WSDM Challenge

Recommender Systems

Kenneth Emeka Odoh

25 Jan, 2018 (Kaggle Meetup | SFU Ventures Labs)

Vancouver, BC

Talk Plan

Bio

Education

- Masters in Computer Science (University of Regina) 2013 - 2016

Work

- Research assistant in the field of Visual Analytics 2013 - 2016
  - (https://scholar.google.com/citations?user=3G2ncgwAAAAJ&hl=en)

- Software engineer working in an IOT startup (Current position) 2017 -

Competition

- Competition rule
    - Only 5 submissions per day
- Data Exploration
    - Data visualization
- Evaluation
- Primer on recommender systems

Competition: Data Exploration

Nature of Data

- Data of Listening habits.
- Data are chronologically ordered.
    - Implicit time series data without timestamps.
    - Timestamps are excluded for sake of avoiding data leakages.
- Balanced data sets.

What to predict

- Given a data set of listening habit of users.
- If a song has been listened in recent past, predict if it will be listening to again in the same month.
- If the song is listened to again by the same user in the same month, output 1 or 0 otherwise

Competition: Background

- URL (https://www.kaggle.com/c/kkbox-music-recommendation-challenge)
- Sister competition on churn prediction by same company.
- Total price $5000

Class of Problem

- Classification
- Sequence prediction ?

Competition: Data files

- train.csv
  - msno, song_id, source_system_tab, source_screen_name, source_type, target

- songs.csv
  - song_id, song_length, genre_ids, artist_name, composer, lyricist, language

- members.csv
  - msno, city, bd, gender, registered_via, registration_init_time, expiration_date

- song_extra_info.csv
  - song_id, song name, isrc

Competition: Visualization of Data

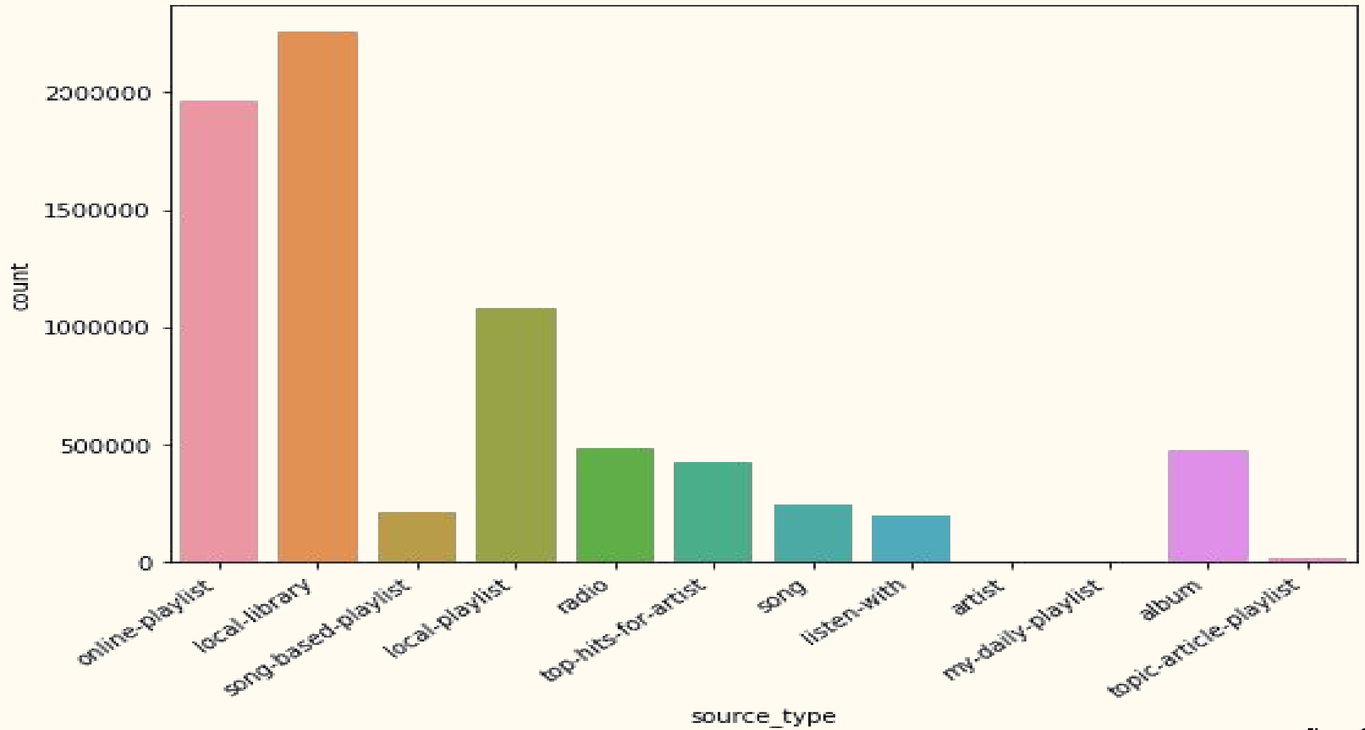


Figure 1: Count vs Source type

[jeru666]

Competition: Data files

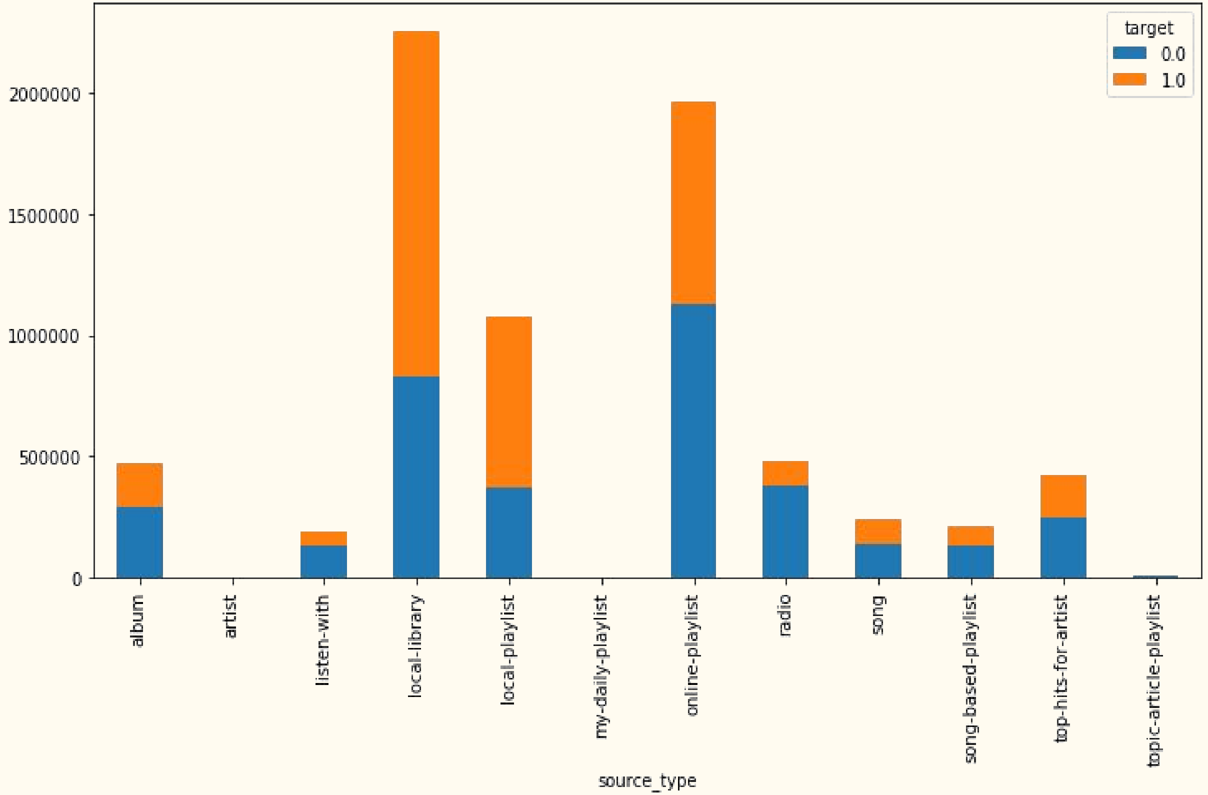


Figure 2: Count vs Source type grouped by labels [jeru666]

Competition: Data files

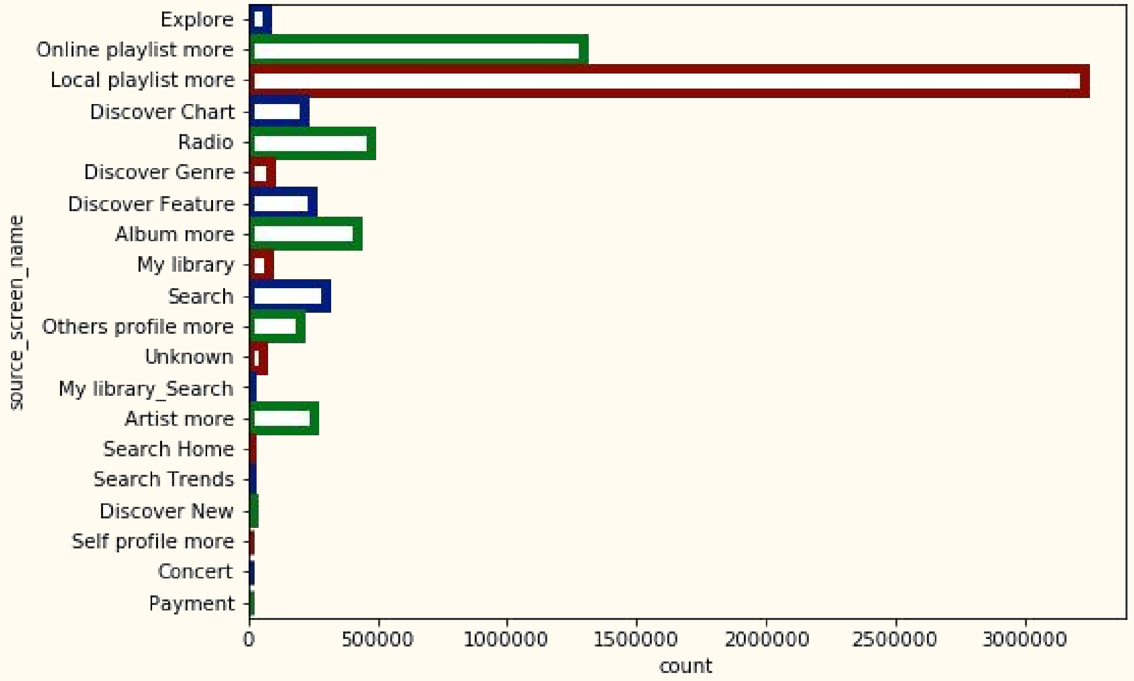
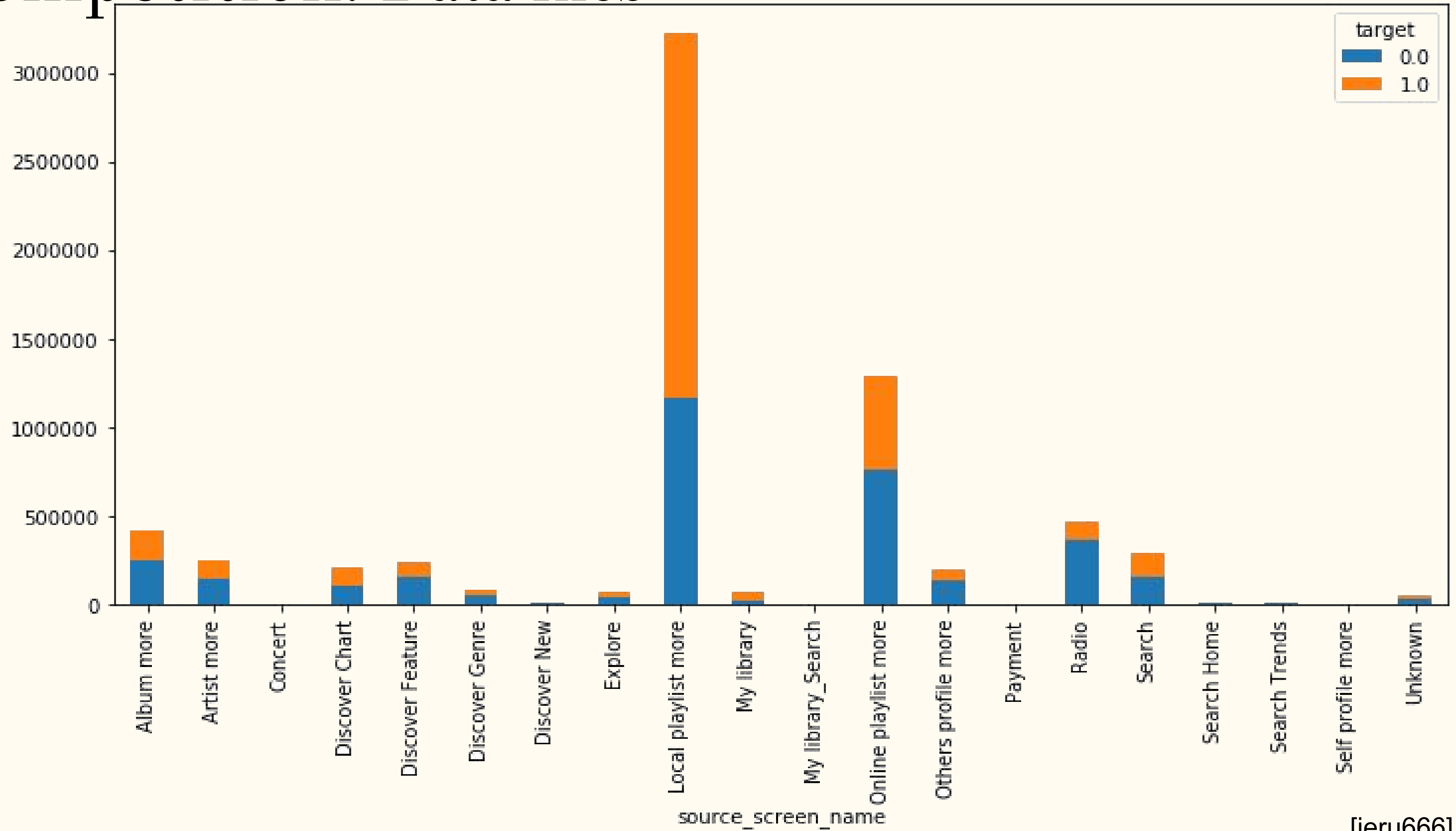


Figure 3: Source screen names vs Count

[jeru666]

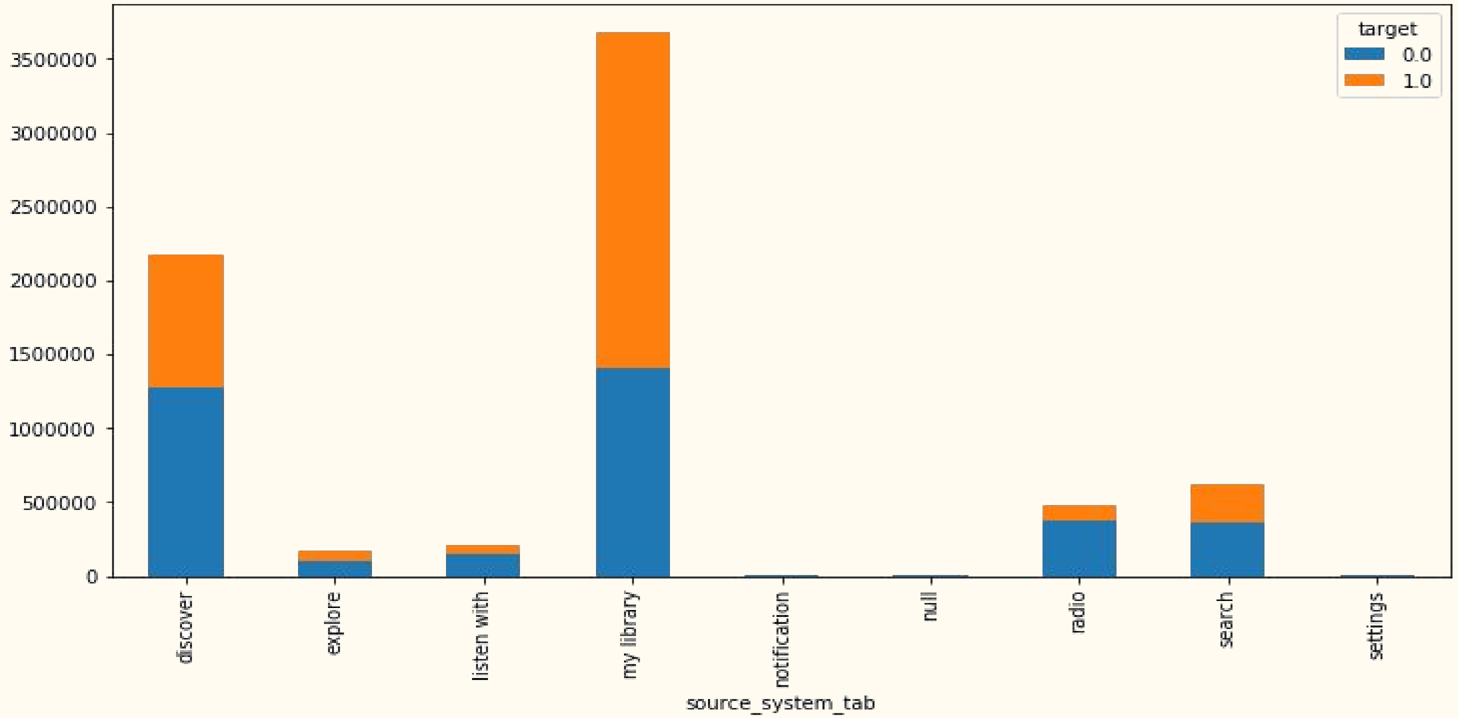
Competition: Data files



[jeru666]

Figure 4: Count vs Source screen names grouped by label

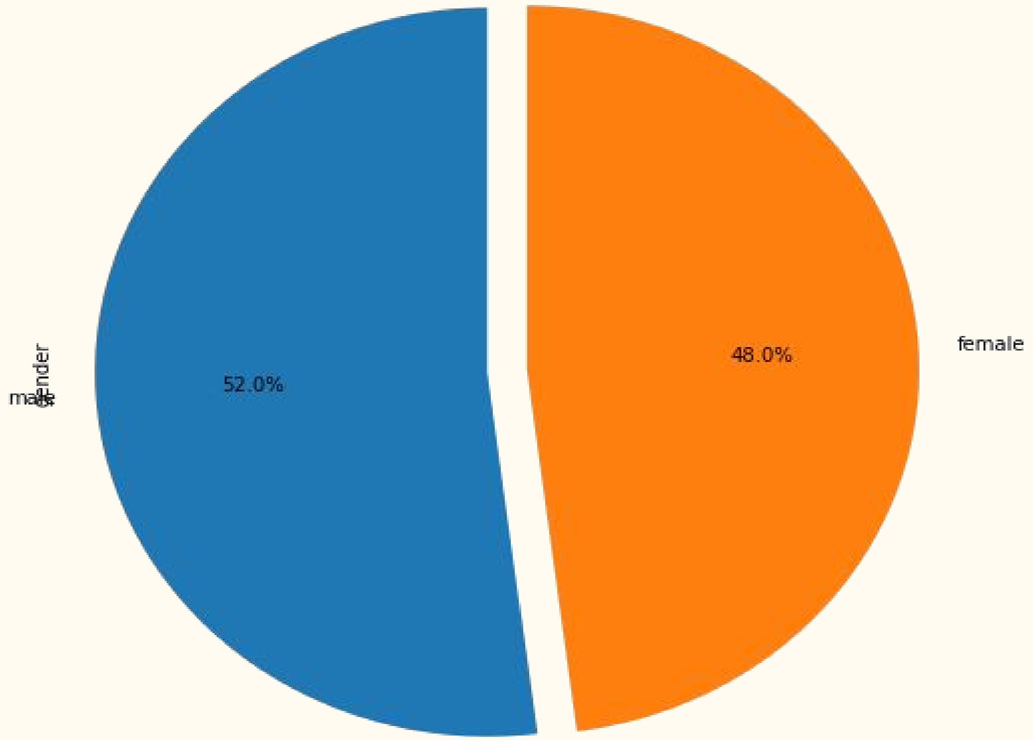
Competition: Data files



[jeru666]

Figure 5: Count vs Source system tab

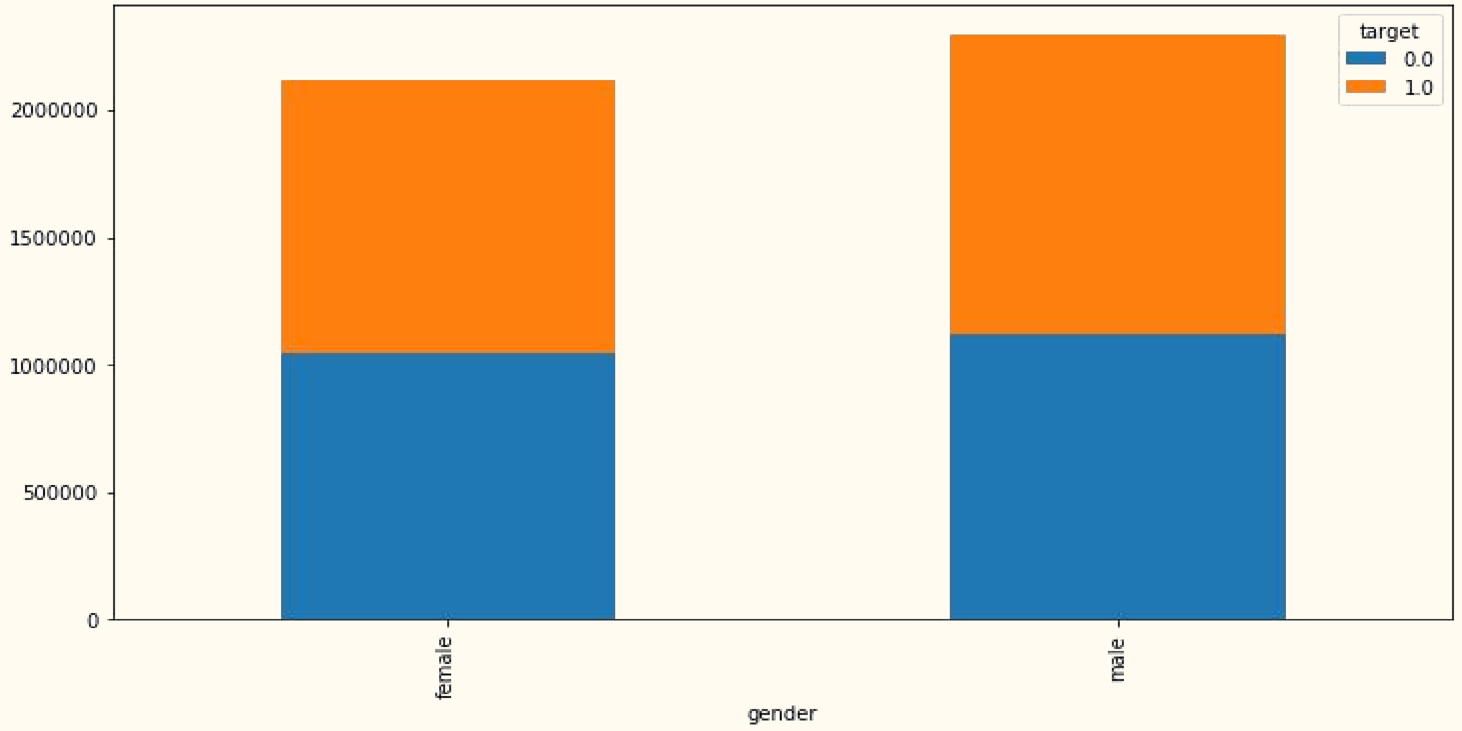
Competition: Data files



[jeru666]

Figure 6: Proportion by Sex

Competition: Data files



[jeru666]

Figure 7: Proportion by Sex grouped by label

Competition: Data files

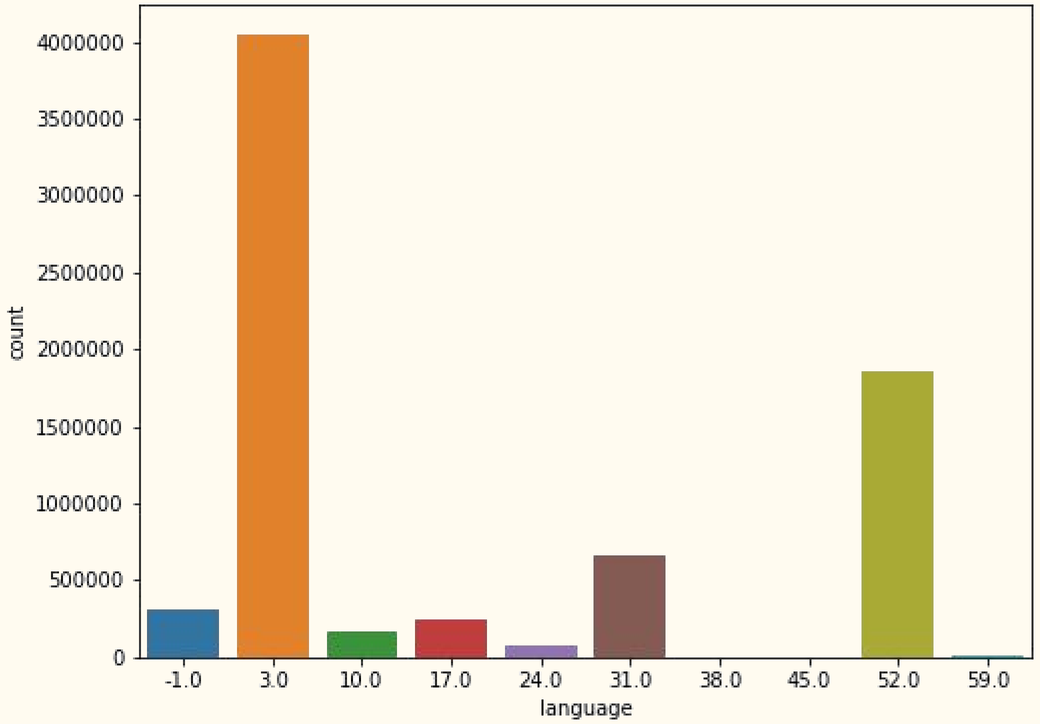


Figure 8: Proportion by Language

[jeru666]

Competition: Data files

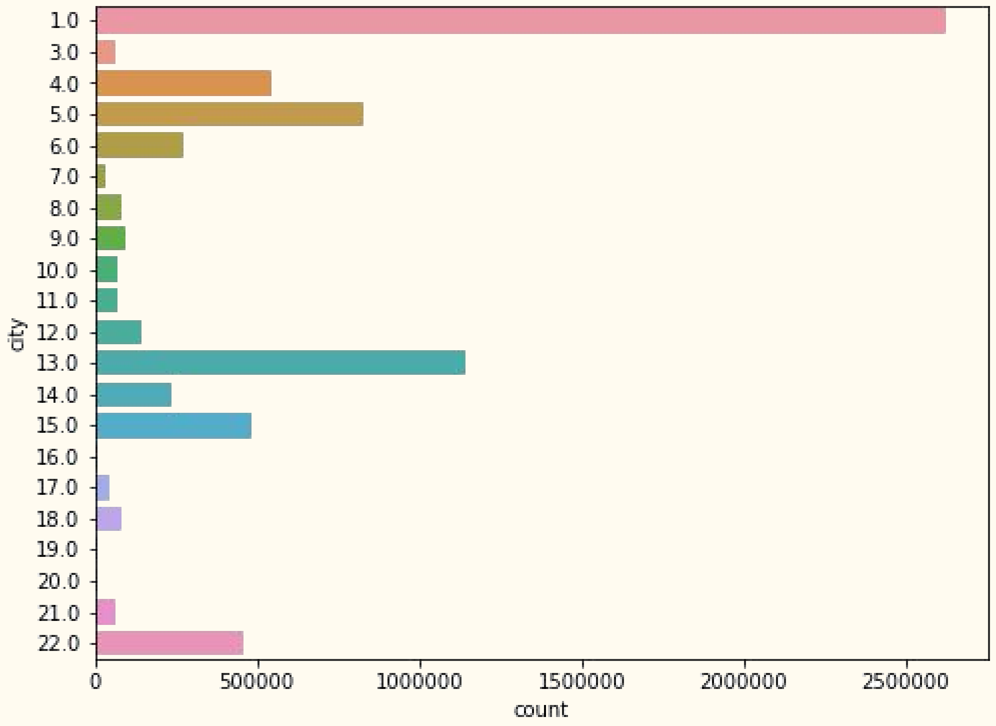
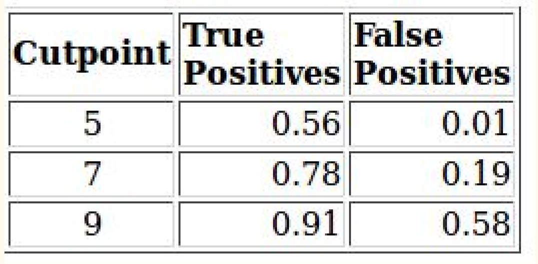


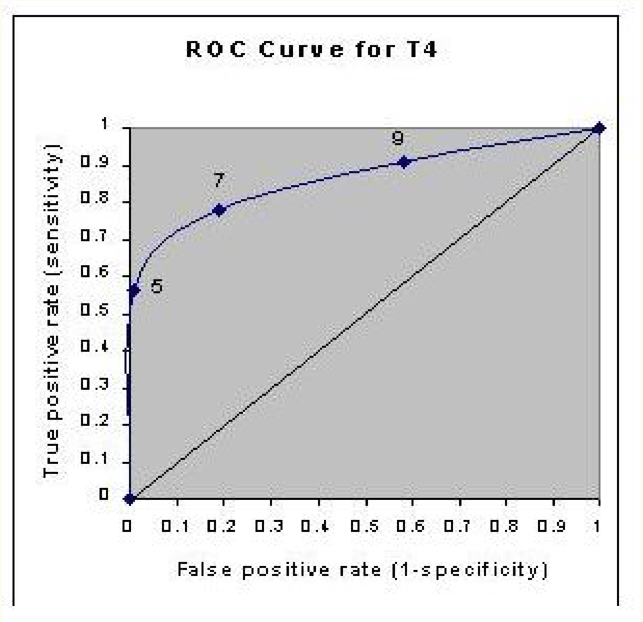
Figure 9: City vs Count

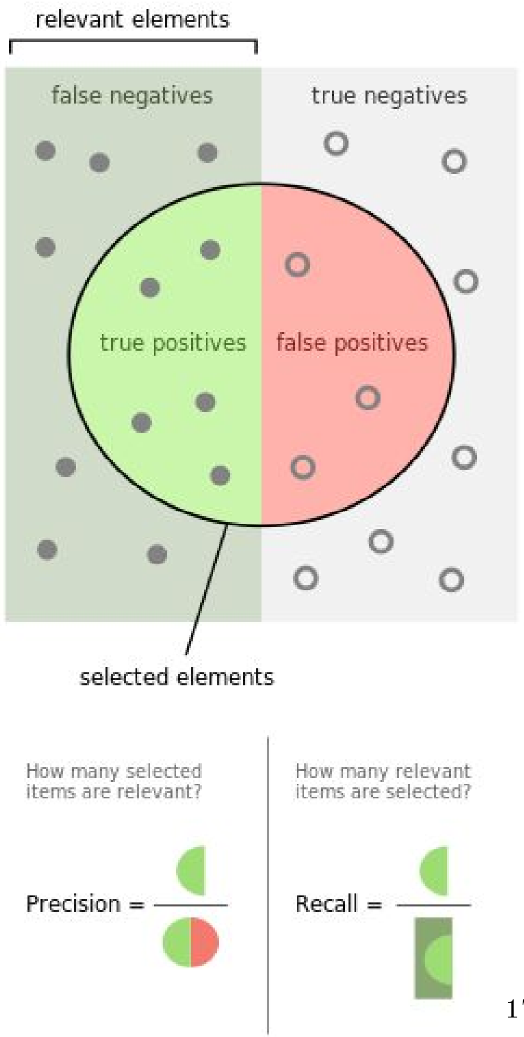
[jeru666]

Competition: Evaluation (AUC)



Cutpoint	True Positives	False Positives
5	0.56	0.01
7	0.78	0.19
9	0.91	0.58





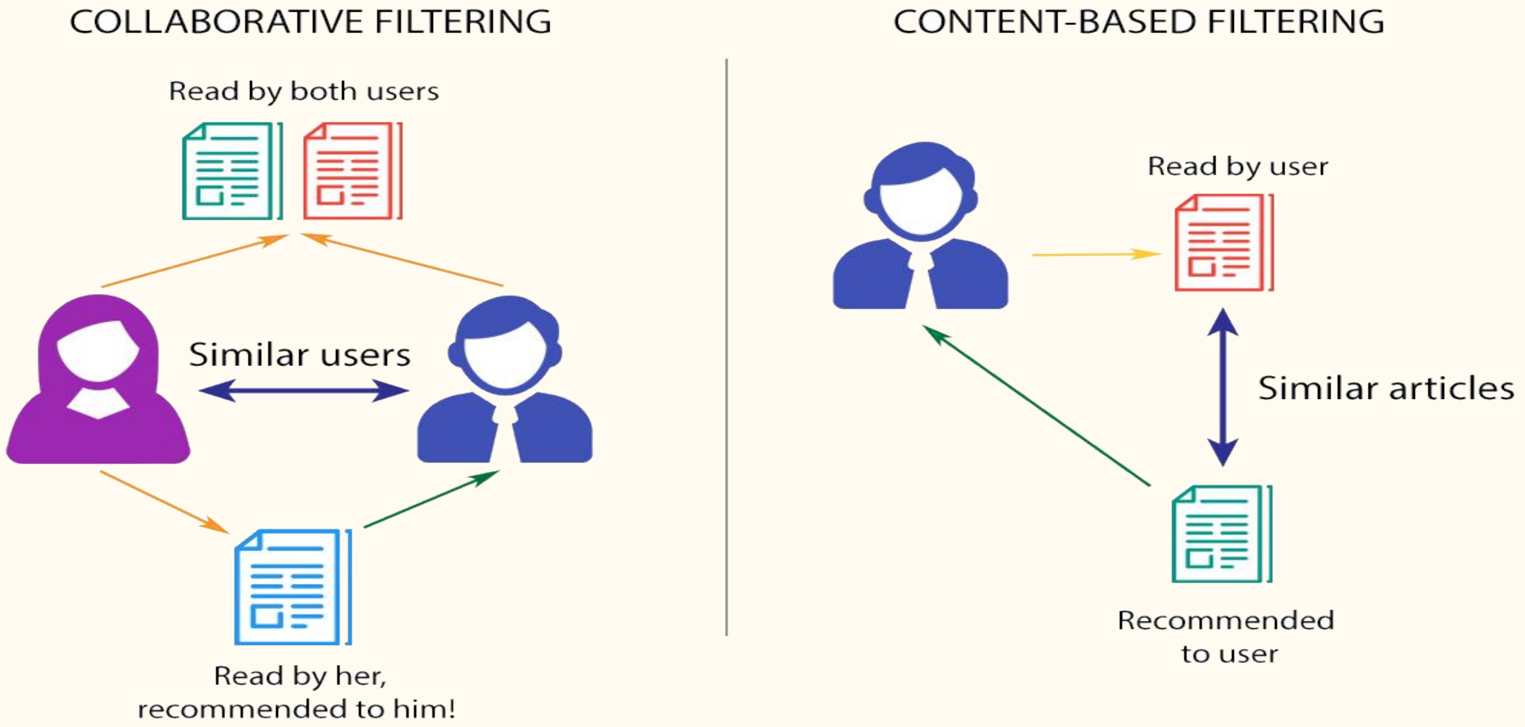
Optimizing PR space -> optimizing ROC space [Davis .et.al]

Image credit: Walber & http://gim.unmc.edu/dxtests/roc2.htm

Evaluation Metric

- AUC per user vs AUC per group ?
- What about **GAUC**?
    - calculate the average of the AUC scores of each user.
        - Cold start problem
- AUC vs F-score vs RMSE ?

Competition: Recommender Systems



U: set of users

I : set of items

R: set of rating of users on items.

$\mathbf{I_u}$**:** set of items rated by a user, u.

f : $\mathbf{U} \times \mathbf{I} \rightarrow \mathbf{R}$

This is usually a sparse matrix and as such so form of data imputation is needed

The most common form of recommender systems

- Best-item recommendation
- Top-N recommendation

For a user, $\mathbf{u_a}$, and **i*** has estimated the highest value

$$\mathbf{i^*} = \underset{\mathbf{j \in I \backslash I_u}}{\operatorname{argmax}} \; f(\mathbf{u_a}, \mathbf{j})$$

WINNING SOLUTIONS

1st Place Solution

- Similar to Click Value Rate (CVR)
    - Click-through rate (CTR) is the ratio of users who click on a specific link to the number of total users who accessed the resource.
- Relistening is like purchasing, and first listening is like clicking.
- Not suitable for latent-factor based or neighborhood-based collaborative filtering is not the best way for this problem.
- **Missing not at random**.
- Data is sparse as a user-song pair is very informative.
- Data is time sensitive.

Ensembling

0.6 * LightGBMs and 0.4 * NNs

- Single LightGBMs is 0.74440 on public LB with a learning rate of 0.1.
- Single LightGBMs is 0.74460 on public LB with a learning rate of 0.05.
- Single LightGBMs with bagging is 0.74584 on public LB with a learning rate of 0.05.

For NNs, my best 30-ensemble on the same feature set can get 0.74215, and bagging ensemble of NNs on different feature set can get 0.74333.

The predictions of LightGBMs and NNs are quite uncorrelated (correlation coefficient is about 0.92). The ensemble of single LightGBM and single 30-ensemble of NN can get 0.7489+. The ensemble of ensembles can reach 0.7498+.

Data preprocessing

https://kaggle2.blob.core.windows.net/forum-message-attachments/259372/8131/model.png

Analyzing Missing Data

Missing-data mechanisms

- Missing completely at random: This occurs when the probability of missing is same for all cases.
- Missing at random: The probability of missing depends only on available information.
- Missingness that depends on unobserved predictors
- Missingness that depends on the missing value itself.

Handling missing data

- Exclude missing data
- Mean / median imputation
- Last value carried forward
- Using information from related observation

[http://www.stat.columbia.edu/~gelman/arm/missing.pdf]

3rd Place Solution

Identified how to generate features from the future. We can listen or not listen to the same artist by the same user in the future.

Add matrix factorization to the feature to use xgboost and catboost.

last 35% for future feature

earlier 65% for history

The size of X_train, X_val was about 20% of data.

The history can be generated using target. Other features uses all data. The generated features in the context of categorical features.

1) mean of target from history by categorical features (pairs and triples)

2) count – the number of observations in all data by categorical features (pairs and triples)

3) regression – linear regression target by categorical features (msno, msno + genre_ids, msno+composer and so on)

4) time_from_prev_heard – time from last heard by categorical features (msno, msno + genre_ids, msno+composer and so on)

5) time_to_next_heard – time to next heard by categorical features (pairs and triples which include msno)

6) last_time_diff – time to the last heard by categorical features

7) part_of_unique_song – the share of unique song of artist that the user heard on all unique song of artist

8) **matrix_factorization** – LightFM. I use as a user msno in different context (source_type) and the msno was the feature for user.

I use last 35% for fit. Also I drop last 5% and again use 35%. Such a procedure could be repeated many times and this greatly improved the score (I did 6 times on validation and this gave an increase of 0.005, but in the end I did not have time to do everything and did only 2 times on test).

As a result, I fit xgboost and catboost on two parts of the data using and without the matrix_factorization feature. And finally I blend all 8 predictions.

The code is available here:

https://github.com/VasiliyRubtsov/wsdm_music_recommendations/blob/master/pipeline.ipynb

Factorization Machine

- By factoring into a number of latent vectors.
- lends itself to kernels.

Let us dive into factorization in a new thread.

https://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf

Observed Characteristics of factorization machine

- Ideal for handling missing data

6th Place Solution

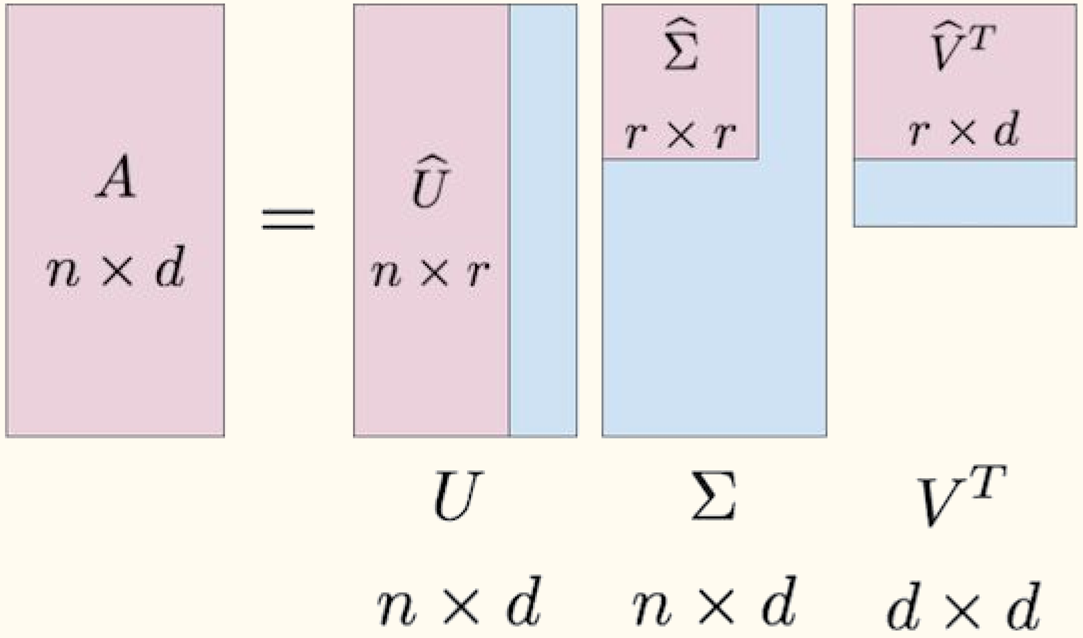
I use an ensemble (average) of lightgbm. The best single model public LB 0.726

Feature Engineering

1) raw features for both song, user and context
2) SVD matrix factorization latent space of user-song interaction matrix
3) SVD matrix factorization score of user-song interaction matrix
4) collaborative filtering (CF) score (top 100 neighborhoods) of user-song interaction
5) similarity of artist, lyricist, composer, genre,language, source_system_tab, source_type, soure_screen_name,within each user.
6) source_system_tab, source_type, soure_screen_name categorical variable rate of each user.
7) **word embedding** of artist, lyricist, composer, genre.

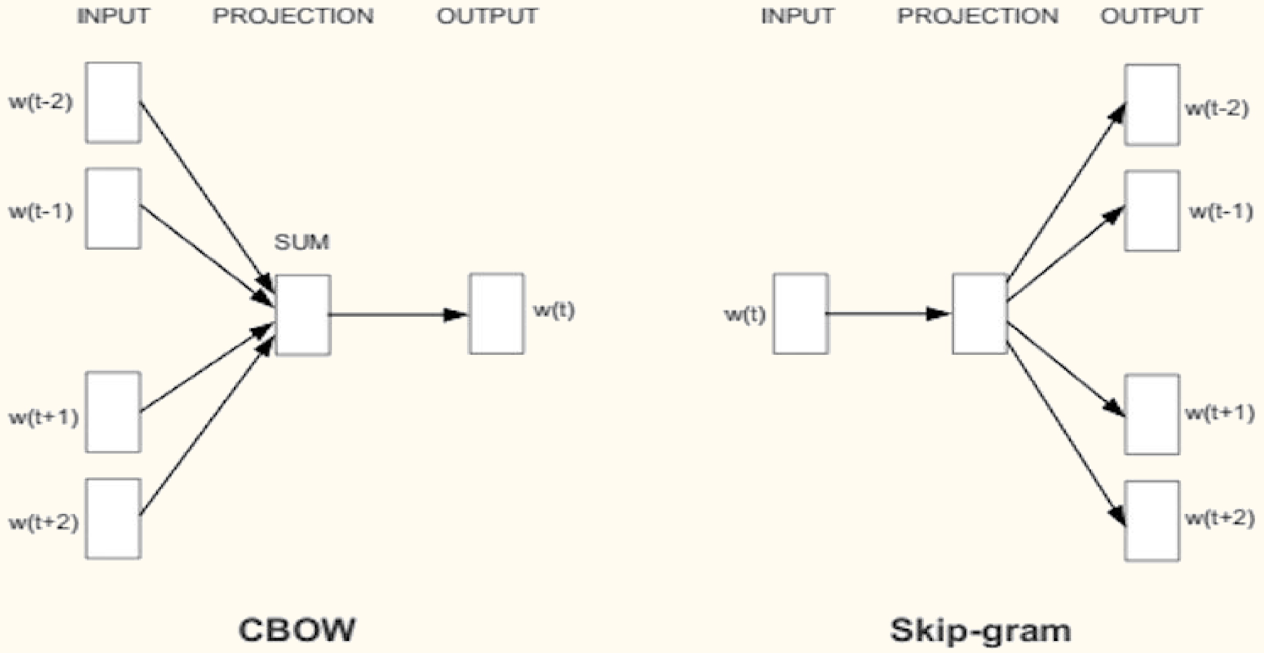
I use 10 cross validation, where train / valid set are splitted randomly.

SVD



[https://intoli.com/blog/pca-and-svd/]

Word Embeddings

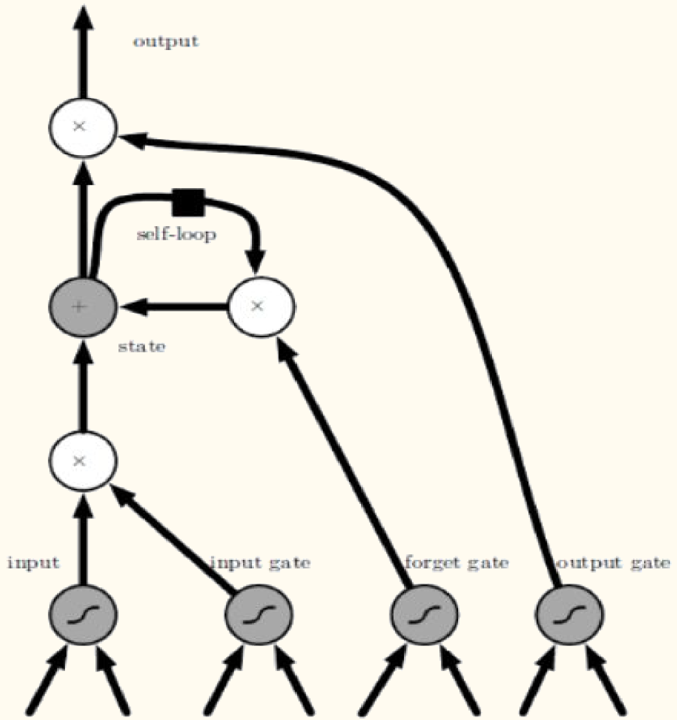


[https://deeplearning4j.org/word2vec.html]

My Solution

- Primer on LSTM architectures.
  - Bidirectional lstm.
  - Batch normalization
- Intuition / guide for building reasonable neural architecture.
- Tricks to avoid overfitting.
- Optimizing loss function using AUC.
- Cross validation of time series data.
- Discuss failures in my traditional ML models.
  - Try same model with different custom loss functions

LSTM

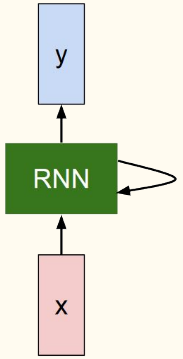


- LSTM is RNN trained using BPTT (solves vanishing gradient problem).
- Capture long term temporal dependency.
- A memory blocks used in place of neuron.

There are three types of gates in a memory block:

  - Forget Gate: decides information to be removed from the unit.
  - Input Gate: decides information to update memory state.
  - Output Gate: decides information to output based on input and the memory state.

Goodfellow, et.al. Deep Learning, 2016

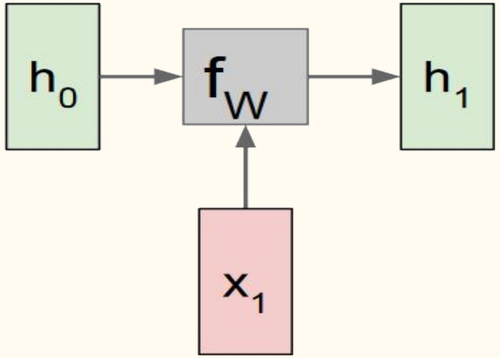


$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

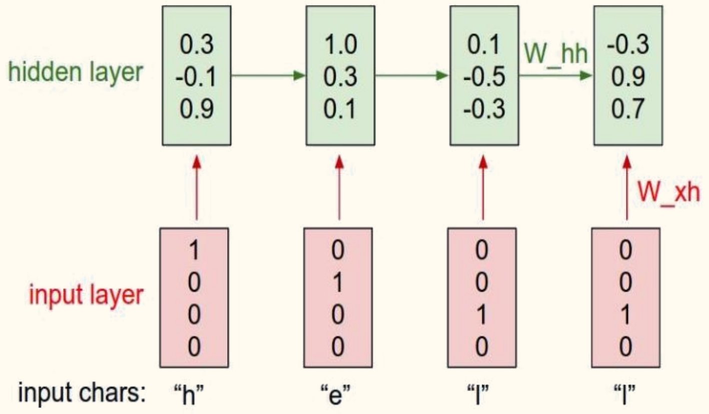
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



[Fei-Fei et. al., 2017]

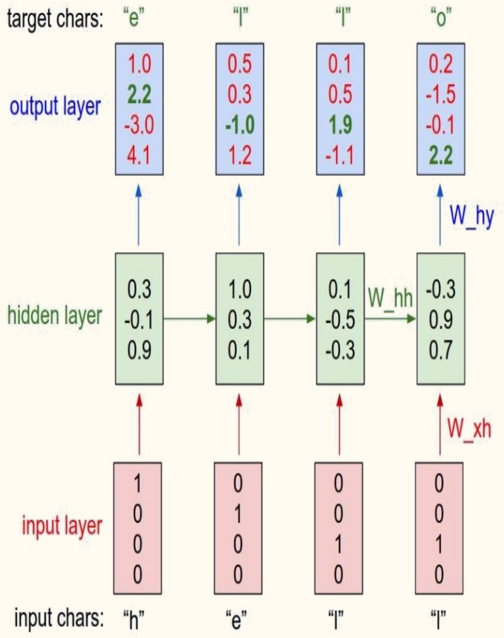
$$\boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$



Example: Character-level Language Model

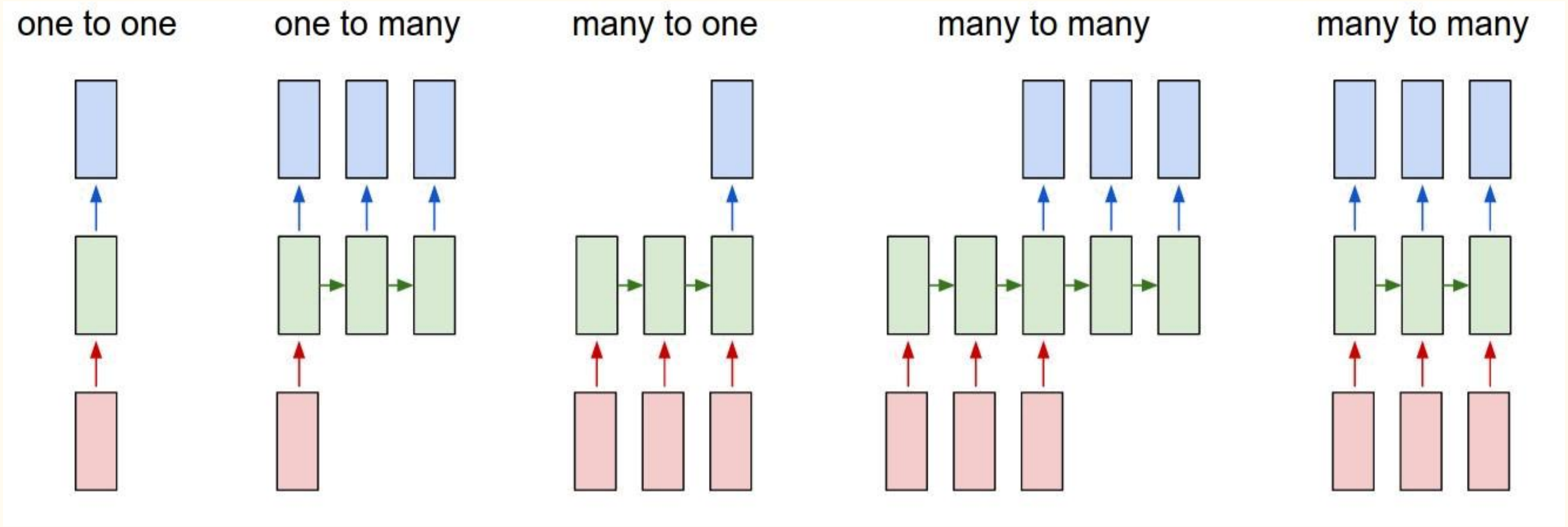
Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**



[Fei-Fei et. al., 2017]

LSTM Architecture



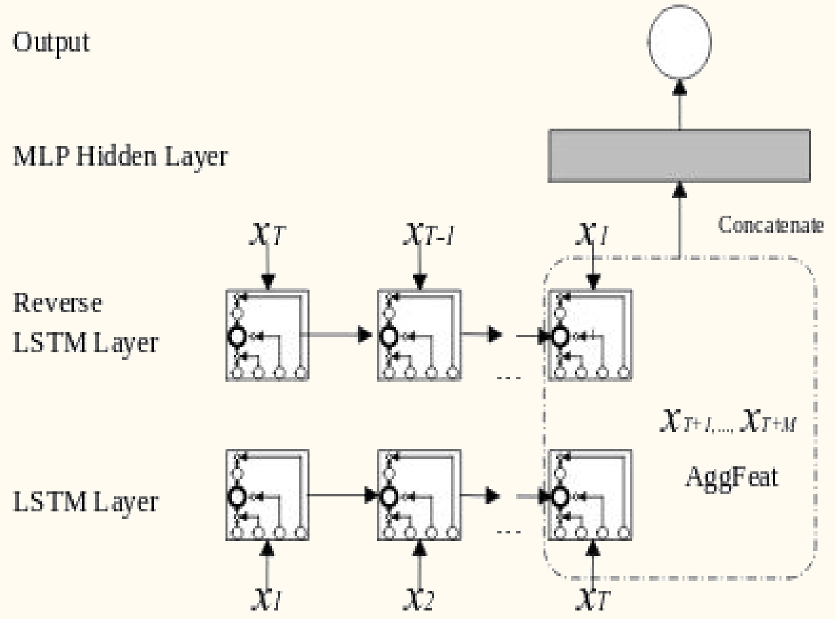
See tutorial on subject [https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/]

[Google images]

Many-to-many model (boundary detection problem)

[Fei-Fei et. al., 2017]

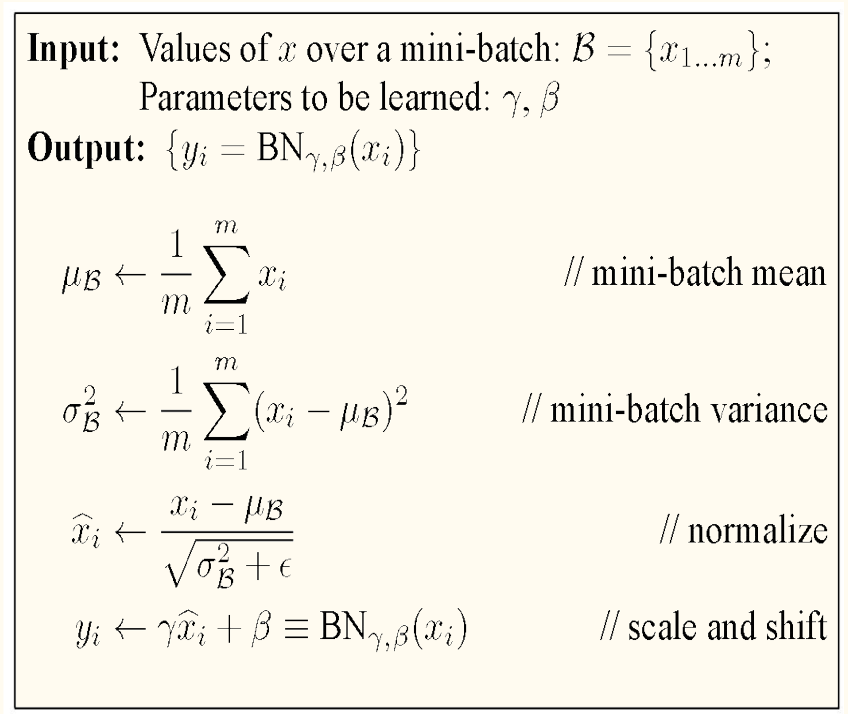
Bidirectional LSTM



[Yu, Zhou et al.]

- Bidirectional LSTMs can be trained using past and future data of a time frame.
- Bidirectional LSTMs can improve performance on a number of sequence classification.
- Bidirectional LSTMs is two lstms where the 1st lstm accept input as-is, and 2nd accept the input in reverse order.

Batch Normalization



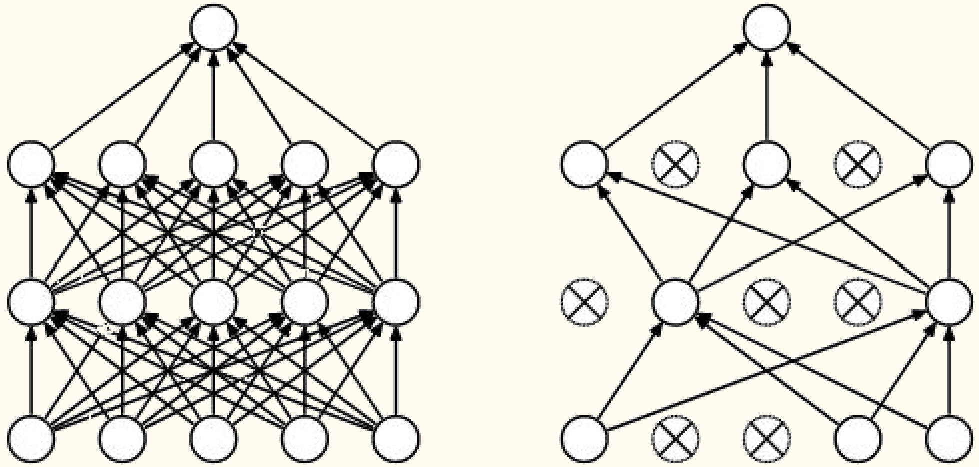
```
**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
           Parameters to be learned: $\gamma, \beta$
**Output:** $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$    // mini-batch mean
    $\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2$    // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$    // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

- Normalization in neural architecture.
- Reduce **covariance shift** thereby improving training speed.
- Control gradient flow in the network thereby minimizing saturation in non-relu based activation.
- Fix distribution between training and testing set thereby enhancing generalization.
- Implicit regularization.

[Ioffe et. al 2015]

Overfitting

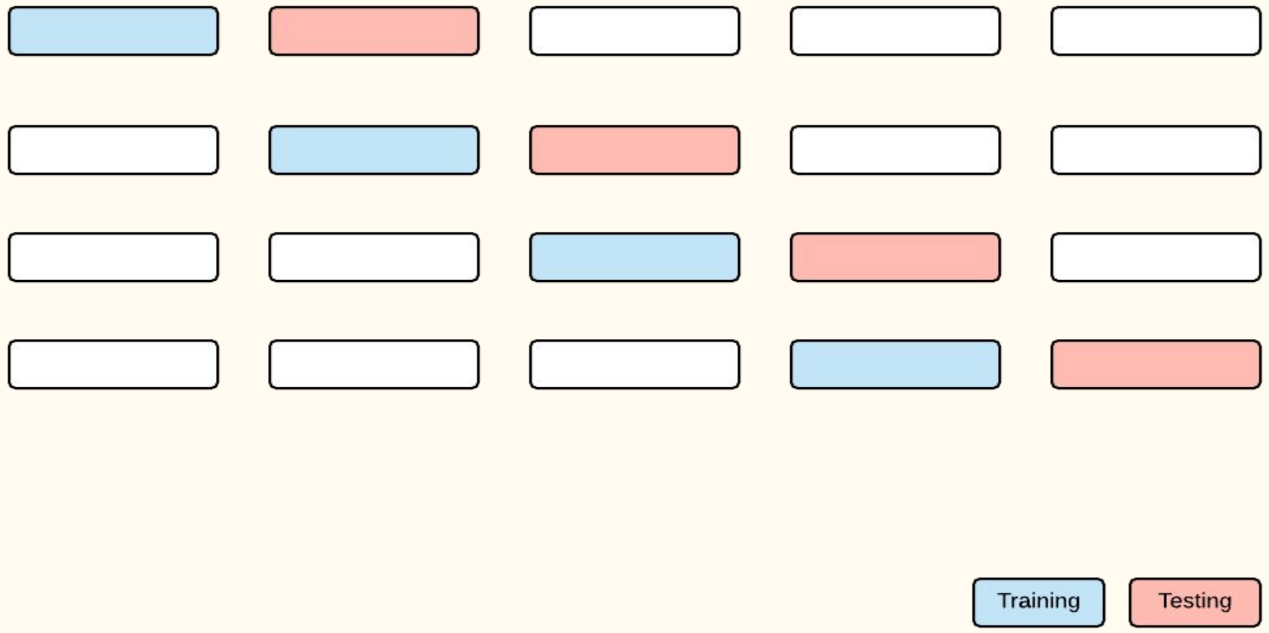
- Implicit Regularization
  - Batch normalization
  - Early stopping [https://en.wikipedia.org/wiki/Early_stopping]
- Dropout



(a) Standard Neural Net (b) After applying dropout.

[Srivastava et. al., 2014]

Cross-Validation of Time-Series



- Don’t trust the leaderboard.
- Trust local CV for model comparison.

My Solution: Code

Private board: 0.59

```
def model_relu6():
            model = Sequential()
            model.add( Embedding(inputDim*inputDim*inputDim, inputDim, dropout=0.2) )
#input vector dimension
            model.add(Convolution1D(nb_filter= nb_filter, filter_length= filter_length,
border_mode='valid', activation='relu',subsample_length=1))
            model.add(MaxPooling1D(pool_length= pool_length))

            model.add( Bidirectional( LSTM(1024, return_sequences=True) ))
            model.add(LeakyReLU())
            model.add(BatchNormalization( ))
            model.add(Dropout(0.5))

            model.add( Bidirectional( LSTM(2048, return_sequences=True)) )
            model.add(Activation('relu'))
            model.add(BatchNormalization( ))
            model.add(Dropout(0.5))

            model.add( Bidirectional( LSTM(512, return_sequences=True)) )
            model.add(Activation('relu'))
            model.add(BatchNormalization( ))
            model.add(Dropout(0.5))

            model.add( Bidirectional( LSTM(256)) )
            model.add(Activation('relu'))
            model.add(BatchNormalization( ))
            model.add(Dropout(0.5))

            model.add(Dense(1, activation='sigmoid'))
            model.compile(loss='binary_crossentropy',
optimizer='adam',metrics=[jacek_auc,discussion41015_auc])

            return model
```

```
#full code here ( https://www.kaggle.com/rspadim/gini-keras-callback-earlystopping-validation )

# FROM https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/41108

def jacek_auc(y_true, y_pred):
  score, up_opt = tf.metrics.auc(y_true, y_pred)
  #score, up_opt = tf.contrib.metrics.streaming_auc(y_pred, y_true)
  K.get_session().run(tf.local_variables_initializer())
  with tf.control_dependencies([up_opt]):
            score = tf.identity(score)
  return score

# FROM https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/41015

# AUC for a binary classifier
def discussion41015_auc(y_true, y_pred):
            ptas = tf.stack([binary_PTA(y_true,y_pred,k) for k in np.linspace(0, 1, 1000)],axis=0)
            pfas = tf.stack([binary_PFA(y_true,y_pred,k) for k in np.linspace(0, 1, 1000)],axis=0)
            pfas = tf.concat([tf.ones((1,)) ,pfas],axis=0)
            binSizes = -(pfas[1:]-pfas[:-1])
            s = ptas*binSizes
            return K.sum(s, axis=0)

# PFA, prob false alert for binary classifier
def binary_PFA(y_true, y_pred, threshold=K.variable(value=0.5)):
            y_pred = K.cast(y_pred >= threshold, 'float32')
            # N = total number of negative labels
            N = K.sum(1 - y_true)
            # FP = total number of false alerts, alerts from the negative class labels
            FP = K.sum(y_pred - y_pred * y_true)
            return FP/N

# P_TA prob true alerts for binary classifier
def binary_PTA(y_true, y_pred, threshold=K.variable(value=0.5)):
            y_pred = K.cast(y_pred >= threshold, 'float32')
            # P = total number of positive labels
            P = K.sum(y_true)
            # TP = total number of correct alerts, alerts from the positive class labels
            TP = K.sum(y_pred * y_true)
            return TP/P
```

Code: https://github.com/kenluck2001/KaggleKenneth/tree/master/wsdm_Kaggle

Alternatives

- Trees (catboost, lightboost)

Other that could work with careful effort

- HMM
- MARS
- ANFIS
- Logistic regression
- Naive Bayes

Conclusions

- Building models versus building infrastructure trade off.
- Debugging a failing model versus trying out a new model.
- Feature engineering versus better algorithm.
- Neural depth versus training size.
- "Ensembling work when the conditions are right".
    - Provide empirical justification.
- The place of **magic numbers**.
- "**Leakages** may not actually be a bad thing".

Thanks for listening

@kenluck2001

https://www.linkedin.com/in/kenluck2001/

kenneth.odoh@gmail.com

References

Yu, Zhou et al. “Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech.” ASRU (2015).

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, in PMLR 37:448-456 (2015).

Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag New York, Inc., New York, NY, USA.

Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.

Jeru666, Visualization script, https://www.kaggle.com/jeru666/data-exploration-and-analysis .

Davis, J. & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. ICML '06: Proceedings of the 23rd international conference on Machine learning 233--240, New York, NY, USA

Fei-Fei Li & Justin Johnson & Serena Yeung, (2017), http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting.. *Journal of Machine Learning Research*, 15, 1929-1958.