Berkeley Engineering | BerkeleyHaas

PROFESSIONAL CERTIFICATE IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Module 15

Gradient Descent and Optimization

Office Hours with Viviana Márquez
January 9, 2024

Happy New Year!

I hope you all had a fantastic break :)
Fun fact: It's a leap year!

AGENDA

- Kaggle Data Science & ML Survey
- Required activities for Module 15
- Content review Module 15: Gradient Descent and Optimization
- Code examples from the industry
- Questions

AGENDA

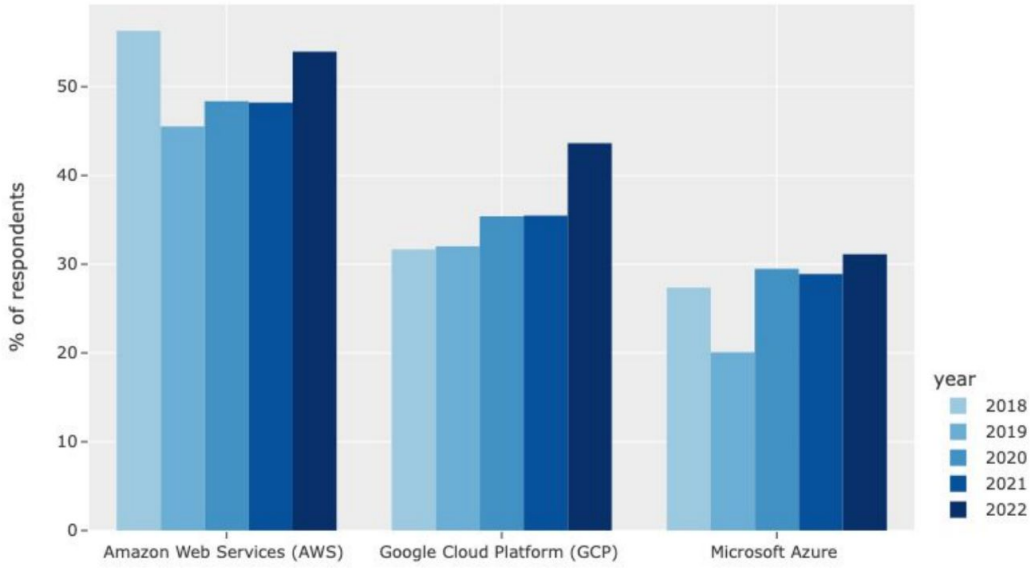
- Kaggle Data Science & ML Survey
- Required activities for Module 15
- Content review Module 15: Gradient Descent and Optimization
- Code examples from the industry
- Questions

Kaggle Data Science & ML Survey

- https://www.kaggle.com/kaggle-survey-2022

Kaggle DS & ML Survey 2022

All major cloud computing providers saw strong year over year growth in 2022



Google Cloud

AGENDA

- ✅ Kaggle Data Science & ML Survey
- Required activities for Module 15
- Content review Module 15: Gradient Descent and Optimization
- Code examples from the industry
- Questions

Required Activities for Module 15

- 🤖 Codio Activity 15.1: Introduction to Gradient Descent
- 🤖 Codio Activity 15.2: Manual Gradient Descent
- 🤖 Codio Activity 15.3: Gradient Descent and Linear Regression
- 🤖 Codio Activity 15.4: Convexity
- 🤖 Codio Activity 15.5: Concavity with Second Derivatives
- 🤖 Codio Activity 15.6: Computing the Gradient of a Two-Dimensional Function
- 🤖 Codio Activity 15.7: Gradient Descent with Two Features
- 🤖 Codio Activity 15.8: Tracing the Descent
- 🤖 Codio Activity 15.9: Stochastic Gradient Descent
- 🤖 Codio Activity 15.10: Comparing GD with Stochastic Gradient Descent
- 🤖 Codio Activity 15.11: The Bias–Variance Trade-Off
- 🤖 Quiz 15.1: Gradient Descent Optimization
- 🤖 Quiz 14.1: Decision Trees

AGENDA

- ✅ Kaggle Data Science & ML Survey
- ✅ Required activities for Module 15
- Content review Module 15: Gradient Descent and Optimization
- Code examples from the industry
- Questions

Gradient descent

- Mathematical **optimization** technique used to find the minimum of a function

Gradient descent

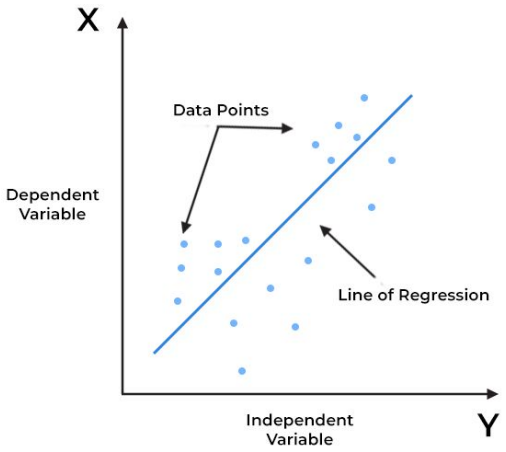
- Mathematical **optimization** technique used to find the minimum of a function

In machine learning, we optimize a lot of stuff…

Gradient descent

- Mathematical **optimization** technique used to find the minimum of a function

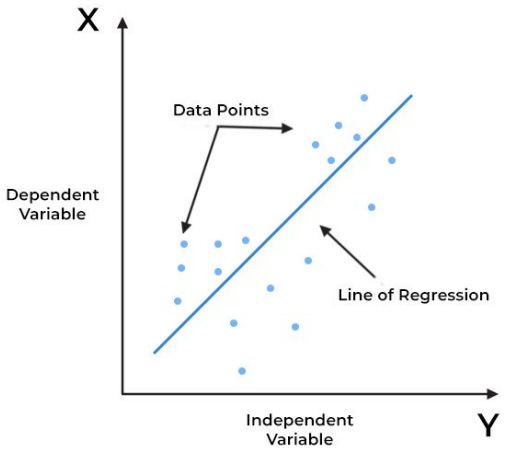
In machine learning, we optimize a lot of stuff…

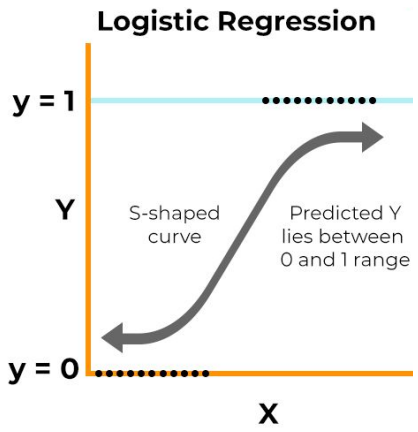


Gradient descent

- Mathematical **optimization** technique used to find the minimum of a function

In machine learning, we optimize a lot of stuff…

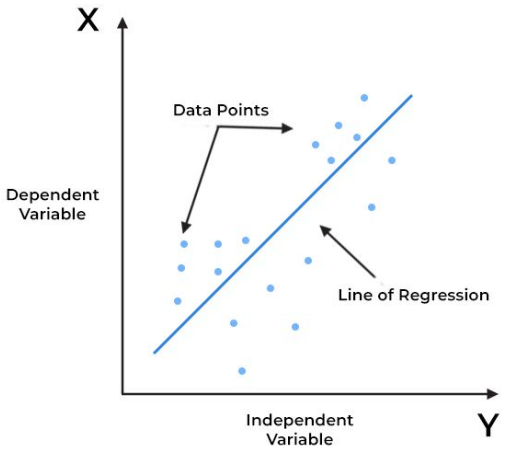


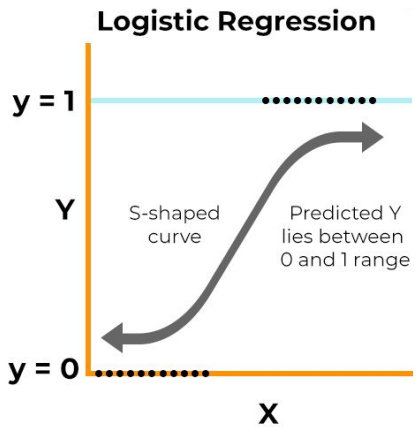


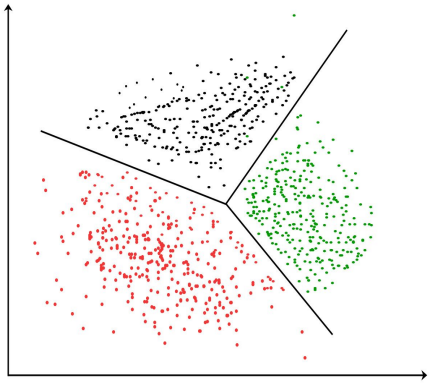
Gradient descent

- Mathematical **optimization** technique used to find the minimum of a function

In machine learning, we optimize a lot of stuff…



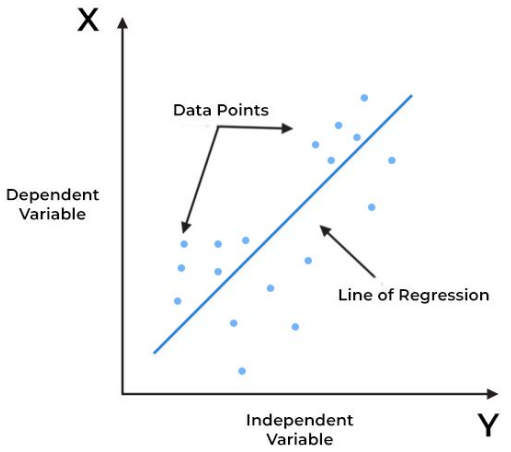


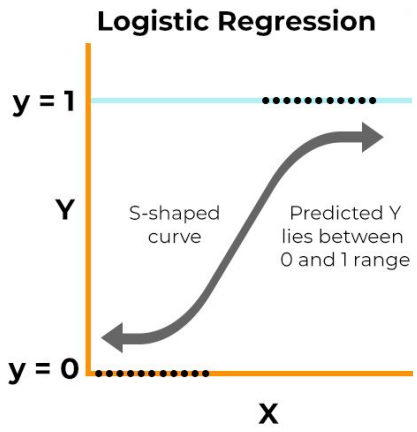


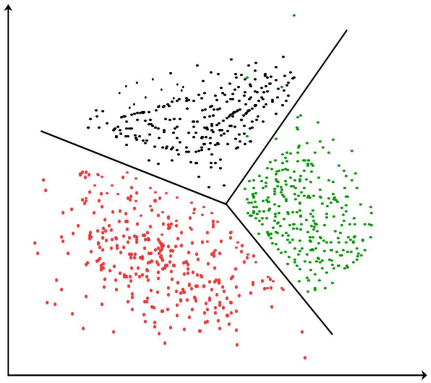
Gradient descent

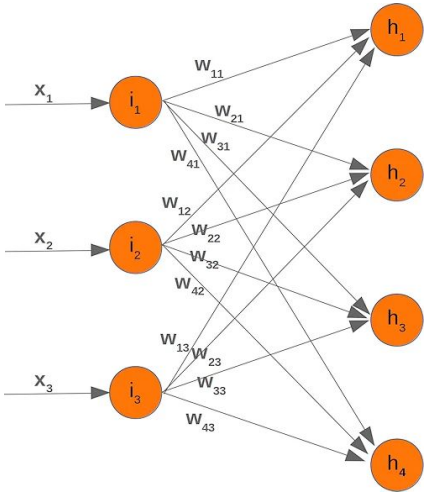
- Mathematical **optimization** technique used to find the minimum of a function

In machine learning, we optimize a lot of stuff…









Gradient descent

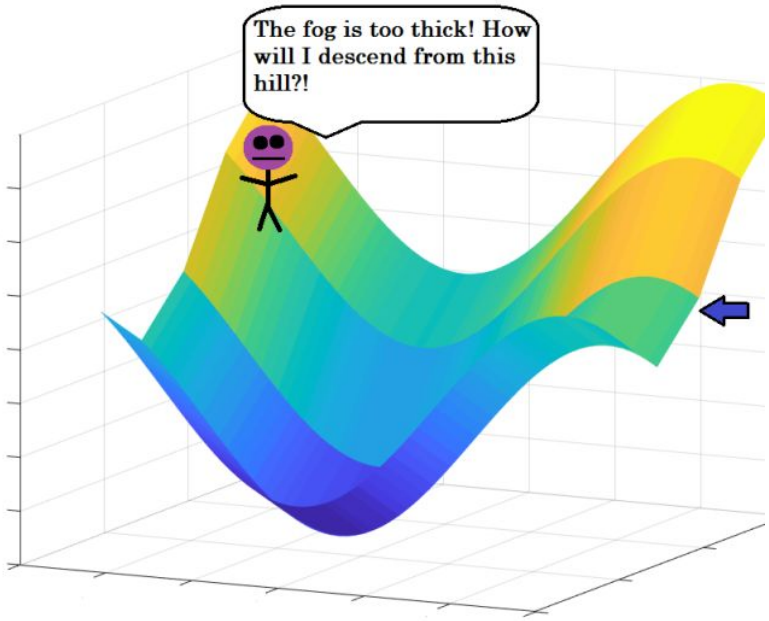
- Mathematical **optimization** technique used to find the minimum of a function
  - **Goal:** Find the minimum value of a function

    In machine learning, this is typically the **loss function** (remember, a loss function measures the extent of **errors** made by a model)
  - **Gradient:** Slope of the function at a particular point (think of the derivative)
  - **Descent:** Refers to moving downwards. You want to know the direction in which the function decreases the fastest
  - **Iteration:** It's an iterative process
  - **Learning rate:** Size of the steps

Gradient descent

- Mathematical **optimization** technique used to find the minimum of a function

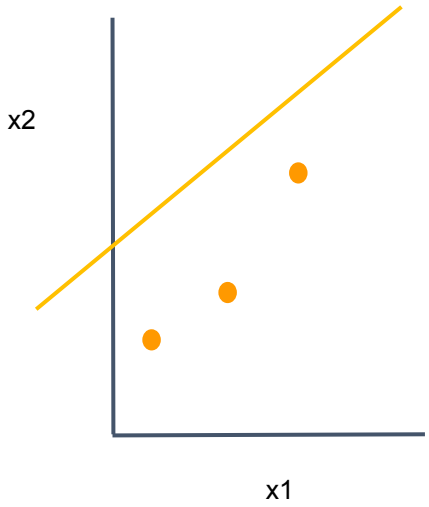
- Gradient descent is like walking down a hill by taking steps in the direction where the slope descends the most steeply, with the aim of reaching the bottom

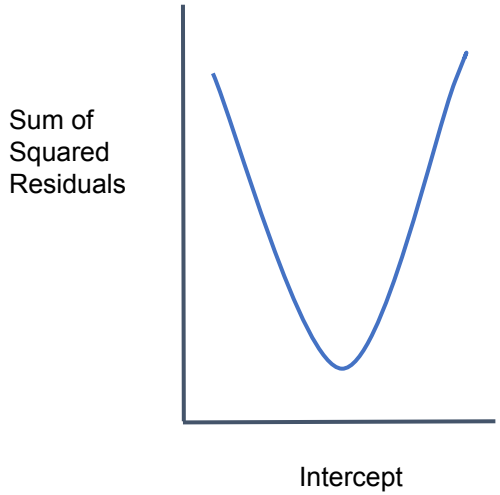


- **Starting point:**
  Imagine you're staring in a hill and your goal is to get to the lowest point of the valley
- **Looking around:**
  You look around to see which way is downhill.
- **Taking a Step:**
  You take a step in the direction that goes most steeply downhill.
- **Repeating the Process:**
  You keep taking steps downhill and reassessing the direction each time. As you get closer to the lowest point, your steps become smaller so you don't overshoot it.

Example:

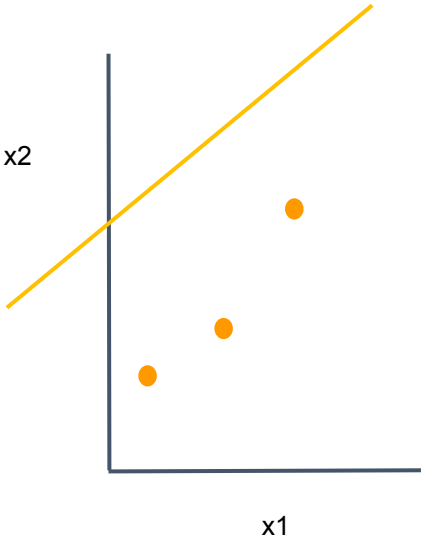
Find the intercept value that minimizes the loss function

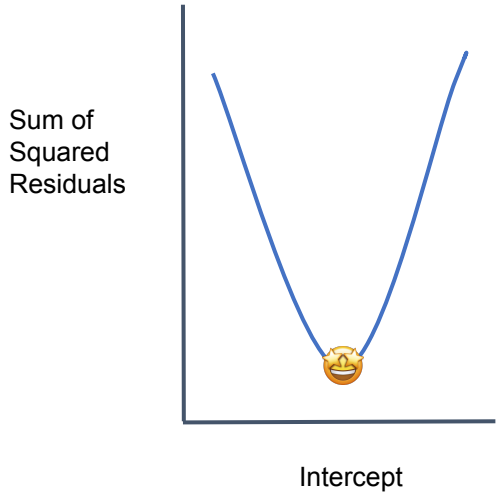




Example:

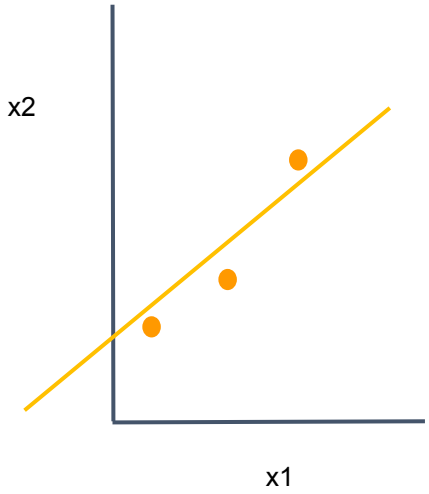
Find the intercept value that minimizes the loss function

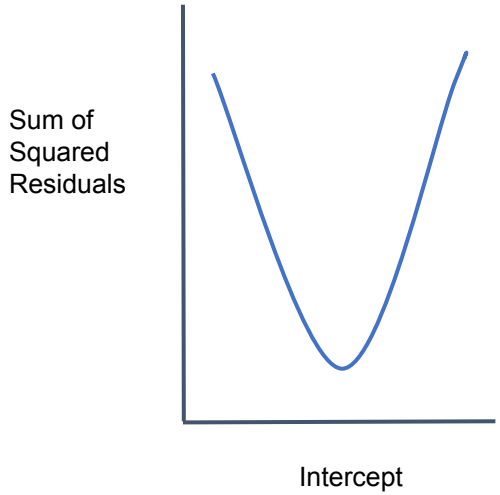




Example:

Find the intercept value that minimizes the loss function

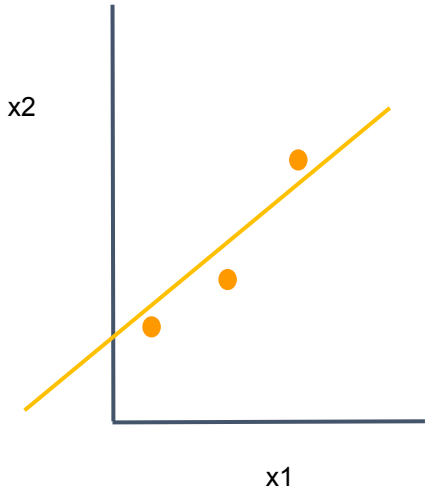


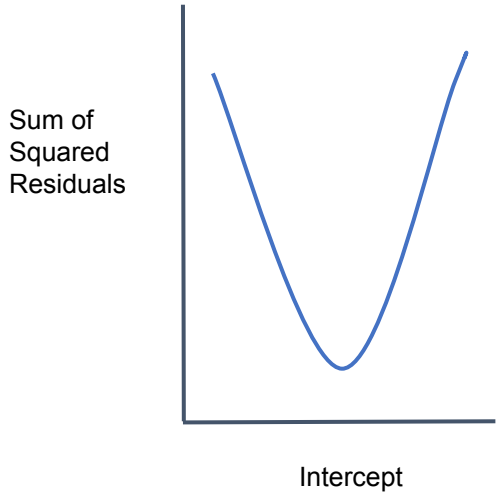


Derivative: Slope (steepness/incline) of a curve at a specific point

Example:

Find the intercept value that minimizes the loss function



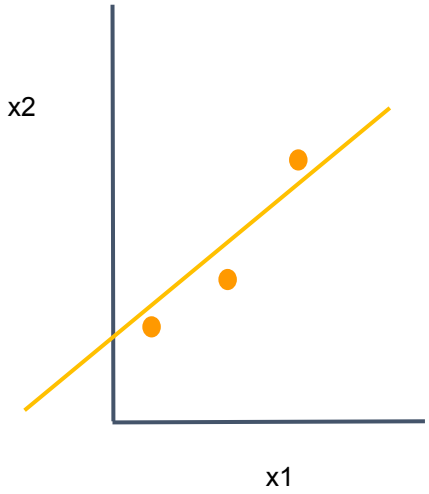


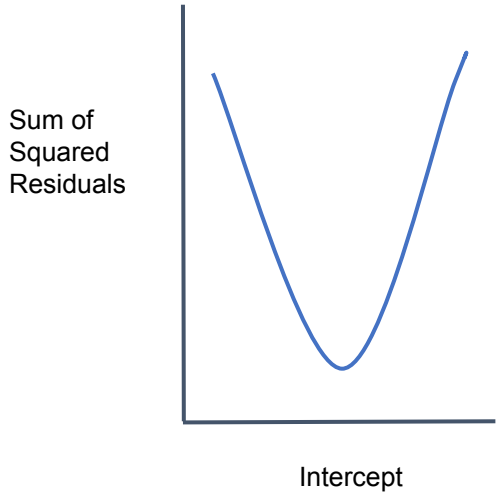
Derivative: Slope (steepness/incline) of a curve at a specific point

Gradient: Generalization of the derivative. It extends this idea to function of multiple variables

Example:

Find the intercept value that minimizes the loss function



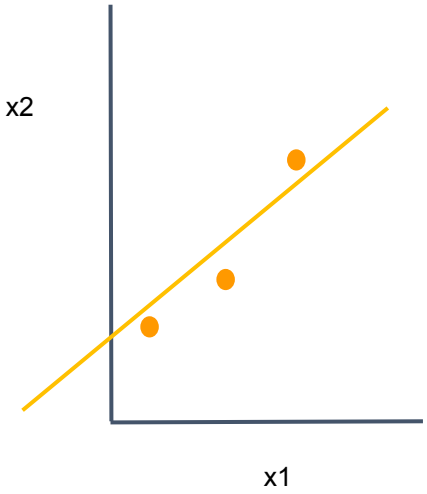


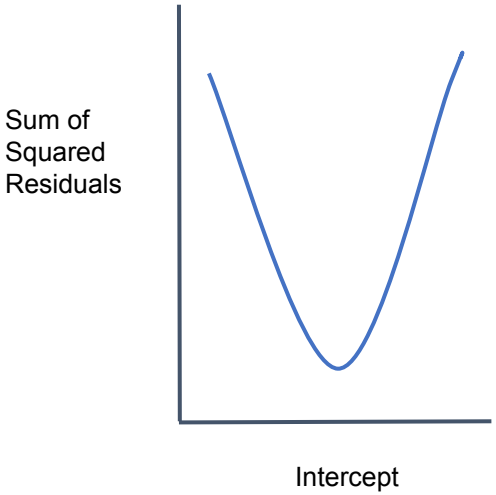
Derivative: Slope (steepness/incline) of a curve at a specific point

Gradient: Generalization of the derivative. It extends this idea to function of multiple variables

Example:

Find the intercept value that minimizes the loss function

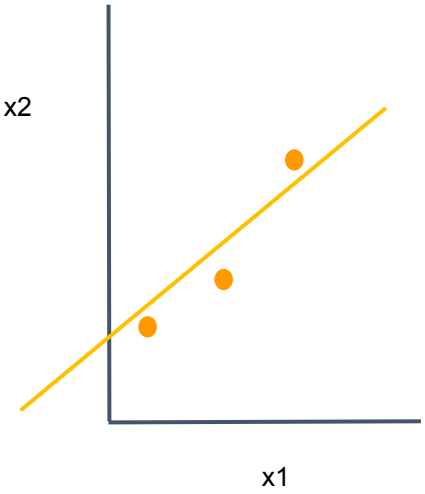


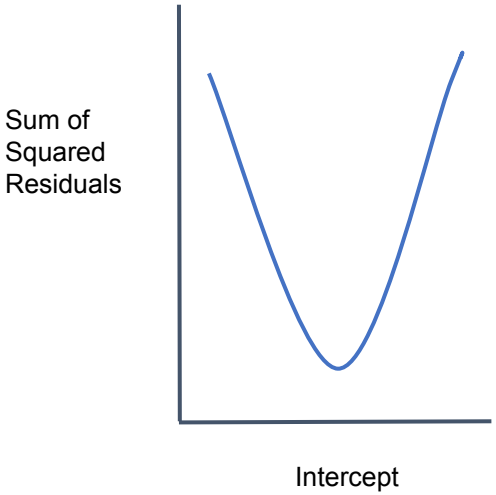


A **gradient** is like an arrow that points in the direction of the steepest increase on a surface.

Example:

Find the intercept value that minimizes the loss function



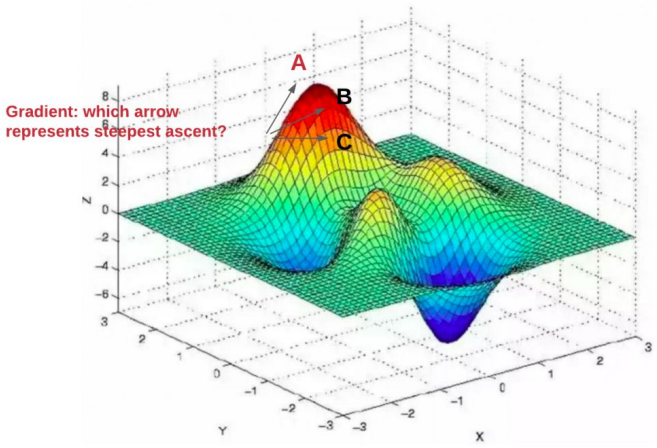


A **gradient** is like an arrow that points in the direction of the steepest increase on a surface.

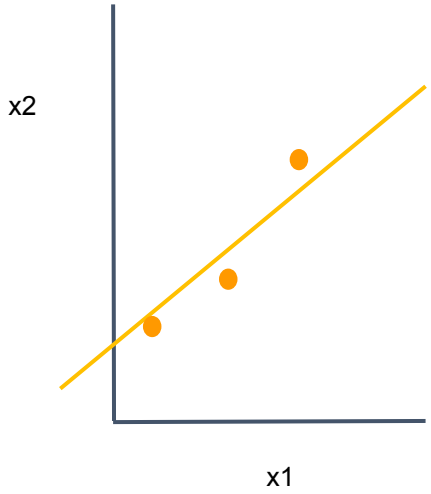
If you imagine being on a hill and you want to know which way is uphill, the gradient would point in that direction. If you want to go downhill (like in gradient descent), you'd go in the opposite direction of the gradient.

Gradient descent

- The **gradient** of the loss gives you the direction of the steepest ascent
- To minimize, you move the opposite way (towards the steepest descent)



Types of gradient descent



Types of gradient descent

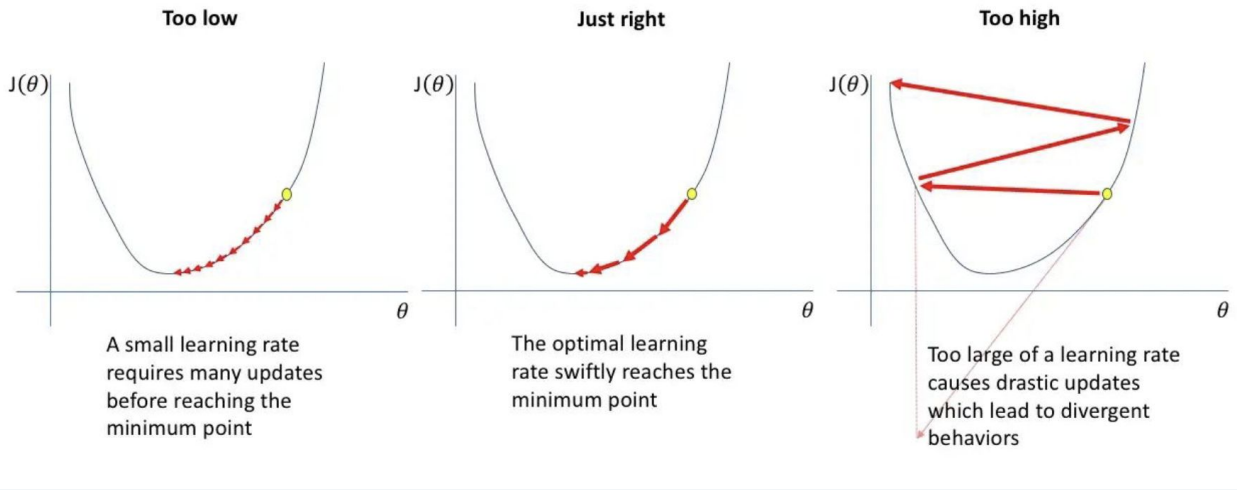
- **Batch Gradient Descent**: Computes the gradient using the entire dataset. This is computationally expensive and infeasible for large datasets.

- **Stochastic Gradient Descent (SGD)**: Computes the gradient using just one example at each iteration. It's faster and can escape local minima because of its inherent noise, but it's also much noisier and might not converge as smoothly.

- **Mini-Batch Gradient Descent**: A compromise between the two. It computes the gradient using a small random sample (mini-batch) of the dataset. This is the most common method in practice, especially in the context of deep learning.

Learning rate in gradient descent

- The learning rate is a hyperparameter (i.e., picked by you before any learning is made!)
- It is the size of the steps taken during the optimization process
  - Too high: The algorithm might overshoot the optimal solution
  - Too low: The algorithm might take too long to converge
  - It needs to be just right



Learning rate in gradient descent

new parameter = old parameter - learning rate x gradient

Additional resources

- Gradient Descent by StatQuest
- Concavity by The Organic Chemistry

AGENDA

- ✅ Kaggle Data Science & ML Survey
- ✅ Required activities for Module 15
- ✅ Content review Module 15: Gradient Descent and Optimization
- Code examples
- Questions

Code

- https://colab.research.google.com/drive/1BCvSxsbLArVo_3rwwQKK1ht2ANRN_jkf?usp=sharing

AGENDA

- ✅ Kaggle Data Science & ML Survey
- ✅ Required activities for Module 15
- ✅ Content review Module 15: Gradient Descent and Optimization
- ✅ Code examples from the industry
- Questions

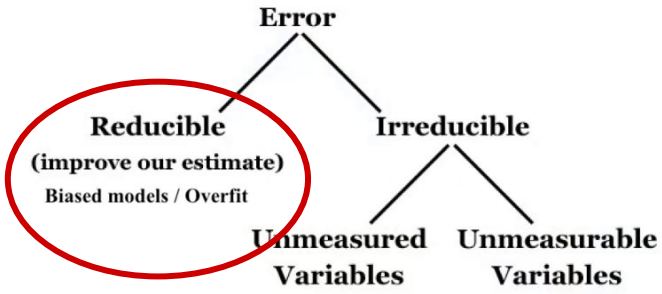
QUESTIONS?

AGENDA

- ✅ Kaggle Data Science & ML Survey
- ✅ Required activities for Module 15
- ✅ Content review Module 15: Gradient Descent and Optimization
- ✅ Code examples from the industry
- ✅ Questions

APPENDIX

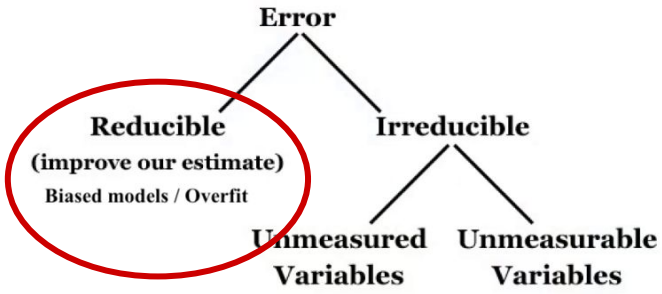
Sources of prediction error



There are three sources of errors in that ***Err*** number:

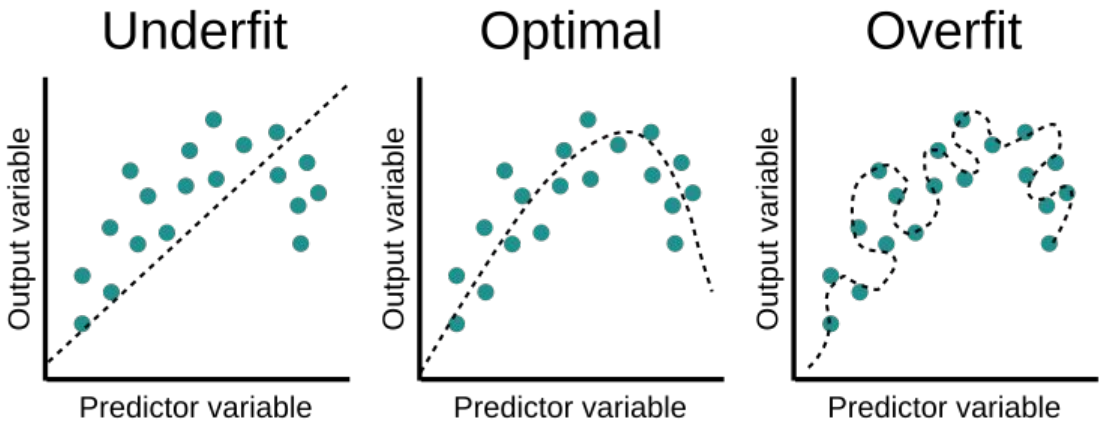
1. Noisy **X** or **y** data, such as inconsistent **X**→**y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Sources of prediction error

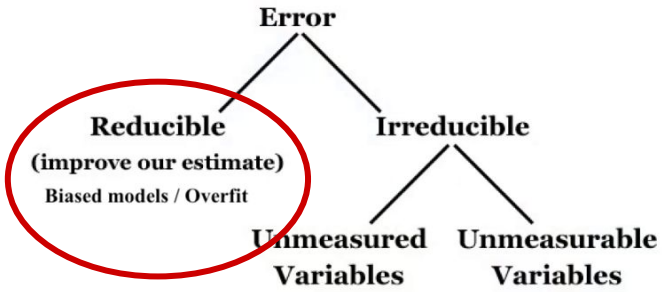


There are three sources of errors in that ***Err*** number:

1. Noisy **X** or **y** data, such as inconsistent **X**→**y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data



Sources of prediction error



There are three sources of errors in that ***Err*** number:

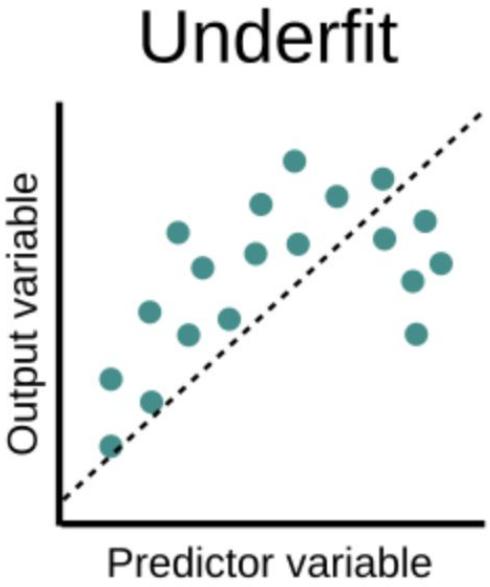
1. Noisy **X** or **y** data, such as inconsistent **X→y**
2. Model underfitting or bias: Too weak or simple
3. Model overfitting: Model too specific to training data

Underfit

Optimal

Overfit

Overly simple models lead to biased models



- **Bias** is the error rate of your model on the training set
- Bias is how much your model **underfits** the training data

Overly simple models lead to biased models

How do you compute bias?

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - y]$$

Expected difference between predicted and observed

Overly simple models lead to biased models

👮‍♀️ Check-in

A model that has a good ability to fit the training data has _____ bias

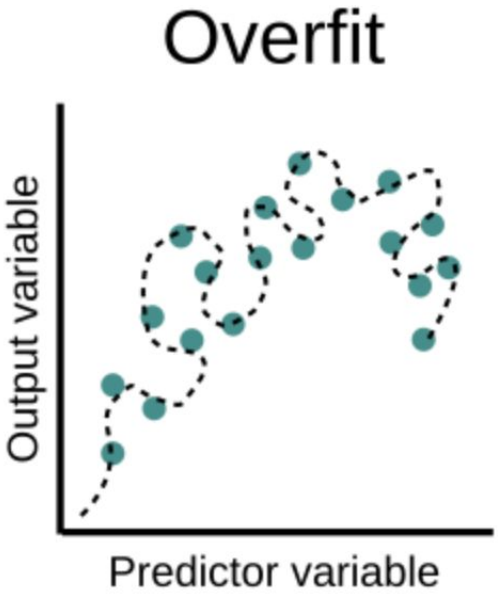
Overly simple models lead to biased models

👮‍♀️ Check-in

A model that has a good ability to fit the training data has **LOW** bias

- We want to minimize bias
- Models with high bias:
  - Fail to capture meaningful patterns in data
  - Under-fit training data
- How to decrease bias? Make model more complex!
  - Add more parameters
  - Pick a different model

Overly complex models can overfit



- **Variance** is the amount a model's prediction will change if a different training data is used, ie, small changes in the training data can result in large changes in the estimated model
- **Variance** in a model is the flexibility to learn patterns in the observed data
- Variance is how much your model **overfits** the training data

Overly complex models can overfit

How do you compute variance?

$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - (E[\hat{f}(x)])^2$$

Intuitively, how much the algorithm will move around its mean

Overly complex models can overfit

👮‍♀️ Check-in

A model that is strongly influenced by the specifics of the training data has _____ variance

Overly complex models can overfit

👮‍♀️ Check-in

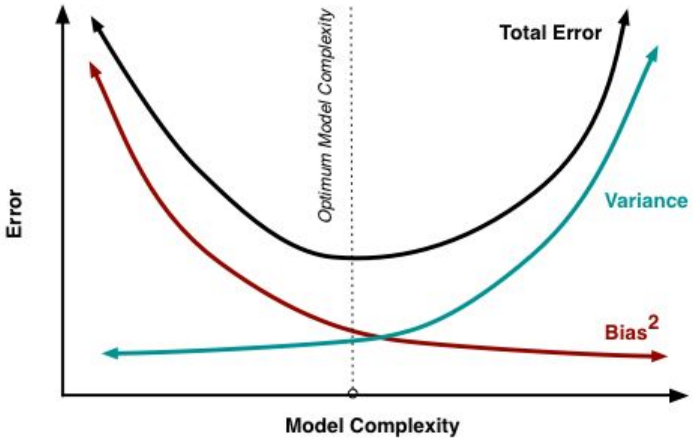
A model that is strongly influenced by the specifics of the training data has **HIGH** variance

- We want to minimize variance. A model that has a good ability to predict test data has **low** variance.
- The more complex the model is, the more data points it will "capture". However, complexity will make the model "move" more to "capture" the data points, and hence its variance will be larger
- How to decrease variance? Make model less complex!
  - A larger training set tends to decrease variance, ie, reduces the chance of overfitting --> increases the chance of generalization
  - Regularization

Bias-variance trade-off

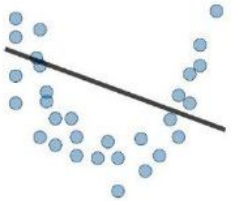
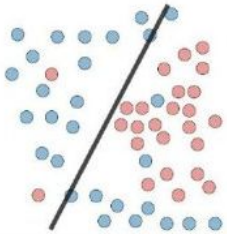
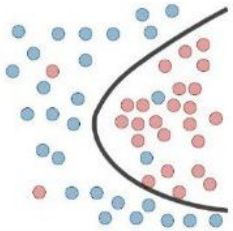
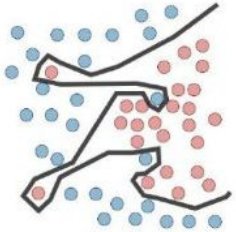
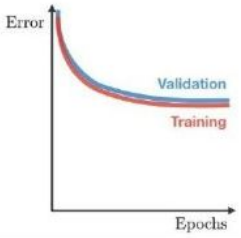
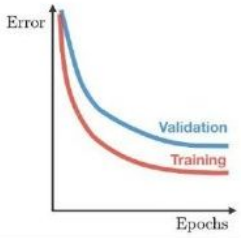
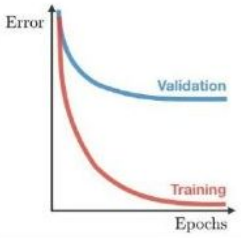
So... we need to decrease both bias and variance (to avoid underfitting and overfitting)

Bias-variance tradeoff



Let the validation dataset be your guide to approximate complexity

- Collect a lot of data
- Engineer good features
- Pick a complex algorithm
- Train the specific model until validation scores starts to go down (smart early stopping)

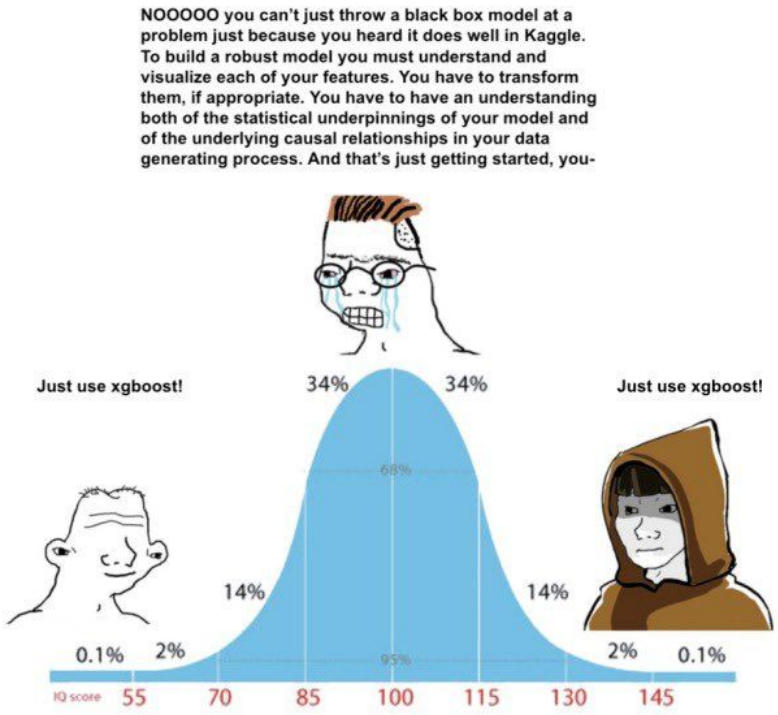
	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			
Deep learning			
Remedies	- Complexify model - Add more features - Train longer		- Regularize - Get more data

Summary

- **Bias** is the error rate of your model on the training set. Bias is how much your model **underfits** the training data
- **Variance** is the amount a model's prediction will change if a different training data is used. Variance is how much your model **overfits** the training data
- We want to always **minimize both bias and variance**, but when one goes down, the other one goes up (more/less complex model)– hence **bias-variance tradeoff**

Overfit memes

Example of boosting: XGBoost



- XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library

- Example