Machine Learning Interview Submission
Sohrab Rahimi
9.29.2017

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?

On the surface, it looks like Page B is slightly better on average because 21% of the views end up in clicking whereas Page A is about 20% only. However, it is not often as simple as I just said. First and foremost, we need to make sure that the two samples that we are comparing are actually quite the same in every other respect. That is, in A/b testing, we have two sets of metrics: first, the evaluation metrics that are those variables that we aim to compare. In this case, we can define the evaluation metric as the number of clicks divided by total number of views. Second, we should have an invariant metric which would be our criteria to make sure that the only thing that is different in the two samples is our evaluation metric.

Based on our evaluation metric, we define a null and alternative hypothesis. In this case, the null hypothesis is: there is no statistically significant difference between the number of clicks in the two samples. We conduct a statistical test to reject the null hypothesis. To do this, we can assume that clicking follows a Bernoulli distribution and based on that we can calculate standard deviation and standard error. We first calculate a pooled probability (i.e. 35/170 ~0.20) and then the standard error for each :

$SE_A = (CR_A * (1-CR_A) / Visitors_A)^{1/2} = ((0.20*0.80)/100)^{1/2} \sim 0.0400$
$SE_B = (CR_B * (1-CR_B) / Visitors_B)^{1/2} = ((0.21*0.79)/100)^{1/2} \sim 0.0401$

The margin of error for the first one with 95% confidence level will be 1.96*0.04 = 0.0784 and for the second one will be about the same as well. What's that mean? It means that you can say with 95% confidence level that Page A has a conversion rate of about 0.20 ± 0.0784 and page B has a conversion rate of 0.21 ± 0.0784. It looks like there are more or less the same. This is not convincing because the number of samples that we have for each is obviously small and that greatly affects the power of analysis. If you want to really understand which page is better, you should calculate the best number of samples for each given a certain power.

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.

This is the case for NLP and there are a multitude of processes one could go through to draw useful information from these tweets. I would start with pretty simple models such as merely looking into the # (hashtag) portion of the text which assumably is intended to summarize the content of that tweet and is the most important part of it. To do this we can use the split() function and only choose those characters where # exists in them. Then we can keep account of those hashtags and for example count them for every tweet. Every tweet, therefore, can be summarized in an n-dimensional vector space, where n is the number of unique hashtags. For every tweet, 1 indicates that that specific hashtag exists in that tweet and 0 means that it doesn't. This matrix, although specifically spars, can later on be grouped by different other variables. For example, we can aggregate all the tweets by user and get a matrix of user-and-hashtag frequency. One interesting thing to explore here is using unsupervised learning methods such as clustering to find groups of similar users. To do this I would first calculate the pairwise cosine similarity matrix from the user matrix and then use spectral clustering which automatically applies a dimension reduction first and then applies k-means on the resulting eigenvectors. This can help us what are the general contents of tweets among groups of users.

Another interesting observation could be to divide up the time-stamp into time bins and aggregate the hashtags on those bins and see what sorts of tweets are more common in different times of the day and year. We can use the same clustering technique to further discover that. We can find the best number of clusters by using for example silhouette score (although it depends on the type of clustering algorithm that we use). I would then visualize the average hashtags for each clusters and only focus on those that are most different in this visualization. Since it is an unsupervised technique, by definition there is no scientific method to see whether our clustering was good unless by making sense of the differences between the clusters.

Supervised techniques can also be used for this data-set. For example, based on a given set of hashtags for a specific user, can we predict the probability of an unseen tweet belonging to that user? This might be a little tricky because we have so many labels (i.e. users). One way to make it more doable is to first apply a clustering algorithm to all tweets and then predict the probability of a tweet belonging to those resulting clusters. However, the classification process is the same: we first choose a classifier, in this case I would definitely start with Naïve Bayes, then we split the data into training, validation and test set. I would live 20% of the tweets out and then split the rest to training and validation using 10-fold CV. Then I will tune the hyper parameters using the grid search cv and choose

the best model by using the f1-score as my performance matrix. The good thing about Naïve Bayes is that it can adapt itself to a new model given a new tweet.

There are many other things that one could do such as sentiment analysis and comparing users sentiments at different times of the year, event detection by merely counting the tweets at different times, training a Doc2Vec model to compare different tweets in terms of their content and etc. I'd be happy to explain any of those!

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.

Overfitting can be prevented through different ways. First, given the size, dimension and types of our features we should select an algorithm which is suitable for that data. Some algorithms are prone to overfitting intrinsically, such as decision tree classifiers. Accordingly, the first step to avoid overfitting, if it is of serious concern, is to choose the right classification algorithm. For example, if we have too many variables and not as many observations, selecting a decision tree is not the best choice and will most likely result in overfitting. In this case, an experience machine learning engineer will first try a Naïve Bayes classifier which better fits to such data-sets.

The second step to avoid overfitting can be done by dividing our data to training, validation and testing sets in a manner that we make sure that our classifier is reliable enough to classify an unseen data point, reasonably. This is a fundamental step in almost every classification process and it aims to minimize overfitting. To do this, we first define a "performance function". This function is our criteria to compare different models. Typically, F-beta score which is a harmonic average between precision and recall is chosen as performance function. We can choose beta depending on whether precision or recall is more important for us. After this step, we split our data-set into training, validation and test set and leave the test set out. Now we have the training and validation set and there are many ways to split them. One common method is 10-fold cross validation where 90% of the other 10% is for validation and this process is repeated until all the 10 folds are used as validation, in turns. So we choose a classifier, we select a set of hyper parameters for it (e.g. using Sklearn's grid search function) we fit it in that 90% and we calculate the error from the performance function, and we repeat this 10 times and average between those iterations. Then we use slightly different model with different hyper parameters. In the end, we draw a comparison between those models and choose the model with highest F-score.

Sometime, this process is not enough. For example, our data may be unbalanced. In this cases we can use stratified sampling when splitting our data-sets. Some methods are explicitly helpful to avoid overfitting, such as ensemble methods. These methods use a combination of models (i.e. weak learners) to build a strong learner. Bagging is one ensemble method that helps us avoid overfitting, by fitting models in different portions of the data and in different variables of the feature space. By averaging all those models, we can get a more stable model that is less likely to be overfit.

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?

One way to go about this problem is by means of Reinforcement Learning, and more specifically, Q learning. In this case, we have a set of actions (e.g. smooth or whatever options that we have). We then have a set of states that represent the context for example, one possible state here is the state of "having a surface selected". What we want here is to have the machine recommend the best set of actions, given a specific state. In order to train the machine to recommend the best option, we can start with a Q table. A Q table is a dictionary-like structure with "state" being the key and "possible actions" being the value. We intend to give every possible action, a score, in a manner that the highest score represents the best possible action. How do we do this? Say we have a given state: ('1 surface selected', ''bird view enabled', 'last action was: slice') in this case, our state is three dimensional. Now we have a set of possible actions: 'slice', 'copy', 'paste', 'smooth'. We start to assign 0 values to all our actions (i.e. slice:0, copy:0, paste:0, smooth:0). Now we start by randomly suggesting an action, say "copy". If the user selects it, we add some value to 0 for 'copy', say +0.2 else we subtract some value. The amount that we add or subtract is some function of our learning rate which can be one of our tuning factors for this learning algorithm. We also define a 'decay function' which is intended to make a balance between the extent to which we suggest randomly, versus suggesting an action that we have already learned (i.e. that with bigger Q value). With this strategy, we can find the optimal policy for all sorts of states and all possible actions. Not surprisingly, as we define more states and actions, we expect more input data and training time.

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?

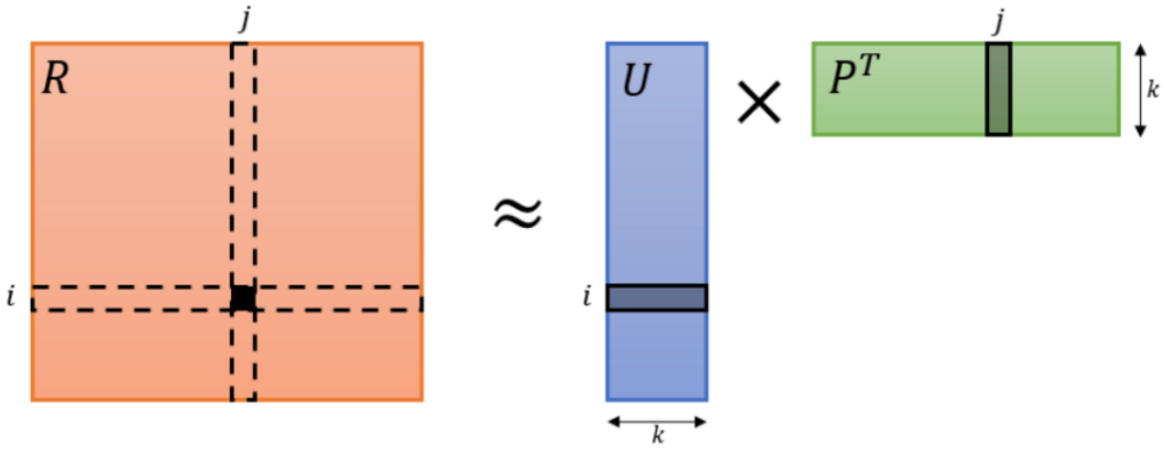
In simple words, regularization is a technique for avoiding overfitting by adding a penalty term to the loss function for every added parameter. That is, it intends to add a restriction for every variable that we add to the model, to make a balance between the optimal bias and variance. For example, imagine that we are predicting housing prices using 4 variables: area, number of bedrooms, median income in neighborhood, and building age. Our model is not as good in predicting. We collect many other variables including neighborhood employment rate, number of schools, parks, churches, ceiling type, architecture style, façade material, window types, and structure type. At this point, we realize that adding all these variables to our model significantly increases our R-square in a regression that is fit on the entire data set. Although adding these terms have improved the R-square, it will probably result in a bad score if we try it on our test data. One common regularization technique is the Lasso regression, where we add a normalization of the weights learned from the linear regression, that is, those terms that are less influential in improving the R-square will add more penalty the loss function. In the example of housing prices, having a lasso regression will help coming up with a better model where only the most important factors are included.

There are cases where regularization doesn't make sense. For example, when our model is already overtly simple. In our housing prices example, recall that we started with 4 variables and our score was already pretty low. In this case our model suffers from high bias and not high variance. Whereas

regularization is only a remedy for high variance. In this case, adding a penalty term would not make sense.

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?

This question actually reminds me of the famous Netflix competition which later became the basis for many recommendation systems. So here is how the problem looks like, we have a set of products, probably a lot of them, and you have a limited categories of coupons. Every user has purchased a certain set of products with different frequencies. So, we have a User-Product matrix where every value denotes the number of times a user has purchased a product. Probably, some users have never purchased any products but they are potential to buy those. There is no guarantee that a given user has actually purchased a product before that we have a coupon for. So, we have to somehow predict the probability that every user purchases every item. To do this, there is a simple math technique named as matrix decomposition. The idea is to decompose the user-product matrix into a multiplication of two matrices and then multiply the resulting matrices and generate a new matrix. The new matrix has a non-zero value set for every product and every user. Here is how the process looks like:



The key question here is what value for k is the best, so that the predicted values by multiplication are closest to the actual values. This is a machine learning question: we split the non-zero values of the main matrix to training, validation and test sets. In every step, we decompose the matrix with different number for k starting with k=2. We then multiply the two resulting matrices and calculate the MSE from the 10-fold cross validation. We keep doing this until a larger value of k (which is smaller than the dimension of the main matrix). What happens is that by increasing k MSE decreases at first until a certain value, and then it starts to increase due to overfitting. After finding the best number k, we multiply the resulting matrices. Now for every user we know which products are preferred the most. For every user, we can select those products for which we have coupons, and then sort them based on values. The largest value shows the best coupon choice for that user.

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?

I expect to see myself involved in real-life machine learning projects where I can apply my theoretical knowledge in practice and learn from my experiences. It is significantly important to me to gain professional experience in this path and familiarize myself with a real-life setting. I would also see myself learning from my colleagues in this process, as many of skills that I have learned so far have been through constructive team works.

I believe that long-term career goals are best achieved once they're broken to short-term goals. I would like to take the steps required to become a senior data-scientist where I can be part of multiple teams and help different groups of researchers and software engineers with my experience and skills. Therefore, finding a position in a company which has the potential to put me in that direction is very important to me and I think this position can help me achieve this milestone by involving me in a multitude of real-life projects, and help me learn from my team mates, and improve my communication and managing skills by constantly communicating with them.