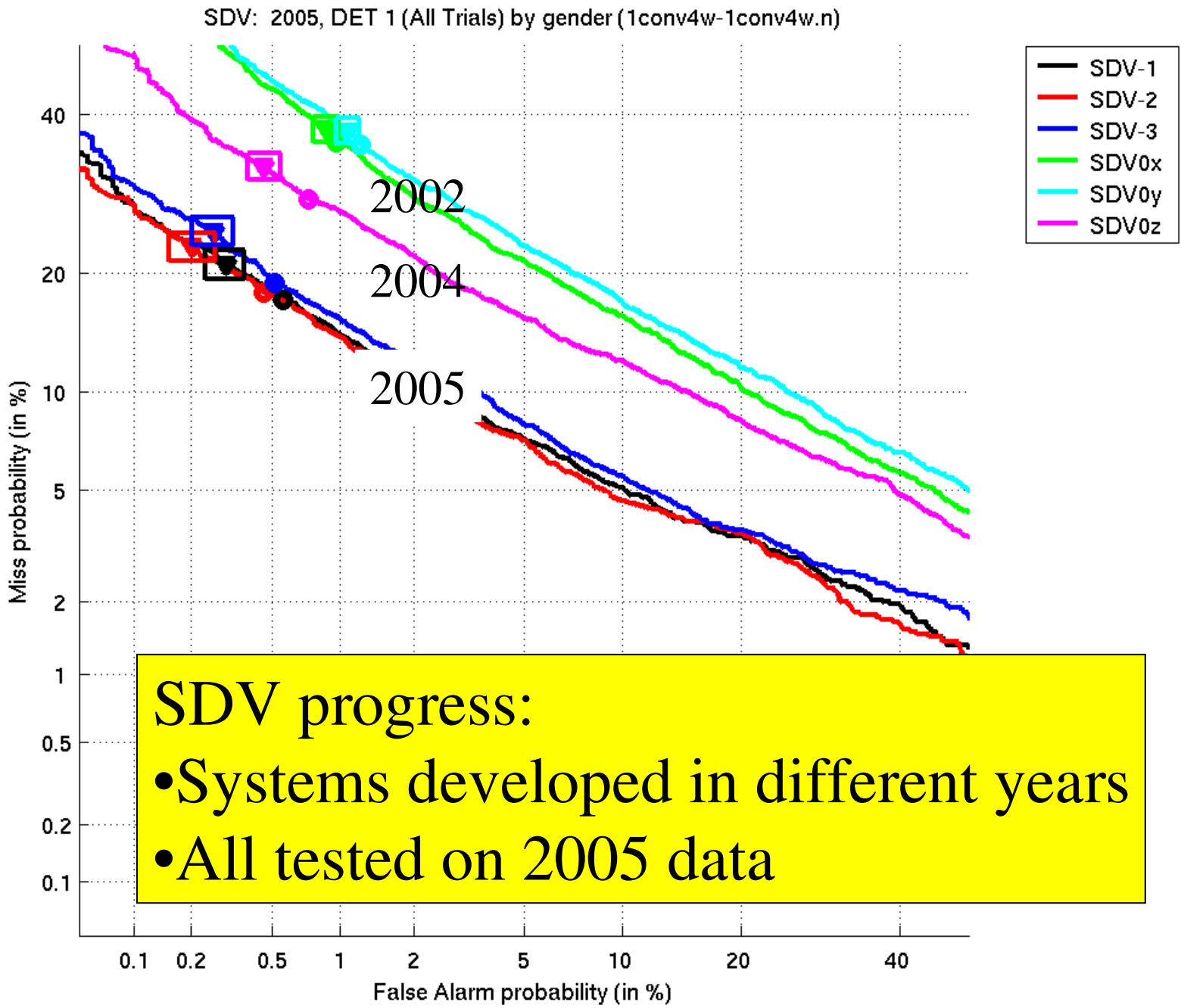
The Spescom DataVoice and University of Stellenbosch NIST SRE 2005 System

Niko Brümmer

Prologue

- What made it work?

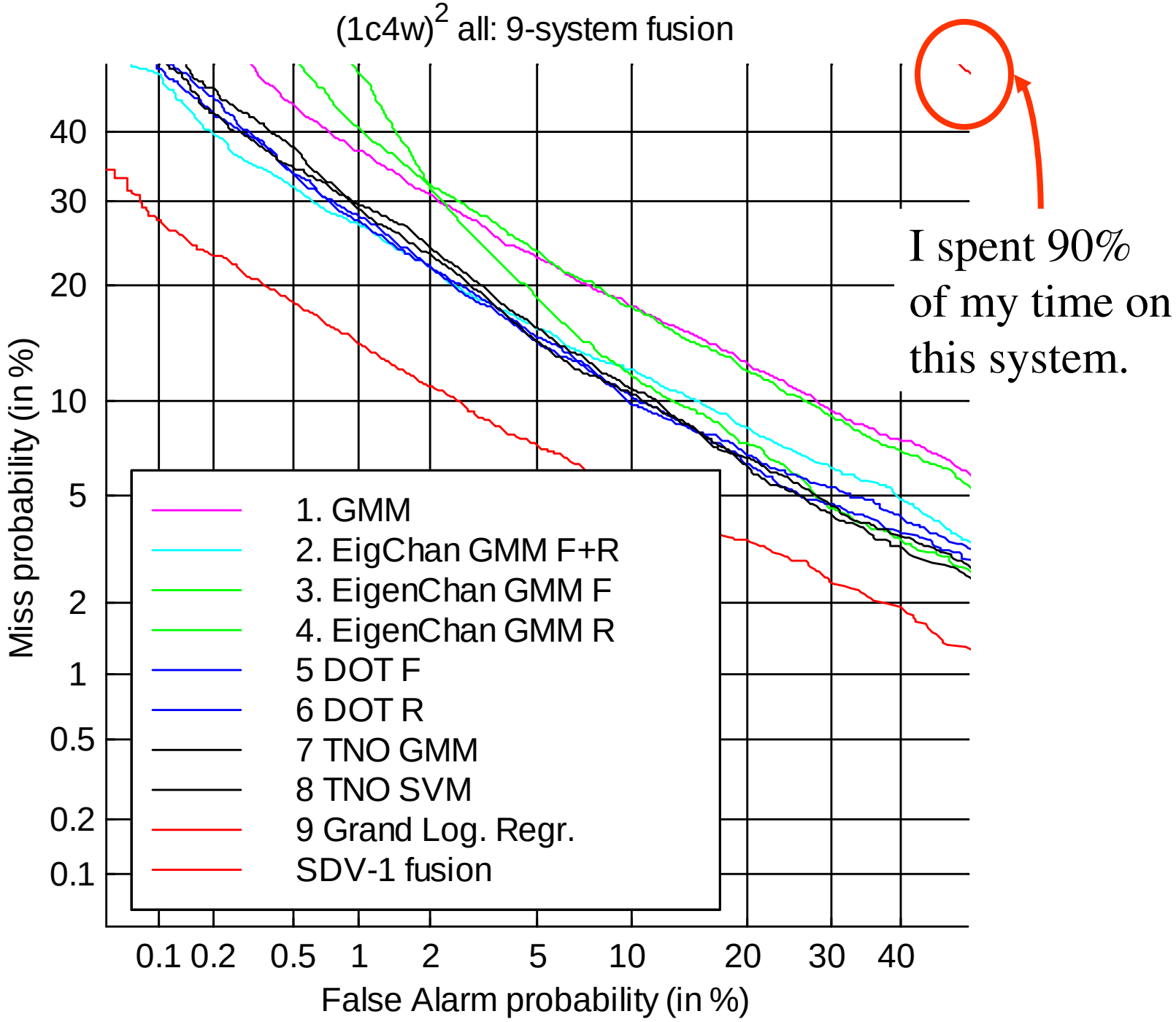


SDV progress:

- Systems developed in different years
- All tested on 2005 data

What made the difference this year?

- *Fusion*
  - Disassemble existing systems.
  - Re-assemble them in a variety of different ways.
  - Swap a system or two with your friends (TNO).
  - Fuse them all together, the more the merrier!


(1c4w)2 all: 9-system fusion
Miss 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
False Alarm probability (in %)
1. GMM
2. EigChan GMM F+R
3. EigenChan GMM F
4. EigenChan GMM R
5 DOT F
6 DOT R
7 TNO GMM
8 TNO SVM
9 Grand Log. Regr.
SDV-1 fusion
I spent 90% of my time on this system.

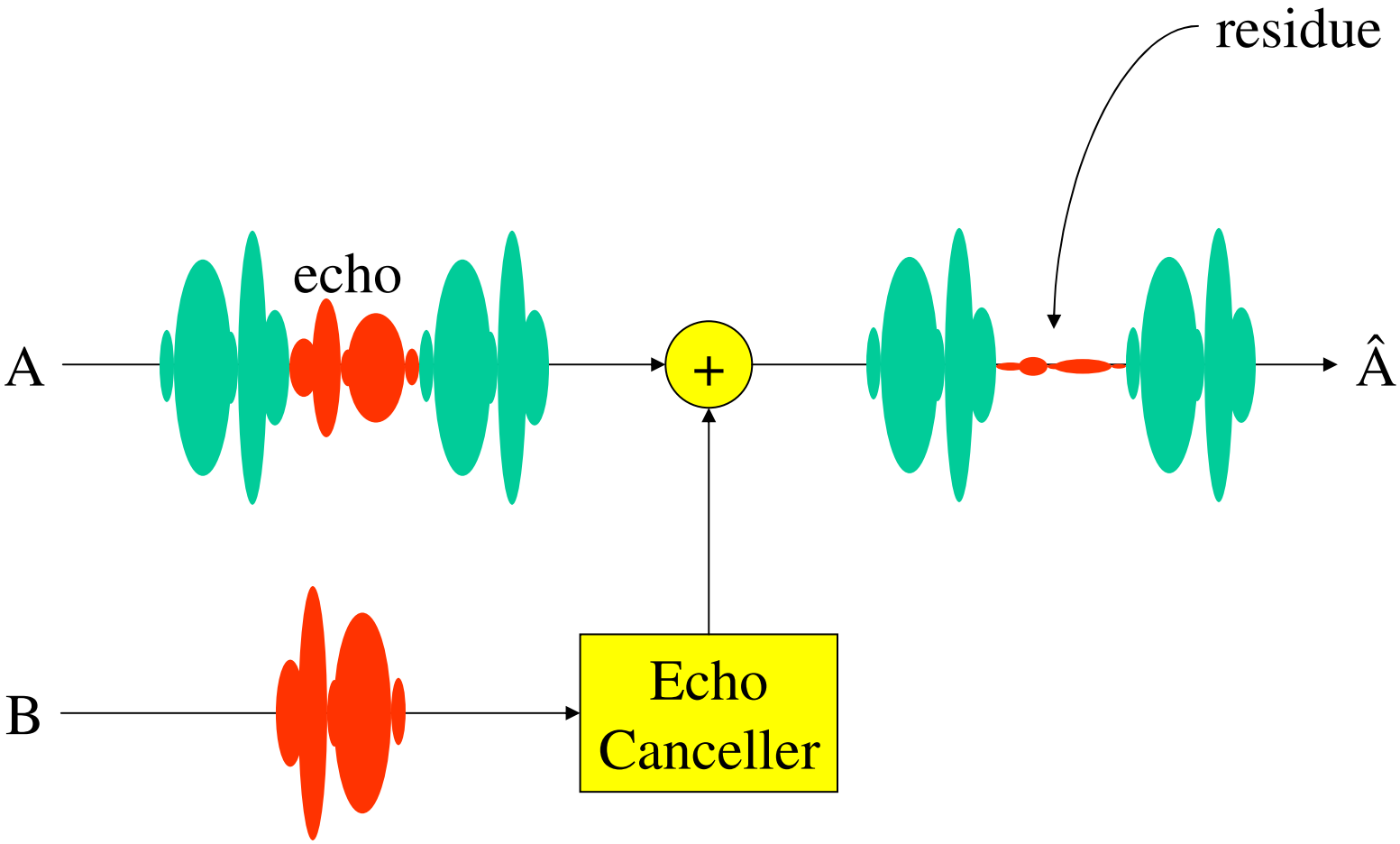
Summary

- $(1C4W)^2$ condition only
- Cross-Channel-Squelch
- New discriminative approaches:
  - new expanded feature set
  - logistic regression
  - *grand* logistic regression (new bilinear kernel)
  - piggyback fusion
- score $\rightarrow$log-likelihood-ratio *calibration*
  - *evaluation* of calibration (*APE* curve)

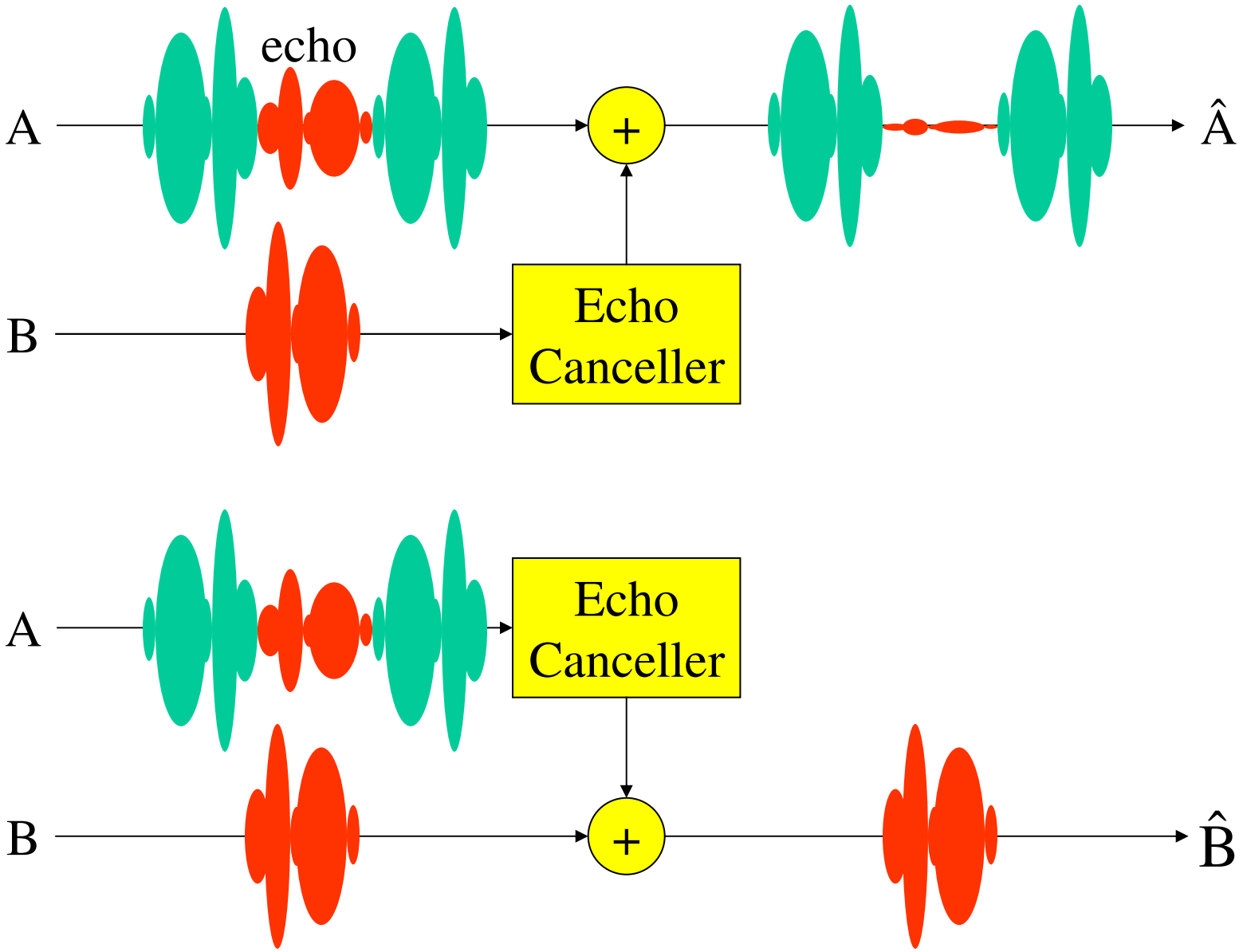
Cross-Channel Squelch

- X-Ch-Sq helps to suppress unwanted echo-cancellation residue. It was made possible this year by the new *stereo* data format.
- This did give a modest improvement.

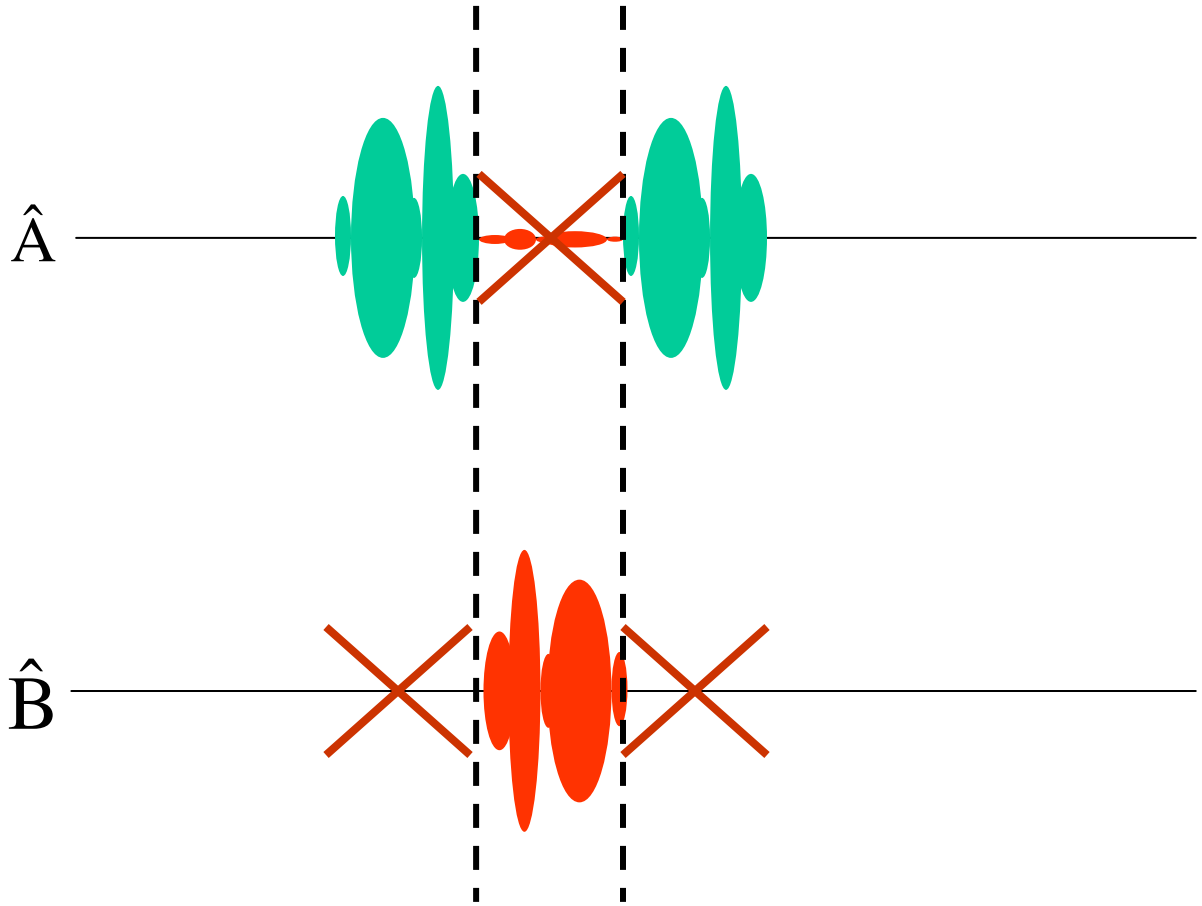
2004 data



2005 data



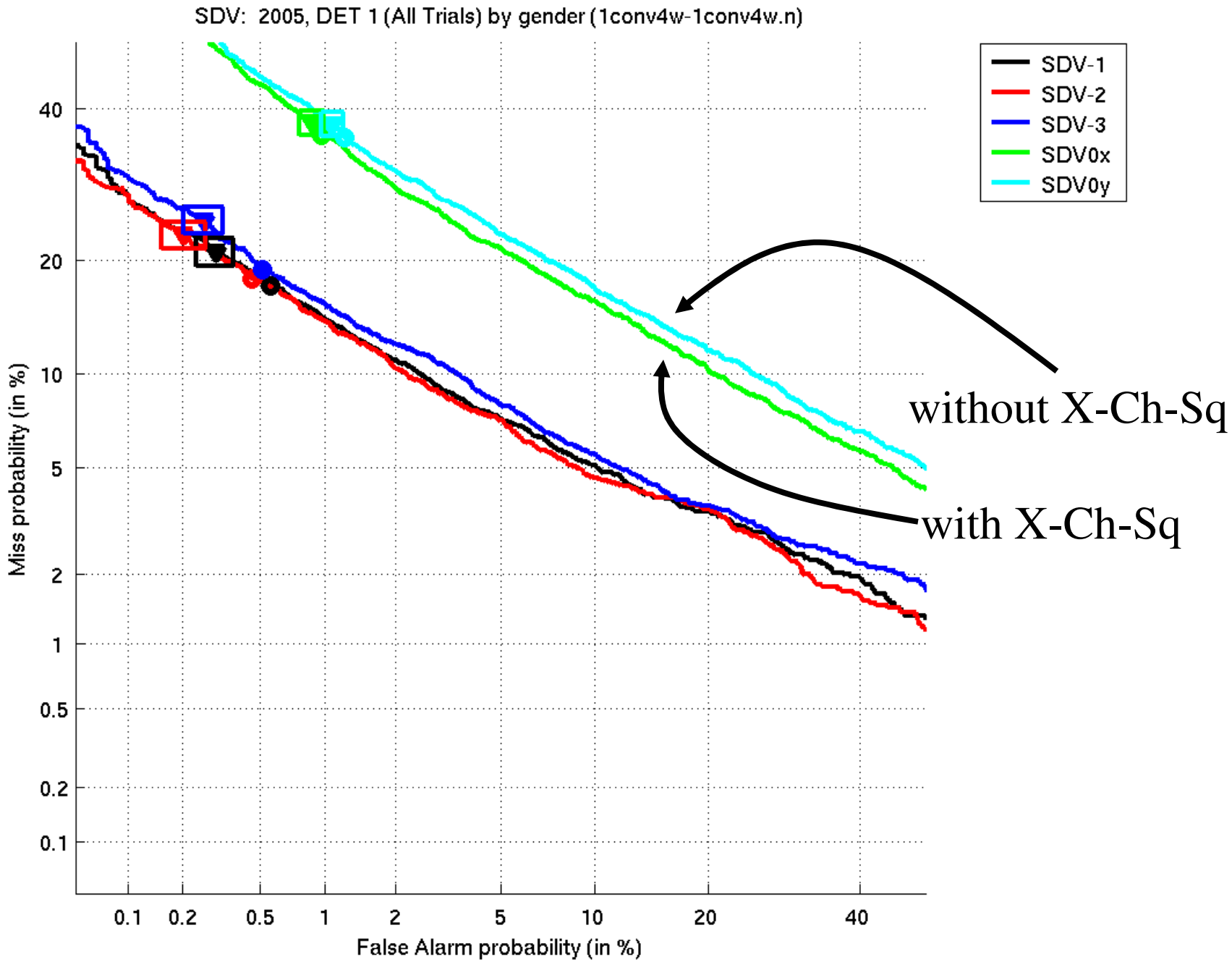
Cross-channel squelch



Discard possible residues via cross-channel energy comparison.

Does it make a difference?

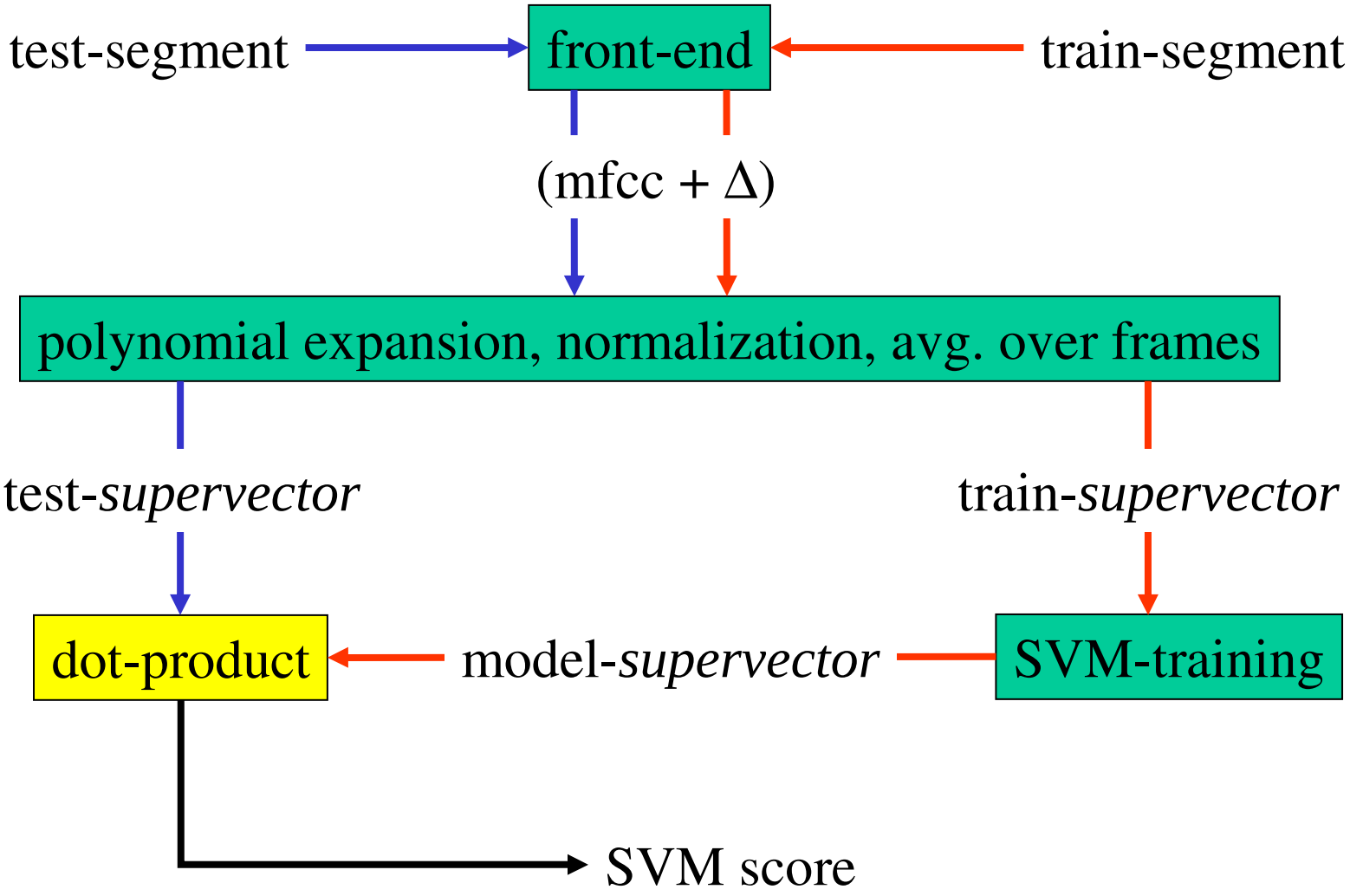
- We experimented with our baseline (2004) GMM system:
  - SDV0y: No cross-channel squelch – only the channel of interest was processed.
  - SDV0x: With cross-channel squelch.
- The DET-curve does show an improvement (see next slide).
  - This is one of the many small improvements that together gave a large improvement.


SDV: 2005, DET 1 (All Trials) by gender (1conv4w-1conv4w.n)
SDV-1
SDV-2
SDV-3
SDV0x
SDV0y
Miss probability (in %)
40
20
10
5
2
1
0.5
0.2
0.1
0.1
0.2
0.5
1
2
5
10
20
40
False Alarm probability (in %)
without X-Ch-Sq
with X-Ch-Sq

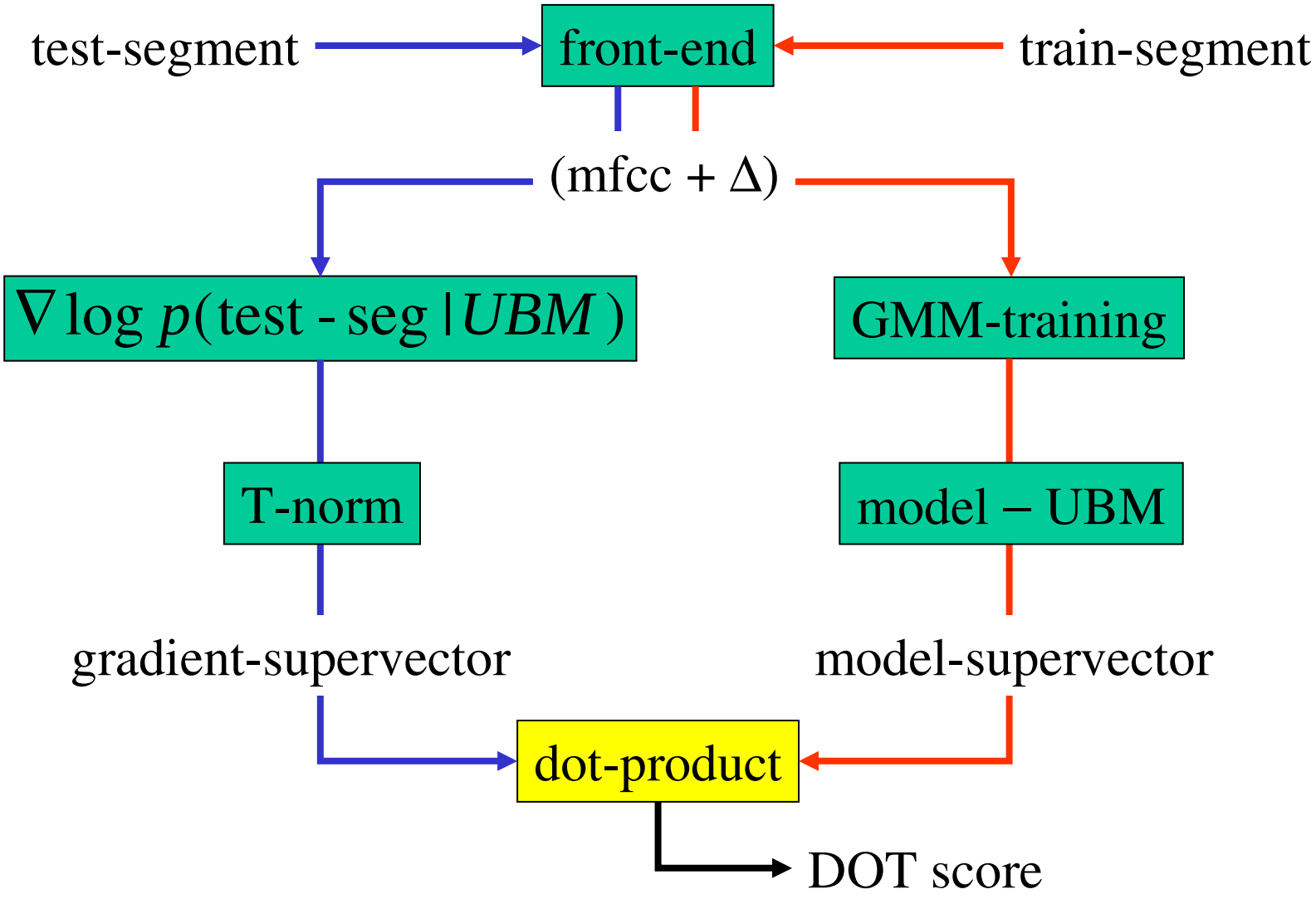
New Discriminative Approaches

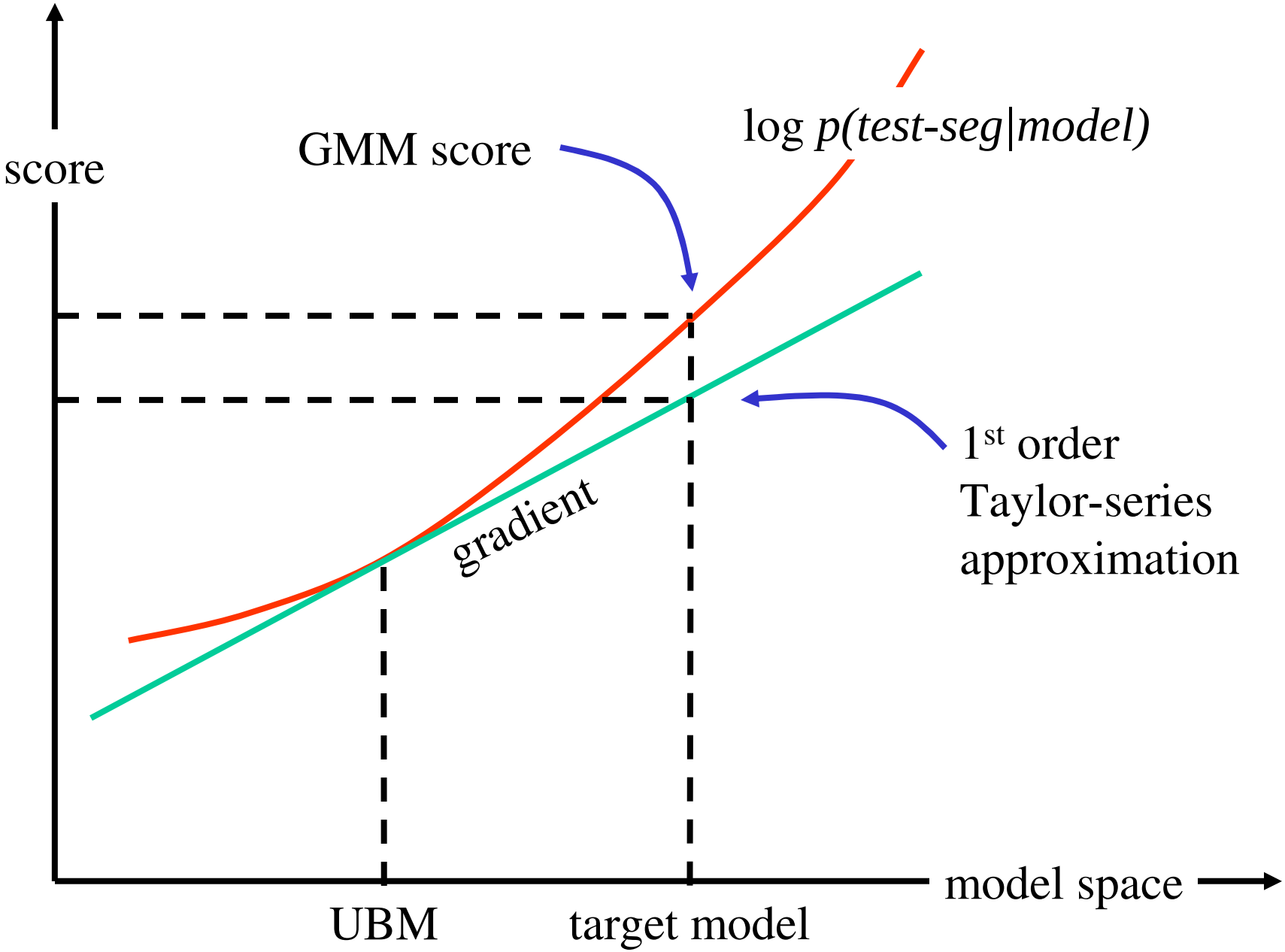
- **Expanded features**
- Objective function
- Optimization procedure
- Grand logistic regression
- Piggyback fusion
- Linear fusion

GLDS+SVM


test-segment
front-end
train-segment
(mfcc + Δ)
polynomial expansion, normalization, avg. over frames
test-*supervector*
train-*supervector*
dot-product
model-*supervector*
SVM-training
SVM score

Linear approx. to GMM score



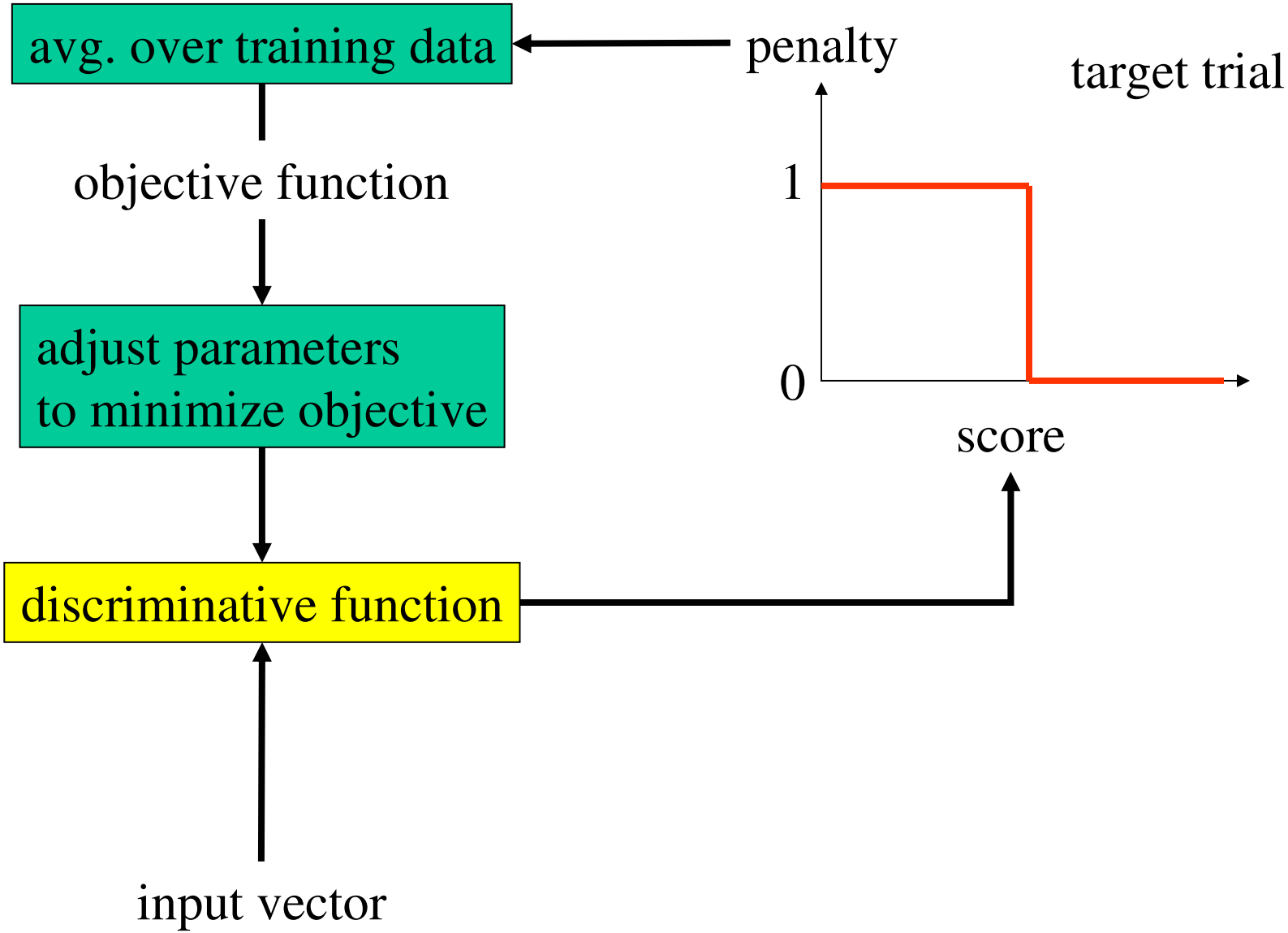

score
GMM score
log p(test-seg|model)
1st order
Taylor-series
approximation
gradient
model space
UBM
target model

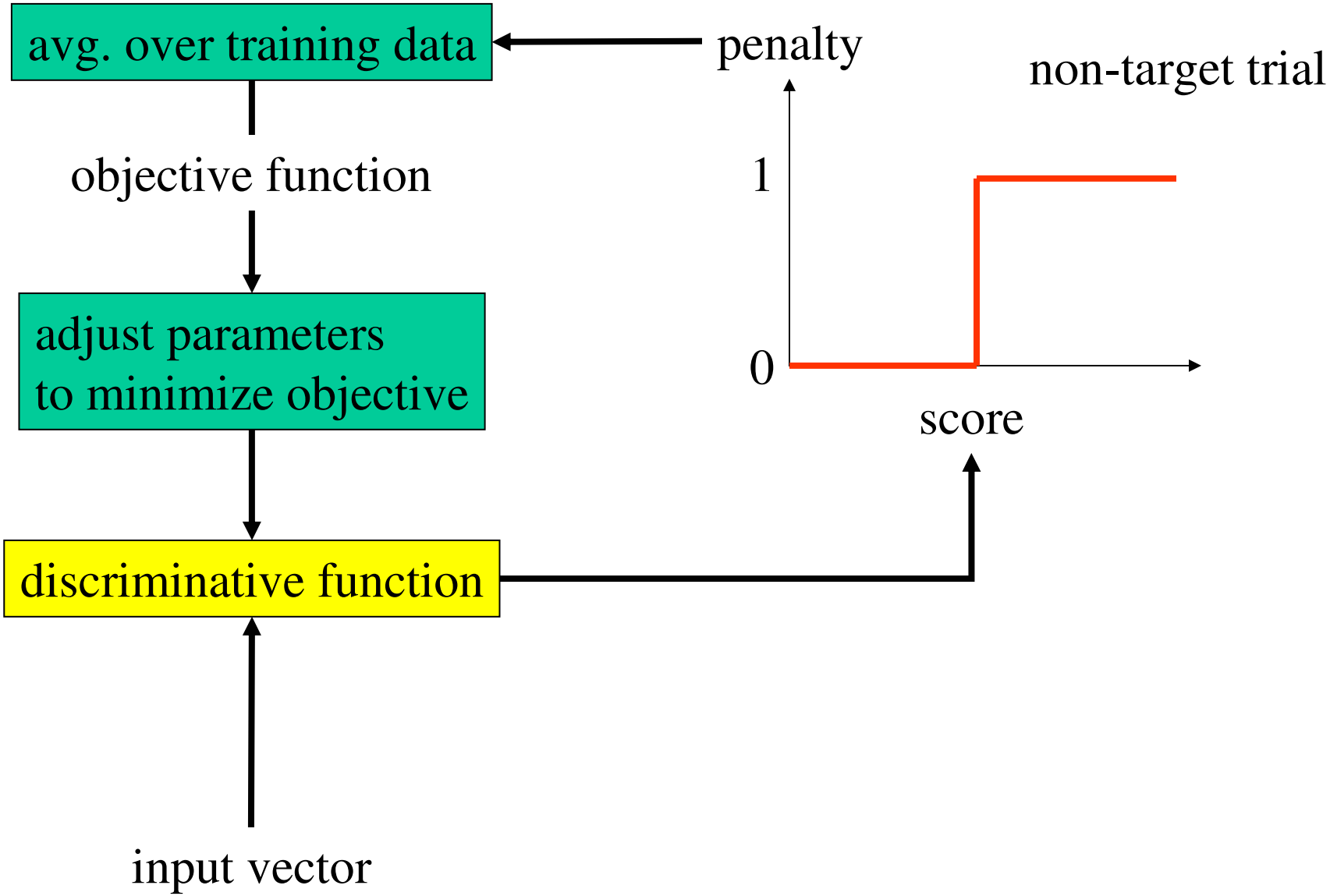
Expanded features

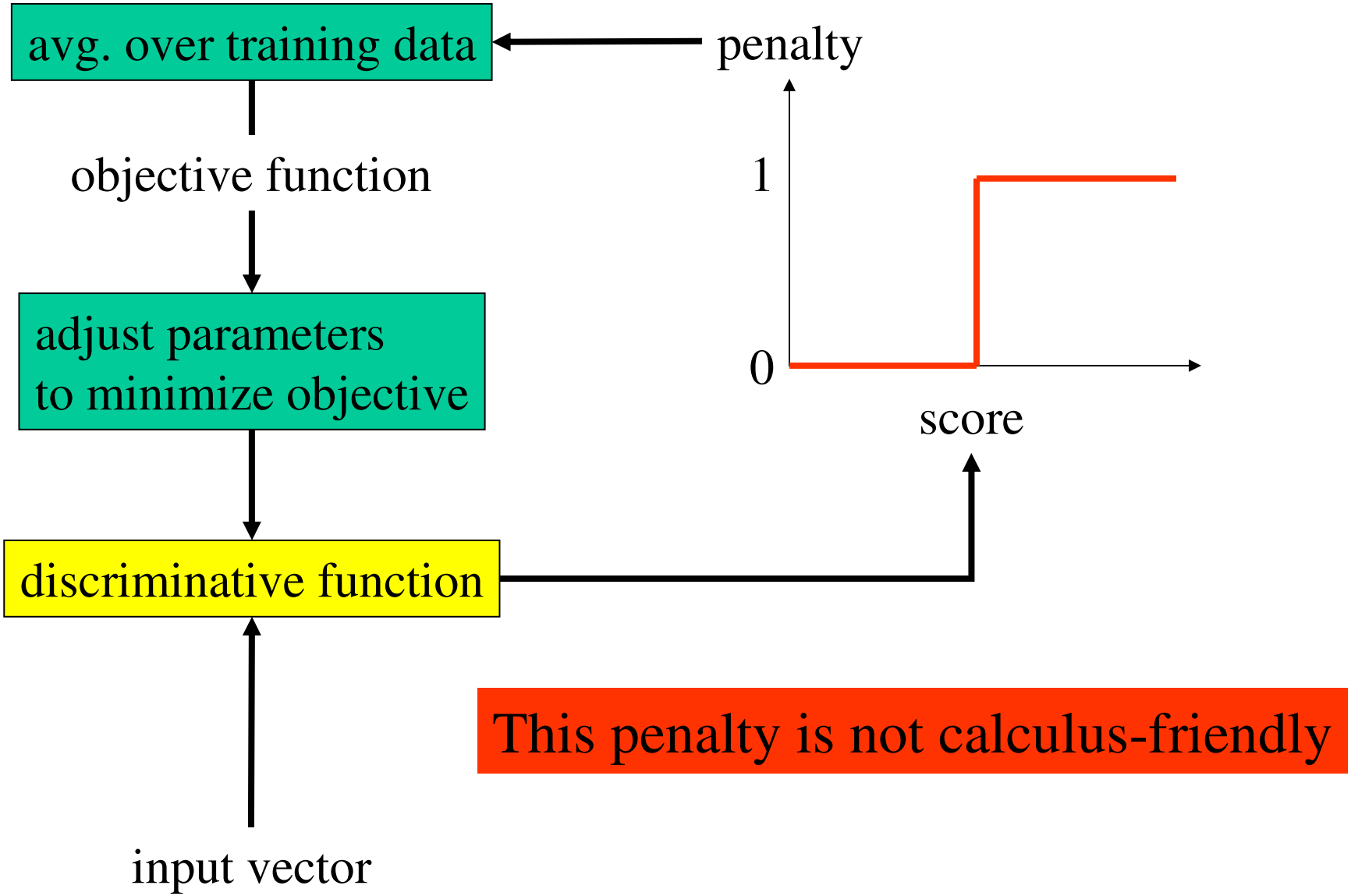
- As a side effect, the DOT score turned out to be complementary when fused with the GMM score!
- But, the primary purpose of this exercise was to generate *supervectors* (gradient and model) to use in a discriminative approach. We wanted to start from a *linearized* GMM score, and then replace the dot product with a discriminatively trained scoring procedure.

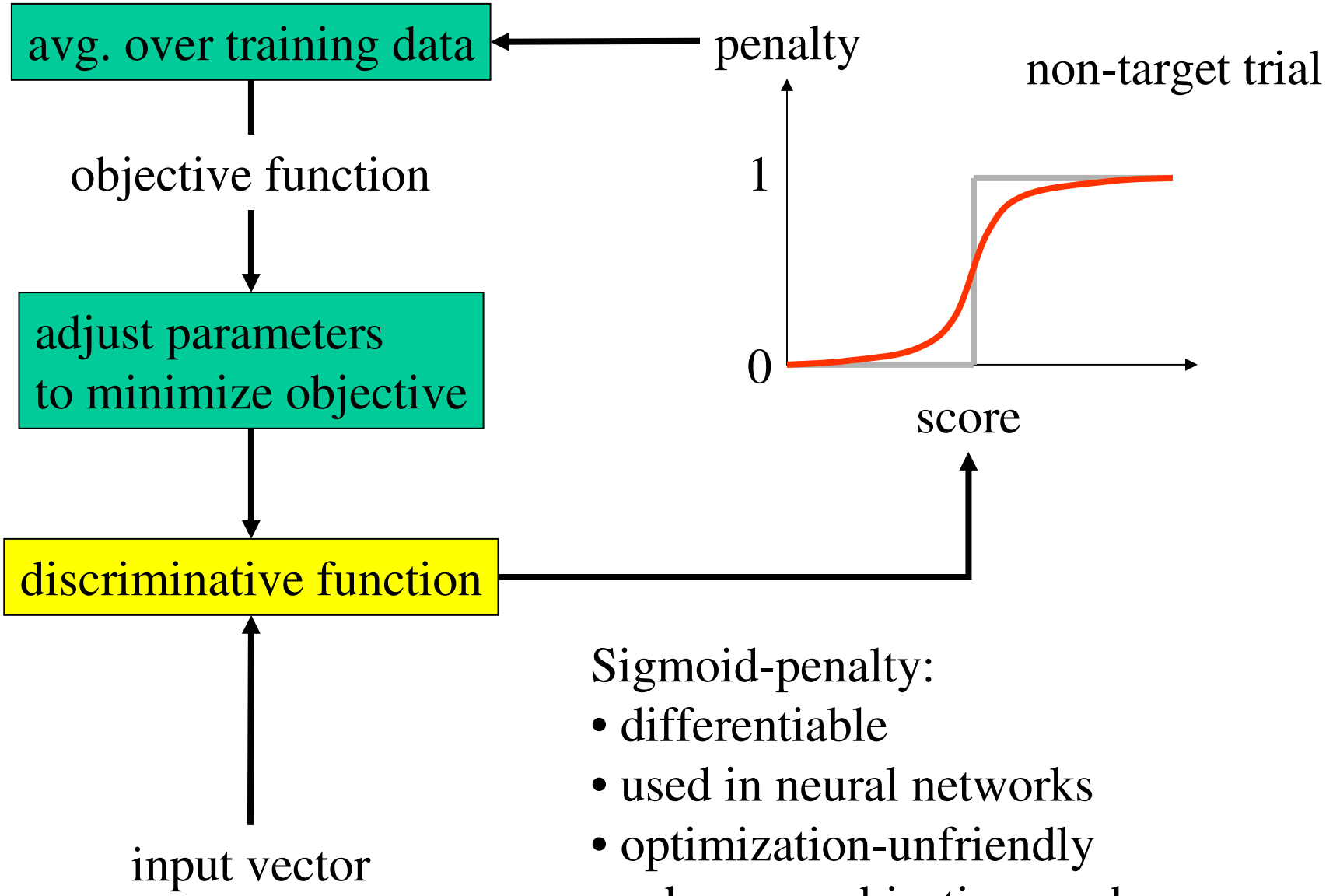
Discriminative approach

- Expanded features
- **Objective function: logistic regression**
- Optimization procedure
- Grand logistic regression
- Piggyback fusion
- Linear fusion


avg. over training data
penalty
target trial
objective function
1
adjust parameters
to minimize objective
0
score
discriminative function
input vector

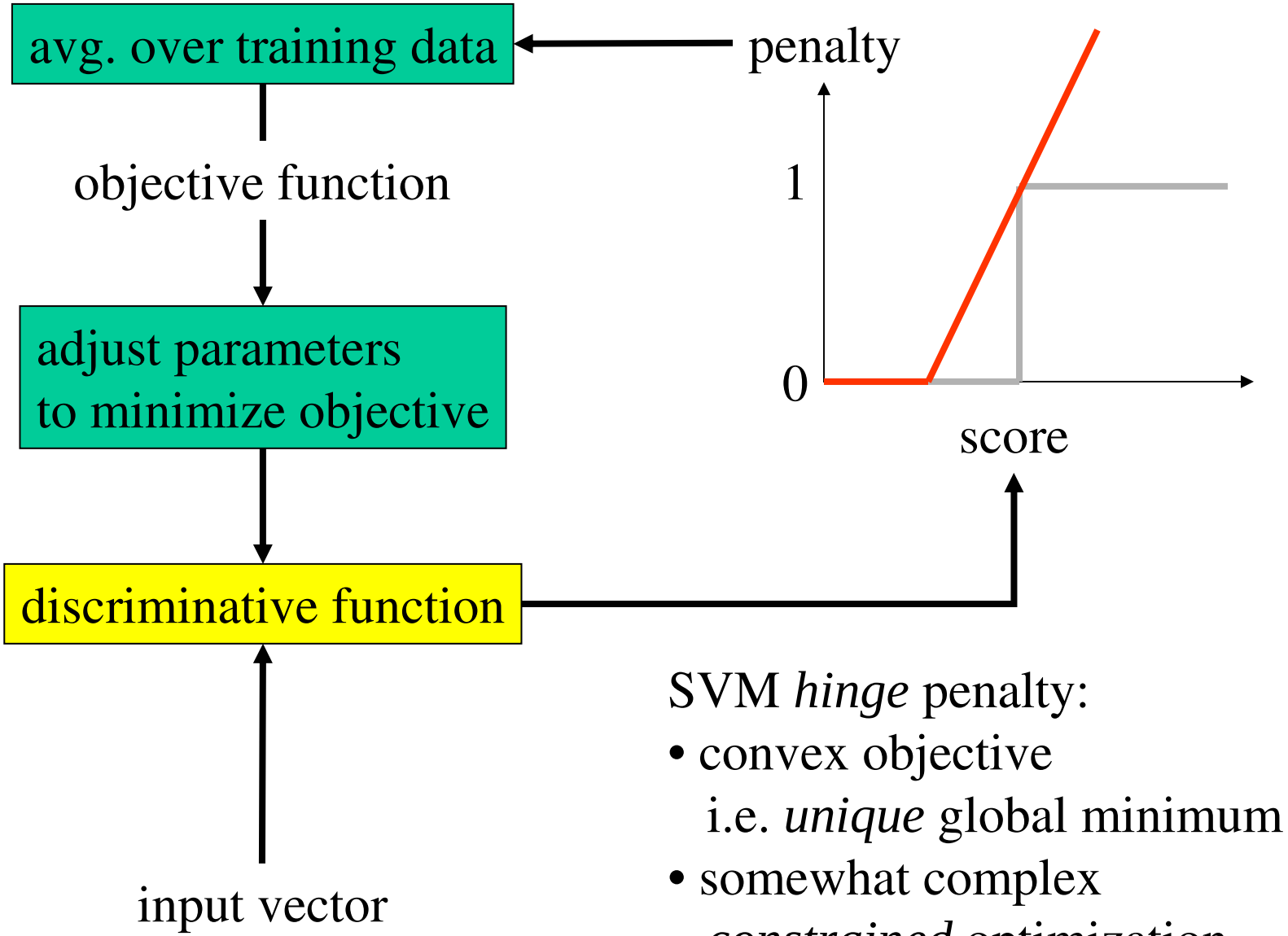

avg. over training data
penalty
non-target trial
objective function
1
adjust parameters
to minimize objective
0
score
discriminative function
input vector


avg. over training data
penalty
objective function
1
adjust parameters
to minimize objective
0
score
discriminative function
This penalty is not calculus-friendly
input vector



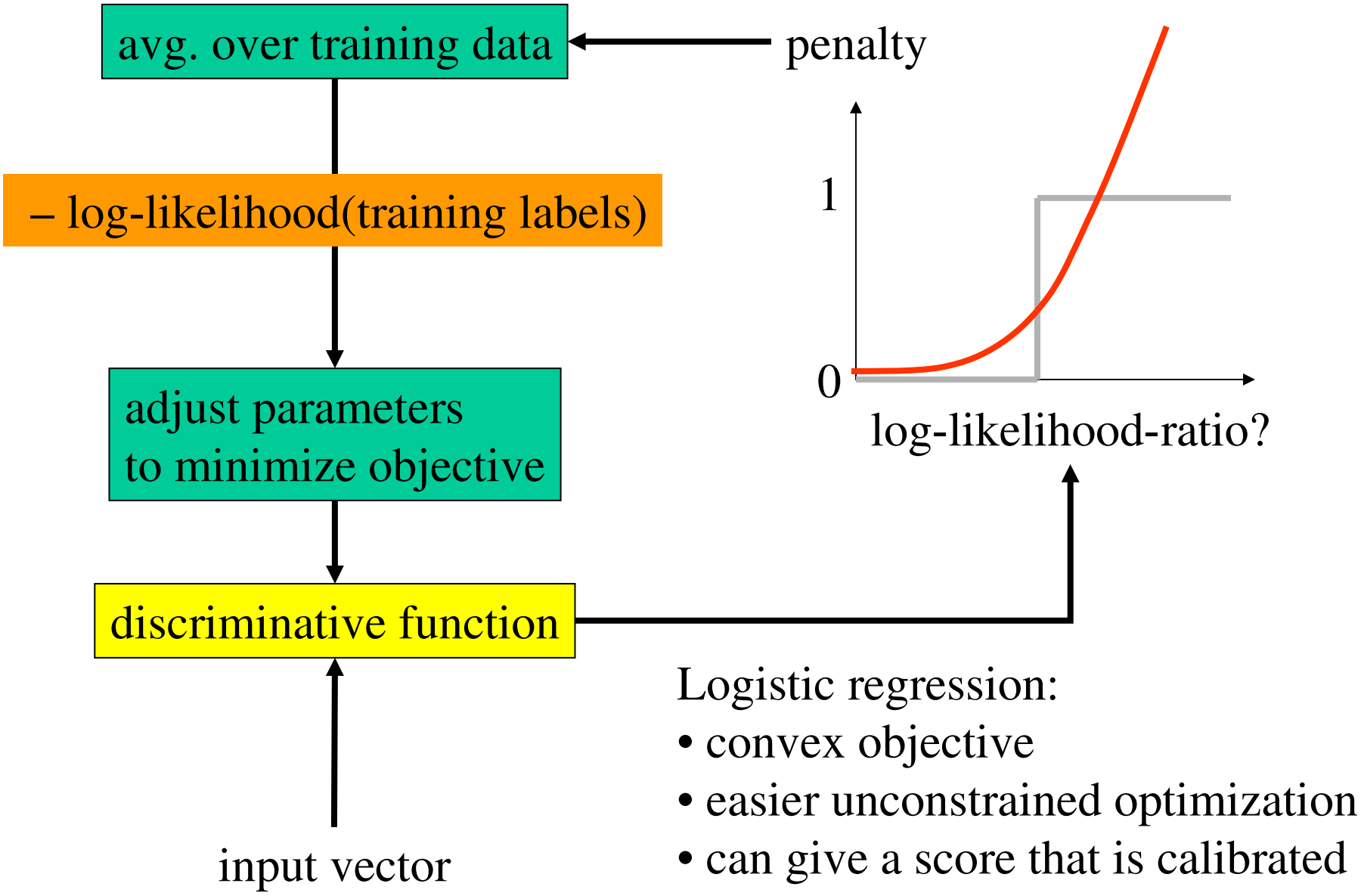
Sigmoid-penalty:

- differentiable
- used in neural networks
- optimization-unfriendly
  because objective can have local optima



SVM *hinge* penalty:

- convex objective
  i.e. *unique* global minimum
- somewhat complex
  constrained optimization
  problem



Logistic regression:

- convex objective
- easier unconstrained optimization
- can give a score that is calibrated as a log-likelihood-ratio

Discriminative approach

- Expanded features
- Objective function
- **Optimization procedure**
- Grand logistic regression
- Piggyback fusion
- Linear fusion

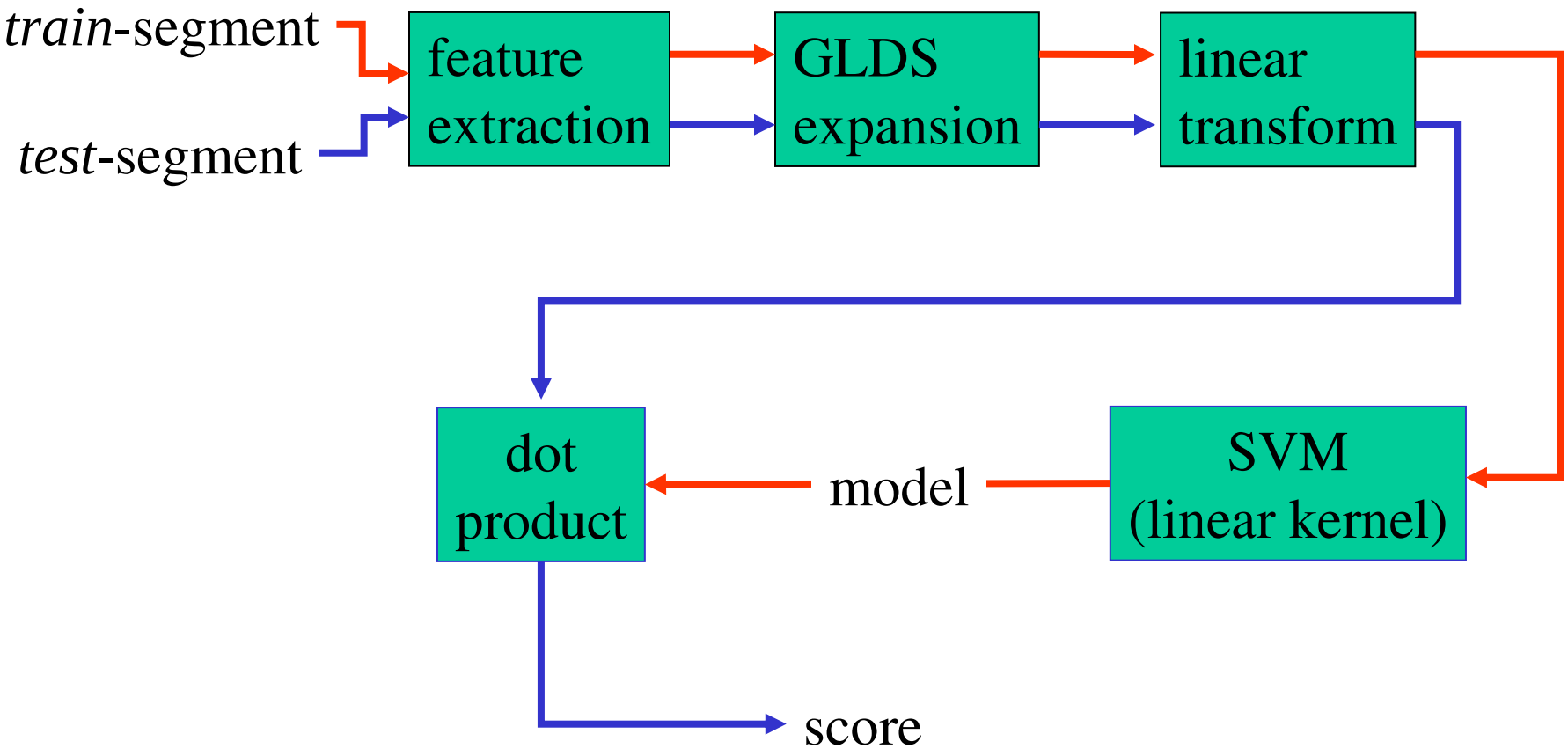
Logistic regression training

- Since the objective function is convex and unconstrained, almost anything works.
- But some methods are a *lot* faster than others.
- We used a conjugate-gradient algorithm that uses 1st and 2nd derivatives of the objective function. (See the system description for a reference to the algorithm of Tom Minka.)

Discriminative approach

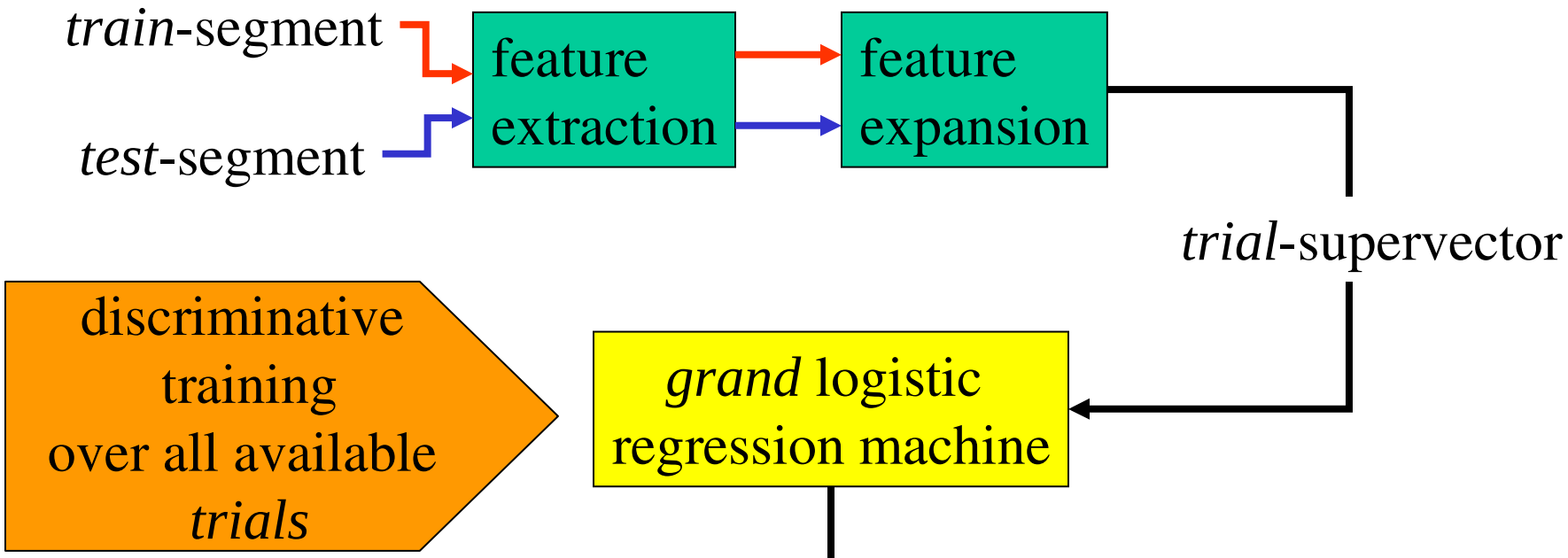
- Expanded features
- Objective function
- Optimization procedure
- **Grand logistic regression**
- Piggyback fusion
- Linear fusion

State-of-the-art discriminative *speaker* recognition



but we tried something different…

Discriminative *detection-trial* recognition



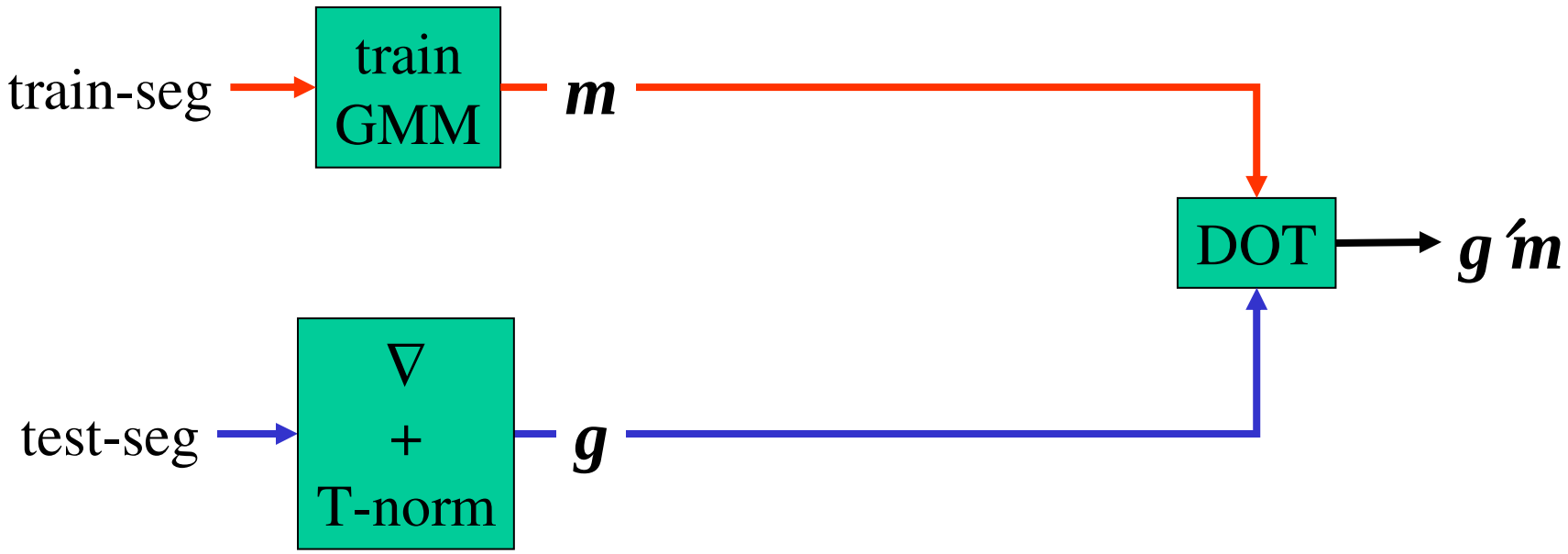
can this solve the channel problem?

The ideal

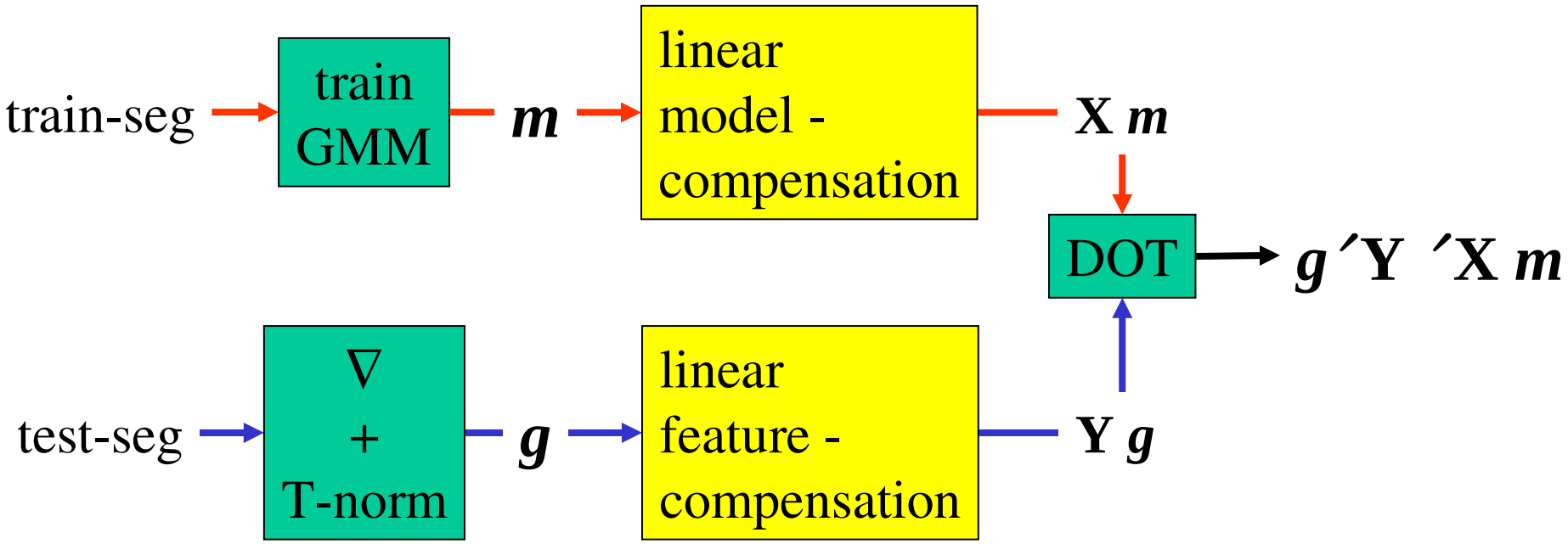
- The ideal was to create a *monolithic discriminative* approach to solve the whole speaker detection problem, including the channel mismatch problem.
- In other words, we tried to create a *streamlined* system that looks like this:

Instead it turned out like this:

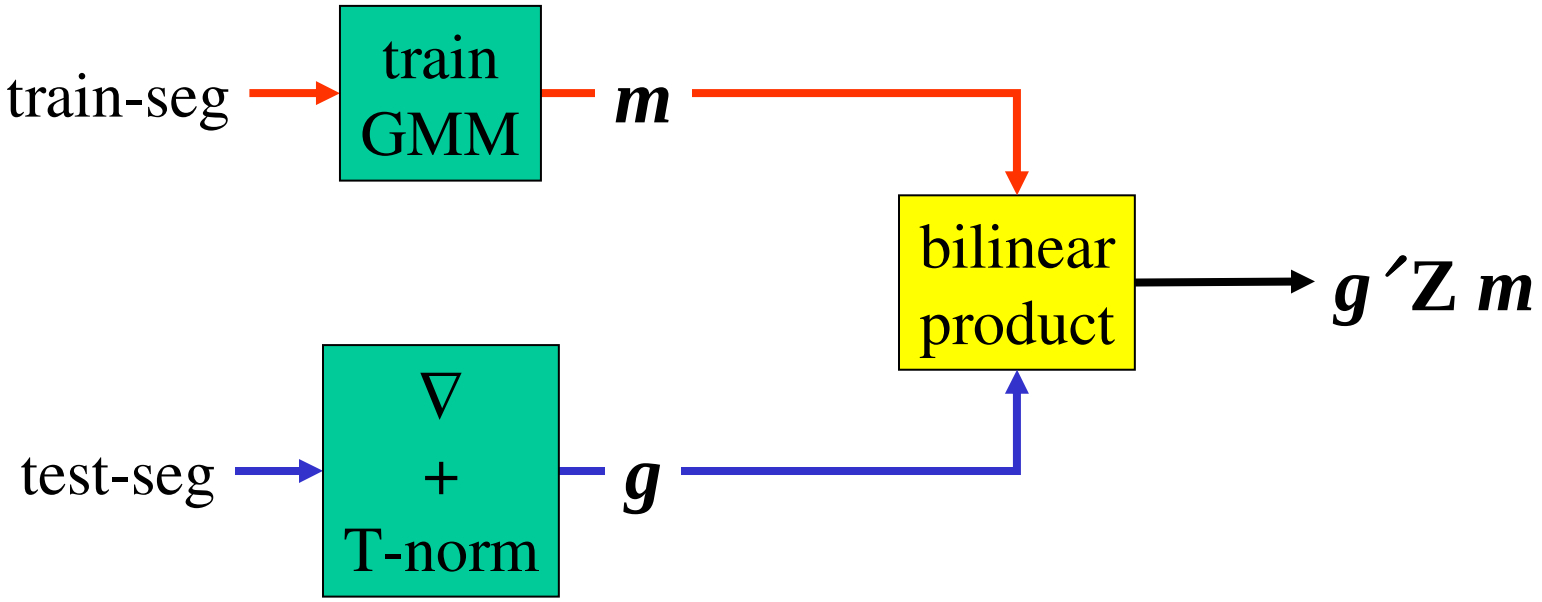
Grand Log.Reg. Baseline



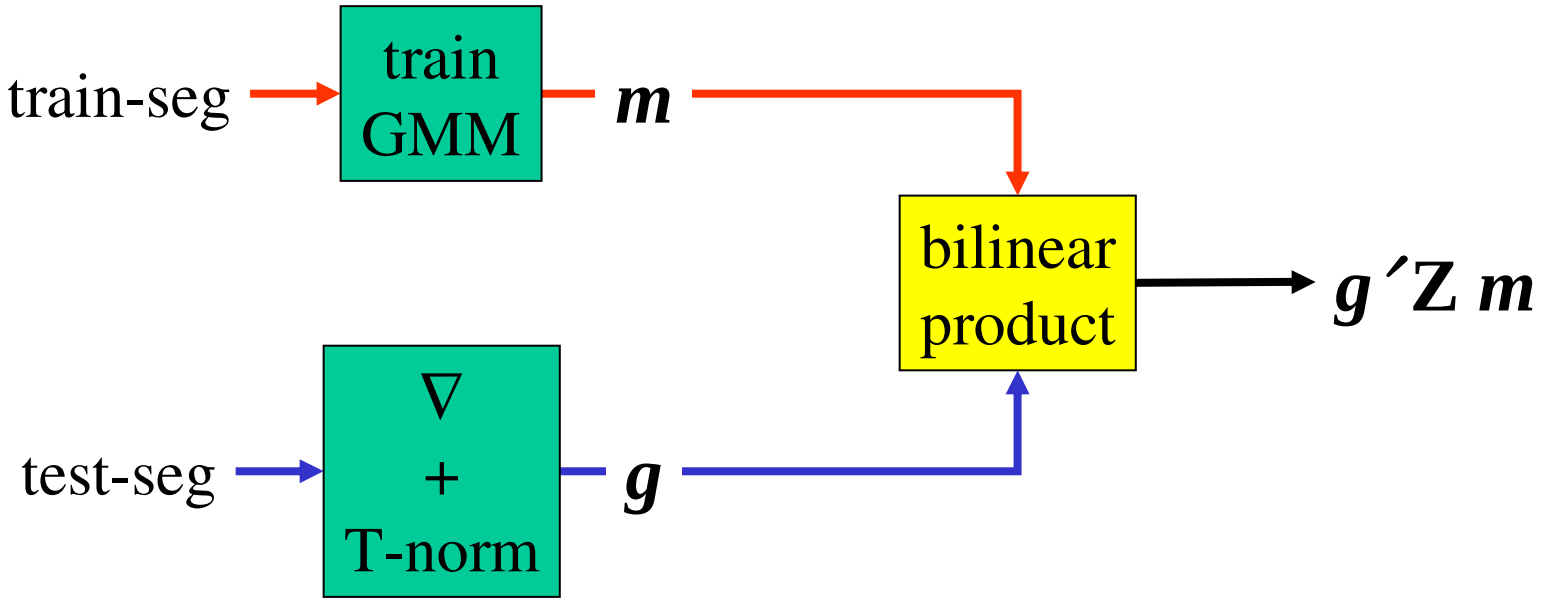
Grand Log.Reg. Construction



Grand Log.Reg.



Grand Log.Reg.



training problem $\equiv$ Find the 150 million elements of $\mathbf{Z}$

Finding $\mathbf{Z}$,
subject to *regularization* (to combat overtraining)

$\equiv$ (by the *Representer Theorem*)

Finding $\{ \alpha_i \}$ where:

$$g' Zm = \sum_{i \in \{\text{training trials}\}} \alpha_i K_{bilinear}\left((g,m),(g_i,m_i)\right)$$

$$K_{bilinear} = (g'g_i)(m'm_i)$$

Grand Logistic Regression

In summary:

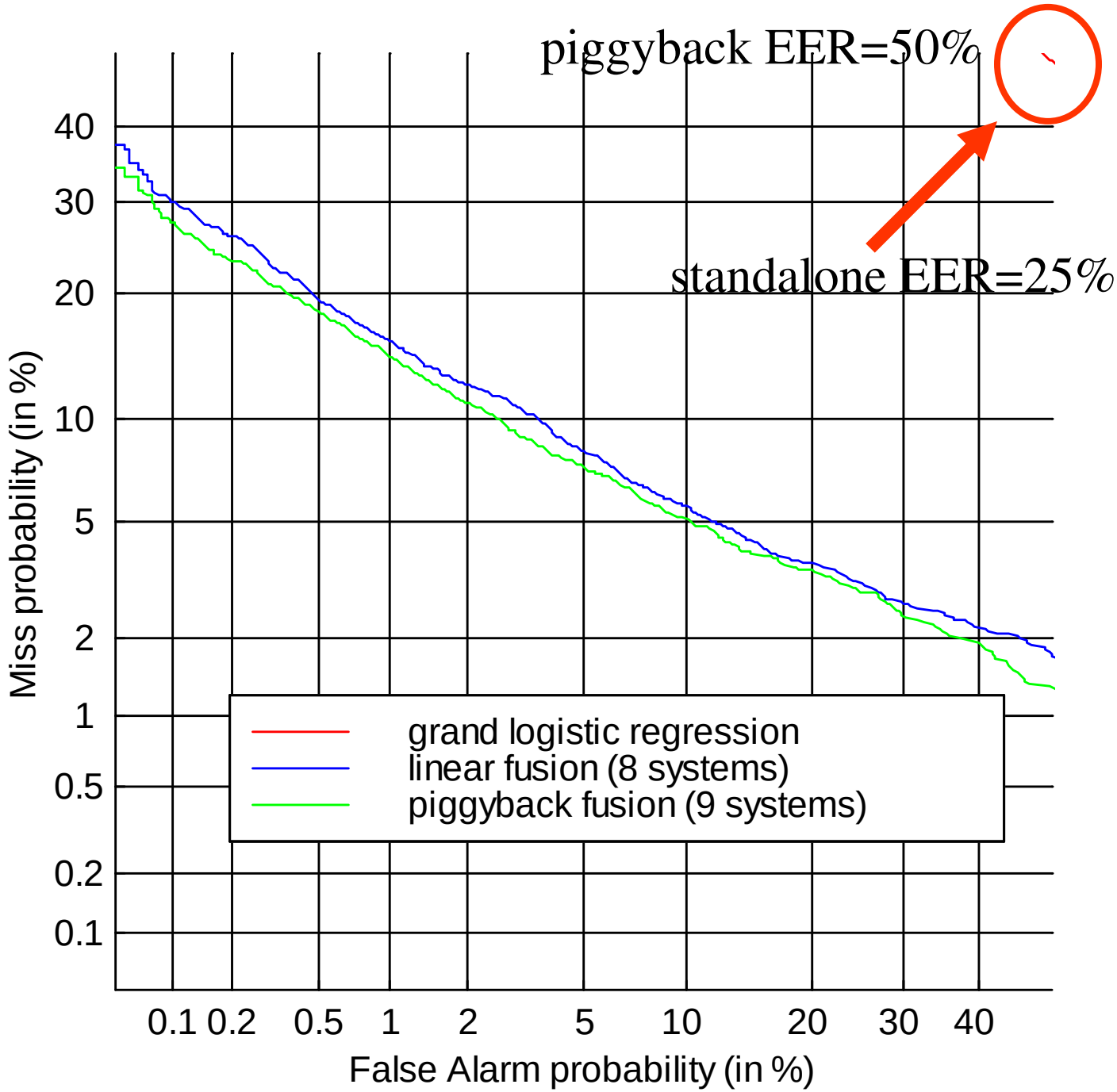
- This is an attempt to improve a linearized (t-normed) GMM score via further linear transformations in supervector space.
- The linear transformations are effected by *regularized kernel logistic regression*.
- Does it work?

Does it work?

- *No*. Performance was much worse (EER=25%) than the baseline dot-product which corresponds to $\mathbf{Z} = \mathbf{I}$. This suggests it may have fared better with more training examples (which were unavailable.)
- *Fortunately*, we found a way to make it pay anyway:
- Instead of using GLR on its own, we trained it to be a fusion *complement* to an already good system. In this case it provides some improvement above this good system. We called this strategy *piggyback fusion*.

New Discriminative Approaches

- Expanded features
- Objective function
- Optimization procedure
- Grand logistic regression
- **Piggyback fusion**
- Linear fusion


piggyback EER=50%
standalone EER=25%
Miss 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
False Alarm probability (in %)
grand logistic regression
linear fusion (8 systems)
piggyback fusion (9 systems)

Grand Logistic Regression Summary

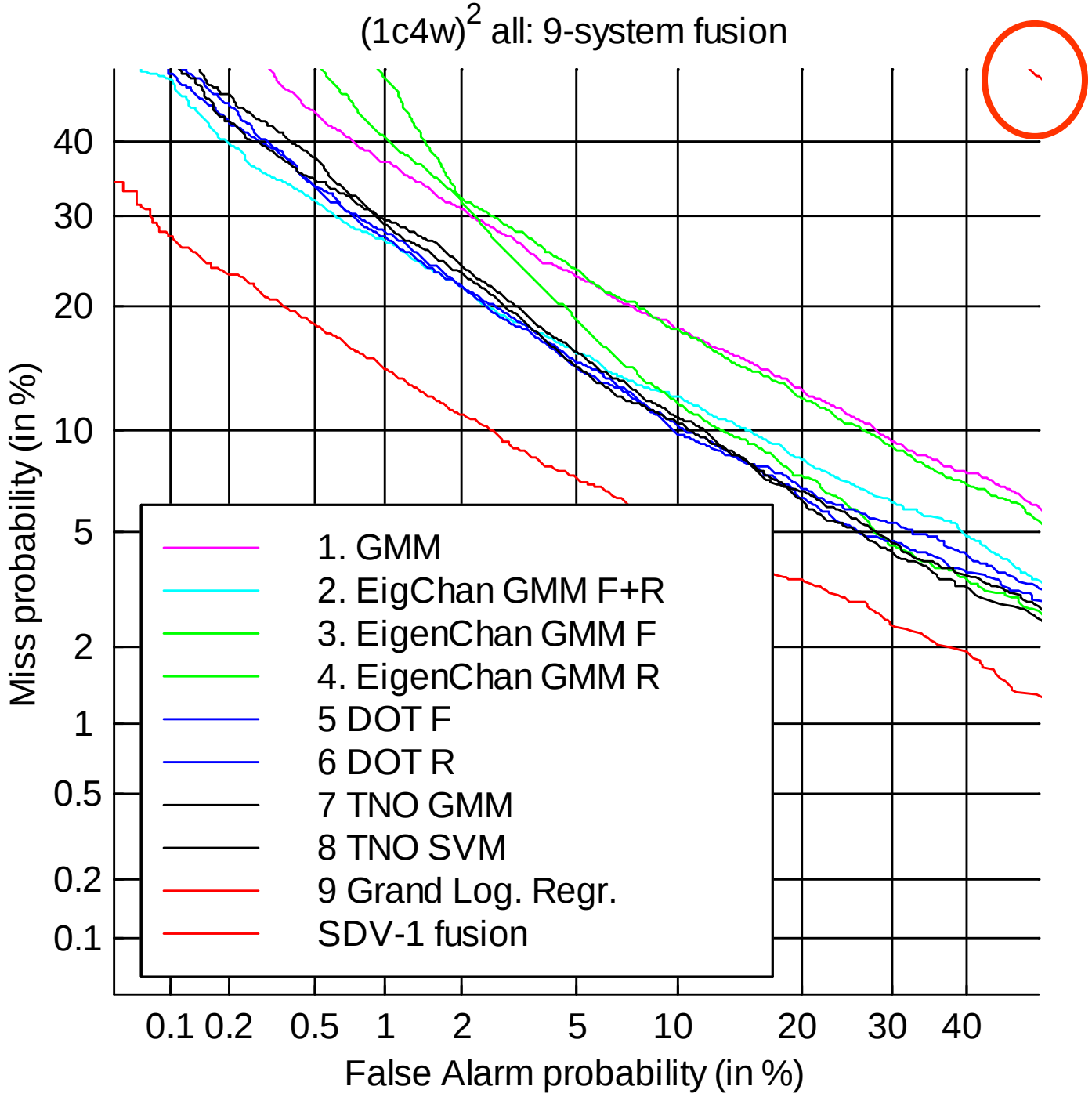
- Tricky to get to work:
  - choice of feature expansion
  - normalization?
  - choice of kernel
  - regularization constant
  - does *not* give a well-calibrated log-likelihood-ratio score.
  - training can take very long
  - poor performance (20-25% EER) on its own (so far)
- *But*, with some effort, can probably be used via piggyback to improve almost any existing system.

New Discriminative Approaches

- Expanded features
- Objective function
- Optimization procedure
- Grand logistic regression
- Piggyback fusion
- **Linear fusion**

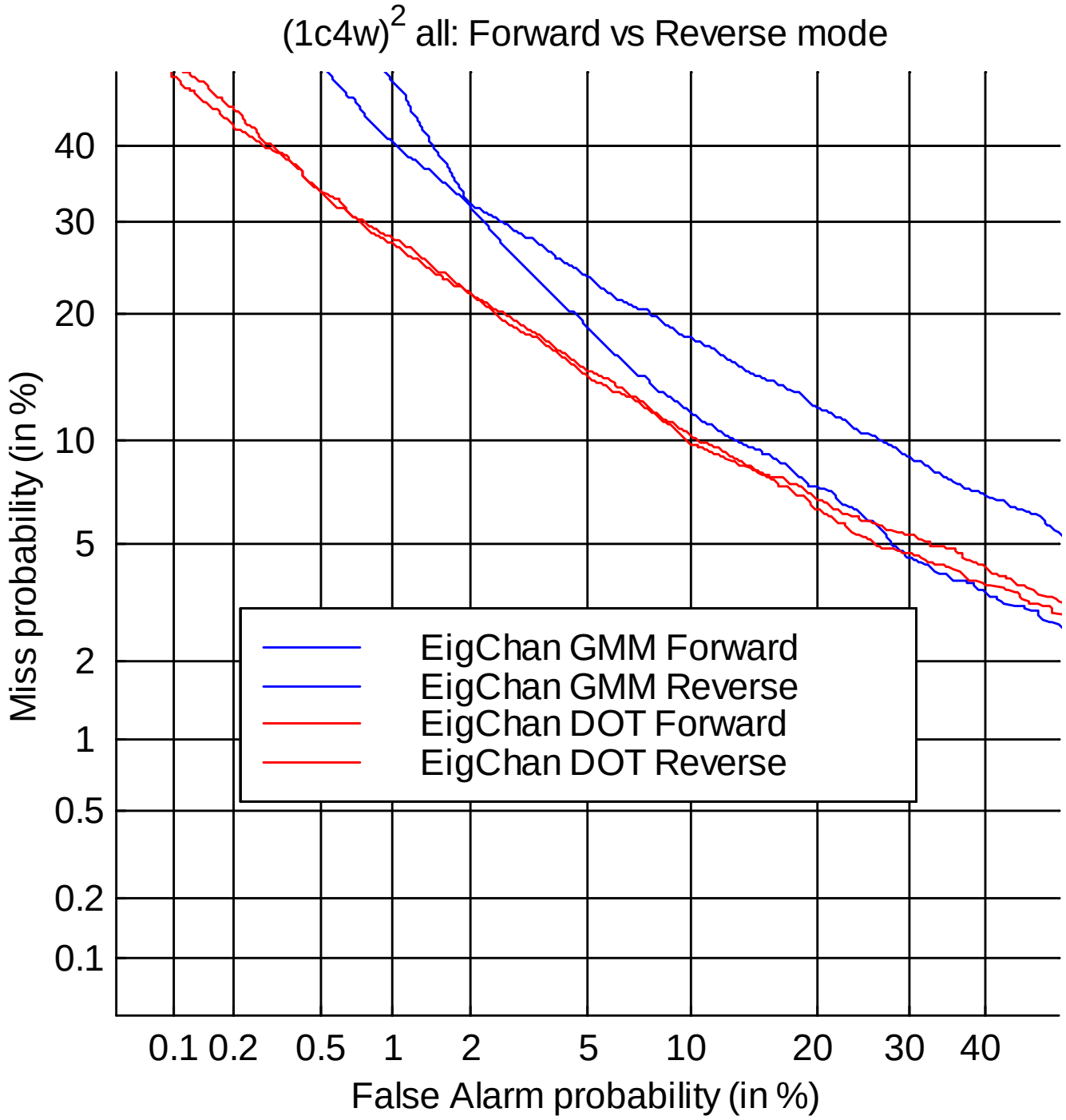
Linear logistic regression fusion

- Easy to implement. (It just works.)
- Not regularized (no constant to choose).
- Training is very fast.
- Gives a *well-calibrated* log-likelihood-ratio score.
- Results seem stable between development and test data.
- See system description for details.
- See next slide for summary of fused systems …


(1c4w)2 all: 9-system fusion
Miss 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
False Alarm probability (in %)
1. GMM
2. EigChan GMM F+R
3. EigenChan GMM F
4. EigenChan GMM R
5 DOT F
6 DOT R
7 TNO GMM
8 TNO SVM
9 Grand Log. Regr.
SDV-1 fusion

Note on asymmetry of eval. data

- Are 1C4W (or 1-side) *test*- and *train*- segments the same thing?
- According to our experiments on 2004 and 2005 data: *No*.
- For our GMM systems, we get a large difference between:
  - **Forward:** Train model on *train*- and score on *test*-segment.
  - **Reverse:** Train model on *test*- and score on *train*-segment.
- (With the *dot-product* approximation, this is not the case.)


(1c4w)2 all: Forward vs Reverse mode
Miss 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
False Alarm probability (in %)
EigChan GMM Forward
EigChan GMM Reverse
EigChan DOT Forward
EigChan DOT Reverse

Asymmetry

- Some part of the fusion success is probably attributable to the fact that this asymmetry had a similar nature in the 2004 and 2005 data sets.
  - Fusion weights for the reverse systems were smaller than for the forward systems.

Calibration

- We relied on linear logistic regression to give scores that are *well-calibrated* log-likelihood-ratios.
- We performed no special optimization for the NIST operating point.
- We chose no special decision thresholds. We simply used the theoretical log-likelihood-ratio threshold of **log 9.9 ≈ 2.29**.

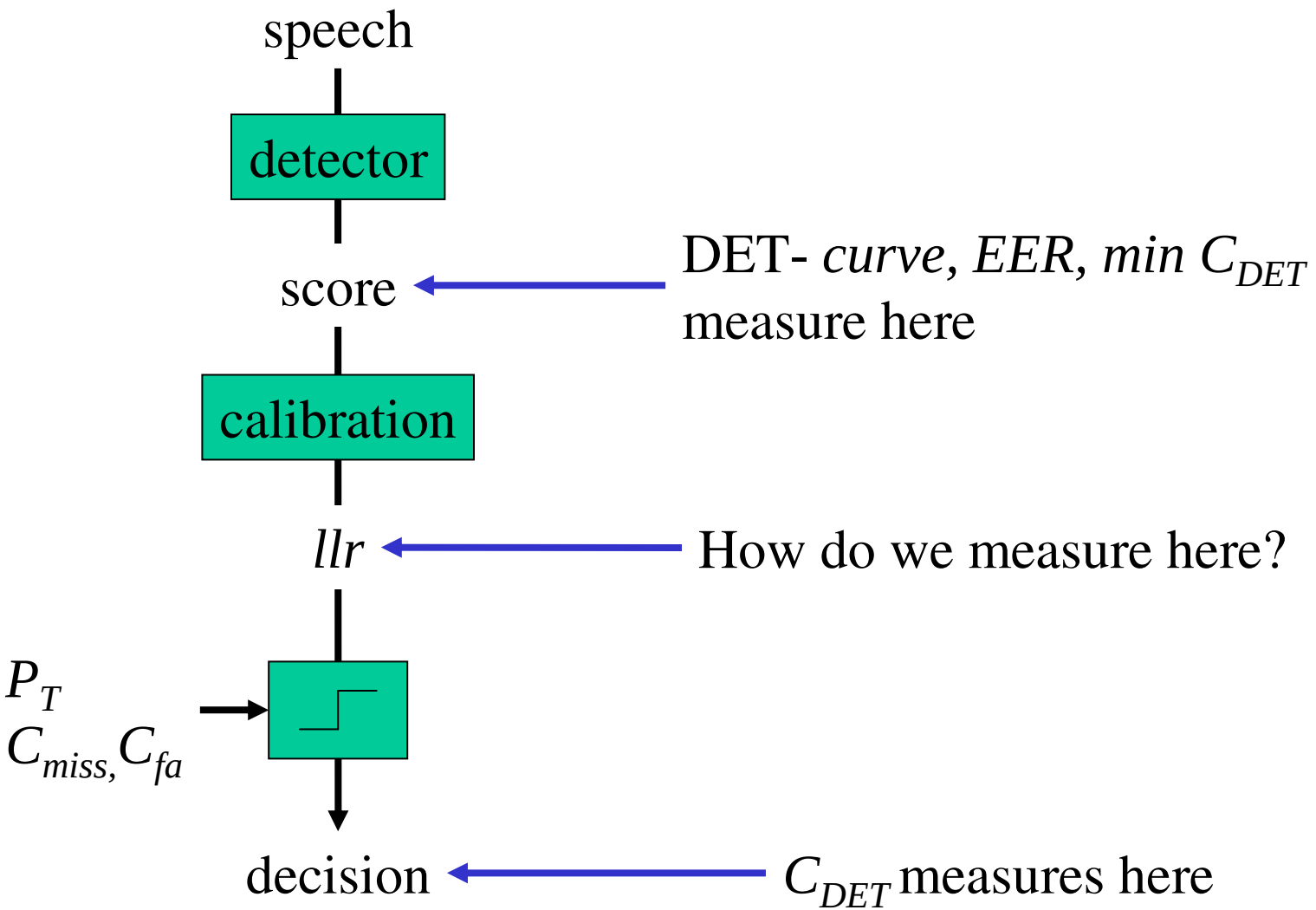
Calibration

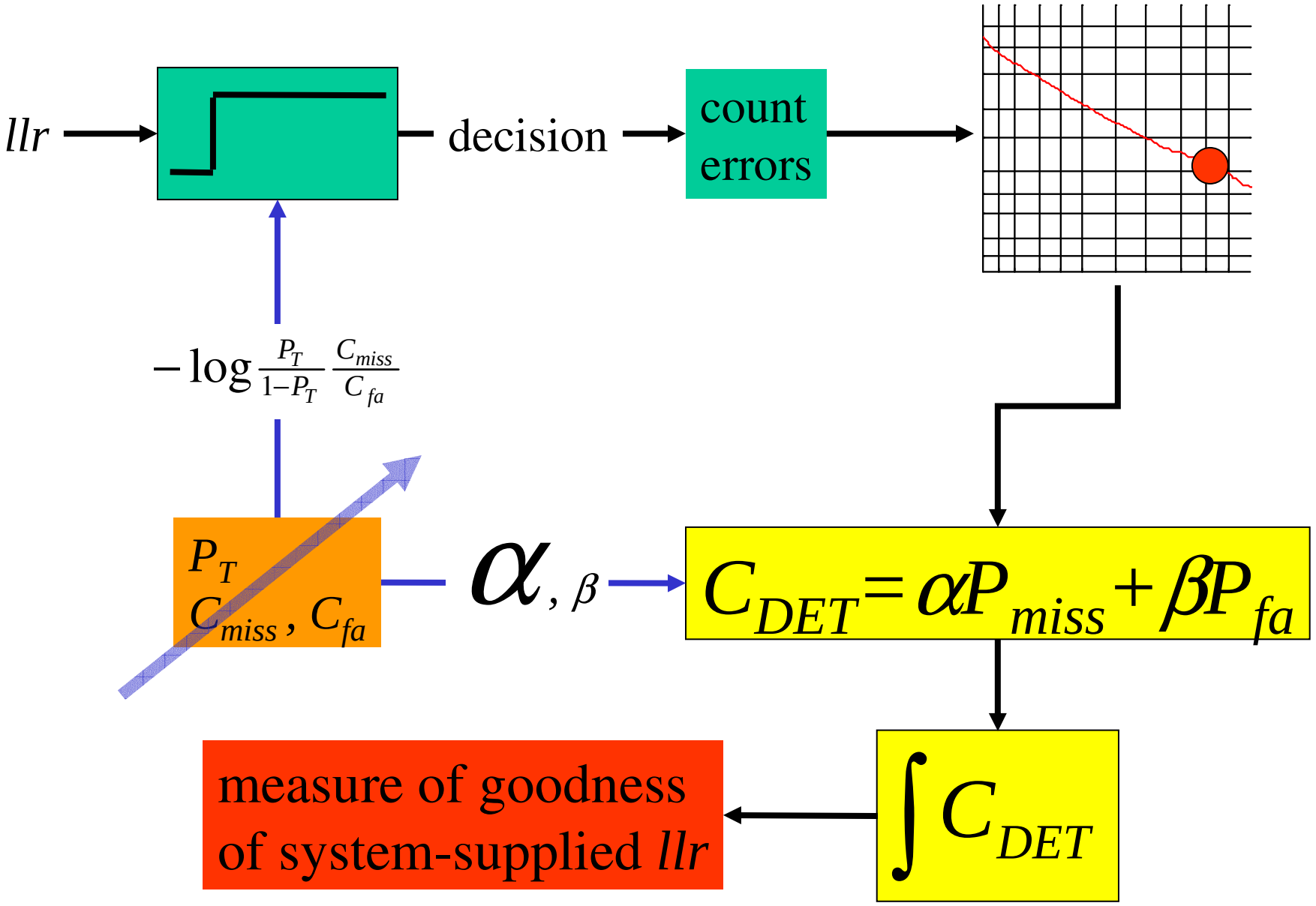
- This linear fusion to give a log-likelihood-ratio is similar to what MIT did this year. The difference is they used a mean-squared-error optimization objective. The MSE can be biased so that it *does* optimize for the NIST operating point. (Indeed MIT found MSE to give better C_{DET} and EER than logistic regression.)

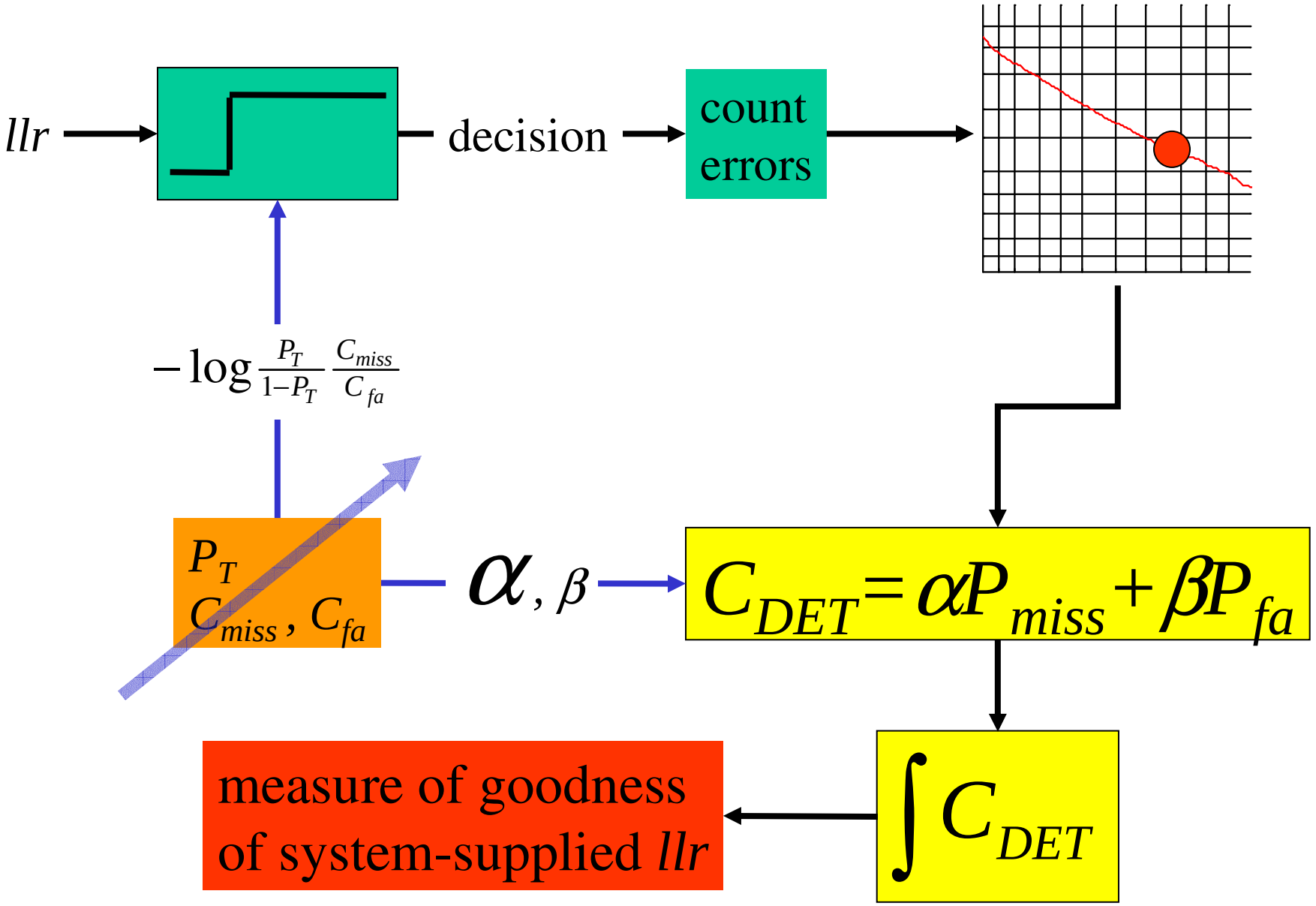
Calibration

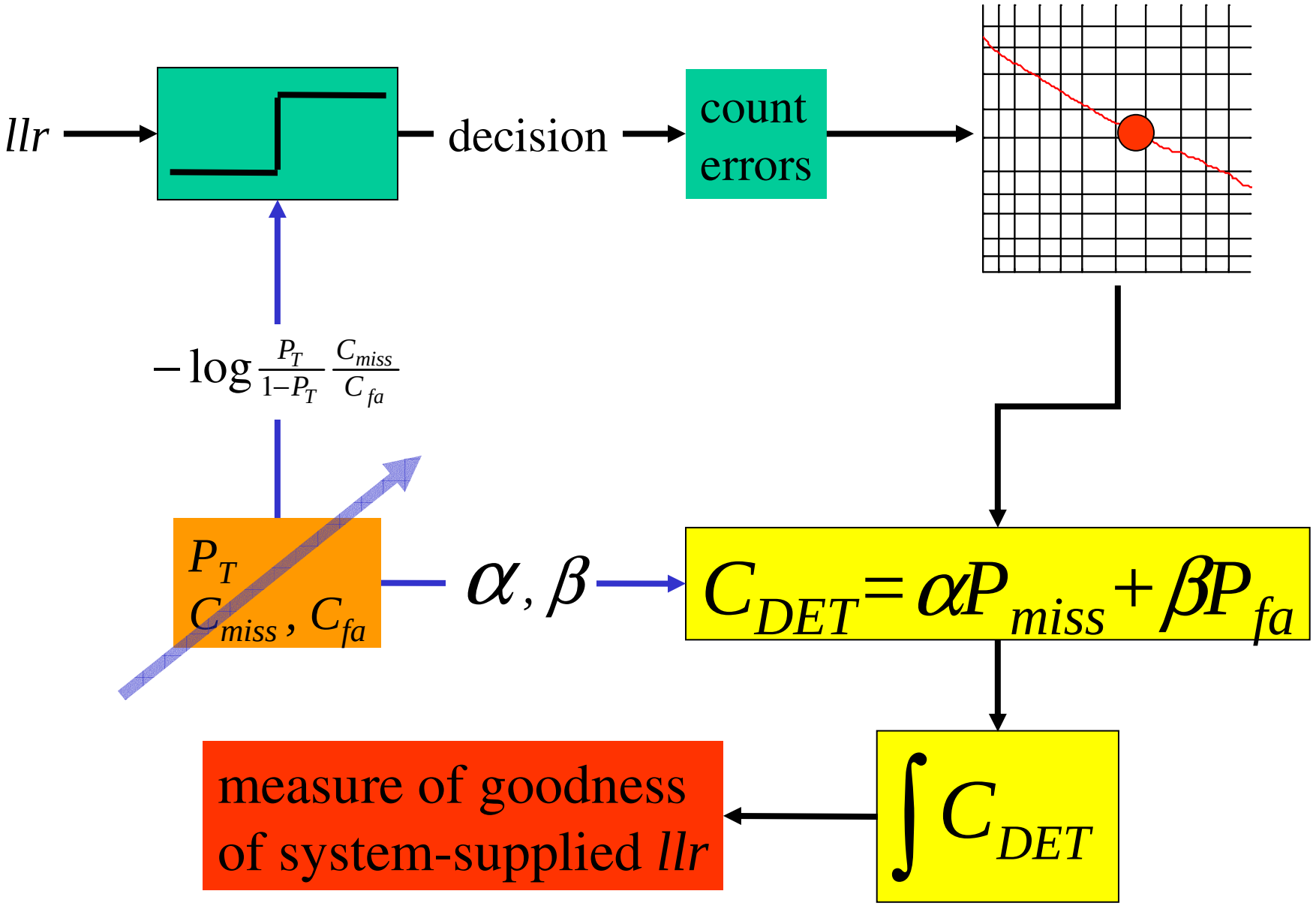
- The reason we stuck with logistic regression instead of MSE, was that for practical applications of our system, we are interested not only in the NIST operating point. We want to be able to apply our system for as wide a range of applications as possible.
- For this purpose, we argue that the logistic regression objective (a.k.a. *cross-entropy*) is more suitable.
- For detailed comparison between MSE and cross-entropy see (Brummer, Computer Speech and Language, 2005).

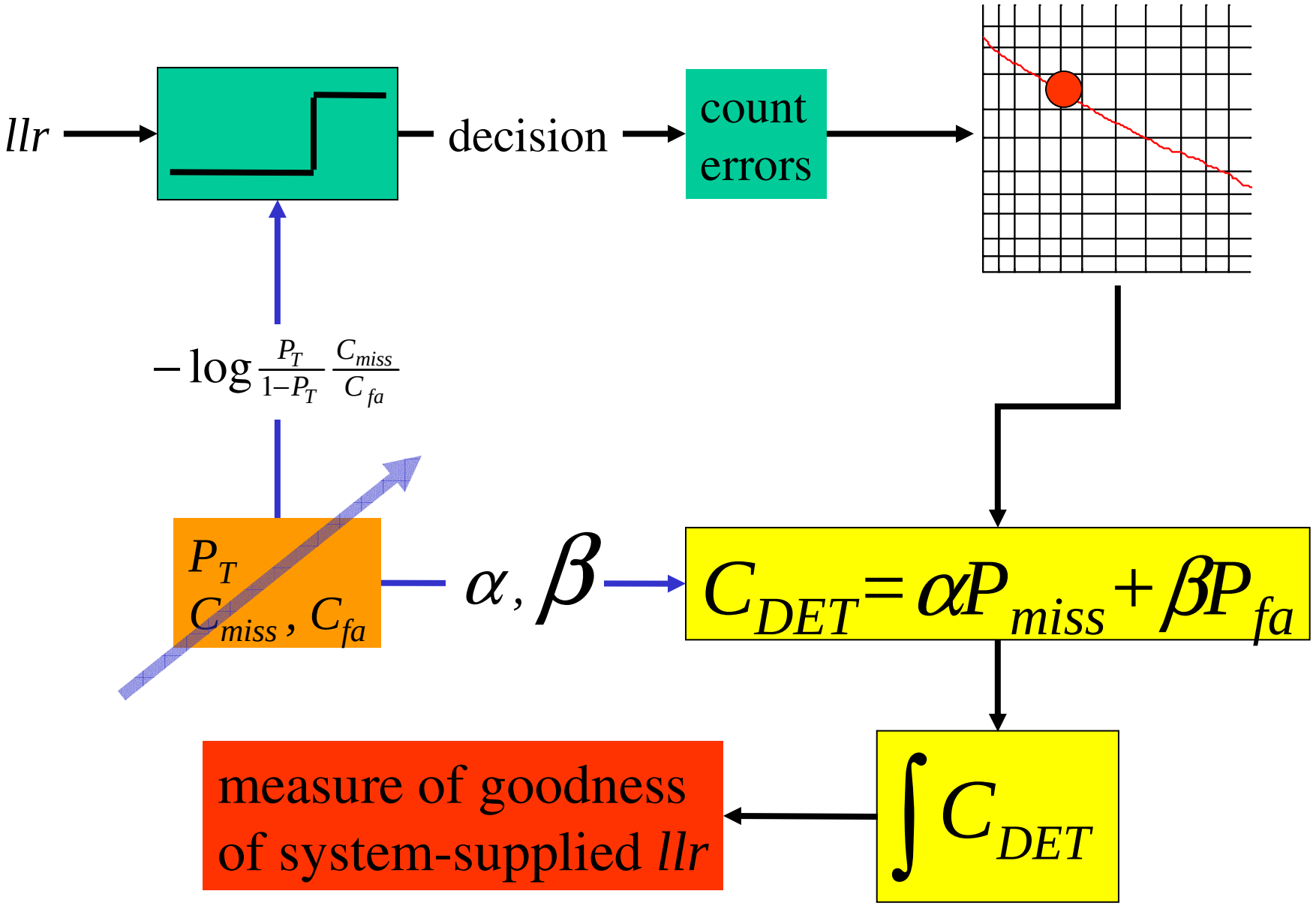
Evaluation of quality of log-likelihood-ratio (*llr*)

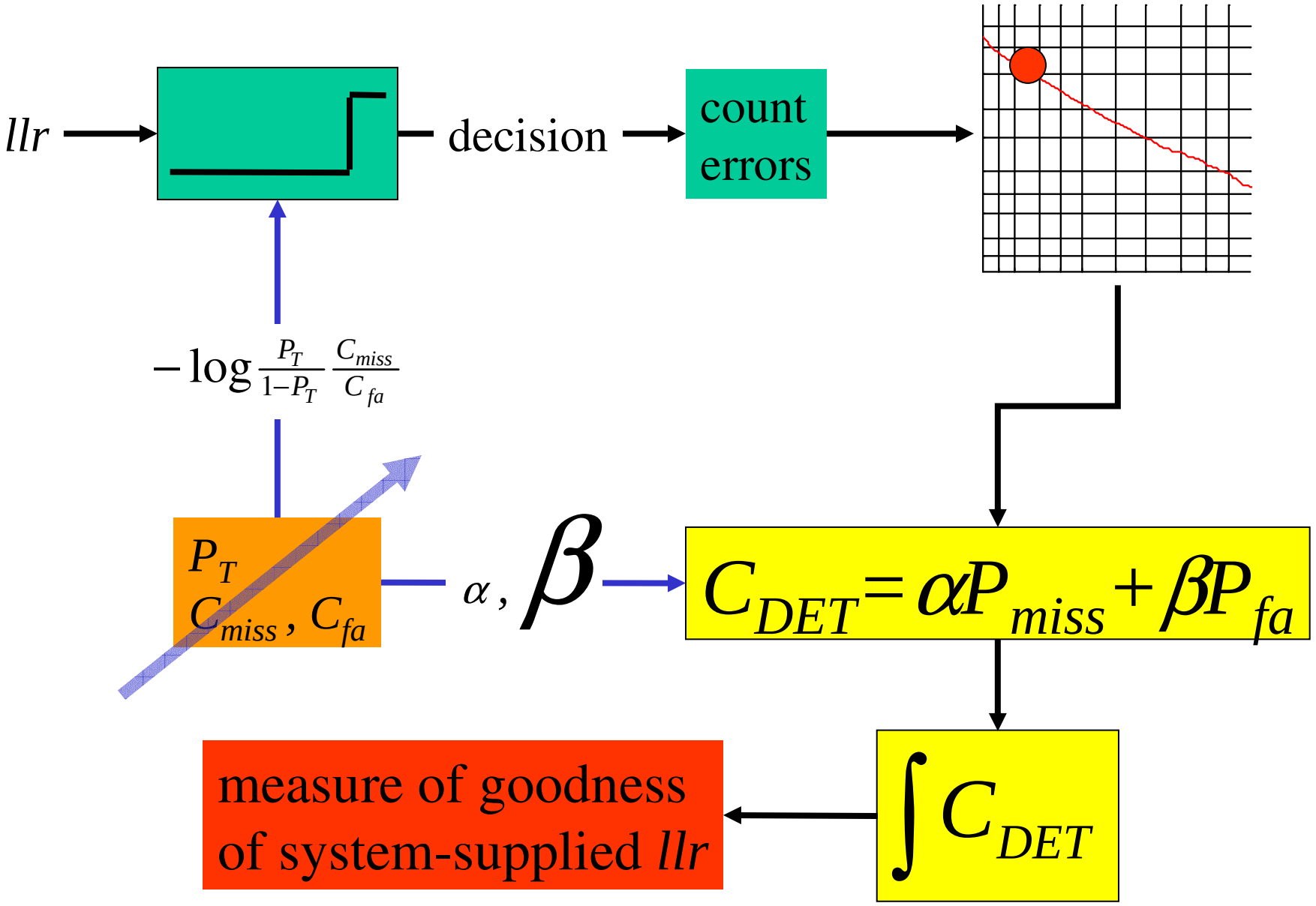


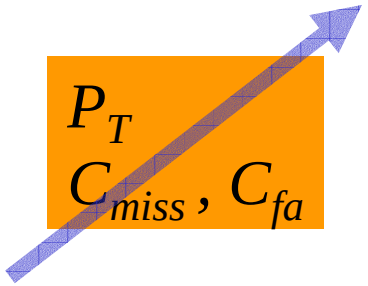

llr
decision
count errors
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{DET} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{DET}$
measure of goodness of system-supplied *llr*


llr
decision
count errors
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{DET} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{DET}$
measure of goodness of system-supplied *llr*


llr
decision
count errors
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{DET} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{DET}$
measure of goodness of system-supplied *llr*


llr
decision
count errors
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{DET} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{DET}$
measure of goodness of system-supplied *llr*


llr
decision
count errors
$-\log \frac{P_T}{1-P_T} \frac{C_{miss}}{C_{fa}}$
P_T
C_{miss}, C_{fa}
α, β
$C_{DET} = \alpha P_{miss} + \beta P_{fa}$
$\int C_{DET}$
measure of goodness of system-supplied *llr*

How do we adjust  ?

- It turns out a *one-dimensional* adjustment is sufficient:

$$P_T = \frac{1}{2} \quad , C_{miss} = \frac{1}{\theta} \quad , \quad C_{fa} = \frac{1}{1-\theta} \quad , \quad 0 < \theta < 1$$

or

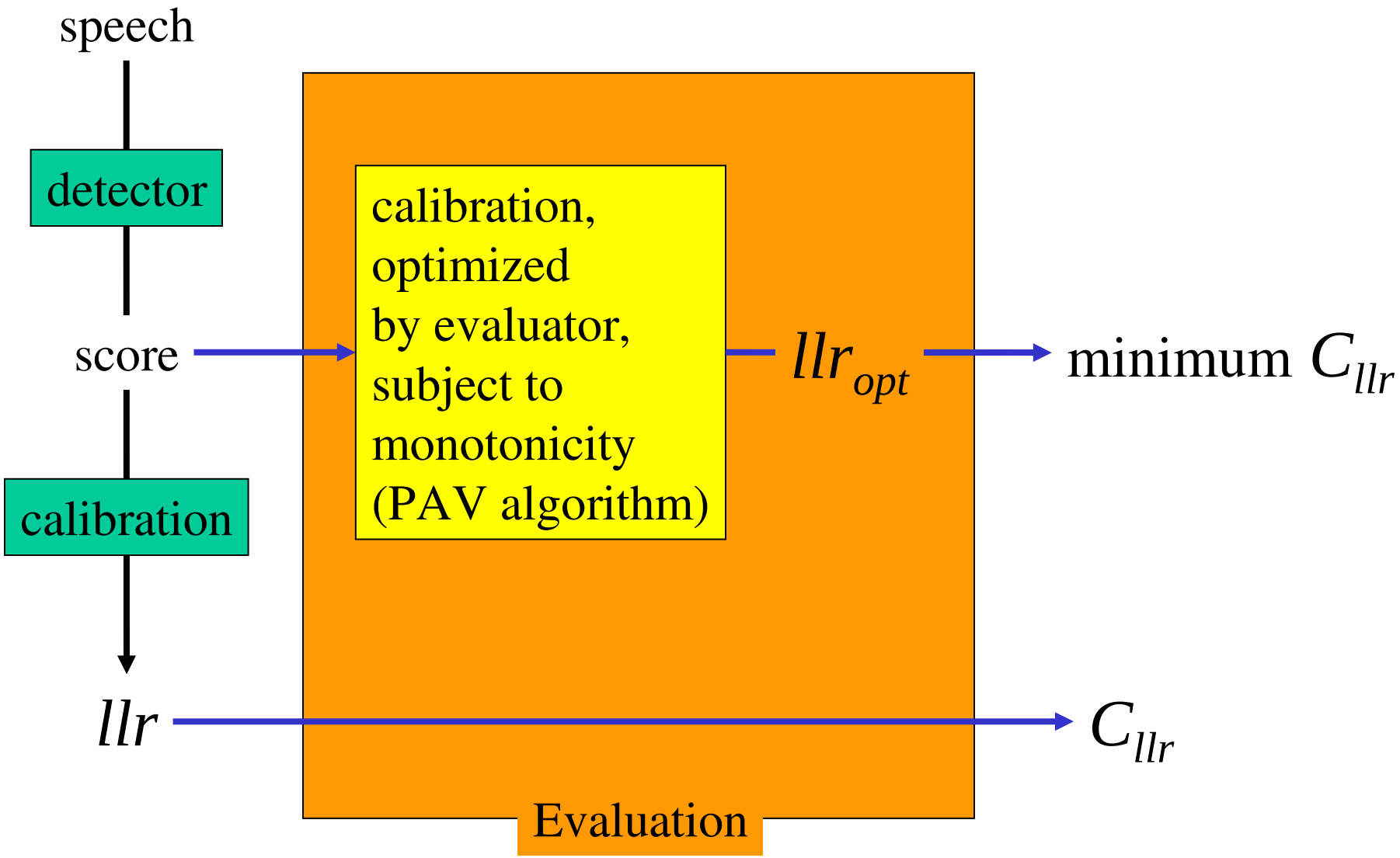
$$C_{miss} = C_{fa} = 1 \quad , \quad P_T = \text{logit}^{-1}\,\phi \quad , \quad -\infty < \phi < \infty$$

$$C_{llr} \equiv \int_0^1 C_{DET}(\theta)\, d\theta = \int_{-\infty}^{\infty} C_{DET}(\phi)\, d\phi$$

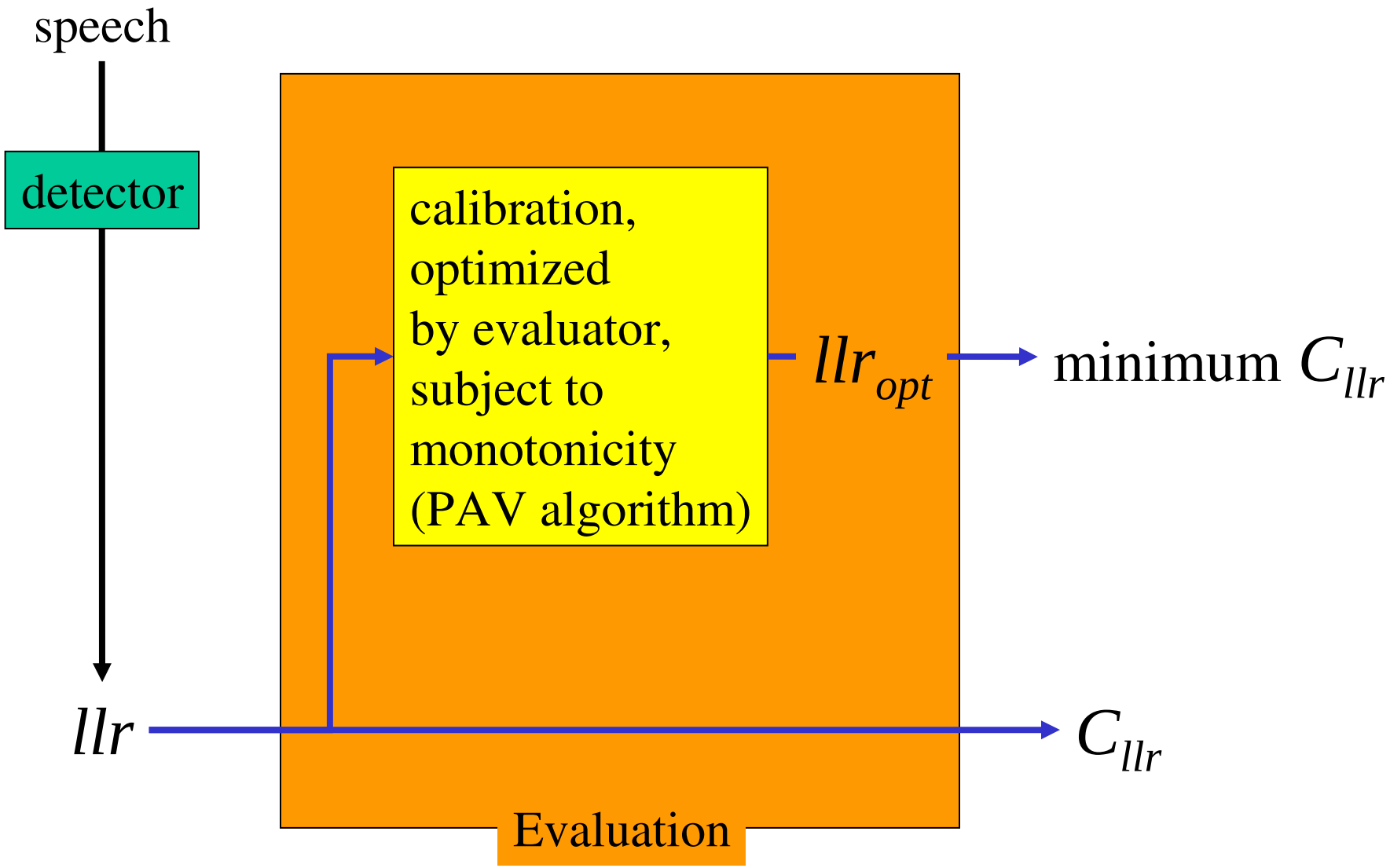
C_{llr} is:

- (minus) *Information* delivered to user by system
- *Expected cost* of using system (over different applications)
- *Total error-rate* (over different applications)
- (minus) log-likelihood of the eval. answer-key
- Logistic regression objective
- a.k.a cross-entropy
- Subject of my Odyssey ‘04 and CSL ‘05 papers.

C_{llr} evaluation of the *score*



(*llr* is also a *score*)



APE-plots

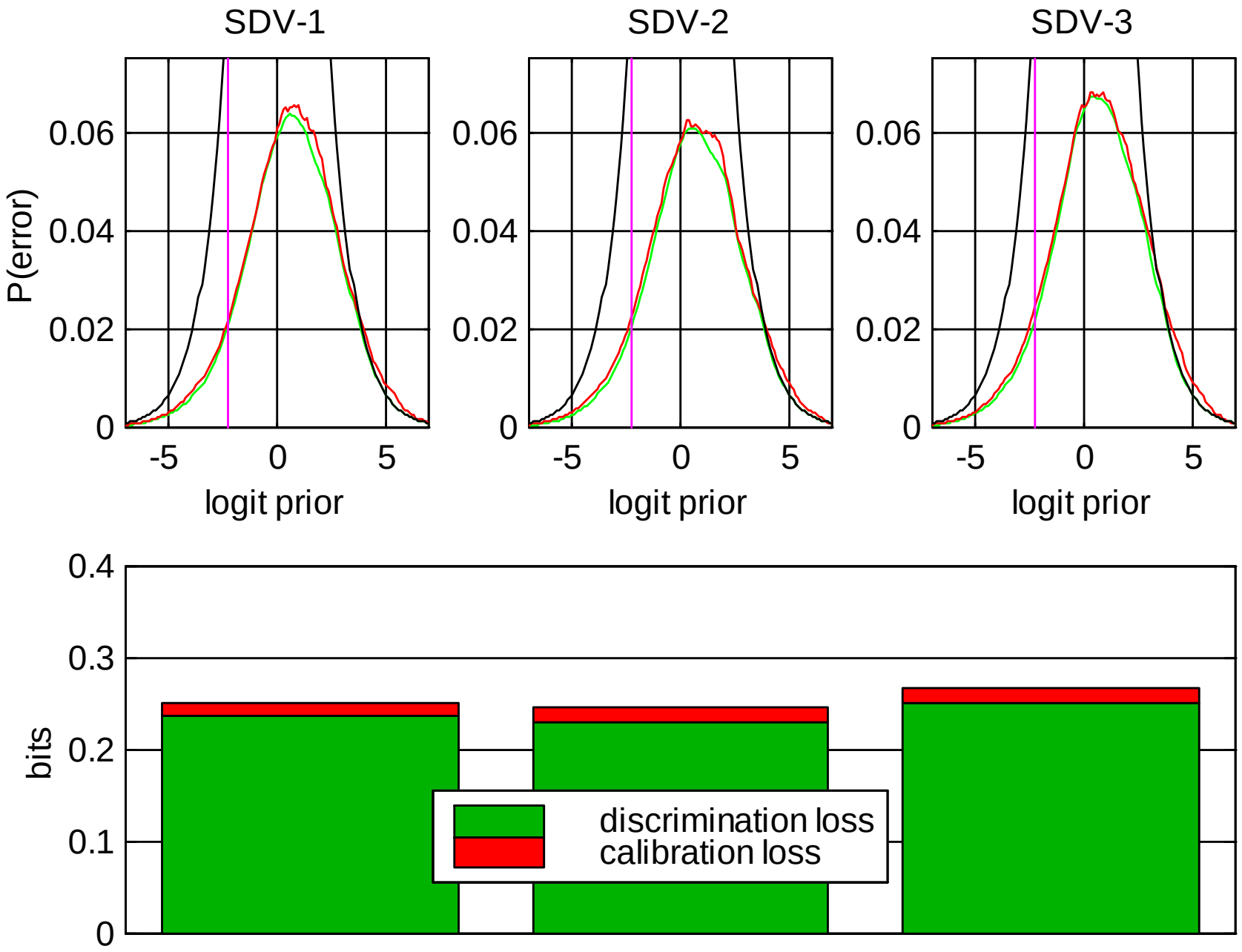
- The interpretation of C_{llr} as an integral over error-rates, has a useful graphic representation, which we called:

  Applied Probability of Error (APE) plot.

- When used for *llr* evaluation it gives information that is complementary to the *DET*-plot.
- We used *APE*-plots in addition to *DET*-plots to make all of our development decisions.

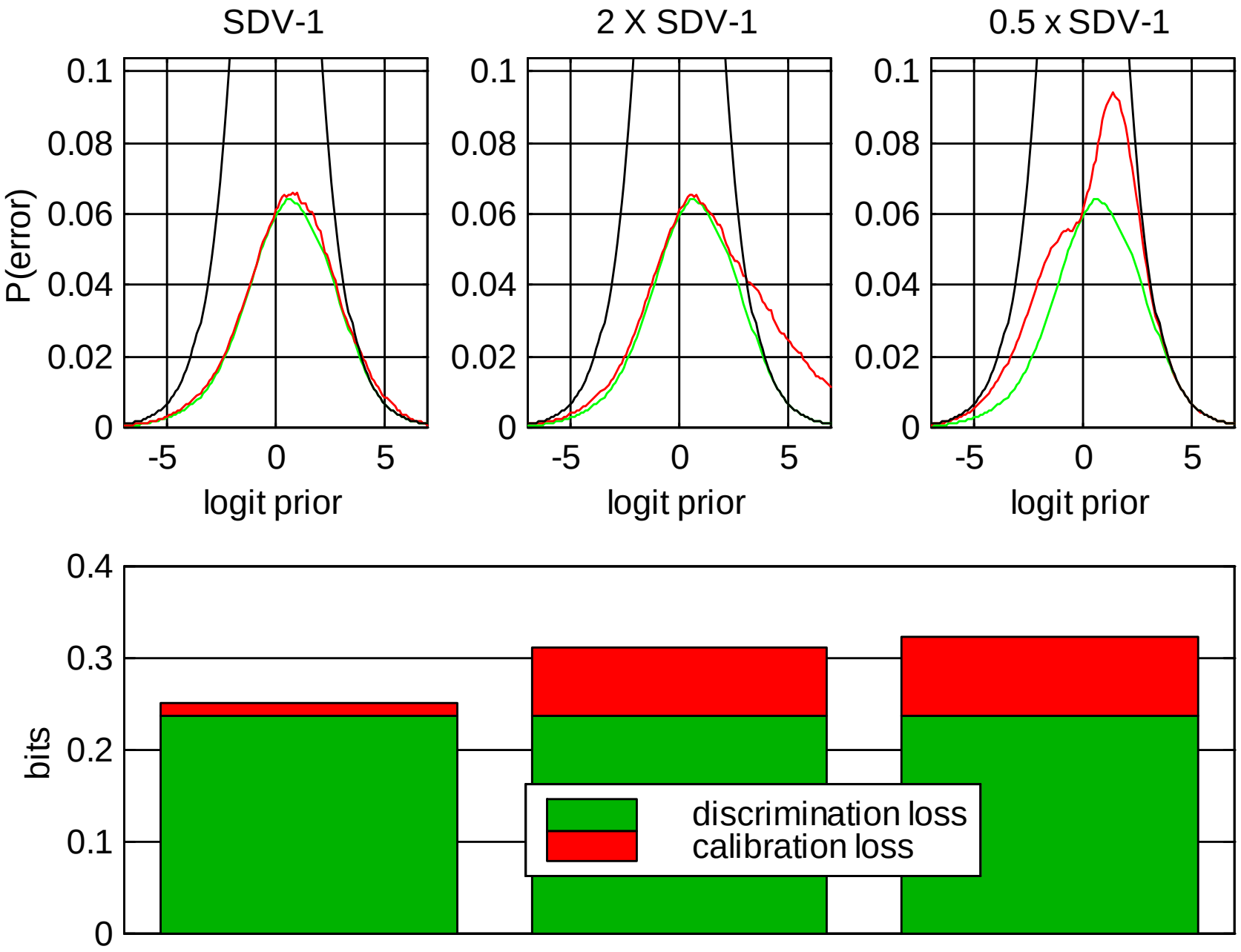
APE-plots show:

- *EER = max of green curve*
- (scaled) C_{DET} and *min* C_{DET}
  = values of red and green curves at –2.29
- C_{llr} = area under red curve
- *minimum* C_{llr} = area under green curve
- *Calibration loss* = area between curves
- Example: SDV-1, SDV-2 and SDV-3:


SDV-1
SDV-2
SDV-3
P(error)
0.06
0.04
0.02
0
-5
0
5
logit prior
0.4
0.3
0.2
0.1
bits
discrimination loss
calibration loss

Examples of bad calibration

- SDV-1 : (not damaged)
- SDV-1 : *llr* multiplied by 2
- SDV-1 : *llr* divided by 2


SDV-1
2 X SDV-1
0.5 x SDV-1
P(error)
0.1
0.08
0.06
0.04
0.02
0
-5
0
5
logit prior
bits
0.4
0.3
0.2
0.1
discrimination loss
calibration loss

Note: *ACE-plot*

- Very similar, and dual to the *APE*-plot is the

  Applied Cost of Error (ACE)-plot

- based on the *expected cost* interpretation of C_{llr} .

Conclusion

- Generative methods of channel compensation formed the mainstay:
  - Eigenchannel (SDV)
  - Feature mapping (TNO)
- But these were complemented, fused and calibrated with new discriminative methods.