

RECITATION 1

BACKGROUND

10-301/10-601: INTRODUCTION TO MACHINE LEARNING

09/02/2022

1 NumPy, Workflow, and Logging

[NumPy Notebook](#)

[Workflow Presentation](#)

[Logging Notebook](#)

2 Linear Algebra and Geometry

1. **Inner Product:** $\mathbf{u} = [6 \ 1 \ 2]^T$, $\mathbf{v} = [3 \ -10 \ -2]^T$, what is the inner product of \mathbf{u} and \mathbf{v} ? What is the geometric interpretation?

The inner product (aka dot product) of the two vectors $\mathbf{u} \cdot \mathbf{v} = 4$. Geometrically, this value is proportional to the projection of \mathbf{u} on \mathbf{v} .

2. **Cauchy-Schwarz inequality** (Optional): Given $\mathbf{u} = [3 \ 1 \ 2]^T$, $\mathbf{v} = [3 \ -1 \ 4]^T$, what is $\|\mathbf{u}\|_2$ and $\|\mathbf{v}\|_2$? What is $\mathbf{u} \cdot \mathbf{v}$? How do $\mathbf{u} \cdot \mathbf{v}$ and $\|\mathbf{u}\|_2\|\mathbf{v}\|_2$ compare? Is this always true?

$$\|\mathbf{u}\|_2 = \sqrt{3^2 + 1^2 + 2^2} = 3.74 \text{ and } \|\mathbf{v}\|_2 = \sqrt{3^2 + (-1)^2 + 4^2} = 5.10$$

$$\mathbf{u} \cdot \mathbf{v} = 16. \text{ Since } \|\mathbf{u}\|_2\|\mathbf{v}\|_2 = 19.074, \|\mathbf{u}\|_2\|\mathbf{v}\|_2 > \mathbf{u} \cdot \mathbf{v}.$$

In the general case, the Cauchy-Schwarz inequality states that $\forall \mathbf{u}, \mathbf{v} : (\mathbf{u} \cdot \mathbf{u})(\mathbf{v} \cdot \mathbf{v}) \geq (\mathbf{u} \cdot \mathbf{v})^2$ where \cdot denotes a valid inner product operation.

3. **Matrix algebra.** Generally, if $\mathbf{A} \in \mathbb{R}^{M \times N}$ and $\mathbf{B} \in \mathbb{R}^{N \times P}$, then $\mathbf{AB} \in \mathbb{R}^{M \times P}$ and $(\mathbf{AB})_{ij} = \sum_k A_{ik}B_{kj}$.

$$\text{Given } \mathbf{A} = \begin{bmatrix} 1 & 2 & 5 \\ 0 & 2 & 2 \\ 0 & 0 & 4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 4 & -3 & 2 \\ 1 & 1 & -1 \\ 3 & -2 & 2 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}.$$

- What is \mathbf{AB} ? Does $\mathbf{BA} = \mathbf{AB}$? What is \mathbf{Bu} ?
- What is rank of \mathbf{A} ?
- What is \mathbf{A}^T ?
- Calculate \mathbf{uv}^T .
- What are the eigenvalues of \mathbf{A} ?

- $\mathbf{AB} = \begin{bmatrix} 21 & -11 & 10 \\ 8 & -2 & 2 \\ 12 & -8 & 8 \end{bmatrix}$, $\mathbf{AB} \neq \mathbf{BA}$, $\mathbf{Bu} = \begin{bmatrix} 8 \\ -2 \\ 9 \end{bmatrix}$

- Rank of $\mathbf{A} = 3$

- $\mathbf{A}^T = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 5 & 2 & 4 \end{bmatrix}$

- $\mathbf{uv}^T = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 4 & 2 \\ 15 & 10 & 5 \end{bmatrix}$

- The eigenvalues of \mathbf{A} are 1, 2 and 4. In general, we find the eigenvalues for square matrices by finding the roots of the matrix's characteristic polynomial.

4. **Geometry:** Given a line $2x + y = 2$ in the two-dimensional plane,

- If a given point (α, β) satisfies $2\alpha + \beta > 2$, where does it lie relative to the line?
- What is the relationship of vector $\mathbf{v} = [2, 1]^T$ to this line?
- What is the distance from origin to this line?

- Above the line.

- This vector is orthogonal to the line.

- The distance is $\frac{2}{\sqrt{5}}$. Generally the distance from a point (α, β) to a line $Ax + By + C = 0$ is given by $\frac{|A\alpha + B\beta + C|}{\sqrt{A^2 + B^2}}$.

3 CS Fundamentals

1. For each (f, g) functions below, is $f(n) \in \mathcal{O}(g(n))$ or $g(n) \in \mathcal{O}(f(n))$ or both?

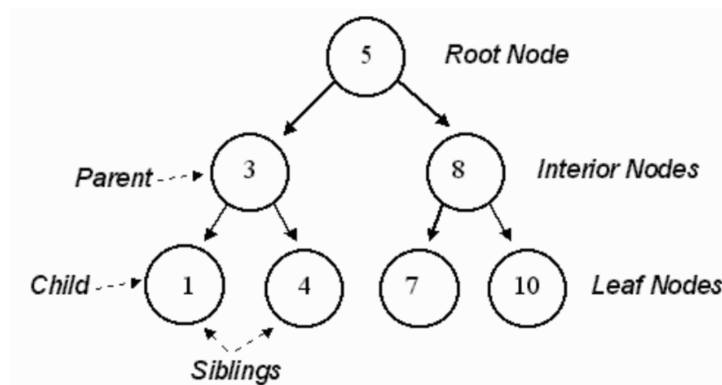
- $f(n) = \log_2(n)$, $g(n) = \log_3(n)$
- $f(n) = 2^n$, $g(n) = 3^n$
- $f(n) = \frac{n}{50}$, $g(n) = \log_{10}(n)$
- $f(n) = n^2$, $g(n) = 2^n$

If $f(n) \in \mathcal{O}(g(n))$, then:

$$\exists c, n_0 : \forall n \geq n_0, f(n) \leq cg(n)$$

- both
- $f(n) \in \mathcal{O}(g(n))$
- $g(n) \in \mathcal{O}(f(n))$
- $f(n) \in \mathcal{O}(g(n))$

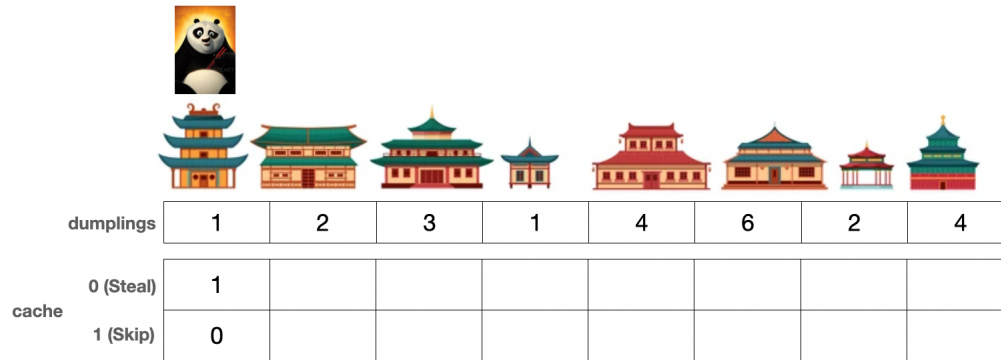
2. Find the DFS Pre-Order, In-Order, Post-Order and BFS traversal of the following binary tree. What are the time complexities of the traversals?











DFS (pre-order): 5, 3, 1, 4, 8, 7, 10
 DFS (in-order): 1, 3, 4, 5, 7, 8, 10
 DFS (post-order): 1, 4, 3, 7, 10, 8, 5
 BFS: 5, 3, 8, 1, 4, 7, 10

Time complexities are all $\mathcal{O}(n)$ where n is the number of nodes in the tree.

3. You are Po, the Kung Fu Panda, whose favourite food is Steamed Pork Dumplings. So, he plans to steal them from village houses arranged in a straight line.



									
dumplings		1	2	3	1	4	6	2	4
cache	0 (Steal)	1							
	1 (Skip)	0							

There is also a special security system in place that alerts Master Shifu if there has been a theft in 2 consecutive houses.

You need to help Po by employing the Bottom-Up Dynamic Programming approach to steal the maximum number of dumplings from the houses without alerting Master Shifu.

For any given house with index i ($i \geq 0$), you just need to write the logical statement to calculate the value after stealing from this house and after skipping this house.

```
def stealDumplings(dumplings: List[int]):
    n = len(nums)

    cache = {}

    cache[(0, 0)] = [a] #value of cache stealing the first dumpling
    cache[(1,0)] = [b] #value of cache skipping the second dumpling

    for i in range(1,n):
        #Stealing from this house
        cache[(0, i)] = [c]

        #Skipping this house
        cache[(1, i)] = [d]

    return [e] # return final answer
```

The number of dumplings per house is stored in a list called *dumplings* that you can access. Here, you need to fill in the Dictionary data structure. The key for each element is a tuple of 2 integers (Steal/Skip, House index) and the value will be the maximum number of dumplings are stolen so far either stealing or skipping this particular house index.

Hint - You can focus on how you can utilize the 2 values of the previous index in order to decide what values to fill for the 1st (steal) position at the current index and what to fill for the 2nd (skip) position.

```

def stealDumplings(dumplings: List[int]):
    n = len(nums)

    cache = {}

    cache[(0, 0)] = nums[0]
    cache[(1,0)] = 0

    for i in range(1,n):
        #Stealing from this house
        cache[(0, i)] = cache[(1, i-1)] + nums[i]

        #Skipping this house
        cache[(1, i)] = max(cache[(0, i-1)], cache[(1, i-1)])

    return max(cache[(0, n-1)], cache[(1, n-1)])

```

4. Complete the following python code for a Binary Search Tree (BST). You only need to complete the class function `insert` that will take a an integer `value` and insert the value at the right position in the tree rooted at the calling node.

Note: In a BST, each node's value is strictly greater than the value of the node to its left (its left child) and less than or equal to the value of node to its right (its right child).

```

class BST:
    def __init__(self, val):
        self.val = val
        self.leftNode = None
        self.rightNode = None

    def insert(self, value):
        if value < self.val: # going left
            if self.leftNode is None: # left node is undefined
                self.leftNode = [b]
            else:
                self.leftNode = [c] # recursive call

        else: # going right
            if self.rightNode is None: # right node is undefined
                self.rightNode = [d]
            else:
                self.rightNode = [e] # recursive call

    return self

```

Following the note above, we know that if the value is strictly less than the value of the node that we are inserting at, the value must reside to the left of that node, and

otherwise to the right. Additionally we can use the `__init__` function of the `BST` class to create a new node if we find that the left or the right child of the node is `None` (not defined yet).

```
class BST:
    def __init__(self, val):
        self.val = val
        self.leftNode = None
        self.rightNode = None

    def insert(self, value):
        if value < self.val:
            if self.leftNode is None:
                self.leftNode = BST(value)
            else:
                self.leftNode = self.leftNode.insert(value)

        else:
            if self.rightNode is None:
                self.rightNode = BST(value)
            else:
                self.rightNode = self.rightNode.insert(value)

        return self
```

5. You are given a sorted list of integers *e.g.*, [1, 2, 5, 7, 10, 13, 14, 15, 22]. Complete the recursive python function below that will take this list, create a Binary Search Tree (BST) with the minimum height possible, and return the root node of that tree.

```
class BST:
    def __init__(self, val):
        self.val = val
        self.leftNode = None
        self.rightNode = None

def minHeightBST(nums: List[int]):
    if len(nums)==0: # If nums is empty
        return [a]
    if len(nums) == 1: # If there's just one number
        return [b]

    mid = [c] # Find the middle element
    root = [d] # Make the root node

    leftNums = [e] # splice the left half of nums
    rightNums = [f] # splice the right half of nums
```

```
root.leftNode = [g] # Make a recursive call to assign the left node
root.rightNode = [h] # Make a recursive call to assign the right node

return root
```

The idea is to basically realize that we get the minimum height when the left and the right subtrees of any node are balanced, i.e. have about the same number of nodes (not differing by more than one). In keeping with the BST property, the easiest way to do this is to pick the middle element in the list of sorted numbers, create a root node, and then recursively call the same function on the left and the right subarrays to create minimum height subtrees at the leftNode and rightNode of the root.

```
class BST:
    def __init__(self, val):
        self.val = val
        self.leftNode = None
        self.rightNode = None

def minHeightBST(nums: List[int]):
    if len(nums)==0: # If nums is empty
        return None
    if len(nums) == 1: # If there's just one number
        return BST(nums[0])

    mid = len(nums)//2 # Find the middle element
    root = BST(nums[mid]) # Make the root node

    leftNums = nums[:mid] # splice the left half of nums
    rightNums = nums[mid+1:] # splice the right half of nums

    root.leftNode = minHeightBST(leftNums) # Make a recursive call to
        assign the left node
    root.rightNode = minHeightBST(rightNums) # Make a recursive call to
        assign the right node

    return root
```

4 Calculus

1. If $f(x) = x^3 e^x$, find $f'(x)$.

$$f'(x) = 3x^2 e^x + x^3 e^x \text{ by product rule}$$

2. If $f(x) = e^x$, $g(x) = 4x^2 + 2$, find $h'(x)$, where $h(x) = f(g(x))$.

$$h'(x) = 8x e^{4x^2+2} \text{ by chain rule}$$

3. If $f(x, y) = y \log(1 - x) + (1 - y) \log(x)$, $x \in (0, 1)$, evaluate $\frac{\partial f(x, y)}{\partial x}$ at the point $(\frac{1}{2}, \frac{1}{2})$.

$$\frac{\partial f(x, y)}{\partial x} = -\frac{y}{1-x} + \frac{1-y}{x}. \text{ Therefore, } \frac{\partial f(x, y)}{\partial x} \Big|_{x=\frac{1}{2}, y=\frac{1}{2}} = 0.$$

4. Find $\frac{\partial}{\partial w_j} \mathbf{x}^T \mathbf{w}$, where \mathbf{x} and \mathbf{w} are M -dimensional real-valued vectors and $1 \leq j \leq M$.

$$\mathbf{x}^T \mathbf{w} = \sum_{i=1}^M x_i w_i. \text{ Therefore, } \frac{\partial}{\partial w_j} \mathbf{x}^T \mathbf{w} = x_j.$$

5 Probability and Statistics

You should be familiar with event notations for probabilities, i.e. $P(A \cup B)$ and $P(A \cap B)$, where A and B are binary events.

In this class, however, we will mainly be dealing with random variable notations, where A and B are random variables that can take on different states, i.e. a_1, a_2 , and b_1, b_2 , respectively. Below are some notation equivalents, as well as basic probability rules to keep in mind.

- $P(A = a_1 \cap B = b_1) = P(A = a_1, B = b_1) = p(a_1, b_1)$
- $P(A = a_1 \cup B = b_1) = \sum_{b \in B} p(a_1, b) + \sum_{a \in A} p(a, b_1) - p(a_1, b_1)$
- $p(a_1 | b_1) = \frac{p(a_1, b_1)}{p(b_1)}$
- $p(a_1) = \sum_{b \in B} p(a_1, b)$

1. Two random variables, A and B, each can take on two values, a_1, a_2 , and b_1, b_2 , respectively. a_1 and b_2 are considered disjoint (mutually exclusive). $P(A = a_1) = 0.5$, $P(B = b_2) = 0.5$.

- What is $p(a_1, b_2)$?
 - What is $p(a_1, b_1)$?
 - What is $p(a_1 | b_2)$?
 - $P(A = a_1, B = b_2) = 0$
 - $P(A = a_1, B = b_1) = p(b_1 | a_1)p(a_1) = 0.5$ since $p(b_1 | a_1) = 1$
 - $P(A = a_1 | B = b_2) = 0$
2. Now, instead, a_1 and b_2 are not disjoint, but the two random variables A and B are independent.
- What is $p(a_1, b_2)$?
 - What is $p(a_1, b_1)$?
 - What is $p(a_1 | b_2)$?
 - $p(a_1, b_2) = 0.25$
 - $p(a_1, b_1) = 0.25$ since now $p(b_1 | a_1) = 0.5$
 - $p(a_1 | b_2) = 0.5$
3. A student is looking at her activity tracker (Fitbit/Apple Watch) data and she notices that she seems to sleep better on days that she exercises. They observe the following:
- | Exercise | Good Sleep | Probability |
|----------|------------|-------------|
| yes | yes | 0.3 |
| yes | no | 0.2 |
| no | no | 0.4 |
| no | yes | 0.1 |
- What is the $P(\text{GoodSleep} = \text{yes} | \text{Exercise} = \text{yes})$?

- Why doesn't $P(\text{GoodSleep} = \text{yes}, \text{Exercise} = \text{yes}) = P(\text{GoodSleep} = \text{yes}) \cdot P(\text{Exercise} = \text{yes})$?
 - The student merges her activity tracker data with her food logs and finds that the $P(\text{Eatwell} = \text{yes} \mid \text{Exercise} = \text{yes}, \text{GoodSleep} = \text{yes})$ is 0.25. What is the probability of all three happening on the same day?
 - $P(\text{GoodSleep} = \text{yes} \mid \text{Exercise} = \text{yes}) = \frac{0.3}{0.3+0.2} = 0.6$
 - Good Sleep and Exercise are not independent.
 - $P(\text{Eatwell} = \text{yes}, \text{Exercise} = \text{yes}, \text{GoodSleep} = \text{yes}) = P(\text{Eatwell} = \text{yes} \mid \text{Exercise} = \text{yes}, \text{GoodSleep} = \text{yes}) * P(\text{Exercise} = \text{yes}, \text{GoodSleep} = \text{yes}) = 0.075$
4. What is the expectation of X where X is a single roll of a fair 6-sided dice ($S = \{1, 2, 3, 4, 5, 6\}$)? What is the variance of X ?
- $E[X] = 3.5$
 $\text{Var}[X] = 2.917$
- For variance, we can do $E[(X - E[X])^2]$ or use the equivalent formulation $E[X^2] - E[X]^2$. In the first method, this gives $\frac{1}{6} \sum_{x \in \{1, 2, 3, 4, 5, 6\}} (x - 3.5)^2$
5. Imagine that we had a new dice where the sides were $S = \{3, 4, 5, 6, 7, 8\}$. How do the expectation and the variance compare to our original dice?
- $E[X] = 5.5$
 $\text{Var}[X] = 2.917$, note $\text{Var}[X + a] = \text{Var}[X]$ for scalar a