National Data Science Challenge 2019

ADVANCED CATEGORY

DoubleDs: CHONG YAN • KANG YU • JIA CHIN • MELISSA

Overview of Models Used

Neural Networks

- LSTM
- Attention
- Deep Convolutional Neural Network (CNN)
- Multichannel CNN

Word Embeddings

- Word2vec/spacy
- Keras Tokenizer

Support Vector Machine

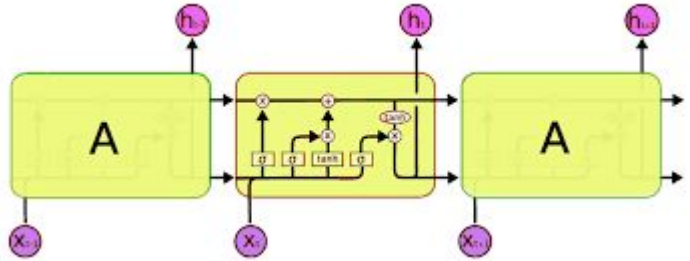
- Hyperplanes
- Parameter tuning and cost function

Limitations: Dirty data set and imbalanced dataset

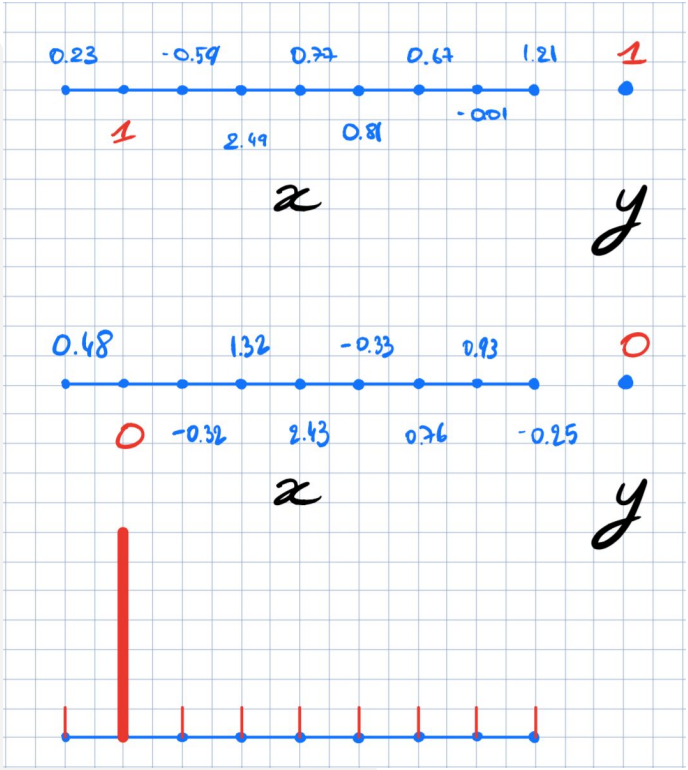
- Utilization on 2nd attribute and statistics to mitigate this problem

Neural Networks: LSTM

- Built around the idea that **sentences are a sequence of words**
- Words that are further back in the sequence can affect words further down the sequence
- Used a **bi-directional LSTM** so that model can **backpropagate and learn from its errors.**
- *Not very successful as words within the given data set are more of 'tokens' rather than 'embeddings' - they are 1 dimensional without additional meaning*



Neural Network: Attention



Similar to LSTM, it is **based around the idea that words further back in the sequence can determine the labels of the data point**.

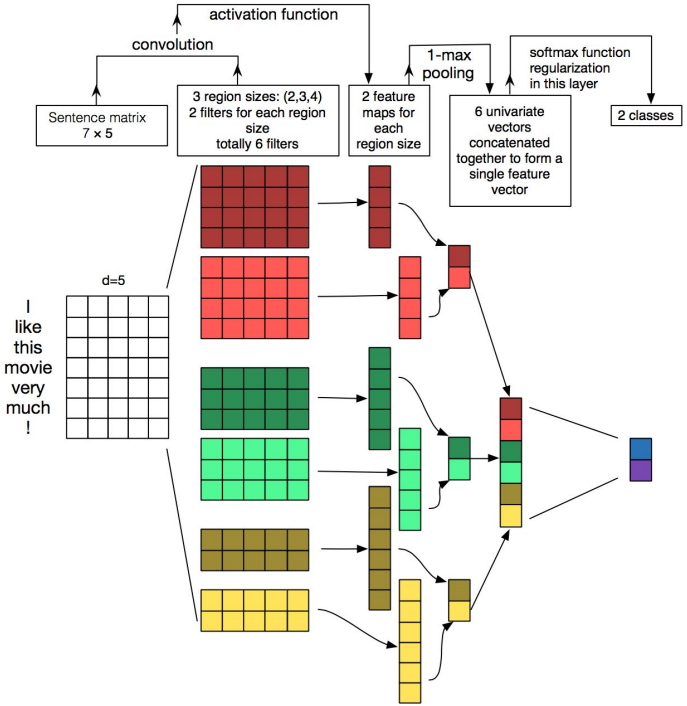
In this case, a word in the sequence essentially determines the label of the data point. Eg: samsung/apple for brand

Hence, we implemented and tried attention, paired together with a LSTM layer before.

Neural Network: Deep Convolutional Neural Network (CNN)

- Used a convolutional layer at the start (kernel size = 2) to extract word n-grams (iphone white, samsung 64gb etc) from the data
- Decided to utilize CNN to extract features from the text due to what we mentioned previously - that the words were more of tokens rather than embeddings.
- Passed the output tensors (weights of the word n-grams) through a dense layer so that model will get the best weights of the n-grams.

Multi-Channel CNN Graphic



Neural Network: Multichannel CNN

- We **fed the input into 2 separate channels** so that the model can **learn different length n-grams.**
- Afterwards, we **parse the results through a global max pooling** to get the most informative n-gram per filter.
- Next, we **concatenate the two outputs of the two separate channels together to get the combined tensors** (most descriptive n-grams) **then pass it into the dense layer** to obtain our output.

Word Embeddings: word2vec/spacy

Used **gensim word2vec** to generate a 300 dimensional word embedding for our models.

Also tried to use **spacy** (en_core_web_md) to get the word embeddings instead.

- Did not work well as **words were better thought of as tokens**. Moreover, **embeddings were not specialised to our dataset** and hence, some embeddings might not be accurate for our data.
- **Swapped over to letting the model dynamically learn word embeddings through convolution** and it performed better than using pre-trained word embeddings.

Word Embeddings: Keras Tokenizer

Defaulted instead to **keras tokenizer to get the 1D representation of the words** (their freq count). Used this and experimented with different 'learnt' words, eventually settling at **10000**.

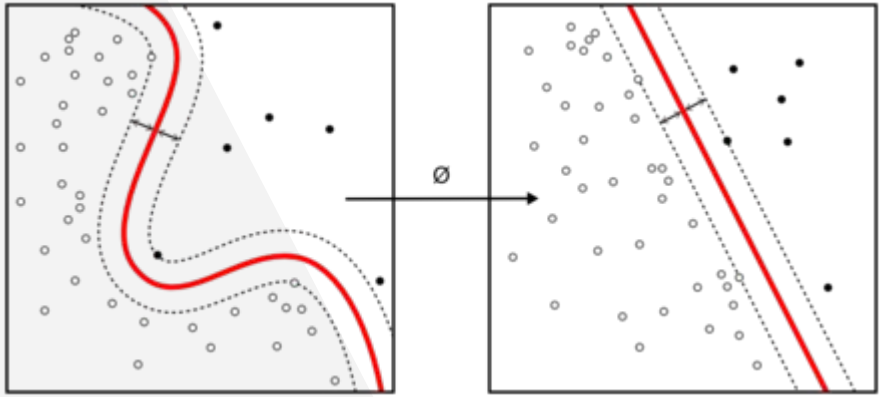
- 10,000 was chosen because the **maximum amount of words in each data set changes but in certain cases such as mobile, the data was not useful as it contained ~22,000 words**, many of which were not descriptive (malay words, a long integer without any meaning).
- Hence, 10,000 vocab size **allowed us to drop unimportant words and focus on the important ones** that determine label.

Creation of Artificial Dataset

- Tried to **address the problem of small (valid) training set** through creating an artificial data set.
- We **permute the words in each sentence and join them with the original data set** to obtain an artificial train set. Allows the model to train on a larger data set and **better its results on valid labels**.
- Could **better address the problem of validity** through supplying valid data points for which there are gaps in the data (eg: 0 data points for windows os for mobile; hence, model will never predict windows os under any circumstance)

Support Vector Machine

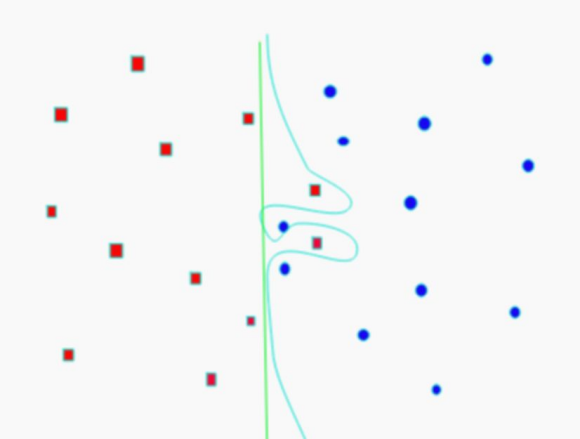
Supervised learning model well-known for its ability to classify items. It is important to note that it takes significantly less time to run the model

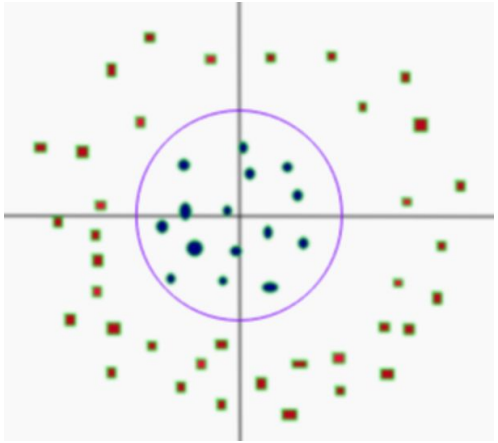


One of the most widely used clustering algorithms in industrial applications

Support Vector Machine

By constructing hyperplane in multiple dimensions, it separates the data into classes.





Support Vector Machine

Parameters taken into consideration:

Kernel
Making the hyperplane decision boundary between the classes

Gamma
How far the influence of a single training

C
Trades off correct classification of training examples against maximisation of the decision function's margin

Degree
Degree of the model

1st Attribute Prediction

We shall use multi-channel CNN as described before.

2nd Attribute Prediction

- We will leverage on using the support vector machine as a layer of filter.
- If the results differs between CNN and SVM, SVM results will be used in the 2nd attribute.i.e. CNN:2, SVM:3 (End results is “2 3”)

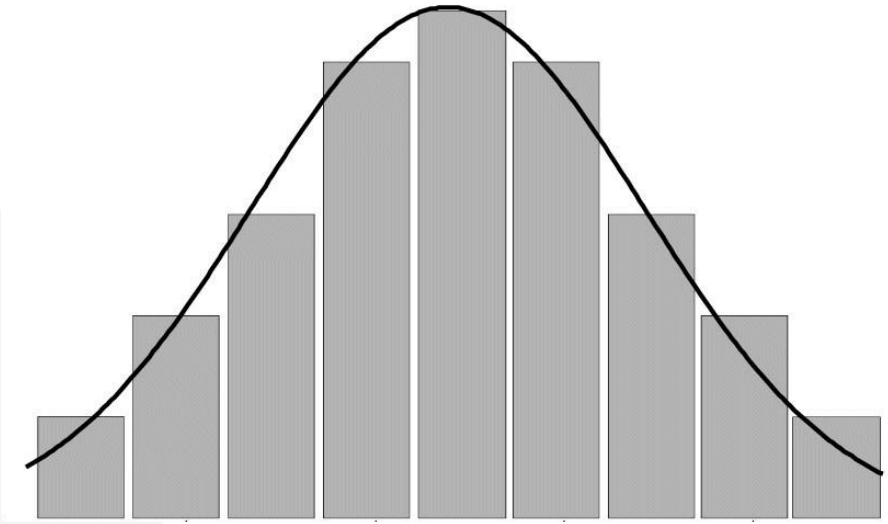
2nd Attribute Prediction

What happens when LSTM and SVM result is the same?

- Accuracy between CNN and SVM is relatively the same (SVM is lower)
- Main reason is because of the **imbalance of dataset** and **both model faces this issue**
- **Little sense to SMOTE** since some dataset are wrongly classified and it will amplify the misclassifications errors

2nd Attribute Prediction

Normal distribution of categorical data since dataset are in such large numbers



2nd Attribute Prediction

Normal distribution of categorical data

Benefits	
1.0	37955
3.0	28419
6.0	26822
4.0	12577
5.0	4930
2.0	2841
0.0	12

Skin_type	
5.0	13061
0.0	12025
6.0	9830
7.0	9077
4.0	5379
1.0	4747
3.0	3187
2.0	1104

Pattern	
6.0	50506
14.0	36178
2.0	16500
12.0	14145
18.0	12125
15.0	11880
5.0	7830

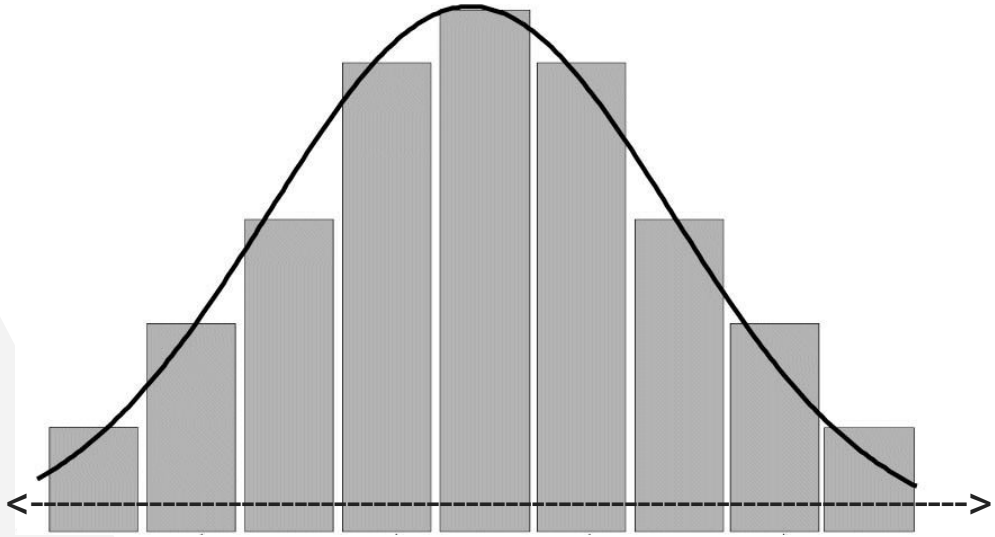
Clothing Material	
3.0	45039
18.0	38540
7.0	28822
4.0	28800
5.0	5899
16.0	5467
2.0	5052
9.0	4612

2nd Attribute Prediction

Since both model are facing the issue of imbalanced dataset, **predictors with the higher occurrences in training dataset are typically predicted correctly** while predictors with lesser occurrences are wrongly predicted.

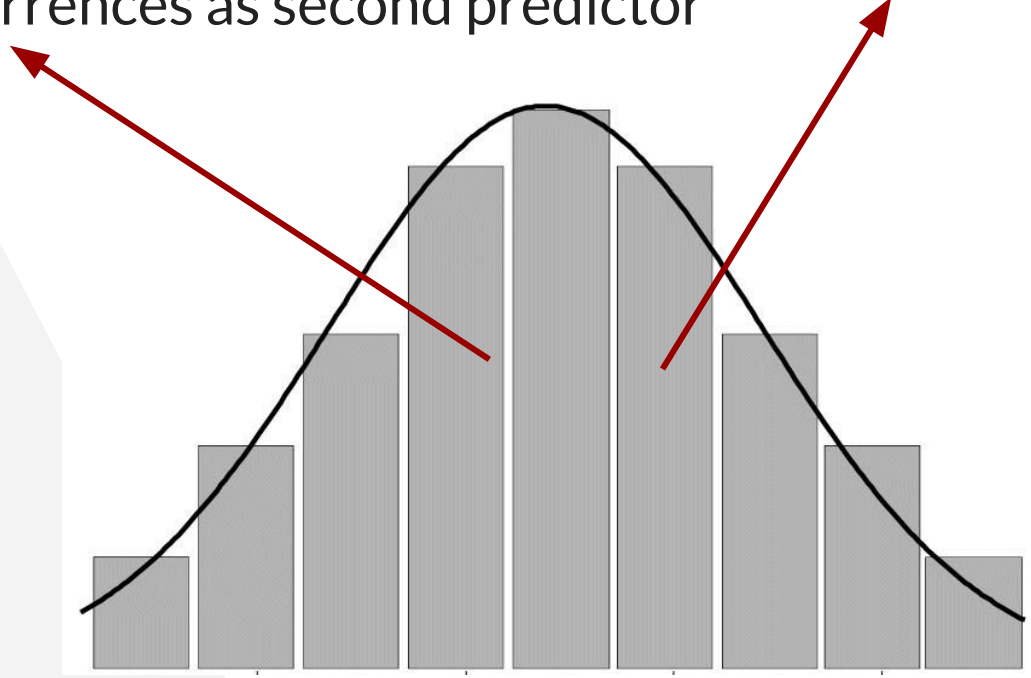
2nd Attribute Prediction

Performance of both models



2nd Attribute Prediction

Leverage on items with the 2nd and 3rd highest occurrences as second predictor



2nd Attribute Prediction

- In the event CNN and SVM have the same results, items with the 2nd or 3rd highest occurance will be used as the 2nd predictor

  ie. SVM & CNN: 2, 2nd: 4, 3rd: 5
  result : “2 4”

  SVM & CNN: 2, 2nd: 2, 3rd: 5
  result : “2 5”

Thank You!