Fundamentals of Probability

Alexander G. Ororbia II and Viet Nguyen

Introduction to Machine Learning

CSCI-335

2/2/2026

NumPy Statistics Functions

Table 2-9. NumPy Functions for Calculating Aggregates of NumPy Arrays

NumPy Function	Description
`np.mean`	The average of all values in the array
`np.std`	Standard deviation
`np.var`	Variance
`np.sum`	The sum of all elements
`np.prod`	The product of all elements
`np.cumsum`	The cumulative sum of all elements
`np.cumprod`	The cumulative product of all elements
`np.min, np.max`	The minimum/maximum value in an array
`np.argmin, np.argmax`	The index of the minimum/maximum value in an array
`np.all`	Returns True if all elements in the argument array are nonzero
`np.any`	Returns True if any of the elements in the argument array is nonzero

Σ = summation (capital sigma)

Π = product (capital pi)

- Can calculate certain statistics over tensors

NumPy aggregation functions (aggregators)

Table 2-9. NumPy Functions for Calculating Aggregates of NumPy Arrays

NumPy Function	Description
`np.sum`	The sum of all elements
`np.prod`	The product of all elements
`np.min, np.max`	The minimum/maximum value in an array
`np.argmin, np.argmax`	The index of the minimum/maximum value in an array

Σ = summation (capital sigma)

Π = product (capital pi)

How I like to think about axis reduction: cross the reduced axis

Shape (x, y) and reduce axis 0 => shape (y,)

Shape (x, y) and reduce axis 1 => shape (x,)

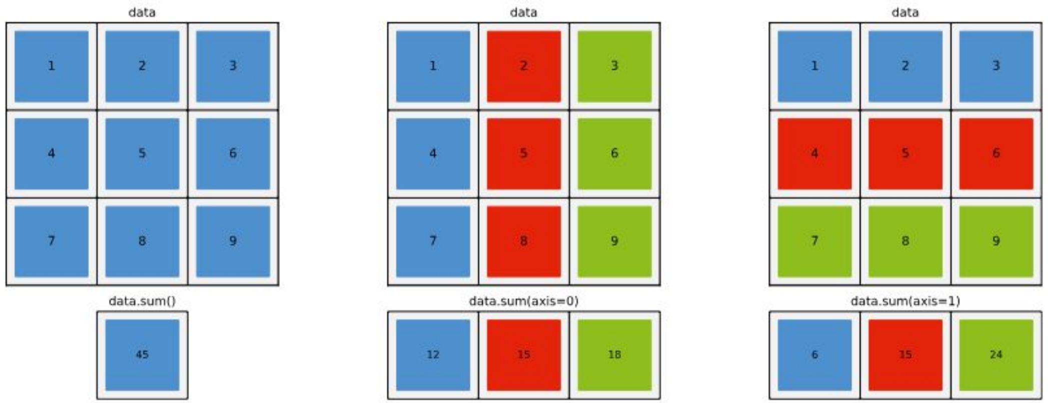


***Figure 2-3.** Illustration of array aggregation functions along all axes (left), the first axis (center), and the second axis (right) of a two-dimensional array of shape 3 × 3*

Elementwise composed functions

We compose/create functions out of the basic/elemental elementwise functions we saw before!

- Applied to each element (i, j) of matrix/vector argument
  - *Can build from simple routines:* *cos(.)*, *sin(.)*, *exp(.)*, etc. (the "." means argument)
- ***Identity***: $\phi(\mathbf{v}) = \mathbf{v}$
- ***Logistic Sigmoid***: $\phi(\mathbf{v}) = \sigma(\mathbf{v}) = \frac{1}{1+e^{-\mathbf{v}}}$
- ***Softmax***: $\phi(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_{c=1}^{C} \exp(\mathbf{v}_c)}$ $\quad \mathbf{v} \in \mathbb{R}^C$
- ***Linear Rectifier***: $\phi(\mathbf{v}) = \max(0, \mathbf{v})$

$\varphi\Big($

1.0	-1.4
-0.69	1.8

$\Big) =$

Elementwise composed functions

We compose/create functions out of the basic/elemental elementwise functions we saw before!

- Applied to each element (i, j) of matrix/vector argument
  - *Can build from simple routines:* *cos(.)*, *sin(.)*, *exp(.)*, etc. (the "." means argument)
- ***Identity***: $\phi(\mathbf{v}) = \mathbf{v}$
- ***Logistic Sigmoid***: $\phi(\mathbf{v}) = \sigma(\mathbf{v}) = \frac{1}{1+e^{-\mathbf{v}}}$
- ***Softmax***: $\phi(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_{c=1}^{C} \exp(\mathbf{v}_c)}$ $\quad \mathbf{v} \in \mathbb{R}^C$
- ***Linear Rectifier***: $\phi(\mathbf{v}) = \max(0, \mathbf{v})$

$$\varphi\left(\begin{bmatrix} 1.0 & -1.4 \\ -0.69 & 1.8 \end{bmatrix}\right) = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.8 \end{bmatrix}$$

Elementwise composed functions

- *Can build from simple routines:*
  cos(.), *sin(.)*, *exp(.)*, etc. (the "." means argument)

Softmax: $\phi(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_{c=1}^{C} \exp(\mathbf{v}_c)}$

Sigmoid: $\phi(\mathbf{v}) = \sigma(\mathbf{v}) = \frac{1}{1+e^{-\mathbf{v}}}$

Let's write these out to our Python interpreter!

NumPy Function
np.cos, np.sin, np.tan
np.arccos, np.arcsin, np.arctan
np.cosh, np.sinh, np.tanh
np.arccosh, np.arcsinh, np.arctanh
np.sqrt
np.exp
np.log, np.log2, np.log10

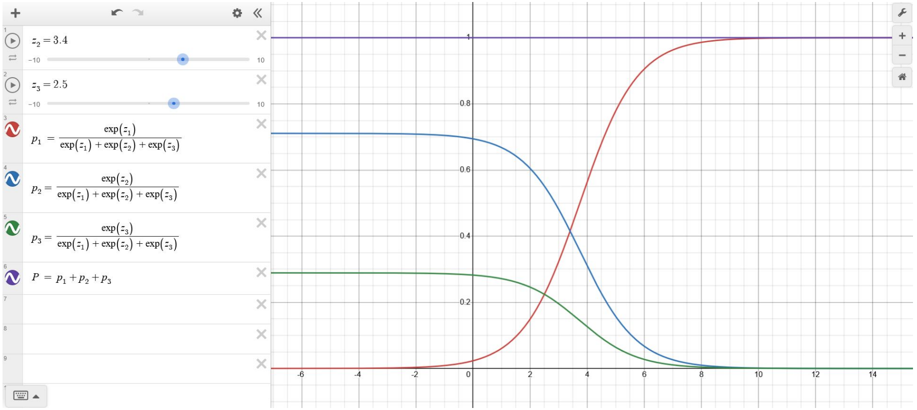
NumPy Function
np.add, np.subtract, np.multiply, np.divide
np.power
np.remainder
np.reciprocal
np.real, np.imag, np.conj
np.sign, np.abs
np.floor, np.ceil, np.rint
np.round

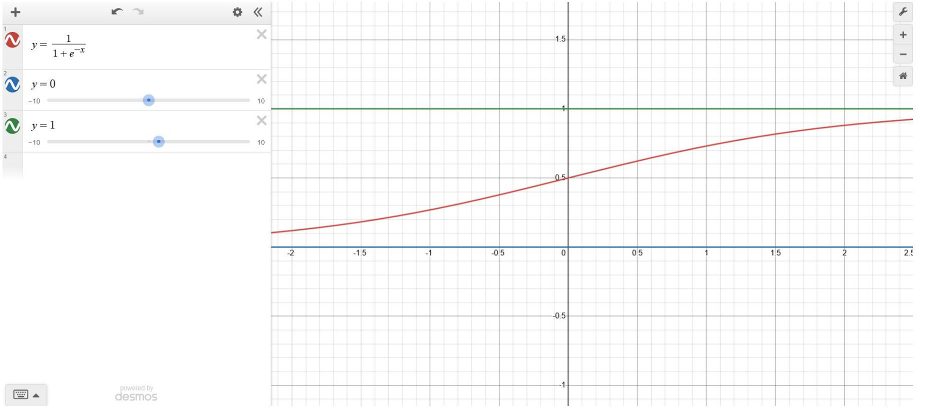
Two Important Functions in ML

- *Can build from simple routines:*
  cos(.), *sin(.)*, *exp(.)*, etc. (the "." means argument)

Softmax: $\phi(\mathbf{v}) = \frac{\exp(\mathbf{v})}{\sum_{c=1}^{C} \exp(\mathbf{v}_c)}$

Sigmoid: $\phi(\mathbf{v}) = \sigma(\mathbf{v}) = \frac{1}{1+e^{-\mathbf{v}}}$





https://www.desmos.com/calculator/tsxekspaud

https://www.desmos.com/calculator/coknirwubg

Tensor logical expression functions

Table 2-10. NumPy Functions for Conditional and Logical Expressions

Function	Description
np.where	Chooses values from two arrays depending on the value of a condition array
np.choose	Chooses values from a list of arrays depending on the values of a given index array
np.select	Chooses values from a list of arrays depending on a list of conditions
np.nonzero	Returns an array with indices of nonzero elements
np.logical_and	Performs an elementwise AND operation
np.logical_or, np.logical_xor	Elementwise OR/XOR operations
np.logical_not	Elementwise NOT operation (inverting)

Uncertainty

Let action A_t = leave for airport $_t$ minutes before flight
Will A_t get me there on time?

Problems:

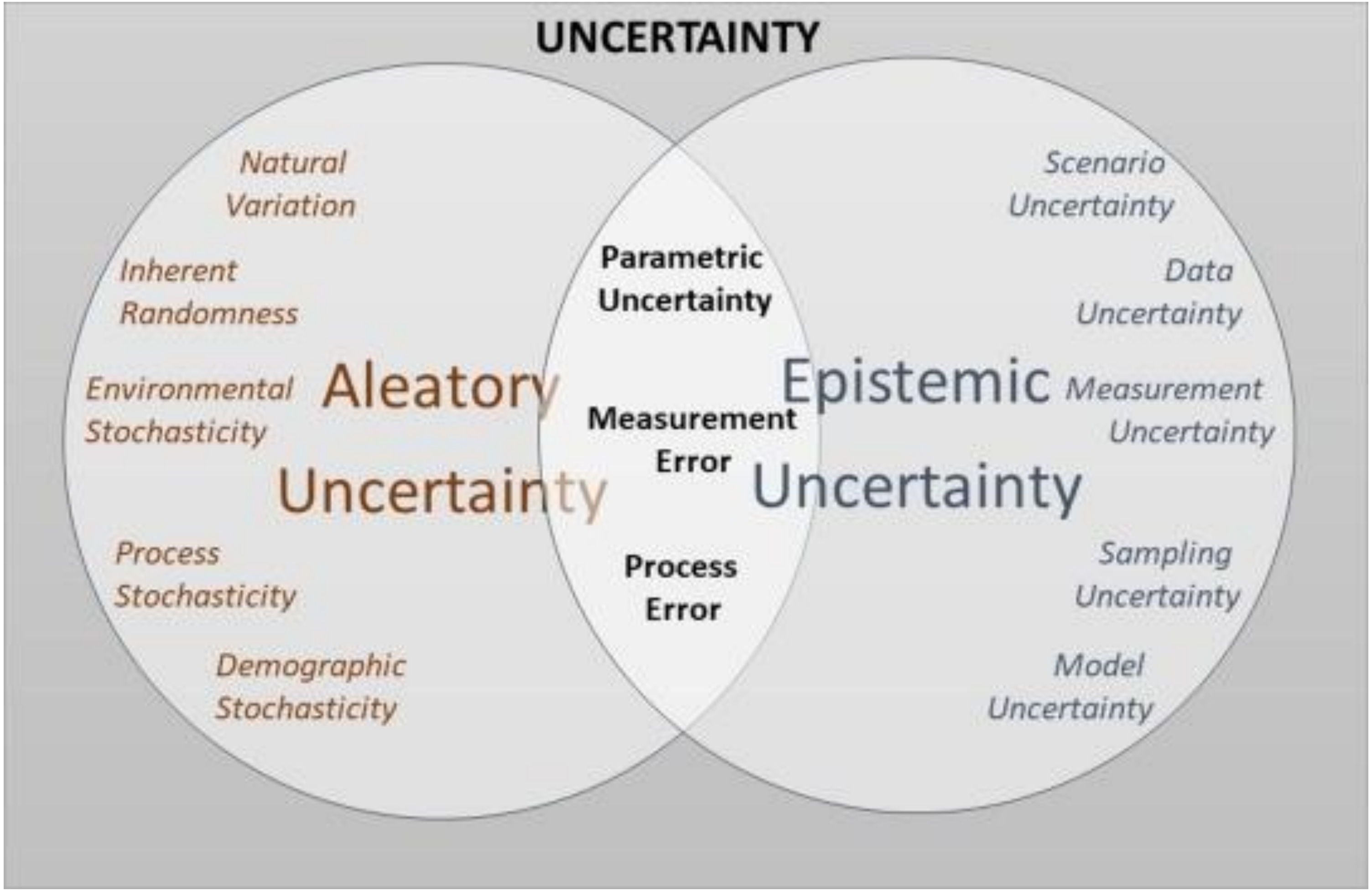
1. partial observability (road state, other drivers' plans, etc.)
2. noisy sensors (traffic reports)
3. uncertain (non-deterministic) action outcomes (flat tire, etc.)
4. immense complexity of modeling and predicting traffic

Set of actions:
{A_1, A_2,...A_t,...,A_T}

Hence a purely logical approach either

1. risks falsehood: "A_{25} will get me there on time", or
2. leads to conclusions that are too weak for decision making: "A_{25} will get me there on time if there's no accident on the bridge and it doesn't rain and my tires remain intact etc etc."

A_{1440} might reasonably be said to get me there on time but I'd have to stay overnight in the airport …


UNCERTAINTY
Natural Variation
Inherent Randomness
Environmental Stochasticity
Aleatory Uncertainty
Process Stochasticity
Demographic Stochasticity
Parametric Uncertainty
Measurement Error
Process Error
Scenario Uncertainty
Data Uncertainty
Epistemic Uncertainty
Measurement Uncertainty
Sampling Uncertainty
Model Uncertainty

Probability in Context

Probability theory

- Branch of mathematics concerned with analysis of random phenomena
  - ***Randomness***: a non-order or non-coherence in a sequence of symbols or steps, such that there is no intelligible pattern or combination

- Central objects of probability theory are:
  random variables, stochastic processes, and **events**
  - Mathematical abstractions of non-deterministic events or measured quantities that may either be single occurrences or evolve over time in an apparently random fashion

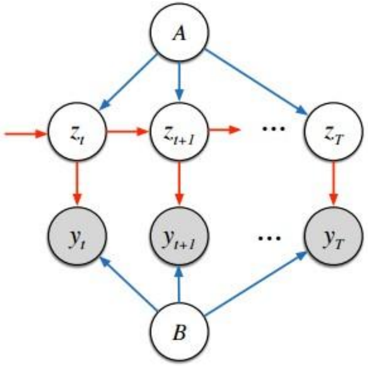
Uncertainty

- A lack of knowledge about an event
- Can be represented by a probability
  - Ex: role a die, draw a card
- Can be represented as an error

A statistic (a measure in **statistics**)

- Can use probability in determining that measure

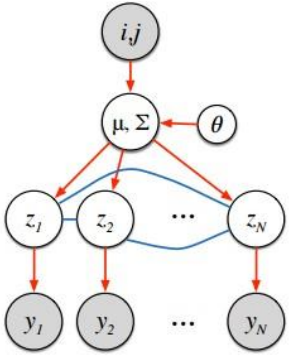
Why do we care? Probability allows us to build models of stochastic, data-generating processes; also, machine learning has historically/originally referred to as *statistical learning*



Gaussian Linear State Space Model
Kalman Filter

$$z_t \sim \mathcal{N}(z_t | A z_{t-1}, \sigma_z^2 I)$$

$$y_t \sim \mathcal{N}(y_t | B z_t, \sigma_y^2 I)$$

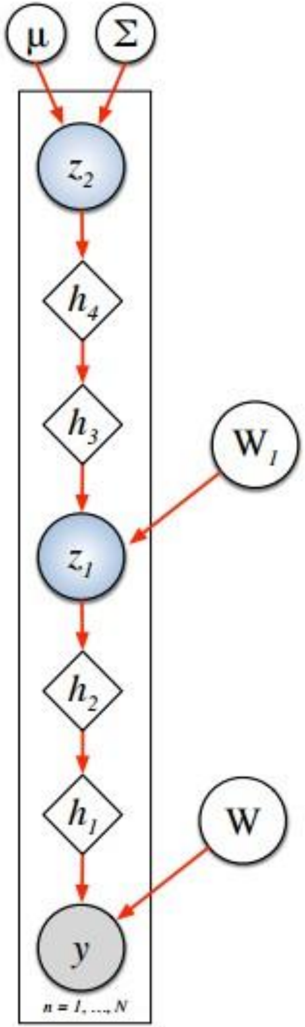


Latent Gaussian Cox Point Process

$$x \sim \mathcal{N}(x | \mu(i,j), \Sigma(i,j))$$

$$y_{ij} \sim \mathcal{P}(c \exp(x_{ij}))$$

Probabilistic graphical models (PGMs)



Why do we care? Probability allows us to build models of stochastic, data-generating processes; also, machine learning has historically/originally referred to as *statistical learning*

Examples:

- Medical diagnostic tool: the model doesn't strictly know if a patient has a bladder cancer; it calculates the likelihood
- We gain insight through the quantification of aleatoric and epistemic uncertainty
- Objective functions (used everywhere in modern ML models): MLE, cross-entropy
- ChatGPT and all generative models use probs to generate contents

Founders of Probability Theory

Blaise Pascal

(1623-1662, France)

Pierre Fermat

(1601-1665, France)

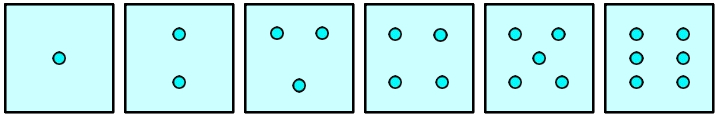
Laid the foundations of the probability theory in a correspondence on a dice game posed by a French nobleman

<u>Sample Spaces</u>: Measures of Events

Collection (list) of all possible outcomes

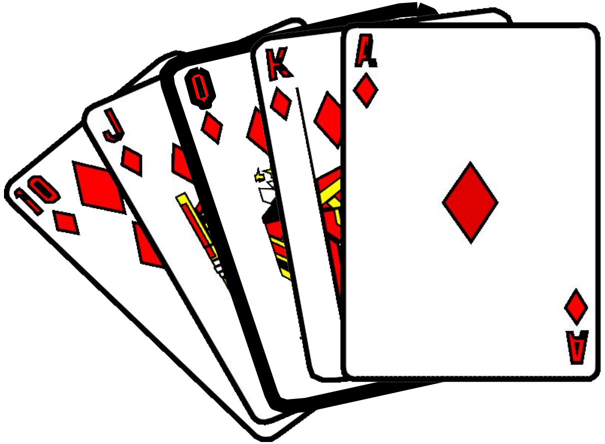
Experiment: Roll a die!

– e.g.: All six faces of a die:



Experiment: Draw a card! (Any card!)

e.g.: All 52 cards in a deck:



Types of Events

Event

- Subset of sample space (set of outcomes of experiment)

Random event

- Different likelihoods of occurrence

Simple event

- Outcome from a sample space with one characteristic in simplest form
- e.g.: King of clubs from a deck of cards

Joint event

- Conjunction (AND, $\wedge$, ","); disjunction (OR, $\vee$)
- Contains several simple events
- e.g.: A red ace from a deck of cards – P(red ace $\vee$ ace of diamonds) (ace on hearts OR ace of diamonds)

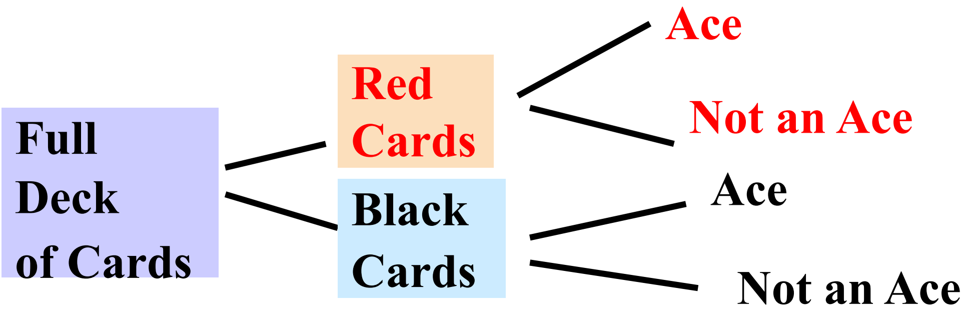
Visualizing Events

Excellent ways of determining probabilities, can be built from data

Contingency tables (nice way to look at probability):

	Ace	Not Ace	Total
Black	2	24	26
Red	2	24	26
Total	4	48	52

Tree diagrams:



Axioms of Probability

Given 2 events: x, y

1) P(x OR y) = P(x) + P(y) - P(x AND y);
   note for **mutually exclusive events** then P(x AND y) = 0
2) P(x and y) = P(x) * P(y|x), also written as P(y|x) = P(x and y)/P(x)
3) If x and y are ***independent***, P(y|x) = P(y), thus P(x AND y) = P(x) * P(y)
4) P(x) > P(x) * P(y); P(y) > P(x) * P(y) [a property!]

Maximum Likelihood Estimation (MLE)

- Uses relative frequencies as estimates
- Maximizes likelihood of training data D under a simple model M, or *P*(D|M)

- With discrete data, we can employ a *counting function* **c(**A=a**)**, that returns frequency of a particular value taken on by attribute A
  - ***Note***: **c(**A=a**)** is actually **c(**A=a, *D*), where *D* is a dataset

- **Issue:** What happens with sparse data?

You're thinking like a frequentist now!

An Example: A Unigram Language Model

w_i is particular word in W, where W is set of unique words (or vocabulary)

- Do not use history:

Probability of a word given a word sequence/history →

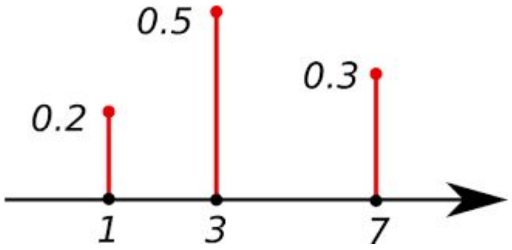
$$P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i) = \frac{c(w_i)}{\sum_{\tilde{w}} c(\tilde{w})}$$

i live in osaka . </s>	P(nara) = 1/20 = 0.05
i am a graduate student . </s>	P(i) = 2/20 = 0.1
my school is in nara . </s>	P(</s>) = 3/20 = 0.15

P(W=i live in nara . </s>) =
$0.1 * 0.05 * 0.1 * 0.05 * 0.15 * 0.15 = 5.625 * 10^{-7}$

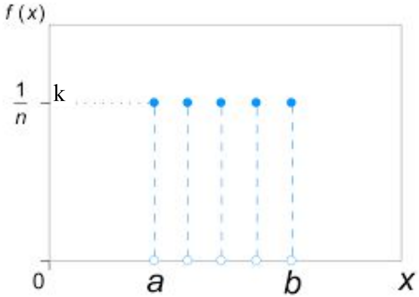
Probability Mass Function (PMF)

- For a discrete random variable X, the PMF f(x) is defined as f(x)=P(X=x), representing the probability that X takes the specific value x.
- The domain of P must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$. An impossible event has probability 0 and no state can be less probable than that. Likewise, an event that is guaranteed to happen has probability 1, and no state can have a greater chance of occurring.
- $\sum_{x \in \mathrm{x}} P(x) = 1$. We refer to this property as being **normalized**. Without this property, we could obtain probabilities greater than one by computing the probability of one of many events occurring.



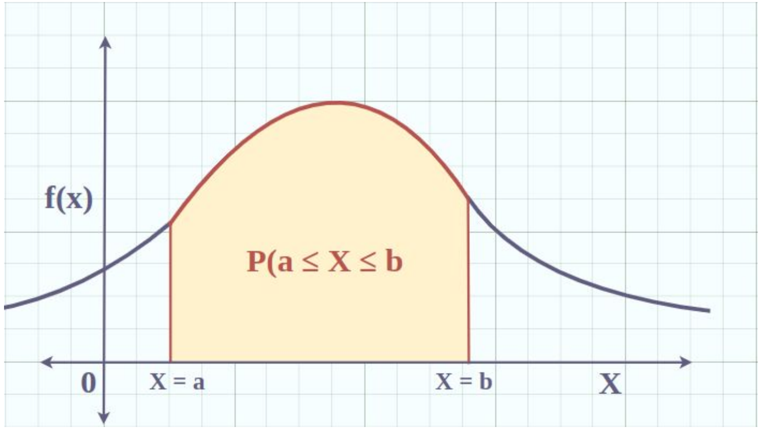
P(X=1) = 0.2;
P(X=3) = 0.5,
and so on

Example: uniform distribution: $$P(\mathrm{x} = x_i) = \frac{1}{k}$$



Probability Density Function (PDF)

- The domain of p must be the set of all possible states of x.
- $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.
- $\int p(x)dx = 1$.



Example: uniform distribution:

$$u(x; a, b) = \frac{1}{b-a}.$$