Neyman-Pearson Classification Paradigm

Jingyi Jessica Li

Department of Statistics

University of California, Los Angeles

joint work with Xin Tong (USC), Yang Feng (Columbia), and Yiling Chen (UCLA)

Binary classification concepts and notations

- Features $X \in \mathcal{X} \subset \mathbb{R}^p$

  e.g., health characteristics of a person

- Class label $Y \in \{0, 1\}$

  e.g., disease status (YES or NO) of a person

Binary classification concepts and notations

- Features $X \in \mathcal{X} \subset \mathbb{R}^p$

  e.g., health characteristics of a person

- Class label $Y \in \{0, 1\}$

  e.g., disease status (YES or NO) of a person

- A classifier is a data dependent binary function $h : \mathcal{X} \to \{0, 1\}$

Binary classification concepts and notations

- Features $X \in \mathcal{X} \subset \mathbb{R}^p$

  e.g., health characteristics of a person

- Class label $Y \in \{0, 1\}$

  e.g., disease status (YES or NO) of a person

- A classifier is a data dependent binary function $h : \mathcal{X} \to \{0, 1\}$
- Training data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ (often assumed as i.i.d.)

Binary classification concepts and notations

- Features $X \in \mathcal{X} \subset \mathbb{R}^p$

  e.g., health characteristics of a person

- Class label $Y \in \{0, 1\}$

  e.g., disease status (YES or NO) of a person

- A classifier is a data dependent binary function $h : \mathcal{X} \to \{0, 1\}$
- Training data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ (often assumed as i.i.d.)
- A classification method is a way to construct a classifier h from training data (e.g., logistic regression, support vector machines, and random forests)

  h is a function of $(X_1, Y_1), \ldots, (X_n, Y_n)$ and thus random

Classification errors

Let (X, Y) denote random variables independent of and identically distributed as (X_i, Y_i) in the training data

- Classification error ("risk")

$$\begin{aligned} R(h) &= \mathbb{P}(h(X) \neq Y) \\ &= \mathbb{P}(Y = 0) R_0(h) + \mathbb{P}(Y = 1) R_1(h), \end{aligned}$$

where

$R_0(h) = \mathbb{P}\left(h(X) \neq Y | Y = 0\right)$ denotes the type I error,
$R_1(h) = \mathbb{P}\left(h(X) \neq Y | Y = 1\right)$ denotes the type II error.

Classification errors

Let (X, Y) denote random variables independent of and identically distributed as (X_i, Y_i) in the training data

- Classification error ("risk")

$$\begin{aligned} R(h) &= \mathbb{P}(h(X) \neq Y) \\ &= \mathbb{P}(Y = 0) R_0(h) + \mathbb{P}(Y = 1) R_1(h), \end{aligned}$$

  where

  $R_0(h) = \mathbb{P}\left(h(X) \neq Y | Y = 0\right)$ denotes the type I error,
  $R_1(h) = \mathbb{P}\left(h(X) \neq Y | Y = 1\right)$ denotes the type II error.

- Randomness

  $\mathbb{P}$ w.r.t. the joint distribution of (X, Y)
  $\mathbb{P}$ w.r.t. the marginal distribution of Y
  $\mathbb{P}$ and $\mathbb{P}$ w.r.t. the conditional distribution of $X \mid (Y = 0)$ and $X \mid (Y = 1)$
  All conditional on h (i.e., conditional on the training data)

- $R(h)$, $R_0(h)$ and $R_1(h)$ are all random if h is random

Classical paradigm (theory)

- Classical goal: find a classifier h to minimize $R(h)$
- Classical oracle classifier

$$h^* = \text{argmin}_h R(h)$$

an unobservable (fixed) classifier among a class of binary functions

- Bayes classifier = classical oracle classifier

$$h^*(x) = \mathbb{I}(\eta(x) \geq 1/2)$$

where $\eta(x) = \mathbb{P}(Y = 1|X = x)$.

Classical paradigm (theory)

Classical oracle inequality

$$\mathbb{P}\left(|R(\hat{h}) - R(h^*)| \leq f(n)\right) > 1 - \delta\,,$$

where

- $\hat{h}$ is a (random) classifier trained from the training data
- h^* is the (fixed) Bayes classifier not observable
- f is a decreasing function of sample size n, $f(n) \to 0$ as $n \to \infty$
- $\delta \in (0, 1)$ is a small constant indicating the violation probability
- $1 - \delta$ is often referred to as high probability

Classical paradigm (theory)

Two approaches to construct $\hat{h}$

- Empirical risk minimization

$$\hat{h} = \text{argmin}_h \hat{R}(h)$$

where $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(h(X_i) \neq Y_i)$ is the empirical risk

- Plug in

$$h^*(x) = \mathbb{I}(\eta(x) \geq 1/2)$$
$$\eta(x) = \mathbb{P}(Y = 1|X = x) = \frac{f_1(x)\mathbb{P}(Y=1)}{f_0(x)\mathbb{P}(Y=0)+f_1(x)\mathbb{P}(Y=1)}$$
$$\hat{\eta}(x) = \widehat{\mathbb{P}}(Y = 1|X = x) = \frac{\hat{f}_1(x)\widehat{\mathbb{P}}(Y=1)}{\hat{f}_0(x)\widehat{\mathbb{P}}(Y=0)+\hat{f}_1(x)\widehat{\mathbb{P}}(Y=1)}$$

$\hat{\eta}(x)$ is constructed from the training data

$$\hat{h}(x) = \mathbb{I}(\hat{\eta}(x) \geq 1/2)$$

Motivations for asymmetric error control

Classification error ("risk")

$$\begin{aligned} R(h) &= \mathbb{P}(h(X) \neq Y) \\ &= \mathbb{P}(Y = 0) R_0(h) + \mathbb{P}(Y = 1) R_1(h), \end{aligned}$$

where

- $R_0(h) = \mathbb{P}\left(h(X) \neq Y | Y = 0\right)$ denotes the type I error,
- $R_1(h) = \mathbb{P}\left(h(X) \neq Y | Y = 1\right)$ denotes the type II error.

Example: cancer diagnosis

- Mispredicting a normal tissue sample as malignant
  $\Rightarrow$ patient anxiety & additional medical costs
- Mispredicting a tumor sample as normal
  $\Rightarrow$ life loss 😭

We need classifiers to enable asymmetric error control

Population and empirical classification errors

Let $\hat{h}$ be a classifier constructed from the training data

$$\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

Denote the test data as

$$\{(X_1', Y_1'), \ldots, (X_m', Y_m')\}$$

Risk

- Population

$$R(\hat{h}) = \mathbb{P}_{(X,Y)}(\hat{h}(X) \neq Y)$$

- Empirical (training)

$$\hat{R}(\hat{h}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\hat{h}(X_i) \neq Y_i)$$

- Empirical (test)

$$\tilde{R}(\hat{h}) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(\hat{h}(X_i') \neq Y_i')$$

Population and empirical classification errors

Let $\hat{h}$ be a classifier constructed from the training data

$$\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

Denote the test data as

$$\{(X_1', Y_1'), \ldots, (X_m', Y_m')\}$$

Type I error

- Population

$$R_0(\hat{h}) = \mathbb{P}_{X|(Y=0)}(\hat{h}(X) = 1)$$

- Empirical (training)

$$\hat{R}_0(\hat{h}) = \frac{\sum_{i=1}^{n} \mathbb{I}(\hat{h}(X_i) = 1)\mathbb{I}(Y_i = 0)}{\sum_{i=1}^{n} \mathbb{I}(Y_i = 0)}$$

- Empirical (test)

$$\tilde{R}_0(\hat{h}) = \frac{\sum_{i=1}^{m} \mathbb{I}(\hat{h}(X_i') = 1)\mathbb{I}(Y_i' = 0)}{\sum_{i=1}^{m} \mathbb{I}(Y_i' = 0)}$$

Population and empirical classification errors

Let $\hat{h}$ be a classifier constructed from the training data

$$\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

Denote the test data as

$$\{(X_1', Y_1'), \ldots, (X_m', Y_m')\}$$

Type II error

- Population

$$R_1(\hat{h}) = \mathbb{P}_{X|(Y=1)}(\hat{h}(X) = 0)$$

- Empirical (training)

$$\hat{R}_1(\hat{h}) = \frac{\sum_{i=1}^{n} \mathbb{I}(\hat{h}(X_i) = 0)\mathbb{I}(Y_i = 1)}{\sum_{i=1}^{n} \mathbb{I}(Y_i = 1)}$$

- Empirical (test)

$$\tilde{R}_1(\hat{h}) = \frac{\sum_{i=1}^{m} \mathbb{I}(\hat{h}(X_i') = 0)\mathbb{I}(Y_i' = 1)}{\sum_{i=1}^{m} \mathbb{I}(Y_i' = 1)}$$

How to control population type I error under α?

Suppose that a user would like to construct a classifier $\hat{h}$ with population type I error $R_0(\hat{h})$ no more than α (e.g., 0.05) with at least probability $(1-\delta)$ (e.g., 0.95). How to achieve it?

Statistical formulation

Given $\alpha, \delta \in [0,1]$, find $\hat{h}$ such that

$$\mathbb{P}\left(R_0(\hat{h}) \leq \alpha\right) > 1-\delta$$

How to control population type I error under α?

Suppose that a user would like to construct a classifier $\hat{h}$ with population type I error $R_0(\hat{h})$ no more than α (e.g., 0.05) with at least probability $(1-\delta)$ (e.g., 0.95). How to achieve it?

Statistical formulation

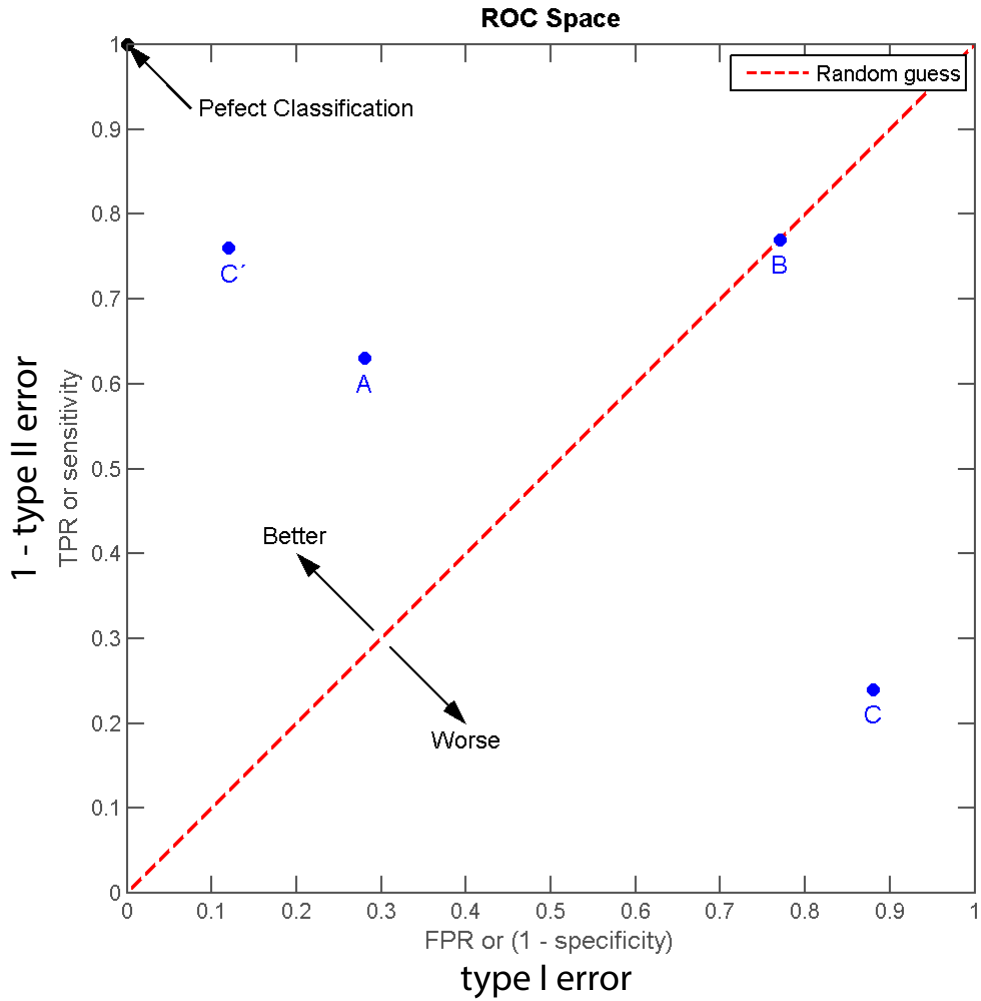
Given $\alpha, \delta \in [0,1]$, find $\hat{h}$ such that

$$\mathbb{P}\left(R_0(\hat{h}) \leq \alpha\right) > 1-\delta$$

Existing approaches for asymmetric error control

- Cost-sensitive learning (Elkan, 2001; Zadrozny et al, 2003)
  - no consensus way to assign costs
  - cannot directly control the population type I error
- Receiver operating characteristics (ROC)?

Receiver operating characteristics (ROC)



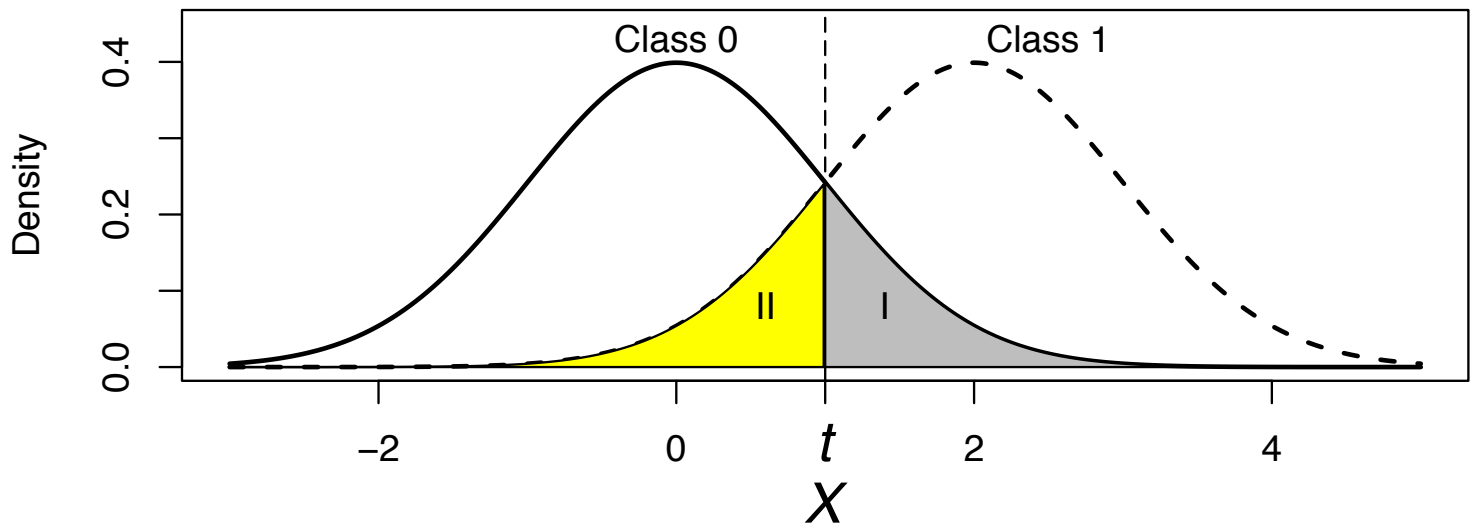
from Wikipedia

- However, ROC curves are plotted based on empirical type I and II errors in practice.

A toy example

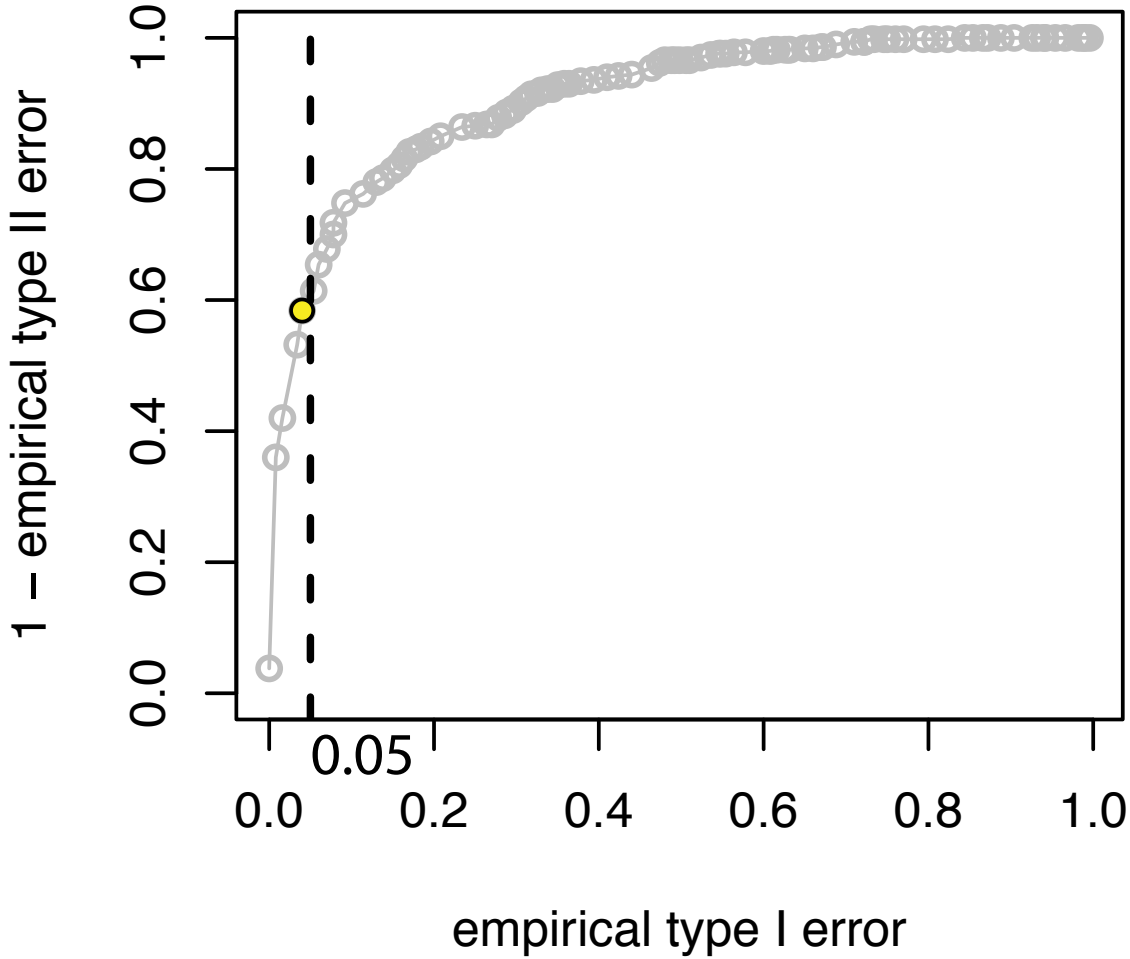
- $X|(Y=0) \sim N(0,1)$
- $X|(Y=1) \sim N(2,1)$
- $\mathbb{P}(Y=0) = 1/2$
- Classifier: $\mathbb{1}(X > t)$
- Type I error $= 1 - \Phi(t)$
- Type II error $= \Phi(t-2)$

- Training sample size $= 1,000$
- Test sample size $= 1,000$
- # of simulations $= 1,000$



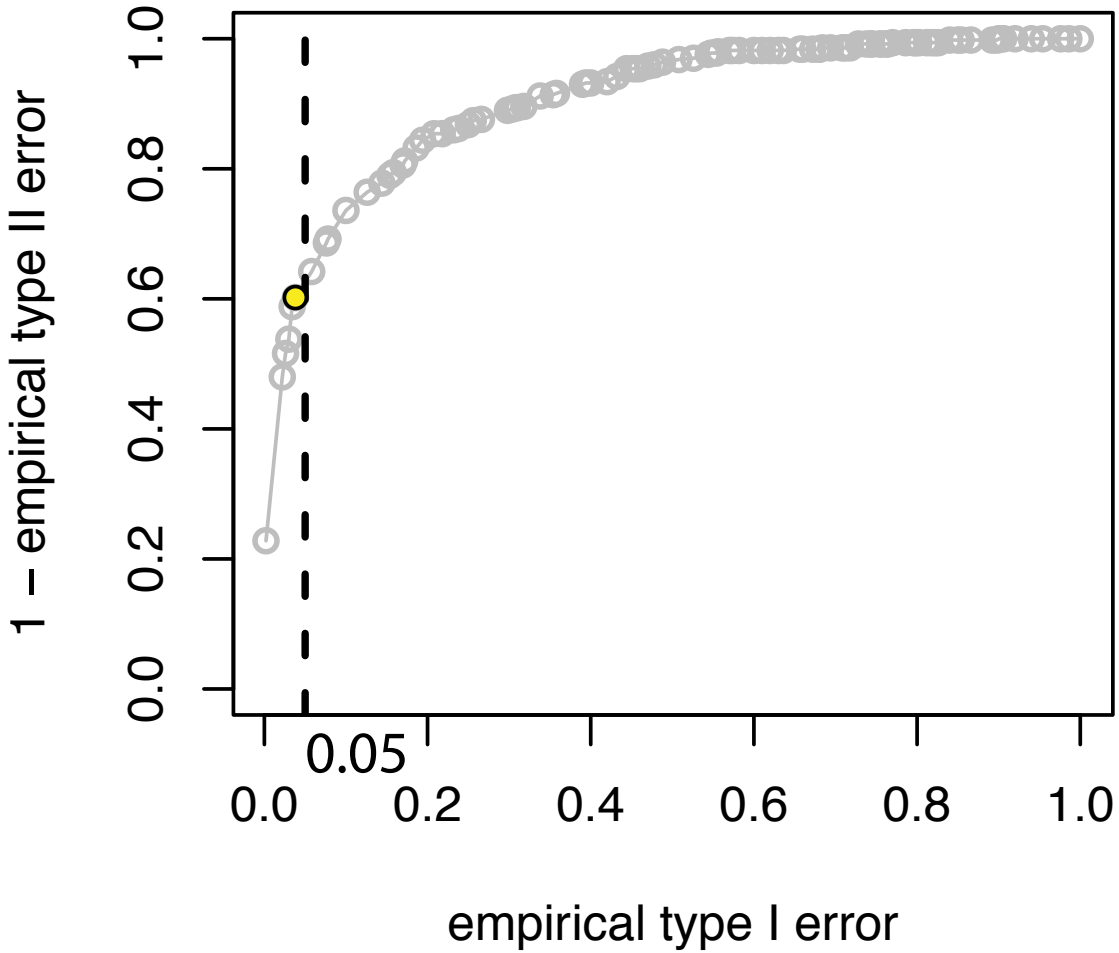
Empirical type I error control != population type I error control

Data set 1



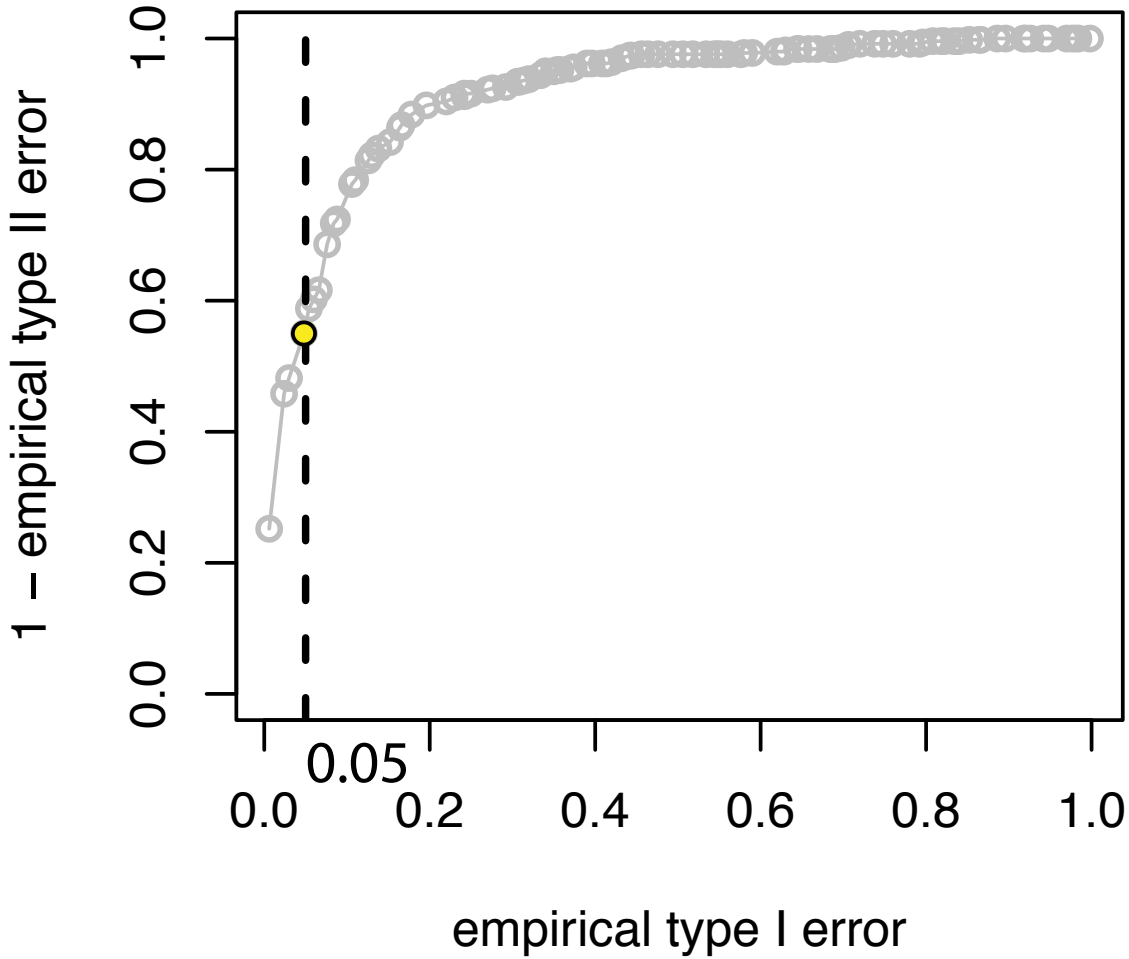
Empirical type I error control != population type I error control

Data set 2



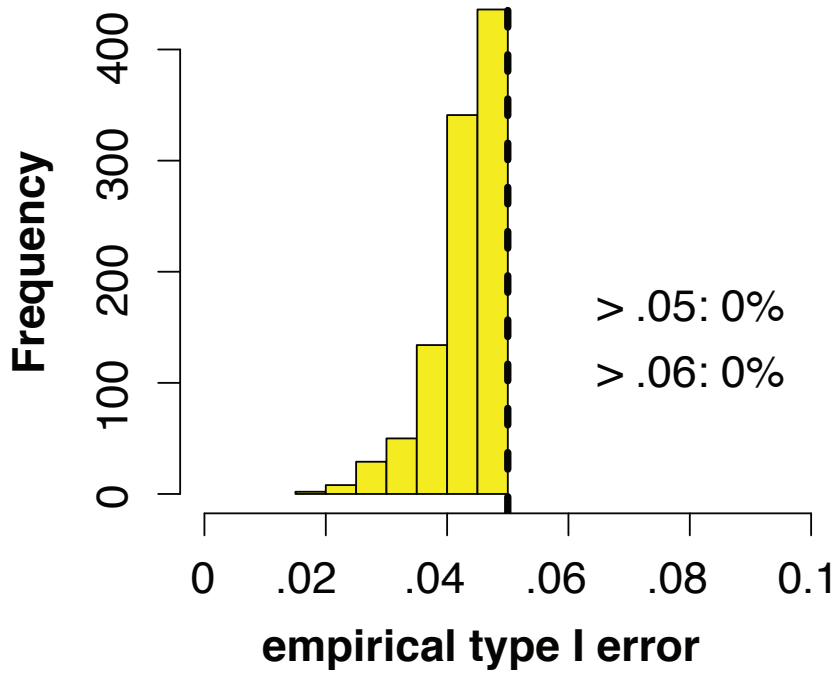
Empirical type I error control != population type I error control

Data set $1,000$



Empirical type I error control != population type I error control

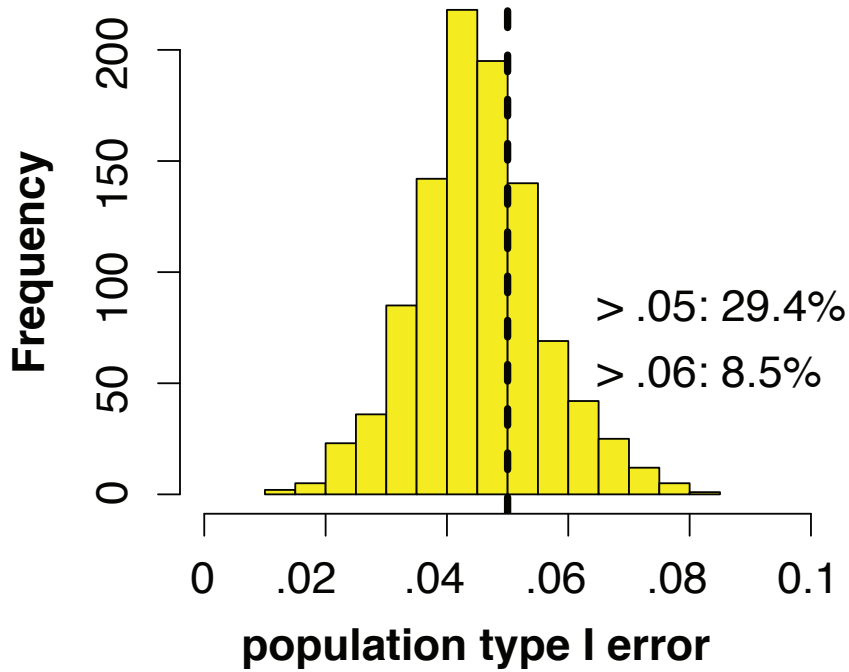
Distribution of the $1,000$ classifiers' empirical type I errors



Empirical type I error controlled? YES

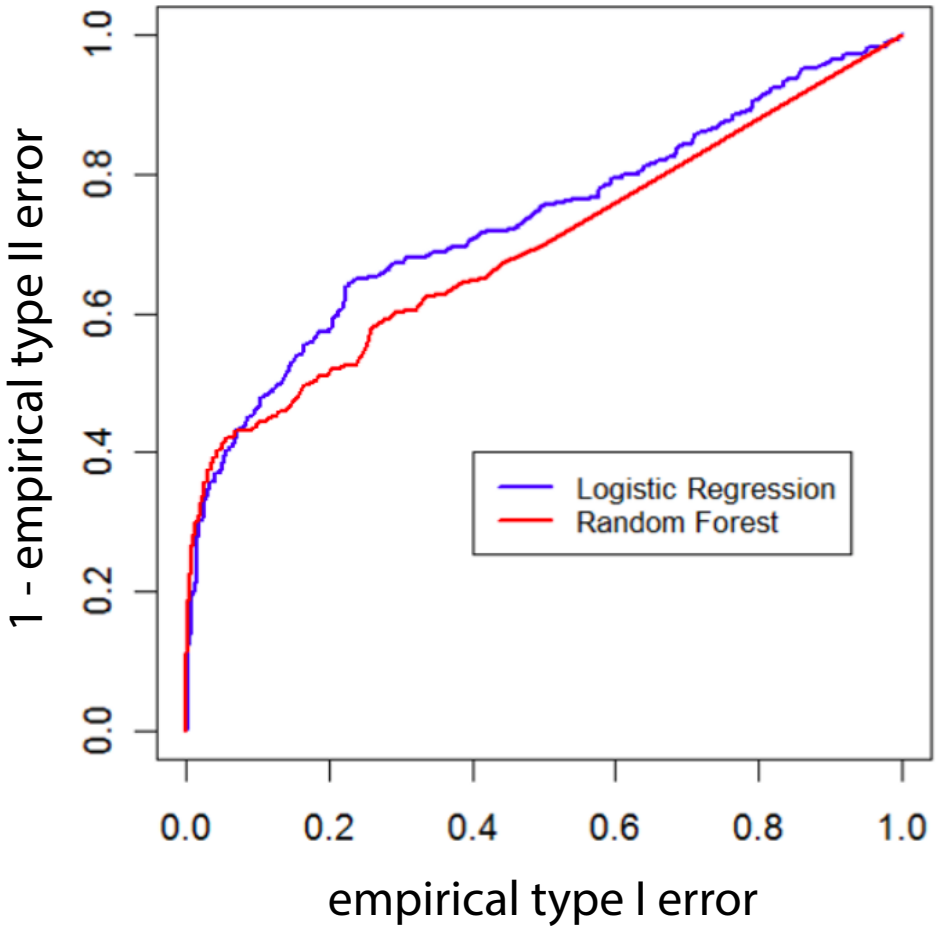
Empirical type I error control != population type I error control

Distribution of the $1,000$ classifiers' population type I errors



Population type I error controlled? NO

Choosing classifiers with population type I error control?



The empirical ROC space provides NO direct information on the population type I error control.

Neyman-Pearson Classification

The Neyman-Pearson (NP) classification paradigm

The NP paradigm seeks a classifier that satisfies:

$$\min_{R_0(h) \leq \alpha} R_1(h)\,,$$

where α is a user-specified level, usually a small value (e.g., 5%).

- Early work in the engineering community: Cannon et.al. (2002); Scott (2005)
- Our work on NP classification:
  - Rigollet and Tong (2011) (NP oracle inequalities)
  - Tong (2013) (plug-in approach)
  - Zhao et al. (2016c) (high-dimensional setting)
  - Tong et al. (2016a) (review paper)
  - Li and Tong (2016b) (genomic applications)

The Neyman-Pearson (NP) classification paradigm

Binary classification

Paradigm	Oracle classifier
Classical	$h^* = \arg\min R(h)$
Neyman-Pearson	$\phi^* = \arg\min_{R_0(\phi) \leq \alpha} R_1(\phi)$

where α reflects users' conservative attitude towards the type I error.

Neyman-Pearson (NP) oracle inequalities

- Rigollet and Tong (2011):
  Under the NP paradigm, a good classifier $\hat{\phi}$ should respect α instead of $\alpha + \epsilon$.

Neyman-Pearson (NP) oracle inequalities

- Rigollet and Tong (2011):
  Under the NP paradigm, a good classifier $\hat{\phi}$ should respect α instead of $\alpha + \epsilon$.
- **NP oracle inequalities**: two theoretical properties should be both satisfied with high probability,
  1) $R_0(\hat{\phi}) \leq \alpha$, i.e., the type I error constraint is respected
  2) $R_1(\hat{\phi}) - R_1(\phi^*) \to 0$, i.e., the excess type II error diminishes with explicit rates (w.r.t. sample size n).

Neyman-Pearson (NP) oracle inequalities

- Rigollet and Tong (2011):
  Under the NP paradigm, a good classifier $\hat{\phi}$ should respect α instead of $\alpha + \epsilon$.

- **NP oracle inequalities**: two theoretical properties should be both satisfied with high probability,
  1) $R_0(\hat{\phi}) \leq \alpha$, i.e., the type I error constraint is respected
  2) $R_1(\hat{\phi}) - R_1(\phi^*) \to 0$, i.e., the excess type II error diminishes with explicit rates (w.r.t. sample size n).

Counterpart of

- Oracle inequality of classifier $\hat{h}$ in the classical paradigm:
  $R(\hat{h}) - R(h^*) \to 0$, i.e., the excess risk diminishes with explicit rates with high probability,

  where $h^*(x) = \mathbb{1}(\eta(x) \geq 1/2)$ is the oracle (Bayes) classifier,
  in which $\eta(x) = \mathbb{E}[Y|X = x] = \mathbb{P}(Y = 1|X = x)$ is the *regression function* of Y on X.

Connection with hypothesis testing

- The Neyman-Pearson paradigm in hypothesis testing:

  Choose ϕ to maximize $\mathbb{E}[\phi(X)|Y=1]$, s.t. $\mathbb{E}[\phi(X)|Y=0] \leq \alpha$,

  where $\alpha \in (0,1)$ is the significance level of the test.

Connection with hypothesis testing

◦ The Neyman-Pearson paradigm in hypothesis testing:

Choose ϕ to maximize $\mathbb{E}[\phi(X)|Y=1]$, s.t. $\mathbb{E}[\phi(X)|Y=0] \leq \alpha$,

where $\alpha \in (0,1)$ is the significance level of the test.

Neyman-Pearson Lemma

Let $\mathbb{P}_0$ and $\mathbb{P}_1$ be probability distributions possessing densities p_0 and p_1 corresponding to Classes 0 and 1. Let C_α be a density ratio threshold such that

$$\mathbb{P}_0(p_1/p_0(X) > C_\alpha) \leq \alpha \text{ and } \mathbb{P}_0(p_1/p_0(X) \geq C_\alpha) \geq \alpha.$$

Then for a given level α, the most powerful test of level α is defined by

$$\phi^*(X) = \begin{cases} 1 & \text{if} \quad p_1/p_0(X) > C_\alpha \\ 0 & \text{if} \quad p_1/p_0(X) < C_\alpha \\ \frac{\alpha - P_0(p_1/p_0(X) > C_\alpha)}{P_0(p_1/p_0(X) = C_\alpha)} & \text{if} \quad p_1/p_0(X) = C_\alpha \end{cases}$$

An NP umbrella algorithm

Adapting popular algorithms to the NP paradigm

For scoring-type classifiers (e.g., logistic regression), a classifier is constructed from

- A scoring function $f(x)$ (e.g., $f(x) = \hat{\mathbb{P}}(Y = 1|X = x)$)
- A threshold t (e.g., $1/2$)

The classifier is

$$h(x) = \mathbb{1}(f(x) \geq t)$$

Adapting popular algorithms to the NP paradigm

For scoring-type classifiers (e.g., logistic regression), a classifier is constructed from

- A scoring function $f(x)$ (e.g., $f(x) = \hat{\mathbb{P}}(Y = 1|X = x)$)
- A threshold t (e.g., $1/2$)

The classifier is

$$h(x) = \mathbb{1}(f(x) \geq t)$$

- Under the classical paradigm, only f needs to be constructed from the training data
- Under the NP paradigm, both f and t needs to be constructed from the training data given α and δ

Adapting popular algorithms to the NP paradigm

Statistical formulation

Given $\alpha, \delta \in [0, 1]$, find $\hat{h}$ such that

$$\mathbb{P}\left(R_0(\hat{h}) > \alpha\right) \leq \delta$$

- Sample splitting: split training data into two parts

  mixed classes 0 and 1 sample $\overset{\text{base algorithm}}{\Longrightarrow}$ trained scoring function

  left-out class 0 sample $\overset{\text{scoring function}}{\Longrightarrow}$ classification scores

- Threshold search: choose the smallest threshold on the classification scores such that the violation rate (i.e., the probability that the population type I error exceeds α) $\leq \delta$.
- Order statistic: find the threshold from ordered classification scores

Theoretical foundation

Lemma 1

Let $T_1, T_2, \cdots, T_n$ be independently and identically distributed (i.i.d.) real-valued random variables following a cumulative distribution function (cdf) F. Denote by $T_{(k)}$ the k-th order statistic (i.e., $T_{(1)} \leq \cdots \leq T_{(n)}$). For any t in the domain of T_i, we have

$$\mathbb{P}\left[T_{(k)} > t\right] = \sum_{i=n-k+1}^{n} \binom{n}{i} \left[1 - F(t)\right]^i \left[F(t)\right]^{n-i} .$$

- **Remark:** Lemma 1 does not have any assumptions on F. Letting $t = F^{-1}(1-\alpha)$, we derive Lemma 2 from Lemma 1.

Theoretical foundation

Lemma 2

In the same settings as in Lemma 1,

$$\mathbb{P}\left[T_{(k)} < F^{-1}(1-\alpha)\right] \;\leq\; \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}\,,$$

where the equality holds for continuous F.

- **Remark:** Lemma 2 leads to the following proposition.

Theoretical foundation

Proposition

Applying a trained classification scoring function to the left-out class 0 sample (with size n), we denote the resulting classification scores as $T_1, \cdots, T_n$, which are real-valued random variables. Then we denote by $T_{(k)}$ the k-th order statistic (i.e., $T_{(1)} \leq \cdots \leq T_{(n)}$). For a new observation, if we denote its classification score, calculated by the trained base algorithm, as T, we can then construct a classifier $\hat{\phi}_k = \mathbb{I}(T > T_{(k)})$. Then the population type I error of $\hat{\phi}_k$ is

$$R_0(\hat{\phi}_k) = \mathbb{P}\left[T > T_{(k)} | T_{(k)}\right] = 1 - F(T_{(k)}).$$

Assuming that $T_1, \cdots, T_n$ are independently and identically distributed (i.i.d.), we have

$$\mathbb{P}\left[R_0(\hat{\phi}_k) > \alpha\right] \leq \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}.$$

Theoretical foundation

- **Remark 1:** The probability that the type I error of $\hat{\phi}_k$ exceeds α is under a constant that only depends on k and α. We call this probability $\mathbb{P}\left[R_0(\hat{\phi}_k) > \alpha\right]$ the violation rate of $\hat{\phi}_k$, and denote its upper bound by

$$v(k) = \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}.$$

  When T_i's are continuous, this bound is tight.

Theoretical foundation

- **Remark 2:** To minimize the type II error, the optimal order should be

$$k^* = \min \{k \in \{1, \cdots, n\} : v(k) \leq \delta\} \ .$$

- **Remark 3:** Minimal sample size to guarantee that $\exists k$ s.t. $v(k) \leq \delta$:

$$n \geq \log \delta / \log(1 - \alpha).$$

The NP umbrella algorithm

Algorithm An NP umbrella algorithm

input:

training data: a mixed i.i.d. sample $\mathcal{S} = \mathcal{S}^0 \cup \mathcal{S}^1$, where $\mathcal{S}^0$ and $\mathcal{S}^1$ are class 0 and class 1 samples respectively

α: type I error upper bound, $0 \leq \alpha \leq 1$; [default $\alpha = 0.05$]

δ: a small tolerance level, $0 < \delta < 1$; [default $\delta = 0.05$]

M: number of $\mathcal{S}^0$ random splits; [default $M = 1$]

function RANKTHRESHOLD(n, α, δ)

  for k in $\{1, \ldots, n\}$ **do** ▷ *for each rank threshold candidate* k

    $v(k) \leftarrow \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}$ ▷ *calculate the violation rate with threshold* k

  $k^* \leftarrow \min \{k \in \{1, \ldots, n\} : v(k) \leq \delta\}$ ▷ *pick the minimal threshold whose violation rate is under* δ

  return k^*

The NP umbrella algorithm (cont'd)

procedure NPCLASSIFIER($\mathcal{S}, \alpha, \delta, M$)

  $n = \lceil |\mathcal{S}^0|/2 \rceil$ ▷ *denote half of the size of* $|\mathcal{S}^0|$ *as* n

  $k^* \leftarrow$RANKTHRESHOLD(n, α, δ) ▷ *find the rank threshold*

  for i in $\{1, \ldots, M\}$ **do** ▷ *randomly split* $\mathcal{S}^0$ *for* M *times*

    $\mathcal{S}^0_{i,1}, \mathcal{S}^0_{i,2} \leftarrow$ random split on $\mathcal{S}^0$ ▷ *each time randomly split* $\mathcal{S}^0$ *into two halves with equal sizes*

    $\mathcal{S}_i \leftarrow \mathcal{S}^0_{i,1} \cup \mathcal{S}^1$ ▷ *combine* $\mathcal{S}^0_{i,1}$ *and* $\mathcal{S}^1$

    $\mathcal{S}^0_{i,2} = \{x_1, \ldots, x_n\}$ ▷ *write* $\mathcal{S}^0_{i,2}$ *as a set of* n *data points*

    $f_i \leftarrow$ `classification algorithm`$(\mathcal{S}_i)$ ▷ *train a classification scoring function* f_i *by inputting* $\mathcal{S}_i$ *into the classification algorithm; let* f_i *output a larger expected value for class* 1 *data*

    $\mathcal{T}_i = \{t_{i,1}, \ldots, t_{i,n}\} \leftarrow \{f_i(x_1), \ldots, f_i(x_n)\}$ ▷ *apply the scoring function* f_i *to* $\mathcal{S}^0_{i,2}$ *to obtain a set of score threshold candidates*

    $\{t_{i,(1)}, \ldots, t_{i,(n)}\} \leftarrow sort(\mathcal{T}_i)$ ▷ *sort elements of* $\mathcal{T}_i$ *in an increasing order*

    $t_i^* \leftarrow t_{i,(k^*)}$ ▷ *find the score threshold corresponding to the chosen rank threshold* k^*

    $\phi_i(X) = \mathbb{1}\left(f_i(X) > t_i^*\right)$ ▷ *construct an NP classifier based on the scoring function* f_i *and the threshold* t_i^*

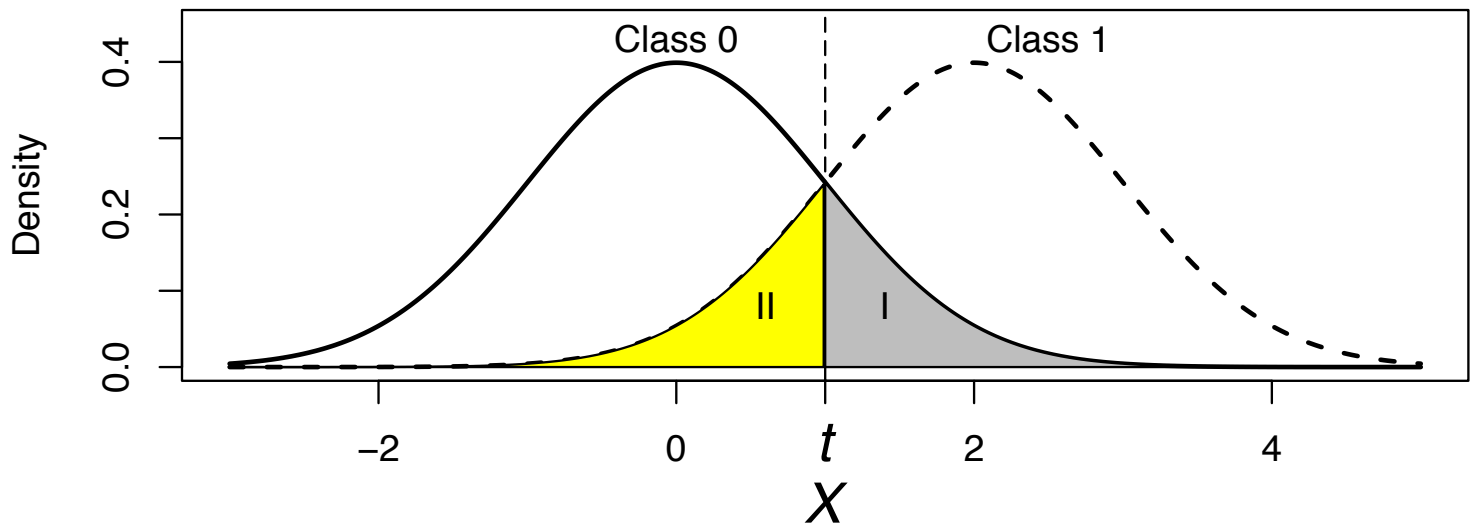
output:

  an ensemble NP classifier $\phi(X) = \mathbb{1}\left(\frac{1}{M}\sum_{i=1}^{M}\phi_i(X) \geq 1/2\right)$ ▷ *by majority vote*

Back to the toy example

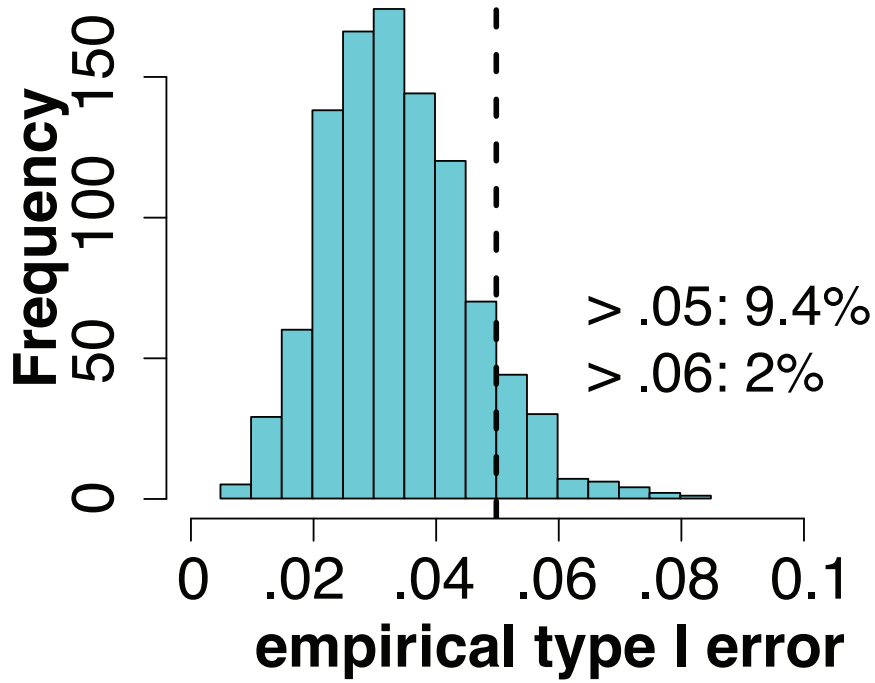
- $X|Y=0 \sim N(0,1)$
- $X|Y=1 \sim N(2,1)$
- $\mathbb{P}(Y=0)=1/2$
- Classifier: $\mathbb{1}(X>t)$
- Type I error $=1-\Phi(t)$
- Type II error $=\Phi(t-2)$

- Training sample size $=1,000$
- Test sample size $=1,000$
- # of simulations $=1,000$



NP Classifier in the toy example

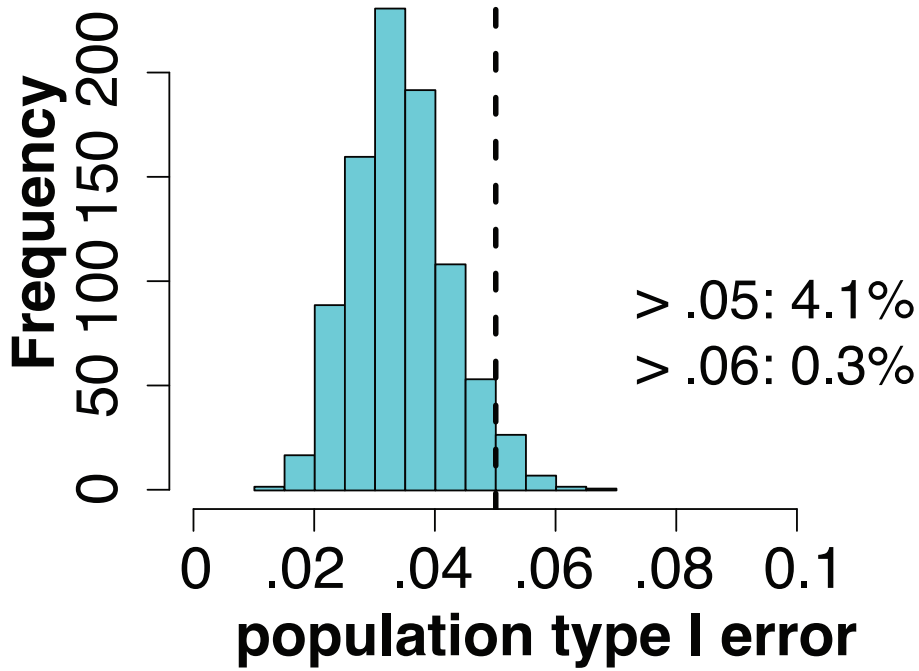
Distribution of the $1,000$ classifiers' empirical type I errors



Empirical type I error controlled? NO/YES

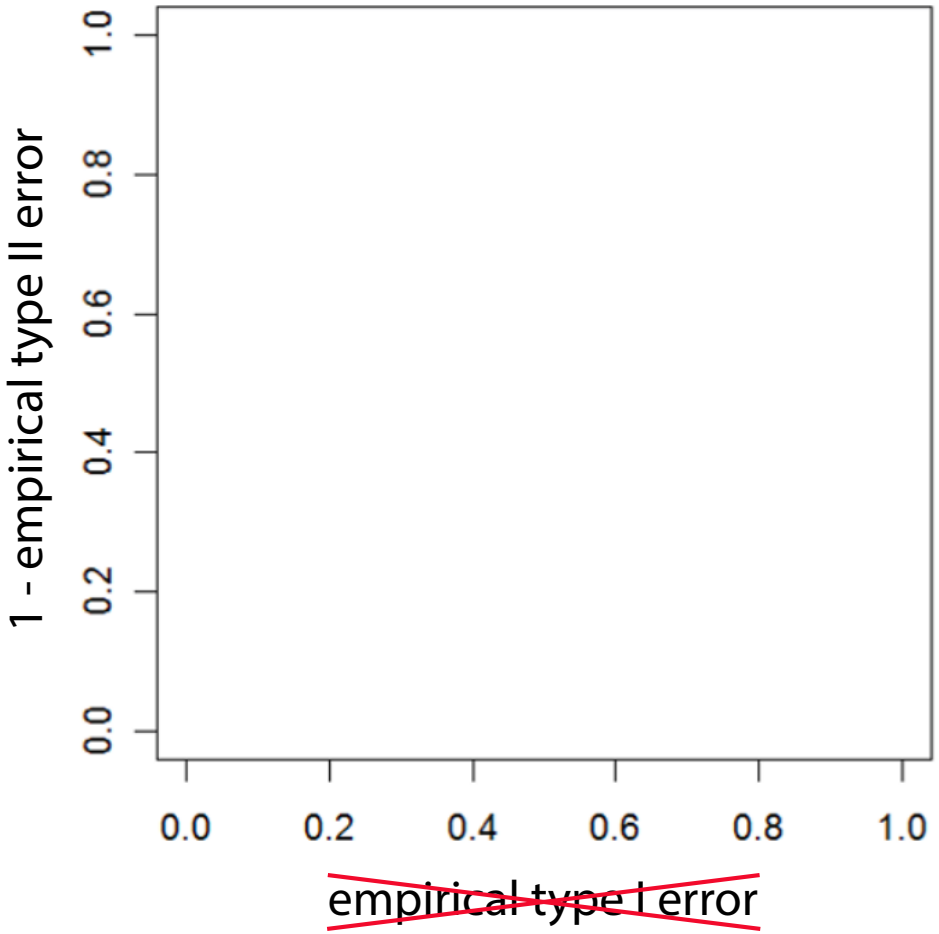
NP Classifier in the toy example

Distribution of the $1,000$ classifiers' population type I errors



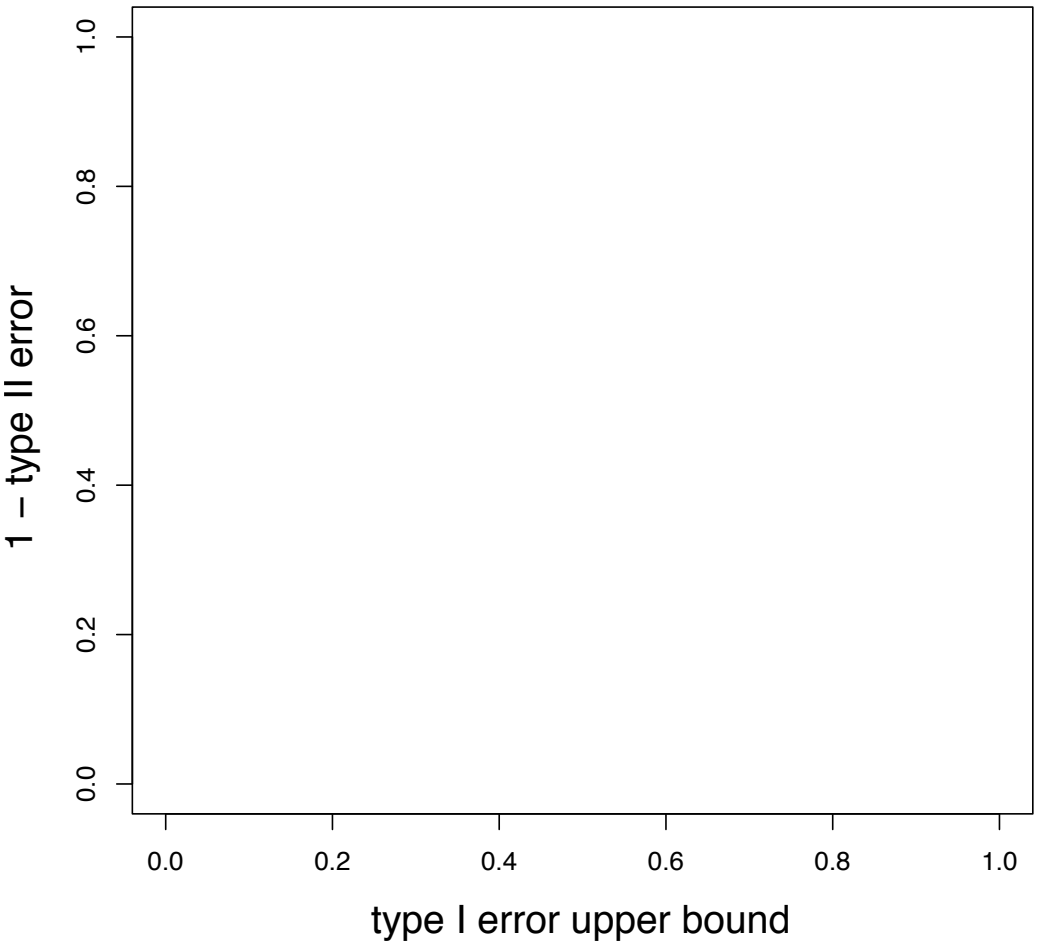
Population type I error controlled? YES

Choosing classifiers with population type I error control?



The empirical ROC space provides NO direct information on the population type I error control.

Choosing classifiers with population type I error control?



NP-ROC

$\Leftarrow$ NP classification methods

NP-ROC Band

Question: now, we have a high-probability type I error control, how about type II error? For this, we would need a left-out sample for class 1

Revised sample splitting:

- (1) mixed classed 0 and 1 sample $\overset{\text{algorithm}}{\Longrightarrow}$ trained model
- (2) left-out class 0 sample $\overset{\text{model}}{\Longrightarrow}$ classification scores
- (3) left-out class 1 sample $\overset{\text{evaluation}}{\Longrightarrow}$ type II error lower and upper bounds

Remark: sometimes we need to pay a price to know how well we will do in the future

Illustration of the band construction process

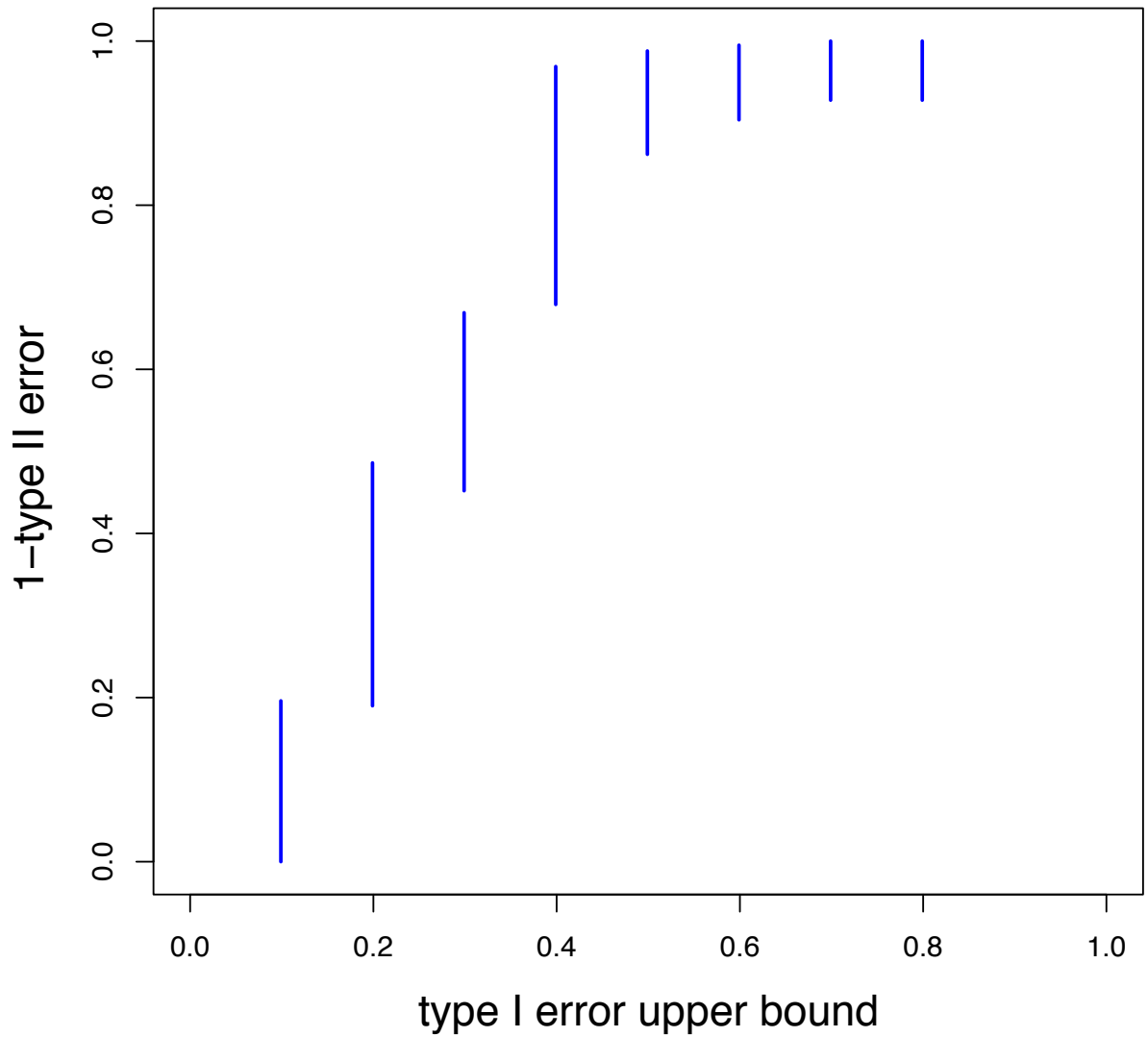
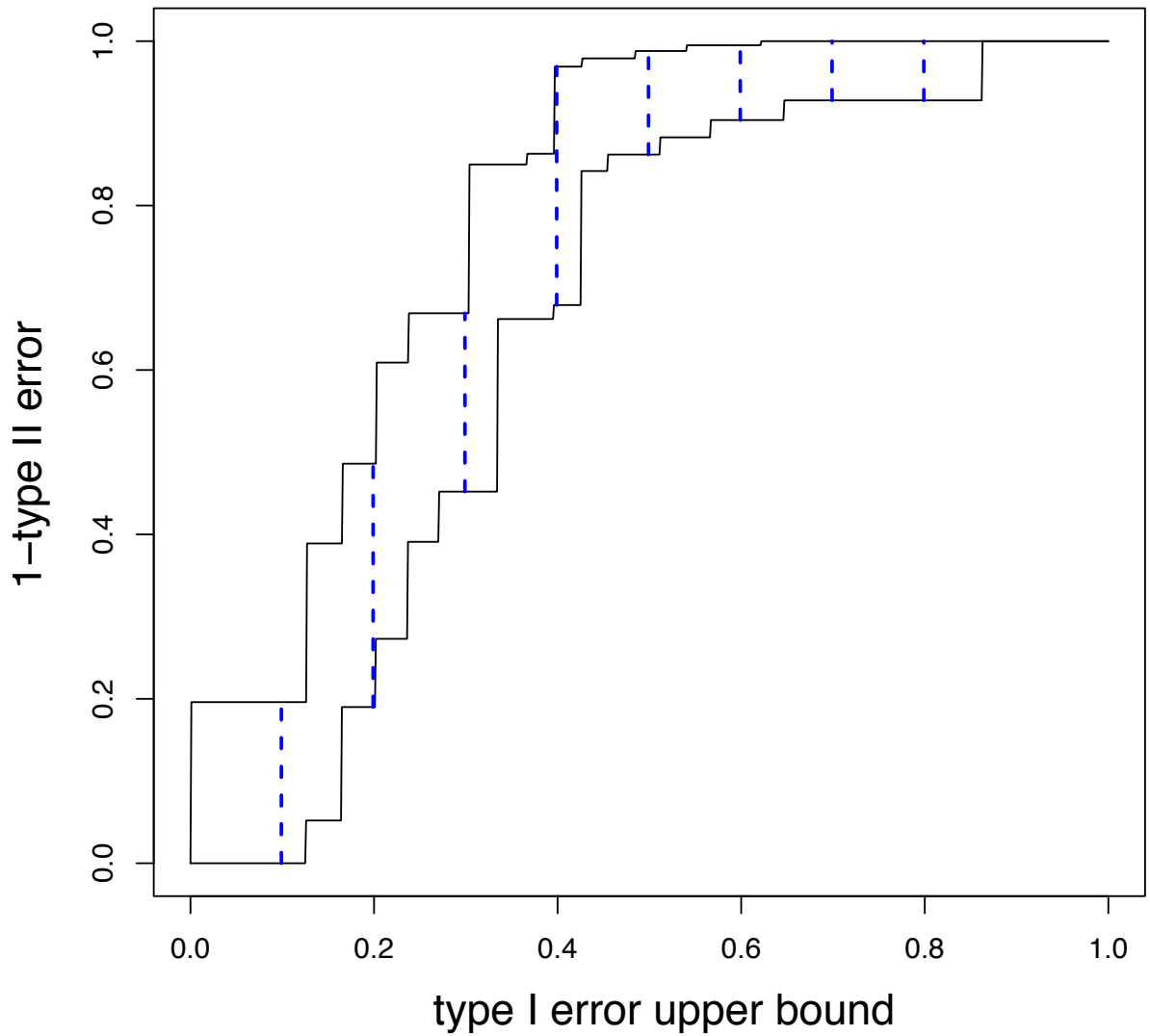
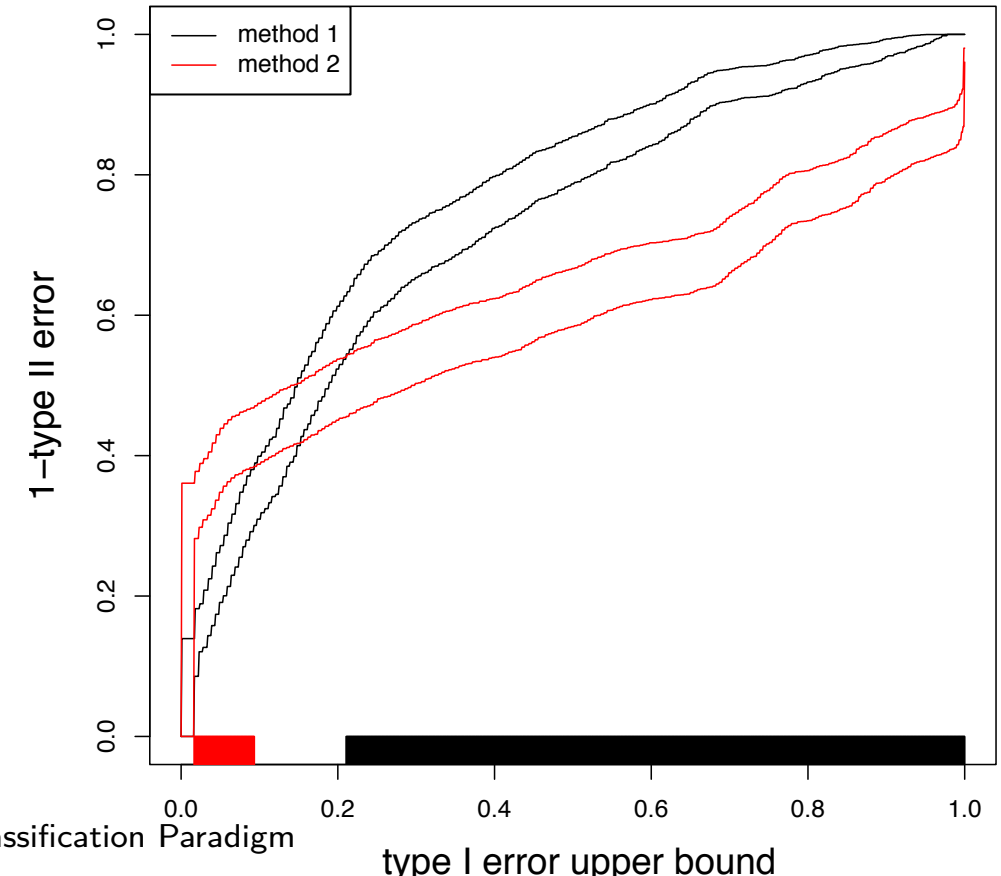


Illustration of the band construction process



Comparison of two NP-ROC bands

$(X_1|Y=0) \sim \mathcal{N}(0,1)$, $(X_1|Y=1) \sim \mathcal{N}(1,1)$, $(X_2|Y=0) \sim \mathcal{N}(0,1)$ and $(X_2|Y=1) \sim \mathcal{N}(1,6)$ with $\mathbb{P}(Y=0) = \mathbb{P}(Y=1) = 0.5$.
$n = 1,000$, $\delta = 0.1$.
We would like to investigate the Linear Discriminant Analysis that uses only X_1 (referred to as f_1) or only X_2 (referred to as f_2)



Ongoing work: feature ranking under the NP paradigm

Motivation

In automated disease diagnosis using gene expression data

- Which genes or gene sets are of the highest diagnostic power?
- Do they also play key roles in disease development and progression?

Misdiagnosing a cancer patient as healthy is much more severe than misdiagnosing a healthy patient with cancer!

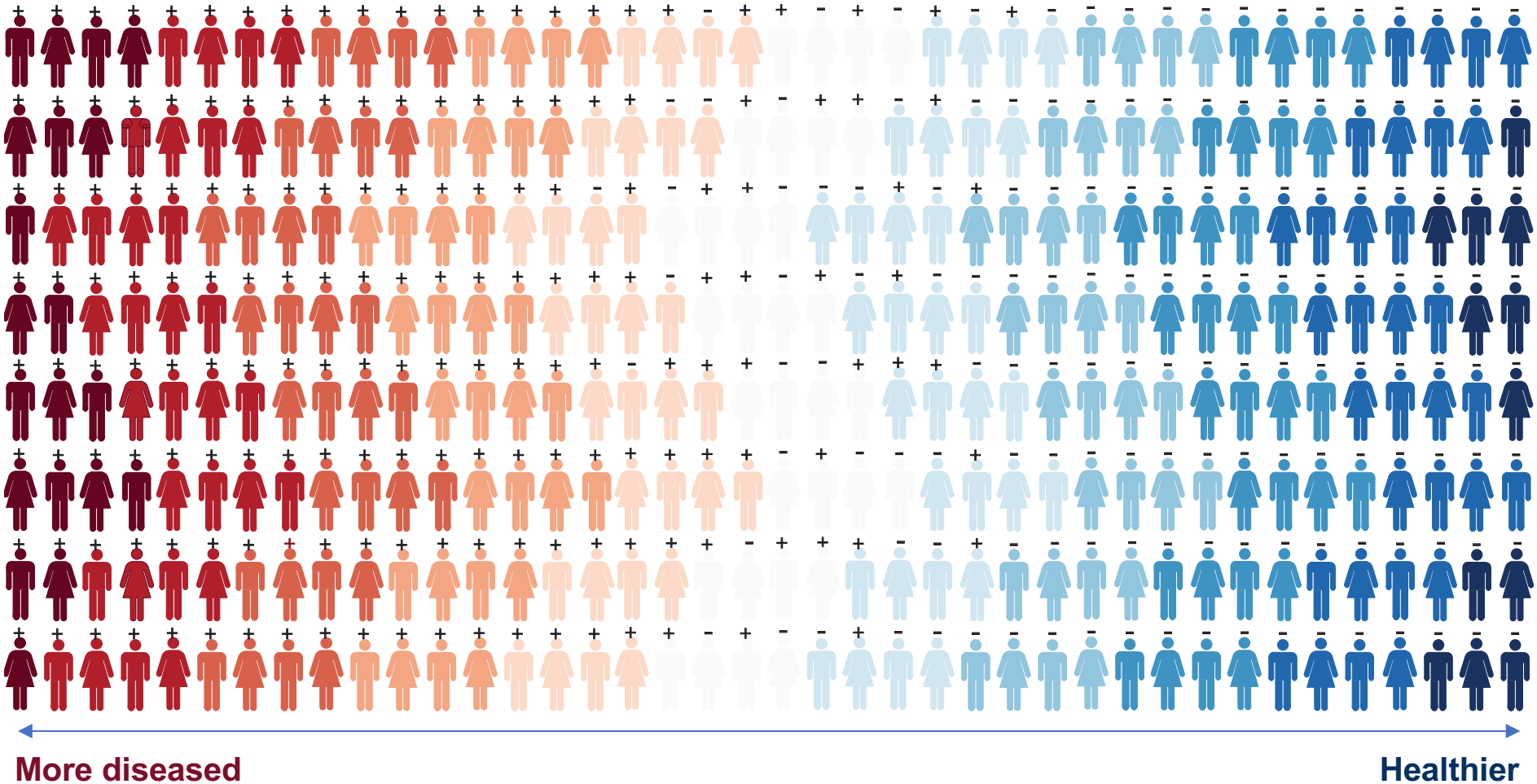
- Gene ranks obtained by minimizing the overall error rate is NOT a proper goal
- A more reasonable approach:
  rank genes by the less severe error rate while controlling the more severe error rate under some user-defined threshold α, usually a small value such as 5%

Summary

- Binary classification
  - Key question: where is the randomness?
- NP classification
  - Control of type I error with high probability
  - NP umbrella algorithm
- NP-ROC
  - NP-ROC bands can be used to compare two classifiers
  - Help choose α adaptively

Controlling the population error is important

The population

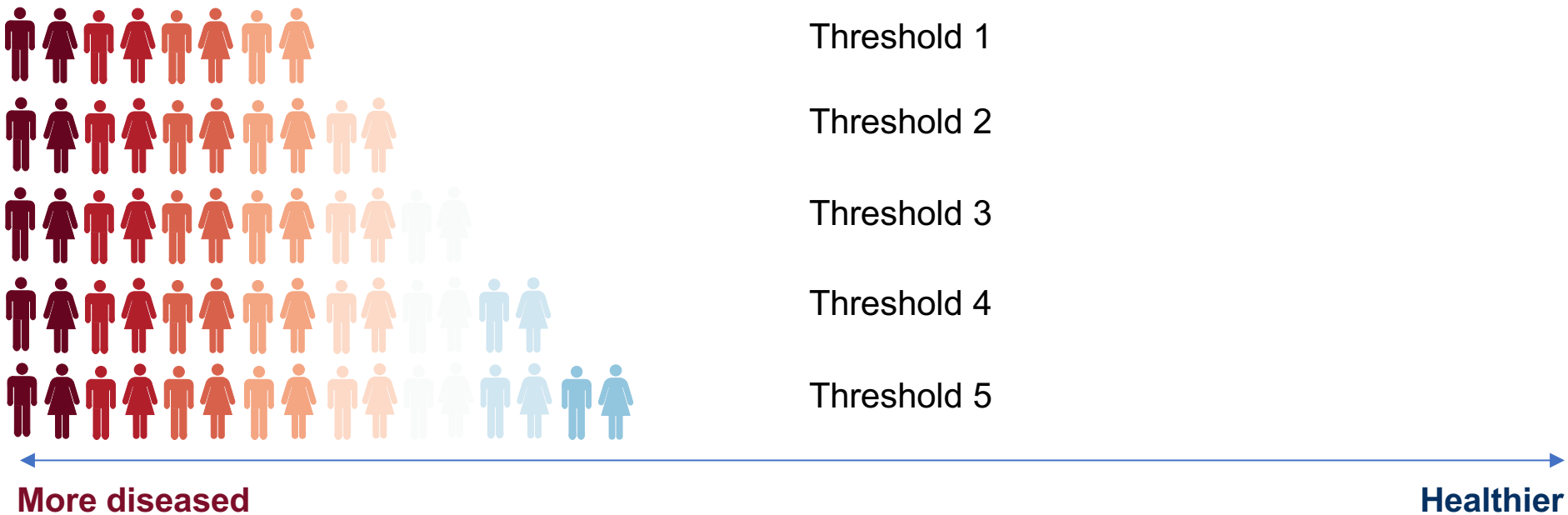


https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

Where to set the threshold?

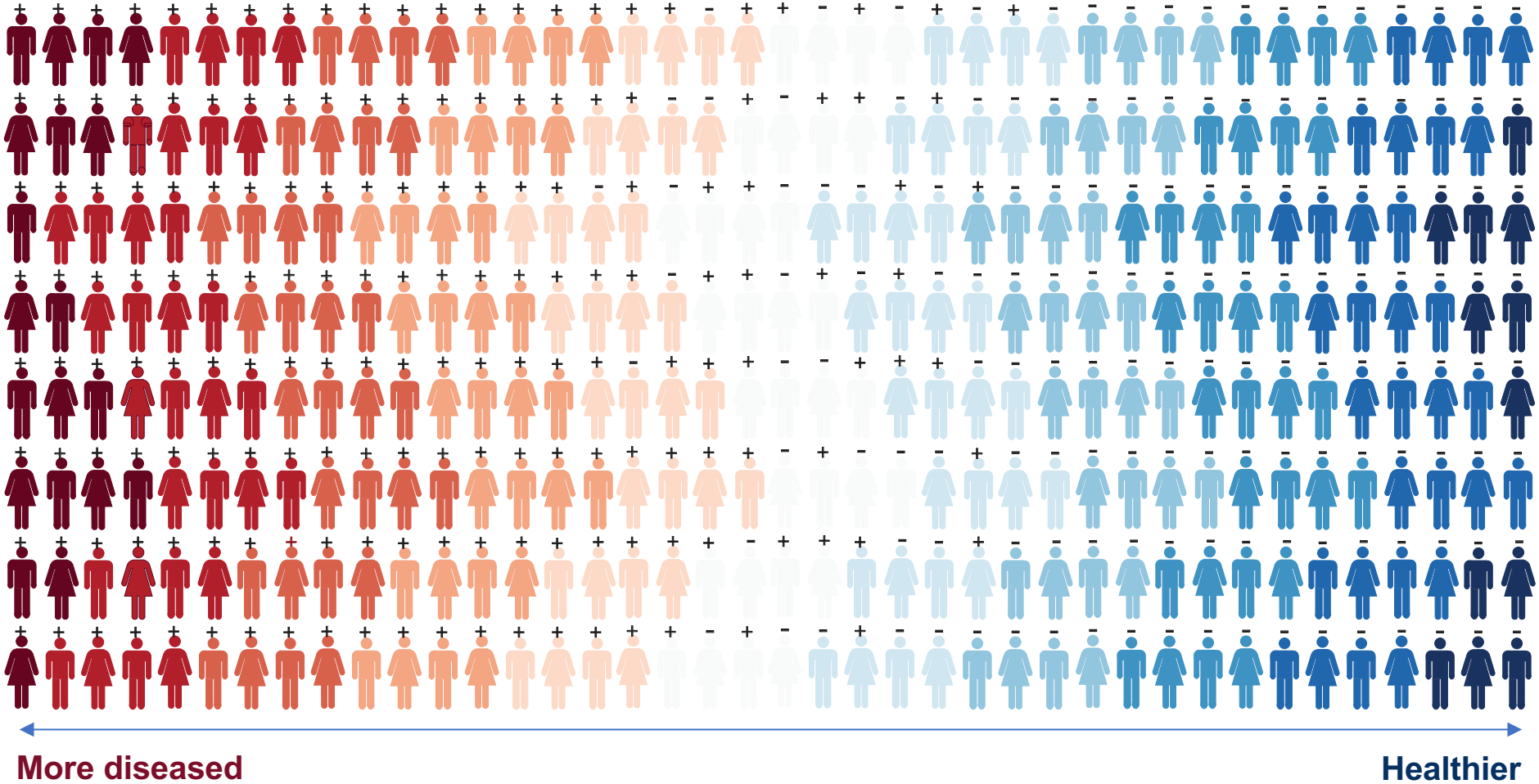
Question:
If doctors would like to control the false negative rate under 1% while minimizing the false positive rate, what should be the diagnosis threshold?



https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

Setting the threshold is not difficult when observing the population



https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

Setting the threshold is not difficult when observing the population

Threshold 1

Threshold 2

Threshold 3

Threshold 4 ⭐

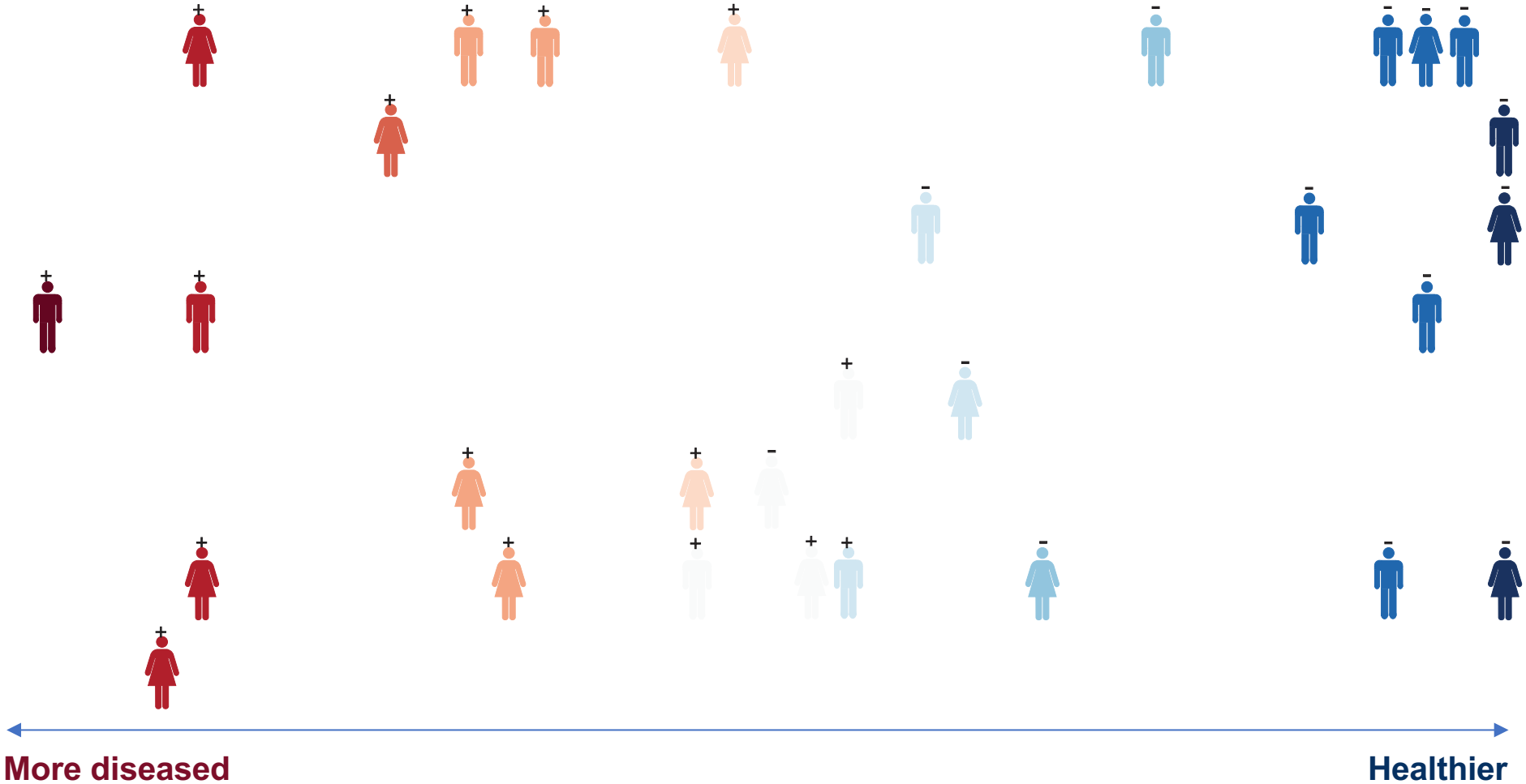
Threshold 5

More diseased — **Healthier**

https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

However in reality, we only observe a random sample from the population



https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

Problem arises if we set the threshold such that the false negative rate (FNR) on a random sample equals 1%

Threshold 1

Threshold 2 ⟶ FNR on the population = 7% 😭

Threshold 3

Threshold 4

Threshold 5

More diseased — **Healthier**

https://youtu.be/vbeeHqJ_PHY

Controlling the population error is important

As we observe more and more random samples...

How likely will we choose each threshold?

Threshold 1 (5%)

Threshold 2 (15%)

Threshold 3 (30%)

Threshold 4 ⭐ (35%)

Threshold 5 (15%)

About half the chance we will choose a threshold with FNR on the population greater than 1% 😭

More diseased ◄——————————————————► **Healthier**

https://youtu.be/vbeeHqJ_PHY

Paper and Software

SCIENCE ADVANCES | RESEARCH ARTICLE

RESEARCH METHODS

Neyman-Pearson classification algorithms and NP receiver operating characteristics

Xin Tong,[1*†] Yang Feng,[2†] Jingyi Jessica Li[3*]

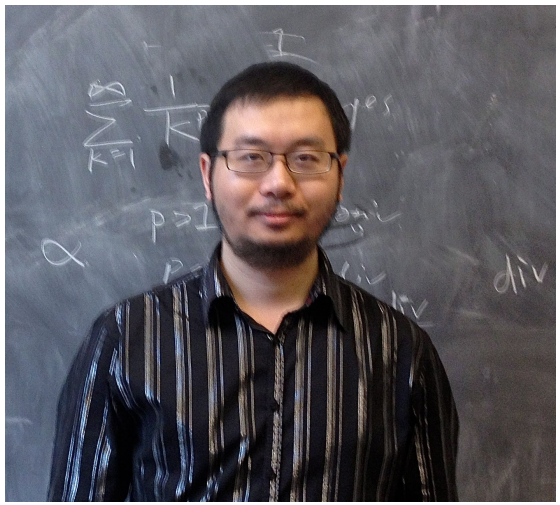
In many binary classification applications, such as disease diagnosis and spam detection, practitioners commonly face the need to limit type I error (that is, the conditional probability of misclassifying a class 0 observation as class 1) so that it remains below a desired threshold. To address this need, the Neyman-Pearson (NP) classification paradigm is a natural choice; it minimizes type II error (that is, the conditional probability of misclassifying a class 1 observation as class 0) while enforcing an upper bound, α, on the type I error. Despite its century-long history in hypothesis testing, the NP paradigm has not been well recognized and implemented in classification schemes. Common practices that directly limit the empirical type I error to no more than α do not satisfy the type I error control objective because the resulting classifiers are likely to have type I errors much larger than α, and the NP paradigm has not been properly implemented in practice. We develop the first umbrella algorithm that implements the NP paradigm for all scoring-type classification methods, such as logistic regression, support vector machines, and random forests. Powered by this algorithm, we propose a novel graphical tool for NP classification methods: NP receiver operating characteristic (NP-ROC) bands motivated by the popular ROC curves. NP-ROC bands will help choose α in a data-adaptive way and compare different NP classifiers. We demonstrate the use and properties of the NP umbrella algorithm and NP-ROC bands, available in the R package nproc, through simulation and real data studies.

R package `nproc`

https://CRAN.R-project.org/package=nproc

Email: jli@stat.ucla.edu

Acknowledgements



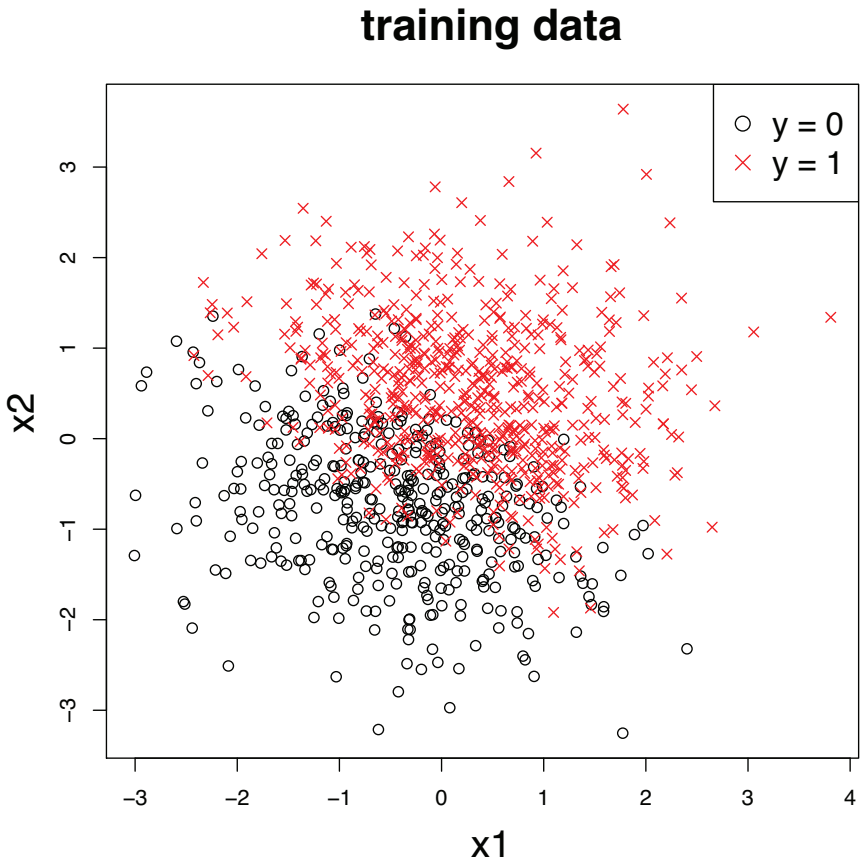
Xin Tong
(USC)

Yang Feng
(Columbia)

Yiling Chen
(UCLA)

R package `nproc` and simulation examples

- Package nproc on CRAN:
  ```
  > library(nproc)
  ```
- Toy data dat simulated from a logistic model
  $\mathbb{P}(Y = 1) = 1/\left(1 + \exp\{-(1 + 2X_1 + 3X_2)\}\right)$, X_1, $X_2 \sim N(0, 1)$.



Logistic regression

- We first train a logistic regression model on the training data under the classical paradigm.

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

Logistic regression

- We first train a logistic regression model on the training data under the classical paradigm.

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

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

```
> summary(lr_model1_err)
Min.     1st Qu.   Median   Mean     3rd Qu.   Max.
0.1290   0.1723    0.1846   0.1849   0.1980    0.2368
```

Logistic regression

- We first train a logistic regression model on the training data under the classical paradigm.

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

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

```
> 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 $\alpha = 0.05$.

```
> lr_model2 <- npc(x=x,y=y,method='logistic',alpha=.05)
```

Logistic regression

- We first train a logistic regression model on the training data under the classical paradigm.

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

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

```
> 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 $\alpha = 0.05$.

```
> lr_model2 <- npc(x=x,y=y,method='logistic',alpha=.05)
```

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

```
> summary(lr_model2_err)
Min.       1st Qu.    Median     Mean       3rd Qu.    Max.
0.009732   0.027830   0.033960   0.034400   0.040880   0.065040
```