Tom Mitchell, a renowned computer scientist, defined machine learning as:

"A computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on T, as measured by P, improves with experience E."

Machine Learning Algorithms



Introduction to Machine Learning

Types of ML:

- **Supervised Learning**: Learning from labeled data (e.g., classifying emails as spam or not spam).

- **Unsupervised Learning**: Finding patterns in unlabeled data (e.g., clustering customers based on purchasing behavior).

- **Reinforcement Learning**: Learning by interacting with an environment to maximize cumulative reward (e.g., training a robot to walk).

Introduction to Machine Learning

- Applications:

  - Healthcare (predicting diseases)
  - Finance (fraud detection)
  - Marketing (customer segmentation)
  - Autonomous vehicles (self-driving cars)

Introduction to Machine Learning

- Workflow:

    1. Data Collection
    2. Data Preprocessing
    3. Model Training
    4. Model Evaluation
    5. Model Deployment

Supervised Learning

Regression

Classification

Supervised Learning

- **Regression** vs **Classification**:

  - Regression: Predicting continuous values (e.g., house prices).

  - Classification: Predicting categorical values (e.g., whether a tumor is benign or malignant).

Supervised Learning

- Key Concepts:

  - **Features**: Independent variables used as input.
  - **Labels**: Dependent variables (target) we want to predict.
  - **Training Set**: Subset of data used to train the model.
  - **Test Set**: Subset of data used to evaluate the model's performance.

Data Preprocessing

- Loading data with pandas

```python
import pandas as pd
df = pd.read_csv('data.csv')
```

- Handling missing values

```python
df.fillna(df.mean(), inplace=True)

```

- Feature scaling and normalization
- Splitting data into training and test sets

Data Preprocessing

- Feature scaling and normalization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
```

- Splitting data into training and test sets

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Regression

Source: [techtarget]

Linear Regression

- A linear approach to modeling the relationship between a dependent variable and independent variable.
  - Mathematical Foundation:

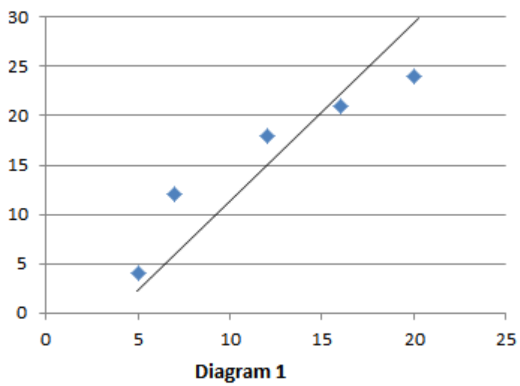
$$y' = b + w_1 x_1$$

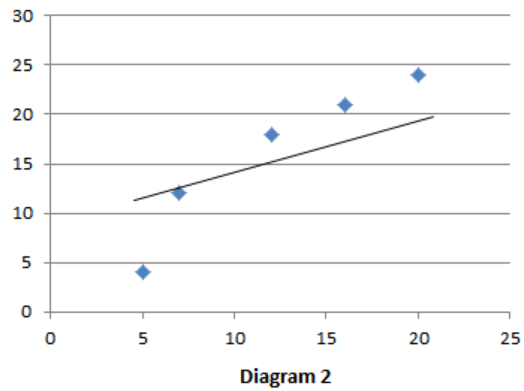
- where:
- y' is the predicted value (a desired output).
- b is the bias (the y-intercept), sometimes referred to as w_0.
- w_1 is the weight of feature 1. Weight is the same concept as the "slope" in the traditional equation of a line.
- x_1 is a feature (a known input).

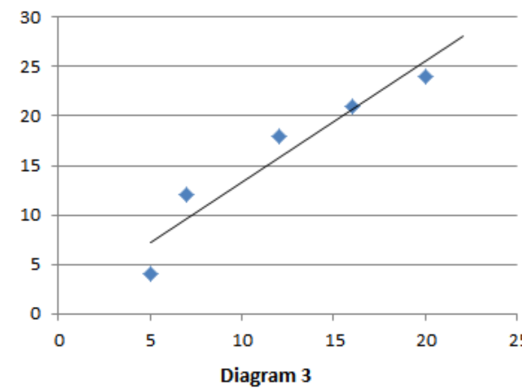
Although this model uses only one feature, a more sophisticated model might rely on multiple features, each having a separate weight:

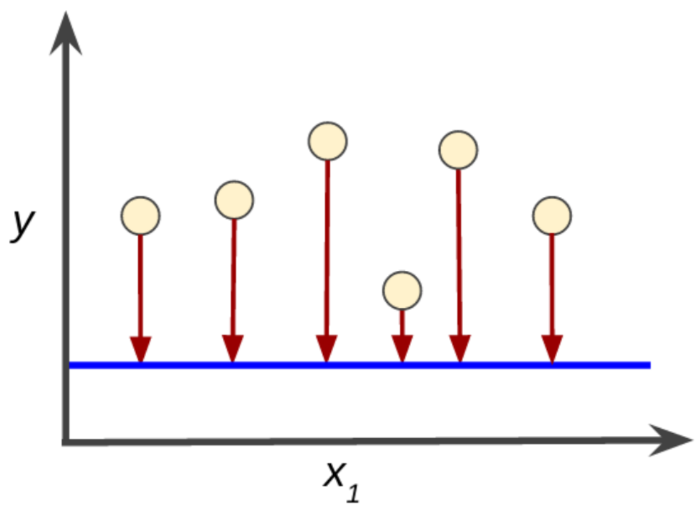
$$y' = b + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_\mathrm{n} x_\mathrm{n}$$

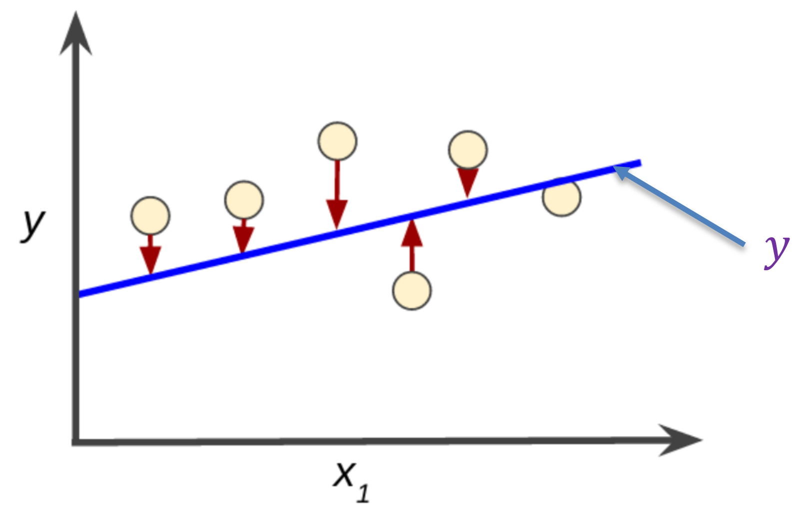
Linear Regression



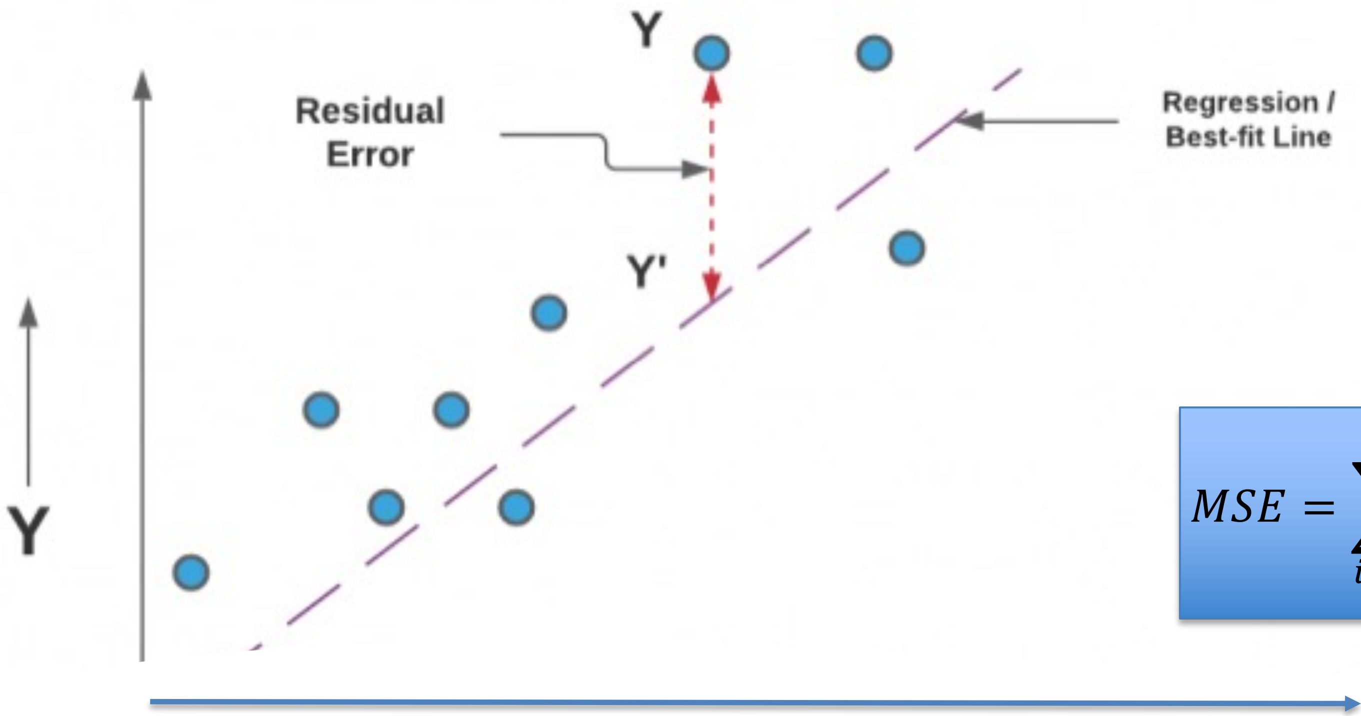








High error (difference between actual value and predicted value) in the left model; low error in the right model.



$$MSE = \sum_{i=0}^{n} \frac{(y_i - y_i')^2}{n}$$

- **Cost Function**: Mean Squared Error (MSE)

The cost function defined as Mean Squared Error is the sum of the squared differences between the prediction and true value. And the output is a single number representing the cost. So, the line with the minimum cost function or MSE represents the relationship between X and Y in the best possible manner.

Linear Regression

Gradient Descent: An iterative optimization algorithm to find values of the model parameters (bias and weights) that minimize the cost function.

$$MSE = \sum_{i=0}^{n} \frac{(y_i - y_i')^2}{n}$$

Cost function

Now , we can use gradient descent to find optimum parameters which is an iterative algorithm and apply the following rule " update rule" .

$$w_i = w_{i-1} - learning\ rate * \partial(MSE(b, w_i))/\partial w$$

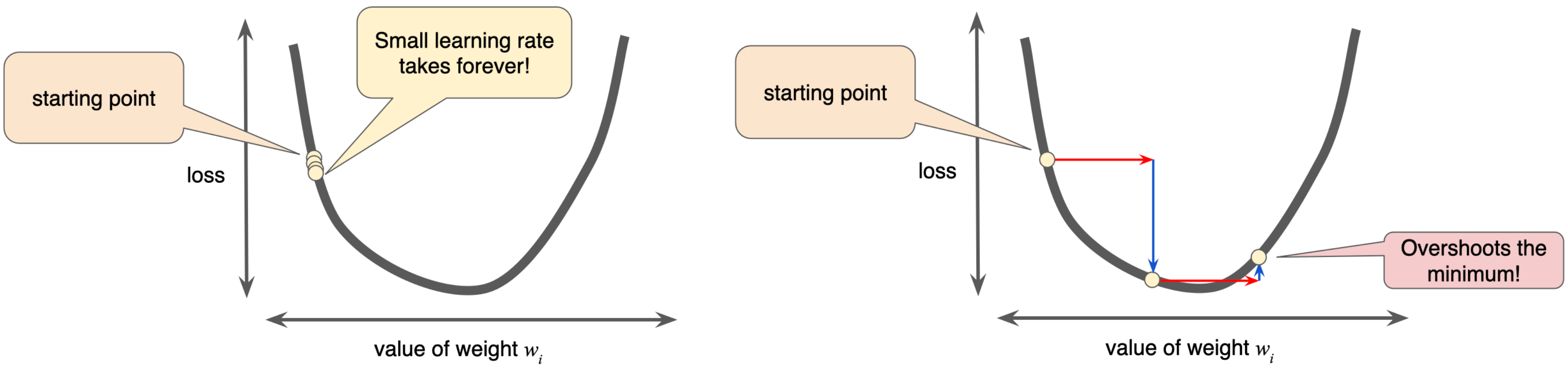
$$b_i = b_{i-1} - learning\ rate * \partial(MSE(b_i, w))/\partial b$$

gradient

Linear Regression

Learning rate decides the size of steps in gradient descent algorithm. The programmer sets this rate. It should not be too large nor too small. if it takes large steps ,it may miss the optimum point . If it takes too small , it may take too many iterations and consume large computation time.

There's a Goldilocks learning rate for every regression problem. The Goldilocks value is related to how flat the loss function is. The common values are [*0.00001,0.0001, 0,001, 0.01, 0.1, 1]*. You can check if your learning rate is doing well by plotting it on a graph.

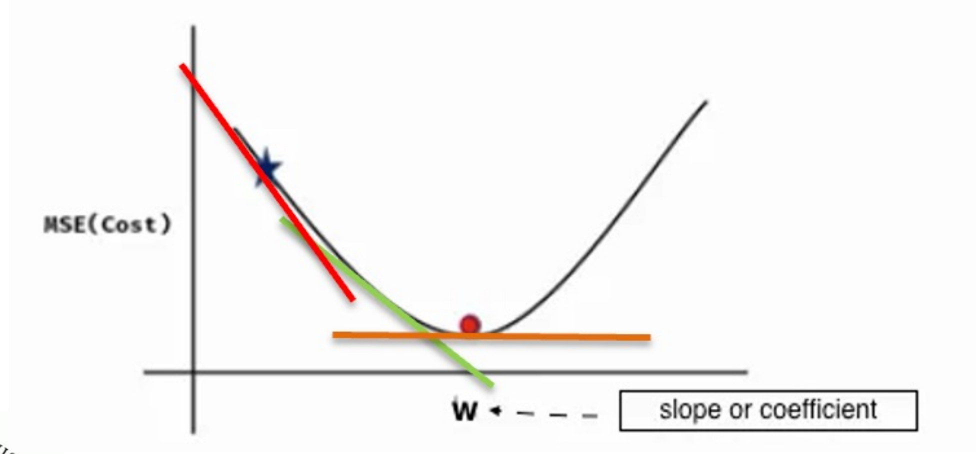


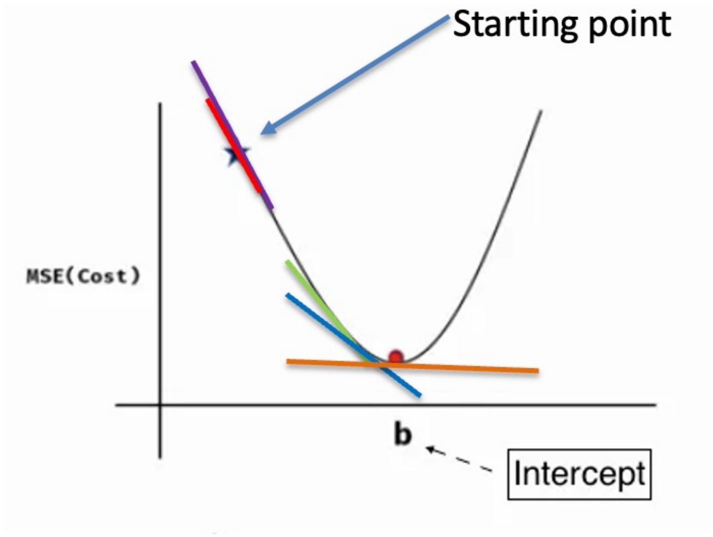
Linear Regression

$$w_i = w_{i-1} - learning\,rate\; * \;\partial(MSE(b, w_i))/\partial w$$

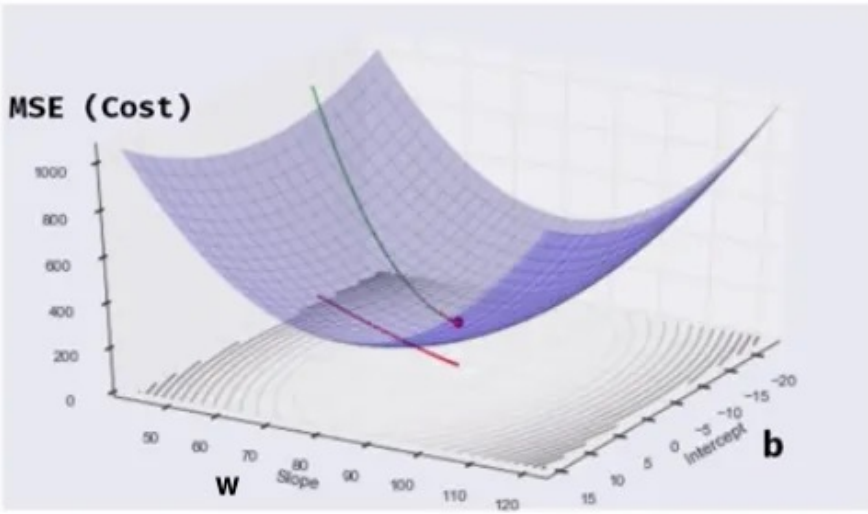
$$b_i = b_{i-1} \; - learning\,rate\; * \;\partial(MSE(b_i, w))/\partial b$$

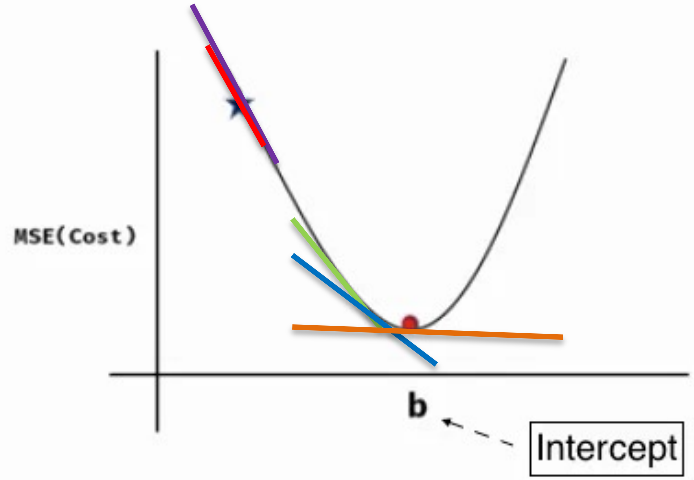
$$precision = abs(w_i - w_{i-1})$$

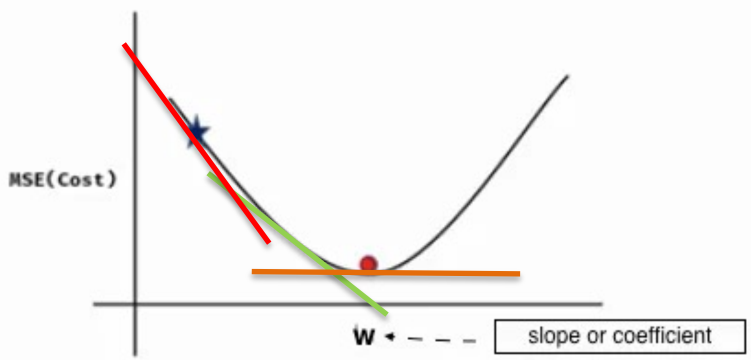




Linear Regression







Gradient vectors: $\frac{\partial(MSE)}{\partial w}$, $\frac{\partial(MSE)}{\partial b}$

3D picture of gradient descent , MSE (cost) with weight (w) , and MSE with bias (b).

[Graph source](#)

Linear Regression

- From Scratch:

- Step 1:
- Initialize Parameters: $w_0, b_0, learning\ rate, precision, \#iterations$

- Step 2: Compute Gradient and update parameters

- Step 3: repeat until the difference between the values of parameters from two consecutive iterations is less than precision or when the number of iterations exceeds number of iterations, the algorithm should be stopped.

Linear Regression

```python
cur_x = 3 # The algorithm starts at x=3
rate = 0.01 # Learning rate
precision = 0.000001 #This tells us when to stop the algori
previous_step_size = 1 #
```

```python
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x #Store current x value in prev_x
    cur_x = cur_x - rate * df(prev_x) #Grad descent
    previous_step_size = abs(cur_x - prev_x) #Change in x
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nX value is",cur_x) #Print iterations

print("The local minimum occurs at", cur_x)
```

Regularized Regression

- In standard linear regression the model trains to minimize the sum of squared error between the true (y_i) and prediction, (y'_i) target values, or residual sum of squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - y'_i)^2$$

- Regularized regression learners are similar, except they attempt to minimize RSS *and* some penalty for the total size of the coefficient values, called a shrinkage penalty because it attempts to "shrink" the model.

$$RSS + \ penalty$$

- There are two common types of regularized learners for linear regression:
  - Lasso regression
  - Ridge regression

  The only formal difference is the type of shrinkage penalty used.

Lasso Regression

- Lasso regression—also known as L1 regularization—is a form of regularization for linear regression models.

- Lasso is similar to ridge, except the shrinkage penalty is a tuning hyperparameter multiplied by the sum of the absolute value of all coefficients:

$$\frac{1}{2n} RSS + \alpha \sum_{j=1}^{p} |\beta'_j|$$

- where n is the number of observations.
- α is the regularization hyperparameter, and $\sum_{j=1}^{p} |\beta'_j|$ is the L_1 penalty.

Ridge Regression

- Ridge regression also known as L1 regularization

- In ridge regression, the shrinkage penalty is a tuning hyperparameter multiplied by the squared sum of all coefficients:

$$\boldsymbol{RSS + \alpha \sum_{j=1}^{p} (\beta'_j)^2}$$

where $\boldsymbol{\beta'_j}$ is the coefficient of the $\boldsymbol{j}$ th of p features and $\boldsymbol{\alpha}$ is a hyperparameter.

- $\boldsymbol{\alpha}$ is the regularization hyperparameter, and $\boldsymbol{\sum_{j=1}^{p} (\beta'_j)^2}$ is the L_2 penalty.