Mercari Price Suggestion Challenge
4th Place Solution

Chenglong Chen
c.chenglong@gmail.com
2018.05

Outline

- Background
- Preprocessing & Feature Engineering
- Model
- Ensemble
- Summary

Background

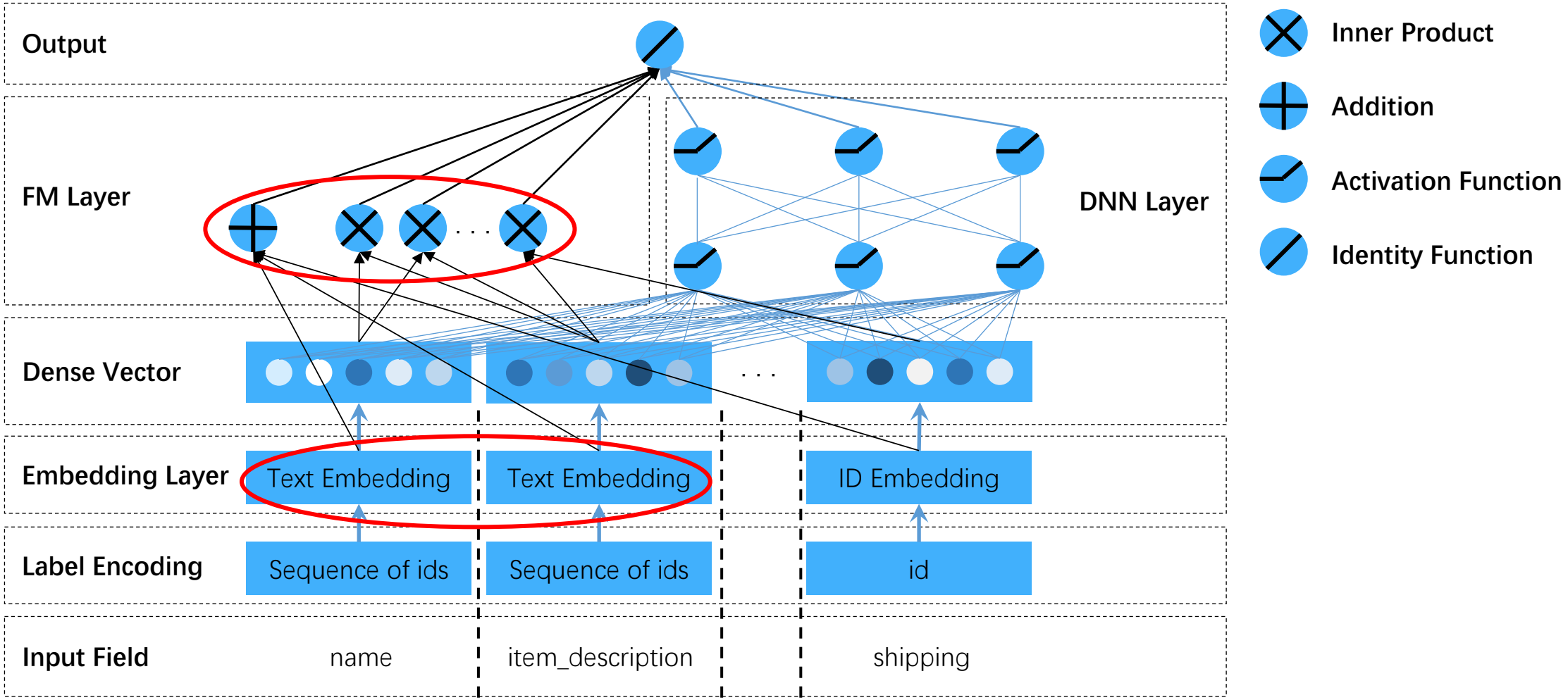
- Ph.D of Sun Yat-sen University, China
  - developed algorithms to detect image forgery
- NLP Algorithm Engineer at Alibaba
  - ALIMe-Knowledge Cloud team
  - working on knowledge graph, KBQA, ChatBot
- Story with Kaggle
  - turned data lover when I meet Kaggle in 2013
  - won two search relevance competitions
    - 1st Place in CrowdFlower, 2015
    - 3rd Place in HomeDepot, 2016

Preprocessing & Feature Engineering

Input type	Input name	Description	Example	Preprocessing	Output
textual	name	the title of the listing	Nike men's dri-fit sleeveless shirt tee	Tokenizing Label Encoding Padding Truncating	Sequence of ids
	item_description	the full description of the item	This is a men's Nike dri-fit shirt which is blue. All items come from a clean smoke and pet free home.	Tokenizing Label Encoding Padding Truncating	Sequence of ids
categorical	brand_name	brand of the listing	Nike	Label Encoding	id
	category_name	category of the listing	Men/Tops/T-shirts	Splitting Label Encoding	id
	item_condition_id	the condition of the items provided by the seller	3		id
	shipping	1 if shipping fee is paid by seller and 0 by buyer	0		id

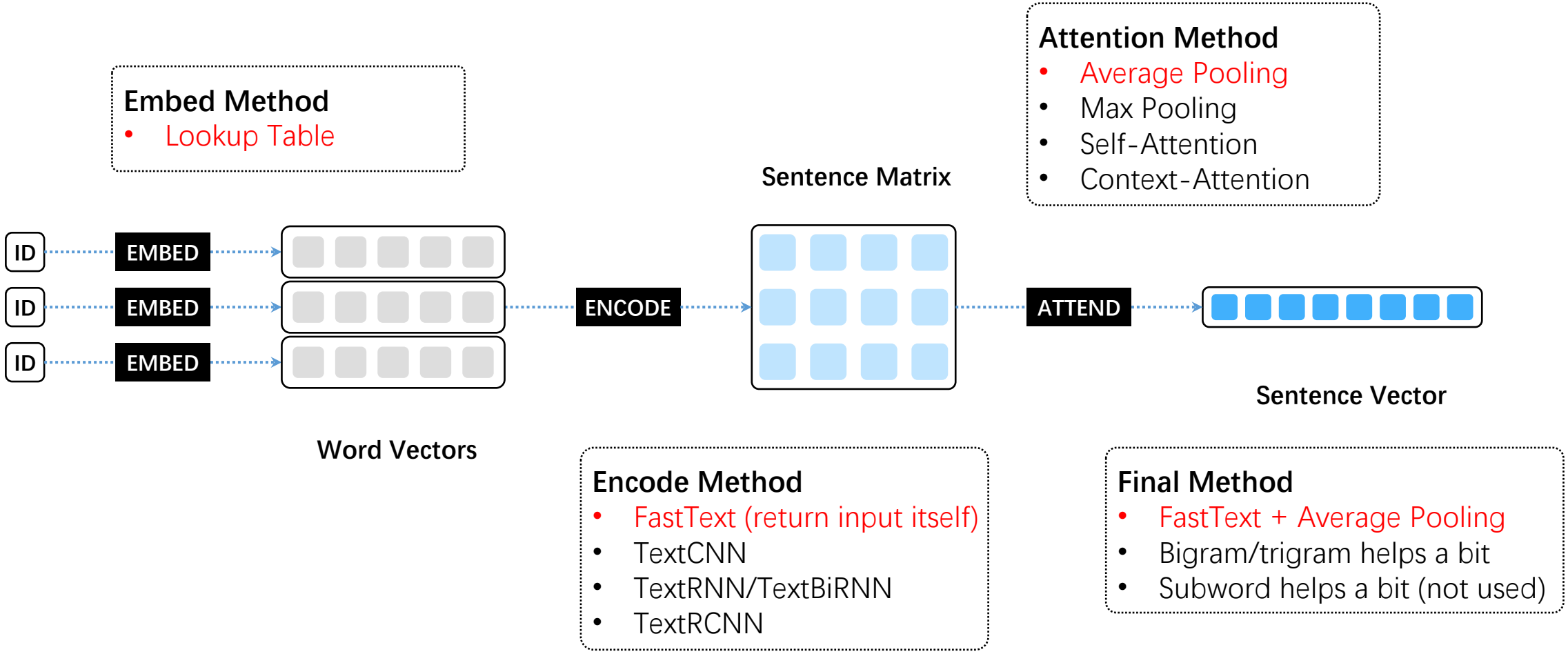
Ref: https://www.kaggle.com/c/mercari-price-suggestion-challenge/data

Network Architecture: DeepFM Variant



Ref: Deepfm: A factorization-machine based neural network for CTR prediction

Text Embedding Layer



Ref: https://explosion.ai/blog/deep-learning-formula-nlp

FM Layer

- Model the interactions between different feature fields
  - suitable for sparse id features
  - widely used in CTR prediction
- Efficient implementation
- Sentence level & word Level

```
# word level fm
fm_list = [
    sentence_matrix_of_name,
    embedding_of_brand_name,
    embedding_of_category_name,
    embedding_of_item_condition,
    embedding_of_shipping,
]
```

```
# efficient implementation to compute second order term
fm_list = [
    sentence_vector_of_name,
    sentence_vector_of_item_description,
    embedding_of_brand_name,
    embedding_of_category_name,
    embedding_of_item_condition,
    embedding_of_shipping,
]
fm_concat = tf.concat(fm_list, axis=1)
# t1 = (\sum_{i=1}^{6} v_i)^2
fm_sum_squared = tf.square(tf.reduce_sum(fm_concat, axis=1))
# t2 = \sum_{i=1}^{6} v_i*v_i
fm_squared_sum = tf.reduce_sum(tf.square(fm_concat), axis=1)
# t = 0.5 * (t1 - t2) = v_1*v_2 + v_1*v_3 + ... + v_5*v_6
fm_second_order = 0.5 * (fm_sum_squared - fm_squared_sum)
```

Ref: Factorization machines with libfm

DNN Layer

- Used Pure MLP
  - efficient
  - accurate
- Tried ResNet and variants
  - ResNet
  - DenseNet

Training Method

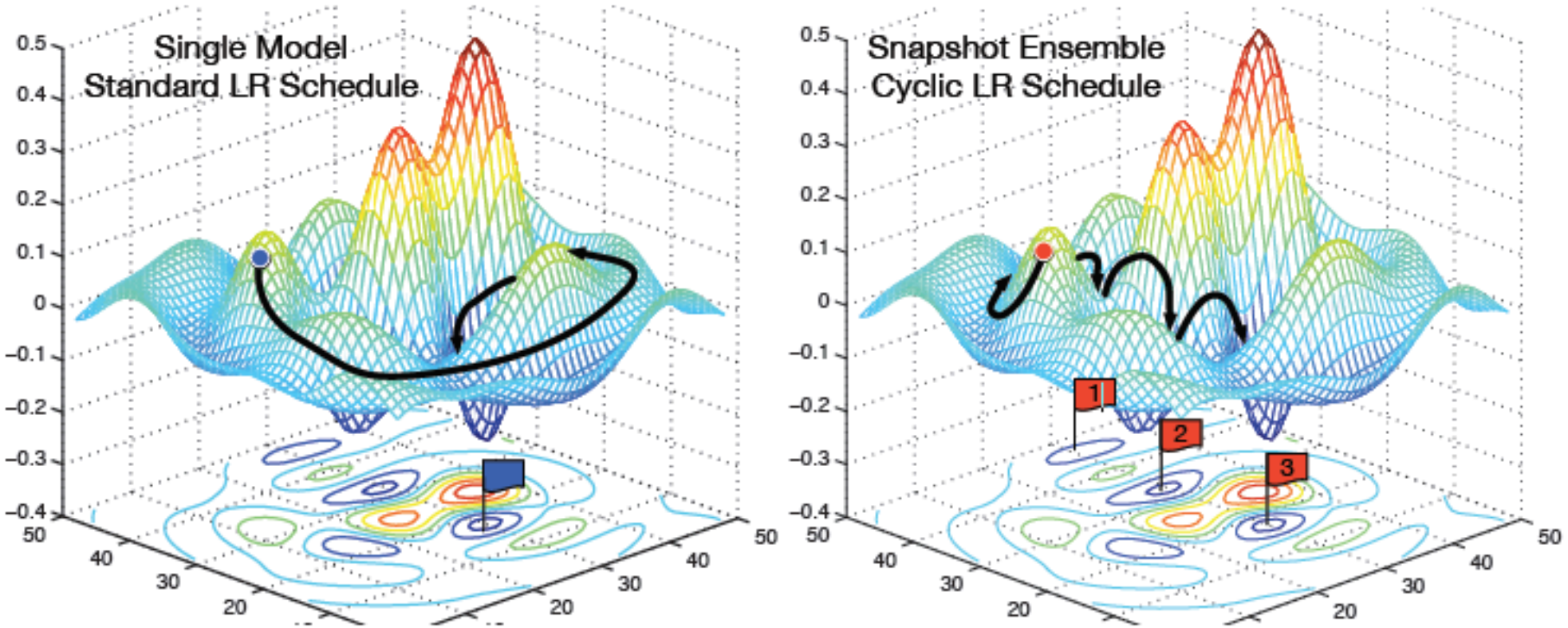
- Huber Loss on standard scaled log(price)
  - slightly better than RMSE
  - robust to outlier, e.g., very large log(price)

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

- Lazy Nadam optimizer
  - slightly better than other optimizers tested (e.g., Adam, RMSProp)
  - lazy update is efficient for sparse input (e.g., large embedding matrix)
- Learning rate schedule
  - lr restart to work with snapshot ensemble

Ref: Incorporating Nesterov Momentum into Adam

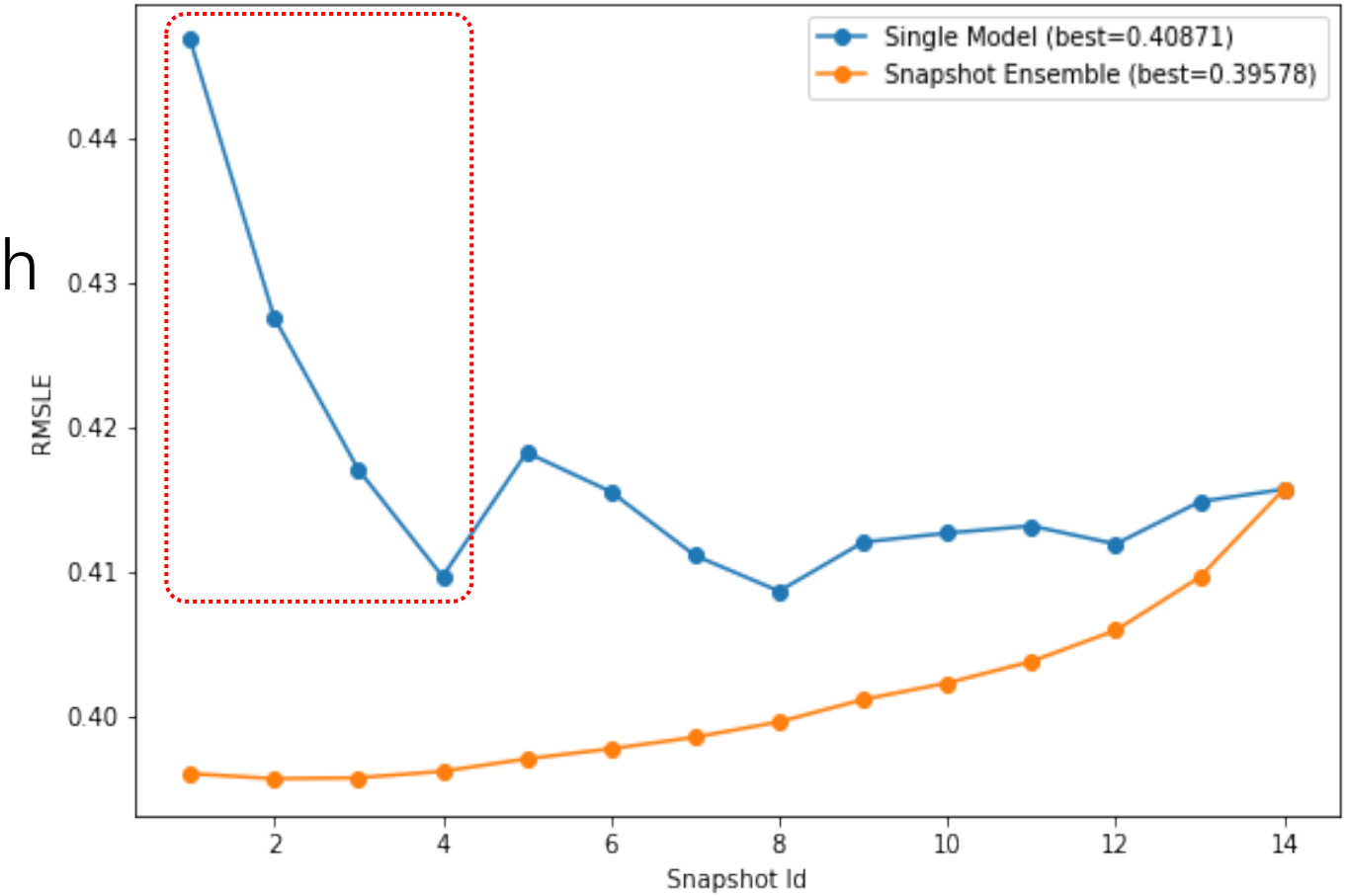
Snapshot Ensemble



Ref: Snapshot Ensembles: Train 1, get M for free

Snapshot Ensemble

- 4 snapshots each epoch
- decays normally 1st epoch
- restart enables from 2nd epoch
- average the last n snapshots



Ref: Sgdr: Stochastic gradient descent with restarts

Efficiency

- Model
  - FastText + Average Pooling
  - Snapshot Ensemble + LR Restarts
- TensorFlow
  - Tune the parallelism of threads
    - config.intra_op_parallelism_threads = 4
    - config.inter_op_parallelism_threads = 4
  - Use optimizers supporting lazy update, e.g., lazynadam or lazyadam
- Python
  - Use bind method outside of loop to reduce overhead
    - lst_append = lst.append, for i in range(1000): lst_append(i)

Ref: https://github.com/scikit-learn/scikit-learn/blob/a24c8b46/sklearn/feature_extraction/text.py#L144

Summary

- Preprocessing & Feature Engineering
  - very minimum preprocessing with focus on end-to-end learning
- Model
  - textual input: embed -> encode -> attend
  - categorical input: embed
  - interactions: FM (factorization machine) layer & DNN layer
- Ensemble
  - snapshot ensemble of NNs of the same architecture
- Code
  - https://github.com/ChenglongChen/tensorflow-XNN

Reference

- Matthew Honnibal, Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models. https://explosion.ai/blog/deep-learning-formula-nlp
- Guo, Huifeng, et al. Deepfm: A factorization-machine based neural network for CTR prediction
- Rendle, Steffen. Factorization machines with libfm
- Timothy Dozat. Incorporating Nesterov Momentum into Adam
- Gao Huang, et al. Snapshot Ensembles: Train 1, get M for free
- Ilya Loshchilov, Frank Hutter. Sgdr: Stochastic gradient descent with restarts

Thank You!

Mercari Price Suggestion Challenge
4th Place Solution

Chenglong Chen
c.chenglong@gmail.com
2018.05