Open Science Framework DOI:10.31219/osf.io/***

Probabilities for machine-learning classifiers

Classifiers as diagnostic tests

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, Bergen, Norway

Draft. 14 April 2022; updated 22 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 What is the goal of a classification?

As the potential of using machine-learning algorithms in important fields such as medicine increases, the machine-learning community ought to keep in mind what the actual needs and goals in such fields are. We must avoid trying (consciously or unconsciously) to convince such fields to change their needs just to fit machine-learning solutions that are available and fashionable at the moment. Rather, we must make sure the available solutions fit those needs, and amend them if they do not.

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*
treatment	*A*	3	−3
	B	1	1
	C	−3	3

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 much 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 are often stated and faced in machine learning:

(i)

2 Sensible probabilities for classifiers

Some machine-learning algorithms for classification, such as support-vector machines, typically output a class label. Others, such as deep networks, output a set of real numbers. These real numbers can be positive, normalized to unity, and can bear some qualitative relation to the plausibilities of the classes. But they cannot be reliably interpreted as sensible probabilities, that is, as the degrees of belief assigned to each possible class by a rational agent[1]; or, in terms of 'populations'[2], 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[3]).

Algorithms that internally do perform probabilistic calculations, such as naive-Bayes or logistic-regression classifiers[4], unfortunately rest on 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[5] output sensible probabilities.

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

[1] **mackay1992d**; **galetal2016russelletal1995_r2022**. [2] **lindleyetal1981fisher1956_r1967**. [3] **bernardoetal1994_r2000dawid2013**. [4] **murphy2012bishop2006barber2007_r2020**. [5] **nealetal2006bishop2006**.

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[6]. 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[7]. 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[8]. 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 –:

situational variable ⟷ difficult variables

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

situational variable ⟷ difficult variables ⟷ aux variable

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

[6] **soxetal1988_r2013**; **huninketal2001_r2014**. [7] **russelletal1995_r2022jeffrey1965**; **north1968**; **raiffa1968_r1970**. [8] **russelletal1995_r2022pearl1988**.

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:

$$\mathrm{p}\left(\begin{smallmatrix}\text{situational}\\ \text{variable}\end{smallmatrix} \,\middle|\, \begin{smallmatrix}\text{aux}\\ \text{variable}\end{smallmatrix}\right) = \sum_{\substack{\text{difficult}\\ \text{variables}}} \mathrm{p}\left(\begin{smallmatrix}\text{situational}\\ \text{variable}\end{smallmatrix} \,\middle|\, \begin{smallmatrix}\text{difficult}\\ \text{variables}\end{smallmatrix}\right) \times \mathrm{p}\left(\begin{smallmatrix}\text{difficult}\\ \text{variables}\end{smallmatrix} \,\middle|\, \begin{smallmatrix}\text{aux}\\ \text{variable}\end{smallmatrix}\right) , \qquad (1)$$

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

In the case of the diagnostic test we determine the probability $\mathrm{p}\left(\begin{smallmatrix}\text{situational}\\ \text{variable}\end{smallmatrix} \,\middle|\, \begin{smallmatrix}\text{aux}\\ \text{variable}\end{smallmatrix}\right)$ 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[9], 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.

[9] **fienberg1980_r2007**; **mostelleretal1983_r2013**.

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

$$\mathrm{p}(\text{class} \mid \text{output}) \tag{2}$$

rather than

$$\mathrm{p}(\text{class} \mid \text{feature})\ . \tag{3}$$

The former probability, as we saw in eq. (1), is the marginal

$$\mathrm{p}(\text{class} \mid \text{output}) = \sum_{\text{feature}} \mathrm{p}(\text{class} \mid \text{feature}) \times \mathrm{p}(\text{feature} \mid \text{output})\ . \tag{4}$$

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 $\mathrm{p}(\text{class} \mid \text{output})$, or $\mathrm{p}(\text{output} \mid \text{class})$ in conjunction with Bayes's theorem. The latter approach enables us to avoid possible base-rate fallacies[10].

[10] **russelletal1995_r2022axelsson2000**; **jennyetal2018**.

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

In § 3 we present some notation and the general procedure for the calculation of the probabilities; more technical details are given in appendix 9. Section 4 explains how to augment a classifier's output with the probability calculation. Results of numerical experiments are presented in § 5.

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

3 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 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 assume to have a sample of M such data pairs denoted collectively by D:

$$D \coloneqq \bigl\{(c_1, x_1), (c_2, x_2), \dots, (c_M, x_M)\bigr\} \,. \tag{5}$$

We call them *calibration data*. Let us emphasize that these are not pairs of class & feature values, but pairs of class & classifier-output values, obtained as described in § 4.

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

$$\mathrm{p}(C, X) \tag{6}$$

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

$$\mathrm{p}(C \mid X) = \frac{\mathrm{p}(C, X)}{\sum_C \mathrm{p}(C, X)} \,, \tag{7}$$

$$\mathrm{p}(X \mid C) = \frac{\mathrm{p}(C, X)}{\sum_X \mathrm{p}(C, X)} \,. \tag{8}$$

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 (6) 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

$$\mathrm{p}(c_0, x_0) = \int F(c_0, x_0)\, w(F)\, \mathrm{d}F\ . \tag{9}$$

This formula is the so-called de Finetti's theorem[11]. 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:

$$w(F) \propto F(D)\, w_{\mathrm{g}}(F)\ . \tag{10}$$

- 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:

  $$F(D) = F(c_1, x_1)\, F(c_2, x_2)\ \cdots\ F(c_M, x_M)\ . \tag{11}$$

- The second factor ('prior') $w_{\mathrm{g}}(F)$ 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[12] 🔧🔧 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 (10) is just Bayes's theorem. Its normalization factor is the integral $\int F(D)\, w_{\mathrm{g}}(F)\, \mathrm{d}F$, 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.

[11] **bernardoetal1994_r2000dawid2013**; **definetti1929**; **definetti1937**.

[12] **goodetal1971**.

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 (9) 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 $\mathbf{R}^{n_\mathrm{c}^2}$ 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 (10). 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 (9) is calculated for all possible values of (c_0, x_0), obtaining the full joint probability distribution $\mathrm{p}(C, X)$. From this joint distribution we calculate the direct and inverse conditional distributions

$$\mathrm{p}(C \mid X) = \frac{\mathrm{p}(C, X)}{\sum_C \mathrm{p}(C, X)} , \tag{12}$$

$$\mathrm{p}(X \mid C) = \frac{\mathrm{p}(C, X)}{\sum_X \mathrm{p}(C, X)} . \tag{13}$$

It is very convenient to have both, as discussed in § 6.

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 (9) also tells us how much the probability distribution $\mathrm{p}(C, X)$ would change if we acquired new data (a sort of 'probability of the probability').

For further mathematical details see appendix 9.

4 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:

$$T \coloneqq \bigl\{(c_1, z_1), (c_2, z_2), \dots, (c_M, z_M)\bigr\} , \tag{14}$$

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 § 3:

$$D \coloneqq \bigl\{(c_1, x_1), (c_2, x_2), \dots, (c_M, x_M)\bigr\} .$$

They are used to find the direct and inverse conditional probability distributions $\mathrm{p}(C \mid X)$ and $\mathrm{p}(X \mid C)$ as described in § 3.

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, $\mathrm{p}(c \mid x_0)$.

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

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:

$$\mathrm{p}(c) = \frac{\mathrm{p}(x_0 \mid c)\; B(c)}{\sum_c \mathrm{p}(x_0 \mid c)\; B(c)} , \tag{15}$$

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

5 Numerical experiments and results

6 Circumventing biases

7 Alternative to ensembling

🔧🔧 The idea is to implement this output-to-probability conversion in several classifiers, *in a generative way*. These probabilities can then be multiplied together and with a base rate, to obtain the 'ensembled' probabilities of the classes. This way of doing ensembling would be a rigorous application of the probability calculus; should be superior to a majority vote or similar.

8 Methods and materials

8.1 Data

The data comes from a publicly open bioactivity database called CheMBL **CheMBL**. The dataset that is used in this paper was introduced in the paper by Koutsoukas et. al. **DL-investigating-bioactivity**. The dataset is a source of structure-activity relationship from version 20 of ChEMBL with protein target Carbonic Anhydrase II (ChEMBL205).

8.2 Pre-processing

For our pre-processing pipeline, we use two different methods to represent the molecule, one for the RF and one for the DNN. The first method turns the molecule into a hashed bit vector of circular fingerprints called ECFP (Extended Connectivity Fingerprints) **Rogers-ECFP**. From our experiments, there was little to no improvement using a 2048-bit vector over a 1024-bit vector.

For our DNN model, the data is represented by converting the molecule into images in 224x224 size. This is done by using the SMILES string from the dataset and using RdKit **rdkit** to generate a canonical graph structure of the molecule. This differs from ECFP in that it represents the actual structure of the molecule rather than properties generated from the molecule.

The dataset has in total 1631 active compounds and 16310 non-active molecules which act as decoys. In training, the active molecules are oversampled to match the same number of non-active molecules.

8.3 Prediction

Virtual screening is the process of finding chemical activity in the interaction between a compound (molecule) and a target (protein). The goal of the machine learning algorithms is to find structural features or chemical properties that show that the molecule is active against the protein. DNNs have previously been shown to outperform RF and other linear models in VHTS and quantitative structure-activity relationship (QSAR) problems **DL-investigating-bioactivity**.

8.4 Models

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

The second model is a pre-trained residual network (ResNet) **resnet** with 18 hidden layers trained on the well-known ImageNet dataset **ImageNet** by using the PyTorch framework **PyTorch**. ResNet has shown to outperform other pre-trained CNN models **resnet**. A ResNet with 34 hidden layers showed little to no performance gain, so we chose to go with the simpler model. The model is trained is with the following hyperparameters:

- **Learning rate:** 0.003
- **Optimization technique:** Stochastic Gradient Descent (SGD)
- **Activation Function:** ReLU
- **Dropout:** 50%
- **Number of epochs:** 20
- **Loss function:** Cross Entropy Loss

8.5 Train, Validation, Test split

The data is split into four parts:

- Test set 1: Uses 20% of the total dataset for the Bayesian inference.
- Test set 2: Uses 20% of the dataset for evaluation.
- Validation set: 25% of the remaining data for validating the model after each epoch.
- Training set: The rest of the dataset is used to train the model.

9 Mathematical details of the nonparametric density regression

The joint probability (6) is calculated nonparametrically, that is, without making any assumptions such as linearity or gaussianity, besides very mild and reasonable assumptions of continuity.

using the versatile computational approach by **dunsonetal2011**, and obtain the probability (2) by conditionalization. The calculation requires Monte Carlo sampling 🔧✏ refs here but needs to be made only once.

10 Summary and discussion

🔧 Add note about how (sequential) decision theory was used during World War I; see **good1950** around § 6.2

PIECES OF TEXT

10.1 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

$$\frac{1}{N} \sum_{dc} U_{dc}\, M_{dc}\ . \tag{16}$$

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[13] also known as Matthews correlation coefficient[14], precision, recall, sensitivity, specificity, and many others[15]. Only

[13] **yule1912**. [14] **matthews1975fisher1925_r1963**. [15] **sammutetal2011_r2017goodmanetal1954**; **goodmanetal1959**; **goodmanetal1963**; **goodmanetal1972b**.

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[16], which for instance do not apply to skewed population distributions[17]. 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[18]. The F_1-measure implicitly gives correct classifications a weight that depends on their frequency or probability[19]; such dependence amounts to saying, for example, "this class is rare, *therefore* its correct classification leads to high gains", which is a form of scarcity cognitive bias[20].

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[21], 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

[16] **fisher1925_r1963**.

[17] **jenietal2013**; **zhu2020**.

[18] **bakeretal2001**; **loboetal2008**.

[19] **handetal2018**.

[20] **camereretal1989**; **kimetal1999**; **mittoneetal2009**.

[21] **sammutetal2011_r2017**; **provost2000**.

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; 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

11 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: $n_{\mathrm{d}} = n_{\mathrm{c}}$.

For simplicity we will focus on binary classification, $n_{\mathrm{d}} = n_{\mathrm{c}} = 2$, but the discussion generalizes to multi-class problems in an obvious way.

11.1 Choice of valuation metric, rationale and consistency (issues i1, i2)

11.2 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:

$$d^* = \arg\max_d \sum_c U_{dc}\, p_{c|z} \ . \tag{17}$$

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 § 10.

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

$$c^* = \arg\max_c p_{c|z} \ . \tag{18}$$

As discussed in § 2, 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 (18) into (17). 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[22]. 🧩 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[23] 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'. 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[24]. 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:

   $$\mathrm{p}(X=0 \mid Z=z, D), \qquad \mathrm{p}(X=1 \mid Z=z, D) \equiv 1 - \mathrm{p}(X=0 \mid Z=z, D)\;, \tag{19}$$

   according to the rules of the probability calculus.

[22] **russelletal1995_r2022pearl1988**. [23] **russelletal1995_r2022**. [24] **kruskaletal1979**; **kruskaletal1979b**; **kruskaletal1979c**; **kruskaletal1980**.

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

$$\begin{aligned}\bar{\mathrm{u}}(\hat{0}) &:= U(\hat{0} \mid X{=}0)\,\mathrm{p}(X{=}0 \mid Z{=}z, D) + U(\hat{0} \mid X{=}1)\,\mathrm{p}(X{=}1 \mid Z{=}z, D)\\ \bar{\mathrm{u}}(\hat{1}) &:= U(\hat{1} \mid X{=}0)\,\mathrm{p}(X{=}0 \mid Z{=}z, D) + U(\hat{1} \mid X{=}1)\,\mathrm{p}(X{=}1 \mid Z{=}z, D)\end{aligned} \tag{20}$$

   and choose the action having maximal expected utility.

How is the probability $\mathrm{p}(X \mid Z{=}z, D)$ determined by the probability calculus? Here is a simplified, intuitive picture. First consider the case where the feature Z can only assume 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 $\mathrm{p}(X = 0 \mid Z = z)$ 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 $X{=}1$.

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 $F_{\mathrm{s}}(X{=}0 \mid Z{=}z)$:

$$F(X{=}0 \mid Z{=}z) \sim F_{\mathrm{s}}(X{=}0 \mid Z{=}z)\,. \tag{21}$$

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 $F_{\mathrm{s}}(X{=}0 \mid Z{=}z)$ 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 assume 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

$$F(z) = F(X\!=\!0, z) + F(X\!=\!1, z) \equiv \sum_x F(x,z)\,. \tag{22}$$

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

$$f(x \mid z) = \frac{F(x,z)}{F(z)}\,. \tag{23}$$

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:

$$\begin{aligned}
&\text{if}\quad U(a \mid X\!=\!0) > U(b \mid X\!=\!0)\quad \text{then choose action } a,\\
&\text{if}\quad U(a \mid X\!=\!0) < U(b \mid X\!=\!0)\quad \text{then choose action } b,\\
&\text{else}\quad \text{it does not matter which action is chosen.}
\end{aligned} \tag{24}$$

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 $\bar{U}(b \mid Z\!=\!z)$:

$$\begin{aligned}
\bar{U}(a \mid z) &\coloneqq U(a \mid X\!=\!0)\, f(X\!=\!0 \mid z) + U(a \mid X\!=\!1)\, f(X\!=\!1 \mid z)\,,\\
\bar{U}(b \mid z) &\coloneqq U(b \mid X\!=\!0)\, f(X\!=\!0 \mid z) + U(b \mid X\!=\!1)\, f(X\!=\!1 \mid z)\,.
\end{aligned} \tag{25}$$

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

$$\begin{aligned}
&\text{if}\quad \bar{U}(a \mid z) > \bar{U}(b \mid z) \quad \text{then choose action } a,\\
&\text{if}\quad \bar{U}(a \mid z) < \bar{U}(b \mid z) \quad \text{then choose action } b,\\
&\text{else}\quad \text{it does not matter which action is chosen.}
\end{aligned} \tag{26}$$

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 (26).

Our main problem is that the full statistics $F(X,Z)$ is almost universally not known. Typically we only have the statistics $F_{\mathrm{s}}(X,Z)$ 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. (25) in place of the unknown $f(x \mid z)$. The decision procedure (26) 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 $F_{\mathrm{s}}(X,Z)$ 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

$$\exp\Bigl[\sum_{X,Z} F_{\mathrm{s}}(X,Z) \ln F(X,Z)\Bigr] \tag{27}$$

The argument of the exponential is the cross-entropy between $F_{\mathrm{s}}(X, Z)$ and $F(X, Z)$; this is the reason of its appearance in the loss function used for classifiers[25].

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.

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 $F_{\mathrm{s}}(X, Z)$ 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 $f_{\mathrm{s}}(X \mid Z)$ 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 $f_{\mathrm{s}}(Z \mid X)$ 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:

$$\frac{\bar{f}(Z \mid X)\, P(X)}{\sum_X \bar{f}(Z \mid X)\, P(X)} \,. \tag{28}$$

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 $f_{\mathrm{s}}(X \mid Z)$ from the data instead of (28) is the 'base-rate fallacy'[26].

[25] **bridle1990**; **mackay1992d**. [26] **russelletal1995_r2022axelsson2000**; **jennyetal2018**.

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 $f_{\mathrm{s}}(X \mid Z)$ of the sampled data; this roughly corresponds to maximizing the first factor (27) described above. The architecture and the parameter regularizer of the classifier play the role of the second factor.