Machine Learning Engineer Nanodegree

Capstone Proposal

Ambuj Agrawal
March 10th, 2019

Domain Background

I work in Qualcomm as a Modem Engineer and I wanted to develop an AI platform which can be used to catch common UE (user equipment-more commonly known as mobile phones or cellular phones) issues. Modem is the chip which allows a mobile phone to communicate wirelessly with a base station so that a user can receive/send data or voice calls.

Right now, for catching these issues we write codes that will contain lots of if-else statements or will have lots of hard-coded values and so they are not easily scalable. We will also look at different plots and manually try to find abnormalities and then debug why such an abnormality occurred.

My goal with this project is to teach an AI agent the work that we Engineers do manually which will result in lots of time saving. I want it to learn some basic relationships between different metrics and figure out if an abnormality occurred or not and if it did then what was the root cause. This way it can learn few of the basic things that I as an engineer learned when I started working in this area. Over time, we can enhance the agent with more complicated scenarios to make it a lot more sophisticated.

Problem Statement

The AI agent needs to find abnormalities in a data set and also needs to figure out why such an abnormality occurred. We will have 4 key metrics that we will use in this project-

1) SNR (signal to noise ratio)-This basically reflects how good or bad the wireless channel being seen by the UE is.
2) CQI (Channel Quality Indicator)-UE measures the SNR that it is observing and feeds back this information to the BS (base station) in the form of CQI. Higher the SNR, higher the CQI and vice-versa.

3) MCS (Modulation and Coding Scheme)-Based on the CQI report received from the UE, the BS will choose an appropriate MCS to use while sending data to the UE. Higher the CQI, higher the MCS and vice-versa.
4) Throughput-The MCS selected by the BS will directly map to a transport block size (TBS) based on the amount of frequency resources available. In this project we will assume that frequency resources assigned to the UE are constant so TBS with respect to a particular MCS will be constant. Throughput will be either equal to TBS or 0 depending on whether UE was able to decode the data or not.

The AI agent needs to figure out if observed Tput makes sense or not. If it doesn't then it needs to flag it along with what it thinks is the root cause of the problem.

We can have 2 abnormalities-

1) Tput is 0 (say BS should have chosen MCS 25 but it ended up choosing MCS 26 and so the UE was not able to decode the data)
2) Tput is lower than expected but not 0 (say BS should have chosen MCS 25 but it ended up choosing MCS 24 and so the UE received lesser data than expected)

The root cause for abnormality can be any of the following-

1) SNR and CQI are correct but MCS chosen is wrong
2) SNR is correct but CQI chosen was wrong resulting in lower/higher MCS than expected for the corresponding SNR
3) SNR estimation itself was wrong which resulted in lower Tput

Basically, the agent will need to learn the relationship between SNR⇔CQI and CQI⇔MCS. Additionally, it will also need to monitor SNR and if it sees abrupt jumps in SNR across time then it needs to flag that as well.

Datasets and Inputs

As I don't want to use proprietary data, a simple model will be used to generate the dataset. The model will be close enough to what is seen in the real world.

Following relationships will be used to generate data-

SNR	CQI	MCS	TBS
0	0	0	0
1	1	2	2000
2	2	4	4000
3	3	6	6000

4	4	8	8000
5	5	10	10000
6	6	12	12000
7	7	14	14000
8	8	16	16000
9	9	18	18000
10	10	20	20000
11	11	22	22000
12	12	24	24000
13	13	25	25000
14	14	26	26000
15	15	28	28000
16	15	28	28000
17	15	28	28000
18	15	28	28000
19	15	28	28000
20	15	28	28000

SNR will be varied from 0⇔20⇔0 at 2 units per sec for 100 seconds. Data will be generated every10ms. That will constitute one dataset. We can have as many datasets as needed based on how well the agent is learning.

In each dataset, approximately 6% of the samples will be corrupted: 2% CQI samples, 2% MCS samples and 2% SNR samples. Example of each of the three corruption scenarios is shown below-

1) CQI corruption: Say the SNR was 13, so from the table above UE should report CQI 13, but this value will be corrupted, and UE may report say CQI 10. Thus, it will be allotted MCS 20 only resulting in TBS/Throughput of 20000 only instead of 25000 which it would have got had it reported the correct CQI.
2) MCS corruption: Say the UE reports CQI 12 but BS somehow interprets it as 13 and thus allocates MCS 25 to the UE. As this MCS is higher than what UE asked for, it will result in UE not able to decode the data as SNR is not good enough to receive such a big TBS. Thus, throughput in this case will be zero.
3) SNR corruption: SNR variation should be smooth as we are varying it at a constant rate of 2 units/per sec. But there can be instances where SNR jumps happen. For ex, SNR goes like this- 9.00,9.02,9.04,5.42,9.06,9.08

The corruption will happen randomly using Gaussian noise.

Solution Statement

As the dataset would mostly be error free, I will employ one of the supervised learning/neural network algorithms to learn the relationships between different metrices. Once the AI agent has developed good enough understanding of these relationships, it can try and predict throughput based on SNR and if the predicted value and the actual value differ by more than a threshold than it can flag the throughput value obtained by the model as erroneous. Then it needs to figure out the root cause using other relationships that it has learned. More details are provided in the "Project Design" section.

Benchmark Model

As a benchmark model, we will use K-Nearest Neighbor algorithm for finding out a jump in the SNR and Stochastic Gradient Descent for learning and predicting other relationships-SNR⇔CQI, CQI⇔MCS and MCS⇔TBS. It is possible that for CQI⇔MCS and MCS⇔TBS SGD will be a pretty good model anyways as the relationships are linear. So it would be interesting to see how much better we can do compared to SGD.

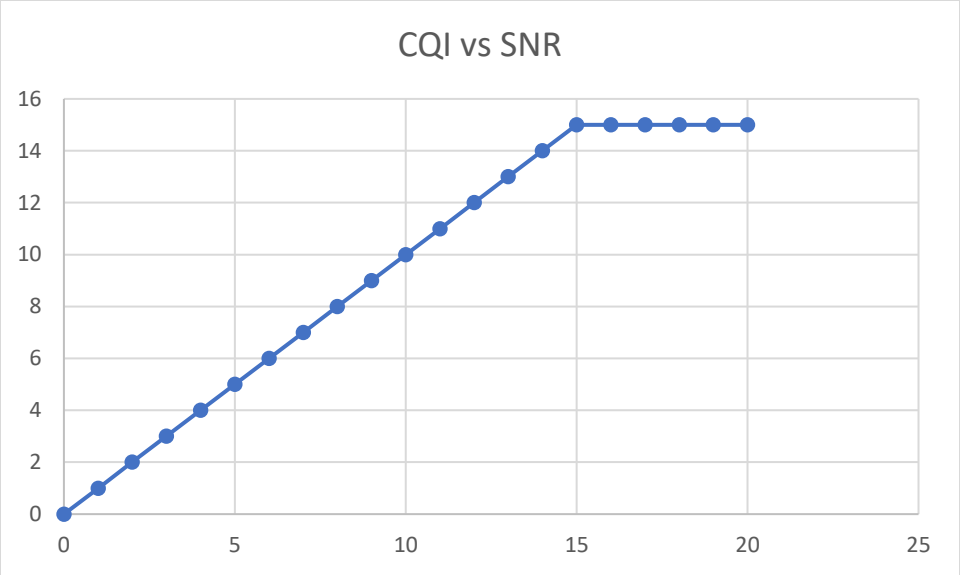
Evaluation Metrics

We will calculate the $F_{0.5}$ scores for the following metrics-

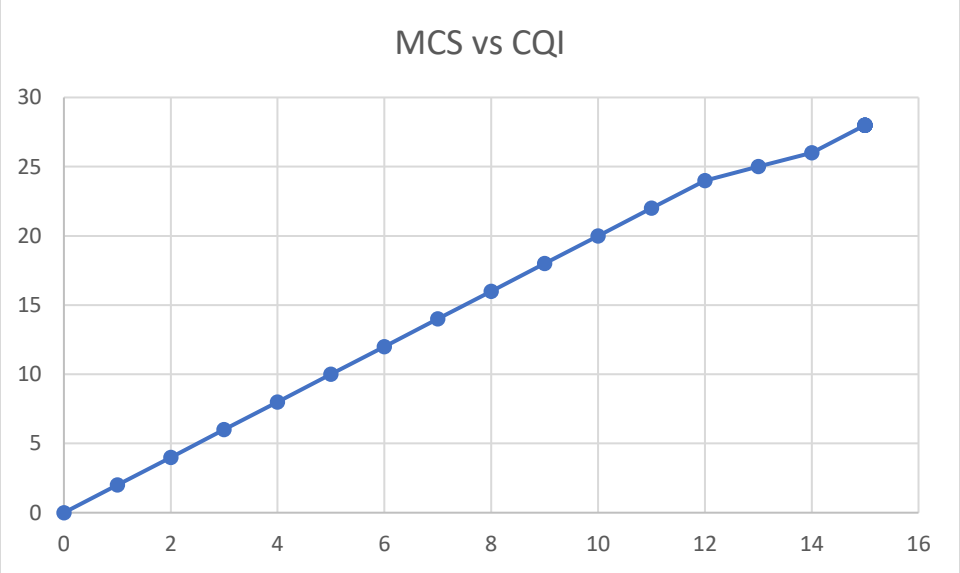
1) Total number of Errors: All the errors in the entire dataset
2) CQI Errors: All the CQI errors in the entire dataset
3) MCS Errors: All the MCS errors in the entire dataset
4) SNR Errors: All the SNR errors in the entire dataset

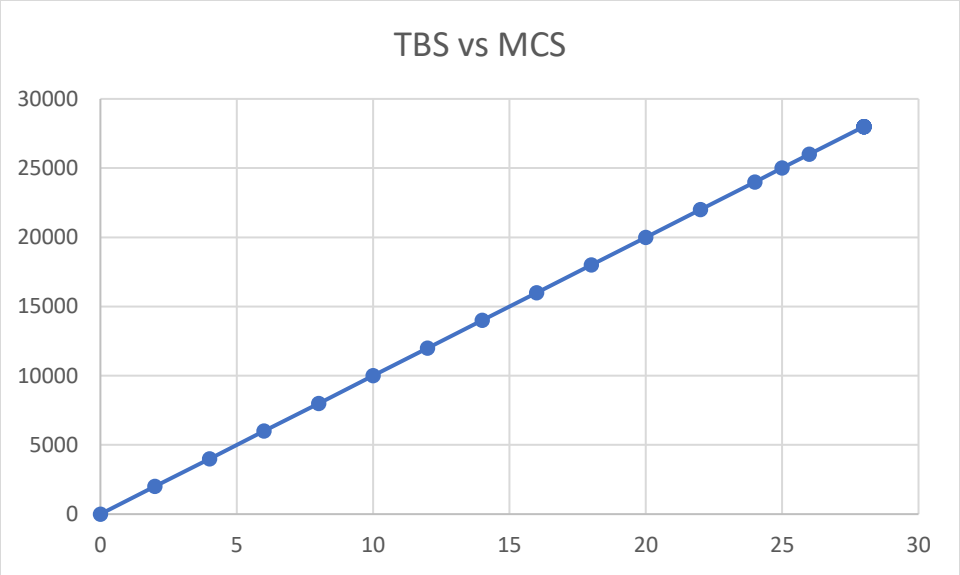
Project Design

As can be seen from the relationship table provided in the "Datasets and Inputs" section, most of the relationships are linear. The SNR⇔CQI relationship is linear up to a point and then it flattens out as shown below



But CQI⇔MCS and MCS ⇔ TBS relationships are almost linear as shown below.





Keeping the above facts in mind, this is how the workflow would be-

1) Use a clustering algorithm (for ex: DBSCAN) to figure out if SNR is varying smoothly or not. Any sudden jumps in SNR should be caught with the help of the clustering algorithm
2) The SNR and CQI relationship will be learnt through a neural network.
3) The CQI⇔MCS and MCS⇔TBS relationship will be learnt through a linear regression algorithm
4) The AI agent will learn the abovementioned relationships.
5) Once the agent has learnt well enough, then we will go about finding the errors in the following manner-
    a. Find out if a particular SNR point is part of a smooth transition or a result of a sudden jump. Flag it if it is a sudden jump.
    b. If SNR point is good then predict CQI value. If the difference b/w predicted and actual value is greater than a threshold then flag it.
    c. If CQI is also good then predict MCS value. If the difference b/w predicted and actual value is greater than a threshold then flag it.
    d. If MCS is also good then predict TBS value. If the difference b/w predicted and actual value is greater than a threshold then flag it.

This way I think we should be able to come up with a good AI agent which should be able to catch some obvious errors. The agent will serve as a good starting point and we can keep on improving it to catch more and more complicated errors in future.