

ML System Design

1. Clarify Requirements

What is the goal? Any secondary goal?

e.g. for CTR - maximizing the number of clicks is the primary goal. A secondary goal might be the quality of the ads/content

Ask questions about the scale of the system - how many users, how much content?

2. How the ML system fits into the overall product backend

Think/draw a very simple diagram with input/output line between system backend and ML system

3. Data Related Activities

Data Explore - whats the dataset looks like?

Understand different features and their relationship with the target

- Is the data balanced? If not do you need oversampling/undersampling?

- Is there a missing value (not an issue for tree-based models)

- Is there an unexpected value for one/more data columns? How do you know if its a typo etc. and decide to ignore?

Feature Importance - partial dependency plot, SHAP values, dataschool video (reference)

(ML Pipeline: Data Ingestion) Think of Data ingestion services/storage

(ML Pipeline: Data Preparation) Feature Engineering - encoding categorical features, embedding generation etc.

(ML Pipeline - Data Segregation) Data split - train set, validation set, test set

4. Model Related Activities

(ML Pipeline - Model Train and Evaluation) Build a simple model (XGBoost or NN)

- How to select a model? Assuming its a Neural Network

1. NLP/Sequence Model

- start: LSTM with 2 hidden layers

- see if 3 layers help,

- improve: check if Attention based model can help

2. Image Models - (Don't care right now)

3. Other

- start: Fully connected NN with 2 hidden layers

- Improve: problem specific

(ML Pipeline - Model Train and Evaluation) What are the different hyperparameters (HPO) in the model that you chose and why?

(ML Pipeline - Model Train and Evaluation) Once the simple model is built, do a bias-variance tradeoff, it will give you an idea of overfitting vs underfitting and based on whether overfit or underfit, you need different approaches to make you model better.

Draw the ML pipeline (reference #3)



By **gokug6**

cheatography.com/gokug6/

Not published yet.

Last updated 7th August, 2023.

Page 1 of 8.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

ML System Design (cont)

Model Debug (reference #1)
 Model Deployment (reference#3)
 (ML Pipeline: Performance Monitoring) Metrics
 AUC, F1, MSE, Accuracy, NDCG for ranking problems etc.
 When to use which metrics?
 5. Scaling

Binary Search: Time - $O(\log n)$, Space - $O(1)$

```
def search(nums, target):
    start = 0
    end = len(nums) - 1
    mid = 0

    while start <= end:

        mid = (start + end) // 2

        # If x is greater, ignore
left half

        if nums[mid] < target:
            start = mid + 1

        # If x is smaller, ignore
right half

        elif nums[mid] > target:
            end = mid - 1

        # means x is present at mid
        else:
            return mid

    # If we reach here, then the
    element was not present
    return -1
```

DFS for In-order Tree Traversal

DFS for In-order Tree Traversal (cont)

```
> break

print()
```

DFS for Graphs

```
def DFS(self,s):
    # prints all vertices in DFS manner from a given
    source.
    # Initially mark all vertices as not visited
    visited = [False for i in range( self.V)]

    # Create a stack for DFS
    stack = []

    # Push the current source node.
    stack.append(s)

    while (len(stack)):
        # Pop a vertex from stack and print
it
        s = stack[-1]
        stack.pop()

        # Stack may contain same vertex
twice. So
        # we need to print the popped item
only
        # if it is not visited.
        if (not visited[s]):
            print(s,end=' ')
            visited[s] = True

        # Get all adjacent vertices of the
popped vertex s
        # If a adjacent has not been
visited, then push it
        # to the stack.
        for node in self.adj[s]:
            if (not visited[node]):
                stack.append(
(node)
```

```
def inOrder(root):
    current = root
    stack = []

    while True:
        if current is not None:
            stack.append(current)

            current = current.left
        elif(stack):
            current = stack.pop()
            print(current.data,
end=" ")

            current = current.right

        else:
```



By **gokug6**
cheatography.com/gokug6/

Not published yet.
Last updated 7th August, 2023.
Page 3 of 8.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Batch Normalization

```
activation_map_sample1 = np.array([
    [1, 1, 1],
    [1, 1, 1],
    [1, 1, 1]
], dtype= np.float32)

activation_map_sample2 = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
], dtype= np.float32)

activation_map_sample3 = np.array([
    [9, 8, 7],
    [6, 5, 4],
    [3, 2, 1]
], dtype= np.float32)

activation_mean_bn = np.mean([activation_map_sample1,
activation_map_sample2,
activation_map_sample3], axis=0)

#get standard deviation across different samples in
batch for each activation
activation_std_bn = np.std([activation_map_sample1,
activation_map_sample2,
activation_map_sample3], axis=0)

activation_map_sample1_bn = (activation_map_sample1 -
activation_mean_bn) / activation_std_bn
```

BCE

$$L_{BCE} = -\frac{1}{n} \sum_{i=1}^n (Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Linked List

```
class Node:

    # Constructor to initialize the node
    object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # Function to reverse the linked list
    def reverse(self):
        prev = None
        current = self.head
        while (current is not None):
            next = current.next
            current.next = prev
            prev = current
            current = next
        self.head = prev

    # Function to insert a new node at the
    beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def deleteN(head, position):
        temp = head
        prev = head
        for i in range(0, position):
            if i == 0 and position == 1:
                head = head.next
```

Linked List (cont)

```
> else:
    if i == position-1 and temp is not None:
        prev.next = temp.next
    else:
        prev = temp

    # Position was greater than
    # number of nodes in the list
    if prev is None:
        break
    temp = temp.next
return head
def search(self, x):

    # Initialize current to head
    current = self.head

    # loop till current not equal to None
    while current != None:
        if current.data == x:
            return True # data found

        current = current.next

    return False # Data Not found
def getCount(self):
    temp = self.head # Initialise temp
    count = 0 # Initialise count

    # Loop while end of linked list is not reached
    while (temp):
        count += 1
        temp = temp.next
    return count

# Utility function to print the LinkedList
def printList(self):
```

Linked List (cont)

```
> temp = self.head
while(temp):
    print(temp.data, end=" ")
    temp = temp.next

# Driver code
l1 = LinkedList()
l1.push(20)
l1.push(4)
l1.push(15)
l1.push(85)
```

Floor/Ceil

```
3//2 # floor == 1
-(-3//2) # ceil == 2
```

BFS for Level-order Tree Traversal

```
class TreeNode:
def __init__(self, key):
    self.data = key
    self.left = None
    self.right = None
def printLevelOrder(root):
    if not root:
        return []

    queue = []
    queue.append(root)

    while len(queue) > 0:
        print(queue[0].data)
        node = queue.pop(0)
        if node.left is not None:
            queue.append(node.left)
        if node.right is not None:
            queue.append(node.right)
```

BFS for Level-order Tree Traversal (cont)

```
> if node.right is not None:
    queue.append(node.right)
root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)
root.left.left = TreeNode(4)
root.left.right = TreeNode(5)
printLevelOrder(root)
```

BFS for Graphs

```
def bfs(r,c):
    q = collections.deque()
    q.append((r,c))
    visited.add((r,c))
    neighbors = [[-1,0],[0,-1],[1,0],[0,1]]
    area = 1
    while q:
        row, col = q.popleft()
        for n in neighbors:
            curInd = (row + n[0], col + n[1])
            if (curInd[0] >= 0) and (curInd[0] < rows) and (curInd[1] >= 0) and (curInd[1] < cols):
                if grid[curInd[0]][curInd[1]] and (curInd not in visited):
                    q.append(curInd)
                    visited.add(curInd)
                    area = area + 1
    return area
```

Softmax Reg PyTorch

```
import time
from torchvision import datasets
from torchvision import transforms
from torch.utils.data import DataLoader
import torch.nn.functional as F
import torch

device = torch.device("cuda: 0" if torch.cuda.is_available() else "cpu")
# Hyperparameters
random_seed = 123
learning_rate = 0.1
num_epochs = 25
batch_size = 256
# Architecture
num_features = 784
num_classes = 10
#####
### MNIST DATASET
#####
train_dataset = datasets.MNIST(root='data',
                               train=True,
                               transform=transforms.ToTensor(),
                               download=True)
test_dataset = datasets.MNIST(root='data',
                               train=False,
                               transform=transforms.ToTensor())
train_loader = DataLoader(train_dataset,
                           batch_size=batch_size,
                           shuffle=True)
test_loader = DataLoader(test_dataset,
                           batch_size=batch_size,
```

Softmax Reg PyTorch (cont)

```
> shuffle=False)
# Checking the dataset
for images, labels in train_loader:
    print('Image batch dimensions:', images.shape) #NCHW
    print('Image label dimensions:', labels.shape)
    break
##### MODEL #####
class SoftmaxRegression(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super(SoftmaxRegression, self).__init__()
        self.linear = torch.nn.Linear(num_features, num_classes)

        self.linear.weight.detach().zero_()
        self.linear.bias.detach().zero_()

    def forward(self, x):
        logits = self.linear(x)
        probas = F.softmax(logits, dim=1)
        return logits, probas
class MLP(torch.nn.Module):
    def __init__(self, num_features, num_hidden, num_classes):
        super().__init__()

        self.num_classes = num_classes

    ### 1st hidden layer
    self.linear_1 = torch.nn.Linear(num_features, num_hidden)
    self.linear_1.weight.detach().normal_(0.0, 0.1)
    self.linear_1.bias.detach().zero_()
    ### Output layer
    self.linear_out = torch.nn.Linear(num_hidden, num_classes)
    self.linear_out.weight.detach().normal_(0.0, 0.1)
    self.linear_out.bias.detach().zero_()

    def forward(self, x):
```

Softmax Reg PyTorch (cont)

```
> out = self.linear_1(x)
out = torch.sigmoid(out)
logits = self.linear_out(out)
#probas = torch.softmax(logits, dim=1)
return logits#, probas

#####
### Model Initialization
#####

torch.manual_seed(RANDOM_SEED)
model = MLP(num_features=28*28,
            num_hidden=100,
            num_classes=10)
model = SoftmaxRegression(num_features=num_features,
                          num_classes=num_classes)
model.to(device)
#####
### COST AND OPTIMIZER
#####
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
torch.manual_seed(random_seed)
def compute_accuracy(model, data_loader):
    correct_pred, num_examples = 0, 0

    for features, targets in data_loader:
        features = features.view(-1, 28*28).to(device)
        targets = targets.to(device)
        logits, probas = model(features)
        _, predicted_labels = torch.max(probas, 1)
        num_examples += targets.size(0)
        correct_pred += (predicted_labels == targets).sum()

    return correct_pred.float() / num_examples * 100
```



By **gokug6**
cheatography.com/gokug6/

Not published yet.
 Last updated 7th August, 2023.
 Page 7 of 8.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Softmax Reg PyTorch (cont)

```
> start_time = time.time()
epoch_costs = []
for epoch in range(num_epochs):
    avg_cost = 0.
    for batch_idx, (features, targets) in enumerate(train_loader):

        features = features.view(-1, 28*28).to(device)
        targets = targets.to(device)

        ### FORWARD AND BACK PROP
        logits, probas = model(features)

        # note that the PyTorch implementation of
        # CrossEntropyLoss works with logits, not
        # probabilities
        cost = F.cross_entropy(logits, targets)
        optimizer.zero_grad()
        cost.backward()
        avg_cost += cost

        ### UPDATE MODEL PARAMETERS
        optimizer.step()

        ### LOGGING
        if not batch_idx % 50:
            print ('Epoch: %03d/%03d | Batch %03d/%03d | Cost: %.4f'
                  %(epoch+1, num_epochs, batch_idx,
                    len(train_dataset)//batch_size, cost))

    with torch.set_grad_enabled(False):
        avg_cost = avg_cost/len(train_dataset)
        epoch_costs.append(avg_cost)
        print('Epoch: %03d/%03d training accuracy: %.2f%%' % (
            epoch+1, num_epochs,
            compute_accuracy(model, train_loader)))
        print('Time elapsed: %.2f min' % ((time.time() - start_time)/60))
```

