Mathematical and Computational Applications

MDPI

Article

Using Kan Extensions to Motivate the Design of a Surprisingly Effective Unsupervised Linear SVM on the Occupancy Dataset

Matthew Pugh *, Jo Grundy, Corina Cirstea and Nick Harris

School of Electronics and Computer Science, University of Southampton, University Road, Southampton SO17 1BJ, UK; j.grundy@soton.ac.uk (J.G.); cc2@ecs.soton.ac.uk (C.C.); nrh@ecs.soton.ac.uk (N.H.)
* Correspondence: mp8g16@soton.ac.uk

Abstract: Recent research has suggested that category theory can provide useful insights into the field of machine learning (ML). One example is improving the connection between an ML problem and the design of a corresponding ML algorithm. A tool from category theory called a Kan extension is used to derive the design of an unsupervised anomaly detection algorithm for a commonly used benchmark, the Occupancy dataset. Achieving an accuracy of 93.5% and an ROCAUC of 0.98, the performance of this algorithm is compared to state-of-the-art anomaly detection algorithms tested on the Occupancy dataset. These initial results demonstrate that category theory can offer new perspectives with which to attack problems, particularly in making more direct connections between the solutions and the problem's structure.

Keywords: category theory; Kan extension; SVM; unsupervised; anomaly; occupancy

check for updates

Citation: Pugh, M.; Grundy, J.; Cirstea, C.; Harris, N. Using Kan Extensions to Motivate the Design of a Surprisingly Effective Unsupervised Linear SVM on the Occupancy Dataset. *Math. Comput. Appl.* **2024**, *29*, 74. https://doi.org/10.3390/mca29050074

Academic Editor: Leonardo Trujillo

Received: 15 August 2024
Revised: 28 August 2024
Accepted: 30 August 2024
Published: 2 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Category theory is not a discipline traditionally known for its practical applications. However, there have been indications that it can benefit the field of machine learning [1]. In particular, Kan extensions, a tool from category theory, have been used to describe the construction of a handful of supervised learning algorithms [2]. This paper will look at applying this construction to motivate the design of an unsupervised classification algorithm, seeking to more closely link the outcomes of the data analysis to the design of the algorithm on a real-world problem.

The Occupancy dataset, first introduced for supervised learning, has also been used to demonstrate the performance of unsupervised anomaly detection algorithms on time series data. Though a seemingly reasonable choice for this task, it possesses two potential issues. Firstly, its classification labels have a limited relationship to the information provided by the timestamps or sequences of recorded points. Secondly, the dataset contains anomalous points whose nature is not reflected in their classifications. These characteristics are indicated by the initial data analysis and included in the construction of a Kan extension, which is used to derive the "Constrained Support Vector Machine" (C-SVM) algorithm.

The C-SVM is an unsupervised linear SVM whose hyperplane is constrained to intersect a given point. It achieved an accuracy of 93.5% and an ROCAUC of 0.98, which is a competitive score with regard to algorithms presented in related works. Motivating the design of the C-SVM from the hypothesised characteristics of the Occupancy dataset allows its performance to validate their relevance to the machine learning problem. The presence of these characteristics indicates that the dataset should be used with caution when benchmarking other time series anomaly detection algorithms.

2. Materials and Methods

2.1. Dataset

The Occupancy dataset is a five-dimensional time series dataset that records temperature (Celsius), relative humidity (percentage), light (Lux), CO_2 (parts per million),

humidity ratio (kilograms of water divided by kilograms of air) in an office room. Each data point has one of two classes, indicating if the room is occupied or not. Introduced by Luis M. Candanedo and Véronique Feldheim [3] (accessible at the UCI ML repository as of 16 January 2022), the original paper explores the use of supervised machine learning models to detect building occupancy and improve the energy efficiency of smart buildings. The dataset contains 20,560 data points recorded over 16 days. In total, 15,843 (77.1%) of the points correspond to the not-occupied class and 4717 (22.9%) to the occupied class. The bias towards the not-occupied class has led to the dataset being used to evaluate unsupervised classification techniques. This dataset has been used in a large number of works. For comparison, these are limited to English language primary research, which tests an unsupervised algorithm to classify occupancy on the unmodified Occupancy dataset. Screening occurred in two phases: the first was based only on abstracts, and the second considered full papers. From an initial pool of 226 papers that cited this dataset, 12 met the inclusion criteria; see Tables 1 and 2.

PCA was used to plot the first three principal components of the dataset. The timestamps for each point were removed, and their sequential nature was ignored (Figure 1).

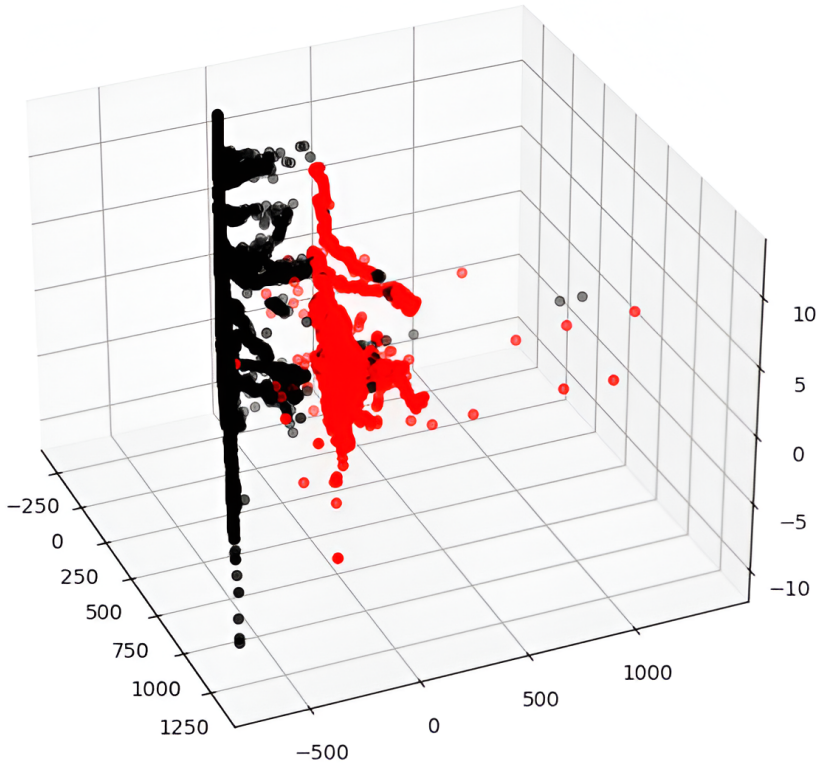


Figure 1. The first three principle components of the Occupancy dataset produced by the PCA algorithm after the data were normalised to have a mean of zero and variance of one in each dimension. The time components of the data were ignored. Points corresponding to the room being unoccupied are coloured black, and occupied rooms are coloured red.

The figure shows a clear separation between the two classes of the Occupancy dataset, indicating that a hyperplane may suitably classify it. As this separation is seen when time is disregarded, it would appear that the temporal component of the dataset is not a particularly relevant feature for the classification problem and should be disregarded by a potential classification algorithm. It is also helpful to note that a small number of points from both classes deviate from the main body of the data. This appears to be due to one of the sensors breaking during data collection.

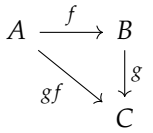
2.2. Category Theory and Kan Extensions for Classification Algorithms

Previous works have suggested that Kan extensions, a tool from category theory, might be used to generalise the notion of extrapolating new data points from previous observations, providing an interesting construction for a supervised classification algorithm from Kan extensions [2]. This construction can be modified to help motivate the unsupervised C-SVM algorithm in the following section. There are three core concepts which are necessary to introduce the definition of a Kan extension: categories, functors, and natural transforms.

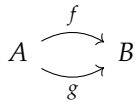
2.2.1. Categories

A category is similar to a graph. Where a graph would have nodes and edges, a category has objects and arrows, called morphisms. A morphism starts at an object (domain) and finishes at an object (codomain). A morphism f from a domain A to a codomain B can be written as $f : A \to B$.

The morphisms in category theory are inspired by functions. As a result, morphisms can be composed like functions. These compositions can be thought of as paths through a graph. If one morphism ends where another starts, they can be composed. The morphisms $f : A \to B$ and $g : B \to C$ can be composed as $g(f(\cdot)) = (g \circ f)(\cdot) = gf(\cdot) : A \to C$. The composition can be represented by a commutative diagram. A diagram is said to commute when all paths with the same domain and codomain (start and finish) are equal, i.e., $g \circ f = gf$.



There can be multiple morphisms between two objects, as shown in the non-commutative diagram below.



Finally, it is required that every object in a category has a morphism to itself which does nothing, called the identity morphism. It can be written (for an object A) as 1_A or Id_A. To say that a morphism does nothing is to say that composing a morphism with an identity morphism yields the same morphism: $Id_A \circ f = f \circ Id_A = f$.

These requirements produce the following definition of a category [4–6].

Definition 1 (Category)**.** *A category C consists of a class of objects $Ob(C)$, and between any two objects $x, y \in Ob(C)$, a class of morphisms is $C(x, y)$, such that*

- *Any pair $f \in C(x, y)$ and $g \in C(y, z)$ can be composed to form $gf \in C(x, z)$.*
- *Composition is associative: $(hg)f = h(gf)$.*
- *Every object $x \in Ob(C)$ has an identity morphism $Id_x \in C(x, x)$.*
- *For any $f \in C(x, y)$, $fId_x = f = Id_y f$.*

2.2.2. Functors

A functor is a morphism between categories. A functor $F : C \to D$ maps objects and morphisms in a category C to objects and morphisms in category D. The structure of a category comes from how its morphisms can be composed together. For the functor to preserve the structure of the category, it must preserve the morphism's composition. Firstly, this requires that identity morphisms in C are mapped to identity morphisms in D. Secondly, if two morphisms in C combine to make a third, then the image of

these three morphisms in D should be a valid composition, i.e., if $f \circ g = h$ in C, then $F(f) \circ F(g) = F(h)$ in D.

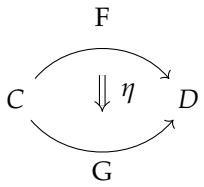
The following defines a functor [4–6].

Definition 2 (Functor). *A functor $F : C \to D$, between categories C and D, sends every object $x \in Ob(C)$ to $F(x) \in Ob(D)$, and every morphism $f \in C(x, y)$ sends every object to $F(f) \in D(F(x), F(y))$, such that*

- *F Preserves composition: $F(gf) = F(g)F(f)$.*
- *F Preserves identities: $F(Id_x) = Id_{F(x)}$.*

2.2.3. Natural Transforms

A natural transform is a morphism between functors. Given two functors $F, G : C \to D$, a natural transform $\eta : F \Rightarrow G$ slides the outputs of F to the outputs of G along morphisms in D.



The following defines a natural transform [4–6].

Definition 3 (Natural Transform). *Given functors $F, G : C \to D$, between categories C and D, a natural transformation $\alpha : F \Rightarrow G$ is a family of morphisms $\alpha_x : F(x) \to G(x)$ in D for each object $x \in Ob(C)$, such that $G(f)\alpha_x = \alpha_y F(f)$ for any $f \in D(x, y)$, i.e., the following diagram commutes.*

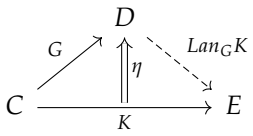
$$\begin{array}{ccc} F(x) & \xrightarrow{\alpha_x} & G(x) \\ \downarrow{\scriptstyle F(f)} & & \downarrow{\scriptstyle G(f)} \\ F(y) & \xrightarrow{\alpha_y} & G(y) \end{array}$$

2.2.4. Kan Extensions

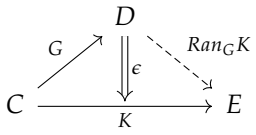
Kan extensions ask how one might extend one functor to produce another. Given two functors $K : C \to E$ and $G : C \to D$, a Kan extension attempts to find a functor $F : D \to E$, such that FG is approximately equal to K. It is overly restrictive (and often less helpful) to ask for FG to be exactly equal to K. So, Kan extensions weaken the equality to the requirement for some universal natural transformation between the two functors. The left Kan extension asks for a natural transform $\eta : K \Rightarrow FG$, and the right asks for $\epsilon : FG \Rightarrow K$. The Kan extensions require that for any natural transform γ and functor H pair, the natural transform can be factored uniquely as a composition of the "best" natural transform, as well as some other natural transform, e.g., $\gamma = \alpha\eta$. The notation for the functors, which satisfy the requirements of the left and right Kan extensions, are $Lan_G K$ and $Ran_G K$, respectively.

The following defines the left and right Kan extensions [4].

Definition 4 (Left Kan Extension). *Given functors $K : C \to E$, $G : C \to D$, a left Kan extension of K along G is a functor $Lan_G K : D \to E$ together with a natural transformation $\eta : K \Rightarrow (Lan_G K)G$, such that for any other such pair $(H : D \to E, \gamma : K \Rightarrow HG)$, γ factors uniquely through η.*



Definition 5 (Right Kan Extension)**.** *Given functors $K : C \to E$, $G : C \to D$, a right Kan extension of K along G is a functor $Ran_G K : D \to E$ together with a natural transformation $\epsilon : (Ran_G K)G \Rightarrow K$, such that for any $(H : D \to E, \delta : HG \Rightarrow K)$, δ factors uniquely through ϵ*



2.2.5. A Supervised Classification Algorithm from Kan Extensions

This subsection summarises the supervised Kan extension classification algorithm described by Dan Shiebler [2], which forms the basis for the unsupervised algorithm developed in this paper.

The unique IDs of a dataset can be represented by a category whose only morphisms are the identity morphisms for each object. This is called a discrete category. For a given discrete category I', functors may assign values to each data point. The input data can be described by a functor $G : I' \to I$ (Figure 2). In order to encode some of the geometric information present within the dataset, rather than a discrete category, I is allowed to be a Preorder. This is a category with at most one morphism between any two objects. An example would be the ordered real numbers $\mathbb{R}_{\leq}$, whose objects are the real numbers and for which a unique morphism $\leq : x \to y$ exists if and only if $x \leq y$.

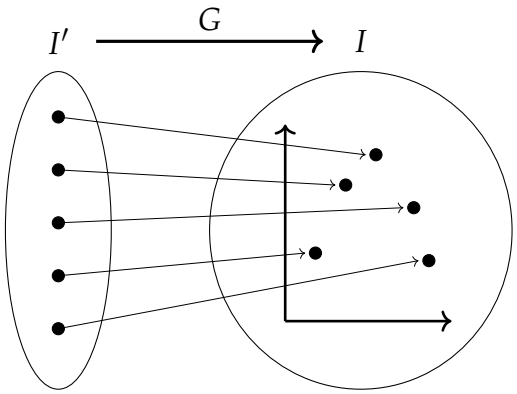


Figure 2. A functor $G : I' \to I$ embedding discrete data points from I' into a richer space I.

For a dataset with binary classification labels, the target data can be represented by a functor $K : I' \to \{\text{false}, \text{true}\}$. In this instance, $\{\text{false}, \text{true}\}$ represents an ordered two-object category whose only non-identity morphism is $\leq : \text{false} \to \text{true}$.

For each of the points in I selected by G, there is information about their classification labels given by K. The general principle of a supervised classification algorithm is to extend information from a subset to the whole space. This can be described in this context as finding a suitable functor $F : I \to \{\text{false}, \text{true}\}$.

An initial attempt may be to assign F to be either the left $(Lan_G K)$ or right $(Ran_G K)$ Kan extensions in Equations (1)–(3) from [2].

$$Lan_G K : I \to \{\text{false}, \text{true}\} \quad Ran_G K : I \to \{\text{false}, \text{true}\} \tag{1}$$

$$Lan_G K(x) = \begin{cases} \text{true} & \exists x' \in I'(G(x') \leq x,\ K(x')) \\ \text{false} & else \end{cases} \tag{2}$$

$$Ran_G K(x) = \begin{cases} \text{false} & \exists x' \in I'(x \leq G(x'),\ \neg K(x')) \\ \text{true} & else \end{cases} \tag{3}$$

An example visualisation of these equations can be seen in Figure 3, in which I was set to $\mathbb{R}_{\leq}$. The figure presents the resulting Kan extensions from datasets with overlapping and non-overlapping classes.

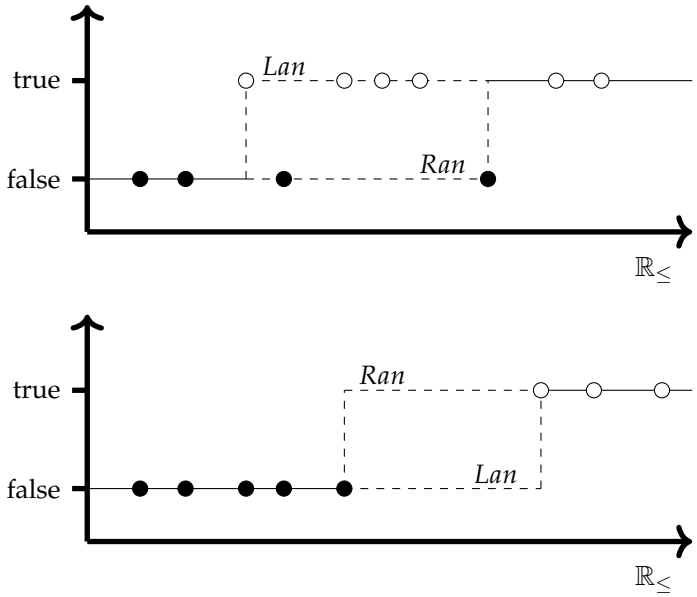


Figure 3. The left and right Kan extensions, $Lan_G K$ and $Ran_G K$, produced from the functors $K : I' \to \{\text{false}, \text{true}\}$ and $G : I' \to \mathbb{R}_{\leq}$, which represent a binary classification dataset over the ordered real numbers. The two graphs show the different extensions produced from a dataset with overlapping classes (**upper**) and separable classes (**lower**).

In the case where I is $\mathbb{R}_{\leq}$, F is forced to become a step function due to the induced ordering of the two categories by their morphisms. This creates a decision boundary at some point in $\mathbb{R}_{\leq}$.

$$F : \mathbb{R}_{\leq} \to \{\text{false}, \text{true}\} \tag{4}$$

$$F(x) := \begin{cases} \text{true} & \alpha \leq x \\ \text{false} & else \end{cases} \tag{5}$$

For two functors $F, F' : \mathbb{R}_{\leq} \to \{\text{false}, \text{true}\}$, a natural transform $\gamma : F \Rightarrow F'$ must select for each object in $\mathbb{R}_{\leq}$ and a morphism in $\{\text{false}, \text{true}\}$. This, at most, may alter the output of F from false to true while retaining its monotonicity. Considering the decision boundary α in F, the effect of γ can only be to increase α. This means that a natural transform can only exist between F and F' if $\alpha \leq \alpha'$. When composed with $G : I' \to \mathbb{R}_{\leq}$, the objects of I and their image under G restrict the components of γ. Consequently, the left and right Kan extensions produce classifying functions with no false negatives and no false positives, respectively.

This approach is not yet sufficient for more complex systems. To extend the utility of this representation, an additional, trainable functor $f : I \to I^*$ can be added.

$$\begin{array}{ccc} I & \xrightarrow{f} & I^* \\ {\scriptstyle G}\uparrow & & \downarrow{\scriptstyle F} \\ I' & \xrightarrow{K} & \{\text{false}, \text{true}\} \end{array}$$

For the case of overlapping classification regions, it is a reasonable assumption that the less these regions overlap, i.e., the smaller the disagreement region, the better the resulting classification is likely to be. For this purpose, by assuming that I^* is $\mathbb{R}^d$, a function known as the ordering loss can be introduced [2], with the guarantee that minimizing the ordering

loss will also minimize the disagreement region. The following equation uses $f(x)[i]$ to represent the i-th component of the vector $f(x) \in \mathbb{R}^d$.

$$l : (I \to \mathbb{R}^d) \to \mathbb{R} \tag{6}$$

$$l(f) = \sum_{i \leq a} max(0,\ max\{f(x)[i] \mid x \in I', \neg K(x)\} - min\{f(x)[i] \mid x \in I', K(x)\}) \tag{7}$$

2.3. Motivating C-SVM through Kan Extensions

Two steps are required to motivate the C-SVM from the construction shown in Section 2.2.5. Firstly, details about the dataset, which were introduced in Section 2.1, can be used to populate the construction with information specific to this task. Secondly, the construction of a classification algorithm through Kan extensions needs to be modified for it to define an unsupervised algorithm.

The morphism $G : I' \to I$ assigns information regarding the input data of the dataset to each of the data points in the discrete category I'. In this case, the measured values in the time series can be represented as a five-dimensional real number vector $\mathbb{R}^5$. The time series information was determined to be irrelevant in the data analysis. By selecting I' as the discrete category $[n]$ with n objects, which act analogously as unique IDs to the n data points in the dataset, $[n]$ disregards any time series information. For the sake of this formulation, rather than using G directly, the data can be shifted to have a mean of $\vec{p}$, giving $G' : [n] \to \mathbb{R}^5$. The choice of $\vec{p}$ is arbitrary, making it a hyper-parameter of this algorithm. By inspection, the data analysis indicates that normalising the datasets to have a mean of zero is a reasonable choice.

The data analysis identified that the data could be suitably separated with a hyperplane. This can be represented by constraining the trainable portion of the construction to be a linear map into the real numbers $f : \mathbb{R}^5 \to \mathbb{R}_{\leq}$. For this construction, $\mathbb{R}^5$ is the discrete category, with $\mathbb{R}_{\leq}$ being the category which represents the ordered set of real numbers. The choice of $\mathbb{R}_{\leq}$ as the codomain of f is equivalent to stating the belief that the points of $\mathbb{R}^5$ can be ordered based on how likely they are to be classified as either "occupied" or "not-occupied". This means that f uses the ordering of $\mathbb{R}_{\leq}$ to induce a partial ordering on the points of $\mathbb{R}^5$ based on their presumed classification. The role of the Kan extensions in this construction is to decide the cutoff, where points greater than a certain value must be classified as "occupied" and less than that value as "not-occupied". These choices result in the diagram in Figure 4.

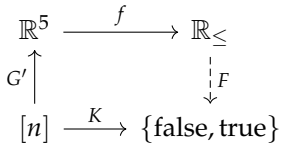


Figure 4. A diagram describing the structure of the classification algorithm for the Occupancy dataset. By defining $K := (0\leq)fG'$ and allowing f to be a trainable hyperplane, this diagram represents an unsupervised linear classifier.

The second problem is modifying the construction to be applied to an unsupervised learning problem. The definition of the Kan extension requires that the morphism $K : [n] \to \{\text{false}, \text{true}\}$ is known. By introducing an additional morphism (Equation (8)), it is possible to define K through composition $K := (0\leq)fG'$, as follows:

$$(0\leq) : \mathbb{R}_{\leq} \to \{\text{false}, \text{true}\} \tag{8}$$

$$(0\leq)(x) := \begin{cases} \text{true} & 0 \leq x \\ \text{false} & else \end{cases} \tag{9}$$

Interpreting this composition, introducing $(0\leq)$ converts the hyperplane f into a binary classifier. The points' classification depends on which side of the hyperplane they lie on. Due to the definition of K by composition, it is guaranteed that there is no overlap in the classification boundaries. As the resulting Kan extensions are from the function space $\mathbb{R}_{\leq} \to \{\text{false}, \text{true}\}$, the resulting left and right Kan extensions correspond with the functions shown in Figure 3. The decision boundaries formed by the three function $(0\leq)f$, $(Lan_{fG'}K)f$, and $(Ran_{fG'}K)f$ can be visualised as in Figure 5.

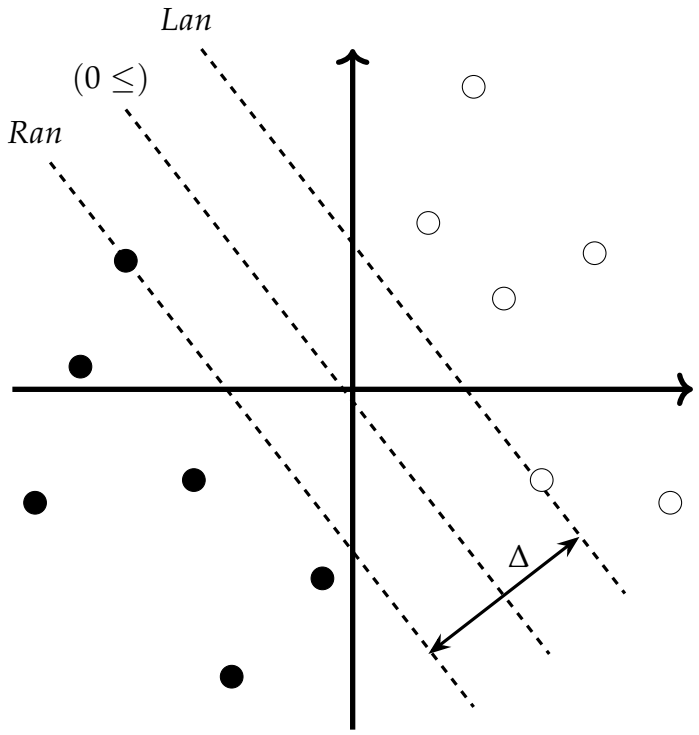


Figure 5. A representation of the classification boundaries produced by the functions $(0\leq)f$, $(Lan_{fG'}K)f$, and $(Ran_{fG'}K)f$ for an imagined dataset on the plane. The white and black circles represent points whose classifications are true and false, respectively.

Any choice of f now produces a preliminary classification of the points in the Occupancy dataset. The left and right Kan extensions identify the points closest to the decision boundary. The classification quality produced by a given f can be judged by the distance between the left and right Kan extensions, Δ.

The ordering loss, as previously defined, cannot be used directly for this version of the problem, as it is zero when there is no overlap between classification regions. However, given that it can be guaranteed that there will never be an overlap of the classification regions, the outer max function of the ordering loss function can be removed, allowing the modified ordering loss function (l') to become negative (Equations (10) and (11)). Minimisation of the modified ordering loss function maximises the separation region between the left and right Kan extensions. In the particular case where f has codomain $\mathbb{R}_{\leq}$, the modified ordering loss is reduced to be $l'(f) = -\Delta$, where Δ can be seen as the difference between the closest points on each side of the hyperplane f when projected down onto the real numbers (Figure 5). Ultimately, the choices and modifications applied to the construction produce an algorithm which appears to be a linear, unsupervised SVM whose hyperplane is constrained to pass through $\vec{p}$

$$l' : (I \to \mathbb{R}^d) \to \mathbb{R} \tag{10}$$

$$l'(f) = \sum_{i \leq a} max\{f(x)[i] \mid x \in I', \neg K(x)\} - min\{f(x)[i] \mid x \in I', K(x)\} \tag{11}$$

2.4. Implementation of C-SVM

From the construction presented in the previous section, the remaining task is to produce an algorithm which finds a suitable, linear transform, $f : \mathbb{R}^5 \to \mathbb{R}_{\leq}$, which maximises the distance (Δ) between the decision boundaries produced by the left and right Kan extensions. As f is a linear transform, it can be defined as $f(\vec{x}) := \vec{x} \cdot \hat{v}$, where $\hat{v}$ is a five-dimensional real number vector. Applying f to every point of the normalised dataset, Δ can be computed as the difference between the data point with the smallest positive value after f, as well as the data point with the largest negative value after f. For the purposes of classification, the process of normalising the dataset and then transforming with f can be understood as assigning a score to each point based on its signed distance from the hyperplane. This corresponds with the image of the point in $\mathbb{R}_{\leq}$ (Equation (12)):

$$\text{score} = (\vec{x} - \vec{p}) \cdot \hat{v} \tag{12}$$

The unsupervised fitting algorithm (Algorithm 1), which this paper introduces, determines the value for each dimension of the normal vector by rotating the vector around axes of a hyper-sphere centred at the constraining point. For each iteration, there are set values, the current value, and the unset values. For each loop, the fitting algorithm checks an integer number of uniformly spaced angles $\Theta_a \in [0, \pi)$, which are given by the following formula:

$$\Theta_a = a\pi/(\text{res} - 1)$$

To determine the value which maximises the plane's distance to the closest point, the algorithm checks integer values of a, where $0 \leq a < \text{res}$. The vector of values this induces can be represented by the notation $\Theta_{a<res}$, where $(\Theta_{a<res})_a = \Theta_a$. The set values remain unchanged, and only the current value and unset values are varied. At the end of the loop, the maximising value is assigned to the current value. The logic of conserving previously determined values, selecting the active dimension based on the angle, and modifying the following dimensions to preserve the unit magnitude of the vector is encoded in the following piece-wise function (Equation (13)):

$$\text{Angle2Value}(\hat{v}, i, j, \Theta_a) = \begin{cases} \hat{v}_j & j < i \\ \sin(\Theta_a) & j = i \\ \frac{\cos(\Theta_a)\sqrt{1-\sin(\Theta_a)}}{\dim(\hat{v})-i} & j > i \end{cases} \tag{13}$$

Algorithm 1 Fitting C-SVM

Input: data $\in \mathbb{R}^{n \times m}$, res $\in \mathbb{N}$
Output: $\vec{\mu}, \hat{v} \in \mathbb{R}^m$

$\vec{\mu} \leftarrow$ mean of the n points in data
$\hat{v}_{a \leq m} \leftarrow 0$
for $i \leftarrow 0$ **to** $i = m$ **do**
  $\Theta_{a<\text{res}} \leftarrow a\pi/(\text{res} - 1)$
  $\text{V}_{j \leq m,\, a<\text{res}} \leftarrow \text{Angle2Value}(\hat{v}, i, j, \Theta_a)$
  $\text{S} \leftarrow \text{MatrixProduct}(data - \vec{\mu},\ \text{V})$
  $\text{L} \leftarrow \text{MinAlongFirstAxisIfTrue}(\text{S}, \text{IsPositive})$
  $\text{R} \leftarrow \text{MaxAlongFirstAxisIfTrue}(\text{S}, \text{IsNegative})$
  $\Delta \leftarrow \text{L} - \text{R}$
  $k \leftarrow \text{MaxIndex}(\Delta)$
  $\hat{v}_i = V_{i,k}$
end for
return $\vec{\mu}, \hat{v}$

Fitting each of the values sequentially the fitting algorithm achieves a time complexity of $O(nm^2 \text{ res})$ with n being the number of data points, m being the number of dimensions, and res being the resolution of the angle search space.

3. Results

The scores given by the fitted C-SVM on every point in the dataset are shown in Figure 6.

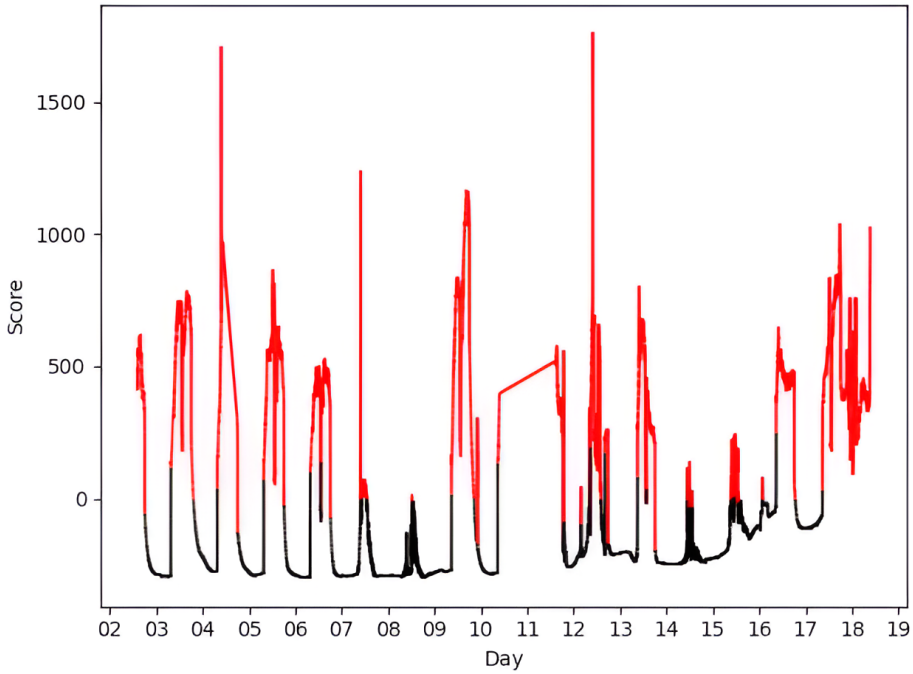


Figure 6. The scores generated by the fitted C-SVM on the Occupancy dataset. Points that are labelled as occupied in the dataset are coloured red, and unoccupied points are coloured black.

Points that the Occupancy dataset labels as occupied were coloured red, and the points corresponding to unoccupied points were coloured black. Via a visual inspection, higher scores assigned with the C-SVM correlate strongly with the building being occupied. The receiver operator curves (ROCs) for these scores, when compared to the ground truth labels of the dataset, are shown in Figure 7. For classification algorithms that return a scalar value (score) which is thresholded to produce true/false classifications, their ROCs can be used to show how modifying the threshold value (sensitivity) changes the rate of false positives and true positives. It is assumed that better classifiers will produce scores that more clearly delineate which points are true/false. This can be measured using the area under the receiver operator curve (ROCAUC). An almost perfect classifier would produce few false positives vs. its number of true positives. The better a classifier is the more its ROC approaches a step change and the closer its ROCAUC is to 1. The C-SVM achieved an ROCAUC of 0.9814 with a maximum accuracy of 93.5%.

Four of the papers found in Section 2.1 provided ROCAUC scores for models tested on the Occupancy dataset. The C-SVM algorithm achieved the fourth highest ROCAUC out of twenty-six models, with a difference in the ROCAUC score of 0.0146 between it and the top-performing model (Table 1). Six of the papers provided classification accuracies (with varying levels of precision) for the tested models. The C-SVM algorithm had the sixteenth highest accuracy out of twenty-eight models, with a difference in accuracy of 3.5% between it and the top-performing model (Table 2).

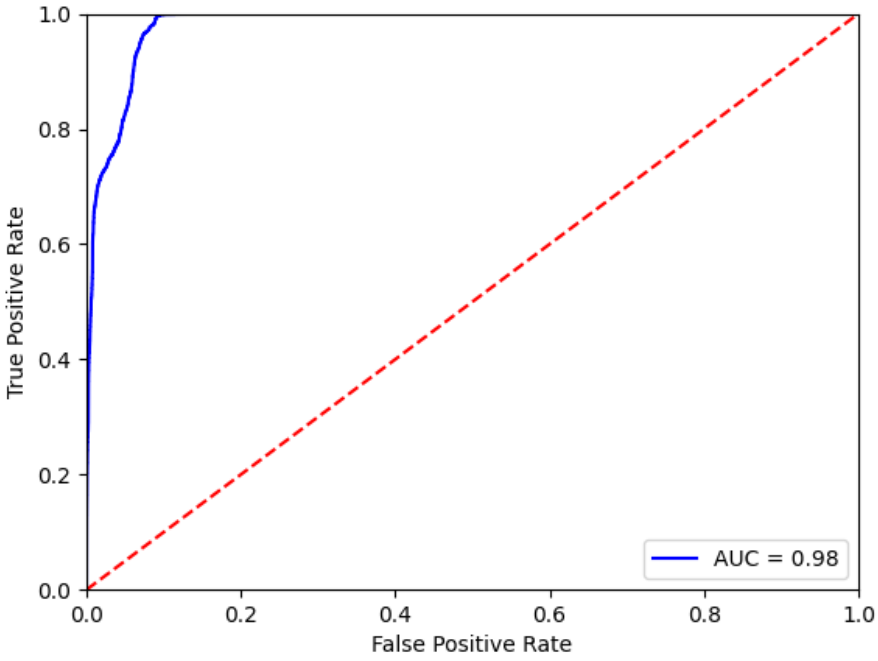


Figure 7. The receiver operator curve (ROC) for the scores given by the C-SVM when compared to the ground truth labels in the Occupancy dataset. The model achieved an ROCAUC of 0.9814.

Table 1. An ordered list of the ROCAUC scores of unsupervised algorithms on the Occupancy dataset, as provided by the papers found in Section 2.1.

Model	Citation	ROCAUC
EIF	[7]	0.9970
VAE	[7]	0.9960
AD HKDE	[8]	0.9907
C-SVM		0.9814
1C-SVM	[7]	0.9780
GRU-GSVDD	[9]	0.9109
GRU-GSVM	[9]	0.9059
KDE-AB	[8]	0.9531
FOGD	[8]	0.9490
IF	[7]	0.9470
K-KDE	[8]	0.9368
Online osPCA	[8]	0.9292
DIFF-RF	[7]	0.9000
LSTM-GSVM	[9]	0.8957
GRU-QPSVM	[9]	0.8719
SVM	[9]	0.8676
LSTM-GSVDD	[9]	0.8609
CSVM	[10]	0.8220
DSVM	[10]	0.8220
LSTM-QPSVM	[9]	0.8197
LSTM-QPSVDD	[9]	0.7869
LSTM	[9]	0.7444
GRU-QPSVDD	[9]	0.7417
SVDD	[9]	0.6715
LSVM	[10]	0.6670
KitNET	[7]	0.6580

Table 2. An ordered list of the accuracy scores of unsupervised algorithms on the Occupancy dataset, as provided by the papers found in Section 2.1. Note that the number of decimal points is inconsistent because different papers quoted the accuracy of their models to different precisions.

Model	Citation	Accuracy (%)
OEMP-4K	[11]	97
BEMG	[11]	96.5
BEMP	[11]	96.5
OEMP-2K	[11]	96.5
OEMP-3K	[11]	96.5
Batch UM	[12]	96
Batch CBM	[12]	96
ARF	[13]	95.9
DA	[13]	95.8
AD	[13]	95.3
OEMP-1K	[11]	95
RF	[14]	94.5
J 48	[14]	94.4
REPTree	[14]	94.4
Occ-STPN	[15]	94
C-SVM		93.5
IFC	[13]	93
HMM	[16]	90.2
LA	[13]	89.1
GeoMA	[11]	87
LB	[13]	86.4
SOL	[13]	85.4
AS	[13]	84
PHT	[11]	83.5
OB	[13]	82.3
AUE	[13]	81.8
RC	[13]	75.6
AWE	[13]	73.4

4. Discussion

4.1. Category Theory

It is reasonable to suggest that a person could design the C-SVM algorithm without any knowledge of category theory. At this initial phase, the utility of category theory with respect to ML may be primarily to be used as a descriptive language, offering occasional intuition. However, developing this more rigorous procedure of encoding the structure of ML problems may start to provide solutions which would not be immediately obvious from a traditional perspective.

Commonly, intuition and experience are used to bridge the gap between knowledge about an ML problem and the design of an algorithm. Category theoretic techniques have the potential to make this connection more explicit. Using the template of the Kan extension, adding information about the dataset began to outline the necessary algorithm. Not only does this form of notation highlight the reasoning for specific choices but it also suggests an alteration to the classic ML design loop. Traditionally, by testing iterations of algorithms, an engineer may understand more about the particular problem they are working on. Intuitively, this means adding information about the dataset into the design of the algorithm. However, with a categorical perspective, it may be more sensible to add new information about the structure of a dataset to its categorical description, trusting that these changes will indicate the appropriate modification to the algorithm design.

It is worth noting that the techniques used to derive the C-SVM algorithm are relatively crude compared to what may be possible with category theory. It was necessary to work inside of the category of categories to make use of the natural transforms required in the definition of the Kan extension. Unfortunately, this led to the use of categories themselves

as the objects of interest. It is true that categories can encode rich mathematical structures, but it is often their objects which represent the structures. For example, vector spaces would have been useful in the definition of the C-SVM and do exist as objects inside of categories, but are not necessarily categories themselves. Future development of the techniques shown may benefit from more nuanced constructions to increase their flexibility and descriptive power.

4.2. The Occupancy Dataset and Anomaly Prediction

The performance of the C-SVM algorithm highlights two issues with using the Occupancy dataset as a benchmark for anomaly prediction algorithms.

The first concern comes from its use in evaluating the performance of algorithms on time series data. Ten of the twelve included papers tested time-sensitive algorithms. However, the PCA analysis in Figure 1 and the ROCAUC of the C-SVM algorithm raised the concern that, effectively, none of the features required for successful classification are present in temporal information. The C-SVM was outperformed in accuracy by fifteen of the algorithms presented in relevant papers. However, with an accuracy difference of only 3.5%, it becomes unclear whether this improvement is due to time series information. For the purpose of investigating the performance of an algorithm on a time series dataset, this uncertainty impairs the utility of the Occupancy dataset as a benchmark. If the accuracy difference is entirely due to time series information, such a slight variation reduces the resolution of the dataset in differentiating the attributes of tested algorithms. This property of the dataset may lead to an improper evaluation of model performance when not accounted for: either by over-representing the performance of an algorithm, which ineffectively utilises time series information, or in the underperformance of models, such as LSTMs, whose additional connection weights increase the dimensionality and symmetries of their loss plane without conferring any significant benefit in this case.

The second concern can be seen in the data points generated by a broken sensor. These data points occur for both occupied and unoccupied classifications; however, considering the use of anomaly detection algorithms, it creates an issue. The Occupancy dataset in this context is a dataset with three classifications: occupied, unoccupied, and true anomaly. For this reason, the labels included in the dataset cannot be taken as ground truth classifications when validating anomaly detection algorithms. An algorithm may correctly identify the erroneous points as anomalies but be punished in its resultant score. In this way, it should be considered that the performance of such algorithms on this dataset is not an entirely accurate indication of their performance as anomaly detection systems. The C-SVM algorithm, due to its simplicity, is largely insensitive to the more nuanced forms of anomalies created by the broken sensor, contributing to its outperformance of comparatively more advanced systems.

The concerns identified with the Occupancy dataset do not necessarily mean it should be completely disregarded. Only in the cases of validating algorithms which are sensitive to time series information, or which identify anomalies, should these concerns be considered in their performance. In summary, it is suggested that the Occupancy dataset should be used with caution.

4.3. Uses of the Centred SVM

Though this paper has primarily utilised the C-SVM to demonstrate certain properties of the Occupancy dataset, it may also provide value in other applications. Its low time complexity allows it to be implemented as a component of a larger system, or as another tool for data analysis. The ability to control the constraining point provides the opportunity for manual selection, or for the point to be provided by another algorithm. Furthermore, the optimising algorithm presented may use any loss function relative to the dataset. Situations, where the separation between datasets is less clear, may cause issues as the hyperplane which maximises the distance to the nearest point may not generate a suitable result. Alterations such as using the distance to the k-th nearest point, the average distance

of k points, or maximising the distance variance (in which its operation becomes similar to the Fischer linear discriminant) may all be varied on a case-by-case basis.

5. Conclusions

The primary contribution of this report was the demonstration of a category theoretic approach to the analysis and presentation of machine learning problems. It was demonstrated that it is possible to describe a simple unsupervised anomaly detection algorithm, for a real-world problem, with Kan extensions. As a result, the design of the algorithm was directly informed by the characteristics of the dataset that were discovered in the initial data analysis. Not only was this algorithm competitive with those presented in related works but its Kan extension-inspired design was also able to add supporting evidence to claims about the characteristics of the Occupancy dataset. Namely, its classification labels have little relationship to the temporal component of the data and truly anomalous data points exist whose nature is not reflected by their classification, which are characteristics which should be considered before utilising the Occupancy dataset as a benchmark for other algorithms.

The development of category theoretic techniques may ultimately generate useful tools for the construction of machine learning algorithms, providing a perspective which is more concerned with the structure of data than the particular implementations of algorithms. Though Kan extensions have provided a promising indication of what these techniques may look like, there are many facets that a future work may improve.

Author Contributions: Conceptualization, M.P.; methodology, M.P.; software, M.P.; validation, M.P.; formal analysis, M.P.; investigation, M.P.; resources, M.P.; data curation, M.P.; writing—original draft preparation, M.P.; writing—review and editing, M.P., J.G., C.C. and N.H.; visualization, M.P.; supervision, J.G., C.C. and N.H.; project administration, N.H.; funding acquisition, N.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly funded by the grant "Early detection of contact distress for enhanced performance monitoring and predictive inspection of machines" (EP/S005463/1) from the Engineering and Physical Sciences Research Council (EP-SRC), UK, and Senseye.

Data Availability Statement: The "Occupancy Detection" dataset can be found at https://archive.ics.uci.edu/dataset/357/occupancy+detection (accessed on 28 February 2016).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shiebler, D.; Gavranović, B.; Wilson, P. Category Theory in Machine Learning. *arXiv* **2021**, arXiv:2106.07032.
2. Shiebler, D. Kan Extensions in Data Science and Machine Learning. *arXiv* **2022**, arXiv:2203.09018.
3. Candanedo, L.M.; Feldheim, V. Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models. *Energy Build.* **2016**, *112*, 28–39. [CrossRef]
4. Riehl, E. *Category Theory in Context*; Dover Publications Inc.: Mineola, NY, USA, 2016.
5. Fong, B.; Spivak, D.I. Seven Sketches in Compositionality: An Invitation to Applied Category Theory. *arXiv* **2018**, arXiv:1803.05316.
6. Leinster, T. Basic Category Theory. *arXiv* **2016**, arXiv:1612.09375.
7. Marteau, P.F. Random Partitioning Forest for Point-Wise and Collective Anomaly Detection-Application to Network Intrusion Detection. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2157–2172. [CrossRef]
8. Kerpicci, M.; Ozkan, H.; Kozat, S.S. Online Anomaly Detection with Bandwidth Optimized Hierarchical Kernel Density Estimators. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4253–4266. [CrossRef] [PubMed]
9. Ergen, T.; Kozat, S.S. Unsupervised Anomaly Detection with LSTM Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3127–3141. [CrossRef] [PubMed]
10. Ergen, T.; Kozat, S.S. A novel distributed anomaly detection algorithm based on support vector machines. *Digit. Signal Process.* **2020**, *99*, 102657. [CrossRef]
11. Sobhiyeh, S.; Naraghi-Pour, M. Online hypothesis testing and non-parametric model estimation based on correlated observations. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; ISSN 2334-0983.

12. Sobhiyeh, S.; Naraghi-Pour, M. Online detection and parameter estimation with correlated data in wireless sensor networks. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; ISSN 1525-3511.
13. Kithulgoda, C.I.; Pears, R.; Naeem, M.A. The incremental Fourier classifier: Leveraging the discrete Fourier transform for classifying high speed data streams. *Expert Syst. Appl.* **2018**, *97*, 1–17. [CrossRef]
14. Khalemsky, A.; Gelbard, R. A dynamic classification unit for online segmentation of big data via small data buffers. *Decis. Support Syst.* **2020**, *128*, 113157. [CrossRef]
15. Tan, S.Y.; Saha, H.; Florita, A.R.; Henze, G.P.; Sarkar, S. A flexible framework for building occupancy detection using spatiotemporal pattern networks. In Proceedings of the 2019 American Control Conference (ACC), Philadelphia, PA, USA, 10–12 July 2019; pp. 5884–5889; ISSN 0743-1619.
16. Candanedo, L.M.; Feldheim, V.; Deramaix, D. A methodology based on Hidden Markov Models for occupancy detection and a case study in a low energy residential building. *Energy Build.* **2017**, *148*, 327–341. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.