Mathematics for Machine Learning

Tanujit Chakraborty

@ Sorbonne

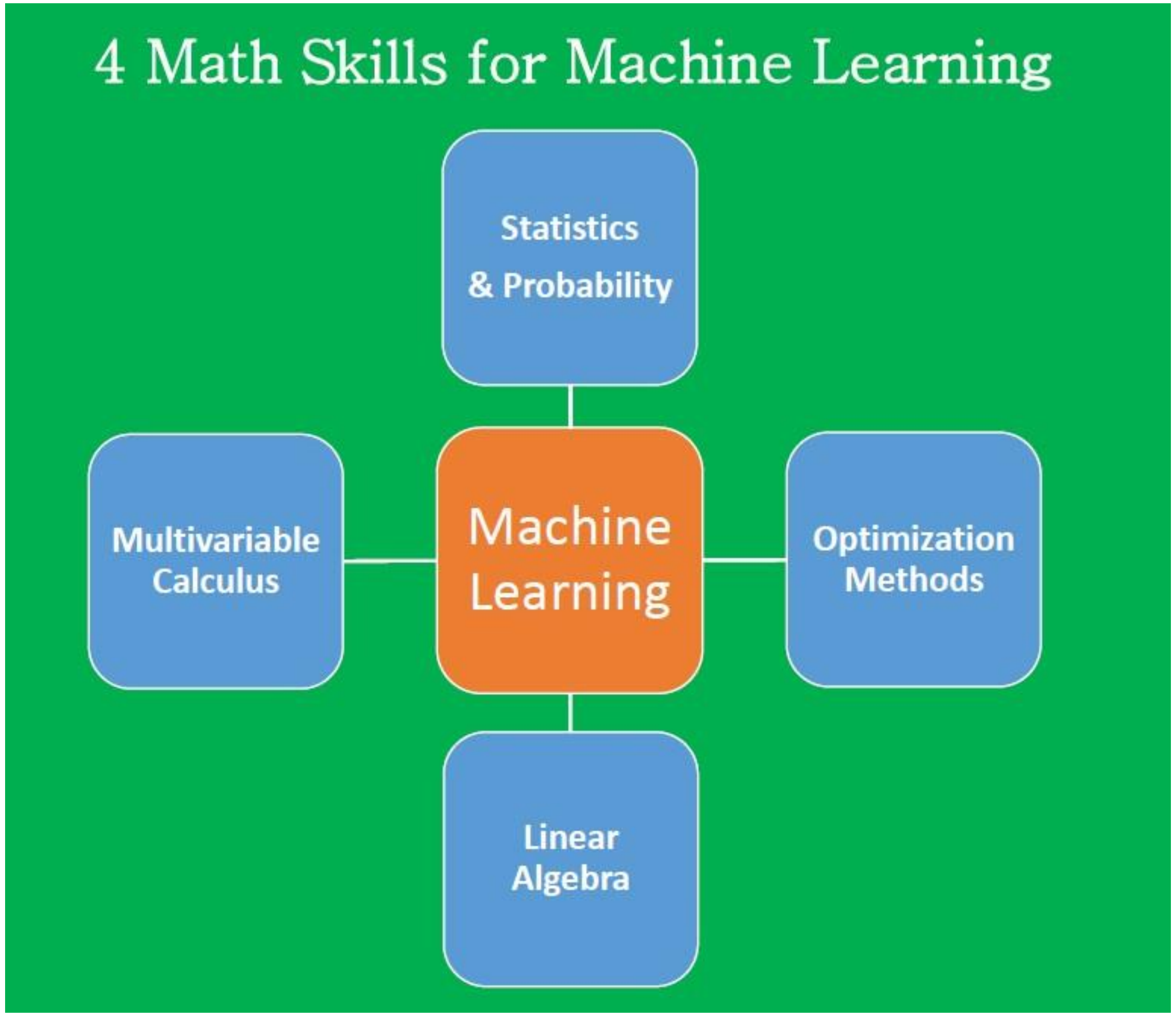
Webpage: https://www.ctanujit.org/

Lecture Outline

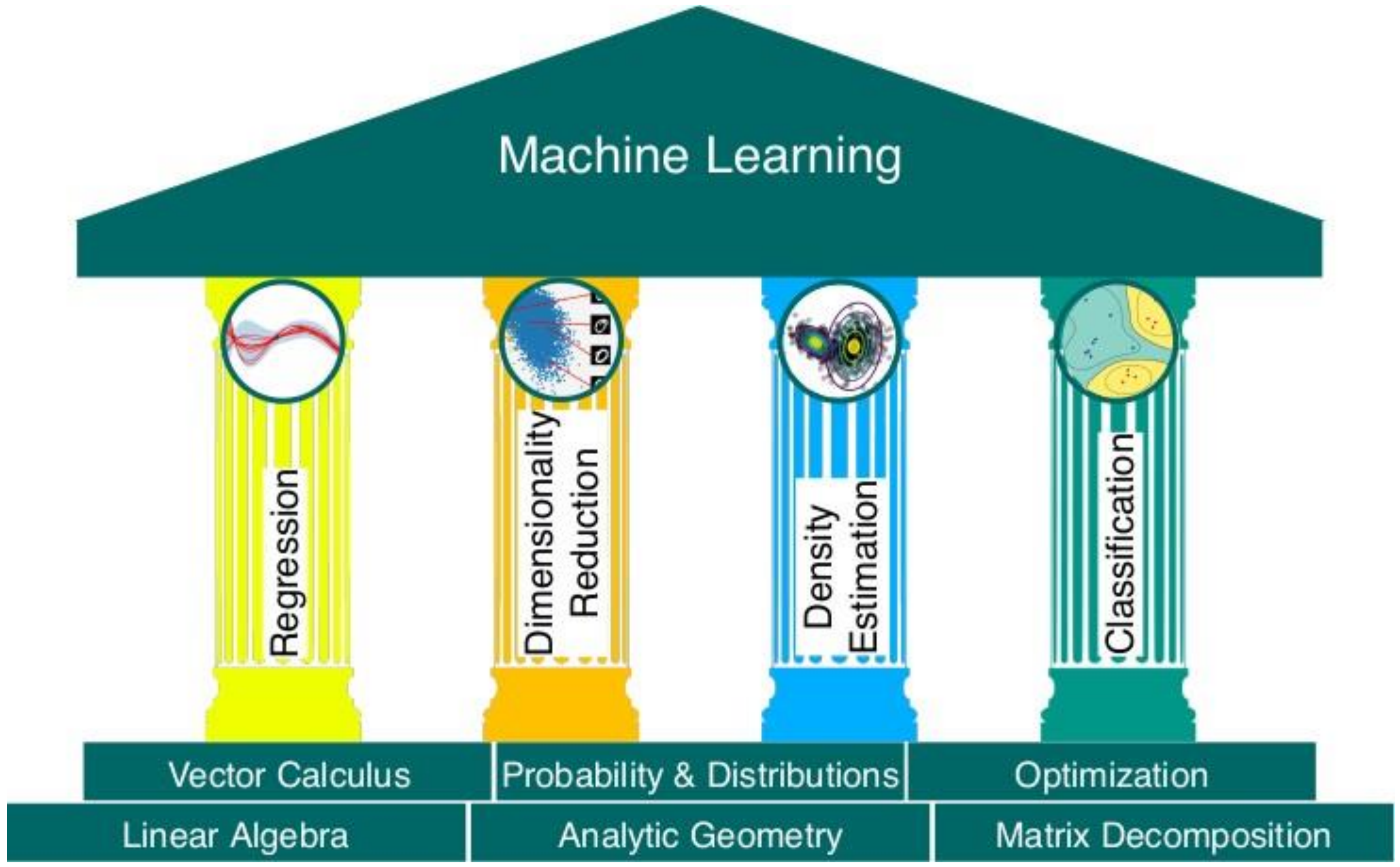
- Introduction and Notation
- Linear Algebra
  - Vectors
  - Matrices
  - Eigen decomposition
- Differential Calculus
- Optimization Algorithms
- Probability
  - Random variables
  - Probability distributions
- Information Theory
  - KL Divergence and Entropy
  - Maximum Likelihood Estimation

* Most of the slides are adapted from MML Book and several online resources that are acknowledged at the end of the presentation.

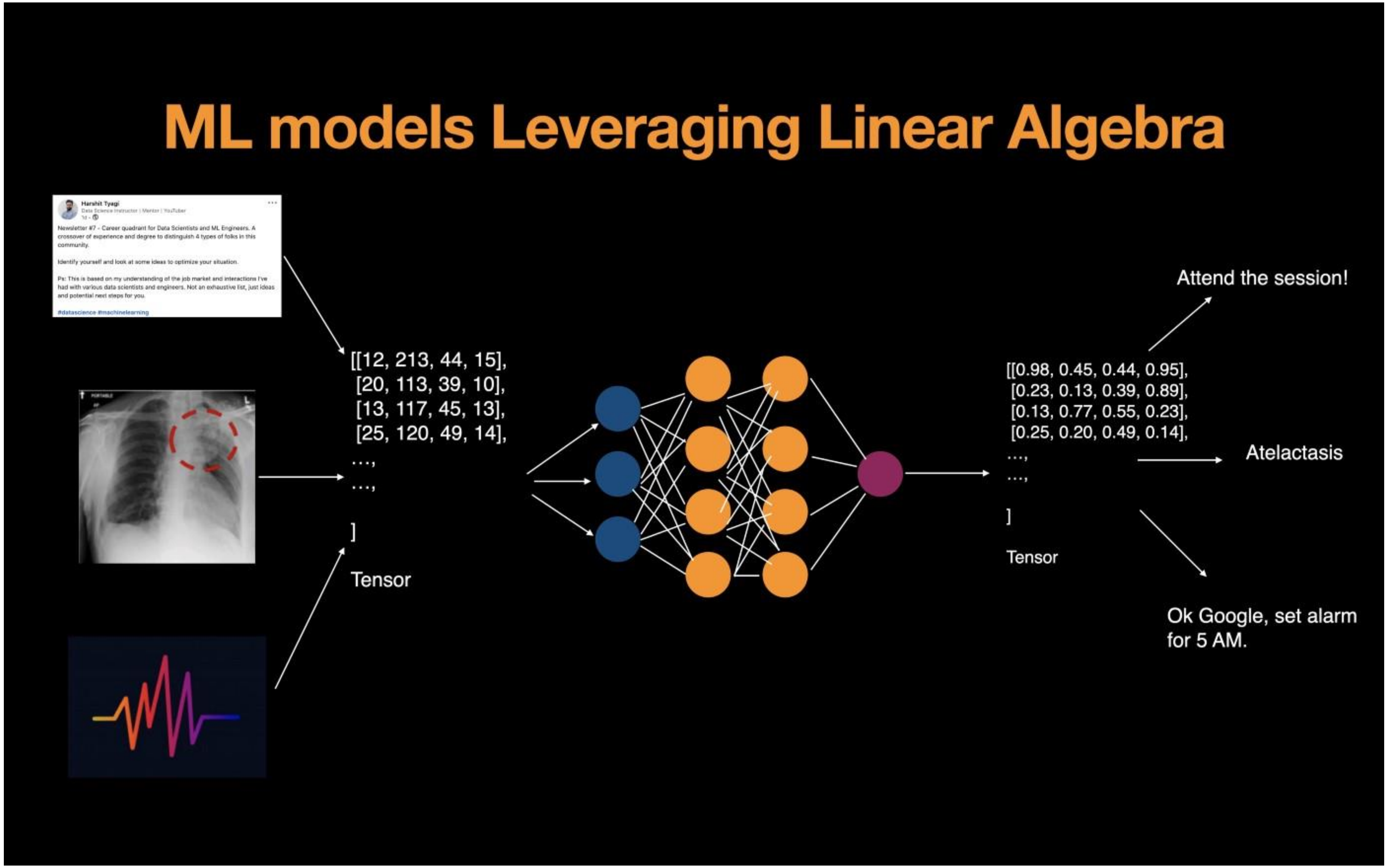
Math for ML


4 Math Skills for Machine Learning
Statistics & Probability
Multivariable Calculus
Machine Learning
Optimization Methods
Linear Algebra

Math for ML


Machine Learning
Regression
Dimensionality Reduction
Density Estimation
Classification
Vector Calculus
Probability & Distributions
Optimization
Linear Algebra
Analytic Geometry
Matrix Decomposition

Math for ML



Harshit Tyagi
Data Science Instructor | Mentor | YouTuber
1d ·

Newsletter #7 - Career quadrant for Data Scientists and ML Engineers. A crossover of experience and degree to distinguish 4 types of folks in this community.

Identify yourself and look at some ideas to optimize your situation.

Ps: This is based on my understanding of the job market and interactions I've had with various data scientists and engineers. Not an exhaustive list, just ideas and potential next steps for you.

#datascience #machinelearning

Notation

a, b, c	Scalar (integer or real)
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Vector (bold-font, lower case)
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Matrix (bold-font, upper-case)
$\boldsymbol{\mathsf{A}}, \boldsymbol{\mathsf{B}}, \boldsymbol{\mathsf{C}}$	Tensor (bold-font, upper-case)
X, Y, Z	Random variable (normal font, upper-case)
$a \in \mathcal{A}$	Set membership: a is member of set $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	Cardinality: number of items in set $\mathcal{A}$
$\lVert\mathbf{v}\rVert$	Norm of vector $\mathbf{v}$
$\mathbf{u} \cdot \mathbf{v}$ or $\langle \mathbf{u}, \mathbf{v} \rangle$	Dot product of vectors $\mathbf{u}$ and $\mathbf{v}$
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Real numbers space of dimension n
$y = f(x)$ or $x \mapsto f(x)$	Function (map): assign a unique value $f(x)$ to each input value x
$f: \mathbb{R}^n \to \mathbb{R}$	Function (map): map an n-dimensional vector into a scalar

Notation

$\mathbf{A} \odot \mathbf{B}$	Element-wise product of matrices $\mathbf{A}$ and $\mathbf{B}$
$\mathbf{A}^{\dagger}$	Pseudo-inverse of matrix $\mathbf{A}$
$\frac{d^n f}{dx^n}$	n-th derivative of function f with respect to x
$\nabla_{\mathbf{x}} f(\mathbf{x})$	Gradient of function f with respect to $\mathbf{x}$
$\mathbf{H}_f$	Hessian matrix of function f
$X \sim P$	Random variable X has distribution P
$P(X\|Y)$	Probability of X given Y
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$\mathbb{E}_{X \sim P}[f(X)]$	Expectation of $f(X)$ with respect to $P(X)$
$\mathrm{Var}\big(f(X)\big)$	Variance of $f(X)$
$\mathrm{Cov}\big(f(X), g(Y)\big)$	Covariance of $f(X)$ and $g(Y)$
$\mathrm{corr}(X, Y)$	Correlation coefficient for X and Y
$D_{KL}(P\|\|Q)$	Kullback-Leibler divergence for distributions P and Q
$CE(P, Q)$	Cross-entropy for distributions P and Q

Vectors

- *Vector* definition
  - **Computer science**: *vector* is a one-dimensional array of ordered real-valued scalars
  - **Mathematics**: *vector* is a quantity possessing both magnitude and direction, represented by an arrow indicating the direction, and the length of which is proportional to the magnitude
- Vectors are written in column form or in row form
  - Denoted by bold-font lower-case letters

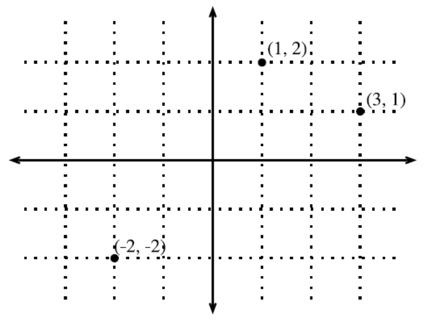
$$\mathbf{x} = \begin{bmatrix} 1 \\ 7 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 1 & 7 & 0 & 1 \end{bmatrix}^T$$

- For a general form vector with n elements, the vector lies in the n-dimensional space $\mathbf{x} \in \mathbb{R}^n$

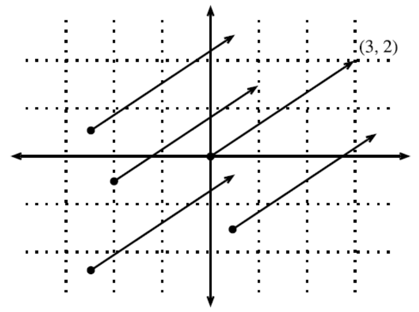
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Geometry of Vectors

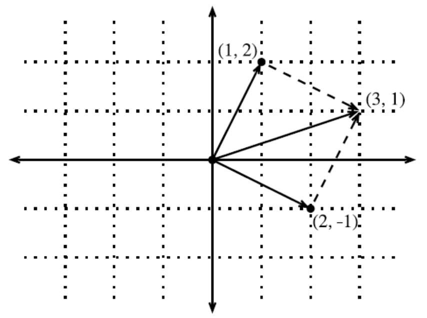
- First interpretation of a vector: point in space
  - E.g., in 2D we can visualize the data points with respect to a coordinate origin



- Second interpretation of a vector: direction in space
  - E.g., the vector $\vec{\mathbf{v}} = [3, 2]^T$ has a direction of 3 steps to the right and 2 steps up
  - The notation $\vec{\mathbf{v}}$ is sometimes used to indicate that the vectors have a direction
  - All vectors in the figure have the same direction



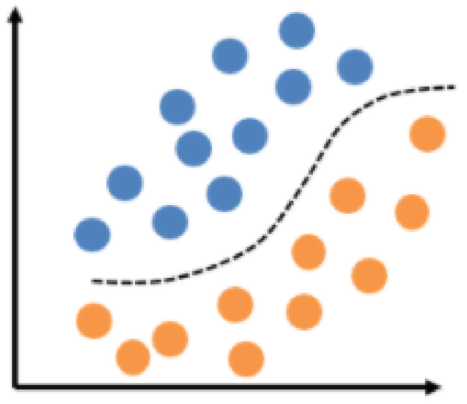
- Vector addition
  - We add the coordinates, and follow the directions given by the two vectors that are added

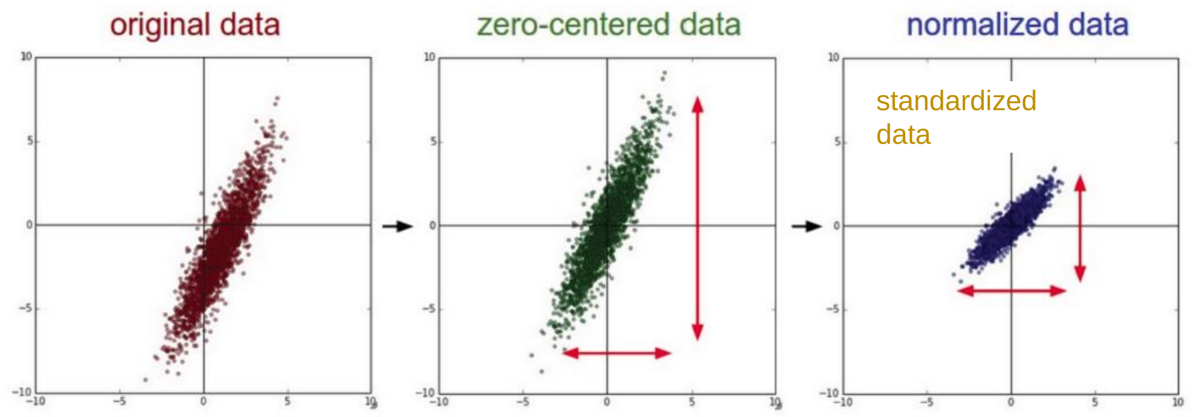


Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/geometry-linear-algebraic-ops.html#geometry-of-vectors

Geometry of Vectors

- The geometric interpretation of vectors as points in space allow us to consider a training set of input examples in ML as a collection of points in space
  - Hence, classification can be viewed as discovering how to separate the clusters of points belonging to different classes (left picture)
    - Rather than distinguishing images containing cars, planes, buildings, for example
  - Or it can help to visualize zero-centering and standardization of training data (right figure)

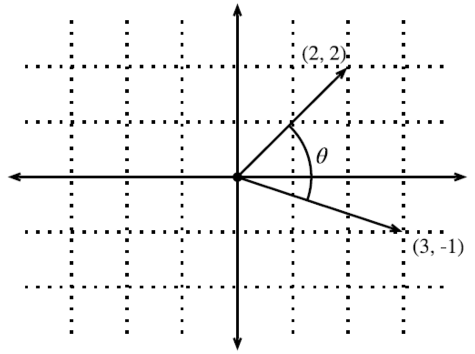




Dot Product and Angles

- *Dot product* of vectors, $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \sum_i u_i \cdot v_i$
  - It is also referred to as inner product, or scalar product of vectors
  - The dot product $\mathbf{u} \cdot \mathbf{v}$ is also often denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$
- The dot product is a symmetric operation, $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u} = \mathbf{v} \cdot \mathbf{u}$

- Geometric interpretation of a dot product: angle between two vectors
  - i.e., dot product $\mathbf{v} \cdot \mathbf{w}$ over the norms of the vectors is $\cos(\theta)$



$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\| cos(\theta) \qquad \cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

- If two vectors are orthogonal: $\theta = 90°$, i.e., $\cos(\theta) = 0$, then $\mathbf{u} \cdot \mathbf{v} = 0$
- Also, in ML the term $\cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$ is sometimes employed as a measure of closeness of two vectors/data instances, and it is referred to as cosine similarity

Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/geometry-linear-algebraic-ops.html#geometry-of-vectors

Norm of a Vector

- A vector *norm* is a function that maps a vector to a scalar value
  - E.g., the norm can be a measure of the size of the vector
- The norm f should satisfy the following properties:
  - Scaling: $f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$
  - Triangle inequality: $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$
  - Must be non-negative: $f(\mathbf{x}) \geq 0$
- The general ℓ_p norm of a vector $\mathbf{x}$ is obtained as: $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$
  - On next page we will review common norms, obtained for $p = 1, 2$, and ∞

Norm of a Vector

- For $p = 2$, we have ℓ_2 norm
  - Also called **Euclidean norm**
  - It is the most often used norm
  - ℓ_2 norm is often denoted just as $\|\mathbf{x}\|$ with the subscript 2 omitted

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$

- For $p = 1$, we have ℓ_1 norm
  - Uses the absolute values of the elements
  - Discriminate between zero and non-zero elements

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

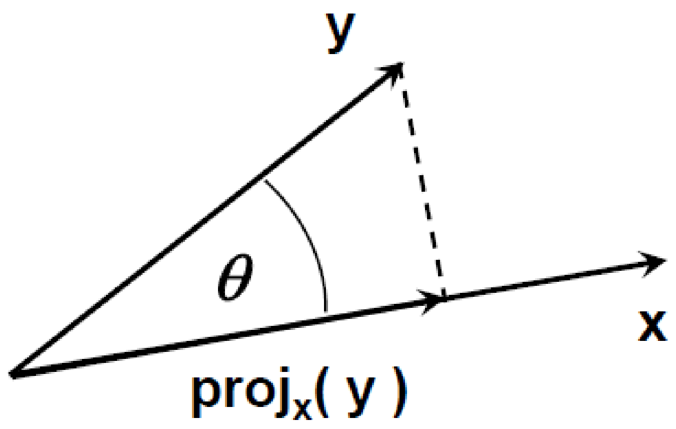
- For $p = \infty$, we have ℓ_∞ norm
  - Known as **infinity norm**, or **max norm**
  - Outputs the absolute value of the largest element

$$\|\mathbf{x}\|_\infty = \max_i |x_i|$$

- ℓ_0 norm outputs the number of non-zero elements
  - It is not an ℓ_p norm, and it is not really a norm function either (it is incorrectly called a norm)

Vector Projection

- *Orthogonal projection* of a vector **y** onto vector **x**
  - The projection can take place in any space of dimensionality ≥ 2
  - The unit vector in the direction of **x** is $\frac{\mathbf{x}}{\|\mathbf{x}\|}$
  - A unit vector has norm equal to 1
  - The length of the projection of **y** onto **x** is $\|\mathbf{y}\| \cdot cos(\theta)$
  - The orthogonal project is the vector $\mathbf{proj_x}(\mathbf{y})$

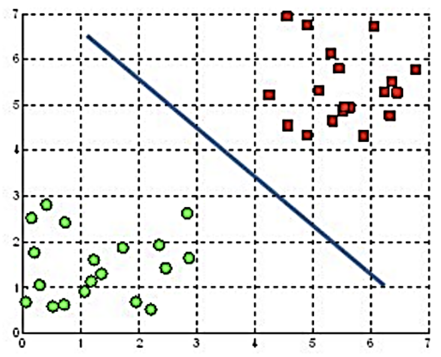


$$\mathbf{proj_x}(\mathbf{y}) = \frac{\mathbf{x} \cdot \|\mathbf{y}\| \cdot cos(\theta)}{\|\mathbf{x}\|}$$

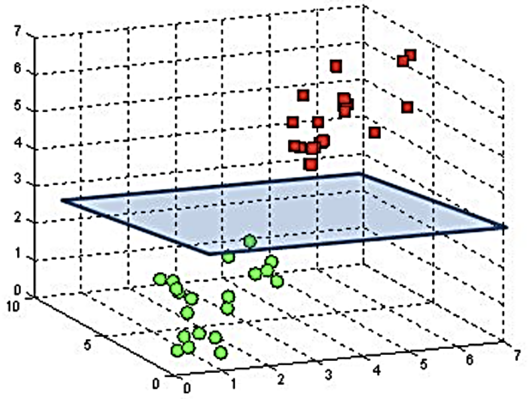
Hyperplanes

- *Hyperplane* is a subspace whose dimension is one less than that of its ambient space
  - In a 2D space, a hyperplane is a straight line (i.e., 1D)
  - In a 3D, a hyperplane is a plane (i.e., 2D)
  - In a d-dimensional vector space, a hyperplane has $d - 1$ dimensions, and divides the space into two half-spaces
- Hyperplane is a generalization of a concept of plane in high-dimensional space
- In ML, hyperplanes are decision boundaries used for linear classification
  - Data points falling on either sides of the hyperplane are attributed to different classes

A hyperplane in $\mathbb{R}^2$ is a line



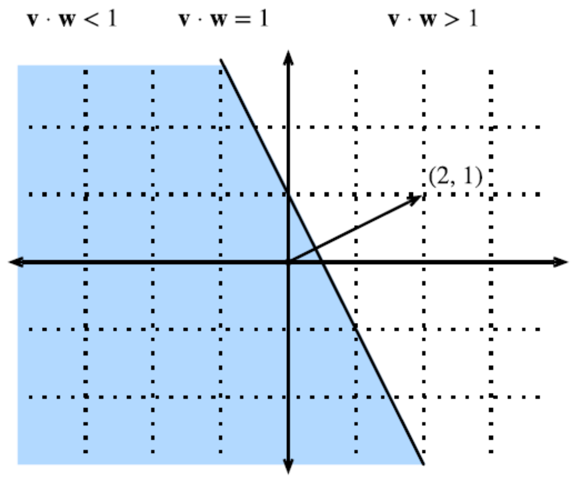
A hyperplane in $\mathbb{R}^3$ is a plane



Picture from: https://kgpdag.wordpress.com/2015/08/12/svm-simplified/

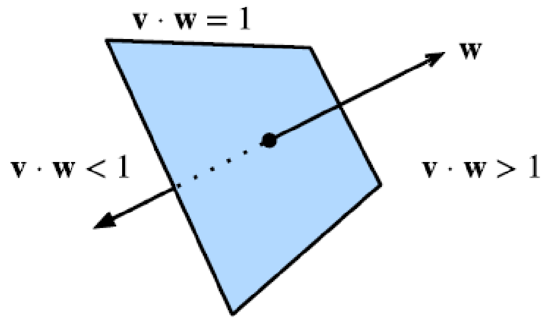
Hyperplanes

- For example, for a given data point $\mathbf{w} = [2, 1]^T$, we can use dot-product to find the hyperplane for which $\mathbf{w} \cdot \mathbf{v} = 1$
  - The solution to $\mathbf{w} \cdot \mathbf{v} = 1$ is the set of points that lay on the line that is orthogonal to the vector $\mathbf{w}$ (the line is $2x + y = 1$)



$\mathbf{v} \cdot \mathbf{w} < 1$
$\mathbf{v} \cdot \mathbf{w} = 1$
$\mathbf{v} \cdot \mathbf{w} > 1$
(2, 1)

- In a 3D space, the points that satisfy $\mathbf{w} \cdot \mathbf{v} = 1$ lay on a plane that is orthogonal to the vector $\mathbf{w}$
  - The inequalities $\mathbf{w} \cdot \mathbf{v} > 1$ and $\mathbf{w} \cdot \mathbf{v} < 1$ define the two subspaces that are created by the plane
  - The same concept applies to high-dimensional spaces as well



$\mathbf{v} \cdot \mathbf{w} = 1$
$\mathbf{w}$
$\mathbf{v} \cdot \mathbf{w} < 1$
$\mathbf{v} \cdot \mathbf{w} > 1$

Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/geometry-linear-algebraic-ops.html#hyperplanes

Matrices

- *Matrix* is a rectangular array of real-valued scalars arranged in m horizontal rows and n vertical columns
  - Each element a_{ij} belongs to the i^{th} row and j^{th} column
  - The elements are denoted a_{ij} or $\mathbf{A}_{ij}$ or $[\mathbf{A}]_{ij}$ or $\mathbf{A}(\boldsymbol{i},\boldsymbol{j})$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- For the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the size (dimension) is $m \times n$ or (m, n)
  - Matrices are denoted by bold-font upper-case letters

Matrices

- Addition or subtraction

$$(\mathbf{A} \pm \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \pm \mathbf{B}_{i,j}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{bmatrix}$$

- Scalar multiplication

$$(c\mathbf{A})_{i,j} = c \cdot \mathbf{A}_{i,j}$$

$$2 \cdot \begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2\cdot 1 & 2\cdot 8 & 2\cdot -3 \\ 2\cdot 4 & 2\cdot -2 & 2\cdot 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$$

- Matrix multiplication

$$(\mathbf{AB})_{i,j} = \mathbf{A}_{i,1}\mathbf{B}_{1,j} + \mathbf{A}_{i,2}\mathbf{B}_{2,j} + \cdots + \mathbf{A}_{i,n}\mathbf{B}_{n,j}$$

  - Defined only if the number of columns of the left matrix is the same as the number of rows of the right matrix
  - Note that **AB** $\neq$ **BA**

$$\begin{bmatrix} \underline{2} & \underline{3} & \underline{4} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \underline{1000} \\ 1 & \underline{100} \\ 0 & \underline{10} \end{bmatrix} = \begin{bmatrix} 3 & \underline{2340} \\ 0 & 1000 \end{bmatrix}$$

Matrices

- ***Transpose*** of the matrix: $\mathbf{A}^T$ has the rows and columns exchanged

$$\left(\mathbf{A}^T\right)_{i,j} = \mathbf{A}_{j,i} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix}$$

  - Some properties

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \qquad \mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$$
$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \boldsymbol{B}^T \qquad \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$$
$$(\mathbf{A}^T)^T = \mathbf{A} \qquad (\mathbf{AB})^T = \boldsymbol{B}^T\mathbf{A}^T$$

- ***Square matrix***: has the same number of rows and columns

- ***Identity matrix*** ($\mathbf{I}_n$): has ones on the main diagonal, and zeros elsewhere

  - E.g.: identity matrix of size 3×3 :

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrices

- *Determinant* of a matrix, denoted by $\det(\mathbf{A})$ or $|\mathbf{A}|$, is a real-valued scalar encoding certain properties of the matrix
  - E.g., for a matrix of size 2×2: $$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$
  - For larger-size matrices the determinant of a matrix id calculated as $$\det(\mathbf{A}) = \sum_j a_{ij}(-1)^{i+j} det\left(\mathbf{A}_{(i,j)}\right)$$
  - In the above, $\mathbf{A}_{(i,j)}$ is a minor of the matrix obtained by removing the row and column associated with the indices i and j
- *Trace* of a matrix is the sum of all diagonal elements $$\mathrm{Tr}(\mathbf{A}) = \sum_i a_{ii}$$
- A matrix for which $\mathbf{A} = \mathbf{A}^T$ is called a *symmetric matrix*

Matrices

- Elementwise multiplication of two matrices **A** and **B** is called the *Hadamard product* or *elementwise product*
  - The math notation is $\odot$

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \dots & a_{mn}b_{mn} \end{bmatrix}$$

Matrix-Vector Products

- Consider a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{x} \in \mathbb{R}^n$
- The matrix can be written in terms of its row vectors (e.g., $\mathbf{a}_1^T$ is the first row)

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_m^\top \end{bmatrix}$$

- The matrix-vector product is a column vector of length m, whose i^{th} element is the dot product $\mathbf{a}_i^T \mathbf{x}$

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_m^\top \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{x} \\ \mathbf{a}_2^\top \mathbf{x} \\ \vdots \\ \mathbf{a}_m^\top \mathbf{x} \end{bmatrix}$$

- Note the size: $\mathbf{A}(m \times n) \cdot \mathbf{x}(n \times 1) = \mathbf{A}\mathbf{x}(m \times 1)$

Matrix-Matrix Products

- To multiply two matrices $\mathbf{A} \in \mathbb{R}^{n\times k}$ and $\mathbf{B} \in \mathbb{R}^{k\times m}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{bmatrix}$$

- We can consider the matrix-matrix product as dot-products of rows in **A** and columns in **B**

$$\mathbf{C} = \mathbf{A}\mathbf{B} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^\top\mathbf{b}_1 & \mathbf{a}_1^\top\mathbf{b}_2 & \cdots & \mathbf{a}_1^\top\mathbf{b}_m \\ \mathbf{a}_2^\top\mathbf{b}_1 & \mathbf{a}_2^\top\mathbf{b}_2 & \cdots & \mathbf{a}_2^\top\mathbf{b}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_n^\top\mathbf{b}_1 & \mathbf{a}_n^\top\mathbf{b}_2 & \cdots & \mathbf{a}_n^\top\mathbf{b}_m \end{bmatrix}$$

- Size: $\mathbf{A}(n \times k) \cdot \mathbf{B}(k \times m) = \mathbf{C}(n \times m)$

Linear Dependence

- For the following matrix $\mathbf{B} = \begin{bmatrix} 2 & -1 \\ 4 & -2 \end{bmatrix}$
- Notice that for the two columns $\mathbf{b}_1 = [2, 4]^T$ and $\mathbf{b}_2 = [-1, -2]^T$, we can write $\mathbf{b}_1 = -2 \cdot \mathbf{b}_2$
  - This means that the two columns are linearly dependent
- The weighted sum $a_1\mathbf{b}_1 + a_2\mathbf{b}_2$ is referred to as a linear combination of the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$
  - In this case, a linear combination of the two vectors exist for which $\mathbf{b}_1 + 2 \cdot \mathbf{b}_2 = \mathbf{0}$
- A collection of vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ are *linearly dependent* if there exist coefficients $a_1, a_2, \dots, a_k$ not all equal to zero, so that

$$\sum_{i=1}^{k} a_i \mathbf{v_i} = 0$$

- If there is no linear dependence, the vectors are *linearly independent*

Matrix Rank

- For an $n \times m$ matrix, the *rank* of the matrix is the largest number of linearly independent columns
- The matrix **B** from the previous example has $rank(\mathbf{B}) = 1$, since the two columns are linearly dependent

$$\mathbf{B} = \begin{bmatrix} 2 & -1 \\ 4 & -2 \end{bmatrix}$$

- The matrix **C** below has $rank(\mathbf{C}) = 2$, since it has two linearly independent columns
  - i.e., $\mathbf{c}_4 = -1 \cdot \mathbf{c}_1,\ \mathbf{c}_5 = -1 \cdot \mathbf{c}_3,\ \mathbf{c}_2 = 3 \cdot \mathbf{c}_1 + 3 \cdot \mathbf{c}_3$

$$\mathbf{C} = \begin{bmatrix} 1 & 3 & 0 & -1 & 0 \\ -1 & 0 & 1 & 1 & -1 \\ 0 & 3 & 1 & 0 & -1 \\ 2 & 3 & -1 & -2 & 1 \end{bmatrix}$$

Inverse of a Matrix

- For a square $n \times n$ matrix $\mathbf{A}$ with rank n, $\mathbf{A}^{-1}$ is its *inverse matrix* if their product is an identity matrix $\mathbf{I}$

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

- Properties of inverse matrices

$$\left(\mathbf{A}^{-1}\right)^{-1} = \mathbf{A}$$

$$\left(\mathbf{AB}\right)^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

- If $\det(A) = 0$ (i.e., $\mathrm{rank}(A) < n$), then the inverse does not exist
  - A matrix that is not invertible is called a singular matrix
- Note that finding an inverse of a large matrix is computationally expensive
  - In addition, it can lead to numerical instability
- If the inverse of a matrix is equal to its transpose, the matrix is said to be orthogonal matrix

$$\mathbf{A}^{-1} = \mathbf{A}^{T}$$

Pseudo-Inverse of a Matrix

- *Pseudo-inverse* of a matrix
  - Also known as Moore-Penrose pseudo-inverse

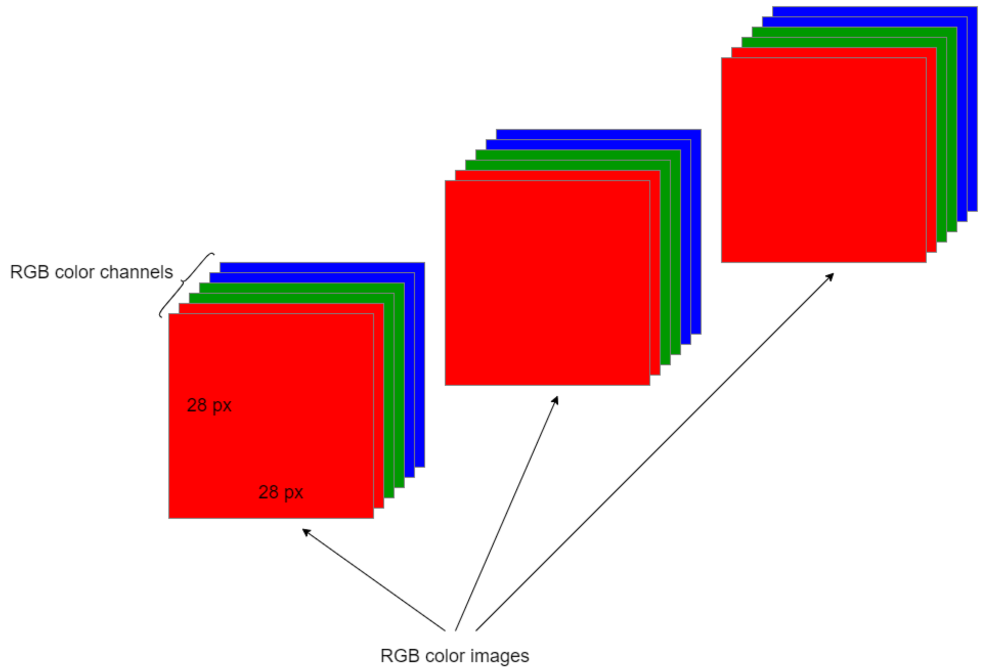
- For matrices that are not square, the inverse does not exist
  - Therefore, a pseudo-inverse is used

- If $m > n$, then the pseudo-inverse is $\mathbf{A}^{\dagger} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{T}$ and $\mathbf{A}^{\dagger}\mathbf{A} = \mathbf{I}$

- If $m < n$, then the pseudo-inverse is $\mathbf{A}^{\dagger} = \mathbf{A}^{T}\left(\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)^{-1}$ and $\mathbf{A}\mathbf{A}^{\dagger} = \mathbf{I}$
  - E.g., for a matrix with dimension $\mathbf{X}_{2\times3}$, a pseudo-inverse can be found of size $\mathbf{X}^{\dagger}_{3\times2}$, so that $\mathbf{X}_{2\times3}\mathbf{X}^{\dagger}_{3\times2} = \mathbf{I}_{2\times2}$

Tensors

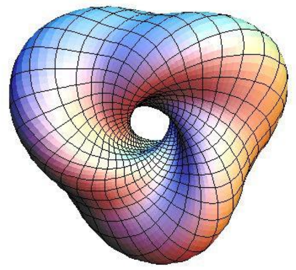
- *Tensors* are n-dimensional arrays of scalars
  - Vectors are first-order tensors, $\mathbf{v} \in \mathbb{R}^n$
  - Matrices are second-order tensors, $\mathbf{A} \in \mathbb{R}^{m \times n}$
  - E.g., a fourth-order tensor is $\mathbf{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$
- Tensors are denoted with upper-case letters of a special font face (e.g., **X**, **Y**, **Z**)
- RGB images are third-order tensors, i.e., as they are 3-dimensional arrays
  - The 3 axes correspond to width, height, and channel
  - E.g., 224 × 224 × 3
  - The channel axis corresponds to the color channels (red, green, and blue)



Manifolds

- Earlier we learned that hyperplanes generalize the concept of planes in high-dimensional spaces
  - Similarly, manifolds can be informally imagined as generalization of the concept of surfaces in high-dimensional spaces

- To begin with an intuitive explanation, the surface of the Earth is an example of a two-dimensional manifold embedded in a three-dimensional space
  - This is true because the Earth looks locally flat, so on a small scale it is like a 2-D plane
  - However, if we keep walking on the Earth in one direction, we will eventually end up back where we started
    - This means that Earth is not really flat, it only looks locally like a Euclidean plane, but at large scales it folds up on itself, and has a different global structure than a flat plane

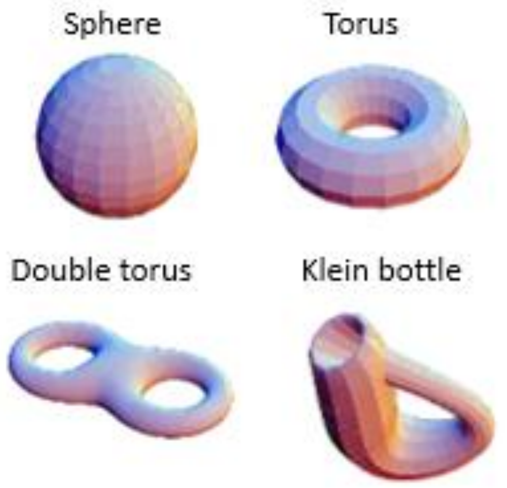


Manifolds

- Manifolds are studied in mathematics under topological spaces

- An n-dimensional ***manifold*** is defined as a topological space with the property that each point has a neighborhood that is homeomorphic to the Euclidean space of dimension n
  - This means that a manifold locally resembles Euclidean space near each point
  - Informally, a Euclidean space is locally smooth, it does not have holes, edges, or other sudden changes, and it does not have intersecting neighborhoods
  - Although the manifolds can have very complex structure on a large scale, resemblance of the Euclidean space on a small scale allows to apply standard math concepts

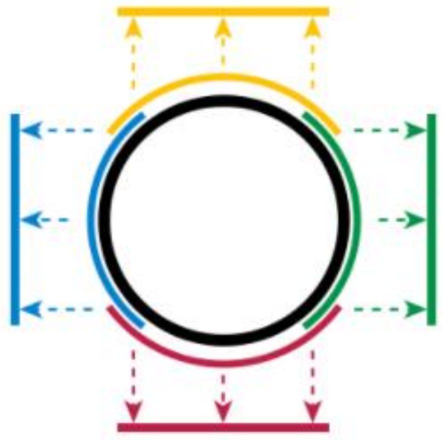
- Examples of 2-dimensional manifolds are shown in the figure
  - The surfaces in the figure have been conveniently cut up into little rectangles that were glued together
  - Those small rectangles locally look like flat Euclidean planes

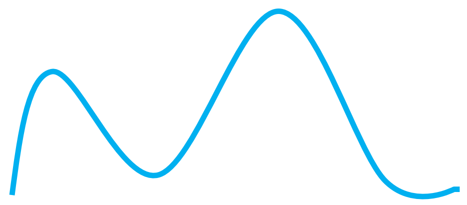


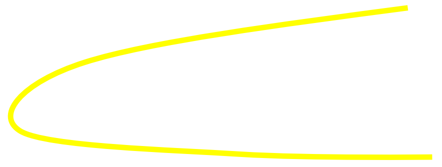
Picture from: http://bjlkeng.github.io/posts/manifolds/

Manifolds

- Examples of one-dimensional manifolds
  - Upper figure: a circle is a l-D manifold embedded in 2-D, where each arc of the circle locally resembles a line segment
  - Lower figures: other examples of 1-D manifolds
  - Note that a number 8 figure is not a manifold because it has an intersecting point (it is not Euclidean locally)
- It is hypothesized that in the real-world, high-dimensional data (such as images) lie on low-dimensional manifolds embedded in the high-dimensional space
  - E.g., in ML, let's assume we have a training set of images with size $224 \times 224 \times 3$ pixels
  - Learning an arbitrary function in such high-dimensional space would be intractable
  - Despite that, all images of the same class ("cats" for example) might lie on a low-dimensional manifold
  - This allows function learning and image classification



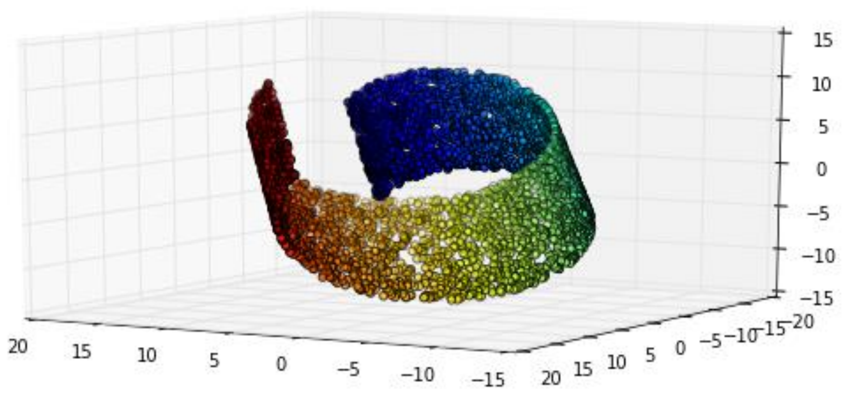


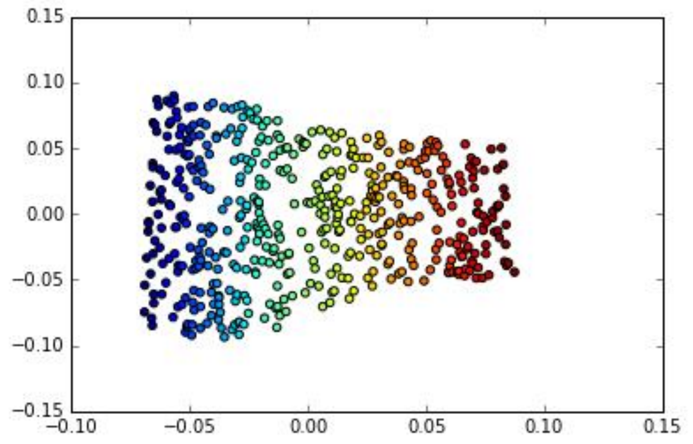


Picture from: http://bjlkeng.github.io/posts/manifolds/

Manifolds

- Example:
  - The data points have 3 dimensions (left figure), i.e., the input space of the data is 3-dimensional
  - The data points lie on a 2-dimensional manifold, shown in the right figure
  - Most ML algorithms extract lower-dimensional data features that enable to distinguish between various classes of high-dimensional input data
    - The low-dimensional representations of the input data are called embeddings





Eigen Decomposition

- *Eigen decomposition* is decomposing a matrix into a set of eigenvalues and eigenvectors
- *Eigenvalues* of a square matrix $\mathbf{A}$ are scalars λ and *eigenvectors* are non-zero vectors $\mathbf{v}$ that satisfy

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- Eigenvalues are found by solving the following equation

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

- If a matrix $\mathbf{A}$ has n linearly independent eigenvectors $\{\mathbf{v}^1, \dots, \mathbf{v}^n\}$ with corresponding eigenvalues $\{\lambda_1, \dots, \lambda_n\}$, the eigen decomposition of $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

  - Columns of the matrix $\mathbf{V}$ are the eigenvectors, i.e., $\mathbf{V} = [\mathbf{v}^1, \dots, \mathbf{v}^n]$
  - $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues, i.e., $\mathbf{\Lambda} = [\lambda_1, \dots, \lambda_n]$

- To find the inverse of the matrix A, we can use $\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^{-1}$
  - This involves simply finding the inverse $\mathbf{\Lambda}^{-1}$ of a diagonal matrix

Eigen Decomposition

- Decomposing a matrix into eigenvalues and eigenvectors allows to analyze certain properties of the matrix
  - If all eigenvalues are positive, the matrix is positive definite
  - If all eigenvalues are positive or zero-valued, the matrix is positive semidefinite
  - If all eigenvalues are negative or zero-values, the matrix is negative semidefinite
    - Positive semidefinite matrices are interesting because they guarantee that $\forall \mathbf{x}, \mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$

- Eigen decomposition can also simplify many linear-algebraic computations
  - The determinant of A can be calculated as

$$\det(\mathbf{A}) = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$$

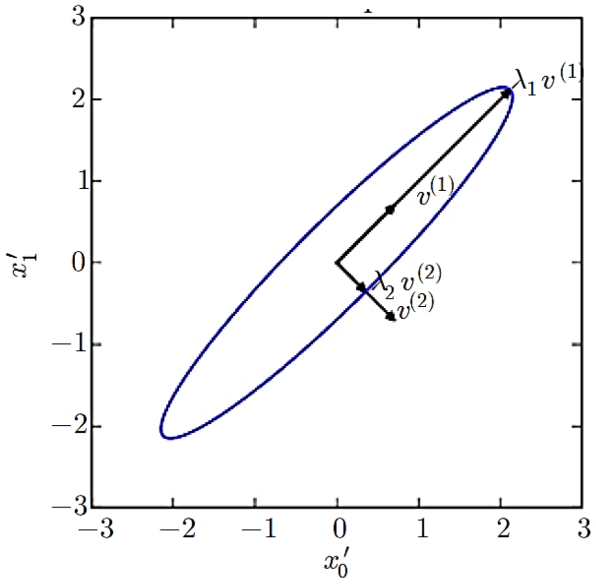
  - If any of the eigenvalues are zero, the matrix is singular (it does not have an inverse)

- However, eigen decomposition is defined only for square matrices
  - Also, in some cases the decomposition may involve complex numbers
  - Still, every real symmetric matrix is guaranteed to have an eigen decomposition according to $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$, where $\mathbf{V}$ is an orthogonal matrix

Eigen Decomposition

- Geometric interpretation of the eigenvalues and eigenvectors is that they allow to stretch the space in specific directions
  - Figure: the two eigenvectors $\mathbf{v}^1$ and $\mathbf{v}^2$ are multiplied with the eigenvalues λ_1 and λ_2
    - We can see how the space is scaled in the direction of the larger eigenvalue λ_1

- E.g., this is used for dimensionality reduction with PCA (principal component analysis) where the eigenvectors corresponding to the largest eigenvalues are used for extracting the most important data dimensions



Picture from: Goodfellow (2017) – Deep Learning

Singular Value Decomposition

- *Singular value decomposition* (SVD) provides another way to factorize a matrix, into singular vectors and singular values
  - SVD is more generally applicable than eigen decomposition
  - Every real matrix has an SVD, but the same is not true of the eigen decomposition
    - E.g., if a matrix is not square, the eigen decomposition is not defined, and we must use SVD
- SVD of an $m \times n$ matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\boldsymbol{T}}$$

  - $\mathbf{U}$ is an $m \times m$ matrix, $\mathbf{D}$ is an $m \times n$ matrix, and $\mathbf{V}$ is an $n \times n$ matrix
  - The elements along the diagonal of $\mathbf{D}$ are known as the singular values of A
  - The columns of $\mathbf{U}$ are known as the left-singular vectors
  - The columns of $\mathbf{V}$ are known as the right-singular vectors
- For a non-square matrix $\mathbf{A}$, the squares of the singular values σ_i are the eigenvalues λ_i of $\mathbf{A}^{\boldsymbol{T}}\mathbf{A}$, i.e., $\sigma_i^2 = \lambda_i$ for $i = 1,2,\ldots,n$
- Applications of SVD include computing the pseudo-inverse of non-square matrices, matrix approximation, determining the matrix rank

Matrix Norms

- ***Frobenius norm*** – calculates the square-root of the summed squares of the elements of matrix **X**
  - This norm is similar to Euclidean norm of a vector

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^2}$$

- ***Spectral norm*** – is the largest singular value of matrix **X**
  - Denoted $\|\mathbf{X}\|_2$
  - The singular values of **X** are $\sigma_1, \sigma_2, \ldots, \sigma_m$

$$\|\mathbf{X}\|_2 = \sigma_{max}(\mathbf{X})$$

- ***$L_{2,1}$ norm*** – is the sum of the Euclidean norms of the columns of matrix **X**

$$\|\mathbf{X}\|_{2,1} = \sum_{j=1}^{n}\sqrt{\sum_{i=1}^{m} x_{ij}^2}$$

- ***Max norm*** – is the largest element of matrix **X**

$$\|\mathbf{X}\|_{\max} = \max_{i,j}(x_{ij})$$

Differential Calculus

- For a function $f: \mathbb{R} \rightarrow \mathbb{R}$, the ***derivative*** of f is defined as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- If $f'(a)$ exists, f is said to be differentiable at a
- If $f'(c)$ is differentiable for $\forall c \in [a, b]$, then f is differentiable on this interval
  - We can also interpret the derivative $f'(x)$ as the instantaneous rate of change of $f(x)$ with respect to x
  - I.e., for a small change in x, what is the rate of change of $f(x)$
- Given $y = f(x)$, where x is an independent variable and y is a dependent variable, the following expressions are equivalent:

$$f'(x) = f' = \frac{dy}{dx} = \frac{df}{dx} = \frac{d}{dx} f(x) = Df(x) = D_x f(x)$$

- The symbols $\frac{d}{dx}$, D, and D_x are differentiation operators that indicate operation of differentiation

Differential Calculus

- The following rules are used for computing the derivatives of explicit functions

  - **Derivative of constants.** $\frac{d}{dx}c = 0.$
  - **Derivative of linear functions.** $\frac{d}{dx}(ax) = a.$
  - **Power rule.** $\frac{d}{dx}x^n = nx^{n-1}.$
  - **Derivative of exponentials.** $\frac{d}{dx}e^x = e^x.$
  - **Derivative of the logarithm.** $\frac{d}{dx}\log(x) = \frac{1}{x}.$
  - **Sum rule.** $\frac{d}{dx}(g(x) + h(x)) = \frac{dg}{dx}(x) + \frac{dh}{dx}(x).$
  - **Product rule.** $\frac{d}{dx}(g(x) \cdot h(x)) = g(x)\frac{dh}{dx}(x) + \frac{dg}{dx}(x)h(x).$
  - **Chain rule.** $\frac{d}{dx}g(h(x)) = \frac{dg}{dh}(h(x)) \cdot \frac{dh}{dx}(x).$

Higher Order Derivatives

- The derivative of the first derivative of a function $f(x)$ is the ***second derivative*** of $f(x)$

$$\frac{d^2 f}{dx^2} = \frac{d}{dx}\left(\frac{df}{dx}\right)$$

- The second derivative quantifies how the rate of change of $f(x)$ is changing
  - E.g., in physics, if the function describes the displacement of an object, the first derivative gives the velocity of the object (i.e., the rate of change of the position)
  - The second derivative gives the acceleration of the object (i.e., the rate of change of the velocity)

- If we apply the differentiation operation any number of times, we obtain the n-th derivative of $f(x)$

$$f^{(n)}(x) = \frac{d^n f}{dx^n} = \left(\frac{d}{dx}\right)^n f(x)$$

Taylor Series

- *Taylor series* provides a method to approximate any function $f(x)$ at a point x_0 if we have the first n derivatives $\{f(x_0), f^{(1)}(x_0), f^{(2)}(x_0), \dots, f^{(n)}(x_0)\}$

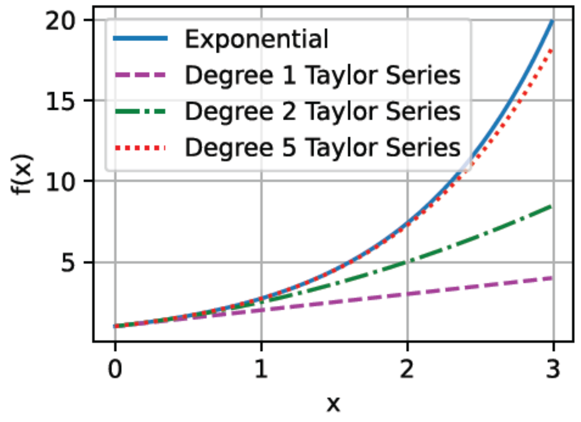
- For instance, for $n = 2$, the second-order approximation of a function $f(x)$ is

$$f(x) \approx \frac{1}{2}\frac{d^2 f}{dx^2}\bigg|_{x_0}(x - x_0)^2 + \frac{df}{dx}\bigg|_{x_0}(x - x_0) + f(x_0)$$

- Similarly, the approximation of $f(x)$ with a Taylor polynomial of n-degree is

$$f(x) \approx \sum_{i=0}^{n} \frac{1}{i!}\frac{d^{(i)} f}{dx^i}\bigg|_{x_0}(x - x_0)^i$$

- For example, the figure shows the first-order, second-order, and fifth-order polynomial of the exponential function $f(x) = e^x$ at the point $x_0 = 0$



Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/single-variable-calculus.html

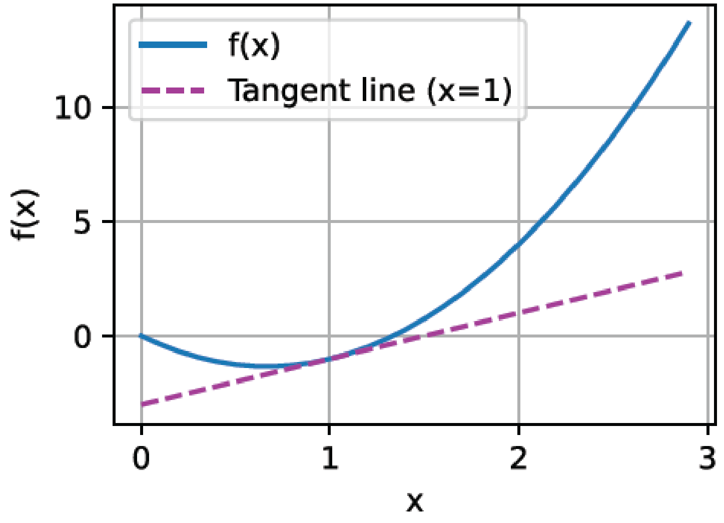
Geometric Interpretation

- To provide a geometric interpretation of the derivatives, let's consider a first-order Taylor series approximation of $f(x)$ at $x = x_0$

$$f(x) \approx f(x_0) + \left.\frac{df}{dx}\right|_{x_0} (x - x_0)$$

- The expression approximates the function $f(x)$ by a line which passes through the point $(x_0, f(x_0))$ and has slope $\left.\frac{df}{dx}\right|_{x_0}$ (i.e., the value of $\frac{df}{dx}$ at the point x_0)

- Therefore, the first derivative of a function is also the slope of the tangent line to the curve of the function



Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/single-variable-calculus.html

Partial Derivatives

- So far, we looked at functions of a single variable, where $f: \mathbb{R} \to \mathbb{R}$
- Functions that depend on many variables are called multivariate functions
- Let $y = f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$ be a multivariate function with n variables
  - The input is an n-dimensional vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and the output is a scalar y
  - The mapping is $f: \mathbb{R}^n \to \mathbb{R}$
- The ***partial derivative*** of y with respect to its i^{th} parameter x_i is

$$\frac{\partial y}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, x_2, \ldots, x_i + h, \ldots, x_n) - f(x_1, x_2, \ldots, x_i, \ldots, x_n)}{h}$$

- To calculate $\frac{\partial y}{\partial x_i}$ (∂ pronounced "del" or we can just say "partial derivative"), we can treat $x_1, x_2, \ldots, x_{i-1}$, $x_{i+1} \ldots, x_n$ as constants and calculate the derivative of y only with respect to x_i
- For notation of partial derivatives, the following are equivalent:

$$\frac{\partial y}{\partial x_i} = \frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} f(\mathbf{x}) = f_{x_i} = f_i = D_i f = D_{x_i} f$$

Gradient

- We can concatenate partial derivatives of a multivariate function with respect to all its input variables to obtain the ***gradient*** vector of the function
- The gradient of the multivariate function $f(\mathbf{x})$ with respect to the n-dimensional input vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, is a vector of n partial derivatives

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]^T$$

- When there is no ambiguity, the notations $\nabla f(\mathbf{x})$ or $\nabla_{\mathbf{x}} f$ are often used for the gradient instead of $\nabla_{\mathbf{x}} f(\mathbf{x})$
  - The symbol for the gradient is the Greek letter ∇ (pronounced "nabla"), although $\nabla_{\mathbf{x}} f(\mathbf{x})$ is more often it is pronounced "gradient of *f* with respect to **x**"

- In ML, the gradient descent algorithm relies on the opposite direction of the gradient of the loss function $\mathcal{L}$ with respect to the model parameters θ ($\nabla_{\theta}\mathcal{L}$) for minimizing the loss function
  - Adversarial examples can be created by adding perturbation in the direction of the gradient of the loss $\mathcal{L}$ with respect to input examples x ($\nabla_{x}\mathcal{L}$) for maximizing the loss function

Hessian Matrix

- To calculate the second-order partial derivatives of multivariate functions, we need to calculate the derivatives for all combination of input variables
- That is, for a function $f(\mathbf{x})$ with an n-dimensional input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, there are n^2 second partial derivatives for any choice of i and j

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right)$$

- The second partial derivatives are assembled in a matrix called the ***Hessian***

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

- Computing and storing the Hessian matrix for functions with high-dimensional inputs can be computationally prohibitive
  - E.g., the loss function for a ResNet50 model with approximately 23 million parameters, has a Hessian of 23 M × 23 M = 529 T (trillion) parameters

Jacobian Matrix

- The concept of derivatives can be further generalized to vector-valued functions (or, vector fields) $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$
- For an n-dimensional input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$, the vector of functions is given as

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \in \mathbb{R}^m$$

- The matrix of first-order partial derivates of the vector-valued function $\mathbf{f}(\mathbf{x})$ is an $m \times n$ matrix called a ***Jacobian***

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

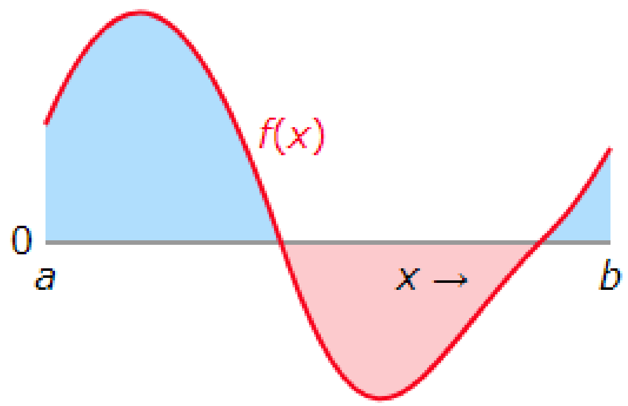
  - For example, in robotics a robot Jacobian matrix gives the partial derivatives of the translational and angular velocities of the robot end-effector with respect to the joints (i.e., axes) velocities

Integral Calculus

- For a function $f(x)$ defined on the domain $[a, b]$, the definite ***integral*** of the function is denoted

$$\int_a^b f(x)dx$$

- Geometric interpretation of the integral is the area between the horizontal axis and the graph of $f(x)$ between the points a and b
  - In this figure, the integral is the sum of blue areas (where $f(x) > 0$) minus the pink area (where $f(x) < 0$)



Picture from: https://mjo.osborne.economics.utoronto.ca/index.php/tutorial/index/1/clc/t

Optimization

- *Optimization* is concerned with optimizing an objective function — finding the value of an argument that minimizes of maximizes the function
  - Most optimization algorithms are formulated in terms of minimizing a function $f(x)$
  - Maximization is accomplished vie minimizing the negative of an objective function (e.g., minimize $-f(x)$)
  - In minimization problems, the objective function is often referred to as a cost function or loss function or error function

- Optimization is very important for machine learning
  - The performance of optimization algorithms affect the model's training efficiency

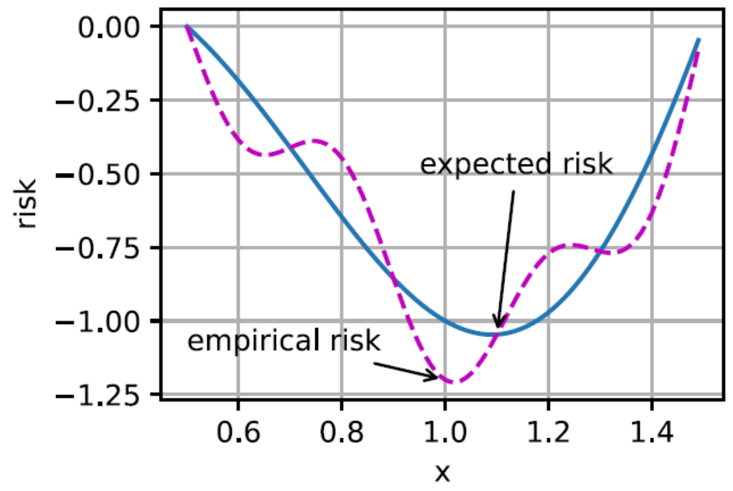
- Most optimization problems in machine learning are nonconvex
  - Meaning that the loss function is not a convex function
  - Nonetheless, the design and analysis of algorithms for solving convex problems has been very instructive for advancing the field of machine learning

Optimization

- Optimization and machine learning have related, but somewhat different goals
  - Goal in optimization: minimize an objective function
    - For a set of training examples, reduce the training error
  - Goal in ML: find a suitable model, to predict on data examples
    - For a set of testing examples, reduce the generalization error

- For a given empirical function *g* (dashed purple curve), optimization algorithms attempt to find the point of minimum empirical risk (error on the training dataset)

- ML algorithms attempt to find the point of minimum expected risk, based on minimizing the error on a set of testing examples (blue curve)
    - Which may be at a different location than the minimum of the training examples



Picture from: http://d2l.ai/chapter_optimization/optimization-intro.html

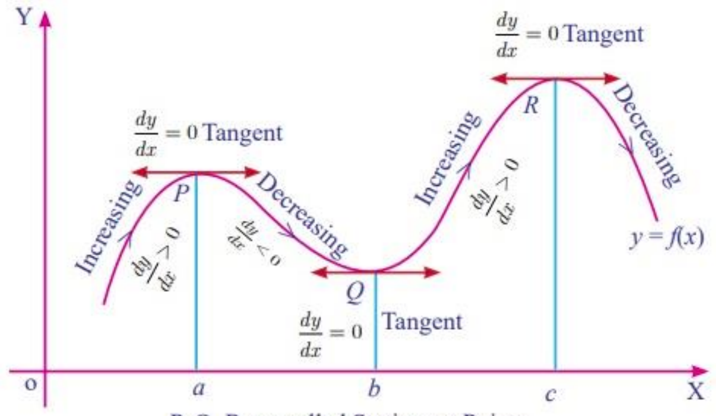
Stationary Points

- *Stationary points* (or critical points) of a differentiable function $f(x)$ of one variable are the points where the derivative of the function is zero, i.e., $f'(x) = 0$
- The stationary points can be:
  - *Minimum*, a point where the derivative changes from negative to positive
  - *Maximum*, a point where the derivative changes from positive to negative
  - *Saddle point*, derivative is either positive or negative on both sides of the point

- The minimum and maximum points are collectively known as extremum points

- The nature of stationary points can be determined based on the second derivative of $f(x)$ at the point
  - If $f''(x) > 0$, the point is a minimum
  - If $f''(x) < 0$, the point is a maximum
  - If $f''(x) = 0$, inconclusive, the point can be a saddle point, but it may not

- The same concept also applies to gradients of multivariate functions

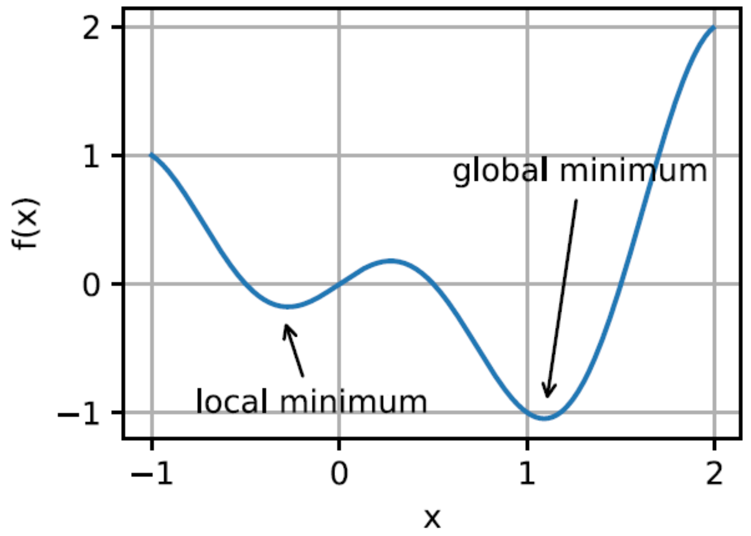


P, Q, R are called Stationary Points

Local Minima

- Among the challenges in optimization of model's parameters in ML involve local minima, saddle points, vanishing gradients
- For an objective function $f(x)$, if the value at a point x is the minimum of the objective function over the entire domain of x, then it is the ***global minimum***
- If the value of $f(x)$ at x is smaller than the values of the objective function at any other points in the vicinity of x, then it is the ***local minimum***

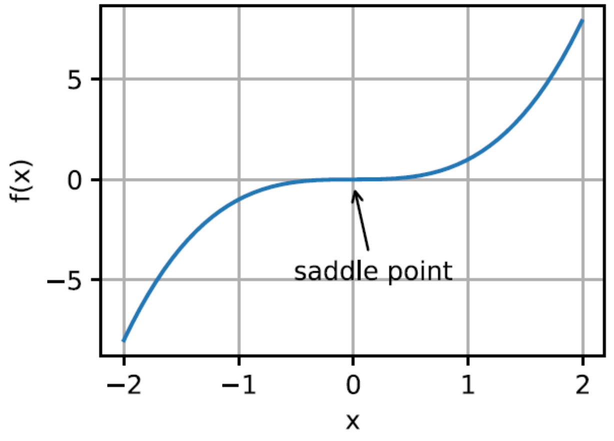
- The objective functions in ML usually have many local minima
  - When the solution of the optimization algorithm is near the local minimum, the gradient of the loss function approaches or becomes zero (vanishing gradients)
  - Therefore, the obtained solution in the final iteration can be a local minimum, rather than the global minimum

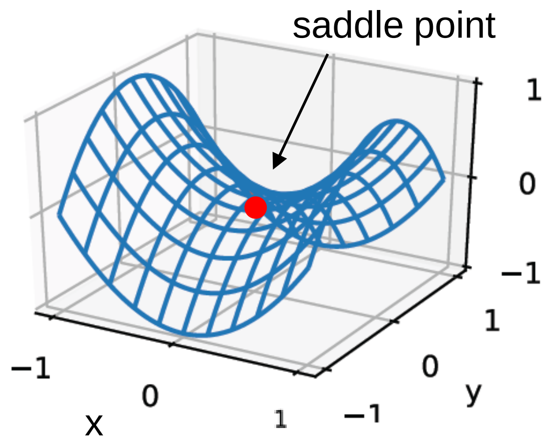


Picture from: http://d2l.ai/chapter_optimization/optimization-intro.html

Saddle Points

- The gradient of a function $f(x)$ at a saddle point is 0, but the point is not a minimum or maximum point
  - The optimization algorithms may stall at saddle points, without reaching a minima
- Note also that the point of a function at which the sign of the curvature changes is called an inflection point
  - An inflection point ($f''(x) = 0$) can also be a saddle point, but it does not have to be
- For the 2D function (right figure), the saddle point is at (0,0)
  - The point looks like a saddle, and gives the minimum with respect to x, and the maximum with respect to y



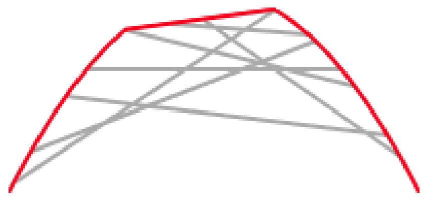


Picture from: http://d2l.ai/chapter_optimization/optimization-intro.html

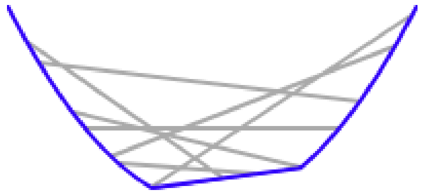
Convex Optimization

- A function of a single variable is *concave* if every line segment joining two points on its graph does not lie above the graph at any point

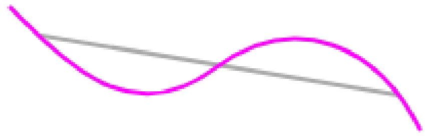
- Symmetrically, a function of a single variable is *convex* if every line segment joining two points on its graph does not lie below the graph at any point



A concave function:
no line segment joining
two points on the graph
lies above the graph
at any point



A convex function:
no line segment joining
two points on the graph
lies below the graph
at any point



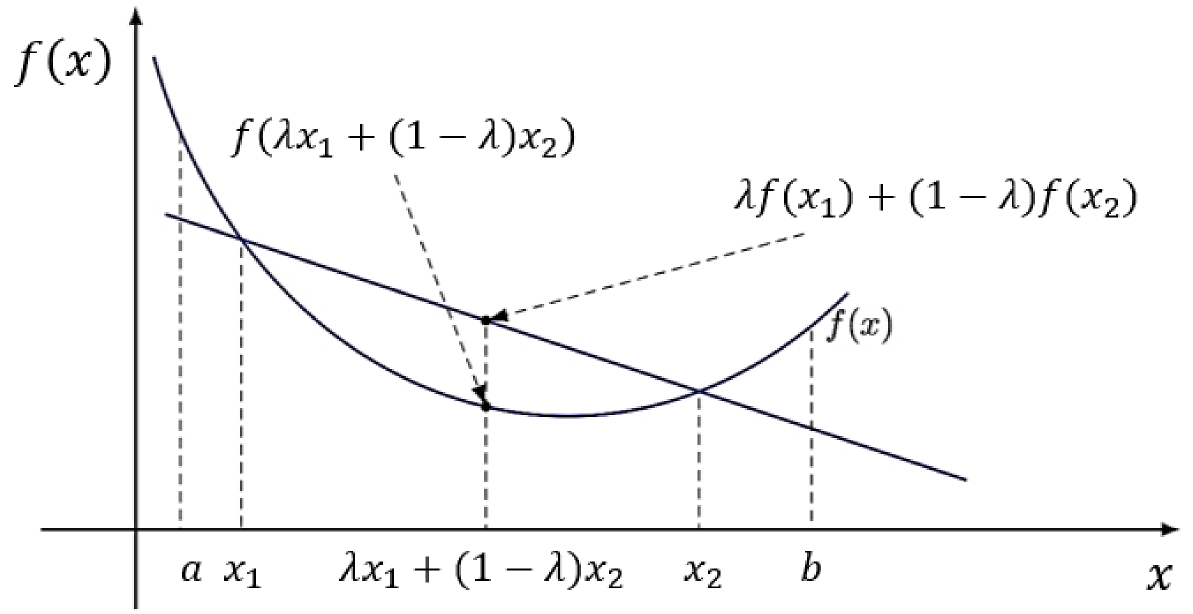
A function that is neither
concave nor convex:
the line segment shown lies
above the graph at some
points and below it at others

Picture from: https://mjo.osborne.economics.utoronto.ca/index.php/tutorial/index/1/cv1/t

Convex Functions

- In mathematical terms, the function f is a *convex function* if for all points x_1, x_2 and for all $\lambda \in [0,1]$

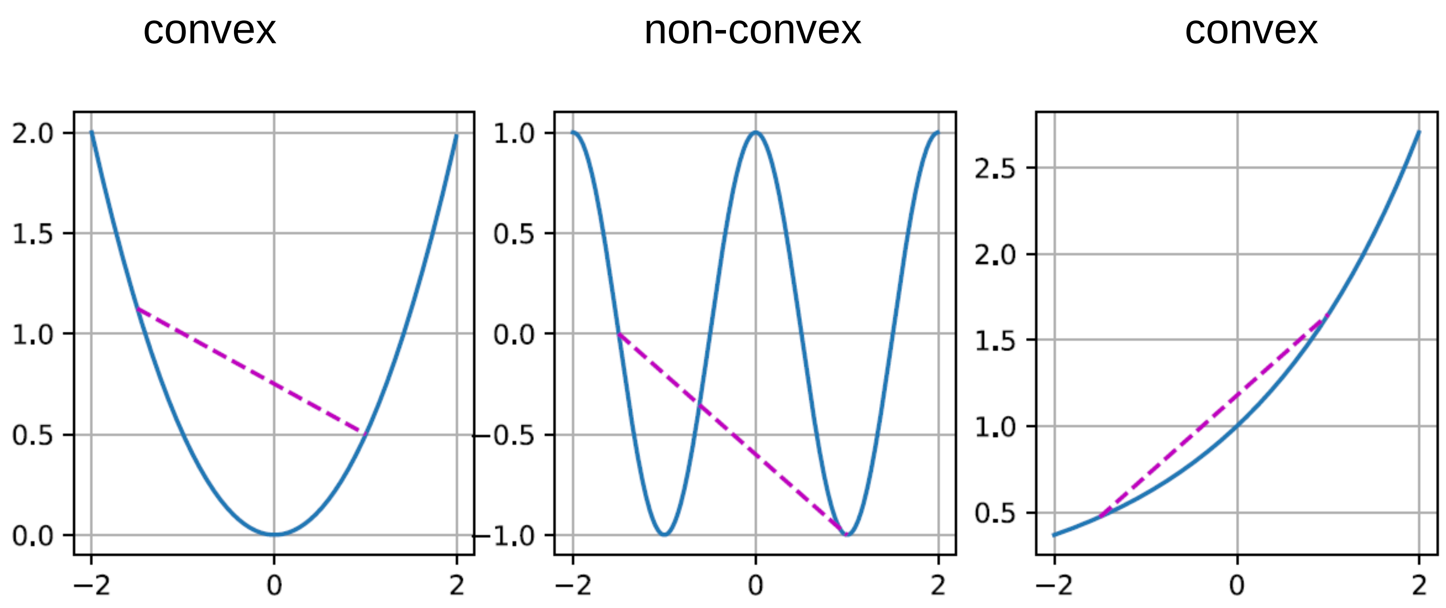
$$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2)$$



$f(x)$
$f(\lambda x_1 + (1-\lambda)x_2)$
$\lambda f(x_1) + (1-\lambda)f(x_2)$
$f(x)$
a
x_1
$\lambda x_1 + (1-\lambda)x_2$
x_2
b
x

Convex Functions

- One important property of convex functions is that they do not have local minima
  - Every local minimum of a convex function is a global minimum
  - I.e., every point at which the gradient of a convex function = 0 is the global minimum
  - The figure below illustrates two convex functions, and one nonconvex function



Picture from: http://d2l.ai/chapter_optimization/convexity.html

Convex Functions

- Another important property of convex functions is stated by the ***Jensen's inequality***

- Namely, if we let $\alpha_1 = \lambda$ and $\alpha_2 = 1 - \lambda$, the definition of convex function becomes

$$\alpha_1 f(x_1) + \alpha_2 f(x_2) \geq f(\alpha_1 x_1 + \alpha_2 x_2)$$

- The Danish mathematician Johan Jensen showed that this can be generalized for all α_i that are non-negative real numbers and $\sum_i \alpha_i$=1, to the following:

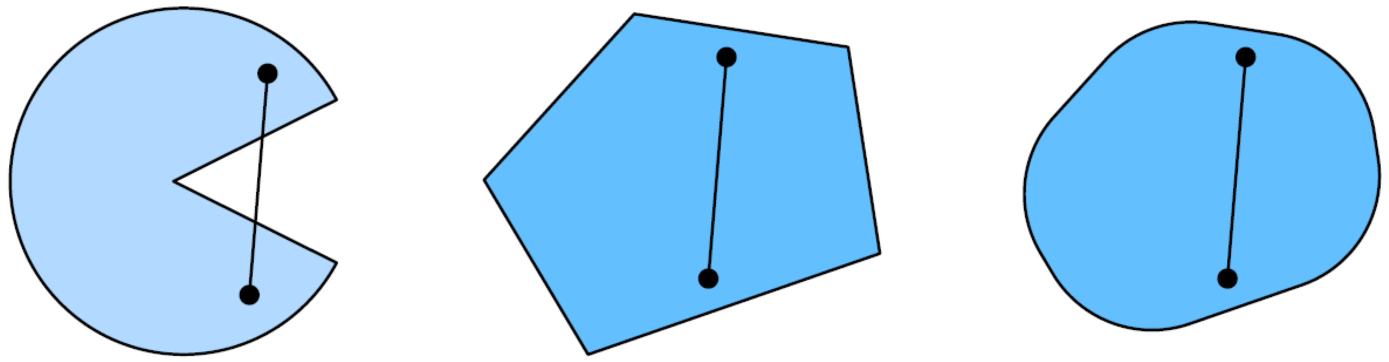
$$\alpha_1 f(x_1) + \alpha_2 f(x_2) + \cdots + \alpha_n f(x_n) \geq f(\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n)$$

Convex Sets

- A set $\mathcal{X}$ in a vector space is a ***convex set*** is for any $a, b \in \mathcal{X}$ the line segment connecting a and b is also in $\mathcal{X}$
- For all $\lambda \in [0,1]$, we have

$$\lambda \cdot a + (1 - \lambda) \cdot b \in \mathcal{X} \text{ for all } a, b \in \mathcal{X}$$

- In the figure, each point represents a 2D vector
  - The left set is nonconvex, and the other two sets are convex
- Properties of convex sets include:
  - If $\mathcal{X}$ and $\mathcal{Y}$ are convex sets, then $\mathcal{X} \cap \mathcal{Y}$ is also convex
  - If $\mathcal{X}$ and $\mathcal{Y}$ are convex sets, then $\mathcal{X} \cup \mathcal{Y}$ is not necessarily convex



Picture from: http://d2l.ai/chapter_optimization/convexity.html

Derivatives and Convexity

- A twice-differentiable function of a single variable $f: \mathbb{R} \to \mathbb{R}$ is convex if and only if its second derivative is non-negative everywhere
  - or, we can write, $\frac{d^2 f}{dx^2} \geq 0$
  - For example, $f(x) = x^2$ is convex, since $f'(x) = 2x$, and $f''(x) = 2$, meaning that $f''^{(x)} > 0$
- A twice-differentiable function of many variables $f: \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its Hessian matrix is positive semi-definite everywhere
  - or, we can write, $\mathbf{H}_f \succcurlyeq 0$
  - This is equivalent to stating that all eigenvalues of the Hessian matrix are non-negative (i.e., ≥ 0)

Constrained Optimization

- The optimization problem that involves a set of constraints which need to be satisfied to optimize the objective function is called ***constrained optimization***

- E.g., for a given objective function $f(\mathbf{x})$ and a set of constraint functions $c_i(\mathbf{x})$

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x})$$
$$\text{subject to } c_i(\mathbf{x}) \leq 0 \;\; \text{for all } \; i \in \{1, 2, \dots, N\}$$

- The points that satisfy the constraints form the feasible region

- Various optimization algorithms have been developed for handling optimization problems based on whether the constraints are equalities, inequalities, or a combination of equalities and inequalities

Lagrange Multipliers

- One approach to solving optimization problems is to substitute the initial problem with optimizing another related function

- The Lagrange function for optimization of the constrained problem on the previous page is defined as

$$L(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_i \alpha_i c_i(\mathbf{x}) \quad \text{where} \quad \alpha_i \geq 0$$

- The variables α_i are called ***Lagrange multipliers*** and ensure that the constraints are properly enforced
  - They are chosen to ensure that $c_i(\mathbf{x}) \leq 0$ for all $i \in \{1, 2, \dots, N\}$

- This is a saddle-point optimization problem where one wants to **minimize** $L(\mathbf{x}, \alpha)$ with respect to $\mathbf{x}$ and simultaneously **maximize** $L(\mathbf{x}, \alpha)$ with respect to α_i
  - The saddle point of $L(\mathbf{x}, \alpha)$ gives the optimal solution to the original constrained optimization problem

Projections

- An alternative strategy for satisfying constraints are projections
- E.g., *gradient clipping* in NNs can require that the norm of the gradient is bounded by a constant value c
- Approach:
  - At each iteration during training
  - If the norm of the gradient $\|g\| \geq$ c, then the update is $g^{new} \leftarrow c \cdot \frac{g^{old}}{\|g^{old}\|}$
  - If the norm of the gradient $\|g\| <$ c, then the update is $g^{new} \leftarrow g^{old}$
- Note that since $\frac{g^{old}}{\|g^{old}\|}$ is a unit vector (i.e., it has a norm = 1), then the vector $c \cdot \frac{g^{old}}{\|g^{old}\|}$ has a norm = c
- Such clipping is the projection of the gradient g onto the ball of radius c
  - Projection on the unit ball is for $c = 1$

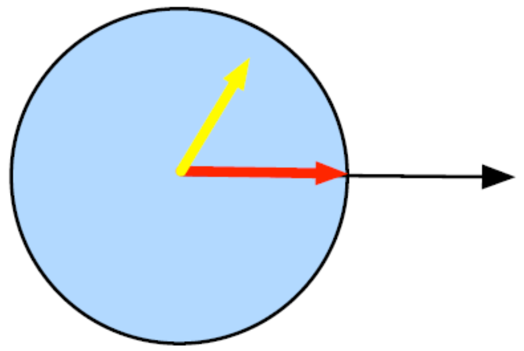
Projections

- More generally, a ***projection*** of a vector $\mathbf{x}$ onto a set $\mathcal{X}$ is defined as

$$\underset{\mathcal{X}}{\mathrm{Proj}}(\mathbf{x}) = \underset{\mathbf{x}' \in \mathcal{X}}{\mathrm{argmin}} \|\mathbf{x} - \mathbf{x}'\|_2$$

- This means that the vector $\mathbf{x}$ is projected onto the closest vector $\mathbf{x}'$ that belongs to the set $\mathcal{X}$

- For example, in the figure, the blue circle represents a convex set $\mathcal{X}$
  - The points inside the circle project to itself
    - E.g., $\mathbf{x}$ is the yellow vector, its closest point $\mathbf{x}'$ in the set $\mathcal{X}$ is itself: the distance between $\mathbf{x}$ and $\mathbf{x}'$ is $\|\mathbf{x} - \mathbf{x}'\|_2 = 0$
  - The points outside the circle project to the closest point inside the circle
    - E.g., $\mathbf{x}$ is the yellow vector, its closest point $\mathbf{x}'$ in the set $\mathcal{X}$ is the red vector
    - Among all vectors in the set $\mathcal{X}$, the red vector $\mathbf{x}'$ has the smallest distance to $\mathbf{x}$, i.e., $\|\mathbf{x} - \mathbf{x}'\|_2$



Picture from: http://d2l.ai/chapter_optimization/convexity.html

First-order vs Second-order Optimization

- ***First-order optimization algorithms*** use the gradient of a function for finding the extrema points
  - Methods: gradient descent, proximal algorithms, optimal gradient schemes
  - The disadvantage is that they can be slow and inefficient

- ***Second-order optimization algorithms*** use the Hessian matrix of a function for finding the extrema points
  - This is since the Hessian matrix holds the second-order partial derivatives
  - Methods: Newton’s method, conjugate gradient method, Quasi-Newton method, Gauss-Newton method, BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, Levenberg-Marquardt method, Hessian-free method
  - The second-order derivatives can be thought of as measuring the curvature of the loss function
  - Recall also that the second-order derivative can be used to determine whether a stationary points is a maximum ($f''(x) < 0$), minimum ($f''(x) > 0$)
  - This information is richer than the information provided by the gradient
  - Disadvantage: computing the Hessian matrix is computationally expensive, and even prohibitive for high-dimensional data

Lower Bound and Infimum

- *Lower bound* of a subset $\mathcal{S}$ from a partially ordered set $\mathcal{X}$ is an element a of $\mathcal{X}$, such that $a \leq s$ for all $s \in \mathcal{S}$
  - E.g., for the subset $\mathcal{S} = \{2, 3, 6, 8\}$ from the natural numbers $\mathbb{N}$, lower bounds are the numbers 2, 1, 0, -3, and all other natural numbers ≤ 2

- *Infimum* of a subset $\mathcal{S}$ from a partially ordered set $\mathcal{X}$ is the greatest lower bound in $\mathcal{X}$, denoted $\inf_{s \in \mathcal{S}} s$
  - It is the maximal quantity h such that $h \leq s$ for all $s \in \mathcal{S}$
  - E.g., the infimum of the set $\mathcal{S} = \{2, 3, 6, 8\}$ is h =2, since it is the greatest lower bound

- Example: consider the subset of positive real numbers (excluding zero) $\mathbb{R}_{\geq 0}$=$\{x \in \mathbb{R}\colon x \geq 0\}$
  - The subset $\mathbb{R}_{\geq 0}$ does not have a minimum, because for every small positive number, there is a another even smaller positive number
  - On the other hand, all real negative numbers and 0 are lower bounds on the subset $\mathbb{R}_{\geq 0}$
  - 0 is the greatest lower bound of all lower bounds, and therefore, the infimum of $\mathbb{R}_{\geq 0}$ is 0

Upper Bound and Supremum

- *Upper bound* of a subset $\mathcal{S}$ from a partially ordered set $\mathcal{X}$ is an element b of $\mathcal{X}$, such that $b \geq s$ for all $s \in \mathcal{S}$
  - E.g., for the subset $\mathcal{S} = \{2, 3, 6, 8\}$ from the natural numbers $\mathbb{N}$, upper bounds are the numbers 8, 9, 40, and all other natural numbers ≥ 8

- *Supremum* of a subset $\mathcal{S}$ from a partially ordered set $\mathcal{X}$ is the least upper bound in $\mathcal{X}$, denoted $\sup_{s \in \mathcal{S}} s$
  - It is the minimal quantity g such that g $\geq s$ for all $s \in \mathcal{S}$
  - E.g., the supremum of the subset $\mathcal{S} = \{2, 3, 6, 8\}$ is $g = 8$, since it is the least upper bound

- Example: for the subset of negative real numbers (excluding zero) $\mathbb{R}_{\leq 0}$=$\{x \in \mathbb{R}: x \leq 0\}$
  - All real positive numbers and 0 are upper bounds
  - 0 is the least upper bound, and therefore, the supremum of $\mathbb{R}_{\leq 0}$

Lipschitz Function

- A function $f(x)$ is a ***Lipschitz continuous function*** if a constant $\rho > 0$ exists, such that for all points x_1, x_2

$$\|f(x_1) - f(x_2)\| \leq \rho \|x_1 - x_2\|$$

- Such function is also called a ρ-Lipschitz function

- Intuitively, a Lipschitz function cannot change too fast
  - i.e., if the points x_1 and x_2 are close (i.e., the distance $\|x_1 - x_2\|$ is small), that means that the $f(x_1)$ and $f(x_2)$ are also close (i.e., the distance $\|f(x_1) - f(x_2)\|$ is also small)
    - The smallest real number that bounds the change of $\|f(x_1) - f(x_2)\|$ for all points x_1, x_2 is the Lipschitz constant ρ of the function $f(x)$
  - For a ρ-Lipschitz function $f(x)$, the first derivative $f'(x)$ is bounded everywhere by ρ

- E.g., the function $f(x) = log(1 + \exp(x))$ is 1-Lipschitz over $\mathbb{R}$
  - Since $\|f'(x)\| = \left\|\frac{\exp(x)}{1+\exp(x)}\right\| = \left\|\frac{1}{\exp(-x)+1}\right\| = \frac{\mathbf{1}}{\|\exp(-x)+1\|} \leq 1$
  - i.e., $\rho = 1$

Lipschitz Continuous Gradient

- A differentiable function $f(x)$ has a *Lipschitz continuous gradient* if a constant $\rho > 0$ exists, such that for all points x_1, x_2

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq \rho \|x_1 - x_2\|$$

- For a function $f(x)$ with a ρ-Lipschitz gradient, the second derivative $f''(x)$ is bounded everywhere by ρ

- E.g., consider the function $f(x) = x^2$
  - $f(x) = x^2$ is not a Lipschitz continuous function, since $f'(x) = 2x$, so when $x \to \infty$ then $f'(x) \to \infty$, i.e., the derivative is not bounded everywhere
  - Since $f''(x) = 2$, therefore the gradient $f'(x)$ is 2-Lipschitz everywhere, since the second derivative is bounded everywhere by 2

Probability

- Intuition:
  - In a process, several outcomes are possible
  - When the process is repeated many times, each outcome occurs with a ***relative frequency***, or ***probability***
  - If a particular outcome occurs more often, we say it is more probable

- Probability arises in two contexts
  - In actual repeated experiments
    - Example: You record the color of 1,000 cars driving by. 57 of them are green. You estimate the probability of a car being green as 57/1,000 = 0.057.
  - In idealized conceptions of a repeated process
    - Example: You consider the behavior of an unbiased six-sided die. The expected probability of rolling a 5 is 1/6 = 0.1667.
    - Example: You need a model for how people's heights are distributed. You choose a normal distribution to represent the expected relative probabilities.

Probability

- Solving machine learning problems requires to deal with uncertain quantities, as well as with stochastic (non-deterministic) quantities
  - Probability theory provides a mathematical framework for representing and quantifying uncertain quantities

- There are different sources of uncertainty:

  - Inherent stochasticity in the system being modeled
    - For example, most interpretations of quantum mechanics describe the dynamics of subatomic particles as being probabilistic

  - Incomplete observability
    - Even deterministic systems can appear stochastic when we cannot observe all the variables that drive the behavior of the system

  - Incomplete modeling
    - When we use a model that must discard some of the information we have observed, the discarded information results in uncertainty in the model’s predictions
    - E.g., discretization of real-numbered values, dimensionality reduction, etc.

Random variables

- A ***random variable*** X is a variable that can take on different values
  - Example: X = rolling a die
    - Possible values of X comprise the **sample space**, or **outcome space**, $\mathcal{S} = \{1, 2, 3, 4, 5, 6\}$
    - We denote the event of "seeing a 5" as $\{X = 5\}$ or $X = 5$
    - The probability of the event is $P(\{X = 5\})$ or $P(X = 5)$
    - Also, $P(5)$ can be used to denote the probability that X takes the value of 5

- A ***probability distribution*** is a description of how likely a random variable is to take on each of its possible states
  - A compact notation is common, where $P(X)$ is the probability distribution over the random variable X
    - Also, the notation $\mathrm{X} \sim P(X)$ can be used to denote that the random variable X has probability distribution $P(X)$

- Random variables can be discrete or continuous
  - Discrete random variables have finite number of states: e.g., the sides of a die
  - Continuous random variables have infinite number of states: e.g., the height of a person

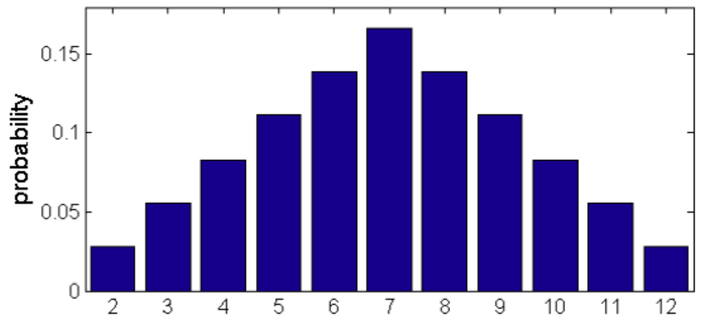
Axioms of probability

- The probability of an event $\mathcal{A}$ in the given sample space $\mathcal{S}$, denoted as $P(\mathcal{A})$, must satisfies the following properties:
  - Non-negativity
    - For any event $\mathcal{A} \in \mathcal{S}$, $P(\mathcal{A}) \geq 0$
  - All possible outcomes
    - Probability of the entire sample space is 1, $P(\mathcal{S}) = 1$
  - Additivity of disjoint events
    - For all events $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{S}$ that are mutually exclusive ($\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$), the probability that both events happen is equal to the sum of their individual probabilities, $P(\mathcal{A}_1 \cup \mathcal{A}_2) = P(\mathcal{A}_1)+P(\mathcal{A}_2)$

- The probability of a random variable $P(X)$ must obey the axioms of probability over the possible values in the sample space $\mathcal{S}$

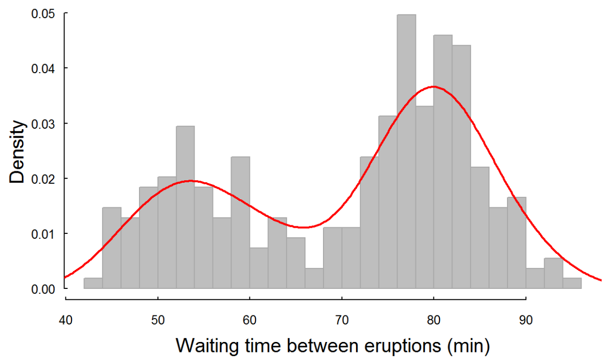
Discrete Vs. Continuous Variables

- A probability distribution over discrete variables may be described using a ***probability mass function*** (PMF)
  - E.g., sum of two dice



- A probability distribution over continuous variables may be described using a ***probability density function*** (PDF)
  - E.g., waiting time between eruptions of Old Faithful
  - A PDF gives the probability of an infinitesimal region with volume δX
  - To find the probability over an interval $[a, b]$, we can integrate the PDF as follows:

$$P(X \in [a, b]) = \int_a^b P(X)dX$$



Multivariate Random Variables

- We may need to consider several random variables at a time
  - If several random processes occur in parallel or in sequence
  - E.g., to model the relationship between several diseases and symptoms
  - E.g., to process images with millions of pixels (each pixel is one random variable)

- Next, we will study probability distributions defined over multiple random variables
  - These include joint, conditional, and marginal probability distributions

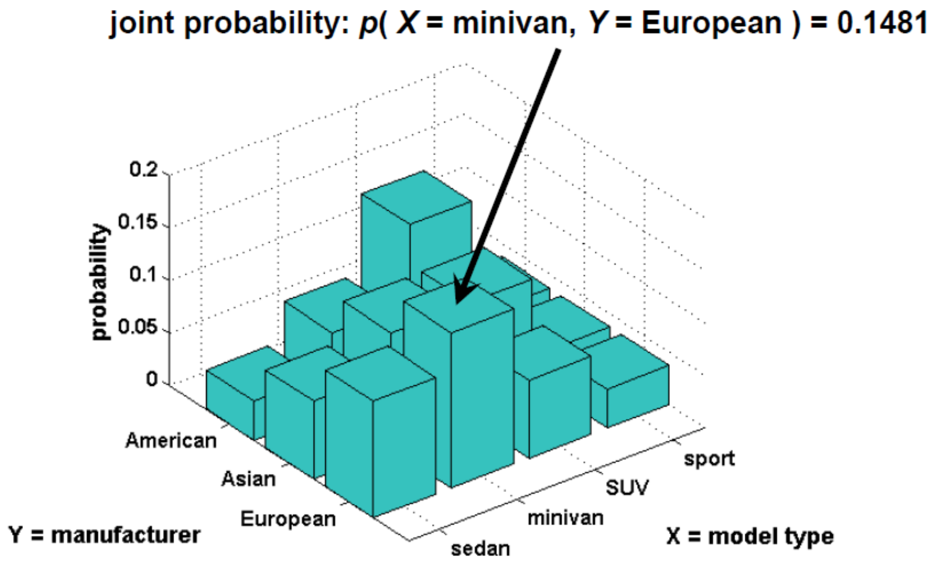
- The individual random variables can also be grouped together into a random vector, because they represent different properties of an individual statistical unit

- A *multivariate random variable* is a vector of multiple random variables $\mathbf{X} = (X_1, X_2, \dots, X_n)^T$
  - It is also referred to as a *random vector*

Joint Probability Distribution

- Probability distribution that acts on many variables at the same time is known as a *joint probability distribution*

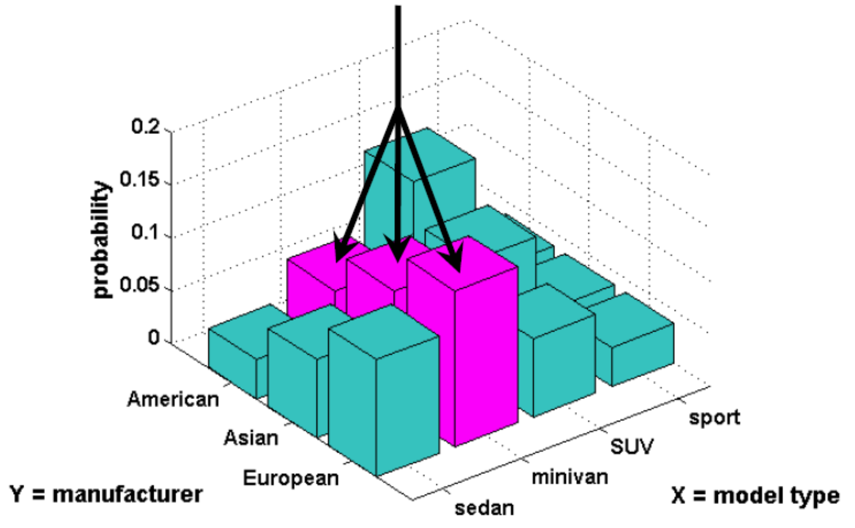
- Given any values x and y of two random variables X and Y, what is the probability that $X = x$ and $Y = y$ simultaneously?
  - $P(X = x, Y = y)$ denotes the joint probability
  - We may also write $P(x, y)$ for brevity



Marginal Probability Distribution

- *Marginal probability distribution* is the probability distribution of a single variable
  - It is calculated based on the joint probability distribution $P(X, Y)$
  - i.e., using the sum rule: $P(X = x) = \sum_y P(X = x, Y = y)$
    - For continuous random variables, the summation is replaced with integration, $P(X = x) = \int P(X = x, Y = y)\, dy$
  - This process is called marginalization

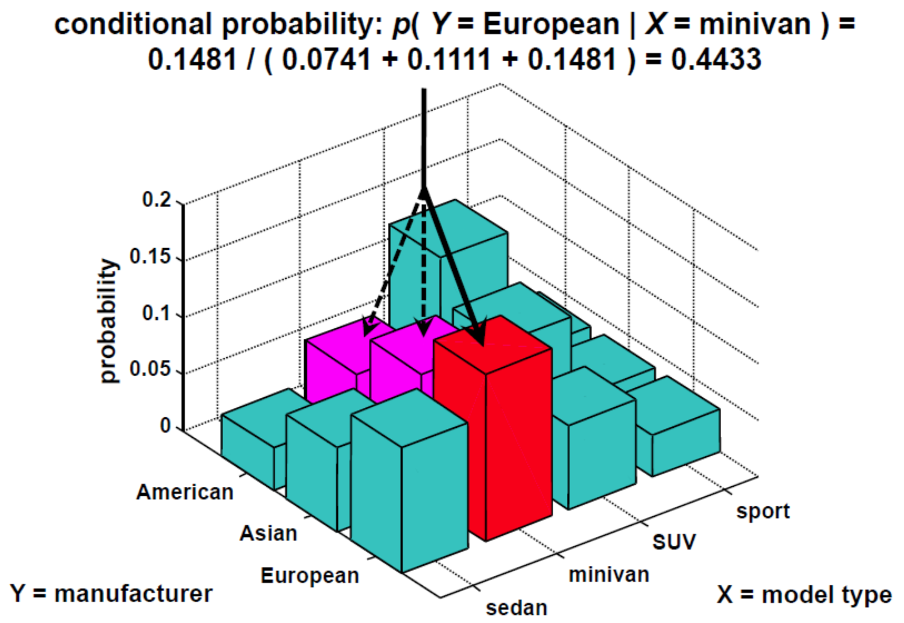
marginal probability: *p*(*X* = minivan) = 0.0741 + 0.1111 + 0.1481 = 0.3333



Conditional Probability Distribution

- ***Conditional probability distribution*** is the probability distribution of one variable provided that another variable has taken a certain value
  - Denoted $P(X = x|\, Y = y)$

- Note that: $P(X = x|\, Y = y) = \frac{P(X=x,\, Y=y)}{P(Y=y)}$



Bayes' Theorem

- *Bayes' theorem* – allows to calculate conditional probabilities for one variable when conditional probabilities for another variable are known

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

- Also known as Bayes' rule
- Multiplication rule for the joint distribution is used: $P(X,Y) = P(Y|X)P(X)$
- By symmetry, we also have: $P(Y,X) = P(X|Y)P(Y)$

- The terms are referred to as:
  - $P(X)$, the prior probability, the initial degree of belief for X
  - $P(X|Y)$, the posterior probability, the degree of belief after incorporating the knowledge of Y
  - $P(Y|X)$, the likelihood of Y given X
  - P(Y), the evidence
  - Bayes' theorem: $\textbf{posterior probability} = \frac{\textbf{likelihood} \times \textbf{prior probability}}{\textbf{evidence}}$

Independence

- Two random variables X and Y are ***independent*** if the occurrence of Y does not reveal any information about the occurrence of X
  - E.g., two successive rolls of a die are independent

- Therefore, we can write: $P(X|\ Y) = P(X)$
  - The following notation is used: $X \perp Y$
  - Also note that for independent random variables: $P(X, Y) = P(X)P(Y)$

- In all other cases, the random variables are ***dependent***
  - E.g., duration of successive eruptions of Old Faithful
  - Getting a king on successive draws form a deck (the drawn card is not replaced)

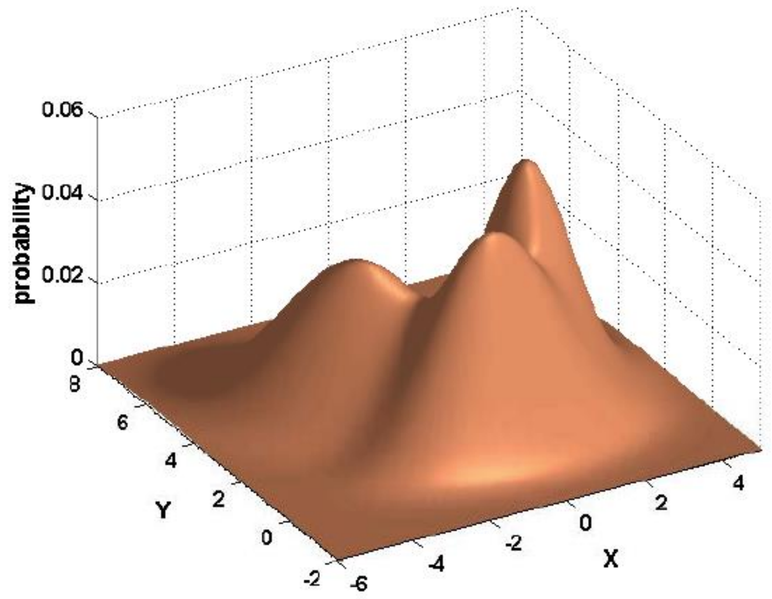
- Two random variables X and Y are ***conditionally independent*** given another random variable Z if and only if $P(X, Y|Z) = P(X|Z)P(Y|Z)$
  - This is denoted as $X \perp Y|Z$

Continuous Multivariate Distributions

- Same concepts of joint, marginal, and conditional probabilities apply for continuous random variables

- The probability distributions use integration of continuous random variables, instead of summation of discrete random variables

    - Example: a three-component Gaussian mixture probability distribution in two dimensions



Expected Value

- The ***expected value*** or ***expectation*** of a random variable X drawn from a probability distribution $P(X)$ is the average (mean) value of all possible outcomes
  - For a discrete random variable X, it is calculated as
$$\mathbb{E}_{X\sim P}[X] = \sum_{X} X\, P(X)$$
  - For a continuous random variable X, it is calculated as
$$\mathbb{E}_{X\sim P}[X] = \int X\, P(X)\, dX$$
  - When the identity of the distribution is clear from the context, we can write $\mathbb{E}[X]$
  - E.g., for a sample of observations: $\mu = \mathbb{E}[X] = \sum_i P(X_i) \cdot X_i = \frac{1}{N}\sum_i X_i$
- Mean is the most common measure of central tendency of a distribution
  - Other measures of central tendency: median, mode
- By analogy, the ***expected value of a function*** $f(X)$ of a discrete random variable X with respect to a probability distribution $P(X)$ is:

$$\mathbb{E}_{X\sim P}[f(X)] = \sum_{X} f(X)\, P(X)$$

Variance

- *Variance* of a random variable X gives the measure of how much the values of X deviate from the expected value as we sample X from $P(X)$

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

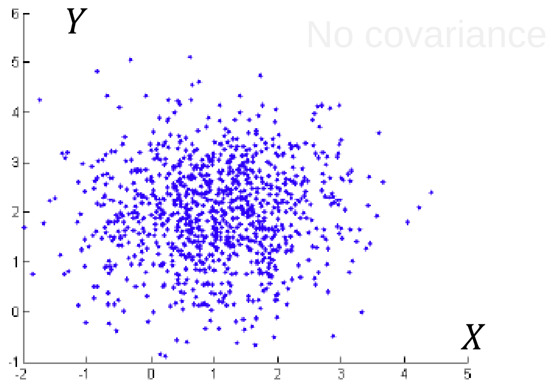
- When the variance is low, the values of X cluster near the expected value
- Variance is commonly denoted with σ^2
  - The above equation is similar to an expected value of a function $f(X) = (X_i - \mu)^2$
  - We can write:

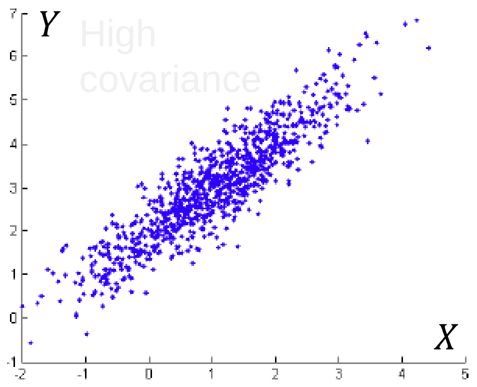
$$\sigma^2 = \mathbb{E}[(X_i - \mu)^2] = \sum_i (X_i - \mu)^2 \cdot P(X_i)$$

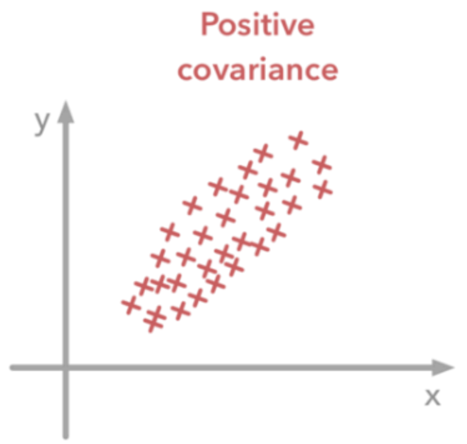
  - Similarly, the variance of a sample of observations can be calculated as:
  - $\sigma^2 = \frac{1}{N} \sum_i (X_i - \mu)^2$
- The square root of the variance is the *standard deviation*
  - $\sigma = \sqrt{\mathrm{Var}(X)}$

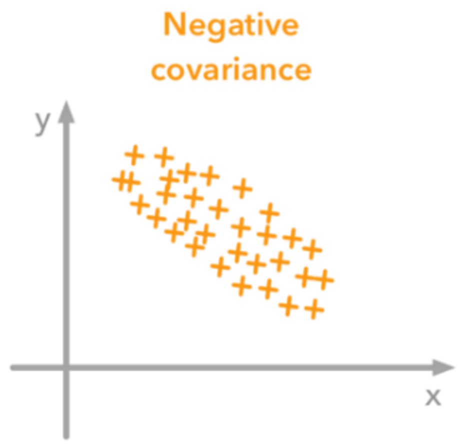
Covariance

- *Covariance* is a measure of the joint variability of two random variables X and Y from their means
$$\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$
- If $f(X) = X_i - \mu_X$ and $g(Y) = Y_i - \mu_Y$
  - Then, the covariance is: $\mathrm{Cov}(X_i, Y_i) = \mathbb{E}[f(X)g(Y)] = \sum_i P(X_i, Y_i) \cdot (X_i - \mu_X) \cdot (Y_i - \mu_Y)$
  - Covariance of samples of observations is: $\mathrm{Cov}(X, Y) = \frac{1}{N}\sum_i (Y_i - \mu_X)(Y_i - \mu_Y)$
- The covariance measures the tendency for X and Y to deviate from their means in the same (or opposite) directions at same time







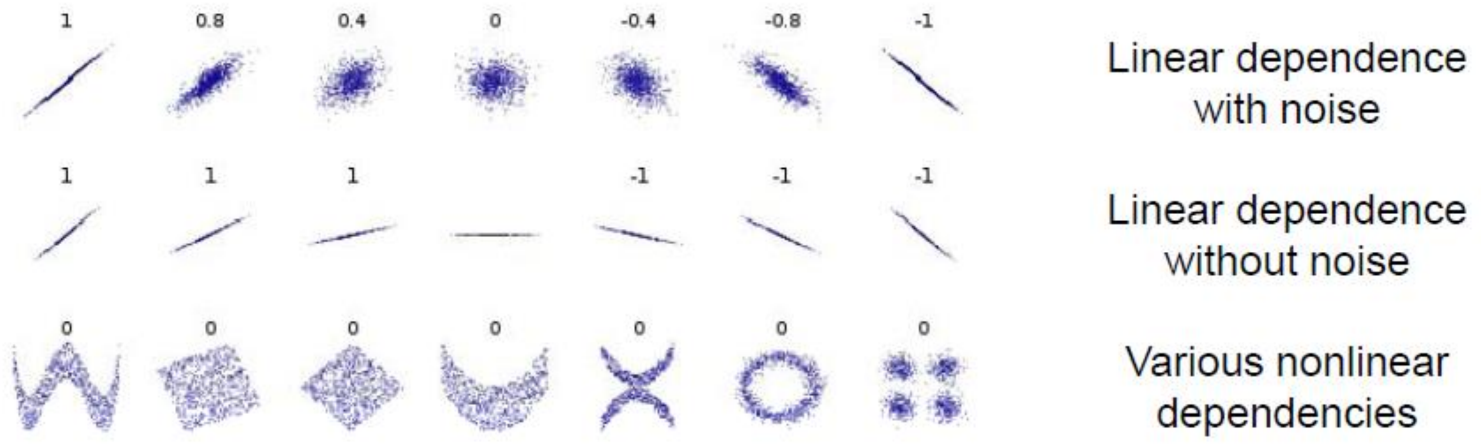


Correlation

- *Correlation coefficient* is the covariance normalized by the standard deviations of the two variables

$$\mathrm{corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y}$$

  - It is also called Pearson's correlation coefficient and it is denoted $\rho(X,Y)$
  - The values are in the interval $[-1, 1]$
  - It only reflects linear dependence between variables, and it does not measure non-linear dependencies between the variables



Covariance Matrix

- *Covariance matrix* of a multivariate random variable $\mathbf{X} = (X_1, X_2, \dots, X_n)^T$ is an $n \times n$ matrix, such that
$$\text{Cov}(\mathbf{X})_{i,j} = \text{Cov}(X_i, X_j)$$

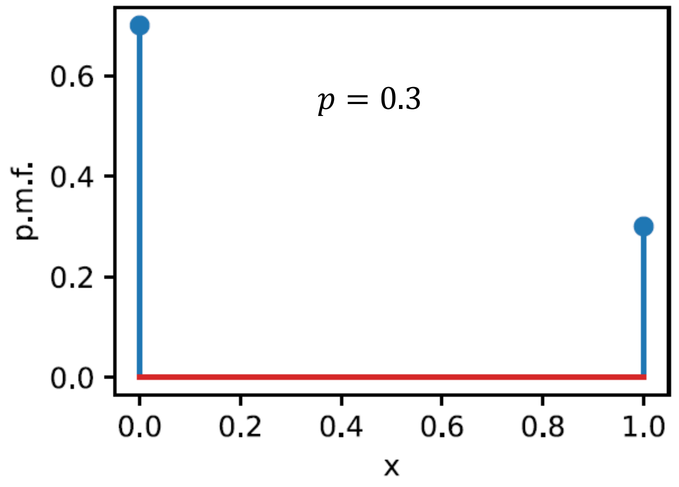
- i.e.,
$$\text{Cov}(\mathbf{X}) = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & & \ddots & \text{Cov}(X_2, X_n) \\ \vdots & & & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \cdots & \text{Cov}(X_n, X_n) \end{bmatrix}$$

- The diagonal elements of the covariance matrix are the variances of the elements of the random vector $\mathbf{X}$
$$\text{Cov}(X_i, X_i) = \text{Var}(X_i)$$

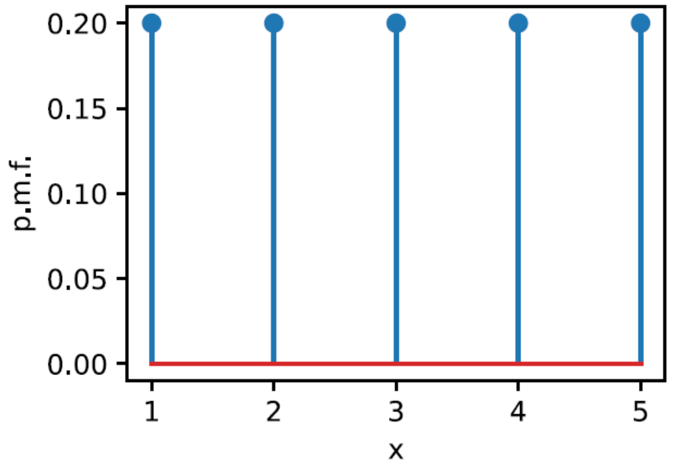
- Also note that the covariance matrix is symmetric, since $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$

Probability Distributions

- ***Bernoulli distribution***
  - Binary random variable X with states $\{0, 1\}$
  - E.g., the random variable can encode a coin flip which comes up 1 with probability p and 0 with probability $1 - p$
  - Notation: $X \sim Bernoulli(p)$



- ***Uniform distribution***
  - The probability of each value $i \in \{1,2, \dots, n\}$ is $p_i = \frac{1}{n}$
  - Notation: $X \sim U(n)$
  - Figure: $n = 5,\ p = 0.2$



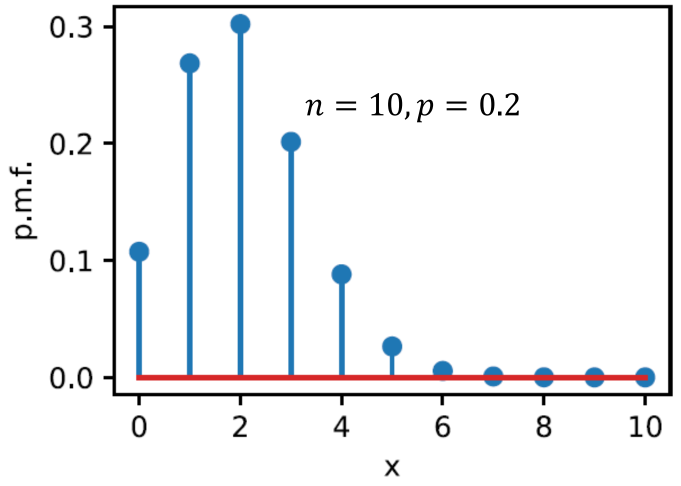
Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/distributions.html

Probability Distributions

- ***Binomial distribution***
  - Performing a sequence of n independent experiments, each of which has probability p of succeeding, where $p \in \{0, 1\}$
    - E.g., tossing a coin 100 times, head probability is 0.5
  - The probability of getting k successes in n trials is

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

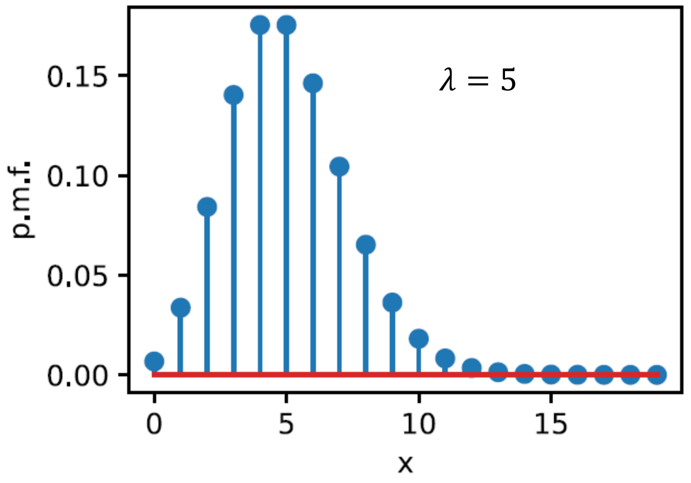
  - Notation: $X \sim Binomial(n, p)$



- ***Poisson distribution***
  - A number of events occurring independently in a fixed interval of time with a known rate λ
    - E.g., number of arriving patients in ER
  - A discrete random variable X with states $k \in \{0, 1, 2, \ldots\}$ has probability

$$P(X = k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!}$$

  - The rate λ is the average number of occurrences of the event
  - Notation: $X \sim Poisson(\lambda)$



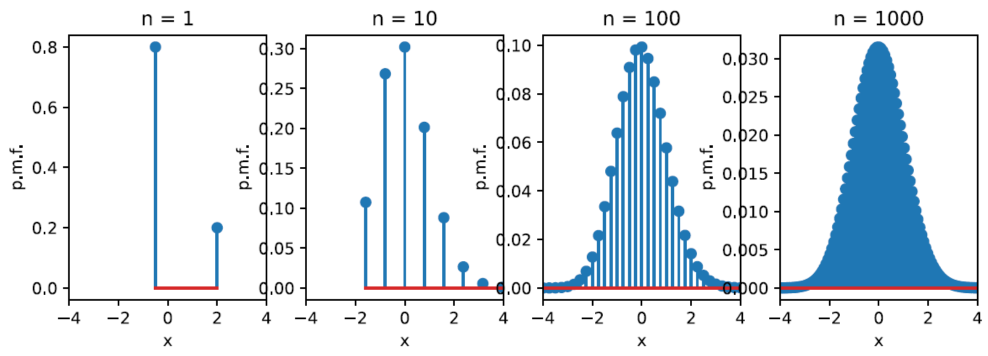
Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/distributions.html

Probability Distributions

- *Gaussian distribution*
  - The most well-studied distribution
    - Referred to as normal distribution or informally bell-shaped distribution
  - Defined with the mean μ and variance σ^2
    - Notation: $X \sim \mathcal{N}(\mu, \sigma^2)$
  - For a random variable X with n independent measurements, the density is

$$P_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - E.g., shown below is a Binomial distribution; as the number of experiments increases from 1 to 1000, it yields a Gaussian distribution
    - ***Central limit theorem***: the distribution of the mean of samples approximates a normal distribution as the sample size becomes larger



Picture from: http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/distributions.html

Probability Distributions

- *Multinoulli distribution*
  - It is an extension of the Bernoulli distribution, from binary class to multi-class
  - Multinoulli distribution is also called categorical distribution or generalized Bernoulli distribution
  - Multinoulli is a discrete probability distribution that describes the possible results of a random variable that can take on one of k possible categories
    - A categorical random variable is a discrete variable with more than two possible outcomes (such as the roll of a die)
  - For example, in multi-class classification in machine learning, we have a set of data examples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, and corresponding to the data example $\mathbf{x}_i$ is a k-class label $\mathbf{y}_i = \{y_{i1}, y_{i2}, \dots, y_{ik}\}$ representing one-hot encoding
    - One-hot encoding is also called 1-of-k vector, where one element has the value 1 and all other elements have the value 0
    - Let's denote the probabilities for assigning the class labels to a data example by $\{p_1, p_2, \dots, p_k\}$
    - We know that $0 \leq p_j \leq 1$ and $\sum p_j = 1$ for the different classes $j = 1, 2, \dots, k$
    - The multinoulli probability of the data example $\mathbf{x}_i$ is $P(\mathbf{x}_i) = {p_1}^{y_{i1}} \cdot {p_2}^{y_{i2}} \cdots {p_k}^{y_{ik}} = \prod_j {p_j}^{y_{ij}}$
    - Similarly, we can calculate the probability of all data examples as $\prod_i \prod_j {p_j}^{y_{ij}}$

Information Theory

- *Information theory* studies encoding, decoding, transmitting, and manipulating information
  - It is a branch of applied mathematics that revolves around quantifying how much information is present in different signals

- As such, information theory provides fundamental language for discussing the information processing in computer systems
  - E.g., machine learning applications use the cross-entropy loss, derived from information theoretic considerations

- A seminal work in this field is the paper *A Mathematical Theory of Communication* by Clause E. Shannon, which introduced the concept of information entropy for the first time
  - Information theory was originally invented to study sending messages over a noisy channel, such as communication via radio transmission

Self-information

- The basic intuition behind information theory is that learning that an unlikely event has occurred is more informative than learning that a likely event has occurred
  - E.g., a message saying "the sun rose this morning" is so uninformative that it is unnecessary to be sent
  - But a message saying "there was a solar eclipse this morning" is very informative

- Based on that intuition, Shannon defined the ***self-information*** of an event X as

$$I(X) = -\log(P(X))$$

  - $I(X)$ is the self-information, and$P(X)$ is the probability of the event X

- The self-information outputs the bits of information received for the event X
  - For example, if we want to send the code "0010" over a channel
  - The event "0010" is a series of codes of length n (in this case, the length is n =4)
  - Each code is a bit (0 or 1), and occurs with probability of $\frac{1}{2}$; for this event $P = \frac{1}{2^n}$

$$I(\text{"0010"}) = -\log(P(\text{"0010"})) = -\log\left(\frac{1}{2^4}\right) = -\log_2(1) + \log_2(2^4) = 0 + 4 = 4 \text{ bits}$$

Entropy

- For a discrete random variable X that follows a probability distribution P with a probability mass function $P(X)$, the expected amount of information through ***entropy*** (or Shannon entropy) is

$$H(X) = \mathbb{E}_{X\sim P}[I(X)] = -\mathbb{E}_{X\sim P}[\log P(X)]$$

- Based on the expectation definition $\mathbb{E}_{X\sim P}[f(X)] = \sum_X P(X)f(X)$, we can rewrite the entropy as

$$H(X) = -\sum_X P(X) \log P(X)$$

- If X is a continuous random variable that follows a probability distribution P with a probability density function $P(X)$, the entropy is

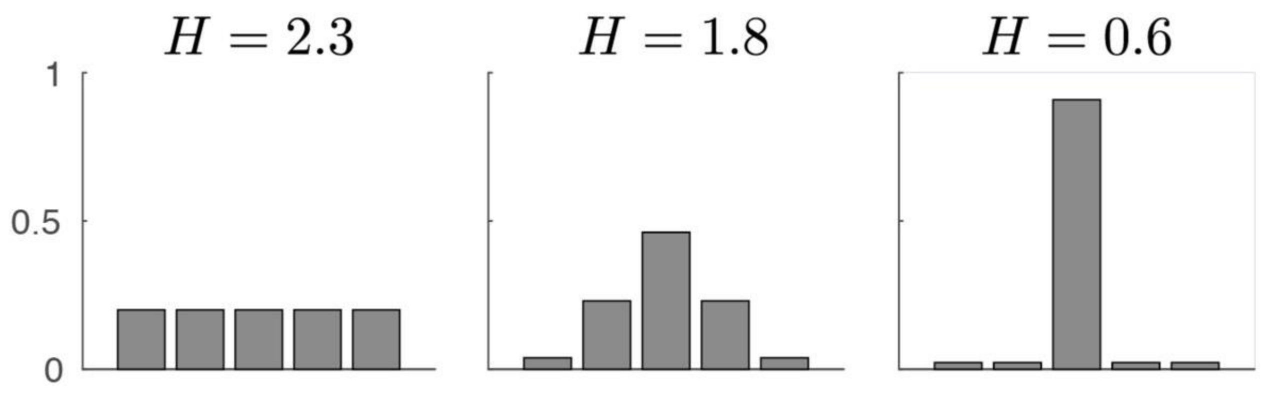
$$H(X) = -\int_X P(X) \log P(X)\, dX$$

  - For continuous random variables, the entropy is also called differential entropy

Entropy

- Intuitively, we can interpret the self-information ($I(X) = -\log(P(X))$) as the amount of surprise we have at seeing a particular outcome
  - We are less surprised when seeing a more frequent event

- Similarly, we can interpret the entropy ($H(X) = \mathbb{E}_{X\sim P}[I(X)]$) as the average amount of surprise from observing a random variable X
  - Therefore, distributions that are closer to a uniform distribution have high entropy
  - Because there is little surprise when we draw samples from a uniform distribution, since all samples have similar values



Kullback–Leibler Divergence

- ***Kullback-Leibler (KL) divergence*** (or relative entropy) provides a measure of how different two probability distribution are
- For two probability distributions $P(X)$ and $Q(X)$ over the same random variable X, the KL divergence is

$$D_{KL}(P||Q) = \mathbb{E}_{X\sim P}\left[\log\frac{P(X)}{Q(X)}\right]$$

- For discrete random variables, this formula is equivalent to

$$D_{KL}(P||Q) = \sum_X P(X)\log\frac{P(X)}{Q(X)}$$

- When base 2 logarithm is used, D_{KL} provides the amount of information in bits
  - In machine learning, the natural logarithm is used (with base e): the amount of information is provided in nats (natural unit of information)
- KL divergence can be considered as the amount of information lost when the distribution Q is used to approximate the distribution P
  - E.g., in GANs, P is the distribution of true data, Q is the distribution of synthetic data

Kullback–Leibler Divergence

- KL divergence is non-negative: $D_{KL}(P||Q) \geq 0$
- $D_{KL}(P||Q) = 0$ if and only if $P(X)$ and $Q(X)$ are the same distribution
- The most important property of KL divergence is that it is non-symmetric, i.e.,

$$D_{KL}(P||Q) \neq D_{KL}(Q||P)$$

- Because D_{KL} is non-negative and measures the difference between distributions, it is often considered as a "distance metric" between two distributions
  - However, KL divergence is not a true distance metric, because it is not symmetric
  - The asymmetry means that there are important consequences to the choice of whether to use $D_{KL}(P||Q)$ or $D_{KL}(Q||P)$

- An alternative divergence which is non-negative and symmetric is the ***Jensen-Shannon divergence***, defined as

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$

  - In the above, M is the average of the two distributions, $M = \frac{1}{2}(P + Q)$

Cross-entropy

- *Cross-entropy* is closely related to the KL divergence, and it is defined as the summation of the entropy $H(P)$ and KL divergence $D_{KL}(P||Q)$

$$CE(P,Q) = H(P) + D_{KL}(P||Q)$$

- Alternatively, the cross-entropy can be written as

$$CE(P,Q) = -\mathbb{E}_{X\sim P}\left[\log Q(X)\right]$$

- In machine learning, let's assume a classification problem based on a set of data examples $\{x_1, x_2, \dots, x_n\}$, that need to be classified into k classes
  - For each data example x_i we have a class label y_i
    - The true labels $\mathbf{y}$ follow the true distribution P
  - The goal is to train a classifier (e.g., a NN) parameterized by θ, that outputs a predicted class label $\hat{y}_i$ for each data example x_i
    - The predicted labels $\hat{\mathbf{y}}$ follow the estimated distribution Q
  - The cross-entropy loss between the true distribution P and the estimated distribution Q is calculated as:

$$CE(\mathbf{y},\hat{\mathbf{y}}) = -\mathbb{E}_{X\sim P}\left[\log Q(X)\right] = -\sum_{X} P(X)\log Q(X) = -\sum_{i} y_i \log \hat{y}_i$$

    - The further away the true and estimated distributions are, the greater the cross-entropy loss is.

Maximum Likelihood

- Cross-entropy is also related to the ***maximum likelihood*** estimation

- In ML, we want to find a model with parameters θ that maximize the probability that the data is assigned the correct class, i.e., $\text{argmax}_\theta\, P(\text{model} \mid \text{data})$
  - For the classification problem from previous page, we want to find parameters θ so that for the data examples $\{x_1, x_2, \dots, x_n\}$ the probability of outputting class labels $\{y_1, y_2, \dots, y_n\}$ is maximized
    - I.e., for some data examples, the predicted class $\hat{y}_j$ will be different than the true class y_j, but the goal is to find θ that results in an overall maximum probability

- From Bayes' theorem, argmax $P(\text{model} \mid \text{data})$ is proportional to argmax $P(\text{data} \mid \text{model})$

$$P(\theta | x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n | \theta)\, P(\theta)}{P(x_1, x_2, \dots, x_n)}$$

  - This is true since $P(x_1, x_2, \dots, x_n)$ does not depend on the parameters θ
  - Also, we can assume that we have no prior assumption on which set of parameters θ are better than any others

- Recall that $P(\text{data}|\text{model})$ is the likelihood, therefore, the maximum likelihood estimate of θ is based on solving

$$\arg\max_{\theta} P(x_1, x_2, \dots, x_n | \theta)$$

Maximum Likelihood

- For a total number of n observed data examples $\{x_1, x_2, \dots, x_n\}$, the predicted class labels for the data example x_i is $\hat{\mathbf{y}}_i$
  - Using the multinoulli distribution, the probability of predicting the true class label $\mathbf{y}_i = \{y_{i1}, y_{i2}, \dots, y_{ik}\}$ is

    $$\mathcal{P}(x_i \,|\theta) = \prod_j \hat{y}_{ij}{}^{y_{ij}}, \text{ where } j \in \{1,2,\dots,k\}$$
  - E.g., we have a problem with 3 classes [car, house, tree], and an image of a car x_i, the true label $\mathbf{y}_i = [1,0,0]$, and let's assume a predicted label $\hat{\mathbf{y}}_i = [0.7, 0.1, 02]$, then the probability is
  - $$\mathcal{P}(x_i \,|\theta) = \prod_j \hat{y}_{ij}{}^{y_{ij}} = 0.7^1 \cdot 0.1^0 \cdot 0.2^0 = 0.7 \cdot 1 \cdot 1 = 0.7$$
- Assuming that the data examples are independent, the likelihood of the data given the model parameters θ can be written as

  $$\mathcal{P}(x_1, x_2, \dots, x_n|\theta) = \mathcal{P}(x_1|\theta) \cdots \mathcal{P}(x_n \,|\theta) \; = \prod_j \hat{y}_{1j}{}^{y_{1j}} \cdot \prod_j \hat{y}_{2j}{}^{y_{2j}} \cdots \prod_j \hat{y}_{nj}{}^{y_{nj}} = \prod_i \prod_j \hat{y}_{ij}{}^{y_{ij}}$$

- Log-likelihood is often used because it simplifies numerical calculations, since it transforms a product with many terms into a summation, e.g., $\log\left(a_1{}^{b_1} \cdot a_2{}^{b_2}\right) = b_1 \log(a_1) + b_2 \log(a_2)$

  $$\log \mathcal{P}(x_1, x_2, \dots, x_n|\theta) = \log\left(\prod_i \prod_j \hat{y}_{ij}{}^{y_{ij}}\right) = \sum_i \sum_j y_{ij} \log \hat{y}_{ij}$$
  - A negative of the log-likelihood allows us to use minimization approaches, i.e.,

    $$-\log \mathcal{P}(x_1, x_2, \dots, x_n|\theta) = -\sum_i \sum_j y_{ij} \log \hat{y}_{ij} = CE(\mathbf{y}, \hat{\mathbf{y}})$$
- Thus, maximizing the likelihood is the same as minimizing the cross-entropy

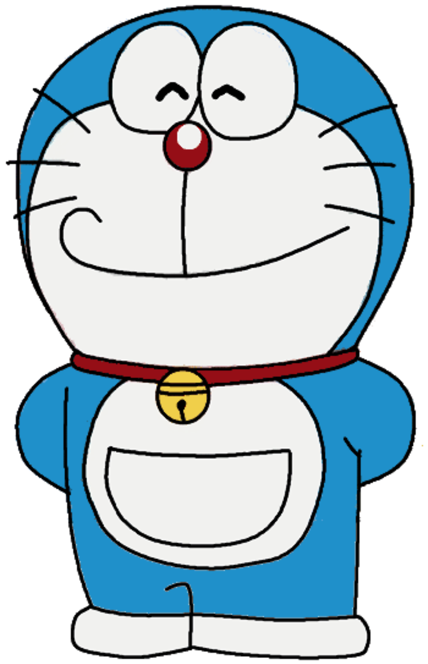
Slide Acknowledgements

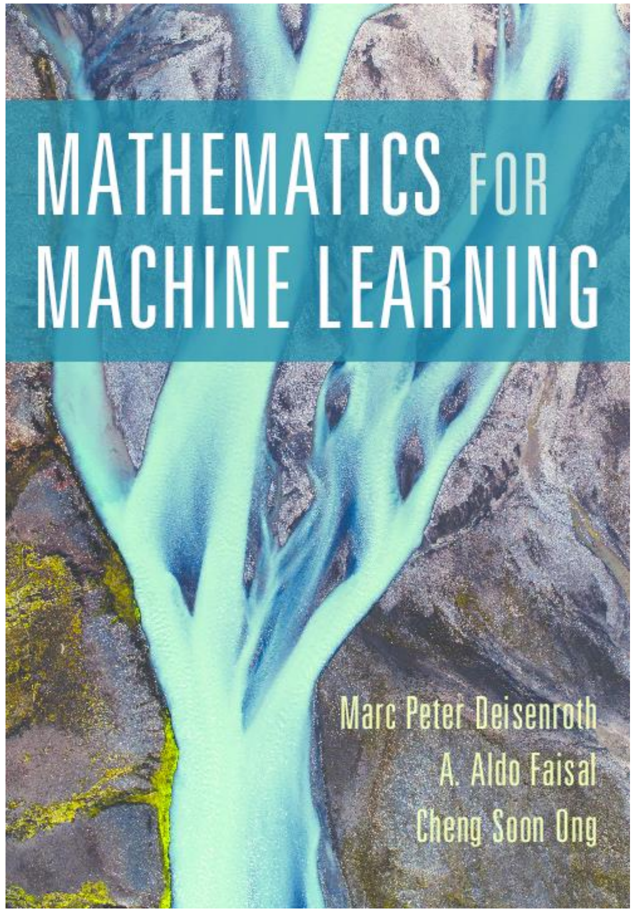
1. A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, *Dive into Deep Learning*, https://d2l.ai, 2020.
2. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2017.
3. Aleksandar (Alex) Vakanski— Machine Learning Math Essentials Presentation
4. Brian Keng – Manifolds: A Gentle Introduction blog
5. Martin J. Osborne – Mathematical Methods for Economic Theory (link)

Video Lectures

1. Linear Algebra for ML
2. Multivariate Calculus for ML
3. Statistics for ML
4. Optimization for ML
5. Stanford CS229 ML: https://cs229.stanford.edu/

REFERENCE





https://mml-book.github.io/