Interview Questions

Chan W. Kim

August 26, 2022

1 Coding Questions

1.1 How do you Debug?

- Print statements in between every statements
- time statements in between possible bottlenecks to fine which code is clotting

2 Algorithms Questions

The Algorithms (finite sequence of instructions) questions were drawn from: src1

2.1 How do compare algorithms?

- Complexity of Time: running time of a program
- Complexity of Space: Required space to execute the program

2.2 Best,average, or worst scenario of an algorithm?

- Performance is defined by **asymptotic analysis**: what happens at end limits
- For ex, $O(n^2)$, $O(n \log(n))$ for time complexity

2.3 Asymptotic notations

- Θ notation: Upper and Lower bounds on the function: Drop low order terms and ignore leading coefficients
- $\emptyset$ notation: Upper bound for the algorithm
- Ωnotation: Lower bound for the algorithm

2.4 Swap two numbers in Java

Using addition and subtraction;

- a = a + b;
- b = a - b; this is like (a+b) - b
- a = a - b; this is like (a+b) - a

XOR (only java) bitwise operator:

- x = xˆy;
- y = xˆy;
- x = xˆy;

2.5 Divide and Conquer algorithm paradigm

Divide and Conquer is a algorithm paradigm which handles a large amount of data by splitting it down into smaller chunks and determine the solution to the problem for each small chunks. Then, it combines all of the piecewise solutions to form a single global solution.

- Separate the problem into set of sub-problems.
- The algorithm solves each sub-problem individually.
- The algorithm combines the solutions to obtain a singular overall solution.

Examples of Divide and Conquer algorithms are:

- Binary Search
- Merge Sort
- Strassen's Matrix Multiplication
- Quick Sort
- Closest pair of points

2.6 Greedy Algorithms

- A greedy algorithm aims to choose the best optimal decision at each sub-step, eventually leading to a globally optimal solution.

- It selects the best immediate or local option.
- Less then perfect and often worse then ideal solutions

2.7 A few types of searching algorithms

Searching algorithms are divided into two categories:

- Sequential Search: Traverse the elements consecutively, checking each element. Ex: Linear Search algorithm.
- Interval Search: Created for sorted data structures: Target the center of the search structure and divide the search in half

2.8 Linear Search Algorithm

Linear search algorithm traverses the list of elements from the beginning to the end and inspect the properties of all the elements encountered along the way.

2.9 Binary Search Algorithm

- Binary search algorithm requires the list of elements to be sorted.
- Based on Divide and Conquers algorithm paradigm
- If the search key's value is less than the item in the interval's **midpoint**, the interval should be narrowed to the lower half.
- If more, narrowed to upper half.

2.10 Algorithm for adding a node to a linked list sorted in ascending order + maintain the order

Do something like a binary search:

- Check if the value to be added to the list is greater than or less than the value of the element at the midpoint of the list

2.11 Binary Search Tree

leaf node is a node with no children nodes.

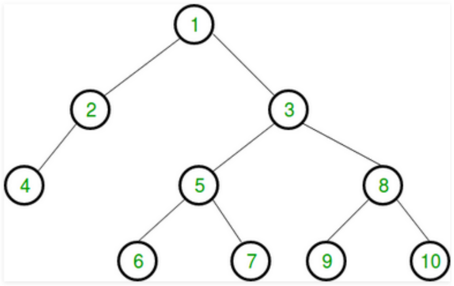


Figure 1: Binary Tree

- Start from the root.
- Compare the searching element with root, if less than root, then recursively call left subtree, else recursively call right subtree.
- If the element to search is found anywhere, return true, else return false.

2.12 What is Data Encapsulation?

Data Encapsulation is an Object Oriented Programming concept that bind a group of related properties, functions, and other members are treated as a single unit.

3 Data Structure Questions

The data structures questions were drawn from: src1

3.1 Data Structure Types

There are two types of data structures:

- **Linear**: If the elements of a data structure result in a **sequence** or a linear list then it is a linear data structure. Exmaple: Arrays, Linked Lists, Stacks, Queues, *etc*.
  - Stack: Stack is a linear data structure which follows a **particular order** in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).
  - Queue: An abstract linear data structure such that its **open at both its ends**. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue).
- **Non-linear**: Data elements are not arranged *sequentially*. Examples: Trees and Graphs.
  - Tree: Consists of various nodes linked together in hierarchy that forms a relationship like that of a parent and a child.
  - Graph: Consists of a definite quantity of vertices and edges. The vertices or the nodes are involved in storing data and the edges show the vertices relationship. No rules for the connection of nodes (opposite of trees).
- **Either**: Hash Table can be implements as linear or non-linear data structures which consists of key-value pairs.

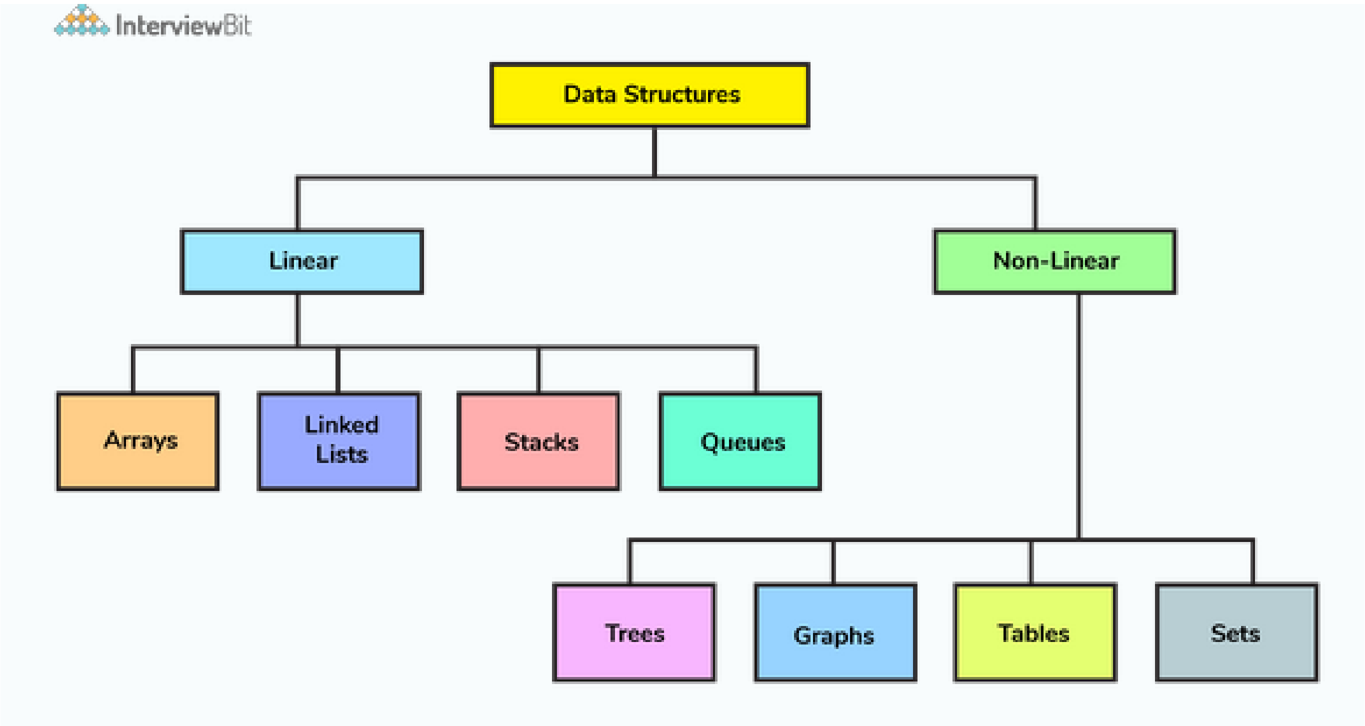


Figure 2: Data Structure Overview

3.2 Explain the difference between file structure and storage structure.

- File Structure: Representation of data into **secondary or auxiliary (additional) memory**. Any device such as hard disk or pen drives that stores data which remains **intact until manually deleted** is known as file structure representation.

- Storage Structure: In this data type, data is stored in the main memory (**RAM**) and is **deleted once the function that uses this data gets completely executed**.

3.3 What is an array?

- Arrays are the collection of similar types of data stored at **contiguous** memory locations.

- Arrays' data elements can be accessed randomly by using its **index**.

3.4 What is a multidimensional array?

4 STATS Questions

4.1 How to check if distribution is Normal? - Normality test

- In *descriptive statistics*, you can measure the goodness of fit of a normal model to the data: src1 src2

  - If the fit is poor, the distribution is not Gaussian
  - Check histograms
  - Check Q-Q plot of the same dataset, but 2 diff variables
    - plot two variables against one another. If straight → Gaussian!
  - KDE plot
  - Skewness
    - pd.Dataframe().skew() $\rightarrow -0.5 < skew < 0.5$ are assumed to be not skewed

- In *frequentist statistics statistical hypothetisis testing*, the data are tested against null hypothesis that it is normally distributed
- In Bayesian statistics, you *DON'T* test the normality, but rather

  - Compute likelihood that the data come from a normal distribution with given parameter (μ, σ) and compare that with the likelihood that the data come from other distributions.
    1. Using a Bayes factor - giving the relative likelihood of seeing the data given different models
    2. Taking a prior distribution on possible models and parameters and computing a posterior distribution given the computed likelihoods.

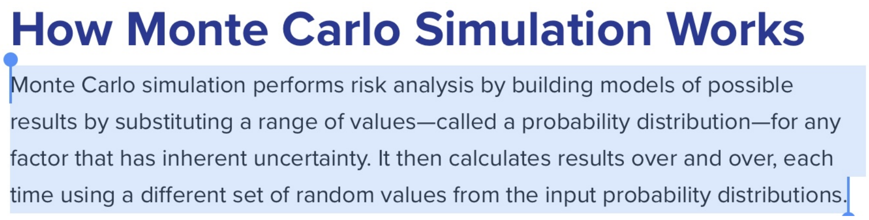
4.2 Central Limit Theorem?

4.3 P-value

Probability of having our data (measurement) if the two variables are actually same (if null hypothesis is true).

- P-value determines whether correlation coefficients is statistically significant (meaningful, trustworthy)
- Lower P value, better: two distributions are different, not by chance, but b/c the variables are in fact different.
- Probability of having a false positive.

4.4 Markov Chain Monte Carlo



How Monte Carlo Simulation Works

Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values—called a probability distribution—for any factor that has inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the input probability distributions.

Monte Carlo

- Randomly sample a probability distribution and approximate a desired quantity.
- Under the assumption that we can efficiently draw samples from the target distribution
- Then, we can estimate the sum or integral quantity as the mean or variance of the drawn sample

Markov Chain

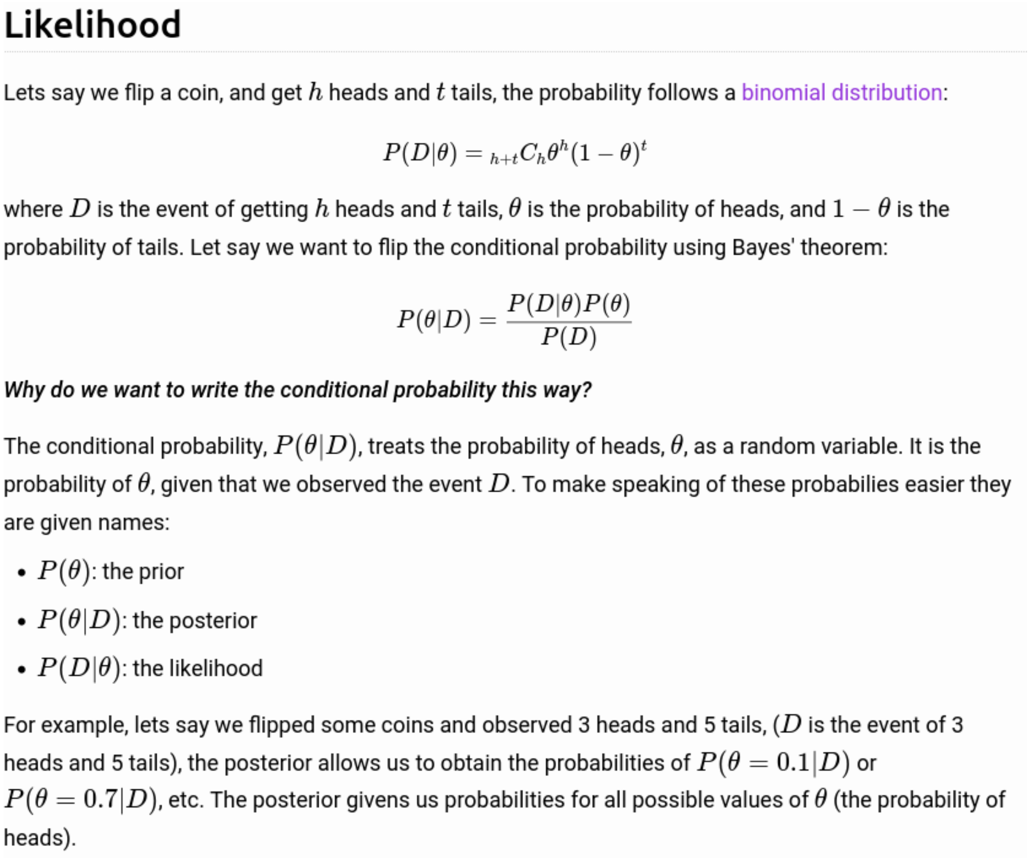
- Generate a sequence of random variables where the current value is probabilistically dependent on the pior variable.
- Selecting the next variable is only dependent upon the last variable in the chain
- Ex: Consider a board game that involves rolling dice.
  - The roll of a die has a uniform prob. distri. across 6 sides.
  - You have a position on the board, but your next position on the board is only based on the current position and the random roll of the dice.

Markov Chain Monte Carlo

- Simply put: Series of Sampling from previous probability distributions that are also sampled from a target distribution.
- Combine Markov Chain and Monte Carlo -¿ sampling high dimensional prob. distributions that honors the probabilistic dependence between samples by constructing a Markov chain that comprise the Monte Carlo sample.
- Samples are drawn from the probaility distribution by constructing a Markov Chain, where the next sample that is drawn from the probability distribution is dependent up the last same that was drawn.
- The idea is that the chain will settle on (find equilibrium) on the desired quanty we are inferring.

4.5 Bayesian Coin Flip

The information below was drawn from Bayesian coin flip source.



Likelihood

Lets say we flip a coin, and get h heads and t tails, the probability follows a binomial distribution:

$$P(D|\theta) = {}_{h+t}C_h \theta^h (1-\theta)^t$$

where D is the event of getting h heads and t tails, θ is the probability of heads, and $1-\theta$ is the probability of tails. Let say we want to flip the conditional probability using Bayes' theorem:

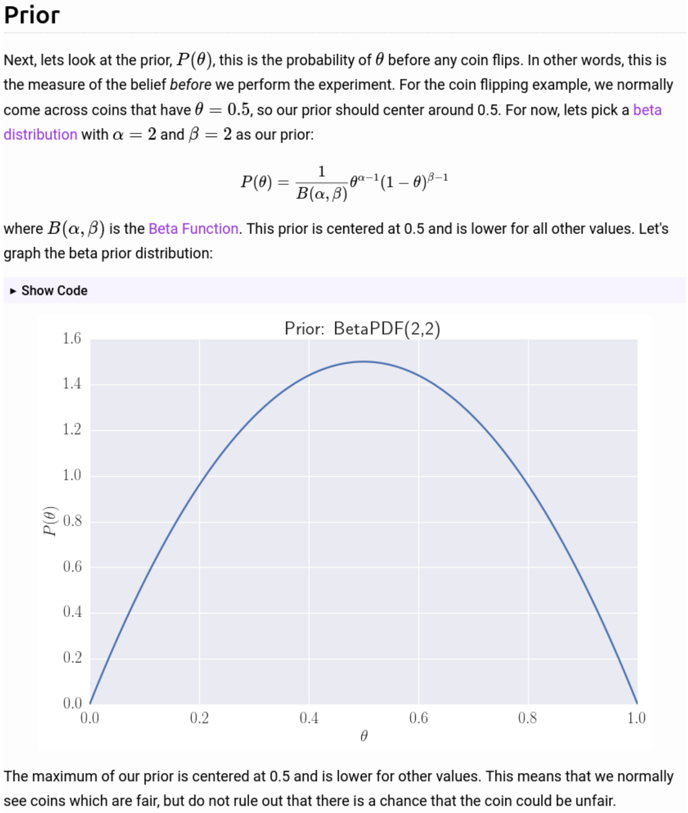
$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

Why do we want to write the conditional probability this way?

The conditional probability, $P(\theta|D)$, treats the probability of heads, θ, as a random variable. It is the probability of θ, given that we observed the event D. To make speaking of these probabilies easier they are given names:

- $P(\theta)$: the prior
- $P(\theta|D)$: the posterior
- $P(D|\theta)$: the likelihood

For example, lets say we flipped some coins and observed 3 heads and 5 tails, (D is the event of 3 heads and 5 tails), the posterior allows us to obtain the probabilities of $P(\theta = 0.1|D)$ or $P(\theta = 0.7|D)$, etc. The posterior givens us probabilities for all possible values of θ (the probability of heads).



Prior

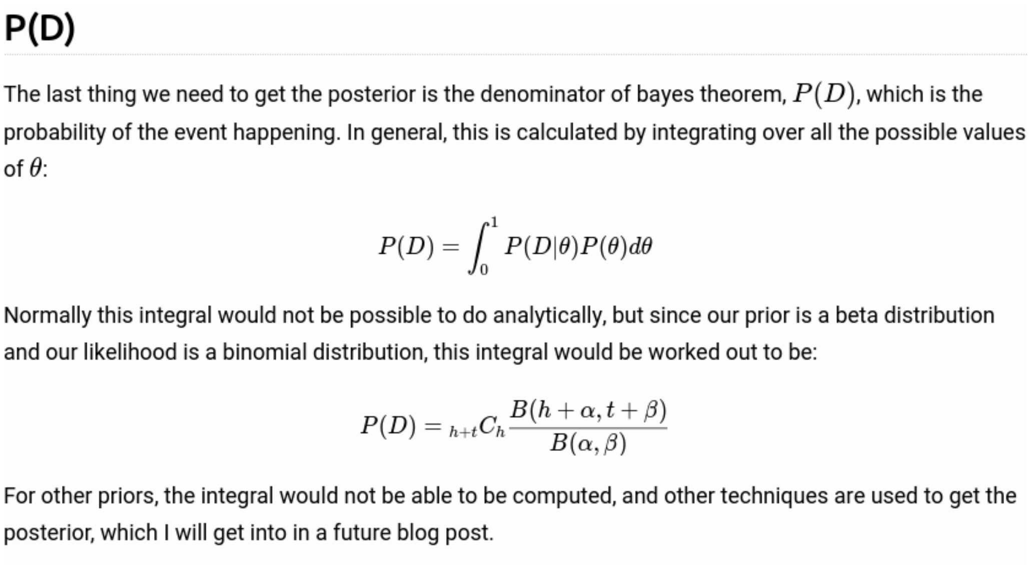
Next, lets look at the prior, $P(\theta)$, this is the probability of θ before any coin flips. In other words, this is the measure of the belief *before* we perform the experiment. For the coin flipping example, we normally come across coins that have $\theta = 0.5$, so our prior should center around 0.5. For now, lets pick a beta distribution with $\alpha = 2$ and $\beta = 2$ as our prior:

$$P(\theta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

where $B(\alpha,\beta)$ is the Beta Function. This prior is centered at 0.5 and is lower for all other values. Let's graph the beta prior distribution:

▸ Show Code

The maximum of our prior is centered at 0.5 and is lower for other values. This means that we normally see coins which are fair, but do not rule out that there is a chance that the coin could be unfair.



P(D)

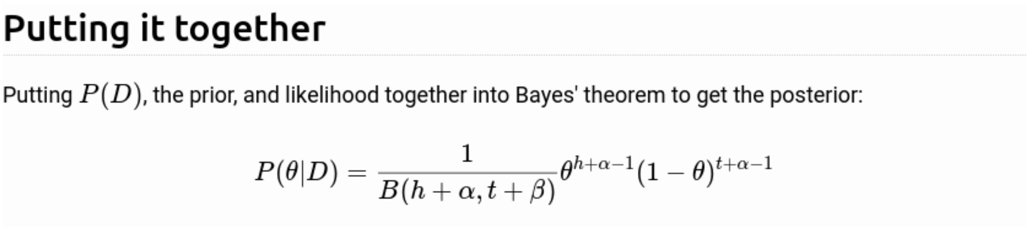
The last thing we need to get the posterior is the denominator of bayes theorem, $P(D)$, which is the probability of the event happening. In general, this is calculated by integrating over all the possible values of θ:

$$P(D) = \int_0^1 P(D|\theta)P(\theta)d\theta$$

Normally this integral would not be possible to do analytically, but since our prior is a beta distribution and our likelihood is a binomial distribution, this integral would be worked out to be:

$$P(D) = {}_{h+t}C_h \frac{B(h+\alpha,t+\beta)}{B(\alpha,\beta)}$$

For other priors, the integral would not be able to be computed, and other techniques are used to get the posterior, which I will get into in a future blog post.



Putting it together

Putting $P(D)$, the prior, and likelihood together into Bayes' theorem to get the posterior:

$$P(\theta|D) = \frac{1}{B(h+\alpha,t+\beta)}\theta^{h+\alpha-1}(1-\theta)^{t+\alpha-1}$$

4.6 Joint, Marginal, & Conditional Probability Distribution

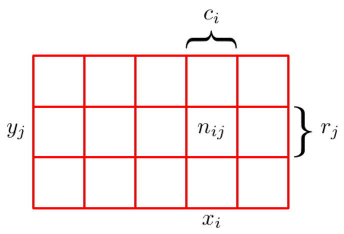


Figure 3

Joint probability distribution is the corresponding probability distribution on all possible pairs of outputs

$$P(X = x_i,\ Y = y_j) = \frac{n_{ij}}{N} \tag{1}$$

Marginal probability distribution is the probability distribution of the variables contained in the subset (integrating out a parameter).

$$p(X = x_i) = \frac{c_i}{N} = \frac{\sum_j n_{ij}}{N} \tag{2}$$

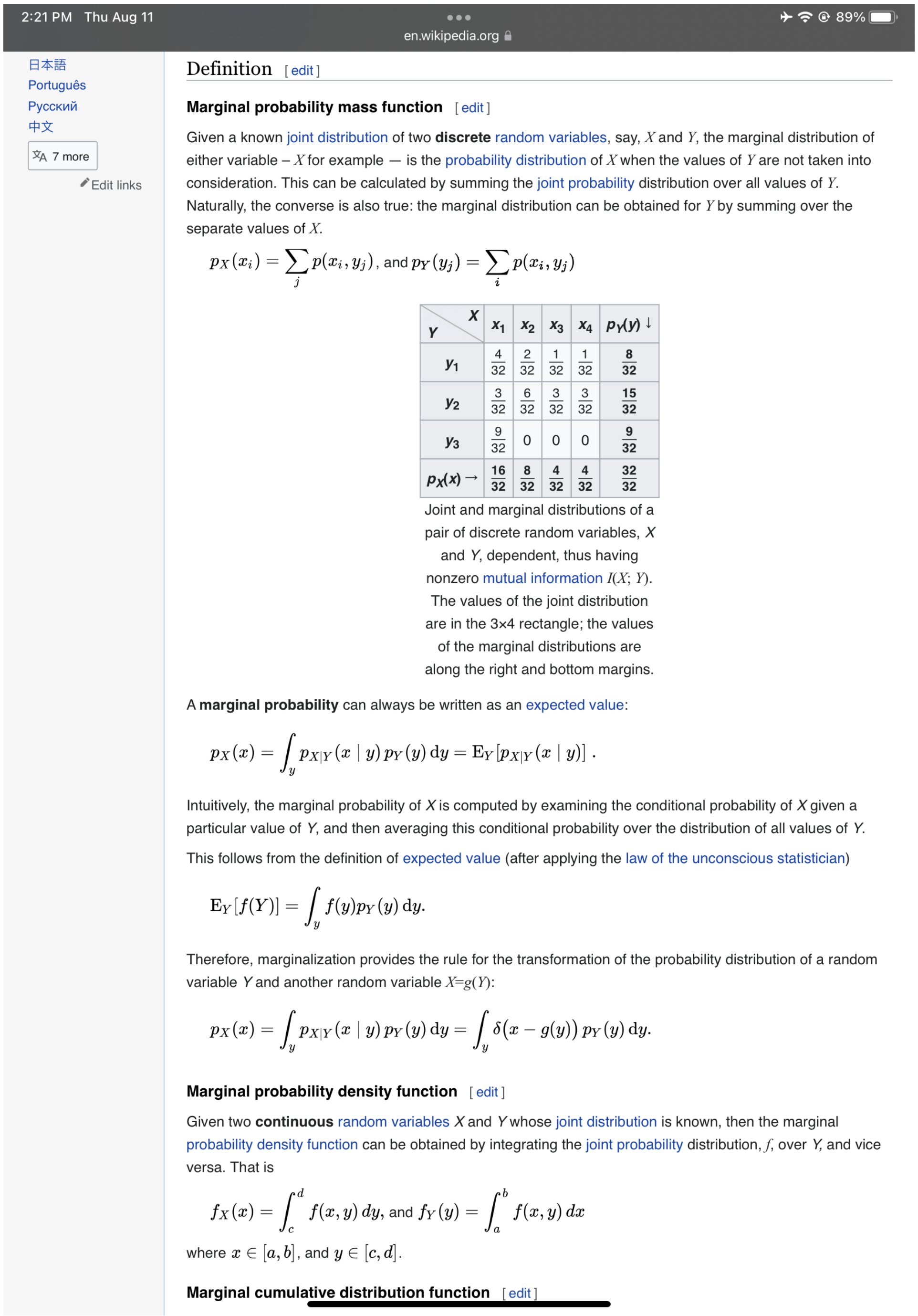
$$\begin{aligned} P(X = x_i) &= \sum_{j=1}^{L} p(X = x_i, Y = y_j) = \sum_{j=1}^{L} P(Y|X)P(X) \\ f_X(x) &= \int f_{X,Y}(x,y)dy = \int f(X|Y)P(Y)dy \\ f_Y(y) &= \int f_{X,Y}(x,y)dx = \int f(Y|X)P(X)dx \end{aligned} \tag{3}$$

Conditional Probability distribution is a probability of an event occuring, given that another event has already occurred.

$$\begin{aligned} P(Y = y_i|X = x_i) &= \frac{n_ij}{c_i} \\ P(Y|X) &= \frac{P(X,Y)}{P(X)} \end{aligned} \tag{4}$$

4.7 Marginal Probability

Marginal probability



Definition [edit]

Marginal probability mass function [edit]

Given a known joint distribution of two **discrete** random variables, say, X and Y, the marginal distribution of either variable – X for example — is the probability distribution of X when the values of Y are not taken into consideration. This can be calculated by summing the joint probability distribution over all values of Y. Naturally, the converse is also true: the marginal distribution can be obtained for Y by summing over the separate values of X.

$$p_X(x_i) = \sum_j p(x_i, y_j), \text{ and } p_Y(y_j) = \sum_i p(x_i, y_j)$$

Y \ X	x_1	x_2	x_3	x_4	$p_Y(y)$ ↓
y_1	$\frac{4}{32}$	$\frac{2}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{8}{32}$
y_2	$\frac{3}{32}$	$\frac{6}{32}$	$\frac{3}{32}$	$\frac{3}{32}$	$\frac{15}{32}$
y_3	$\frac{9}{32}$	0	0	0	$\frac{9}{32}$
$p_X(x)$ →	$\frac{16}{32}$	$\frac{8}{32}$	$\frac{4}{32}$	$\frac{4}{32}$	$\frac{32}{32}$

Joint and marginal distributions of a pair of discrete random variables, X and Y, dependent, thus having nonzero mutual information $I(X; Y)$. The values of the joint distribution are in the 3×4 rectangle; the values of the marginal distributions are along the right and bottom margins.

A **marginal probability** can always be written as an expected value:

$$p_X(x) = \int_y p_{X|Y}(x \mid y)\, p_Y(y)\, \mathrm{d}y = \mathrm{E}_Y[p_{X|Y}(x \mid y)]\,.$$

Intuitively, the marginal probability of X is computed by examining the conditional probability of X given a particular value of Y, and then averaging this conditional probability over the distribution of all values of Y.

This follows from the definition of expected value (after applying the law of the unconscious statistician)

$$\mathrm{E}_Y[f(Y)] = \int_y f(y) p_Y(y)\, \mathrm{d}y.$$

Therefore, marginalization provides the rule for the transformation of the probability distribution of a random variable Y and another random variable $X{=}g(Y)$:

$$p_X(x) = \int_y p_{X|Y}(x \mid y)\, p_Y(y)\, \mathrm{d}y = \int_y \delta\big(x - g(y)\big)\, p_Y(y)\, \mathrm{d}y.$$

Marginal probability density function [edit]

Given two **continuous** random variables X and Y whose joint distribution is known, then the marginal probability density function can be obtained by integrating the joint probability distribution, f, over Y, and vice versa. That is

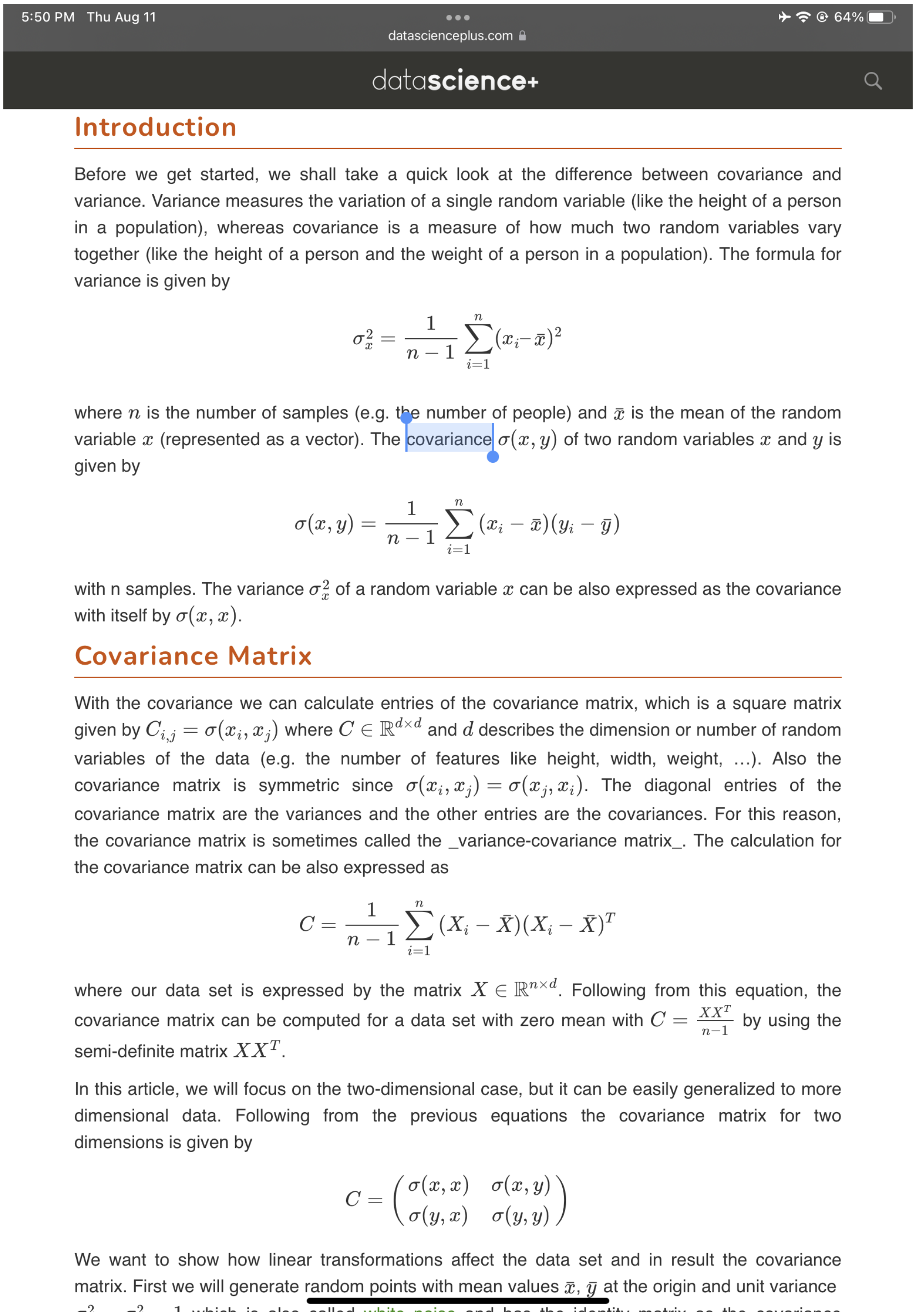
$$f_X(x) = \int_c^d f(x, y)\, dy, \text{ and } f_Y(y) = \int_a^b f(x, y)\, dx$$

where $x \in [a, b]$, and $y \in [c, d]$.

Marginal cumulative distribution function [edit]

4.8 Covariance Matrix

Covariance Matrix



Introduction

Before we get started, we shall take a quick look at the difference between covariance and variance. Variance measures the variation of a single random variable (like the height of a person in a population), whereas covariance is a measure of how much two random variables vary together (like the height of a person and the weight of a person in a population). The formula for variance is given by

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where n is the number of samples (e.g. the number of people) and $\bar{x}$ is the mean of the random variable x (represented as a vector). The covariance $\sigma(x, y)$ of two random variables x and y is given by

$$\sigma(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

with n samples. The variance σ_x^2 of a random variable x can be also expressed as the covariance with itself by $\sigma(x, x)$.

Covariance Matrix

With the covariance we can calculate entries of the covariance matrix, which is a square matrix given by $C_{i,j} = \sigma(x_i, x_j)$ where $C \in \mathbb{R}^{d \times d}$ and d describes the dimension or number of random variables of the data (e.g. the number of features like height, width, weight, …). Also the covariance matrix is symmetric since $\sigma(x_i, x_j) = \sigma(x_j, x_i)$. The diagonal entries of the covariance matrix are the variances and the other entries are the covariances. For this reason, the covariance matrix is sometimes called the _variance-covariance matrix_. The calculation for the covariance matrix can be also expressed as

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T$$

where our data set is expressed by the matrix $X \in \mathbb{R}^{n \times d}$. Following from this equation, the covariance matrix can be computed for a data set with zero mean with $C = \frac{XX^T}{n-1}$ by using the semi-definite matrix XX^T.

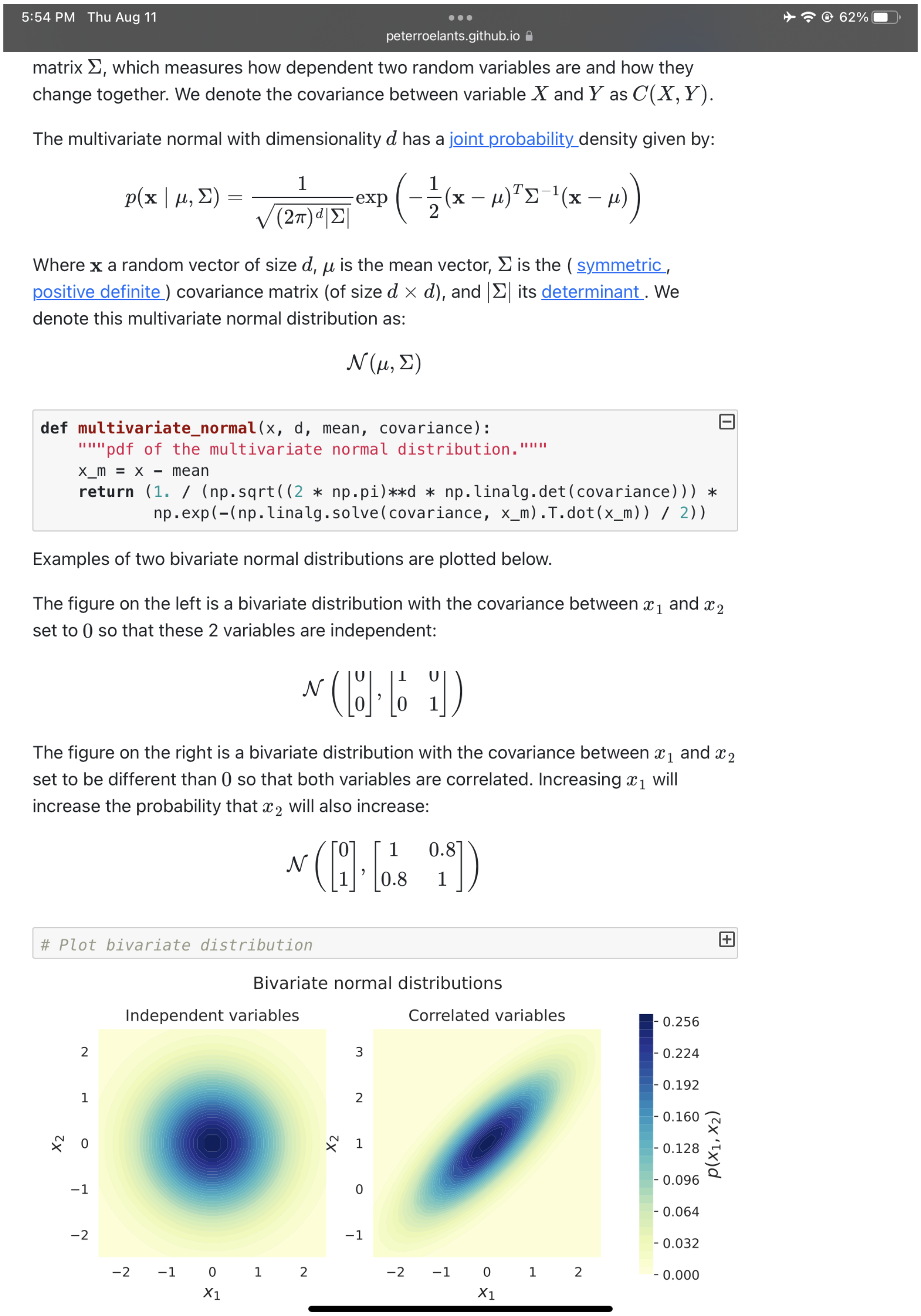
In this article, we will focus on the two-dimensional case, but it can be easily generalized to more dimensional data. Following from the previous equations the covariance matrix for two dimensions is given by

$$C = \begin{pmatrix} \sigma(x, x) & \sigma(x, y) \\ \sigma(y, x) & \sigma(y, y) \end{pmatrix}$$

We want to show how linear transformations affect the data set and in result the covariance matrix. First we will generate random points with mean values $\bar{x}$, $\bar{y}$ at the origin and unit variance $\sigma_x^2 = \sigma_y^2 = 1$ which is also called white noise and has the identity matrix as the covariance

4.9 Multivariate Gaussian

Multivariate Gaussian



matrix Σ, which measures how dependent two random variables are and how they change together. We denote the covariance between variable X and Y as $C(X, Y)$.

The multivariate normal with dimensionality d has a joint probability density given by:

$$p(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$

Where $\mathbf{x}$ a random vector of size d, μ is the mean vector, Σ is the (symmetric , positive definite) covariance matrix (of size $d \times d$), and $|\Sigma|$ its determinant . We denote this multivariate normal distribution as:

$$\mathcal{N}(\mu, \Sigma)$$

```
def multivariate_normal(x, d, mean, covariance):
    """pdf of the multivariate normal distribution."""
    x_m = x - mean
    return (1. / (np.sqrt((2 * np.pi)**d * np.linalg.det(covariance))) *
            np.exp(-(np.linalg.solve(covariance, x_m).T.dot(x_m)) / 2))
```

Examples of two bivariate normal distributions are plotted below.

The figure on the left is a bivariate distribution with the covariance between x_1 and x_2 set to 0 so that these 2 variables are independent:

$$\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

The figure on the right is a bivariate distribution with the covariance between x_1 and x_2 set to be different than 0 so that both variables are correlated. Increasing x_1 will increase the probability that x_2 will also increase:

$$\mathcal{N}\left(\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1 & 0.8\\0.8 & 1\end{bmatrix}\right)$$

```
# Plot bivariate distribution
```

Bivariate normal distributions

5 DL/ML Questions

DL/ML questions are drawn from: src1, src2

5.1 What Neural network?

- Neural networks is a network of artificial neurons designed to replicate the way humans learn.
- A series of algorithms are operated as the input data is fed into the network and gets passed down to next layer of neurons to extract underlying relationships in the fed data.

5.2 What is the difference AI, ML, and DL?

- AI represents **simulated** intelligence in machine that replicated ways humans think and learn.
- AI is a subset of Data Science
- ML is the practice of getting machines to **extract patterns** and make decisions by feeding data into it.
- ML is a subset of AI & Data Science
- DL is a process of using **artificial neural networks to extract patterns** in the fed data and make requested predictions.
- DL is a subset of ML, AI, & Data Science.

5.3 What are different types of AI?

- Reactive Machines AI
  - Simplest level of robot such that they cannot create memories or use information learnt to influence future decisions. They only react to presently existing situations
  - IBM's Deep Blue, a machine designed to play chess against a human is an example.

5.4 Different domains of AI

- Natural Language Processing: AI method that **analyze natural human language** to extract patterns and classify the meaning of the language.
  - Google Search Engine AI uses predictive analytic, natural language processing and machine learning to recommend relevant searches to you.
  - The recommendations are based on data that Google collects about you and make prediction on what you might be looking for.

5.5 Bayesian network vs Markov networks

- Bayesian entworks are directed and acyclic graphs
- Makov networks are undirected and may be cyclic graphs

5.6 What is transfer learning?

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

5.7 Supervised, Unsupervised, Reinforcement learning

- Supervised: Training done on a set of known data
- Unsupervised: Training done on data that is not labeled: learn from scratch
- Reinforcement: Training by trail and error: Reward desired behavior + Punish undesired behaviors: Data not labeled.

5.8 SGF vs Adam optimizer

Comparison

- SGD generalizes better: improved result
- Adam converges faster

Adam Opti

- Adam also keeps average of past gradients

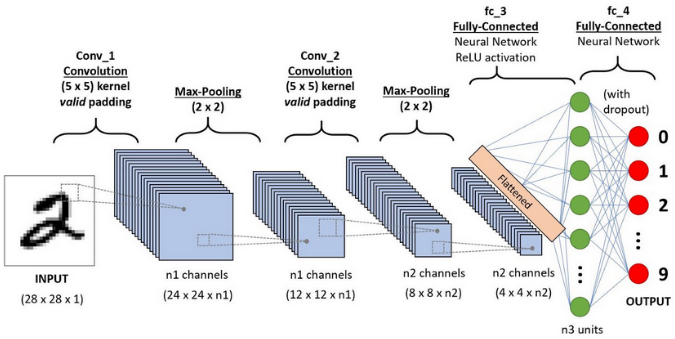
SGD

- Stochastic gradient descent: update weight for every mini-batch

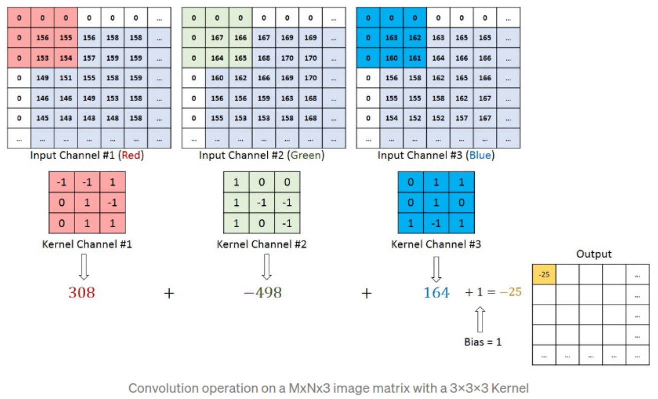
5.9 Convolutional Neural Network

A **Convolutional Neural Network (CNN)** is a Deep Learning algorithm which can take in an input image, assign importance to various **specific portions in the image** using filters (**pooling**).

- The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex
- Individual neurons respond to **stimuli only in a restricted region** of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



(a)



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

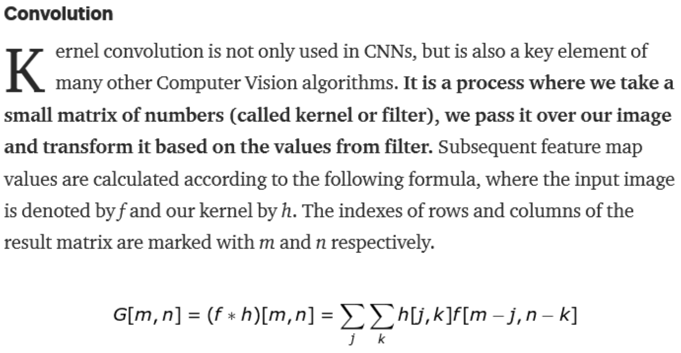
Kernel Channel #3

Output

-25				...
				...
				...
				...
...	...	...	...	...

Convolution operation on a MxNx3 image matrix with a 3×3×3 Kernel

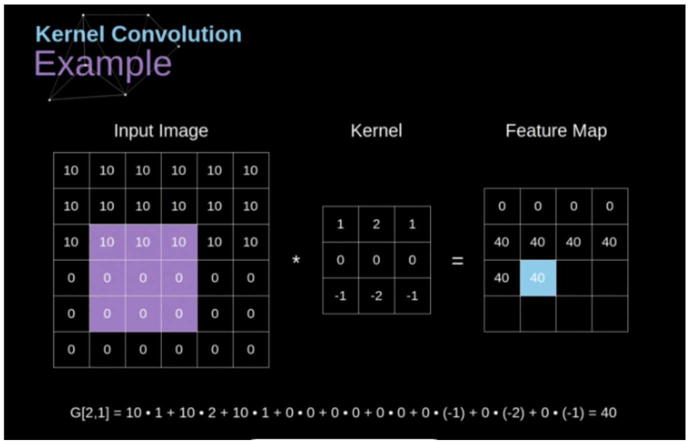
(b)



Convolution

Kernel convolution is not only used in CNNs, but is also a key element of many other Computer Vision algorithms. **It is a process where we take a small matrix of numbers (called kernel or filter), we pass it over our image and transform it based on the values from filter.** Subsequent feature map values are calculated according to the following formula, where the input image is denoted by *f* and our kernel by *h*. The indexes of rows and columns of the result matrix are marked with *m* and *n* respectively.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$



Input Image

10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Kernel

1	2	1
0	0	0
-1	-2	-1

Feature Map

0	0	0	0
40	40	40	40
40	40		

G[2,1] = 10 • 1 + 10 • 2 + 10 • 1 + 0 • 0 + 0 • 0 + 0 • 0 + 0 • (-1) + 0 • (-2) + 0 • (-1) = 40

(c)

Figure 4: (a) Overview of CNN. (b) Specific region of image: Pooling. (c) Filter map math

Pooling layers

- Sliding 2dim filter over each region of feature map
- Summarizing the features lying within the region covered by the filter
  - Average Pooling
  - Max pooling

Why Pooling?

- Reduce the dimensions of the feature maps → Reduce classifying parameters
- Since summarizes the features → model more robust to variations.

Padding

- When filters used → size gets reduced → dont wan't that
- Add pixel value of 0 in the cropped out portion → preserve the original image size

5.10 Confusion matrix

- Table (error matrix) that allows visualization of the classification algorithm performance
- Combinations of
  - Condition positive: number of real positive cases in the data
  - Condition negative
  - true positive: correctly classified positive
  - true negative: correctly classified negative
  - false positive
  - false negative

5.11 Bayesian Neural Network

Bayesian NN, Probabilistic NN, Probabilistic Bayesian NN

- Bayesian Neural Network: Model weights are probability distributions, instead of scalar weights
- Probabilistic Neural Network: Outputs are probability distributions, but weights are scalar
- Probabilistic Bayesian Neural Network: Both model weights and outputs are probability distributions

Bayesian Deep learning - Bayesian Inference

- Bayesian Inference: Estimating the posterior probability of a hypothesis using the prior information (prior prob. distri.) and new evidence (data) about the system.
- Contrasts frequentist inference which makes predictions using only data from the current experiment: only a sample from total population.

- ANALOGY: I have misplaced my phone somewhere in the home. I can use the phone locator on the base of the instrument to locate the phone and when I press the phone locator the phone starts beeping. Which area of my home should I search?
    - I can hear the phone beeping. I also have a mental model which helps me identify the area from which the sound is coming. Therefore, upon hearing the beep, I infer the area of my home I must search to locate the phone.
    - I can hear the phone beeping. Now, apart from a mental model which helps me identify the area from which the sound is coming from, I also know the locations where I have misplaced the phone in the past. So, I combine my inferences using the beeps and my prior information about the locations I have misplaced the phone in the past to identify an area I must search to locate the phone.

Bayesian NN advantages and disadvantages

- Advantages
    - Less overfitting
    - Ability to quantify uncertainty in the predictions
    - Most applicable to real-world data
- Disadvantages
    - Slower and require more data to converge since distributions instead of scalar weights

Naive Bayes Classifier

- Supervised Probabilistic Classification method using Bayesian theorem but with assumption that the features of data are independent of each other
- Not a Neural Network
- When coupled with kernel density estimation, they can achieve high accuracy

Why is Naive Bayes so fast?

- Because it only needs the prior probability that do not change and can be stored ahead of time.
- No iteration, epoch, optimization of a cost equation, error back-propagation.

Difference between Naive Bayesian Classifier and Bayesian Neural Network

- Naive Bayesian classifier is a tool for performing classification
- Bayesian Neural Network is a graphical model that represent the interactions between random variables using Bayesian Inference.

Bayesian Neural Network: When is it used?

- In cases where some information is known about the prior distribution of the random variables

How are Bayesian Neural Networks trained?

- For the most part, BNN are trained just like other neural nets.

- **Maximum a Posteriori (MAP) estimation**: Produces the probability distributions weights

Bayesian Convolutional Neural Network?

- Just CNN + using prob. distri. weight

Bayes Backpropagation: Bayes-backprob

- Regularizes the weights by minimizing a compression cost, known as the variational free energy o r the expected lower bound on the marginal likelihood.

5.12 Variational Inference: BNN

Goal: Approximate difficult-to-compute probability densities via optimization

- First, estimate a family of densities that might resemble the target density
- Second, Find the member of the family which is closest to the target density
  - that minimizes KL divergence

5.13 Markov Chain Monte Carlo

Monte Carlo

- Randomly sample a probability distribution and approximate a desired quantity
  - Under the assumption that we can efficiently draw samples from the target distribution
  - Then, we can estimate the sum or integral quantity as the mean or variance of the drawn sample

Markov Chain

- Generate a sequence of random variables where the current value is probabilistically dependent on the pior variable.
- Selecting the next variable is only dependent upon the last variable in the chain
- Ex: Consider a board game that involves rolling dice. You have a position on the board, but your next position on the board is only based on the current position and the random roll of the dice.

Markov Chain Monte Carlo

- Series of Random Sampling from previous probability distributions that are also sampled from a target distribution.
- Samples are drawn from the probability distribution by constructing a Markov Chain, where the next sample that is drawn from the probability distribution is dependent up the last same that was drawn.
- The idea is that the chain will settle on (find equilibrium) on the desired quantity we are inferring.

5.14 Regularization methods

L1 and L2

- Suppress weights on certain classifying parameters
- Useful when prior knowledge of parameter is available

Drop-out

- Randomly select and removes some nodes in the hidden layer at every iteration of training.
- Useful when the network is large and the user want to introduce some randomness.

Early-stopping

- Monitoring of the error while training on training data and testing on the testing data simultaneously.
- If the error from the testing data reaches certain value, stop training.

Batch Normalization

- Normalization: Re-scaling of the data (or weights) to achieve the numerical data to a common scale: so that total would always sum to 1.
  - Re-centering
  - Re-scaling
- In detail:
  - Determines the mean and variance of the activation values (input to next layer) across the batch
  - Then normalize the activation values (input to next layer)
  - Linear transformation: Re-scaling (adjust standard dev.) and Re-centering (adjust bias.)
- Benefits
  - Speed up training
  - Handles internal covariate shift
    - Changes in distribution of the inputs of each layer affect the learning rate.
  - Smoothens the loss function

Weight Decay

- To help avoiding overfitting → penalize complexity!
- Add the parameters (weights) to the loss funtion
  - Since weights could be negative or positive → Add the squares of all the weights to the loss function
  - To suppress the loss not to explode → multiply by **weight decay**, wd.

$$Loss_{new} = Loss_{old} + wd * \sum(weight^2)$$

5.15 Case Studies

How would you build a trigger word detection algorithm to spot the word "activate" in a 10 second long audio clip?

1.

An e-commerce company is trying to minimize the time it takes customers to purchase their selected items. As a machine learning engineer, what can you do to help them?

1.

You are given a data set of credit card purchases information. Each record is labeled as fraudulent or safe. You are asked to build a fraud detection algorithm. How would you proceed?

1.

You are provided with data from a music streaming platform. Each of the 100,000 records indicates the songs a user has listened to in the past month. How would you build a music recommendation system?

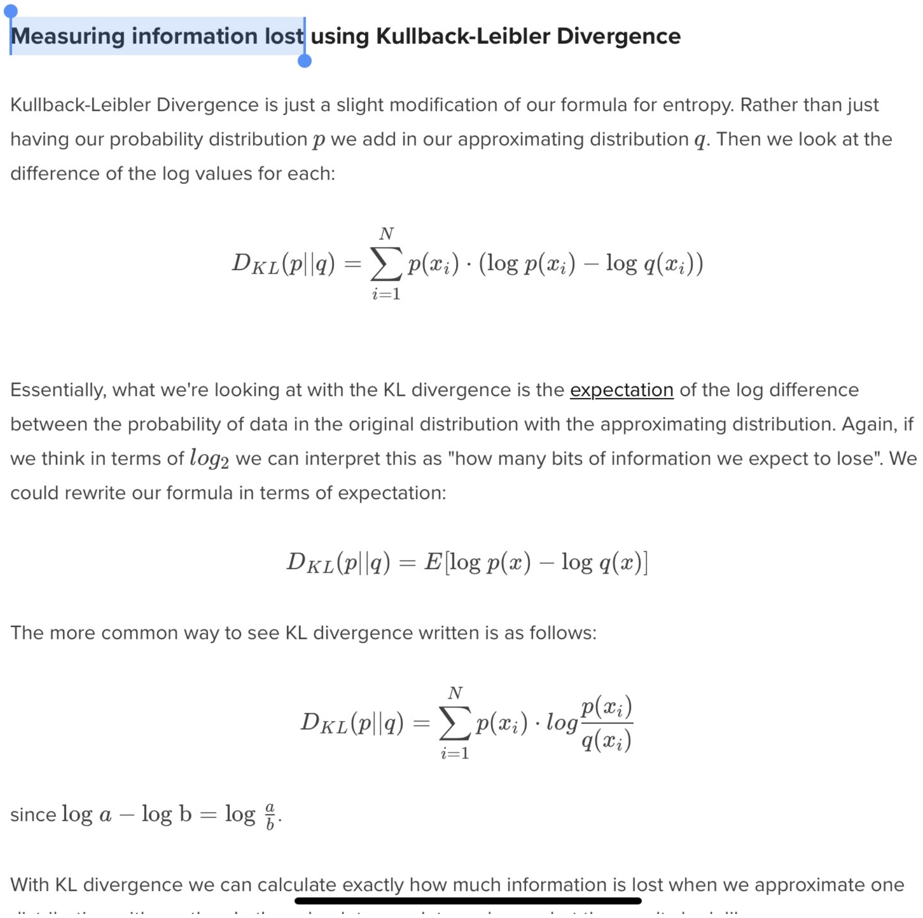
1.

ML-Powered Search Ranking of Airbnb Experiences

1. Data collection
   - Rate the "Experiences" accordingly → Ratings are Labels
   - Rating based on Features: 1 to 5? or 10?
   - Features: Experience duration, Price, category, reviews, number of bookings, occupancies, maximum number of seats, and click-through rates.
   - Clicks of users who ended up booking *vs* users who didn't book
2. Feature selection
   - Plot histograms of all features → Remove outliers
   - Plot 2d histogram: every possible pair of features → Deduce relations
   - Examine + Remove outliers
   - Correlation matrix
   - Systematic study on how features are computed
   - Estimate errors for each features
3. Training
   - Simple NN with output layer with same # of nodes as the rating system: 1 to 10 or 1 to 5.
   - Regularization depending on the result → avoid overfitting
   - Fine tune: Optimizer, loss function, layers, nodes

5.16 KL-Divergence

Kullback-Leibler Divergence



Measuring information lost using Kullback-Leibler Divergence

Kullback-Leibler Divergence is just a slight modification of our formula for entropy. Rather than just having our probability distribution p we add in our approximating distribution q. Then we look at the difference of the log values for each:

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

Essentially, what we're looking at with the KL divergence is the expectation of the log difference between the probability of data in the original distribution with the approximating distribution. Again, if we think in terms of log_2 we can interpret this as "how many bits of information we expect to lose". We could rewrite our formula in terms of expectation:

$$D_{KL}(p||q) = E[\log p(x) - \log q(x)]$$

The more common way to see KL divergence written is as follows:

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

since $\log a - \log \mathrm{b} = \log \frac{a}{b}$.

With KL divergence we can calculate exactly how much information is lost when we approximate one

6 General Government Questions - Gov. Contract

6.1 Opener

- Hello, Thank you for this opportunity to interview with APL.

6.2 Describe yourself - work-wise

- I am a recent phd graduate majoring in experimental nuclear physics.
- Most of PhD work was data analysis of physics experiments
- Revisited and analyzed accumulated experimental data → new conclusions using new data analysis tech.
- Came across the potential of machine learning applications in physics
    - Advancement of ML is advancement of nuclear physics experiments
    - Save experimental data
    - From theory to data → data to theory
- So, I came across this job posting from STR and I really liked how the company supported ML research and applies to it problems pertaining to NATIONAL SECURITY.
- Jefferson Lab → Hardware to data management
    – shifts
    – on-call duty 24hr experiments
    – calibrate the detector
    – validate physics reconstruction algorithms

6.3 Briefly explain your previous work / PhD Dissertation

Physics

- Goal → validate quark models → nuclear force
- In detail, excited states of nucleons *resonances* → just below breaking apart energy 2GeV.
- highly energized photon scatter stationary proton targets → study excisted states of protons
- Various quark models predict certain excited states of nucleons, → far less are observed in experiment
- Specifically, spin states → by polarizing the photon and proton into certain directions.
- Quark models predict certain spin states to occur in a specific way → try to confirm experimentally.
- In conclusion, we don't have a working nuclear force theory in low energy region.

Deep learning

- While revisiting old experiments → experimental error was spotted
  - Previously thrown out → ML to save data
  - vacuum leakage in the low temperature controlled target (carbon) → ice formed
  - So events both stemmed from carbon and ice events
  - goal was to classify events of photons hitting ice from photons hitting the carbon target
  - Simple fully connected neural nets
  - Concerned my colleagues → inability to quantify uncertainties when imperfect train data
  - Started study probabilistic neural networks
  - Give probability scores to each groups of training data → fitting, level of contamination
  - Teach neural net to learn less on events with high uncertainty.

6.4 Research interests

- Estimation of ML prediction's uncertainties → most important part of the ML research
- Real world data → training data always imperfect
- Estimate degree of uncertainty in training data
- Need probabilistic model → Bayesian to better incorporate uncertainties.

6.5 Why Bayesian Neural Network?

Where I am:

- In pytorch: only μ and σ
- In my case, we guess certain % of data has correct labels → no μ and σ
- For certain portion of the distribution, we say certain % is ICE and other % is carbon.

What I can do:

- Path integral formulation QFT to BNN: Core idea resembles
- Imperfect data ex: Elephant vs Rhinoceros: Fit length of tusk and give rating (prior information)

Why interested?

- Tried NN in nuclear physics experiments
- Regardless of how sophisticated (CNN, complex regularization, optimization), PREDICTION FAILS when using training data with high degree of uncertainty and not correctly estimated uncertainties in training data.
- Simulated data as training data → Need to quantify the likeness to real experiment.
- Need efficient way to rate the training data → Models selectively train itself based on the rating of training data.

6.6 Why are you applying for this position?

- Personal level → respected those who serve the country.
  - federal, state grands and grants from DOE for research → undergraduate, graduate
  - Enabled me to purse my dream of studying physics
  - I want to pay back
  - *Working in defense sector is a form of service*
- Certain professional freedom: research, conferences
- Opportunity to work with other ML researchers and apply to real-world data.
  - Avinash Pujala

6.7 What can you bring to STR for ML research?

- Data analysis skills: Fitting, Mathematical background, Ability to see data in many diff. angles
- Any new tech, I am a good self-learner
- QFT Path integral formulation to BNN formulation

6.8 Why leave physics and join ML?

- advancement in ML is also advancement in physics.
- In low energy QCD, theory failed: last 40 years, no advancements
- NOT theory → data: Formulating a potential to match data → OLD WAY
- NEED data → theory: ML on DATA to derive the potential

6.9 Weakness and strengths

Weakness

- Too focused on a problem → lose track of time and hard to prioritize other important tasks.
- Update colleagues and gain perspective

Strengths

- Curiosity
- Physical healthy to stay CURIOUS - REHAB - Accident in 2010
- Make people interested in projects
- self-learn & self-propagate when problems
- Well organized in logic → mental picture of the problem

6.10 How would you lead a team?

- Goal: Make someone interested/passionate about a project.
- Input time - Weekly meetings
- Understand the projects of every individuals - Don't have to fix
- Fun simplified toy example
- Attitude that I may be wrong during discussions

6.11 How would you start analyzing a dataset?

Key points

- No need for a sophisticated ML or fitting techniques → Simpler is better
- Understanding of data is much more important → regularization and feature selection.

Pre-ML

- 1 axis - Histograms
  - Refine binning
  - For ex: Time data: first bin by months, then by weeks, then by days.
- Remove extreme outliers
- Introduce more axis - 2D plots
  - Partition in bins: Energy ranges, angle ranges, etc → Show BINNING plot
    - Refine binning
    - For ex: Spatial data: First bin by km, then by m, then by cm.
  - Plot 2D histogram or scatter: every possible pair of features → Deduce relations
  - Table of Histograms in each partitions
  - Remove outliers again
- Correlation between parameters → Avoid -1 or 1 *or* too many uncorrelated params
- Systematic study how each parameters are computed during data taking → Subsystems
- Select classifying parameters based on systematic studies
- Estimate error/contamination level of data coming from how each parameter is computed.

ML

- Selection of training data ("LABELING") → Depends on data → Boostrap or not
- Size of training data per classes important → Match data sizes of each classes in total data
- Simple models first
- Apply regularization methods depending on systematic studies
- Fine tune: optimizer, loss function, layers, nodes

6.12 Situation when you failed

- Teaching → anxious in front of crowd → gave wrong information
- Email with corrections with slides for what was taught wrong
- Rehearse as many times as possible

6.13 Situation when you faced a problem within a group

- Physicist hating ML
    - Simply state current status of ML and potential of ML
- Adviser plagiarism on overleaf - from his paper, TRACKING OFF
    - Have to be delicate, but confront to never repeat

6.14 Best collaboration experience

- Jefferson Lab 30 universities internationally, 150 physicists → Need good collaboration
- CTOF path length miscalculation when reconstruction in Scintillation bars
    - Sources of errors tracked by different experts
    - reconstruction software people, detector geometry people, calibrators, shift takers during experiment
    - Fixed one thing while holding everything unchanged → checking everything
    - problem → a bug in the detector geometry code → cylinder but sphere
- Currently my thesis on overleaf for publication → 6 professors and me as first author. 30-40 co-authors.

6.15 Opinion on Diversity

- I sincerely feel the need for the diversity in all parts of our society.
- Also, focus into diversifying people's cultures, ideas, and , philosophies.
- physical appearances shouldn't impact on the opportunities that one can receive.

6.16 Questions

I have a few administrative questions and some more questions related to culture at STR.

- How much time is spent in: Theory (developing algorithms) VS data handling and application of ML
- If there is a theory (specific algorithm) I want to focus researching on, Could I submit a proposal to ask for time and resources?
- What are some of the common challenges the ML team is facing?

- Typical day in ML researcher at STR?
- What was the biggest reason you came to STR?
- Is the work open public? If research done publishable to public journals?
- What is the culture like at STR?
- How many people are there? What are their academic background like?
- What are top qualities that you look for in a machine learning researcher?
- How do you envision STR to grow within next 5-10 years?
- Do you encourage researcher to attend ML conferences?
- What are the reasons for expansion in Arlington office?
- If I am to lead a research team, how many people will be allocated?
- Thank you, I had more questions, but you answered all of them.

7 NASA: Heliospheric Physics

7.1 Why interested in Heliospheric physics?

- B/c Diagram of the heliosphere looks alot like atoms, where the sun is the nucleon
- Willing to learn any necessary astrophysics

7.2 NASA Questions/Comments

- Many detectors, but Will I have a chance to study data from WIND spacecraft?
- Which of the WIND instruments and What kind of data will I be looking at?
  - Magnetic Field?
  - Ion Data?
  - Electron Data?
  - High energy particle data?
  - Radio and Plasma wave data?
- Interested in WIND EPACT High Energy Particle Data
  - Surprised by the large collection power → 3He, 4He, even to Fe.
  - Stupid question: How do you trace back the origin of these particles? When we don't know the source of the acceleration of these particles?
  - If the measured kinetic energies for particles from solar wind and Interplanetary Shock happen to be in same range, How do you classify?
- Will I be able to try new ways of classifying particles? and their sources?
  - For example, machine learning.
- Will I also be conducting simulations? or Are there simulation experts?
- Typically, How big is the data size for calibration work? C++ or python

7.3 CTOF Overview

Starting from Bethe-Block eq:

$$-\frac{dE}{dx} \propto \frac{AZ^2}{E} \tag{5}$$

- Measure the energy loss per distance
  - inversely proportional to total energy E
  - proportional to charge Z and mass A

CTOF Overview

- Measure the position and time at which the particle arrives at the scintillar bars
- Array of scintillation bars → forms a barrel

  - Scintillator material: plastic, liquid, crystal
  - Interaction in scintillator bar → electrons excited & de-exited → emit light signal

- Each bar had PMT (photomultiplier) at ends away from B-field
  - long light guides to guide signals to PMTs.
  - light-sensitive electron-amplification vacuum tube
  - detect tiny amounts of visible light

CTOF detector Calibration Steps

- Gain Calibrtion
- PMT High Voltage Calibration
- Timing Calibration
  - Upstream-Downstream PMT Timing Alignment
  - Attenuation Length in each counter
    - Distance into a material when the prob. drops to $1/e$ that a particle has not been absorbed.
  - Effective velocity
    - AVG speed that scintillation light propagates along the scintillation bars
  - RF Offset Calibration
  - Paddle to Paddle Calibration
  - TDC Calibration
    - time walk - time it takes for a signal to reach the threshold amplitude
- Path length correction
  - Since curved scintillation bars, sometimes particles interact with the bar twice
  - need to Select the first interaction

CTOF Reconstruction algorithms

- Hit time reconstruction
- Energy reconstruction
- Position coordinates
- Hit Clustering and Matching

7.4 WIND Spacecraft

Solid-state detector telescopes

- dE/dX *vs* total energy technique

8 Nuclear Physics

8.1 Why do heavier/larger nucleus (more N and P) have more neutrons?

[Knight pg1254-1256]

- Since the range of electrostatic force is greater than the range of nuclear force
- Mode protons in heavier/larger nucleus → more electrostatic force when only neighboring nucleons provide nuclear force
- Therefore, need more neutrons to compensate for the increase in EM force.

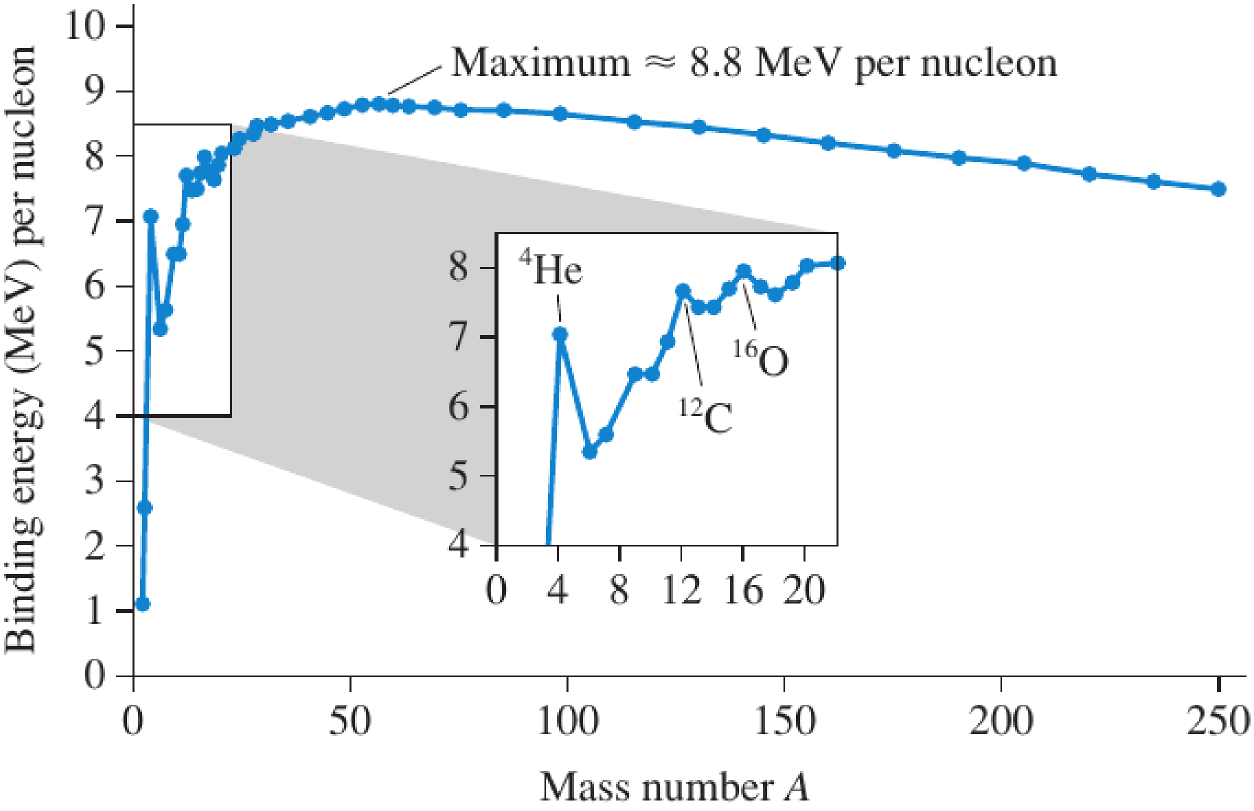


Figure 5

8.2 Slow and fast neutrons

SLOW

After a series of collisions with different nuclei, the energy of neutrons produced by fission reactions drops to the order of a few electronvolts or a few fractions of an electronvolt. Neutrons with energies in this range are collectively referred to as 'slow', and neutrons whose energies match those of the surrounding atoms are known as 'thermal'.

It is these slow neutrons that allow for nuclear reactors to run with fuel based on natural uranium or uranium lightly-enriched in fissile isotope 235. Without them, the most common pressurised (PWR) and boiling water (BWR) reactors would not operate. As a result, the neutrons emitted by nuclear fission have to be slowed down by collisions within a medium called a moderator. Reactors operating with natural uranium fuel, which contains only 0,7% of fissile uranium 235, require efficient moderators which absorb very few of the neutrons : such moderators are heavy water and pure graphite.

FAST

Before they are slowed down by a large number of nuclear collisions, neutrons produced by fission reactions are known as 'fast'. They usually have energies between 0.1 and 2 or 3 MeV.

The fact that they possess a substantial amount of kinetic energy allows fast neutrons to fission more easily nuclei once they get captured. They can therefore split not only nuclei reputed fissile by slow neutrons, but also minor actinides, the heavy nuclei which build up inside nuclear fuel as radioactive waste. Fast neutrons are needed to eliminate these waste products.

The use of fast neutrons in so-called ‘fast reactors’ allows for the production of more fissile nuclei than are destroyed, as the absorption of at least one neutron per fission by an uranium 238 nucleus transforms this uranium 238 into a fissile plutonium 239 nucleus. This process is known as breeding, leading to an almost inexhaustible supply of nuclear fuel.

One drawback of fast neutrons in reactors is that the probabilities of their capture by nuclei are comparatively small. Travelling in matter, neutrons see nuclei as targets. The apparent cross-section of these targets is much more smaller for fast neutrons than it is for slower neutrons. As a result, an intense neutron flux and a fuel rich in fissile elements are both needed to compensate for this lower probability.