Layer-wise Relevance Propagation based Sample Condensation for Kernel Machines

Daniel Winter[1], Ang Bian[2][0000−0002−7667−9780], and Xiaoyi Jiang[1][0000−0001−7678−9528]

[1] Faculty of Mathematics and Computer Science, University of Münster, Münster, Germany

[2] College of Computer Science, Sichuan University, Chengdu, China

Abstract. Kernel machines are a powerful class of methods for classification and regression. Making kernel machines fast and scalable to large data, however, is still a challenging problem due to the need of storing and operating on the Gram matrix. In this paper we propose a novel approach to sample condensation for kernel machines, preferably without impairing the classification performance. To our best knowledge, there is no previous work with the same goal reported in the literature. For this purpose we make use of the neural network interpretation of kernel machines. Explainable AI techniques, in particular the Layer-wise Relevance Propagation method, are used to measure the relevance (importance) of training samples. Given this relevance measure, a decremental strategy is proposed for sample condensation. Experimental results on three data sets show that our approach is able to achieve the goal of substantial reduction of the number of training samples.

1 Introduction

A fundamental result of learning theory is the family of representer theorems [7], which lead to the powerful kernel machines. Although trained to have zero classification error, kernel machines generalize well to unseen test data [4]. Compared to deep neural networks (DNN), they can be interpreted as two-layer NNs. Despite the simplicity, however, kernel machines turned out to be a good alternative to DNNs, capable of matching and even surpassing their performance while utilizing less computational resources in training [8,9].

Making kernel machines fast and scalable to large data is still a challenging problem. A major limiting factor is the need of saving all training samples, computing the corresponding Gram matrix, and solving the related linear equation system (see Section 2). In this paper we thus consider the problem of condensing the training samples, preferably without impairing the classification performance. Based on the interpretation of kernel machines as two-layer neural networks, we make use of explainable AI techniques [15], in particular Layer-wise Relevance Propagation (LRP) [14], as a means to measure the relevance (importance) of training samples. A decremental strategy is proposed to use this measure for sample condensation.

Sample condensation has been studied in other contexts, where the whole training set has to be saved and used for classification. Starting from the pioneer work [6], more advanced techniques have been proposed to boost the performance of nearest neighbor based classifiers [2,12]. In addition, nearest neighbor condensation has been applied to speed up the training of support vector machines [1] and convolutional neural networks [12].

The remainder of the paper is organized as follows. In Section 2 we introduce the fundamentals of kernel machines and discuss the need of sample condensation, thus motivating our work. Our sample condensation method is described in Section 3. Experimental results are reported in Section 4. Finally, Section 5 concludes the paper.

2 Kernel machines

Kernels are an efficient way to compute the similarity of two samples in a higher dimensional space. In this section we introduce a technique to fully interpolate the training data using kernel functions, known as kernel machines. Let $X = \{x_1, x_2, \ldots, x_n\} \subset \Omega^n$ be a set of n training samples with their corresponding targets $Y = \{y_1, y_2, \ldots, y_n\} \subset \mathcal{T}^n$ in the target space. A function $f : \Omega \to \mathcal{T}$ interpolates this data iif

$$f(x_i) \;=\; y_i, \quad \forall i \in 1, \ldots, n \tag{1}$$

Representer Theorem [7]. Let $k : \Omega \times \Omega \to \mathbb{R}$ be a positive definite kernel, X and Y a set of training samples and targets as defined above, and $g : [0, \infty) \to \mathbb{R}$ a strictly monotonically increasing function for regulation. We define E as an error function that calculates the loss l of f on the whole sample set with

$$E(X, Y) \;=\; E((x_1, y_1), ..., (x_n, y_n)) \;=\; \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) + g(\|f\|) \tag{2}$$

Then, the function f^* that minimizes E, $f^* = \text{argmin}_f\{E(X, Y)\}$, has the form

$$f^*(z) \;=\; \sum_{i=1}^{n} \alpha_i k(z, x_i) \quad \text{with } \alpha_i \in \mathbb{R} \tag{3}$$

We now can use f^* from Eq. (3) to interpolate our training data. Note that the only learnable parameters are $\alpha = (\alpha_1, \ldots, \alpha_n)$. Learning α is equivalent to solving the system of linear equations

$$K(\alpha_1^*, ..., \alpha_n^*)^T \;=\; (y_1, ..., y_n)^T \tag{4}$$

where $K \in \mathbb{R}^{n \times n}$ is the Gram matrix with elements $K_{ij} = k(x_i, x_j)$. Since the kernel function k is assumed to be positive definite, the Gram matrix K is invertible. Therefore, we can find the optimal α^* to construct f^* by

$$(\alpha_1^*, ..., \alpha_n^*)^T \;=\; K^{-1}(y_1, ..., y_n)^T \tag{5}$$

After learning, the kernel machine then uses the interpolating function from Eq. (3) to make prediction for test samples. In this work we focus on classification problems. In this case $f(z)$ is encoded as a one-hot vector $f(z) = (f_1(z), \dots f_t(z))$ with $t \in \mathbb{N}$ being the number of output classes. When predicting a test sample z, the output vector $f(z)$ is not a one-hot vector, in fact not even a probability vector, in general. The class which gets the highest output value is considered as the predicted class. If needed, e.g. for the purpose of classifier combination, the output vector $f(z)$ can also be converted into a probability vector by applying the softmax function.

The practical usability of kernel machines strongly depends on the size n of training set. Solving the optimal α^* in (5) in a naive manner requires computation of order $\mathcal{O}(n^3)$ and is thus not feasible for many applications. Recently, a highly efficient solver EigenPro has been developed [13] to enable significant speedup for training on GPUs.

Sample condensation is another way of efficiency boosting, which is required even when using high-performance solvers like EigenPro. After training, the testing using (3) still needs the whole set of training samples, which is similar to the situation with nearest neighbor based classifiers. In complex domains like strings and graphs the kernel computation may be costly [3,11,18] so that the need of considerably reducing the number of samples remains. Even in case of easy-to-compute kernel functions, it can be typically expected that not all training samples are relevant to the classification. This observation has been made before, e.g. when working with nearest neighbor based classifiers [2,12]. Thus, there is a general need of sample condensation for kernel machines. In this work we propose a novel approach tailored to sample condensation for kernel machines. To our best knowledge, there is no previous work with the same goal reported in the literature.

3 Sample condensation method

We make use of the neural network interpretation of kernel machines and apply the Layer-wise Relevance Propagation method to measure the relevance of training samples. Given the relevance estimation of training samples, a decremental strategy is then applied to select the most relevant samples out of a training set.

3.1 LRP for relevance measure of kernel machine

The kernel machine (3) can be seen as a network with one hidden layer. Let z be the test sample to which the target $f(z)$ should be computed. Given a training sample x_i out of the training set $X \subset \Omega^n$, we denote α_{it} as the trained weight between x_i and the value in the output $f_t(z)$. Figure 1 shows the network architecture of a kernel machine. The input z is represented by a single input neuron. Each training sample is represented by a single neuron in the hidden layer and connected to the input by a special connection applying the kernel function.

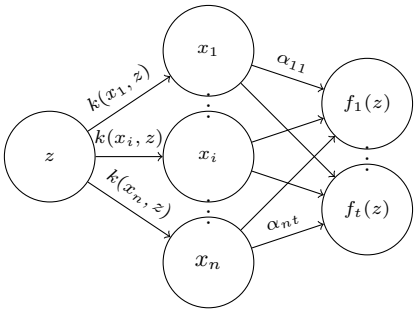


Fig. 1: Neural network representation of a kernel machine.

Each output class is represented by a neuron in the output layer, connected by the individually learned weight α_{it}.

The recent research on explainable AI has spawned many techniques, e.g. for studying the influence of hyper-parameters on training deep neural networks [5] and interpreting the behavior of neural networks [15]. In particular, it is possible to estimate the relevance of features (also hidden neurons) to the network decision. We apply such relevance estimation, concretely Layer-wise Relevance Propagation (LRP) [14], to determine the relevance of training samples.

Overall, the relevance estimation in our proposed approach consists of two steps. In the first phase the optimal α^* in (3) is computed, by means of a highly efficient solver like EigenPro [13] if needed. In the second phase a set of validation samples is used to estimate the relevance of training samples, which builds the foundation for sample condensation described in Section 3.

We apply LRP to propagate the relevance back from the output layer to the hidden layer and so assign each training sample a relevance measure. The formula to compute the relevance of a neuron x_i with connection to neurons x_t on a given validation sample z is given by

$$R(x_i, z) \;=\; \sum_t \frac{x_i w_{it}^+}{\sum_{i'} x_{i'} w_{i't}^+} \cdot R(x_t) \tag{6}$$

where $w^+ = \max(w, 0)$. Since x_t is in the output layer, its relevance is the target itself $R(x_t) = f_t(z)$. The weight w_{it} between the neurons representing x_i and x_t is given by the weight of the kernel machine $w_{it} = \alpha_{it}$. The activation on the neuron representing x_i is given by the kernel function $k(z, x_i)$. We thus can express the relevance $R(x_i, z)$ of a training sample x_i to the output $f(z)$ on a given validation sample z by

$$R(x_i, z) \;=\; \sum_t \frac{k(x_i, z)\alpha_{it}^+}{\sum_{i'} k(x_{i'}, z)\alpha_{i't}^+} \cdot f_t(z) \tag{7}$$

The target vector $f(z)$ is a one-hot vector in our case, i.e. the value in the vector corresponding to the correct target class t^* is 1 and all others are 0. Therefore,

we do not need to apply the outer sum but only calculate it for t^* since all other elements in the sum would be 0, which leads to

$$R(x_i, z) \;=\; \frac{k(x_i, z)\alpha^+_{it^*}}{\sum_{i'} k(x_{i'}, z)\alpha^+_{i't^*}} \tag{8}$$

Eq. (8) only focuses on the relevance of one validation sample z. To get a good estimation of the general relevance of a training sample x_i, we split the training set $X \subset \Omega^n$ in two distinct subsets X_{train} and X_{val} with $X = X_{train} \cup X_{val},\ X_{train} \cap X_{val} = \emptyset$. We train a kernel machine only using the set X_{train}. For each training sample $x_i \in X_{train}$ we then add all relevances on validation samples $x_j \in X_{val}$

$$R(x_i) \;=\; \sum_{x_j \in X_{val}} R(x_i, x_j) \;=\; \sum_{x_j \in X_{val}} \frac{k(x_i, x_j)\alpha^+_{it^*}}{\sum_{i'} k(x_{i'}, x_j)\alpha^+_{i't^*}} \tag{9}$$

3.2 Relevance-based sample condensation

Given the relevance estimation of training samples, we first sort the training samples by the relevance measure. A decremental strategy is then applied to select the most relevant samples out of a training set X_{train} by slowly eliminating the least relevant samples until only m ($< n$) samples are left.

The idea is to select the m samples with the highest relevance scores. A problem with this simple approach is a proper choice of the parameter m. If m is chosen too small, the selected samples will not suffice to reach a model of good accuracy. On the other hand, if m is chosen too big, the selection is sub-optimal since the same accuracy could be reached with fewer samples. Therefore, we define a parameter μ that expresses the minimum share of the original score (on the whole training set) that we like to retain. This means that for the whole training set X_{train}, the selected samples $X_{selected_m} \subset X_{train}$, and a score measure s, e.g. the accuracy, the following should apply

$$s(X_{selected_m}) \;\geq\; s(X_{train}) \cdot \mu \tag{10}$$

We assume that the evolution of the score is approximately monotonously rising, i.e. the score is in general higher for greater m, but may have small local noise, which is however small enough to be ignorable. Later in Section 4 we will show that the score indeed is of such a form.

A possible way to find m samples that represent the data best is to drop the Δ least relevant samples in each step. We train a new kernel machine in each step with the remaining samples and re-calculate the relevances with this machine. In general, we hope that in each step there is less redundancy. For example, a medium relevant sample can become more relevant in the next iterations, when other samples that are similar to it are dropped out. In each iteration, we thus train a kernel machine and drop the least relevant Δ samples regarding to the validation set. The algorithm is depicted in Algorithm 1. Due to the fact that a

Algorithm 1 Decremental sample condensation

procedure DECREMENTAL _ SELECTION($X_{train}, Y_{train}, X_{val}, Y_{val}, X_{test}, Y_{test}, k, \Delta, \mu$)
  $model$ = TRAIN_KM(X_{train}, Y_{train}, k)
  $score$ = TEST_KM($model, X_{test}, Y_{test}, k$)

  $score_i = score$
  while $score_i > score * \mu$ **do**
    $relevances$ = GET_RELEVANCES($model.\alpha, X_{train}, X_{val}, Y_{val}, k$)
    X_{train} = ARGSORT($X_{train}, relevances$) ▷ Sort by the relevances
    Y_{train} = ARGSORT($Y_{train}, relevances$)

    $X_{train} = X_{train}[1 : -\Delta]$ ▷ drop the last m elements
    $Y_{train} = Y_{train}[1 : -\Delta]$

    $model$ = TRAIN_KM(X_{train}, Y_{train}, k)
    $score_i$ = TEST_KM($model, X_{test}, Y_{test}, k$)
  end while
  return $X_{train}, Y_{train}, score_i$
end procedure

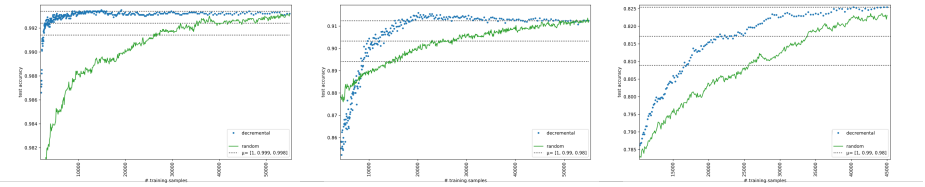
new kernel machine is computed in each iteration and its weight vector is used in the next iteration, the runtime therefore is always of order $\mathcal{O}(\frac{n_{train}}{\Delta})$.

Another way to find m samples that represent the data best is to start at the other side of the set, i.e. add the Δ most relevant samples in each step. This incremental strategy, however, turns put not to be competitive against the decremental strategy [17] and is thus not further discussed in this paper.

4 Experimental validation

4.1 Data sets

For our purposes, we have chosen three data sets for image classification that are broadly used and well studied. The **MNIST** data set contains 60,000 handwritten digits (graylevel images are of 28×28 pixels) for training and 10,000 handwritten digits for testing, written by 250 different people. The **MNIST-Fashion** data set has the same structure as the original MNIST data set (i.e. 60,000 training images and 10,000 test images, all of size 28×28). It contains images of clothes of 10 different classes (t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot). The **CIFAR-10** data set is formed by selecting and labeling proper images out of the *80 million tiny images* data set. It contains 50,000 training images plus 10,000 designated test images, each being a 32×32 RGB-image and labeled with one of the ten classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The objects in the images were captured from different view points and from different distances, which leads to



(a) MNIST (b) MNIST-Fashion (c) CIFAR-10

Fig. 2: Accuracy with selected training samples for the three data sets.

Data set	μ	approach	
		decremental	random
MNIST	0.998	2,500	26,000
	0.999	3,000	35,200
MNIST-Fashion	0.98	8,400	14,200
	0.99	9,700	22,900
CIFAR-10	0.98	16,600	25,600
	0.99	21,600	33,900

Table 1: Required number of samples to reach a certain accuracy level.

more variety in the data set compared to the other two data sets. For all three data sets, 90% is of the training data is really used as training data while the remainder 10% serves as validation data for relevance estimation.

The state-of-the-art classification results on these data sets can be found in [10,16,20], respectively. It is important to emphasize that it is not our goal to beat these results. Instead, we use them to study the ability of our approach to sample condensation without impairing the classification performance of kernel machines. The power of kernel machines themselves as classifier has already been demonstrated in the literature [4,9].

4.2 Results

Since convolutional neural networks (CNN) are powerful in feature learning, we train a CNN with all the samples and resort to using the learned features from the convolutional layers as input to a kernel machine. In all our experiments we use the Laplacian kernel $k(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||}{2\sigma}\right)$ with a bandwidth $\sigma = 7$. We chose the bandwidth $\sigma = 7$ since the experiments only show minor improvements with larger bandwidths.

In Figure 2 we show the performance of our approach on the three test sets (the step size is set to $\Delta = 100$). For comparison purpose, we also show the performance of the same number of randomly selected samples. We marked the original accuracy with the whole test set and $\mu = 99\%$ and $\mu = 98\%$ of this accuracy as stop criterion described in the algorithm. Note that for the MNIST

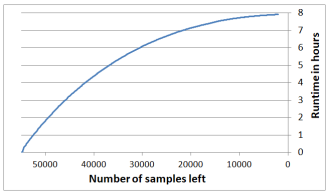


Fig. 3: Runtime with decremental approach on MNIST.

Data set	number of samples	selected samples	random samples
MNIST	2,500	0.9799	0.9580
	3,000	0.9831	0.9654
MNIST-Fashion	8,400	0.8067	0.8698
	9,700	0.8449	0.8779
CIFAR-10	16,600	0.7191	0.7436
	21,600	0.8146	0.8108

Table 2: Comparison of the accuracies of CNN trained using the selected samples.

set we alternately chose $\mu = 99.9\%$ and $\mu = 99.8\%$ since the accuracy for this set was still high even with only 2,500 of 60,000 samples left. The required number of samples to reach a certain accuracy is shown in Table 1.

When decreasing the number of samples (i.e. reading the figures from right to left), the accuracy of the randomly selected samples considerably decreases, while the accuracy of the samples selected with our approach only decreased slowly (CIFAR-10) or stays static (MNIST and MNIST-Fashion). For MNIST-Fashion, reducing the samples even increases the accuracy slightly.

To produce the data for the decremental approach in Figure 2 we reduced the training set down to 2,000 samples. This took 7.9h for the two MNIST data sets and 11.6h for CIFAR-10 because the feature vectors contain more elements here. Note that the runtime is recorded on a computer with 16 GB memory and Nvidia GTX 760. As an example, Figure 3 shows the accumulated runtime in dependency to the number of samples left on the MNIST data set.

We now investigate if the selected samples have a higher expressiveness in general or if it is limited only to our special experiment with kernel machines. Therefore, we only use the individual selected samples to train the original CNN and compare the accuracy of the resulting model to the accuracy with a model trained of the same number of randomly selected features. Table 2 shows the result of this comparison for the samples selected with our approach. We can see that the selected samples do not seem to have a higher expressiveness in general. Only for the MNIST data set, where we managed to select very few samples, the accuracy of the selected samples is higher. On the other data sets, the accuracy is mostly the same or slightly worse.

Overall, our LRP-based sample condensation technique can reduce the number of training samples while still preserving the high accuracy of kernel machines. As input for these studies we have chosen the output of the convolutional (feature learning) part of a CNN, which was trained once with the whole data set. In the comparison shown in Table 2, the CNN, especially its convolutional part, was only trained with the remaining, selected samples of the previous experiments. Since we could not show that the selected samples do lead to greater accuracy than randomly selected ones, we come to the following conclusion: The convolutional (feature learning) part of a CNN really benefits from a large base of training samples, whereas for the fully connected (classification) part a smaller base on training samples is sufficient.

It is important to mention that achieving a general higher expressiveness of the selected training samples is not the goal of this work. In fact, it cannot necessarily be expected since our approach is tailored to kernel machines. Our goal is to reduce the number of training samples to store for model inference and classification of unseen patterns, which is clearly achieved. We could reduce both the size of the model and the complexity of computation for kernel machines. To maintain 99% (99.9% for MNIST) of the original accuracy, we could reduce the number of training samples to 5% of the original training set for an easier task like MNIST and 43% for a more complex task like CIFAR-10.

5 Conclusion

This work intends to achieve substantial reduction of training data to store for model inference and classification of unseen patterns for kernel machines. Based on the neural network interpretation of kernel machines, we apply explainable AI techniques, in particular the Layer-wise Relevance Propagation method, to measure the relevance (importance) of training samples. A decremental strategy has been proposed for sample condensation. Our experimental results demonstrated the ability of our approach to considerably condense the training set without impairing the classification performance. Currently, we apply a rather straightforward decremental strategy for the condensation purpose. More sophisticated techniques can be studied in future. For instance, the concept of sparse representations [19] may be an option to model the importance of training samples.

To our best knowledge, our work is the first contribution to sample condensation tailored to kernel machines. As such it contributes to making kernel machines fast and scalable to large data. In addition, it also represents a novel application of explainable AI techniques.

Acknowledgment. This work was partly supported by the EU Horizon 2020 RISE Project ULTRACEPT under Grant 778062.

References

1. Angiulli, F., Astorino, A.: Scaling up support vector machines using nearest neighbor condensation. IEEE Transactions on Neural Networks **21**(2), 351–357 (2010)

2. Batchanaboyina, M.R., Devarakonda, N.: Design and evaluation of outlier detection based on semantic condensed nearest neighbor. Journal of Intelligent Systems **29**(1), 1416–1424 (2020)
3. Belazzougui, D., Cunial, F.: A framework for space-efficient string kernels. Algorithmica **79**(3), 857–883 (2017)
4. Belkin, M., Ma, S., Mandal, S.: To understand deep learning we need to understand kernel learning. In: 35th Int. Conf. on Machine Learning. pp. 540–548 (2018)
5. Hamid, S., Derstroff, A., Klemm, S., Ngo, Q.Q., Jiang, X., Linsen, L.: Visual ensemble analysis to study the influence of hyper-parameters on training deep neural networks. In: Proc. of 2nd Workshop on Machine Learning Methods in Visualisation for Big Data (MLVis@EuroVis). pp. 19–23 (2019)
6. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory **14**(3), 515–516 (1968)
7. Herbrich, R.: Learning Kernel Classifiers: Theory and Algorithms. The MIT Press (2002)
8. Huang, P., Avron, H., Sainath, T.N., Sindhwani, V., Ramabhadran, B.: Kernel methods match deep neural networks on TIMIT. In: IEEE Int. Conf. on Acoustics, Speech and Signal Processing. pp. 205–209 (2014)
9. Hui, L., Ma, S., Belkin, M.: Kernel machines beat deep neural networks on mask-based single-channel speech enhancement. In: Proc. of 20th Annual Conf. of Int. Speech Communication Association. pp. 2748–2752 (2019)
10. Kowsari, K., Heidarysafa, M., Brown, D.E., Meimandi, K.J., Barnes, L.E.: RMDL: random multimodel deep learning for classification. In: Proc. of the 2nd Int, Conf. on Information System and Data Mining. pp. 19–28 (2018)
11. Kriege, N.M., Johansson, F.D., Morris, C.: A survey on graph kernels. Applied Network Science **5**(1), 6 (2020)
12. Liang, T., Xu, X., Xiao, P.: A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. Pattern Recognition Letters **94**, 105–111 (2017)
13. Ma, S., Belkin, M.: Kernel machines that adapt to GPUs for effective large batch training. In: Proc. of Conf. on Machine Learning and Systems (2019)
14. Montavon, G., Binder, A., Lapuschkin, S., Samek, W., Müller, K.: Layer-wise relevance propagation: An overview. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K. (eds.) Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, LNCS, vol. 11700, pp. 193–209. Springer (2019)
15. Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K. (eds.): Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, LNCS, vol. 11700. Springer (2019)
16. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. In: Proc. of 3rd Int. Conf. on Learning Representations (2015)
17. Winter, D.: Sample Condensing for Kernel Machine based Classification. Master's thesis, University of Münster (2020)
18. Xu, L., Bai, L., Jiang, X., Tan, M., Zhang, D., Luo, B.: Deep Rényi entropy graph kernel. Pattern Recognition **111**, 107668 (2021)
19. Zhang, Z., Xu, Y., Yang, J., Li, X., Zhang, D.: A survey of sparse representation: Algorithms and applications. IEEE Access **3**, 490–530 (2015)
20. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proc. of The Thirty-Fourth AAAI Conf. on Artificial Intelligence. pp. 13001–13008 (2020)