Blindspot assignment: report

Kobza Ondrej

May 13, 2019

Contents

1 Raw data parsing and features extraction

Each http request consists of the following information:

1. Type of request (GET/POST/PUT) and url
2. User-Agent
3. Cache-control
4. Pragma
5. Accept
6. Accept-Encoding
7. Accept-Charset
8. Accept-Language
9. Cookie
10. Host
11. Connection
12. Content-Length (not for GET request)
13. Content-Type (not for GET request)

As every request in all provided datasets has the same value in the following attributes [1]

1. User-Agent
2. Pragma
3. Cache-control
4. Accept
5. Accept-Language
6. Connection
7. Accept-Encoding
8. Accept-Charset
9. Content-Type

They do not provide any information, which could help to distinguish between normal and anomalous data. Hence, these attributes are not taken into account.

The cookie feature contains a session id for every request. As this kind of information is useless for applied models, it was removed too. Additionally, the URL feature was also removed from the dataset as it contains URLs from the e-commerce application, that cannot be used.

Also, Content-Type and Host attributes are not taken as features, because these attributes are not very informative.

As there are only three possible values for request method, and requests with all of these values are presented in all three datasets, this information is also not taken into account.

So only two attributes of each request are considered: URL, Content-Length. The second one is considered as a feature itself (for GET request, this attribute is not present. In the feature represemntation of GET request, this feature is setted to 0), from the first one, eleven features are extracted:

1. max_arg_length (length of the longest argument)
2. number_of_arguments
3. number_of_digits_in_arguments
4. number_of_special_chars_in_args
5. length_of_arguments
6. number_of_letters_in_arguments
7. length_of_request
8. number_of_special_chars_in_path
9. avg_arg_length (average length of argument in the request)
10. number_of_letter_chars_in_path
11. length_of_path

The feature selection is inspired by [2], [1] One of the features proposed in [2] was not taken into account, two other features were added (+ Content-Length feature).

2 Data preprocessing

As pointed out here [1] the datasets have some duplications. I decided not to remove those because a request, which is present multiple times is likely to be very "normal" and this information can be very important in learning a classification model. My assumption is, that requests labeled as "normal" are not

equally normal between each other, hence some requests can be more "normal" than others.

It is necessary to do some data preprocessing, concretely data whitening. Whitening consists of two steps: The first operation decorrelates the data. The second operation, scaling, can be thought of squeezing the data, if the variance along a dimension is larger than one, or stretching the data if the variance along a dimension is less than one [3], [4]. After data whitening it one can estimate a probability distribution of the normal data and form a classifier, which will label samples which likely belonged to this distribution as "normal" and samples, which were not likely belongs to that distribution as "anomalous".

3 Unsupervised classifier

I assumed that the normal data after whitening are normally distributed. Therefore I decided to estimate a normal distribution of the normal training data (using EM algorithm) with two mixture components.

The number of mixture components is a hyperparameter of the classifier and can be tuned. This parameter is tuned on validation set. My validation set was created by randomly picking samples from the training set. Hence, the estimated accuracy on the validation set did not take into account how well the model can detect anomalous HTTP requests; it just took into account the number of normal samples classified as anomalous.

The classifier works as follows:
The classifier estimates the probability distribution of the training data. Then, it selects a threshold: the least normal sample from the training set - the sample with the lowest log probability according to the learned distribution. Then, when this classifier is required to classify a sample, it computes the log probability of this sample, compares this log probability to the threshold and if the log probability is higher than the threshold, the sample is classified as normal, else as anomalous.

The estimated accuracy of this classifier is 0.99998 (for number of mixture components = 1), which corresponds to 1 misclassification (but it depends on parameters initialization in EM algorithm, which is random so that the EM algorithm can get stuck in bad local optima. It also depends on the tuned hyperparameter - there is some randomness in hyperparameter tuning.).

4 Supervised classifier

In the task, you suggested using supervised learning. I tried logistic regression and support vector machines with a linear kernel (according to [2], a linear classifier should be optimal).

SVM classifier has a nice property - it can find a global optimum. The reason

is that it is transforming the classification problem to a dual one, which can be solved by QP. For QP problems, it is always possible to find a global optimum.

If we have classification problem with two classes with labels $y_k \in \{-1, +1\}$, the desired behaviour is:
$w^T x_k + b \geq +1$ if $y_k = +1$ and $w^T x_k + b \leq -1$ if $y_k = -1$. We can formalize this as optimization problem:

$$min_{w,b} \frac{1}{2} w^T w \quad s.t. \quad y_k[w^T x_k + b] \geq 1, k = 1, .., N.$$

The Lagrangian for this problem is

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{k=1}^{N} \alpha_k \{y_k[w^t x_k + b] - 1\} \quad \alpha_k \geq 0, k = 1, .., N$$

The solution is given by the saddle point of the Lagrangian, which results to the classifier

$$y(x) = sign[\sum_{k=1}^{N} \alpha_k y_k x_k^T x + b]$$

The partials derivations of Lagrangian are

$$\frac{\partial L}{\partial w} = 0 \quad -> \quad w = \sum_{k=1}^{N} \alpha_k y_k x_k$$

$$\frac{\partial L}{\partial b} = 0 \quad -> \quad \sum_{k=1}^{N} \alpha_k y_k = 0$$

By replacing the expression for w in the Lagranginian, one obtaines the dual problem, which leads to quadratic programming:

$$max_\alpha Q(\alpha) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \quad s.t. \quad \sum_{k=1}^{N} \alpha_k y_k = 0, \quad \alpha_k \geq 0$$

Using QP, global solution will be found.

4.1 Linear SVM classifier: non separable case

If our data are not linearly separable, we may allow that missclassifications can be tolerated. This can be formalize using so called slack variables:

$$min_{w,b,\xi} = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k \quad s.t. \quad y_k[w^T x_k + b] \geq 1 - \xi_k, \quad \xi_k \geq 0, \quad k = 1, .., N$$

The dual problem will be

$$max_{\alpha_k} Q(\alpha) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_K \quad s.t.$$

$$\sum_{k=1}^{N} \alpha_k y_k = 0$$

$$0 \leq \alpha_k \leq c, \quad k = 1, .., N$$

Firstly, one must create training and test set: I split the provided datasets into one and randomly choose samples from this big dataset for training and test set (60% for training set, 40% for test set).

Then, I tuned the C parameter with the grid search algorithm (with 10-fold cross-validation). The accuracy on the test set after parameter tuning was 1.

5 Conclusion

By trying both supervised and unsupervised approaches, I end up with a classifier with very high accuracy. A question is if the true performance is also that good. However, if I assume, the training data generalize well enough, I would be confident to use this classifier in a production.

The unsupervised classifier does not need any part of the test data for training, whereas for the supervised classifier one needs also some data in the provided test sets. This is a big advantage of the unsupervised classifier. However, the supervised svm classifier has a little bit better performance. So both classifiers have some pros and cons.

References

[1] Neural Analysis of HTTP Traffic for Web Attack Detection, David Atienza, Álvaro Herrero, and Emilio Corchado

[2] Analyzing HTTP requests for web intrusion detection, Sara Althubiti North Carolina A and T State University, saalthub@aggies.ncat.edu Xiaohong Yuan North Carolina A and T State University, xhyuan@ncat.edu Albert Esterline North Carolina A and T State University, esterlin@ncat.edu

[3] https://multivariatestatsjl.readthedocs.io/en/latest/whiten.html

[4] https://theclevermachine.wordpress.com/2013/03/30/the-statistical-whitening-transform/