EPSRC Vision Summer School

Performance Characterization

Adrian F. Clark
⟨alien@essex.ac.uk⟩
VASE Laboratory, Comp Sci & Elec Eng
University of Essex

Overview of the session

1. What we’re trying to do
2. Assessing an individual algorithm
3. Comparing algorithms
4. Automating things
5. Concluding remarks

What are we trying to do?

- Why are you doing a PhD?
- To extend the body of human knowledge
- Improving the robustness of vision techniques is the main challenge for you, the upcoming generation of vision researchers

What are we trying to do?

- Why are you doing a PhD?
- To extend the body of human knowledge
- Improving the robustness of vision techniques is the main challenge for you, the upcoming generation of vision researchers

What are we trying to do?

- Why are you doing a PhD?
- To extend the body of human knowledge
- Improving the robustness of vision techniques is the main challenge for you, the upcoming generation of vision researchers

A statistical preamble

Imagine taking a fair coin and tossing it a number of times. Let us say that we obtain:

1. 3 heads from 10 tosses;
2. 30 heads from 100 tosses;
3. 300 heads from 1,000 tosses.

Let us now consider which of these is the most surprising.

A coin toss obeys a *binomial* distribution

$$P(t) = \binom{N}{t} p^t (1-p)^t \qquad (1)$$

with mean is Np and variance $Np(1-p)$. As the coin is fair, $p = \frac{1}{2}$ and the three cases work out as:

1. The expected number of heads (*i.e.,* the mean) is $Np = 5$. The sd is $\sqrt{Np(1-p)} = \sqrt{10 \times \frac{1}{2} \times \frac{1}{2}} = 1.58$. Hence, 3 heads is $(5-3)/1.58 \approx 1.3$ sds from the mean.
2. 30 heads is $(50-30)/5 = 4$ sds from the mean.
3. 300 heads is $(500-300)/15.8 \approx 13$ sds from the mean.

“Which algorithm is best?”

- This is a natural question to ask, and is essentially what the competitions we often see associated with conferences ask
- But that isn’t what such competitions find: they measure which algorithm **performs** best — and there is no attempt to understand **why** algorithms’ performances differ

“Which algorithm is best?”

- This is a natural question to ask, and is essentially what the competitions we often see associated with conferences ask
- But that isn’t what such competitions find: they measure which algorithm **performs** best — and there is no attempt to understand **why** algorithms’ performances differ

What questions should we be asking?

- Some researchers contend that we should be asking **why does one algorithm out-perform another?**
- To answer that, we need to explore what characteristics of the input affect an algorithm's performance and by how much — this is what is meant by performance **characterization**
- It is important that we ask (and answer) this question if we are to develop vision algorithms that work robustly outside the research lab

What questions should we be asking?

- Some researchers contend that we should be asking **why does one algorithm out-perform another?**
- To answer that, we need to explore what characteristics of the input affect an algorithm's performance and by how much — this is what is meant by performance **characterization**
- It is important that we ask (and answer) this question if we are to develop vision algorithms that work robustly outside the research lab

What questions should we be asking?

- Some researchers contend that we should be asking **why does one algorithm out-perform another?**
- To answer that, we need to explore what characteristics of the input affect an algorithm's performance and by how much — this is what is meant by performance **characterization**
- It is important that we ask (and answer) this question if we are to develop vision algorithms that work robustly outside the research lab

Assessing performance

- There are few occasions where an algorithm's performance can be predicted analytically, so most performance assessment work is performed empirically
- The procedure is almost always to run the program under test against a large set of input data for which the correct outputs ("ground truth") are known
- The numbers of correct and incorrect results are then counted

The result from a single test

Each individual test yields one of four possible results:

True positive: when a test that should yield a correct result does so

True negative: when a test that should yield an incorrect result does so

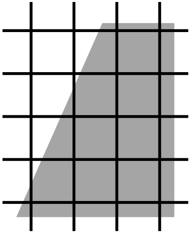
False negative: when a test that should yield a correct result actually yields an incorrect one

False positive: when a test that should yield an incorrect result actually yields a correct one

But things aren't always clear-cut

(a) Original image

(b) Corresponding template


(c) Ambiguity at edge of template

Why can't we just compare TP *etc.* rates?

- Most algorithms have tuning parameters
- Results obtained on one dataset don't guarantee similar figures from another ($\Rightarrow$ the test set was too small or not representative)
- Results obtained on simulated data don't necessarily match those from real data ($\Rightarrow$ the simulated data are poor)

Technology evaluation

- This involves identifying and testing any underlying assumptions of an algorithm — *e.g.*, Canny’s edge detector assumes that noise is additive
- The tuning parameters of the algorithm are also listed (*e.g.*, Gaussian kernel width, thresholds) and the consequence of varying them assessed
- It is best done using **simulated** data, as it provides the only way that all characteristics of the data can be known

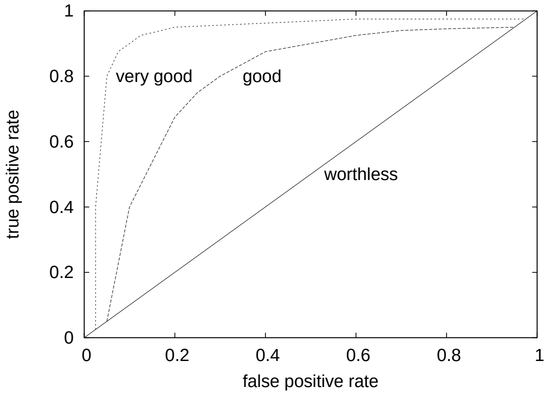
Scenario or application evaluation

- This assesses the effectiveness of the technique for a particular task, such as locating line-segments in fMRI datasets
- This must, of course, be performed using real data
- If technology evaluation is performed well, the researcher will have some idea of how well the algorithm is likely to perform in an application simply by estimating the characteristics of the fMRI data — how much and what type of noise, and so on

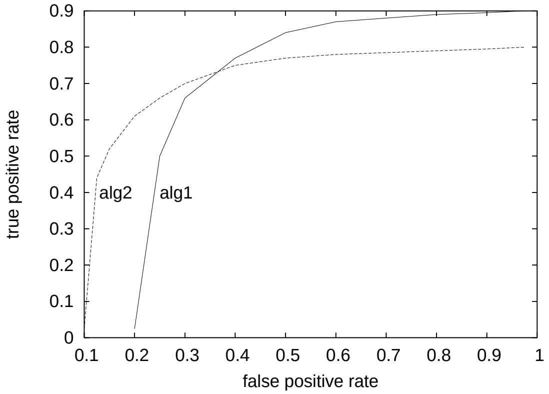
Assessing a single algorithm

- As most algorithms can be tuned to the test data, it is common to plot sensitivity (TP rate) against specificity (1 − FP rate) or similar
- In such a curve, the area under the curve gives a measure of overall accuracy and can be used as the basis of a comparison — though there are better ways

Receiver operating characteristic (ROC) curves



Which algorithm is better?



Confusion matrices

- For classification-type problems, it is common to tabulate the class returned by an algorithm against the correct value
- When there are two classes, success and failure, this reduces to

	predicted negative	predicted positive
actual negative	TN	FP
actual positive	FN	TP

actual	predicted									
	0	1	2	3	4	5	6	7	8	9
0	20	0	0	6	0	0	1	0	10	0
1	0	25	0	0	0	0	0	6	0	0
2	0	0	31	0	0	0	0	0	0	0
3	0	0	0	21	0	0	0	0	10	0
4	0	0	0	0	31	0	0	0	0	0
5	0	0	0	0	0	22	0	0	9	0
6	1	0	0	1	0	2	23	0	3	1
7	0	8	0	0	0	0	0	23	0	0
8	4	0	1	3	2	1	3	0	13	4
9	0	0	0	2	0	0	0	3	1	27

Comparing algorithms

- The only viable way to compare algorithms is to run them on the same data with a specific set of tuning parameter values
- When this is done, an appropriate statistical test can be employed which takes into account not only the number of false positives *etc.* but also the number of tests

McNemar's Test

	Alg A failed	Alg A succeeded
Alg B failed	N_{ff}	N_{sf}
Alg B succeeded	N_{fs}	N_{ss}

McNemar's test is:

$$\chi^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{(N_{sf} + N_{fs})}$$

This is a form of chi-square test for matched paired data.

Why McNemar’s test is so useful

- When the two algorithms being tested give similar performances, $\chi^2 \approx 0$
- If we assessing whether two algorithms differ, a two-tailed test should be used; and if we are determining whether one algorithm is better than another, a one-tailed test should be used
- If both N_{sf} and N_{fs} are large, then Algorithm A tends to succeed where Algorithm B fails and *vice versa* — so an algorithm that combines both is likely to be better than either in isolation

Towards principled vision systems design

- McNemar’s test tells us not only **how to do it** but also **when it is appropriate to do it** on the basis of technology evaluation — in other words, technology evaluation needs to be an inherent part of the algorithm design process
- It is ironic that McNemar’s test works principally by considering where algorithms fail — yet almost all the testing we see in computer vision concentrates on where algorithms succeed

Towards principled vision systems design

- McNemar's test tells us not only **how to do it** but also **when it is appropriate to do it** on the basis of technology evaluation — in other words, technology evaluation needs to be an inherent part of the algorithm design process
- It is ironic that McNemar's test works principally by considering where algorithms fail — yet almost all the testing we see in computer vision concentrates on where algorithms succeed

Determining robustness to noise

- Take a typical input image and generate (say) 100 versions with added Gaussian-distributed noise of zero mean and known standard deviation, then feed to the algorithm under test
- If the algorithm produces a class label, that label should not change
- If the algorithm produces a measurement, plotting a histogram of the error in that measurement will show how the algorithm is affected by noise. If the input noise distribution is Gaussian, the distribution of the measurement error should also be Gaussian and the sd of the errors gives an idea of how well the algorithm withstands noise

Vision systems ‘in the wild’

- Most successful vision systems work in controlled environments, though thankfully there is an increasing trend towards working in uncontrolled ones
- There is a host of problems you won’t have thought of, ranging from a lack of handy main power to illumination changes due to the time of day and clouds
- Don’t expect lab-measured performance to be achieved in the field!

Concluding remarks

- Most of the vision community *still* does not take assessment seriously
- By performing assessment and comparison intelligently **as part of the algorithm development process** you can improve algorithms as you develop them
- This gives us a handle on being able to **design** vision systems