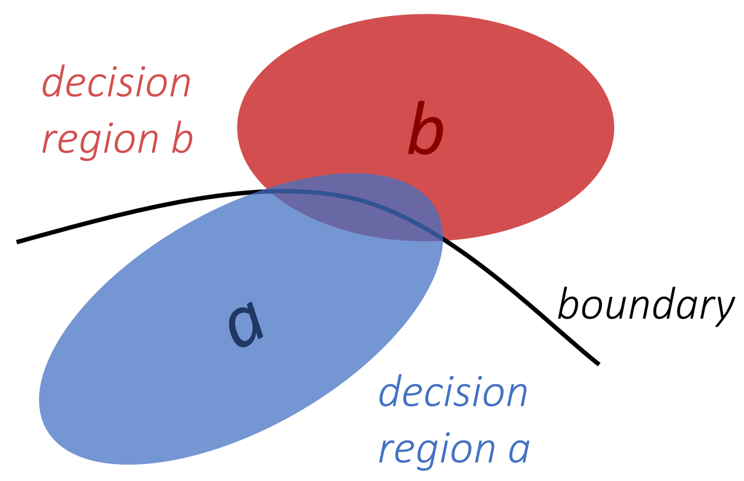
A new method to calculate classification accuracy

Abhranil Das, R Calen Walshe, Wilson Geisler · Center for Perceptual Systems · The University of Texas at Austin

THE UNIVERSITY OF TEXAS AT AUSTIN

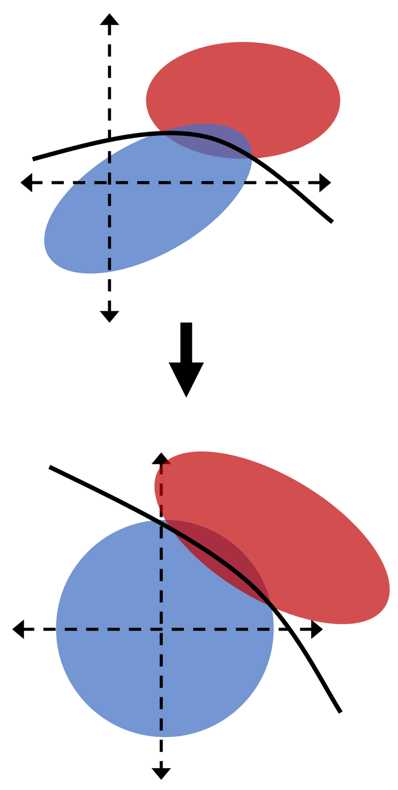
Summary

- A decision boundary can optimally classify observations into one of two Gaussians.
- Accuracy is the fraction of the Gaussian masses in the correct decision regions.
- These mass integrals have no closed form for general Gaussians. Standard numerical methods would be cumbersome, inaccurate and inefficient.
- We present a solution that is fast, accurate and robust across cases.
- Our MATLAB implementation is a complete suite of tools for such classification.

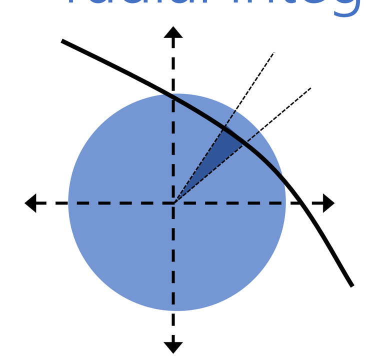


Method

1. Whiten Gaussian



2. Use closed-form radial integral



$$dm(\theta) = \int_0^{r_b(\theta)} \frac{e^{-\frac{r^2}{2}}}{2\pi}\, dr\, \Delta\theta = \frac{\Delta\theta}{2\pi}\left(1 - e^{-\frac{r_b^2(\theta)}{2}}\right)$$

In polar form, the standard normal has a closed-form integral over a small angle within the boundary.

3. Sum radial integral over fixed angle grid

$$m = \sum_\theta dm(\theta)$$

The closed-form integral can be summed over a finite grid of angles that is the same in all cases.

Example applications

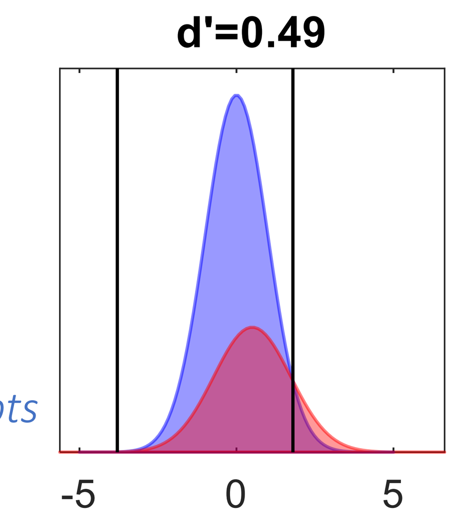
download our MATLAB toolbox from **github.com/abhranildas/classify**

Discriminate 1D Gaussians, unequal priors

Classification accuracy and d' between Gaussians with specified parameters

```
>> mu_a=0; v_a=1;
>> mu_b=0.5; v_b=1.5;
>> results=classify([mu_a,v_a],[mu_b,v_b],'p_a',.7)

 struct with fields:

           acc_gauss: 0.7181  ← accuracy
         acc_gauss_a: 0.9634
         acc_gauss_b: 0.1458
             d_gauss: 0.4923  ← discriminability d'
        d_gauss_aprx: 0.4472  ← approx. d'
    bd_pts_gauss_opt: [-3.7929 1.7929]  ← boundary pts
 bd_coeffs_gauss_opt:  struct with fields:

                              a2: -0.3333
                              a1: -0.6667
                              a0: 2.2667
```

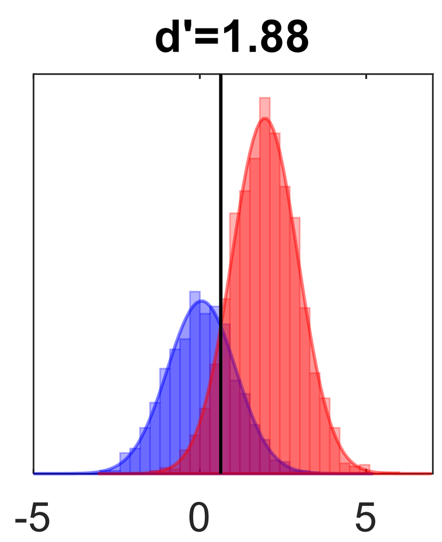
coefficients of the boundary quadratic $x'a_2x + a_1'x + a_0 = 0$ →



Discriminate observations

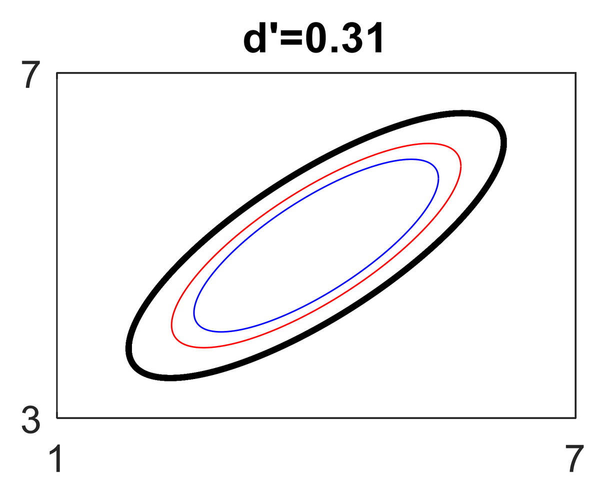
Directly input observations instead of parameters

```
>> mu_a=0; v_a=1;
>> obs_a=normrnd(mu_a,sqrt(v_a),[1000 1]);

>> mu_b=2; v_b=1;
>> obs_b=normrnd(mu_b,sqrt(v_b),[2000 1]);

>> results=classify(obs_a,obs_b,'type','obs');
```

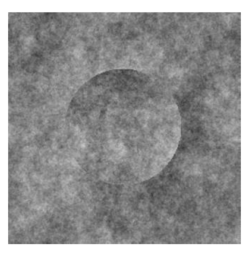


Discriminate 2D Gaussians

```
>> mu_a=[4; 5];
>> v_a=[2 1.1; 1.1 1];

>> mu_b=[4; 5];
>> v_b=1.4*v_a;

>> results=classify([mu_a,v_a],[mu_b,v_b]);
```

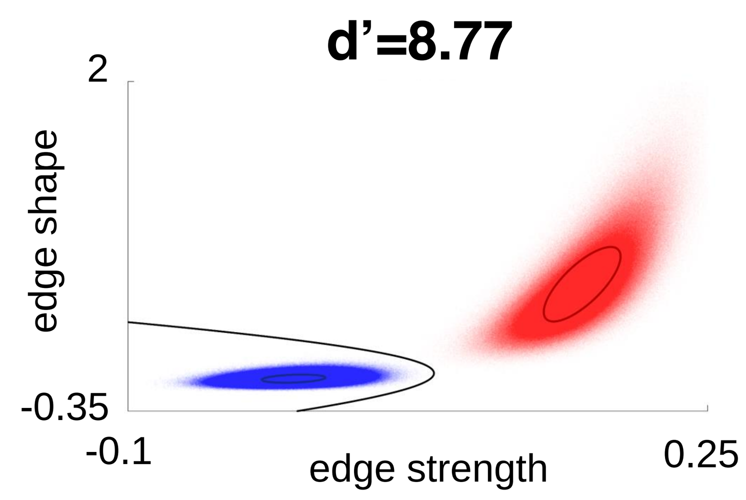


Real detection task data



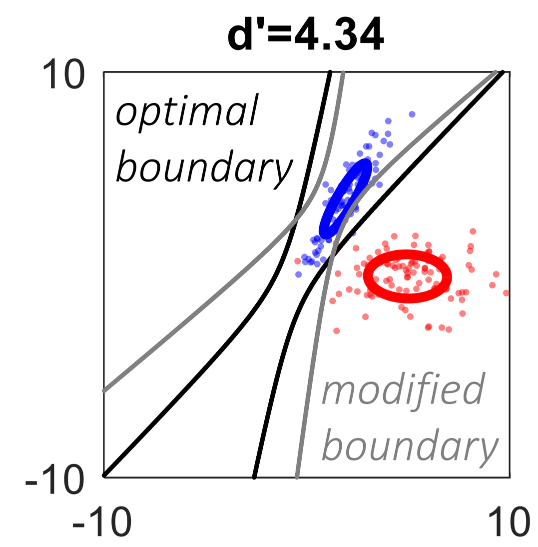
Decision variables from an actual camouflage detection task.

```
>> load('camouflage_data.mat', ...
   'data_blank','data_target')

>> results=classify(data_blank,...
   data_target,'type','obs');
```



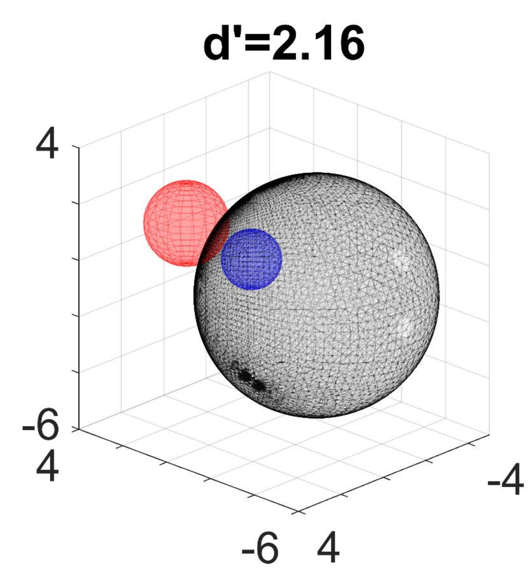
Modify decision boundary

```
>> mu_a=[2 4]; v_a=[1 1.5; 1.5 3];
>> obs_a = mvnrnd(mu_a,v_a,100);

>> mu_b=[5 0]; v_b=[3 0; 0 1];
>> obs_b = mvnrnd(mu_b,v_b,100);

>> results=classify(obs_a,obs_b,'type','obs');

>> % modify boundary
>> custom_bd_coeffs=results.bd_coeffs_obs_opt;
>> custom_bd_coeffs.a2=custom_bd_coeffs.a2+.2;
>> custom_bd_coeffs.a1=custom_bd_coeffs.a1-5;
>> custom_bd_coeffs.a0=custom_bd_coeffs.a0+10;

>> results_mod=classify(obs_a,obs_b,'type','obs',...
   'custom_bd_coeffs',custom_bd_coeffs);
```

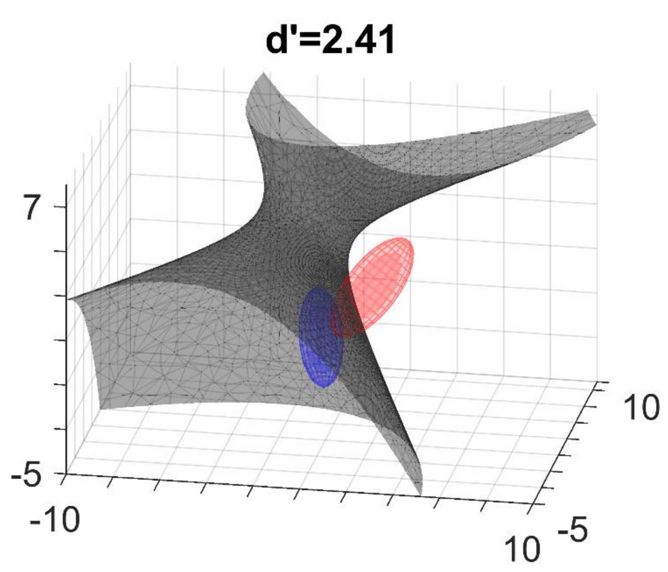


Discriminate 3D Gaussians

```
>> mu_a=[0;0;0];
>> v_a=eye(3);

>> mu_b=[2;1;1];
>> v_b=2*eye(3);

>> results=classify([mu_a,v_a],[mu_b,v_b]);
```




```
>> mu_a=[0;0;0];
>> v_a=[1 .5 0;
       .5 2 1;
        0 1 4];

>> mu_b=[2;1;2];
>> v_b=[2    1 -1.5;
        1    3 -2;
       -1.5 -2  4];

>> results=classify([mu_a,v_a],[mu_b,v_b]);
```



Estimate d' between far-apart distributions

Get the approximate d' even when the overlap is too small to measure (e.g. for ideal observers).

```
>> mu_a=[0;0;0]; v_a=eye(3);
>> mu_b=[100;0;0] v_b=1.5*eye(3);

>> results=classify([mu_a,v_a],[mu_b,v_b])
```