1. 프로젝트 소개

본 프로젝트는 IITP에서 진행하는 IITP-Purdue 프로그램 내 캡스톤 디자인 프로젝트의 결과이다. 본 팀의 분야는 ‘Big data and Machine learning’ 이었으며 Kaggle사의 Competition 중 ‘KKBox’s Music Recommendation Challenge’를 주제로 프로젝트를 진행하였다. KKBOX는 대만의 음악 스트리밍 서비스를 제공하는 기업으로 이 대회에서 이용자와 노래에 관련된 데이터 셋을 제공한다. 이 데이터 셋을 기반으로 이용자가 어떠한 특정 노래를 청취 했을 때, 그 특정 노래를 한 달 내에 재 청취할 확률을 예측한다. 주어진 여러 종류의 데이터를 가공하고 머신러닝 알고리즘을 통해 예측에 적용한다. 개발 라이브러리로는 ‘TensorFlow’, ‘Scikit-Learn’을 사용하였고 시행 모델로는 ‘Light Gradient Boosting’, ‘Neural Network’, ‘Random Forest’ 가 있으며 개발언어로는 ‘Python’, ‘R’을 사용하였다.

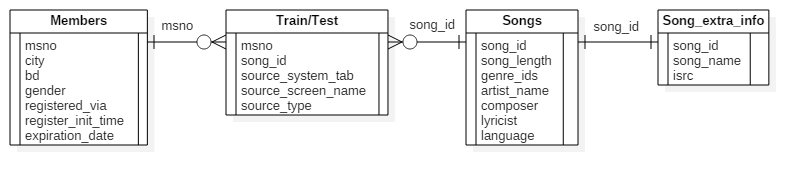
2. 프로젝트 목표

본 프로젝트는 음악 스트리밍 서비스 사의 입장에서 최대 이익을 창출하는 목표의 부분으로서, 이용자(user)와 관련된 방대한 양의 데이터를 통해 소비자의 행동을 예측한다. 데이터 셋을 기반으로 이용자가 어떠한 특정 노래를 청취 했을 때, 그 특정 노래를 한 달 내에 재 청취할 확률을 예측한다. 이 예측은 이용자의 새로운 노래와 아티스트에 대한 호감도와 연계될 수 있으며 새로운 이용자에게 효율적인 노래 추천 서비스까지 제공할 수 있게 된다. 이에 각 사용자의 특정 노래 별 재 청취율의 예측 정확도를 최대로 높이는 것을 목표로 한다.

3.

4. 모델

4.1. 데이터 분석



|  |  |  |  |
| --- | --- | --- | --- |
| 데이터 | 의미 | 형식 | 비고 |
| msno | 회원 일련번호 | Hashed ID(256bit) | Base64 Encoding |
| city | 도시 | Nominal |  |
| bd | 연령(세) | Integer |  |
| gender | 성별 | Binary(male/female) |  |
| registered\_via | 가입경로 | Nominal |  |
| registered\_init\_time | 가입일 | Date(yyyymmdd) |  |
| expiration\_date | 회원 만료일 | Date(yyyymmdd) |  |
| song\_id | 곡 일련번호 | Hashed ID(256bit) | Base64 Encoding |
| song\_length | 음악 길이(ms) | Integer |  |
| genre\_ids | 장르 번호 | (Nominal)\* |  |
| artist\_name | 가수 | (String)\* |  |
| composer | 작곡가 | (String)\* |  |
| lyricist | 작사가 | (String)\* |  |
| language | 언어 | Nominal |  |
| song\_name | 곡 제목 | String |  |
| isrc | ISRC 일련번호 | String | 곡 발매연도 포함 |
| source\_system\_tab | 감상 메뉴 1 | Nominal |  |
| source\_screen\_name | 감상 메뉴 2 | Nominal |  |
| source\_type | 감상 메뉴 3 | Nominal |  |

The characteristics of the data includes:

* Most of the features in the data are nominal form.
* Some features can have multiple values separated by bar(|), which means the data is not even in a 1st-normal form(atomized values).
* A large portion of the member data is missing. For example, the age feature(bd) has more than 60% of zeroes(missing values). The values should be dealt properly for the good predictions.

<작성중… 4-1 나원이가 맡게 되면 이거 공유해도 좋을 듯>

4.2. GBM

4.3. Neural Network

Artificial Neural Network(NN) is a re-arising method of machine learning, which mimics the activity on brain cells (neurons). If the states of the input neurons is given, the output is given by

where represents weights, bias, and an activation function, respectively. Weights and biases would be randomly set first, and improved throughout the training, by manually calculating back-propagation formula or just running some optimizations. Such neurons are linked to build a model; layers (groups of neurons) are used for the simplicity.

Recently researchers developed some variants of NNs including Convolutional NN for image recognition and Recurrent NN for serial data analysis. However, as these models assume spatial and temporal locality of input layers, they are not appropriate for this project as the project analyzes separated features. In this project, we used layers of dense networks mainly.

Some python libraries including Tensorflow and Theano aim to support neural networks. We used GPU implementation of Tensorflow for this project.

4.3.1. Preprocessing the data

4.3.1.1. Categorical Data

Neurons only accept numbers as the level of activation, making the NN rather ineffective to deal with categorical data. To use NN with nominal data, we could have following approaches to convert them into numbers.

* Gender can be directly processed such as (male=1, female=0). Nulls can be mapped into 0.5, or the portion of males from gender-known members
* Complete Disjunctive Table (CDT, or informally, ‘Dummy variables’)

CDT introduces columns for each source categories, and encodes the value as 0 or 1 depend on whether the column indicates the value of source category.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Genre |  | ID | Genre\_Pop | Genre\_Rock | Genre\_Electronic |
| 1 | Pop |  | 1 | 1 | 0 | 0 |
| 2 | Rock | → | 2 | 0 | 1 | 0 |
| 3 | Pop|Electronic |  | 3 | 1 | 0 | 1 |
| 4 | (null) |  | 4 | 0 | 0 | 0 |

(The details how to deal with nulls(ID 4) and multiple values(ID 3) may vary by methods.)

CDT is still one of the simplest methods to encode categorical variables and becomes the base of other categorical data processing methods. Large number of distinct values in the categories may lead to the explosion of features; one can use some workarounds such as retaining ‘Top n’ categories.

* Multiple Correspondence Analysis(MCA)

For multiple related categorical variables, MCA converts them to tuples of numbers, by applying PCA to their CDTs after having some transformations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Source1 | Source2 | Source3 |  | ID | F1 | F2 |
| 1 | My Library | My Profile | My Profile more |  | 1 | 0.88 | 0.79 |
| 2 | Radio | Radio | Listen to | → | 2 | 0.22 | 0.17 |
| 3 | Playlist | Others Profile | Purchase |  | 3 | 0.65 | 0.23 |
| … | … | … | … |  | … | … | … |

If one visualizes the table using the numbers in the tuples as coordinates, he or she may notice some underlying structures and relationships between the categories.

Note that nominal features other than sources and genres are not used in this analysis. The features include city, registered\_via, language, and artists; we expect the performance rise is negligible if we add these features by converting them into numbers. We only created CDT for Top 32 genres and all source features to use in NNs.

4.3.1.2. Data Transformation and Normalization

The relative magnitude of those numbers are sometimes important. If we use ReLU(Rectified Linear Unit) as the activation function, large input values may make the model converge slow, compared to the use of Logistic function. Optimally, we can adjust the input data to be at between 0 and 1. We applied some nonlinear transformations to reflect the property of data or remove skewness on the data.

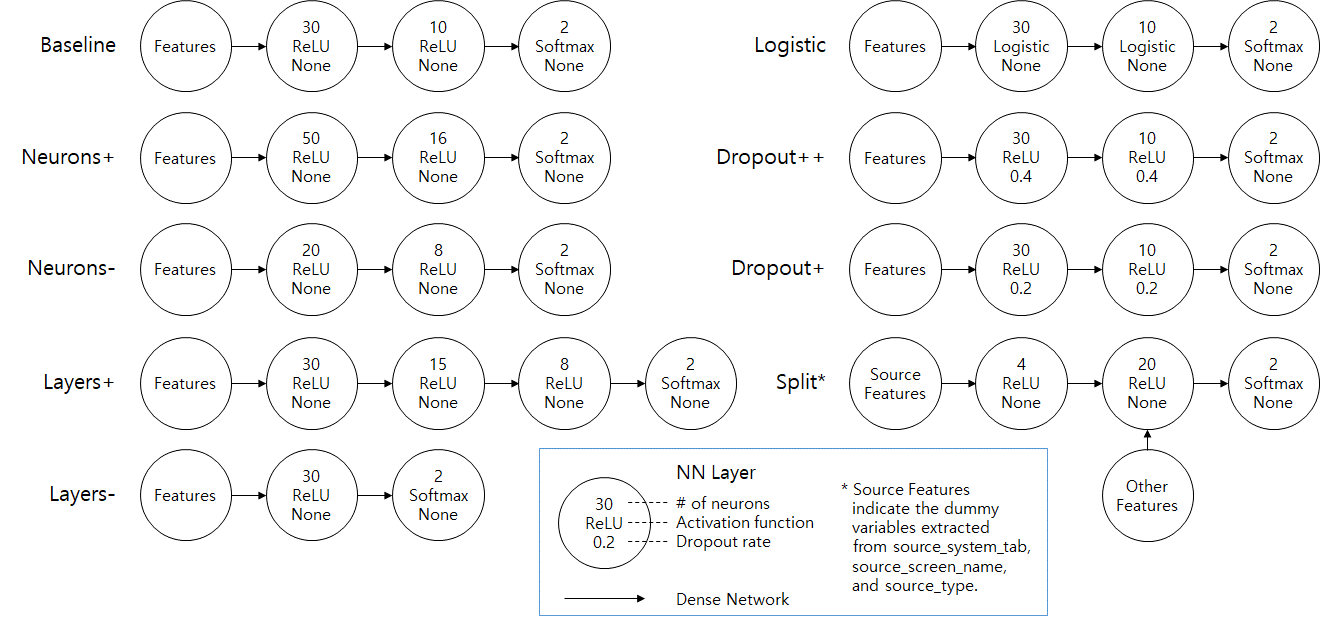
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Normalizer** generates a nonlinear transformation mapping min\_value to 0 and max\_value to 1.  The function should be an increasing function to obtain meaningful results. | | | | |
| def trimmer(x):  return 0 if x<0 else 1 if x>1 else x  def normalizer(min\_value, max\_value, func, positive=False, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) or (positive and x<=0) \  else trimmer((func(float(x))-func(max\_value))/(func(min\_value)-func(max\_value))) | | | | |
| Features | func | min\_value | max\_value | Notes |
| bd (age, yr) | Sqrt | 9 | 49 | Adjusting median(~27) near 0.5 |
| song\_length (ms) | Log | 32768 | 1048576 |  |
| member\_duration (s) | Log | 1 | 444873600 | reg\_init\_time – expiration\_date |

|  |  |  |  |
| --- | --- | --- | --- |
| **Decayer** generates an exponential decay transformation when the current point(which mapped to 1) and the decaying rate(as the half-life) are given. | | | |
| def decayer(current, halflife, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) else 1 if float(x)>current \  else math.pow(0.5,(current-float(x))/halflife) | | | |
| Features | current | halflife | Notes |
| song\_year (yr) | 2017 | 10 | Data from isrc |
| registration\_init\_time (s) | 1488258000 | 100000000 | current: Feb 28, 2017 |

For example, as recent songs have more listens than older ones and the popularity of songs usually have exponential decay, it is reasonable that we preprocess the release years of the songs using a decaying function (we chose half-life 10 years in this case). The decaying function relatively spreads out the recent song, making the recent songs more distinguishable.

4.3.2. Network Design

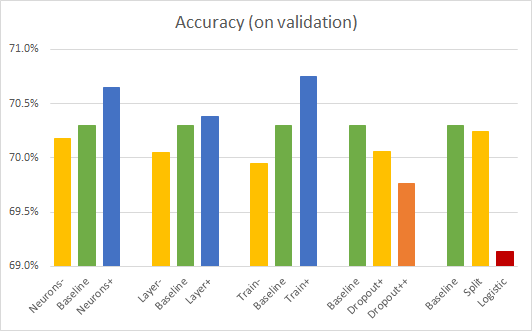
To examine effects of network modification, we set the different configurations of the networks. The baseline network is a 3-layered ReLU network with no dropouts. We changed the number of the neurons and the number of layers, changed the dropout rates, used Logistic function as the activation function, and split the inputs to follow different path on the network.



Aside from the shape of the networks, we modified iteration number of trains. The baseline network uses 50000 steps; we also tried 30000(Train-) and 100000(Train+) steps of trains.

4.3.3. Results

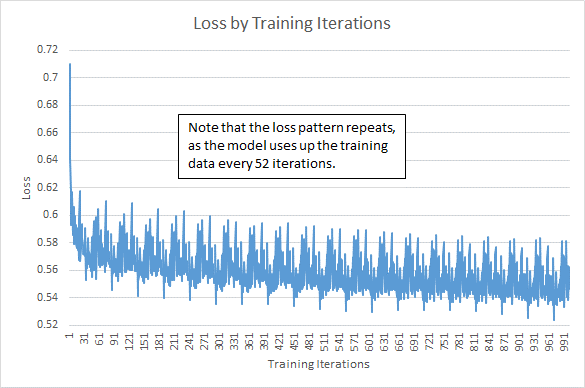
As we have expected from ineffectiveness of treating nominal data by NN, the general results are around 70% validation accuracy, which is far more inaccurate than the results of LightGBM.



Still, we could observe tendency of accuracy by the model parameters including number of neurons and the number of layers. While addition of the neurons, the layers, or the training iterations improves the model as expected, the dropout technique seems to prevent the models to improve. When we split data and pass it to different locations of the network separately, the results are better than Neurons-(28 neurons) and Layers-(30 neurons) models while using only 24 neurons.

It is normal that the score from Kaggle uploads are lower than local validation. The difference roughly indicates how much the model overfits the data. Upload of results from Neurons+ model yields 62.172% score. Compared to validation accuracy 70.652%, the difference is roughly 8.5%p.

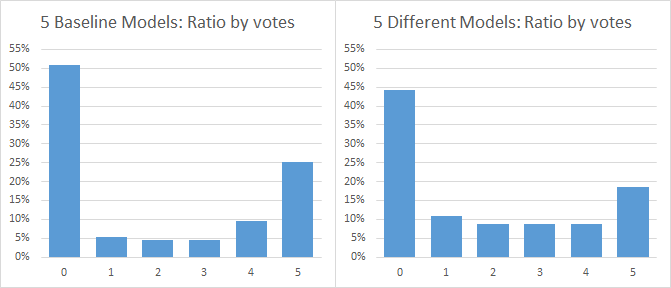
Observing the losses during the iterations, we could see decrease of the losses like the following diagram of Train+ run.



4.3.4. Aggregation of multiple networks by voting

Voting is one of the simplest ensemble methods, letting multiple models to have multiple predictions and choosing the result which most of the models predicted. The voting procedure is more effective if the models have similar accuracy and produce different predictions.

We applied voting to 5 executions of single model (Baseline), and another 5 executions of different models (2 Baselines, Neurons+, Layers+, Split).



It turns out that even the different networks produce similar predictions. We have 61.262% submission score for single model voting, and 61.983% for different model voting, which is not very different from the original baseline model.

4.3. Neural Network

인공 신경망(NN)은 최근 재조명받고 있는 머신러닝 기법 중 하나로, 뇌세포의 활동을 모방하여 데이터를 처리한다. 입력 뉴런의 상태 가 주어질 때, 현재 뉴런의 상태는 다음과 같이 계산된다.

여기서 는 각각 weight, bias, 그리고 activation function을 의미한다. Weight와 bias는 처음 무작위로 설정된 후, 학습 과정에서 back-propagation 식을 직접 계산하거나 최적화 알고리즘을 수행함으로써 개선되어 나간다. 이러한 뉴런이 모여 전체 모델을 구성하는데, 편의를 위해 뉴런들의 layer가 주로 사용된다.

최근 연구자들은 이미지 인식 등을 위한 Convolutional NN과 순차 데이터 분석을 위한 Recurrent NN 등 여러 변형된 layer 구성을 활용하고 있다. 그러나 이러한 모델은 각각 입력 layer의 공간적, 시간적 locality를 가정하고 있으므로, 그러한 연관이 없는 여러 feature들을 분석하는 우리 프로젝트에는 적절하지 않다. 이 프로젝트는 layer간의 모든 뉴런이 연결된 dense network 구성을 사용하였다.

Tensorflow와 Theano 등의 몇몇 파이썬 라이브러리가 NN 구현에 초점이 맞추어져 있다. 이 중 GPU를 활용하는 Tensorflow 버전을 이번 프로젝트에 사용하였다.

4.3.1. 데이터 전처리

4.3.1.1. 분류 데이터의 처리

각각의 뉴런은 활성 상태를 숫자로 나타내며 모든 계산이 수치로 이루어지므로, 분류 데이터를 다루기에 NN은 비교적 부적합하다. NN을 nominal data에 사용하기 위해서는, 다음의 과정을 거쳐 데이터를 숫자로 변환해 주어야 한다.

* 성별과 같은 Binary Data는 (male=1, female=0)처럼 바로 변환할 수 있다. 이 경우 Null 값은 단순히 0.5로 매핑하거나, 성별이 알려진 사람 중 남성의 비율을 사용해도 된다.
* Complete Disjunctive Table (CDT, 또는 ‘Dummy variable’)

CDT는 각각의 분류마다 하나씩의 열을 만들어서, 원래 테이블에 있던 값이 현재 열이 나타내는 분류와 같으면 1, 다르면 0을 부여하는 방식이다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Genre |  | ID | Genre\_Pop | Genre\_Rock | Genre\_Electronic |
| 1 | Pop |  | 1 | 1 | 0 | 0 |
| 2 | Rock | → | 2 | 0 | 1 | 0 |
| 3 | Pop|Electronic |  | 3 | 1 | 0 | 1 |
| 4 | (null) |  | 4 | 0 | 0 | 0 |

(Null(ID 4)과 다중 값 처리(ID 3)의 세부사항은 방법마다 다를 수 있다.)

CDT는 분류 데이터를 인코딩하는 가장 간단한 방법 중 하나이며, 다른 분류 데이터 처리 방법의 기반이 되기도 한다. 서로 다른 분류의 수만큼의 열이 생성되므로 분류의 수가 많은 경우 ‘explosion of features’ 현상이 발생할 수 있으며, 이를 피하기 위해 상위 n개의 분류에 대해서만 열을 생성하는 등의 방법을 사용할 수 있다.

* Multiple Correspondence Analysis(MCA)

분류 데이터가 여러 개 있는 경우, MCA는 데이터들을 실수로 변환한다. 내부적으로는 분류 데이터에 대한 CDT를 생성한 후 변환을 거쳐 PCA에 넣는 방식으로 계산한다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Source1 | Source2 | Source3 |  | ID | F1 | F2 |
| 1 | My Library | My Profile | My Profile more |  | 1 | 0.88 | 0.79 |
| 2 | Radio | Radio | Listen to | → | 2 | 0.22 | 0.17 |
| 3 | Playlist | Others Profile | Purchase |  | 3 | 0.65 | 0.23 |
| … | … | … | … |  | … | … | … |

MCA에서 출력된 숫자를 좌표로 생각하여 시각화 해 보면, 데이터의 구조나 관계에 대해서 추가 정보를 얻을 수도 있다.

분석 과정에서 감상 경로와 장르를 제외한 다른 분류 데이터(city, registered\_via, language, and artists)는 NN 모델에서 입력값으로 사용되지 않았다. 이는 데이터 분석 및 다른 모델의 수행 과정에서 이들을 사용했을 때 처리 과정에 비해 정확도의 향상이 높지 않은 수준이라 추정했기 때문이다. 상위 32개 장르와 감상 경로에 대해서는 CDT를 적용한 후 모델의 입력으로 사용했다.

4.3.1.2. 데이터 변환 및 정규화

수치 입력값의 경우에는 상대적인 크기가 간혹 문제가 될 수 있다. ReLU(Rectified Linear Unit)를 activation function으로 사용하는 경우 상대적인 수치가 크게 다르면 모델의 수렴 속도가 저하될 수 있다. 가장 좋은 방법은 입력값을 비슷한 크기(예: 0과 1 사이)로 조정해 주는 것인데, 이를 위해 데이터의 특성을 반영하고 skewness를 완화시키는 비선형 변환을 사용하였다.

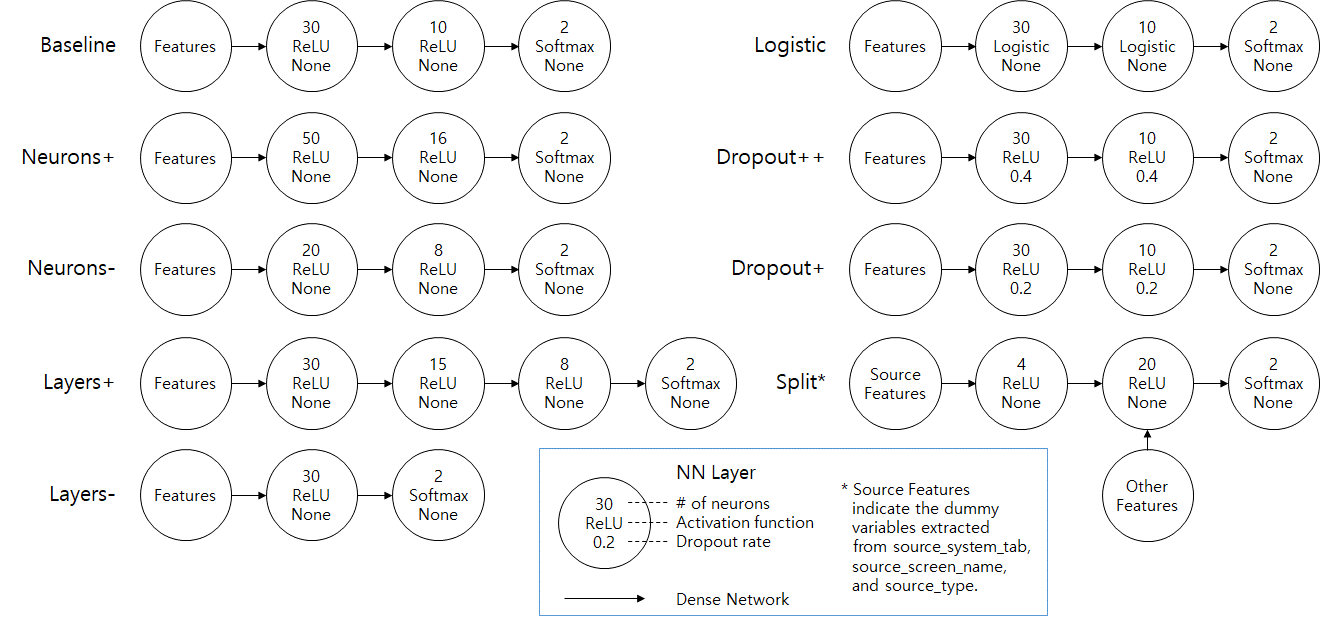
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Normalizer**는 비선형 함수 func를 적용한 뒤, min\_value를 0으로, max\_value를 1로 맞추어 주는 변환 함수를 생성한다. 유효한 변환을 하기 위해서 func는 증가함수여야 한다. | | | | |
| def trimmer(x):  return 0 if x<0 else 1 if x>1 else x  def normalizer(min\_value, max\_value, func, positive=False, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) or (positive and x<=0) \  else trimmer((func(float(x))-func(max\_value))/(func(min\_value)-func(max\_value))) | | | | |
| Features | func | min\_value | max\_value | Notes |
| bd (age, yr) | Sqrt | 9 | 49 | Adjusting median(~27) near 0.5 |
| song\_length (ms) | Log | 32768 | 1048576 |  |
| member\_duration (s) | Log | 1 | 444873600 | reg\_init\_time – expiration\_date |

|  |  |  |  |
| --- | --- | --- | --- |
| **Decayer**는 현재 시점을 1로 할 때, 지수 함수에 따라 해당 시점의 감소한 값을 계산하는 함수를 생성한다. 이 때 인자로 주어진 반감기가 적용된다. | | | |
| def decayer(current, halflife, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) else 1 if float(x)>current \  else math.pow(0.5,(current-float(x))/halflife) | | | |
| Features | current | halflife | Notes |
| song\_year (yr) | 2017 | 10 | Data from isrc |
| registration\_init\_time (s) | 1488258000 | 100000000 | current: Feb 28, 2017 |

예를 들어, 최신 곡은 과거의 곡들에 비해 감상 빈도가 더 높을 것이며, 노래의 인기는 보통 지수적 감소 패턴을 보인다. 따라서 곡의 발매 연도를 Decayer(여기에서는 반감기 10년을 사용함)로 전처리할 수 있으며, 이렇게 하면 최신 곡들의 수치가 퍼지는 대신 과거 곡들의 수치가 비교적 비슷한 값으로 바뀌어 최신 곡들에 대해 더 민감한 분석을 할 수 있게 된다.

4.3.2. Network 디자인

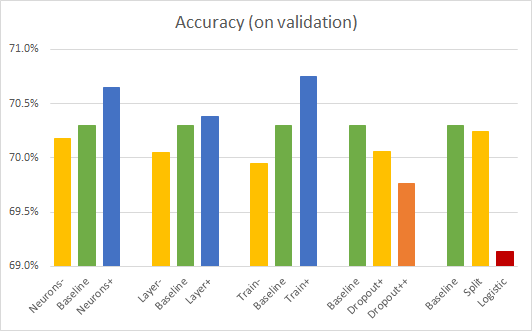
Network 조정에 대한 효과를 알아보기 위해, network를 여러 다른 방법으로 구성하였다. 기준이 되는 network(Baseline)는 dropout이 없는 3-layer network로, ReLU를 activation function으로 사용한다. 뉴런의 수, layer의 수, dropout 비율, activation function 등을 변화시켜 정확도를 확인했으며, input을 분할해 보기도 하였다.



Network의 형태와는 별도로, train 반복 횟수도 조정하였다. Baseline의 경우 50000 step을 사용했으며, 이를 30000(Train-) 및 100000(Train+)으로 조정한 모델도 분석해 보았다.

4.3.3. 결과

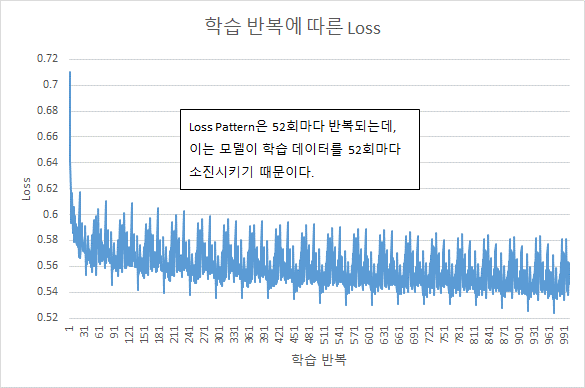
NN으로 분류 데이터를 처리하기 어렵다는 점에서 예상했듯이, 전체적인 validation 결과는 약 70% 정도의 정확도를 보여 LightGBM에 비해 부정확한 것으로 나타났다.



다만 뉴런과 layer의 수, Train 기간 등에 따른 정확도 변화의 경향은 확인할 수 있었다. 뉴런이나 layer의 추가, 긴 학습이 정확도를 향상시킴과 달리, Dropout 기법은 그에 방해가 되는 것으로 나타났다. 데이터를 감상 경로와 그 외의 데이터로 분리해서 처리한 결과(Split), 24개의 뉴런을 사용했음에도 불구하고 Neurons-(28개)와 Layers-(30개)보다 좋은 결과를 얻을 수 있었다.

Kaggle에 업로드한 결과 점수는 validation에서 얻은 정확도보다 낮은 것이 일반적이며, 그 차이는 대략적으로 모델이 데이터를 얼마나 overfitting하고 있나를 나타낸다. Neurons+ 모델로부터 얻은 결과를 업로드하여 62.172%의 점수를 얻었으며, validation 정확도 70.652%와 비교했을 때 그 차이는 약 8.5%p 정도이다.

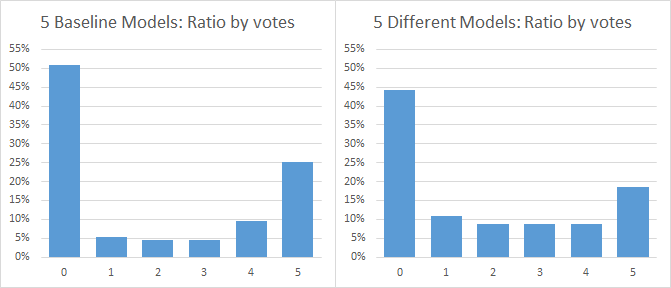
학습을 반복하면서 loss의 변화를 관찰하면 다음 그림(Train+ 모델)과 같은 loss의 감소를 확인할 수 있다.



4.3.4. Voting을 통한 여러 NN 결과의 통합

Voting은 가장 간단한 앙상블 모델 중 하나로, 여러 모델의 결과를 모아 그 중 가장 많은 모델이 나타내는 결과를 선택하는 방법이다. Voting 과정은 사용된 모델들이 비슷한 정확도를 가지면서 서로 다른 결과를 도출하는 상황에서 효과적으로 사용될 수 있다.

동일한 NN 모델(Baseline)을 5회 수행한 결과와 서로 다른 모델(Baseline 2회, Neurons+, Layers+, Split)을 수행한 결과를 이용하여 Voting을 적용하였다.



그 결과, 서로 다른 모델에서조차 비슷한 결과를 도출하고 있는 것으로 확인되었다. 정확도는 Kaggle 업로드 기준으로 61.262%(Baseline 5회), 61.983%(서로 다른 모델 5회) 정도로, Baseline 단독 수행과 크게 다르지 않은 것으로 보인다.