Using K-NN, SVM to Classify Tactile Objects

Erik Larsen

December 11th 2017

1 Introduction

There are multiple ways to classify data using machine learning. Here, two widely-used algorithms, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM), are employed and compared.

K-Nearest Neighbors can be a supervised or an unsupervised machine learning algorithm. In this study, it will be supervised. K-Nearest Neighbors essentially classifies data into a group, given some features, labels, and some general parameters [**2**]. This can be implemented geometrically, depending on the number of features in the dataset: given two features with discrete values, KNN serves to classify new data points based on how close to a classified group that new point is, using, for example, Euclidean distance [**3**].

Although visual representation is simplest, imagine a doctor wants to diagnose whether Patient X's skin growth is malignant or benign. Using KNN, the doctor could take previously-acquired (trained) data on tumors designated as malignant or benign based on the radius of the growth and how much the tumor protrudes above the skin (concavity). The doctor can measure Patient X's tumor and include it on a 2D plot (visually representing the KNN algorithm), providing a quick screen as to what Patient X's growth may be by determining where that new tumor falls in relation to the previous (trained) data.

KNN, in this case, would be trained to take a pre-determined number of points nearest to Patient X's and decide (neighbors), based on shortest distance to those neighbors, whether the tumor is malignant or benign, assuming that data can be classified based on similar features: in other words, closer data points are likely to be more similar and therefore may be classified as such.

This raises interesting questions that are wrestled by data analysts: how many neighbors (k) in a dataset should be used to classify a new data point? Is it simply a "majority rules" system of classification with these neighbors? Are they weighted equally? Can this method be extended into a feature-space that cannot be depicted, graphically? How many groups can be used? Can groups overlap?

Cross-validation is commonly used to determine how many neighbors to employ by comparing performance of how many mistakes were made in classification for different neighbor conditions [**3**].

There are different variations of KNN in terms of its classification system, but generally, the "majority rules" approach (uniform weights) is employed, where the class with the most neighbors nearest the new data point determines the new data point's class. However, weighted distance can be employed: weight is given to the magnitude of the distance and use the corresponding aggregate values to determine class [**3**], possibly negating ties in classifiers with even-numbered ks. If, in the tumor classification example, a $k = 5$ system is used, and one benign neighbor is more than four times closer to Patient X's point than the sum of the other three malignant neighbors, Patient X's tumor would be classified as benign. The equation for such a KNN can be described as []:

$$\mathrm{t}_{q_i} = \frac{1}{\mathbf{d_{t_i}}^{\mathbf{2}}}, \qquad (Eq.1)$$

where:

$$\mathbf{d_t} = \sum_{i=1}^{k_t} \sqrt{(q_i - p_{i_t})^2} \qquad (Eq.2)$$

Here, t_q represents the predicted classes of query points, q. $\mathbf{d_t}$ represents the vector of aggregate Euclidean distances between each query point and all the neighbors of their respective classes, p_{i_t}. To normalize the distances between each query point and its neighbors so that the closest neighbors, k, of a given class, t, carry more weight than neighbors further from the query points, the inverse square of the distance, $\frac{1}{\mathbf{d_t}_\mathbf{i}^\mathbf{2}}$, is taken, assigning the class, t, of each query point, q, as the highest quantity within the vector.

KNN class numbers and shapes can vary widely. There can be overlap between classifications— a different class with an area around it can be found within an area predominantly describing another class— and the number of classes and the decision boundaries governing the classes can be enormous and irregular, respectively [**5**], making KNN an ideal tool to use for this paper.

A Support Vector Machine is another kind of supervised machine learning algorithm that can be used to classify data [**2**], whose decision boundary line separating data (the support *vector)* is governed by the following equation [**3**]:

$$\mathrm{t}_n(\mathrm{w}^\top \mathbf{x}_n + \mathrm{b}) \geq 1 - \zeta_n \qquad (Eq.3)$$

Here, t_n represents the class boundary the SVM predicts; $\mathbf{w}^\top \mathbf{x}_n$ constitute the width of the boundary (vector) based on the data on which the SVM is trained; b is the "slope," "shape," or "orientation" of the boundary with respect to the feature axes; ζ_n is a "slack" term allowing for data to be classified on the "wrong side" of the boundary, scaled by a factor not shown, C, that incorporates a relative number of training data into the shape and orientation of the boundary. The specific points involved in the boundary's shaping is called the support.

In a linear sense, SVMs take data and provide a boundary whereupon data can be classified. A more specific SVM that decides multiple classes employs multiple boundaries, otherwise an SVM can decide a "one-versus-all" boundary (*line*) that classifies data into one category and everything on the other side of that boundary (hyperplane, if classifying based on numerous features) is considered not that category [**2**].

Disadvantages of a KNN are that by changing the k, class labels can change [**5**]. It also suffers from being a "lazy learner," where it simply uses the training data for classification. This requires additional data points to be classified (tested) by sorting through all the training data for each new point; it also may not generalize well [**5**]. In comparison, SVMs may also not generalize well [**5**], based on hyper-parameter selection. Another similar problem to KNN is the difficulty in choosing an appropriate kernel, as choosing k may be in KNN [**2**].

A common problem for limb amputees is the ability to manipulate objects with their hands. Using their hands to do this involves sensory feedback about the object they are manipulating— feedback they lack, having lost the requisite nerves in their hand. One way to supplement this vital deprivation is to provide a new modality: electrically stimulating coded information about the parameters of objects to be grasped in an area where these subjects *can* feel is promising. The theoretical paradigm that generated the data used in this study was as follows: 27 different objects were classified based on the levels (low, medium, high) of their physical parameters (width, rigidity, weight) and fed into an electrical stimulus pattern that a subject was tasked to associate blindly.

A machine learning algorithm would be a vital control for humans' performance on classifying data across days and trials to determine whether an electrical stimulus feedback system is a viable tactile substitute. Here, the goal is determine the performances of KNN and SVM supervised machine learning algorithms on new data and assess whether one would be better suited for the aforementioned experiment.

2 Procedure

Data Acquisition

The experiment conducted to acquire the electrical stimulation data was as follows:

Custom force transducers were fixed to the thumb and index finger of a golf glove, sensing pressure between the two digits. Polhemus LIBERTY electromagnetic hand micro-sensors were fixed onto the backside of the finger and thumb to detect the change in aperture of thumb and index finger. Signals from both were transmitted to a MATLAB2017a program to convert the data into values fed into a current pulse generator in real time to generate electrical stimulus pulses. Pulses were sent to surface electrodes on the neck of a blindfolded subject, detecting the change in patterns when 27 different objects were each picked up and squeezed by the experimenter wearing the golf glove. The subject was tested on each object 3 times per day over 5 total days for a total 15 repetitions of each unique object, totaling 405 trials. The three measured features were each object's weight (force required to hold), compliance (rigidity; change in aperture), and width (aperture at onset). (Data was acquired by Spike2 Version 7.07 Spectrum Analyzer software, and processed in excel before use in machine learning algorithms.

It should be noted that compliance required three measurements, alone: the aperture change between initial touch of the object and holding the object above a surface, the aperture change once held steady, and the aperture change from a steady hold throughout a squeeze.

Algorithms

Since the classification of the objects was known beforehand, supervised machine learning algorithms, a KNN and a SVM, were compared to assess the proper utilization in this study: whether the objects, and thereby, the stimulation paradigm, were distinct enough to be used for classification. Thus, the algorithms could help determine whether the stimulation paradigm is feasible for use by insensate humans. Grid search was performed for both algorithms via scikitlearn to determine the optimal parameters to train and test the accuracy of the algorithms (based on one-vs-all since there were 27 unique objects, 405 in total).

As previously mentioned, a KNN relies upon the distance from (a) test point(s) to the k-numbered nearest neighbors to determine its classification [**3**]. In this experiment, weight was not uniformly given to each neighbor in the KNN algorithm as a grid search maximized the parameters to determine the optimum number of neighbors: more weight was given to training points nearer to the data upon which the algorithm was tested.

An SVM relies upon the shape, width, and "flexibility" of its decision hyperplane [**3**]. In this experiment, different values for C, the scaling factor allowing for the rigidity of a decision boundary [**3**], were also run through and maximized (along with the boundary itself, given by Eq. 3) via grid search.

The values for the number of nearest neighbors (k) to train the algorithm were run through a "grid search," a parameter optimization function. The values ranged from 2 - 7, roughly half of the total number of each unique object in the experiment. The range of k included the number of repetitions of unique objects the algorithm would be tested on if the data were split by number of days, and trained on 1 day, as would be the case in 5-Fold CV (10). The range also included number of times each object was seen per day, 15, and the number of unique objects, 27. the number of unique objects to determine if the algorithm would require more than the total number of each unique objects, or an example of each to be accurate.

An exhaustive search along a continuous range for the ideal nearest neighbor was not conducted for practicality and to parallel the range of parameters for the "slack" term, C, in the SVM model, which were determined over a range of orders of magnitude. Since the relative "rigidity" of the boundary increases as C and ζ_n approach 0, values ranged from near 0 (0.01) to 100. A more rigid boundary includes more data points that determine the boundary; a less rigid boundary involves fewer training data to orient the direction and width of the boundary, allowing for more flexibility [**3**].

Cross-validation was employed to find the best parameters for each model (k, C, respectively) [**3**]. However, multiple types of cross-validation were employed, namely, Leave-One-Out Cross-Validation (LOOCV), and K-Folds Cross-Validation (KF CV). In LOOCV, all the data except one sample is used to train the model; the last sample tests it [**1**]. Therefore, one may assume a larger amount of training data would yield a more accurate classifier— much moreso than KF CV, which partitions the data into equal *folds*. The drawback of LOOCV is time: the algorithm must be run through each time to train on all but one of a large dataset, testing on the other, and iterating through all the permutations, averaging the number of errors of all the permutations to find the best parameters [**1**]. However, since the size of the dataset in this study (405) is relatively small, computation time would be a negligible factor if both models were classifying based on one feature. Both KNN and SVM were trained on different features to determine which, if any, was most identifiable. Parameter optimization results are displayed in **Tables 1, 2**.

Thus, both a KNN and SVM were used with LOOCV and KFCV (5 folds) to determine how many of the 405 objects could be correctly classified into their either their 27 groups, which combines all the available features, or based on their weight, or compliance, or width. The optimal parameters for both KNN and SVM were chosen based on the highest 5-fold KF CV grid search score because objects were presented the same number of times over 5 different days, thus making the most natural partition of the data, even if the parameters determined by this cross-validation did not yield the highest accuracies.

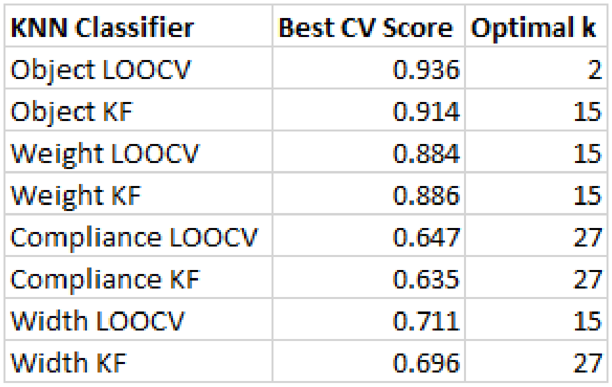
Each model was re-fit to the data based on the aforementioned methodology (and parameters): an SVM with a C of 10, and a KNN with a k of 15, and a confusion matrix was generated. Accuracy was assessed based solely on the proportion of correct classifications to the total number of objects being tested (324).

3 Evaluation

Supervised learning algorithms are somewhat simple to evaluate, given that the true classification of the data that is fitted and test is already known [**2**]. As mentioned above, cross-validation was used to optimize parameters for different classifiers in two different algorithms over a similar range of parameters. Once selected, the data was re-fit to train and test each algorithm's best condition to produce a confusion matrix (**Figures 1, 2**). The accuracies for each were determined as the proportion of correct classifications to the total number of objects tested from these matrices.

4 Results

Table 1 shows the most optimal KNN classification parameters. The most accurate model, trained and tested via LOOCV, achieved a 93.6% accuracy, using only two neighbors to identify a new data point, based on the uniqueness of the object, which factored in the object's range of weight, compliance, and width features **Table 1**.



KNN Classifier	Best CV Score	Optimal k
Object LOOCV	0.936	2
Object KF	0.914	15
Weight LOOCV	0.884	15
Weight KF	0.886	15
Compliance LOOCV	0.647	27
Compliance KF	0.635	27
Width LOOCV	0.711	15
Width KF	0.696	27

Table 1: Optimization of the parameters for K-Nearest Neighbor classifiers for one of four classifications (Object, Object Weight, Object Compliance, Object Width), with either Leave-One-Out or K-Fold (k = 5) Cross-Validation, over a range of neighbor sizes, 2-7, 10, 15, 27.

The most optimal parameters for the SVM algorithm, cross-validated by LOOCV, had a very flexible decision boundary, with a C of 100 **Table 2**. Its accuracy score was 94.3%.

SVM Classifier	Best CV Score	Optimal C
Object LOOCV	0.943	100
Object KF	0.916	10
Weight LOOCV	0.886	1
Weight KF	0.884	1
Compliance LOOCV	0.662	100
Compliance KF	0.654	10
Width LOOCV	0.721	1000
Width KF	0.701	100

Table 2: Optimization of the parameters for Support Vector Machine classifiers for one of four classifications (Object, Object Weight, Object Compliance, Object Width), with either Leave-One-Out or K-Fold (k = 5) Cross-Validation, over a range of boundary flexibilities 0.01 - 1000 each increasing by a factor of 10.

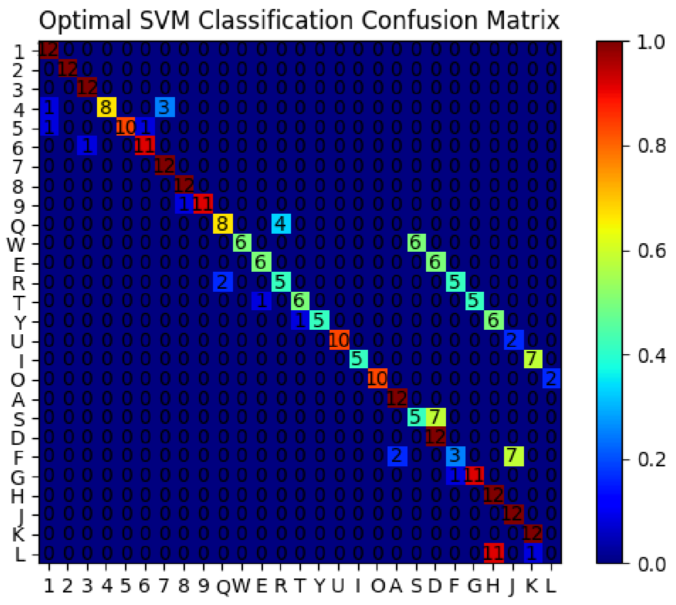


Figure 1: Confusion matrix of the most optimal SVM classifier, classifying an object (with a C of 10) based on which group to which it belongs, each having a unique combination of weight, rigidity, and width. Diagonals indicate correct classifications (% Correct = 74.1)

The optimal parameter results are not too surprising, given that the most accurate models were based on (object) classification that incorporated all three features to make it unique: more data, in this case, yielded more accurate classifiers.

The accuracy results (**Figures 1, 2**; KNN = 76.5%, SVM = 74.1%) are somewhat expected, as well: each algorithm is comparable at a classification for which it was designed . However, to correct group an object based on the combination of each features intuitively seems difficult, but as mentioned before, more data should yield higher accuracy. An even more exhaustive study would show the performances of each model through each classification based on their optimized parameters and then compared, regardless of "realism" in data parsing, or expense. Explicitly, if the data could be easily reorganized to have each object evenly distributed throughout the dataset to then be iteratively trained and tested on, then the results would be more interesting.

To have a KNN algorithm with a k of 15, the number of objects in each of 27 groups, be an optimal parameter is none too surprising: the algorithm needed 15 neighbors to determine which object was which. Yet, given the true nature of the clustering of these objects, this can be understandable: **Figure 3** shows how the objects group along the feature-space and there is considerable overlap and variability between and within objects (each color represents a unique object based on all the grasps using the electro-tactile system throughout an experiment). It is also understandable why such a high C was necessary: it is difficult to imagine a hyper-plane weaving through all the objects; yet it was flexible enough to correctly predict an "unknown" object nearly three-quarters of the time. This figure also yields some insight into which features would likely have been the most easily identifiable, and could explain why compliance did so poorly in cross-validation.

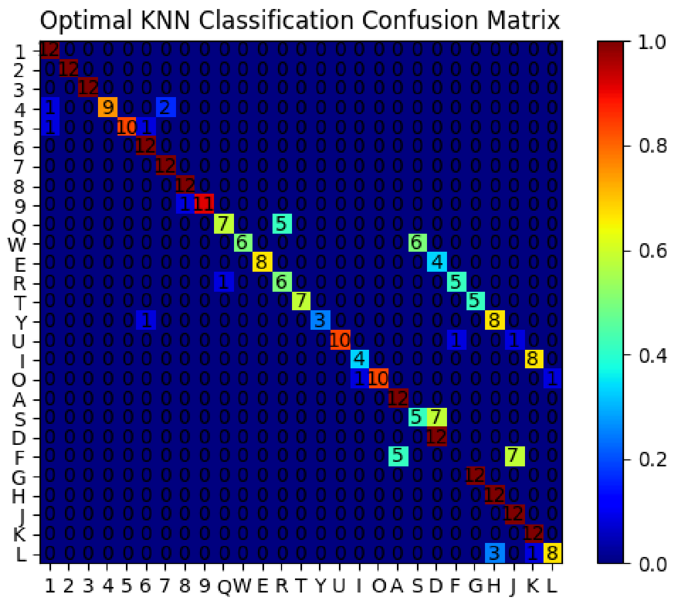


Figure 2: Confusion matrix of the most optimal KNN classifier, classifying an object (with a 5 of 15) based on which group to which it belongs, each having a unique combination of weight, rigidity, and width. Diagonals indicate correct classifications (%Correct = 76.5)

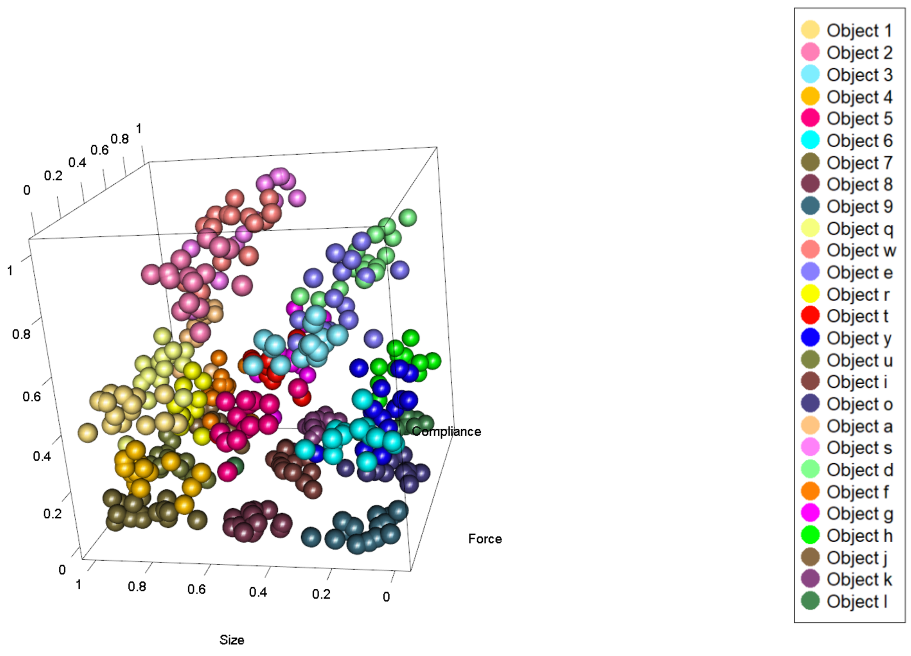


Figure 3: Graphical representation of 27 unique objects, parameterized by discrete and equal values of weight (force), rigidity (compliance), and width (size). Each object contains 15 separate trials of grasping and holding, and each delivered a corresponding strength of electrical stimulation, all normalized between 0 and 1.

In conclusion, it is reasonable to determine either algorithm could be used to faithfully classify tactile objects, as the objects truly cluster fairly well: one could assume tighter clustering would have yielded higher accuracy scores than 75% for each model and smaller parameter values. A future, more rigid experiment would be to train and test these algorithms on objects with qualities that are less distinguished.

References

[1] James, Gareth, D. Witten, T. Hastie, R. Tibshirani. "An Introduction to Statistical Learning with Applications in R." 7th Edition. Springer Science and Business Media. 2013.

[2] Lantz, Brett. "Machine Learning with R." 2nd Edition. Packt Publishing. 2015.

[3] Rogers, Simon, Mark Girolami. "A First Course in Machine Learning." 2nd Edition. CRC Press. 2017.

[4] SciKit-Learn Developers. "Support Vector Machines." scikit-learn: Machine Learning in Python. SciKit-Learn Developers. 2017. `http://scikit-learn.org/stable/modules/svm.html#multi-class-classification`

[5] Vanderplas, Jake. "Nearest Neighbors." scikit-learn: Machine Learning in Python. SciKit-Learn Developers. 2017. `http://scikit-learn.org/stable/modules/neighbors.html#choice-of-nearest-neighbors-algorithm`