

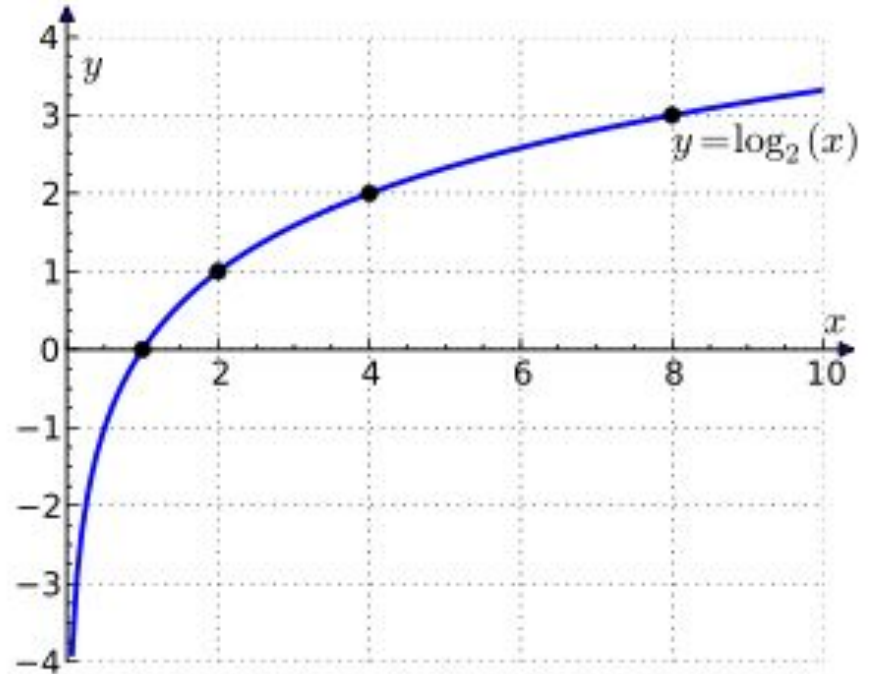
# Math Intro

Dr. Chelsea Parlett-Pelleriti

*Things are a little less  
overwhelming the second  
time you see them*

# Logarithms

Log rules:



# Logarithms

- Logarithms take values between 0 to  $+\infty$  and output values from  $-\infty$  to  $+\infty$
- Inputs between 0-1 turn into negative numbers
- Inputs between 1- $\infty$  turn into positive numbers
- The opposite of exponentiation

# Lp Norms

$$\|x\|_1 = \sum_{i=0}^n |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=0}^n x_i^2}$$

$$\|x\|_\infty = \lim_{p \rightarrow \infty} \left( \sum_{i=0}^n x_i^p \right)^{\frac{1}{p}}$$

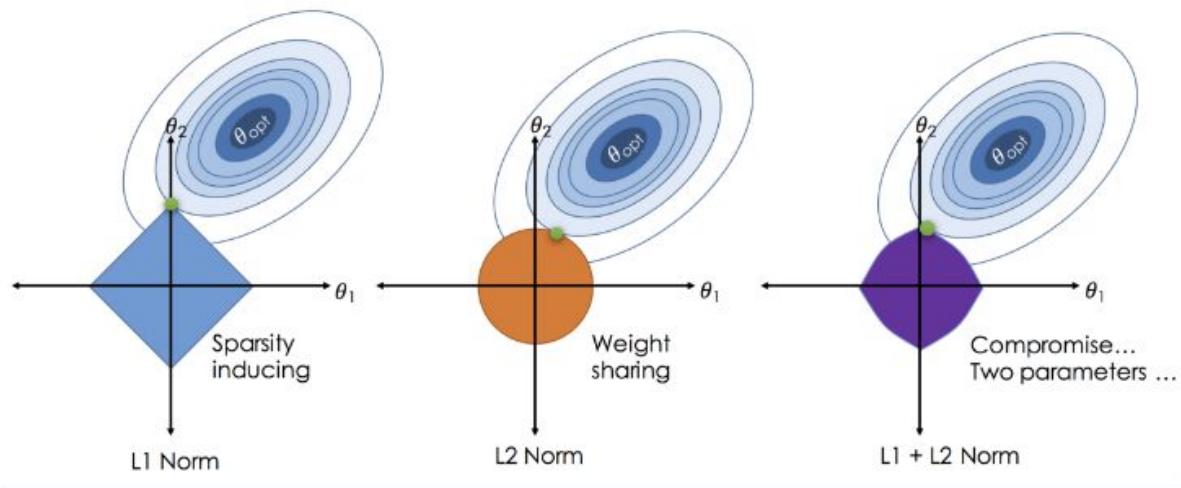
```
np.linalg.norm(x-y)
```

# Lp Norms

$$\|x\|_1 = \sum_{i=0}^n |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=0}^n x_i^2}$$

$$\|x\|_\infty = \lim_{p \rightarrow \infty} \left( \sum_{i=0}^n x_i^p \right)^{\frac{1}{p}}$$



# Lp Norms

- Lp norms measure the **size** of an object

## Dot Product

$$\begin{bmatrix} a & b \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = ax + by$$



# Dot Product

- Dot products are the sum of all the element wise multiplications of two vectors

# Linear Combinations

Example:

Course Grade:	
Attendance/Participation (in-class activities, quizzes)	
	15 %
Challenge Activities*:	
	5%
Programming Assignments:	
	40 %
Exam 1:	
	10 %
Exam 2:	
	10 %
Final:	
	20 %

# Linear Combinations

Generally:

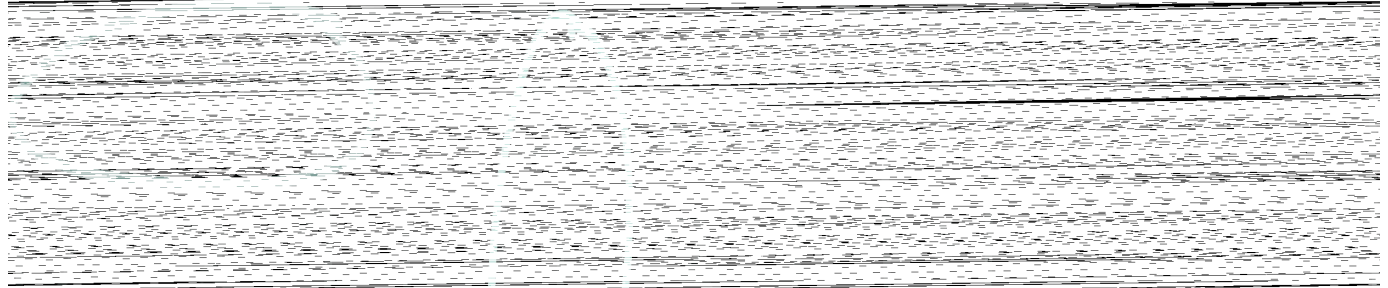
$$\mathbf{w}^T \mathbf{x} = \sum_{i=0}^n w_i * x_i = \mathbf{w} \cdot \mathbf{x}$$

# Linear Combinations

- Linear Combinations take a weight vector  $\mathbf{w}$  and dot it with our variable vector  $\mathbf{x}$
- This gives us a new composite value that uses weights ( $\mathbf{w}$ ) to combine the variables in  $\mathbf{x}$
- Linear Combinations you might know:
  - Linear Regression
  - Principal Components

# Matrices and Vectors

- Data as a Matrix/Vector (it's an excel spreadsheet)
- Matrix Algebra



# Matrices and Vectors

- Vectors are arrays of numbers that can represent a point, or a line from the origin
- Matrices are collections of vectors and have both rows and columns
  - Data frames are like matrices
  - Correlation matrices

## One Hot Encoding

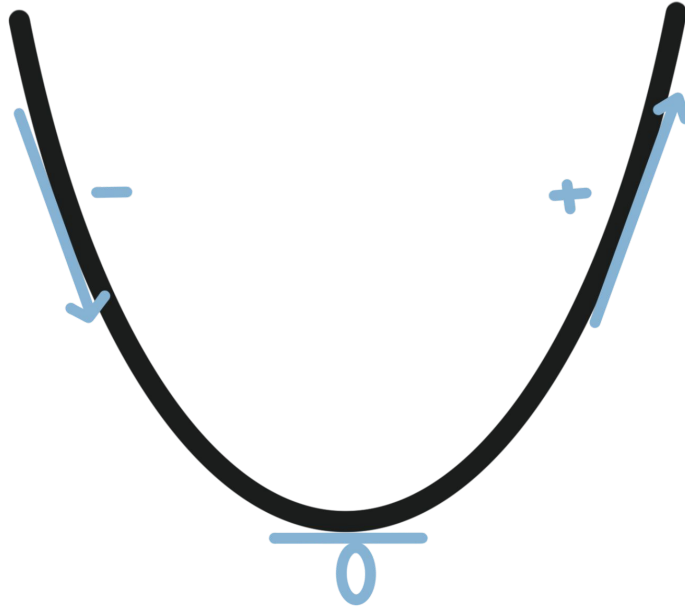
$$\begin{bmatrix} apple \\ carrot \\ celery \\ carrot \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

# One Hot Encoding

- One Hot Encoding is like dummy variables
- To convert to One Hot Encoding you create a vector with length **n**
  - **n** is the number of different elements you can have (e.g. words, categories)
- If a variable is the *n*th option, it has a 1 for the *n*th element of the vector and 0's everywhere else
- One Hot Encoding is *sparse*



# Derivatives



Derivatives tell you the “instantaneous rate of change”  
As you change a variable, how does the output change?

# Derivatives

- Derivatives tell us the rate of change of a function
- Derivatives are **0** at **minima**, **maxima**, and **saddle points**
- Derivatives are **positive** when the function is **increasing**
- Derivatives are **negative** when the function is **decreasing**

# The Chain-Rule

$$f(x) = \cos(x)$$

$$g(x) = x^2$$

$$f(g(x)) = \cos(x^2)$$

If we want to know how **changing  $x$  affects  $f(g(x))$**  we *first* need to think about how **changing  $x$  affects  $g(x)$**  *then* how **changing  $g(x)$  affects  $f(g(x))$**

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

# The Chain Rule

- To calculate the derivative of composite functions, we need to think about how changing the variable changes the inner part, then how changing the inner part changes the outer part

## Partial Derivatives

$$f(x, y) = x^2 + xy + y^2$$

$$\frac{\partial f}{\partial y} = x + 2y$$

$$\frac{\partial f}{\partial x} = 2x + y$$

For a function of with multiple variables, how does the *function change* when you change a *single variable*

# Partial Derivatives

- Partial Derivatives tell us how a multivariable function changes with respect to a *single* variable (holding all other variables constant)

## Gradient

$$f(x, y) = x^2 + xy + y^2$$

$$\frac{\partial f}{\partial y} = x + 2y$$

$$\frac{\partial f}{\partial x} = 2x + y$$

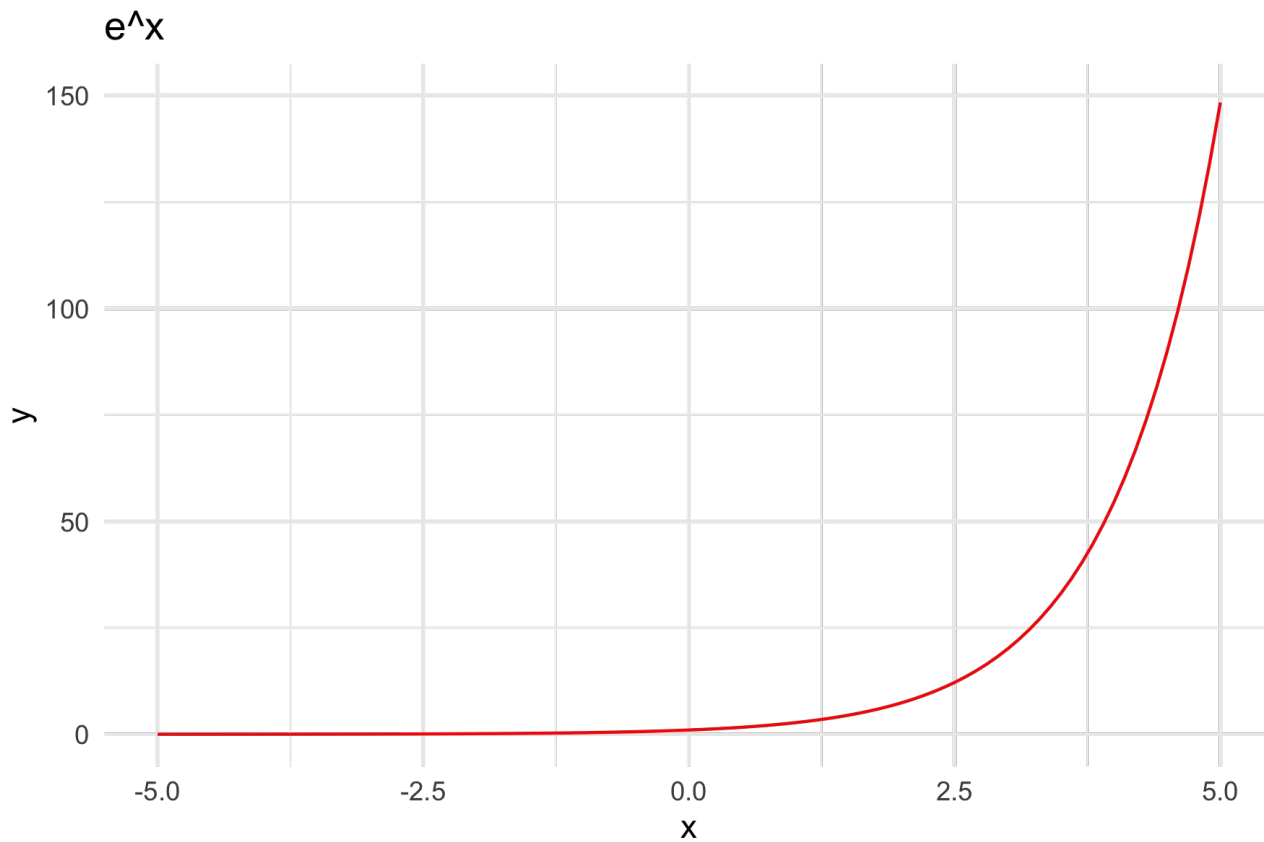
$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

# Gradient

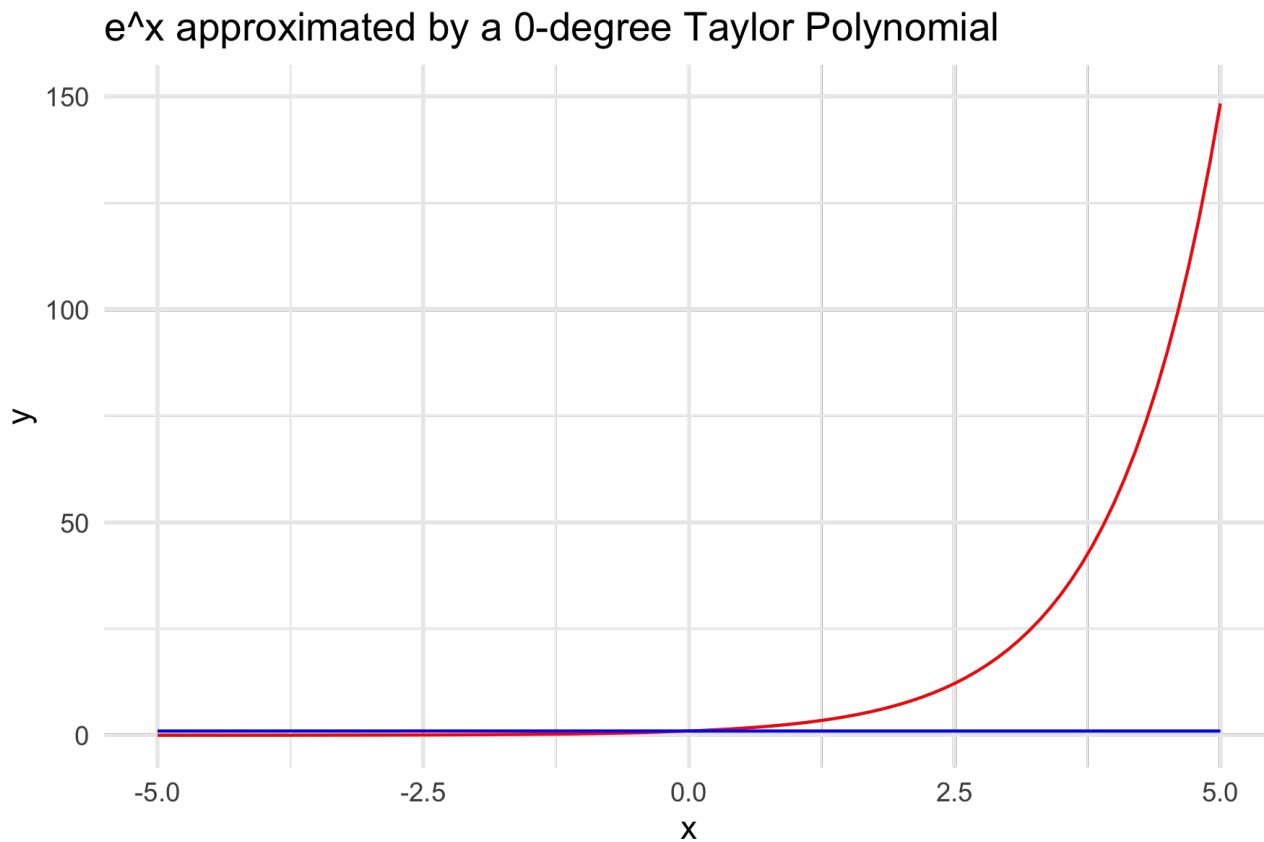
- A Gradient is a vector of partial derivatives
- it tell us how a multivariable function changes with respect to a *each* variable (holding all other variables constant)



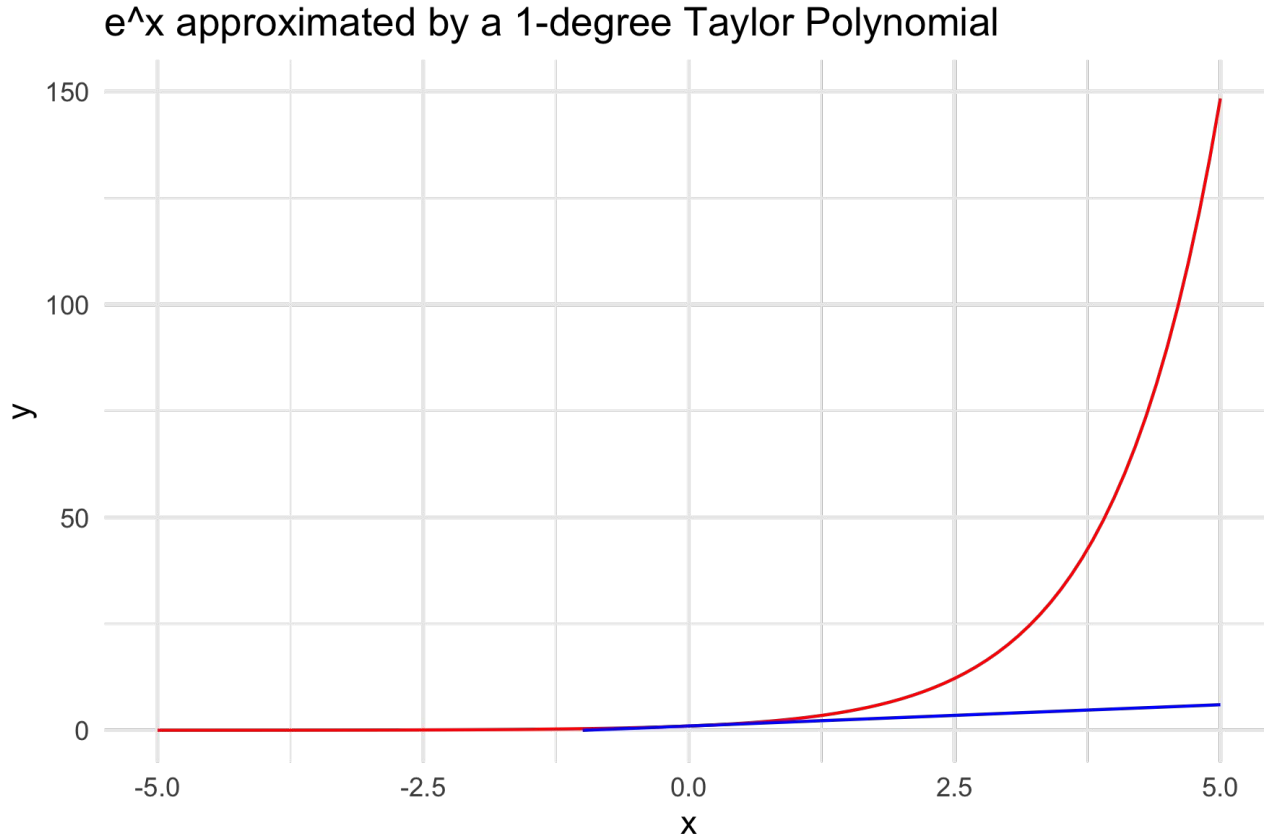
# Taylor Series (visually)



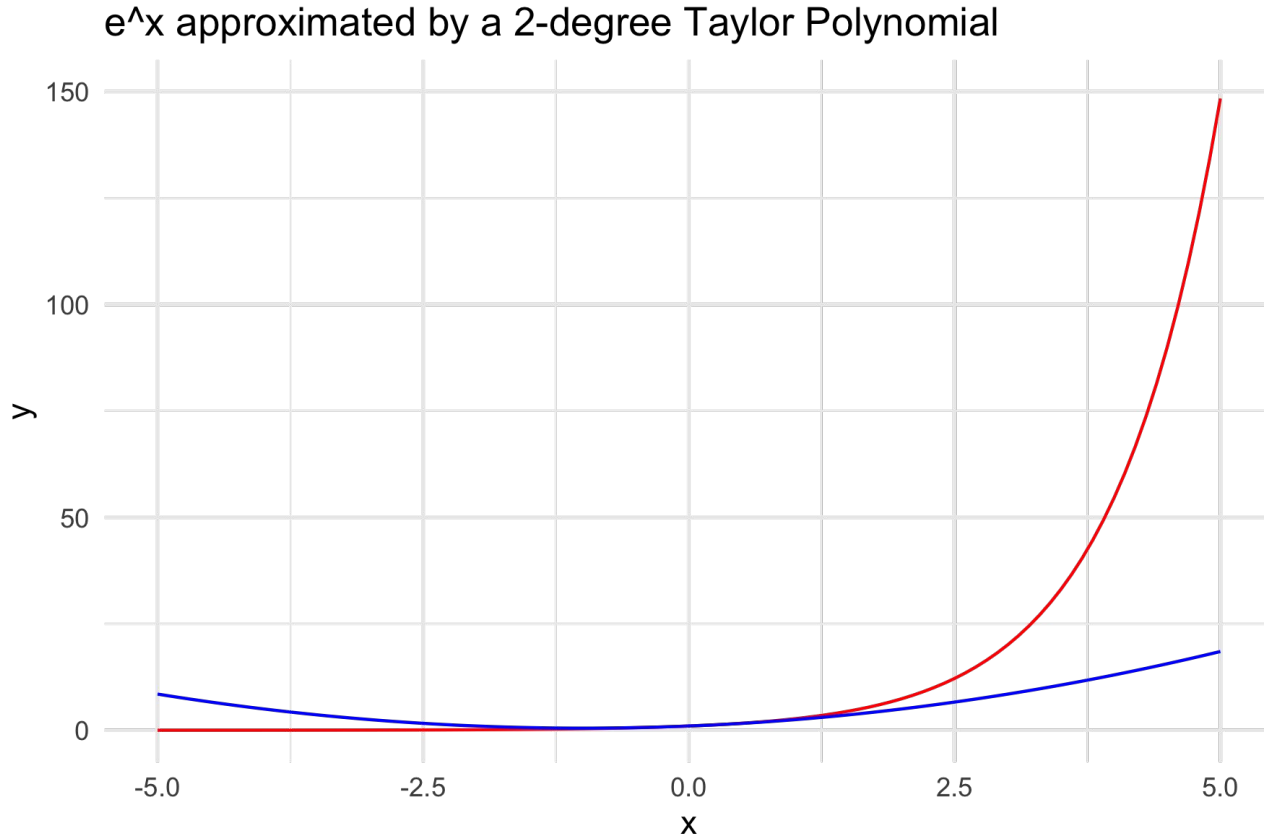
# Taylor Series (visually)



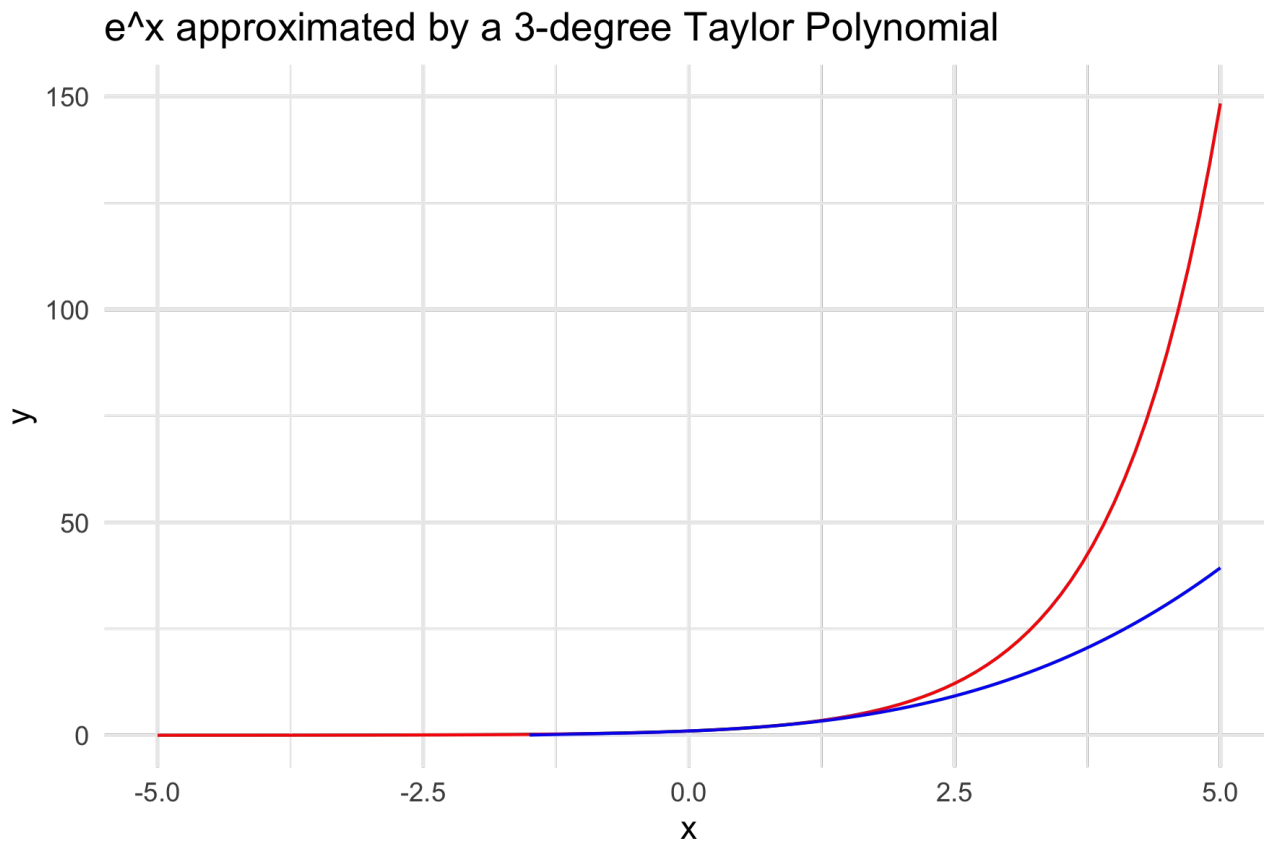
# Taylor Series (visually)



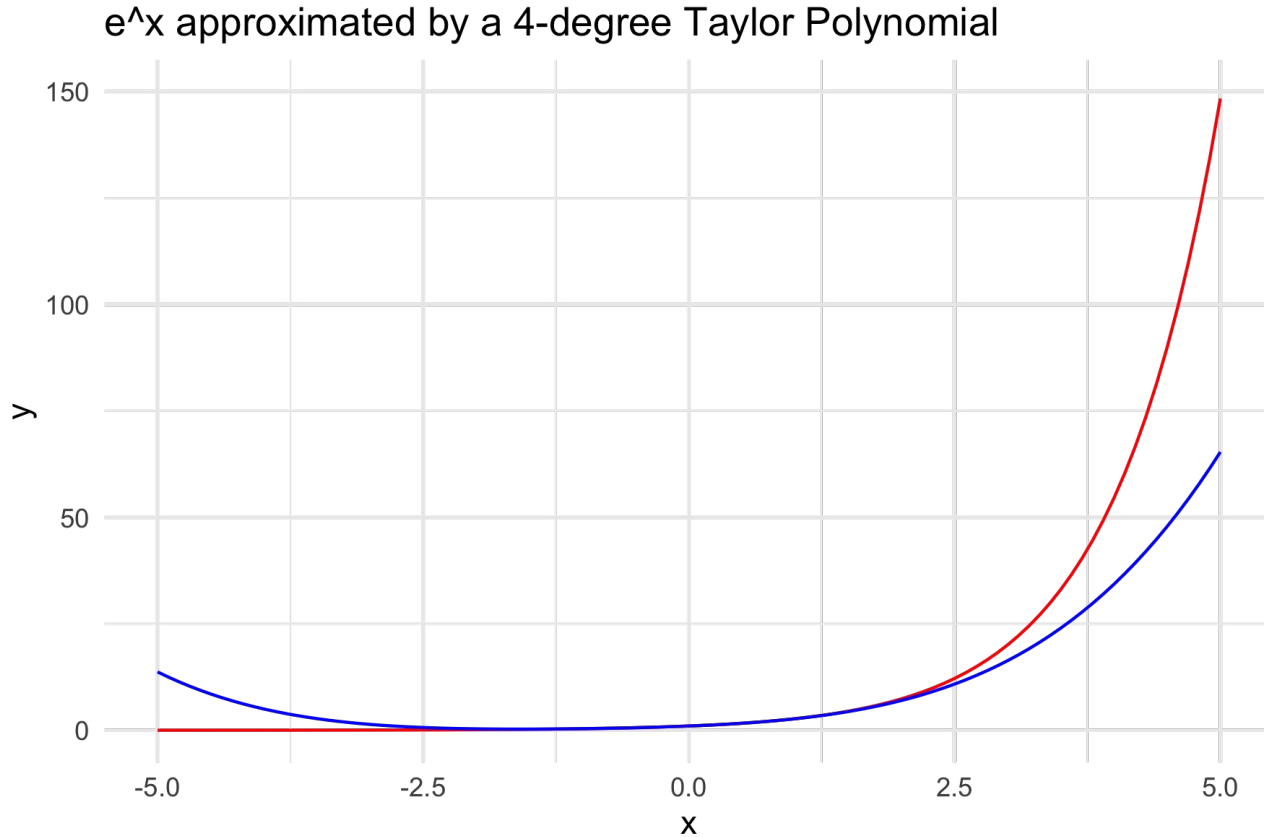
# Taylor Series (visually)



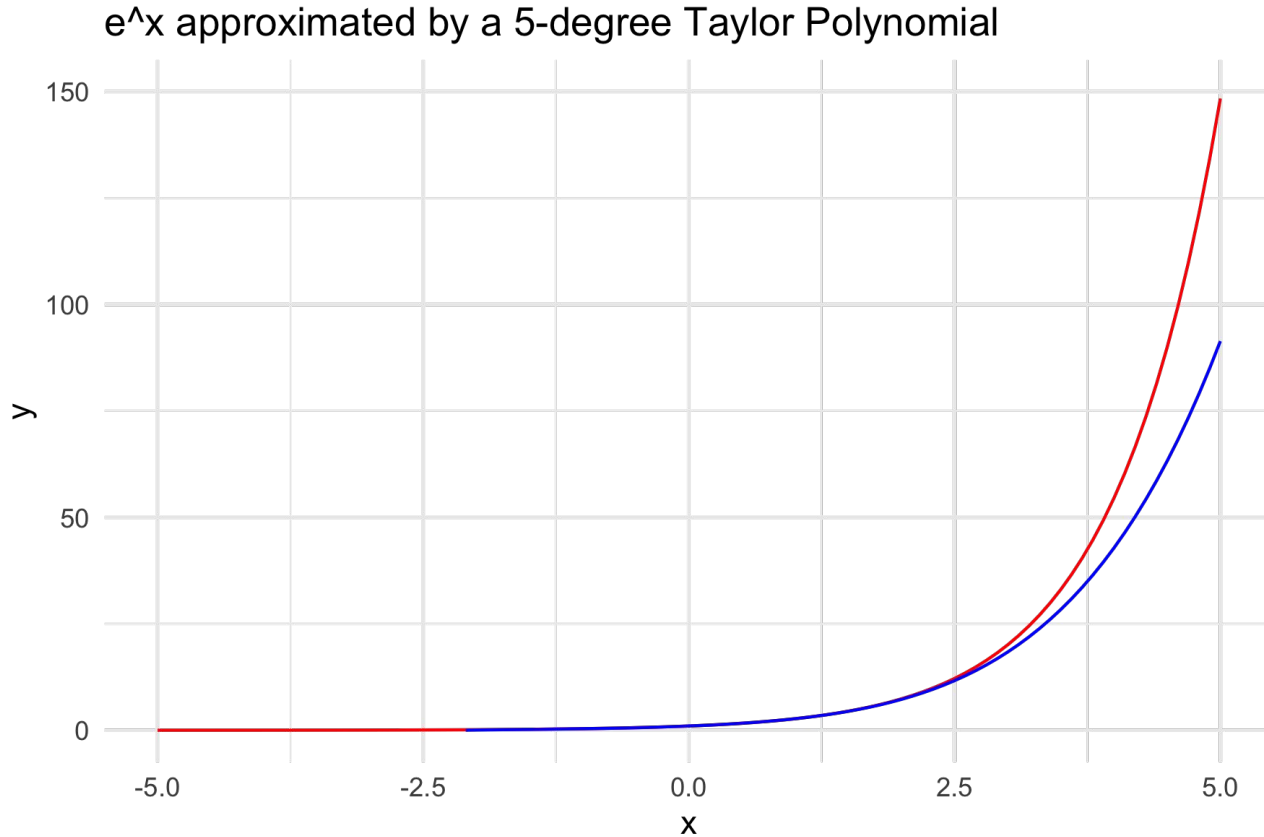
# Taylor Series (visually)



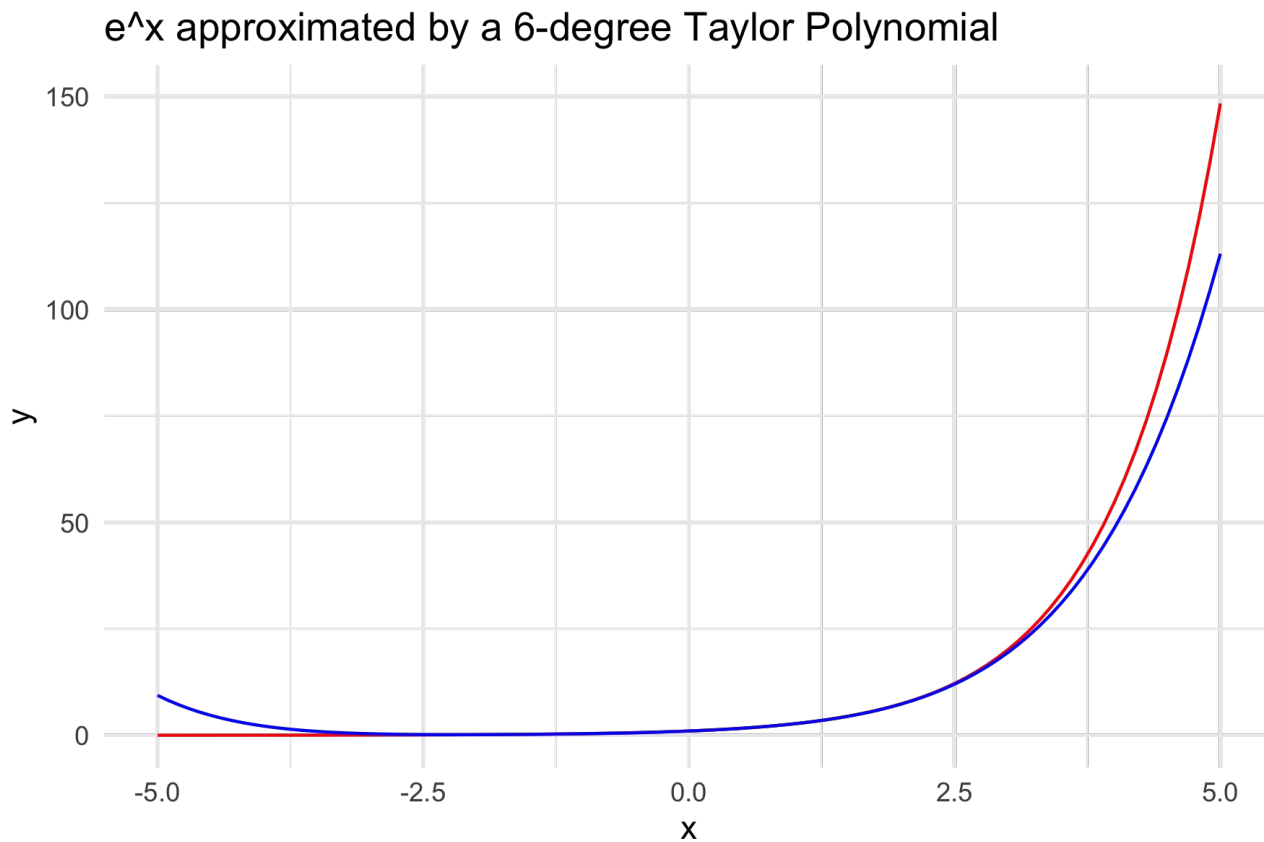
# Taylor Series (visually)



# Taylor Series (visually)



# Taylor Series (visually)





# Taylor Series

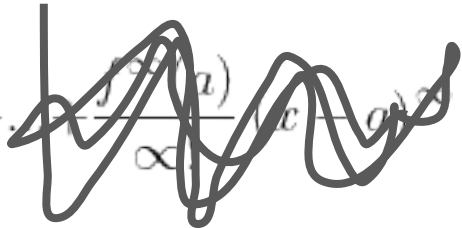
- Is there a simpler function that approximates  $f(x)$ ?

Yes!

$$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!} (x-a)^n = f(a) + \frac{f'(a)}{1} (x-a) + \frac{f''(a)}{2!} (x-a)^2 + \dots + \frac{f^{\infty}(a)}{\infty!} (x-a)^{\infty}$$

# Taylor Series

$$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!} (x-a)^n = f(a) + \frac{f'(a)}{1} (x-a) + \frac{f''(a)}{2!} (x-a)^2 + \dots + \frac{f^{\infty}(a)}{\infty!} (x-a)^{\infty}$$

$$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!} (x-a)^n = \underbrace{f(a) + \frac{f'(a)}{1} (x-a) + \frac{f''(a)}{2!} (x-a)^2}_{\text{use just this for approximation of } f(x)} + \dots$$


# Taylor Series

- Taylor Series are ways to **re-write a function** using its derivatives
- A Taylor Series is the sum of an **infinite** number of terms
- We can use just a few of these terms if we want an **approximation** of the function

# KL Divergence

$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i) \cdot \frac{\log p(x_i)}{\log q(x_i)}$$

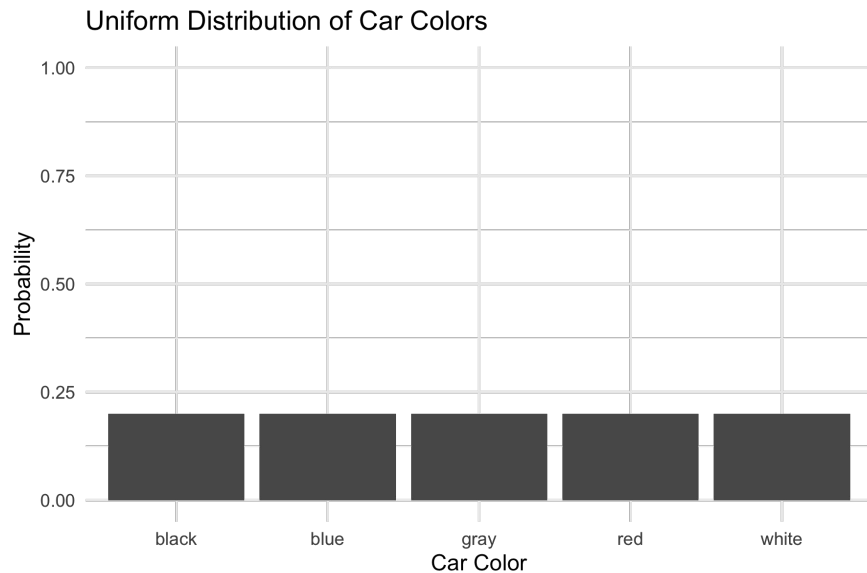
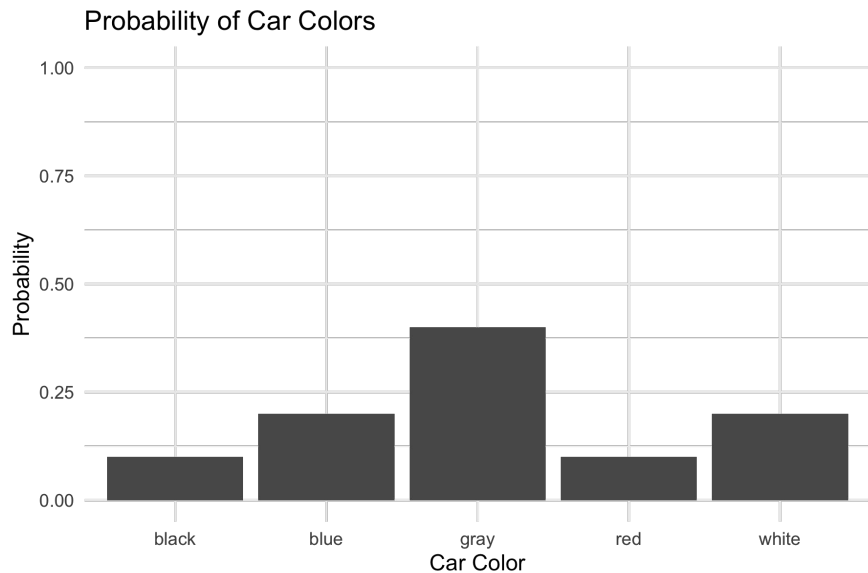
A similarity metric between two distributions **p** and **q**.

How much information is lost when we use **q** to approximate **p**.

# KL Divergence

$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

	black	blue	gray	red	white
<b>p(x)</b>	0.1	0.2	0.4	0.1	0.2
<b>q(x)</b>	0.2	0.2	0.2	0.2	0.2



# KL Divergence

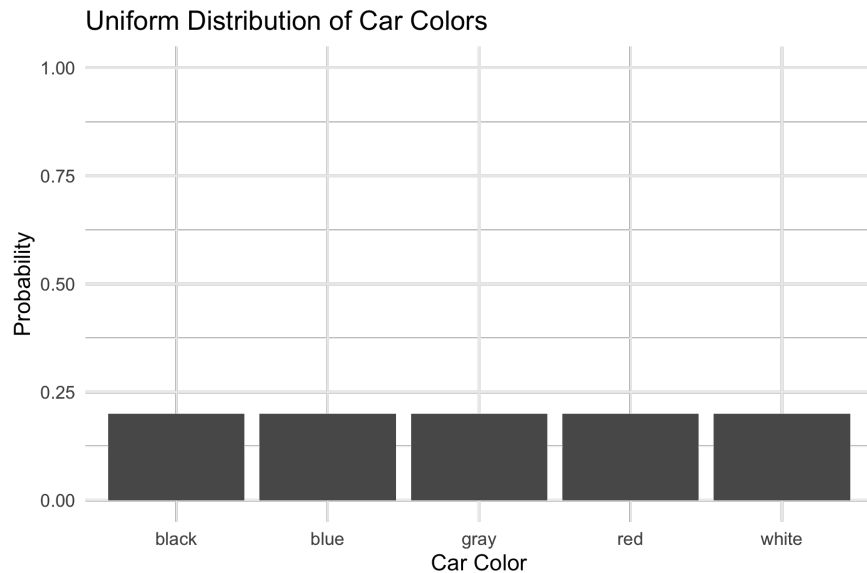
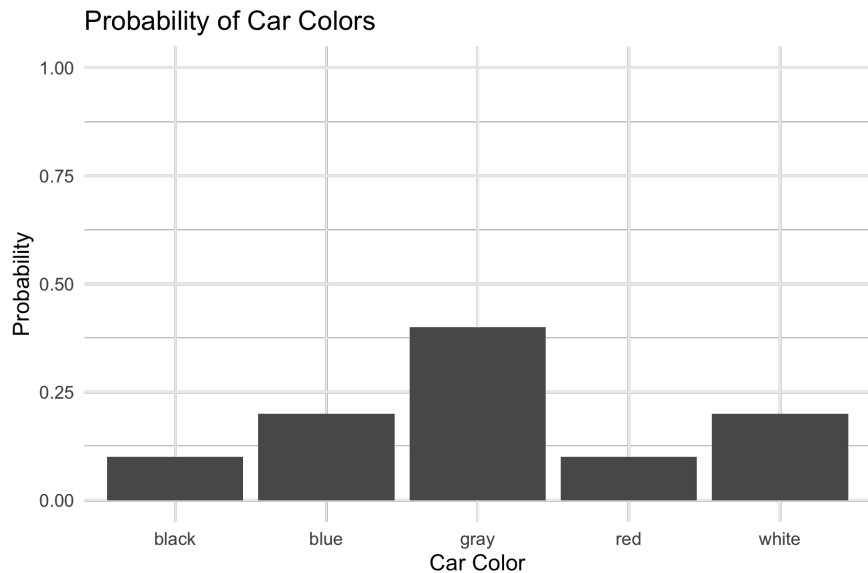
$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

	black	blue	gray	red	white
<b>p(x)</b>	0.1	0.2	0.4	0.1	0.2
<b>q(x)</b>	0.2	0.2	0.2	0.2	0.2

# KL Divergence

$$D_{KL}(p||q) = \sum_{i=1}^N p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

	black	blue	gray	red	white
<b>p(x)</b>	0.1	0.2	0.4	0.1	0.2
<b>q(x)</b>	0.2	0.2	0.2	0.2	0.2



# KL Divergence

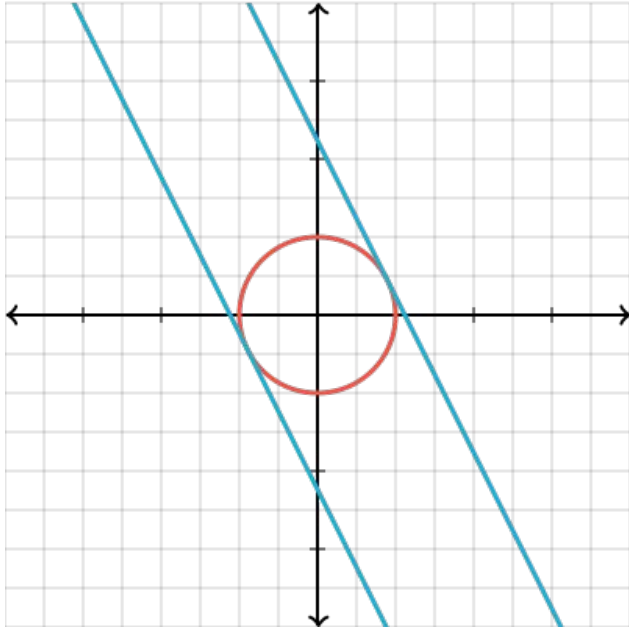
- KL Divergence is a **non-symmetric** similarity metric that measures the similarity between two distributions



# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

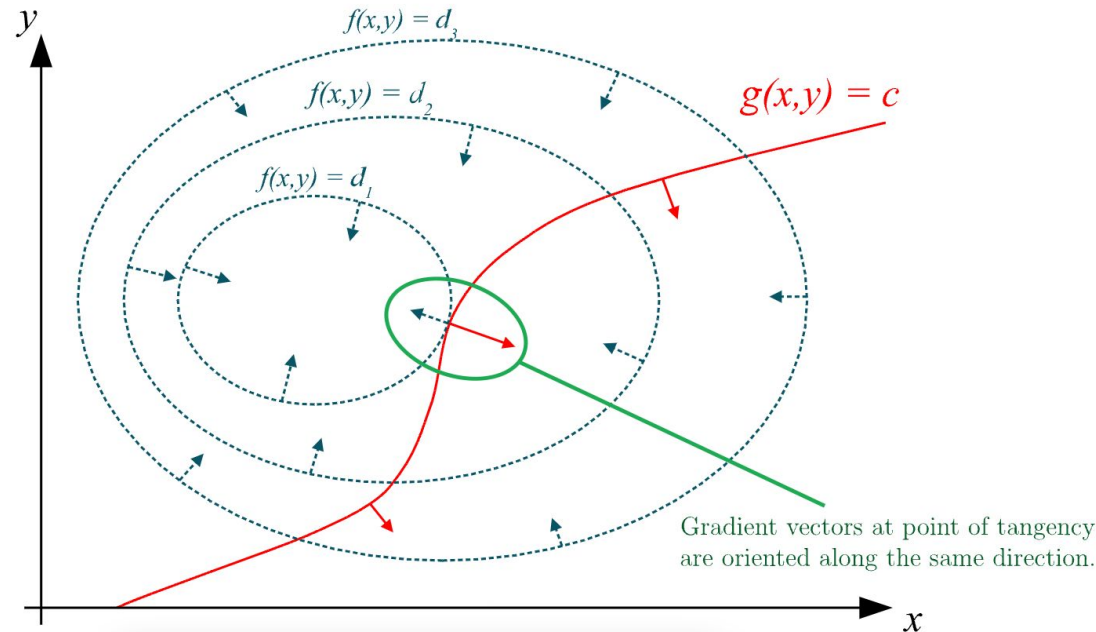
1



# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

2



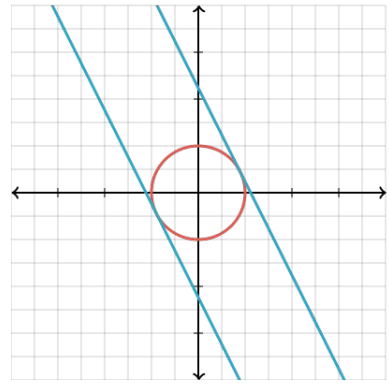
# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

$$3 \quad \nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0)$$

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$



$$3 \quad \nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0)$$

$$f(x, y) = 2x + y$$

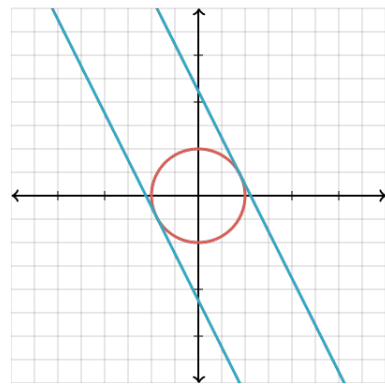
$$g(x, y) = x^2 + y^2 = 1$$

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} 2x + y \\ \frac{\partial}{\partial y} 2x + y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\nabla g(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} x^2 + y^2 \\ \frac{\partial}{\partial y} x^2 + y^2 \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$



$$\textcircled{3} \quad \nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0)$$

$$f(x, y) = 2x + y$$

$$g(x, y) = x^2 + y^2 = 1$$

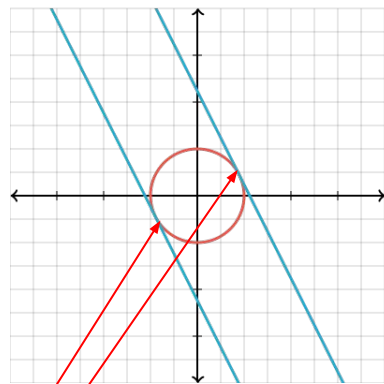
$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} 2x + y \\ \frac{\partial}{\partial y} 2x + y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\nabla g(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} x^2 + y^2 \\ \frac{\partial}{\partial y} x^2 + y^2 \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \lambda_0 \begin{bmatrix} 2x_0 \\ 2y_0 \end{bmatrix}$$

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$



$$\textcircled{3} \quad \nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0)$$

$$f(x, y) = 2x + y$$

$$g(x, y) = x^2 + y^2 = 1$$

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} 2x + y \\ \frac{\partial}{\partial y} 2x + y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\nabla g(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} x^2 + y^2 \\ \frac{\partial}{\partial y} x^2 + y^2 \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \lambda_0 \begin{bmatrix} 2x_0 \\ 2y_0 \end{bmatrix}$$

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

4  $\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$

$\mathcal{L}(x, y, \lambda) = 2x + y - \lambda(x^2 + y^2 - 1)$  ← For example

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

4

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$$

$$\mathcal{L}_\lambda(x, y, \lambda) = \frac{\partial \mathcal{L}}{\partial \lambda} = 0 - (g(x, y) - c) = -g(x, y) + c$$

$$\mathcal{L}_\lambda(x, y, \lambda) = -g(x, y) + c = 0 \text{ when } g(x, y) = c$$



# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

4

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$$

$$\mathcal{L}_\lambda(x, y, \lambda) = \frac{\partial \mathcal{L}}{\partial \lambda} = 0 - (g(x, y) - c) = -g(x, y) + c$$

$$\mathcal{L}_\lambda(x, y, \lambda) = -g(x, y) + c = 0 \text{ when } g(x, y) = c$$

$$\mathcal{L}_x(x, y, \lambda) = 0$$

$$\frac{\partial}{\partial x}(f(x, y) - \lambda(g(x, y) - c)) = 0$$

$$f_x(x, y) - \lambda g_x(x, y) = 0$$

$$f_x(x, y) = \lambda g_x(x, y)$$

# Lagrangians and Constrained Optimization

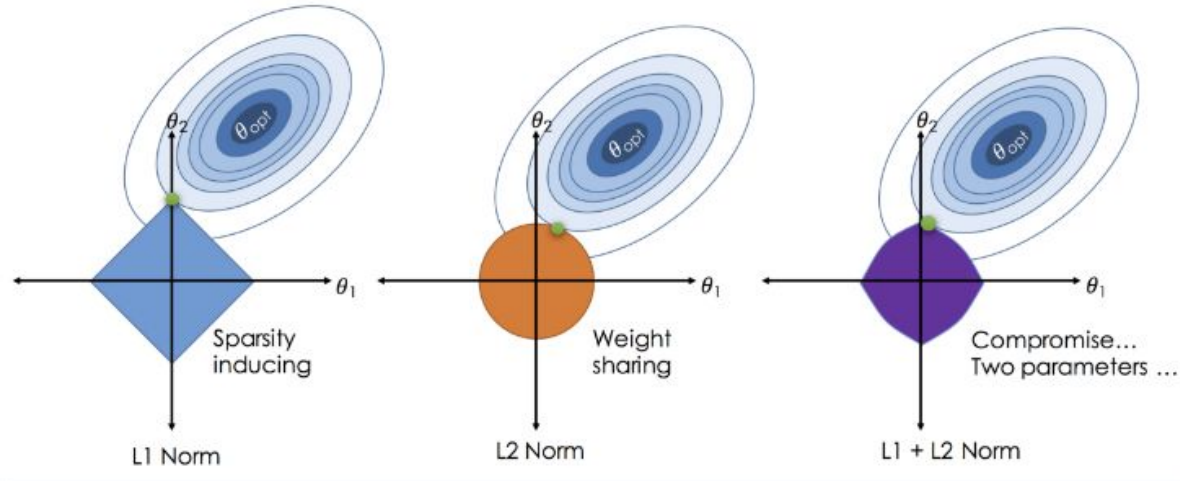
Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$

4

$$\nabla \mathcal{L} = \begin{bmatrix} \frac{\partial}{\partial x}(2x + y - \lambda(x^2 + y^2 - 1)) \\ \frac{\partial}{\partial y}(2x + y - \lambda(x^2 + y^2 - 1)) \\ \frac{\partial}{\partial \lambda}(2x + y - \lambda(x^2 + y^2 - 1)) \end{bmatrix} = \begin{bmatrix} 2 - 2\lambda x \\ 1 - 2\lambda y \\ -x^2 - y^2 + 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

# Lagrangians and Constrained Optimization

Optimize  $f(x, y, z)$  subject to  $g(x, y, z) = k$



# Lagrangians

- Lagrangians (and Lagrangian Multipliers) are a way to do **constrained optimization**
- Constrained Optimization is finding the min (or max) of a function within some boundary
- Because we know that the **optima** of a function (with constraints) occur at places where the two functions (function and constraint) are **tangent**, we can find where their **gradients are parallel** and that will be an optima!
- **Lagrangians** are a clever way of solving for these points by shoving them all into an equation and setting the gradient to 0
- The variable  $\lambda$  is called a Lagrangian Multiplier and represents the scaling factor between the two gradients
  - Can be interpreted as how much your function will change as you change your constraint