

# Welcome to SEAS Online at George Washington University

**Class will begin shortly**

**Audio:** To eliminate background noise, please be sure your audio is muted. To speak, please click the hand icon at the bottom of your screen (**Raise Hand**). When instructor calls on you, click microphone icon to unmute. When you've finished speaking, ***be sure to mute yourself again.***

**Chat:** Please type your questions in Chat.

**Recordings:** As part of the educational support for students, we provide downloadable recordings of each class session to be used exclusively by registered students in that particular class. **Releasing these recordings is strictly prohibited.**

# SEAS 8520 – Lecture 2: Regression

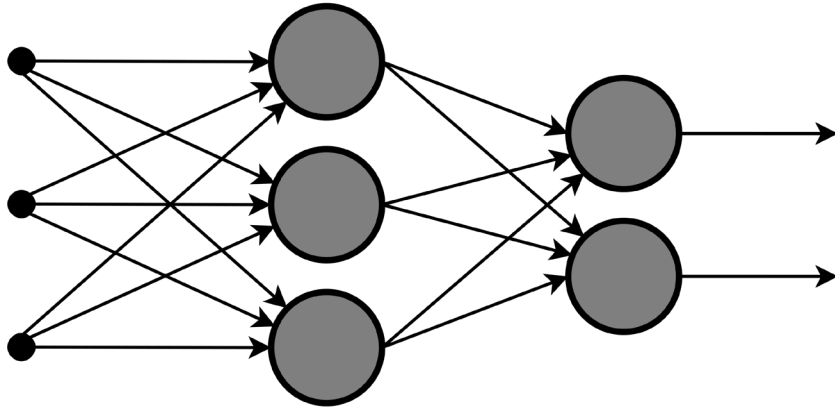
Walid Hassan, M.B.A, D.Eng.

# Agenda

- Linear Regression with One Variable
- Linear Regression - Cost Function
- Linear Regression - Normal Equation and Gradient Descent
- Logistic Regression
- Logistic Regression - Cost Function
- Linear Regression - Gradient Descent for Multiple Variables

# Linear Regression with One Variable

---



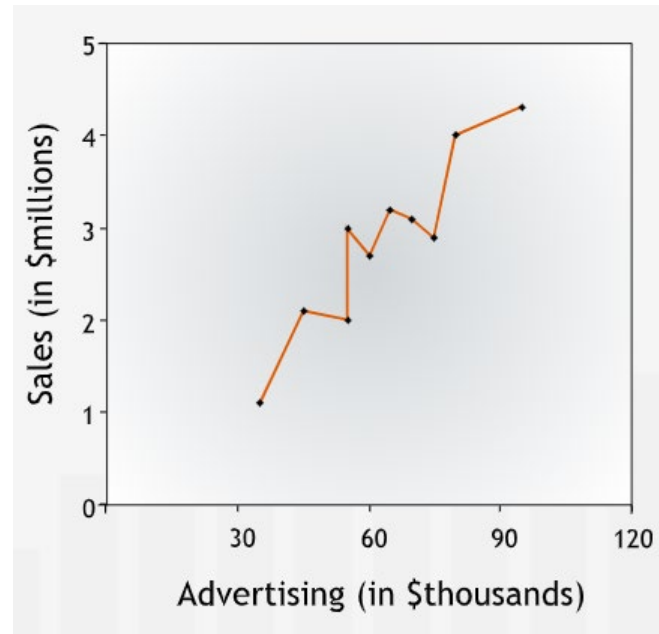
Model  
representation

# Linear Regression

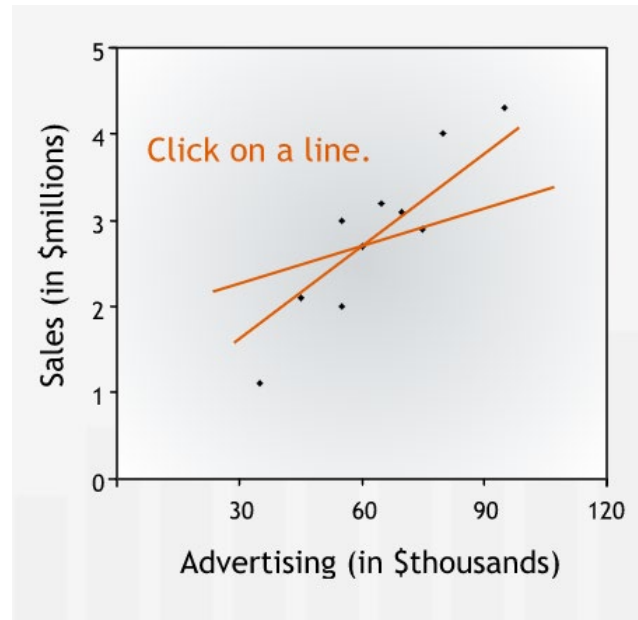
- Linear regression is a statistical method used to model and analyze the relationships between a numeric dependent variable and one or more independent variables.
- Linear Regression is used for problems where the output is a continuous value.
- The main goal of linear regression is to find the best fit straight line that accurately predicts the output values.
- Example: predicting house prices, stock market prices, or temperatures.

# Regression Line Explained - 1

Depicted below is a relationship between advertising and sales for a company. As we can see from the graph, no straight line could be drawn that would pass through every point in the dataset.

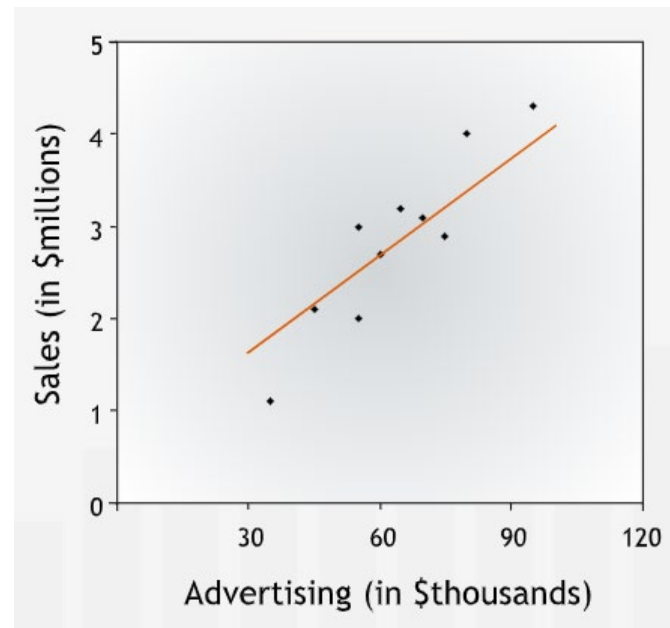


We can draw multiple lines. Which of the two lines best fits the data?  
it is useful to have a precise **measure** of a line's accuracy.



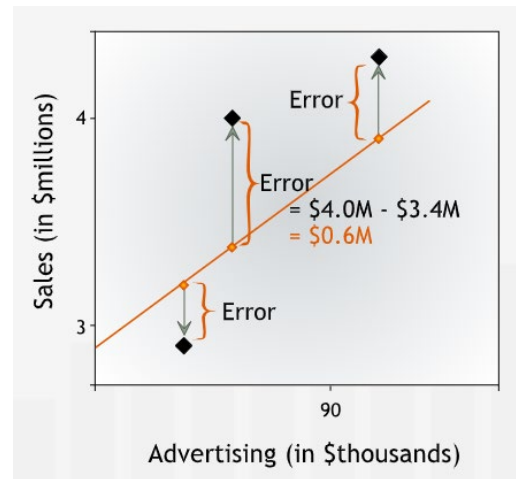
Let's look at one line we could draw through our dataset.

To measure the accuracy of a line, we'll quantify the dispersion of the data around the line.

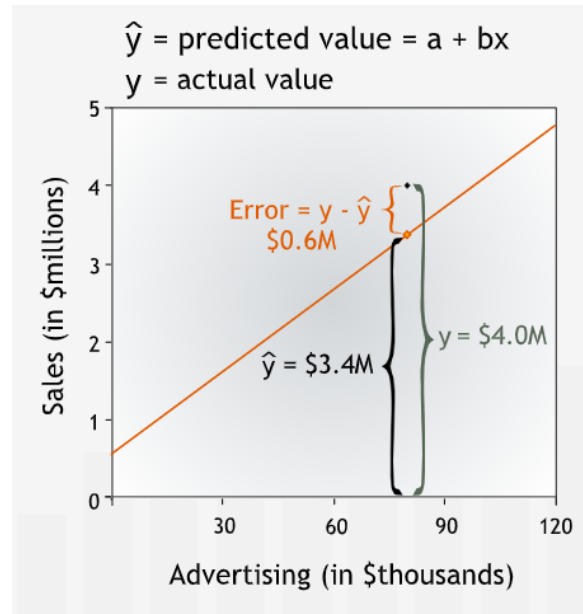




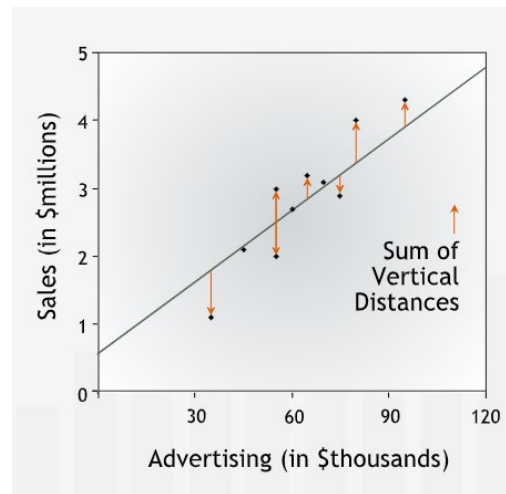
- To quantify how accurately a line fits a dataset, we measure the vertical distance between each data point and the line.
- We refer to this vertical distance between a data point and the line as the **error**. The error is the difference between the observed value and the line's prediction for our dependent variable.



We refer to the value of the dependent variable predicted by the line as  $\hat{y}$  and to the actual value of the dependent variable as  $y$ . Then the error is  $y - \hat{y}$ , the difference between the actual and predicted values of the dependent variable.



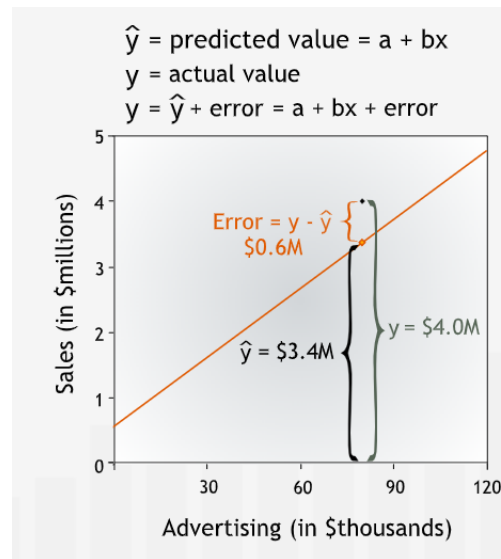
- Collectively, the errors in prediction for all the data points measure how accurately a line fits a set of data.
- To quantify the total size of the errors, we cannot just sum each of the vertical distances. If we did, positive and negative distances would cancel each other out.
- Instead, we take the square of each distance and then sum all the squares.



- The line that most accurately fits the data — the regression line — is the line for which the Sum of Squared Errors is minimized.

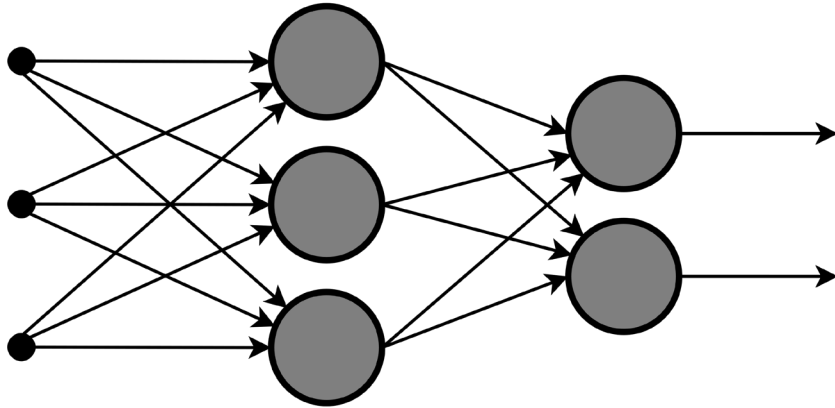
● Lower SSE ⇔ Higher Accuracy  
● Lowest SSE ⇔ Regression Line

The complete mathematical description of the relationship between the dependent and independent variables is  $y = bx + a + \text{error}$ .



# Linear Regression with One Variable

---



Cost function

# Error

Measures the difference between the observed (actual) and predicted values for a single training instance. This can be a simple subtraction (either predicted - observed or observed - predicted).

## Dataset Example:

Data Point	Observed(Actual)	Predicted
1	10	12
2	15	14
3	20	18

## Error Calculation

Data Point	Error (Predicted - Observed)
1	$12 - 10 = 2$
2	$14 - 15 = -1$
3	$18 - 20 = -2$

# Loss Function

Takes the error for a single training instance and applies additional properties to it (ex. squaring the error) .

Loss Function:  $L(y, \hat{y}) = (y - \hat{y})^2$  Where  $\hat{y} = bx + a$

Using the Squared Error as our loss function.

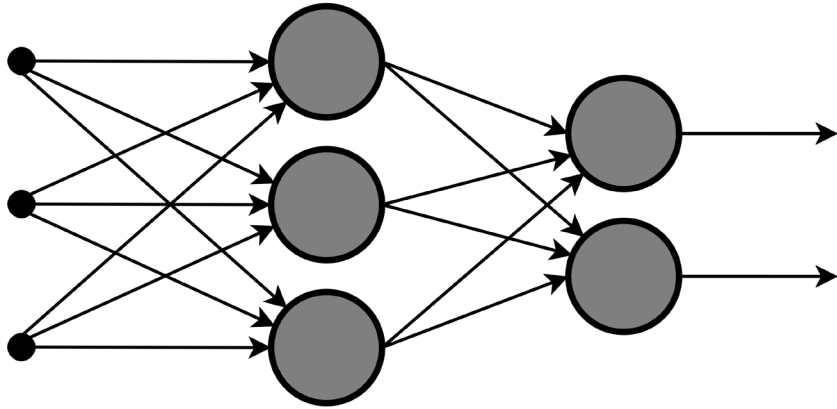
Data Point	Error	Squared Error (Loss)
1	2	$2^2 = 4$
2	-1	$(-1)^2 = 1$
3	-2	$(-2)^2 = 4$



# Cost Function

- Aggregates the loss over all training instances to provide a single measure of how well the model is performing across the entire training dataset.
- Often involves averaging the individual losses
- The cost function is what you typically aim to minimize in the training of the model.
- Cost Function  $J(w) = (4+1+4)/3 = 3$ .
- This value represents the average of the squared errors (losses) over all the data points in the dataset.

# Linear Regression



---

Normal equation  
and Gradient  
Descent

# Simplified Linear Regression

Simplified linear equation  $h(x) = wx + b$

Cost function:  $J(w) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$

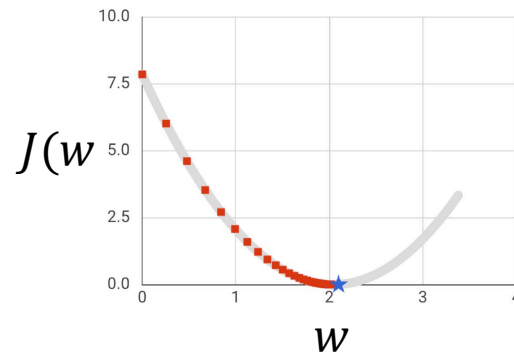
Cost function: MSE (Mean Squared Error) of the entire training set.

Need to minimize the cost function:  $J(w) = aw^2 + bw + c$

# Two ways to find minimum

1. Analytical Method

2. Gradient Descent

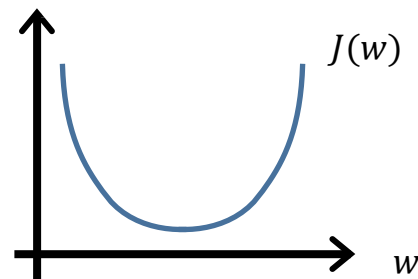


# Analytical Method

Intuition:  $h(x) = wx + b$

$$J(w) = aw^2 + bw + c$$

$$\frac{d}{dw}J(w) = 2aw + b$$



We can then set it to equal 0 and solve it!

$$J(b, w_1, w_2, \dots, w_{n_x}) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial b} J(b, w_1, w_2, \dots, w_{n_x}) = \dots = 0$$

$$\frac{\partial}{\partial w_j} J(b, w_1, w_2, \dots, w_{n_x}) = \dots = 0$$

We can then solve for  $b, w_1, w_2, \dots, w_{n_x}$

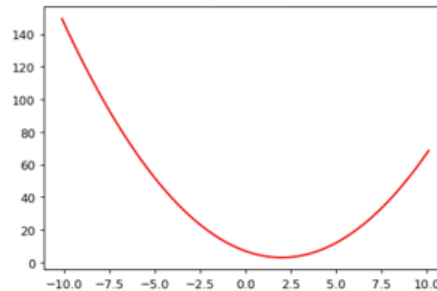
## Normal Equation - Example

$$f(x) = 3 + (x - 2)^2$$

$$\frac{\partial f(x)}{\partial x} = 2x - 4$$

$$\text{Set } 2x - 4 = 0$$

Minimum at  $x = 2$



$$f(x) = e^{-x} + x^2$$

$$0 = -e^{-x} + 2x$$

$$\frac{\partial f(x)}{\partial x} = -e^{-x} + 2x$$

How do you find  $x$ ?

# Strategy

Have some function:  $J(b, w)$

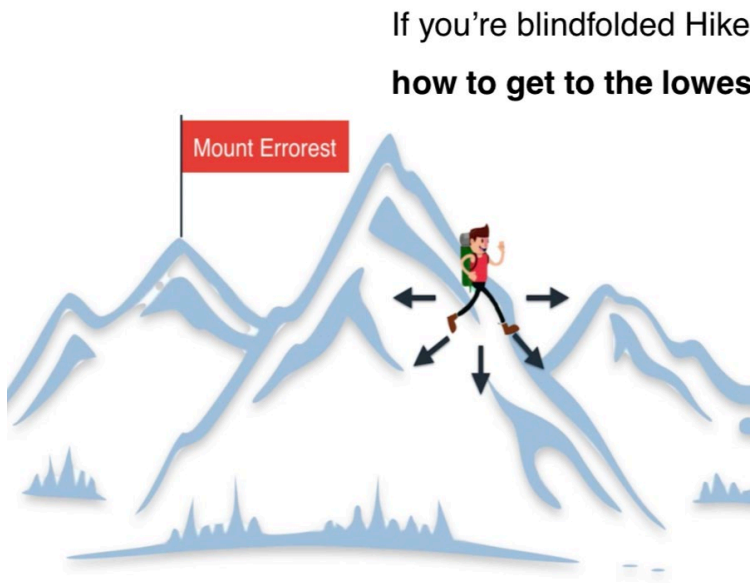
Want  $\operatorname{argmin}_{b,w} J(b, w)$

Outline

- Start with some random  $b, w$

- Keep changing  $b, w$  to reduce  $J(b, w)$  until we hopefully end up at a minimum

# Gradient Descent Intuition



A good strategy to get to the bottom of the valley quickly is to go downhill in the **direction of the steepest slope**.

This is exactly what GD does: it measures the local gradient of the cost function with regards to the parameters, and it goes in the direction of descending gradient. Once the gradient is zero, you have reached a minimum.



# What is a Derivative?

- A derivative is a mathematical tool that tells you the "slope" of a function at a specific point—basically, whether you're going uphill/downhill and how steeply.
- In one dimension, it's simply the derivative  $dy/dw$ .
- In multiple dimension, it's the partial derivative:
  - $\partial J(b, w) / \partial b$  represents the rate and direction of change of  $J$  with respect to  $b$
  - $\partial J(b, w) / \partial w$  is the rate and direction of change with respect to  $w$ .

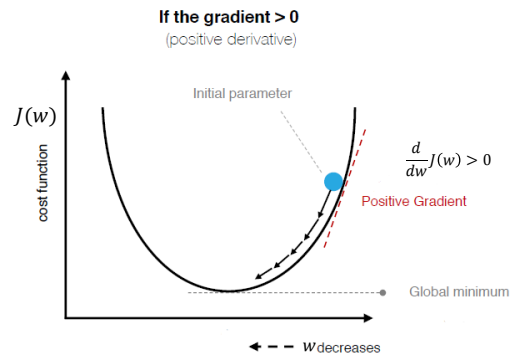
# The Gradient Descent Update Equations

- $b = b - \alpha * (\partial J(b, w) / \partial b)$
- $w = w - \alpha * (\partial J(b, w) / \partial w)$
- These equations tell us how to adjust  $b$  and  $w$  to minimize  $J$ .
- $\alpha$  is the learning rate — how big of a "step" to take in each update.

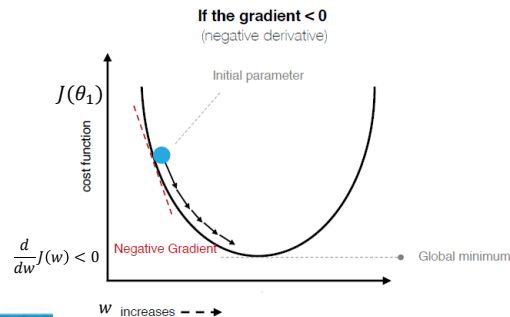
# Intuition Behind Update Equations

Simplified:  $h(x) = wx$

$$J(w) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$



$$w = w - \alpha \boxed{\frac{d}{dw}J(w)} > 0$$



$$w = w - \alpha \boxed{\frac{d}{dw}J(w)} < 0$$

# The Steepness and Sign of the Slope

High Magnitude: A larger absolute value of the partial derivative indicates a steeper slope. This means we are far from the minimum, and a larger step could be beneficial.

Low Magnitude: A smaller absolute value suggests that we are closer to a flat area, possibly near the minimum. In this case, taking smaller steps can help fine-tune the optimization.

The update equations  $b = b - \alpha * (\partial/\partial b J(b, w))$  and  $w = w - \alpha * (\partial/\partial w J(b, w))$  combine the sign and magnitude to adjust  $b$  and  $w$  in a way that moves us closer to the minimum of  $J$ .

The learning rate  $\alpha$  scales these steps, allowing us to control the speed of convergence.

# Gradient descent algorithm

Repeat until convergence {

$$b = b - \alpha \frac{\partial}{\partial b} J(b, w)$$

$$w = w - \alpha \frac{\partial}{\partial w} J(b, w)$$

Simultaneously update  $b$  and  $w$

}

Learning rate      Partial derivative



# Gradient descent algorithm

Correct: Simultaneous update

$$\begin{aligned}temp0 &= b - \alpha \frac{\partial}{\partial b} J(b, w) \\temp1 &= w - \alpha \frac{\partial}{\partial w} J(b, w) \\b &= temp0 \\w &= temp1\end{aligned}$$

Incorrect:

$$\begin{aligned}b &= b - \alpha \frac{\partial}{\partial b} J(b, w) \\w &= w - \alpha \frac{\partial}{\partial w} J(b, w)\end{aligned}$$

# Gradient descent Summary

## Linear Regression Model

Hypothesis:

$$h(x) = b + wx$$

Parameters:

$$b, w$$

Cost function:

$$\begin{aligned} J(b, w) \\ = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 \end{aligned}$$

Goal:

$$\operatorname{argmin}_{b, w} J(b, w)$$

## Gradient descent algorithm

Repeat until convergence {

$$b = b$$

$$- \alpha \frac{\partial}{\partial b} J(b, w)$$

$$w = w$$

$$- \alpha \frac{\partial}{\partial w} J(b, w)$$

}

*Simultaneously update  $b$  and  $w$*

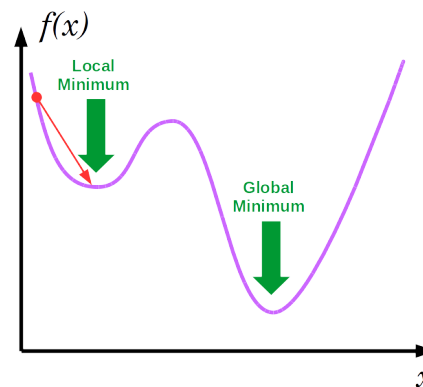
# Pros and Cons of Gradient Descent

## Advantages:

- Quite effective on ML tasks
- Often very scalable

## Drawbacks

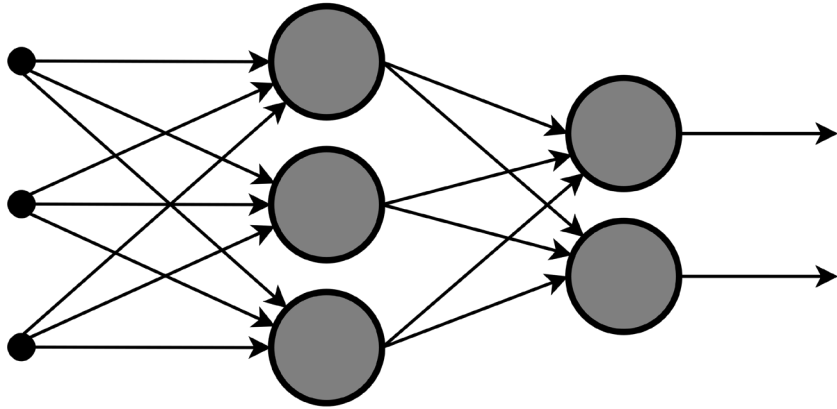
- Might find a local minimum
- Only applies to differentiable function.





# Linear Regression with One Variable

---



## Learning Rate

# Learning Rate ( $\alpha$ )

Gradient descent algorithm:

Repeat until convergence {

$$b = b - \alpha \frac{\partial}{\partial b} J(b, w)$$

$$w = w - \alpha \frac{\partial}{\partial w} J(b, w)$$

}

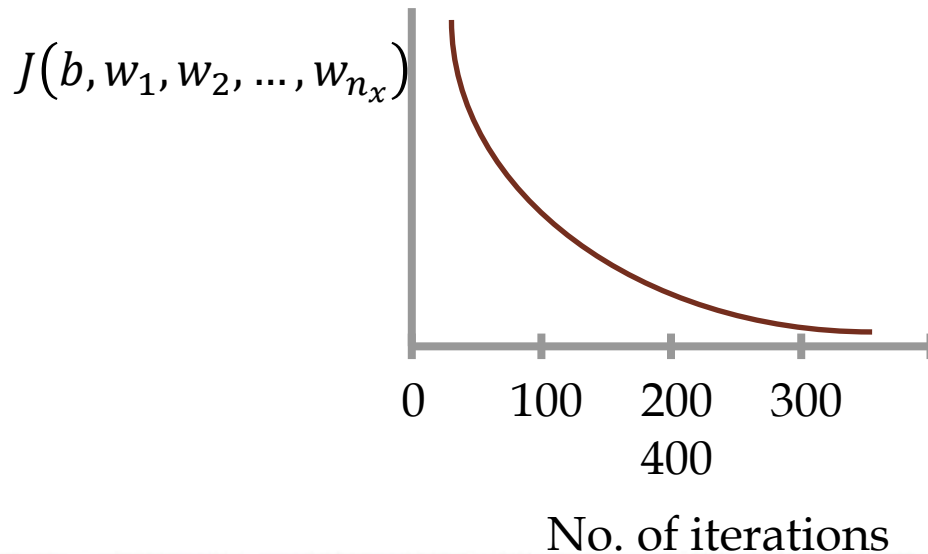
- $\alpha$  controls the step size in gradient descent.
- Too small: slow convergence.
- Too large: risk of overshooting or divergence.
- Finding the right  $\alpha$  is crucial for efficient and accurate optimization.

# Analogy

- Imagine you are in a valley, blindfolded, and you are trying to find the lowest point.
- A small  $a$  is like taking small, careful steps. It will take you a long time to get to the bottom, but you are less likely to walk past it.
- A large  $a$  is like taking large, careless steps. You'll get to the bottom quickly, but you might step over it and start climbing up the other side of the valley.

# Making sure gradient descent is working correctly

$J(b, w_1, w_2, \dots, w_{n_x})$  should decrease  
after every iteration!



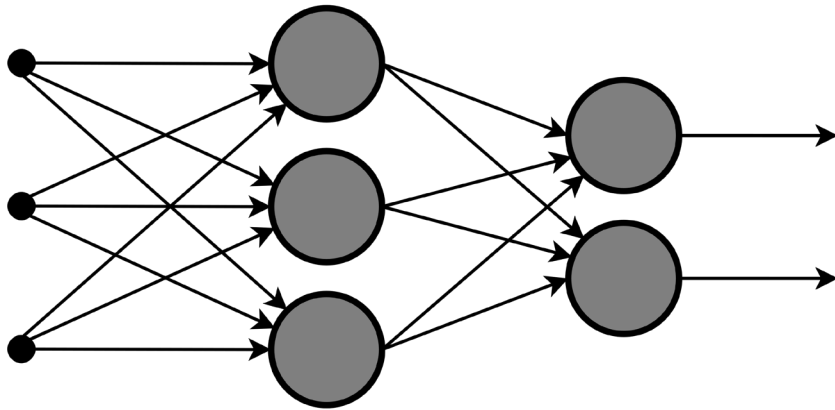
Example:

Declare convergence if cost  
function decreases by less  
than  $10^{-3}$  in one iteration.

# Finding the Right $\alpha$

- Empirical Testing: Try several values of  $\alpha$  and use cross-validation to find the most effective one.
- Adaptive Learning Rate: Some algorithms adjust  $\alpha$  dynamically based on how quickly the loss is decreasing.
- Momentum: Use a weighted average of past gradients to smooth out updates.

# Regression



---

Impact of Learning  
Rate on Convergence

## Learning Rate Impact

- Assume we have the following cost function:  $J(w) = (w - 4)^2$
- Use Analytical solution: we can find the minimum for this function by setting the derivative to 0. In this case,  $2(w-4) = 0 \Rightarrow w = 4$ .
- Using Gradient Descent: We will first calculate the new values for  $w$  assuming we don't have a learning rate and then compute with the learning rate.
- Gradient of Cost Function:  $dJ(w)/dw = 2(w - 4)$

## Updating w - No Learning Rate Scenario

Iteration 1: Starting Point:  $w = 0$

Gradient:  $dJ(w)/dw = 2(0 - 4) = -8$

Update Rule:  $w = w - \alpha * (-8) = 0 + 8 = 8$

Iteration 2: Starting Point:  $w = 8$

Gradient:  $dJ(w)/dw = 2(8 - 4) = 8$

Update Rule:  $w = w - \alpha * 8 = 8 - 8 = 0$

Iteration 3: Starting Point:  $w = 0$

Gradient:  $dJ(w)/dw = 2(0 - 4) = -8$

Update Rule:  $w = w - \alpha * (-8) = 0 + 8 = 8$



# Updating - Learning Rate Scenario

Learning Rate ( $\alpha$ ):  $\alpha = 0.1$  Starting point:  $w = 0$

Iteration 1: Gradient:  $dJ(w)/dw = 2(0 - 4) = -8$

Update Rule:  $w = w - \alpha * (-8) = 0 + 0.1 * 8 = 0.8$

Iteration 2: Gradient:  $dJ(w)/dw = 2(0.8 - 4) = -6.4$

Update Rule:  $w = w - \alpha * (-6.4) = 0.8 + 0.1 * 6.4 = 1.44$

Iteration 3: Gradient:  $dJ(w)/dw = 2(1.44 - 4) = -5.12$

Update Rule:  $w = w - \alpha * (-5.12) = 1.44 + 0.1 * 5.12 = 1.944$

Iteration 4: Gradient:  $dJ(w)/dw = 2(1.944 - 4) = -4.112$

Update Rule:  $w = w - \alpha * (-4.112) = 1.944 + 0.1 * 4.112 = 2.3552$

Iteration 5: ?

# Learning Rate - Sample Code

File: learning\_rate\_importance\_demo.ipynb

This program demonstrates how to find the minimum of the cost function  $J(w) = (w - 4)^2$  using the gradient descent algorithm in a simple linear regression problem. It examines the impact of two different learning rates on the speed and accuracy of convergence.

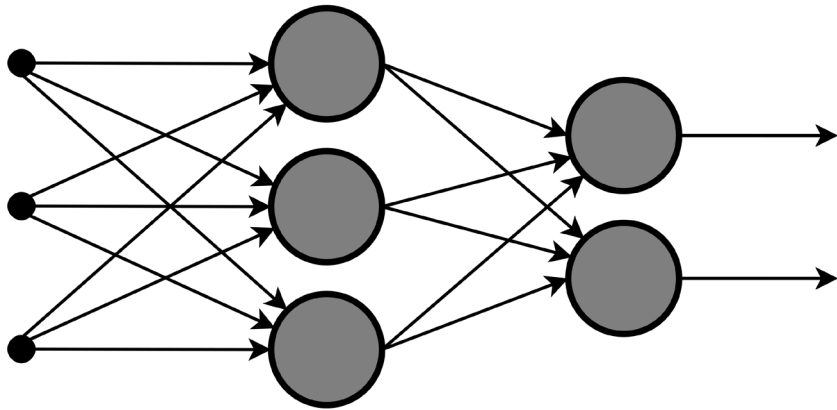
Learning Rates:

High Learning Rate (1.0):

Runs the algorithm with a learning rate of 1.0. In this case, the algorithm overshoots the minimum, leading to oscillation and failure to converge.

Low Learning Rate (0.1):

Runs the algorithm with a learning rate of 0.1. The algorithm gradually converges to the minimum at  $w = 4$ .



# Logistic Regression

---

Binary  
classification

# What is Classification?

- The linear regression model discussed earlier assumes that the response variable  $Y$  is **quantitative**. But in many situations, the response variable is instead qualitative. For example, eye color is **qualitative**. Often qualitative variables are referred to as **categorical**; we will use these qualitative terms interchangeably. The study for predicting qualitative responses is known as classification.
- Predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class. On the other hand, often the methods used for classification first predict the probability that the observation belongs to each of the categories of a qualitative variable, as the basis for making the classification.
- In this sense they also behave like regression methods.

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

# Examples

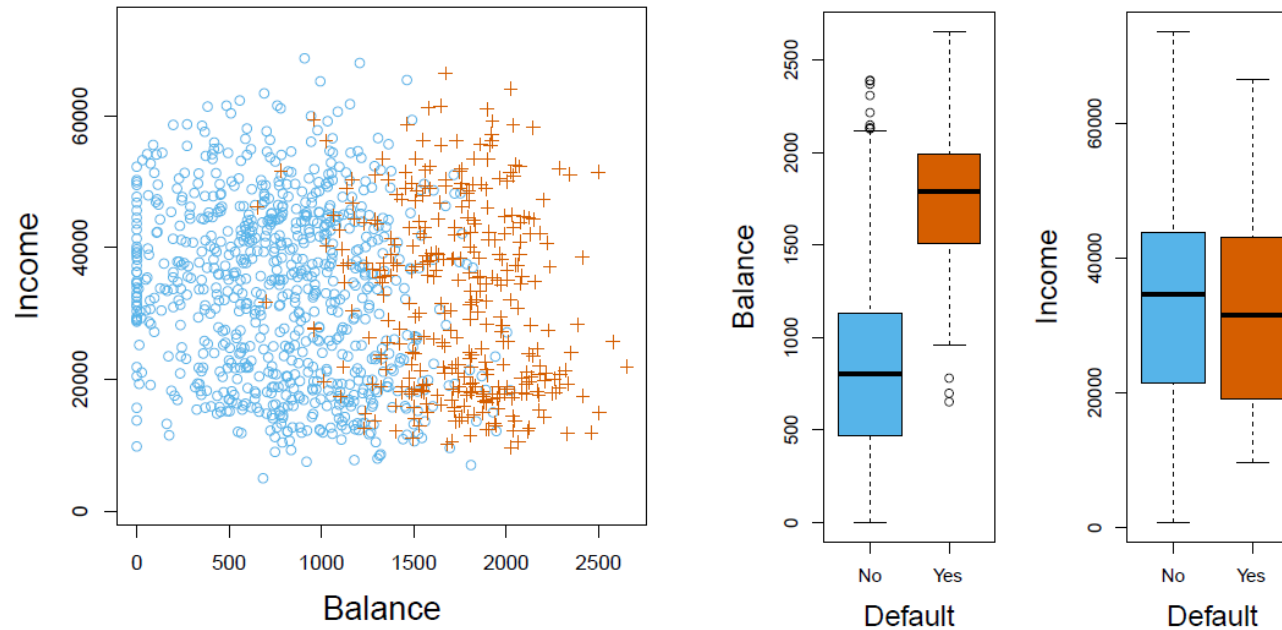
- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.

Just as in the regression setting, in the classification setting we have a set of training observations  $(x_1, y_1), \dots, (x_n, y_n)$  that we can use to build a classifier. We want our classifier to perform well not only on the training data, but also on test observations that were not used to train the classifier.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Examples



Predict whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Classification Error

Suppose that we seek to estimate  $f$  on the basis of training observations  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ , where now  $y_1, \dots, y_n$  are qualitative. The most common approach for quantifying the accuracy of our estimate  $\hat{f}$  is the training *error rate*, the proportion of mistakes that are made if we apply our estimate  $\hat{f}$  to the training observations

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i). \quad \text{i.e. The fraction of incorrect classifications}$$

Here  $\hat{y}_i$  is the predicted class label for the  $i$ th observation using  $\hat{f}$ . And  $I(y_i \neq \hat{y}_i)$  is an *indicator variable* that equals 1 if  $y_i \neq \hat{y}_i$  and zero if  $y_i = \hat{y}_i$ . If  $I(y_i \neq \hat{y}_i) = 0$  then the  $i$ th observation was classified correctly by our classification method; otherwise it was misclassified

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

# Classification Error

## Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

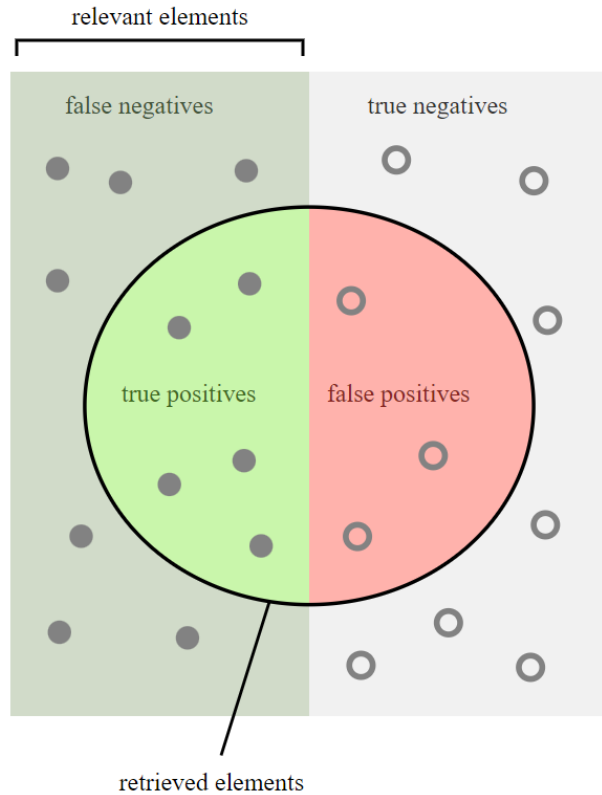
$$F1 \text{ score} = \frac{2 * (precision * recall)}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

Src:<https://medium.com/swlh/how-to-remember-all-these-classification-concepts-forever-761c065be33>

© Walid Hassan, M.B.A, D.Eng.



# Classification Error



How many retrieved items are relevant?

Precision =



How many relevant items are retrieved?

Recall =



Src: [https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall)

© Walid Hassan, M.B.A, D.Eng.

# Classification Error

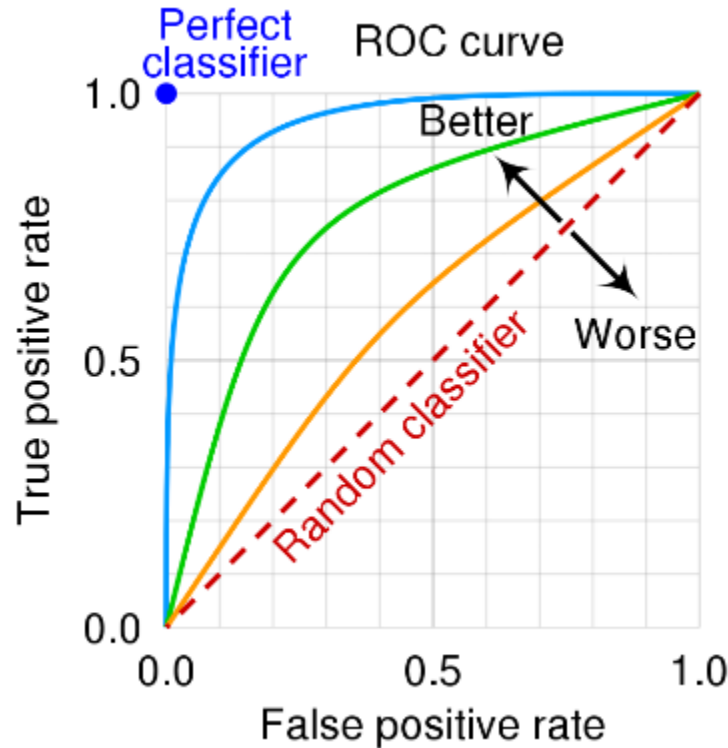
- **Accuracy** is the proportion of the total number of correct predictions.
- **Precision** is the ratio of the total number of correctly classified positive classification and the total number of predicted positive classification. aka: **positive predictive value** (Triple-P rule).
- **Recall (Sensitivity)**, also known as the sensitivity, or the **true positive rate (TPR)**, is the proportion of the total number of actual positives that were identified correctly.
- **Specificity** is the proportion of the total number of the actual negative that were identified correctly. **true negative rate (TNR)**
- When we increase the recall rate by adjusting the classification threshold of a model, the precision rate is decreased.
- high precision and high recall are what every model optimizes for. Consider the tradeoff, there is a balancing metric called **F1 score** that combines the two terms

Src:<https://medium.com/swlh/how-to-remember-all-these-classification-concepts-forever-761c065be33>

# Classification Error

- ROC : receiver operating characteristic curve, or ROC curve

True Positive Rate  
=  $(TP / (TP + FN))$  =  
Sensitivity/Recall



FPR: The proportion of actual negative cases that the model incorrectly predicts as positive

False Positive Rate  
=  $(FP / (Negatives))$   
=  $FP / (TN + FP)$   
1- TNR

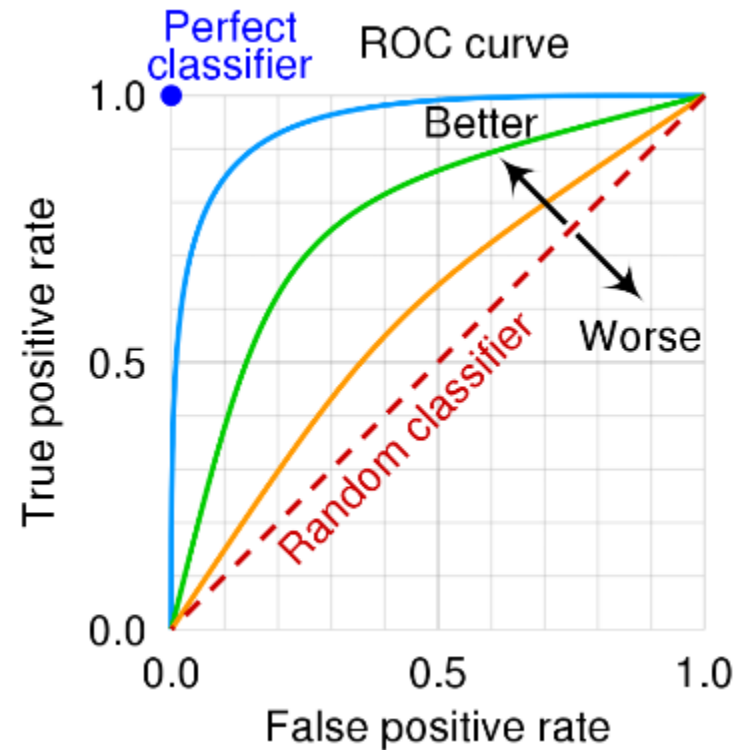
Src:<https://medium.com/swlh/how-to-remember-all-these-classification-concepts-forever-761c065be33>

© Walid Hassan, M.B.A, D.Eng.

# Classification Error

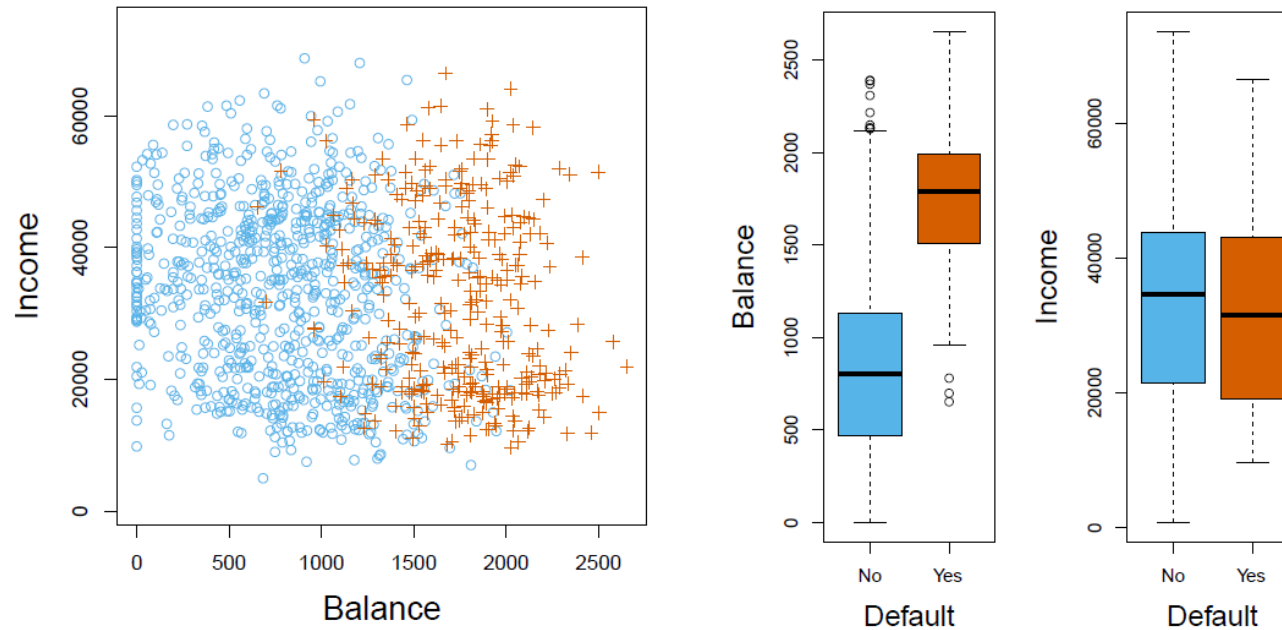
- **ROC** : receiver operating characteristic curve, or ROC curve:
  - Use it to find out the best threshold within the same algorithm
- **Area Under the Curve**: Use it to compare different algorithms

The perfect classifier is the one with coordinates 0,1 because we want all True positive rate to be 1, and the false positive rate to be zero



Src:<https://medium.com/swlh/how-to-remember-all-these-classification-concepts-forever-761c065be33>

# Examples



Predict whether an individual will default on his or her credit card payment, on the basis of annual income and monthly credit card balance.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Why not Linear Regression?

Suppose that we are trying to predict the medical condition of a patient in the emergency room on the basis of her symptoms. In this simplified example, there are three possible diagnoses: stroke, drug overdose, and epileptic seizure.

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Why not Linear Regression?

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

Using this coding, least squares could be used to fit a linear regression model to predict  $Y$  on the basis of a set of predictors  $X_1, \dots, X_p$ . Unfortunately, this coding implies an ordering on the outcomes, putting drug overdose in between stroke and epileptic seizure, and insisting that the difference between stroke and drug overdose is the same as the difference between drug overdose and epileptic seizure. In practice there is no particular reason that this needs to be the case.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Why not Linear Regression?

If the response variable's values did take on a natural ordering, such as mild, moderate, and severe, and we felt the gap between mild and moderate was similar to the gap between moderate and severe, then a 1, 2, 3 coding would be reasonable. Unfortunately, in general there is no natural way to convert a qualitative response variable with more than two levels into a quantitative response that is ready for linear regression.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.



# Binary Classification ~ Linear Regression

Assume there are only two possibilities for the patient's medical binary condition: stroke and drug overdose

$$Y = \begin{cases} 0 & \text{if stroke;} \\ 1 & \text{if drug overdose.} \end{cases}$$

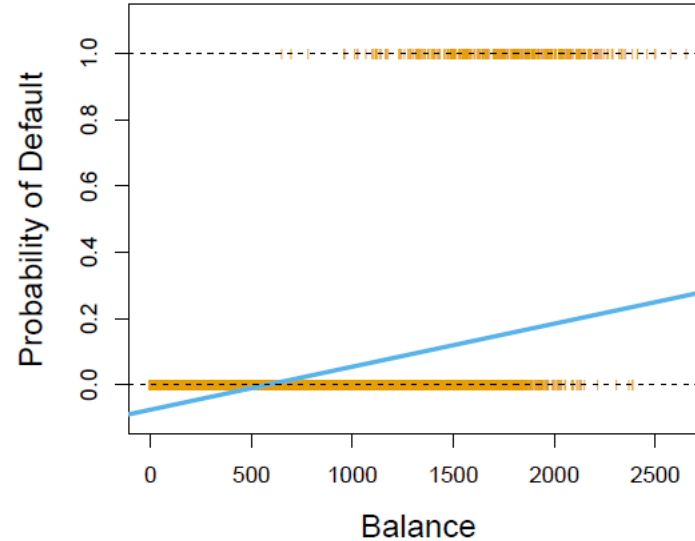
We could then fit a linear regression to this binary response, and predict drug overdose if  $\hat{Y} > 0.5$  and stroke otherwise. In the binary case even if we flip the above coding, linear regression will produce the same final predictions.

In this binary case, regression by least squares is not completely unreasonable: it can be shown that the  $\hat{X}$  obtained using linear regression is in fact an estimate of  $\Pr(\text{drug overdose} | X)$  in this special case. However, if we use linear regression, some of our estimates might be outside the  $[0, 1]$  interval!

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Binary Classification ~ Linear Regression



Probability of Default using Linear Regression. What Shortcomings do you see???

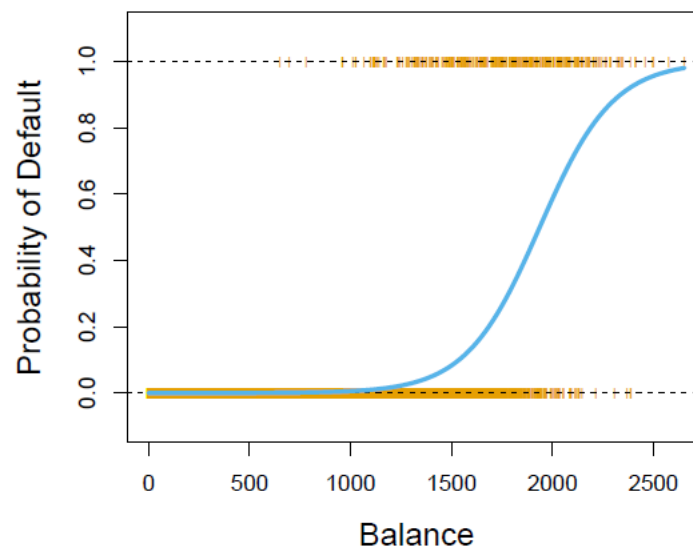
Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Logistic Regression

Rather than modeling this response  $Y$  directly, logistic regression models the probability that  $Y$  belongs to a particular category

$\Pr(\text{default} = \text{Yes}|\text{balance})$ . The values of  $\Pr(\text{default} = \text{Yes}|\text{balance})$ , which we abbreviate  $p(\text{balance})$ , will range between 0 and 1. Then for any given value of balance, a prediction can be made for default. This will be used rather than using the linear regression model to represent these probabilities.



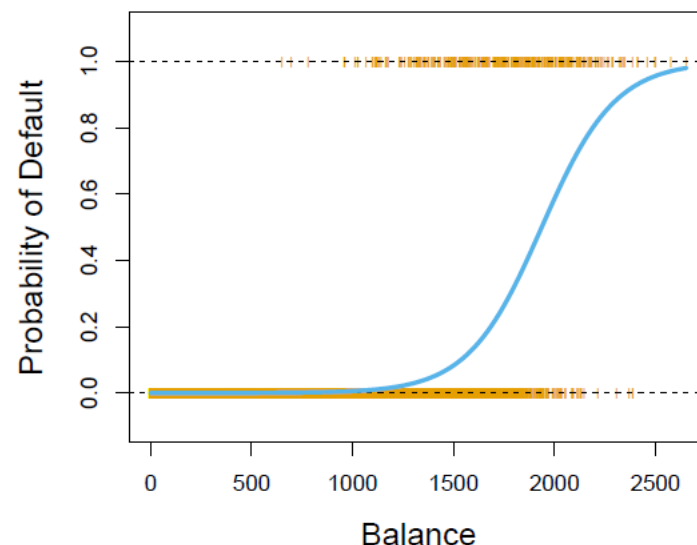
Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

# Logistic Regression

We use the logistic function defined as :

we now predict the probability of default as close to, but never below, zero. Likewise, for high balances we predict a default probability close to, but never above, one. The logistic function will always produce an S-shaped curve of this form, and so regardless of the value of  $X$ , we will obtain a sensible prediction.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$



Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.

# Logistic Regression

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X} \quad \left. \vphantom{\frac{p(X)}{1 - p(X)}} \right\} \text{Odds}$$

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X. \quad \left. \vphantom{\log \left( \frac{p(X)}{1 - p(X)} \right)} \right\} \text{Logit Function}$$

The quantity  $p(X)/[1-p(X)]$  is called the odds, and can take on any value between odds 0 and  $\infty$ . Values of the odds close to 0 and  $\infty$  indicate very low and very high probabilities of default (in this example).

We use a method called maximum likelihood for optimization (not least squares as in LR).

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

# Maximum Likelihood Function

we seek estimates for  $\beta_0$  and  $\beta_1$  such that the predicted probability  $\hat{p}(x_i)$  of default for each individual, corresponds as closely as possible to the individual's observed default status. In other words, we try to find  $\beta_0$  and  $\beta_1$  such that plugging these estimates into the model for  $p(X)$ , yields a number close to one for all individuals who defaulted, and a number close to zero for all individuals who did not.

This intuition can be formalized using a mathematical equation called a **likelihood function**.

$$\ell(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'})).$$

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

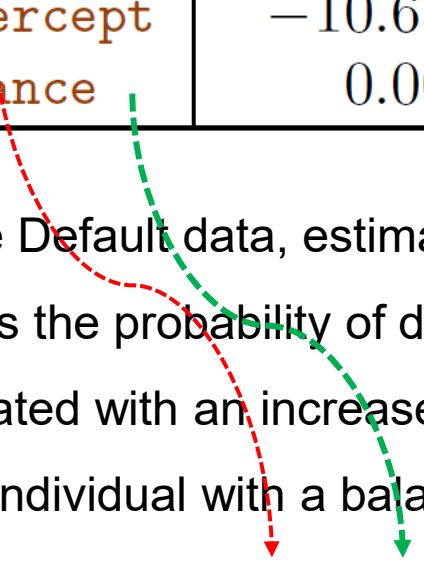
© Walid Hassan, M.B.A, D.Eng.

# Logistic Regression - Example

	Coefficient	Std. error	z-statistic	p-value
Intercept	-10.6513	0.3612	-29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

For the Default data, estimated coefficients of the logistic regression model that predicts the probability of default using balance. A one-unit increase in balance is associated with an increase in the log odds of default by 0.0055 units.

for an individual with a balance of \$1, 000 is < 1% vs. for 2000\$ balance it is 58.6%


$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1,000}}{1 + e^{-10.6513 + 0.0055 \times 1,000}} = 0.00576$$

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

# Multiple Logistic Regression

Predicting a binary response using multiple predictors: by analogy with the extension from simple to multiple linear regression

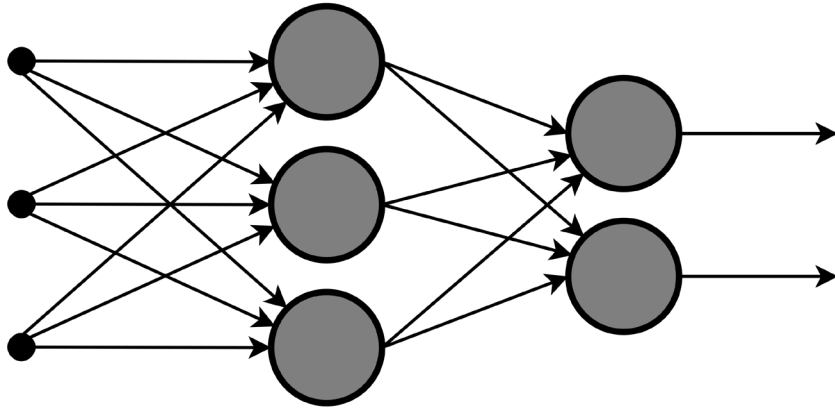
$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}.$$

Src: Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

© Walid Hassan, M.B.A, D.Eng.





# Logistic Regression

---

Binary  
classification

# Binary classification

**Binary Classification:** The process of classifying the elements of a dataset into one of two groups.

## Example:

Email: Spam / Not Spam?

Online Transactions: Fraudulent (Yes / No)?

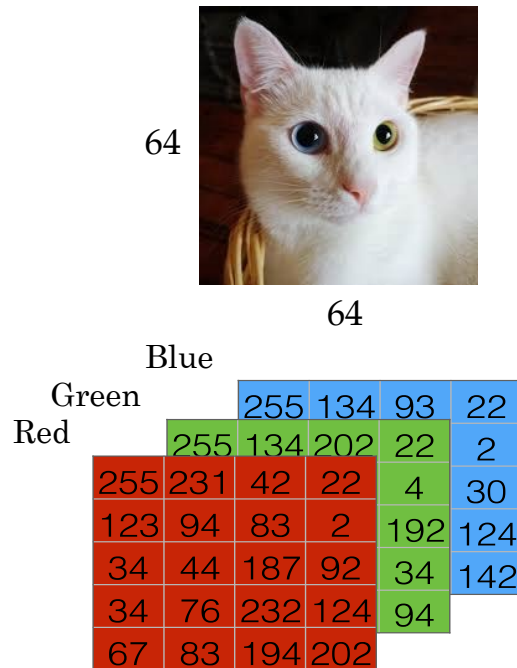
Tumor: Malignant / Benign ?

Image Recognition: Cat / Non-cat?

$y \in \{0, 1\}$       0: “Negative Class” (e.g., Non-cat)  
                             1: “Positive Class” (e.g., Cat)

# Binary classification

Image Recognition: Cat / Non-cat?



$y \in \{0, 1\}$

0: "Negative Class" (e.g., Non-cat)

1: "Positive Class" (e.g., Cat)

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ 22 \\ \dots \\ 255 \\ 134 \\ \dots \end{bmatrix}$$

$$n_x = 64 \times 64 \times 3 = 12288$$

# Binary classification

Tumor: Malignant / Benign ?

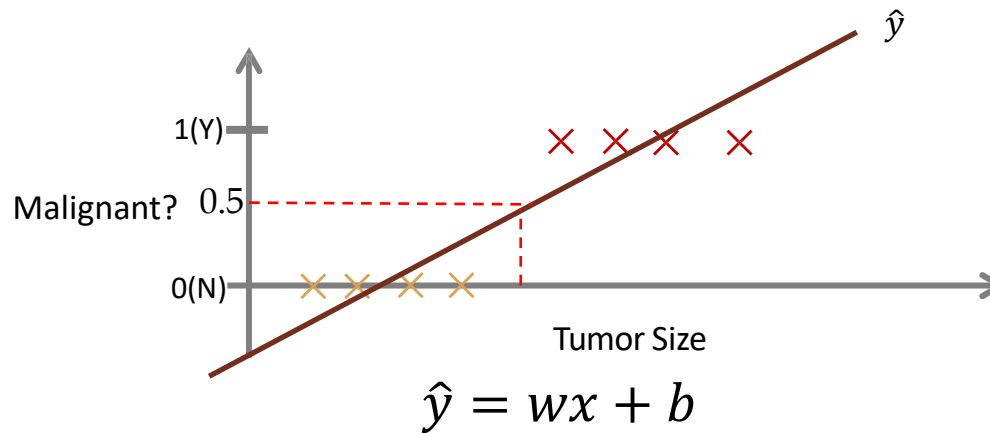
Input: size of tumor

$y \in \{0, 1\}$

0: “Negative Class” (e.g., non-Malignant)

1: “Positive Class” (e.g., Malignant)

## Initial solution – Linear Classification



Threshold classifier output  $\hat{y}$  at 0.5

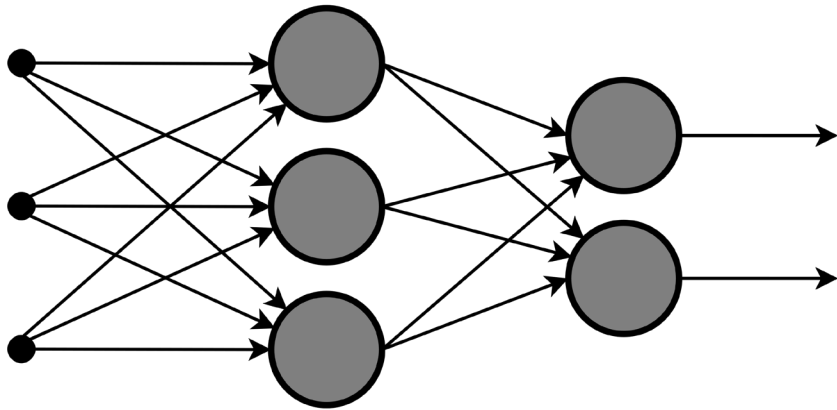
If  $\hat{y} \geq 0.5$ , predict 1

If  $\hat{y} < 0.5$ , predict 0

# Problem of using linear regression for classification

**1.Probabilistic Interpretation:** Linear regression does not provide probabilities for class memberships. Classification often benefits from knowing the certainty of predictions.

**2.Non-linear Decision Boundaries:** Linear regression assumes a linear relationship between the input variables and the output, which is often not the case in classification problems where the decision boundary between classes could be non-linear.



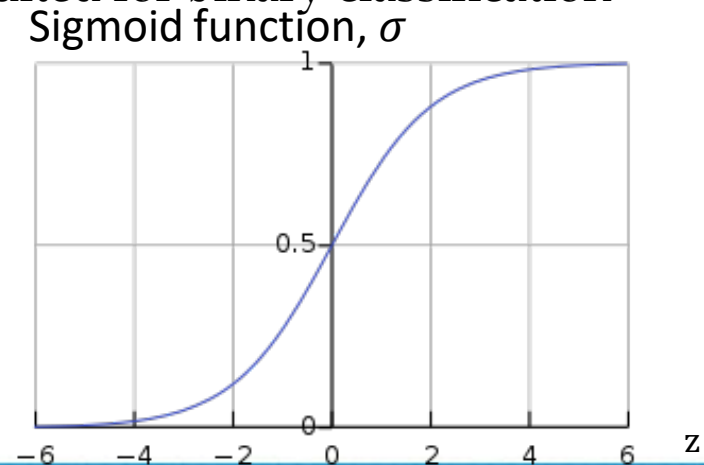
# Logistic Regression

---

## Hypothesis Representation

# Logistic regression model

- This model predicts the probability that a given input belongs to a particular class.
- It uses the logistic (or sigmoid) function to squeeze the output of a linear equation into the  $[0, 1]$  range.
- Logistic regression also has a different cost function, known as the log loss or binary cross-entropy, which is better suited for binary classification problems.
- In classification:  $y = 0 \text{ or } 1$
- Logistic Regression:  $0 \leq \hat{y} \leq 1$





# Logistic regression

$$z = wx + b \quad \hat{y} = \sigma(z) = \sigma(wx + b)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Suppose predict 1 if  $\hat{y} \geq 0.5$

$$wx + b \geq 0$$

$$\sigma(z) \geq 0.5$$

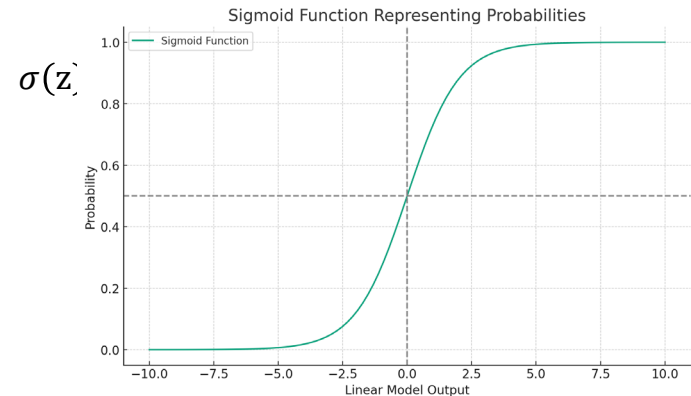
When  $z \geq 0$

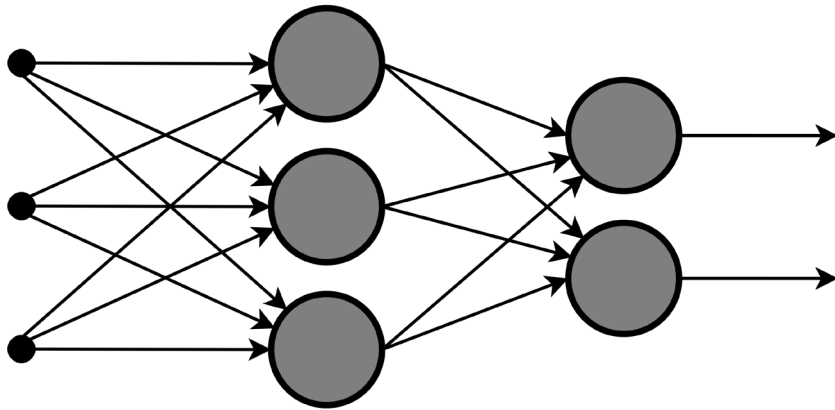
Suppose predict 0 if  $\hat{y} < 0.5$

$$wx + b < 0$$

$$\sigma(z) < 0.5$$

When  $z < 0$





# Logistic Regression

---

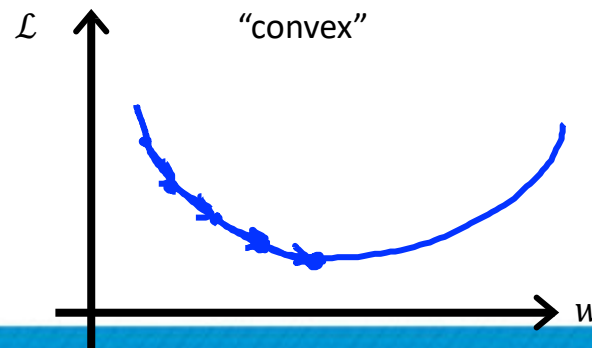
## Cost Function

## Cost function review

Cost function of linear regression:  $J(b, w) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (\hat{y}^{(i)} - y^{(i)})^2$

Linear regression is using squared loss function:

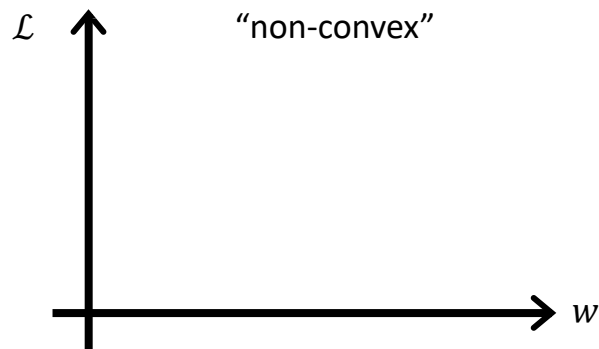
$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2} (\hat{y}^{(i)} - y^{(i)})^2$$



## Can we use the same loss in logistic?

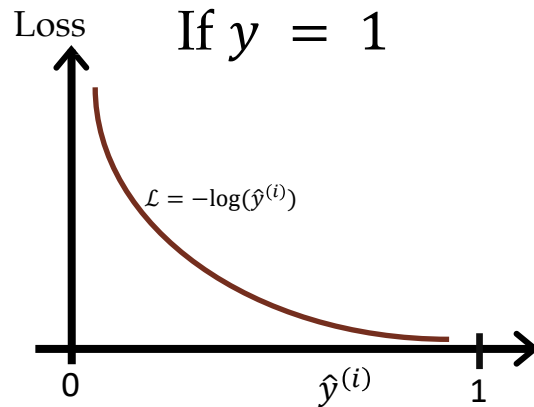
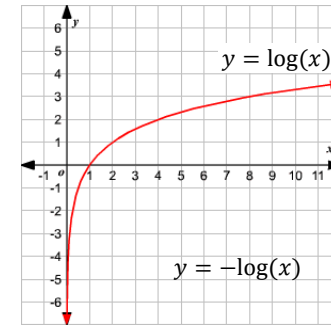
If logistic regression were to use squared loss function:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2} (\hat{y}^{(i)} - y^{(i)})^2$$



# Logistic regression loss function (1)

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}), & y = 1 \end{cases}$$



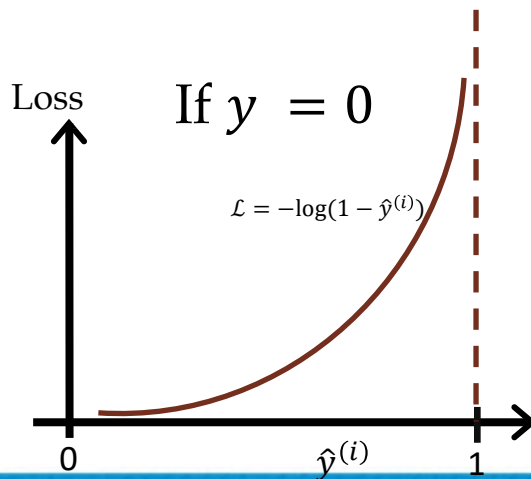
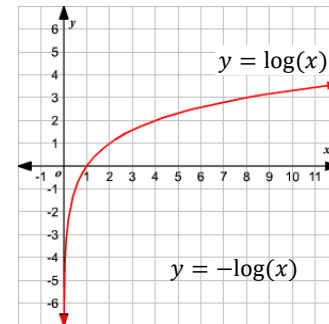
Loss = 0 if  $y = 1$  and  $\hat{y} = 1$

But Loss  $\rightarrow +\infty$  if  $y = 1$  and  $\hat{y} \rightarrow 0$

The intuition here is that if  $\hat{y} = 0$ , but  $y = 1$   
We will penalize learning algorithm by a very large loss.

## Logistic regression loss function (2)

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(1 - \hat{y}^{(i)}), & y = 0 \end{cases}$$



Loss = 0 if  $y = 0$  and  $\hat{y} = 0$

But Loss  $\rightarrow +\infty$  if  $y = 0$  and  $\hat{y} \rightarrow 1$

The intuition here is that if  $\hat{y} = 1$ , but  $y = 0$   
We will penalize learning algorithm by a very large loss.

# Logistic regression loss function

Overall cost function of logistic regression:

Loss function: 
$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}), & y = 1 \\ -\log(1 - \hat{y}^{(i)}), & y = 0 \end{cases}$$

Compact version of logistic loss function: **Binary cross-entropy loss**

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} * \log(\hat{y}^{(i)}) - (1 - y^{(i)}) * \log(1 - \hat{y}^{(i)})$$

<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>

# Logistic regression loss function

Overall cost function of logistic regression:

$$J(b, w) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

To fit parameters,  $b$  and  $w$   $\underset{b, w}{\operatorname{argmin}} J(b, w)$

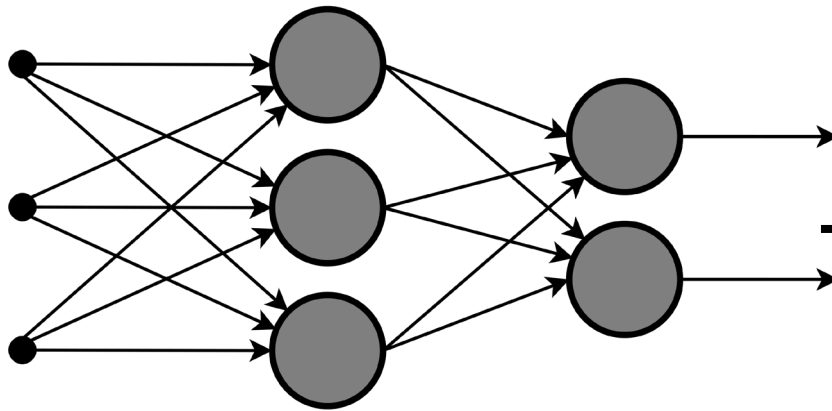


# References

In addition to the references in each slide, the following are leveraged throughout this lecture:

DeepLearning.AI(<https://www.deeplearning.ai/>)

# Backup



# Linear Regression with Multiple Variables

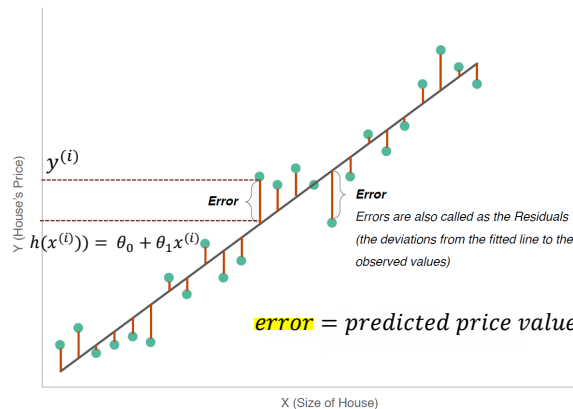
---

Gradient descent  
for multiple  
variables

# Review cost function for univariate Linear Regression

Idea: Choose  $b$  and  $w$  so that  $h(x^{(i)})$  is close to  $y^{(i)}$  for our  $m$  training examples  $(x^{(i)}, y^{(i)})$

But what do we mean by “close?”



$$J(b, w) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

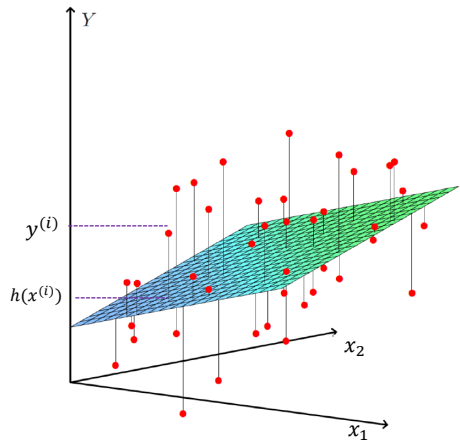
An arrow points from the term  $(h(x^{(i)}) - y^{(i)})$  in the equation to the text 'error' in the following block.

$$h(x) = b + wx$$

Cost function: MSE (Mean Squared Error)  
Minimize the cost function!!!

# Cost function for multivariate L.R.

We want the hyper-plane that “best fits” the training examples (red points).  
In other words, minimize the sum of squared residuals (error).



$$h(x) = b + w_1x_1 + w_2x_2$$

$$J(b, w_1, w_2) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

# Derivatives for a Linear Cost Function

$$J(b, w) = \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial b} J(b, w) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (b + wx^{(i)} - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})$$

$$\frac{\partial}{\partial w} J(b, w) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (b + wx^{(i)} - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

# Gradient descent algorithm

Gradient descent for univariate L.R.

Repeat until convergence {

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})$$

$$w = w$$

$$- \alpha \frac{1}{m} \sum_{i=1}^m [(h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}]$$

}

Gradient descent for multivariate L.R.

Repeat until convergence {

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})$$

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m [(h(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}]$$

$$w_2 = w_2 - \alpha \frac{1}{m} \sum_{i=1}^m [(h(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}]$$

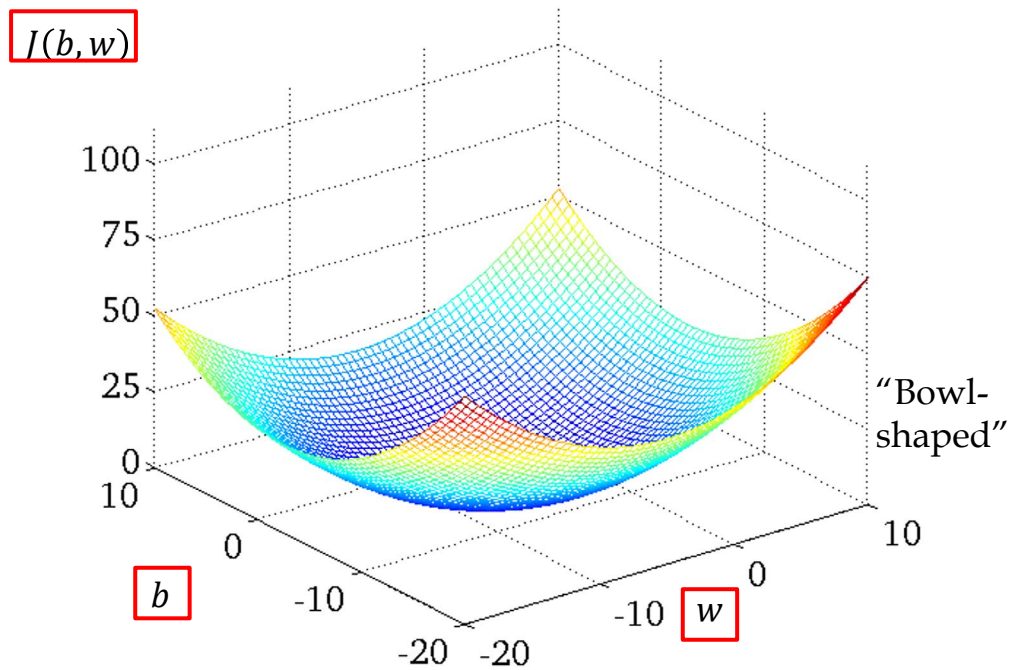
$$w_3 = w_3 - \alpha \frac{1}{m} \sum_{i=1}^m [(h(x^{(i)}) - y^{(i)}) \cdot x_3^{(i)}]$$

...

$$w_{n_x} = w_{n_x} - \alpha \frac{1}{m} \sum_{i=1}^m [(h(x^{(i)}) - y^{(i)}) \cdot x_{n_x}^{(i)}]$$

}

# A slice of cost function



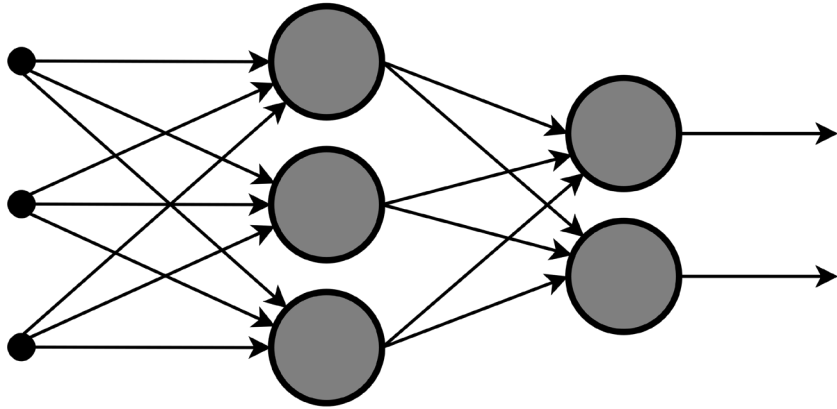
<https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/ways-to-represent-multivariable-functions/a/contour-maps>

© Walid Hassan, M.B.A, D.Eng.



# Linear Regression with One Variable

---



## Derivatives and Chain Rule

# Understanding the Chain Rule

## 1. What is a Composite Function?

1. A function made up of two or more functions.
2. Example:  $g(f(x))$ , where  $g(x)$  and  $f(x)$  are both functions.

## 2. Why Use the Chain Rule?

1. Essential for finding the derivative of a composite function.

## 3. The Chain Rule Formula:

1.  $(d/dx)[g(f(x))] = g'(f(x)) * f'(x)$
2.  $g'(f(x))$  is the derivative of  $g(x)$  evaluated at  $f(x)$
3.  $f'(x)$  is the derivative of  $f(x)$

# Understanding the Chain Rule

1.Example:

1.If  $g(x) = x^2$  and  $f(x) = 3x + 2$

2.Then  $g(f(x)) = (3x + 2)^2$

3.Using the Chain Rule:  $(d/dx)(g(f(x))) = 2 * (3x + 2) * 3 = 6 * (3x + 2)$

# Loss Function and Convexity

Squared Error Loss Function is convex with respect to  $w$  and has a minimum point:

Loss Function:  $L(y, \hat{y}) = (y - \hat{y})^2$  Where  $\hat{y} = wx$

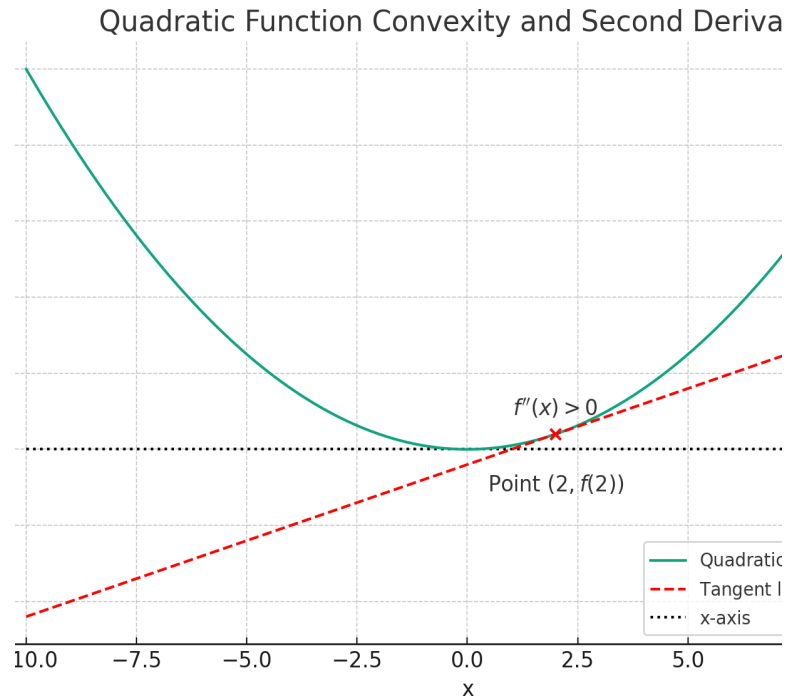
First Derivative with respect to  $w$ :

$$dL/dw = -2(y - \hat{y})x = -2(y - wx)x$$

Second Derivative with respect to  $w$ :  $d^2L/dw^2 = 2x^2$

Interpretation:

Since  $2x^2$  is always non-negative, this confirms that the loss function  $L(y, \hat{y})$  is convex with respect to  $w$  and has a minimum point.



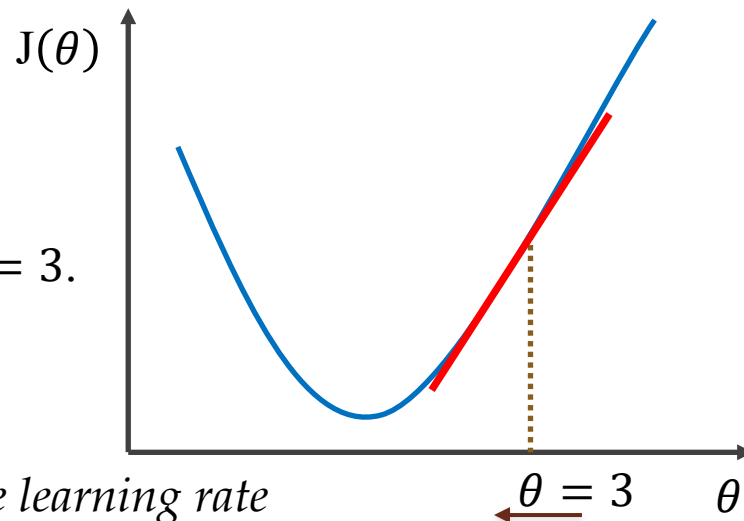
# Summary of Gradient Descent (GD) idea (One Parameter)

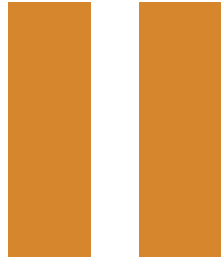
$$\hat{y} = h_{\theta}(x) = \theta x$$

$$\text{Cost Function: } J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

1. Start with a random value of  $\theta$  (e.g.  $\theta = 3$ )
2. Compute the gradient (derivative) of  $J(\theta)$  at point  $\theta = 3$ .
3. Recompute  $\theta$  as:

$$\theta = \theta - \alpha * \text{gradient} \text{ where } \alpha \text{ is the learning rate}$$





**BREAK**



**Please come back @  
2:30 PM EST  
1:30 PM CST**



Hands On.....

# BACK-UP



# References

In addition to the references in each slide the below are leveraged throughout this course.

Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

VanderPlas, J. (2017). Python Data Science Handbook. O'Reilly.

Wolff, S. G. (2018). Less is more: optimizing classification performance through feature selection in a very-high-resolution remote sensing object-based urban application. GIScience & Remote Sensing. doi:10.1080/15481603.2017.1408892

Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press.