Evaluation of classifier accuracy

Tomi Kinnunen

tkinnu@cs.uef.fi

University of Eastern Finland

UEF summer school 2017

Neural Network Follies

by Neil Fraser, September 1998

In the 1980s, the Pentagon wanted to harness computer technology to make their tanks harder to attack.

The Plan

The preliminary plan was to fit each tank with a digital camera hooked up to a computer. The computer would continually scan the environment outside for possible threats (such as an enemy tank hiding behind a tree), and alert the tank crew to anything suspicious. Computers are really good at doing repetitive tasks without taking a break, but they are generally bad at interpreting images. The only possible way to solve the problem was to employ a neural network.

The Implementation

The research team went out and took 100 photographs of tanks hiding behind trees, and then took 100 photographs of trees - with no tanks. They took half the photos from each group and put them in a vault for safe-keeping, then scanned the other half into their mainframe computer. The huge neural network was fed each photo one at a time and asked if there was a tank hiding behind the trees. Of course at the beginning its answers were completely random since the network didn't know what was going on or what it was supposed to do. But each time it was fed a photo and it generated an answer, the scientists told it if it was right or wrong. If it was wrong it would randomly change the weightings in its network until it gave the correct answer. Over time it got better and better until eventually it was getting each photo correct. It could correctly determine if there was a tank hiding behind the trees in any one of the photos.

Verification

But the scientists were worried: had it actually found a way to recognize if there was a tank in the photo, or had it merely memorized which photos had tanks and which did not? This is a big problem with neural networks, after they have been trained you have no idea how they arrive at their answers, they just do. The question was did it understand the concept of tanks vs. no tanks, or had it merely memorized the answers? So the scientists took out the photos they had been keeping in the vault and fed them through the computer. The computer had never seen these photos before -- this would be the big test. To their immense relief the neural net correctly identified each photo as either having a tank or not having one.

Independent testing

The Pentagon was very pleased with this, but a little bit suspicious. They commissioned another set of photos (half with tanks and half without) and scanned them into the computer and through the neural network. The results were completely random. For a long time nobody could figure out why. After all nobody understood how the neural had trained itself.

Grey skies for the US military

Eventually someone noticed that in the original set of 200 photos, all the images with tanks had been taken on a cloudy day while all the images without tanks had been taken on a sunny day. The neural network had been asked to separate the two groups of photos and it had chosen the most obvious way to do it - not by looking for a camouflaged tank hiding behind a tree, but merely by looking at the colour of the sky. The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not.

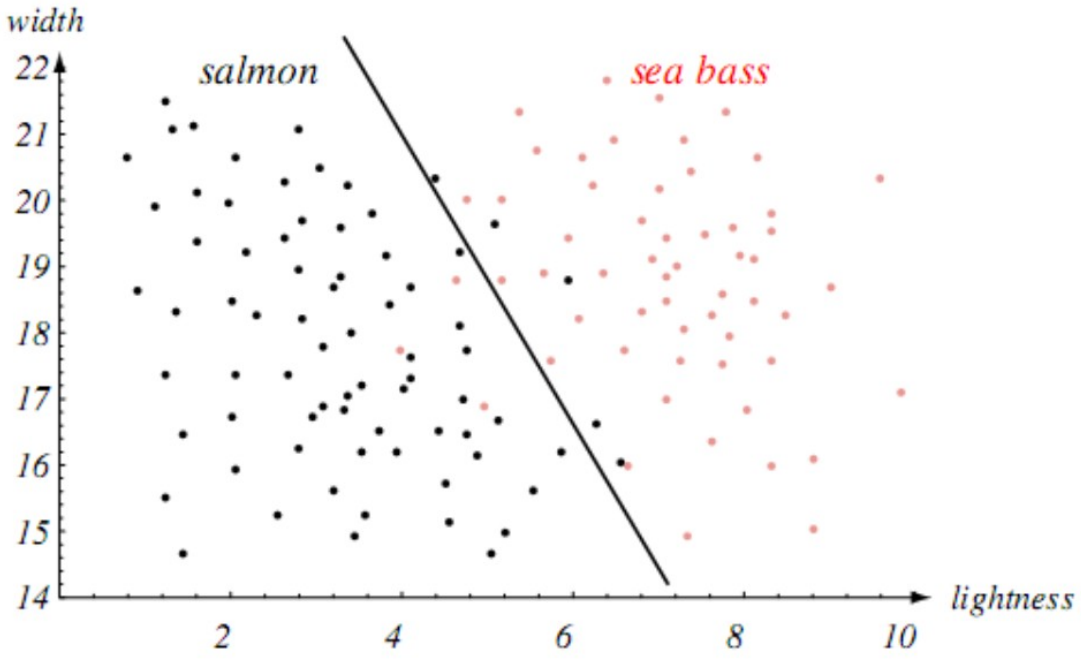
---

*This story might be apocryphal, but it doesn't really matter. It is a perfect illustration of the biggest problem behind neural networks. Any automatically trained net with more than a few dozen neurons is virtually impossible to analyze and understand. One can't tell if a net has memorized inputs, or is 'cheating' in some other way. A promising use for neural nets these days is to predict the stock market. Even though initial results are extremely good, investors are leery of trusting their money to a system that **nobody** understands.*

Two key things

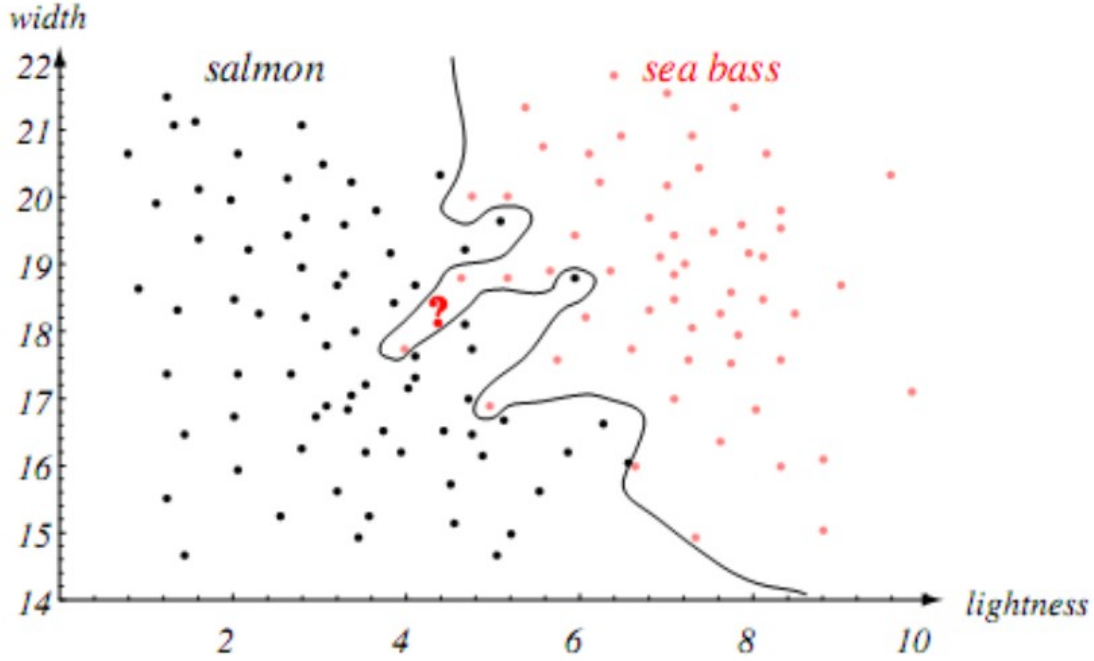
1. CHOICE OF DATA PARTITIONING
2. CHOICE OF THE EVALUATION METRIC

Overfitting vs. underfitting



MODEL WITH A SMALL NUMBER OF PARAMETERS

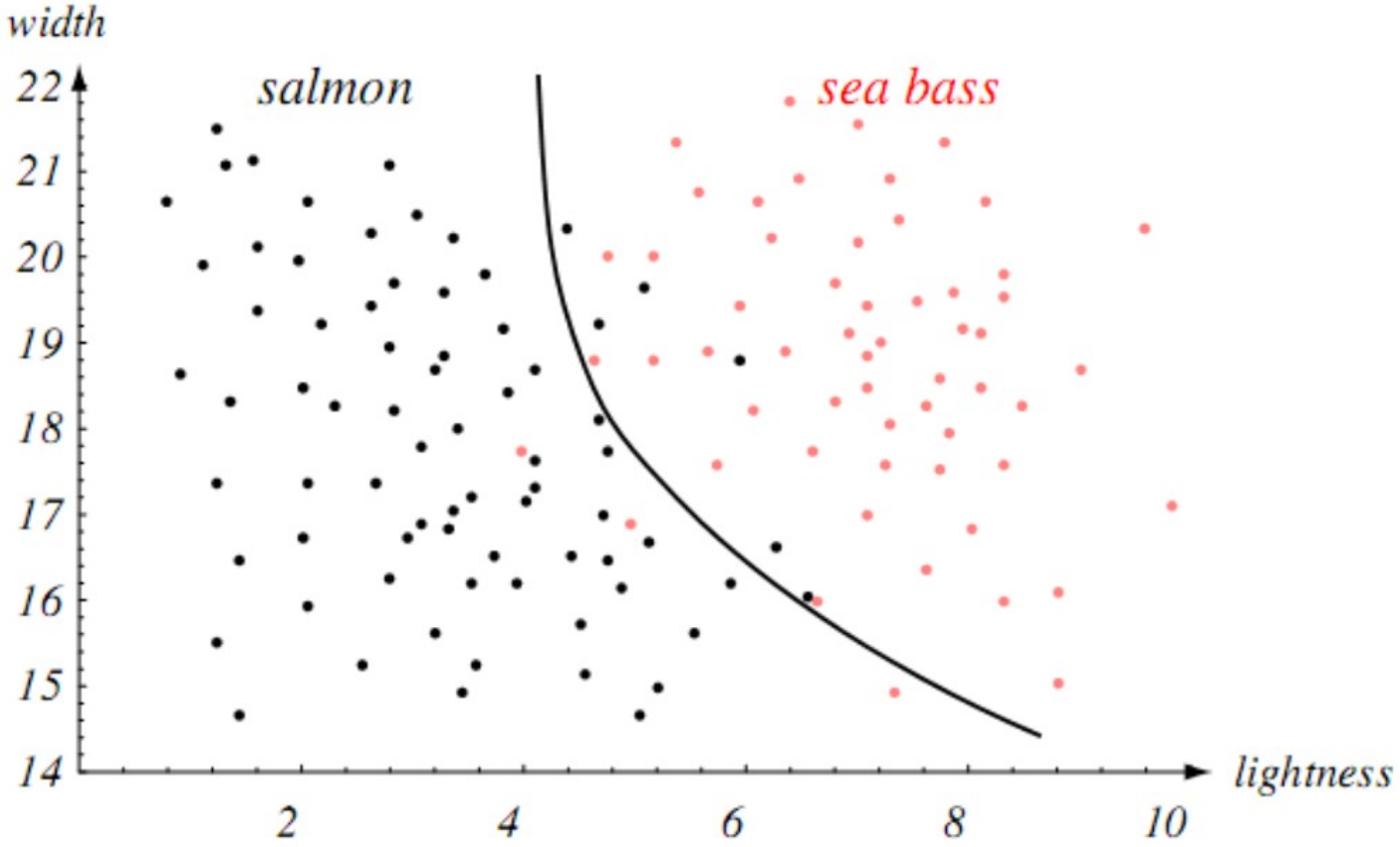
f(x) = a*x + b



PERFECT CLASSIFIER … FOR TRAINING SET

FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

A compromise



ALWAYS split your data into meaningful partitions

- **Training set:** data used for infering the parameters of your model(s)
  - To set weights and biases of an artificial neural network (ANN)
  - To find the mean vectors, covariance matrices and mixing weights of a Gaussian mixture model (GMM)
- **Validation set**: a ”pseudo” test data (disjoint from the training set) to optimize the architecture or to set control parameters
  - Number of layers, regularization terms, number of Gaussians…

- **Test set:** data to evaluate the model on an independent test set – Ideally ”one shot” – no optimizations or modeling decisions based on this data any more !

Multi-class: counting errors

- **GIVEN:**
  - M classes (categories/states of nature), $\omega_1,\ldots, \omega_M$
  - Assume N_j samples from class ω_j
  - Labeled set X = $\{(\mathbf{x}_i, y_i), i=1,2,\ldots,N\}$ on which the classifier "black-box" is evaluated on ($\mathbf{x}$=datapoints, y=labels), $N = N_1 + N_2 + \ldots + N_M$
  - Classifier g($\mathbf{x}$) that predicts the correct label of any datapoint $\mathbf{x}$
- **TASK:** estimate the probability of error P_{err} of classifier g
  Error rate: ratio of misclassified data points to the total number of data points:

$$E = \frac{1}{N}\sum_{i=1}^{N} e(\boldsymbol{x}_i)$$

$e(\mathbf{x}_i) = I\{ g(\mathbf{x}_i) \neq y_i \}$, where I{.}=1 for a true assertion, otherwise I{.}=0

Problem: error rate is not balanced

$$E = \frac{1}{N}\sum_{i=1}^{N} e(\boldsymbol{x}_i) = \frac{1}{N}\sum_{j=1}^{M} \underbrace{\sum_{\boldsymbol{x}_i \in \omega_j} e(\boldsymbol{x}_i)}_{\text{TOTAL ERROR COUNT FOR CLASS } \omega_j}$$

Classes with more test samples contribute more to the sum, biasing the evaluation (or system optimization) towards ”larger” classes. For this reason, we ocassionally call the above metric **weighted** error rate (weighted accuracy)

Balanced (unweighted) alternative: average of class-specific error rates

$$E = \frac{1}{M}\sum_{j=1}^{M} \underbrace{\frac{1}{N_j}\sum_{\boldsymbol{x}_i \in \omega_j} e(\boldsymbol{x}_i)}_{\text{ERROR RATE FOR CLASS } \omega_j}$$

Looking beyond error rates: confusion matrix

		Predicted label							
		TUR	SPA	ALB	ARA	KUR	RUS	EST	ENG
True label	TUR	**70**	3	1	10	5	5	2	4
	SPA	1	**51**	3	8	2	2	3	2
	ALB	1	3	**62**	3	1	11	5	2
	ARA	12	9	7	**128**	10	9	8	8
	KUR	9	3	3	6	**60**	5	3	4
	RUS	43	30	51	20	16	**379**	25	26
	EST	6	8	8	12	6	13	**120**	13
	ENG	7	10	3	6	3	7	6	**63**

TUR: Turkish
SPA: Spanish
ALB: Albanian
ARA: Arabic
KUR: Kurdish
RUS: Russian
EST: Estonian
ENG: English

Quizz/homework

- How many Estonian test samples we have ?
- What is the error rate for Russian dialect?
- How many Turkish samples are mistakenly classified as Arabic ?
- What are the weighted and unweighted error rates ?

Two-class (detection) tasks

- Example: Speaker verification (do two speech utterances originate from the same speaker or not?)
- Example: Medical diagnosis (does the person has cancer or not?)
- … name your favorite task here that demands a ”yes” or ”no” answer.
- Assume a **detector** that takes a test sample in and produces a real-valued output **score**
- Higher score values indicate high confidence towards either one of the hypotheses (positive or negative)
- Examples of a detectors:

$$s(\mathbf{x}) = log\frac{p(\mathbf{x}|H_0)}{p(\mathbf{x}|H_1)}$$

Log-likelihood ratio between two statistical models

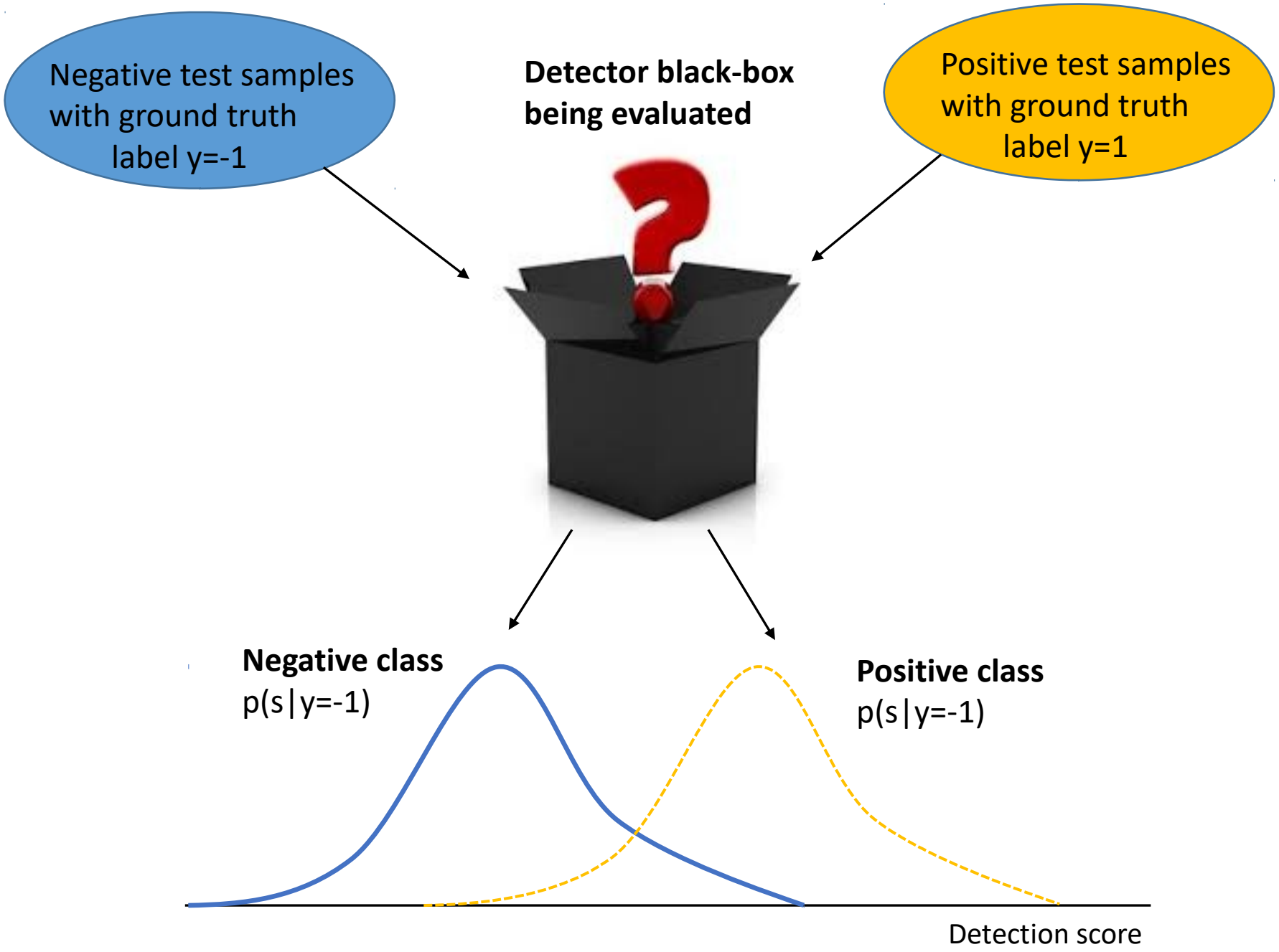
$$s(\mathbf{x}) = P(H_0|\mathbf{x})$$

Posterior probability of null hypothesis

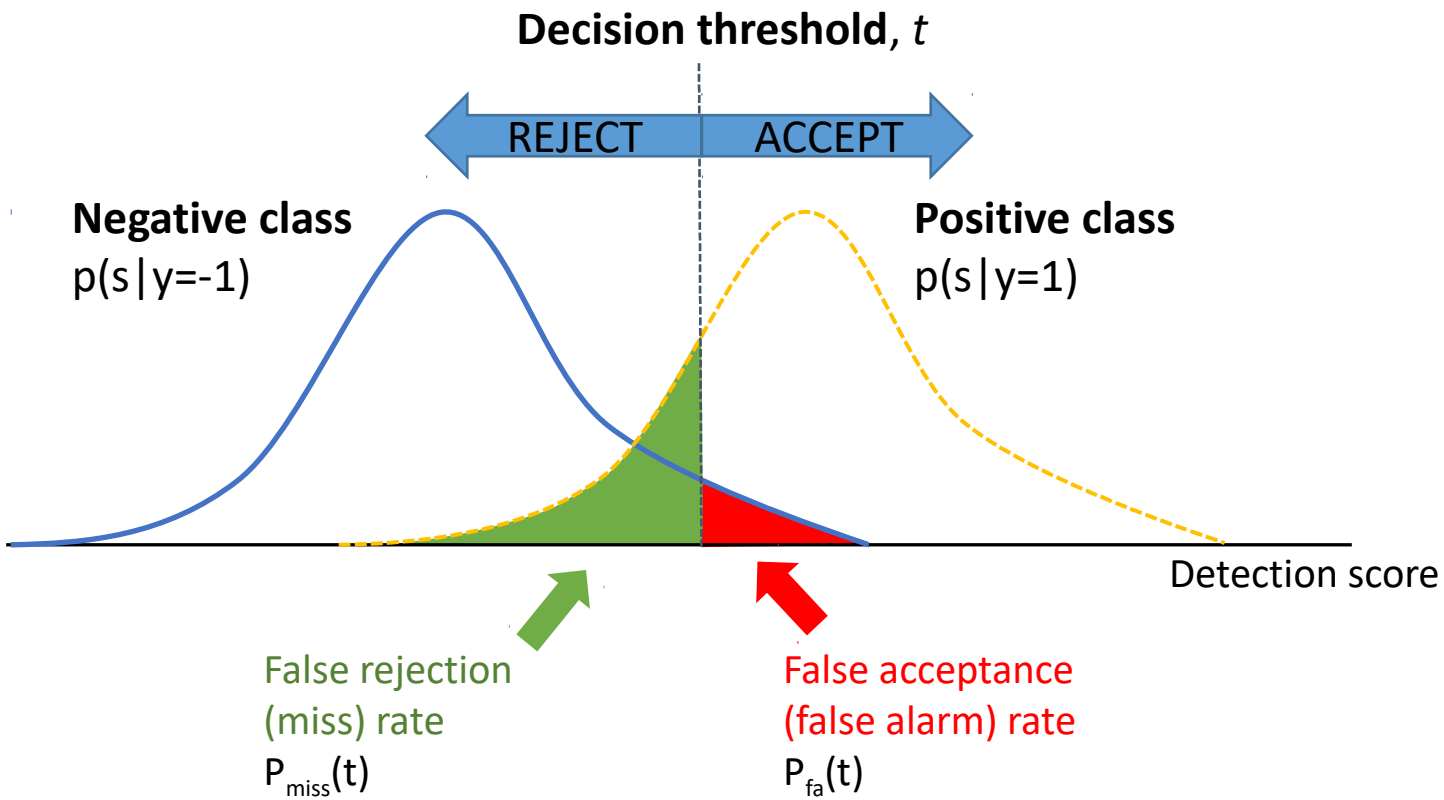
$$s(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{x} + w_0$$

Linear classifier
(e.g. support vector machine)

From the perspective of evaluation, we really do not care what the detector does internally.
The task of the evaluator is to assess the accuracy of the detector based on distribution of s(**x**) only.


Negative test samples with ground truth label y=-1
Detector black-box being evaluated
Positive test samples with ground truth label y=1
Negative class
p(s|y=-1)
Positive class
p(s|y=-1)
Detection score

Decision threshold and the two error rates



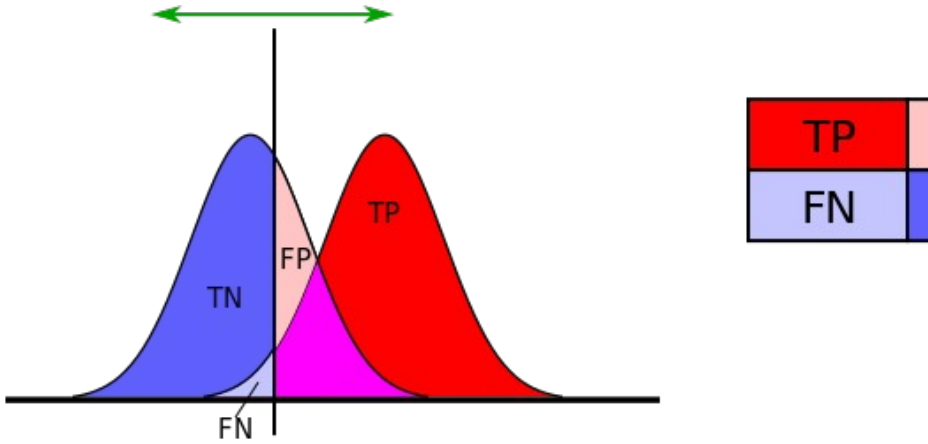
$$P_{miss}(t) = \int_{-\infty}^{t} p(s \mid y = 1)\ \mathrm{ds}$$

Miss probability is an increasing function of the threshold t

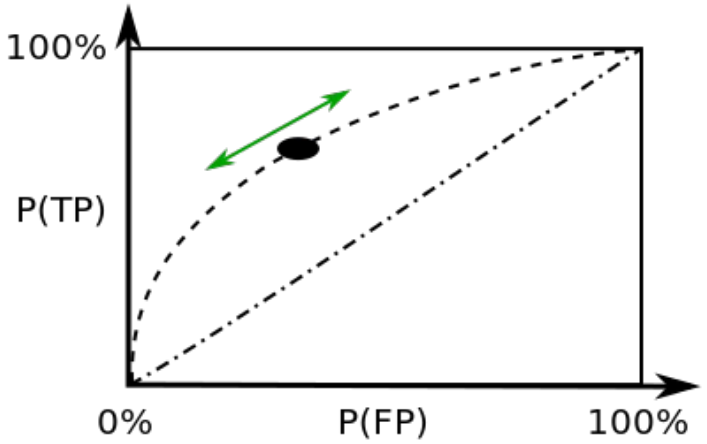
$$P_{fa}(t) = \int_{t}^{\infty} p(s \mid y = -1)\ \mathrm{ds}$$

False alarm probability is a decreasing function of the threshold t

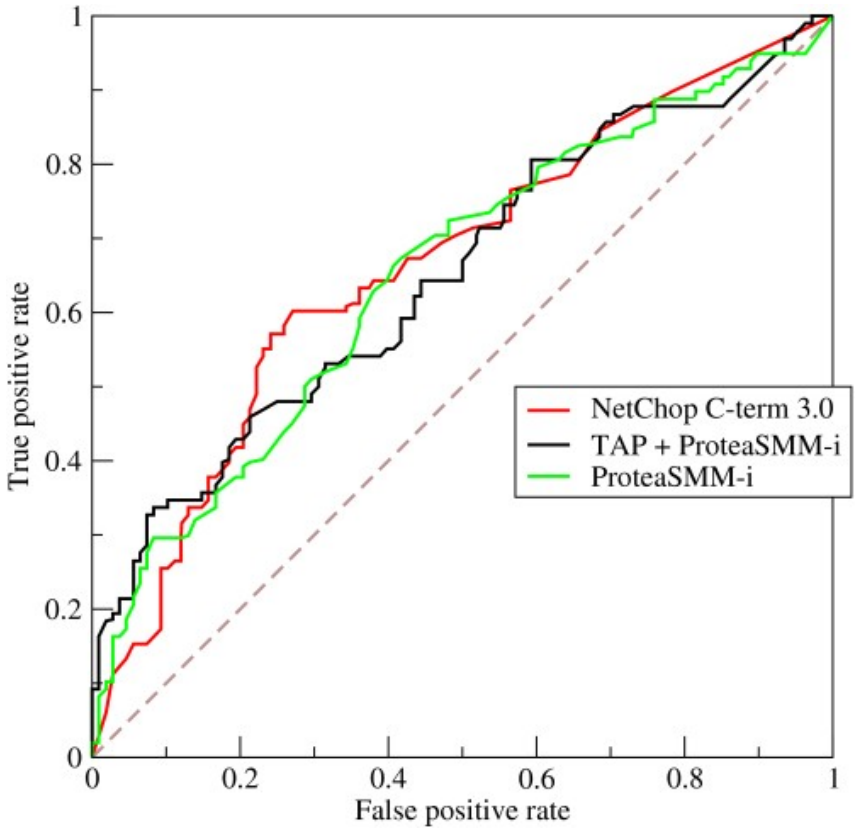
True positive, false positive, false negative, true negative, ROC curve [Wikipedia]



TP	FP
FN	TN



Receiver operating characteristics (ROC) curve displays the trade-off betweel false positive and true positive rates when the detection threshold is varied



- The extreme cases:

  $t = -\infty$ ➔ miss rate = 0%, false alarm rate = 100%

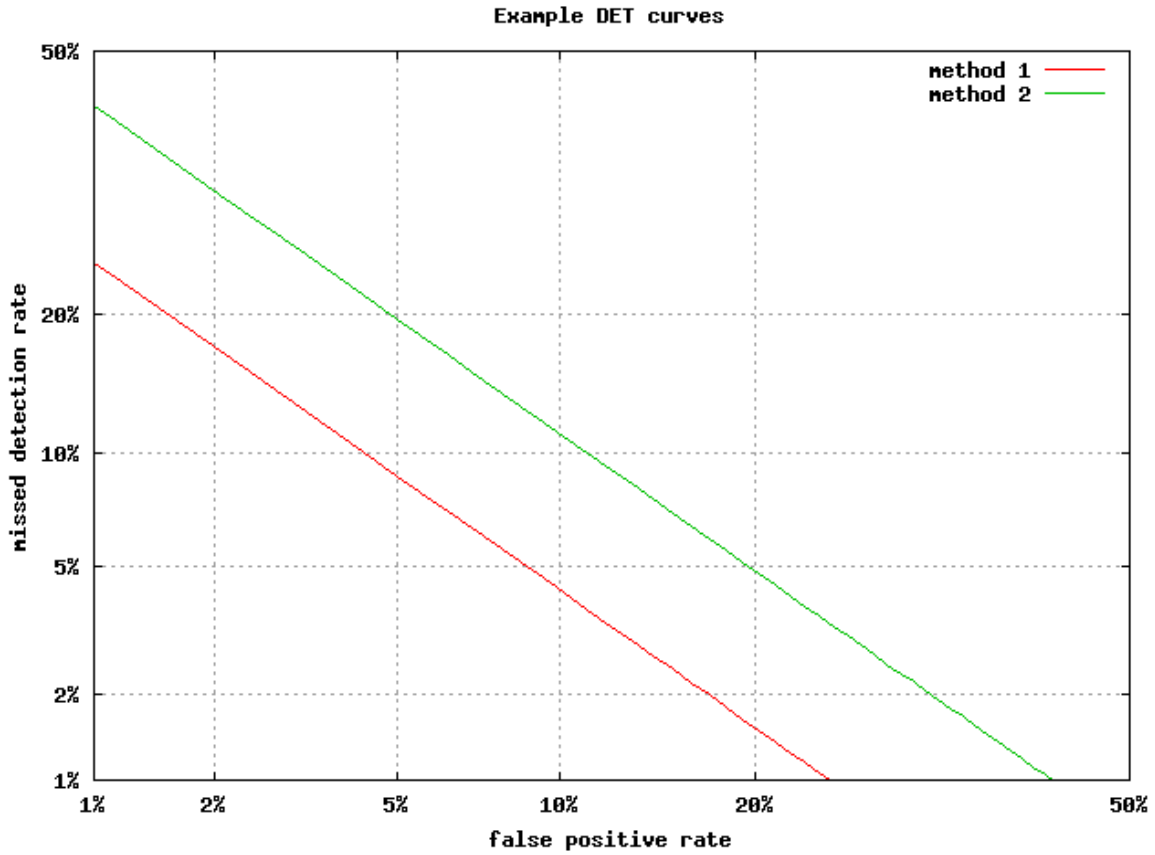
  $t = +\infty$ ➔ miss rate = 100%, false alarm rate = 0%

- Since the false alarm rate increases with t while the miss rate decreases, there exists a unique threshold, t_{EER} at which the two error rates must equal each other: $P_{fa}(t_{EER})=P_{miss}(t_{EER})$.
- The error rate at this specific decision threshold is known as the **equal error rate** (EER)
- EER is a widely used overall summary measure of a detector's error rate
- Other popular metric is area under the ROC curve (AUC or AUROC). For detailed discussion why AUC should not be used, see e.g.
  - D.J. Hand and C. Anagnostopoulos. When is the area under the receiver operating characteristic curve an appropriate measure of classifier performance? *Pattern Recognition Letters*, 34:492–496, 2013.
  - D.J. Hand. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, 29:1502–1510, 2010.

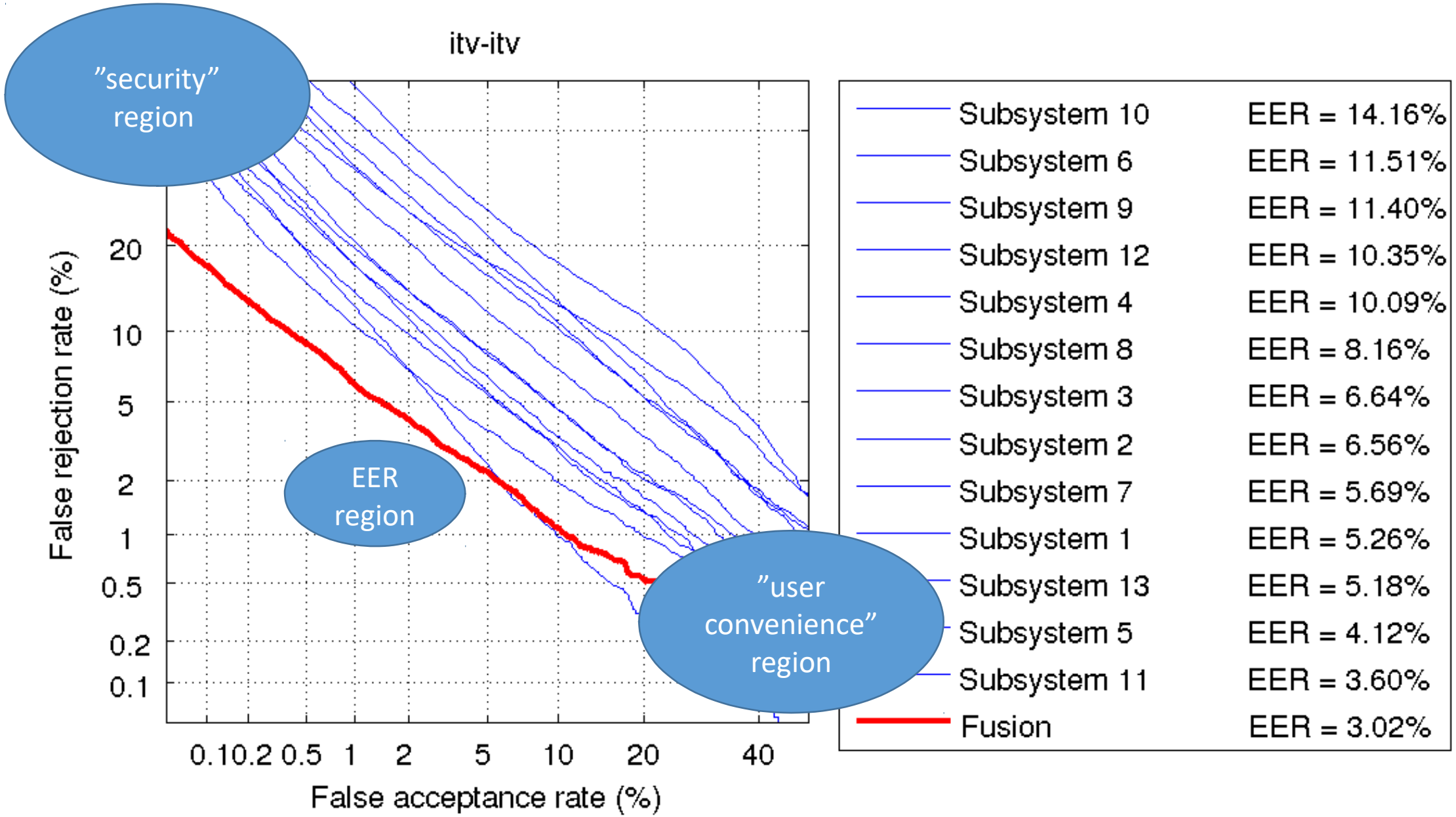
Homework*

- Assume the negative and positive score distributions are Gaussians with known parameters, say means μ_0 and μ_1 and standard deviations σ_0 and σ_1. Find out the expressions of equal error rate threshold t_{EER} and the corresponding EER value itself.

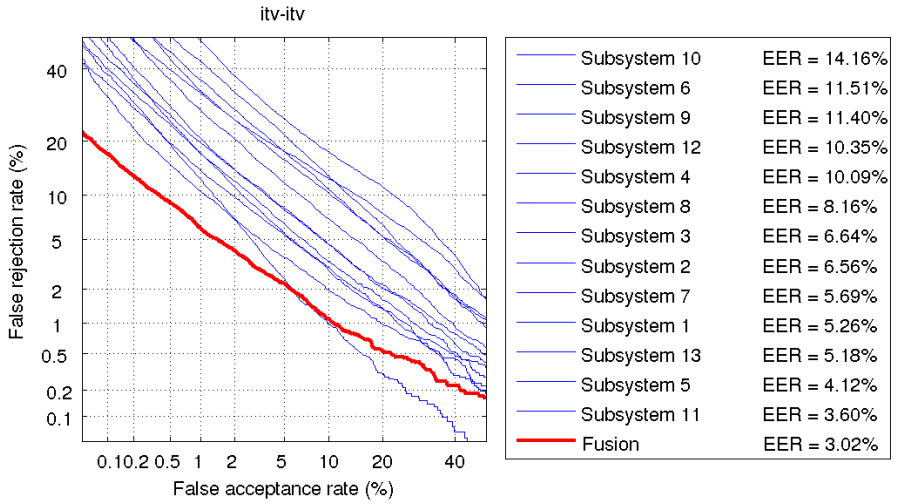
Detection error trade-off (DET) curve - an alternative to ROC

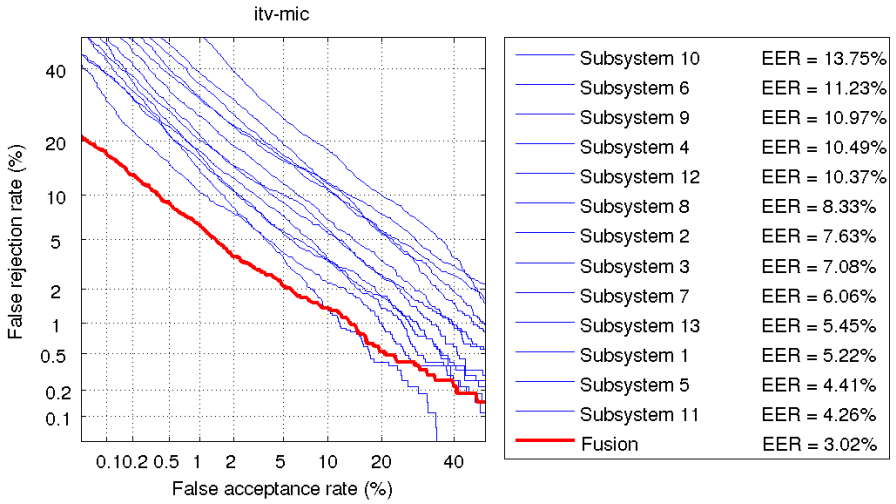


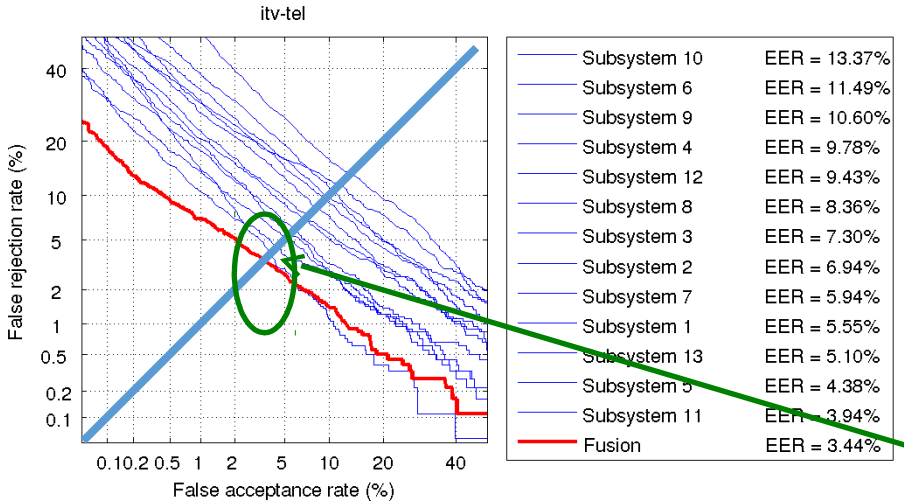
A. Martin, A., G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. "The DET Curve in Assessment of Detection Task Performance", Proc. Eurospeech '97, Rhodes, Greece, September 1997, Vol. 4, pp. 1895-1898.


itv-itv
”security” region
EER region
”user convenience” region
False rejection rate (%)
20
10
5
2
1
0.5
0.2
0.1
0.1 0.2 0.5 1 2 5 10 20 40
False acceptance rate (%)
Subsystem 10 EER = 14.16%
Subsystem 6 EER = 11.51%
Subsystem 9 EER = 11.40%
Subsystem 12 EER = 10.35%
Subsystem 4 EER = 10.09%
Subsystem 8 EER = 8.16%
Subsystem 3 EER = 6.64%
Subsystem 2 EER = 6.56%
Subsystem 7 EER = 5.69%
Subsystem 1 EER = 5.26%
Subsystem 13 EER = 5.18%
Subsystem 5 EER = 4.12%
Subsystem 11 EER = 3.60%
Fusion EER = 3.02%

Performance measures: DET plots, DCF, EER







EER point: Pfa = Pmiss

Weighting the two errors: decision cost function (DCF)

$$\mathrm{DCF(t)} = C_{\mathrm{Miss}} P_{\mathrm{target}} P_{\mathrm{miss}}(t) + C_{\mathrm{fa}} (1 - P_{\mathrm{target}}) P_{\mathrm{fa}}(t)$$

Cost of missing the target class

Prior probability of the target class

Miss rate at threshold t

Cost of doing a false alarm

Prior of the nontarget class

False alarm rate at threshold t

EXAMPLE: typical NIST speaker recognition evaluation

$$\mathrm{DCF} = C_{\mathrm{FR}} P_{\mathrm{FR}} P_{\mathrm{target}} + C_{\mathrm{FA}} P_{\mathrm{FA}} (1 - P_{\mathrm{target}})$$

$C_{\mathrm{FR}} = 1$ Cost of false rejection

$C_{\mathrm{FA}} = 1$ Cost of false acceptance

$P_{\mathrm{target}} = 0.001$ Prior probability of the target speaker

”Effective prior”

$$P_{\text{eff}} = \text{logit}^{-1}(\text{logit}(P_{\text{tar}}) + \log(C_{\text{miss}}/C_{\text{fa}}))$$

$$\text{logit}\, P = \log\left(P/(1-P)\right)$$

Actual and minimum decision costs

1. Find the best possible decision threshold on a **development set**
2. Evaluate the DCF at that threshold on an **evaluation data set;** this is the **actual detection cost**:

$$t_{dev} = \arg\min_{-\infty<t<\infty} DCF(t)$$

$$actDCF = DCF(t_{dev})$$

”Oracle” value is provided by the **minimum DCF** directly on the evaluation data

$$minDCF = \min_{-\infty<t<\infty} DCF(t)$$

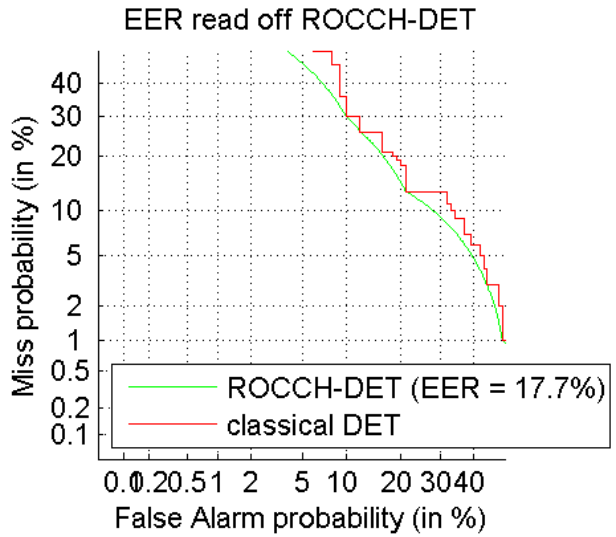
Other cost functions to be aware of…

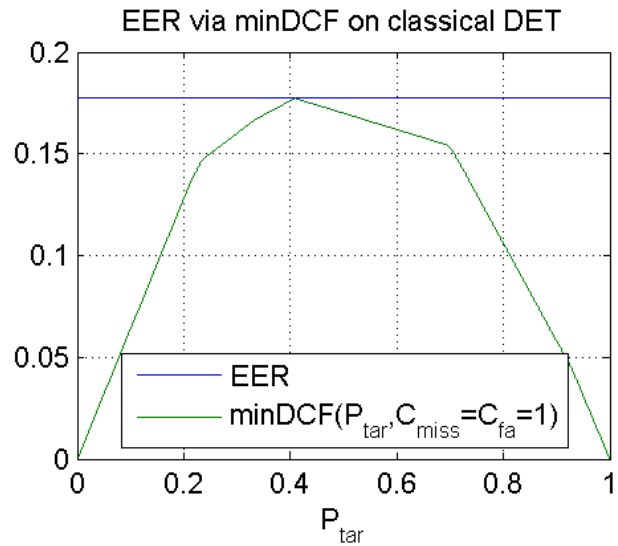
- **Half total error rate** (HTER) – again, find a threshold on a development set. Then just report the average of both types of errors on the evaluation set:

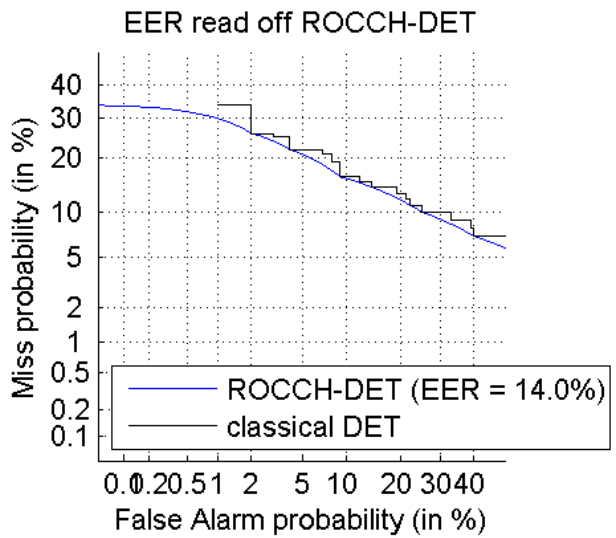
$$HTER = \frac{P_{\text{fa}}(t_{\text{dev}}) + P_{miss}(t_{\text{dev}})}{2}$$

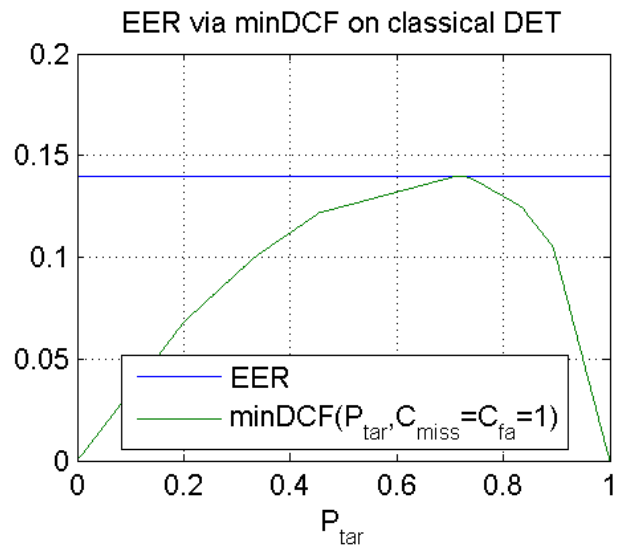
A practical note on DET and ROC curves for small data

- go for convex hull method









Do not re-invent the wheel… use these:

- Bosaris toolkit: https://sites.google.com/site/bosaristoolkit/
- NIST DETware: https://www.nist.gov/file/65996
- SIDEKIT: https://pypi.python.org/pypi/SIDEKIT