Job: https://g.co/kgs/PMYNzz

1. We A/B tested two styles for a sign-up button on our company's product page. ***100*** *visitors viewed page* ***A****, out of which* ***20*** *clicked on the button; whereas,* ***70*** *visitors viewed page* ***B****, and only* ***15*** *of them clicked on the button. Can you confidently say that page* ***A*** *is a better choice, or page* ***B****? Why?*

No. To understand why, we need to conduct a two-sided test where our null hypothesis is p_A = p_B where p represents the probability of a user clicking a button. Our alternative hypothesis is that p_A != p_B. I will conduct the test at the 95% confidence level (tail alphas equal 0.025). In our samples, our total degrees of freedom is equal to 168, or the total degrees of freedom in p_A (99) plus the total degrees of freedom in p_B (69). Our critical values for rejecting the null hypothesis are +/- 1.96.

The standard error for sample means is defined as the root of p times (1-p) over n where n is the count of samples. For page A, p_A is 20%, and the standard error is 8.9%. For page B, p_B is 21.4% and the standard error is 10.6%. The pooled standard deviation of the two samples is 4.4%. The z-score with this standard deviation is -0.33 (p-value 0.3745), which is not outside our critical value range of +/-1.96. Therefore, I cannot reject the null hypothesis, and I cannot say that A is better than B since, statistically, A is no different than B. However, if we were to increase our samples, and thereby decrease both the standard error and our confidence intervals, then I might be able reach a confident conclusion.

2. Can you devise a scheme to group Twitter users by looking only at their tweets? No demographic, geographic or other identifying information is available to you, just the messages they've posted, in plain text, and a timestamp for each message.

In JSON format, they look like this:

```
{
 "user_id": 3,
 "timestamp": "2016-03-22_11-31-20",
 "tweet": "It's #dinner-time!"
}
```

Assuming you have a stream of these tweets coming in, describe the process of collecting and analyzing them, what transformations/algorithms you would apply, how you would train and test your model, and present the results.

First, I would decode the tweet data from JSON into Python using the json module. I would ultimately convert the JSON object into a Python dictionary where the keys are the JSON keys and the values are the JSON values. This way, I can also add additional pieces of information to each tweet by simply calling the dictionary as needed. I would also create another dictionary, one reserved for information about users. User_id would be the first key entered into this user dictionary.

There are some interesting pieces of information we can get from each tweet that we can use to group users. We have all their tweets, which can tell us how long a user has been active. The timestamps can show us their specific tweeting activities over time – when they tweet, how frequently they tweet. The hashtag provides meaning to the tweet as it's an author-created index for other Twitter users. Even combining the timestamp with the hashtag can provide insights into the inner thoughts and personal daily activities of Twitter users.

I would use this data and organize it to reflect information at the user level. Through data preprocessing, I would extrapolate metrics on frequency, user engagement, user longevity, group interests by hashtag, and group common hashtags by date or time of day to understand how the user acts during the day. I would use principal components analysis to reduce the number of dimensions and data into composite dimensions that represent archetypes of Twitter users. From this PCA-reduced data, I would either run a maximum expectation clustering algorithm for clustering or K-means if we know in advance the number of desired clusters.

To present the results, I would display the clusters visually in multidimensional space if possible with clear boundaries for the clusters, and I would display the clusters as a table that showed how each cluster corresponds to the principal components, and what types of Twitter users this implies.

3. In a classification setting, given a dataset of labeled examples and a machine learning model you're trying to fit, describe a strategy to detect and prevent overfitting.

One excellent strategy to prevent overfitting is the use of dropout layers combined with cross-validated grid search using dropout rates as hyperparameters. Dropout layers are vectors of 0 or 1 that are multiplied to tensors in neural networks. For each training epoch, dropout weights are assigned 0 at the dropout rate. When multiplied by a 0-dropout weight, output from preceding nodes essentially becomes ignored by the network. Thus, we can 'shut off' nodes randomly during training.

Because we are 'shutting off' nodes randomly, and thereby allowing only random pieces of information for training, the model will be less prone to train on one specific piece or category of information and instead be forced to account for all other pieces of information as well. That is, the model is less likely to be overfitted to any one piece from a dataset.

However, that doesn't guarantee the model won't be overfitted. For this, we need to review the training curves of both our training accuracy scores and our testing accuracy scores. If a model is being trained well, we would expect the training and validation scores to be close together and improve in sync with each epoch. However, if a model is being overfitted, then we would see improved training scores relative to stagnant or falling validation scores. Divergence in these two curves as the number of epochs increases is a tell-tale sign of overfitting. If a curve starts to diverge, early-stopping, another technique used to prevent overfitting, can be used.

So, in short, a good strategy to prevent overfitting is the use of dropout layers and early stopping, and to detect is the review of training curves.

*4. Your team is designing the next generation user experience for your flagship 3D modeling tool. Specifically, you have been tasked with implementing a smart context menu that learns from a modeler's usage of menu options and shows the ones that would be most beneficial. E.g. I often use **Edit** > **Surface** > **Smooth Surface**, and wish I could just right click and there would be a **Smooth Surface** option just like **Cut, Copy** and **Paste**. Note that not all commands make sense in all contexts, for instance I need to have a surface selected to smooth it. How would you go about designing a learning system/agent to enable this behavior?*

I would create a reinforcement agent to learn from a modeler's usage. For this reinforcement agent, I would define the state space by the class of object selected and the last action performed on that instance of the object. My rationale is that a power user will often repeat the same type of process or set of steps on similar instances objects. The action space would consist of the actions a user could take. To keep the reinforcement algorithm from recommending actions that are invalid or would cause the system to crash, I would separate the state-action matrices of different types of objects. That is, I would have the reinforcement agent use one state-action matrix for one type of object, and a separate state-action matrix for another.

Anytime a user interacts with an object in a certain way (which will always be valid since the actions are just mappings to the object's properties and methods), the agent will record the interaction by adding 1 to the appropriate state-action value Q(s,a). In this context, the Q value represents how valuable a given action is to a user given the state. In this algorithm, there are no penalties; only rewards. We are defining value by how many times a given action was selected by the user. Whenever a new object is encountered, a new Q matrix is created with 'INITIALIZE' in its state space. Whenever a new action is discovered by the algorithm, it is added to both the action space (since it was the action chosen) and the state space (since it now defines the current state).

The objective of the reinforcement agent will then be to select the policy (or set of policies) with the highest expected utility value of recommending that policy. Though I think users may be predictable in how they choose their immediate action given a last action and type of object, it becomes difficult to predict beyond that (especially if the user is switching between types objects). I don't think the algorithm should consider actions beyond the next state. Therefore, the agent should only value it's immediate actions, and not consider actions of other states. The value function for each policy is equal to Q, the value of the action. The objective for the agent then becomes finding the argmax of Q(s,a) (or N argmaxes where N is the number of commands to display in the smart menu).

5. Give an example of a situation where regularization is necessary for learning a good model. How about one where regularization doesn't make sense?

Variance can be a good indicator of when to use regularization. When a feature on a dataset has a high variance, training will result in overfitting since more extreme values will be used in the training. Therefore, a regularization factor, lambda, is used to penalize large errors. Furthermore, high variance can be symptomatic of bias in the data (i.e. some unmeasured phenomenon) in which case it would be best to regularize, and therefore minimize, the effect of this bias. Situations that could lead to a feature with high variance include outliers and large datasets with limited features.

When a feature on a dataset has values within a contained set or when variance of the data is low suggesting a well-designed dataset with quality information, regularization is not necessary. We regularize data to eliminate error, so in cases where our feature options are limited, or all required information is included to make a classification, and therefore little to no error, we don't regularize. Example situations include simple multiple-choice datasets where only one of a few answers can be selected, and situations where the model is designed knowing the process by which data is generated

6. Your neighborhood grocery store would like to give targeted coupons to its customers, ones that are likely to be useful to them. Given that you can access the purchase history of each customer and catalog of store items, how would you design a system that suggests which coupons they should be given? Can you measure how well the system is performing?

Reinforcement learning is a good approach here. You need to add more detail here though. Which RL algorithm would be appropriate in this case? What would be the state space and action space? What would the rewards and penalties be given for? How would the trained model be used in the real world?

I would design a recommender system that combines both clustering and collaborative filtering to make decisions on what coupons to issue. The clustering will help simplify the data to allow for a collaborative filter. The filter will find a subset of other customers that exhibit similar properties as the target customer, but unlike the target, these customers have acted overwhelmingly positive for some item. The coupon is issued and depending on how the customer uses the coupon or not within a given time frame, the customer's response is recorded. This record is reused for other target customers in further collaborative filters.

Now, how we define 'overwhelmingly positive' is key and I would look to A/B testing to determine this. I would have the system analyze the habits of similar customers relative to the rest of the population. In this test, the offer is the same, but the target customer is different. Any offer that showed statistically significant response among similar customers relative to the customer base at large is a target candidate for prediction.

7. If you were hired for your machine learning position starting today, how do you see your role evolving over the next year? What are your long-term career goals, and how does this position help you achieve them?

If I started today, I would anticipate the first month of my work mainly helping my teammates with their projects. This way, I would get to know my teammates, the IT infrastructure, the purpose and scopes of our current projects, the company culture, and department protocol. Once I've onboarded successfully, I would be handed small projects to own. My manager and teammates would provide feedback which I would incorporate into additional small projects. Within three or four months, I hope to obtain my AWS certification and to lead a regular workflow. Over the course of the next year, I want to prove myself through the quality of my work and value to my team. My professional one-year goal is to earn a mid-manager position where I can interact with cross-functional stakeholders within the company.

For me, this job provides great technical and leadership opportunities. My long-term career goal is to be involved in a leadership role at a company that specializes in providing AI-empowered tools in traditionally non-technology fields. AI, by its ability to receive complex inputs and by its ability to learn complex tasks in an efficient manner, opens new opportunities in every industry and promises the disruptive power of the Internet – everything from decision making to tedious workload balancing. Clayton Christensen defines what a product is by the job that it does. There are plenty of jobs AI can do, which means there are plenty of non-tech market opportunities for AI.