

# Genomic Applications of the Neyman–Pearson Classification Paradigm

Jingyi Jessica Li and Xin Tong

**Abstract** The Neyman–Pearson (NP) classification paradigm addresses an important binary classification problem where users want to minimize type II error while controlling type I error under some specified level  $\alpha$ , usually a small number. This problem is often faced in many genomic applications involving binary classification tasks. The terminology Neyman–Pearson classification paradigm arises from its connection to the Neyman–Pearson paradigm in hypothesis testing. The NP paradigm is applicable when one type of error (e.g., type I error) is far more important than the other type (e.g., type II error), and users have a specific target bound for the former. In this chapter, we review the NP classification literature, with a focus on the genomic applications as well as our contribution to the NP classification theory and algorithms. We also provide simulation examples and a genomic case study to demonstrate how to use the NP classification algorithm in practice.

**Keywords** Classification • Genomic applications • Neyman–Pearson • Statistical learning • Methodology

## 1 Introduction

As an important statistical and machine learning method, classification has been widely used in genomic studies. Binary classification is the basis of all types of classification problems, and there exist many approaches to ensemble binary classifiers to solve multi-class classification problems or to reduce multi-class to binary classification. Important genomic applications of binary classification include labeling microarray data as tumor or non-tumor samples [18, 53], dividing

---

J.J. Li (✉)

Department of Statistics, University of California, Los Angeles, Los Angeles, CA, USA  
e-mail: [jli@stat.ucla.edu](mailto:jli@stat.ucla.edu)

X. Tong

Department of Data Sciences and Operations, University of Southern California,  
Los Angeles, CA, USA  
e-mail: [xint@marshall.usc.edu](mailto:xint@marshall.usc.edu)

genes into housekeeping and single-tissue specific groups [54], classifying genomic hairpin structures into precursor microRNAs and pseudo hairpins (i.e., genomic inverted repeats that are not precursor microRNAs) [32], and predicting transcription factor binding sites and other DNA regulatory elements based on genomic features [7, 20].

The aim of binary classification is to accurately predict binary (i.e., 0 or 1) labels for new observations on the basis of labeled training data. There are two types of errors: *type I error* (the conditional probability that the predicted label for a new observation is 1 given that the observation has a true label 0) and *type II error* (the conditional probability that the predicted label for a new observation is 0 given that the observation has a true label 1). For more than half a century, significant advances have been made in the development of binary classification theory and methods to construct good classifiers with various desirable properties [24]. Most existing binary classification methods aim to optimize the *risk*, which is the expected classification error (the probability that the predicted label is different from the true label) and can be expressed as a weighted sum of the type I and II errors, where the two weights are the marginal probabilities of the true label being 0 and 1, respectively. In real-world applications, however, users' priorities for type I and type II errors may differ from these weights, and then minimizing the risk may lead to unsatisfactory classifiers. For example, in tumor diagnosis, suppose that we label a tumor sample as 0 and a normal sample as 1, the risk minimization approach fails if it leads to a classifier with type I error (i.e., the conditional probability that a tumor sample is misclassified as a normal sample) equal to 0.3 but doctors prefer to constrain the type I error under 0.05.

There are many scenarios where users need asymmetric error control, and they often occur when the two types of classification errors lead to vastly different consequences. Again in the example of tumor diagnosis, mispredicting a normal sample as a tumor sample may increase a patient's anxiety and impose additional medical costs, but misclassifying a tumor sample as a normal sample may delay a patient's treatment and even cause a life loss. Hence, the latter type of error—type I error—is more severe and should be controlled at a low level. In another example of classifying genes into housekeeping ones (say class 0) and cell-specific ones (say class 1), if the research aim is to identify novel cell-specific genes for a cell type (e.g., human embryonic stem cells) and the identified genes will be validated by experiments, researchers would generally prefer to control the type I error (the conditional probability of misclassifying a housekeeping gene as a cell-specific gene) at a low level to reduce experimental costs.

One common approach to addressing asymmetric classification errors is cost-sensitive learning, which allows users to assign two different costs as weights for type I and type II errors [14, 56]. Although this approach has many merits, its effectiveness is largely limited when there lacks a consensus way to assign costs. Cost-sensitive learning is also unable to serve the purpose when users desire a specific high probabilistic bound  $\alpha$  on the type I or II error [e.g.,  $\mathbb{P}(\text{type I error} \leq \alpha) > 1 - \delta$ , the probability that a chosen classifier has type I error not exceeding  $\alpha$  is greater than  $1 - \delta$  for some small positive  $\alpha$  (e.g., 0.05) and  $\delta$  (e.g., 0.05)], even though users may vary the two costs to achieve a

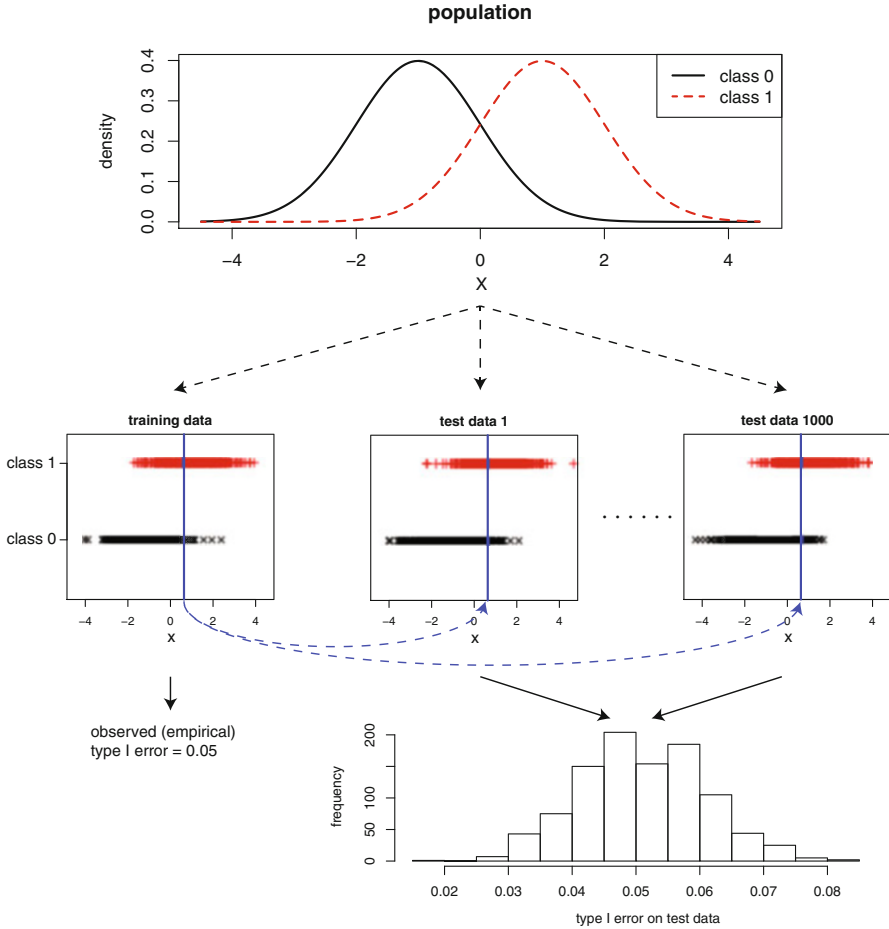
small type I or type II error. There are several other classification approaches that target on small type I errors. Examples include asymmetric support vector machines (SVM) [52] and  $p$ -values for classification [13]. But like cost-sensitive learning, these approaches also provide no probabilistic guarantee for the type I error bound and could lead to non-negligible probability of large type I errors. In practice, there has been a long-time intuitive and straightforward common practice, that is to tune the observed type I error (also called empirical type I error) on the training data to the desired type I error bound  $\alpha$ , for example, by adjusting the costs of errors or by changing the classification threshold. However, this approach cannot control the type I error of the chosen classifier on a new data set to be under  $\alpha$  with high probability; in fact, a classifier chosen in this way will have type I error greater than  $\alpha$  for approximately half the chance. Figure 1 illustrates this phenomenon with a simple two-class Gaussian example.

Unlike the above approaches, the Neyman–Pearson (NP) classification, which was motivated by the century-long Neyman–Pearson paradigm in hypothesis testing, specifically aims to bound one type of error with high probability and meanwhile minimize the other type error. The main advantage of the NP classification is that it provides high probability guarantee on controlling one type of error under a user desired level.

This chapter is organized as follows. Section 2 provides a review of the Neyman–Pearson classification paradigm, including its theoretical and algorithmic advances. Section 3 presents three simulation examples to demonstrate how to implement the NP classification with popular classification algorithms (logistic regression, SVM, and random forests) that are widely used in genomic applications. Section 4 implements the NP classification on a genomic case study, where the goal is to classify DNA regions containing transcription factor motifs into two classes: transcription factor binding sites and non-binding sites, using two genomic features (absolute DNase-seq tag counts and DNase-seq footprint scores). Section 5 describes future research directions and potential genomic applications of the NP classification.

## 2 Neyman–Pearson Paradigm

A few commonly used notations are set up to facilitate our discussion. Let  $(X, Y)$  be random variables where  $X \in \mathcal{X} \subset \mathbb{R}^d$  is a vector of features and  $Y \in \{0, 1\}$  is a class label. A classifier  $h(\cdot)$  is a mapping  $h : \mathcal{X} \rightarrow \{0, 1\}$  that returns the predicted class given  $X$ . An error occurs when  $h(X) \neq Y$ , and the binary loss is  $\mathbb{I}(h(X) \neq Y)$ , where  $\mathbb{I}(\cdot)$  denotes the indicator function. The risk is the expected loss with respect to the joint distribution of  $(X, Y)$ :  $R(h) = \mathbb{E}[\mathbb{I}(h(X) \neq Y)] = \mathbb{P}(h(X) \neq Y)$ , which can be expressed as a weighted sum of type I and II errors:  $R(h) = \mathbb{P}(Y = 0)R_0(h) + \mathbb{P}(Y = 1)R_1(h)$ , where  $R_0(h) = \mathbb{P}(h(X) \neq Y | Y = 0)$  denotes the type I error, and  $R_1(h) = \mathbb{P}(h(X) \neq Y | Y = 1)$  denotes the type II error. While the classical binary classification aims to minimize the risk  $R(\cdot)$ , the NP classification aims to mimic the NP oracle classifier  $\phi^*$ , which is defined as



**Fig. 1** An example to illustrate that tuning the empirical type I error on training data to  $\alpha$  cannot control the type I error on test data under  $\alpha$  with high probability. The population is a two-class Gaussian distribution, where  $X$  follows  $N(-1, 1)$  under class 0 and  $N(1, 1)$  under class 1. The two classes have equal probabilities. A training data set with size  $n = 1000$  is generated from this population, and a threshold  $t = 0.635$  (the dark blue vertical line) is chosen so that the resulting classifier  $\mathbb{I}(X \geq t)$  has the observed (empirical) type I error equal to  $\alpha = 0.05$  on the training data. This classifier is then applied to  $B = 1000$  test data sets from the same population, and the resulting empirical type I errors on each of these test data sets are summarized in the histogram, which shows that approximately 50% of the type I errors are greater than  $\alpha$  and 18.1% of the errors are even greater than 0.06

$$\phi^* = \arg \min_{\phi: R_0(\phi) \leq \alpha} R_1(\phi),$$

where the user-specified level  $\alpha$  reflects a conservative attitude (priority) towards the type I error. Figure 2 shows a toy example that demonstrates the difference between a classical classifier that aims to minimize the risk and an NP classifier.

Earlier work on the NP classification came from the engineering community. Earlier theoretical development for the NP classification includes traditional statistical learning results such as probably approximately correct bounds and oracle type inequalities [8, 38, 40]. Then performance measures for the NP classification were proposed [39]. More recently, a general solution to semi-supervised novelty detection via reduction to NP classification was developed [3]. There are also other related work [9, 19]. All these work follow an empirical risk minimization (ERM) approach, and suffer a common limitation: a relaxed empirical type I error constraint is used in the optimization program, and as a result, the type I error can only be shown to satisfy some relaxed upper bound, which is bigger than  $\alpha$ .

We have worked extensively on NP classification using ERM and plug-in approaches [37, 46, 58]. We initiated a significant departure from the previous NP classification literature in [37] by arguing that a good classifier  $\hat{\phi}$  under the NP paradigm should respect the chosen significance level  $\alpha$ , rather than some relaxation of it. More concretely, the following two properties should both be satisfied with high probability.

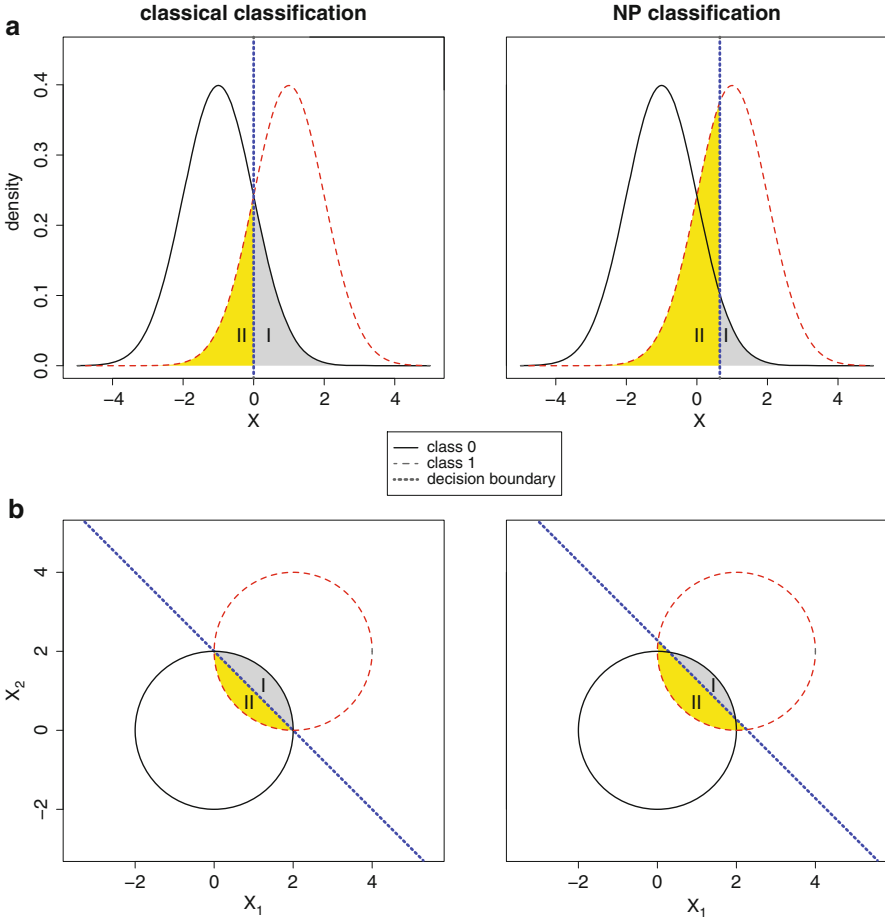
- (1) The type I error constraint is respected, i.e.,  $R_0(\hat{\phi}) \leq \alpha$ .
- (2) The excess type II error  $R_1(\hat{\phi}) - R_1(\phi^*)$  diminishes with an explicit rate (w.r.t. sample size).

We say a classifier satisfies the NP oracle inequalities if it has properties (1) and (2) with high probability. The NP oracle inequalities measure the theoretical performance of classifiers under the NP paradigm, as well as define a new NP counterpart of the well-established oracle inequalities under the classical paradigm (see [23] and the references within). In contrast, for a classifier  $\hat{h}$ , the classical oracle inequality insists that with high probability,

$$\text{the excess risk } R(\hat{h}) - R(h^*) \text{ diminishes with an explicit rate,}$$

where  $h^*(x) = \mathbb{I}(\eta(x) \geq 1/2)$  is the Bayes classifier under the classical paradigm, with  $\eta(x) = E[Y|X = x] = \mathbb{P}(Y = 1|X = x)$  denoting the regression function of  $Y$  on  $X$ .

Using a more stringent empirical type I error constraint (less than  $\alpha$ ), we established NP oracle inequalities for their proposed classifiers under convex loss functions (as opposed to the binary loss) [37]. We also proved a negative result by constructing a counterexample: under the binary loss, ERM approaches cannot guarantee diminishing excess type II error if one insists that type I error of the classifier be bounded from above by  $\alpha$  with high probability. This negative result motivated us to develop a plug-in approach to NP classification, described in [46].



**Fig. 2** Classical vs. NP classification in two binary classification examples. (a) A one-dimensional toy example where  $X$  has a two-class Gaussian distribution.  $X$  follows  $N(-1, 1)$  under class 0 and  $N(1, 1)$  under class 1. The two balanced classes have equal marginal probabilities. Suppose that a user prefers a type I error  $\leq 0.05$ . The classical classifier  $I(X \geq 0)$  that minimizes the risk would result in a type I error =  $0.16 > 0.05$ . On the other hand, the NP classifier  $I(X \geq 0.65)$  that minimizes the type II error under the type I error constraint ( $\leq 0.05$ ) delivers the desirable type I error. (b) A two-dimensional toy example where  $(X_1, X_2)$  has a two-class uniform distribution over circles.  $(X_1, X_2)$  follows a uniform distribution on  $\{X_1^2 + X_2^2 \leq 4\}$  under class 0 and a uniform distribution on  $\{(X_1 - 2)^2 + (X_2 - 2)^2 \leq 4\}$  under class 1. The two balanced classes have equal marginal probabilities. Suppose that a user prefers a classifier that is linear in  $X_1$  and  $X_2$  and has type I error  $\leq 0.05$ . The classical classifier  $I(X_1 + X_2 \leq 2)$  that minimizes the risk would result in a type I error =  $0.29 > 0.05$ . On the other hand, the NP classifier  $I(X_1 + X_2 \leq 2.28)$  that minimizes the type II error under the type I error constraint ( $\leq 0.05$ ) delivers the desirable type I error

In classical binary classification, plug-in methods that target the Bayes classifier  $\mathbb{I}(\eta(x) \geq 1/2)$  have been studied. The earliest works cast doubt on the efficacy of the plug-in approach to classification. For example, it was shown that plug-in estimators cannot achieve excess risk with rates faster than  $O(1/\sqrt{n})$  under certain assumptions [55], while direct methods can achieve fast rates up to  $O(1/n)$  under margin assumption [30, 45, 49, 50]. However, a more recent work combined a smoothness condition on  $\eta$  with the margin assumption and showed that plug-in classifiers  $\mathbb{I}(\hat{\eta}_n \geq 1/2)$  based on local polynomial estimators can achieve rates faster than  $O(1/n)$  [1].

The oracle classifier under the NP paradigm arises from its close connection to the Neyman–Pearson Lemma in statistical hypothesis testing. Hypothesis testing bears a strong resemblance to binary classification if we assume the following model. Let  $P_1$  and  $P_0$  be two known probability distributions on  $\mathcal{X} \subset \mathbb{R}^d$ . Let  $\zeta \in (0, 1)$  and assume that  $Y \sim \text{Bernoulli}(\zeta)$ . Assume further that the conditional distribution of  $X$  given  $Y$  is denoted by  $P_Y$ . Given such a model, the goal of statistical hypothesis testing is to determine whether  $X$  was generated from  $P_1$  or from  $P_0$ . To this end, we construct a randomized test  $\phi : \mathcal{X} \rightarrow [0, 1]$  and the conclusion of the test based on  $\phi$  is that  $X$  is generated from  $P_1$  with probability  $\phi(X)$  and from  $P_0$  with probability  $1 - \phi(X)$ . Two types of errors arise: type I error occurs when  $P_0$  is rejected given  $X \sim P_0$ , and type II error occurs when  $P_0$  is not rejected given  $X \sim P_1$ . The Neyman–Pearson paradigm in hypothesis testing amounts to choosing  $\phi$  that

$$\text{maximizes } E[\phi(X)|Y = 1], \text{ subject to } E[\phi(X)|Y = 0] \leq \alpha,$$

where  $\alpha \in (0, 1)$  is the significance level of the test. A solution to this constrained optimization problem is called a most powerful test of level  $\alpha$ . The Neyman–Pearson Lemma gives mild sufficient conditions for the existence of such a test.

**Theorem 1 (Neyman–Pearson Lemma).** *Let  $P_0$  and  $P_1$  be probability distributions possessing densities  $q$  and  $p$ , respectively, with respect to some measure  $\mu$ . Let  $r(x) = p(x)/q(x)$  and  $C_\alpha$  be such that  $P_0(r(X) > C_\alpha) \leq \alpha$  and  $P_0(r(X) \geq C_\alpha) \geq \alpha$ . Then for a given level  $\alpha$ , the most powerful test of level  $\alpha$  is defined by*

$$\phi^*(X) = \begin{cases} 1 & \text{if } r(X) > C_\alpha \\ 0 & \text{if } r(X) < C_\alpha \\ \frac{\alpha - P_0(r(X) > C_\alpha)}{P_0(r(X) = C_\alpha)} & \text{if } r(X) = C_\alpha \end{cases}.$$

Therefore, our plug-in target under the NP paradigm is the **oracle classifier**

$$\phi^*(x) = \mathbb{I}(r(x) \geq C_\alpha) = \mathbb{I}(\eta(x) \geq D_\alpha), \text{ where } D_\alpha = \frac{\mathbb{P}(Y = 1)C_\alpha}{\mathbb{P}(Y = 1)C_\alpha + \mathbb{P}(Y = 0)}.$$

Note that under the classical paradigm, the oracle classifier  $\mathbb{I}(\eta(x) \geq 1/2)$  puts a threshold on the regression function  $\eta$  at precisely  $1/2$ , so plug-in methods do not involve estimating the threshold level. In contrast, the NP paradigm poses more challenges because the threshold level  $C_\alpha$  or  $D_\alpha$  needs to be estimated in addition to  $r(x)$  or  $\eta(x)$ .

Also note that in practice, the threshold on  $\hat{\eta}(x)$  is often not set to  $1/2$  but is chosen by data-driven approaches such as cross validation and bootstrap. In contrast to the NP classification, these data-driven approaches aim to minimize an estimated classification risk, not the type II error with a type I error constraint.

## 2.1 An Estimate of $C_\alpha$

Pinning down a good estimate of  $C_\alpha$  is of central importance for classifiers under the NP paradigm. Contrary to common intuition, naïvely tuning the empirical type I error to  $\alpha$  does not deliver a desirable classifier, as we have shown in Fig. 1. To facilitate our discussion, we assume that our sample contains  $n$  i.i.d. observations  $\mathcal{S}^1 = \{U_1, \dots, U_n\}$  from class 1 with density  $p$ , and  $m$  i.i.d. observations  $\mathcal{S}^0 = \{V_1, \dots, V_m\}$  from class 0 with density  $q$ . The sample  $\mathcal{S}^0$  is decomposed as follows:  $\mathcal{S}^0 = \mathcal{S}_1^0 \cup \mathcal{S}_2^0$ , where  $|\mathcal{S}_1^0| = m_1$  and  $|\mathcal{S}_2^0| = m_2$ . Below is a generic procedure introduced in our recent paper [58].

### General Neyman–Pearson Plug-In Procedure

*Step 1:* Use  $\mathcal{S}^1$  and  $\mathcal{S}_1^0$  to construct a density ratio estimate  $\hat{r}$ .

*Step 2:* Given  $\hat{r}$ , choose a threshold estimate  $\hat{C}_\alpha$  from the set  $\hat{r}(\mathcal{S}_2^0) = \{\hat{r}(V_{i+m_1})\}_{i=1}^{m_2}$ . Denote by  $\hat{r}_{(k)}(\mathcal{S}_2^0)$  the  $k$ th order statistic of  $\hat{r}(\mathcal{S}_2^0)$ ,  $k \in \{1, \dots, m_2\}$ . The corresponding plug-in classifier by setting  $\hat{C}_\alpha = \hat{r}_{(k)}(\mathcal{S}_2^0)$  is

$$\hat{\phi}_k(x) = \mathbb{I}\{\hat{r}(x) \geq \hat{r}_{(k)}(\mathcal{S}_2^0)\}. \quad (1)$$

The general strategy is that for any given estimate  $\hat{r}$ , we want to find a proper order statistic  $\hat{r}_{(k)}(\mathcal{S}_2^0)$  to estimate the threshold  $C_\alpha$ , so that type I error of the classifier defined in (1) will be controlled from above by  $\alpha$  with high probability  $1 - \delta$ . To achieve this, it is necessary to study the distribution of order statistics, which we find to be beta-distributed. Based on a concentration inequality for beta distributed variables, we have derived the following high probability bound for  $R_0(\hat{\phi}_k)$ :



**Proposition 1.** *Suppose  $\hat{r}$  is such that  $F_{0,\hat{r}}(t) = P_0(\hat{r}(X) \leq t)$  is continuous almost surely. For any  $\delta \in (0, 1)$  and  $k \in \{1, \dots, m_2\}$ , it holds that*

$$\mathbb{P}\left(R_0(\hat{\phi}_k) > g(\delta, m_2, k)\right) \leq \delta,$$

where

$$g(\delta, m_2, k) = \frac{m_2 + 1 - k}{m_2 + 1} + \sqrt{\frac{k(m_2 + 1 - k)}{\delta(m_2 + 2)(m_2 + 1)^2}}.$$

Let  $\mathcal{H} = \mathcal{H}(\alpha, \delta, m_2) = \{k \in \{1, \dots, m_2\} : g(\delta, m_2, k) \leq \alpha\}$ . Proposition 1 implies that  $k \in \mathcal{H}(\alpha, \delta, m_2)$  is a sufficient condition for the classifier  $\hat{\phi}_k$  to satisfy the NP oracle inequality (1). The next step is to characterize  $\mathcal{H}$ . The smallest  $k \in \mathcal{H}$  accommodates small excess type II error for  $\hat{\phi}_k$ ; for details, please see [58].

**Proposition 2.** *The minimum  $k$  that satisfies  $g(\delta, m_2, k) \leq \alpha$  is  $k_{\min} := \lceil A_{\alpha,\delta}(m_2) \cdot (m_2 + 1) \rceil$ , where  $\lceil z \rceil$  denotes the smallest integer larger than or equal to  $z$  and*

$$A_{\alpha,\delta}(m_2) = \frac{1 + 2\delta(m_2 + 2)(1 - \alpha) + \sqrt{1 + 4\delta(1 - \alpha)\alpha(m_2 + 2)}}{2[\delta(m_2 + 2) + 1]}.$$

The choice  $k_{\min}$  coupled with a good estimate of  $r$  or  $\eta$  delivers a plug-in NP classifier that satisfies the NP oracle inequalities. We have worked out estimates based on parametric and nonparametric naïve Bayes models [58], but estimates for more complex model assumptions are not yet developed. While these directions are interesting to explore, we would like to note a limitation in the use of theoretical estimates for the threshold  $C_\alpha$ . That is, the theoretical results require concentration inequalities, which are not specific to certain types of data distributions and sometimes give threshold estimates that are too conservative in practice. Therefore, we have developed an alternative route to implement the NP paradigm. This route makes the NP classification more adaptable to popular classification algorithms and thus more useful in practice.

## 2.2 The NP Umbrella Algorithm

Here we present the alternative route, the NP umbrella algorithm we developed in [47], as pseudocodes in Algorithm 1. The essential idea is to use bootstrap to

approximate the distribution of type I errors and determine a threshold such that the corresponding classifier has type I errors bounded by a predefined level with high probability. This algorithm is widely applicable to the scoring type of classification methods, which include a wide range of popular methods, such as logistic regression [11], SVM [10], random forests [6], naïve Bayes [26], and neural networks [43]. Methods of the scoring type output a numeric value, i.e., a classification score, to represent the degree to which a test data point belongs to class 1. The classification scores can be strict probabilities or uncalibrated numeric values, as long as a higher score indicates a higher probability of an observation belonging to class 1. Many other classification methods that only output class labels can be converted to the scoring type via bagging to generate an ensemble of classifiers, each of which predicts a class label for a test data point, and the proportion of predicted labels being 1 serves as a classification score. Since almost all the state-of-the-art classification methods belong to or can be converted to the scoring type, this NP umbrella algorithm is easily adaptable in practice, though its theoretical properties are difficult to establish.

---

**Algorithm 1:** The NP umbrella classification algorithm
 

---

1: **input:**  
     training data with two parts: a mixed i.i.d. sample  $\mathcal{S}$  and a class 0 sample  
      $\mathcal{S}^0 = \{X_1, \dots, X_m\}$   
     type I error upper bound  $\alpha \in [0, 1]$   
     small tolerance level  $\delta \in (0, 1)$   
     number of bootstrap samples  $B$

2: **procedure** NPThreshold( $\mathcal{S}, \mathcal{S}^0, \alpha, \delta, B$ )

3:  $f \leftarrow \text{classificationAlgorithm}(\mathcal{S})$   
     ▷ train a classification scoring function  $f$  by inputting  $\mathcal{S}$  into the classification algorithm;  
     let  $f$  have a larger expected value for class 1 data

4:  $T_0 = (T_{0,1}, \dots, T_{0,m})^T \leftarrow (f(X_1), \dots, f(X_m))^T$   
     ▷ apply the scoring function  $f$  to  $\mathcal{S}^0$  to obtain a set of threshold candidates

5: **for**  $b$  in  $\{1, \dots, B\}$  **do** ▷ bootstrap  $T_0$  for  $B$  times

6:  $T_0^{(b)} = (T_{0,1}^{(b)}, \dots, T_{0,m}^{(b)})^T \leftarrow \text{sample}(T_0, \text{size} = m, \text{replace} = \text{TRUE})$  ▷ sample  $m$   
points with replacement from  $T_0$

7: **for**  $t$  in  $T_0$  **do** ▷ for each threshold candidate  $t$

8: **for**  $b$  in  $\{1, \dots, B\}$  **do** ▷ for each bootstrap sample  $b$

9:  $e^{(b)}(t) \leftarrow \frac{1}{m} \sum_{i=1}^m \mathbb{I}(T_{0,i}^{(b)} > t)$  ▷ calculate the type I error of threshold  $t$  in bootstrap sample  $b$

10:  $v(t) \leftarrow \frac{1}{B} \sum_{b=1}^B \mathbb{I}(e^{(b)}(t) > \alpha)$  ▷ calculate the violation rate of threshold  $t$

11:  $t^* \leftarrow \min\{t : v(t) \leq \delta\}$  ▷ pick the minimal threshold whose violation rate is under  $\delta$

12: **output:**  
     an NP classifier  $\phi(X) = \mathbb{I}(f(X) > t^*)$

---

### 3 Simulation

In this section, we demonstrate the use of NP classification with three popular classification algorithms: logistic regression [11], SVM [10], and random forests [6]. The three simulation examples, each employing one algorithm, are implemented by calling the R package `nproc` [15] we developed in recent work [47].

The `nproc` package can be installed by calling

```
> install.packages("nproc")
```

and loaded into the R environment using the next command. All the following numerical results were generated by the `nproc` package version 0.1.

```
> library(nproc)
```

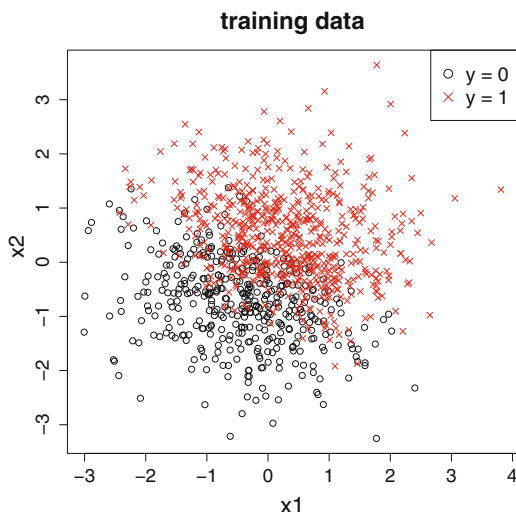
We first simulate a training data set from a logistic regression model. The training data have 1000 observations with binary responses and two-dimensional features.

```
> # training data
> set.seed(1)
> x1 <- rnorm(1000)      # feature 1
> x2 <- rnorm(1000)      # feature 2
> x <- cbind(x1, x2)     # matrix of features
> z <- 1 + 2*x1 + 3*x2    # linear combination of the
                        # two features
> pr <- 1/(1+exp(-z))    # logistic function to
                        # generate probability
> y <- rbinom(1000,1,pr) # response as Bernoulli
                        # variable
> df <- data.frame(x1=x1, x2=x2, y=y)
```

Figure 3 shows a scatterplot of the training data. We also simulate 1000 test data sets with 1000 observations from the same model, to evaluate the performance of classifiers.

```
> # test data
> test_data <- lapply(1:1000, FUN=function(i) {
+   set.seed(i+1)
+   x1 <- rnorm(1000)
+   x2 <- rnorm(1000)
+   x <- cbind(x1, x2)
+   z <- 1 + 2*x1 + 3*x2
+   pr <- 1/(1+exp(-z))
+   y <- rbinom(1000, 1, pr)
+   df <- data.frame(x1=x1, x2=x2, y=y)
+   return(list(x=x, y=y, df=df))
+ })
```

**Fig. 3** Scatterplot of the training data with 1000 observations and two-dimensional features. *Black circles* and *red crosses* represent class 0 and class 1, respectively



### 3.1 Logistic Regression

Logistic regression is a type of generalized linear model. For binary classification purposes, logistic regression can serve as a classification method. One way to interpret logistic regression is that it models

$$p := \mathbb{P}(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta^T X)}}$$

for a binary response  $Y$  with  $d$  features  $X \in \mathbb{R}^d$ . Given training data  $\{(x_i, y_i)\}_{i=1}^n$ , logistic regression estimates  $\beta_0$  and  $\beta$  as maximum likelihood estimates  $\hat{\beta}_0$  and  $\hat{\beta}$ , and then estimates  $p$  as  $\hat{p} = \left(1 + e^{-(\hat{\beta}_0 + \hat{\beta}^T X)}\right)^{-1}$ . This estimated probability can be interpreted as a classification score, as logistic regression predicts  $Y$  by a linear decision rule, i.e.,  $\hat{Y} = \mathbb{I}(\hat{\beta}^T X \geq c) = \mathbb{I}\left(\hat{p} \geq \left(1 + e^{-\hat{\beta}_0 - c}\right)^{-1}\right)$  for some threshold  $c \in \mathbb{R}$ . Under the classical classification paradigm,  $c = -\hat{\beta}_0$ , which corresponds to  $\hat{p} \geq 1/2$ . Under the NP paradigm, we potentially have different choices for  $c$ . By regarding  $\hat{p}$  as a classification scoring function of  $X$ , we use the umbrella algorithm to find a threshold on  $\hat{p}$  so that the resulting classifier will have type I error below the desired level  $\alpha$  with high probability close to  $1 - \delta$ .

We use the simulated data to demonstrate the use of NP classification with logistic regression and compare it with the classical paradigm. We first train a logistic regression model on the training data under the classical paradigm.

```
> lr_model1 <- glm(y~x1+x2, data=df, family="binomial")
```

Then we apply the trained model `lr_model1` to the 1000 test data sets to evaluate the distribution of its empirical type I errors on test data.

```
> lr_model1_err <- sapply(test_data, FUN=function(tdat)
{ + pred <- predict(lr_model1, tdat$df,
type="response") > 0.5 + ind0 <- which(tdat$y==0)
+ typeI <- mean(pred[ind0]!=tdat$y[ind0])
+ })
> summary(lr_model1_err)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.1290 0.1723 0.1846 0.1849 0.1980 0.2368
```

We next train a logistic regression model on the training data under the NP paradigm with type I error bound  $\alpha = 0.05$ , using the `npc` function.

```
> set.seed(1001)
# for reproducible purposes, because npc()
  involves bootstrap,
# whose results will be reproducible with a fixed seed
> lr_model2 <- npc(x=x, y=y, method='logistic',
  alpha=0.05)
```

Then we also applied the trained model `lr_model2` to the test data.

```
> lr_model2_err <- sapply(test_data, FUN=function(tdat){
+ pred <- predict(lr_model2, tdat$x)
+ ind0 <- which(tdat$y==0)
+ typeI <- mean(pred$pred.label[ind0]!=tdat$y[ind0])
+ })
> summary(lr_model2_err)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.009732 0.027230 0.033530 0.033950 0.040400 0.062330

> sum(lr_model2_err <= 0.05) / 1000

[1] 0.948
```

Comparing the empirical type I errors of the two logistic regression classifiers found under the classical and the NP paradigm, respectively, we can see that the NP classifier gives much smaller type I errors, 94.8% of which are under  $\alpha = 0.05$ .

### 3.2 Support Vector Machines

Similar to logistic regression, SVM is also a scoring type of classification method, for which approximate posterior probabilities of class labels proposed by Platt [35]

can be used as classification scores. We demonstrate the use of NP classification with SVM on the simulated data as follows.

We first train an SVM model on the training data under the NP paradigm.

```
> set.seed(1001)
> svm_model <- npc(x=x, y=y, method='svm', alpha=0.05)
```

Then we apply the trained model `svm_model` to the 1000 test data sets to evaluate the distribution of its empirical type I errors on test data.

```
> svm_model_err <- sapply(test_data, FUN=function(tdat)
+ {
+   pred <- predict(svm_model, tdat$x)
+   ind0 <- which(tdat$y==0)
+   typeI <- mean(pred$pred.label[ind0]!=tdat$y[ind0])
+ })
> summary(svm_model_err)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.007772 0.023530 0.029560 0.029900 0.035350 0.062330

> sum(svm_model_err <= 0.05) / 1000

[1] 0.987
```

We see that the SVM classifier found by the NP algorithm has empirical type I errors under  $\alpha = 0.05$  with high probability.

### 3.3 Random Forests

Random forests is another popular and powerful classification method. It is an ensemble method of tree-based classifiers. We can also interpret it as a scoring type of method, if we consider the proportion of output votes for class 1 (i.e., the proportion of trees that predict an observation as class 1) as classification scores. We demonstrate the use of NP classification with random forests on the toy data as follows.

We first train a random forest model on the training data under the NP paradigm.

```
> set.seed(1001)
> rf_model <- npc(x=x, y=y, method='randomforest',
+   alpha=0.05)
```

Then we apply the trained model `rf_model` to the 1000 test data sets to evaluate the distribution of its empirical type I errors on test data.

```

> rf_model_err <- sapply(test_data, FUN=function(tdat) {
+   pred <- predict(rf_model, tdat$x)
+   ind0 <- which(tdat$y==0)
+   typeI <- mean(pred$pred.label[ind0]!=tdat$y[ind0])
+ })
> summary(rf_model_err)

      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.002451 0.021000 0.026280 0.026810 0.032260 0.054460

> sum(rf_model_err <= 0.05) / 1000

[1] 0.996

```

We see that the random forest classifier found by the NP algorithm has empirical type I errors under  $\alpha = 0.05$  with high probability.

## 4 Case Study

We demonstrate the use of NP classification in a genomic case study on the prediction of transcription factor binding sites. A recent study [20] found that DNase-seq signals can well predict whether genomic regions containing transcription factor sequence motifs are transcription factor binding sites. DNase-seq is a recent high-throughput technology that combines traditional DNaseI footprinting [17] and next-generation DNA sequencing to identify genomic regions where regulatory factors interact with DNA to modify chromatin structure [5, 12, 31, 42]. An important question investigated in this study is which DNase-seq features can well predict binding sites of CTCF, a transcription factor that acts as an insulator to regulate the 3D structure of chromatin [34].

The study [20] formulated this question as a binary classification problem, where the goal is to classify genomic regions that contain CTCF sequence motifs into CTCF binding sites (i.e., class 1) and non-binding sites (i.e., class 0). For this task, two one-dimensional genomic features extracted from DNase-seq data are compared: (1) the number of DNase-seq tags in a 200 base pair window centered in each CTCF motif site and (2) the DNaseI footprint score calculated using the formula  $f = -[(n_C + 1)/(n_R + 1) + (n_C + 1)/(n_L + 1)]$ , where  $n_C$ ,  $n_R$ , and  $n_L$  represent, respectively, the tag count in the motif region and the flanking regions to the right and left of the motif (the lengths of the flanks are both the same as that of the motif). In the data, there are  $n = 216,929$  genomic regions, each with one tag count and one footprint score. Among these regions, 27,220 regions were found as CTCF binding sites (class 0), and the rest 189,709 regions were considered as non-binding sites (class 1). By varying the threshold on each feature, the study showed that the footprint score outperforms the tag count at low FPR and underperforms at higher FPR. In other words, if users desire a small type I error, for example, if they

prefer to predict fewer but more confident CTCF binding sites, the footprint score is a better genomic feature; otherwise if users prefer a small type II error, for example, if they prefer to predict more potential CTCF binding sites, the tag count is a better feature.

However, this study only reported the observed (empirical) type I and II errors on one data set, without assessing the randomness of these empirical errors. If a user is interested in knowing which feature is better when the type I error (or FDR) is constrained under  $\alpha$  (e.g., 5%) with high probability, this analysis cannot provide a good answer. Here we address this question using the NP classification algorithm described in Sect. 2.2. We also compare the performance of the NP classification with the common practice, which is to tune the empirical type I error on the training data to  $\alpha$ . At three different type I error bounds  $\alpha = 0.01, 0.05, \text{ and } 0.1$ , we find their corresponding thresholds on the number of DNase-seq tags or the footprint scores via the common practice or the NP approach. Specifically, in the common practice we find the thresholds as the 99th, 95th, and 90th percentiles of the number of DNase-seq tags or the footprint scores of the 189, 709 non-binding sites; in the NP approach we use the NP algorithm with the number of bootstrap runs  $B = 1000$  and the violation tolerance level  $\delta = 0.05$  to find the thresholds on the number of DNase-seq tags or the footprint scores of the 189, 709 non-binding sites. Since values of each feature serve as natural classification scores in this case, we do not need to train a classification scoring function, and all the class 0 data points can be used to find the NP thresholds. That is, in the algorithm,  $\mathcal{S}^0$  contains all the class 0 data points, and the  $f$  function in Step 3 is just the identity map.

We evaluate the thresholds and their corresponding classifiers found by the NP approach or the common practice via bootstrap. We generate  $B' = 1000$  sets of  $n = 216, 929$  bootstrap regions with corresponding tag counts and footprint scores from the original data via random sampling with replacement. Then we evaluate the empirical type I and type II errors of each threshold on each bootstrap data set, and summarize the distribution of the empirical type I and type II errors of all bootstrap data sets. The results in Table 1 show that the classifiers found by the NP approach have empirical type I errors bounded by  $\alpha$  with high probability, while the classifiers found by the common practice have large portions of empirical type I errors above  $\alpha$ . Comparing the means of the empirical classification errors, we can see that the classifiers found by the NP approach have slightly smaller mean type I errors and slightly larger mean type II errors, a reasonable result given its high probability bound on the type I errors. The standard deviations are similar for the two approaches. Back to the question about which feature is better when the type I error is under  $\alpha$  with high probability, the NP classification results suggest that at  $\alpha = 0.01$  and  $0.05$ , the footprint score is a better feature, while the tag count is a better feature at  $\alpha = 0.1$ .



**Table 1** Comparison of classifiers established by the common practice vs. the NP approach

Features	$\alpha$	Approaches	Type I errors			Type II errors	
			Mean <sup>a</sup>	sd <sup>b</sup>	% > $\alpha$ <sup>c</sup>	Mean	sd
Tag counts	0.01	Common	0.010	$2.32e^{-4}$	46.5	0.976	$9.34e^{-4}$
		NP	0.009	$2.25e^{-4}$	0.00	0.979	$8.84e^{-4}$
	0.05	Common	0.050	$4.83e^{-4}$	48.4	0.707	$2.78e^{-3}$
		NP	0.049	$4.78e^{-4}$	1.10	0.716	$2.78e^{-3}$
	0.10	Common	0.100	$6.89e^{-4}$	39.8	0.335	$2.86e^{-3}$
		NP	0.099	$6.83e^{-4}$	3.90	0.340	$2.87e^{-3}$
Footprint scores	0.01	Common	0.010	$2.25e^{-4}$	48.6	0.766	$2.59e^{-3}$
		NP	0.009	$2.07e^{-4}$	0.00	0.775	$2.59e^{-3}$
	0.05	Common	0.050	$4.96e^{-4}$	33.5	0.596	$2.97e^{-3}$
		NP	0.049	$4.90e^{-4}$	1.90	0.598	$2.96e^{-3}$
	0.10	Common	0.100	$6.58e^{-4}$	49.5	0.493	$3.07e^{-3}$
		NP	0.099	$6.54e^{-4}$	1.40	0.494	$3.06e^{-3}$

<sup>a</sup> Mean of the empirical classification errors over the  $B' = 1000$  bootstrap runs

<sup>b</sup> Standard deviation of the empirical classification errors over the  $B' = 1000$  bootstrap runs

<sup>c</sup> Percentage of the empirical type I errors that are greater than  $\alpha$  in the  $B' = 1000$  bootstrap runs

## 5 Future Research and Genomic Applications of the Neyman–Pearson Classification

Neyman–Pearson classification paradigm handles binary class classification problem, but it can be extended to address multi-class problems where errors are asymmetric in nature. Neyman–Pearson classification has wide application potentials in genomics. In Sects. 5.2–5.4 we describe three potential applications as future research directions: sample size determination, automatic disease diagnosis, and disease marker detection. The latter two applications were also discussed in our recent review paper [48].

### 5.1 Extension to Multi-class

Originating from binary trade-offs, the NP classification methods can also be applied to multi-class ( $Y \in \{1, \dots, K\}$ ,  $K \geq 3$ ) problems using the following two strategies:

- **[Strategy 1]** Missing class 1 has more severe consequences than missing other classes. A two-step procedure can be implemented: Apply an NP method to classify a subject into class 1 versus the other classes. Stop if the subject is assigned to class 1. Otherwise, continue

and apply a (multi-class) classification algorithm to assign this subject to one of the other classes  $\{2, \dots, K\}$ .

- **[Strategy 2]** There is a hierarchical order (class 1  $>$   $\dots$   $>$  class  $K$ ) of class priorities (i.e., severity of missing each class). A possible procedure is to first apply an NP method to classify a subject into class 1 versus other classes  $\{2, \dots, K\}$ . Stop if this subject is assigned to class 1. Otherwise, apply again the NP method to classify it into class 2 versus classes  $\{3, \dots, K\}$ . Continue along this line until this subject is assigned to a class.

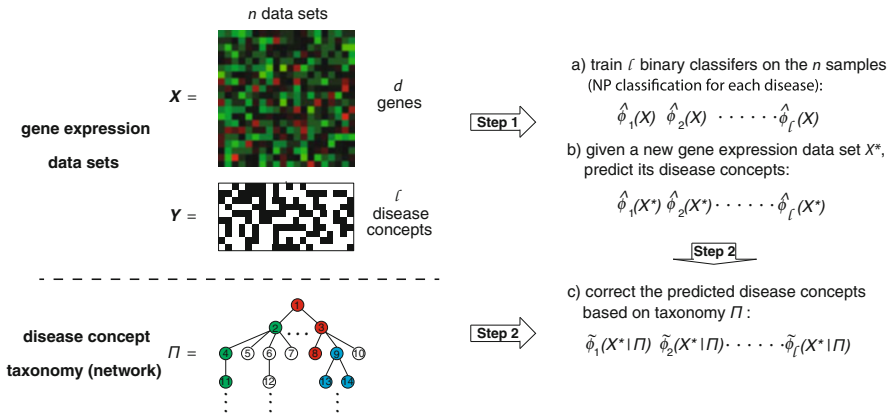
## 5.2 Sample Size Determination

In clinical trials and other experimental designs, sample acquisition can be quite expensive. Therefore, how to determine the minimal sufficient sample size is an important question. Admittedly, different criteria would lead to different sample size selection procedures; so there is no hope of finding a universally golden rule. The Neyman–Pearson paradigm inspires one procedure to choose the sample size based on the theoretical upper bound of the excess type II error in the NP oracle inequalities.

We describe a contrived version of this procedure in the following. Suppose a classifier  $\hat{\varphi}$  satisfies the NP oracle inequalities, and the excess type II error of  $\hat{\varphi}$  is bounded from above by  $f(m, n)$ , where  $m$  is the sample size of class 0, and  $n$  is the sample size of class 1, and  $f(m, n) \rightarrow 0$  as  $m$  and  $n$  go to infinity. If the user has a target to control the excess type II error at some  $\gamma > 0$ . Then  $m$  and  $n$  can be selected such that  $f(m, n) \leq \gamma$ . As this procedure is based on concentration inequalities, it is conservative and might overestimate the sample size. Yet this procedure provides a valid upper bound of the needed sample size. Future statistical research is need for more accurate sample size determination.

## 5.3 Automatic Disease Diagnosis

Another application is a long-time challenge in clinical research: automatic disease diagnosis from patient genomic data. This challenge involves a classification problem, where diseases correspond to different classes, and the goal is to predict the diseases that are most likely associated with a patient’s genomic sample. Thanks to the development of high-throughput genomic technologies [e.g., microarray and next-generation sequencing (NGS)], a large amount of disease related genomic data can serve as training data in this classification problem. Taking gene expression data as an example, the National Center for Biotechnology Information (NCBI) Gene Expression Omnibus (GEO) contains more than 690,000 human gene expression samples that are related to hundreds of diseases, such as heart diseases, mental illnesses, infectious diseases, and various cancers.

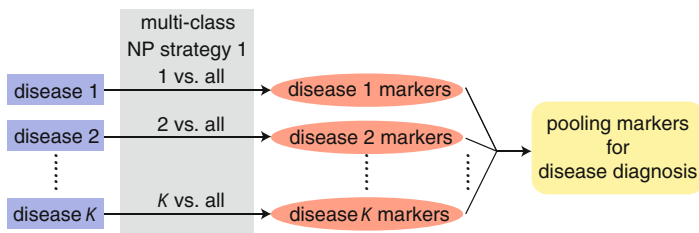


**Fig. 4** Automatic disease diagnosis via NP classification and network-assisted correction

We can study automatic disease diagnosis by NP classification and network-assisted correction using a two-step approach (Fig. 4). Step 1: using public microarray and NGS gene expression data sets with disease labels (e.g., > 100 Unified Medical Language System standardized disease concepts), we can (a) use NP classification to build a binary classifier for each disease class, and (b) classify a patient’s microarray gene expression sample into these disease classes. Step 2: (c) correct the predicted diseases based on the disease taxonomy (network). In Step 1, since the disease classes are non-exclusive (one data set may have multiple disease class labels), this multi-label classification problem is inherently composed of multiple binary classification problems, where every disease class needs a binary decision. In previous works [21, 27], binary classifiers such as SVM and naive Bayes classifiers were used, and all disease classes were treated as interchangeable. This raises an important issue, though: some diseases are more life-threatening than others, e.g., lung cancer vs. arthritis. Therefore, it is important to allow doctors to have different levels of conservativeness, i.e., different thresholds  $\alpha$  on the type I error (the chance of missing a disease when a patient in fact has it), for different diseases. Although previous researchers have attempted to address this trade-off between false positives and false negatives in disease diagnosis [16], they failed to control false negative rates under a desired threshold with high probability. Given the pressing need for precise disease diagnosis, the developed NP classification algorithms are in high demand to address this issue.

### 5.4 Disease Marker Detection

The multi-class extension of NP classification has application potentials in detecting and screening for key markers (i.e., genes and genomic features) to aid disease diagnosis as well as to understand molecular mechanisms of diseases. In early cancer diagnosis studies that aimed to determine which genes should be included



**Fig. 5** Marker detection via multi-class NP strategy 1

as features (markers) [16, 36, 41], classification error of each disease class versus others was used as a criterion. In other words, “the smallest set” of genes that results in low classification error for a disease class was retained as markers for that disease. However, this criterion lacks consideration of asymmetric classification errors, and as a result, the selected markers for a disease could lead to high false negative rates in the diagnosis—a dangerous situation for severe diseases such as cancers. Therefore, in the diagnosis of severe diseases, a more reasonable criterion would be to minimize the FPR given a pre-specified false negative rate control. The multi-class NP classification (Strategy 1) serves the purpose: key markers are selected so that low FPR are attained while the false negative rates are constrained below a threshold (see Fig. 5). Markers selected by this new detection strategy can be pooled to make disease prediction, in the hope of increasing the sensitivity of disease diagnosis. To implement and evaluate this strategy, we need to compare it with the more recent state-of-the-art disease prediction methods, which are for example based on multi-task learning [2, 28, 57, 59], group lasso [29], multicategory support vector machines [25], partial least squares regression [4, 33], neural networks [22, 51], and others [44].

**Acknowledgements** Dr. Jingyi Jessica Li’s work was supported by the start-up fund of the UCLA Department of Statistics and the Hellman Fellowship. Dr. Xin Tong’s work was supported by Zumberge Individual Award from University of Southern California and summer research support from Marshall School of Business. We thank Dr. Yang Feng in Department of Statistics at Columbia University and Ms. Anqi Zhao in Department of Statistics at Harvard University for their help in developing the Neyman–Pearson classification algorithms. We also thank Dr. Wei Li and Mr. Sheng’en Shawn Hu in Dr. X. Shirley Liu’s group in Department of Biostatistics and Computational Biology at Dana-Farber Cancer Institute and Harvard School of Public Health for kindly sharing the data for our genomic case study in Sect. 4.

## References

1. Audibert, J., Tsybakov, A.: Fast learning rates for plug-in classifiers under the margin condition. *Annals of Statistics* **35**, 608–633 (2007)
2. Bi, J., Xiong, T., Yu, S., Dundar, M., Rao, R.B.: An improved multi-task learning approach with applications in medical diagnosis. In: *Machine Learning and Knowledge Discovery in Databases*, pp. 117–132. Springer (2008)

3. Blanchard, G., Lee, G., Scott, C.: Semi-supervised novelty detection. *Journal of Machine Learning Research* **11**, 2973–3009 (2010)
4. Booij, B.B., Lindahl, T., Wetterberg, P., Skaane, N.V., Sæbø, S., Feten, G., Rye, P.D., Kristiansen, L.I., Hagen, N., Jensen, M., et al.: A gene expression pattern in blood for the early detection of Alzheimer's disease. *Journal of Alzheimer's Disease* **23**(1), 109–119 (2011)
5. Boyle, A.P., Song, L., Lee, B.K., London, D., Keefe, D., Birney, E., Iyer, V.R., Crawford, G.E., Furey, T.S.: High-resolution genome-wide in vivo footprinting of diverse transcription factors in human cells. *Genome research* **21**(3), 456–464 (2011)
6. Breiman, L.: Random forests. *Machine learning* **45**(1), 5–32 (2001)
7. Bulyk, M.L., et al.: Computational prediction of transcription-factor binding site locations. *Genome biology* **5**(1), 201–201 (2004)
8. Cannon, A., Howse, J., Hush, D., Scovel, C.: Learning with the Neyman-Pearson and min-max criteria. Technical Report LA-UR-02-2951 (2002)
9. Casasent, D., Chen, X.: Radial basis function neural networks for nonlinear fisher discrimination and Neyman-Pearson classification. *Neural Networks* **16**(5–6), 529–535 (2003)
10. Cortes, C., Vapnik, V.: Support-vector networks. *Machine learning* **20**(3), 273–297 (1995)
11. Cox, D.R.: The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 215–242 (1958)
12. Degner, J.F., Pai, A.A., Pique-Regi, R., Veyrieras, J.B., Gaffney, D.J., Pickrell, J.K., De Leon, S., Michelini, K., Wellen, N., Crawford, G.E., et al.: DNase I sensitivity QTLs are a major determinant of human expression variation. *Nature* **482**(7385), 390–394 (2012)
13. Dümmbgen, L., Igl, B., Munk, A.: P-values for classification. *Electronic Journal of Statistics* **2**, 468–493 (2008)
14. Elkan, C.: The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* pp. 973–978 (2001)
15. Feng, Y., Li, J., Tong, X.: nproc: Neyman-Pearson Receiver Operator Curve (2016). URL <http://CRAN.R-project.org/package=nproc>. R package version 0.1
16. Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16**(10), 906–914 (2000)
17. Galas, D.J., Schmitz, A.: DNase footprinting a simple method for the detection of protein-DNA binding specificity. *Nucleic acids research* **5**(9), 3157–3170 (1978)
18. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., et al.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science* **286**(5439), 531–537 (1999)
19. Han, M., Chen, D., Sun, Z.: Analysis to Neyman-Pearson classification with convex loss function. *Anal. Theory Appl.* **24**(1), 18–28 (2008). DOI 10.1007/s10496-008-0018-3
20. He, H.H., Meyer, C.A., Chen, M.W., Zang, C., Liu, Y., Rao, P.K., Fei, T., Xu, H., Long, H., Liu, X.S., et al.: Refined DNase-seq protocol and data analysis reveals intrinsic bias in transcription factor footprint identification. *Nature methods* **11**(1), 73–78 (2014)
21. Huang, H., Liu, C.C., Zhou, X.J.: Bayesian approach to transforming public gene expression repositories into disease diagnosis databases. *Proceedings of the National Academy of Sciences* **107**(15), 6823–6828 (2010)
22. Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine* **7**(6), 673–679 (2001)
23. Koltchinskii, V.: Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems (2008)
24. Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: Supervised machine learning: A review of classification techniques. *Informatica* **31**, 249–268 (2007)
25. Lee, Y., Lee, C.K.: Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics* **19**(9), 1132–1139 (2003)

26. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: *Machine learning: ECML-98*, pp. 4–15. Springer (1998)
27. Liu, C.C., Hu, J., Kalakrishnan, M., Huang, H., Zhou, X.J.: Integrative disease classification based on cross-platform microarray data. *BMC Bioinformatics* **10**(Suppl 1), S25 (2009)
28. Liu, F., Wee, C.Y., Chen, H., Shen, D.: Inter-modality relationship constrained multi-modality multi-task feature selection for Alzheimer’s disease and mild cognitive impairment identification. *NeuroImage* **84**, 466–475 (2014)
29. Ma, S., Song, X., Huang, J.: Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics* **8**(1), 1 (2007)
30. Mammen, E., Tsybakov, A.: Smooth discrimination analysis. *Annals of Statistics* **27**, 1808–1829 (1999)
31. Neph, S., Vierstra, J., Stergachis, A.B., Reynolds, A.P., Haugen, E., Vernot, B., Thurman, R.E., John, S., Sandstrom, R., Johnson, A.K., et al.: An expansive human regulatory lexicon encoded in transcription factor footprints. *Nature* **489**(7414), 83–90 (2012)
32. Ng, K.L.S., Mishra, S.K.: De novo svm classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics* **23**(11), 1321–1330 (2007)
33. Park, P.J., Tian, L., Kohane, I.S.: Linking gene expression data with patient survival times using partial least squares. *Bioinformatics* **18**(suppl 1), S120–S127 (2002)
34. Phillips, J.E., Corces, V.G.: Ctf: master weaver of the genome. *Cell* **137**(7), 1194–1211 (2009)
35. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* **10**(3), 61–74 (1999)
36. Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C.H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J.P., Poggio, T., Gerald, W., Loda, M., Lander, E.S., Golub, T.R.: Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences* **98**(26), 15,149–15,154 (2001)
37. Rigollet, P., Tong, X.: Neyman-Pearson classification, convexity and stochastic constraints. *Journal of Machine Learning Research* **12**, 2831–2855 (2011)
38. Scott, C.: Comparison and design of Neyman-Pearson classifiers. Unpublished (2005)
39. Scott, C.: Performance measures for Neyman-Pearson classification. *IEEE Transactions on Information Theory* **53**(8), 2852–2863 (2007)
40. Scott, C., Nowak, R.: A Neyman-Pearson approach to statistical learning. *IEEE Transactions on Information Theory* **51**(11), 3806–3819 (2005)
41. Segal, N.H., Pavlidis, P., Antonescu, C.R., Maki, R.G., Noble, W.S., DeSantis, D., Woodruff, J.M., Lewis, J.J., Brennan, M.F., Houghton, A.N., Cordon-Cardo, C.: Classification and subtype prediction of adult soft tissue sarcoma by functional genomics. *The American Journal of Pathology* **163**(2), 691–700 (2003)
42. Song, L., Zhang, Z., Grasfeder, L.L., Boyle, A.P., Giresi, P.G., Lee, B.K., Sheffield, N.C., Gräf, S., Huss, M., Keefe, D., et al.: Open chromatin defined by DNaseI and faire identifies regulatory elements that shape cell-type identity. *Genome research* **21**(10), 1757–1767 (2011)
43. Specht, D.F.: Probabilistic neural networks. *Neural networks* **3**(1), 109–118 (1990)
44. Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multiclassification methods for microarray gene expression cancer diagnosis. *Bioinformatics* **21**(5), 631–643 (2005)
45. Tarigan, B., van de Geer, S.: Classifiers of support vector machine type with  $l_1$  complexity regularization. *Bernoulli* **12**, 1045–1076 (2006)
46. Tong, X.: A plug-in approach to Neyman-Pearson classification. *Journal of Machine Learning Research* **14**, 3011–3040 (2013)
47. Tong, X., Feng, Y., Li, J.J.: Neyman-pearson (np) classification algorithms and np receiver operating characteristic (np-roc) curves Manuscript
48. Tong, X., Feng, Y., Zhao, A.: A survey on Neyman-Pearson classification and suggestions for future research. *Wiley Interdisciplinary Reviews: Computational Statistics* **8**, 64–81 (2016)
49. Tsybakov, A.: Optimal aggregation of classifiers in statistical learning. *Annals of Statistics* **32**, 135–166 (2004)

50. Tsybakov, A., van de Geer, S.: Square root penalty: Adaptation to the margin in classification and in edge estimation. *Annals of Statistics* **33**, 1203–1224 (2005)
51. Wei, J.S., Greer, B.T., Westermann, F., Steinberg, S.M., Son, C.G., Chen, Q.R., Whiteford, C.C., Bilke, S., Krasnoselsky, A.L., Cenacchi, N., et al.: Prediction of clinical outcome using gene expression profiling and artificial neural networks for patients with neuroblastoma. *Cancer research* **64**(19), 6883–6891 (2004)
52. Wu, S., Lin, K., Chen, C., M., C.: Asymmetric support vector machines: low false-positive learning under the user tolerance (2008)
53. Xing, E.P., Jordan, M.I., Karp, R.M., et al.: Feature selection for high-dimensional genomic microarray data. In: *ICML*, vol. 1, pp. 601–608. Citeseer (2001)
54. Yanai, I., Benjamin, H., Shmoish, M., Chalifa-Caspi, V., Shklar, M., Ophir, R., Bar-Even, A., Horn-Saban, S., Safran, M., Domany, E., et al.: Genome-wide midrange transcription profiles reveal expression level relationships in human tissue specification. *Bioinformatics* **21**(5), 650–659 (2005)
55. Yang, Y.: Minimax nonparametric classification-part i: rates of convergence. *IEEE Transaction Information Theory* **45**, 2271–2284 (1999)
56. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. *IEEE International Conference on Data Mining* p. 435 (2003)
57. Zhang, D., Shen, D., Initiative, A.D.N., et al.: Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer’s disease. *NeuroImage* **59**(2), 895–907 (2012)
58. Zhao, A., Feng, Y., Wang, L., Tong, X.: Neyman-Pearson classification under high dimensional settings (2015). URL <http://arxiv.org/abs/1508.03106>
59. Zhou, J., Yuan, L., Liu, J., Ye, J.: A multi-task learning formulation for predicting disease progression. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 814–822. ACM (2011)