Optimization of the Accuracy and Calibration of Binary and Multiclass Pattern Recognizers, for Wide Ranges of Applications

Niko Brümmer

Spescom DataVoice

~~Optimization of the Accuracy and Calibration of Binary and Multiclass Pattern Recognizers, for Wide Ranges of Applications~~

Niko Brümmer

Spescom DataVoice

SPESCOM

El Ingenioso Hidalgo
Don Quijote de la Ciudad del Cabo

A very long story in two parts, in which we will violently charge some old windmills and gently re-calibrate some others.

Part I: Binary Pattern Recognition

Part II: Multiclass Pattern Recognition

Don Quijote de la Ciudad del Cabo

Contents

Part I

1. **Introduction**
2. Good old error-rate
3. Binary case:

   Error-Rate $\rightarrow$ ROC $\rightarrow$ C_{det} $\rightarrow$ C_{llr}

Part II: Multiclass

Introduction

1. What do we mean by *pattern recognition?*
2. What is our *goal* with the methodology discussed in this talk.
3. Why the emphasis on *evaluation*?

1. What do we mean by *pattern recognition*?

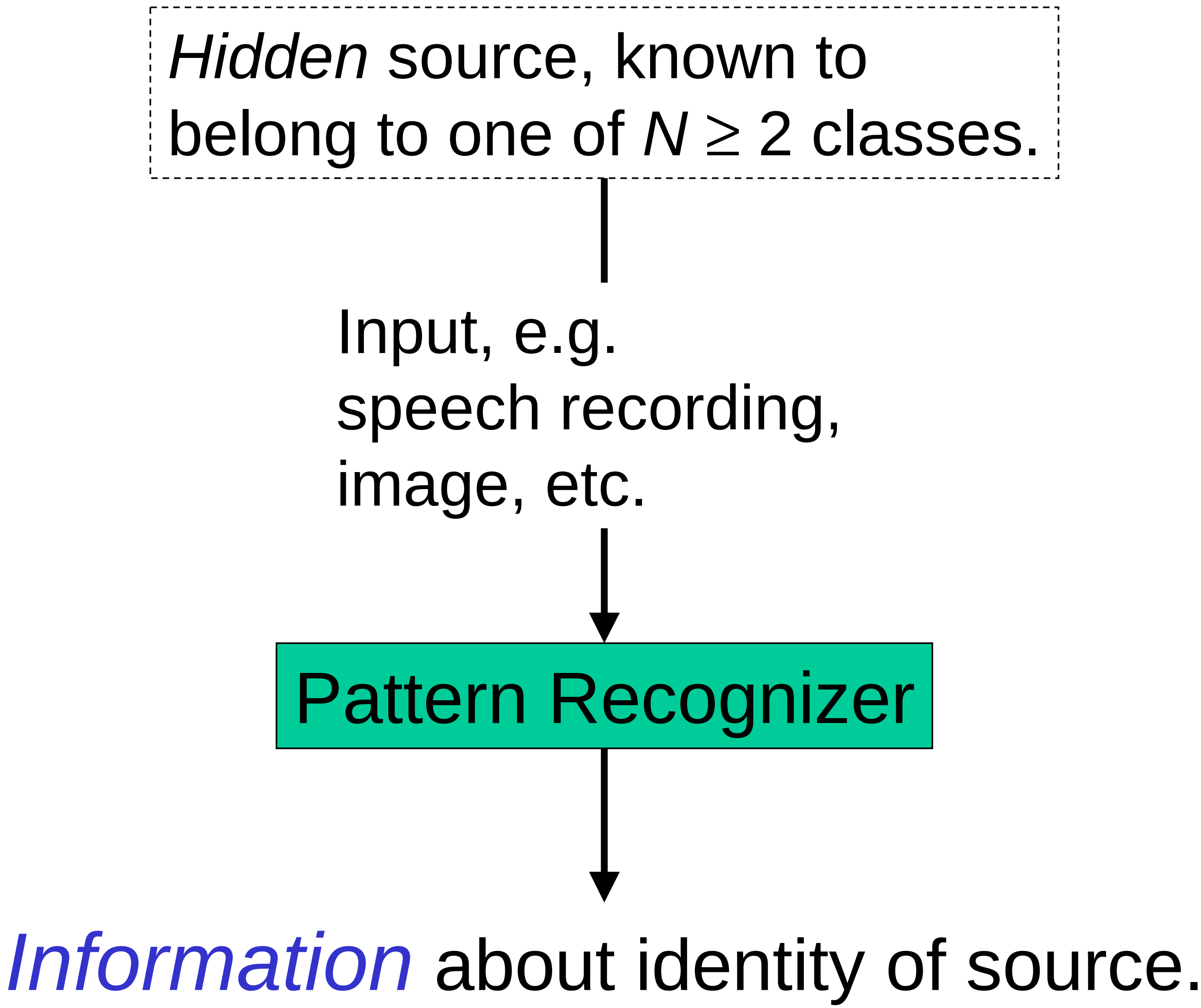
Hidden source, known to belong to one of $N \geq 2$ classes.

Input, e.g.
speech recording,
image, etc.

Pattern Recognizer

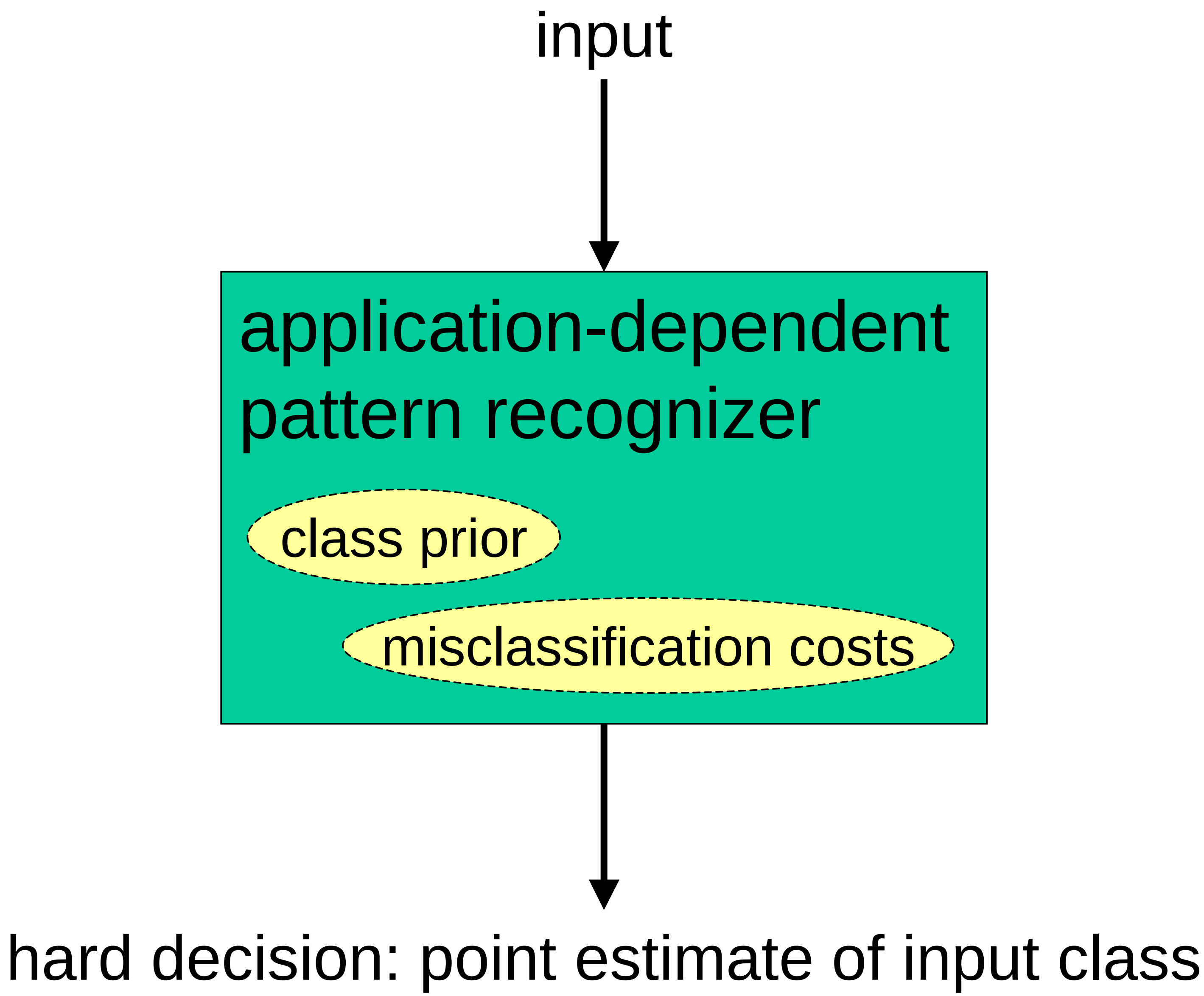
Estimated class of input.

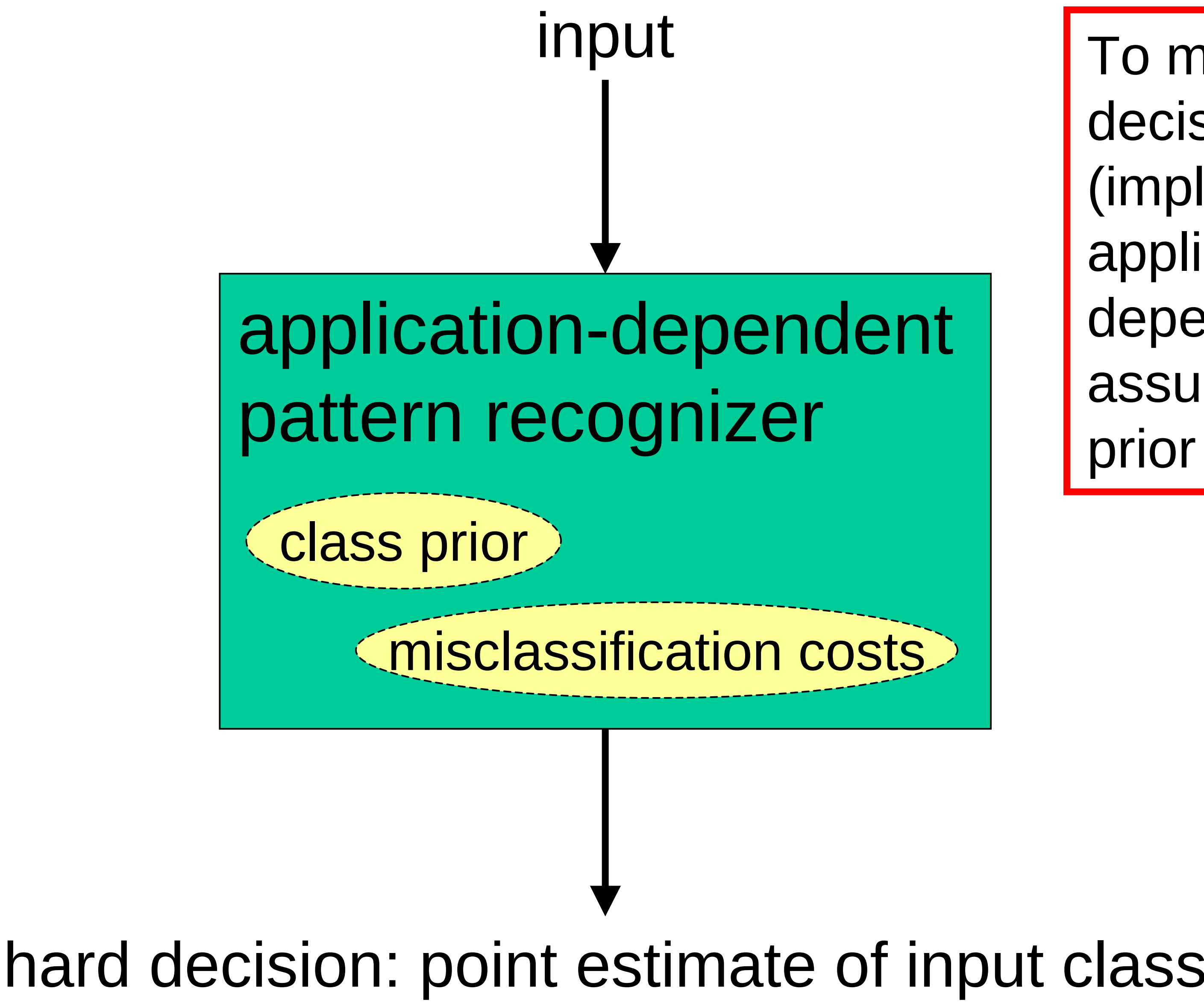
… or more general and potentially more useful:


Hidden source, known to belong to one of $N \geq 2$ classes.
Input, e.g. speech recording, image, etc.
Pattern Recognizer
Information about identity of source.

2. Goal

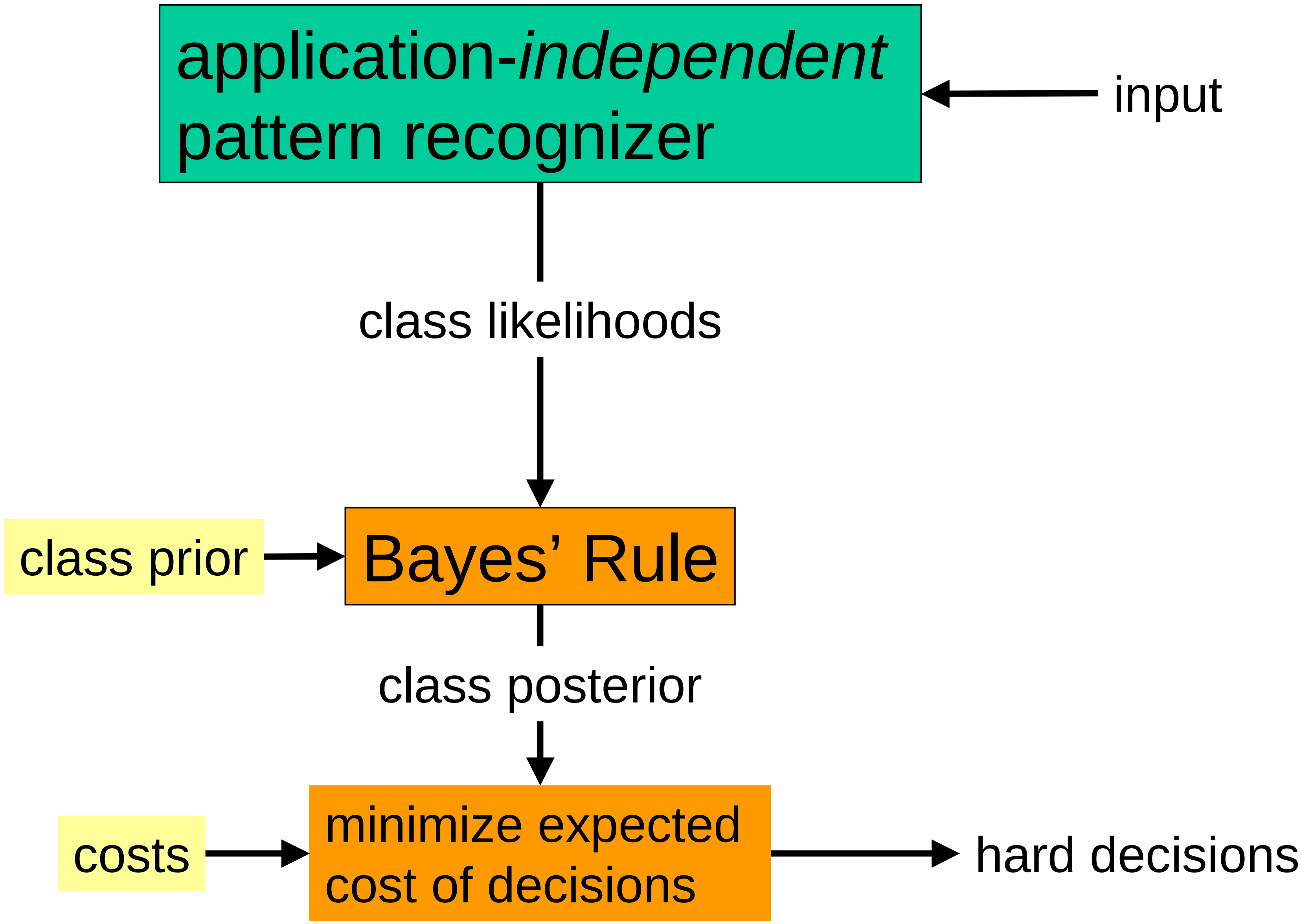
To create accurate, well-calibrated and application-independent pattern recognizers.

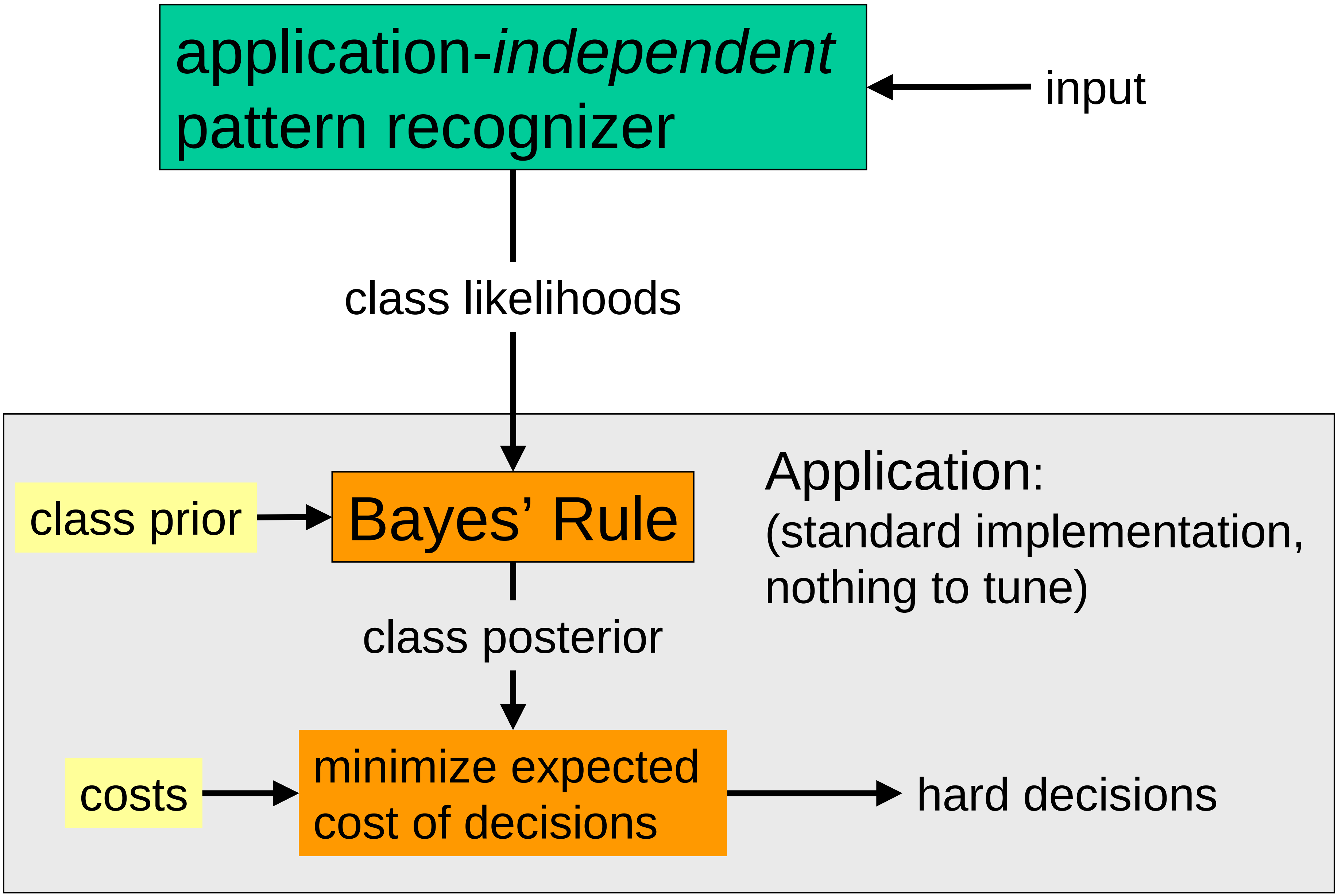

input
application-dependent pattern recognizer
class prior
misclassification costs
hard decision: point estimate of input class

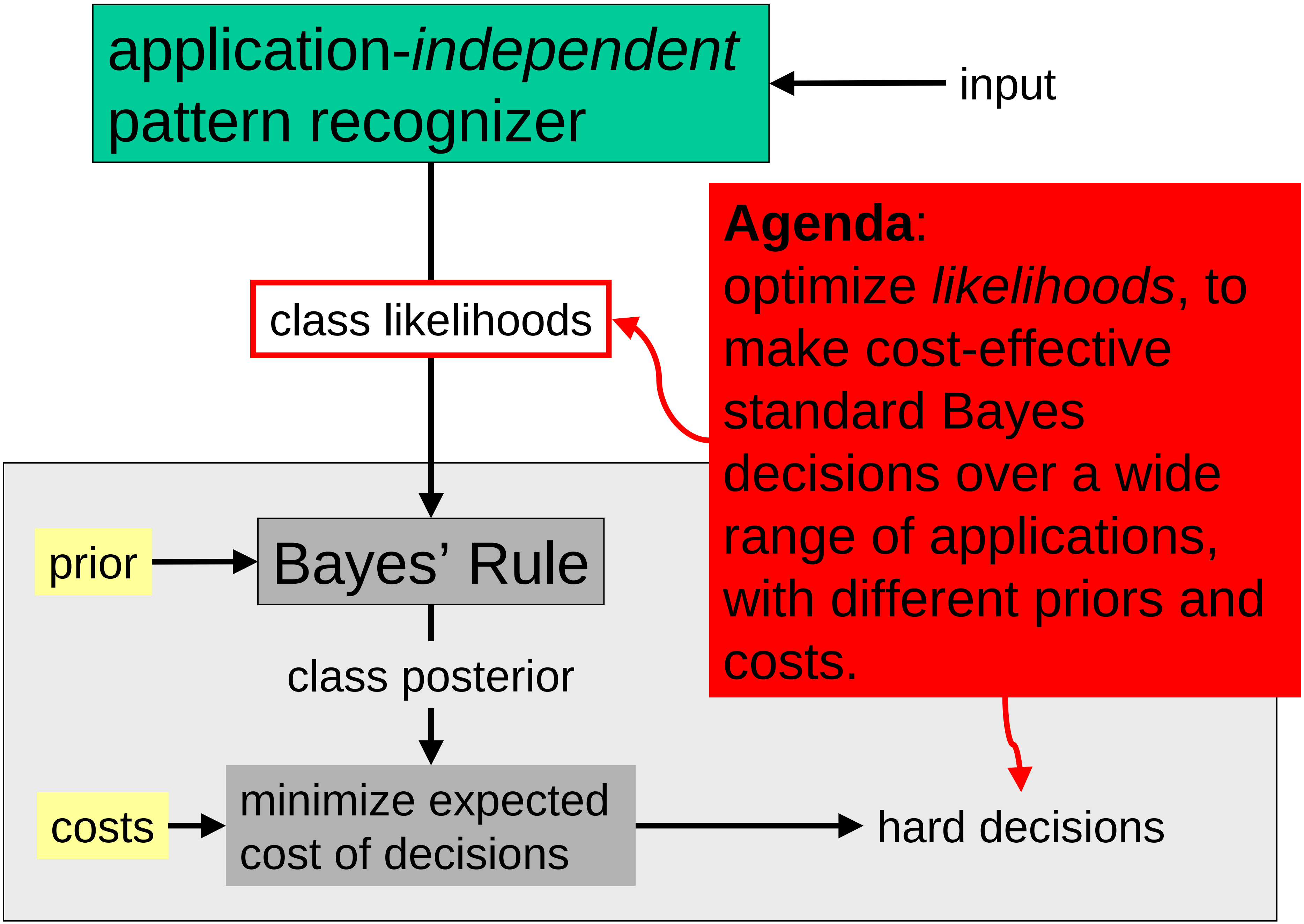


To make hard decisions, you need (implicit) application-dependent assumptions about prior and costs.

A principled approach to factoring out the role of priors and costs is the *Bayes decision* framework:


application-*independent*
pattern recognizer
input
class likelihoods
class prior
Bayes' Rule
class posterior
costs
minimize expected
cost of decisions
hard decisions


application-*independent*
pattern recognizer
input
class likelihoods
Application:
(standard implementation,
nothing to tune)
class prior
Bayes' Rule
class posterior
costs
minimize expected
cost of decisions
hard decisions


application-*independent*
pattern recognizer
input
class likelihoods
Agenda:
optimize *likelihoods*, to make cost-effective standard Bayes decisions over a wide range of applications, with different priors and costs.
prior
Bayes' Rule
class posterior
costs
minimize expected cost of decisions
hard decisions

3. Evaluation and Optimization

Before we can optimize performance, we need to be able to *evaluate* performance!

- Much of this talk will concentrate on *evaluation*.
- It is *very* useful if the evaluation criterion can also be used as numerical optimization objective function.

Part I

1. Introduction
2. **Good old error-rate**
3. Binary case:

   Error-Rate $\rightarrow$ ROC $\rightarrow$ C_{det} $\rightarrow$ C_{llr}

Traditional evaluation by error-rate

1. Use supervised evaluation database with input patterns of *N* classes.
2. Recognizer makes hard decisions,
   - i.e. *point* estimates of input class.
3. Evaluator counts misclassification errors:
   - average error-rate
   - class-conditional errors (confusion matrix)

Advantages of error-rate:

- Intuitive, easy to understand.
- Easy to compute.

These advantages are very important! We don't want to lose them. So we generalize error-rate to create new evaluation criteria which remain easy to understand and to compute.

Disadvantages of error-rate

- Application dependent, assumes:
  - fixed, *equal costs* for all types of misclassifications.
  - *class priors* are *fixed* and equal to relative proportions in evaluation database.
- Forces recognizer to make hard decisions:
  - Hard decisions are non-invertible and therefore *lose information*.
  - Recognizer can be applied only to that one fixed recognition task.

Disadvantages of error-rate

- Average error-rate tends to increase with perplexity (number of classes, *N*).
  - results difficult to compare for different *N*.
  - results look pessimistic for large *N*.

Disadvantages of error-rate

- Poor objective function for numerical optimization in discriminative training:
  - Not differentiable.
  - Even if approximated with smooth differentiable function, tends to lead to non-convex optimization problems.
  - Vulnerable to over-training.

Part I

1. Introduction
2. Good old error-rate
3. **Binary case:**

   Error-Rate $\rightarrow$ ROC $\rightarrow$ C_{det} $\rightarrow$ C_{llr}

How to fix the disadvantages of error-rate?
Binary recognizer case

Previous approaches

We summarize these to:

- Appreciate their advantages and disadvantages, and to
- Review some terminology that we will need later.

Previous approaches

1. **ROC / DET- curves**
2. Detection Cost Function: C_{det}
   – used in NIST Speaker/Language Recognition Evaluations.

ROC / DET-curves

- ROC = *Receiver Operating Curve*

- DET = *Detection-Error-Tradeoff* curve (equivalent to ROC, but with specially warped axes).

ROC / DET

- ROC/DET works best for *binary* classification problems.
  - Several kinds of ROC analysis for multiclass pattern recognition have been proposed, but it remains an open problem …

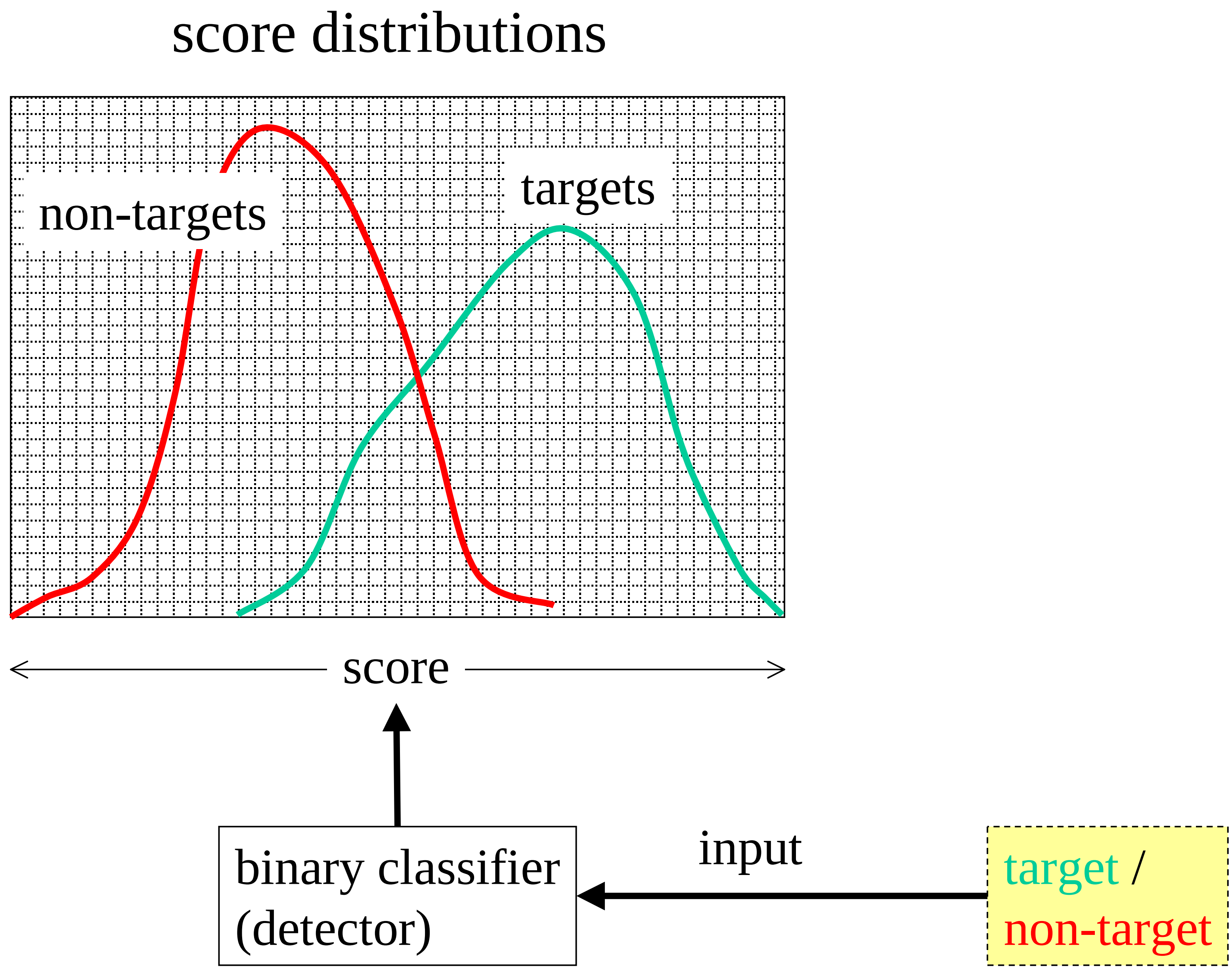
ROC / DET

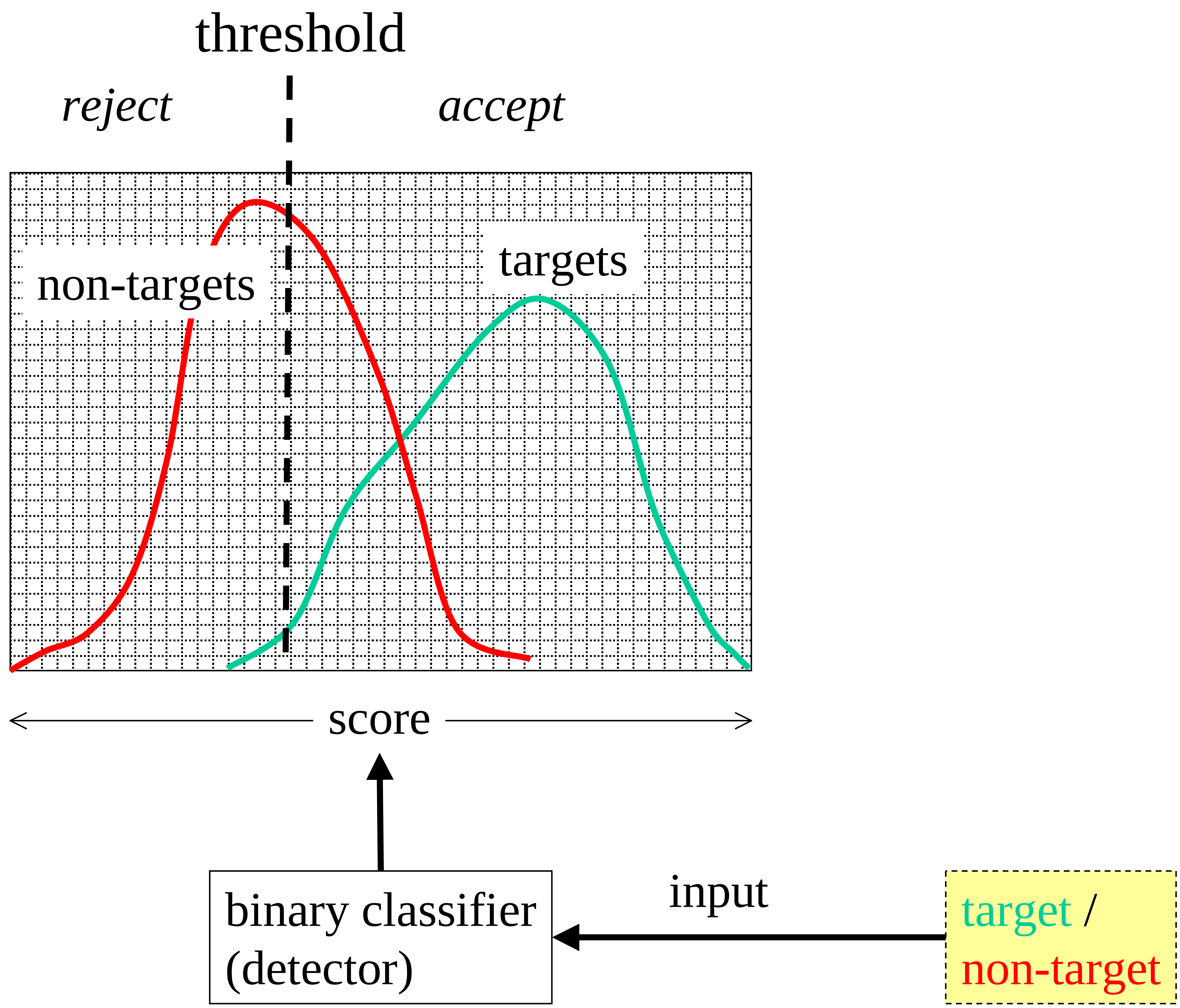
- ✓ This analysis attains *independence of prior and costs* by requiring *soft decisions* from recognizer.
- ✗ but it *ignores calibration.*
  - Does *not* test ability to set decisions thresholds.
  - *Cannot* be used as sole evaluation criterion in applications where hard decisions need to be made.

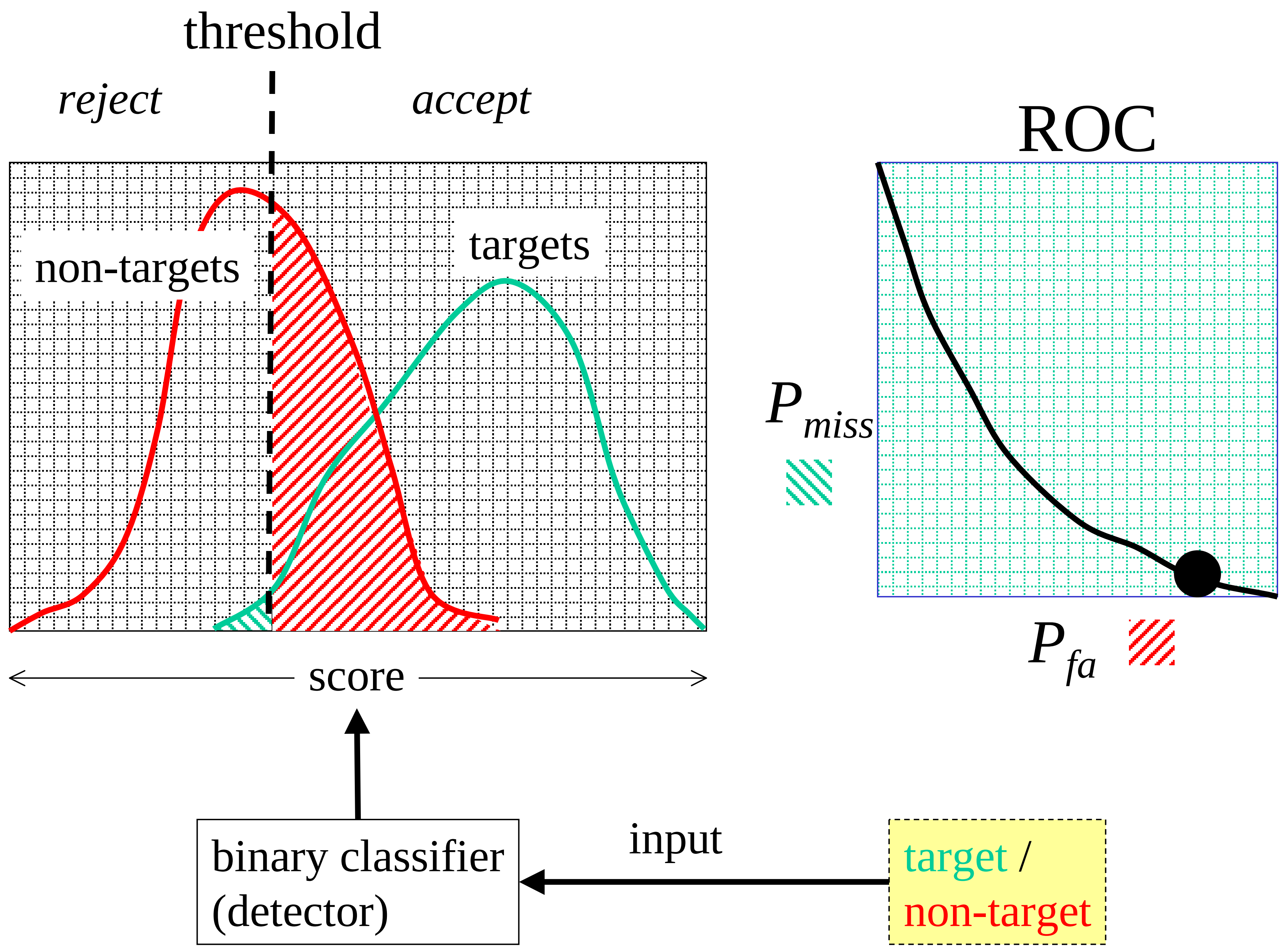
ROC: 5-minute tutorial

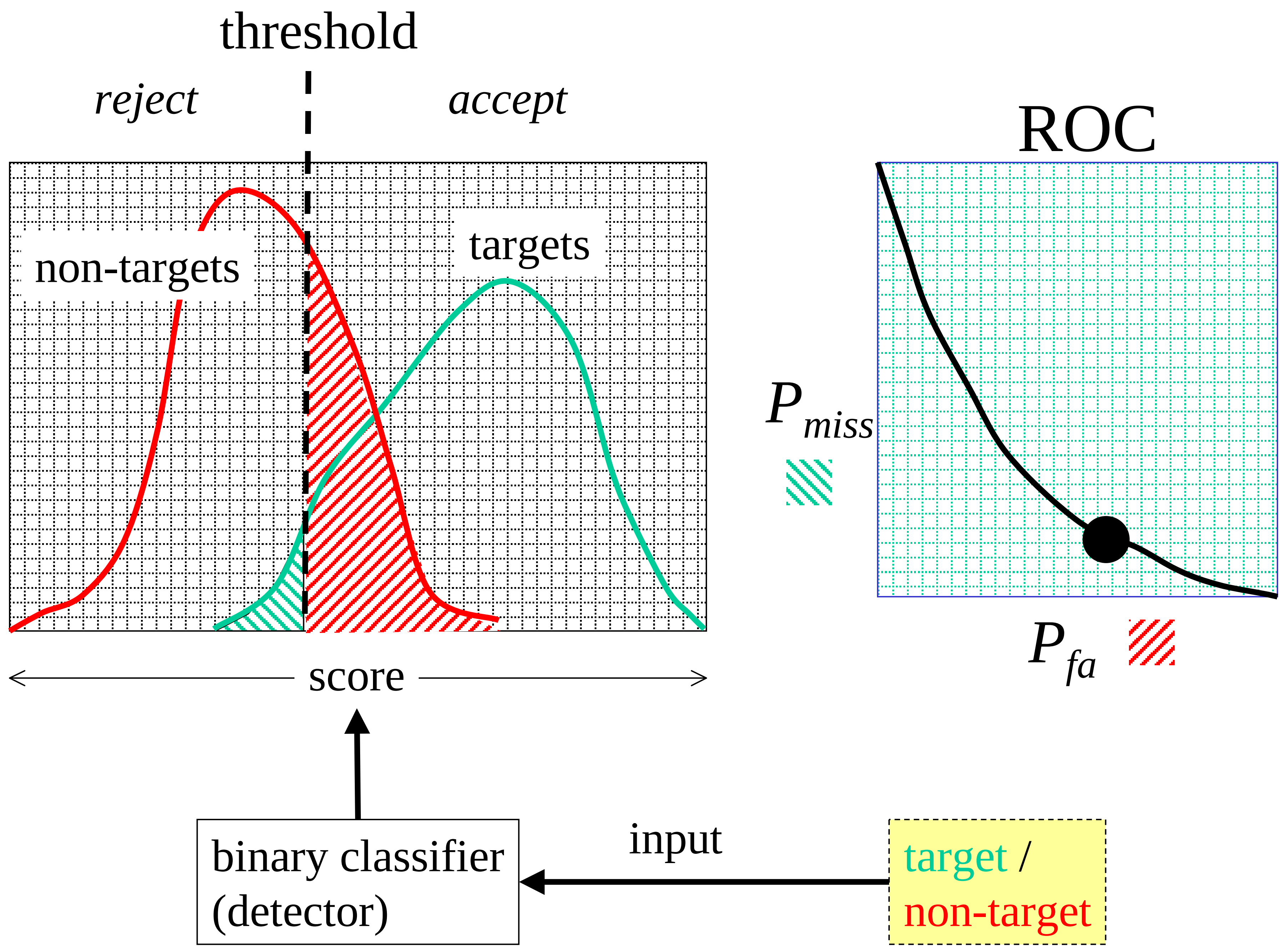
Binary misclassification errors (detection terminology)

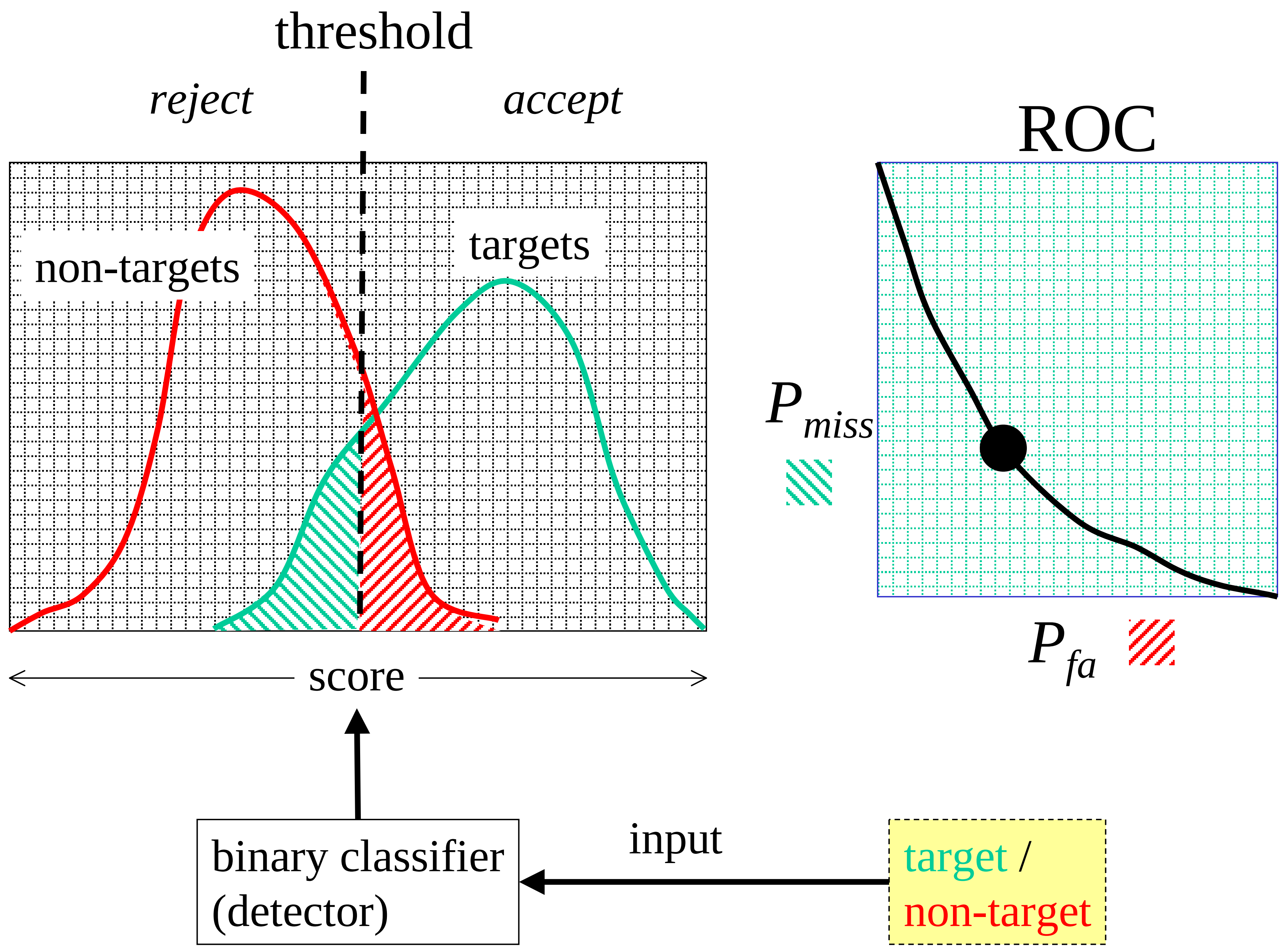
		decisions	
		accept	**reject**
classes	**target**	✓	*miss*
	non-target	*false-alarm*	✓

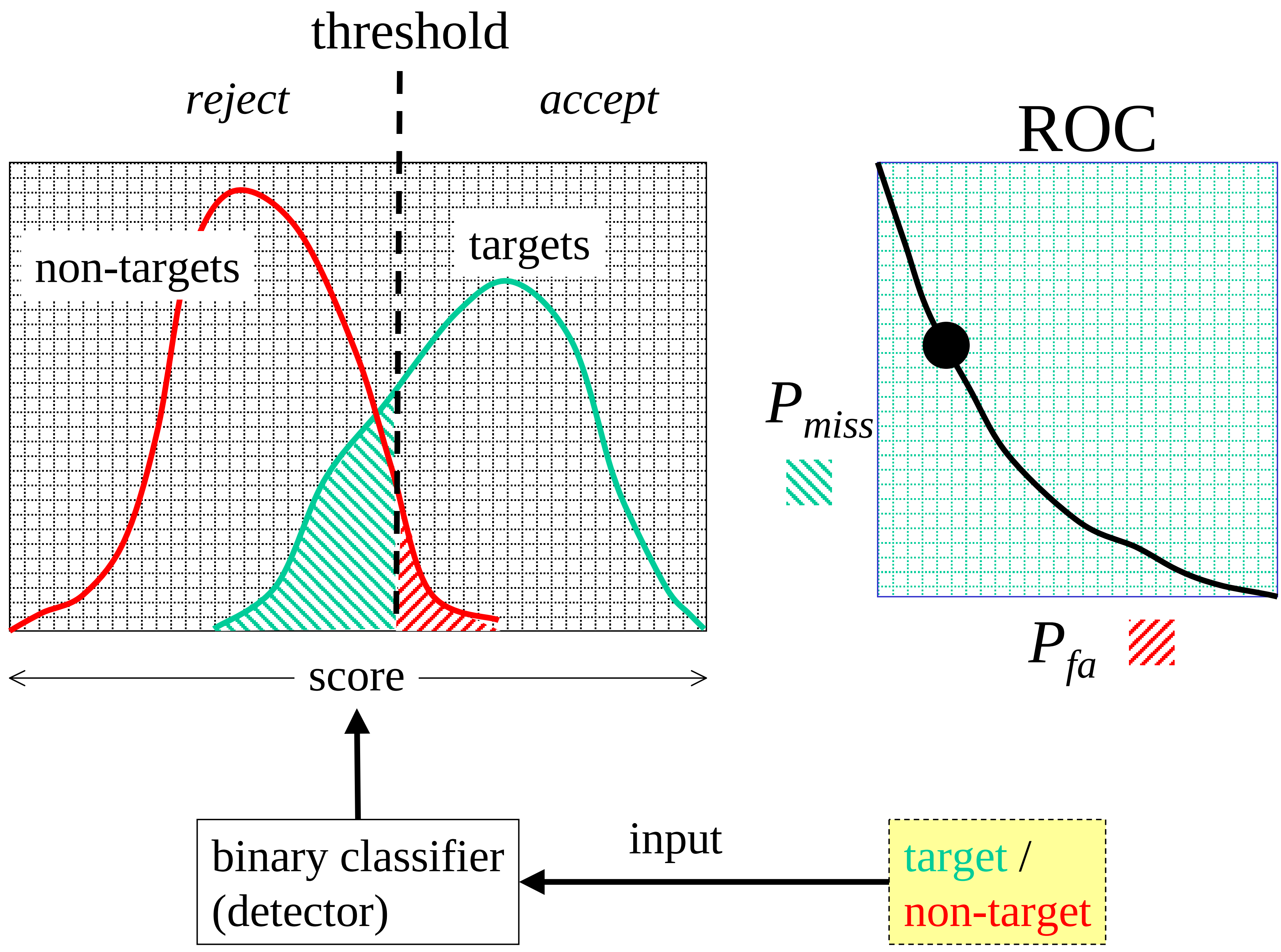

score distributions
non-targets
targets
score
binary classifier
(detector)
input
target /
non-target

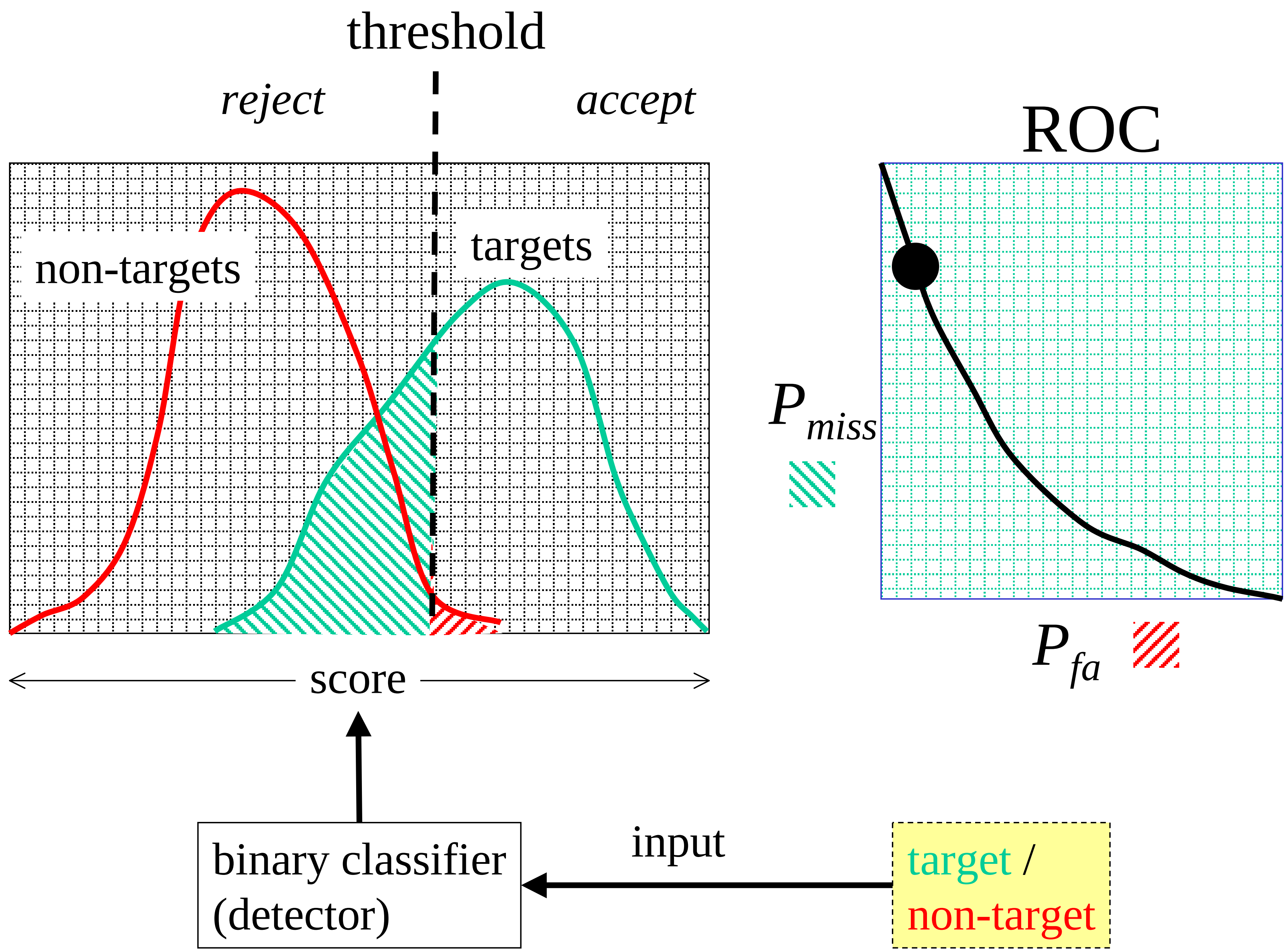

threshold
reject
accept
non-targets
targets
score
binary classifier
(detector)
input
target /
non-target

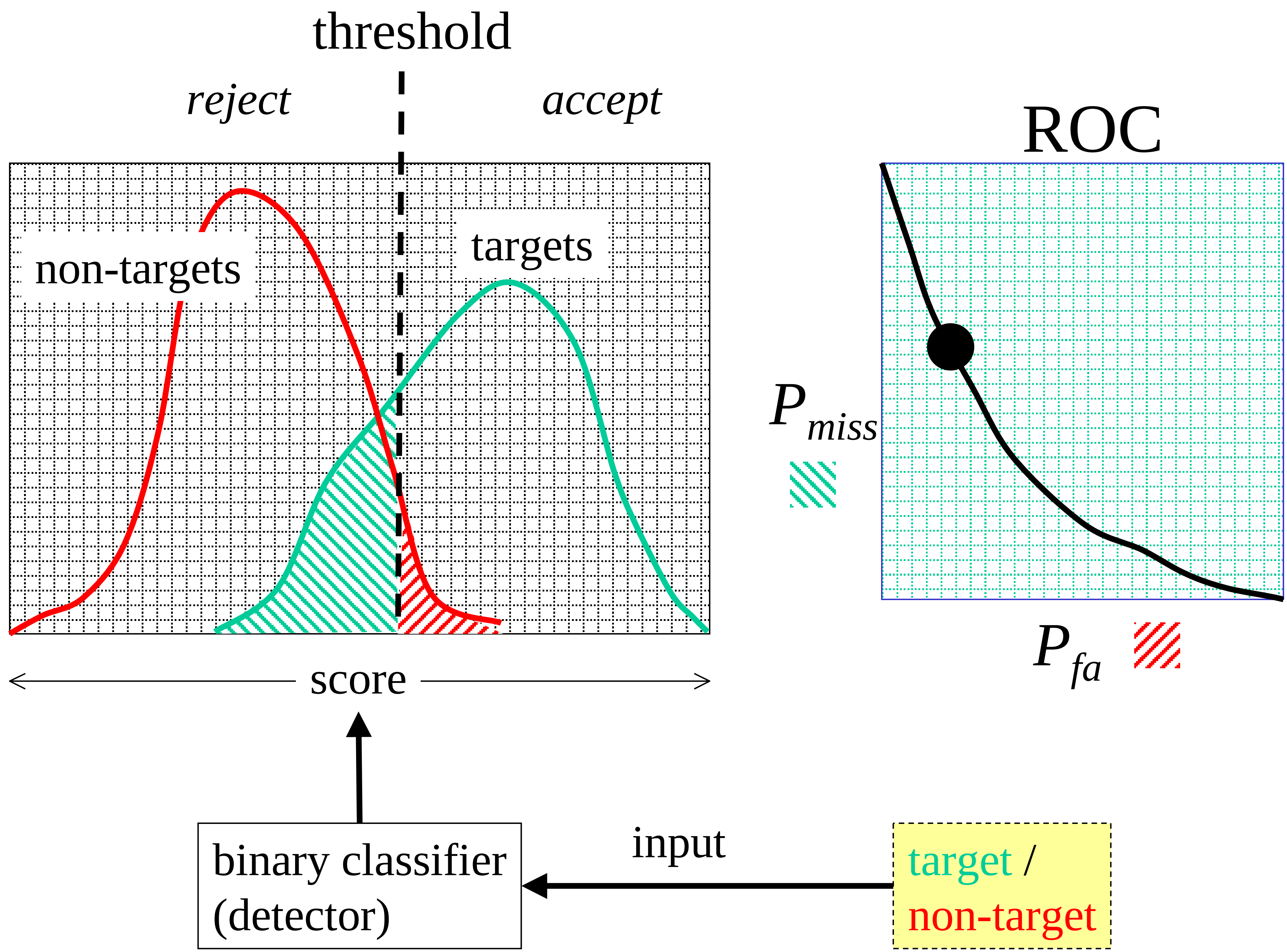

threshold
reject
accept
non-targets
targets
score
ROC
P_{miss}
P_{fa}
binary classifier
(detector)
input
target /
non-target

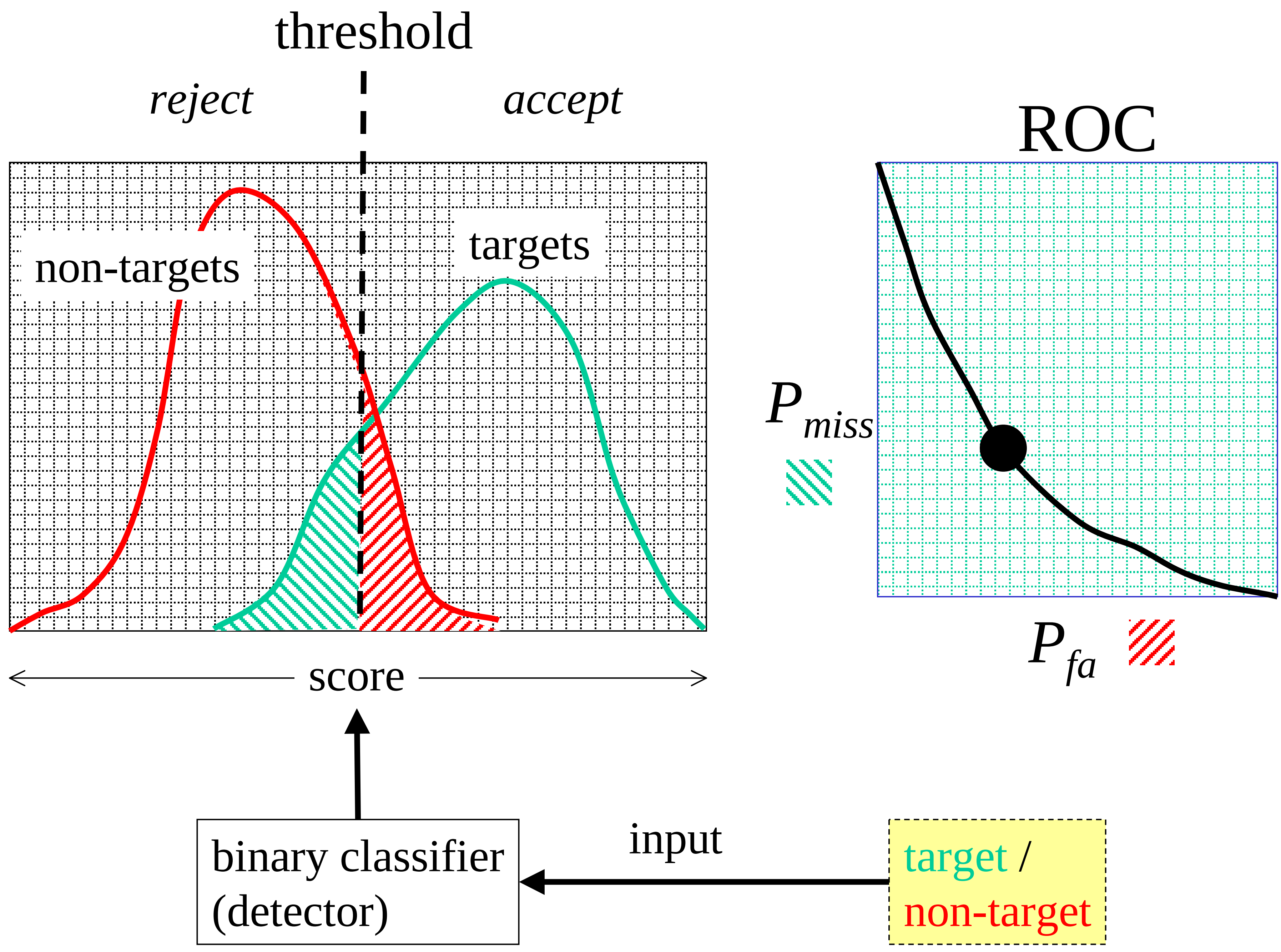

threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
input
binary classifier
(detector)
target /
non-target

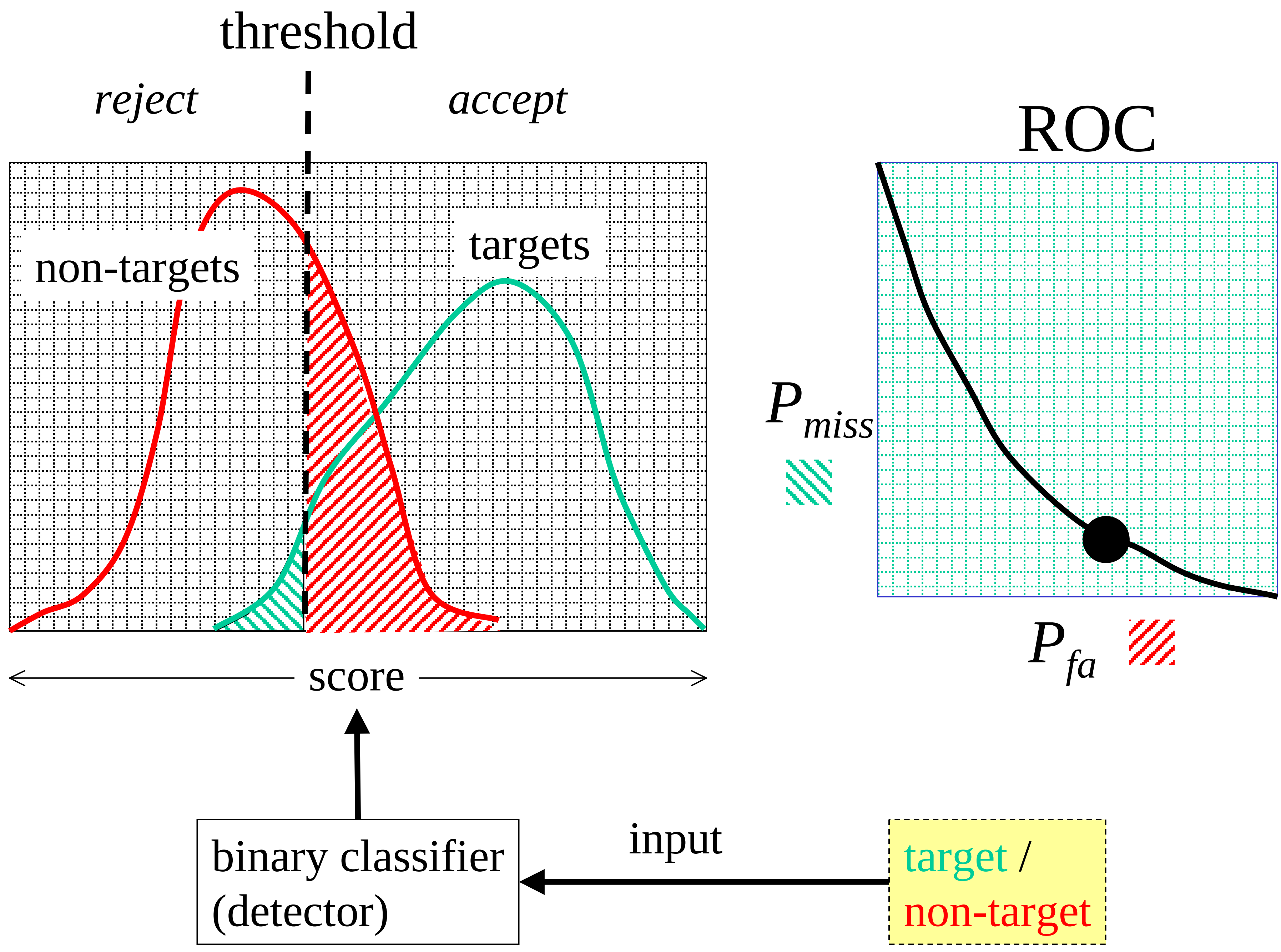

threshold
reject
accept
non-targets
targets
score
ROC
P_{miss}
P_{fa}
binary classifier
(detector)
input
target /
non-target

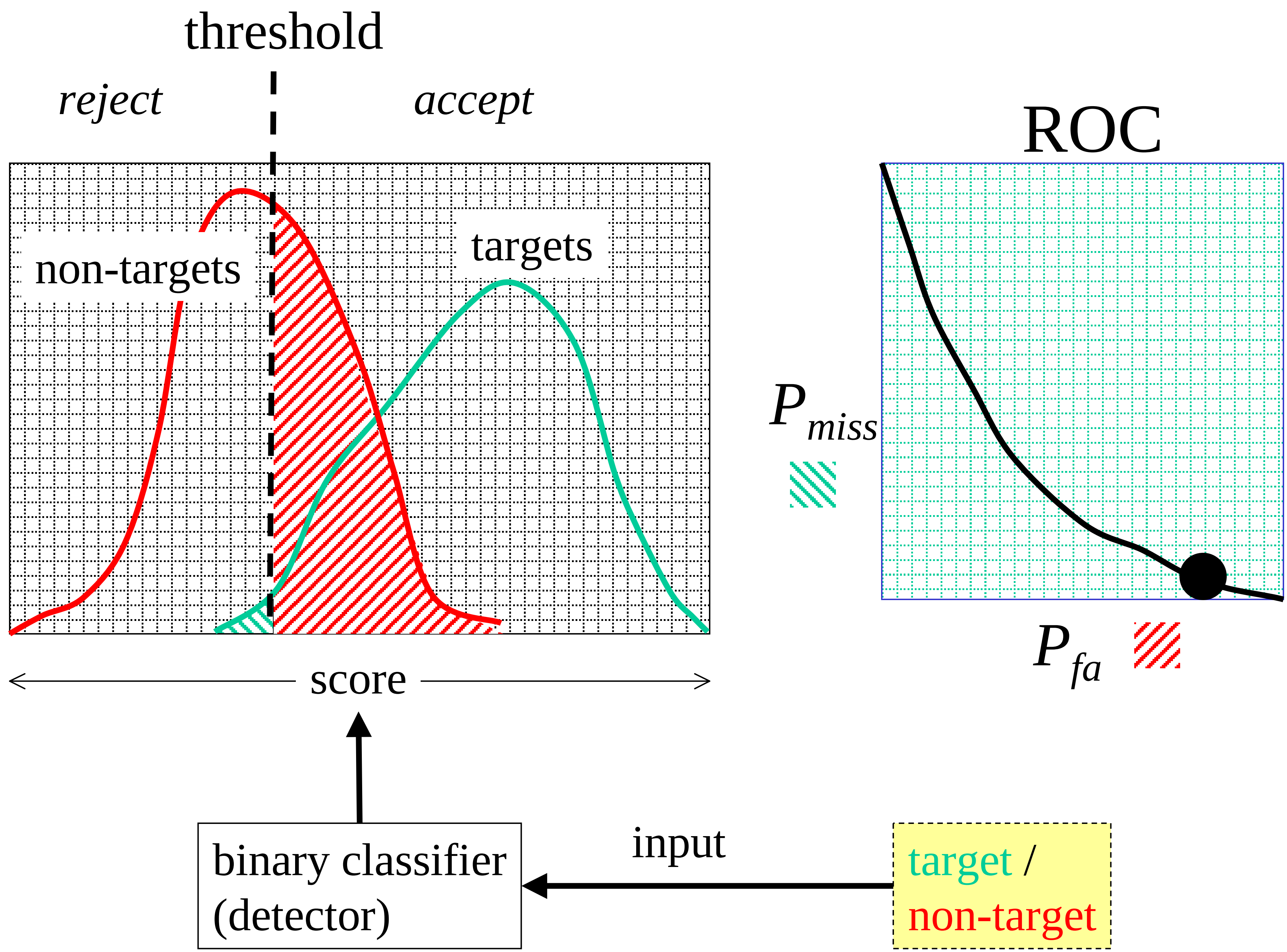

threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
input
binary classifier
(detector)
target /
non-target


threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
input
binary classifier
(detector)
target /
non-target


threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
input
binary classifier
(detector)
target /
non-target


threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
binary classifier
(detector)
input
target /
non-target

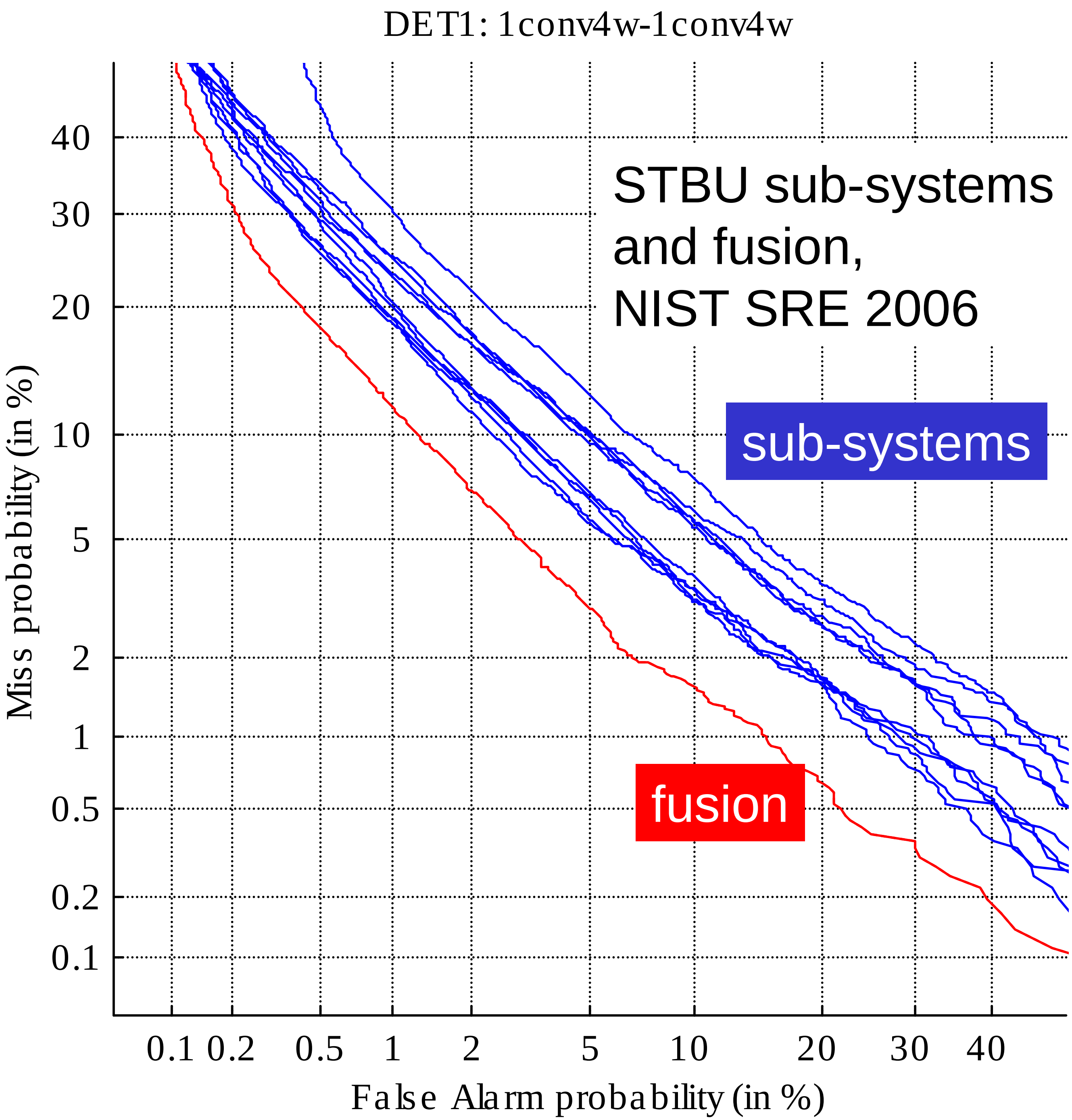

threshold
reject
accept
non-targets
targets
score
ROC
P_{miss}
P_{fa}
binary classifier
(detector)
input
target /
non-target


threshold
reject
accept
ROC
non-targets
targets
P_{miss}
P_{fa}
score
binary classifier
(detector)
input
target /
non-target

DET ≡ ROC

- DET-curve is *equivalent* to ROC
- uses warping of axes to make curves approximately linear
- good for *displaying* comparative performance of multiple binary classifiers

Example

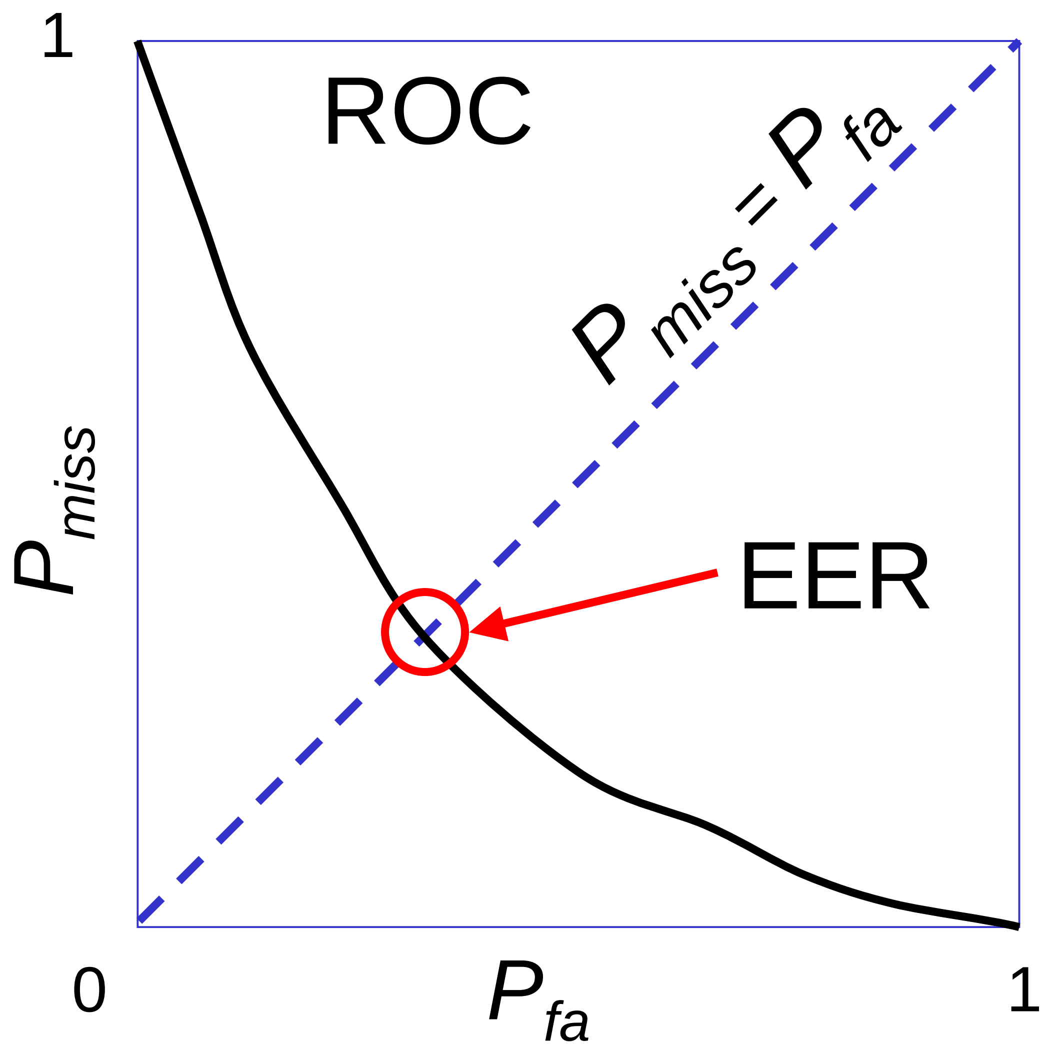

DET1: 1conv4w-1conv4w
STBU sub-systems
and fusion,
NIST SRE 2006
sub-systems
fusion
Miss probability (in %)
False Alarm probability (in %)
40
30
20
10
5
2
1
0.5
0.2
0.1
0.1
0.2
0.5
1
2
5
10
20
30
40

Scalar Representations of DET/ROC

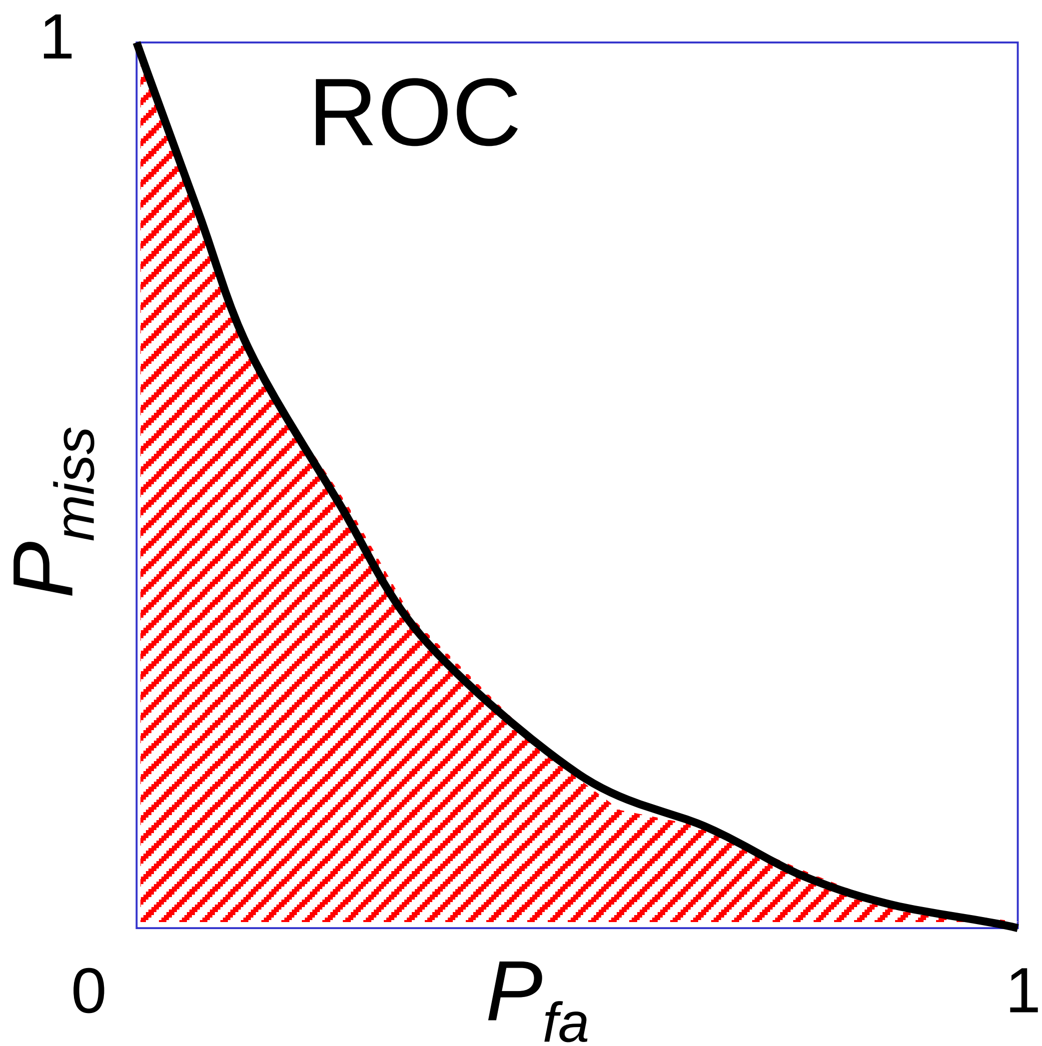
- AUC = *Area-Under-Curve*
  - works for ROC (but undefined for DET-curve)
  - popular in medical literature

- EER = *Equal-Error-Rate*
  - works for both ROC and DET
  - popular in speaker recognition

EER: *Equal-Error-Rate*



AUC: *Area-Under-Curve*



DET/ROC Advantages

- Very useful summary of the *potential* of recognizers *to discriminate* between two classes over a wide spectrum of applications involving different priors and/or misclassification costs.

(Speaker recognition researchers *love* DET-curves!)

DET/ROC *Disadvantages*

- Does *not* measure actual decision-making ability
  - thresholds are set by evaluator, not by the technology under evaluation.
- Gives *overoptimistic* estimates of the average cost, or average error-rate, when applying the recognizer to make hard decisions.

DET/ROC *Disadvantages*

- EER and AUC are difficult to use as numerical optimization objectives.

DET/ROC *Disadvantages*

- Problematic for multiclass ($N > 2$)
  - Conflicting definitions
  - Difficult to compute
  - Error-rate increases with N

Previous approaches

1. ROC / DET
2. **Detection Cost Function: $\boldsymbol{C_{det}}$**

Detection Cost Function

- Naturally applicable and well-defined for *binary* classification problems (i.e. speaker detection)
- Can be applied (with moderate complexity and some pitfalls) to *indirectly* evaluate *multiclass* recognizers (e.g. NIST LRE Language Detection Task)

Detection Cost Function

✓ Does require hard decisions and therefore *does evaluate calibration* of recognizer under evaluation, but

✗ Evaluates only for a fixed application. The ability of the recognizer in *other applications is not exercised*.

C_{det}: 5-minute tutorial

Detection Cost Function

$$\boxed{\begin{aligned} C_{det} &= P_{tar} C_{miss} P_{miss} \\ &\quad + (1 - P_{tar}) C_{fa} P_{fa} \end{aligned}}$$

Expected cost of using recognizer for a *specific application*.

Detection Cost Function

$$C_{det} = \boxed{P_{tar}} C_{miss} P_{miss} + \boxed{(1 - P_{tar})} C_{fa} P_{fa}$$

P_{tar} : application-dependent target *prior*.

Detection Cost Function

$$C_{det} = P_{tar} C_{miss} P_{miss} + (1 - P_{tar}) C_{fa} P_{fa}$$

C_{miss}, C_{fa} : application-dependent mis-classification *costs*.

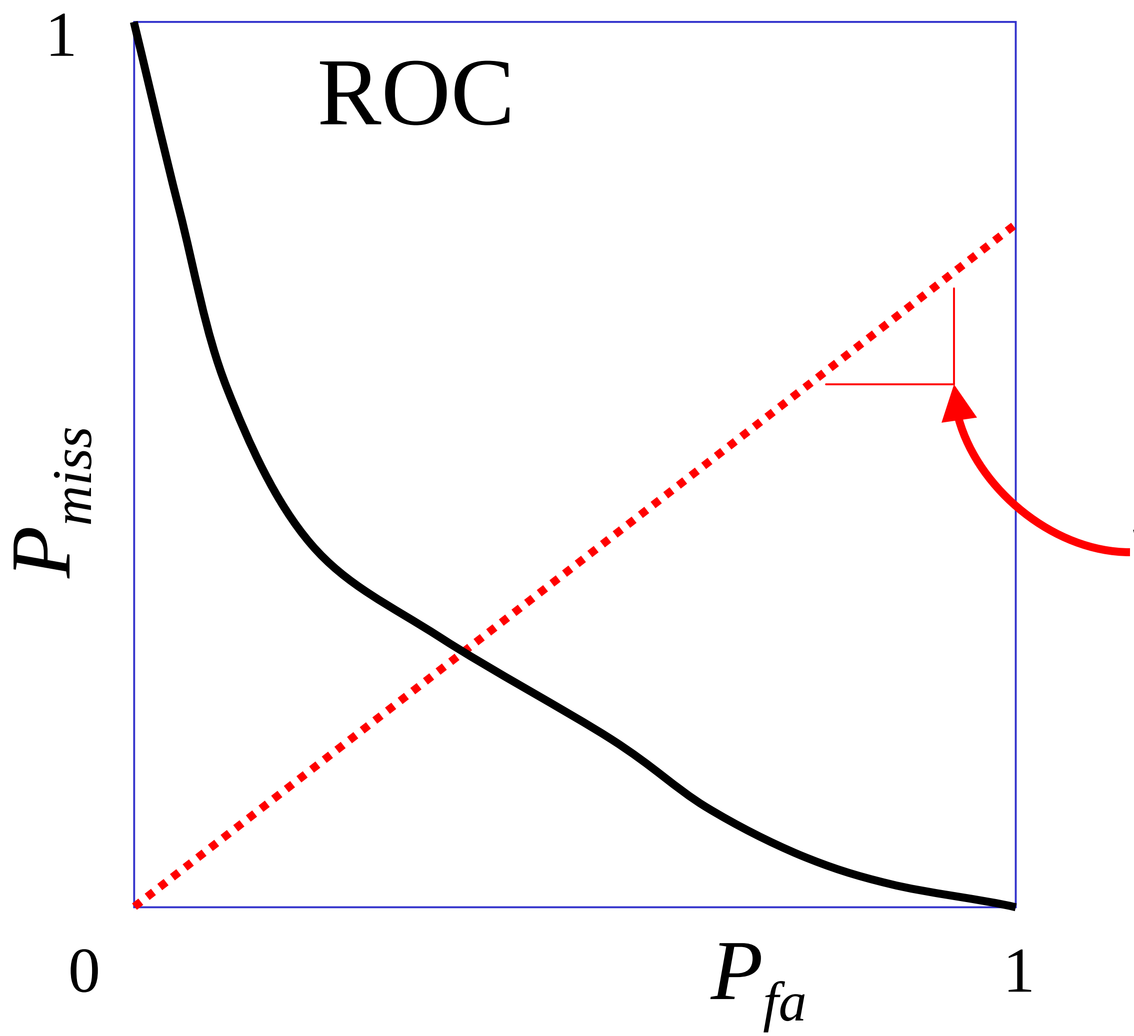
$$
\begin{aligned}
C_{det} &= P_{tar} C_{miss} \boxed{P_{miss}} \\
&+ (1 - P_{tar}) C_{fa} \boxed{P_{fa}}
\end{aligned}
$$

C_{det} is a *weighted* linear combination of the two misclassification error-rates, P_{miss} and P_{fa}.

$$C_{det} = P_{tar} C_{miss} P_{miss} + (1 - P_{tar}) C_{fa} P_{fa}$$

geometric interpretation

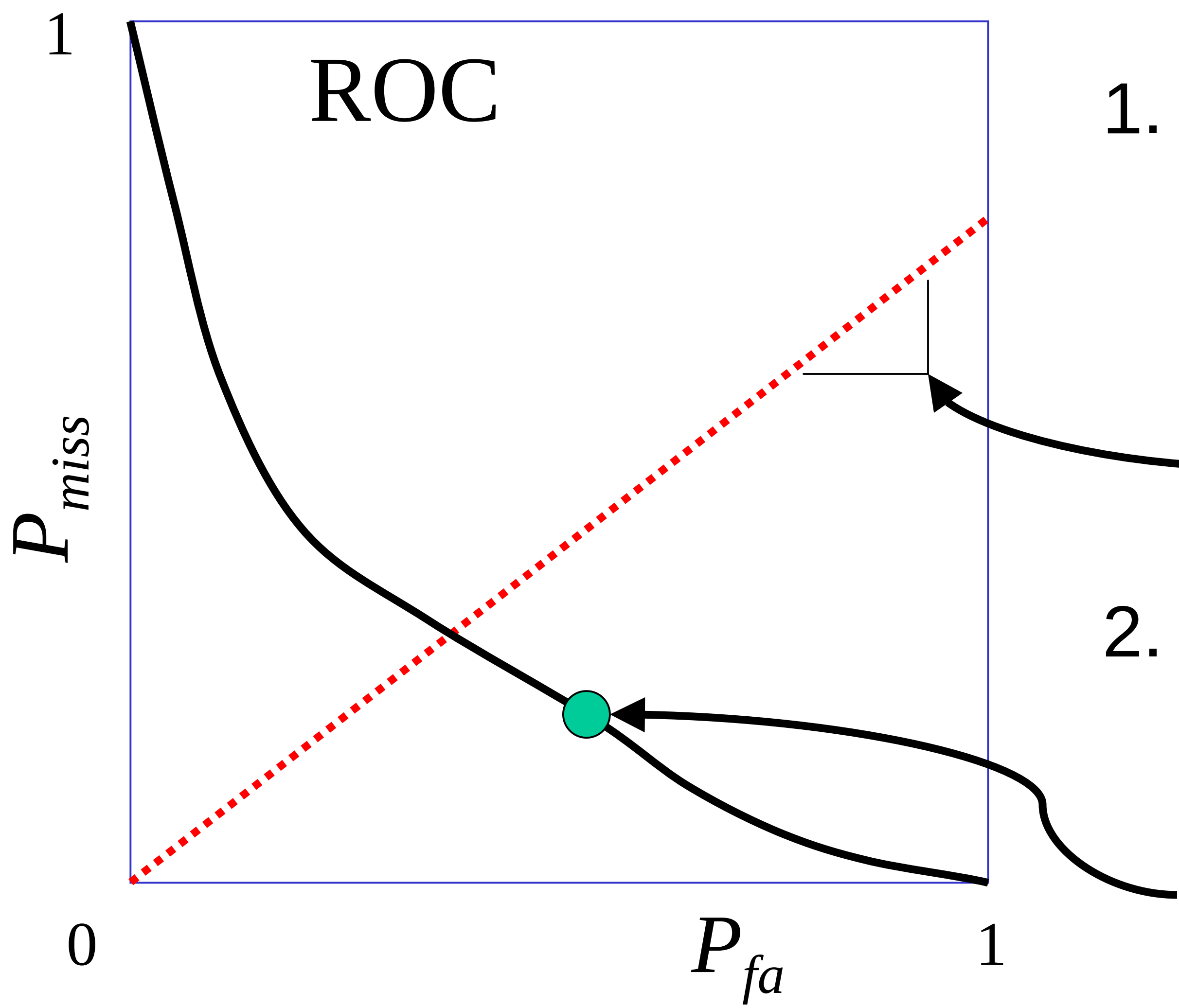
C_{det}



Weighted linear combination of P_{miss} and P_{fa} .

- Relative weighting (slope) depends on priors and costs of a specific application.

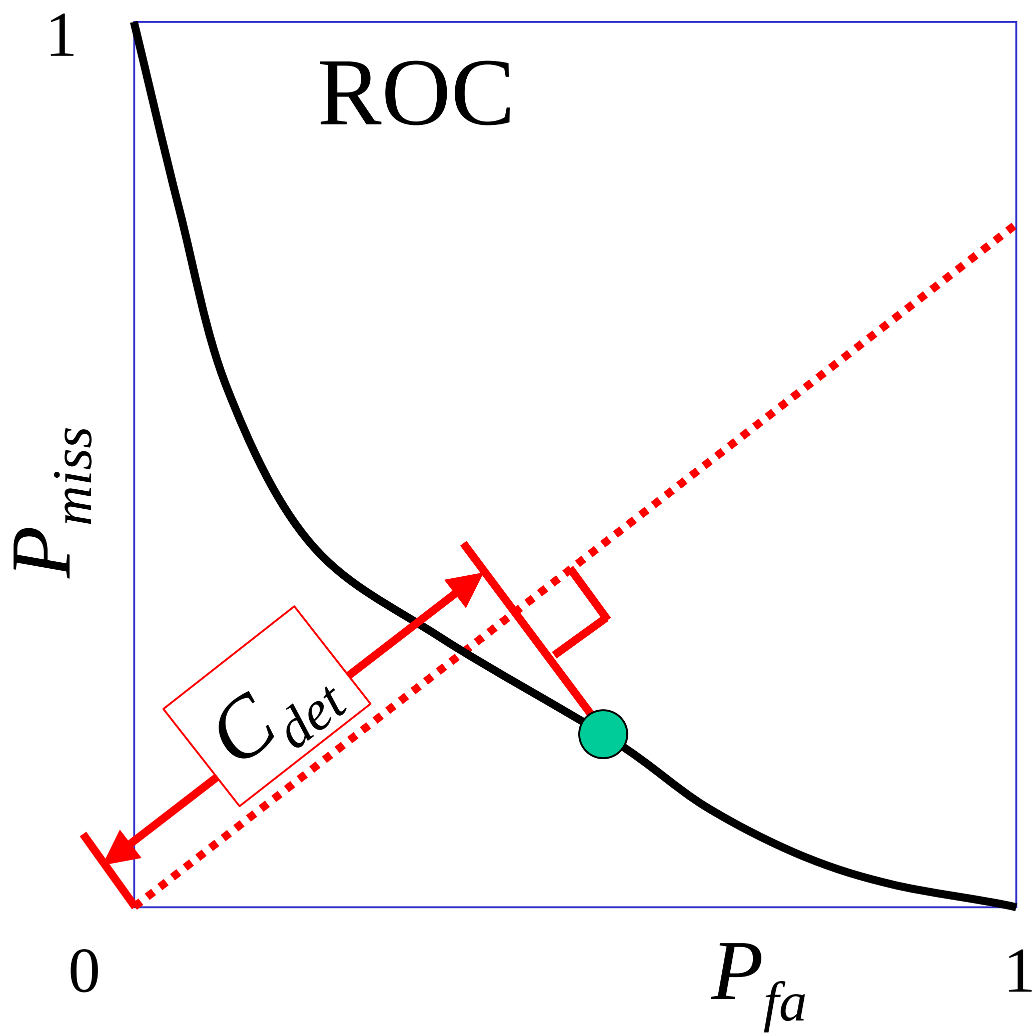
C_{det}



1. Evaluator chooses *application* (cost, prior) which determines *slope*.

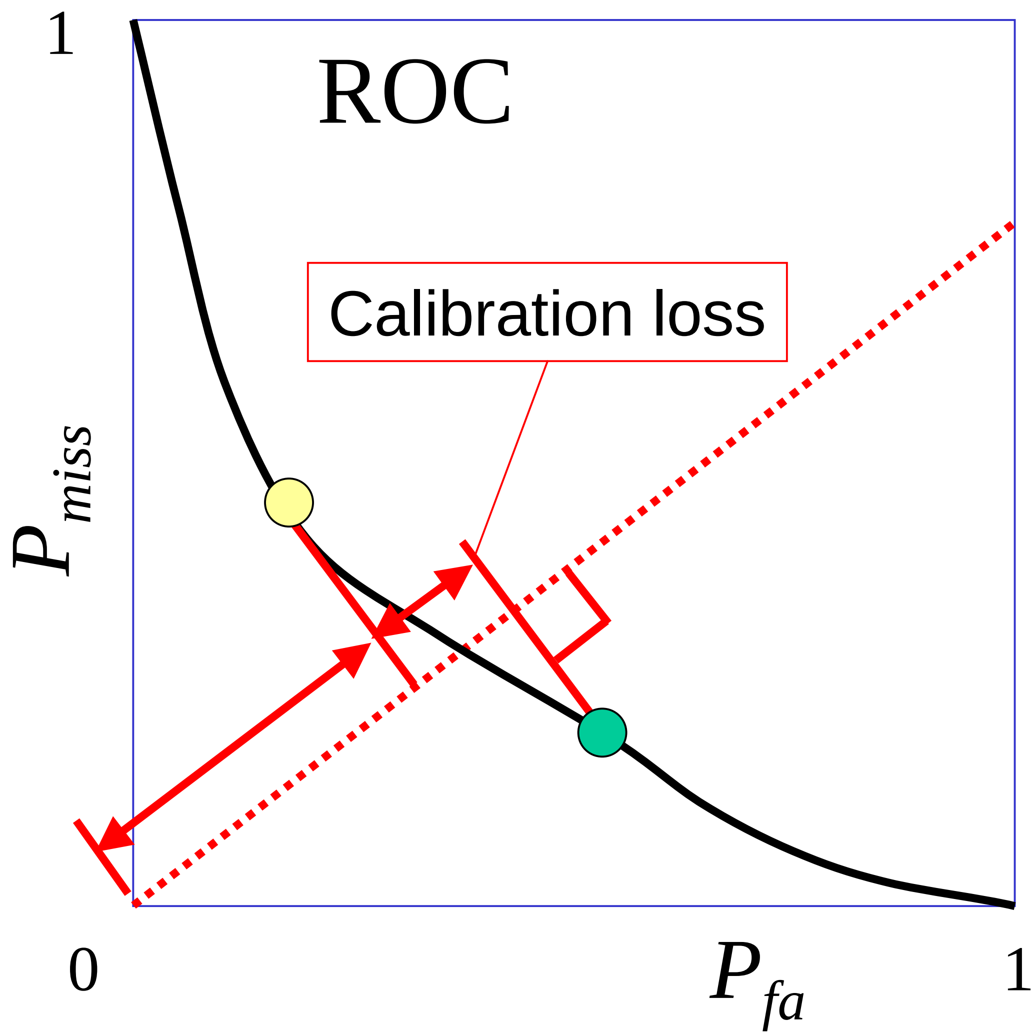
2. Evaluee chooses threshold, which determines operating *point*.

C_{det}



1. Evaluator chooses *slope*.
2. Evaluee chooses operating *point*.
3. C_{det} is the linear combination of the two error-rates, as indicated by the *projection* onto the sloped line.

C_{det}



- Evaluee chooses operating point (threshold) *without* access to true class labels. (Cannot see ROC curve.)
- Threshold may be suboptimal.
- Difference is *calibration loss*.

C_{det}

✓ Does *evaluate calibration*, but

✗ Evaluates only for a fixed application.

Summary of previous approaches

	ROC	C_{det}
application-independent	✓	✗
evaluates calibration	✗	✓

Part I

1. Introduction
2. Good old error-rate
3. **Binary case:**

$$\text{Error-Rate} \rightarrow \text{ROC} \rightarrow C_{det} \rightarrow \boldsymbol{C_{llr}}$$

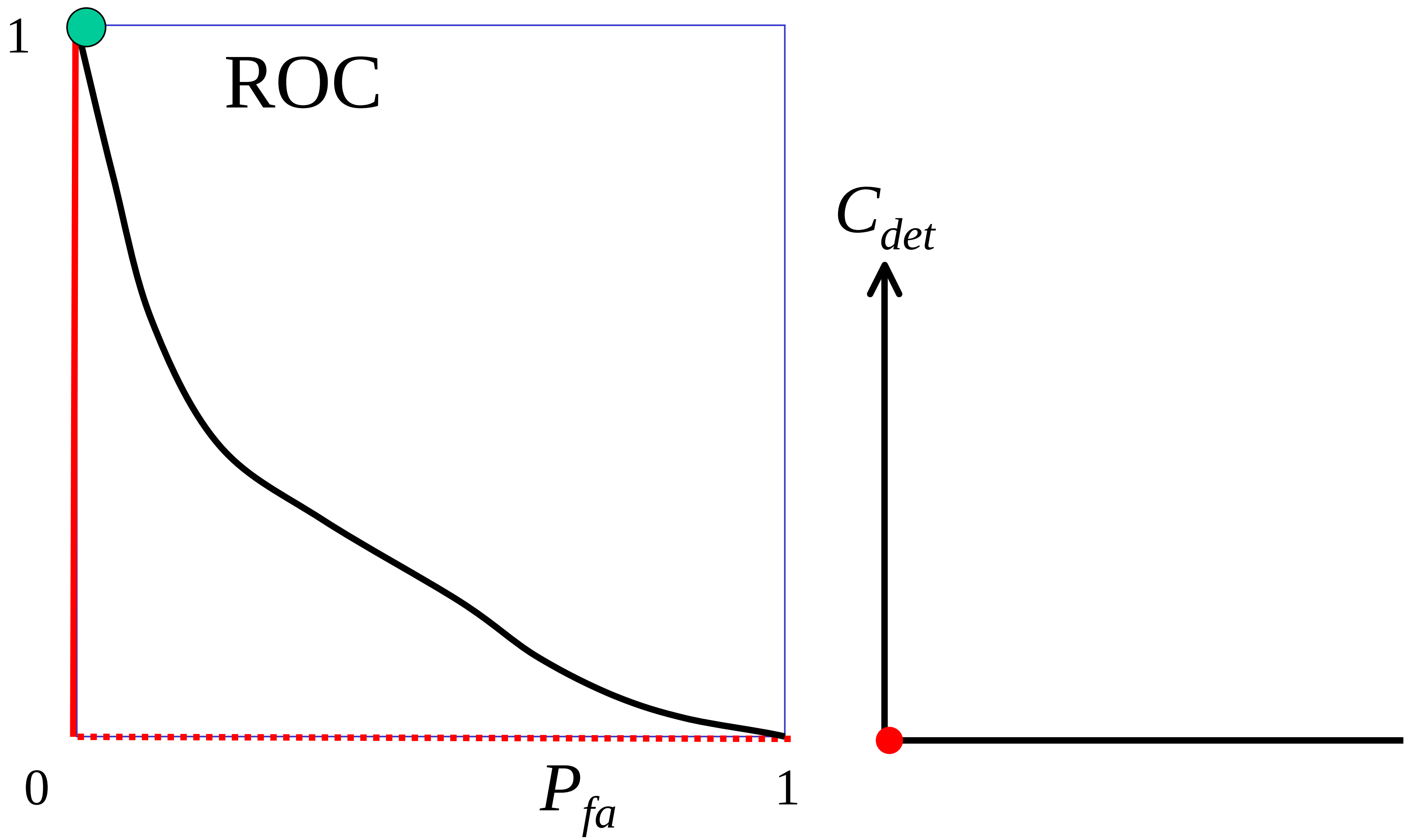
New approach: C_{llr}

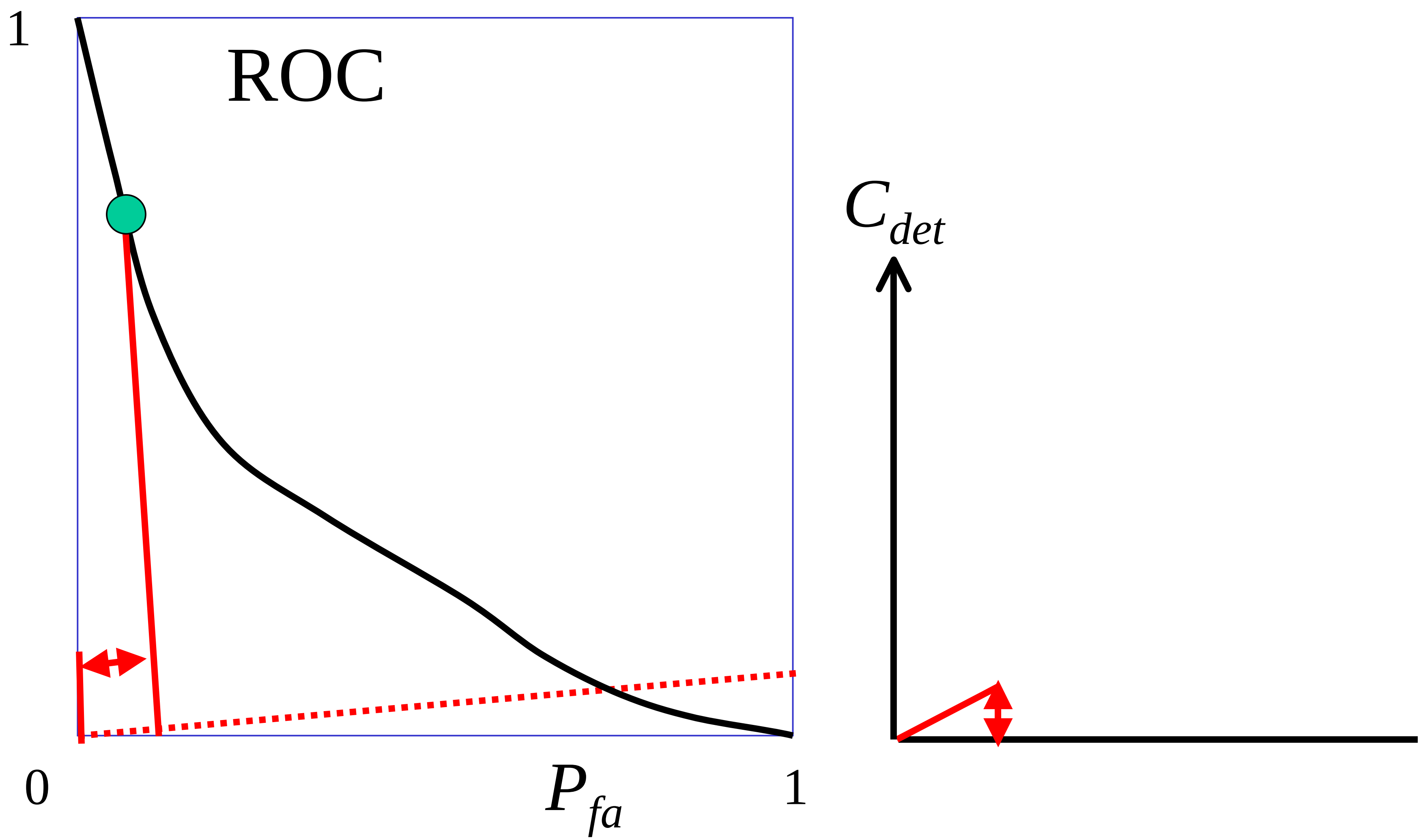
- Combines ROC and C_{det} to get good qualities of both:
  - Application-independent
  - Evaluates Calibration

New approach: C_{llr}

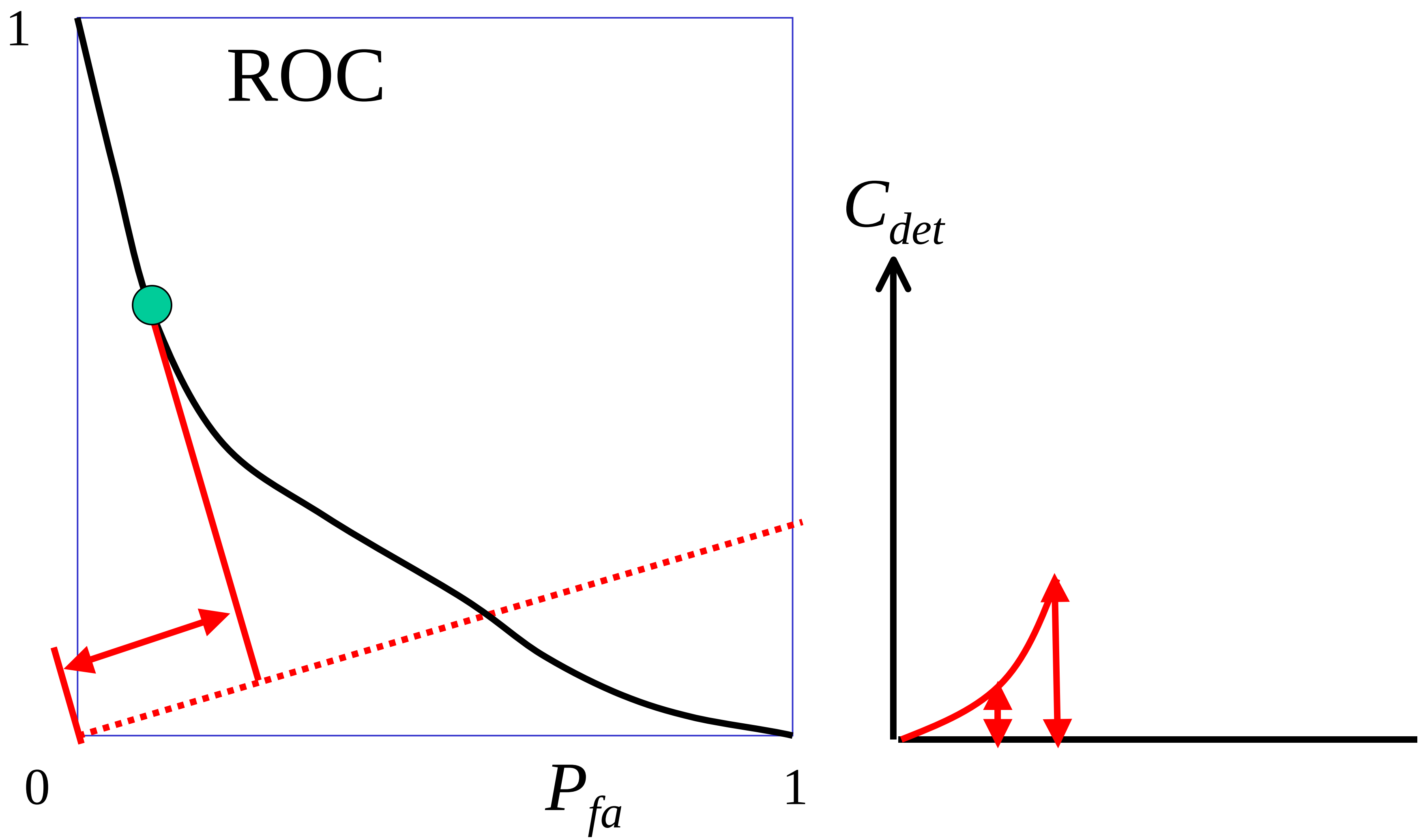
ROC and C_{det} are combined by *integrating* C_{det} over the whole ROC-curve.

$$\boxed{C_{llr} = \int_{ROC} C_{det}}$$

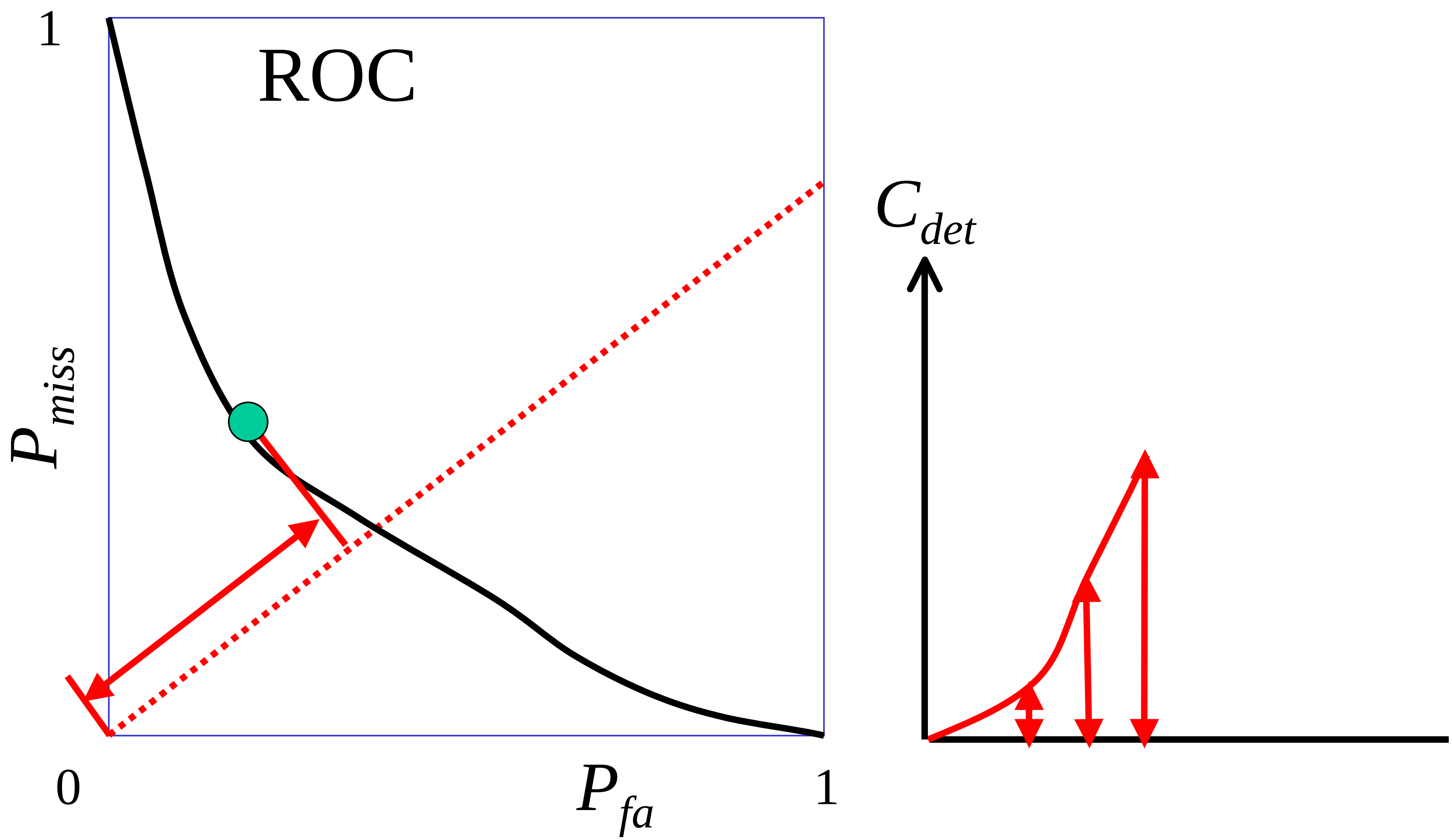

1
ROC
C_{det}
0
P_{fa}
1



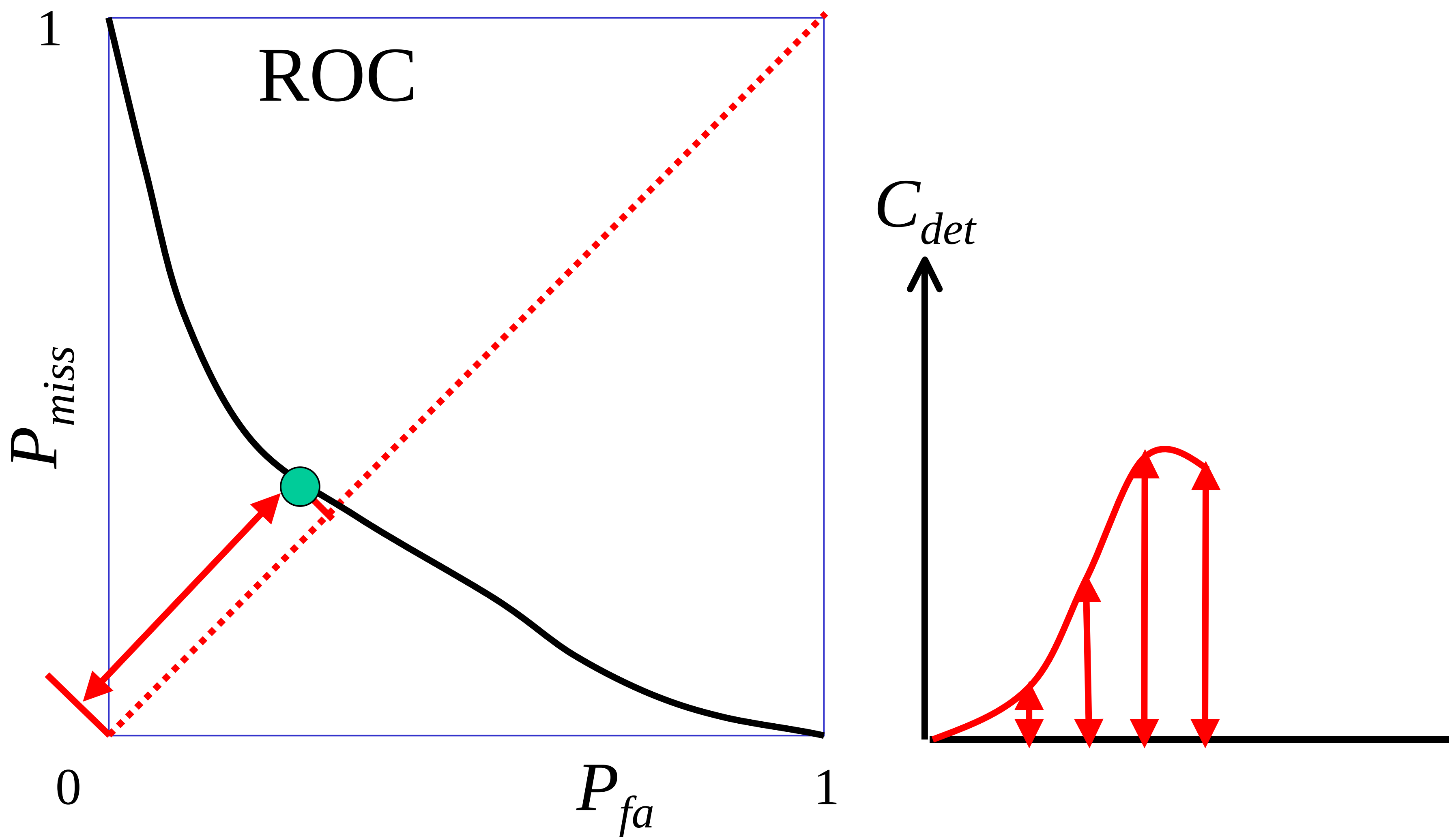
1
ROC
C_{det}
0
P_{fa}
1



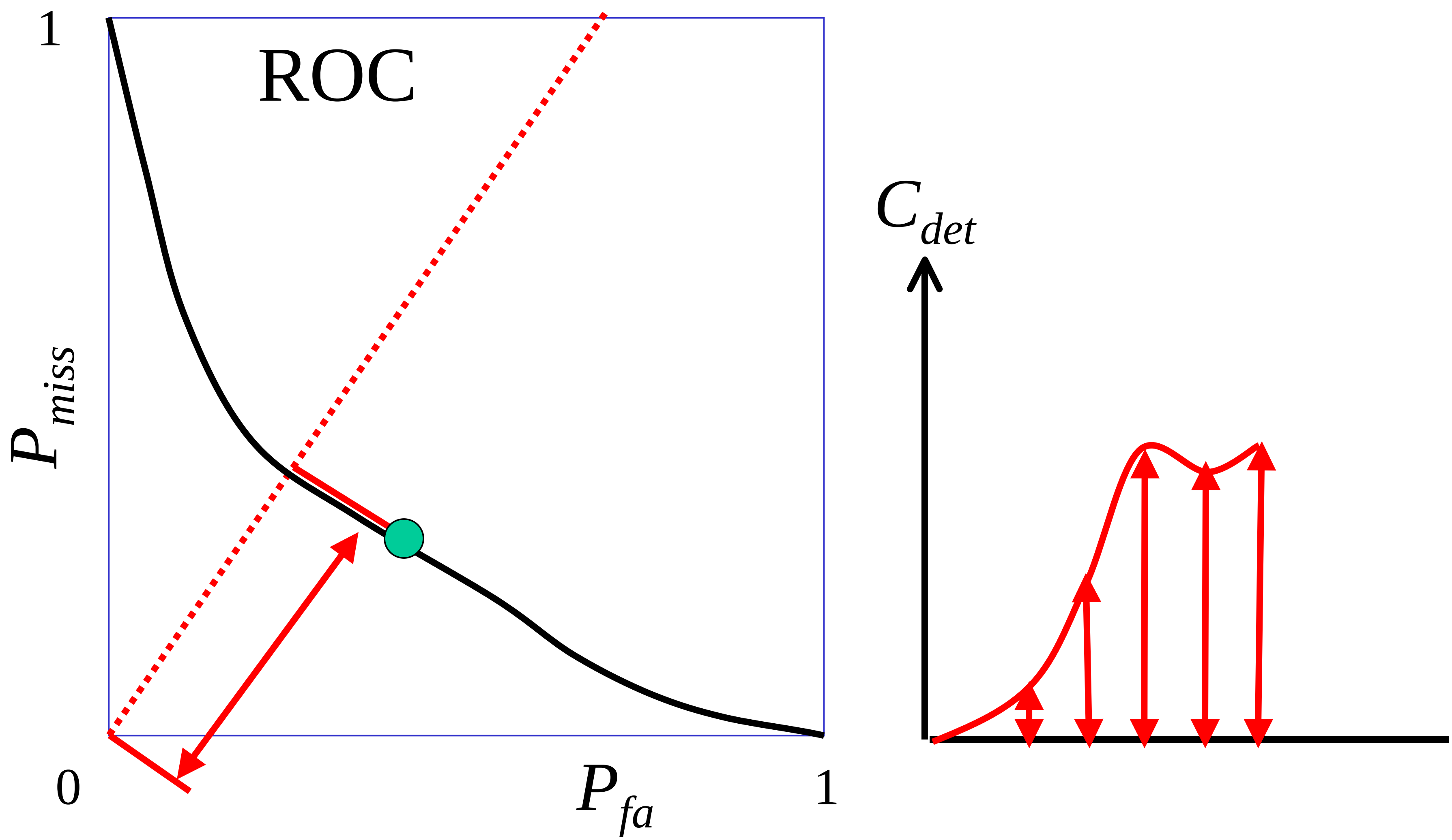
1
ROC
0
P_{fa}
1
C_{det}



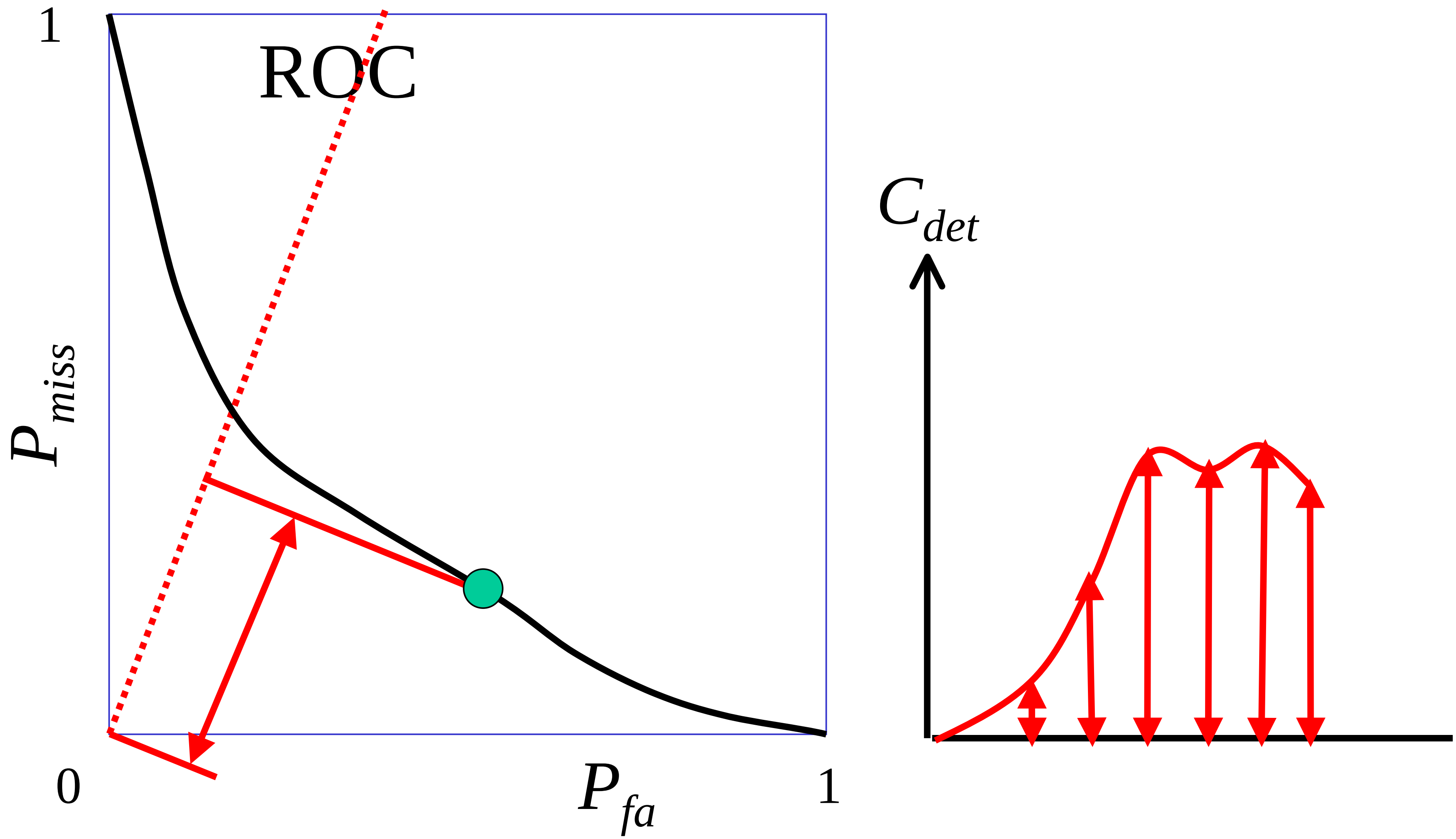
1
ROC
P_{miss}
0
P_{fa}
1
C_{det}



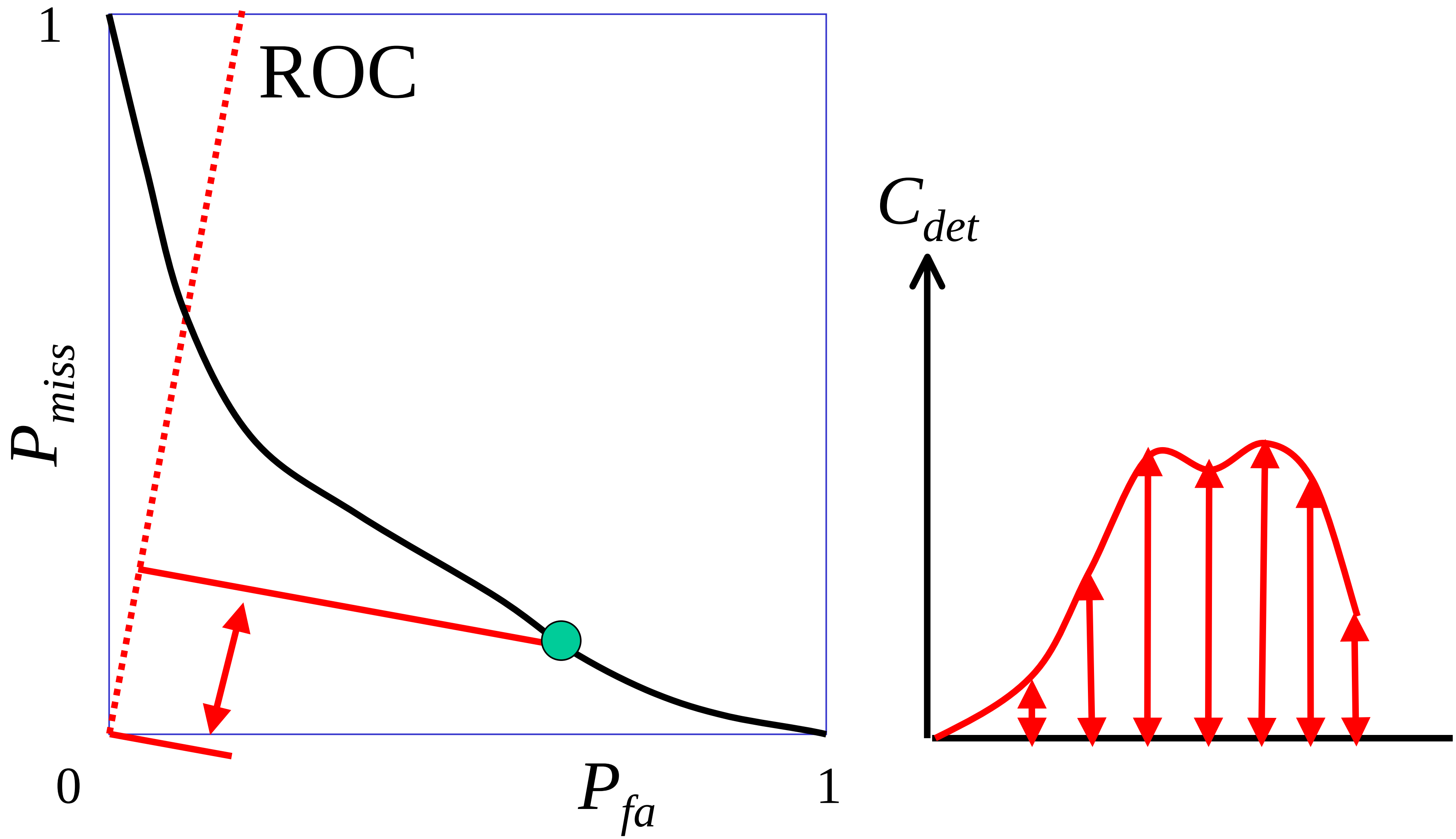
1
ROC
P_{miss}
0
P_{fa}
1
C_{det}



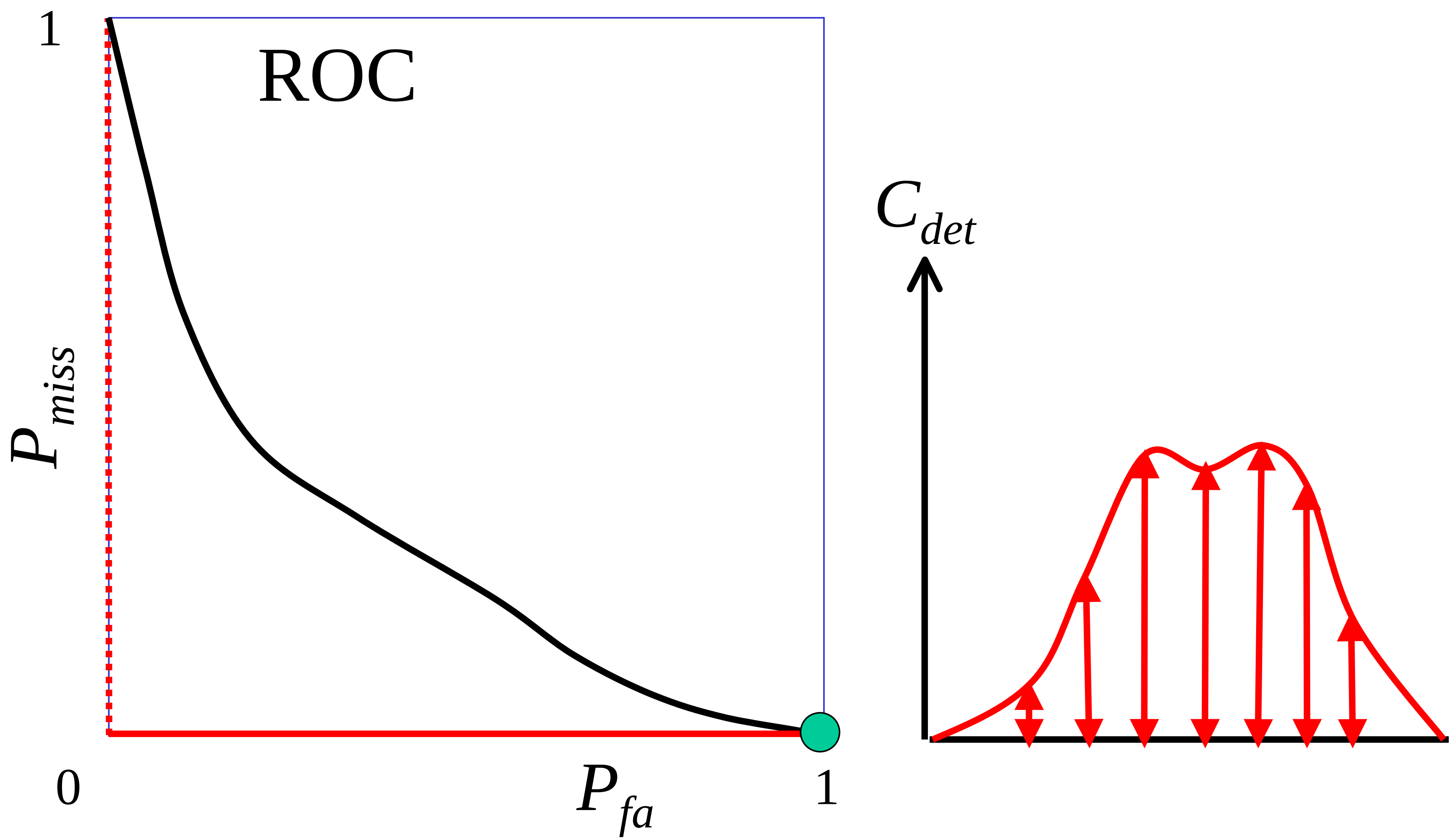
1
ROC
P_{miss}
0
P_{fa}
1
C_{det}



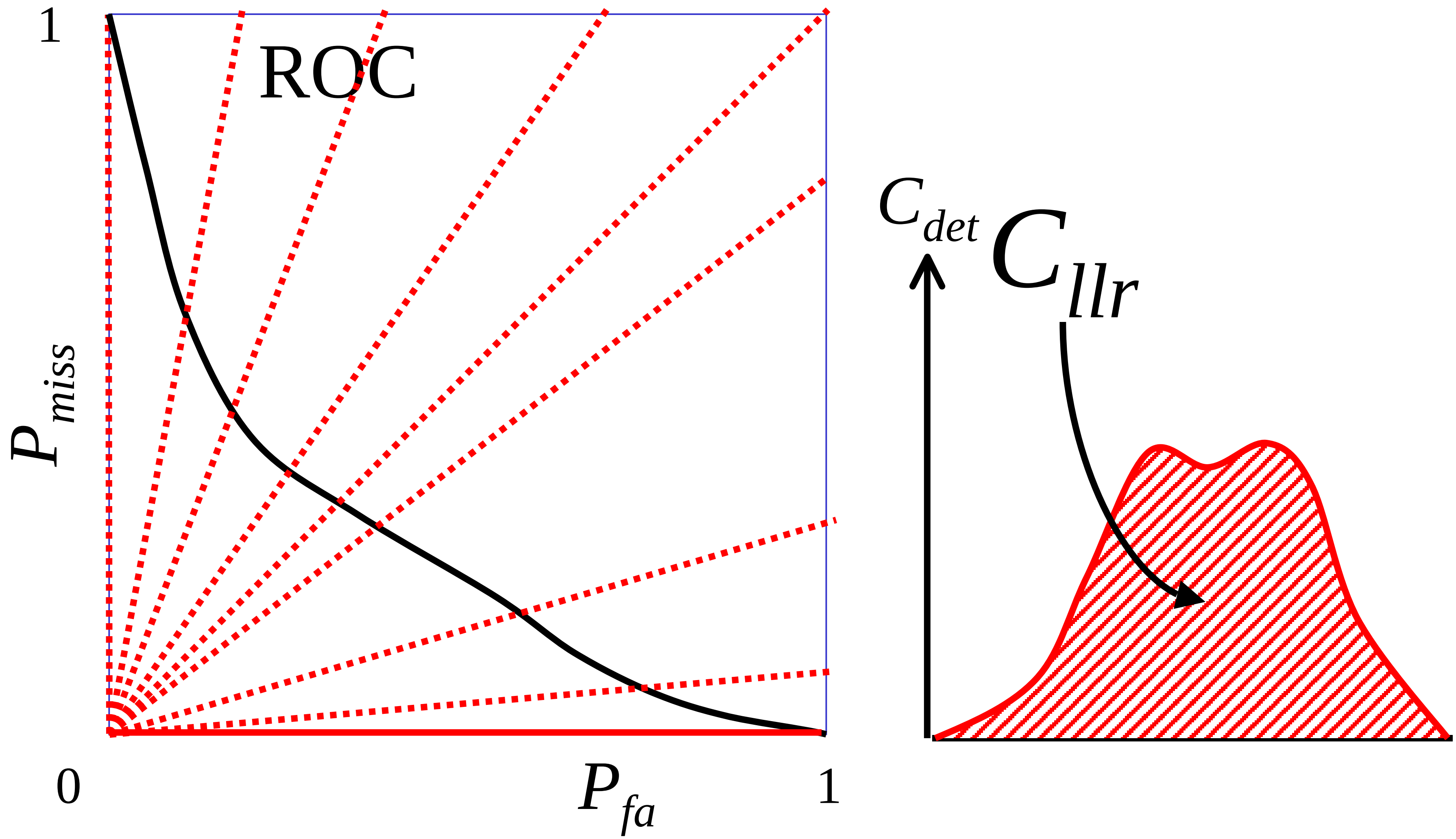
1
ROC
P_{miss}
0
P_{fa}
1
C_{det}



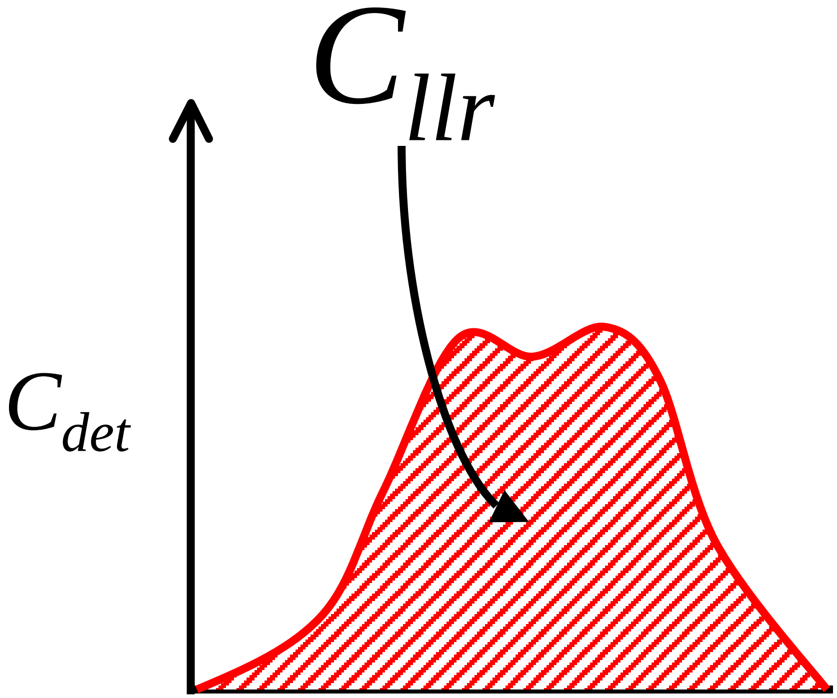
1
ROC
P_{miss}
0
P_{fa}
1
C_{det}



1
ROC
P_{miss}
0
P_{fa}
1
C_{det}

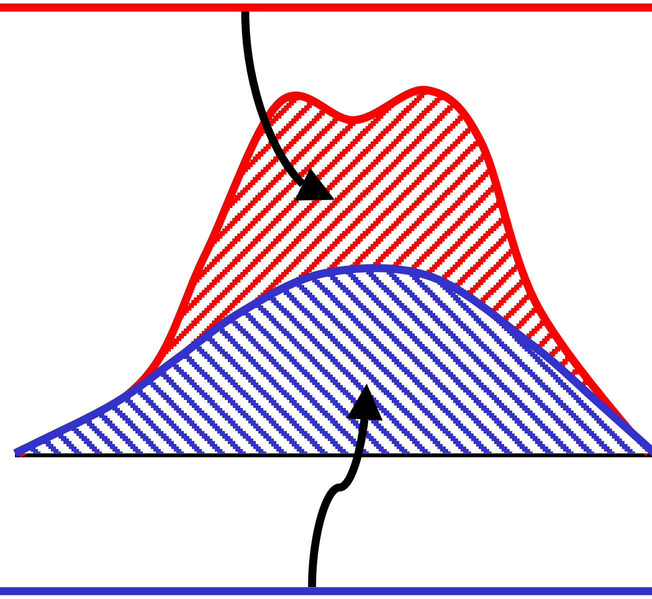


1
ROC
P_{miss}
0
P_{fa}
1
C_{det}
C_{llr}



C_{llr} integrates cost of decisions made with *suboptimal thresholds* chosen by evaluee.

Calibration loss:
extra cost because of
suboptimal thresholds



Discrimination loss:
integrated decision cost at
thresholds optimized by evaluator.

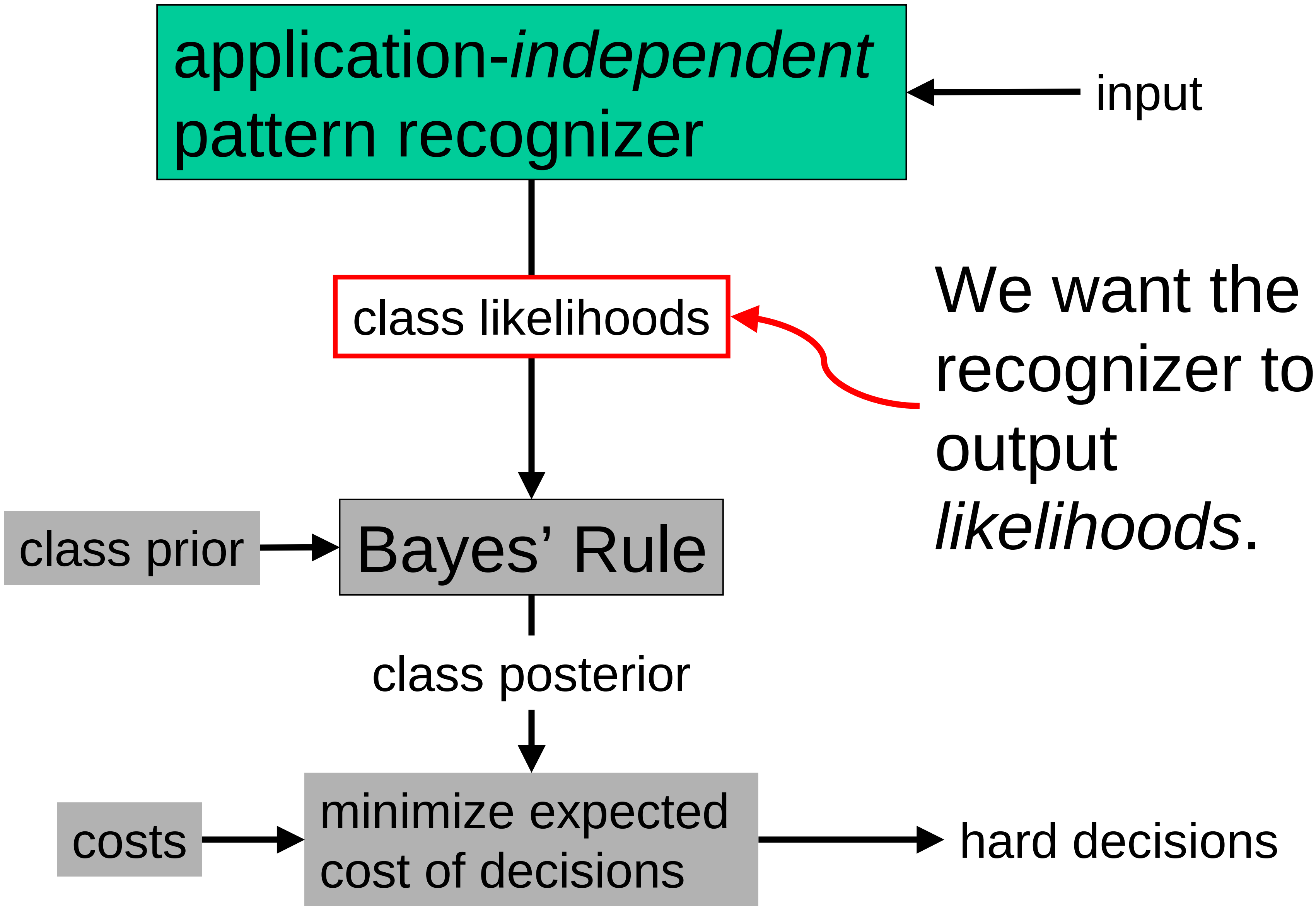
Questions

1. **How does evaluee set thresholds?**

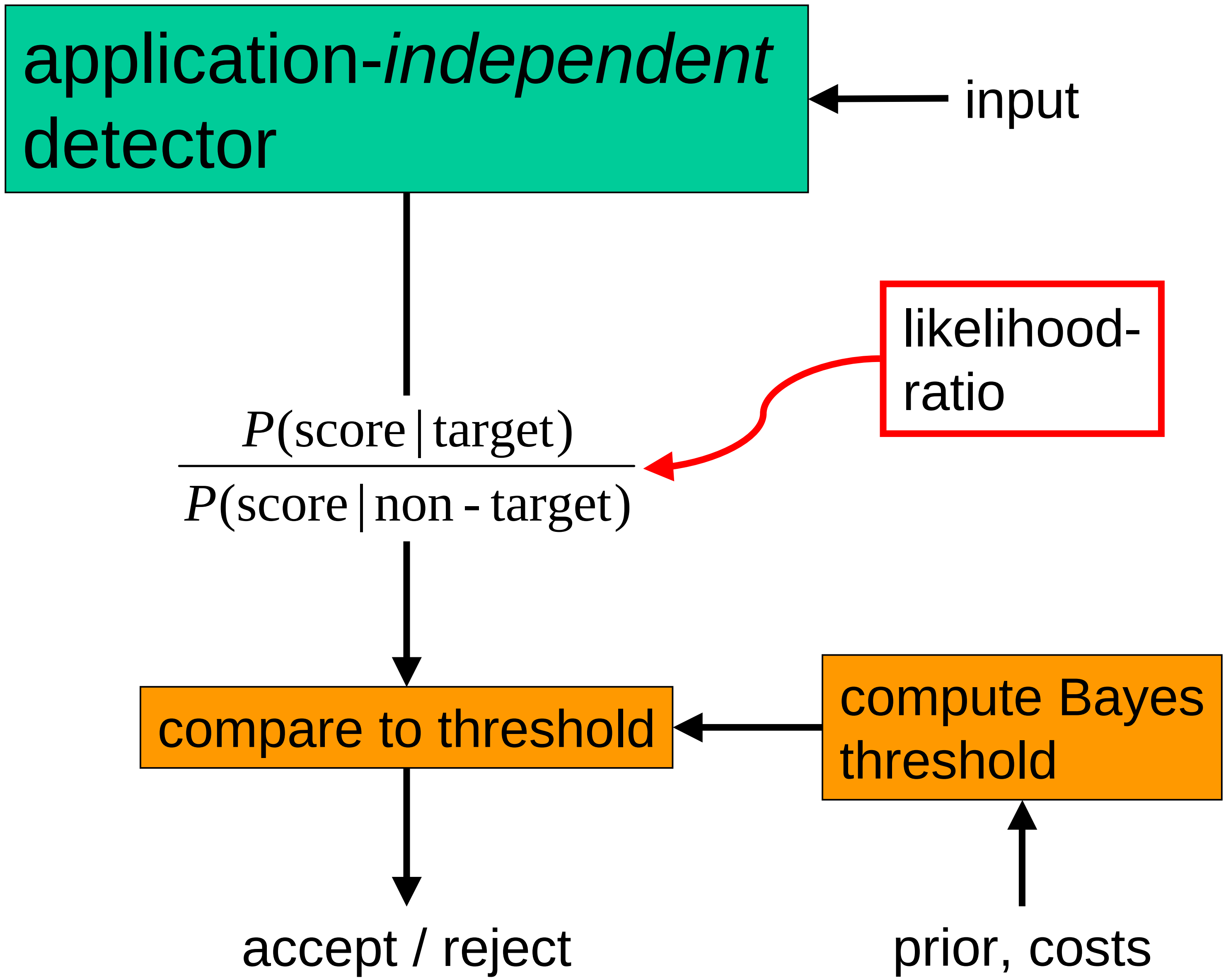
2. How does evaluator perform the integral?

How does evaluee (recognizer) set thresholds?

To answer this, let's review our agenda.

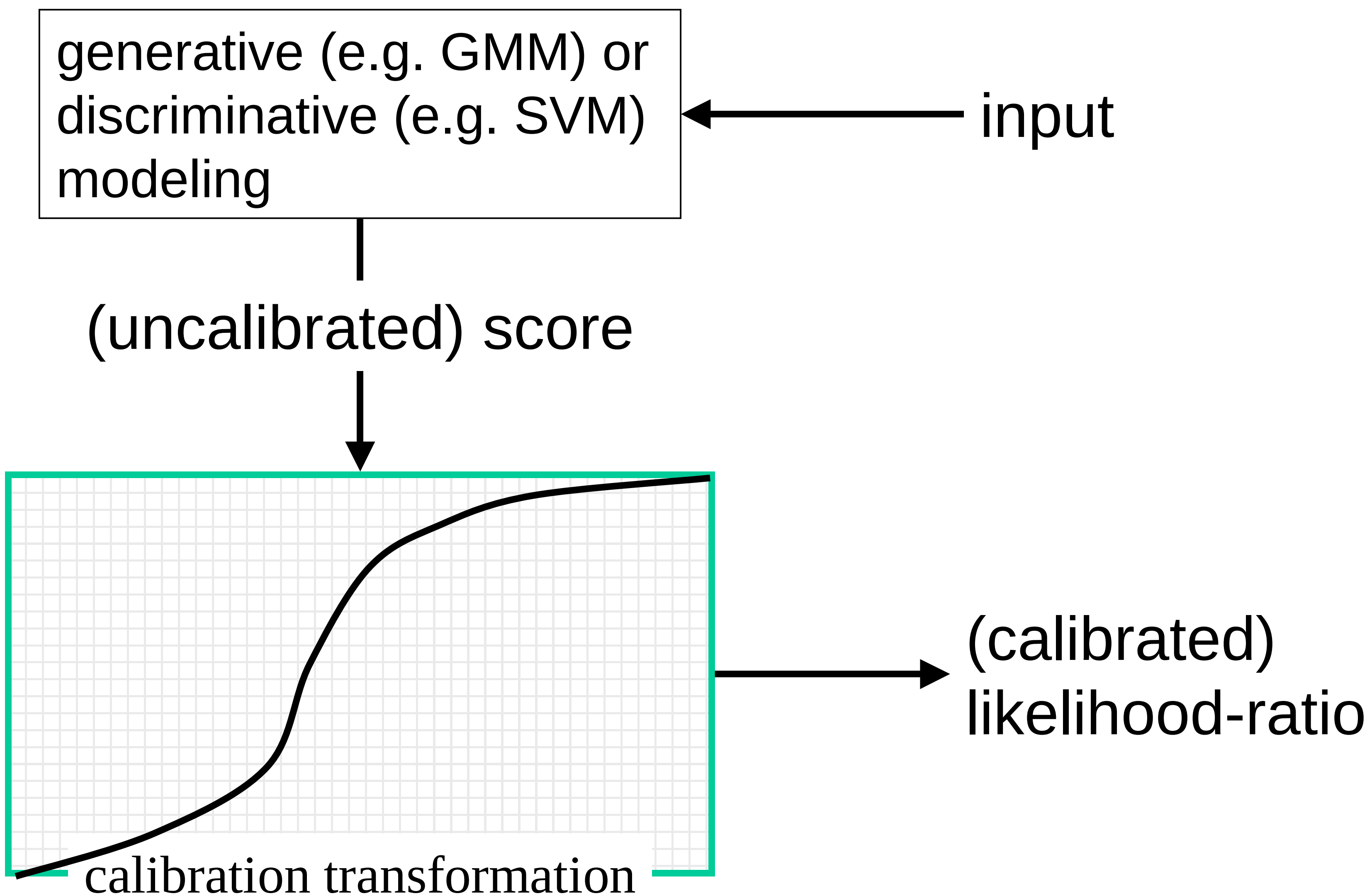

application-*independent*
pattern recognizer
input
class likelihoods
Bayes' Rule
class prior
class posterior
costs
minimize expected
cost of decisions
hard decisions
We want the recognizer to output *likelihoods*.

In the binary (detection) case,
this can be simplified …

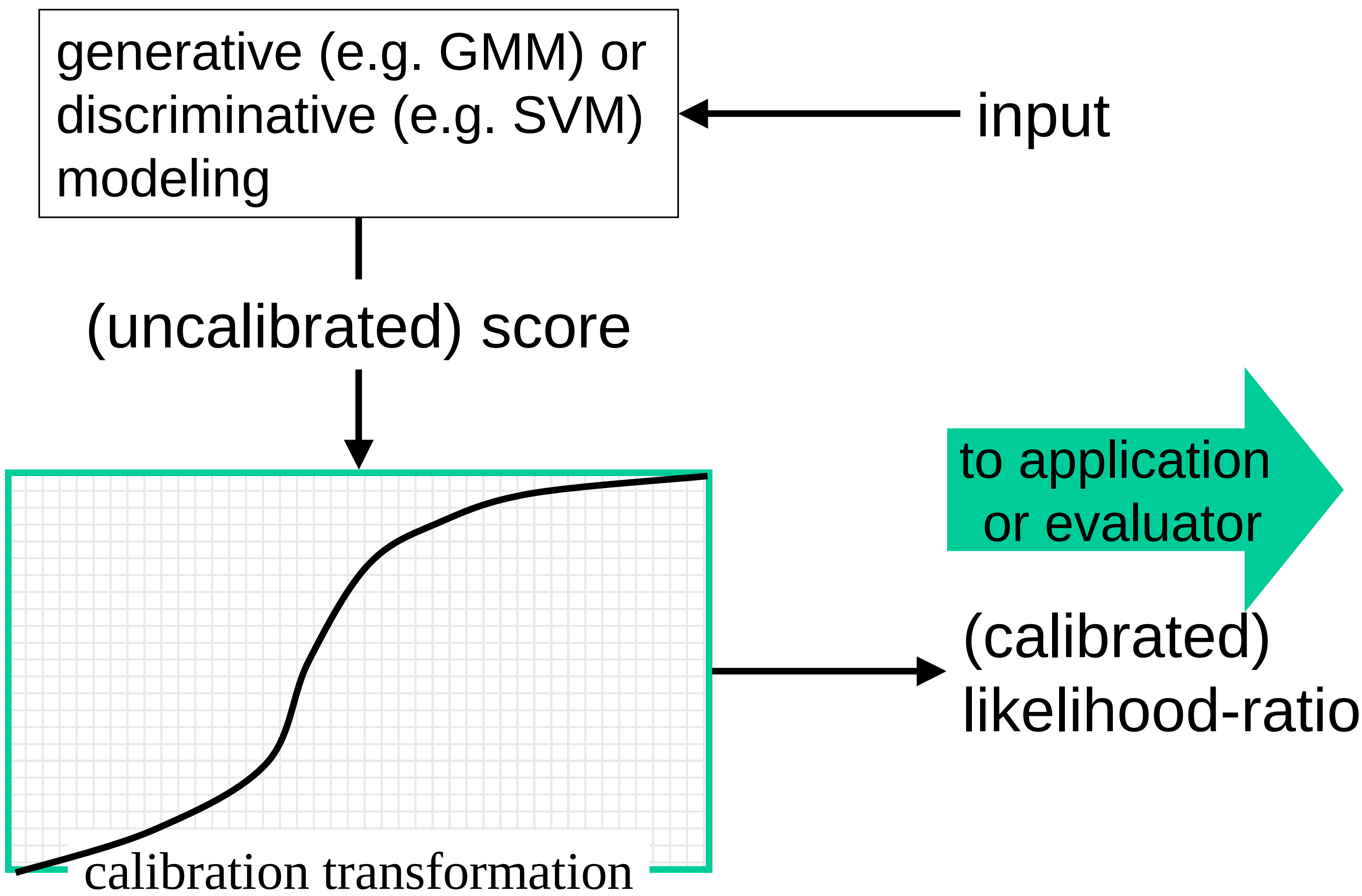


application-*independent* detector
input
likelihood-ratio
$$\frac{P(\text{score}\,|\,\text{target})}{P(\text{score}\,|\,\text{non-target})}$$
compare to threshold
compute Bayes threshold
accept / reject
prior, costs

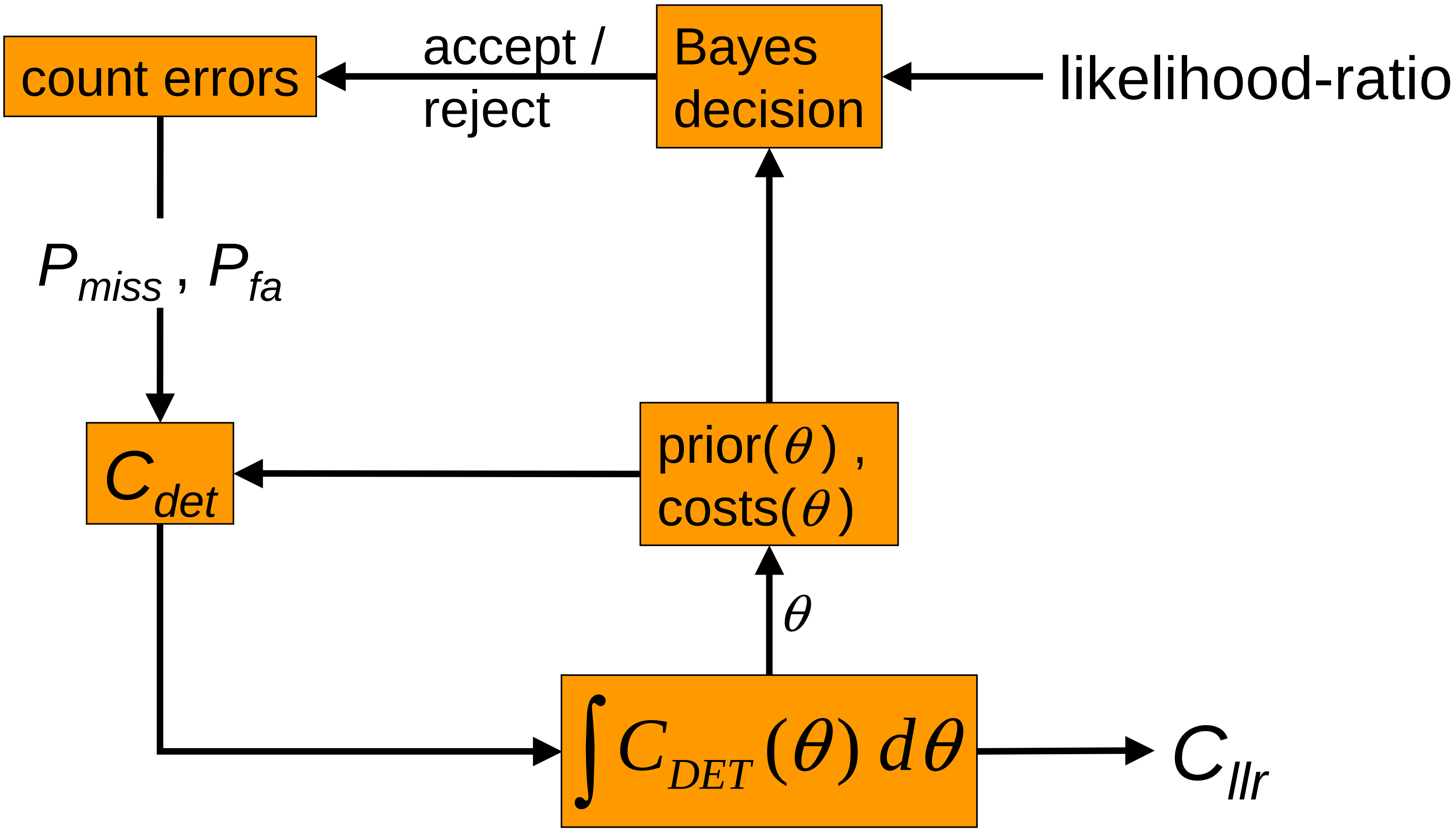
What detector (evaluee) does



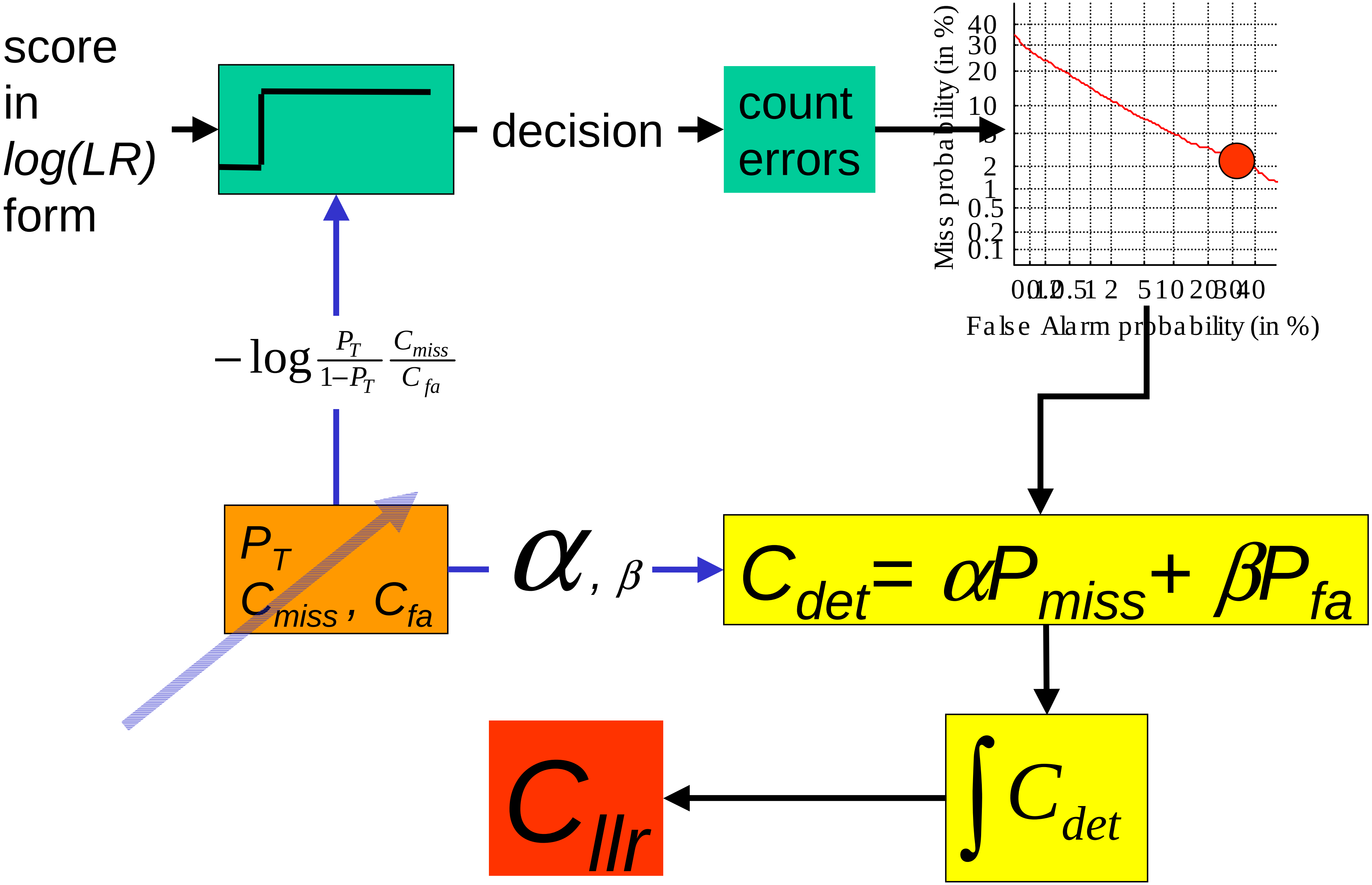
What detector (evaluee) does

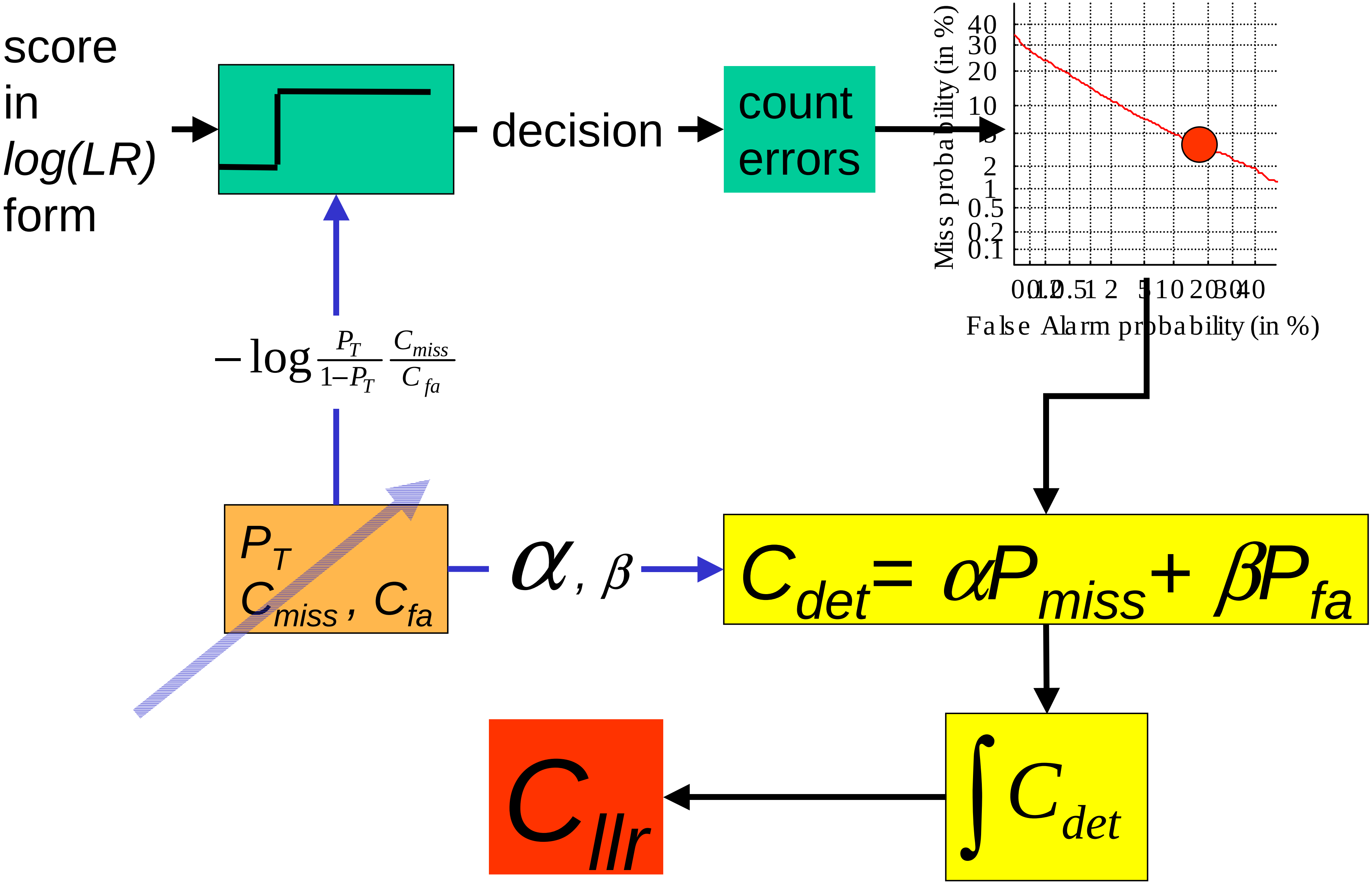


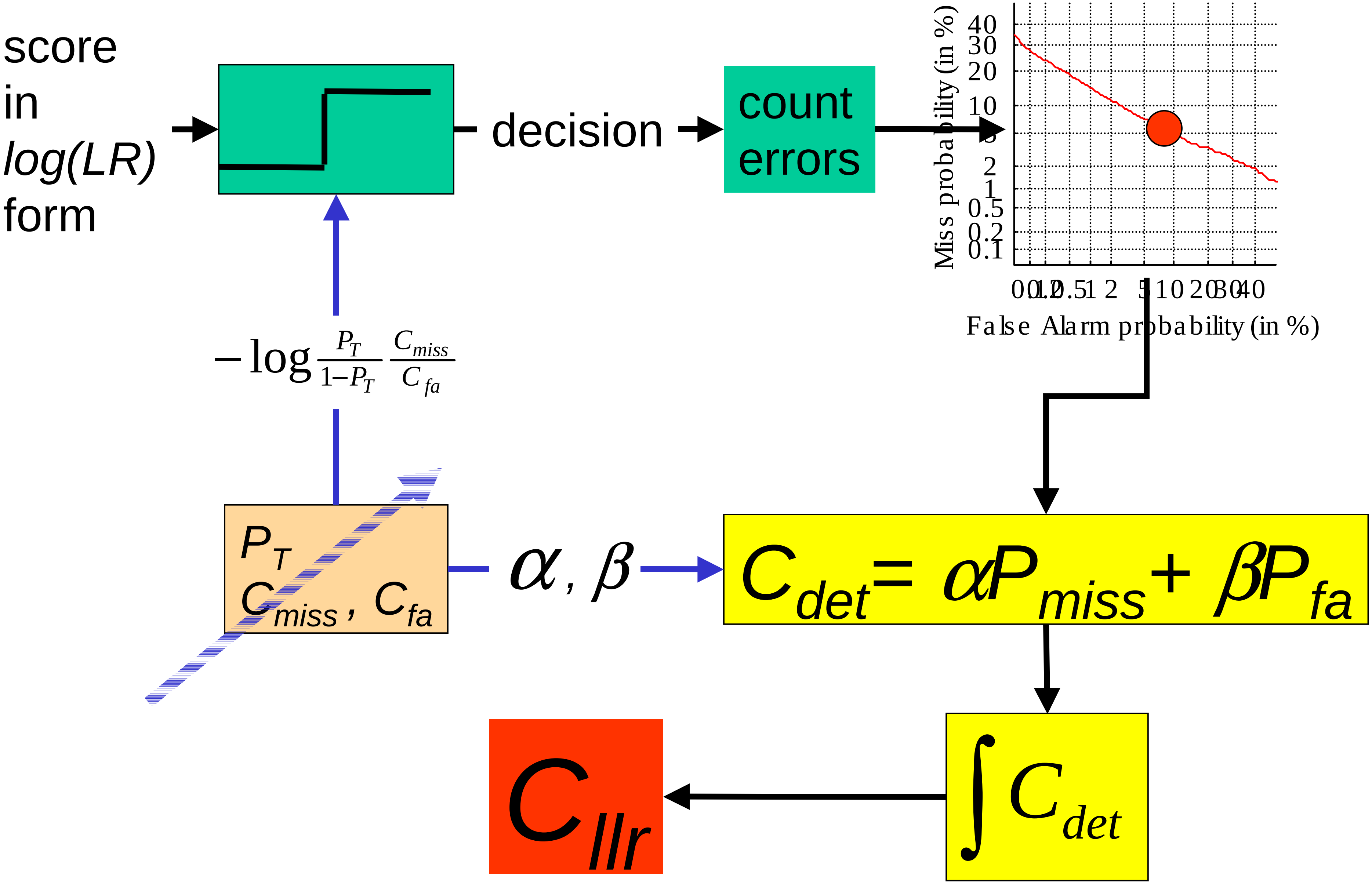
What evaluator does

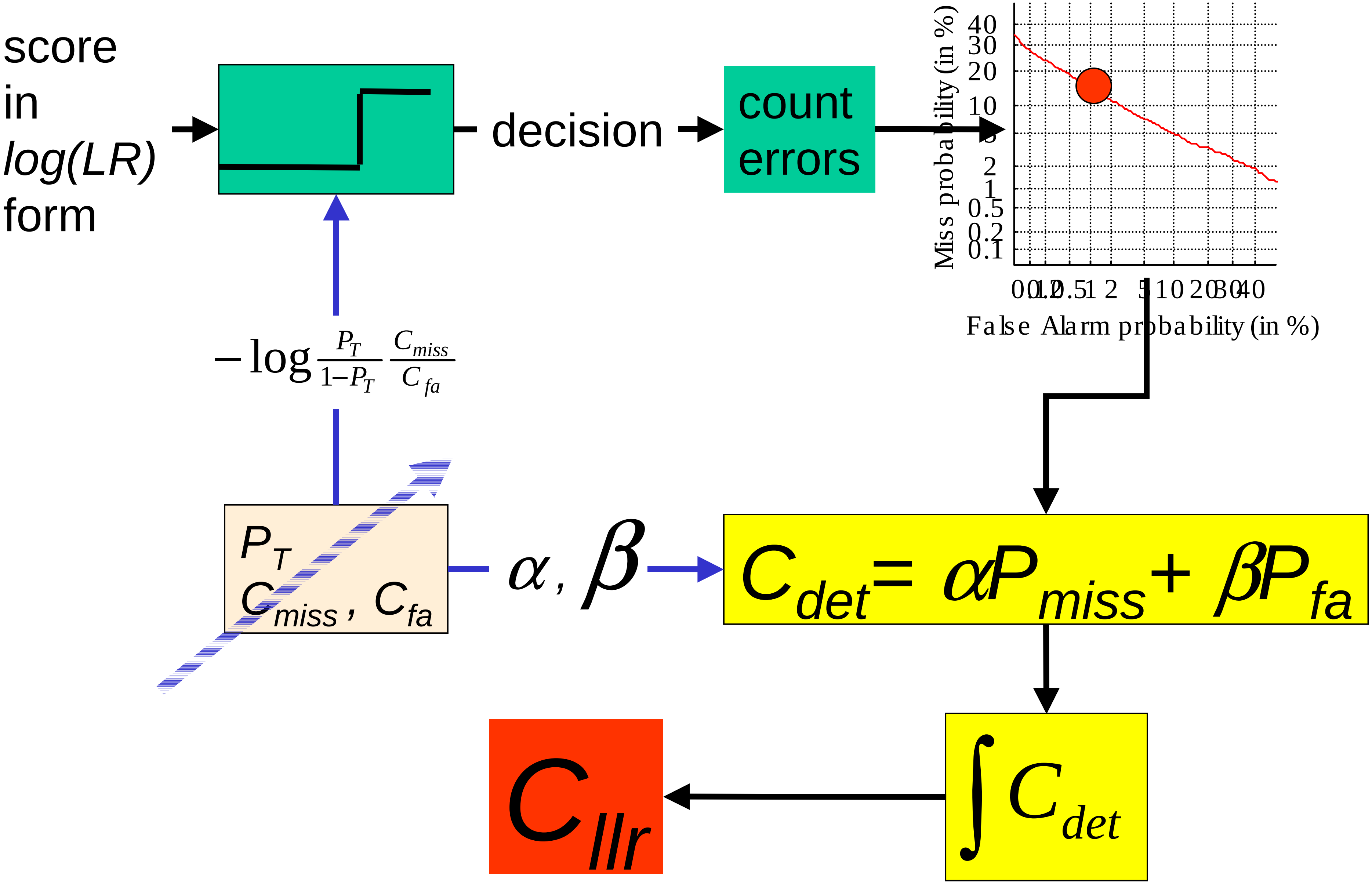


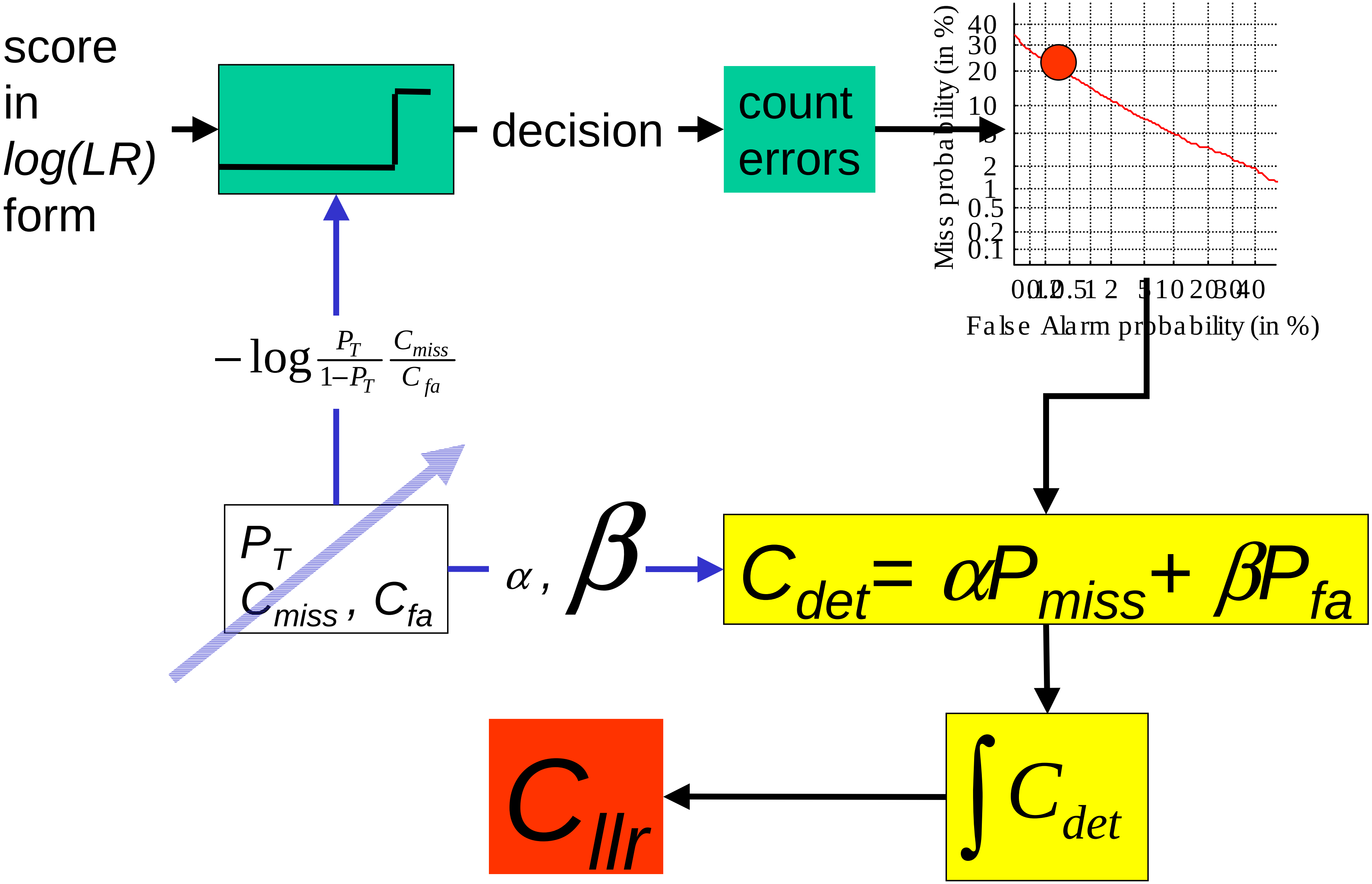
Here is another view …


score
in
log(LR)
form
decision
count
errors
40
30
20
10
2
1
0.5
0.2
0.1
Miss probability (in %)
False Alarm probability (in %)
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{det} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{det}$
C_{llr}


score
in
log(LR)
form
decision
count
errors
Miss probability (in %)
False Alarm probability (in %)
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{det} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{det}$
C_{llr}


score in *log(LR)* form
decision
count errors
Miss probability (in %)
False Alarm probability (in %)
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{det} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{det}$
C_{llr}


score
in
log(LR)
form
decision
count
errors
Miss probability (in %)
False Alarm probability (in %)
$-\log \frac{P_T}{1-P_T}\frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{det} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{det}$
C_{llr}


score
in
log(LR)
form
decision
count
errors
Miss probability (in %)
False Alarm probability (in %)
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{det} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{det}$
C_{llr}

Questions

1. How does evaluee set thresholds?

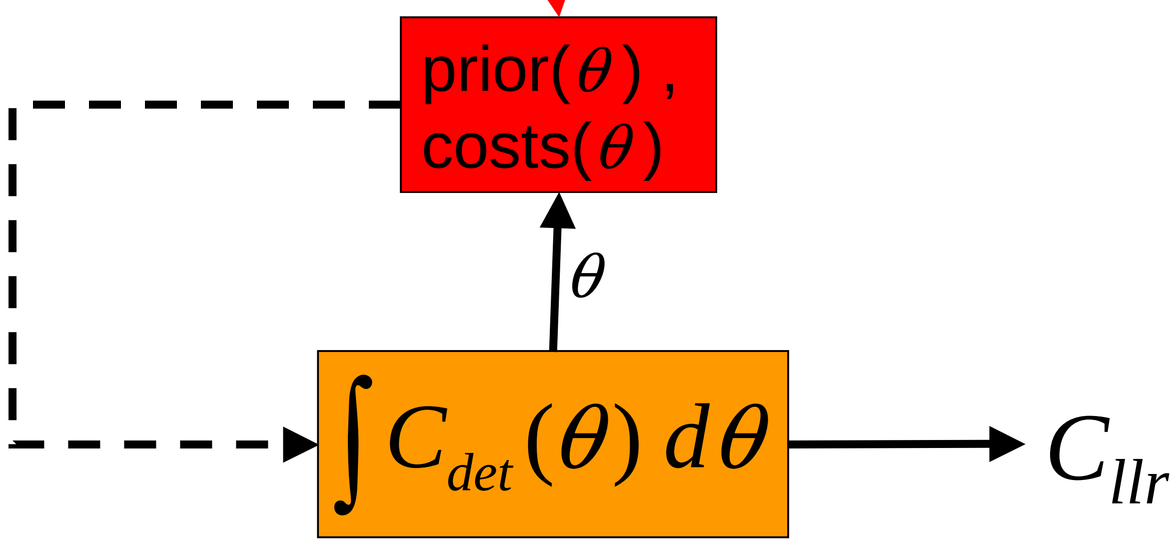
2. **How does the evaluator perform the integral?**

How does the evaluator perform the integral?

$$\boxed{\int C_{det}(\theta)\, d\theta} \longrightarrow C_{llr}$$

How does the evaluator perform the integral?

We have to choose *appropriate* ways to vary prior and cost as a function of θ.



Choice of functions: prior(θ) and costs(θ)

There exist choices for these functions, so that:

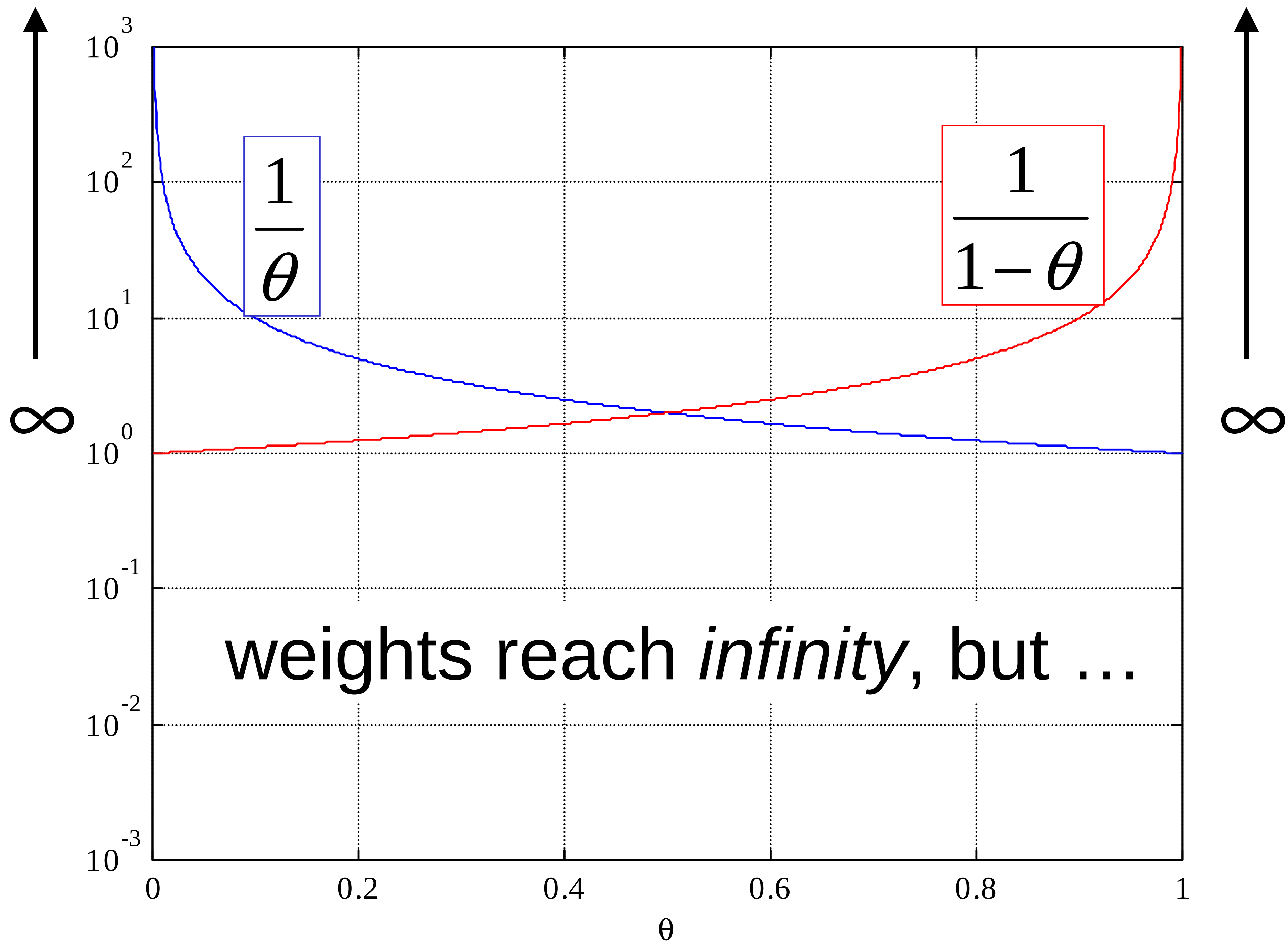
- Integral is solved analytically (easy to compute).
- C_{llr} represents recognizer performance over a *wide* range of applications.
- C_{llr} has intuitive information-theoretic interpretation (cross-entropy).
- C_{llr} serves as good numerical optimization objective function (logistic regression).

The magic formula:

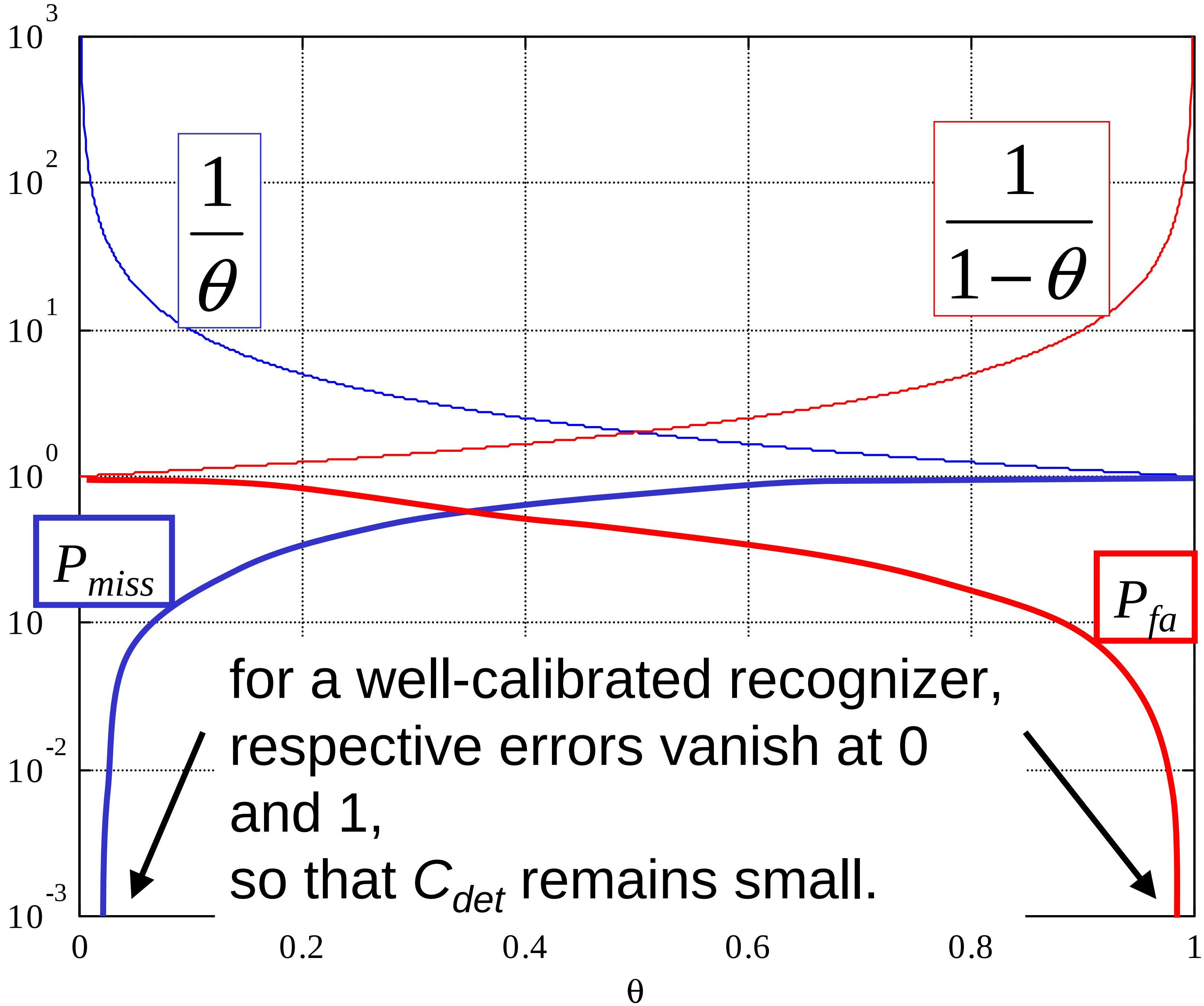
$$P_{tar} C_{miss} = \frac{1}{\theta}, \quad (1 - P_{tar}) C_{fa} = \frac{1}{1-\theta}$$

$$0 \le \theta \le 1$$

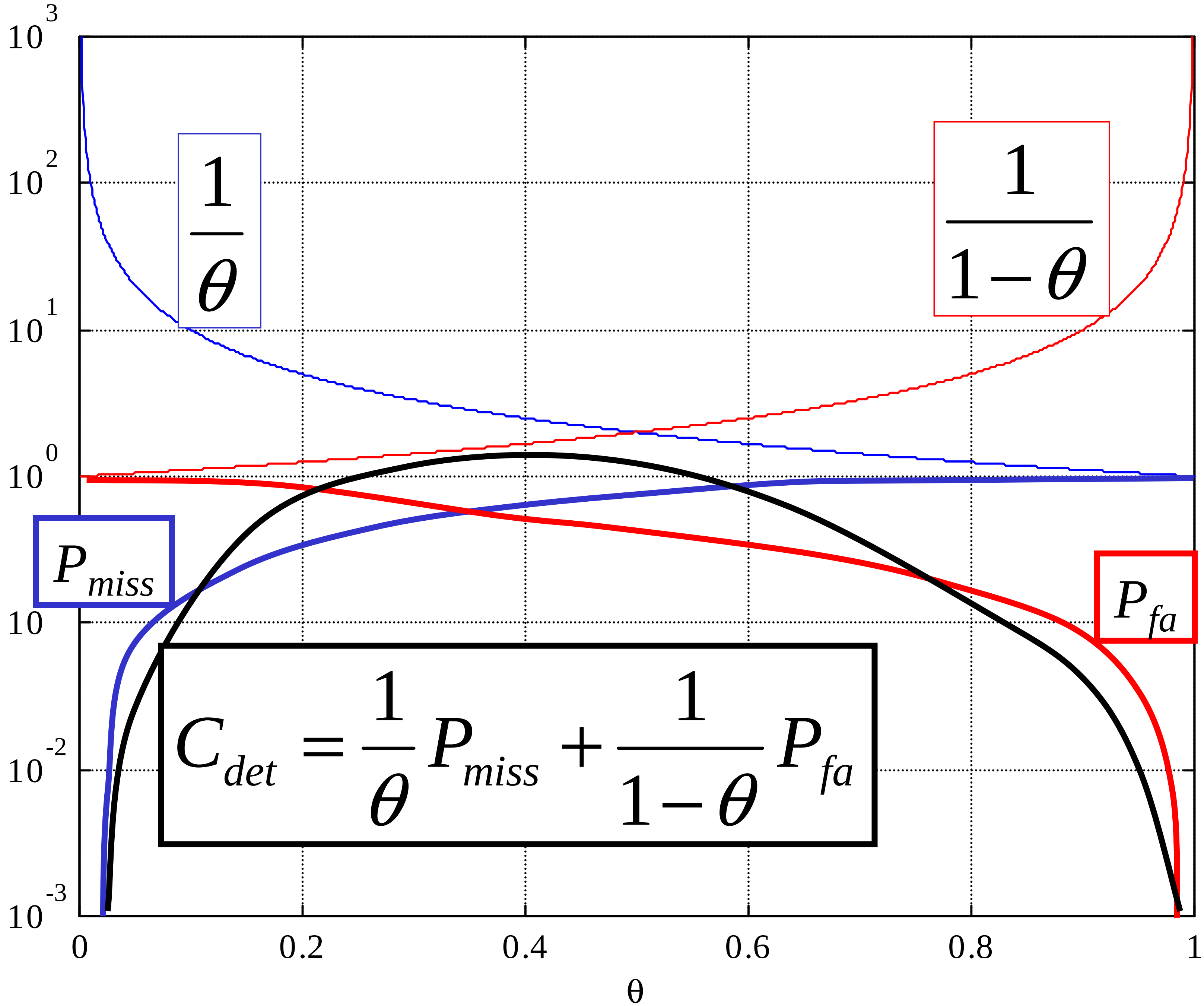
Notice *infinities* at $\theta = 0$ and $\theta = 1$. This is good! We want C_{llr} to represent a wide range of applications, including those with very high misclassification costs.



$\frac{1}{\theta}$
$\frac{1}{1-\theta}$
∞
∞
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
0.2
0.4
0.6
0.8
1
θ
weights reach *infinity*, but …



10^3
10^2
10^1
10^0
10
10^-2
10^-3
0
0.2
0.4
0.6
0.8
1
θ
1/θ
1/(1−θ)
P_miss
P_fa
for a well-calibrated recognizer,
respective errors vanish at 0
and 1,
so that C_det remains small.



10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
0
0.2
0.4
0.6
0.8
1
θ
$\frac{1}{\theta}$
$\frac{1}{1-\theta}$
P_{miss}
P_{fa}
$C_{det} = \frac{1}{\theta} P_{miss} + \frac{1}{1-\theta} P_{fa}$

This gives:

$$\boxed{C_{llr} = k \int_0^1 \frac{1}{\theta} P_{miss}(\theta) + \frac{1}{1-\theta} P_{fa}(\theta)\, d\theta}$$

- Ok, this looks like a nice integral, but how does one compute it?
- and why is it called C_{llr}?

Why is it called C_{llr} ?

C_{llr} is a cost function to evaluate detector scores in ***l**og-**l**ikelihood-**r**atio* format.

(For practical reasons ***log***-likelihood-ratio format is better than likelihood-ratio format.)

How to compute C_{llr}

- Computation of C_{llr} from a supervised evaluation database is just as easy as computation of error-rates, or C_{det} .

How to compute C_{llr}

$$\boxed{C_{llr} = \frac{1}{2\|S_T\|}\sum_{t\in S_T}\log_2\left(1+\exp(-llr_t)\right) + \frac{1}{2\|S_N\|}\sum_{t\in S_N}\log_2\left(1+\exp(llr_t)\right)}$$

S_T is the set of target trials
S_N is the set of non-target trials
llr_t is the *log-likelihood-ratio* under evaluation for trial t

Properties of C_{llr}

- $C_{llr} = 0$ (perfect): $llr = +\infty$ for every target and $llr = -\infty$ for every non-target.
- $0 < C_{llr} < 1$ (useful): well-calibrated, real-world detector.
- $C_{llr} = 1$ (reference): well-calibrated but *useless,* gives no discrimination, outputs $llr = 0$ for every trial.
- $1 < C_{llr} \leq \infty$ (badly calibrated): makes worse decisions than not using any detector at all (i.e. worse than reference detector).

Information-theoretic interpretation

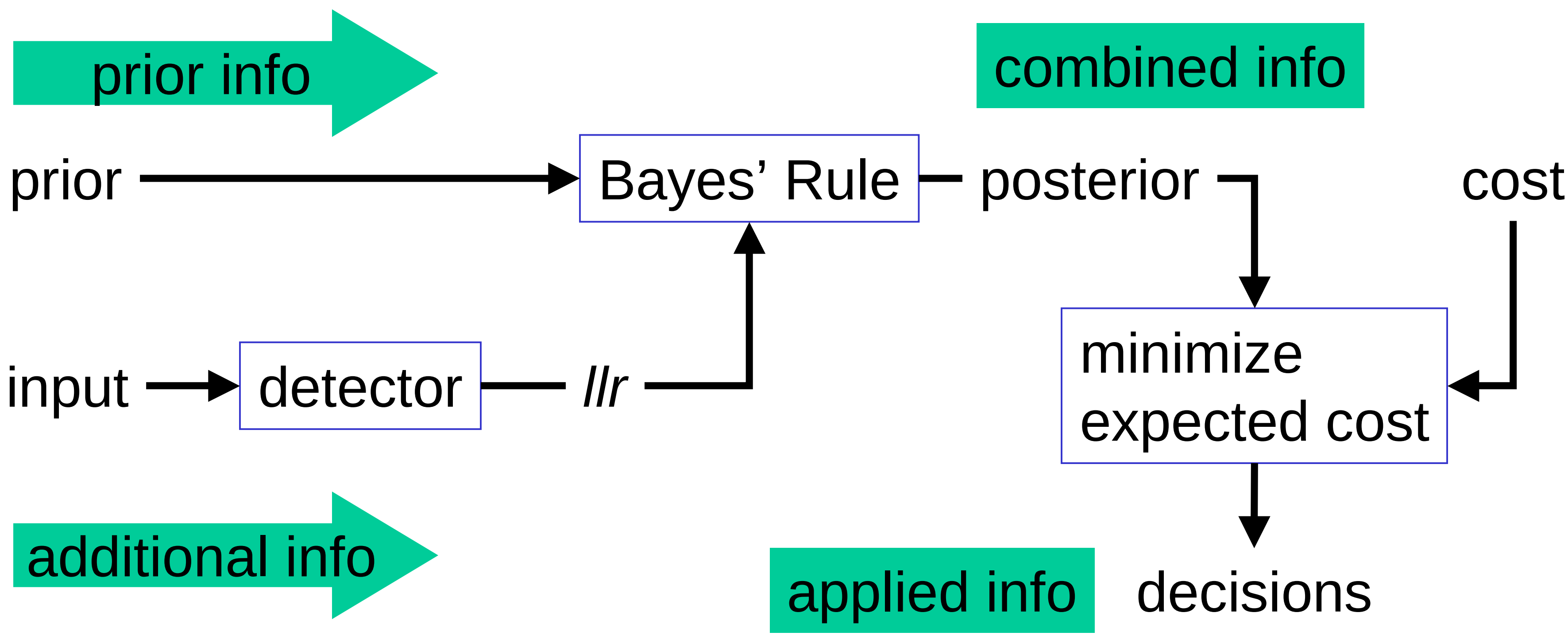
C_{llr} can be shown to be equivalent to an *empirical cross-entropy,* which gives the effective amount of *information* (in the sense of Shannon's Information Theory) that the detector delivers to the user.

Information-theoretic interpretation

The target *prior* gives *information* about the presence of the target, e.g.:

- If $P_{tar} = 1$, we know the target is there. This is *1 bit* of information.
- If $P_{tar} = 0$, we know the target is not there. This is also *1 bit* of information.
- If $P_{tar} = 0.5$, this gives least information, namely *0 bits*.

A detector that outputs a *llr* score gives *additional information* about the presence of the target, which (if well-calibrated) can be optimally combined (via Bayes' Rule) with the prior information.



Information-theoretic interpretation

The *complement*, $1 - C_{llr}$ measures the average amount of additional information (in bits per trial) contributed by the *llr* scores of the detector, when there is least prior information: $P_{tar} = 0.5$.

- Note, if the detector is badly calibrated, then

$$1 - C_{llr} < 0,$$

(negative amount of info!) with the interpretation that the information is *misleading* and would lead to bad decisions.

C_{llr} and Forensics

Daniel Ramos and others have done much work to motivate that measures based on C_{llr} are suitable for evaluating the quality of detection likelihood-ratios, when likelihood-ratios are used as evidence in Forensic Speaker Recognition.

C_{llr} as numerical optimization objective

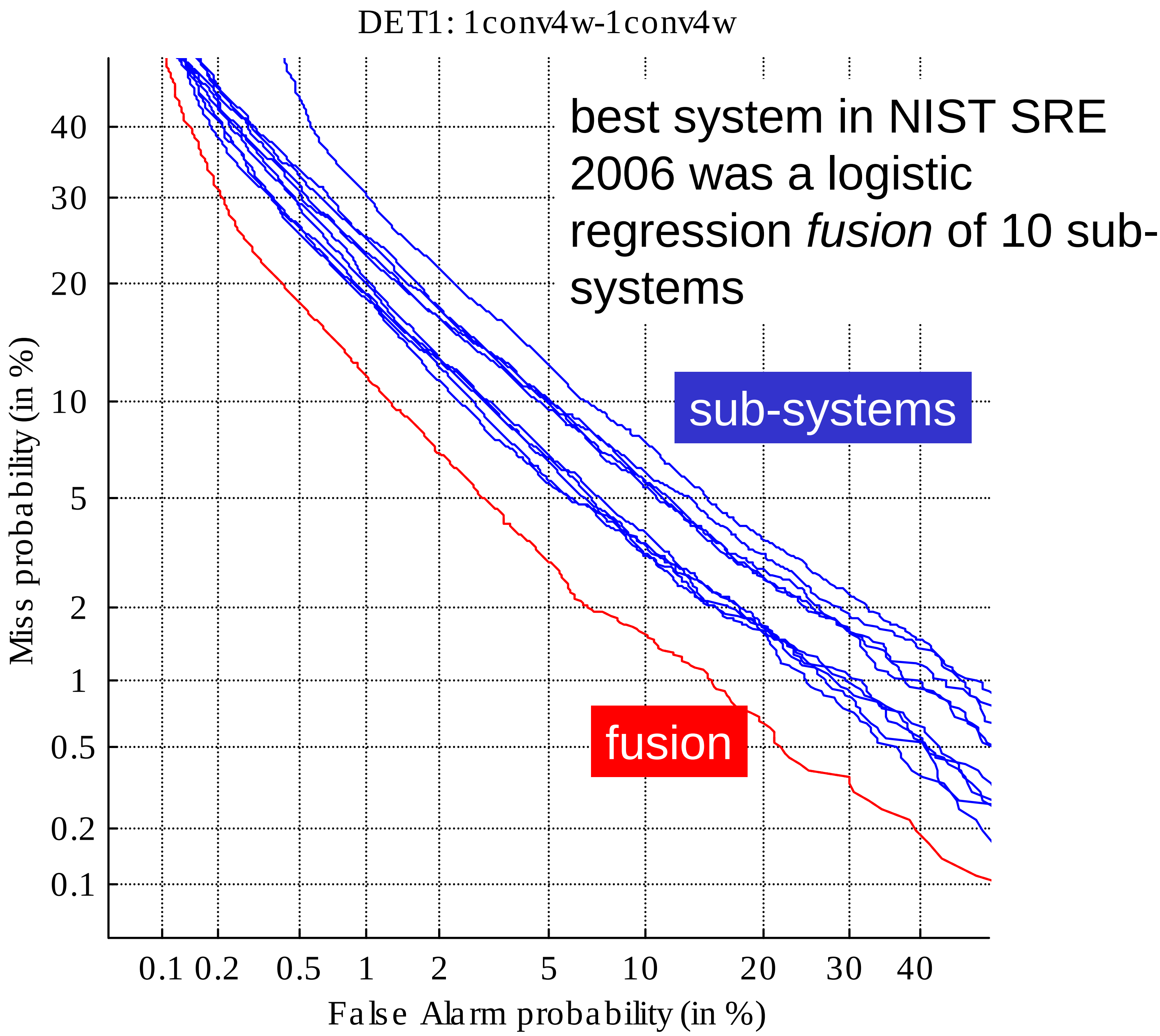
Numerically optimizing C_{llr} is just a form of the well-known *logistic regression*. It is an attractive optimization objective because:

- It tends to lead to a convex optimization surface with a unique optimum.
- Efficient algorithms, like conjugate gradient can be used to find the optimum.

C_{llr} as numerical optimization objective

Logistic regression can be used to perform supervised training of the parameters of:

- a *calibration* stage for any existing binary recognizer that outputs a score, or
- a *fusion* of multiple speaker recognition sub-systems to give a single, well-calibrated and more accurate output.


DET1: 1conv4w-1conv4w
best system in NIST SRE 2006 was a logistic regression *fusion* of 10 sub-systems
sub-systems
fusion
Miss probability (in %)
40
30
20
10
5
2
1
0.5
0.2
0.1
False Alarm probability (in %)
0.1
0.2
0.5
1
2
5
10
20
30
40

C_{llr} : Adopted by others

- C_{llr} has been the basis of further publications by MIT Lincoln Lab (USA), ATVS-UAM (Spain), TNO (Netherlands) and others;
- has been adopted by NIST for use as evaluation metric in both speaker detection (2006,2008) and language detection (2007);
- was used (as numerical optimization objective) by 5 of the best-performing teams at the last NIST Speaker Recognition Evaluation (2006).

C_{llr} tools

FoCal: Tools for Fusion and Calibration

- Free MATLAB toolkit.
- Applicable to *binary* pattern recognizers.
- Evaluation with C_{llr}
  - including graphical calibration/discrimination decompositions (APE-curves).
- Calibration and Fusion with logistic regression.

See: www.dsp.sun.ac.za/~nbrummer/focal

Contents

Part I

1. Introduction
2. Good old error-rate
3. Binary case:

   Error-rate $\rightarrow$ ROC $\rightarrow$ C_{det} $\rightarrow$ C_{llr}

Part II: Multiclass

Error-rate $\rightarrow$ ~~ROC~~ $\rightarrow$ C_{miss} $\rightarrow$ C_{mxe}

Part II: Multiclass

1. What we want to do
2. Why cost and error-rate don’t work.
3. How NIST did it.
4. How we propose to do it.
5. Experimental demonstration of our proposal.

1. What we want to do

To create an evaluation criterion that is:

- application-independent,
- sensitive to calibration, and
- useful as numerical optimization objective

2. Why cost and error-rate don't work.

- Average error-rate is
  - *application-dependent*
  - increases with N
- Conditional error-rate analysis
  - ROC is ill-defined and computationally problematic
  - *does not evaluate calibration*
- Misclassification Cost Functions
  - *application-dependent*
  - complexity increases as N^2
- None of the above give good numerical optimization objectives.

3. How NIST did it.

- NIST’s Language Recognition Evaluation (LRE) is a multiclass pattern recognition problem (14 languages in 2007).
- NIST presented it as 14 different, one-against-the-rest *detection* tasks, with the evaluation criterion being average detection cost over all 14.
- This approach has both good and bad consequences:

Advantages of LRE strategy

- LRE'07 averaged over 14 *different* detection tasks. This encouraged *some* application-independence in the resulting recognizers.
- The evaluation is calibration-sensitive.

Disadvantages of LRE strategy

Casting language recognition in the mold of a detection task gives it some of the attributes of a binary recognition problem.

- But treating it as binary pattern recognition task, has contributed to 2 significant problems. (Both problems arose because the 14 detection tasks are *not* independent, but were treated as such.)

Problems induced by rotated language detection:

1. Pooling of scores across targets for ROC analysis produces *meaningless* results. (many researchers did this, me too.)

Problems induced by rotated language detection:

2. Sub-optimal calibration strategies were used by several teams:
   - *pooling* scores across targets and then attempting a global 2-class calibration

     (A very bad idea---see experiments below!)
   - calibrating *separate* detectors for every target (Suboptimal compared to global multiclass calibration---see experiments below.)

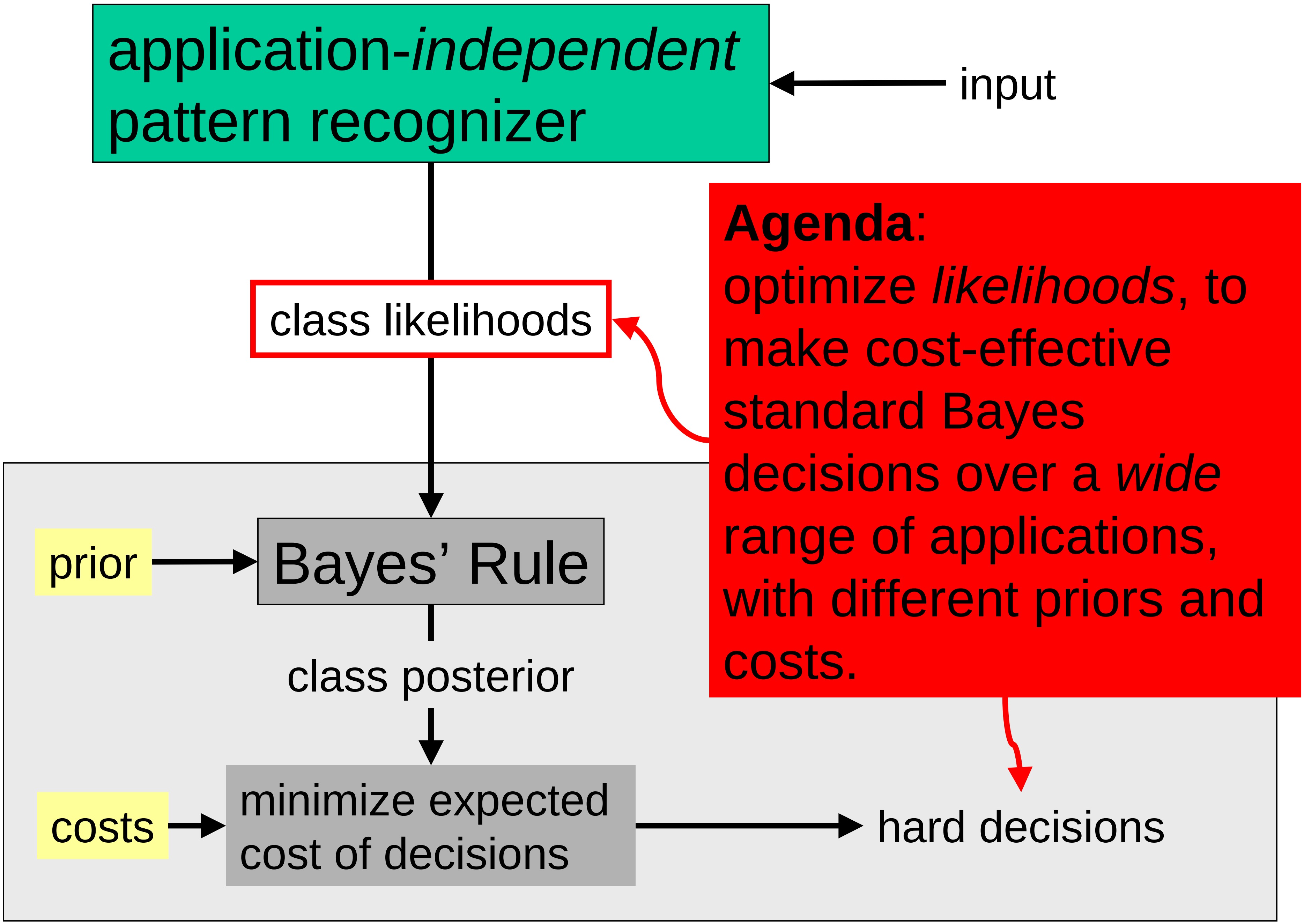
Back to our own agenda …

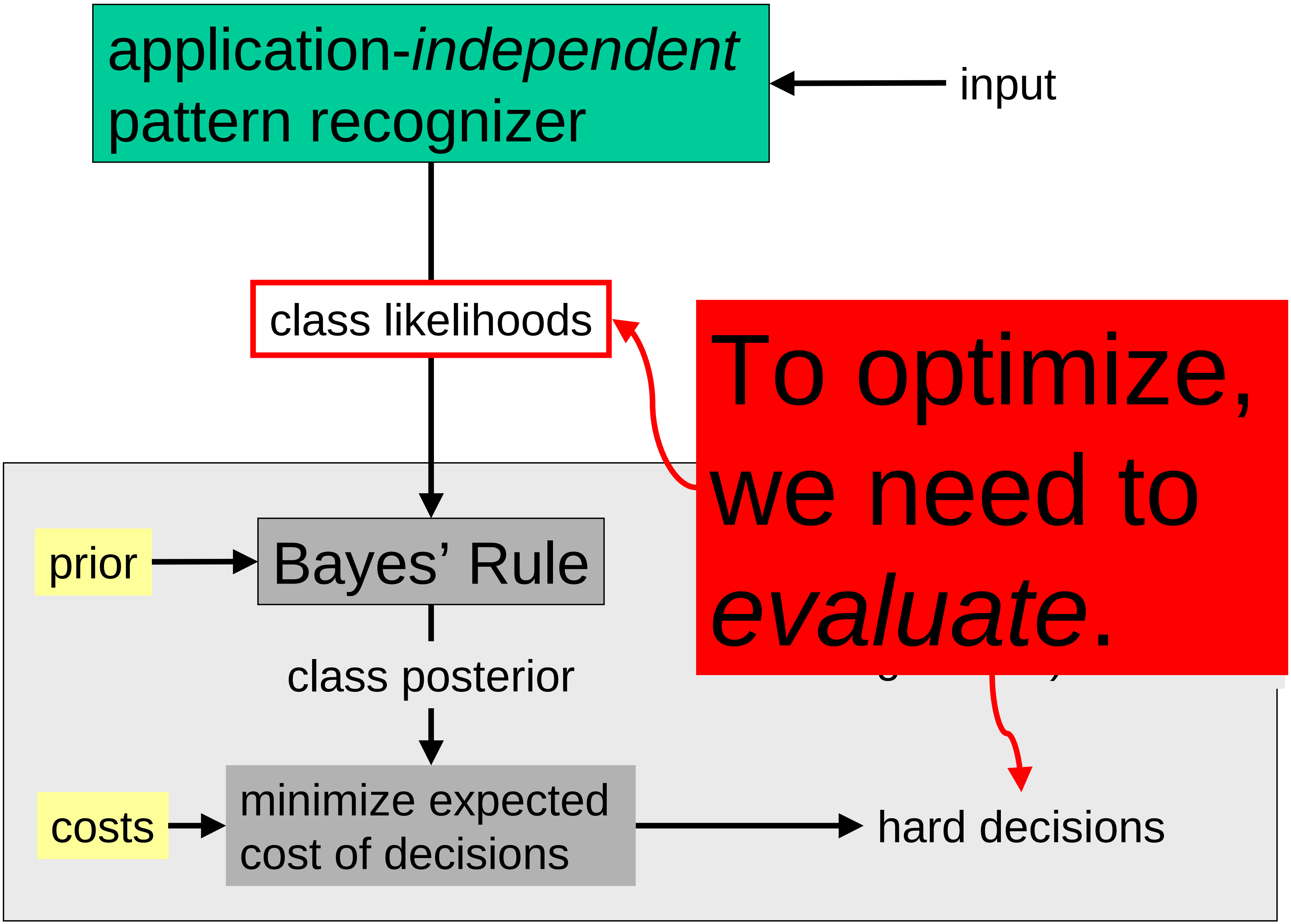
- Let us return to treating language recognition as a full multiclass problem and to solving the calibration problem in a way which is as application-independent as possible.
- Then we can solve not only language detection tasks, but also many other recognition tasks.

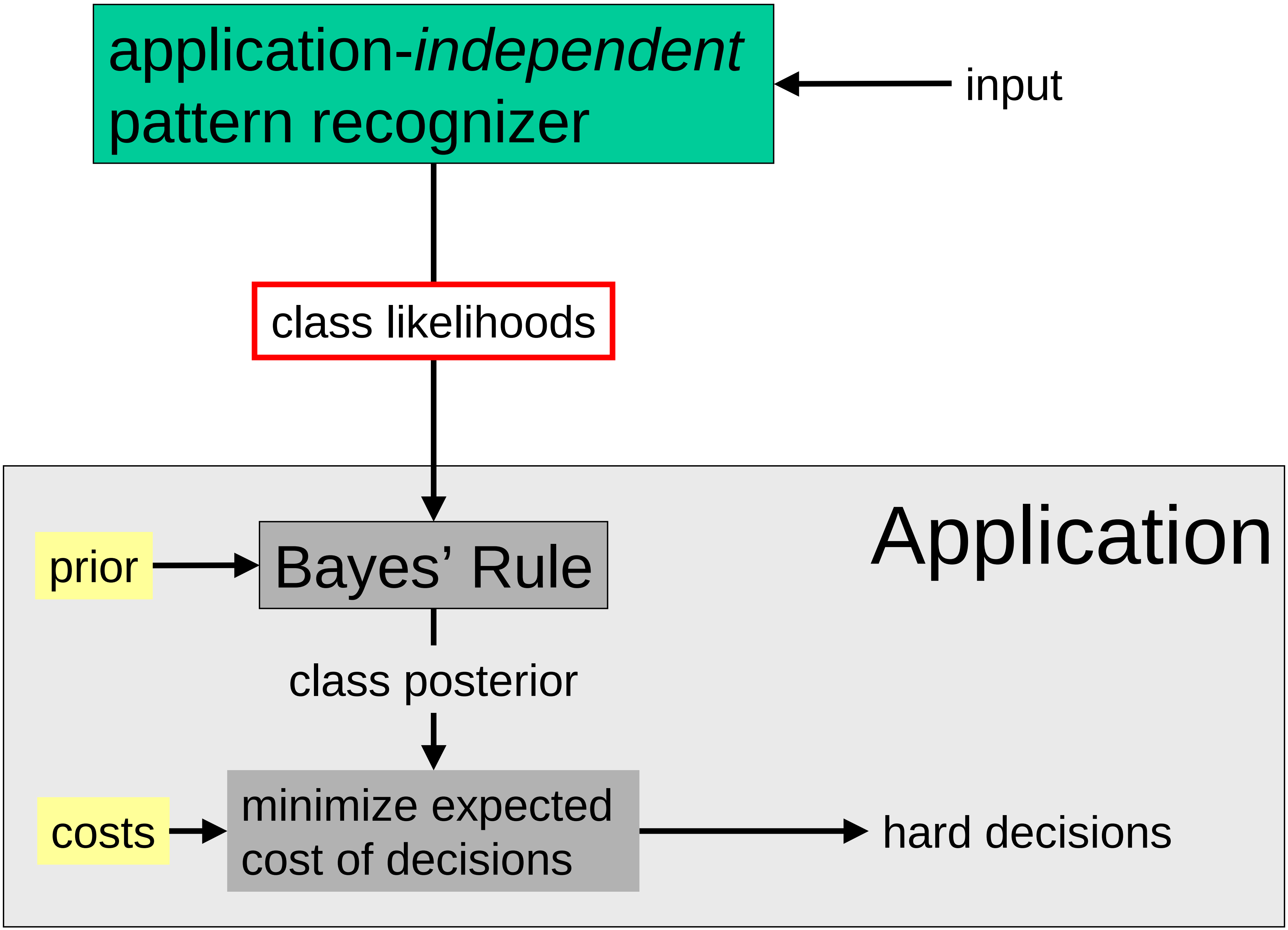
Part II: Multiclass

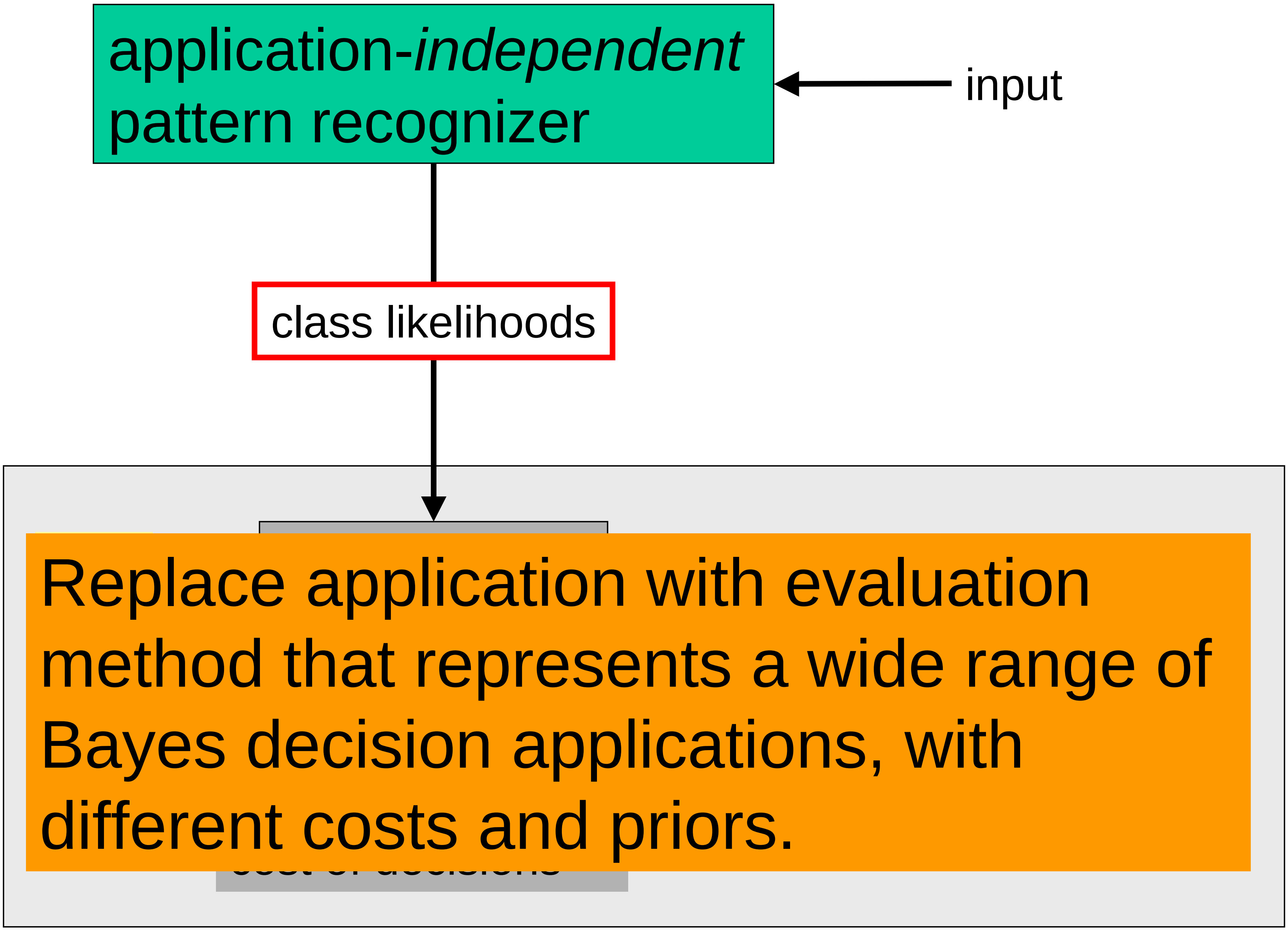
1. What we want to do
2. Why cost and error-rate don't work.
3. How NIST did it.
4. **How we propose to do it.**
5. Experimental demonstration of our proposal.

Let’s recapitulate our application-independent recipe.


application-*independent*
pattern recognizer
input
class likelihoods
Agenda:
optimize *likelihoods*, to make cost-effective standard Bayes decisions over a *wide* range of applications, with different priors and costs.
prior
Bayes' Rule
class posterior
costs
minimize expected cost of decisions
hard decisions


application-*independent*
pattern recognizer
input
class likelihoods
To optimize,
we need to
evaluate.
prior
Bayes' Rule
class posterior
costs
minimize expected
cost of decisions
hard decisions


application-*independent*
pattern recognizer
input
class likelihoods
Application
prior
Bayes' Rule
class posterior
costs
minimize expected
cost of decisions
hard decisions


application-*independent*
pattern recognizer
input
class likelihoods
Replace application with evaluation method that represents a wide range of Bayes decision applications, with different costs and priors.

Evaluation over different priors and costs

To understand how to vary the application, we need to understand how to vary the *prior* and *costs*.

Prior

$$P_i = P(\text{class } i), \quad \sum_{i=1}^{N} P_i = 1$$

The prior can be varied inside an (*N-1*) dimensional simplex.

Multiclass Cost Functions:
2-minute tutorial

(In which many important and interesting facts are ignored.)

Cost Function Complexity

There are $N^2 - N$ different types of misclassification error, all of which could have different costs.

There can be $N^2 - N$ different cost coefficients.

		estimated class			
		1	2	…	N
true class	1		C_{12}	…	C_{1N}
	2	C_{21}		…	C_{2N}
	…	…	…	…	…
	N	C_{N1}	C_{N2}	…	

We will instead use a *simplified* cost function:

Simplified cost function

Let the cost be:

- Dependent on the true class,
- but independent of the estimated class:

$C_{miss}(i)$ is the cost of *missing* true class i when misclassifying it as *any* other class.

true class \ estimated class	1	2	…	N
1		$C_{miss}(1)$	…	$C_{miss}(1)$
2	$C_{miss}(2)$		…	$C_{miss}(2)$
…	…	…		…
N	$C_{miss}(N)$	$C_{miss}(N)$	…	

There are N different types of *misses*, all of which could have different costs.

Simplified cost function:
Expected miss cost

$$\boxed{C_{miss} = \sum_{i=1}^{N} P_i C_{miss}(i) P_{miss}(i)}$$

Simplified cost function:
Expected miss cost

$$\boxed{C_{miss} = \sum_{i=1}^{N} P_i C_{miss}(i) P_{miss}(i)}$$

P_i : prior for class *i*.

Simplified cost function: *Expected miss cost*

$$\boxed{C_{miss} = \sum_{i=1}^{N} P_i C_{miss}(i) P_{miss}(i)}$$

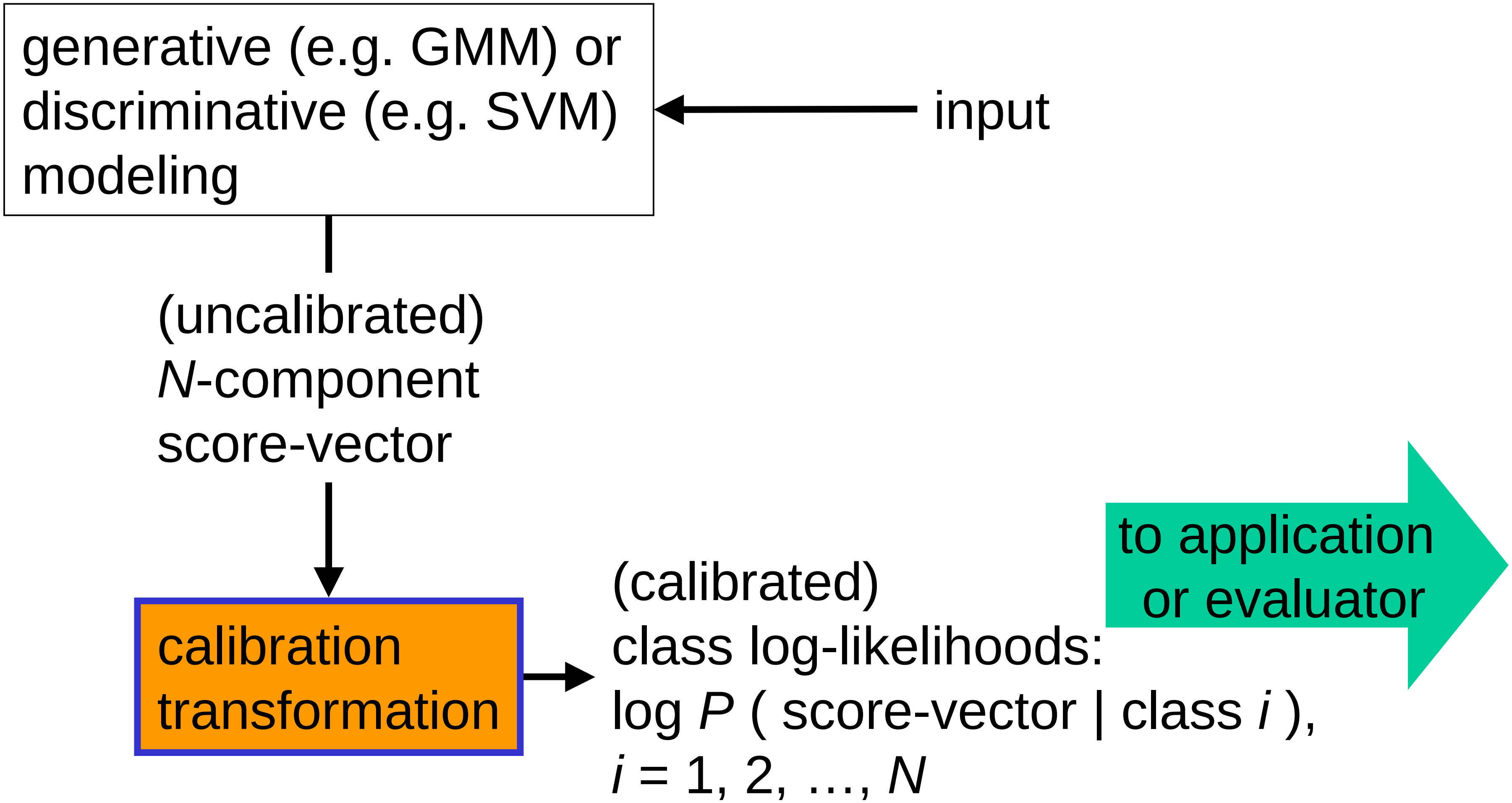
$C_{miss}(i)$: cost of missing class *i*.

Simplified cost function:
Expected miss cost

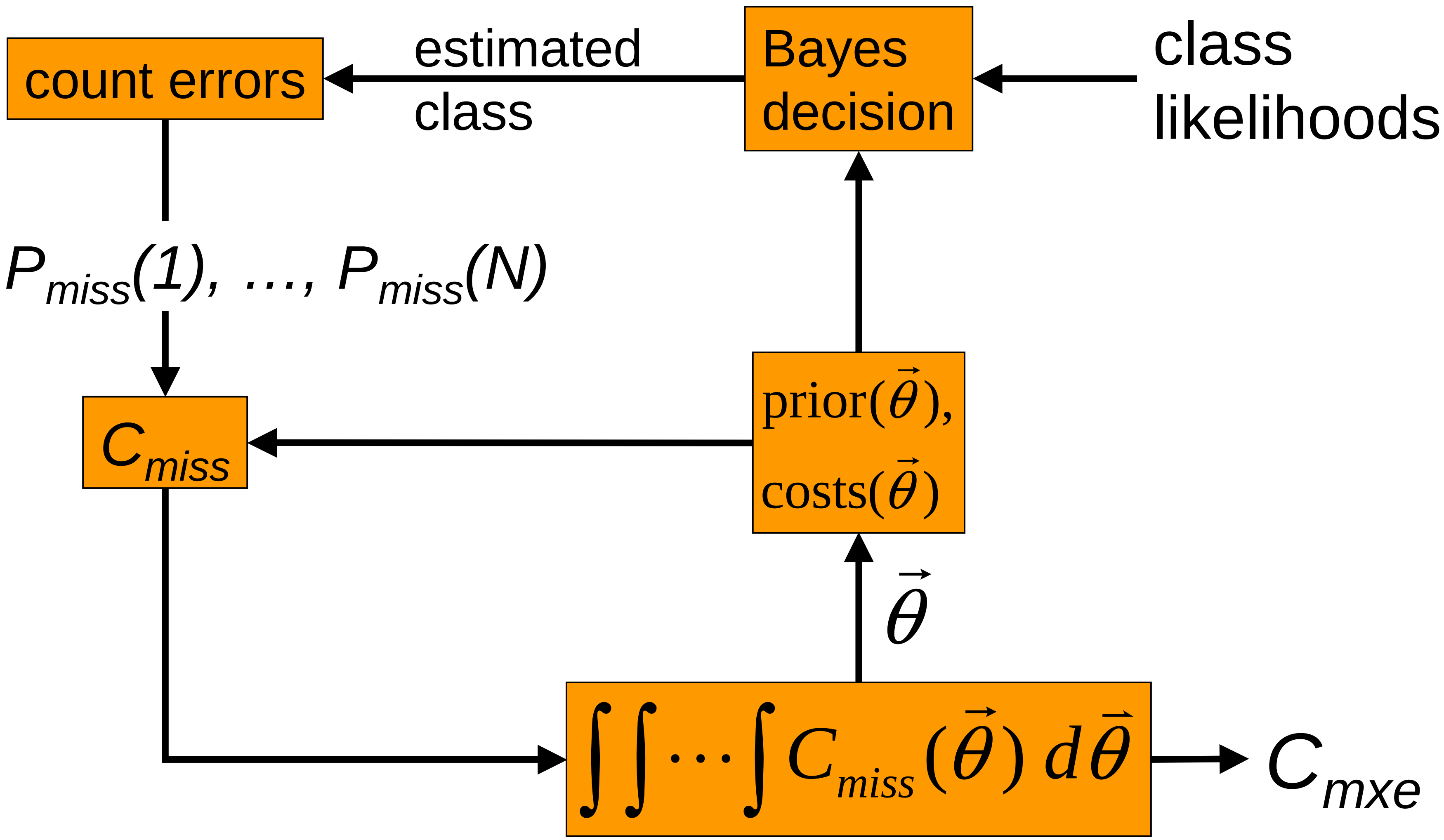
$$\boxed{C_{miss} = \sum_{i=1}^{N} P_i C_{miss}(i) P_{miss}(i)}$$

$P_{miss}(i)$: empirical miss-rate for class i.

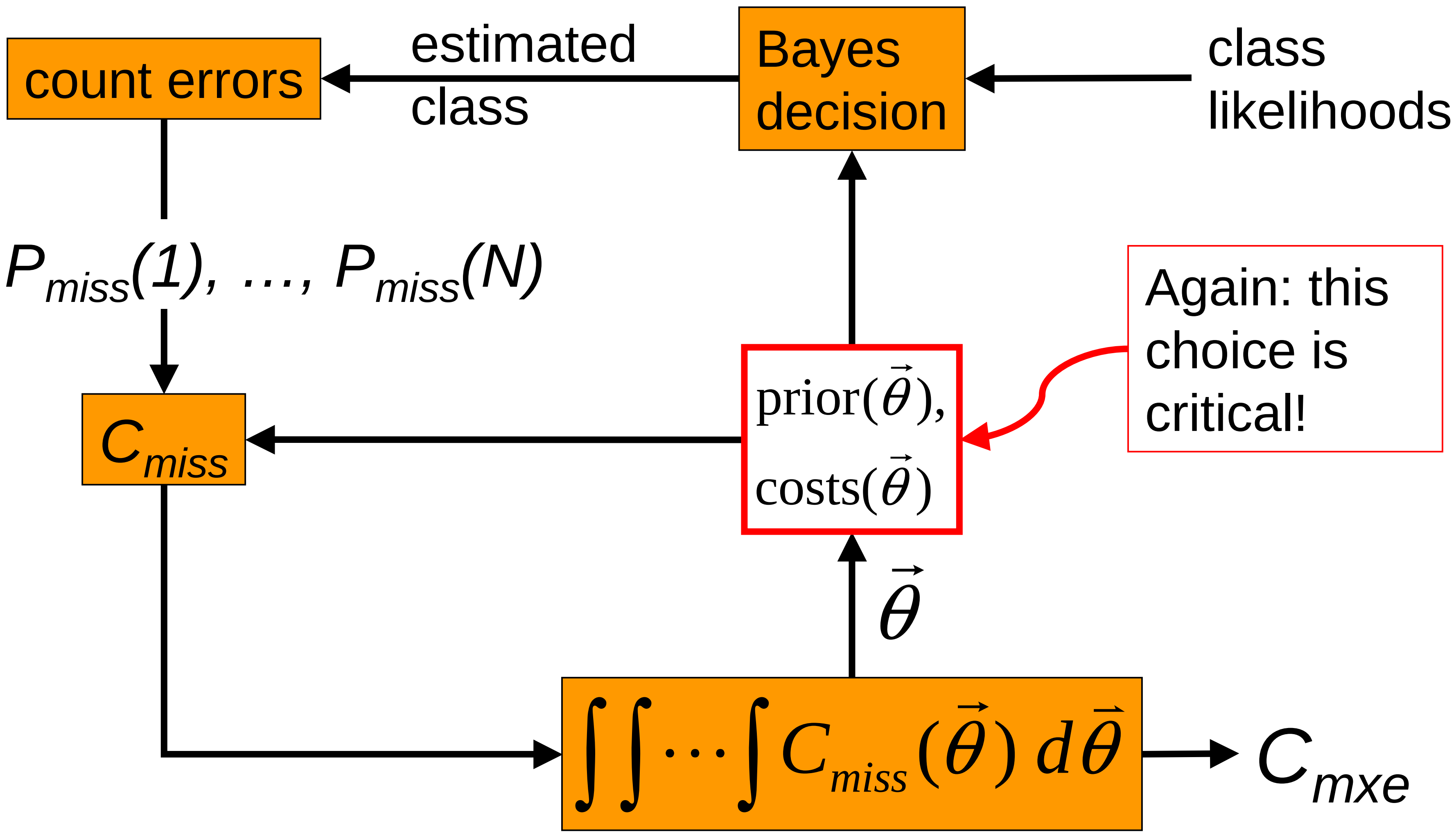
What recognizer (evaluee) does



What evaluator does



What evaluator does



The magic formula:

$$\vec{\theta} : 0 \leq \theta_i \leq 1, \quad \sum_{i=1}^{N} \theta_i = 1$$

$$P_i C_{miss}(i) = \frac{1}{\theta_i}$$

The magic formula:

$$\vec{\theta} : 0 \le \theta_i \le 1, \quad \sum_{i=1}^{N} \theta_i = 1$$

$$P_i C_{miss}(i) = \frac{1}{\theta_i}$$

Note 1: $\vec{\theta}$ is an *N*-vector of parameters, which has the *form* of a probability distribution.

The magic formula:

$$\vec{\theta} : 0 \leq \theta_i \leq 1, \quad \sum_{i=1}^{N} \theta_i = 1$$

$$P_i C_{miss}(i) = \frac{1}{\theta_i}$$

Note 2: We don't need to vary cost and prior separately, because in expected-cost calculations they always act together as *prior-cost products*.

The magic formula:

$$\vec{\theta} : 0 \le \theta_i \le 1, \quad \sum_{i=1}^{N} \theta_i = 1$$

$$P_i C_{miss}(i) = \frac{1}{\theta_i}$$

Note 3: Notice again the *infinities* at the edges of the parameter simplex (at $\theta_i = 0$). This ensures that we include applications with *arbitrarily* large cost in our evaluation.

which gives our new evaluation objective:

$$C_{mxe} = k \int_0^1 \int_0^1 \cdots \int_0^1 \sum_{i=1}^{N} \frac{1}{\theta_i} P_{miss}(i) \, d\theta_{N-1} \cdots d\theta_2 \, d\theta_1$$

$$\theta_N = 1 - \sum_{i=1}^{N-1} \theta_i$$

$$C_{mxe} = k \int_0^1 \int_0^1 \cdots \int_0^1 \sum_{i=1}^{N} \frac{1}{\theta_i} P_{miss}(i)\, d\theta_{N-1} \cdots d\theta_2\, d\theta_1$$

$$\theta_N = 1 - \sum_{i=1}^{N-1} \theta_i$$

We integrate a weighted combination of empirical miss-rate over the whole parameter simplex.

$$C_{mxe} = k\int_0^1\int_0^1 \cdots \int_0^1 \sum_{i=1}^{N} \frac{1}{\theta_i} P_{miss}(i)\, d\theta_{N-1} \cdots d\theta_2\, d\theta_1$$

$$\theta_N = 1 - \sum_{i=1}^{N-1} \theta_i$$

- OK, this is another impressive-looking integral, but how do you compute it?
- And why is it called C_{mxe}?

Why is it called C_{mxe} ?

- C_{mxe} refers to ***m**ulticlass-**c**ross-**e**ntropy*

(In the 2-class case: $C_{llr} = C_{mxe}$)

- When there are $N > 2$ classes, scores in likelihood-ratio form are inconvenient---so we work with scores in log-likelihood form. (Again: log is for practical reasons.)

How to compute C_{mxe}

- Computation of C_{mxe} from a supervised evaluation database is just as easy as computation of error-rates.

How to compute C_{mxe}

$$\boxed{C_{mxe} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\|S_i\|}\sum_{t\in S_i}\log_2\frac{\sum_{j=1}^{N}\exp(ll_{jt})}{\exp(ll_{it})}}$$

N is the number of classes.
S_i is the set of trials of class i .
ll_{it} = log P (trial t | class i).
is the *log-likelihood* under evaluation for class *i,* given the data of trial *t.*

Properties of C_{mxe}

- $C_{mxe} = 0$ (perfect): for $i \neq j$, outputs ll_{it} - $ll_{jt} = +\infty$, whenever i is the true class.
- $0 < C_{mxe} < \log_2 N$ (useful): well-calibrated, real-world detector.
- $C_{llr} = \log_2 N$ (reference): well-calibrated but *useless,* gives no discrimination, outputs $ll_{it} = ll_{jt}$ for any i, j and t.
- $\log_2 N < C_{llr} \leq \infty$ (badly calibrated): makes worse decisions than not using any detector at all (i.e. worse than reference detector).

Information-theoretic interpretation

C_{mxe} can be shown to be equivalent to an *empirical cross-entropy.*

$\Delta = \log_2 N - C_{mxe}$ gives the effective amount of *information* (in bits of Shannon entropy) that the recognizer delivers to the user, relative to a maximally uncertain prior of

$P_i = 1 / N$.

Information view is optimistic!

- This *information* view of multiclass recognizer performance gives an *optimistic* view for large *N*: For a given recognizer strategy, the amount of effective recognized information,

  $\Delta = \log_2 N - C_{mxe}$ tends to *increase* with *N*.

  As problem perplexity increases, experiments show we can also manage to extract more and more information.
- This is in marked contrast to *error-rates*, which appear to be more and more *pessimistic* for large *N*.

C_{mxe} as numerical optimization objective

Again:

Numerically optimizing C_{mxe} is just a form of *multiclass logistic regression,* which can also be solved with conjugate gradient methods.

Part II: Multiclass

1. What we want to do
2. Why cost and error-rate don't work.
3. How NIST did it.
4. How we propose to do it.
5. **Experimental demonstration of our proposal.**

Experimental demonstration

We experiment with 7 different language recognizers, which were submitted by 7 different teams for NIST 2007 Language Recognition Evaluation.

- Here $N = 14$ languages.

Experimental demonstration

- We demonstrate that we can calibrate (by multiclass logistic regression) the scores of several different language recognizers, to act as well-calibrated language likelihoods.
- We practically demonstrate well-calibratedness by successfully applying these likelihoods to make Bayes decisions for *thousands* of different applications.

Calibration strategies

2 of the 7 submitted recognizers had used the same multiclass logistic regression calibration that we are proposing in this talk.

- Both used my own calibration software. (Available as MATLAB toolkit, see below.)

Score transformation strategies

The other 5 recognizers were designed specifically for the LRE task of detecting one target language at a time, while the 13 other languages are considered non-targets.

- Their scores were presented in an *application-dependent* form, suitable for that task.
- We transformed these scores to act as multiclass language likelihoods. We used two different transformation strategies:

Score Transformations

1. **Projection**: A quick-and-dirty, parameterless, non-linear, non-invertible score transformation which converted the 14 separate detection-log-likelihood-ratios to assume the *form* of a 14-dimensional multiclass log-likelihood-vector.
2. **Re-calibration:** A parametrized, affine, invertible calibration transformation of scores to obtain the multiclass log-likelihood-vectors. Parameters were trained with multiclass logistic regression, using a *separate* set of training data specially provided by each team.

16369 Applications!

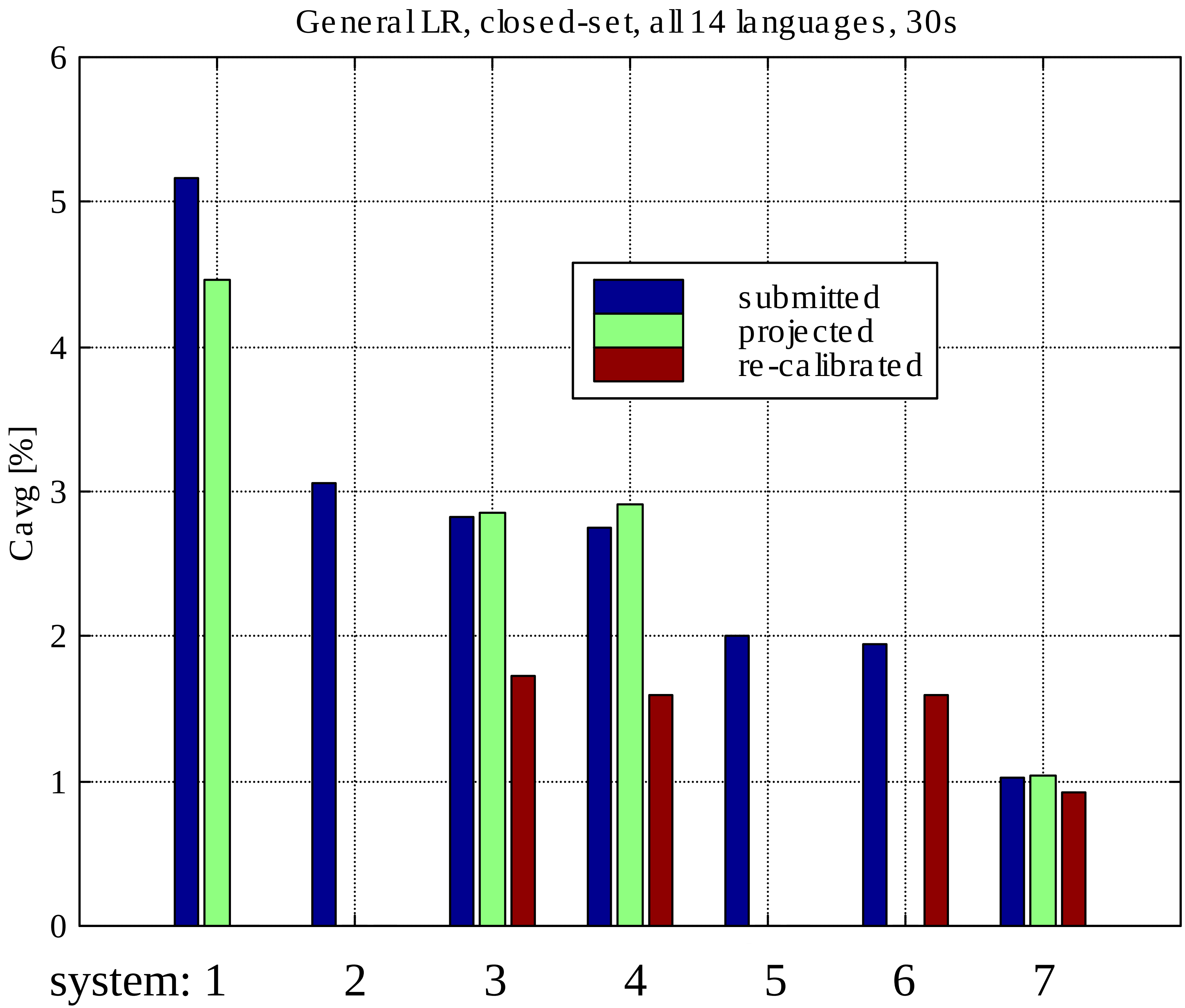
- The focus of LRE'07 was *one-against-the-rest detection,* within a closed subset of 14 languages.
  - The LRE'07 evaluation criterion was called C_{avg} which is an average of 14 detection cost functions, one for each target.
- We used this *same* framework for our demonstration, using the *same* C_{avg} evaluation criterion, but we applied it also to the other 16368 non-trivial *subsets* of these 14 languages.

16369 Applications!

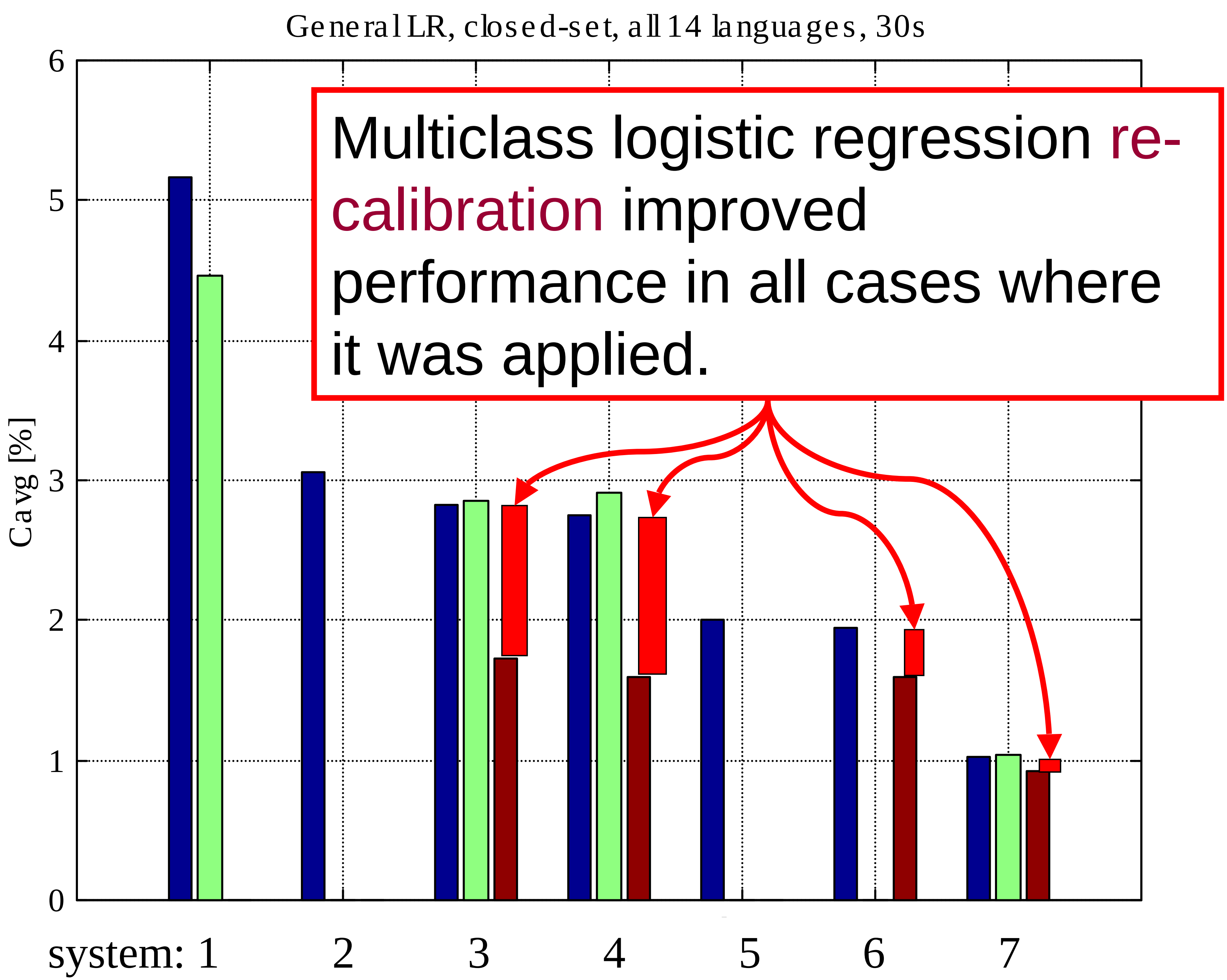
In summary: We did a total of 16369 different NIST evaluations, with language sets of sizes 2, 3, …, 14.

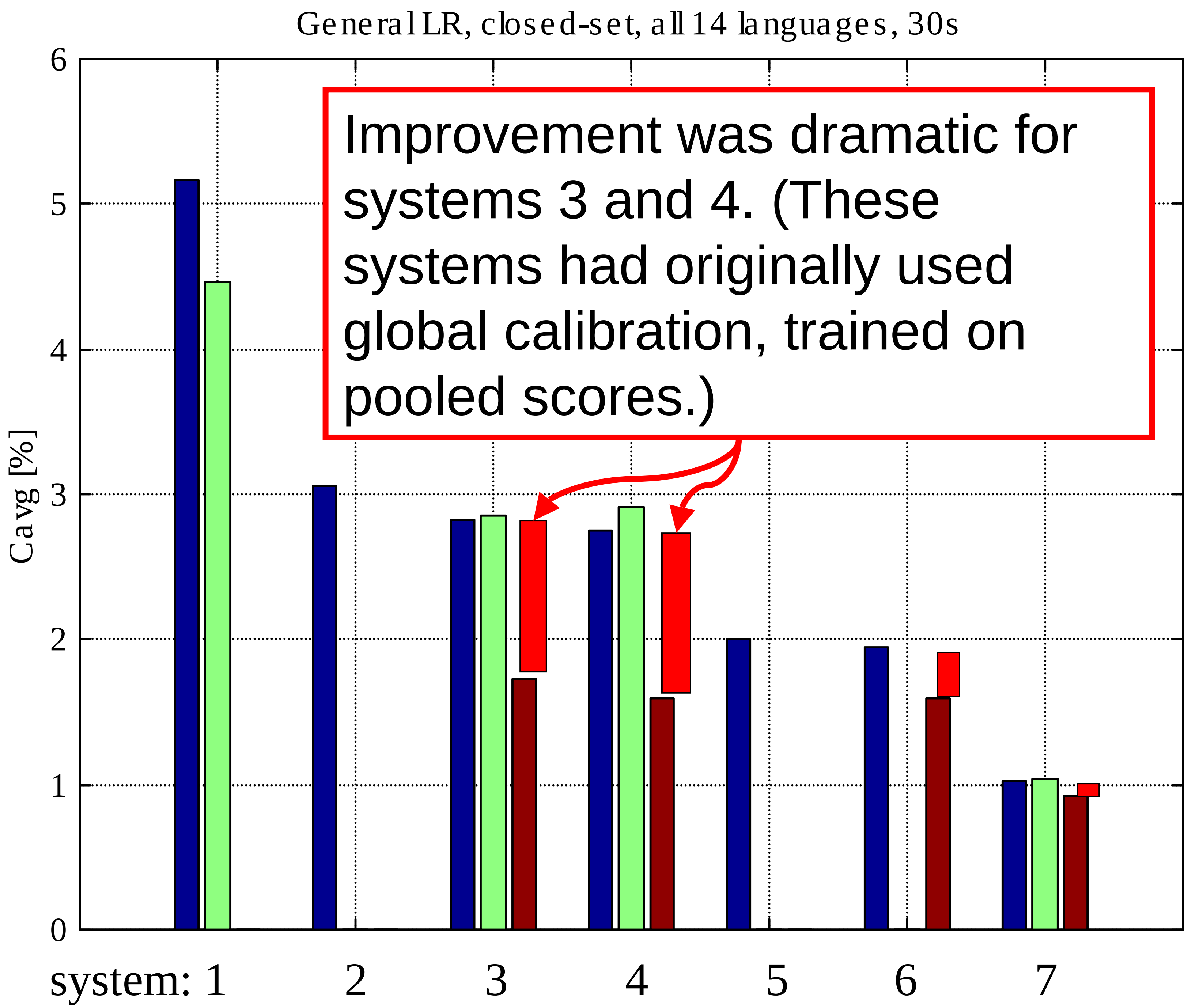
Sanity check:

Did re-calibration affect the original
14-language C_{avg} ?

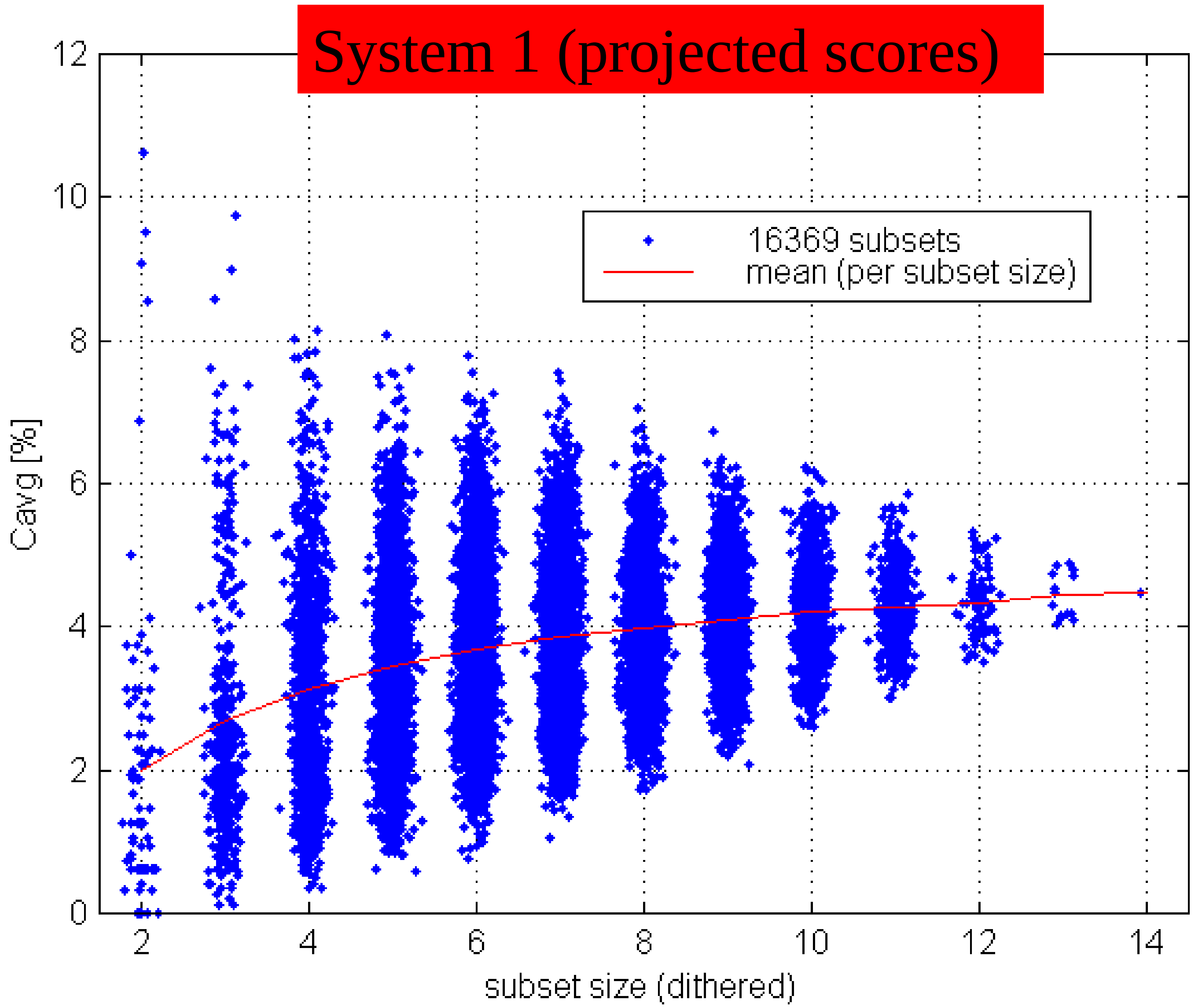


General LR, closed-set, all 14 languages, 30s
Cavg [%]
0
1
2
3
4
5
6
submitted
projected
re-calibrated
system: 1
2
3
4
5
6
7

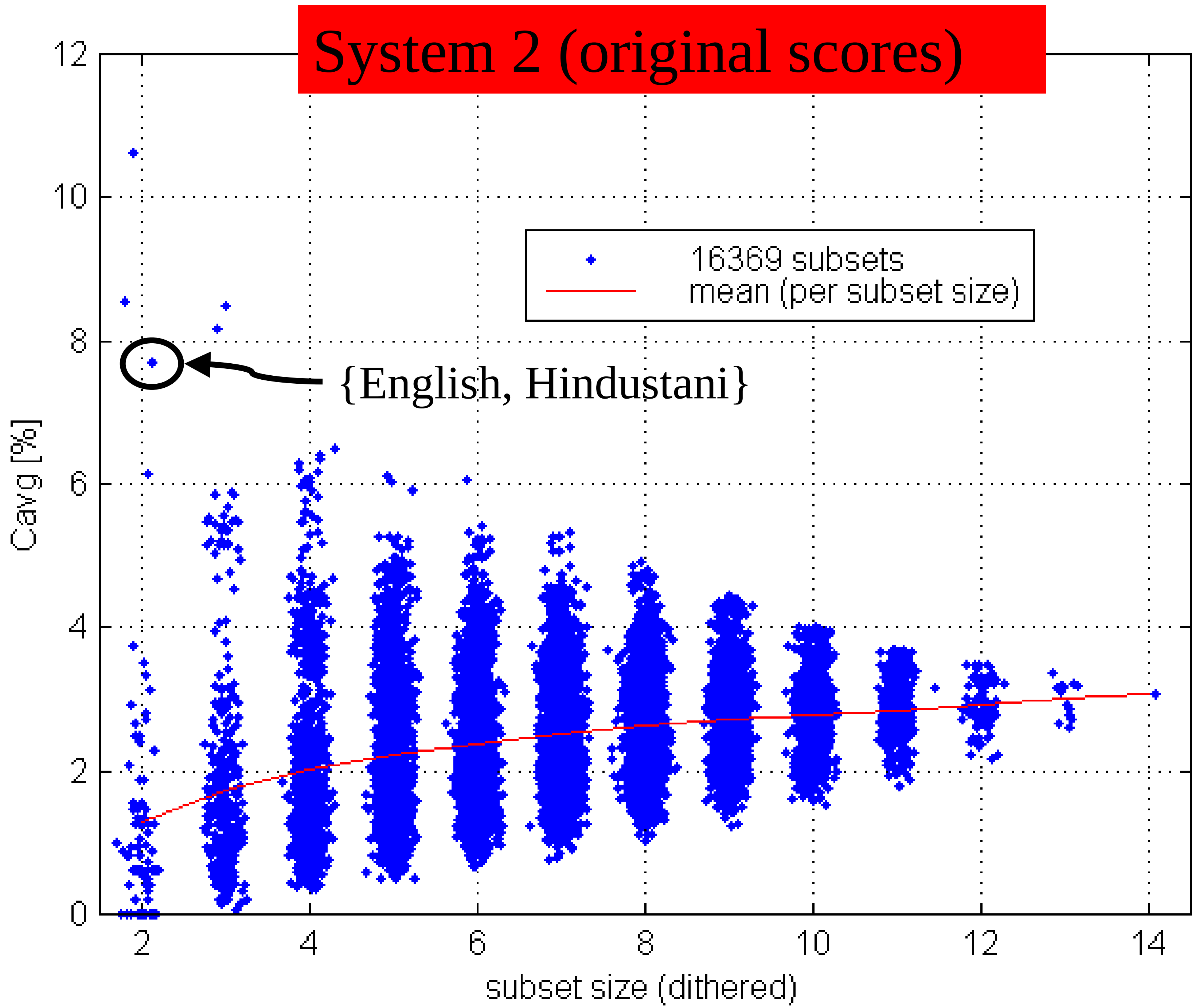

General LR, closed-set, all 14 languages, 30s
Cavg [%]
0
1
2
3
4
5
6
system: 1
2
3
4
5
6
7
Multiclass logistic regression re-calibration improved performance in all cases where it was applied.


General LR, closed-set, all 14 languages, 30s
Cavg [%]
0
1
2
3
4
5
6
system: 1
2
3
4
5
6
7
Improvement was dramatic for systems 3 and 4. (These systems had originally used global calibration, trained on pooled scores.)

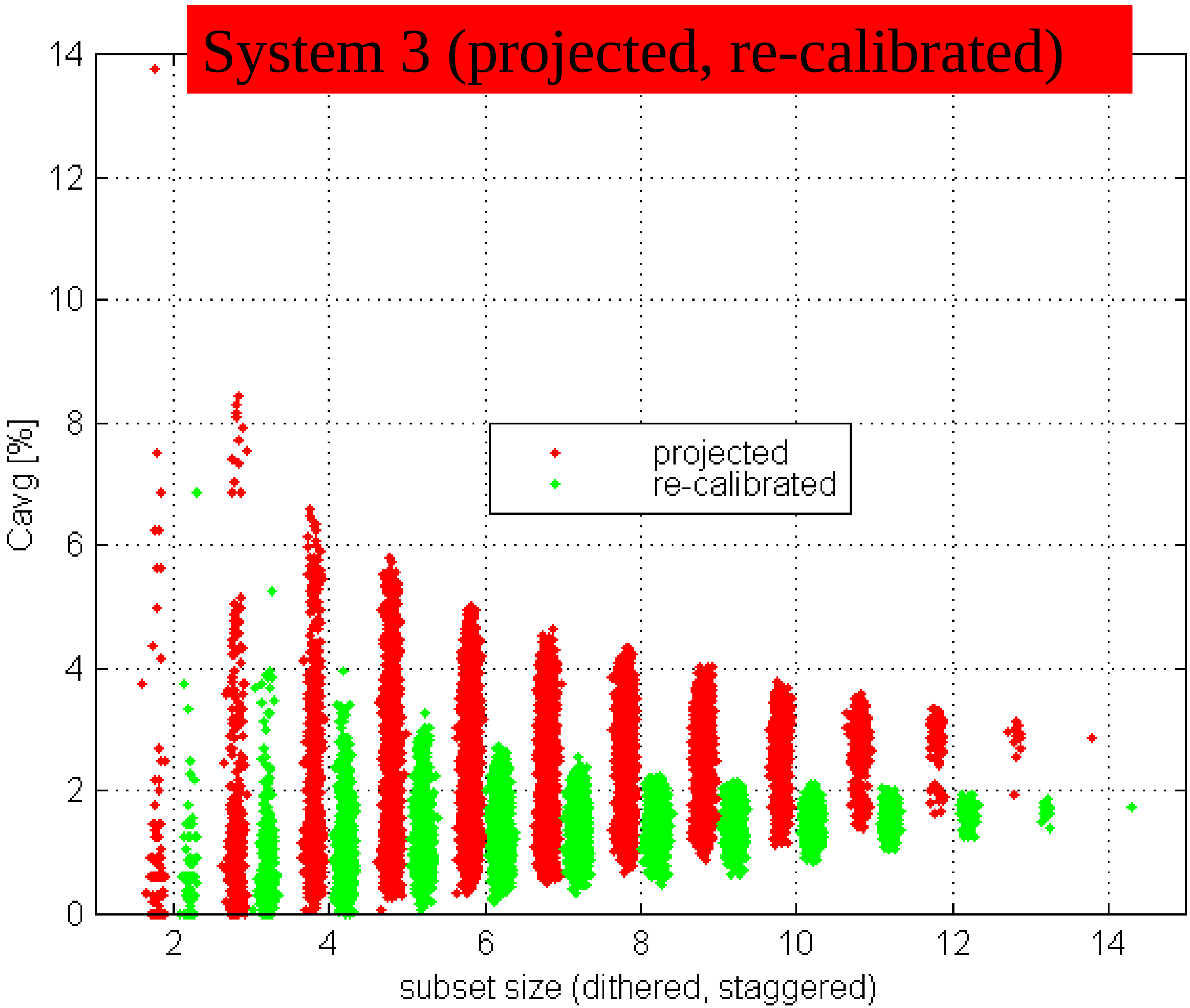
Now all 16369 subsets:

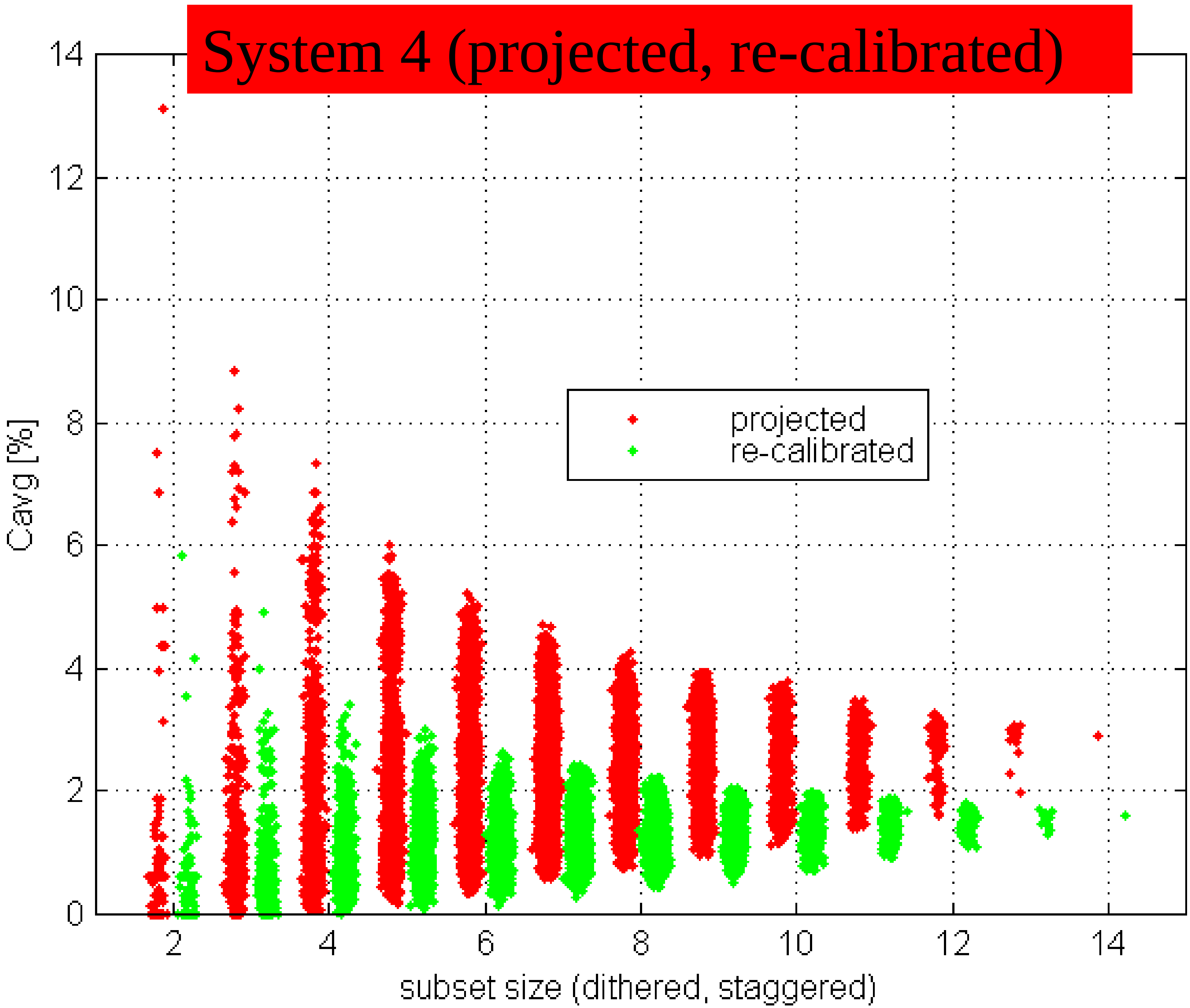


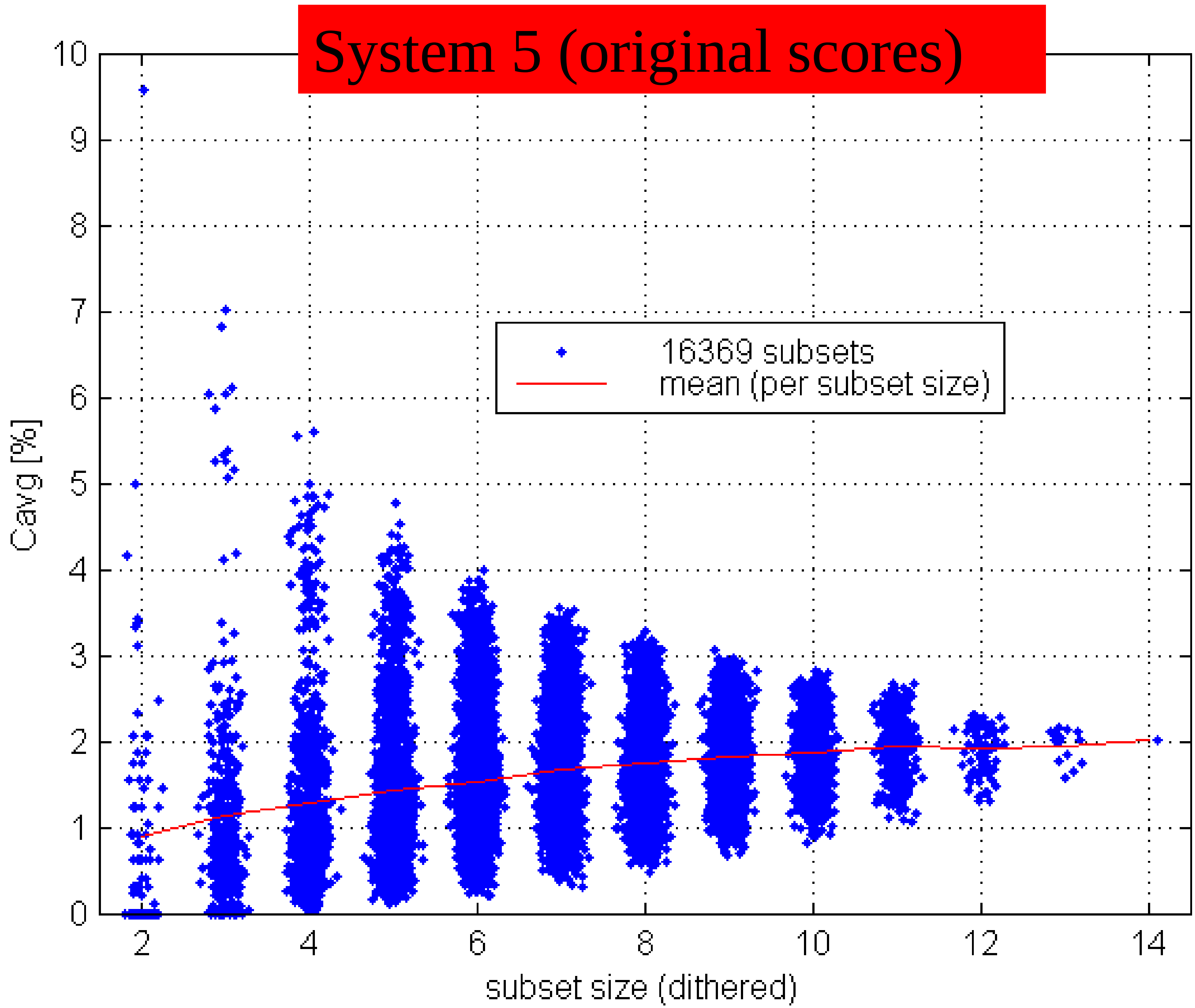
System 1 (projected scores)
16369 subsets
mean (per subset size)
Cavg [%]
subset size (dithered)
0
2
4
6
8
10
12
2
4
6
8
10
12
14

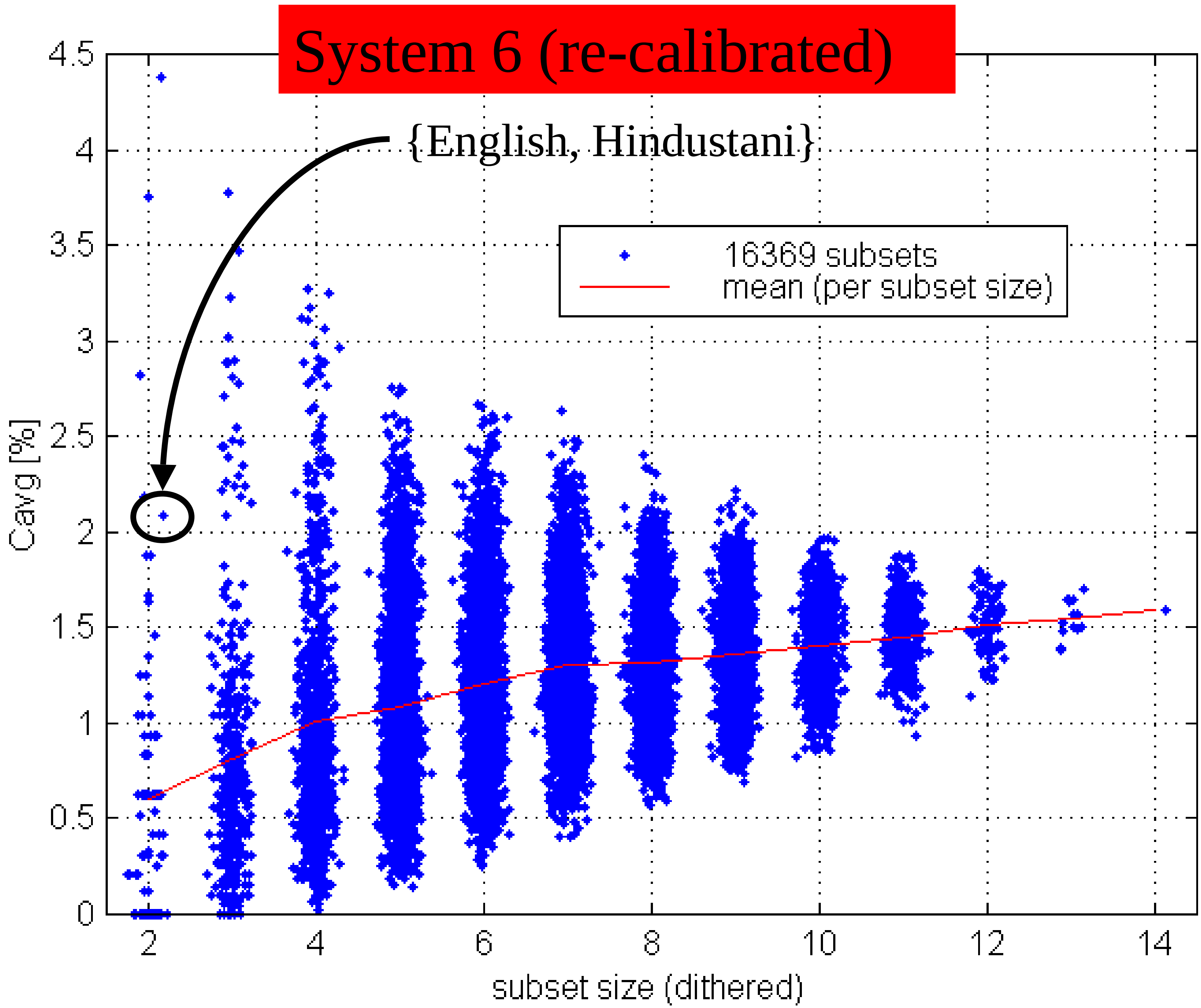


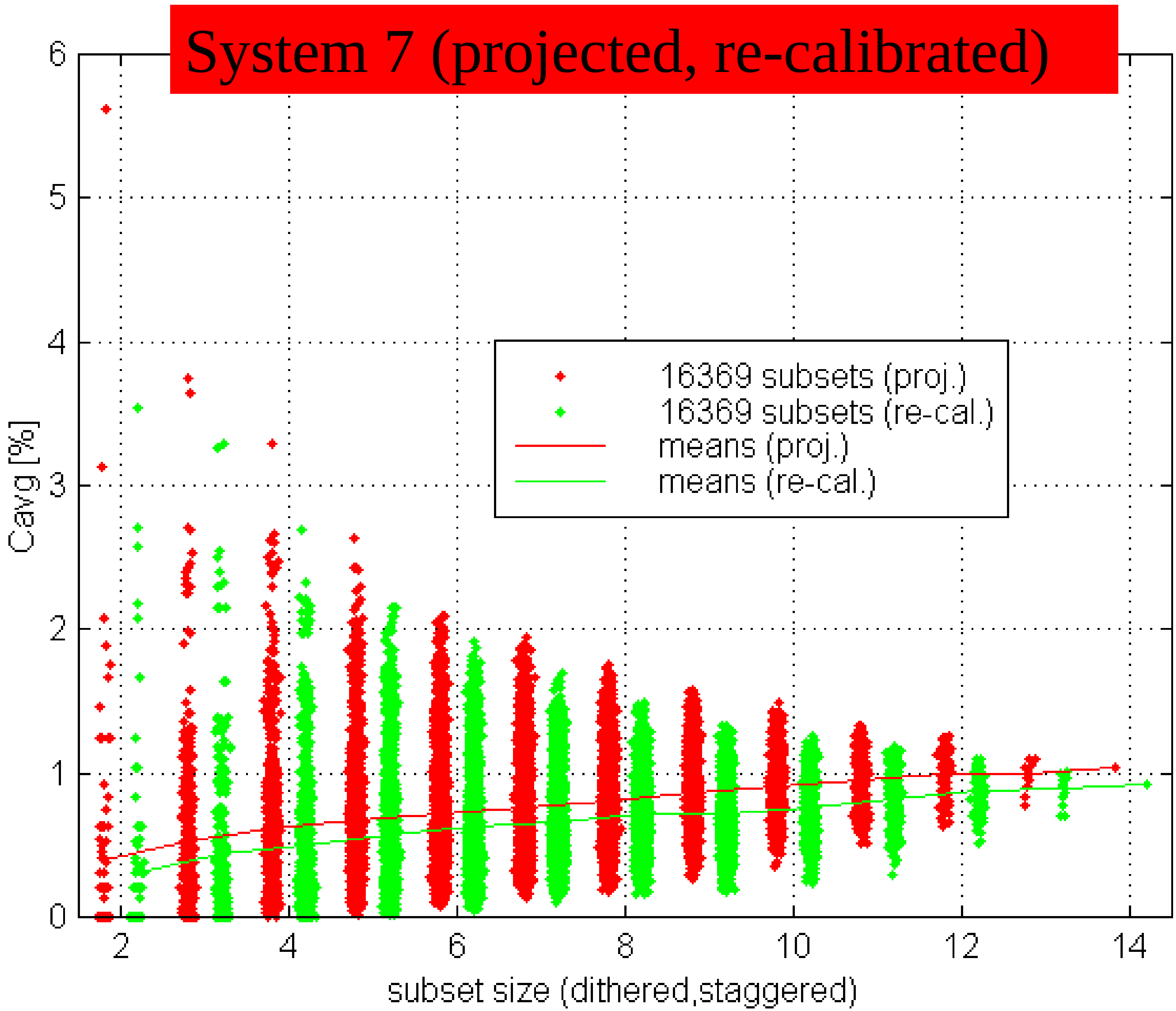
System 2 (original scores)
16369 subsets
mean (per subset size)
{English, Hindustani}
Cavg [%]
subset size (dithered)
0
2
4
6
8
10
12
14


System 3 (projected, re-calibrated)
Cavg [%]
0
2
4
6
8
10
12
14
subset size (dithered, staggered)
2
4
6
8
10
12
14
projected
re-calibrated


System 4 (projected, re-calibrated)
projected
re-calibrated
Cavg [%]
subset size (dithered, staggered)
0
2
4
6
8
10
12
14
2
4
6
8
10
12
14


System 5 (original scores)
16369 subsets
mean (per subset size)
Cavg [%]
subset size (dithered)
0
1
2
3
4
5
6
7
8
9
10
2
4
6
8
10
12
14


System 6 (re-calibrated)
{English, Hindustani}
16369 subsets
mean (per subset size)
Cavg [%]
subset size (dithered)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
2
4
6
8
10
12
14


System 7 (projected, re-calibrated)
16369 subsets (proj.)
16369 subsets (re-cal.)
means (proj.)
means (re-cal.)
Cavg [%]
subset size (dithered,staggered)
0
1
2
3
4
5
6
2
4
6
8
10
12
14

LRE Subsets Experiment: Conclusions

1. We used multiclass logistic regression (optimization of C_{mxe}), to *improve* the performance of systems that had specifically been designed for LRE'07.

   Our calibration was application-independent (not targeted at a specific application) but nevertheless it improved upon systems that had been *specially designed* for the LRE'07 detection application.

LRE Subsets Experiment: Conclusions

2. Our re-calibration allowed these same (previously application-dependent) language recognizers to also be applied successfully to thousands of other applications, even though those recognizers were not specifically designed for such use.

C_{mxe} tools: **FoCal Multiclass**

Free MATLAB toolkit for fusion, calibration, evaluation and bayes decisions for Multiclass Pattern Recognition

See:

http://niko.brummer.googlepages.com/focalmulticlass

Contents

Part I

Part II

Conclusion

Bibliography

Conclusion

The following are all *equivalent* (and all are good things to do):

- *Calibrating* pattern recognition outputs as *likelihoods*.
- *Optimizing* the amount of effective *information* delivered by pattern recognizers.
- *Optimizing* recognition *error-rates* over wide ranges of the priors.
- *Optimizing* recognizer *decision cost* over wide ranges of cost and prior parameters.

Bibliography

1. C_{llr} references
2. FSR References
3. C_{mxe} References

C_{llr} references

1. Niko Brümmer, “Application-Independent Evaluation of Speaker Detection”, Odyssey 2004: The Speaker and Language Recognmition Workshop, Toledo.
2. Niko Brümmer and Johan du Preez, “Application-Independent Evaluation of Speaker Detection”, Computer Speech and Language, 2006, pp. 230-275.
3. David van Leeuwen and Niko Brümmer, “An Introduction to Application-Independent Evaluation of Speaker Recognition Systems”, in *Speaker Classification vol 1*, Ed. Christian Müller, Springer, 2007, pp. 330-353.

FSR References

1. Daniel Ramos-Castro et al., “Likelihood Ratio Calibration in a Transparent and Testable Forensic Speaker Recognition Framework”, Proc. IEEE Odyssey 2006: The Speaker and Language Recognition Workshop, San Juan, 2006. (Best student paper award!)
2. W.M. Campbell et al., “Understanding scores in Forensic Speaker Recognition”, Proc. IEEE Odyssey 2006: The Speaker and Language Recognition Workshop, San Juan, 2006.

C_{mxe} References

1. David van Leeuwen and Niko Brümmer,”On Calibration of Language Recognition Scores”, Proc. IEEE Odyssey 2006: The Speaker and Language Recognition Workshop, San Juan, 2006.
2. Niko Brümmer, *FoCal Multi-class: Toolkit for Evaluation, Fusion and Calibration of Multi-class Recognition Scores: Tutorial and User Manual*, online: http://niko.brummer.googlepages.com/FoCal_MultiClass_Manual.pdf

Summary

It is common practice in many fields of basic pattern recognition research to evaluate performance as the misclassification error-rate on a given evaluation database. A limitation of this approach is that it implicitly assumes that all types of misclassification have equal cost and that the prior class distribution equals the relative proportions of classes in the evaluation database.

In this talk, we generalize the traditional error-rate evaluation, to create an evaluation criterion that allows optimization of pattern recognizers for wide ranges of applications, having different class priors and misclassification costs. We further show that this same strategy optimizes the amount of relevant information that recognizers deliver to the user.

In particular, we consider a class of evaluation objectives known as "proper scoring rules", which effectively optimize the ability of pattern recognizers to make minimum-expected-cost Bayes decisions. In this framework, we design our pattern recognizers to:

- extract from the input as much relevant information as possible about the unknown classes, and
- to output this information in the form of well-calibrated class likelihoods.

We refer to this form of output as "application-independent". Then when application-specific priors and costs are added, the likelihoods can be used in a straight-forward and standard way to make minimum-expected-cost Bayes decisions.

A given proper scoring rule can be interpreted as a weighted combination of misclassification costs, with a weight distribution over different costs and/or priors. On the other hand, proper scoring rules can also be interpreted as generalized measures of uncertainty and therefore as generalized measures of information. We show that there is a particular weighting distribution which forms the logarithmic proper scoring rule, and for which the associated uncertainty measure is Shannon's entropy, which is the canonical information measure. We conclude that optimizing the logarithmic scoring rule not only minimizes error-rates and misclassification costs, but it also maximizes the effective amount of relevant information delivered to the user by the recognizer.

We discuss separately our strategies for binary and multiclass pattern recognition:

- We illustrate the binary case with the example of speaker recognition, where the calibration of detection scores in likelihood-ratio form is of particular importance for forensic applications.
- We illustrate the multiclass case with examples from the recent 2007 NIST Language Recognition Evaluation, where we experiment with the language recognizers of 7 different research teams, all of which had been designed with one particular language detection application in mind. We show that by re-calibrating these recognizers by optimization of a multiclass logarithmic scoring rule, they can be successfully applied to a range of thousands of other applications.