Calculate Slope

$$m = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 0/0 \quad \text{or undefined}$$

Introduction to Limits.

Limits: A limit describes the ~~future~~ value ~~of a~~ function ~~is~~ approaches when the ~~in~~ input variable to a function approaches a specific value. In our case, the input variable is x_2 and our function is (see above).

Calc for machine learning / Understanding Limits

→ Mission of Undefind Limit to defined limit

$f(x_2) = -x^2 + 3x - 1$

$\lim_{x_2 \to 3} \frac{f(x_2) - f(x_1)}{x_2 - x_1}$

$f(x_1) = -1$

$x_1 = +3$

expand:

$$\lim_{x_2 \to 3} \frac{-(x_2)^2 + 3x_2 - 1 + 1}{x_2 - 3} \Rightarrow \frac{-x_2^2 + 3x_2}{x_2 - 3} \Rightarrow \frac{x_2(-x_2 + 3)}{x_2 - 3}$$

$$\Rightarrow \frac{-x_2(x_2 - 3)}{x_2 - 3} \Rightarrow -x_2$$

$$\lim_{x_2 \to 3} -x_2 = -3$$

Sub chapter: Finding Extreme Points (maxima)

~~03 Differentiation~~

02 Introduction to Derivatives

Confusion: Derivative is defined as a function that can determine the slope of a tangent line for any x value along the function.

The widget is confusing as the green line is only tangent when $h = 0$. I guess my confusion is because the axes are not labeled.

03 Differentiation

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \quad f(x) = -(x)^2 + 3x - 1$$

$$\lim_{h \to 0} \frac{\left(-(x+h)^2 + 3(x+h) - 1\right) - \left(-(x)^2 + 3x - 1\right)}{h}$$

$$-(x^2 + 2xh + h^2) = -x^2 - 2xh - h^2$$

$$\lim_{h \to 0} \frac{-x^2 - 2xh - h^2 + 3x + 3h - 1 + x^2 - 3x + 1}{h}$$

$$\lim_{h \to 0} \frac{-2xh - h^2 + 3h}{h} = -2x - h + 3$$

to simplify this using direct substitution do the following see green line above

$$f(x) = -2x - 0 + 3 = -2x + 3$$

or

$y = -2x + 3$ this is the derivative which is expressed as y ~~prime~~ prime or...

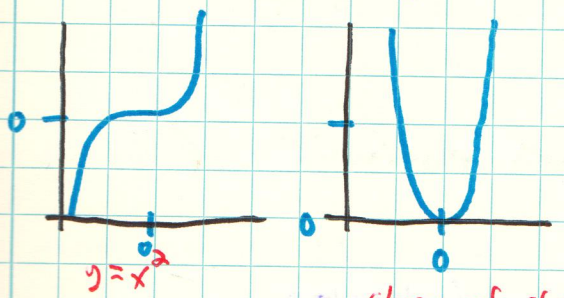
$y' = -2x + 3$ or $f'(x) = -2x + 3$

The last common notation is the following which can be read as "the derivative w/ respect to x is"

$$\frac{d}{dx}\left[-x^2 + 3x - 1\right] = -2x + 3$$

04 Critical Points: a point on a curve where the gradient = 0

function: $y = x^2$



slope of the tangent @ different x values

- Critical points are interesting in data science when they represent extreme values, which can be split into two categories: maximum & minimum.
  1) Maximums → when slope transitions from positive to negative.
  2) Minimum → When slope transitions from negative to positive
  3) When slope doesn't transition signs then it is not an extreme. See the graphs above.

06 Power Rule

$f'(x) = r x^{r-1}$

So...

1. $f(x) = x^2$, r would be 2 : & it derivative would be: $f'(x) = 2x$

2. $f(x) = -(x)^2 + 3x - 1$

   $f'(x) = -2x + 3$ wow so much easier

The actual exercise, calculate the derivative

1. $f(x) = x^5$ $\quad f'(x) = 5x^4$

2. $f(x) = x^9$ $\quad f'(x) = 9x^8$

Next, look up the slope at $x = 2$ for (1) & $x = 0$ for (2)

1. $f'(2) = 5(2)^4 = 80$

2. $f'(0) = 9(0)^8 = 0$

07 Linearity of Differentiation

Consists of 2 rules:

1) The Sum Rule

$$\frac{d}{dx}[f(x)+g(x)] = \frac{d}{dx}[f(x)] + \frac{d}{dx}[g(x)]$$

2) The constant factor rule

$$\frac{d}{dx}[c\,f(x)] = c\frac{d}{dx}[f(x)]$$

example:

$$\frac{d}{dx}[-x^3+x^2] = \frac{d}{dx}[-x^3] + \frac{d}{dx}[x^2] = -3x^2 + 2x$$

The actual exercise...

Calculate the derivative of

$f(x) = x^5 - x$, the determine the slope when $x=1$

$\frac{d}{dx} = 5x^4 - 1 \quad f'(1) = 5(1)^4 - 1 = 4$

$f(x) = x^3 - x^2$, slope when $x=2$

$\frac{d}{dx} = 3x^2 - 2x \quad f'(2) = 3(2)^2 - 2(2) = 8$

oops, I used the power rule & not the linearity of differentiation rule.

08 Practicing Finding Extreme Values

1. Find the critical points for $x^3 - x^2$

To do this I'll need to determine the derivative, set it equal to 0 & solve for x.

$f(x) = x^3 - x^2 \quad \frac{d}{dx} = 3x^2 - 2x$

this is the easy part I'm not sure how to solve for x, when there are more than 1 solution

$0 = 3x^2 - 2x$

ok, I'm just going to try some stuff here.

$3x^2 = 2x \Rightarrow 3x = 2 \Rightarrow x = \frac{2}{3}$

wholy shit, that's one solution...

divide both sides by x

the other solution is probably the easy one, 0

critical points = $\frac{2}{3}$ & 0

2. Determine if a critical point is a relative maximum or minimum

Recall if slope goes from (+) to (-) it's a max & vice versa if it's a min

01 Overview of Linear Algebra

a. Optimal Salary Problem

	base pay/week	$/hour
1. $y = 1000 + 30x$	$1000/wk	$30/hr
2. $y = 100 + 50x$	$100/wk	$50/hr

y: the total earned each week

If we know how much we'd like to ~~earn~~ earn each week.

02 Solving Linear Systems By Elimination

The point where both functions intersect is called the solutions to the system.

Using equations 1 & 2 above we can solve for x.

$$1000 + 30x = 100 + 50x \Rightarrow 1000 - 100 = 50x - 30x$$

$$900 = 20x$$

$$x = 45$$

The same can be done to solve for y.

$$y = 30(45) + 1000 = 2,350$$

$$y = 50(45) + 100 = 2,350$$

This illustrates that after 45 hours we will have earned the same amount of $2,350.00

03 Representing Functions In General Form & 04 Representing An Augmented Array Matrix In NumPy

In linear algebra we usually represent linear functions in the general form.

$$Ax + By = C$$

In the general form, variables & their coefficients are on the left side & the constant term is on the right. We can switch from point-slope form to the general form by rearranging the terms:

$$y = mx + b \Rightarrow mx - y = -b$$

function 1 in general form:

$$30x - y = -1000$$

function 2 in general form

$$50x - y = -100$$

To represent both linear functions in a system we use an augmented matrix:

$$\left[\begin{array}{cc|c} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{array}\right]$$

06 (Matrix) Row Operations

1) Any two rows can be swapped
(continue from previous matrix)

$\longrightarrow$ Swap R1 & R2 $\longrightarrow$ $\begin{bmatrix} 50 & -1 & -100 \\ 30 & -1 & -1000 \end{bmatrix}$

2) Any row can be multiplied by a nonzero constant.

$$\begin{bmatrix} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{bmatrix} \xrightarrow{\substack{2 \cdot R1 \\ 3 \cdot R2}} \begin{bmatrix} 60 & -2 & -2000 \\ 150 & -3 & -300 \end{bmatrix}$$

3) Any row can be added to another row

$$\begin{bmatrix} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{bmatrix} \xrightarrow{R2 = R2 + R1} \begin{bmatrix} 30 & -1 & -1000 \\ 80 & -2 & -1100 \end{bmatrix}$$

07 Simplifying Matrix To Echelon Form

Two Steps to find the solution of a matrix

1) Rearrange the matrix into echelon form. In this form the values on the diagonal locations are all equal to (1) and the values below the diagonal locations are all equal to (0).

$$\begin{bmatrix} 1 & ? & ? \\ 0 & 1 & ? \end{bmatrix}$$

First, divide R1 by 30 so that the diagonal value in the 1st row is 1:

$$\begin{bmatrix} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{bmatrix} \longrightarrow R1 = R1/30 \rightarrow \begin{bmatrix} 1 & -1/30 & -1000/30 \\ 50 & -1 & -100 \end{bmatrix}$$

Then, subtract 50 times the 1st row from the second:

$$\begin{bmatrix} 1 & -1/30 & -1000/30 \\ 50 & -1 & -100 \end{bmatrix} \rightarrow R2 = R2 - (R1 \times 50)$$

$$\begin{bmatrix} 1 & -1/30 & -1000/30 \\ 0 & 20/30 & 47{,}000/30 \end{bmatrix}$$

Next, multiply R2 by 1.5 to get the diagonal value in the second row to be (1).

$$\begin{bmatrix} 1 & -1/30 & -1000/30 \\ 0 & 20/30 & 47{,}000/30 \end{bmatrix} \rightarrow R2 = R2 \cdot \frac{3}{2} \rightarrow \begin{bmatrix} 1 & -1/30 & -1000/30 \\ 0 & 1 & 141{,}000/60 \end{bmatrix}$$

To complete the transformation we'll have to zero out the second value in the first row. To do this lets ~~subtract R2x~~ add R2 x 1/30 to R1.

$$\begin{bmatrix} 1 & -\frac{1}{30} & \Big| & -\frac{1000}{30} \\ 0 & 1 & \Big| & \frac{141,000}{60} \end{bmatrix} \rightarrow R1 = R1 + \underbrace{(R2 \times \frac{1}{30})}_{\rightarrow\ 0 \ +\ \frac{1}{30}\ \Big|\ \frac{141,000}{1800}}$$

$$\begin{bmatrix} 1 & +0 & \Big| & -\frac{2250}{30} \\ 0 & 1 & \Big| & \frac{141,000}{60} \end{bmatrix}$$

not sure my math is right here, let me run it through python

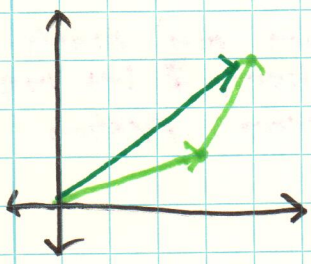
$$\Longrightarrow \begin{bmatrix} 1 & 0 & \Big| & -\frac{2250}{30} \\ 0 & 1 & \Big| & \frac{2350}{30} \end{bmatrix}$$

03 Vector Operations

Vector Summation

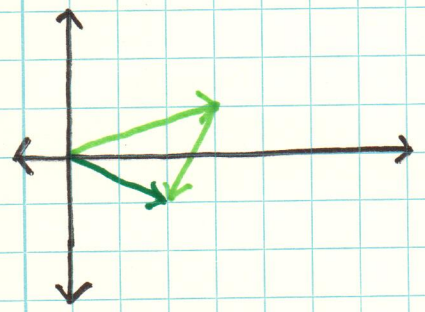
Stop putting the horizontal line in there!

$$\begin{bmatrix}3\\1\end{bmatrix} + \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}4\\3\end{bmatrix}$$



Vector Subtraction

$$\begin{bmatrix}3\\1\end{bmatrix} - \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}2\\-1\end{bmatrix}$$



04 Scaling Vectors

$$3 \times \begin{bmatrix}3\\1\end{bmatrix} = \begin{bmatrix}9\\3\end{bmatrix}$$

05 Vectors in NumPy

See code..

06 Dot Product

$$\vec{a} * \vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

What this looks like visually

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} = (1 \times 3) + (2 \times 0) + (1 \times 1) = 4$$

Unlike the other vector operations, the result of the ~~scalar value~~ dot product is a scalar value not a vector.

To compute the dot product between 2 vectors, we need to use the numpy.dot() function. This function accepts NumPy ndarray objects as the required parameters. The main quirk is that one of the two vectors need to be represented as a row vector while the other a column vector.

07 Linear Combination

Using vectors we can determine if a certain vector can be obtained by combining other vectors. For example we may want to know if we can combine the vectors $\begin{bmatrix}3\\1\end{bmatrix}$ & $\begin{bmatrix}1\\2\end{bmatrix}$ to obtain $\begin{bmatrix}4\\-2\end{bmatrix}$

Here's what it looks like mathematically:

$$c_1\begin{bmatrix}3\\1\end{bmatrix} + c_2\begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}4\\-2\end{bmatrix}$$

Being able to scale vectors using scalar multiplication then adding or subtracting these scaled vectors is known as <u>linear combination</u>. This concept is crucial to being able to being able to apply linear algebra to solve useful problems.

if

$$w = \begin{bmatrix}1\\2\end{bmatrix}$$

$$v = \begin{bmatrix}3\\1\end{bmatrix}$$

08 Linear Combination & Vectors

Circle back to the salary problem represented as a matrix.

$$\left[\begin{array}{cc|c} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{array}\right]$$

We can now ~~tie~~link this augmented matrix to the linear combination of vectors idea we just discussed.

We want to know if $\begin{bmatrix} -1000 \\ -100 \end{bmatrix}$ is a linear combination of $\begin{bmatrix} 30 \\ 50 \end{bmatrix}$ & $\begin{bmatrix} -1 \\ -1 \end{bmatrix}$. To find the solution to this system

we need to find the constants x & y where the following equation is true.

$$x\begin{bmatrix} 30 \\ 50 \end{bmatrix} + y\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

In the last mission, we solved the augmented matrix by using row operations to obtain the following form.

$$\left[\begin{array}{cc|c} 1 & 0 & x \\ 0 & 1 & y \end{array}\right]$$

09 the Matrix Equation

The matrix equation is the representation of a linear system using only matrices & vectors
Here's the augmented matrix we started out with:

$$\left[\begin{array}{cc|c} 30 & -1 & -1000 \\ 50 & -1 & -100 \end{array}\right]$$

This augmented matrix is the shorthand representation for the matrix equation:

$$\begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

On the left side we're multiplying a matrix containing the coefficients w/ the vector containing the variables. The right side contains the constant values. This separation of coefficients & variables from the constants should be familiar. This is exactly what we did in the general form as well.

It's common practice to use $x_1, x_2, \ldots x_n$ instead of x & y to represent the individual values in the solution vector.

$$\begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

This allows us to work with vectors w/ any number of elements (instead of just 26 for the number of letters in the english dictionary). We can now introduce the arithmetic representation of the matrix equation:

$$A\vec{x} = \vec{b}$$

Where A represents the coefficient matrix, $\vec{x}$ represents the solution vector, and $\vec{b}$ represents the constants. Note that $\vec{b}$ can't be a vector containing all zero's also known as the zero factor & represented using $\vec{0}$.

WITNESSED AND UNDERSTOOD
SIGNED DATE SIGNED

Before we can work with this form of the system we need to learn about the following topics in the following mission.

1. The rules that ~~describe~~ describe how matrices can be combined
2. How to multiply a matrix w/ a vector
3. How to calculate the solution vector X w/o using Gaussian elimination.

01 Basic Matrix Operations

$$A = \begin{bmatrix} 0.7 & 3 & 9 \\ 1.7 & 2 & 9 \\ 0.7 & 9 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 3 & 3 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 0.7 & 6 & 12 \\ 2.7 & 4 & 10 \\ 0.7 & 10 & 4 \end{bmatrix}$$

$$5A = \begin{bmatrix} 3.5 & 15 & 45 \\ 8.5 & 10 & 45 \\ 3.5 & 45 & 10 \end{bmatrix}$$

02 Matrix Vector Multiplication

$$A c = \begin{bmatrix} 0.7 & 3 & 9 \\ 1.7 & 2 & 9 \\ 0.7 & 9 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} (0.7 \times 1) + (3 \times 2) + (9 \times 1) \\ (1.7 \times 1) + (2 \times 2) + (9 \times 1) \\ (0.7 \times 1) + (9 \times 2) + (2 \times 1) \end{bmatrix} = \begin{bmatrix} 15.7 \\ 14.7 \\ 20.7 \end{bmatrix}$$

- When we multiply a matrix by a vector we are essentially combining each row in the matrix w/ the column vector (pen died)
- To multiply a matrix by a vector, the # of colums in the matrix needs to match the number of rows in the vector.
- To multiply a matrix with a vector in Numpy, we need to use the numpy.dot() function.
- The process is known as the dot product between each row in the matrix & the column vector.

03 Matrix Multiplication

$$A = \begin{bmatrix} 0.7 & 3 \\ 1.7 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$

$$A \times B = \begin{bmatrix} (0.7 \times 1 + 3 \times 2) & (0.7 \times 2 + 3 \times 3) \\ (1.7 \times 1 + 2 \times 2) & (1.7 \times 2 + 2 \times 3) \end{bmatrix} = \begin{bmatrix} 6.7 & 10.4 \\ 5.7 & 9.4 \end{bmatrix}$$

→ In matrix multiplication we extend the dot product done in vector matrix multiplication, to preform a dot product between each row in the first matrix and each column in the second matrix.

→ As with matrix vector multiplication the number of colums in the 1st matrix must match the number of rows in the second matrix.

→ Order of operations matters.

$$A \times B \neq B \times A$$

DATE 03/13/2018 SUBJECT Math Refresher - Matrix Algebra PROJECT NO.

04 Matrix Transpose

- The transpose of a matrix switches the rows and columns of a matrix. Think of this operation as a rotation.

- This is sometimes necessary because of the requirements for matrix multiplication. Transposing a matrix allows us to multiply matrices together that, by default, don't overlap in # of rows & columns.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

→ Rules

1.) ~~The transpose~~ when taking the transpose of the sum of two matrices, we can distribute the transpose operation to each matrix.

$$(A + B)^T = A^T + B^T$$

2) No description provided other than being a counter-intuitive.

$$(AB)^T = B^T A^T$$

Seems intutive to me...?

3) Also: $(A \times B)^T = A^T \times B^T$ I know this is a repeat of #1 but I proved it to myself & I get it now.

05 Identity Matrix

In the matrix equation that we discussed in the last mission, we're trying to solve for the vector $\vec{x}$.

$$A\vec{x} = \vec{b}$$

Right now, the matrix A multiplies the vector $\vec{x}$ and we need a way to cancel A.

Lets look at the identity matrix, which we touched on briefly at the end of the first mission in this course. If you recall the identity matrix contains 1 along the diagonals and zero elsewhere. Here's what the 2×2 identity matrix looks like, often represented as I_2:

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

When we multiply I_2 w/ any vector containing 2 elements, the resulting vector matches the original vector exactly:

$$I_2\vec{x} = \vec{x}$$

This is because each element in the vector is multiplied once by the diagonal 1 value in the identity matrix.

specific case

$$I_2 x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1\times3 + 0\times2 \\ 0\times3 + 1\times2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

general case

$$I_2 x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1\times x_1 + 0\times x_2 \\ 0\times x_1 + 1\times x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

If we can transform matrix A & convert it to the identity matrix, then only the solution vector $\vec{x}$ will remain.

Lets practice working w/ the identity matrix before exploring how to transform A into I.

We can create any I_n identity matrix using the numpy.identity() function. This function only has 1 required parameter, n, which specifies the $n \times n$ identity matrix we want.

06 Matrix Inverse

Now we can revisit canceling A by transforming it to the identity matrix. Multipilying the inverse of a matrix by a matrix does this task.

The matrix inverse is similar to the idea of the multiplicative inverse. For example, let's say we want to solve for x in the equation: $5x = 10$. To do so we need to multiply both sides by the multiplicative inverse of 5, which is 5^{-1} or $(\frac{1}{5})$.

To solve for the vector $\vec{x}$ in the matrix equation, we need to multiply both sides by the inverse of A:

$$A^{-1} \times A\vec{x} = A^{-1} \times \vec{b}$$

This simplifies to $I\vec{x} = A^{-1}\vec{b}$ and we're left w/ the formula for calculating the solution vector:

$$\vec{x} = A^{-1}\vec{b}$$

Matrix Calculation for a 2×2 matrix:

if $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

The term $ad-bc$ is known as the <u>determinant</u> & is often written as $det(A) = ad-bc$ or as $|A| = ad-bc$. Because were dividing by the determinant when calculating the matrix inverse, a 2×2 matrix is only invertible if the determinant is not $= 0$.

In this & the next step we'll focus on finding the matrix inverse when A is a 2×2 matrix. We'll get to higher dimensional matrix later.

07 Solving the matrix equation

Now that we know how to calculate the matrix inverse, we can solve our system using the matrix equation $A\vec{x} = \vec{b}$:

$$\begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

We start by left multiplying A^{-1} on both sides

$$\begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}^{-1}\begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}^{-1}\begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

This simplifies to:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 30 & -1 \\ 50 & -1 \end{bmatrix}^{-1}\begin{bmatrix} -1000 \\ -100 \end{bmatrix}$$

numpy.linalg.inv() can be used to calculate the inverse of an np array.

08 Determinant For Higher Dimensions

More Terminology

Square matrices: Matrices that contain the same number of rows & columns. We can only compute the determinants & matrix inverse for square matrices.

Minor matrices: The determinant of a higher-dimensional system involves down the full matrix into minor matrices.

General Example

$$\det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) = a \times \det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) \text{ crap}$$

$$\det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) = a \times \det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) - b \times \det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) + c \times \det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right)$$

Specific case

$$\det\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) = 1 \times \det\left(\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}\right) - 2 \times \det\left(\begin{bmatrix} 4 & 6 \\ 7 & 8 \end{bmatrix}\right) + 3 \times \det\left(\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}\right)$$

We can use `numpy.linalg.det()` to accomplish this.

01 Introduction

So far we've explored two different ways to find a solution to $A\vec{x} = \vec{b}$ when b isn't a vector containing all zeros.

1.) Gaussian Elimination, involving row operations to transform the augmented representation of a linear system to echelon form, and then finally to reduced row echelon form.

2.) Compute the matrix inverse of A & left multiplying both sides of the equation to find $\vec{x}$.

These techniques will not help us in the following situations:

1.) The solution set for a linear system does not exist.

2.) The solution set for a linear system isn't a single vector.

3.) b is equal to $\vec{0}$.

this mission will explore these situations.

02 Inconsistent Systems

Inconsistent Systems: An inconsistent system will have two or more equations that don't overlap in values, making it impossible to find a solution.

Let's explore the following example:

$$\left[\begin{array}{cc|c} 8 & 4 & 5 \\ 4 & 2 & 5 \end{array}\right]$$

Reduce by dividing ~~both~~ the 1st row by 2

$$\left[\begin{array}{cc|c} 4 & 2 & 2.5 \\ 4 & 2 & 5 \end{array}\right]$$

Subtract the top row from the bottom row ($R_2 - R_1$) & divide the 1st row by 4 ($R_1/4$)

$$\left[\begin{array}{cc|c} 4 & 2 & 2.5 \\ 4 & 2 & 5 \end{array}\right] \rightarrow R_2 = R_2 - R_1 \rightarrow \left[\begin{array}{cc|c} 4 & 2 & 2.5 \\ 0 & 0 & 2.5 \end{array}\right] \rightarrow R_1 = R_1/4 \rightarrow \left[\begin{array}{cc|c} 1 & \frac{1}{2} & 0.625 \\ 0 & 0 & 2.5 \end{array}\right]$$

The augmented matrix ends up with zeros for the coefficients in R_2.

By plotting these two systems it will be clear why no solution exists.

02 Inconsistent Systems

Extract slope-intercept form from the following matrix:

$$\left[\begin{array}{cc|c} 8 & 4 & 5 \\ 4 & 2 & 5 \end{array}\right]$$

1st attempt w/o looking anything up.

$y = mx + b \Rightarrow x\,m \ominus y = \ominus b$

$8x - 4y = 5 \Rightarrow 4y = -8x - 5 \Rightarrow y = -2x - \frac{5}{4}$

$4x - 2y = 5 \Rightarrow 2y = -4x - 5 \Rightarrow y = -2x - \frac{5}{2}$

I can already see why this won't have a solution, these functions will never intersect.

Close but no cigar, ignore the signs, and drop the coefficients and constants as is from the matrix...
Try again:

$8x + 4y = 5 \Rightarrow 4y = -8x + 5 \Rightarrow y = -2x + \frac{5}{4}$

$4x + 2y = 5 \Rightarrow 2y = -4x + 5 \Rightarrow y = -2x + \frac{5}{2}$

03 Singular Matrix

If we calculate the determinate from the coefficient matrix from the previous step we see that the result is 0. Recall that this means that the matrix has no inverse. When this is the case the matrix is Singular.

04 Possible Solutions For Nonhomogenous Systems

Nonhomogenous Systems: Systems where the ~~coefficient matrix~~ constants vector $\vec{b}$ doesn't contain all zeros. $\begin{bmatrix} 8 & 4 & | & 5 \\ 4 & 2 & | & 5 \end{bmatrix}$

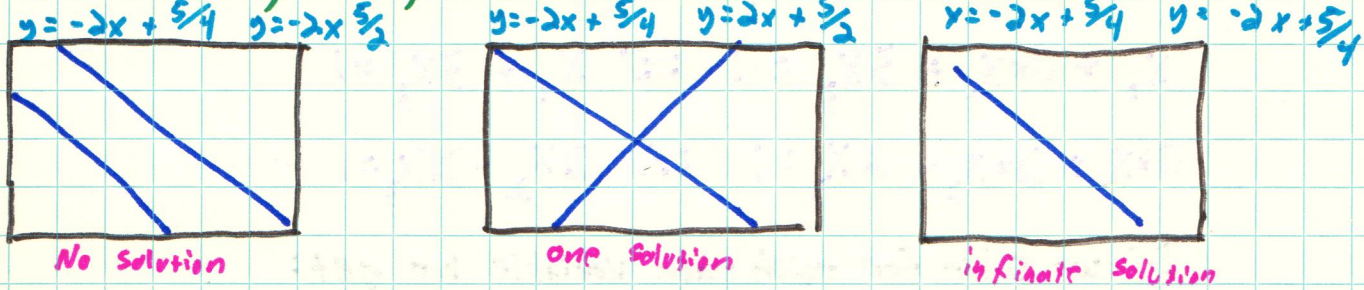
Homogenous Systems: Systems where the constants vector $\vec{b}$ is equal to the zero vector.

$$\left[\begin{array}{cc|c} 8 & 4 & 0 \\ 4 & 2 & 0 \end{array}\right]$$

The distinction is that ~~the~~ Homogeneous systems always have a solution, the zero vector.

Square, Nonhomogenous Systems: For square, nonhomogenous systems, there are 3 possible solutions.

1.) no solution
2.) a single solution
3.) infinately many solutions



04 Possible Solutions For Nonhomogenous Systems

Rectangular, Nonhomogeneous System

For a rectangular (nonsquare), ~~matrix~~ nonhomogeneous systems there are 2 possible solutions:

1.) no solution
2.) Infinitely many solutions

Example:

$$\left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 1 & 2 & 1 & 2 \end{array}\right]$$

Transform the matrix into echelon form. Start by subtracting $R_2 - R_1$

$$\left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 1 & 2 & 1 & 2 \end{array}\right] \rightarrow R_2 = R_2 - R_1 \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 0 & 2 & -2 & 1 \end{array}\right]$$

Next divide $R_2/2$

$$\left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 0 & 2 & -2 & 1 \end{array}\right] \rightarrow R_2 = \frac{R_2}{2} \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 3 & 1 \\ 0 & 1 & -1 & 0.5 \end{array}\right]$$

Convert back to equation form:

$$x_1 + 0 \times x_2 + 3x_3 = 1 \Rightarrow x_1 + 3x_3 = 1$$

$$0 \times x_1 + 2 \times x_2 - 2x_3 = 1 \Rightarrow 2x_2 - 2x_3 = 1$$

If we solve for ~~each~~ each variable we get:

$x_1 = 1 - 3x_3$

$x_2 = \frac{1}{2} + 3x_3$

04 Possible Solutions For Nonhomogeneous Systems

X_3 is known as a free variable, because it is allowed to vary freely. You'll notice that both x_1 & x_2 are expressed in terms of X_3. This system has infinitely many solutions, because for any real number we plug in for X_3, we'll get different values for X_1 & X_2.

Said another way, there are infinate solutions to this system because there's an infinite number of values that X_3 could be that make the system true. This set of infinate solutions is know as a Solution Space. Let's test this using some random examples.

05 Homogeneous Systems

Trivial Solution: We can solve any homogeneous system $A\vec{x} = 0$, by setting $\vec{x}$ to the zero vector. This is known as the trivial solution. This isn't very useful were interested in determining if infinitely many solutions exist or not. Here's an example:

Reduce to echelon form

$$\begin{bmatrix} 6 & 10 & -8 & 0 \\ -6 & -4 & 8 & 0 \\ 3 & 1/2 & -4 & 0 \end{bmatrix} \rightarrow R_2 = R_2 + R_1 \rightarrow \begin{bmatrix} 3 & 5 & -4 & 0 \\ 0 & 6 & 0 & 0 \\ 3 & 1/2 & -4 & 0 \end{bmatrix} \rightarrow R_2 = R_2/2 \rightarrow \begin{bmatrix} 3 & 5 & -4 & 0 \\ 0 & 3 & 0 & 0 \\ 3 & 1/2 & -4 & 0 \end{bmatrix}$$

$$\rightarrow R_1 = R_1/2$$

$$R_3 = R_3 - R_1$$

$$\begin{bmatrix} 3 & 5 & -4 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 9/2 & 0 & 0 \end{bmatrix} \leftarrow R_3 = R_3 - (R_2/3/3) \leftarrow \begin{bmatrix} 3 & 5 & -4 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_1 = R_1/3$$

$$\begin{bmatrix} 1 & 5/3 & -4/3 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow R_2 = R_2/3 \rightarrow \begin{bmatrix} 1 & 5/3 & -4/3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_1 = R_1 - (R_2/3/5)$$

$$\begin{bmatrix} 1 & 0 & -4/3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

05 Homogeneous Systems

If we convert this to equation form we get

$$x_1 - \tfrac{4}{3}x_3 = 0 \Rightarrow x_1 = \tfrac{4}{3}x_3$$

$$x_2 = 0$$

$$0 = 0$$

In this system x_3 is a free variable, x_3 can be any real number. Thus there is an infinite number of solutions to this system

Parametric Vector Form: When the solution is a solution space (and not just a unique set of values), it's common to rewrite it into parametric vector form

$$\vec{x} = x_3 \begin{bmatrix} 4/3 \\ 0 \\ 1 \end{bmatrix}$$

Here's where the values in the parametric vector form come from:

1.) Since x_3 is the free variable w/ respect to x_3.

2.) The solution to each variable makes up the vector:

- x_1: $4/3$
- x_2: 0
- x_3: is the free variable, that's why we put 1

05 Homogeneous Systems

The Exercise...

- Create a function that uses the solution functions to calculate X_1 & X_2
- Plug these values into the original linear system and check if the values turn out to be true

03 Derivative of the Cost Function

$$MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}(a_1 x_1^{(i)} - y^{(i)})^2$$

$$\frac{d}{da_1} MSE(a_1) = \frac{d}{da_1}\frac{1}{n}\sum_{i=1}^{n}(a_1 x_1^{(i)} - y^{(i)})^2$$

apply linearity of differentiation property of calculus

$$\frac{d}{da_1} MSE(a_1) = \frac{1}{n}\sum_{i=1}^{n}\frac{d}{da_1}(a_1 x_1^{(i)} - y^{(i)})^2$$

applying both the chain rule & the power rule yields:

$$= \frac{1}{n}\sum_{i=1}^{n} 2(a_1 x_1^{(i)} - y^{(i)})\frac{d}{da_1}(a_1 x_1^{(i)} - y^{(i)})$$

→ Here's how this works

$$F'(x) = f'(g(x))(g'(x))$$

- f': derivative of outside function
- $g(x)$: inside function left alone
- $g'(x)$: derivative of inside function

→ In our case...

derivative of inside function $\Rightarrow \frac{d}{da_1}(a_1 x_1^{(i)} - y^{(i)})$

inside function left alone $\Rightarrow (a_1 x_1^{(i)} - y^{(i)})$

Because we are $a_1 x_1^{(i)} - y^{(i)}$ w/ respect to a_1, we treat $y^{(i)}$ as constants:

$$\frac{d}{da_1}\left(a_1 x_1^{(i)} - y^{(i)}\right) = x_1$$

Then we can simplify:

$$\frac{d}{da_1} MSE(a_1) = \frac{2}{n}\sum_{i=1}^{n} x_1^{(i)}\left(a_1 x_1^{(i)} - y^{(i)}\right)$$

05 Gradient of the cost function

In this case w/ have two parameters values (a_0 & a_1), the cost function is now a function of two variables not 1.

$$MSE(a_0, a_1) = \frac{1}{n}\sum_{i=0}^{n}\left(a_0 + a_1 x^{(i)} - y^{(i)}\right)^2$$

$$\frac{d}{da_0} MSE(a_0, a_1) = \frac{2}{n}\sum_{i=0}^{n} a_0 + a_1 x^{(i)} - y^{(i)}$$

$$\frac{d}{da_1} MSE(a_0, a_1) = \frac{2}{n}\sum_{i=0}^{n} x^{(i)}\left(a_0 + a_1 x^{(i)} - y^{(i)}\right)$$

Note, if you are wondering/forget where $\left[a_0 + a_1 x^{(i)} - y\right]$ is coming from. It's $y = mx + b$ set equal to zero. Remember the derivative of this is going to give us our extremes.

01 Introduction

Ordinary Least Squares (OLS) estimation formula which results in the optimal vector A:

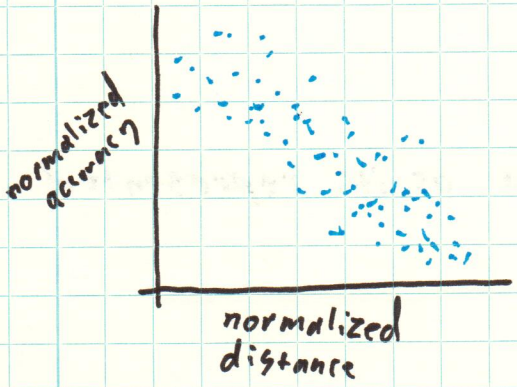
$$a = (X^T X)^{-1} X^T y$$

X^T: X transposed

To solve for the vector A or a as its represented above we use

```
np.linalg.inv()
np.linalg.dot
np.transpose()
```

He we are going to predict the ~~dis~~ accuracy of a golf ~~dri~~ drive using the distance. When we plot the normalized accuracy vs normalized distance we ~~can~~ get something simarl to ~~what~~ what I've crudely drawn below:



From this plot we see there is a linear correlation w/ a negative slope. As a result we can use a linear model for this data set. This model can be written as:

$$accuracy_i = \theta_i \, distance_i + \theta_0 + \varepsilon_i$$

θ's are coefficients and ε are error terms.

If there is little enough data where it all can fit into memory then sklearn can be used to estimate the coefficients using least squares.

Gradient Discent is a general method that can be used to estimate coefficients of nearly any model, including linear models. It minimized the residuals in the estimated model by updating each coefficient based upon it gradient.

To start we must understand cost functions. Most cost functions measure the difference between a models predictions & it's corresponding observations with the coefficients as parameters. Let's say the model is:

$$h_\theta(x) = \theta_1 x + \theta_0$$

the cost function is:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

The cost here is one half the average distance between our prediction & observation squared. As we change the coefficients of the model this cost changes. During modeling we will randomly choose the coefficients & update them intelligently to minimize this cost.

The goal being to minimize the the ~~cost~~ / error / residuals to write ~~a function~~ derive a function ~~to do this~~ to estimate this is through partial derivitaves.

$$\frac{d(\theta_0, \theta_1)}{d\theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

This is the partial of $J(\theta_0, \theta_1)$ in terms of θ_0.

$$\frac{d(\theta_0, \theta_1)}{d\theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) * x_i$$

DATE 04/09/2018 SUBJECT Math Refresher - Gradient Descent PROJECT NO.

To execute gradient descent we randomly initialize a set of parameters & update them by moving in the direction of the cost functions steepest slope, ie descending down the function. If we can find the downward slope in terms of each parameter we can move in the direction of the global minumum. Eventually the updates will converge to a near optimal set of parameters. When parameters converge the hypothesized parameters become very close to the optimal parameters. We measure convergence by finding the difference between the previous iterations cost & versus the current cost.

The general gradient descent algorithm for two variables is:

repeat until convergence:

$$\left\{ \theta_1 := \theta_1 - \alpha \frac{dJ(\theta_0, \theta_1)}{d\theta_1} \quad \theta_0 := \theta_0 - \alpha \frac{dJ(\theta_0, \theta_1)}{d\theta_0} \right\}$$

θ_1 = the current valve of our coefficient, ie how much accuracy is lost per yard of distance (slope)

α = the learning rate. This value is set by the user & controls how fast the algorithm with ~~contirge con~~ converge by changing the parameters by some % of the slope. Learning rate varies, but in ~~gene~~ general ranges from 0.0001 & 1.

Too Large → Overshoot

Too Small → Will take too many iterations

max_epoch

& = is used to limit the number of iterations