Additional file 1

Methods

Developing an algorithm for behavioural state classification

Feature characteristics

The algorithms in this study used two different components of the raw acceleration data: a static component and a dynamic component. The static component of the acceleration is caused by the orientation of the sensor relative to the gravity field of the earth and can be calculated as a running mean, to determine the body posture [1,2].

The dynamic component of the acceleration is caused directly by the movement of the animal. Isolation of the dynamic component of the acceleration can be obtained through the recently proposed overall dynamic body acceleration (ODBA) [3,4]. The ODBA, and its vectorial variation (VeDBA), are feature characteristics that have been used as a proxy of energy expenditure for animals with attached data loggers. This feature characteristic represents an aggregated acceleration measure for the subject [3,4]. Therefore, the dynamic body acceleration is a good candidate to discriminate between behaviours with high dynamic movement (such as feeding) from behaviours with low dynamic movement (such as standing or lying). Calculations of the static and dynamic component were computed according the derivations in [3,4].

The static component of the acceleration was calculated using a running mean. A simple running mean of a sequence of data points $\{a_i\}_{i=1}^{N}$is a new sequence of data points $\{s_i\}_{i=1}^{N-n-1}$ where each point is calculated by taking the arithmetic mean of the previous *n* points.

The static and dynamic body acceleration (DBA) can be calculated in two steps. Firstly, the static acceleration for each axis is derived using a running mean over the corresponding raw accelerometer data. Secondly, the resulting static acceleration is subtracted from the corresponding raw accelerometer data. This can be expressed as:

$DBA_i = A_{i_t} = \left|A^*_{i_t} - \mu_{i_t}\right|$, where $\mu_{i_t} = \sum_{t-win_{size}/2}^{t+win\_size/2} A^*_{i_t}/win\_size$ and $i = x, y, z$.

Here A_i, A_i^*, μ_{i_t} correspond to the static acceleration, raw accelerometer data and the running mean, respectively. The values obtained for DBA are then used to calculate the overall dynamic body acceleration (ODBA) and the vectorial dynamic body acceleration (VeDBA) as follows:

$ODBA = |A_X + A_Y + A_Z|$, and $VeDBA = \sqrt{A_x^2 + A_Y^2 + A_Z^2}$.

The first and second feature characteristics used in the decision-tree classification algorithm are calculated as the mean of the static component of the acceleration in the *y*-axis (SCAY) and the mean of VeDBA, respectively. Both means are taking over the duration of the window selected.

Machine learning algorithms

A full description of the machine learning algorithms used in this study is provided here:

k-means

A *k*-means algorithm is an example of a non-parametric classifier. In this method the data input set is partitioned into *k* predefined and mutually exclusive clusters. In other words given a set of *d*- dimensional observations $(x_1, x_2, .., .x_n)$ is partitioned into a set of predefined *k* number of clusters **S** = $\{S_1, S_2, .., S_k\}$, by minimising the within-cluster sum of squares. This can also be expressed with the function:

$\operatorname{argmin}_S \sum_{i=1}^{k} \sum_{x \epsilon S_i} \|x - \mu_i\|^2$, with μ_i the centroids of S_i.

The *k*-means algorithm can be computed through an iterative technique with the following steps:

1- Initial cluster seeds are chosen at random.
2- For each observation the distance to each cluster is calculated and then the observation is assigned to the closest cluster.
3- The new centroid of each cluster is calculated and each seed value is replaced with its respective centroid cluster.
4- The squared distance from an observation to a cluster and is computed and the observation is assigned to the cluster with the minimum squared distance.
5- Based on the new membership assignment, cluster centroids are recalculated.

The algorithm converges when the assignment no longer changes with the addition of observations. Several different measures can be used to calculate the squared distance from an object to a cluster but typically the Euclidean distance is used. Clusters formed by applying this algorithm are typically isolated elliptic regions. Note that although the *k*-means is a relatively simple algorithm, the recursive (iterative) procedure outlined above can be computationally expensive [5,6,7].

Hidden Markov model

A hidden Markov model (HMM) is a particular class of stochastic model in which a sequence of observations (visible) *X* is generated by a sequence of internal hidden states (not visible) *Y* [8,9,10,11,12]. The hidden states are assumed to follow a (first-order) Markov process (i.e. the probability of changing to a given state at any point in time is only dependent on the most recent state and not any earlier states). The transitions between hidden states can be specified by the transition probability *A* and a start probability vector π. In a HMM, hidden states emit the observations according to a probability distribution, also called the emission probability, which can be represented by B. Gaussian, multinomial and Poisson distributions are typical examples used for the emission probabilities in a HMM. A HMM can be fully described by (π, A, B). A HMM has three fundamental problems.

- Given the observational data only, how to find the best-fitting set of transition and output parameters. This problem is also known as learning and can be solved iteratively by a special case of expectation maximization, known as the Baum-Welch algorithm.
- Given the model parameters (π, A, B) and observational data, how to calculate the likelihood of the data conditional to the model. This is also

known as evaluation and can be solved using the forward-backward algorithm.

- Given the model parameters (π, A, B) and observational data, how to infer the most like sequence of hidden states that produce such observations. This problem is also known as decoding and can be solved by a dynamic programming algorithm known as the Viterbi algorithm [8,9,10,11,12].

Support Vector Machines

A support vector machine (SVM) algorithm is a supervised learning algorithm for binary classification which classifies data by finding the best hyperplane that separates the data points into two different classes. This hyperplane is constructed by maximising its distance from each training class [13]. New data is classified according to which side of the hyperplane they fit. Multiple classification can be achieved with SVM by constructing binary classifiers that can distinguish one class and the rest (called one-versus-all) or by constructing binary classifiers that can distinguish between every pair of classes (one-versus-one) [14,15]. The upgrade of a SVM binary classifier to a multi-class classifier is not easy and can result in classes ambiguously labelled [11].

SVM requires training and a testing process to be implemented. For these two processes, data has to be partitioned into a training set and a testing set. Results on the performance of the algorithm can depend on a particular random choice of the pair of sets. A possible solution to this problem is to use a model validation technique called cross validation. This technique partitions the data into complementary subsets where the training process is realised in one set and testing is performed in the other. Variability of the results with a cross validation technique can be reduced by running multiple partitions and averaging over the validation results. There are several types of cross validation, but in the present study we used a *k-fold* type where all the samples of the data are divided into *k* groups (called *k*-folds) of same size [11].

Decision-tree

Decision-trees are simple algorithms that use basic decision rules to classify the input data [11,16]. At each node of the decision-tree, rules partition the data into different clusters with each cluster containing samples with similar properties. Nodes are represented by the decision rules. The decision rules can be applied very quickly and are easy to interpret. In addition, decision-trees do not need any knowledge of the distribution of the data (non-parametric) and the number of leaves in the tree can be decided by the user. Leaves in the tree represent the labelled clusters, i.e., the behaviours. Typically a decision-tree uses one parameter per step. In our algorithm an unsupervised decision tree is used to discriminate and classify different behaviours of the dairy cows (Figure 2, in the main paper). In Figure 2 (main paper) each leaf represents a cluster and the tree consists of two decision rules, each using a different characteristic feature.

Decision-tree algorithm parameters and structure

Moving window size

When calculating the feature characteristics (VeDBA and SCAY) from the full time series of tri-axial acceleration data it is necessary to predefine both a window size to

smooth the data and the two thresholds (*A* and *B*) for each step of the decision tree. The choice of window size or the threshold values used can affect the performance of the decision-tree algorithm. To explore the effect of the choice of window size, the performance of the algorithm was investigated using the same input data and windows ranging from 1 - 600 seconds (window sizes above 600 seconds resulted in too few data points for a fair comparison of performance). In this initial exploration we used fixed values for the decision-tree thresholds: 0.0413g for threshold *A* and -0.055g for threshold *B*; these values were motivated by an exploratory study into algorithm performance relative to threshold value as described in the next section.

Figures S1 A) and B) show the decision-tree algorithm performance using the same input data for window sizes between 1 and 600 seconds. The results are presented in terms of sensitivity and precision, calculated by directly comparing the algorithm's classification of behaviour to the true observed behaviour at each time point. In each plot we also show the 'overall' sensitivity and precision which we define to be the arithmetic mean of the corresponding sensitivity or precision for each of the individual behaviours (feeding, lying or standing). Note that this definition of 'overall' sensitivity and precision is used in order to give equal weighting to each behaviour; we do not attempt to directly take into account the number of observations for each behaviour (see Table 1 in main paper). Figure S1 A) clearly shows that the classification sensitivity for feeding rapidly increases for window sizes of 1 to 60 seconds (from just below 60% to over 90%), and subsequently remains very similar for window sizes above 60 seconds with sensitivity ranging from 96% to 99%. The classification sensitivity of lying and standing changes less obviously with window size and there is low variation across all window sizes (sensitivity ranges between 78-88% for standing and between 73-79% for lying), although there is generally an upward linear trend in increasing sensitivity with increasing window size. As might be expected, the 'overall' sensitivity matches the trends observed for the individual behaviours (increasing from 64% to 84% as the window size increases from 1 to 60 seconds, and then ranging between 84% and 87% for window sizes larger than 60 seconds). The change in classification precision for feeding and lying show similar qualitative trends (Figure S1 B), with precision increasing with window size and ranging between 86-94% (feeding, window sizes above 60 seconds) and 94-97% (lying, window sizes above 60 seconds), respectively. In contrast, the classification precision for standing is quite low and ranges from 39-55% (with generally higher values with larger window sizes). Motivated by the results shown in Figure S1, we consider window sizes of 1 minute, 5 minutes and 10 minutes in the subsequent comparative study of algorithm performance.

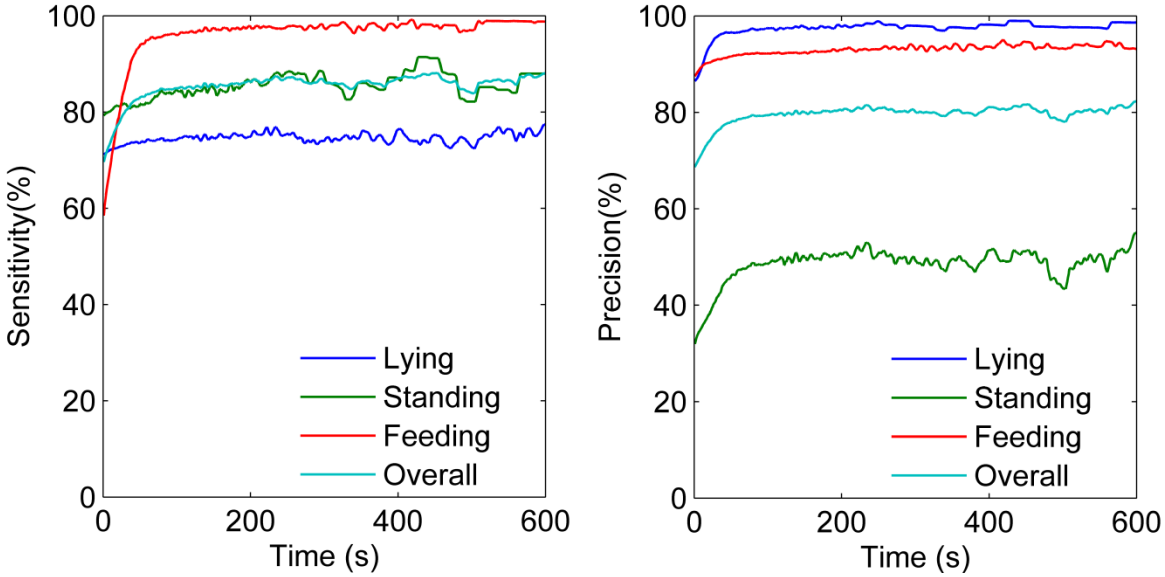

Figure S1. Performance of the decision-tree algorithm across a range of different window sizes. **A)** Sensitivity of the decision-tree algorithm with window sizes ranging between 1to 600 seconds. The overall sensitivity is defined as the arithmetic mean sensitivity of the three behaviours. **B)** Precision of the decision-tree using the same range of window sizes as for the sensitivity. Overall precision is defined in a similar way to overall sensitivity.

Threshold values for the decision-tree

The decision-tree algorithm contains two different thresholds (*A* and *B*) that need to be predefined, the choice of which can subsequently have an effect on the performance of the algorithm. We explored the performance of the decision-tree algorithm relative to the choice of thresholds using two different complementary approaches. In both approaches we used a 10-minute window size (see previous section).

In the first approach, optimal performance thresholds were calculated separately for each step of the decision-tree. Firstly, threshold *A* was used to discriminate between high (feeding) and low (lying or standing) activities through the VeDBA time series. The optimal value for threshold A of 0.0431g was obtained by exploring a receiver operating characteristic (ROC) curve. Figure S2 A), illustrates the ROC curve obtained by varying threshold *A* between -0.1 to 0.9g in increments of 0.001g. The plot shows the true positive rate (TPR =TP/TP+FN, also known as sensitivity) against the false positive rate (FPR= FP/FP+TN, also known as the false alarm rate) when varying the threshold. In this curve, the optimal threshold value is the one that generates a pair of TPR and FPR values closest to the top left corner, which represents the value that separates perfectly the positives from the negatives. The selection of the optimal threshold value *B* was based on a comparison of the overall TPR and overall FPR of the discrimination between lying and standing. Overall TPR (FPR) can be calculated as the mean of the TPR (FPR) for lying and TPR (FPR) for standing. All comparisons were performed using a 10-minute window size. Figure S2 B) shows the TPR and FPR values for lying and standing calculated when varying threshold *B* between -0.9 to 0.9g. The optimal value of threshold *B* was found to be -0.055g.

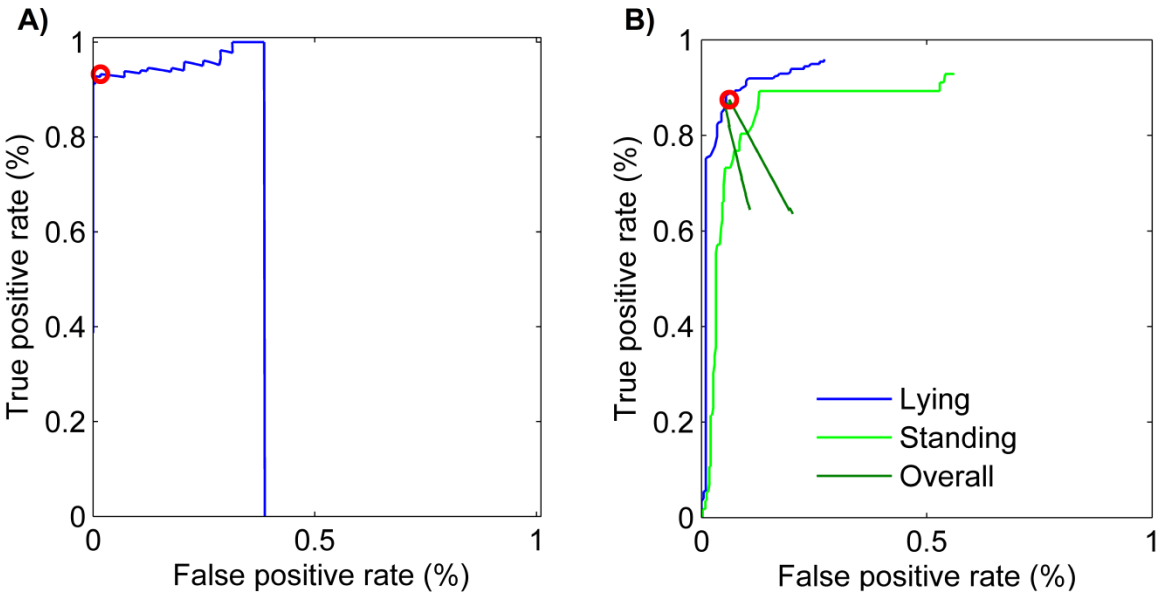

Figure S2. Receiver operating characteristic (ROC) curves to obtain optimal decision-tree threshold values. **A)** ROC curve displaying the true positive rate (TPR) against the false positive rate (FPR) when varying threshold *A* in a range between -0.1 and 0.9g. Threshold *A* helps to discriminate between high and low energy expenditure activities. The optimal threshold (in red), with value of 0.0413g, was selected by taking the closest point to the left top corner. **B)** Graphical display of the true positive rate against the false positive rate for lying, standing and overall. The rate of true positive and false positives was obtained by varying threshold *B* in a range between -0.9 and 0.9g. The overall TPR and overall FPR are defined as the mean of their respective rates for lying and standing. The optimal threshold (in red), with value of -0.055g, was selected by the closest point to the top left corner. Values for plots A) and B) were obtained using a 10-minute window.

The second method we considered for determining the best choice of threshold values selects both thresholds at the same by running a comparison of the performance of the algorithm over a 2-dimensional space range (Threshold *A* v Threshold *B*). Figure S3 A) displays the overall sensitivity of the algorithm using a 2-dimensional space range with values for threshold *A* between -0.3 and 0.3, and values for threshold *B* between 0.1 and 0.08; Figure S3 B) displays the precision for the same range of threshold values. Contours in this graph delimit the regions with the same overall classification accuracy. Figure S3 A) and B) illustrate contour plots of overall sensitivity and overall precision when threshold *A* (VeDBA) varies between 0.01 and 0.08g, and threshold *B* (SCAY) varies between -0.3 and 0.3g. Figure 5A (supplementary file) shows that the highest overall sensitivity (>82%) occurs when threshold A is in the range 0.03 to 0.05g and when threshold B is in the range -0.08 to 0.1g. Similarly, Figure S3B shows that the highest overall precision (>80%) occurs when threshold A is in the range 0.025g to 0.06g and threshold B is in the range -0.11 to 0.08g. The results obtained here were using a 10 minute window.

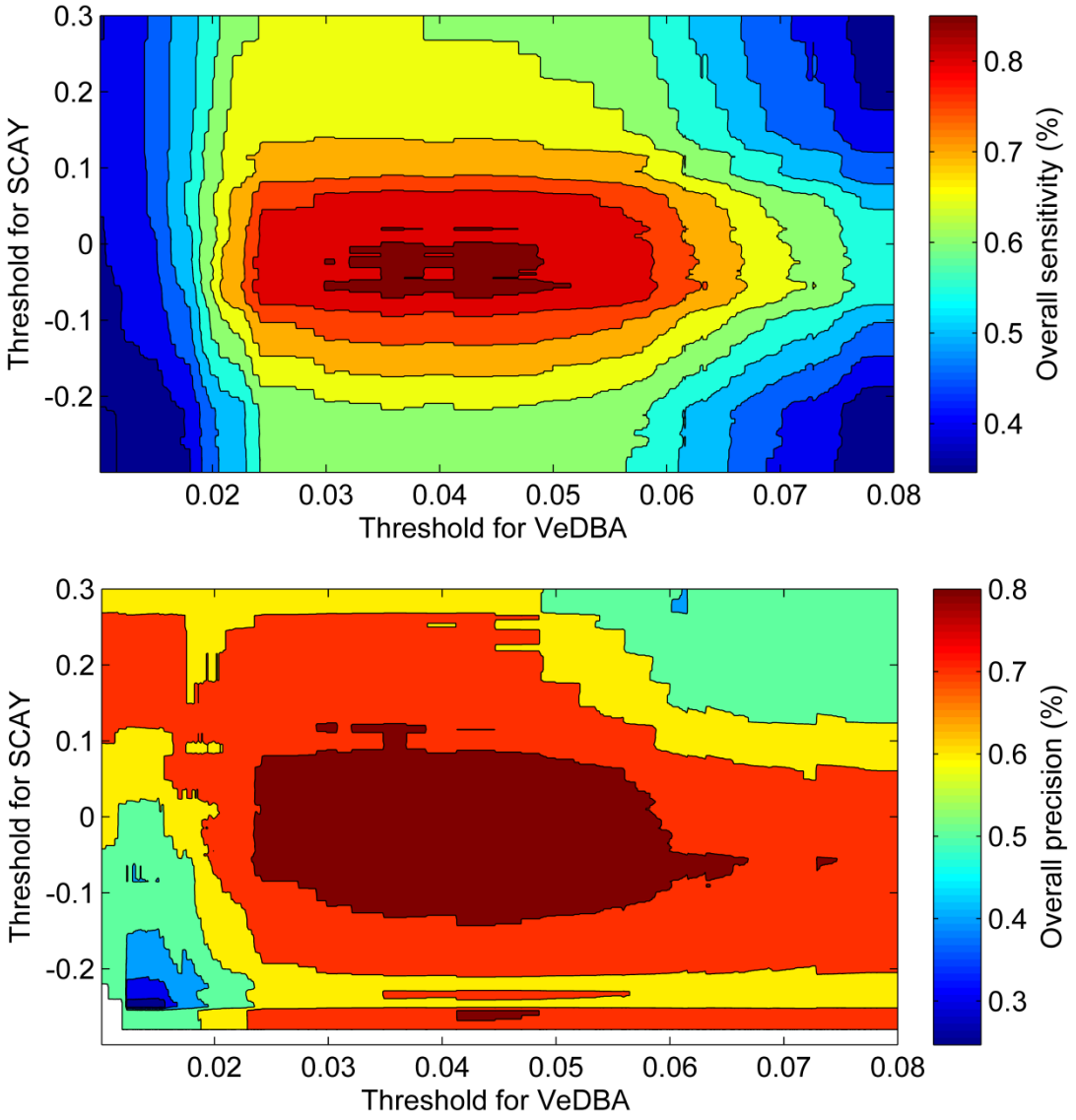

Figure S3. Performance of the decision-tree algorithm across a range of different threshold values. Performance of the algorithm was evaluated by varying both threshold A and B in the decision-tree algorithm at the same time. The range of variation was 0.01 to 0.08g for threshold *A* and -0.3 to 0.3g for threshold *B*. Contours illustrate regions with the same sensitivity or precision. **A)** Sensitivity of the algorithm using this approach. The region contained in the range between 0.3 to 0.5g for threshold *A* and -0.08 to 0.1g for threshold *B* resulted in sensitivity values of approximately 82%. **B)** Precision of the algorithm using this approach. A region contained in a range between 0.025 to 0.06g for threshold *A* and -1.1 to 0.8g for threshold *B* resulted in precision values of approximately 80%.It is clear that the region with the highest sensitivity matches the region with highest precision.

The threshold values obtained using the first approach (0.0413g for threshold *A* and -0.055g for threshold *B*) are contained in the regions with the highest overall performance obtained using the second approach (Figure S3). Hence, in the subsequent comparative study of algorithm performance we used these values for the thresholds in the decision-tree algorithm.

Algorithm structure

The impact of changing the order of the steps in the decision-tree (i.e., applying threshold *B* to the SCAY values before applying threshold *A* to the VeDBA values) was explored. The performance of the algorithm with this new structure was much

worse than that obtained with the original decision-tree structure: sensitivity was much lower (24.73% for lying, 52% for standing and 40.24% for feeding) compared with the original decision-tree (77.42% for lying, 88% for standing and 98.78% for feeding). Similarly, the precision was also much lower (92% for lying, 14.61% for standing and 38.37% for feeding) compared with the original decision-tree (97.95% for lying, 55% for standing and 88.06% for feeding). This comparison was made with a 10-minute window and the threshold values given in the previous section.

Classification performance across individual cows

The performance of an automated behavioural classification algorithm can often vary across individuals or breeds of the same species [17]. Hence we also consider the performance of the decision-tree algorithm at the level of the individual cow. In order to do this we computed the sensitivity and precision performance metrics for each individual cow at a window size of 1 minute (Table 3 in main paper). The 1 minute window was selected in this context to avoid having only a small number of samples for each individual cow (which can occur at larger window sizes).

An algorithm for detection of transitions between lying and standing

The algorithm developed to detect the transitions between lying and standing is a two-step algorithm. In the first step of the algorithm we detect when a transition occurs using a threshold value for the change in the range of acceleration in the *y*-axis without discriminating between standing up and lying down. The second step is performed by applying the decision-tree classification algorithm described previously to infer the anterior and posterior behaviour either side of the transition, and hence discriminate between standing up and lying down. The detection algorithm was applied to data extracted from the spreadsheet of aligned visual observations and accelerometer recordings. The data was extracted by taking 2 minutes of data either side of the time each transition was visually recorded. Data extracted for each transition was aggregated to form a unique time series with all the transitions included. The range of the *y*-axis was calculated on a window size of 8 seconds, selected based on the Nyquist sampling criterion and a minimal duration of 4 seconds for the transitions according to the visual recordings. The Nyquist sampling criterion establishes that the sampling frequency should be at least twice the most rapid movement that is necessary to characterise the behaviour [18]. A threshold value of 1.4g was selected for the range in *y*-axis. Selection of this value was performed by a comparison of the performance across different threshold values varying in a range between 1g to 3g in 0.1g increments. Performance of the detection algorithm was calculated using the sensitivity and precision at each step of the algorithm (Table 4 in main paper). This means that we first calculated sensitivity and precision of detecting a transition without including the type of transition (non-specific) and subsequently we included the type of transition (lying down or standing up). True positives (TP), false negatives (FN) and false positives (FP) were defined in a similar manner as previously described.

References

[1] Brown DD, Kays R, Wikelski M, Wilson R, Klimley AP: **Observing the unwatchable through acceleration logging of animal behaviour.** *Animal Biotelem* 2013, **1(1):**20.

[2] Yoda K, Sato K, Niizuma Y, Kurita M, Bost CA, Le Maho Y,Naito Y: **Precise monitoring of proposing behaviour of Adélie penguins determined using acceleration data loggers**. *J. Exp. Biol* 1999, **202:** 3121-3126.

[3] Qasem L, Cardew A, Wilson A, Griffiths I, Halsey LG, Shepard ELC, Gleiss AC, Wilson R**: Tri-axial dynamic acceleration as a proxy for animal energy expenditure; should we summing values or calculating the vector?** *PLoS One* 2012, **7:** e31187. doi:10.1371/journal.pone.0031187.

[4] Gleiss AC, Wilson RP, Shepard ELC. **Making overall dynamic body acceleration work: on the theory of acceleration as a proxy for energy expenditure.** *Methods Ecol Evol* 2011, **2:** 23-33.

[5] Bidder OR, Campbell HA, Gómez–Laich A, Urgé P, Walker J, Cai Y, Gao L, Quintana F, Wilson RP: **Love thy neighbour: automatic animal behavioural classification of acceleration data using the k-nearest neighbour algorithm.** *PLoS ONE* 2014, **9:** e88609. doi:10.1371/journal.pone.0088609.

[6] Vattani A: **k-means requires exponentially many iterations even in the plane.** *Discrete and Comput Geom* 2011,**45:** 596-616.

[7] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY: **An efficient k-means clustering algorithm: Analysis and implementation.** *IEEE Trans Pattern Anal Mach Intell* 2002, **24**: 881-892.

[8] Langrock R, King R, Mathiopoulus J, Thomas L, Fortin D, Morales JM: **Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions.** *Ecology* 2012 **93**:2336-2342.

[9] Guo Y, Poulton G, Corke P, Bishop-Hurley GJ, Wark T, Swain DL: **Using accelerometer, high sample rate GPS and magnetometer data to develop a cattle movement and behaviour model** *Ecol Model* 2009, **220:** 2068-2075.

[10] Rezek I, Roberts SJ: **Ensemble hidden Markov models for biosignal analysis.** In, *14th International Conference on Digital Signal Processing* 2002, Santori, Greece.

[11] Murphy KP: **Machine Learning:** a Probabilistic Perspective. MIT Press; 2012.

[12] Rabiner L: **Tutorial on hidden Markov models and selected applications on speech recognition.** *Proc. IEEE* 1989, **77:** 257-286.

[13] Cortes C, Vapnik V: **Support-vector networks**. *Mach Learn,* **20:** 273-297.

[14] Duan KB, Keerthi SS: **Which is the best multiclass SVM method?** *Lect Notes Comput Sc* 2005, **3541**: 278-285.

[15] Hsu CW, Lin CJ: **A comparison of methods of multiclass support vector machines.** *IEEE Trans Neural Netw* 2002, **13:** 415-425.

[16] Hastie T, Tibshirani R, Friedman J: **The Elements of Statistical Learning.** *New York: Springer Verlag; 2001.*

[17] Bailey DW, Van Wagoner HC, Weinmeister R. **Individual animal selection has the potential to improve uniformity of grazing on foothill rangeland.** *Rangl Ecol Manag* 2006, **59:** 351-358.

[18] Chen KY, Bassett DR Jr.: **The technology of accelerometry-based activity monitors: Current and future.** *Med Sci Sports Exer* 2005, **37:** 490-500.