TDT4171 Methods in AI

Assignment 5

Written spring 2022 by Ø. Solbø

1 Machine learning using scikit

1.1 Parameters

The HashingVectorizer contains three important parameters, namely stopwords, binary and number of features. As it is assumed the entire data is written in a form of English, the English stopwords are set. The problem with this method however, is that this selection will split some of the stopwords into two words. For example, 'isn't' will be split into 'isn' and 't', causing the enumeration of 't' to be out of proportion. Thus, the naive usage of stopwords, may cause skew or bias in the data. Binary is set as boolean true, because the algorithm tries to detect the existence or nonexistence of words in a sentence. Number of features is left as a variable of experimentation, due to its importance on the performance. By increasing the number of features, the algorithm becomes better at separating the data, thus giving improved results. The downside is that the problem becomes computationally more difficult.

For the BernoulliNB and DecisionTree classes, most of the default arguments are used. The only exception, is the maximum depth for the decision tree. By leaving this as default, namely 'None', the tree could potentially become really long. It would be very computationally heavy to create and maintain, and is more likely to be overfitted. Depending on the circumstances, this is either fixed to 14, or being a variable parameter.

Based on the documentation, it was difficult to understand the optimality of some parameters. Having more experience would have simplified the process of choosing parameters, and may have slightly improved performance. However, the parameters are often a tradeoff between accuracy and computational demands. For example, choosing the decision tree's maximum features, might have improved the computational performance. By checking all features during training, one would expect an optimal splitting point for each split. The downside is the computational demands for each iteration, especially when number of features grows large.

1.2 Results

As mentioned previously, the number of features is a varying parameter. It is tested with the maximum tree depth set to 14, and the number of tested features ranging from 2^2 to 2^{20}. The results corresponding to the naive bayesian classifier are shown in figure 1, with accuracies given in table 1. As expected, increasing the number of features results in a general improvement in the algorithm's accuracy. This is linked to the algorithm becoming better to structure the observed data, and generalize it to new cases.

The naive bayesian performs better with number of features set as 2^2 compared to 2^6. The same observation, although not as dramatic, can be observed for the decision tree classifier's testing results in figure 2 and table 2. One could also observe that both models accuracies are identical for 2^2 features. No explanation to neither of these observations are found, however the former is theorized to be caused by the number of possible hashings. With 2^2 number of features, the hashing will hash the words into similar 'buckets'. With low distance between the words, the likelihood of incorrectly choose correctly is high. With an increasing number of features, fewer words will be hashed to the same 'bucket'. The resolution however, is not good enough until the number of features achieves roughly 2^{10}. At this point, the resolution is good enough that the classifiers manages to classify more 'correctly'.

By comparing the naive bayesian classifier in figure 1 with the decision tree classifier in figure 2, one can see that the naive bayesian classifier has little difference between the training and

testing results, while the decision tree does not. This may be caused by the assumptions of the different algorithms. As the naive bayesian assumes that each word in the sentence is independent of the other words, more words in the sentence may be taken into account during classification. The decision tree classifier may end up classifying based on a single word, potentially causing the observed difference.

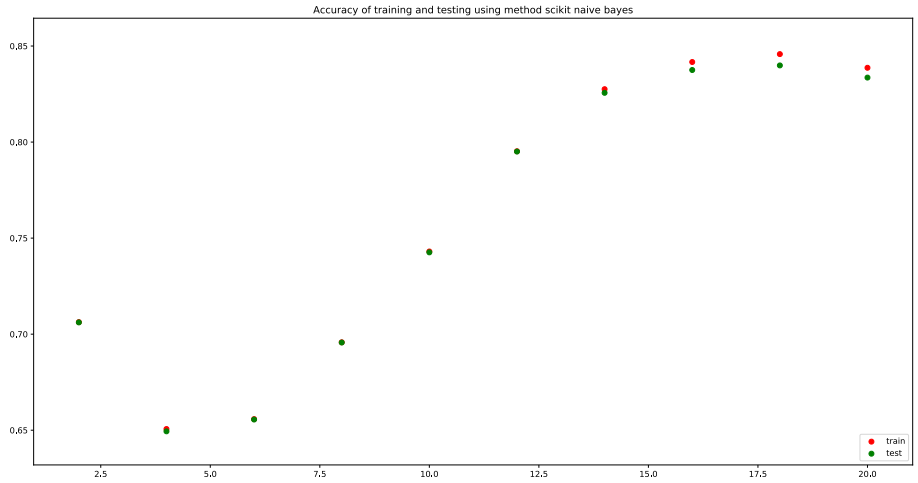

Figure 1: Training and testing results for naive bayesian classifier on the vertical axes, with varying $n_{features}$ on the horizontal axes.

$\mathbf{n_{features}}$	**Training accuracy**	**Test accuracy**
2^2	0.70626099	0.70612436
2^4	0.65061455	0.64941622
2^6	0.65576907	0.65556049
2^8	0.69577385	0.69565151
2^{10}	0.74303974	0.74258397
2^{12}	0.795221	0.79507079
2^{14}	0.82753725	0.8256466
2^{16}	0.84167784	0.83752145
2^{18}	0.84579688	0.83990408
2^{20}	0.83868588	0.83357594

Table 1: Training and testing accuracies corresponding to the naive bayes classifier in figure 1.

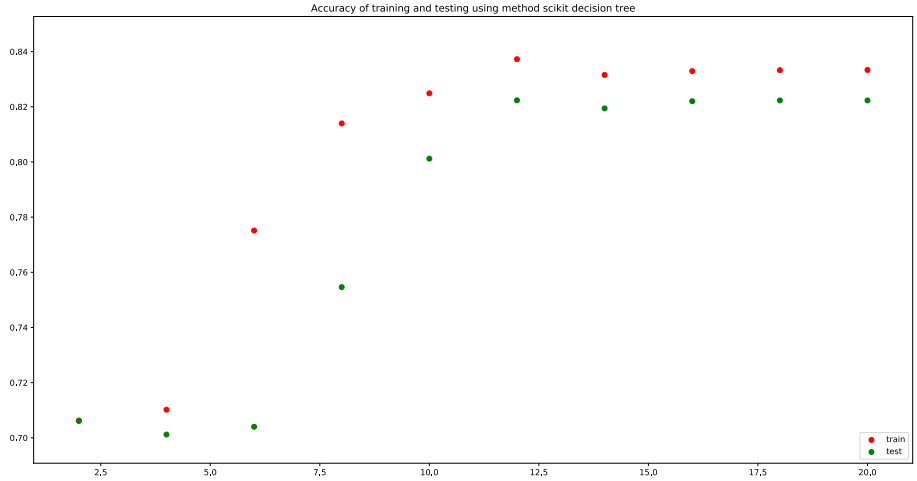

Figure 2: Training and testing results for naive bayesian classifier on the vertical axes, with varying $n_{features}$ on the horizontal axes. These values are obtained with the maximum tree depth set as 14.

$\mathbf{n_{features}}$	**Training accuracy**	**Test accuracy**
2^2	0.70626099	0.70612436
2^4	0.71018158	0.70118289
2^6	0.77514737	0.70400221
2^8	0.8139742	0.75462736
2^{10}	0.82488621	0.8012074
2^{12}	0.83725604	0.82234463
2^{14}	0.83156216	0.81941806
2^{16}	0.83293347	0.8220152
2^{18}	0.83328203	0.82231399
2^{20}	0.83335835	0.82230633

Table 2: Training and testing accuracies corresponding to the decision tree classifier in figure 2. These values are obtained with the maximum tree depth set as 14.

Based on figure 2, one could argue that the decision tree classifier appears to converge with an increasing number of features. This is theorized to be caused by the maximum depth of the decision tree. By limiting how deep the tree will become, it has to stop splitting at some point. Thus implying that an increase in number of features may not fully correspond to an improvement in the decision tree accuracy. To test this theory, number of features where set as 2^{14}, and the maximum depth was varied. The obtained accuracies are shown in table 3, where it is clear that an increase in depth results in improved accuracies. A deeper tree is therefore better able to capture the dynamics in the data, due to having more splits to fine-tune the attributes. However, using a deeper tree will increase the risk of overfitting, as clearly shown with **Max depth** as 'None'. This is caused by the tree to fully split all the training data, which will not necessarily match the testing data. Choosing an optimal depth is therefore a tradeoff between increasing accuracy, and preventing the classifier from overfitting. A maximum tree depth of around 14 seems to work well here.

One could critique the former paragraph on multiple points. First of, it only studies the effects on the accuracies, and does not take into account the computational power necessary. To get a proper understanding of the different values, the CPU-cycles for training or testing should be included.

In general, if the processing power is limited, choosing a lower tree depth may be necessary to have a maintainable tree with somewhat good accuracy. The second point, is that the analysis is only performed with number of features set as 2^{14}. The tendency to overfit could be linked to number of features, and a different chosen number of features could have a dramatically different performance.

Max depth	**Training accuracy**	**Test accuracy**
2	0.74480032	0.74266824
4	0.78385867	0.7814875
6	0.80833628	0.80514526
8	0.81082704	0.80673878
10	0.81434565	0.80806417
12	0.81808051	0.80915206
14	0.8315418	0.81918822
16	0.84396252	0.82762319
18	0.84890333	0.82799093
20	0.86396237	0.83746782
None	0.99997201	0.85355633

Table 3: Training and testing accuracies when testing the effect from the maximum depth of the tree. These values are obtained with the number of features set as 2^{14}.

Based on the performance of the classifiers, it appears that the optimum number of features is 2^{16}, with the optimum maximum tree depth being 14. Both classifiers are able to achieve good accuracies during training and testing, while not taking forever to train (looking at you **Max depth** as 'None'). A better performance might have been obtained using a different set of parameters, or modifying the stopwords to counteract incorrect splitting affecting the hashing. As currently implemented, the naive usage of stop words implies that the stopwords contain crucial information in the sentences, which is clearly incorrect.

2 Machine learning using keras

2.1 Parameters

When developing a deep learning model, there are multiple parameters to set. As it may be difficult to get a thorough understanding of the parameters effect, the following sections contain the results from much of the experimentation performed. By analysing the performance of the different parameters, and choosing the optimum parameters for each type, it was theorized that an optimal algorithm could be achieved by such iterative testing. This assumes however that the parameters are independent of each other, such that one could use the optimum of one parameter to choose the next optimum parameter. In reality, this will likely be a slightly incorrect assumption to make, however it should have little impact on the total performance of such a small system. To reduce the time complexity during experimentation, the testing is performed on a single epoch. There might occur a discrepancy where some other combination would have been slightly better if it was run for n epochs.

The performance from analyzing some of the optimizers, loss- and activation-functions are shown in tables 4 to 6. When choosing the optimum parameters, it is a choice between reducing the loss and achieving high enough accuracy for both training and testing. As the loss represents the error between the predicted state and the actual state, optimizing in terms of the loss, might give better performance over multiple training-epochs. It is also a tradeoff whether to choose between the training accuracy or the testing accuracy. Optimizing the training accuracy might result in overfitting, while optimizing the testing accuracy might result in underfitting. Based of the results, the optimal {optimizer, loss function, activation function} are chosen as {'adam', 'huber', 'tanh'}, since these minimize the loss and maximize the testing accuracy. With respect to the training

data, one could argue that 'sigmoid' performs slightly better than 'tanh'. It would however be slightly worse for the test-data, thus potentially inducing a slight overfitting in the long run. It is however assumed that the model will not be trained over sufficiently many epochs to achieve neither overfitting nor underfitting.

Optimizer	**Training loss**	**Training accuracy**	**Testing loss**	**Testing accuracy**
Adamax	0.2674	0.8872	0.2417	0.8984
Adadelta	0.6351	0.7002	0.5903	0.7061
Adam	0.2253	0.9081	0.2002	0.9210

Table 4: Measured performance for different optimizers.

Loss function	**Training loss**	**Training accuracy**	**Testing loss**	**Testing accuracy**
Cross entropy	-	0.7063	-	0.7061
Binary cross entropy	0.2253	0.9081	0.2002	0.9210
Binary focal cross entropy	0.0695	0.8955	0.0558	0.9118
MSE	0.0720	0.9034	0.0543	0.9280
Huber	0.0357	0.9044	0.0276	0.9248

Table 5: Measured performance for different loss functions. '-' means that the algorithm was unable to calculate a scalar loss.

Activation function	**Training loss**	**Training accuracy**	**Testing loss**	**Testing accuracy**
Softmax	0.2281	0.7063	0.1856	0.7061
Sigmoid	0.2253	0.9081	0.2002	0.9210
tanh	0.2470	0.8996	0.1899	0.9276
Linear	3.0896	0.7264	4.4815	0.7061
exp	0.4351	0.8107	0.2720	0.8914
Swish	10.8938	0.2937	10.8920	0.2939

Table 6: Measured performance for different activation functions.

The structure of the network may have a huge impact on the performance of the algorithm. This includes evaluating both the number and choice of layers, including the number of internal units per layer. During the development and testing, it was initially assumed to be using three different layers, namely embedded, LSTM and dense. The results from tuning the number of internal units, are given in table 7. This shows that the network performs 'best' with 64 embedded output units, 32 LSTM internal units, and a single dense unit. With these parameters, the tree has enough units to capture the information embedded into the data, and generalize it onto new unseen cases.

Embedded	**LSTM**	**Dense**	**Training loss**	**Training accuracy**	**Testing loss**	**Testing accuracy**
32	32	1	0.0357	0.9044	0.0276	0.9248
32	16	1	0.0362	0.9029	0.0271	0.9266
64	16	1	0.0523	0.8572	0.0319	0.9123
64	32	1	0.0346	0.9080	0.0256	0.9312
64	64	1	0.0473	0.8732	0.0307	0.9190
16	16	1	0.0373	0.8989	0.0289	0.9233

Table 7: Measured performance for different number of internal units, given three layers. Trained for 1 epoch, but does not take training time into consideration. The number of internal layers are always selected as a factor of 2, however better choices will exist.

The modelled network assumes that it is sufficient with three layers. One could however argue that

a more narrow, but deeper tree might perform better. Using a narrower network, it is hypothesized that each layer will capture a different semantic of the data, becoming more generalizable. The results are not displayed in a table, as previously, to save some space in the report. Testing upwards to 6 layers; the best result was obtained using a 6 layered network, which achieved training loss and accuracy of 0.0360 and 0.9039 respectively, and testing loss and accuracy of 0.0289 and 0.9227 respectively. This is slightly worse compared to the optimum three-layered tree, and one could therefore argue that increasing the number of layers would not necessarily improve performance. One should note that the tree was restructured by adding dense layers, and modifying the internal units of all layers for it to become more narrow. There might consist a different set of parameters, be it layer-types or layer-parameters, where the narrower network would outperform the tuned network.

2.2 Results

To evaluate the performance of the tree, and to better understand the effect from retraining, the optimal three-layered network was trained over 10 epochs. The final achieved results are shown in table 8. Both the training and the testing accuracies are improved dramatically. As both the training and the testing results contain sufficient losses, it does not appear that the model has been overfitted.

Training loss (last epoch)	**Training accuracy (last epoch)**	**Testing loss**	**Testing accuracy**
0.0139	0.9654	0.0214	0.9444

Table 8: Measured performance for the optimal tree when trained over 10 epochs.

3 Comparison of the algorithms

3.1 'Deep' learning compared to more traditional machine learning

Using traditional methods for machine learning, such as naive bayesian classifier and decision tree classifier, it was possible to accuracies upwards to 0.85. The deep learning network trained over 10 epochs, managed to achieve accuracies upwards to 0.96. This shows that deep learning outperforms classical machine learning in this case.

To understand the different results, one must take into account the problem formulation itself. The algorithm is tasked to classify the existence of some features in a text. The problem is that text is often not formally defined, and every person will write somewhat differently. The changes might occur due to usage of slang, dialects or other informal words, but it could also be caused by the different vocabularies that are used. This means that sentences written by different people, with different education and life situation, could have a huge variance, despite being written about the same subject. It will therefore be extremely difficult to train a traditional machine learning algorithm to properly classify this problem, as one would require huge amounts of training data. Despite vast training data, there might always be some combination of words that the algorithm has not been trained on, and which it will struggle to generalize.[1, pp. 876-877]

Traditional machine learning algorithms are very useful to exploit formalism and structure. A natural language has, as mentioned previously, a lack of structure which the algorithms could exploit. The assumptions which are used to built the algorithms, are therefore incorrect. For example, the naive bayesian classifier assumes that all words in a sentence are independent, which clearly is incorrect. Any change in words, be it placement or synonyms, could have a dramatic effect, causing the algorithm to lose generalization.[1, p. 879] Another example, would be the decision tree classifier, which would split according to the sentences it has trained on. Even though some sentences will appear similar, the sentences will not necessarily convey the same information. A naive split could therefore produce vastly incorrect results. In summary, due to the lack of formalism in a natural language, traditional machine learning algorithms are unable to properly generalize.

Deep learning on the other hand, does not make any crucial assumptions regarding structure in the input data. Using a LSTM RNN, the network is able to preserve information over multiple time steps, causing it to maintain an internal state.[1, pp. 824-826] This enables the neural network to infer features from multiple places in a sentence. It is therefore independent of the lack of formalism in a natural language, causing it to easier extract important information from the text, independent of their placement in the sentences. Thus, deep learning is generally unaffected by some strange words in the middle of the sentence, which could have a huge impact on more traditional machine learning.[1, p. 876]

Due to the sheer complexity of deep learning models, which contains many internal weights, the models may still be retrained over the same dataset. There will often, especially for rich problems such as text-classification, exist a way to fine tune the weights by retraining. As observed in section 2.2, the deep learning model improved by several percentage by running it over 10 epochs. The traditional machine learning algorithms, will often risk overfitting if retrained.

References

[1] S. Norvig, P. Russel, *Artificial Intelligence - A modern approach.* Pearson, 2022.