Practical Machine Learning Documentation

Iordachescu Anca Mihaela, 407

The assigned regression task was estimating the probability of a word being complex, given the training (14002 examples) and test (1764 examples) sets.

The two machine learning models that I have chosen to implement for this assignment are NuSVR and Neural Networks. Each of them are presented in the next paragraphs having regard to the next points:

- Feature Engineering
- Description of Model
- Hyperparameter Tuning and Cross Validation
- Observations

1. *First approach - NuSVR*

i) Feature Engineering

The features for this machine learning model were extracted only from the target word (the word that the model has to estimate the complex probability for). I selected and concatenated the next features:

- **Number of vowels** of the target word - word tends to be less complex if there are more vowels;
- **Number of consonants** of the target word - word tends to be more complex if there are more consonants;
- **Number of characters** (it does not matter if they are alpha, numbers or punctuation) of the target word – word tends to be more complex if there are more characters;
- **Number of words** of the target word (the target word could be multi-word expression) - word tends to be more complex if there are more than a word;
- **Number of uppercase letters** of the target word - word tends to be less complex if there are at least one uppercase (this can indicate that the word might be a proper name);

- **Ngrams** of the target word with $n \in \{2,3,4\}$ - the motivation of using this feature is that there are groups of letters that are more or less common in English language;
- **The syllables** (or the pronunciation) of the target word - the motivation of using this feature is that there are some syllables that are more common in English language and can indicate a not-so complex word;

For the last two I applied the term frequency-inverse document frequency (TF-IDF) in order to get features(that are a large sparse matrix) that show how important the words are and reduce their dimensionality using the TruncatedSVD transformer.

ii) Description of Model

NuSVR is a model used for supervised learning tasks that consist in regression problems and is implemented using support vector machine technique. The main advantages of using support vector machine technique are that they perform very well in high dimensional space and, also, when the sample set size is smaller than the number of dimensions.

The parameters I have used for training the model are:

- **C** - the parameter that sets the misclassifying. A large value for C might make the model overfit while a small value for C might make the model underfit. Its default value is 1.
- **nu** - epsilon value of the SVR and, at the same time, bounds for two fractions: training errors(upper) and support vectors(lower). Its default value is 0.5 and the setted value should be in (0,1].

The kernel type used for this model is ‘rbf’, the default value for param kernel.

iii) Hyperparameter Tuning and Cross Validation

In order to find the best performance of the NuSVR model, I have tried different combinations of the next parameters:

- $C \in \{0.01, 0.1, 1\}$
- $nu \in \{0.25, 0.35, 0.5\}$

Parameters		Mean Absolute Error - MAE					
C	**nu**	**Fold 1 MAE**	**Fold 2 MAE**	**Fold 3 MAE**	**Fold 4 MAE**	**Fold 5 MAE**	**Cross-validation MAE**
0.01	0.25	0.124	0.121	0.118	0.124	0.122	0.122
0.01	0.35	0.108	0.103	0.102	0.102	0.105	0.104
0.01	0.5	0.091	0.087	0.085	0.085	0.088	0.087
0.1	0.25	0.121	0.117	0.115	0.116	0.118	0.118
0.1	0.35	0.101	0.096	0.095	0.096	0.097	0.097
0.1	0.5	0.088	0.083	0.082	0.083	0.085	0.084
1	0.25	0.115	0.112	0.109	0.111	0.112	0.112
1	0.35	0.094	0.089	0.088	0.090	0.091	0.090
1	0.5	0.084	0.080	0.078	0.080	0.082	0.081

iv) Observations

The best performance of the NuSVR model was obtained by setting the C=1 and nu=0.5. The submission was created by training the whole training set on the model with the parameters mentioned above and predict the probability for the test data.

2. *Second approach - Neural Network*

i) Feature Engineering

The features of this machine learning model are the ones that I also used for the first approach, detailed above, and plus:

- **Word embeddings for the target word** - if the target word is a multi-word expression, I add the word embeddings for every word and divide by number of words in the target word;
- **Word embeddings for the context** of the target word. The context is created by deleting the target word from the full sentence. If the context is a multi-word expression, I add the word embeddings for every word and divide by the number of words in the context.

These two features are important because it shows how different the target word is to the context. I have trained a word2vec from scratch to learn the word embeddings, using library gensim. The training data (param *sentences*) consisted in the Text8 corpora and the words extracted from the training data. The size of word embeddings (param *vector_size)* is 100.

ii) Description of Model

An artificial neural network is a model used for supervised learning tasks, and not only them, that consist in regression or classification problems and they are composed of layers.

In the architectures of neural networks that I have created, I used the next layers:

- ***Dense*** - output = activation(dot product(input of the tensor, kernel matrix) + bias)
- ***Dropout*** - helps the model from overfitting by dropping out a number of units in a NN, number calculated depending on *the param rate.*

The only activation function I used is **ReLu** - max(0, x) - all the negative values are “converted” to 0 while the positive values stay the same.
The only optimizer I used is **Adam** and I used values {0.0001, 0.00001} for learning rate.
I have used **ReduceLROnPlateau** in order to decrease the learning rate value when there is no improvement of the model.

Architecture used:

1) **Model M1**

Model M1 is composed of:

- Dense(200) with activation function - ReLu
- Dense(1) - layer that predict the probability value of a word being complex

2) **Model M2**

Model M2 is composed of:

- Dense(256) with activation function - ReLu
- Dropout with rate=0.2
- Dense(1) - layer that predict the probability value of a word being complex

3) Model M3

Model M3 is composed of:

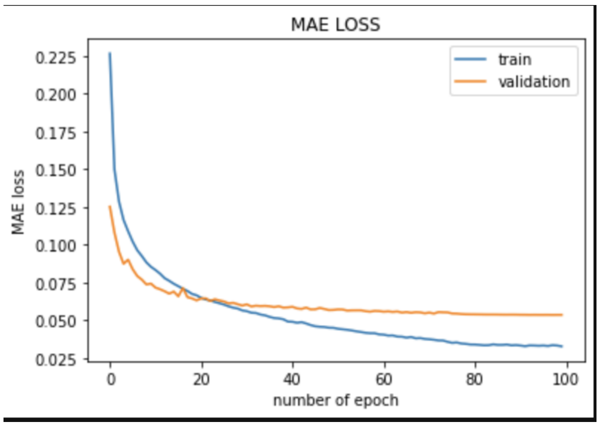
- Dense(256) with activation function - ReLu
- Dropout with rate=0.2
- Dense(200) with activation function - ReLu
- Dense(1) - layer that predict the probability value of a word being complex

iii) Hyperparameter Tuning and Cross Validation

Learning rate	ReduceLR	Model	Epoch	Fold 1 MAE	Fold 2 MAE	Fold 3 MAE	Fold 4 MAE	Fold 5 MAE	Cross-validation MAE
0.0001	Yes	M1	80	0.081	0.080	0.083	0.079	0.081	0.081
0.00001	Yes	M1	80	0.096	0.092	0.099	0.092	0.100	0.096
0.0001	Yes	M2	80	0.064	0.059	0.062	0.062	0.065	0.062
0.00001	No	M2	80	0.088	0.084	0.084	0.083	0.088	0.085
0.0001	Yes	M3	100	0.054	0.054	0.052	0.051	0.056	0.053
0.00001	Yes	M3	100	0.078	0.069	0.069	0.071	0.086	0.075

iv) Observations

The best performance of the neural network model was obtained by choosing model M3 and setting the learning rate for Adam optimizer=0.0001 and applied ReduceLR. In the Figure, the training data set was split in order to see the evolution of the loss. It seems that this model tends to overfit, more data could help improving.


MAE LOSS
train
validation
MAE loss
number of epoch