Stance Classification

Jonatan Hellgren
Author contact details
gusheljot@student.gu.se

Abstract

The recent covid-19 vaccines has been controversial, some people have an positive stance while others have a negative one. Many opinions have been posted online on web-pages such as youtube, facebook and twitter, and by looking at these opinions we can get a overall view on what the general opinion on the subject is. To classify all these opinions by hand isn't however scalable with the amounts of opinions posted. Here in this report we will attempt to automate this with machine learning. We show that a support vector machine classifier trained on 7516 samples of negative and positive opinions are able to achieve an accuracy of 0.88 on the test data of 392 samples. Where the data trained upon was gathered by student at Chalmers. We will also go through some examples where the model made a faulty prediction and the reasoning behind this.

Introduction

The support vector machine classifier was trained on data scraped from the web by students of the course Applied Machine Learning at Chalmers. The data contains opinions about the covid-19 vaccine and weather the stance of the given comment is positive or negative. The predicting features will thus be the words used in the texts and the predicting feature will be weather the opinion expressed where positive or negative. By only considering if the opinion is positive or negative the problem will be in the domain of binary classification.

The data has first been collected and classified by one student and later been reclassified by at least one other student. The data used for training will be the data where all students agree on the classification of the text will be used to train the model, this brought the training data from the original size of 8788 to 7516. This decision was made to ensure the quality of the data, since if the students have problem classifying it, it doesn't make sense to use as training data for our model.

We will try out some different models with tuned hyper-parameters on the data to see which model is able to capture the underlying patterns in the vector-space of this domain. The model we found to be most effective where a support vector machine classifier, and we will later take a closer look at the performance of this model on the test data containing of 392 samples.

Method

Data preprocessing

The original format of the data is plain text, this makes it such that we can not just import the data into our model without any prepossessing. To turn our data into features we will first use a count vectorizer to transform all the observations into a matrix of tokens, this makes the data interpretable for our model. However just performing this transformation won't be that useful since every word will have the same weight which may affect the performance of our classifier.

To solve this issue we will perform a term frequency-inverse document frequency (tf-idf) transformation on the matrix of tokens, this first calculates the term frequency then multiplies it with the inverse document frequency. The term frequency $tf(t, d)$ for a term t is equal the number of occurrences of the term in the sentence divided by the total number of words in that sentence. The idf is computed with the following formula:

$$idf = -\log \frac{n_t}{N},$$

where n_t is the number of sentances that the term t occurs in and N is the total numbers of sentances. The tf-idf transformation thus gives a higher weight to the more informative less frequent words and lower weight to the more frequent less informative words.

Model selection

The model selection part of this analysis was done by trying out three different candidate models, the models that we will try for this task where perceptron, logistic regression and support vector machine. First we will tune the hyper parameters of each model using a randomized search with cross-validation. When we have found the best performing hyper-parameters for each model we will compare the mean cross-validation score of the models using the training data, here will continue with the best performing model.

Cross-validation is performed by splitting the data into n (in this analysis $n = 5$) folds, then training the data on four of the folds and later evaluating it on the last fold, this is repeated until every fold has been us for evaluation. This gives us information about how the model will be able to perform on data not trained on.

Model evaluation

To evaluate the model we got to choose from the model selection part of the analysis we will try the model out on the not seem before test data, with this data we will construct a confusion matrix. A confuison matrix for a binary classification problem is a grid where the column represents the predicted class and the rows represent the actual classes, see table 1 for an abstract example. With this confusion matrix we will then take a look at the metric called accuracy, the accuray is computed witht the following formula:

$$\text{accuracy} = \frac{TN + TP}{TN + FN + FP + TP},$$

so it basically gives us the percentage of correctly classified observations.

	Predicted 0	Predicted 1
Actual 0	TN	FN
Actual 1	FP	TP

Table 1: A example of how a confusion matrix looks for a binary classification problem. Here the TN stands for true negative and it is where the number of correctly classified negatives will be written, likewise for the labels FN, FP and TP, which stands for false negative, false positive and true positive.

Support vector machine

A model we will take a closer look at is the support vector classifier is a machine learning model that fits weights to each feature and a intercept, to classify an observation the model takes the sum of the weights times the feature value, if that sum is negative the model will classify it the observation as negative (0) and vice versa.

$$\mathbf{w}^T\mathbf{x} + b < 0 \implies \text{negative}$$
$$\mathbf{w}^T\mathbf{x} + b \geq 0 \implies \text{positive},$$

where the column vectors $\mathbf{w}$ and $\mathbf{x}$ are the weighs and the feature values respectively and b is the intercept of the model.

A support vector machine can be fitted on different kernels, the one described above is using the linear kernel since that is what will be further used in this analysis. The other possible kernels we will try out are rbf and polynomial.

To fit a support vector machine with soft-margins (allowing outliers) we minimize the following expression:

$$\frac{1}{n}\left(\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b))\right) + C\|\mathbf{w}\|^2,$$

where C is a regularization hyper-parameter that adds cost when the model uses higher parameter values.

Results

In our model selection step of the analysis the support vector classifier achieved the best mean accuracy during cross-validation, the best scoring kernel where a linear one with the hyper parameter $C = 0.919$. The test resulting confusion matrix on the test data can be seen in 2, as we can see this gave the model an accuracy of 0.88.

	Predicted 0	Predicted 1
Actual 0	169	31
Actual 1	16	178

Table 2: In the table we can see the results from our model evaluated on the test data.

Highest weighted words

With this model we are able to tell what features has the highest impact on the models decision, we can do this by taking a look at the values in the vector $\mathbf{w}$, the results can be seen in table 3.

Word	value	Word	value
yes	2.63	not	-4.69
science	2.58	never	-3.66
vaxxers	2.54	no	-2.98
anti	2.39	rushed	-2.96
get	2.32	experimental	-2.85
hope	2.18	poison	-2.8
every	2.17	years	-2.63
available	2.16	test	-2.53
vaccinated	2.07	term	-2.44
great	2.06	don	-2.41

Table 3: Above we can see the top ten positive impacting features and the top ten negative impacting features

A closer look at some classifications

Now we will take a close look at a few observations that the model wrongfully classified in the test data to see what the reason is for why they where miss classified. Here is an example of false negative: *"I think the vaccine is ok."* and here is an example of false positive: *"the vaccines don't even work"*. By taking a look at the tf-idf transformation of the sentences and the corresponding weights for these words we will get a deeper insight why the model miss classifies these observations.

We begin with *"I think the vaccine is ok."* when doing the math on this we get the following:

$$w_{\text{think}}x_{\text{think}} + w_{\text{the}}x_{\text{the}} + w_{\text{vaccine}}x_{\text{vaccine}} +$$
$$w_{\text{is}}x_{\text{is}} + w_{\text{great}}x_{\text{great}} + b$$
$$= -0.89 \cdot 0.56 + 0.81 \cdot 0.19 + 0.12 \cdot 0.22 +$$
$$0.49 \cdot 0.3 + -0.16 \cdot 0.72 - 0.16 = -0.45,$$

here we can see that the majority of the words have an positive value but the two that has a negative value have higher tf-ifd scores at thus makes the sum en up as negative.

Now we are going to take a closer look at the false negative sample *"the vaccines don't even work"*:

$$w_{\text{the}}x_{\text{the}} + w_{\text{vaccines}}x_{\text{vaccines}} + w_{\text{don't}}x_{\text{don't}} +$$

$$\begin{aligned}&w_{\text{even}}x_{\text{even}} + w_{\text{work}}x_{\text{work}} + b\\ &= 0.81 \cdot 0.2 + 0.74 \cdot 0.38 + -2.41 \cdot 0.43 +\\ &-0.05 \cdot 0.56 + 1.44 \cdot 0.57 - 0.16 = 0.04,\end{aligned}$$

here we can see that the words *"the"*, *"vaccine"* and *"work"* has an higher positive impact then the words *"don't"* and *"even"*, however not by much since the resulting sum is close to zero.

Conclusion

Even one of the more simpler models as a linear support vector machine is able to tackle this large scale problem and doing so with quite pleasing performance. The observations that the model miss classified aren't all that obvious, take for example *"Alright, so now look into the covid protein spike and it's prion capable modification. Make sure you understand what you are getting yourself into when you take those shots, and it will also shed some light on the side effects you may have seen."*, this is a false positive, however it isn't that clear for me either when I read it, one could interpret this as just being general advice not a clear opinion. When performing the calculation showed in the previous section to this observation we get a score very close to zero: 0.027. This is also the case for many other wrongly classified observations. That is therefore to say that even tough the model is a off sometimes it is at least only a bit off.