A probability transducer
and decision-theoretic augmentation
for machine-learning classifiers

K. Dyrland ◉
<kjetil.dyrland@gmail.com>

A. S. Lundervold ◉[†]
<alexander.selvikvag.lundervold@hvl.no>

P.G.L. Porta Mana ◉
<pgl@portamana.org>

(listed alphabetically)

Dept of Computer science, Electrical Engineering and Mathematical Sciences,
Western Norway University of Applied Sciences, Bergen, Norway

[†] & Mohn Medical Imaging and Visualization Centre, Dept of Radiology,
Haukeland University Hospital, Bergen, Norway

Draft. 14 April 2022; updated 30 May 2022

✎ The present work sells probabilities for machine-learning algorithms at a bargain price. It shows that these probabilities are just what those algorithms need in order to increase their sales.

1   The inadequacy of common classification approaches

As the potential of using machine-learning algorithms in important fields such as medicine or drug discovery increases[1], the machine-learning community ought to keep in mind what the actual needs and inference contexts in such fields are. We must avoid trying (intentionally or unintentionally) to convince such fields to change their needs, or to ignore their own contexts just to fit machine-learning solutions that are available and fashionable at the moment. Rather, we must make sure the that solutions fit needs & context, and amend them if they do not.

The machine-learning mindset and approach to problems such as classification in such new important fields is often still inadequate in many respects. It reflects simpler needs and contexts of many inference problems successfully tackled by machine learning earlier on.

---

[1] Lundervold & Lundervold 2019; Chen et al. 2018; Green 2019.

This document is designed for screen reading and two-up printing on A4 or Letter paper

A stereotypical 'cat vs dog' image classification, for instance, has four very important differences from a 'disease *I* vs disease *II*' medical classification, or from an 'active vs inactive' drug classification:

(i) Nobody presumably dies or loses large amounts of money if a cat image is misclassified as dog or vice versa. But a person can die if a disease is misdiagnosed; huge capitals can be lost if an ultimately ineffective drug candidate is pursued. The *gains and losses* – or generally speaking the *utilities* – of correct and incorrect classifications in the former problem and in the two latter problems are vastly different.

(ii) To what purpose do we try to guess whether an image's subject is a cat or a dog? For example because we must decide whether to put it in the folder 'cats' or in the folder 'dogs'. To what purpose do we try to guess a patient's disease or a compound's chemical activity? A clinician does not simply tell a patient "You probably have such-and-such disease. Goodbye!", but has to decide among many different kinds of treatments. The candidate drug compound may be discarded, pursued as it is, modified, and so on. *The ultimate goal of a classification is always some kind of decision*, not just a class guess. In the cat-vs-dog problem there is a natural one-one correspondence between classes and decisions. But in the medical or drug-discovery problems *the set of classes and the set of decisions are very different*, and have even different numbers of elements.

(iii) If there is a 70% probability that an image's subject is a cat, then it is natural to put it in the folder 'cats' rather than 'dogs' (if the decision is only between these two folders). If there is a 70% probability that a patient has a particular health condition, it may nonetheless be better to dismiss the patient – that is, to behave as if there was no condition. This is the optimal decision, for example, when the only available treatment for the condition would severely harm the patient if the condition were not present. Such treatment would be recommended only if the probability for the condition were much higher than 70%. Similarly, even if there is a 70% probability that a candidate drug is active it may nonetheless be best to discard it. This is the economically most advantageous choice if pursuing a false-positive leads to large economic losses. *The target of a classification is not what's most probable, but what's optimal.* ✎ add something on proper probabilities, connecting to (iv)

(iv) The relation from image pixels to house-pet subject may be almost deterministic; so we are effectively looking for or extrapolating

a function pet = f(pixels) contaminated by little noise. But the relation between medical-test scores or biochemical features on one side, and disease or drug activity on the other, is typically *probabilistic*; so a function disease = f(scores) or activity = f(features) does not even exist. *We are assessing statistical relationships* P(disease, scores) or P(activity, features) instead, which include deterministic ones as special cases.

In summary, there is place to improve classifiers so as to (i) quantitatively take into account actual utilities, (ii) separate classes from decisions, (iii) target optimality rather than 'truth', (iv) output and use proper probabilities.

In artificial intelligence and machine learning it is known how to address all these issues in principle – the theoretical framework is for example beautifully presented in the first 18 chapters or so of Russell & Norvig's (2022) text[2].

Issues (i)–(iii) are simply solved by adopting the standpoint of *Decision Theory*, which we briefly review in § 3 below and discuss at length in a companion work[3]. In short: The set of decisions and the set of classes pertinent to a problem are separated if necessary. A utility is associated to each decision, relative to the occurrence of each particular class; these utilities are assembled into a utility matrix: one row per decision, one column per class. This matrix is multiplied by a column vector consisting in the probability for the classes. The resulting vector of numbers contains the expected utility of each decision. Finally we select the decision having *maximal expected utility*, according to the principle bearing this name. Such procedure also takes care of the class imbalance problem[4].

Clearly this procedure is computationally inexpensive and ridiculously easy to implement in any machine-learning classifier. The difficulty is that this procedure requires *sensible probabilities* for the classes, which brings us to issue (iv), the most difficult.

Some machine-learning algorithms for classification, such as support-vector machines, output only a class label. Others, such as deep networks, output a set of real numbers that can bear some qualitative relation to the plausibilities of the classes. But these numbers cannot be reliably interpreted as proper probabilities, that is, as the degrees of belief assigned

---

[2] Self & Cheeseman 1987; Cheeseman 1988; 2018; Pearl 1988; MacKay 2005 see also.
[3] Dyrland et al. 2022a.   [4] cf. the analysis by Drummond & Holte 2005 (they use the term 'cost' instead of 'utility').

to the classes by a rational agent[5]; or, in terms of 'populations'[6], as the expected frequencies of the classes in the hypothetical population of units (degrees of belief and frequencies being related by de Finetti's theorem[7]). Algorithms that internally do perform probabilistic calculations, for instance naive-Bayes or logistic-regression classifiers[8], unfortunately rest on strong probabilistic assumptions, such as independence and particular shapes of distributions, that are often unrealistic (and their consistency with the specific application is rarely checked). Only particular classifiers such as Bayesian neural networks[9] output sensible probabilities, but they are computationally very expensive. The stumbling block is the extremely high dimensionality of the feature space, which makes the calculation of the probabilities

$$P(class, feature \mid training\ data)$$

(a problem opaquely called 'density regression' or 'density estimation'[10]) computationally unfeasible.

If we solved the issue of outputting proper probabilities then the remaining three issues would be easy to solve, as discussed above.

In the present work we propose an alternative solution to calculate proper class probabilities, which can then be used in conjunction with utilities to perform the final classification or decision.

This solution consists in a sort of 'transducer' or 'calibration curve' that transforms the algorithm's raw output into a probability. It has a low computational cost, can be applied to all commonly used classifiers and to simple regression algorithms, does not need any changes in algorithm architecture or in training procedures, and is grounded on first principles. The probability thus obtained can be combined with utilities to perform the final classification task.

Moreover, this transducer has three other great benefits, which come automatically with its computation. First, it allows us to calculate both the probability of class conditional on features, and *the probability of features conditional on class*. In other words it allows us to use the classification

[5] MacKay 1992a; Gal & Ghahramani 2016; Russell & Norvig 2022 chs 2, 12, 13.   [6] Lindley & Novick 1981.   [7] Bernardo & Smith 2000 ch. 4; Dawid 2013.   [8] Murphy 2012 § 3.5, ch. 8; Bishop 2006 §§ 8.2, 4.3; Barber 2020 ch. 10, § 17.4.   [9] Neal & Zhang 2006; Bishop 2006 § 5.7.   [10] Ferguson 1983; Thorburn 1986; Hjort 1996; Dunson et al. 2007.

algorithm in both 'discriminative' and 'generative' modes[11], even if the algorithm was not designed for a generative use. Second, its computation also gives a quantification of how much the probability would change if we had further training data. Third, it can give an *evaluation of the whole classifier* – including an uncertainty about such evaluation – that allows us to compare it with other classifiers to choose the optimal one.

In § 2 we present the general idea behind the probability-transducer and its calculation. Its combination with the rule of expected-utility maximization to perform classification is discussed in § 3; we call this combined use the 'augmentation' of a classifier.

In § 4 we demonstrate the implementation, use, and benefits of classifier augmentation in a concrete drug-discovery classification problem and dataset, with a random forest and a convolutional neural network classifiers.

Finally, we give a summary and discussion in § 6, including some teasers of further applications to be discussed in future work.

2   An output-to-probability transducer

In the present section we explain the main idea in informal and intuitive terms, and give its mathematical essentials only.

2.1   Main idea: algorithm output as a proxy for the features

Let us first consider the essentials behind a classification (or regression) problem. We have the following quantities:

- the *feature* values of a set of known units,
- the *classes* of the same set of units,

which together form our *learning* or *training data*; and

- the feature value of a *new* unit,

where the 'units' could be widgets, images, patients, drug compounds, and so on, depending on the classification problem. From these quantities we would like to infer

- the class of the new unit.

---

[11] Russell & Norvig 2022 § 21.2.3; Murphy 2012 § 8.6.

This inference consists in probabilities

P(class of new unit | feature of new unit, classes & features of known units)

$$(1)$$

for each possible class.

These probabilities are obtained through the rules of the probability calculus[12]; in this case specifically through the so-called de Finetti theorem[13] which connects training data and new unit. This theorem is briefly summarized in appendix B.1.

Combined with a set of utilities, these probabilities allow us to determine an optimal, further decision to be made among a set of alternatives. Note that the inference (1) includes deterministic interpolation, i.e. the assessment of a function class = f(feature), as a special case, when the probabilities are essentially 0s and 1s.

A trained classifier should ideally output the probabilities above when applied to the new unit. Machine-learning classifiers trade this capability for computational speed – with an increase in the latter of several orders of magnitude[14]. Thus their output cannot be considered a probability, but *it still carries information about both class and feature variables*.

Our first step is to acknowledge that the information contained in the feature and in the training data, relevant to the class of the new unit, is simply inaccessible to us because of computational limitations. We do have access to the output for the new unit, however, which does carry relevant information. Thus what we can do is to calculate the probability

P(class of new unit | output for new unit)   (2)

for each class.

This idea can also be informally understood in two ways. First: the classifier's output is regarded as a proxy for the feature. Second: the classifier is regarded as something analogous to a *diagnostic test*, such as any common diagnostic or prognostic test used in medicine for example. We do not take diagnostic-test results at face value – if a flu test is

---

[12] Jaynes 2003; Russell & Norvig 2022 chs 12–13; Gregory 2005; Hailperin 2011; Jeffreys 1983:see further references in appendix B. [13] Bernardo & Smith 2000 ch. 4; Dawid 2013.
[14] to understand this trade-off in the case of neural-network classifiers see e.g. MacKay 1992b,c,a; Murphy 2012 § 16.5 esp. 16.5.7; see also the discussion by Self & Cheeseman 1987.

'positive' we do not conclude that the patient has the flu – but rather arrive at a probability that the patient has the flu, given some statistics about results of tests performed on 'verified samples' of true-positive and true-negative patients[15].

2.2 Calibration data

To calculate the probability above, however, it is necessary to have examples of further pairs (class of unit, output for unit), of which the new unit's pair can be considered a 'representative sample'[16] and vice versa – exactly for the same reason why we need training data in the first place to calculate the probability of a class given the feature. Or, with a more precise term, the examples and the new unit must be *exchangeable*[17].

For this purpose, can we use the pairs (class of unit, output for unit) of the training data? This would be very convenient, as those pairs are readily available. But answer is no. The reason is that the outputs of the training data are produced from the features *and the classes* jointly; this is the very point of the training phase. There is therefore a direct informational dependence between the classes and the outputs of the training data. For the new unit, on the other hand, the classifier produces its output from the feature alone. *As regards the probabilistic relation between class and output, the new unit is not exchangeable with (or a representative sample of) the training data.*

We need a data set where the outputs are generated by simple application of the algorithm to the feature, as it would occur in its concrete use, and the classes are known. The *test data* of standard machine-learning procedures are exactly what we need. The new unit can be considered exchangeable with the test data. We rename such data 'transducer-calibration data', owing to its new purpose.

The probability we want to calculate is therefore

$$P(\text{class of new unit} \mid \text{output for new unit, classes \& outputs of calibr. data}) . \tag{3}$$

For classification algorithms that output a quantity much simpler than the features, like a vector of few real components for instance, the probability above can be exactly calculated. Thus, once we obtain the

---

[15] Sox et al. 2013 ch. 5; Hunink et al. 2014 ch. 5; see also Jenny et al. 2018. [16] for a critical analysis of the sometimes hollow term 'representative sample' see Kruskal & Mosteller 1979a,b,c; 1980. [17] Lindley & Novick 1981.

classifier's output for the new unit, we can calculate a probability for the new unit's class.

The probability values (4), for a fixed class and variable output, constitute a sort of 'calibration curve' (or hypersurface for multidimensional outputs) of the output-to-probability transducer for the classifier. See the concrete examples of figs 1 on page 15, and 3 on page 17. It must be stressed that such curve needs to be calculated only once, and it can be used for all further applications of the classifier to new units.

What is the relation between the ideal incomputable probability (1) and the probability (4) obtained by proxy? If the output y of the classifier is already very close to the ideal probability (1), or a monotonic function thereof, isn't the proxy probability (4) throwing it away and replacing it with something different? Quite the opposite. Owing to de Finetti's theorem, if the output y is almost identical with the ideal probability, then it becomes increasingly close to the frequency distribution of the training data, as their number increases (see appendix B.1); the same happens with the proxy probability and the frequency distribution of the calibration data. But these two data sets should be representative of each other and of future data – otherwise we would be 'learning' from irrelevant data – and therefore their frequency distributions should also converge to each other. Consequently, by transitivity we expect the proxy probability to become increasingly close to the output y. Actually, if the output is not exactly the ideal probability (1) but a monotonic function of it, the proxy probability (4) will reverse such monotonic relationship, giving us back the ideal probability.

Obviously all these considerations only hold if we have good training and calibration sets, exchangeable with (representative of) the real data that will occur in our application.

Since we are using as calibration data the data traditionally set aside as 'test data' instead, an important question arises. Do we then need a third, separate test dataset for the final evaluation and comparison of candidate classifiers or hyperparameters? This would be inconvenient: it would reduce the amount of data available for training.

The answer is no: *the calibration set automatically also acts as a test set*. It may be useful to explain why this is the case, especially for those who may mistakenly take for granted the universal necessity of a test set.

Many standard machine-learning methodologies need a test set because they are only an approximation of ideal inference performed with the probability calculus. The latter needs no division of available data into different sets: something analogous to such division is automatically made internally, so to speak. It can be shown[18] that the mathematical operations behind the probability rules correspond to making *all possible* divisions of available data between 'training' and 'test', as well as all possible cross-validations with folds of all orders. This is the reason why ideal inference by means of the probability calculus is almost computationally impossible in some cases, and we have to resort to approximate but faster machine-learning methods. The latter typically do not do such data partitions automatically; the latter need to be made – and only approximately – by hand.

🖉 explain how testing is done

2.3   Calculation of the probabilities

Let us denote by c the class value of a new unit, by y the output of the classifier for the new unit, and by $D := \{c_i, y_i\}$ the classes and classifier outputs for the transducer-calibration data.

It is more convenient to focus on the *joint* probability of class and output given the data,

$$p(c, y \mid D) \,, \tag{4}$$

rather than on the conditional probability of the class given the output and calibration data, (4).

The joint probability is calculated using standard non-parametric Bayesian methods[19]. 'Non-parametric' in this case means that we do not make any assumptions about the shape of the probability curve as a function of c, y (contrast this with logistic regression, for instance), or about special independence between the variables (contrast this with naive-Bayes). The only assumption made – and we believe it is quite realistic – is that the curve must have some minimal degree of smoothness. This assumption allows for much leeway, however: fig.🖉 ... for instance shows that the probability curve can still have very sharp bends, as long as they are not cusps.

---

[18] Porta Mana 2019; Fong & Holmes 2020; Wald 1949; many examples of this fact are scattered across the text by Jaynes 2003.   [19] for introductions and reviews see e.g. Walker 2013; Müller & Quintana 2004; Hjort 1996.

Non-parametric methods differ from one another in the kind of 'coordinate system' they select on the infinite-dimensional space of all possible probability curves, that is, in the way they represent a general positive normalized function.

We choose the representation discussed by Dunson & Bhattacharya[20]. The end result of interest in the present section is that the probability density $p(c, y \mid D)$, with c discrete and y continuous and possibly multi-dimensional, is expressed as a sum

$$p(c, y \mid D) = \sum_k q_k\, A(c \mid \alpha_k)\, B(y \mid \beta_k) \tag{5}$$

of a finite but large number of terms. Each term is the product of a positive weight q_k, a probability distribution $A(c \mid \alpha_k)$ for c depending on parameters α_k, and a probability density $B(c \mid \beta_k)$ for y depending on parameters β_k. These distributions are chosen by us according to convenience; see the appendix B.1 for further details. The parameter values can be different from term to term, as indicated by the index k. The weights $\{q_k\}$ are normalized. For simplicity we shall from now on omit the dependence '$\mid \dots, D$)' on the calibration data, leaving it implicitly understood.

This mathematical representation can approximate (under some norm) any smooth probability density in c and y. It has the advantages of being automatically positive and normalized, and of readily producing the marginal distributions for c and for y:

$$p(c) = \sum_k q_k\, A(c \mid \alpha_k)\,, \qquad p(y) = \sum_k q_k\, B(y \mid \beta_k)\,, \tag{6}$$

from which also the conditional distributions are easily obtained:

$$\boxed{p(c \mid y) = \sum_k \frac{q_k\, B(y \mid \beta_k)}{\sum_l q_l\, B(y \mid \beta_l)}\, A(c \mid \alpha_k)} \tag{7a}$$

$$p(y \mid c) = \sum_k \frac{q_k\, A(c \mid \alpha_k)}{\sum_l q_l\, A(c \mid \alpha_l)}\, B(y \mid \beta_k)\,. \tag{7b}$$

In the rest of the paper we shall use formula (7a), the probability of the class given the algorithm's output, as typically done with discriminative algorithms.

---

[20] Dunson & Bhattacharya 2011; see also the special case presented by Rasmussen 1999.

The weights and parameters $\{q_k, \alpha_k, \beta_k\}$ are the heart of this representation, because the shape of the probability curve $p(c \mid y, D)$ depends on their values. They are determined by the test data D. Their calculation is done via Markon-chain Monte Carlo sampling, discussed in appendix B.2. For low-dimensional y and discrete c (or even continuous, low-dimensional c, which means we are working with a regression algorithm), this calculation can be done in a matter of hours, and *it only needs to be done once*.

Once calculated, these parameters are saved in memory and can be used to compute any of the probabilities (5), (6), (7) as needed, as discussed in the next subsection. Such computations take less than a second.

Note that the role of the classifier in this calculation is simply to produce the outputs y for the calibration data, after having been trained in any standard way on a training data set. No changes in its architecture or in its training procedure have been made, nor are any required.

3   Utility-based classification

We refer to our companion work Dyrland et al. 2022a, § 2, for a more detailed presentation of decision theory and for references. In the following we assume familiarity with the material presented there.

Our classification or decision problem has a set of decisions, which we can index by $i = 1, 2, \dots$. As discussed in § 1, these need not be the same as the possible classes; the two sets may even be different in number. But the true class, which is unknown, determines the *utility* that a decision yields. If we choose decision i and the class c is true, our eventual utility will be U_{ic}.[21] These utilities are assembled into a rectangular matrix (U_{ic}) with one row per decision and one column per class. Note that the case where decisions and classes are in a natural one-one correspondence, as in the cat-vs-dog classification example of § 1, is just a particular case of this more general point of view. In such a specific case we may replace 'decision' with 'class' in the following discussion, and the utility matrix is square.

Now let us consider the application of the algorithm, with the probabilities calculated in the preceding section, to a new unit.

---

[21] We apologize for the difference in notation from our companion work, where the class variable is 'j' and the utilities 'U_{ij}'

1. Fed the unit's features to the classifier, which outputs the real value y.

2. Calculate $p(c \mid y)$, for each value of c, from formula (7a), using the parameters $\{q_k, \alpha_k, \beta_k\}$ stored in memory. These are the probabilities of the classes, which are collected in a column vector (p_c).

3. The expected utility $\bar{U}_i$ of decision i is given by the matrix product of an appropriate utility matrix (U_{ic}) and the column vector (p_c):

$$\bar{U}_i := \sum_c U_{ic}\, p_c \ . \tag{8}$$

4. Choose the decision i^* having largest $\bar{U}_i$, according to the principle of maximum expected utility:

$$\boxed{\text{choose} \quad i^* = \arg\max_i \{\bar{U}_i\} \equiv \arg\max_i \left\{\sum_c U_{ic}\, p_c\right\}} \tag{9}$$

We call the procedure above, especially steps 2.–4., the *augmentation* of the classifier.

In step 2. we have effectively translated the classifier's raw output into a more sensible probability. From this point of view the function $p(c \mid y)$ can be considered as a more appropriate substitute of the softmax function, for instance, at the output of a neural network (compare fig. 3).

The matrix multiplication of and subsequent selection of steps 3.–4. are computationally inexpensive; they can be considered as substitutes of the 'argmax' selection that typically happen at the continuous output of a classifier.

It should be noted that the utilities U_{ic} used in step 3. can either be the same for each new unit, or different from unit to unit. The augmentation procedure is therefore extremely flexible, at no additional computational cost.

4   Demonstration

4.1   Overview

We illustrate the implementation of the probability transducer and its combination with utility-based decisions in a concrete example. The

evaluation of the results is also made from the standpoint of decision theory, using utility-based metrics, as explained in our companion paper[22].

A couple of remarks may clarify the purpose of this illustration and our choice of classification problem.

The internal consistency of decision theory guarantees that utility-based decisions always improve on, or at least give as good results as, any other procedure, including the standard classification procedures used in machine learning. This is intuitively obvious: we are, after all, grounding our single class choices upon the same gains & losses that underlie our classification problem and that are used in its evaluation. The present illustration is therefore not a proof for such improvement – none is needed. It is a reassuring safety check, though: if the results were negative it would mean that errors were made in applying the method or in the computations.

Rather than looking for some classification problem and dataset on which the decision-theoretic approach could lead to astounding improvements, we choose one where machine-learning classifiers already give excellent results, therefore difficult to improve upon; and which is characterized by a naturally high class imbalance. The classification problem is moreover of interest to us for other ongoing research projects.

The binary-classification task is a simplified version of an early-stage drug-discovery problem: to determine whether a molecule is chemically 'inactive' (class 0) or 'active' (class 1) towards one specific target protein.

Two machine-learning classifiers are considered: a Random Forest and a residual Convolutional Neural Network (ResNet), details of which are given in appendix ✎ …. The random forest takes as input a set of particular physico-chemical characteristics of a molecule, and outputs a real number in the range $[0, 1]$, corresponding to the fraction of decision trees which vote for class 1, 'active'. The convolutional-neural-network takes as input an image representing the chemical and spatial structure of the molecule, and outputs two real numbers roughly corresponding to scores for the two classes.

We use data from the ChEMBL database[23], previously used in the literature for other studies of machine-learning applications to drug discovery[24]. One set with 60% of the data is used to train and validate

---

[22] Dyrland et al. 2022a.   [23] Bento et al. 2014.   [24] Koutsoukas et al. 2017.

the two classifiers. One set with 20% is used for the calibration of the probability transducer and evaluation of the classifiers. One further data set with 20% is here used as fictive 'real data' to illustrate the results of our procedure; we call this the 'demonstration set'.

Note that *the additional demonstration dataset has an illustrative purpose only* for the sake of the present example. In a real design & evaluation of a set of candidate classifiers, the calibration set will at the same time be the evaluation test set, and no further data subset will be necessary, as explained in § 2.2.

In all data sets, class 0 ('inactive') occurs with a 91% relative frequency, and class 1 ('active') with 9%; a high class imbalance.

Technical details about the setup and training of the two classifiers and of the calculation of the probability-transducer parameters are given in appendix B.2.

4.2   Probability-transducer curves

Random forest

The joint probability of class c and output y, eq. (5), for the random forest is expressed by the sum

$$p(c, y) = \sum_k q_k \left[c \; \alpha_k + (1 - c) \; (1 - \alpha_k) \right] \mathrm{N}(y \mid \mu_k, \sigma_k) \qquad (10)$$

where $\mathrm{N}(\cdot)$ is a Gaussian as in eq. (27). The sum contains $2^{18} \approx 260\,000$ terms.

Figure 1 shows the probabilities of classes 1 and 0 conditional on the random-forest output: p(class 1 | output) and p(class 0 | output). It also shows the range of variability that these probabilities could have if more data were used for the calibration: with a 75% probability they would remain within the shaded regions. This variability information is provided for free by the calculation; we plan to discuss and use it more in future work.

The probabilities increase (class 1) or decrease (class 0) monotonically up to output values of around 0.9. The minimum and maximum probabilities are 0.14% and 92.9%; these values will be important for a later discussion. The output, if interpreted as a probability for class 1 ('active'), tends to be too pessimistic for this class (and too optimistic for the other) in a range from roughly 0.25 to 0.95; and too optimistic
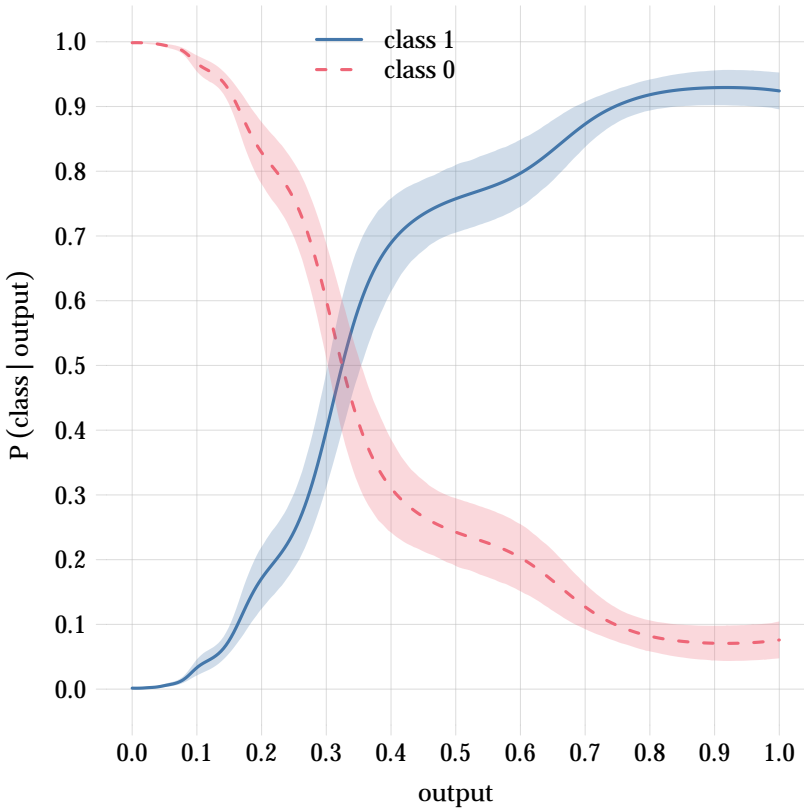
Figure 1   Probabilities of class 1 ('active', blue solid curve) and class 0 ('inactive', red dashed curve) conditional on the random-forest output. Their extremal values are 0.0014 and 0.929. The shaded region around each curve represents its 12.5%–87.5% range of possible variability if more data were used to calculate the probabilities.

outside this range. For instance, for an output of 0.3 the probability for class 1 is 40%; for an output of 1 the probability for class 1 is 92%.

Figure 2 shows the 'generative' probability densities of the random-forest output conditional on each class, p(output | class 1) and p(output | class 0). These probabilities densities are also provided for free by the parameter calculation, as explained in § 2.3. The shaded regions are again 75% intervals of possible variability upon increase of the calibration dataset.

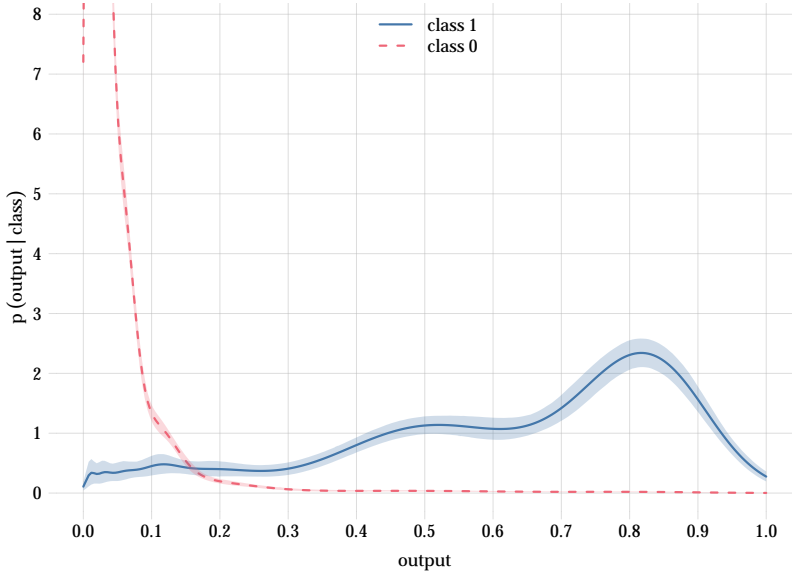There is a high probability of output values close to 0 when the true

Figure 2   Probability densities of the random-forest output conditional on class 1 ('active', blue solid curve) and on class 0 ('inactive', red dashed curve, truncated). The shaded region around each curve represents its 12.5%–87.5% range of possible variability upon increase of the calibration dataset.

class is 0 ('inactive'), and a peak density around 0.8 when the true class is 1 ('active'), as expected. The density conditional on class 0 is narrower than the one conditional on class 1 owing to the much larger proportion data in the former class. Intuitively speaking, we have seen that most data in class 1 correspond to high output values, but we have seen too few data in this class to reliably conclude, yet, that future data will show the same correspondence.

Convolutional neural network

The joint probability of class c and the bivariate output $y \equiv (y_0, y_1)$ of the convolutional neural network is expressed by the sum

$$p(c, y_0, y_1) = \sum_k q_k \left[c\, \alpha_k + (1 - c)\, (1 - \alpha_k) \right] \mathrm{N}(y_0 \mid \mu_{0k}, \sigma_{0k})\, \mathrm{N}(y_1 \mid \mu_{1k}, \sigma_{1k}) \quad (11)$$
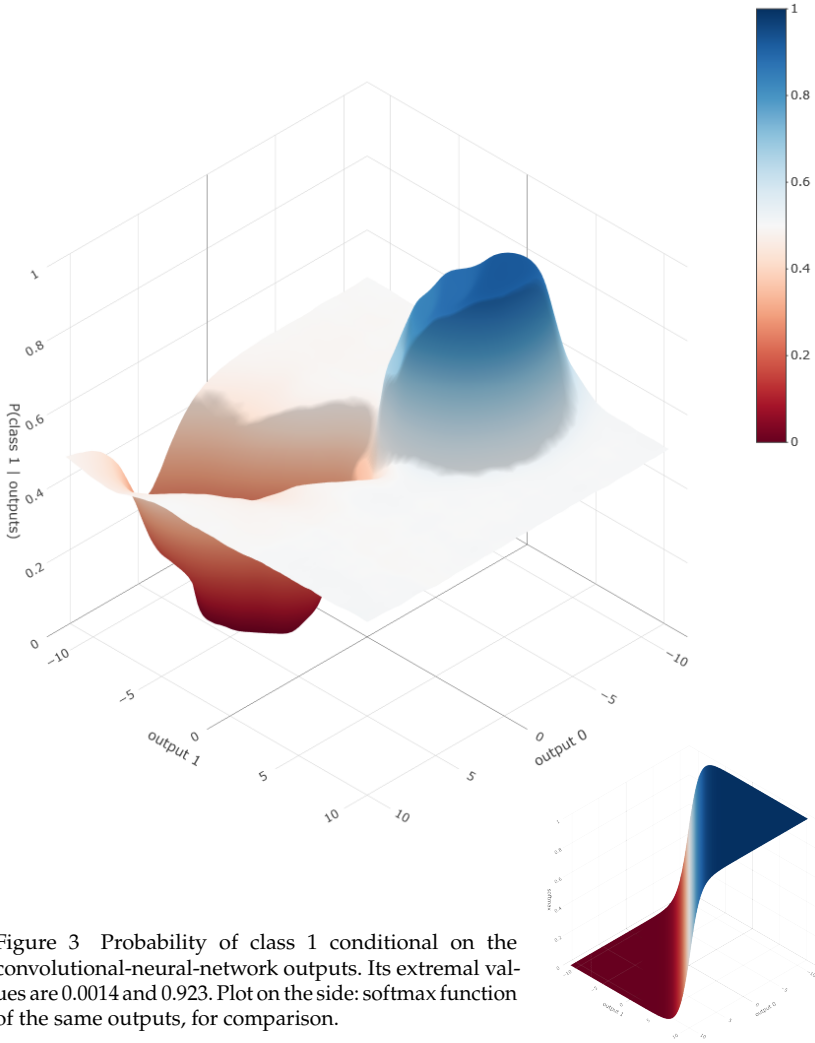
Figure 3 Probability of class 1 conditional on the convolutional-neural-network outputs. Its extremal values are 0.0014 and 0.923. Plot on the side: softmax function of the same outputs, for comparison.

containing again $2^{18} \approx 260\,000$ terms, and with parameters analogous to those of eq. (10).

Figure 3 shows the probability of class 1 conditional on the bivariate output of the convolutional neural network, p(class 1|outputs). Its extremal values are 0.14% and 92.3%. It is interesting to compare this probability with the softmax function of the outputs, shown in the smaller side plot,
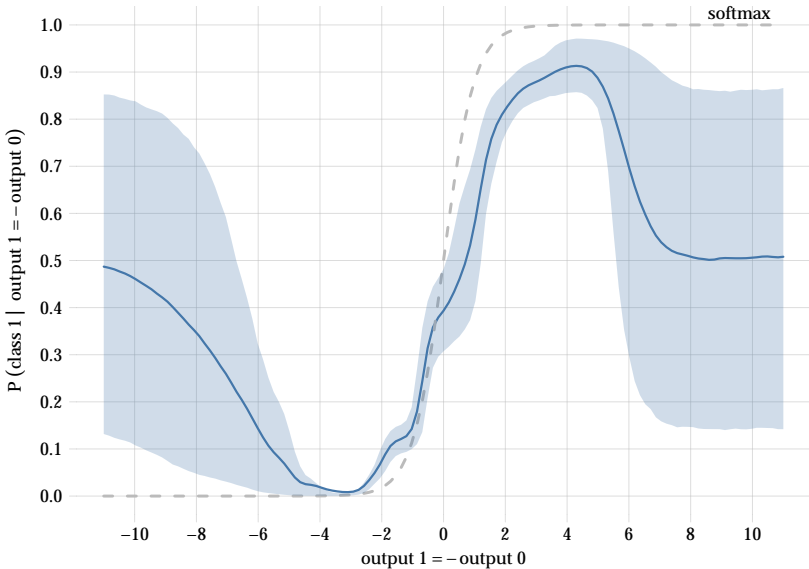
Figure 4   Cross-section of the probability surface of fig. 3 across the bisector of the II and IV quadrants of output space. The shaded region represents the 12.5%–87.5% range of possible variability upon increase of the calibration dataset. The cross-section of the softmax function (grey dashed curve) is also shown for comparison.

typically used as a proxy for the probability.

A cross-section of this probability surface along the bisector of the II and IV quadrants of the output space is shown in fig. 4, together with the cross-section of the softmax. The probability takes on extremal values, around 1% and 90%, only in very narrow ranges, and quickly returns and extrapolates to 50% everywhere else. The softmax, on the other hand, extrapolates to extreme probability values – a known problem of neural networks[25]. The conservative extrapolation of the transducer is also reflected in the 75% interval of possible variability of the probability (shaded region), which becomes extremely wide at the extremities.

4.3   Results on demonstration data

The essential point of the decision-theoretic approach is that we first need to specify the utilities involved in the classification problem, because they

---

[25] Gal & Ghahramani 2016.

determine (i) together with the probabilities, which class we choose in each single instance; (ii) the metric to evaluate a classifier's performance. The utilities are assembled into a utility matrix which we write in the format

$$\begin{array}{cc} & \text{true class} \\ & \begin{array}{cc} 0 & \quad\ 1 \end{array} \\ \text{decision} \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} \text{True 0} & \text{False 0} \\ \text{False 1} & \text{True 1} \end{bmatrix} \end{array}. \tag{12}$$

We call *equivalent* two utility matrices that differ by a constant additive term and a positive multiplicative term, since changes in the zero or unit of measurement of utilities do not affect comparative evaluations.

For illustration we choose four utility matrices:

$$\begin{array}{cccc} \text{utility case I} & \text{utility case II} & \text{utility case III} & \text{utility case IV} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix} & \begin{bmatrix} 1 & -100 \\ 0 & 100 \end{bmatrix} & \begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix} \end{array}. \tag{13}$$

Case I represents any case where the correct classification of either class is equally valuable; and the incorrect classification, equally invaluable. Note that this utility matrix is equivalent to any other of the form $\begin{bmatrix} a & b \\ b & a \end{bmatrix}$ with $a > b$. Accuracy is the correct metric to evaluate this case. Case II represents any case where the correct classification of class 1, 'active', is ten times more valuable than that of class 0 'inactive', and its incorrect classification is as damaging as correct classification is valuable. The remaining two cases are interpreted in an analogous way. The 'value' could simply be the final average monetary revenue at the end of the drug-discovery project that typically follows any of these four situations. Of particular relevance to drug discovery, where false positives are known to be especially costly[26], is the utility matrix of case II and possibly that of case III.

We consider each of these utility matrices, in turn, to be the one underlying our classification problem. In each case we perform the classification of every item – a molecule – in the demonstration data as follows:

1. feed the features of the item to the classifier and record its output

2. feed this output to the probability transducer and record the resulting probability for class 1; form the normalized probability vector for the two classes

---

[26] Sink et al. 2010; Hingorani et al. 2019.

3. multiply the probability vector by the utility matrix to determine the expected utility of each class choice, eq. (8)

4. choose the class with higher expected utility (ties have to be decided unsystematically, to avoid biased results), eq. (9).

Confusion matrices – and a peculiar situation

Once all items in the demonstration dataset are classified, we compare their chosen classes with their true ones and compute the resulting confusion matrix, which we also write in the format (12). The confusion matrices for all cases, methods, and algorithms are presented in table 1 on page 22. Ties (both classes were equally preferable) are solved by giving half a point to each class.

The standard method produces the same confusion matrix in all four utility cases because it does not use utilities to choose a class. The augmentation produces instead a different confusion matrix in each utility case: even if the class probabilities for a give datum are the same in all cases, the threshold of acceptance varies so as to always be optimal for the utilities involved.

A peculiar case of this automatic optimization of the threshold is visible for the transducer applied to either algorithm in case IV: it leads to the confusion matrix

$$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix} \tag{14}$$

which means that *all* items were classified as '0', 'inactive'. How can this happen? Let us say that the probabilities for class 0 and 1, determined by the transducer from the random-forest output, are $1 - p$ and p. In case IV, the expected utilities of choosing class 0 or 1 are given by the matrix multiplication $\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix} \begin{bmatrix} 1-p \\ p \end{bmatrix}$:

$$\begin{aligned} \text{choose 0: expect} \quad & 10 \cdot (1 - p) + 0 \cdot p = 10 - 10\,p \ , \\ \text{choose 1: expect} \quad & -10 \cdot (1 - p) + 1 \cdot p = -10 + 11\,p \ . \end{aligned} \tag{15}$$

It is optimal to choose class 1 only if (disregarding ties)

$$-10 + 11\,p > 10 - 10\,p \quad \text{or} \quad p > 20/21 \approx 0.952 \ , \tag{16}$$

that is, only if the probability of class 1 is higher than 95%. The threshold is so high because on the one hand there's a high cost (−10) if the class

isn't actually 1, and on the other hand a high reward (10) if the class is actually 0. Now, a look at the transducer curve for the random forest, fig. 1, shows that the transducer never gives a probability higher than 93% to class 1. Similarly the transducer for the convolutional neural network, fig. 3, never reaches probabilities above 92%. So the threshold of 95% will never be met in either case, and no item will be classified as 1. It is simply never rewarding, on average, to do so[27]. It can be seen that this situation will occur with any utility matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, with $a > c$ and $d > b$, such that $\frac{a-c}{a-c+d-b} \approx 0.93$.

This peculiar situation has a notable practical consequence. We have found the transducer curve for a classifier, and see that its maximum probability for class 1 is 93%. We have assessed that the utilities involved are $\left[\begin{smallmatrix} 10 & 0 \\ -10 & 1 \end{smallmatrix}\right]$, so the threshold to classify as 1 is 95%. Then we immediately find that *there is no need to employ that classifier, in this utility case*: it is simply more profitable to automatically treat all data as class 0.

A look at the other confusion matrices of table 1 shows the effect of the automatic threshold optimization also in utility cases II and III: as the cost of misclassifying true class 1 increases (-10 or -100), the number of such misclassifications decreases. It does so at the detriment of class 0 classification, which, however, incurs lower costs and benefits.

Utility yields

We can finally assess the performance of both classifiers, with and without augmentation, on the demonstration dataset. As explained in our companion work[28] and summarized in § 3, the correct metric for such performance must naturally depend on the utilities that underlie the problem. It is the utility yield per datum produced by the classifier on the dataset, obtained by taking the grand sum of the products of the homologous elements of the utility matrix (U_{ij}) and the confusion matrix (C_{ij}):

$$\sum_{ij} U_{ij}\, C_{ij} \,. \tag{17}$$

The utility yields for the different cases, classifiers, and methods are presented in table 2. The maximum and minimum theoretically

---

[27] Drummond & Holte 2005 cf. the analysis by.   [28] Dyrland et al. 2022a.

		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & -100 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
Random Forest	standard method		$\begin{bmatrix} 3225 & 79.5 \\ 37 & 246.5 \end{bmatrix}$		
	augmentation	$\begin{bmatrix} 3207 & 38 \\ 55 & 288 \end{bmatrix}$	$\begin{bmatrix} 3050 & 7 \\ 212 & 319 \end{bmatrix}$	$\begin{bmatrix} 2358 & 2 \\ 904 & 324 \end{bmatrix}$	$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix}$
Neural Network	standard method		$\begin{bmatrix} 3165 & 49 \\ 97 & 277 \end{bmatrix}$		
	augmentation	$\begin{bmatrix} 3189 & 65 \\ 73 & 261 \end{bmatrix}$	$\begin{bmatrix} 2882 & 12 \\ 380 & 314 \end{bmatrix}$	$\begin{bmatrix} 2011 & 1 \\ 1251 & 325 \end{bmatrix}$	$\begin{bmatrix} 3262 & 326 \\ 0 & 0 \end{bmatrix}$

Table 1   Confusion matrices from demonstration dataset

		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & -100 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
	min achievable utility	0	−0.91	−9.09	−9.09
	max achievable utility	1	1.82	9.99	9.18
Random Forest	standard method	0.968	1.36	5.55	8.95
	augmentation	**0.974**	**1.72**	**9.63**	**9.09**
Neural Network	standard method	0.959	1.52	7.24	8.63
	augmentation	**0.962**	**1.64**	**9.59**	**9.09**

Table 2   Utility yields from demonstration dataset

Random Forest	standard method	0.968	0.834	0.767	0.988
	augmentation	**0.974**	**0.964**	**0.981**	**0.995**
Neural Network	standard method	0.959	0.890	0.855	0.970
	augmentation	**0.962**	**0.937**	**0.979**	**0.995**

Table 3   *Rescaled* utility yields from demonstration dataset

achievable yields, which are obtained when all data are correctly classified or incorrectly misclassified, are also shown for each case. Since the maximum and minimum differ from case to case, we also report the *rescaled* utilities in table 3. For each case, the rescaled utility is obtained by a shift and a scaling of its measurement unit, which set the minimum and maximum achievable yields to 0 and 1:

$$\text{rescaled utility} = \frac{\text{utility} - \text{theoretical min}}{\text{theoretical max} - \text{theoretical min}} . \tag{18}$$

Let us first compare the two algorithms when employed in the standard way (red). Their performance is very close to the theoretical maximum in most cases, the only exception being the random forest in case III. The random forest outperforms the convolutional neural network in cases I and II; vice versa in cases II and III.

Then let us look at the performances obtained with the augmentation (blue bold). We note the following:

- The augmentation improves the performance of each algorithm in all cases. The improvement also occurs also in case IV for the random forest, where the standard method already had an extremely high performance (rescaled utility of 0.988).

- In cases II–IV the augmentation improves the originally worse algorithm above the originally better one. This almost happens also in case I, the difference possibly being at the level of statistical variability.

- In case IV the augmentation brings the utility yield to above 99% of the theoretical maximum; remember from the previous section that this is achieved by classifying all data as class 0.

- With augmentation, the random forest outperforms the convolutional neural network in all cases, with a possible tie for case IV.

These were four particular cases only, though. Does the augmentation lead to an improvement (or at least to no change) for *all* possible utility matrices? This must indeed be the case, owing to the internal consistency of decision theory.

We give evidence of this fact by considering a large number (10 000) of utility matrices selected uniformly from the utility-matrix space for
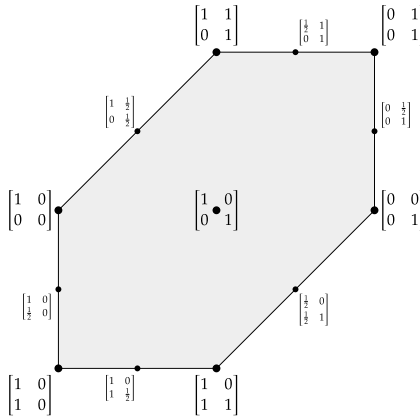
Figure 5   Space of utility matrices (modulo equivalence) for binary classification.

binary classification. This two-dimensional space, shown in fig. 5, is discussed in our companion work[29].

For each of these utility matrices, we calculate the rescaled yields obtained by using either classifier in the standard way, and with the augmentation – probability-transducer & utility-based classification. The yields obtained in the two ways are plotted against each other in fig. 6. Only 2 000 are plotted, in order to show the fact that most of the yields are quite high (they accumulate on the top right corner); but all 10 000 lie along the jagged curves evident in the plot.

Clearly, the augmentation always leads to increased utility yields, especially for those cases where the standard performance of the two algorithms is particularly low or high (in relative terms); hence the U-shapes of the scattered points. Note that the minimum values of the plot's axes is 0.75, so the improvement is upon yields that are already quite high. Numerical rounding can lead to apparent decreases in some cases (maximum relative decreases: -0.09% for random forest, -0.2% for convolutional neural network); this is actually useful for estimating the magnitude of numerical-rounding errors occurring in the computation and the number of significant digits.

🔧 add note about generalization to other decisions
🔧 add final remark that this is not evaluation

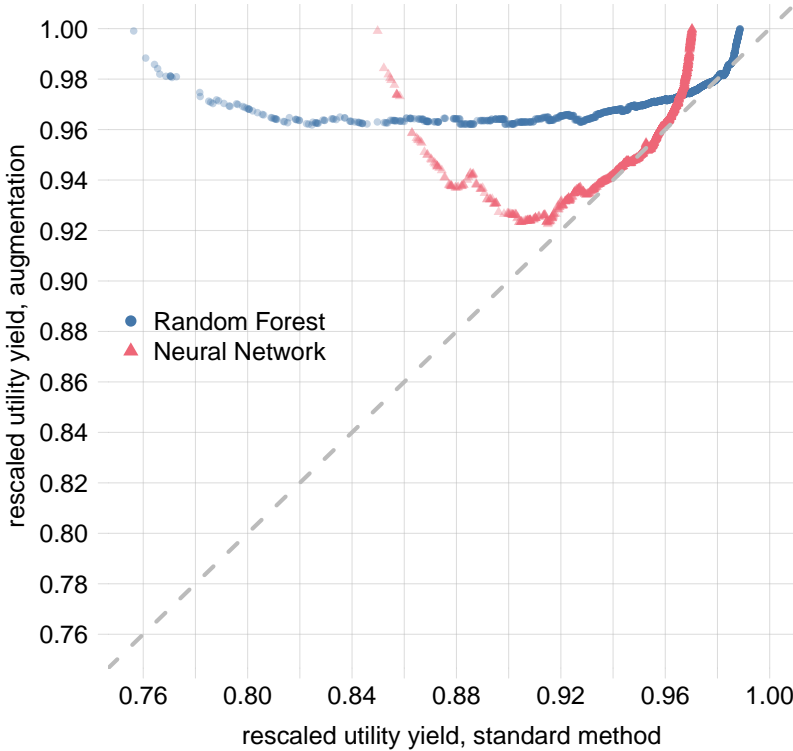[29] Dyrland et al. 2022a § 3.2.

Figure 6   Rescaled utility yields obtained using the two classifiers in the standard way, vs those obtained with augmentation, for a uniform distribution of possible utility matrices over the utility-matrix space of fig. 5. The augmentation always leads to an improved utility yield, especially in cases where the standard method has a low or high performance. Owing to noise coming from numerical rounding, in some cases the yield from augmentation may appear lower than from the standard method (points below the dashed grey line).

5   Additional uses of the probability-transducer: an overview

The probability-transducer presented in § 2 and illustrated in the previous section has several other uses and advantages, all of which come for free or almost for free with its calculation. We give a brief overview of them in the present section, leaving a more thorough discussion and applications to future works.

25

The additional uses are mainly three:

- Quantification of the possible variability of the transducer probability curve.

- Evaluation of the optimal algorithm, including the uncertainty about such evaluation.

- 'Generative use' of the augmented algorithm, even if the original algorithm is not designed for generative use.

5.1   Variability of the transducer probability curve

The output-to-probability function, eq. (4), such as those plotted in figs 1 and 3–4, is determined by the data in the calibration set. There is the question, then, of how the function could change if we used more calibration data. Such possible variability could be of importance. For example, we may find that the transducer only yields class probabilities around 0.5, and wonder whether this is just a statistical effect of a too small calibration dataset, or whether it would persist even if we used more calibration data.

The calculation of the transducer parameters automatically tells us the probabilities of these possible variations, in the form of a set of possible alternative transducer curves, from which we can for example calculate quantiles. The shaded regions in figs 1, 2 and 4 are examples of such probability intervals. Their calculation is sketched in appendix B.3.

5.2   Expected utility of the classifying algorithm

At the end of the discussion about the calibration dataset, § 2.2, we gave our assurances that no additional data must be set apart – with a detrimental reduction in training data – for evaluation or testing purposes. This is because from the probabilities (4), obtained from the calibration data, we can also calculate *the expected, future utility yield of the augmented algorithm*, once we have specified the utility matrix underlying the particular application. More details about this calculation, which amounts to a low-dimensional integration, are given in appendix B.4.

For the random forest and convolutional neural network of the demonstration § 4, for instance, this calculation gives the expected utilities (non-rescaled) of table 4. The augmented random forest is expected to be optimal for cases I and II, possibly also in case III, although

	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -10 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & -100 \\ 0 & 100 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ -10 & 1 \end{bmatrix}$
Random Forest	0.973	1.68	9.59	9.08
Neural Net	0.962	1.62	9.56	9.08

Table 4   Expected utilities for the two algorithms of § 4

case I

case II

case III

case IV
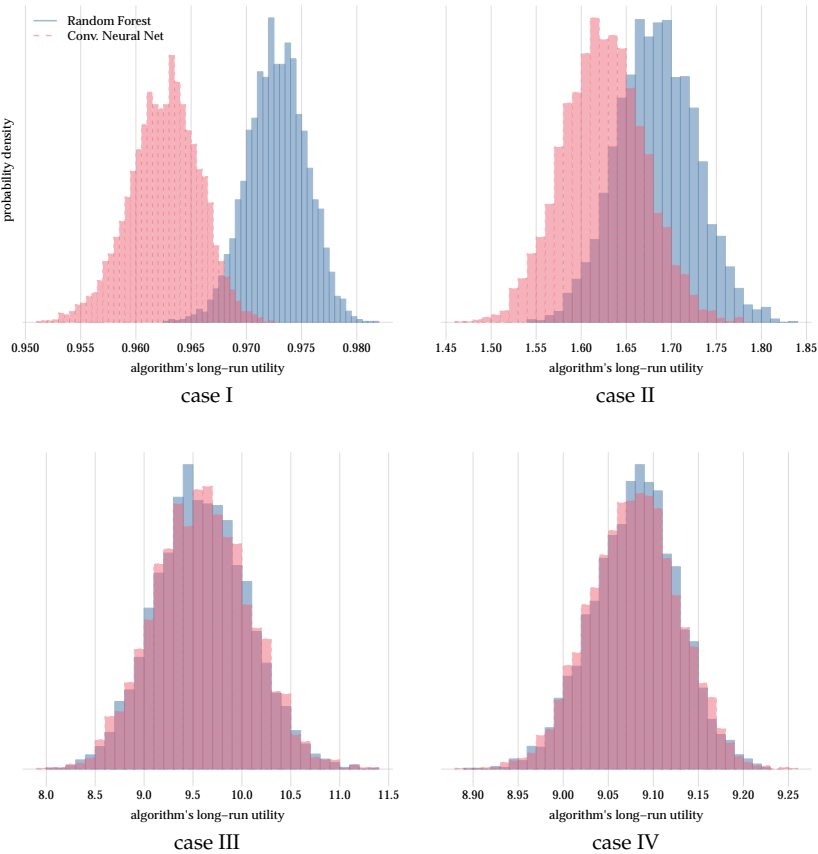
Figure 7   Probability distributions of the long-run utility yields of random forest and convolutional neural network in the four cases of § 4

the difference in utilities is likely affected by numerical-precision error. There is no preference in case IV.

Let us emphasize again that these values are obtained *from the parameters of the transducer curve, without the need of any additional dataset*. The demonstration dataset discussed in § 4.1 was not used for their calculation. The results from that dataset, reported in table 2, corroborate these values.

One may ask: but how can you be sure that what you basically found from the calibration data will generalize to new data? The answer goes back to the discussion at the end of § 2.2, about how the probability calculus works, and to the technical details explained in appendix B: the probability calculus automatically considers all possible sets of new data that could be encountered in the future application.

In fact, the calculation of an algorithm's expected utility automatically produces a probability distribution of the possible long-run yields the algorithm could give. The distributions for the long-run utilities of the random forest and the convolutional neural network in cases I–IV of § 4 are shown in fig. 7. We see that in case I the random forest will very probably (and the actual probability can be easily calculated) be superior to the convolutional neural network. In case II the probability is somewhat lower. In cases III and IV it is completely uncertain which algorithm will be best.

The evaluation of candidate classifiers' performances and their uncertainties are obviously extremely important for the choice and final deployment of the optimal classifier.

5.3   Discriminative and generative modes

The transducer parameters, calculated as discussed in § 2.3 and appendix B, allow us to calculate not only the 'discriminative' probability of the class given the algorithm's output, formula (7a), but also the inverse, 'generative' probability of the output given the class, formula (7b).

Generative mode: probability of class given output and base rates
    We calculate $p(y \mid c)$ from formula (7b), again for each value of c. The probability of each class c is then obtained through Bayes's theorem after supplying the *population prevalence* r_c of the class:

$$p(c \mid y, \text{prevalences}) = \frac{p(y \mid c)\, r_c}{\sum_c p(y \mid c)\, r_c} \ . \tag{19}$$

The population prevalences[30], also called *base rates*[31], are the relative frequencies of occurrence of the various classes in the population whence our unit originates. This notion is very familiar in medicine and epidemiology. For example, a particular type of tumour can have a prevalence of 0.01% among people of a given age and sex, meaning that 1 person in 10 000 among them has that kind of tumour, as obtained through a large survey.

🔧 maybe move parts below to independent section

The generative mode, generally not available for classifiers with a discriminative design, is the required way to calculate the class probabilities if the calibration set does not have the same class frequencies as the population on which the classifier will be employed – let us call the latter the 'real population'. For example, two classes may appear in a 50%/50% proportion in the calibration set but in a 90%/10% proportion in the real population. This frequency discrepancy can occur for several reasons. Examples: samples of the real population are unavailable or too expensive to be used for calibration purposes; the class statistics of the real population has suddenly changed right after the deployment of the classifier; it is necessary to use the classifier on a slightly different population; the sampling of calibration data was poorly designed.

6   Summary, discussion, and teasers of further developments

🔧 Maybe add note about how (sequential) decision theory was used during World War I; see Good (1950) around § 6.2

---

[30] Sox et al. 2013 ch. 3; Hunink et al. 2014 § 5.1.   [31] Bar-Hillel 1980; Axelsson 2000.

Appendices: mathematical and technical details

A Algorithms and data used in the demonstration

A.1 Data

The data comes from the open-access CheMBL bioactivity database[32]. The dataset used in the present work was introduced by Koutsoukas et al. (2017). The data consist in structure-activity relationships from version 20 of ChEMBL, with Carbonic Anhydrase II (ChEMBL205) as protein target.

A.2 Pre-processing

For our pre-processing pipeline, we use two different methods to represent the molecule, one for the Random Forest (RF) and one for the Convolutional Neural Network (CNN). The first method turns the molecule into a hashed bit vector of circular fingerprints called Extended Connectivity Fingerprints (ECFP)[33]. From our numerical analysis, there was little to no improvement using a 2048-bit vector over a 1024-bit vector.

For our convolutional neural network, the data is represented by converting the molecule into images of 224 pixels × 224 pixels. This is done by taking a molecule's SMILES (Simplified Molecular Input Line Entry System) string[34] from the dataset and converting it into a canonical graph structure by means of RdKit[35]. This differs from ECFP in that it represents the actual spatial and chemical structure (or something very close to it) of the molecule rather than properties generated from the molecule.

The dataset has in total 1631 active molecules and 16310 non-active molecules which act as decoys. For training, the active molecules are oversampled, as usually done with imbalanced datasets[36], to match the same number of non-active molecules.

A.3 Prediction

Virtual screening is the process of assessing chemical activity in the interaction between a compound (molecule) and a target (protein). The

---

[32] Bento et al. 2014.   [33] Rogers & Hahn 2010.   [34] David et al. 2020.   [35] Landrum et al. 2017.   [36] Provost 2000.

goal of the machine learning algorithms is to find structural features or chemical properties that show that the molecule is active towards the protein[37]. Deep neural networks have previously been shown to outperform random forests and various linear models in virtual high-throughput screening and in quantitative structure-activity relationship (QSAR) problems[38].

A.4   Chosen classifiers

The algorithms and methods used to create the models have previously been shown to give great results for a lot of different fields.

Random Forest

The first machine learning model used in the experiments is an RF model implemented in sci-kit learn[39]. RF is an ensemble of classifying or regression trees where the majority of votes is chosen as the predicted class[40]. It is known for being robust when dealing with a large number of features (as in our case), being resilient to over-fitting, and achieving good performance. And has already been shown to deliver powerful and accurate results in compound classification and QSAR analysis[41]. The following parameters were used when training the model:

Number of trees: 200

Criterion: Entropy

Max Features: Square root

Convolutional Neural Network

The second model is a pre-trained residual network (ResNet)[42] with 18 hidden layers trained on the well-known ImageNet dataset[43] by using the PyTorch framework[44]. ResNet has shown to outperform other pre-trained convolutional neural network models[45]. A ResNet with 34 hidden layers

---

[37] Green 2019.   [38] Koutsoukas et al. 2017.   [39] Pedregosa et al. 2011.   [40] Breiman 2001.
[41] Svetnik et al. 2003.   [42] He et al. 2016.   [43] Russakovsky et al. 2015.   [44] Paszke et al. 2019.
[45] He et al. 2016.

showed little to no performance gain, so we chose to go with the simpler model. The model is trained with the following hyperparameters:

Learning rate: 0.003

Optimization technique: Stochastic Gradient Descent

Activation Function: Rectified linear unit (ReLU)

Dropout: 50%

Number of epochs: 20

Loss function: Cross-entropy loss

A.5 Train, Validation, Test split

The data set is split into four parts:

- Training set: 45% of the dataset to train the model.
- Validation set: 15%, for validating the model after each epoch.
- Calibration set: 20%, for calibrating the probability transducer.
- Test set: 20%, for evaluation.

B   Mathematical details and computation of the transducer

The notation is the one used in § 2.3: the class is denoted c and the algorithm output y. In our demonstration c takes on values in $\{0, 1\}$, and y either in $[0, 1]$ or in $\mathbf{R}^2$; but the method can be applied to more general cases, such as continuous but low-dimensional spaces for both c and y, or combinations of continuous and discrete spaces. For convenience we use a single symbol for the pair $d := (c, y)$.

For general references about the probability calculus and concepts, and specific probability distributions see Jaynes 2003; MacKay 2005; Jeffreys 1983; Gregory 2005; Bernardo & Smith 2000; Hailperin 1996; Johnson et al. 1996; 2005; 1994; 1995; Kotz et al. 2000.

B.1   Exchangeability and expression for the probability transducer

There is a fundamental theorem in the probability calculus that tells us how extrapolation from known units – molecules, patients, widgets, images – to new, unknown units takes place: de Finetti's theorem[46]. It is a consequence of the assumption that our uncertainty is invariant or 'exchangeable' under permutations of the labelling or order of the units (therefore it does not apply to time series, for example).

De Finetti's theorem states that the probability density of a value d_0 for a new unit '0', conditional on data D, is given by the following integral:

$$p(d_0 \mid D) = \int F(d_0) \, w(F \mid D) \, dF \,, \tag{20}$$

which can be given an intuitive interpretation. We consider every possible long-run frequency distribution $F(d)$ of data; give it a weight density $w(F|D)$ which depends on the observed data; and then take the weighted sum of all such long-run frequency distributions.

The weight $w(F|D)$ given to a frequency distribution F is proportional to two factors:

$$w(F \mid D) \propto F(D) \, w_g(F) \,. \tag{21}$$

- The first factor ('likelihood') $F(D)$ quantifies how well F *fits* known data of the same kind, in our case the calibration data $D := \{d_1, \dots, d_M\}$. It is simply proportional to how frequent the known data would be, according to F:

$$F(D) := F(d_1) \cdot F(d_2) \cdots F(d_M) \equiv \exp\left[M \sum_d \hat{F}(d) \ln F(d)\right] , \tag{22}$$

  where $\hat{F}(d)$ is the frequency distribution *observed* in the data.

- The second factor ('prior') $w_g(F)$ quantifies how well F *generalizes* beyond the data we have seen, owing to reasons such as physical or biological constraints for example. In our case we expect F to be somewhat smooth in X when this variable is continuous[47]. No assumptions are made about F when X is discrete.

Formula (51) is just Bayes's theorem. Its normalization factor is the integral $\int F(D) \, w_g(F) \, dF$, which ensures that $w(F)$ is normalized.

---

[46] Bernardo & Smith 2000 ch. 4; Dawid 2013; de Finetti 1929; 1937.   [47] Cf. Good & Gaskins 1971.

The exponential expression in eq. (52) is proportional to the number M of data, and in it we recognize the cross-entropy between the observed frequency distribution $\hat{F}$ and F. This has two consequences. First, it makes the final probability $p(d_0)$ increasingly identical with the distribution $\hat{F}(d)$ observed in the data, because the average (50) gets more and more concentrated around $\hat{F}$. Second, a large amount of data indicating a non-smooth distribution F will override any smoothness preferences embodied in the second factor. Note that no assumptions about the shape of F – Gaussians, logistic curves, sigmoids, or similar – are made in this approach.

The integral in (50) is calculated in either of two ways, depending on whether d is discrete or continuous. For d discrete, the integral is over $\mathbf{R}^n$, where n is the number of possible values of d, and can be done analytically. For d with continuous components, the integral is numerically approximated by a sum over T representative samples, obtained by Markov-chain Monte Carlo, of distributions F according to the weights (51):

$$\mathrm{p}(d_0 \mid D) = \int F(d_0)\, w(F \mid D)\, \mathrm{d}F \approx \frac{1}{T} \sum_{t=1}^{T} F_t(d_0)\,. \tag{23}$$

The error of this approximation can be calculated and made as small as required by increasing the number of Monte Carlo samples.

We must find a way to express any kind of distribution $F(d)$. As mentioned in § 2.3, this is done by writing it as

$$F(c, y) = \sum_l w_l\, A(c \mid \alpha_l)\, B(y \mid \beta_l)\,, \tag{24}$$

where the sum as a very large number of terms, $\{w_l\}$ are normalized weights, and $A(c \mid \alpha)$, $B(y \mid \beta)$ are distributions, possibly the product of further one-dimensional distributions. Effectively we are expressing $F(\cdot)$ by the 'coordinates' (w_l, α_l, β_l) in a space of extremely high dimensions.

This representation[48] has several advantages:

- Its marginal distributions for c and y are also of the form (24), as shown in § 2.3, and easily computable.

- Its conditional distributions for c given y and vice versa have also a form similar to eq. (24) and easily computable.

---

[48] Dunson & Bhattacharya 2011 promoted by; see also Rasmussen 1999.

- It can be used with conjugate priors.

- The final probability $p(c, y)$, approximated by the sum (23), has also the form (24):

$$p(c_0, y_0) \approx \sum_{t,l} \frac{w_{t,l}}{T} A(c_0 \mid \alpha_{t,l}) B(y_0 \mid \beta_{t,l}) . \qquad (25)$$

This is the expression given in § 2.3 with k running over both indexes t, l and with $q_k \coloneqq w_{t,l}/T$.

In our demonstration of § 4, where the class variable c takes on conventional values $\{0, 1\}$, we use a Bernoulli distribution for the first:

$$A(c \mid \alpha) = c \, \alpha + (1 - c) \, (1 - \alpha) \equiv \begin{cases} \alpha & \text{if } c = 1, \\ 1 - \alpha & \text{if } c = 0 \; ; \end{cases} \qquad (26)$$

and a Gaussian distribution for the second, $\beta \equiv (\mu, \sigma)$ being its mean and standard deviation:

$$B(y \mid \beta) = \mathrm{N}(y \mid \mu, \sigma) \coloneqq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y - \mu)^2}{2\sigma^2} \right], \qquad (27)$$

or a product of such Gaussians, each with its own parameters, if y is multidimensional. For the random-forest output $y \in [0, 1]$ such a Gaussian should in principle be truncated; we did not use any truncation as the error committed is small and the computation much faster.

B.2   Markov-chain Monte Carlo sampling

The samples $\{F_t(\,\cdot\,)\}$ of the sum (23) – these are samples of the distribution $w(F)$ – are obtained through Markov-chain Monte Carlo; specifically Gibbs sampling[49]. Effectively we obtain samples of the coordinates (w_l, α_l, β_l), and the prior $w_\mathrm{g}(F)$ is a prior over these coordinates.

For the demonstration of § 4 we use a Dirichlet distribution for (w_l), a beta distribution for (α_l), a Gaussian distribution for (μ_l), and a gamma distribution for $(1/\sigma_l^2)$.

The Markov-chain Monte Carlo sampling scheme is implemented using the R[50] package NIMBLE[51], and uses 16 parallel chains[52]. The sampling took approximately 45 min for the random forest and 75 min

---

[49] Neal 1993; MacKay 2005 ch. 29.   [50] R Core Team 2022.   [51] NIMBLE 2021.   [52] scripts available in Dyrland et al. 2022b.

for the convolutional neural network, wall-clock time. The resulting parameters are available in our supplementary data[53].

B.3 Assessment of the possible variability of the probability

In §🔧 we mentioned that the calculation of the transducer parameters automatically also tells us how much the probabilities curves could change if we used more data for the calibration. The range of this possible variability was shown for example in figs 1, 2, and 4 of the demonstration.

This possible variability is encoded in the weight $w(F \mid D)$, which can be interpreted as the probability distribution of the long-run frequency distribution F; remember that the probability $\mathrm{p}(d_0 \mid D)$, for the new unit, eq. (50), becomes closer and closer to the distribution F at which $w(F \mid D)$ peaks, as the number of data increases.

In the approximation (23), the probability $w(F \mid D)$ is effectively represented by a large number of samples $\{F_t\}$ from it. Plotting these samples alongside $\mathrm{p}(d_0 \mid D)$ gives an approximate idea of how the latter probability could change with new data. For fixed d, the samples $\{F_t(d)\}$ also give estimates of the quantiles of such change. This is how the ranges of figs 1, 2, 4 were obtained.

An analogous discussion holds for the marginal and conditional probabilities that we can obtain from $\mathrm{p}(d_0 \mid D)$.

B.4 Assessment of the augmented algorithm's long-run utility yield

Besides making a decision – such as choosing a class – for each new unit, we generally must also decide which algorithm to use for such a future task, among a set of candidates. This latter decision depends on the future performance of each algorithm, which in turn depends on the decision that the algorithm will make for each new unit. Two kinds of unknown accompany this double decision: we do not know the classes of the future units, and we do not know which outputs each algorithm will give for the future data.

---

[53] Dyrland et al. 2022b, files `transducer_params-Random_Forest.zip` and `transducer_params-Neural_Net.zip`.
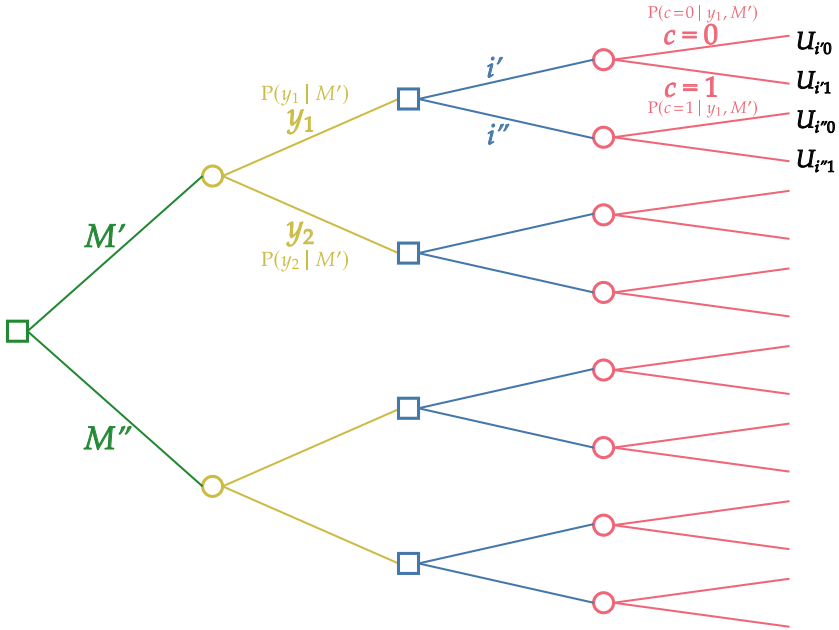
Figure 8   Decision tree for the choice of algorithm

This more complex kind of decision & uncertainty problems are also dealt with decision theory. Their theory is presented and applied step-by-step in the humorous lectures by Raiffa 1970 ch. 2; other references are[54]. We here give only a sketch and refer to the works above for details.

Our double decision & uncertainty problem can be represented as a *decision tree*. A very simplified example is illustrated in fig. 8.

We imagine to have to choose between two classification algorithms M' and M''. This choice corresponds to the *decision node* on the left (green). Decision nodes are represented by squares.

If we choose to use algorithm M', then it may happen that it will give either output y_1 or y_2 when applied to a new unit. We are uncertain about which output will occur, with probabilities $P(y_1 \mid M')$ and $P(y_2 \mid M')$. This uncertainty corresponds to an *uncertainty node* (yellow). Uncertainty nodes are represented by circles.

---

[54] Bernardo & Smith 2000 § 2.2; Pratt et al. 1996; Raiffa & Schlaifer 2000; Luce & Raiffa 1957.

Once the output of the algorithm is known, we must decide (blue decision nodes) among choices i' and i'', for example to choose whether to consider the new unit as class 0 or class 1.

The unit will turn out to be class 0 or class 1: we are uncertain about which (red decision nodes), with probabilities $P(c = 0 \mid y_1, M')$, $P(c = 1 \mid y_1, M')$ if the output was y_1, and with probabilities $P(c = 0 \mid y_2, M')$, $P(c = 1 \mid y_2, M')$ if the output was y_2.

Finally, depending on our choice between i' and i'' and on the actual class, we will gain one of the four utility amounts U_{i10}, $U_{i'1}$, $U_{i''0}$, $U_{i''1}$. These are the elements of the utility matrix discussed in § 3; we have seen concrete numerical examples in the demonstration of § 4.

An analogous analysis and probabilities hold if we choose algorithm M'' (the output space of this algorithm can be different from that of M').

The basic procedure of this decision problem is to first calculate expected utilities starting from the terminal uncertainty nodes, making optimal decisions at the immediately preceding decision nodes. Each such decision will therefore have an associated utility equal to its corresponding maximal expected utility. The same procedure is then applied to the uncertainty nodes about the outputs. In this way each algorithm receives a final expected utility; in formulae,

$$\text{utility of algorithm } M = \sum_y \left[\max_i \left\{ \sum_c U_{ic}\, P(c \mid y, M) \right\} \right] P(y \mid M) . \quad (28)$$

The sum are replaced by integrals over densities if the quantities c or y are continuous.

What is important in the formula above is that the probabilities for the outputs and the conditional probabilities for the classes given the outputs are known: *they are the ones calculated for the transducer from the calibration set*.

We have performed this calculation for the random forest and convolutional neural network (both with augmentation) of § 4, obtaining the values shown in table 4 (these utilities are not rescaled).

These values are *expected* utilities, though. One may ask: what is the probability that the final utility of one model will actually be higher or lower than the other's?

We can answer this question, again thanks to de Finetti's formula (50), similarly to how we did with the variability of the transducer curves,

explained in the previous § B.3. The long-run utility of the algorithm M is given by formula (28) but with the probabilities replaced by the long-term frequencies $F(c \mid y)$ and $F(y)$. The probability of this long-run utility is then determined by the density $w(F)$ represented by a set of samples. Calculating the long-run utility for each sample we can finally construct a probability histogram for each algorithm's utility. These are the histograms shown in §⚒

Author contributions

The authors were so immersed in the development of the present work, that unfortunately they forgot to keep a detailed record of who did what.

Thanks

Bibliography

('de *X*' is listed under D, 'van *X*' under V, and so on, regardless of national conventions.)

Axelsson, S. (2000): *The base-rate fallacy and the difficulty of intrusion detection*. ACM Trans. Inf. Syst. Secur. **3**[3], 186–205. DOI:10.1145/357830.357849, http://www.scs.carleton.ca/~soma/id-2007w/readings/axelsson-base-rate.pdf.

Baker, S. G., Pinsky, P. F. (2001): *A proposed design and analysis for comparing digital and analog mammography special receiver operating characteristic methods for cancer screening*. J. Am. Stat. Assoc. **96**[454], 421–428. DOI:10.1198/016214501753168136.

Bar-Hillel, M. (1980): *The base-rate fallacy in probability judgments*. Acta Psychol. **44**[3], 211–233. DOI:10.1016/0001-6918(80)90046-3.

Barber, D. (2020): *Bayesian Reasoning and Machine Learning*, online update. (Cambridge University Press, Cambridge). http://www.cs.ucl.ac.uk/staff/d.barber/brml. First publ. 2007.

Bento, A. P., Gaulton, A., Hersey, A., Bellis, L. J., Chambers, J., Davies, M., Krüger, F. A., Light, Y., et al. (2014): *The ChEMBL bioactivity database: an update*. Nucleic Acids Res. **42**[D1], D1083–D1090. DOI:10.1093/nar/gkt1031. Release DOI:10.6019/CHEMBL.database.20.

Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M., West, M., eds. (2011): *Bayesian Statistics 9*. (Oxford University Press, Oxford). DOI: 10.1093/acprof:oso/9780199694587.001.0001.

Bernardo, J.-M., Berger, J. O., Dawid, A. P., Smith, A. F. M., eds. (1996): *Bayesian Statistics 5*. (Oxford University Press, Oxford).

Bernardo, J.-M., Smith, A. F. (2000): *Bayesian Theory*, repr. (Wiley, New York). DOI: 10.1002/9780470316870. First publ. 1994.

Bishop, C. M. (2006): *Pattern Recognition and Machine Learning*. (Springer, New York). https://www.microsoft.com/en-us/research/people/cmbishop/prml-book.

Breiman, L. (2001): *Random forests*. Mach. Learn. **45**[1], 5–32. DOI:10.1023/A:1010933404324.

Bridle, J. S. (1990): *Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition*. Neurocomputing **68**, 227–236. DOI: 10.1007/978-3-642-76153-9_28.

Camerer, C. F., Kunreuther, H. (1989): *Decision processes for low probability events: policy implications*. J. Policy Anal. Manag. **8**[4], 565–592. DOI:10.2307/3325045.

Cheeseman, P. (1988): *An inquiry into computer understanding*. Comput. Intell. **4**[2], 58–66. DOI:10.1111/j.1467-8640.1988.tb00091.x.

— (2018): *On Bayesian model selection*. In: Wolpert (2018): 315–330. First publ. 1995.

Chen, H., la Engkvist, Wang, Y., Olivecrona, M., Blaschke, T. (2018): *The rise of deep learning in drug discovery*. Drug Discov. Today **23**[6], 1241–1250. DOI:10.1016/j.drudis.2018.01.039.

Cifarelli, D. M., Regazzini, E. (1979): *Considerazioni generali sull'impostazione bayesiana di problemi non parametrici. Le medie associative nel contesto del processo aleatorio di Dirichlet*. Riv. mat. sci. econ. soc. **2**[1, 2], 39–52, 95–111.

Damien, P., Dellaportas, P., Polson, N. G., Stephens, D. A., eds. (2013): *Bayesian Theory and Applications*. (Oxford University Press, Oxford). DOI:10.1093/acprof:oso/9780199695607.001.0001.

David, L., Thakkar, A., Mercado, R., Engkvist, O. (2020): *Molecular representations in AI-driven drug discovery: a review and practical guide*. J. Cheminf. **12**, 56. DOI:10.1186/s13321-020-00460-5.

Dawid, A. P. (2013): *Exchangeability and its ramifications*. In: Damien, Dellaportas, Polson, Stephens (2013): ch. 2:19–29. DOI:10.1093/acprof:oso/9780199695607.003.0002.

de Finetti, B. (1929): *Funzione caratteristica di un fenomeno aleatorio*. In: *Atti del Congresso Internazionale dei Matematici:* ed. by S. Pincherle (Zanichelli, Bologna): 179–190. https://www.mathunion.org/icm/proceedings, http://www.brunodefinetti.it/Opere.htm. Transl. in Cifarelli, Regazzini (1979). See also de Finetti (1930).

— (1930): *Funzione caratteristica di un fenomeno aleatorio*. Atti Accad. Lincei: Sc. Fis. Mat. Nat. **IV**[5], 86–133. http://www.brunodefinetti.it/Opere.htm. Summary in de Finetti (1929).

— (1937): *La prévision: ses lois logiques, ses sources subjectives*. Ann. Inst. Henri Poincaré **7**[1], 1–68. http://www.numdam.org/item/AIHP_1937__7_1_1_0. Transl. in Kyburg, Smokler (1980), pp. 53–118, by Henry E. Kyburg, Jr.

de Valpine, P., Paciorek, C., Turek, D., Michaud, N., Anderson-Bergman, C., Obermeyer, F., Wehrhahn Cortes, C., Rodríguez, A., et al. (2021): *NIMBLE: MCMC, particle filtering, and programmable hierarchical modeling*. https://cran.r-project.org/package=nimble, DOI:10.5281/zenodo.1211190. First publ. 2016.

Drummond, C., Holte, R. C. (2005): *Severe class imbalance: why better algorithms aren't the answer*. Eur. Conf. Mach. Learn. **2005**, 539–546. DOI:10.1007/11564096_52, https://webdocs.cs.ualberta.ca/~holte/Publications.

Dunson, D. B., Bhattacharya, A. (2011): *Nonparametric Bayes regression and classification through mixtures of product kernels*. In: Bernardo, Bayarri, Berger, Dawid, Heckerman, Smith, West (2011): 145–158. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.178.1521, DOI:10.1093/acprof:oso/9780199694587.003.0005, older version at https://www.researchgate.net/publication/228447342_Nonparametric_Bayes_Regression_and_Classification_Through_Mixtures_of_Product_Kernels.

Dunson, D. B., Pillai, N., Park, J.-H. (2007): *Bayesian density regression*. J. R. Stat. Soc. B **69**[2], 163–183.

Dyrland, K., Lundervold, A. S., Porta Mana, P. G. L. (2022a): *Does the evaluation stand up to evaluation?: A first-principle approach to the evaluation of classifiers*. Open Science Framework DOI:10.31219/osf.io/7rz8t.

— (2022b): *Bayesian augmentation of machine-learning algorithms: supplementary data*. Open Science Framework DOI:10.17605/osf.io/mfz5w.

Ferguson, T. S. (1983): *Bayesian density estimation by mixtures of normal distributions*. In: Rizvi, Rustagi, Siegmund (1983): 287–302.

Fienberg, S. E. (2007): *The Analysis of Cross-Classified Categorical Data*, 2nd ed. (Springer, New York). DOI:10.1007/978-0-387-72825-4. First publ. 1980.

Fisher, R. A. (1963): *Statistical Methods for Research Workers*, rev. 13th ed. (Hafner, New York). First publ. 1925.

Fong, E., Holmes, C. C. (2020): *On the marginal likelihood and cross-validation*. Biometrika **107**[2], 489–496. DOI:10.1093/biomet/asz077.

Gal, Y., Ghahramani, Z. (2016): *Dropout as a Bayesian approximation: representing model uncertainty in deep learning*. Proc. Mach. Learn. Res. **48**, 1050–1059. See also Appendix at arXiv DOI:10.48550/arXiv.1506.02157.

Good, I. J. (1950): *Probability and the Weighing of Evidence*. (Griffin, London).

Good, I. J., Gaskins, R. A. (1971): *Nonparametric roughness penalties for probability densities*. Biometrika **58**[2], 255–277. DOI:10.1093/biomet/58.2.255.

Goodman, L. A., Kruskal, W. H. (1954): *Measures of association for cross classifications*. J. Am. Stat. Assoc. **49**[268], 732–764. DOI:10.1080/01621459.1954.10501231. See corrections Goodman, Kruskal (1957; 1958) and also Goodman, Kruskal (1959; 1963; 1972).

— (1957): *Corrigenda: Measures of association for cross classifications*. J. Am. Stat. Assoc. **52**[280], 578. DOI:10.1080/01621459.1957.10501415. See Goodman, Kruskal (1954).

— (1958): *Corrigenda: Measures of association for cross classifications*. J. Am. Stat. Assoc. **53**[284], 1031. DOI:10.1080/01621459.1958.10501492. See Goodman, Kruskal (1954).

— (1959): *Measures of association for cross classifications. II: Further discussion and references*. J. Am. Stat. Assoc. **54**[285], 123–163. DOI:10.1080/01621459.1959.10501503. See also Goodman, Kruskal (1954; 1963; 1972).

— (1963): *Measures of association for cross classifications. III: Approximate sampling theory*. J. Am. Stat. Assoc. **58**[302], 310–364. DOI:10.1080/01621459.1963.10500850. See correction Goodman, Kruskal (1970) and also Goodman, Kruskal (1954; 1959; 1972).

— (1970): *Corrigenda: Measures of association for cross classifications. III: Approximate sampling theory*. J. Am. Stat. Assoc. **65**[330], 1011. DOI:10.1080/01621459.1970.10481142. See Goodman, Kruskal (1963).

Goodman, L. A., Kruskal, W. H. (1972): *Measures of association for cross classifications, IV: Simplification of asymptotic variances*. J. Am. Stat. Assoc. **67**[338], 415–421. DOI: 10.1080/01621459.1972.10482401. See also Goodman, Kruskal (1954; 1959; 1963).

Green, D. V. S. (2019): *Using machine learning to inform decisions in drug discovery: an industry perspective*. In: *Machine learning in chemistry: data-driven algorithms, learning systems, and predictions*, ed. by E. O. Pyzer-Knapp, T. Laino (American Chemical Society, Washington, DC): ch. 5:81–101. DOI:10.1021/bk-2019-1326.ch005.

Gregory, P. C. (2005): *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with* Mathematica *Support*. (Cambridge University Press, Cambridge). DOI: 10.1017/CBO9780511791277.

Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. A., eds. (2006): *Feature Extraction: Foundations and Applications*. (Springer, Berlin). DOI:10.1007/978-3-540-35488-8.

Hailperin, T. (1996): *Sentential Probability Logic: Origins, Development, Current Status, and Technical Applications*. (Associated University Presses, London).

— (2011): *Logic with a Probability Semantics: Including Solutions to Some Philosophical Problems*. (Lehigh University Press, Plymouth, UK).

Hand, D., Christen, P. (2018): *A note on using the F-measure for evaluating record linkage algorithms*. Stat. Comput. **28**[3], 539–547. DOI:10.1007/s11222-017-9746-6.

He, K., Zhang, X., Ren, S., Sun, J. (2016): *Deep residual learning for image recognition*. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) **2016**, 770–778. DOI:10.1109/CVPR.2016.90, https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Hingorani, A. D., Kuan, V., Finan, C., Kruger, F. A., Gaulton, A., Chopade, S., Sofat, R., MacAllister, R. J., et al. (2019): *Improving the odds of drug development success through human genomics: modelling study*. Sci. Rep. **9**, 18911. DOI:10.1038/s41598-019-54849-w.

Hjort, N. L. (1996): *Bayesian approaches to non- and semiparametric density estimation*. In: Bernardo, Berger, Dawid, Smith (1996): 223–253. With discussion by M. Lavine, M. Gasparini, and reply.

Hunink, M. G. M., Weinstein, M. C., Wittenberg, E., Drummond, M. F., Pliskin, J. S., Wong, J. B., Glasziou, P. P. (2014): *Decision Making in Health and Medicine: Integrating Evidence and Values*, 2nd ed. (Cambridge University Press, Cambridge). DOI:10.1017/CBO9781139506779. First publ. 2001.

Jaynes, E. T. (2003): *Probability Theory: The Logic of Science*. (Cambridge University Press, Cambridge). Ed. by G. Larry Bretthorst. First publ. 1994. DOI:10.1017/CBO9780511790423, https://archive.org/details/XQUHIUXHIQUHIQXUIHX2, http://www-biba.inrialpes.fr/Jaynes/prob.html.

Jeffrey, R. C. (1965): *The Logic of Decision*. (McGraw-Hill, New York).

Jeffreys, H. (1983): *Theory of Probability*, 3rd ed. with corrections. (Oxford University Press, London). First publ. 1939.

Jeni, L. A., Cohn, J. F., De La Torre, F. (2013): *Facing imbalanced data: recommendations for the use of performance metrics*. Proc. Int. Conf. Affect. Comput. Intell. Interact. **2013**, 245–251. DOI:10.1109/ACII.2013.47.

Jenny, M. A., Keller, N., Gigerenzer, G. (2018): *Assessing minimal medical statistical literacy using the Quick Risk Test: a prospective observational study in Germany*. BMJ Open **8**, e020847, e020847corr2. DOI:10.1136/bmjopen-2017-020847, DOI:10.1136/bmjopen-2017-020847corr2.

Johnson, N. L., Kemp, A. W., Kotz, S. (2005): *Univariate Discrete Distributions*, 3rd ed. (Wiley, New York). First publ. 1969.

Johnson, N. L., Kotz, S., Balakrishnan, N. (1994): *Continuous Univariate Distributions. Vol. 1*, 2nd ed. (Wiley, New York). First publ. 1970.

— (1995): *Continuous Univariate Distributions. Vol. 2*, 2nd ed. (Wiley, New York). First publ. 1970.

— (1996): *Discrete Multivariate Distributions*. (Wiley, New York). First publ. 1969 in chapter form.

Kim, H., Markus, H. R. (1999): *Deviance or uniqueness, harmony or conformity? A cultural analysis.* J. Pers. Soc. Psychol. **77**[4], 785–800. DOI:10.1037/0022-3514.77.4.785.

Kotz, S., Balakrishnan, N., Johnson, N. L. (2000): *Continuous Multivariate Distributions. Vol. 1: Models and Applications*, 2nd ed. (Wiley, New York). First publ. 1972 by N. L. Johnson and S. Kotz.

Koutsoukas, A., Monaghan, K. J., Li, X., Huan, J. (2017): *Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data*. J. Cheminf. **9**, 42. DOI:10.1186/s13321-017-0226-y.

Kruskal, W., Mosteller, F. (1979a): *Representative sampling, I: Non-scientific literature*. Int. Stat. Rev. **47**[1], 13–24. See also Kruskal, Mosteller (1979b,c; 1980).

— (1979b): *Representative sampling, II: Scientific literature, excluding statistics*. Int. Stat. Rev. **47**[2], 111–127. See also Kruskal, Mosteller (1979a,c; 1980).

— (1979c): *Representative sampling, III: The current statistical literature*. Int. Stat. Rev. **47**[3], 245–265. See also Kruskal, Mosteller (1979a,b; 1980).

— (1980): *Representative sampling, IV: The history of the concept in statistics, 1895–1939*. Int. Stat. Rev. **48**[2], 169–195. See also Kruskal, Mosteller (1979a,b,c).

Kyburg Jr., H. E., Smokler, H. E., eds. (1980): *Studies in Subjective Probability*, 2nd ed. (Robert E. Krieger, Huntington, USA). First publ. 1964.

Landrum, G., Kelley, B., Tosco, P., sriniker, NadineSchneider, Vianello, R., gedeck, adalke, et al. (2017): *RDKit: open-source cheminformatics software*. https://www.rdkit.org. Release DOI:10.5281/zenodo.268688.

Lindley, D. V., Novick, M. R. (1981): *The role of exchangeability in inference*. Ann. Stat. **9**[1], 45–58. DOI:10.1214/aos/1176345331.

Lobo, J. M., Jiménez-Valverde, A., Real, R. (2008): *AUC: a misleading measure of the performance of predictive distribution models*. Glob. Ecol. Biogeogr. **17**[2], 145–151. DOI:10.1111/j.1466-8238.2007.00358.x, https://www2.unil.ch/biomapper/Download/Lobo-GloEcoBioGeo-2007.pdf.

Luce, R. D., Raiffa, H. (1957): *Games and Decisions: introduction and critical survey*. (Wiley, New York).

Lundervold, A. S., Lundervold, A. (2019): *An overview of deep learning in medical imaging focusing on MRI*. Z. Med. Phys. **29**[2], 102–127. DOI:10.1016/j.zemedi.2018.11.002.

MacKay, D. J. C. (1992a): *The evidence framework applied to classification networks*. Neural Comput. **4**[5], 720–736. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.5.720.

— (1992b): *Bayesian interpolation*. Neural Comput. **4**[3], 415–447. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.3.415.

— (1992c): *A practical Bayesian framework for backpropagation networks*. Neural Comput. **4**[3], 448–472. http://www.inference.phy.cam.ac.uk/mackay/PhD.html, DOI:10.1162/neco.1992.4.3.448.

— (2005): *Information Theory, Inference, and Learning Algorithms*, Version 7.2 (4th pr.) (Cambridge University Press, Cambridge). https://www.inference.org.uk/itila/book.html. First publ. 1995.

Matthews, B. W. (1975): *Comparison of the predicted and observed secondary structure of T4 phage lysozyme*. Biochim. Biophys. Acta **405**^{2}, 442–451. DOI:10.1016/0005-2795(75)90109-9.

Mittone, L., Savadori, L. (2009): *The scarcity bias*. Appl. Psychol. **58**^{3}, 453–468. DOI:10.1111/j.1464-0597.2009.00401.x.

Mosteller, F., Fienberg, S. E., Rourke, R. E. K. (2013): *Beginning statistics with data analysis*, repr. (Dover, Mineola, USA). First publ. 1983.

Müller, P., Quintana, F. A. (2004): *Nonparametric Bayesian data analysis*. Stat. Sci. **19**^{1}, 95–110. http://www.mat.puc.cl/~quintana/publications/publications.html.

Murphy, K. P. (2012): *Machine Learning: A Probabilistic Perspective*. (MIT Press, Cambridge, USA). https://probml.github.io/pml-book/book0.html.

Neal, R. M. (1993): *Probabilistic inference using Markov chain Monte Carlo methods*. Tech. rep. CRG-TR-93-1. (University of Toronto, Toronto). http://www.cs.utoronto.ca/~radford/review.abstract.html, https://omega0.xyz/omega8008/neal.pdf.

Neal, R. M., Zhang, J. (2006): *High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees*. In: Guyon, Gunn, Nikravesh, Zadeh (2006): ch. 10:265–296. DOI:10.1007/978-3-540-35488-8_11.

North, D. W. (1968): *A tutorial introduction to decision theory*. IEEE Trans. Syst. Sci. Cybern. **4**^{3}, 200–210. DOI:10.1109/TSSC.1968.300114, https://stat.duke.edu/~scs/Courses/STAT102/DecisionTheoryTutorial.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., et al. (2019): *PyTorch: an imperative style, high-performance deep learning library*. Adv. Neural Inf. Process. Syst. (NIPS) **32**, 8026–8037. https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library. https://pytorch.org.

Pearl, J. (1988): *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, rev. 2nd pr. (Kaufmann, San Francisco). DOI:10.1016/C2009-0-27609-4.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., et al. (2011): *Scikit-learn: machine learning in python*. J. Mach. Learn. Res. **12**^{85}, 2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html. https://scikit-learn.org.

Porta Mana, P. G. L. (2019): *A relation between log-likelihood and cross-validation log-scores*. Open Science Framework DOI:10.31219/osf.io/k8mj3, HAL:hal-02267943, arXiv DOI:10.48550/arXiv.1908.08741.

Pratt, J. W., Raiffa, H., Schlaifer, R. (1996): *Introduction to Statistical Decision Theory*, 2nd pr. (MIT Press, Cambridge, USA). First publ. 1995.

Provost, F. (2000): *Machine learning from imbalanced data sets 101*. Tech. rep. WS-00-05-001. (AAAI, Menlo Park, USA). https://aaai.org/Library/Workshops/2000/ws00-05-001.php.

R Core Team (2022): *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org. First released 1995.

Raiffa, H. (1970): *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, 2nd pr. (Addison-Wesley, Reading, USA). First publ. 1968.

Raiffa, H., Schlaifer, R. (2000): *Applied Statistical Decision Theory*, repr. (Wiley, New York). First publ. 1961.

Rasmussen, C. E. (1999): *The infinite Gaussian mixture model*. Adv. Neural Inf. Process. Syst. (NIPS) **12**, 554–560. https://www.seas.harvard.edu/courses/cs281/papers/rasmussen-1999a.pdf.

Rizvi, M. H., Rustagi, J. S., Siegmund, D., eds. (1983): *Recent Advances in Statistics: Papers in Honor of Herman Chernoff on His Sixtieth Birthday*. (Academic Press, New York).

Rogers, D., Hahn, M. (2010): *Extended-connectivity fingerprints*. J. Chem. Inf. Model. **50**[5], 742–754. DOI:10.1021/ci100050t.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., et al. (2015): *ImageNet large scale visual recognition challenge*. Int. J. Comput. Vis. **115**[3], 211–252. DOI:10.1007/s11263-015-0816-y. https://www.image-net.org.

Russell, S. J., Norvig, P. (2022): *Artificial Intelligence: A Modern Approach*, Fourth Global ed. (Pearson, Harlow, UK). First publ. 1995.

Sammut, C., Webb, G. I., eds. (2017): *Encyclopedia of Machine Learning and Data Mining*, 2nd ed. (Springer, Boston). DOI:10.1007/978-1-4899-7687-1. First publ. 2011.

Self, M., Cheeseman, P. C. (1987): *Bayesian prediction for artificial intelligence*. In: *Proceedings of the third conference on uncertainty in artificial intelligence (uai'87)*, ed. by J. Lemmer, T. Levitt, L. Kanal (AUAI Press, Arlington, USA): 61–69. Repr. in arXiv DOI:10.48550/arXiv.1304.2717.

Sink, R., Gobec, S., Pečar, S., Zega, A. (2010): *False positives in the early stages of drug discovery*. Curr. Med. Chem. **17**[34], 4231–4255. DOI:10.2174/092986710793348545.

Sox, H. C., Higgins, M. C., Owens, D. K. (2013): *Medical Decision Making*, 2nd ed. (Wiley, New York). DOI:10.1002/9781118341544. First publ. 1988.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., Feuston, B. P. (2003): *Random forest: a classification and regression tool for compound classification and QSAR modeling*. J. Chem. Inf. Comput. Sci. **43**[6], 1947–1958. DOI:10.1021/ci034160g.

Thorburn, D. (1986): *A Bayesian approach to density estimation*. Biometrika **73**[1], 65–75.

Wald, A. (1949): *Statistical decision functions*. Ann. Math. Stat. **20**[2], 165–205. DOI:10.1214/aoms/1177730030.

Walker, S. G. (2013): *Bayesian nonparametrics*. In: Damien, Dellaportas, Polson, Stephens (2013): ch. 13:249–270.

Wolpert, D. H., ed. (2018): *The Mathematics of Generalization*, repr. (CRC Press, Boca Raton, USA). DOI:10.1201/9780429492525. First publ. 1995.

Yule, G. U. (1912): *On the methods of measuring association between two attributes*. J. R. Stat. Soc. **75**[6], 579–652. DOI:10.1111/j.2397-2335.1912.tb00463.x.

Zhu, Q. (2020): *On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset*. Pattern Recognit. Lett. **136**, 71–80. DOI:10.1016/j.patrec.2020.03.030.

C   An example

🔧 will delete this section

An example. Imagine you're a clinician and must attend to a patient with a particular disease. The disease may appear in two variants *I* and *II*; you are not sure which type affects your patient. For this disease there are three kinds of treatment *A*, *B*, *C* available at present, and you must choose one of them. Their efficacies, measured on some scale, depend on the disease type according to the following table:

	disease type	
	I	*II*
A	2	−2
B	1	1
C	−2	2

treatment

Treatment *A* is very effective for disease type *I* but causes harm (hence the negative value) if administered to a patient of type *II*; vice versa for treatment *C*. Treatment *B* never causes harm but is less effective on either disease type.

Which treatment do you choose?

You could say "the answer depends on the patient's disease type". This would be correct if you knew the disease type: type *I*, choose treatment *A*; type *II*, choose treatment *C*. But you do *not* know the disease type, so cannot make your answer depend on unavailable knowledge.

Rather, the answer depends *on how sure you are about the disease type*, given whatever evidence you have. This is clear in the case where you are completely uncertain about the type (and have no way to dispel your uncertainty), which could equally be either way. The rational, safest choice in this case is treatment *B*: the patient will have some relief for sure, albeit not the highest, and no harm will be done. If you are *extremely sure* that the disease type is *I* instead, then you recommend your patient to go for treatment *A*, as there's low risk for harm and greater benefits to be reaped (surgical treatments are typical examples of this case: there's always some minimal risk that a surgery goes awry, but such risk may be offset by the benefits of the more likely successful surgery). Analogously if you are sure about type *II* instead, recommending treatment *C*.

Now suppose that a shiny new machine-learning algorithm has been acquired by your clinic to help diagnosing the disease type quickly and with minimal expenses. This algorithm takes as input the results of various cheap clinical tests performed on a patient, and outputs the disease type. You use it for your patient, it outputs '*I*', so you go for treatment *A*, and the patient indeed gets better. You use it for two more patients. The outputs are *II* and *I*, so you administer treatments *C* and *A*. Both patients get better. Bliss! You use the algorithm for a third patient. Clinical tests are again made and their results fed into the algorithm. It outputs '*II*'. You therefore administer treatment *C*. But the patient is badly harmed and gets worse (and possibly sues you). To check for human error, the tests are repeated; the algorithm consistently outputs '*II*'. Eventually you perform an expensive and invasive but reliable test. It turns out the disease type is *I*, not *II*. The algorithm is simply wrong, and you chose the worst treatment.

You ask the algorithm's vendor or developers for an explanation: "isn't the algorithm supposed to tell me the actual disease type?". They tell you that no, there's always a chance that the algorithm is wrong, depending on the case. For some input values (test results) the prediction is virtually certain, but for others there may be a larger margin of error. You tell them:

> Sorry, but there has been a misunderstanding then. I don't need an algorithm that gives me a best guess, making me sometimes almost kill my patients. *I need an algorithm that tells me how sure a disease type is*. Because if there's much uncertainty I can give my patients a treatment that's guaranteed harm-free even if not the best!

This example exposes several misconceptions and issues in how classification problems and also some regression problems are often stated and faced in machine learning:

(i) The ultimate goal in a classification is often not the guess a class, but the choice among a set of alternative decisions. In a 'cat vs dog' image classification the classes are 'cat' and 'dog', and the decisions might be 'put into folder Cats' vs 'put into folder Dogs'. A clinician, however, does

not simply tell a patient "you probably have such-and-such disease", but has to decide among different kinds of treatment[55].

(ii) The alternative decisions, for example treatments in medicine, often do not correspond to the unknown classes in a one-one way; they may be more numerous than the classes.

(iii) The target is not what's most probable, but what's *optimal*. The two may be different. In the clinical example above, the clinician and patients would opt for treatment *B*, rather than *A*, even if type *I* had slightly higher probability than *II* – say 60% vs 40% (we shall show later that the threshold for this case is 70%). This is also true when the decisions have some natural one-one correspondence with the classes. Consider for example two other treatments *A′*, *B′* with this efficiency table:

	disease type	
	I	*II*
A′	2	−2
B′	1	1

(rows labelled "treatment")

Clearly *A′* is best for type *I*, and *B′* for type *II*, but we would choose *A′* only if type *I* had quite a higher probability than *II* (the threshold is again 70%).

C.1 Actual utility yield

The utility matrix is not only the basis for making optimal decisions by means of expected-utility maximization. It also provides the metric to rank a set of decisions already made – for example on a test set – by some algorithm, if we know the corresponding true classes. Suppose we have N test instances, in which each class c occurs N_c times, so that $\sum_c N_c = N$. A decision algorithm made decision d when the true class was c a number M_{dc} of times. These numbers form the confusion matrix (M_{dc}) of the algorithm's output. The numbers M_{dc} are must satisfy the constraints $\sum_d M_{dc} = N_c$ for each c.

For given decision d and class c, in each of the M_{dc} instances the algorithm yielded a utility U_{dc}. The actual average utility yield in the test set is then

$$\frac{1}{N} \sum_{dc} U_{dc} M_{dc} . \tag{29}$$

[55] Sox et al. 2013; Hunink et al. 2014.

It is convenient to consider the average utility yield, rather than the total utility yield (without the $1/N$ factor), because if we shift the zero or change the measurement unity of our utilities then the yield changes in the same way.

The summary of decision theory just given suffices to address issues **i1–i4**.

In comparing, evaluating, and using machine-learning classifiers we face a number of questions and issues; some are well-known, others are rarely discussed:

i1 Choice of valuation metric. When we have to evaluate and compare different classifying algorithms or different hyperparameter values for one algorithm, we are avalanched by a choice of possible evaluation metrics: accuracy, area under curve, F_1-measure, mean square contingency[56] also known as Matthews correlation coefficient[57], precision, recall, sensitivity, specificity, and many others[58]. Only vague guidelines are usually given to face this choice. Typically one computes several of such scores and hopes that they will lead to similar ranking.

i2 Rationale and consistency. Most or all of such metrics were proposed only on intuitive grounds, from the exploration of specific problems and relying on tacit assumptions, then heedlessly applied to new problems. The Matthews correlation coefficient, for example, relies on several assumptions of Gaussianity[59], which for instance do not apply to skewed population distributions[60]. The area under the receiver-operating-characteristic curve is heavily affected by values of false-positive and false-negative frequencies, as well as by misclassification costs, that have nothing to do with those of the specific application of the classifier[61]. The F_1-measure implicitly gives correct classifications a weight that depends on their frequency or probability[62]; such dependence amounts to saying, for

[56] Yule 1912 denoted 'r' there.   [57] Matthews 1975; Fisher 1963 § 31 p. 183.   [58] Sammut, Webb 2017; see also the analysis in Goodman, Kruskal 1954; 1959; 1963; 1972.   [59] Fisher 1963 § 31 p. 183 first paragraph.   [60] Jeni et al. 2013; Zhu 2020.   [61] Baker, Pinsky 2001; Lobo et al. 2008.   [62] Hand, Christen 2018.

example, "this class is rare, *therefore* its correct classification leads to high gains", which is a form of scarcity cognitive bias[63].

We are therefore led to ask: are there valuation metrics that can be proven, from first principles, to be free from biases and unnecessary assumptions?

i3 Class imbalance. If our sample data are more numerous for one class than for another – a common predicament in medical applications – we must face the 'class-imbalance problem': the classifier ends up classifying all data as belonging to the more numerous class[64], which may be an undesirable action if the misclassification of cases from the less numerous class entails high losses. 🔧 discussion and refs about cost-sensitive learning

i4 Optimality vs truth.

All the issues above are manifestly connected: they involve considerations of importance, gain, loss, and of uncertainty.

In the present work we show how issues **i1–i4** are all solved at once by using the principles of *Decision Theory*. Decision theory gives a logically and mathematically self-consistent procedure to catalogue all possible valuation metrics, to make optimal choices under uncertainty, and to evaluate and compare the performance of several decision algorithms. Most important, we show that implementing decision-theoretic procedures in a machine-learning classifier does not require any changes in current training practices 🧩 (possibly it may even make procedures like under- or over-sampling unnecessary!), is computationally inexpensive, and takes place downstream after the output of the classifier.

The use of decision theory requires sensible probabilities for the possible classes, which brings us to issue*** above. In the present work we also present and use a computationally inexpensive way of calculating these probabilities from the ordinary output of a machine-learning classifier, both for classifiers such as 🧩 example here that can only output a class label, and for classifiers that can output some sort of continuous score.

🔧 Write here a summary or outlook of the rest of the paper and a summary of results:
• The admissible valuation metrics for a binary classifier form a two-dimensional family;

---

[63] Camerer, Kunreuther 1989; Kim, Markus 1999; Mittone, Savadori 2009.   [64] Sammut, Webb 2017; Provost 2000.

that is, the choice of a specific metric corresponds to the choice of two numbers. Such choice is problem-dependent and cannot be given a priori. ● Admissible metrics are only those that can be expressed as a linear function of the elements of the population-normalized confusion matrix. Metrics such as the F_1-measure or the Matthews correlation coefficient are therefore inadmissible

D   Classification from the point of view of decision theory

In using machine-learning classifiers one typically considers situations where the set of available decisions and the set of possible classes have some kind of natural correspondence and equal in number. In a 'cat vs dog' image classification, for example, the classes are 'cat' and 'dog', and the decisions could be 'put into folder Cats' vs 'put into folder Dogs'. In a medical application the classes could be 'ill' and 'healthy' and the decisions 'treat' vs 'dismiss'. In the following when we speak of 'classification' we mean a *decision* problem of this kind. The number of decisions thus equals that of classes: $n_d = n_c$.

🦗 For simplicity we will focus on binary classification, $n_d = n_c = 2$, but the discussion generalizes to multi-class problems in an obvious way.

D.1   Choice of valuation metric, rationale and consistency (issues i1, i2)

D.2   Optimality vs truth (issue i4)

According to decision theory a classification algorithm should, at each application, calculate the probabilities $(p_{c|z})$ for the possible classes, given the feature z provided as input; calculate the expected utility of the available decisions according to eq.***, using the probabilities and the utility matrix; and finally output the decision d^* having maximal expected utility:

$$d^* = \arg\max_d \sum_c U_{dc}\, p_{c|z} \,. \tag{30}$$

We assume here that the utilities are given and the same at each application – the latter assumption could be dropped, however; see the discussion in § 6.

Current common practice with algorithms capable of outputting some sort of probability-like score is simply to output the class c^* having highest probability:

$$c^* = \arg\max_c p_{c|z} \,. \tag{31}$$

As discussed in § E, issue **i4**, this means choosing the *most probable* class, not the *optimal* class, and the two are often different, the second being what we typically want. This choice is also the one that would be made with an identity utility matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$.

How can we amend current practice for this kind of classifiers, so that they look for optimality rather than truth?

A first idea could be to simply modify the standard output step (31) into (30). It is an easily implementable and computationally cheap modification: we just multiply the probability tuple by a matrix. Such simple modification, however, has a profound implication for the training procedure: we are modifying the algorithm to output the optimal class, and therefore it should also *learn what is optimal*, not what is true: *the targets in the training and validation phases should be the optimal classes, not the true classes*. But optimality depends on the value of sensible probabilities for the specific situation of uncertainty, in this case conditional on the input features. Determining the optimal classes would thus require a probabilistic analysis that is computationally unfeasible at present for problems that involve very high-dimensional spaces, such as image classification – if an exact probabilistic analysis were possible we would not be developing machine-learning classifiers in the first place[65]. 🧩

Maybe useful to add a reminder that probability theory is the *learning* theory par excellence (even if there's no 'learning' in its name)? Its rules are all about making logical updates given new data.

Appendix: broader overview of binary classification

Let us consider our binary-classification problem from a general perspective and summarize how it would be approached and solved from first principles[66] if our computational resources had no constraints.

In our long-term task we will receive 'units' of a specific kind; the units for example could be gadgets, individuals, or investment portfolios. Each new unit will belong to one of two classes, which we can denote $X = 0$ and $X = 1$; for example they could be 'defective' vs 'non-defective', 'ill' vs 'healthy'. The class will be unknown to us. For each new unit we shall need to decide among two possible actions, which we can denote $A = \hat{0}$ and $A = \hat{1}$; for example 'discard' vs 'keep', or 'treat' vs 'dismiss'.

---

[65] Russell, Norvig 2022 chs 2, 12; Pearl 1988. [66] Russell, Norvig 2022 part IV.

The utility of each action depends on the unknown class of the unit; we denote these utilities by $U(A \mid X)$. For each new unit we will be able to measure a 'feature' Z of a specific kind common to all units; for example Z could be a set of categorical and real quantities, or an image such as a brain scan. We have a set of units – our 'sample units' or 'sample data' – that are somehow "representative" of the units we will receive in our long-term task[67]. we know both the class and the feature of each of these sample units. Let us denote this sample information by D.

According to the principles of decision theory and probability theory, for each new unit we would proceed as follows:

1. Assign probabilities to the two possible values of the unit's class, given the value of the unit's feature $Z=z$, our sample data D, and any other available information:

$$p(X=0 \mid Z=z, D), \qquad p(X=1 \mid Z=z, D) \equiv 1 - p(X=0 \mid Z=z, D),$$
$$(32)$$

according to the rules of the probability calculus.

2. Calculate the expected utilities $\bar{U}$ of the two possible actions:

$$\bar{U}(\hat{0}) := U(\hat{0} \mid X=0)\, p(X=0 \mid Z=z, D) + U(\hat{0} \mid X=1)\, p(X=1 \mid Z=z, D)$$
$$\bar{U}(\hat{1}) := U(\hat{1} \mid X=0)\, p(X=0 \mid Z=z, D) + U(\hat{1} \mid X=1)\, p(X=1 \mid Z=z, D)$$
$$(33)$$

and choose the action having maximal expected utility.

How is the probability $p(X \mid Z=z, D)$ determined by the probability calculus? Here is a simplified, intuitive picture. First consider the case where the feature Z can only take on a small number of possible values, so that many units can in principle have the same value of Z.

Consider the collection of all units having $Z = z$ that we received in the past and will receive in the future. Among them, a proportion $F(X=0 \mid Z=z)$ belong to class 0, and a proportion $1 - F(X=0 \mid Z=z) \equiv F(X=1 \mid Z=z)$ to class 1. For example these two proportions could be 74% and 26%. Our present unit with $Z=z$ is a member of this collection. The probability $p(X=0 \mid Z=z)$ that our unit belongs to class 0, given that its feature has value z, is then intuitively equal to the proportion $F(X=0 \mid Z=z)$. Analogously for $X=1$.

---

[67] for a critical analysis of the sometimes hollow term 'representative sample' see Kruskal, Mosteller 1979a,b,c; 1980.

The problem is that we do not know the proportion $F(X=0 \mid Z=z)$. However, we expect it to be roughly equal to the analogous proportion seen in our sample data; let us denote the latter by $F_s(X=0 \mid Z=z)$:

$$F(X=0 \mid Z=z) \sim F_s(X=0 \mid Z=z) . \tag{34}$$

this is indeed what we mean by saying that our sample data are 'representative' of the future units. Later we shall discuss the case in which such representativeness is of different kinds. We expect the discrepancy between $F(X=0 \mid Z=z)$ and $F_s(X=0 \mid Z=z)$ to be smaller, the larger the number of sample data. Vice versa we expect it to be larger, the smaller the number of sample data.

If Z can take on a continuum of values, as is the case for a brain scan for example, then the collection of units having $Z=z$ is more difficult to imagine. In this case each unit will be unique in its feature value – no two brains are exactly alike.

🔧

old text below

Given the unit's feature Z we will assign probabilities to the possible values of the unit's class: according to the rules of the probability calculus.

As mentioned in §***, a decision problem under uncertainty is conceptually divided into two steps

The Suppose we have a population of units or individuals characterized by a possibly multidimensional variable Z and a binary variable $X \in \{0, 1\}$. Different joint combinations of (X, Z) values can appear in this population. Denote by $F(X=x, Z=z)$, or more simply $F(x, z)$ when there is no confusion, the number of individuals having specific joint values $(X=x, Z=z)$. This is the absolute frequency of the values (x, z). We can also count the number of individuals having a specific value of $Z=z$, regardless of X; this is the marginal absolute frequency $F(z)$. It is easy to see that

$$F(z) = F(X=0, z) + F(X=1, z) \equiv \sum_x F(x, z) . \tag{35}$$

Analogously for $F(x)$.

Select only the subpopulation of individuals that have a specific value $Z=z$. In this subpopulation, the *proportion* of individuals having

a specific value $X = x$ is $f(x \mid Z = z)$. This is the conditional relative frequency of x given that z. It is easy to see that

$$f(x \mid z) = \frac{F(x, z)}{F(z)} \ . \tag{36}$$

Now suppose that we know all these statistics about this population. An individual coming from this population is presented to us. We measure its Z and obtain the value z. What could be the value of X for this individual? We know that among all individuals having $Z = z$ (and the individual before us is one of them) a proportion $f(x \mid z)$ has $X = x$. Thus we can say that there is a probability $f(x \mid z)$ that our individual has $X = x$. And this is all we can say if we only know Z.

For this individual we must choose among two actions $\{a, b\}$. The utility of performing action a if the individual has $X = x$, and given any other known circumstances, is $U(a \mid x)$; similarly for b. If we knew the value of X, say $X = 0$, we would simply choose the action leading to maximal utility:

$$\text{if} \quad U(a \mid X{=}0) > U(b \mid X{=}0) \quad \text{then choose action } a,$$
$$\text{if} \quad U(a \mid X{=}0) < U(b \mid X{=}0) \quad \text{then choose action } b, \tag{37}$$
$$\text{else} \quad \text{it does not matter which action is chosen.}$$

But we do not know the actual value of X. We have probabilities for the possible values of X given that $Z = z$ for our individual. Since X is uncertain, the final utilities of the two actions are also uncertain; but we can calculate their *expected* values $\bar{U}(a \mid Z{=}z)$ and $\bar{U}(b \mid Z{=}z)$:

$$\bar{U}(a \mid z) := U(a \mid X{=}0)\, f(X{=}0 \mid z) + U(a \mid X{=}1)\, f(X{=}1 \mid z) ,$$
$$\bar{U}(b \mid z) := U(b \mid X{=}0)\, f(X{=}0 \mid z) + U(b \mid X{=}1)\, f(X{=}1 \mid z) . \tag{38}$$

Decision theory shows that the optimal action is the one having the maximal expected utility. Our choice therefore proceeds as follows:

$$\text{if} \quad \bar{U}(a \mid z) > \bar{U}(b \mid z) \quad \text{then choose action } a,$$
$$\text{if} \quad \bar{U}(a \mid z) < \bar{U}(b \mid z) \quad \text{then choose action } b, \tag{39}$$
$$\text{else} \quad \text{it does not matter which action is chosen.}$$

The decision procedure just discussed is very simple and does not need any machine-learning algorithms. It could be implemented in a

simple algorithm that takes as input the full statistics $F(X,Z)$ of the population, the utilities, and yields an output according to (39).

Our main problem is that the full statistics $F(X,Z)$ is almost universally not known. Typically we only have the statistics $F_s(X,Z)$ of a sample of individuals that come from the population of interest or from populations that are somewhat related to the one of interest. This is where probability theory steps in. It allows us to assign probabilities to all the possible statistics $F(X,Z)$. From these probabilities we can calculate the *expected* value $\bar{f}(x \mid z)$ of the conditional frequencies $f(x \mid z)$. Decision theory says that the expected value $\bar{f}(x \mid z)$ should then be used, in this uncertain case, in eq. (38) in place of the unknown $f(x \mid z)$. The decision procedure (39) can then be used again.

Probability theory says that in this particular situation the probability of a particular possible statistics $F(X,Z)$ is the product of two factors having intuitive interpretations:

- the probability of observing the statistics $F_s(X,Z)$ of our data sample, assuming the full statistics to be $F(X,Z)$. With some combinatorics it can be shown that this probability is proportional to

$$\exp\left[\sum_{X,Z} F_s(X,Z) \ln F(X,Z)\right] \tag{40}$$

  The argument of the exponential is the cross-entropy between $F_s(X,Z)$ and $F(X,Z)$; this is the reason of its appearance in the loss function used for classifiers[68].

  This factor tells us how much the possible statistics *fit* the sample data; it gives more weight to statistics with a better fit.

- the probability of the full statistics $F(X,Z)$ for reasons not present in the data, for example because of physical laws, biological plausibility, or similar.

  This factor tells us whether the possible statistics should be favourably considered, or maybe even discarded instead, for reasons that go beyond the data we have seen; in other words, whether the hypothetical statistics would *generalize* well beyond the sample data.

---

[68] Bridle 1990; MacKay 1992a.

The final probability comes from the balance between these 'fit' and 'generalization' factors. Note that the first factor becomes more important as the sample size and therefore $F_s(X, Z)$ increases; the sample data eventually determine what the most probable statistics is, if the sample is large enough.

A similar probabilistic reasoning applies if our sample data come not from the population of interest but from a population having at least the same *conditional* frequencies of as the one of interest, either $f(X \mid Z)$ or $f(Z \mid X)$. The latter case must be examined with care when our purpose is to guess X from Z. In this case we cannot use the conditional frequencies $f_s(X \mid Z)$ that appear in the data to obtain the expected value $\bar{f}(X \mid Z)$: they could be completely different from the ones of the population of interest. We must instead use the sample conditional frequencies $f_s(Z \mid X)$ to obtain the expected value $\bar{f}(Z \mid X)$, and then combine the latter with an appropriate probability $P(X)$ through Bayes's theorem:

$$\frac{\bar{f}(Z \mid X)\, P(X)}{\sum_X \bar{f}(Z \mid X)\, P(X)} \ . \tag{41}$$

The probability $P(X)$ cannot be obtained from the data, but requires a separate study or survey. In medical applications, where X represents for example the presence or absence of a disease, the probability $P(X)$ is the base rate of the disease. Direct use of $f_s(X \mid Z)$ from the data instead of (41) is the 'base-rate fallacy'[69].

In supervised learning the classifier is trained to learn the most probable $f(X \mid Z)$ from the data. The training finds the $f(X \mid Z)$ that most closely fits the conditional frequency $f_s(X \mid Z)$ of the sampled data; this roughly corresponds to maximizing the first factor (40) described above. The architecture and the parameter regularizer of the classifier play the role of the second factor.

E   Sensible probabilities for classifiers

🔧 only pieces will be taken from this section

Why are probability values important? As we argue in a companion work 🔧✏, our ultimate purpose in classification is seldom only to guess a class; most often it is to choose a specific course of action, or to make a

---

[69] Russell, Norvig 2022 § 12.5; Axelsson 2000; Jenny et al. 2018.

decision, among several available ones. A clinician, for example, does not simply tell a patient "you will probably not contract the disease", but has to decide among dismissal or different kinds of preventive treatment[70]. Said otherwise, in classification we must choose the *optimal* class, not the probably true one. Making optimal choices in situations of uncertainty is the domain of Decision Theory[71]. In order to make an optimal choice, decision theory requires the use of probability values that properly reflect our state of uncertainty.

Determining class probabilities conditional on the input features is unfortunately computationally unfeasible at present for problems that involve very high-dimensional spaces, such as image classification; in fact if an exact probabilistic analysis were possible we would not be developing machine-learning classifiers in the first place[72]. 🔧 👣 Maybe useful to add a reminder that probability theory is the *learning* theory par excellence (even if there's no 'learning' in its name)? Its rules are all about making logical updates given new data.

In the present work we propose an alternative solution that has a low computational cost and that can be applied to all commonly used classifiers, even those that only output class labels.

The essential idea comes from seeing an analogy between a classifier and a diagnostic test, such as any common diagnostic or prognostic test used in medicine for example. There are many parallels in the way machine-learning classifiers and diagnostic tests, a flu test for instance, are devised and work. Our basic motivation in using either is that we would like to assess some situational variable – class, pathological condition – by means of its correlation (in the general sense of the word, not the linear Pearson one; and including deterministic dependence as a particular case) with a set of 'difficult' variables that are either too complex or hidden – image pixels, presence of replicating viral agents –:

$$\text{situational variable} \longleftrightarrow \text{difficult variables}$$

We devise an auxiliary variable – algorithm output, test result – to be correlated with the difficult variables:

$$\text{situational variable} \longleftrightarrow \text{difficult variables} \longleftrightarrow \text{aux variable}$$

We can now assess the situational variable by observing the more easily accessible auxiliary variable instead of the difficult ones. In probability

[70] Sox et al. 2013; Hunink et al. 2014.   [71] Russell, Norvig 2022 ch. 15; Jeffrey 1965; North 1968; Raiffa 1970.   [72] Russell, Norvig 2022 chs 2, 12; Pearl 1988.

language we are *marginalizing* over the difficult variables. This is the procedure dictated by the probability calculus whenever we do not have informational access to a set of variables. The correlation of the auxiliary variable is achieved by the training process in the case of the machine-learning algorithm, and by the exploitation of biochemical processes or reactions in the case of the flu test.

The situational variable is *informationally screened* from the auxiliary variable by the difficult variables. That is, the auxiliary variable does not – in fact, cannot – contain any more information about the situational variable than that contained in the difficult variables. This means that the probability relationship between the three variables is as follows:

$$p\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right) =$$
$$\sum_{\substack{\text{difficult}\\\text{variables}}} p\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{difficult}\\\text{variables}\end{smallmatrix}\right) \times p\left(\begin{smallmatrix}\text{difficult}\\\text{variables}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right), \quad (42)$$

the sum running over all possible values of the difficult variables.

In the case of the diagnostic test we determine the probability $p\left(\begin{smallmatrix}\text{situational}\\\text{variable}\end{smallmatrix}\,\middle|\,\begin{smallmatrix}\text{aux}\\\text{variable}\end{smallmatrix}\right)$ by carrying out the test on a representative sample of cases and collecting joint statistics between the test's output and the true situation, the presence of the flu in our example. These statistics are typically displayed in a so-called contingency table[73], akin to a confusion matrix.

Unlike the case of a diagnostic test, the output of a machine-learning classifier is usually taken at face value: if the output is a class label, that label is regarded as the true class; if the output is a unity-normalized tuple of positive numbers, that tuple is regarded as the probability distribution for the classes.

We instead propose *to treat the classifier's output just like a diagnostic test's result*. This output, discrete or continuous, is regarded as a quantity that has some correlation with the true class. This correlation can be analysed in a set of representative samples and used to calculate a sensible probability for the class given the classifier's output. This analysis only needs to be made once and is computationally cheap, because the classifier output takes values in a discrete or low-dimensional space.

---

[73] Fienberg 2007; Mosteller et al. 2013.

This approach differs from the computationally infeasible one discussed above in that we are calculating the computationally easier probability

$$p(\text{class} \mid \text{output}) \tag{43}$$

rather than

$$p(\text{class} \mid \text{feature}) \ . \tag{44}$$

The former probability, as we saw in eq. (42), is the marginal

$$p(\text{class} \mid \text{output}) = \sum_{\text{feature}} p(\text{class} \mid \text{feature}) \times p(\text{feature} \mid \text{output}) \ . \tag{45}$$

We can thus think of this approach as a marginalization over the possible features, which is a necessary operation as have no effective access to them.

A hallmark of this approach is that we are calculating exact probabilities conditional on reduced information, rather than approximate probabilities conditional on full information. This protects us from biases that are typically present in the approximation method. The price of using reduced information is that the probabilities may be open to more variability as we collect more representative data. But as we shall see this variability is actually quite low, and moreover it can be exactly assessed.

This approach also offers the following advantages:

- It does not require any changes of the standard training procedures.

- It is easily implemented as an additional low-cost computation of a function at the end of the classifier's output, or as a replacement of a softmax-like computation.

- It does not make any assumptions such as linearity or Gaussianity.

- It yields not only the probability distribution for the classes, but also a measure of how much this distribution could change if we collected more test data (the 'probability of the probability', so to speak).

- It allows us to use the classifier both in a discriminative and generative way. That is, we can use either $p(\text{class} \mid \text{output})$, or $p(\text{output} \mid \text{class})$ in conjunction with Bayes's theorem. The latter approach enables us to avoid possible base-rate fallacies[74].

---

[74] Russell, Norvig 2022 § 12.5; Axelsson 2000; Jenny et al. 2018.

- It can be seamlessly integrated with a utility matrix to compute the optimal class, as shown in the companion work 🔧🖊.

In § F we present some notation and the general procedure for the calculation of the probabilities; more technical details are given in appendix***. Section*** explains how to augment a classifier's output with the probability calculation. Results of numerical experiments are presented in §***.

🔧🔧 We could show that even if we used a biased test set, the method corrects the bias (provided we know what the bias is).

F   Calculation of the probabilities: general procedure

Let us denote the class variable by C and the classifier-output variable by X. We assume that C is discrete and finite, its values can be simply renamed $1, 2, 3, \dots$. We also assume that X is either discrete and finite (it is isomorphic to C for many classifying algorithms) or a low-dimensional tuple of real variables; a combination of both cases can also be easily accommodated. We imagine to have a sample of M such data pairs denoted collectively by D:

$$D := \big\{ (c_1, x_1), (c_2, x_2), \dots, (c_M, x_M) \big\} . \tag{46}$$

Let us call them *calibration data*. It must be emphasized that these are not pairs of class & feature values, but pairs of class & classifier-output values, obtained as described in § G.

Instead of the conditional probability p(class | output), that is, $\mathrm{p}(C \mid X)$, we can actually calculate the joint probability

$$\mathrm{p}(C, X) \tag{47}$$

given the sample data. The computational cost is the same, but from the joint probability we can easily derive both conditionals

$$\mathrm{p}(C \mid X) = \frac{\mathrm{p}(C, X)}{\sum_C \mathrm{p}(C, X)} , \tag{48}$$

$$\mathrm{p}(X \mid C) = \frac{\mathrm{p}(C, X)}{\sum_X \mathrm{p}(C, X)} . \tag{49}$$

It is advantageous to have both, as we shall see in §🔧🖊: if one of them is biased owing to the way the test samples were obtained, we can still use the other.

In our specific inference problem, where no time trends are assumed to exist in future data (the probability distribution for future data is exchangeable), probability theory dictates that the joint probability (47) for a new datapoint (c_0, x_0) is equal to the *expected* frequency of that datapoint in a hypothetically infinite run of observations, that is, the average

$$p(c_0, x_0) = \int F(c_0, x_0)\, w(F)\, dF \, . \tag{50}$$

This formula is the so-called de Finetti's theorem[75]. It is derived from first principles but can be intuitively interpreted: We are considering every possible long-run frequency distribution $F(\cdot, \cdot)$, giving it a weight $w(F)$, and then taking the weighted sum of all such distributions. The result is still a distribution, and its value at (c_0, x_0) is the probability of this datapoint.

The weight $w(F)$ – a probability density – given to a frequency distribution F is proportional to two factors:

$$w(F) \propto F(D)\, w_g(F) \, . \tag{51}$$

- The first factor ('likelihood') $F(D)$ quantifies how well F *fits* known data of the same kind, the sample data D in our case. It is simply proportional to how frequent the known data would be, according to F:

$$F(D) = F(c_1, x_1)\, F(c_2, x_2)\, \cdots\, F(c_M, x_M) \, . \tag{52}$$

- The second factor ('prior') $w_g(F)$ quantifies how well F *generalizes* beyond the data we have seen, owing to reasons such as physical or biological constraints for example. In our case we expect F to be somewhat smooth in X when this variable is continuous[76] 🔧🔧 add a picture – a sample from the prior over F – to illustrate the expected range of smoothness. No assumptions are made about F when X is discrete.

Formula (51) is just Bayes's theorem. Its normalization factor is the integral $\int F(D)\, w_g(F)\, dF$, which ensures that $w(F)$ is normalized.

The first factor becomes larger as the number of known data increases. Thus a large amount of data indicating a non-smooth distribution F will override any smoothness preferences embodied in the second factor.

---

[75] Bernardo, Smith 2000 ch. 4; Dawid 2013; de Finetti 1929; 1937.   [76] Cf. Good, Gaskins 1971.

Note that no assumptions about the shape of F – no Gaussians, logistic curves, sigmoids, and so on – are made in this approach.

The integral in (50) is calculated in either of two ways, depending on whether X is discrete or continuous. For X discrete and taking on the same values as the class variable C, the integral is over $\mathbf{R}^{n_c^2}$ where n_c is the number of possible classes, and can be done analytically. For X continuous, the integral is numerically approximated by a sum over a representative sample, obtained by Markov-chain Monte Carlo, of distributions F according to the weights (51). The error of this approximation can be calculated and made as small as required by increasing the number of Monte Carlo samples.

The expected value (50) is calculated for all possible values of (c_0, x_0), obtaining the full joint probability distribution $p(C, X)$. From this joint distribution we calculate the direct and inverse conditional distributions

$$p(C \mid X) = \frac{p(C, X)}{\sum_C p(C, X)} \, , \tag{53}$$

$$p(X \mid C) = \frac{p(C, X)}{\sum_X p(C, X)} \, . \tag{54}$$

It is very convenient to have both, as discussed in § **??**.

The conditional distributions above are just matrices when X is discrete. For continuous X they can be regarded as n_c tuples of functions in X. We can find convenient approximate expressions, such as polynomial interpolants, for faster numerical implementations of these functions.

The integration procedure for (50) also tells us how much the probability distribution $p(C, X)$ would change if we acquired new data (a sort of 'probability of the probability').

For further mathematical details see appendix***

G Implementation in the classifier output

The implementation of our approach takes place after the training of the classifier has been carried out in the usual way. We assume that a collection of M *test data* were set aside as usual:

$$T := \big\{ (c_1, z_1), (c_2, z_2), \dots, (c_M, z_M) \big\} \, , \tag{55}$$

where the c_i are the true classes and z_i the corresponding feature values.

The M feature values z_i are given as inputs to the classifier, which produces M corresponding outputs x_i. We now consider data pairs consisting in the true classes c_i and the outputs x_i: these are the *calibration data* discussed in § F:

$$D := \{(c_1, x_1), (c_2, x_2), \ldots, (c_M, x_M)\} .$$

They are used to find the direct and inverse conditional probability distributions $p(C \mid X)$ and $p(X \mid C)$ as described in § F.

We can finally augment our classifier either in a 'direct' or 'discriminative' way, or an 'inverse' or 'generative' way, by adding one computation step at the end of the classifier's operation:

Direct: from its output x_0 we obtain the probability for each class, $p(c \mid x_0)$.

Inverse: from its output x_0 we obtain the probability of the output itself, conditional on each class, $p(x_0 \mid c)$.

These n_c probabilities are the final output of the augmented classifier.

In the direct or discriminative case, at each new use of the classifier the output probabilities can be used together with a utility matrix to choose the *optimal* class for that case, as discussed in the companion paper✦✦.

In the inverse or generative case, at each new use of the classifier the probabilities for the classes are obtained via Bayes's theorem:

$$p(c) = \frac{p(x_0 \mid c)\, B(c)}{\sum_c p(x_0 \mid c)\, B(c)} , \tag{56}$$

where $B(c)$ is the base rate of class c. The probabilities $p(c)$ can finally be used together with a utility matrix to choose the *optimal* class.