Silhouettes and Quasi Residual Plots for Neural Nets and Tree-based Classifiers

Jakob Raymaekers & Peter J. Rousseeuw

View supplementary material

Published online: 04 Apr 2022.

Submit your article to this journal

Article views: 1160

View related articles

View Crossmark data

Citing articles: 1 View citing articles

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS

Check for updates

Silhouettes and Quasi Residual Plots for Neural Nets and Tree-based Classifiers

Jakob Raymaekers and Peter J. Rousseeuw

Section of Statistics and Data Science, Department of Mathematics, KU Leuven, Leuven, Belgium

ABSTRACT

Classification by neural nets and by tree-based methods are powerful tools of machine learning. There exist interesting visualizations of the inner workings of these and other classifiers. Here we pursue a different goal, which is to visualize the *cases* being classified, either in training data or in test data. An important aspect is whether a case has been classified to its given class (label) or whether the classifier wants to assign it to a different class. This is reflected in the (conditional and posterior) *probability of the alternative class* (PAC). A high PAC indicates label bias, that is, the possibility that the case was mislabeled. The PAC is used to construct a *silhouette plot* which is similar in spirit to the silhouette plot for cluster analysis. The average silhouette width can be used to compare different classifications of the same dataset. We will also draw *quasi residual plots* of the PAC versus a data feature, which may lead to more insight in the data. One of these data features is how far each case lies from its given class. The graphical displays are illustrated and interpreted on datasets containing images, mixed features, and tweets. Supplementary materials for this article are available online.

1. Introduction

Classification by neural nets and by tree-based methods are powerful tools of machine learning. Hastie, Tibshirani, and Friedman (2017) give a broad overview of classification methods. There exist interesting visualizations of the inner workings of classification by neural nets; see, for example, Shahroudnejad (2021) and the references cited therein. Classification trees such as those produced by CART (Breiman et al. 1984) and the corresponding R package `rpart` (Therneau, Atkinson, and Ripley 2019) can plot the tree and list its decision rules, which makes the classification explainable. The random forest classifier (Breiman 2001) can be understood as an ensemble of such trees.

In this article we propose visualizations of the *cases* being classified, rather than the mechanism of the classifier. We are convinced that visualizing the objects being classified is equally useful, and can reveal different and very relevant aspects of the classification task. The visualization of cases is well-established for regression tasks, with for example, residual plots, whereas it is lagging behind for classification. The purpose of the new visualizations is to draw our attention to interesting aspects of the data that we might not have known otherwise or did not expect. Examples are the strength of the classification per label, noticing patterns in subsets of the data, detecting mislabeled instances, and discovering overlap between classes. Interpreting such clues may provide insight into the structure and quality of the data. The graphical displays can also reveal underlying causes of misclassifications, telling us something about the appropriateness of the classifier. This will be illustrated in various examples throughout the article.

In classification, a key concept is the conviction with which an observation is assigned to its own class or a different class. This information is captured by the probability of the alternative class (PAC) which is fundamental to our visualizations. We use the PAC to construct a *silhouette plot* which is similar in spirit to the silhouette plot for unsupervised classification (Rousseeuw 1987). The average silhouette width (on test data or cross-validated) can be used to compare different classifiers applied to the same dataset. We will also draw *quasi residual plots* of the PAC versus a data feature. Patterns in such plots may reveal interesting trends, and may help unearth factors explaining why some cases are easier to classify correctly than others, potentially informing model choice.

In Section 4.1 we will focus on neural nets, and analyze the results of a classification of images from 10 categories. Section 4.2 applies the general principles to classification by CART, and Section 4.3 does the same for random forests, each illustrated on a well-known dataset. Section 5 concludes and describes the available software.

2. Silhouette Plots for Classification

The silhouette plot of Rousseeuw (1987) is a graphical display of a clustering (unsupervised classification) in k clusters. The silhouette width

$$s(i) := \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)} \qquad (1)$$

describes how well case i is clustered. Here $a(i)$ is the average interpoint dissimilarity of case i to the members of its own

cluster. In contrast, $b(i)$ is the smallest average dissimilarity of case i to a nonself cluster. That cluster can be considered the "best alternative" cluster for case i. From (1) we see that $s(i)$ is between -1 and 1. When $s(i)$ is high (close to 1) it means that case i has much more in common with its own cluster than with any other cluster, so it was clustered well. On the other hand, an $s(i)$ close to -1 means that case i would much prefer to be assigned to its best alternative cluster.

The silhouette plot displays the s(i) values, ranked in decreasing order in each cluster. The silhouette of a cluster reflects how well its members are clustered. The left panel of Figure 1 shows the silhouettes of a partition with $k = 3$ of a toy dataset. The $s(i)$ are on the horizontal axis. For instance, in the second cluster from the top the cases range from well-clustered (high $s(i)$) to poorly clustered (low $s(i)$). The silhouette of the third cluster is the widest, and indeed the average of its $s(i)$, shown on the left as $\bar{s} = 0.75$, is the highest. The *overall average silhouette width* of 0.63 listed at the bottom is the average $s(i)$ over all cases i in the dataset. Usually the number of clusters k is not given in advance, and then one often selects the value of k that makes the overall average silhouette width the highest.

In our setting of supervised classification, the situation is somewhat different. Not all classifiers are based on interpoint dissimilarities; in particular, neural nets combine the outputs of cells in neurons, and tree-based classifiers use variable splits. We denote a class (label, group) by the letter g, with $g = 1, \ldots, G$. Consider a case $i = 1, \ldots, n$ in the training set or a test set. Typically, a classifier will provide posterior probabilities $\hat{p}(i, g)$ of object i belonging to each of the classes g, with $\sum_{g=1}^{G} \hat{p}(i, g) = 1$ for each i. The object i is then classified according to the *maximum a posteriori* rule

$$\text{assign object } i \text{ to class } \operatorname{argmax}_{g=1,\ldots,G} \hat{p}(i, g) . \quad (2)$$

Now consider the object i with its known given label g_i. In analogy with the $s(i)$ above, we want to measure to what extent the given label g_i agrees with the classification of i. For this purpose we define the highest $\hat{p}(i, g)$ attained by a class *different from* g_i as

$$\tilde{p}(i) := \max\{\hat{p}(i, g) ; g \neq g_i\} . \quad (3)$$

The class attaining this maximum can be seen as the best alternative class, so it plays the same role as the class yielding $b(i)$ in clustering. If $\hat{p}(i, g_i) > \tilde{p}(i)$ it follows that g_i attains the overall highest value of $\hat{p}(i, g)$, so the classifier agrees with the given class g_i. On the other hand, if $\hat{p}(i, g_i) < \tilde{p}(i)$ the classifier will not assign object i to class g_i.

We now compute the conditional posterior probability of the best alternative class when comparing it with the given class g_i as

$$\text{PAC}(i) = \frac{\tilde{p}(i)}{\hat{p}(i, g_i) + \tilde{p}(i)} . \quad (4)$$

The abbreviation PAC stands for Probability of the Alternative Class. It always lies between 0 and 1, and smaller values are better than larger values. When $\text{PAC}(i) < 0.5$ the classifier does predict the given class g_i, whereas $\text{PAC}(i) > 0.5$ indicates that the best alternative class outperforms g_i in the eyes of the classifier. $\text{PAC}(i) \approx 0$ indicates that the given class fits very well, and $\text{PAC}(i) \approx 1$ means the given class fits very badly. The PAC can

be seen as a continuous alternative to the more crude distinction between "correctly classified" and "misclassified" that is used in the misclassification rate.

In order to draw the silhouette plot of a classification, we put

$$s(i) := 1 - 2\,\text{PAC}(i) . \quad (5)$$

Like (1) this $s(i)$ again ranges from -1 to 1, with high values reflecting that the given class of case i fits very well, and negative values indicating that the given class fits less well than the best alternative class. The actual silhouette plot is then drawn as before, for example in the right panel of Figure 1 with $G = 4$ classes shown in different colors. The fact that the $s(i)$ have a continuous range allows us to see finer detail than if we would only display whether a case is classified correctly or not.

The data and the classifier leading to this plot will be explained in Section 4.3, but the display alone already tells us a lot. The silhouettes have unequal heights, which are proportional to the number of cases in each given class. Each class has several cases with high $s(i)$, that are predicted strongly in it, but also some cases with negative $s(i)$, which the classifier predicts in a different class. The overall silhouette width is 0.79 . Class "fear" has the highest average silhouette width ($\bar{s} = 0.84$), similar to that of class "joy" (0.82), and followed by classes "anger" (0.76) and "sadness" (0.71). This indicates that classes "fear" and "joy" are fit best by this particular classifier.

In supervised classification the number of classes G is known in advance, so the overall average silhouette width cannot play the same role as in cluster analysis, where it is used to select the number of clusters. But when different classifiers are applied to the same dataset, it measures the quality of each classification, so one could select the classification with the highest overall average silhouette width.

3. Quasi Residual Plots

Another graphical display is obtained by plotting the PAC versus a relevant data variable. This is not unlike plotting the absolute residuals in regression, since small values of $\text{PAC}(i)$ indicate that the model fits the data point nearly perfectly, whereas a high $\text{PAC}(i)$ alerts us to a poorly fitted data point. We call it a *quasi residual plot* because of this analogy. The data feature on the horizontal axis does not have to be part of the classification model, and it could also be a quantity derived from the data features such as a principal component score or a prediction, or just the index i of the data point if the data were recorded sequentially.

Figure 2 shows two quasi residual plots. The data will be described in Section 4.1, and consists of 10,000 color images with $G = 10$ classes. The classifier has operated on the pixels of each image, yielding the PAC on the vertical axis. The variables on the horizontal axis were not in the classification model. The left panel plots the PAC versus the intensity of each image, which was averaged over all pixels and the three colors (red, green, and blue). Note that the bottom half of the plot has a light gray background. Points in this region have $\text{PAC}(i) < 0.5$, meaning that they are predicted in their given class, whereas the classifier would put the points in the top half in a different class.

Since the density on the horizontal axis is far from uniform, three curves were superimposed. The red curve is the average
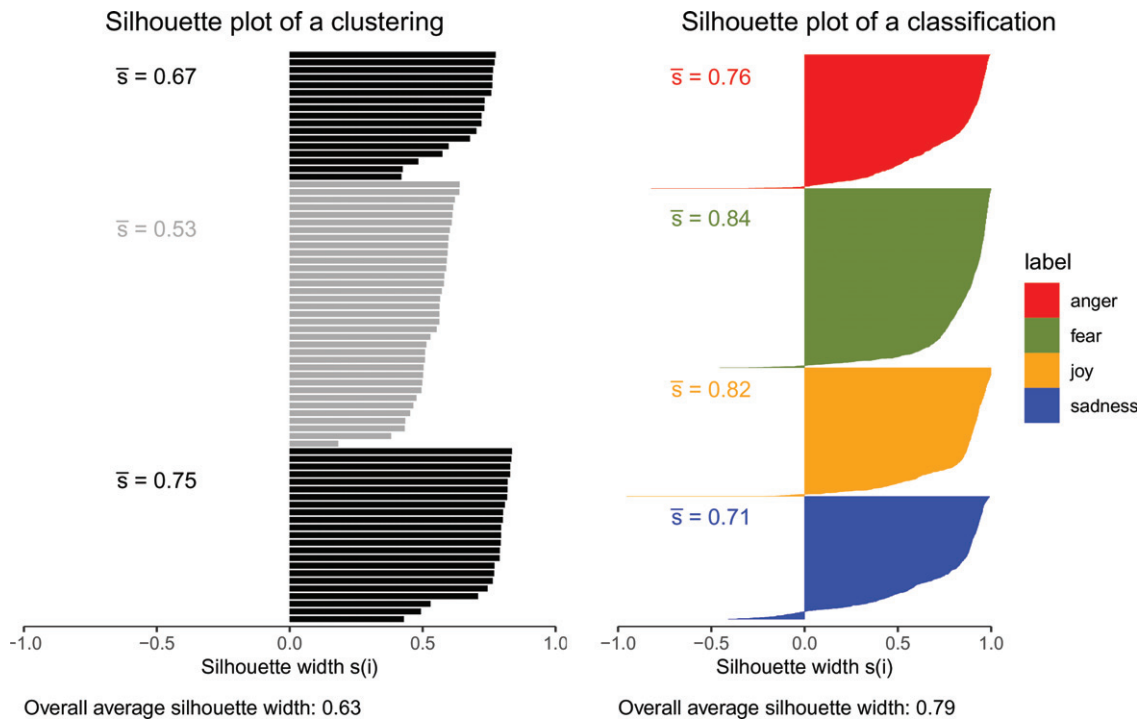
Figure 1. Silhouette plots of (left) a partitioning of a toy dataset in three clusters, and (right) a classification with four classes.
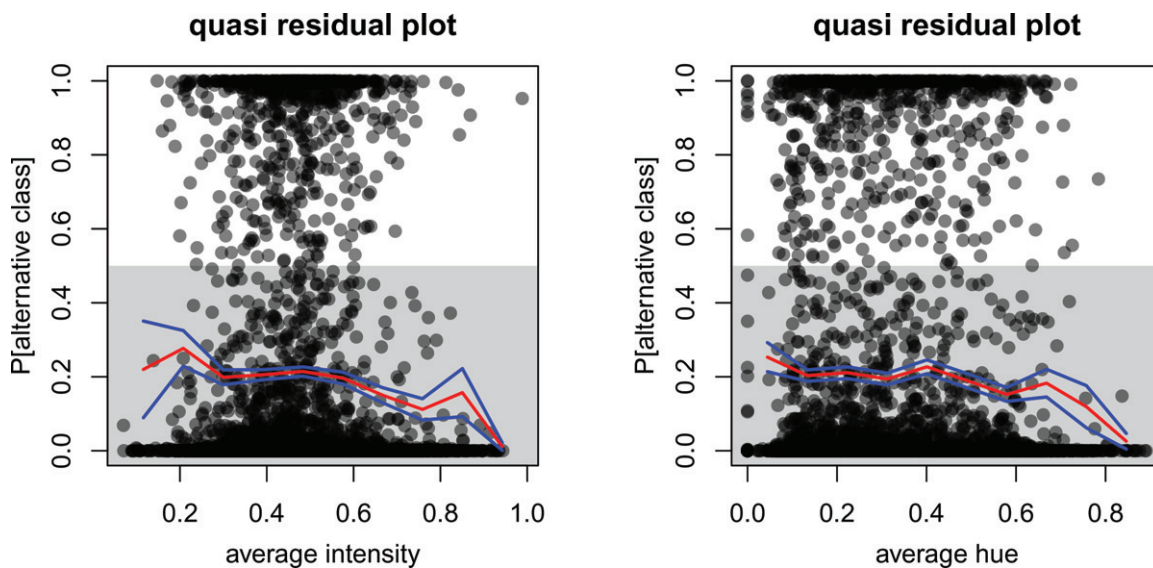

Figure 2. Quasi residual plots of images versus their intensity and hue.

PAC on 10 equispaced intervals, positioned in the midpoint of each interval, after which the averages were connected by line segments. The blue curves correspond to the average plus or minus one standard error. We see that higher intensities tend to yield lower PAC, meaning that the classifier did a better job on them. In the right hand panel we see a similar effect in function of the hue of each image.

A possible strategy is to record, collect, or construct a number of features and to run a regression method with the PAC as response variable. This may tell the user which factors affect the precision of the classifier. With that information one could finetune the classifier, or select additional features for the classification.

3.1. Class Maps

Class maps (Raymaekers, Rousseeuw, and Hubert 2022) are quasi residual plots versus a feature reflecting how far each case is from its class. This is based on some distance measure $D(i, g)$ of a case i relative to a class g. For each classifier in Section 4 we will specify an appropriate measure $D(i, g)$.

Next we estimate the cumulative distribution function of $D(\mathbf{x}, g)$ where $\mathbf{x}$ is a random object generated from class g. The *farness* of the object i to the class g is then defined as

$$\text{farness}(i, g) := P[D(\mathbf{x}, g) \leqslant D(i, g)] . \tag{6}$$

Therefore, farness(i, g) lies in the [0, 1] range, just like PAC(i). The computation of (6) is described in section A.1 of the supplementary material.

The class map plots PAC(i) versus farness(i, g) for all cases i with given label g. The colors of the points are those of the predicted class. Points with high farness to *all* classes are marked by a black border, as in Figure 4.

4. Silhouette and Quasi Residual Plots Applied to Classifiers

The silhouette plot, quasi residual plot, and class map can be drawn for training data as well as labeled test data, with the same interpretation. But the motives for looking at them are slightly different. Plots of the training data can help to assess whether the classifier is appropriate, discover overlap between classes, and find mislabeled points so their labels can be corrected to improve the trained model. Making plots of the test data can yield the same type of conclusions, but can also highlight aspects specific to the test data, such as differences between training and test data. They can also help identify gaps in the training data. If a test image of a cat in the snow is classified as a dog because only dogs appeared in the snow in the training data, it might be good to add images of cats in the snow to the training data.

4.1. Neural Nets

In this section we illustrate the proposed graphical displays in the setting of classification by a neural network. Neural networks encompass a broad class of classifiers which are based on a structure with an input layer, hidden layers, and an output layer, each consisting of a number of nodes. For a classification into G groups based on p-variate data, the input layer has p nodes, each corresponding to one input variable, and the output layer has G nodes, one for each class. The number and sizes of hidden layers and their connections determine the structure of the network and have to be fixed beforehand.

For an introduction to neural networks we refer to Hastie, Tibshirani, and Friedman (2017). In its simplest form, a neural network classifier has one hidden layer with M nodes. Case i is described by a p-dimensional vector x_i of input variables. The response is its given class g_i. This is encoded as a G-variate vector y_i which has 1 in the position g_i and 0 in all other positions. (This is called "one-hot encoding.") We aim to approximate the response by a function f, that is, $y_i \approx f(x_i)$. The neural network will then create M new features in the intermediate layer, given by

$$(z_i)_m = \sigma(\alpha_{0m} + \alpha'_m x_i) \quad \text{for} \quad m = 1, \ldots, M.$$

For the activation function σ one often takes the rectified linear unit $\sigma(t) := \max(0, t)$. Next, G-variate vectors v_i are obtained as linear combinations of the vectors z_i by

$$(v_i)_g = \beta_{0g} + \beta'_g z_i \quad \text{for} \quad g = 1, \ldots, G. \quad (7)$$

These vectors v_i do not yet contain probabilities. To that end one applies the G-variate *softmax* function h given by

$$(h(v_i))_g = \frac{e^{(v_i)_g}}{\sum_{j=1}^{G} e^{(v_i)_j}} . \quad (8)$$

Silhouette plot of neural net on CIFAR−10 test data

Classes

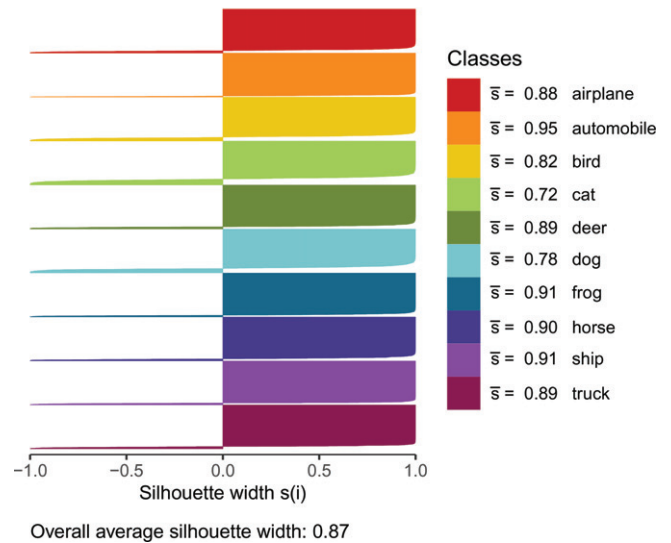|		
$\bar{s} = 0.88$	airplane
$\bar{s} = 0.95$	automobile
$\bar{s} = 0.82$	bird
$\bar{s} = 0.72$	cat
$\bar{s} = 0.89$	deer
$\bar{s} = 0.78$	dog
$\bar{s} = 0.91$	frog
$\bar{s} = 0.90$	horse
$\bar{s} = 0.91$	ship
$\bar{s} = 0.89$	truck

Silhouette width s(i)

Overall average silhouette width: 0.87

Figure 3. Silhouette plot on the CIFAR-10 test data.

The end result is the vector $f(x_i) := h(v_i)$ with positive entries. These can be seen as posterior probabilities $\hat{p}(i, g) := f(x_i)_g$. They indeed satisfy $\sum_{g=1}^{G} \hat{p}(i, g) = 1$ by virtue of (8) in the final layer of the network. The PAC can then be calculated from (3) and (4).

In practice most neural networks have multiple hidden layers, that are chained to each other to allow more complex structures to be learned from the data. These hidden layers are connected in the same way as the layers above, that is, by applying a nonlinear activation function on a linear combination of the outputs of the previous layer. Some of these layers can have specialized connections depending on the classification task at hand. For instance, for classifying images, convolutional neural networks (CNNs) are very popular. They incorporate "convolutional layers" and "pooling layers" that combine the information in nearby pixels ("local connectivity") to capture spatial information and reduce the dimension. Neural networks are most commonly trained by backpropagation, which allows gradient-based optimization of a loss function. The model is good when the fitted vectors $f(x_i)$ are close to the response vectors y_i. Training can take long, but for a new case x the prediction $f(x)$ is fast.

As an illustration we use the well-known CIFAR-10 benchmark dataset. It consists of 60,000 color images with 32×32 pixels. They depict objects from 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images.

The CIFAR-10 data have been classified by the residual neural network with 56 layers (ResNet-56) of He et al. (2016). We have looked at the proposed graphical displays on the training data (not shown), but they are not very eventful because the model obtains a perfect classification on the training data. On the test data the accuracy is a realistic 94%, allowing for more interesting visualization. Figure 3 shows the silhouette plot of the test data. With an overall average silhouette width of 0.87, we can conclude that the test data is classified quite well. We also see clear differences between the classes. The class of automobiles has the highest average silhouette width $\bar{s} = 0.95$, so
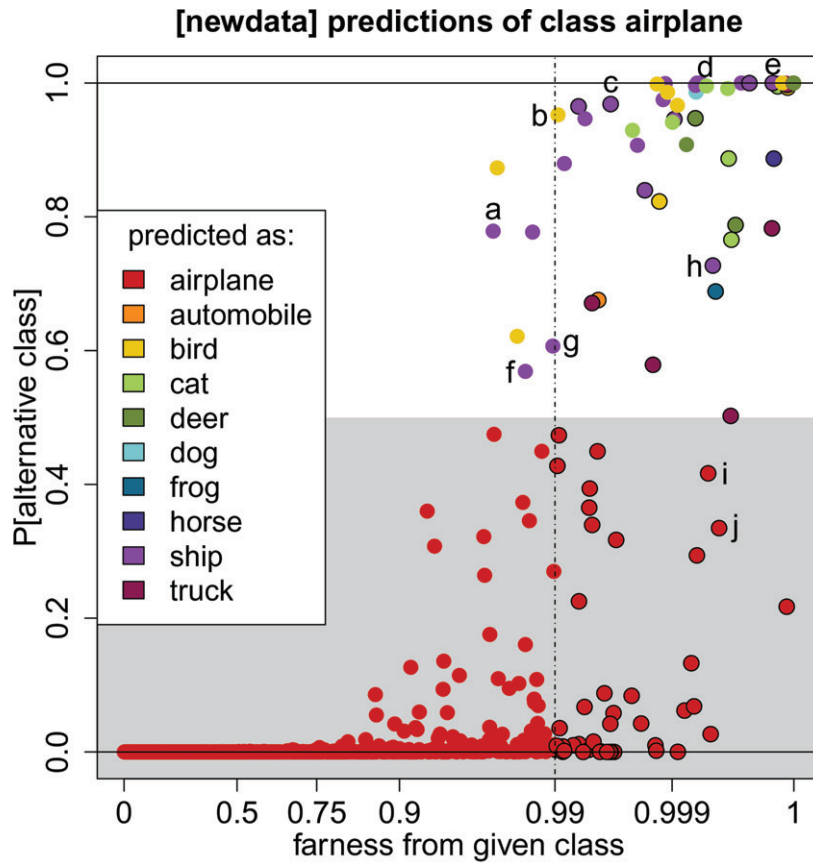
Figure 4. Class map of the airplane class in the test data, with the images corresponding to the marked points.

the classifier did best on this class. Animals seem to be harder to classify, with cats and dogs obtaining an average silhouette width $\bar{s}$ below 0.8. Also note that this classifier often had a high conviction, with many $s(i) \approx 1$ (when classified correctly) or $s(i) \approx -1$ (when misclassified).

Now we turn to quasi residual plots. For image data, it would not be natural to plot the PAC versus a single input feature, since this would correspond to the red, green or blue value in one pixel. Instead, we used the summary properties of intensity and hue of an image in Figure 2, already shown in Section 3.

For the class maps described in Section 3 we start by computing the Mahalanobis distance $D(i, g)$ of each case i relative to

each class g, given by

$$D(i, g) := \sqrt{(\mathbf{v}_i - \bar{\mathbf{v}}_g)' \widehat{\boldsymbol{\Sigma}}_g^{-1} (\mathbf{v}_i - \bar{\mathbf{v}}_g)} \qquad (9)$$

where $\mathbf{v}_i$ is given by (7), $\bar{\mathbf{v}}_g$ is the average of all $\mathbf{v}_j$ in class g in the training data, and $\widehat{\boldsymbol{\Sigma}}_g$ is their covariance matrix. This requires that all the $G \times G$ matrices $\widehat{\boldsymbol{\Sigma}}_g$ are nonsingular, which is typically the case when each class has many members compared to G. The resulting farness is then given by (6).

Figure 4 is the class map of airplane images in the test data. We see that most of the images get classified correctly, as the
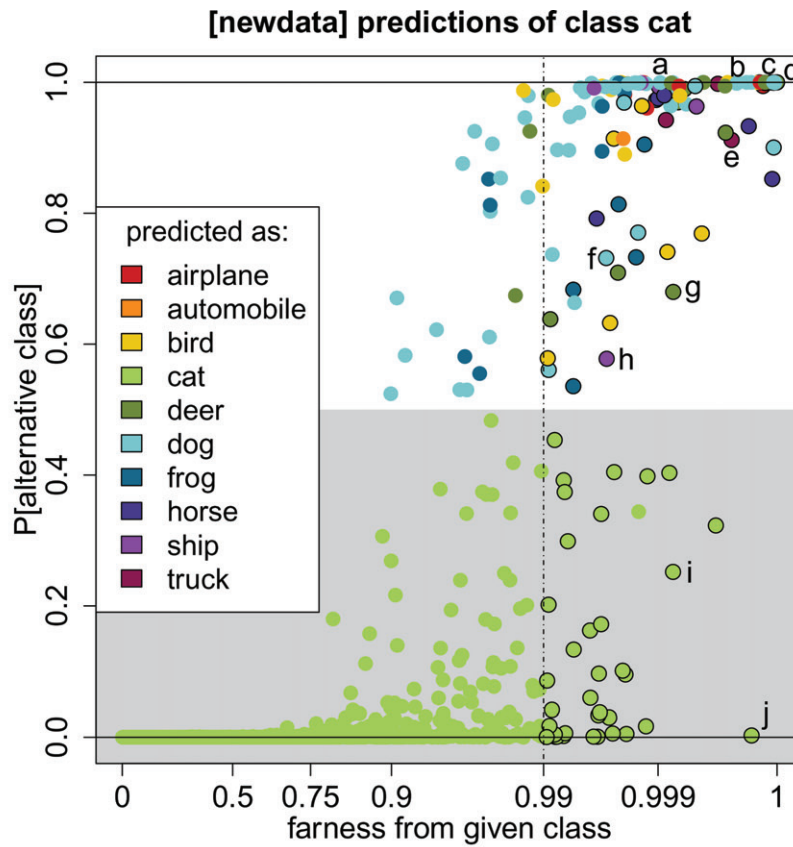
[newdata] predictions of class cat

Figure 5. Class map of the cat class in the test data, with the corresponding images.

majority of points have a PAC below 0.5 and are shown in red, the color of this class. The misclassified cases are the points with PAC(*i*) above 0.5, with many purple points being classified as ships and some yellow points as birds. These unusual cases stood out in the class map, and deserve to be looked at. The images corresponding to the marked points are shown below the class map. Images a, c–e and g assigned to class "ship" have water in them. Three of these are seaplanes (c–e), and g contains both an airplane and a ship. Image b is classified as a bird, which is not too surprising as it does look like one. The object covers so few pixels that it is hard to classify, even for a human. Finally, points i and j are correctly classified as airplanes, but with high farness. The first is an airplane photographed from an unusual

angle, with a city as background. The second has a strange shape, and could be a stealth plane.

The farness probabilities on the horizontal axis are not equi-spaced: they are shown on the scale of quantiles of the standard Gaussian distribution restricted to the interval [0,4]. This makes high farness values stand out more. The vertical dashed line is at a cutoff value, which can be chosen by the user and is 0.99 by default. Cases which are far from every class in the data, that is with farness(*i*, *g*) above the cutoff for all *g*, are called *farness outliers* and plotted with a black border in the class map. Such cases do not lie well within any class, for example the images c, e, and h–j. Note that class maps are similar in spirit to the regression outlier maps of Rousseeuw and van Zomeren (1990)

and Rousseeuw et al. (2004), which plot residuals versus farness to the entire dataset.

As a second example we consider the class map of cat images, shown in Figure 5. As could be expected, there is some confusion with the dog class. Not only do cats and dogs often have similar features, they also tend to be photographed with similar backgrounds, as illustrated in the images a, b, and f. Point c has both a very high farness and the highest PAC. This is clearly a mislabeled image, which should belong to the frog class! We detected this image because of its extremely high PAC and farness, whereas it would have been harder to find if we only had a long list of misclassified cases. Image d is classified as deer, but flagged as a farness outlier, suggesting that this image does not lie well within any class. Images e and h are misclassified as a truck and a ship, respectively. The first may be explained by the straight lines in the image, whereas the second is of a cat in a blue container. As both of these are farness outliers, they are not close to any of the classes, making them hard to classify. Finally, images i and j are farness outliers but classified correctly. They contain cats, but the images are dominated by other objects.

In both class maps, higher PAC values are associated with higher farness. This suggests that in the CIFAR-10 dataset, the inaccuracy of the neural net classifier is caused more by *feature noise* (atypical images) than by *label noise* (randomness in the response).

Some other classes are shown in section A.2 of the supplementary material.

4.2. Classification Trees

In this section we will visualize the cases in a tree-based classification. Here we use CART (Breiman et al. 1984), for which many implementations exist such as the R package rpart (Therneau, Atkinson, and Ripley 2019), but other tree-based classifiers such as C4.5 (Quinlan 1993) can be visualized as well. As an illustrative example we analyze the Titanic data. This dataset is freely available on *https://www.kaggle.com/c/titanic/data*. The data contains information on the passengers of the RMS Titanic. The binary response variable indicates whether the passenger survived or was a casualty. It also contains a mix of nominal, ordinal and numerical variables describing several characteristics. Strong points of CART are its ability to deal with features of mixed types as well as missing values, which are abundant in these data. We train the classification tree predicting the survival of the passengers from the features Pclass, Sex, Sibsp, Parch, Fare, and Embarked. Pclass is an ordinal variable ranging from 1 (first class) to 3 (third class), Sex is male or female, and Fare is in British Pounds. The variables Sibsp and Parch count the number of siblings+spouses and parents+children aboard. Embarked is the port (out of three) at which the passenger got on the ship. The resulting tree is shown in Figure 6, drawn with the rpart.plot package (Milborrow 2020). We see that only 4 out of 6 variables are actually used in the model, and that the tree starts with a very crude split which predicts all males as casualties. This tree has an accuracy of about 82% on the training data.

At the bottom of the tree in Figure 6 we see the leaves. The leftmost leaf contains all males, which make up 65% of the passengers, with the probability of survival being 19%. So for all cases i that end up in this leaf, the posterior probabilities are

$$\hat{p}(i, \text{survived}) = 0.19 \quad \text{and} \quad \hat{p}(i, \text{casualty}) = 1 - 0.19 = 0.81.$$

The classification by the maximum a posteriori rule (2) thus assigns all objects in this leaf to the casualty class, which is listed as the top line inside the leaf. Analogously, the rightmost leaf represents 19% of all passengers, its posterior probability of survival is 95%, so all members of this leaf are predicted as survived.

Now that we know the posterior probabilities for each case i in the dataset, it is trivial to compute the probability of the alternative class PAC(i) from (4). Next, (5) immediately yields the silhouette plot, shown in Figure 7. With an average silhouette width of 0.44, we conclude that this classification of the Titanic data is far from perfect, but it may be hard to do much better given the presumably chaotic decision making at the time of the disaster. By comparing the average silhouette widths, we see that the class of survived passengers (in blue) is hardest to predict. But the correct predictions in this class are made with a relatively high conviction ($s(i) \approx 1$). This is in contrast to the predictions of the casualty class which contains fewer misclassified cases, but the correct classifications for this class are made with only moderate conviction.

Figure 8 shows an interesting quasi residual plot, of PAC versus age for the males in the data. The PAC only takes two values in this subset of the data, corresponding to the leftmost leaf in Figure 6. As a visual aid the loess curve (Cleveland 1979) was superimposed, using the loess function in base R with default settings. This indicates that the PAC values (here linked to survival) for very young males are substantially higher than for older males. The graph thus uncovered a phenomenon in a subset of the data.

Note that each leaf of a classification tree such as Figure 6 corresponds to a number of splits in the features, so its domain in feature space can be called "rectangular." The region that is assigned the same prediction is thus a union of such "rectangles," which does not need to be connected. Therefore, a tree-based classifier takes a completely different view of the data than, for instance, linear discriminant analysis. In order to construct class maps, we need a measure of farness in feature space which is in line with how the classifier looks at the data. Therefore, it is natural to take a distance measure that is additive in the features. Moreover, the distance measure needs to be able to handle features of mixed types, as well as missing values. For these reasons we elect to use the daisy function introduced by Kaufman and Rousseeuw (1990), which is a dissimilarity version of the similarity coefficient of Gower (1971) for nominal, asymmetric binary, and numerical variables, extended to encompass ordinal variables. It is implemented as the function daisy in the R package cluster (Maechler et al. 2019).

Simply applying daisy to the set of features would ignore an important property of tree-based classifiers, which is that they do not consider all features equally important. To take this aspect into account, we use a weighted daisy dissimilarity where the weights are equal to each variable's importance. The variable importance is a standard output of rpart, computed as described in (Breiman et al. 1984). The weighted daisy computation provides us with a dissimilarity $d(i, j)$ between all cases i and j. When computing the farness of a case i to a class g, we
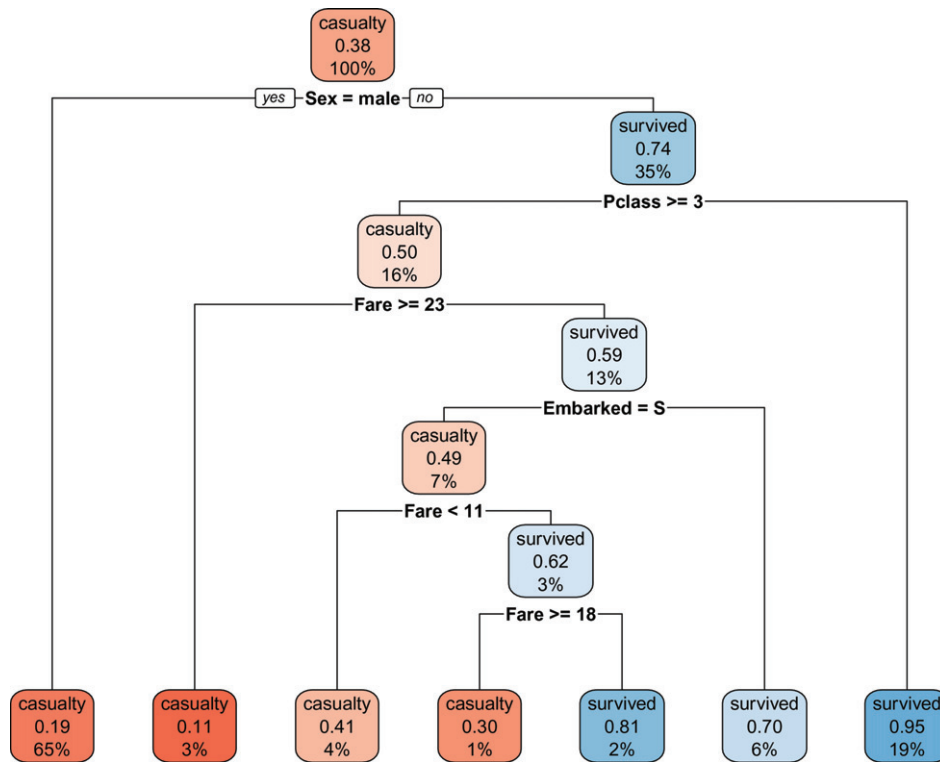
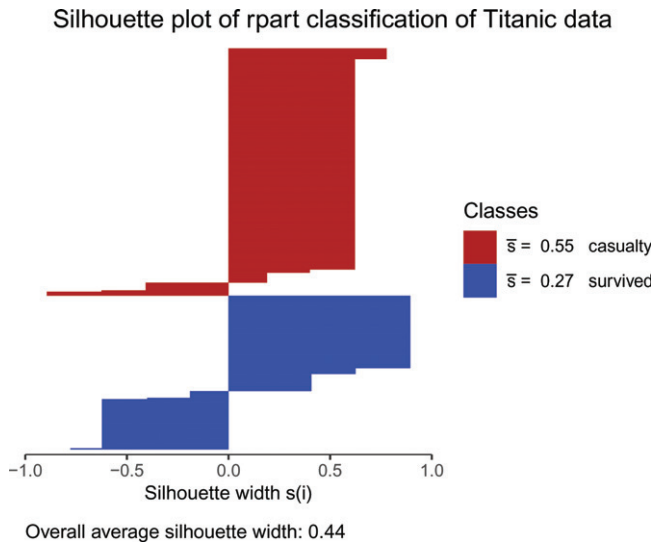**Figure 6.** CART classification tree trained on the Titanic data.

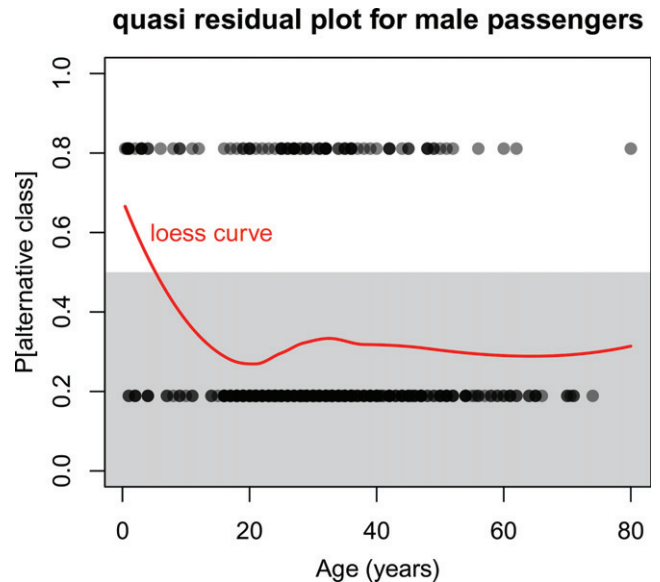**Figure 7.** Silhouette plot of the CART classification on the Titanic training data.

Figure 8. Quasi residual plot of male passengers versus their age, with loess curve in red.

need to take into account the local nature of classification trees. To this end we use a nearest-neighbors type approach, described in detail in section A.3, supplementary material.

Figure 9 shows the class maps of the Titanic training data, in which we note a few extreme points. Cases a and b have the highest PAC in the casualty class, combined with a relatively high farness. These are a woman and female child traveling in first class, for which the classifier predicted survival. The elevated farness is due to some unusual characteristics for the casualty class, such as a high fare for the child, the gender and traveling class of both subjects, and an uncommon port of embarkation for the woman. Passenger c has a low PAC but a

rather high farness. This is a male passenger who was correctly predicted as a casualty. His high farness is caused by paying a huge fare (in the top 1%), traveling in first class, and having as many as 4 children+parents traveling with him.

The class map of the survived passengers is shown in the right hand panel of Figure 9. Case d is a woman who traveled in third class and is misclassified as a casualty. Her relatively high farness is caused by the fact that she was traveling with as many as 5 parents+children. Passengers e and f are two males, and thus predicted as casualties with high PAC. Their substantial farness
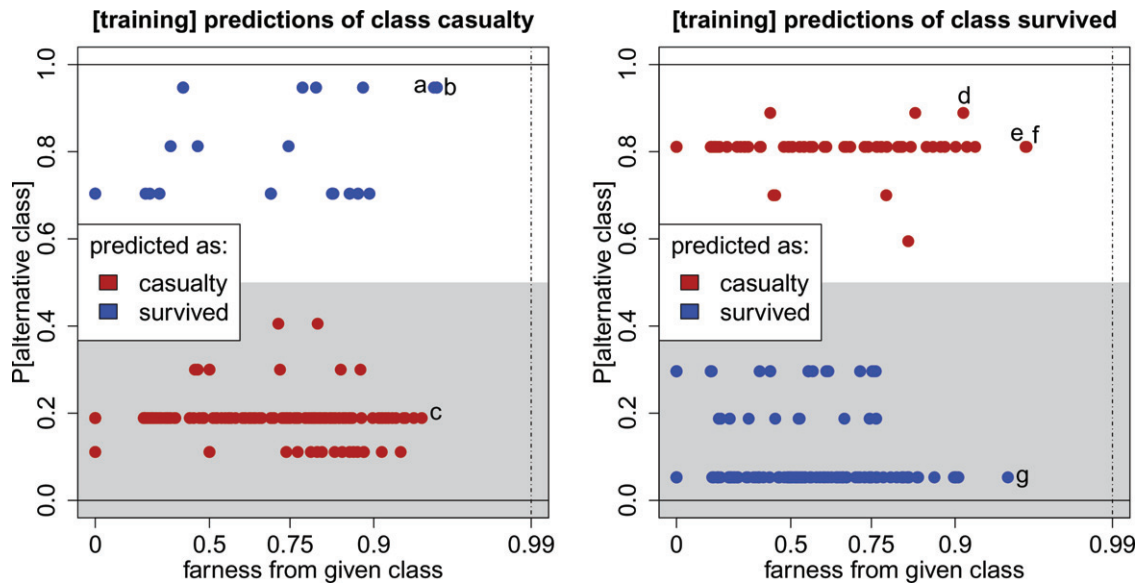
Figure 9. Titanic training data: class maps of casualties (left) and survivors (right).
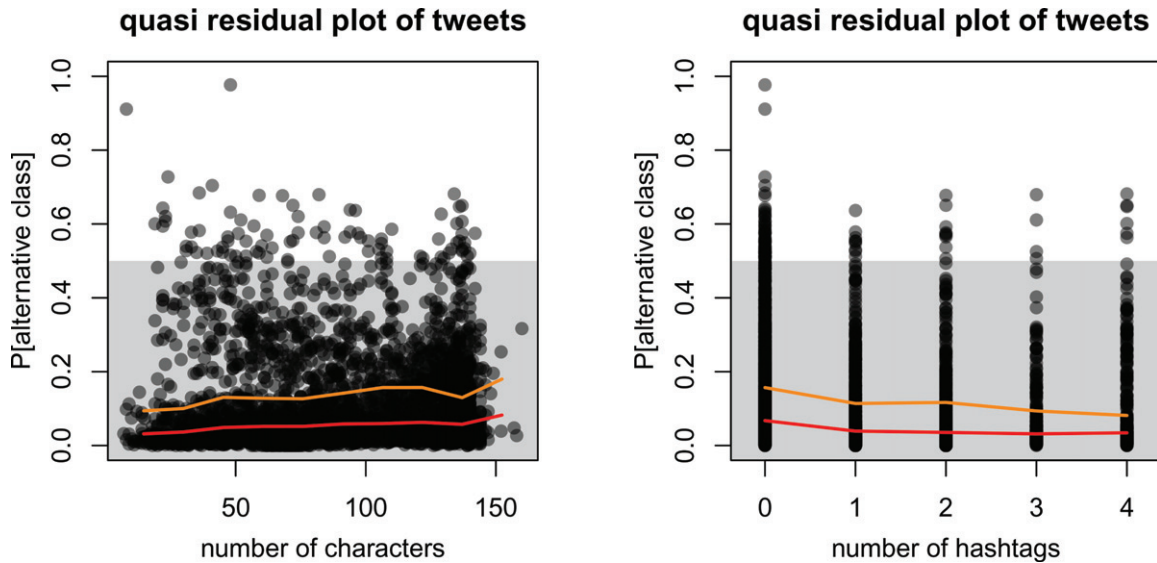

Figure 10. Quasi residual plots, with their medians (red) and 75th percentiles (orange).

is explained by having paid a high fare. Point g is a woman traveling first class, correctly predicted as survived, with much conviction since her PAC is close to zero. She paid the highest fare of all passengers in the training data, causing her relatively high farness.

In Figure 9 we note that high PAC values occur over the whole farness range, telling us that for the Titanic data the misclassifications were mainly caused by label noise, that is, much randomness in the survival label. Section A.4 in the supplementary material shows the corresponding class maps for the Titanic test data, where again some individuals stand out.

4.3. Random Forests

Random forests were introduced by Breiman (2001) and are based on an ensemble of decision trees. The idea is to train many different classification trees for the same task. In order to

generate sufficiently diverse trees, two techniques are exploited. The first is bagging, which means that only a subsample of the observations is used when training a single tree. The second is the random sampling of potential variables at each split. This forces the various trees to use a wide variety of variables. To classify case i, we let it go down all of the trees in the forest. Its posterior probability $\hat{p}(i, g)$ in class g is then the number of trees that assigned it to class g, divided by the total number of trees. These posterior probabilities clearly add up to 1. Applying the maximum a posteriori rule (2), we then assign case i to the class with highest $\hat{p}(i, g)$. Random forests often perform well in real world classification problems. Here we use the implementation in the R package `randomForest` by Liaw and Wiener (2018).

As an illustration we analyze the emotion dataset of Mohammad and Bravo-Marquez (2017). It contains a training set of 3613 tweets and a test set of 3142 tweets, which have been labeled with the four classes anger, fear, joy, and sadness. The

Figure 11. Class map of the anger class, with the corresponding tweets.
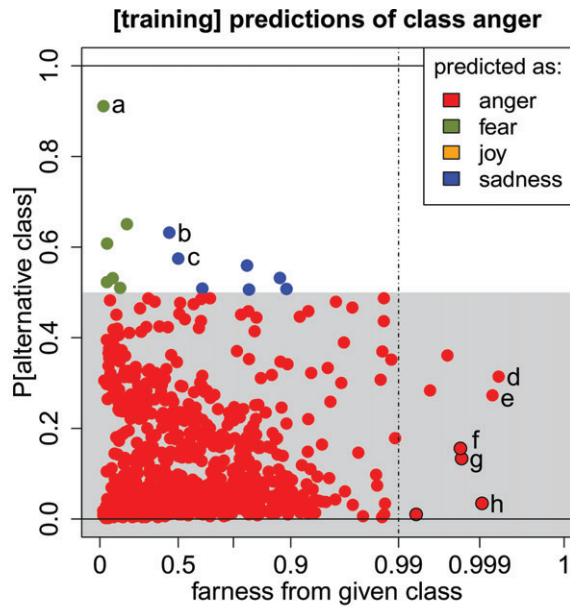
label	tweet
a	testing
b	@jwolfie_ why you gotta use the dark skin emoji
c	@shopgreenwich #ldf16 what shall we do this weekend? #spraypainting in #greenwichmarket with @SNUB23 #core246 #lilylou #fret & #benoakley
d	@MHChat sadness with resentment is the past, sadness with fear is the future. try to live in the now #MHchat
e	To'Why can't we brothers protect one another? No one's serious, and it makes me furious. Don't be misled, just think of Fred.' #mayfield
f	Focus on yourself. Don't let other's love define you when you're happy and don't let it make you bitter if you get hurt.
g	How can I tell Happy Anniversary, when u are not happy.. #bitter #ampalaya #paitpaitanangpeg
h	And she got all angry telling me 'but what would be doing a 40 year old guy looking for a girl like you' and I felt #offended

goal is to train a random forest to predict the emotion of a tweet. We preprocessed the data by removing word contractions and elongations using the R package `textclean` (Rinker 2018). We also replaced the emojis with unique words. Finally, we used the R package `text2vec` (Selivanov, Bickel, and Wang 2020) to convert these texts into numerical data based on n-grams of at most length 3. This procedure leaves us with a 3613×2705 training data matrix. The class anger has 857 tweets, fear 1147, joy has 823, and sadness 786. We then ran `randomForest()` with its default options. The trained forest achieves an accuracy of 97.6% on the training data, and 80.8% on the test data. Since each case i gets posterior probabilities $\hat{p}(i, g)$ for each class g, we can easily compute the probability of the alternative class PAC(i) from (3) and (4). The silhouette plot for the training data was shown in Figure 1.

The left panel of Figure 10 is the quasi residual plot of PAC versus the number of characters in each tweet, with the median (in red) and 75th percentile (orange) on 10 equispaced intervals. As most tweets are classified correctly with low PAC values, these trend lines are near the bottom of the plot. But we still see an upward trend, which indicates that longer tweets were somewhat harder to classify, perhaps due to containing words linked with more than one class. The quasi residual plot in the right hand panel is versus the number of hashtags (0, 1, 2, 3, and 4+). This time we see a downward trend, suggesting that tweets with more hashtags are typically easier to classify. Both plots revealed an effect that we could not have predicted beforehand.

We now turn to the class maps of the emotion data. The farness is computed along the same lines as in the previous section, starting from a weighted daisy dissimilarity between

[training] predictions of class joy
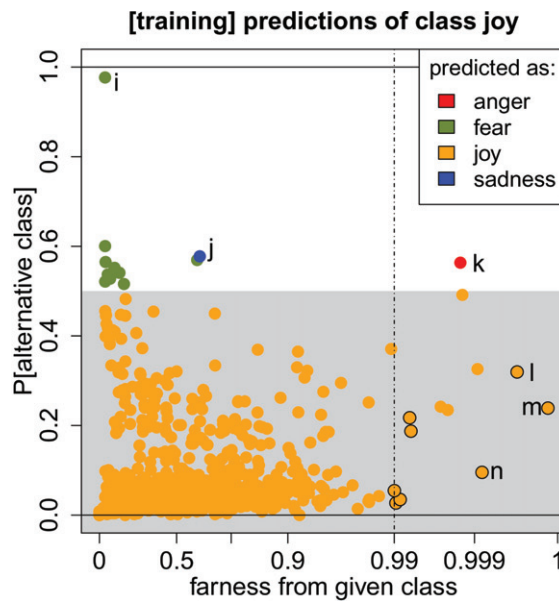
label	tweet
i	@niceoneWern @Alotta_Pain the gleesome threesome
j	Quinn's short hair makes me sad.
k	Never make a #decision when you're #angry and never make a #promise when you're #happy. #wisewords
l	@rosyIipH glee glee glee glee gLEE GLEE i LOST LOST LOST lowe much
m	I'm worried someday I'll be yelling at my kids and they'll be confused about if or not I'm being serious 😭😭😭I'm too damn playful
n	@yoshi_lucky Good morning.Let's start with a smile! Let's enjoy life in a cheerful way! Don't worry be happy!

Figure 12. Class map of the joy class, with the corresponding tweets.

cases. The weights are again given by each variable's importance, corresponding to the total decrease of the Gini index by splitting on the variable, and averaged over all trees in the forest. This is a standard output of the function `randomForest()`. Deriving the farness of case i from class g is also done in the same way, described in section A.3 of the supplementary material.

Figure 11 shows the class map of the anger class. Only a few points aren't red, meaning they are assigned to a different class. We marked some points that jump out, and the corresponding tweets are listed below the class map. Point a has the highest PAC in this class. It corresponds to the uninformative tweet "testing" which does not contain any word in the constructed vocabulary, so it gets assigned to the fear class simply because that class has the most members. The class map drew our attention to this atypical tweet. Tweets b and c are assigned to sadness. The first might indeed be sad, but the context is lacking. The second does not seem to carry a clear emotion. We also look at some tweets that are classified correctly, but lie far from their given class. Tweet d seems more sad than angry, and contains a lot of words which would not immediately be associated with anger. Tweet e is in fact part of a song lyric. Finally, f, g, and h have a black border which indicates that their farness to *all* classes is above

the cutoff value, so they do not lie well within any class. Indeed, they refer to emotions belonging to none of the classes, such as bitterness and feeling offended.

The class map of the joy class is shown in Figure 12. There are only a few misclassified points, as well as a handful of farness outliers on the bottom right. Tweet i contains the word gleesome, which does suggest joy, but this word only occurs once in the dataset so it is not in the constructed vocabulary, leading i to be assigned to the largest class (fear). Tweet j appears to be mislabeled, as it suggests sadness instead of joy. Tweet k is classified as angry, but not with high conviction since its PAC(i) is only slightly above 0.5 . It also has a rather high farness, indicating that it doesn't lie well within the joy class. The text is a proverb about two emotions. Tweets l to n have a black border indicating farness outliers, which suggests that they do not lie well within any of the classes. Tweet l is indeed strange with many repetitions, and m mixes emotions so it is hard to give it a single label. Tweet n is definitely in the joy class, but contains an unusual number of joy-related words compared to other tweets of this class.

In both class maps the PAC > 0.5 values are not associated with high farness. This suggests that the misclassifications are

mainly driven by label noise, caused by the difficulty of labeling the emotion of some tweets. The class maps of the remaining emotions fear and sadness are discussed in section A.5 of the supplementary material.

5. Conclusions

The proposed visualizations focus on the cases in a classification. The examples illustrated the benefits of this approach. The new silhouette plot describes the strength of each object's classification, grouped by class. For instance, we noticed that images of mechanical objects were typically classified more reliably than images of animals. Quasi residual plots yielded other insights, such as trends in subsets of the data like the effect of age for male passengers on the Titanic. They also revealed factors affecting the classification accuracy, such as the length of tweets and their number of hashtags. The class map provides additional information, as it can tell us which cases lie between classes, which cases are far from their given class, and some cases may be far from all classes. The class map allowed us to distinguish between feature noise and label noise in the examples: in the image data the misclassifications were mainly driven by atypical images, whereas in the other examples the dominant effect was some randomness in the response (the labels), such as survival in the Titanic data. The displays also drew our attention to atypical cases that were inspected in more detail, providing further insights in the data.

The visual displays in this article were produced with the R package `classmap` (Raymaekers and Rousseeuw 2021) on CRAN. Its vignettes `Neural_net_examples`, `Rpart_examples` and `Random_forest_examples` correspond to the classifiers in Section 4. Note that all three visualizations make use of the posterior probabilities that a case belongs to the available classes (labels). Since most classifiers provide such probabilities, the graphical displays can be employed not only with neural nets and tree-based classifiers, but also with other methods such as discriminant analysis, k-nearest neighbors, and support vector machines. The displays for these methods are also available in the *classmap* package.

Supplementary Materials

These consist of a text with additional information and figures, and an R script that reproduces the examples in the article.

Acknowledgments

The reviewers made helpful comments improving the presentation.

References

Breiman, L. (2001), "Random Forests," *Machine Learning*, 45, 5–32. [1332,1340]

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*, Boca Raton, FL: Chapman and Hall. [1332,1338]

Cleveland, W. S. (1979), "Robust Locally Weighted Regression and Smoothing Scatterplots," *Journal of the American Statistical Association*, 74, 829–836. [1338]

Gower, J. C. (1971), "A General Coefficient of Similarity and Some of its Properties," *Biometrics*, 27, 857–871. [1338]

Hastie, T., Tibshirani, R., and Friedman, J. (2017), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 12th printing*. Springer Series in Statistics, New York: Springer. [1332,1335]

He, K., Zhang, X., Ren, S., and Sun, J. (2016), "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. [1335]

Kaufman, L., and Rousseeuw, P. J. (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Hoboken, NJ: Wiley-Interscience. [1338]

Liaw, A., and Wiener, M. (2018), *randomForest: Breiman and Cutler's Random Forests*. CRAN, R package. Available at *https://CRAN.R-project.org/package=randomForest*. [1340]

Maechler, M., Rousseeuw, P. J., Struyf, A., and Hubert, M. (2019), *cluster: Methods for Cluster Analysis.* CRAN, R package. Available at *https://CRAN.R-project.org/package=cluster*. [1338]

Milborrow, S. (2020), *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. CRAN, R package. [1338]

Mohammad, S., and Bravo-Marquez, F. (2017), "WASSA-2017 Shared Task on Emotion Intensity," in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark, pp. 34–49. Association for Computational Linguistics. [1340]

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann Publishers. [1338]

Raymaekers, J., and Rousseeuw, P. J. (2021), *classmap: Visualizing Classification Results*. CRAN, R package. Available at *https://CRAN.R-project.org/package=classmap*. [1343]

Raymaekers, J., Rousseeuw, P. J., and Hubert, M. (2022), "Class Maps for Visualizing Classification Results," *Technometrics*, to appear. [1334]

Rinker, T. W. (2018), *textclean: Text Cleaning Tools*. R package. version 0.9.3. [1341]

Rousseeuw, P. J. (1987), "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Journal of Computational and Applied Mathematics*, 20, 53–65. [1332]

Rousseeuw, P. J., Van Aelst, S., Van Driessen, K., and Agulló, J. (2004), "Robust Multivariate Regression," *Technometrics*, 46, 293–305. [1338]

Rousseeuw, P. J., and van Zomeren, B. (1990), "Unmasking Multivariate Outliers and Leverage Points," *Journal of the American Statistical Association*, 85, 633–651. [1337]

Selivanov, D., Bickel, M., and Wang, Q. (2020), *text2vec: Modern Text Mining Framework for R*. CRAN, R package. [1341]

Shahroudnejad, A. (2021), "A Survey on Understanding, Visualizations, and Explanation of Deep Neural Networks." arXiv:2102.01792. [1332]

Therneau, T., Atkinson, B., and Ripley, B. (2019), *Package 'rpart'*. CRAN, R package. Available at *https://CRAN.R-project.org/package=rpart*. [1332,1338]