# Machine Learning (ML)

**Chapter 5:**

Gradient Descent (GD), Stochastic Gradient Descent (SGD),

and Cross-validation

**Saeed Saeedvand, Ph.D.**

# Outline

## In this Chapter:

- ✓ Gradient Descent (GD)
- ✓ Stochastic Gradient Descent (SGD)
- ✓ Stochastic Gradient Descent (Mini-Batch)
- ✓ Cross-Validation approaches
  - K-Fold Cross-Validation
  - Stratified K-Fold Cross-Validation
  - Leave-One-Out Cross-Validation (LOOCV)
  - Time Series Cross-Validation
  - Group K-Fold Cross-Validation

## Aim of this chapter:

- ✓ Understanding the Gradient Descent and Stochastic Gradient Descent in practical way as practical optimization algorithms. Understand Cross-Validation approaches solutions and techniques.

# What is the Gradient Descent?

- ✓ An optimization algorithm to find the minimum of a cost function.
- ✓ Works based on the slope of the cost function.
- ✓ We compute the partial derivatives of the cost function regarding each parameters of the model.

**Gradient Descent applications:**

- Linear regression:
- Logistic regression (popular classification algorithm)
- Neural networks
- Support vector machines
- Principal component analysis (PCA)
- Clustering
- Recommendation systems
- …

# What is the partial derivative? (Reminder)

- ✓ **If we have a function** that is constantly **changing** through time (change in input changes output), we use derivative to determine **rate of change.**
- ✓ A partial derivative is a derivative, that shows the change in only one chosen variable.
- ✓ we want to **understand how** the function changes with **respect to each variable** while keeping the other constant.

In multivariable functions, such as $f(x,y)$, for each independent variable we calculate slopes separately (because the function's behavior can be different in each direction).

If we assemble slopes of all variables into a vector we call it the **gradient**.

$$f(x,y) = x^2y^4, \quad \frac{\partial f(x^2y^4)}{\partial(x)} = 2xy^4 \quad 2(1)(3)^4 = 162$$

With respect to x

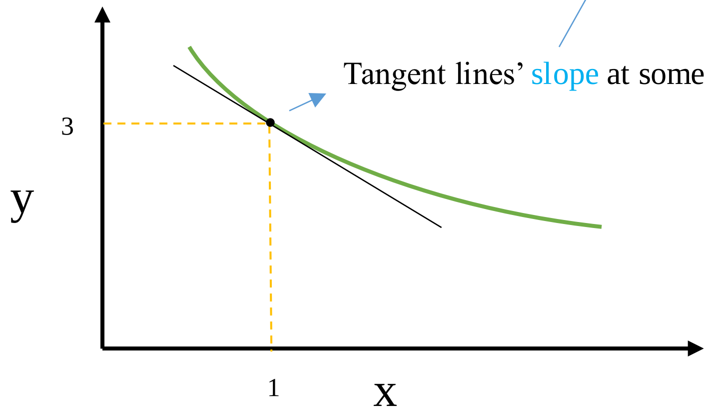
Slope in x-direction

$$f(x,y) = x^2y^4, \frac{\partial f(x^2y^4)}{\partial(y)} = 4x^2y^3 \quad 4 \times 2^2\ 3^3 = 432$$

With respect to y

Slope of the function in the $y$-direction at a given point $(x, y)$.

Tangent lines' slope at some point.




If we assemble partial derivatives of all variables into a matrix we call it the **Jacobian matrix.**

# Gradient Descent (GD)

## How to apply the GD?

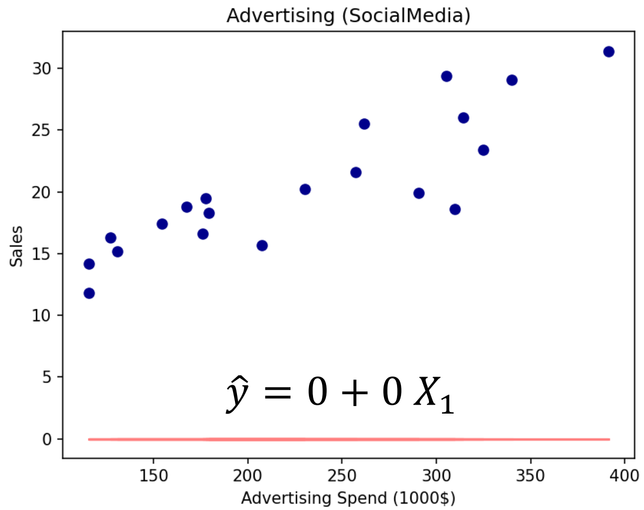
✓ In linear regression we needed to estimate the coefficients $\hat{\theta}_0$ and $\hat{\theta}_1$ for the model.

$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x_1$$

**Step 1**

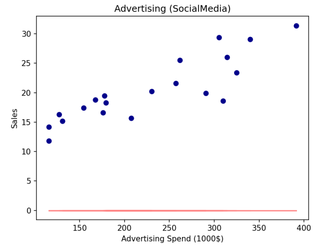
**Initialize the parameters randomly:**

- Choose initial values for $\hat{\theta}_0$ and $\hat{\theta}_1$
- For example, $\hat{\theta}_0 = 0$ and $\hat{\theta}_1 = 0$



# Gradient Descent (GD)

## How to apply the GD?



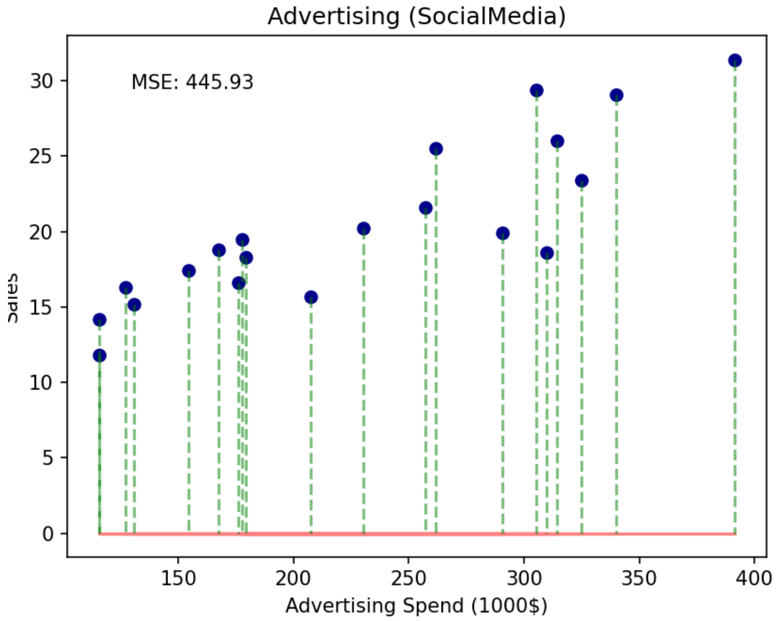

**Step 2** **Calculate the cost function:**

- ✓ We can Calculate different cost functions: RSS, MSE, …
  - ➢ Here compute the (MSE) between the $\hat{y}$ and $y$ for currenting values of $\hat{\theta}_0$ and $\hat{\theta}_1$.
  - ➢ The MSE is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2$$

$$J(\hat{\theta}_0, \hat{\theta}_1) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2$$

Loss or cost function



# Gradient Descent (GD)

## How to apply the GD?

**Step 3** **Calculate the gradient:**

$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x_1$

$$J(\hat{\theta}_0, \hat{\theta}_1) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2 \Rightarrow J(\hat{\theta}_0, \hat{\theta}_1) = \frac{1}{n}\sum_{i=1}^{n}[y_i - (\hat{\theta}_0 + \hat{\theta}_1 x_1)]^2$$

- ✓ We calculate the partial derivatives of the cost function regarding $\hat{\theta}_0$ and $\hat{\theta}_1$ independently, (we called it **gradient** witch we assemble slopes into a vector).

Partial derivative of the cost function with respect to $\hat{\theta}_0$, or when it changes (sign is important)

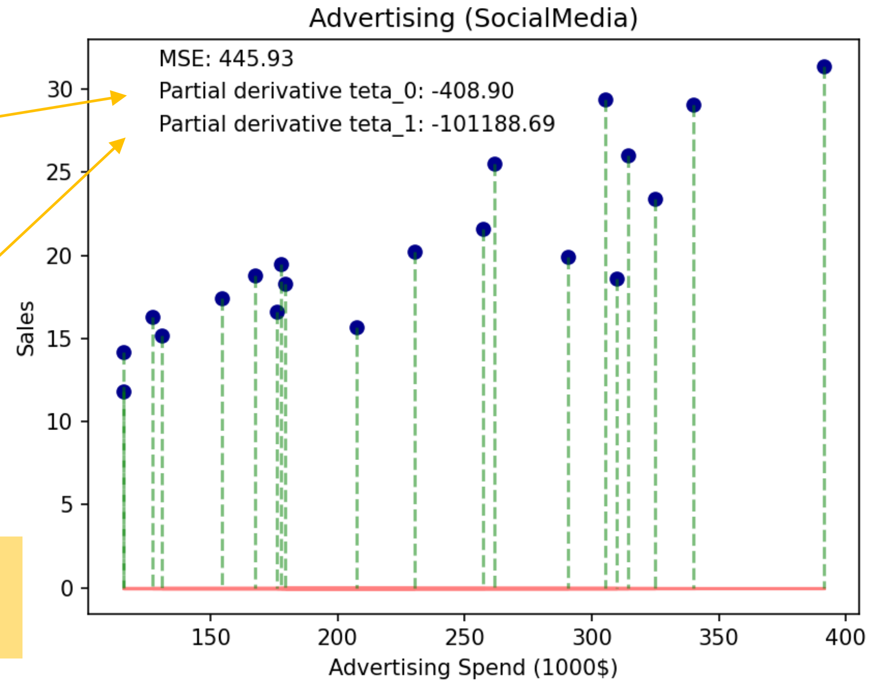
This is simplified result, (since we have learning rate $\alpha$ we can ignore constant -2).

$$\frac{\partial J}{\partial(\hat{\theta}_0)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]$$

$$\nabla J(\hat{\theta}) = \frac{\partial J}{\partial(\hat{\theta})}$$

$$\frac{\partial J}{\partial(\hat{\theta}_1)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]\, x_{i\_1}$$

Gradient



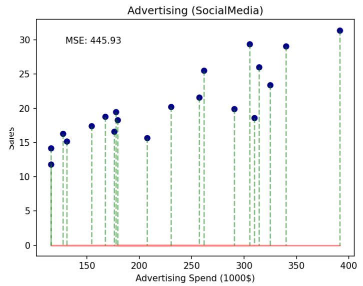

**Note:** these term should be defined based on the objective function and model that we have here MSE).

# Gradient Descent (GD)

## How to apply the GD?

$$\frac{\partial J}{\partial(\hat{\theta}_0)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]$$

$$\frac{\partial J}{\partial(\hat{\theta}_1)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]\, x_{i\_1}$$



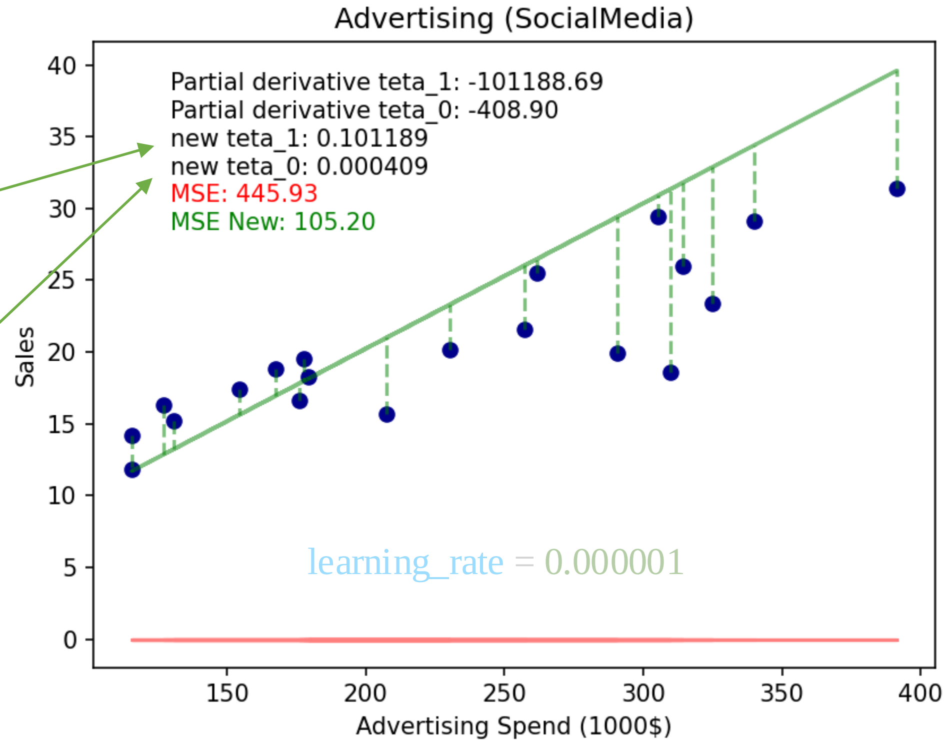

**Step 4** **Update the parameters:**

✓ We Update the values of $\hat{\theta}_0$ and $\hat{\theta}_1$ using the gradients and a learning rate α:

New $\hat{\theta}_0$ and $\hat{\theta}_1$

$$\hat{\theta}_0 = \hat{\theta}_0 - \alpha \frac{\partial J}{\partial(\hat{\theta}_0)}$$

$$\hat{\theta}_1 = \hat{\theta}_1 - \alpha \frac{\partial J}{\partial(\hat{\theta}_1)}$$



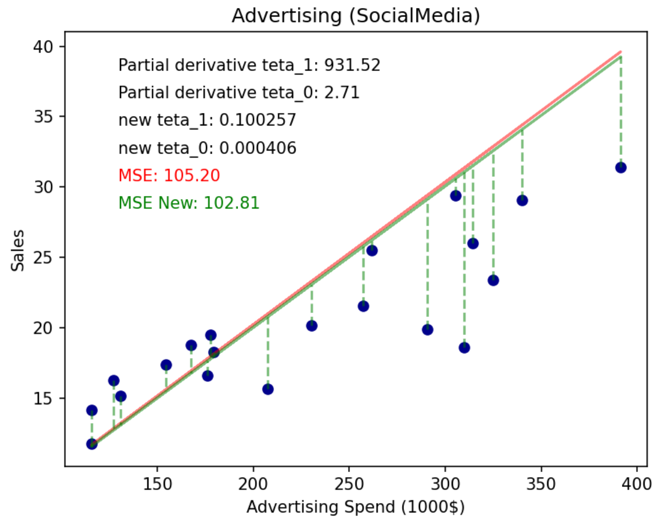

**First iteration**

# Gradient Descent (GD)

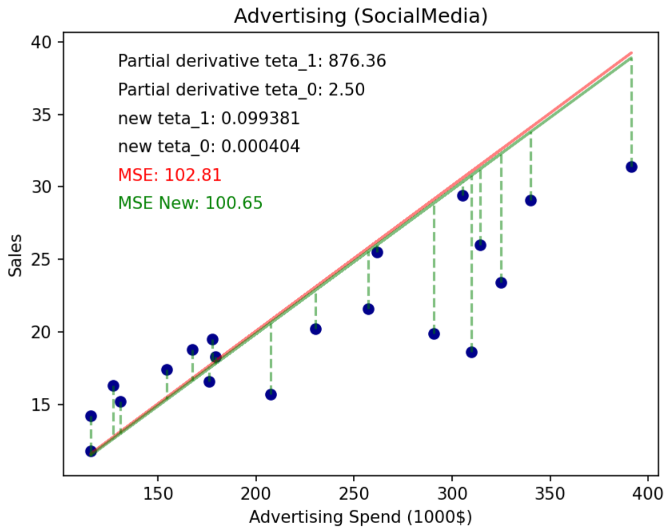
## How to apply the GD?

**Step 5** **Repeat**:

✓ Run steps 2 to 4 until the cost function gets a minimum or a stopping criterion is met (changes are very small).

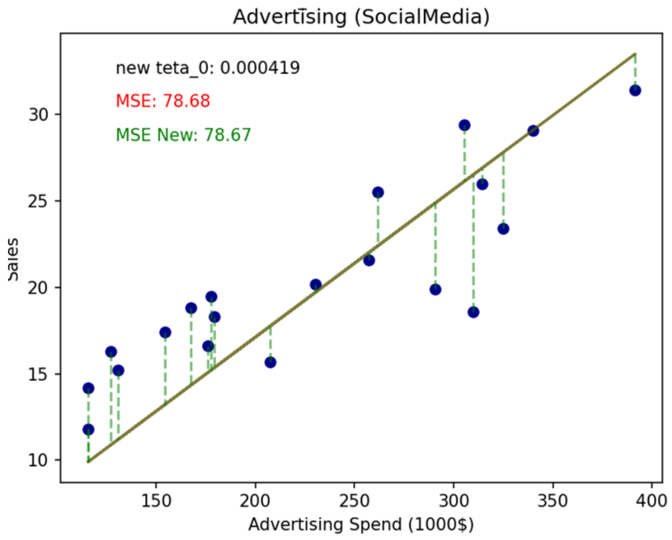



**Second iteration**




**Third iteration**

**...**



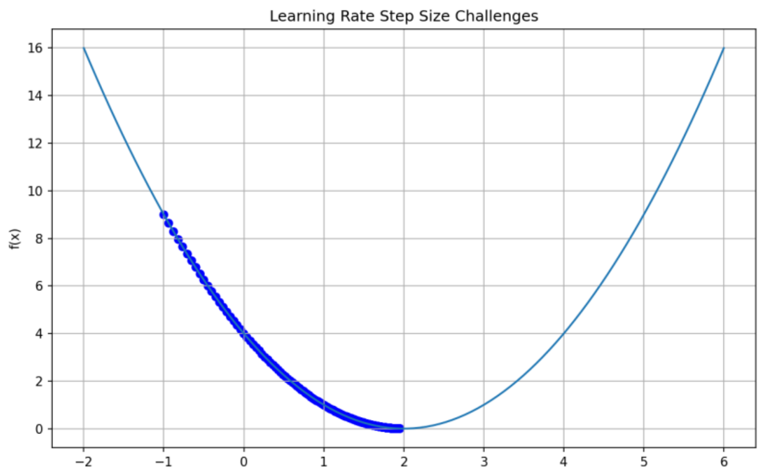

**~15th iteration**

# Gradient Descent (GD)

## Learning Rate Challenges

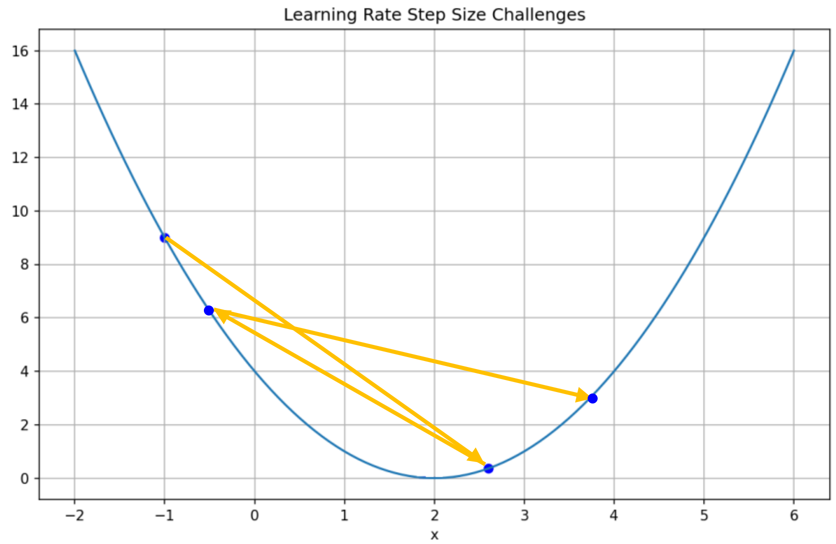
### The Small learning rate:

- Converges slowly and can stuck in local minima




### Large Learning Rate:

- Overshoot, become unstable and diverge



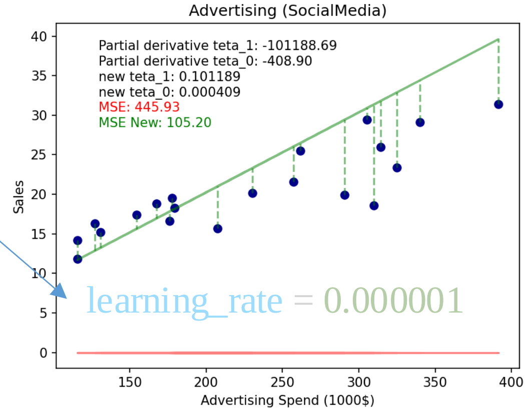

**Stable learning rates:** can avoid local minima and converges smoothly

# Gradient Descent (GD)

## Learning Rate Challenges

✓ For our example if we set big learning rate what happens:

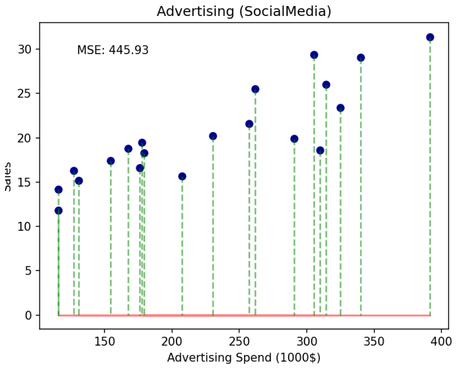
**What was Good learning rate?**



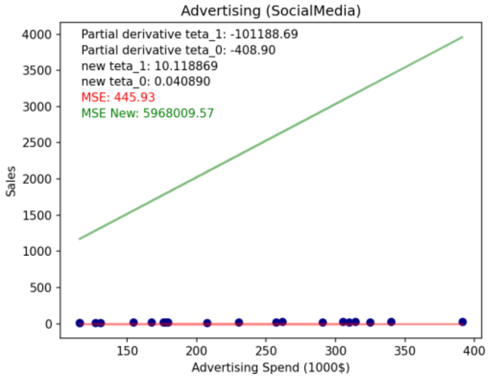

**First iteration**

**Lets try a Bad learning rate for our example:**

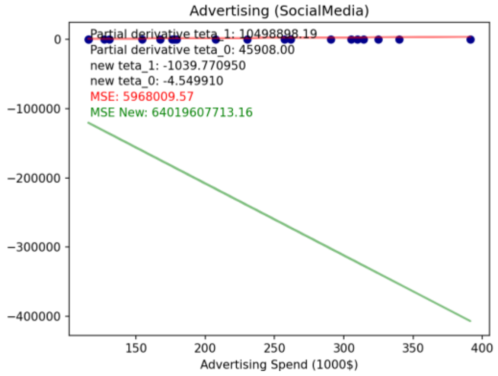
learning_rate = 0.0001



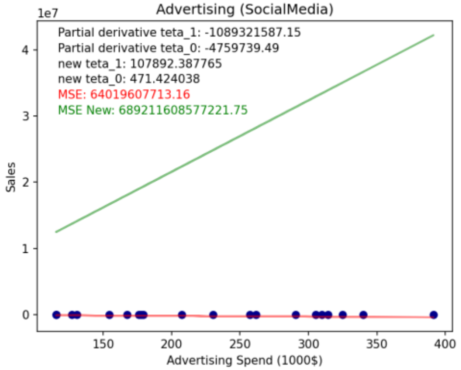

**Initial**




**First iteration**




**second iteration**




**Third iteration**

**...**

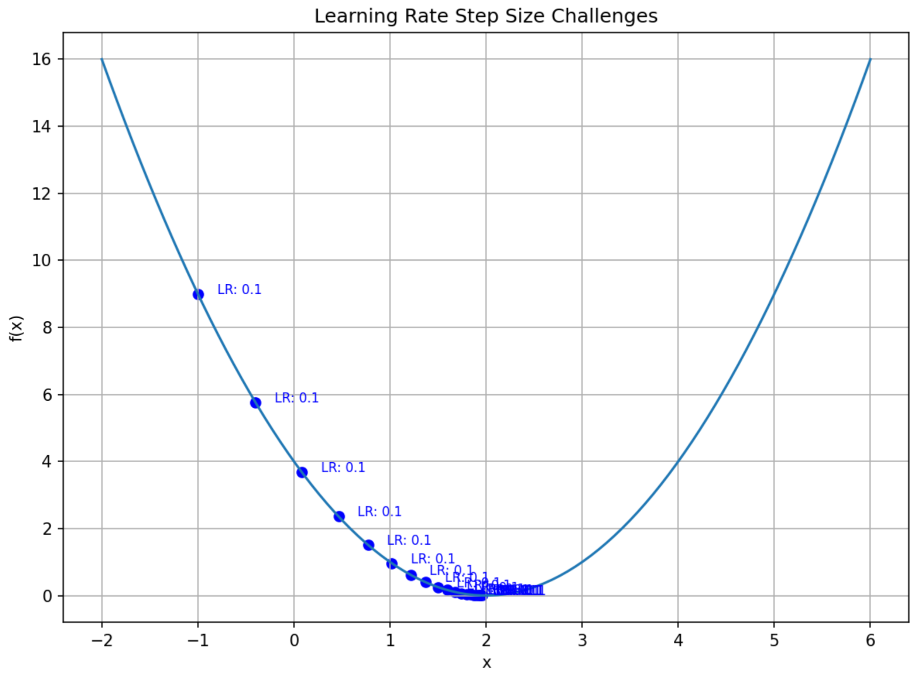
Python Example

# Gradient Descent (GD)

## Learning Rate Challenges

**Proper learning rate:**

- Converges slowly enough to find optimal solution.



# Gradient Descent (GD)

## GD on non-linear regression problem

Step 3 and 4

**Calculate the gradient:**

- ✓ All steps same except partial derivatives of the cost function.

$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x_{i\_1} + (\hat{\theta}_2 x_{i\_1})^2$$

Partial derivatives of the cost function

$$\frac{\partial J}{\partial(\hat{\theta}_0)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]$$

$$\frac{\partial J}{\partial(\hat{\theta}_1)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]\, x_{i\_1}$$

$$\frac{\partial J}{\partial(\hat{\theta}_2)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]\, (x_{i\_1})^2$$

Update of coefficients

$$\hat{\theta}_0 = \hat{\theta}_0 - \alpha \frac{\partial J}{\partial(\hat{\theta}_0)}$$

$$\hat{\theta}_1 = \hat{\theta}_1 - \alpha \frac{\partial J}{\partial(\hat{\theta}_1)}$$

$$\hat{\theta}_2 = \hat{\theta}_2 - \alpha \frac{\partial J}{\partial(\hat{\theta}_2)}$$

# Gradient Descent (GD)

**Summarize GD Algorithm**

Initialize weights (random)

Loop until convergence:

Compute gradient $\frac{\partial J(\hat{\theta})}{\partial \hat{\theta}}$

Update weights, $\hat{\theta} \longleftarrow \hat{\theta} - \alpha \frac{\partial J(\hat{\theta})}{\partial (\hat{\theta})}$

Return weights

Is there any problem with GD?

Computationally expensive for big datasets (calculate for all points)

$$\frac{\partial J}{\partial(\hat{\theta}_0)} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]$$

$\hat{\theta}$ in this general equation is considered as a vector $(\hat{\theta}_0, \hat{\theta}_1, \dots)$

# Stochastic Gradient Descent

**Solve the Complexity problem by SGD!**

Initialize weights (random or zero)

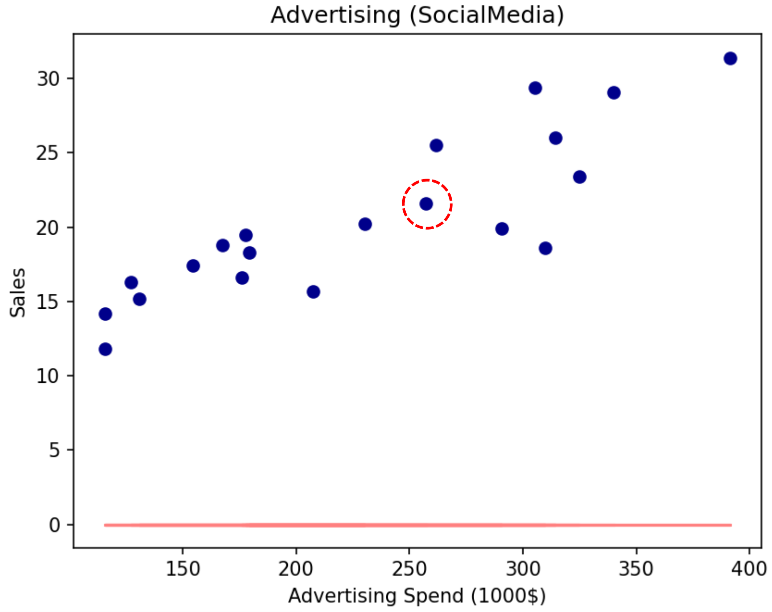
Loop until convergence:

Pick single data Point i

Compute gradient $\frac{\partial J(\hat{\theta})}{\partial \hat{\theta}}$

Update weights, $\hat{\theta} \longleftarrow \hat{\theta} - \alpha \frac{\partial J(\hat{\theta})}{\partial (\hat{\theta})}$

Return weights




Problem?

Noisy because we use only one sample!

# Stochastic Gradient Descent (Mini-Batch)

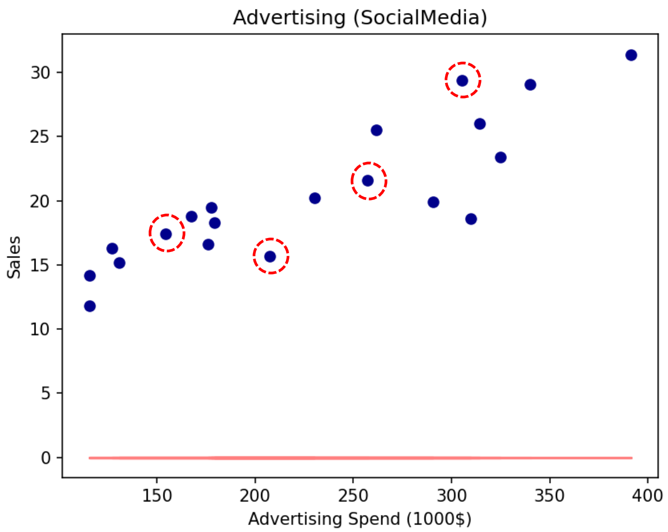
**SGD Mini-Batch Algorithm**

Initialize weights (random or zero)

Loop until convergence:

New loop

For each mini-batch of N data points




Compute gradient $\boxed{\frac{\partial J(\hat{\theta})}{\partial \hat{\theta}}}$

Computationally less expensive

Update weights, $\hat{\theta} \longleftarrow \hat{\theta} - \alpha \frac{\partial J(\hat{\theta})}{\partial (\hat{\theta})}$

Return weights

Benefits

- ✓ Smooth convergence.
- ✓ Higher accuracy for estimating Gradient.
- ✓ We can increase learning rate.
- ✓ Parallelized computation.

# Stochastic Gradient Descent (Mini-Batch)

## Mini-batch Selection

- ✓ As we saw, in SGD we update the weights using the gradients computed for a small subset (mini-batch) of the dataset.
  1. Shuffle the training dataset (to ensure we get more uniform random).
  2. Chose or divide the dataset into mini-batches of size: 16, 32, 64, …, 512 (the most common ones).

**What is the Idea of having multiped baches?**

- ✓ Iterating over the data multiple times can lead to have **better results**.
- ✓ Hardware and memory limitations.

**Practice**

Run SGD on given example, then change to:
A) quadratic, B) multivariate

# Cross-Validation

## Definition

- ✓ The technique to assessing the performance of **machine learning models**.
- ✓ To evaluate how well a model generalizes to unseen or new data.
- ✓ In cross-validation, we divide the dataset is into a number of smaller subsets and we name this folds.
- ✓ Generally **we train model** and test it multiple times with different folds as **test set** (validation set) and **train set**.
- ✓ We repot the average performance of the model over all iterations (metrics we already studied like RSS, MSE, p-value, R-squared, …

# Cross-Validation

## Different Approaches (common ones to use):

- ✓ K-Fold Cross-Validation
- ✓ Stratified K-Fold Cross-Validation
- ✓ Leave-One-Out Cross-Validation (LOOCV)
- ✓ Time Series Cross-Validation
- ✓ Group K-Fold Cross-Validation
- ✓ ..

# Cross-Validation

## k-fold cross-validation

✓ k-fold cross-validation is one of the most common approach of Cross-Validation statistical technique.

1. Divide the dataset into k equally sized folds.
2. Train the model on k-1 folds.
3. Test the model on remaining one fold.
4. Repeat this k times, so that each fold being used as a test set once.
5. Report the average performance of the model over the k iterations.

## Main Advantages

- Can help to fine-tune model hyperparameters.
- Can help to select the best model from a set of candidate models.
- Can help to overcome the risk of overfitting.

# k-fold cross-validation

## How to choose k?

✓ Choosing the value of k for k-fold cross-validation depends on several factors:

- **Dataset size**: If you have a small dataset, choose a larger value of k to ensure that each fold has enough data points.
- **Computational cost**: Increasing the value of k, increases the number of times the model needs to be trained.
- **Bias-Variance trade-off**: A smaller k may lead to higher variance in the performance estimate.
- **Nature of the data**: If the data has a specific structure, you may need to use a specialized cross-validation technique.

A **common choice for k** is 5 or 10, (these values experimentally shown a good balance between computational cost and performance).

# Cross-Validation

## Challenge of the K-Fold Cross-Validation

- ✓ In K-Fold we divide data into K equally sized folds without considering the distribution of the classes (i.e., the labels).

- ✓ In K-Fold the dataset **we may have** many samples of a particular class but others only a few (in imbalanced datasets can cause problems).

# Cross-Validation

## Stratified K-Fold Cross-Validation

- ✓ Similar to K-Fold Cross-Validation but **differ** in how the samples are distributed among the folds.
- ✓ Stratified K-Fold ensures that each fold maintaining the same proportion of samples.
- ✓ So we have **similar distribution** of the data from each class as in the original dataset.

# Cross-Validation

## Leave-One-Out Cross-Validation (LOOCV)

- ✓ A special case of K-Fold Cross-Validation, so that K equals the number of samples in the dataset.
- ✓ **At each iteration** uses a single data point for validation and the remaining points for training.
- ✓ LOOCV is computationally expensive.
- ✓ LOOCV has higher variance in performance estimates.
- ✓ LOOCV is useful in Small datasets, and Stable models (less sensitive to small changes in the training data).

# Cross-Validation

## Time Series Cross-Validation

- ✓ Specifically designed for time series data, **specially** if the order of the samples is important.
- ✓ In each iteration, we train the model up to a certain time point on the data then we validate the model data after that time point.
- ✓ We **repeat** this process by moving this time window forward.

# Group K-Fold Cross-Validation

✓ When the **dataset contains groups** of related samples:

1. **Split the data into k groups:** First, identify the groups within the dataset (for example one hospital data, or one university data).
2. **Exclude validation set:** At each iteration, exclude one group as the validation set, and **train the model**.
3. **Evaluate models**: calculate the average performance.
4. **Repeat:** Run steps 2-4 this for each of the k groups (iterations).
5. **Report:** Return total average performance metric.

**Assignment**

Use or extend a sample dataset for our advertising problem, apply one of the proper Cross-Validation approaches and train SGD minibatch to fit the data (use 3 medias).

# Summery

- ✓ We understood the Gradient Descent (GD) optimization approach.
- ✓ We extended GD to Stochastic Gradient Descent (SGD).
- ✓ W improved SGD by Mini-Batch technique.
- ✓ We introduced Cross-Validation approaches including:
  - K-Fold Cross-Validation
  - Stratified K-Fold Cross-Validation
  - Leave-One-Out Cross-Validation (LOOCV)
  - Time Series Cross-Validation
  - Group K-Fold Cross-Validation.