Introduction to Neural Networks
U. Minn. Psy 5038

Linear discriminant

Initialize

■ **Read in Statistical Add-in packages:**

```
Off[General::"spell1"];
<< "Statistics`DescriptiveStatistics`"
<< "Statistics`DataManipulation`"
<< "Statistics`NormalDistribution`"
<< "MultivariateStatistics`"; << "ComputationalGeometry`"
<< "MultivariateStatistics`"
```

Review Discriminant functions

Let's review earlier material on discriminant functions. Perceptron learning is an example of nonparametric statistical learning, because it doesn't require knowledge of the underlying probability distributions generating the data (such distributions are characterized by a relatively small number of "parameters", such as the mean and variance of a Gaussian distribution). Of course, how well it does will depend on the generative structure of the data. Much of the material below is covered in Duda and Hart (1978).

Linear discriminant functions: Two category case

A discriminant function, g(x) divides input space into two category regions depending on whether g(x)>0 or g(x)<0. (We've switched notation, x=**f**). The linear case corresponds to the simple perceptron unit we studied earlier:

$$g(x) = w.x + w_0 \quad (1)$$

where w is the weight vector and $w_0$is the threshold (sometimes called bias, although this "bias" has nothing to do with statistical "bias").

Discriminant functions can be generalized, for example to quadratic decision surfaces:

$$g(x) = w_0 + \sum_{i=1} w_i x_i + \sum_{i=1} \sum_{j=1} w_{ij} x_i x_j \quad (2)$$

We've seen how g(x)=0 defines a decision surface which in the linear case is a hyperplane. Suppose x_1 and $x_2$are points sitting on the hyperplane, then their difference is a vector lying in the hyperplane

$$\begin{aligned} w.x_1 + w_0 &= w.x_2 + w_0 \\ w.(x_1 - x_2) &= 0 \end{aligned} \qquad (3)$$

so the weight vector w is normal to any vector lying in the hyperplane. Thus w determines how the plane is oriented. The normal vector w points into the region for which g(x)>0, and -w points into the region for which g(x)<0.

Let x be a point on the hyperplane. If we project x onto the normalized weight vector x.w/|w|, we have the normal distance of the hyperplane from the origin equal to:

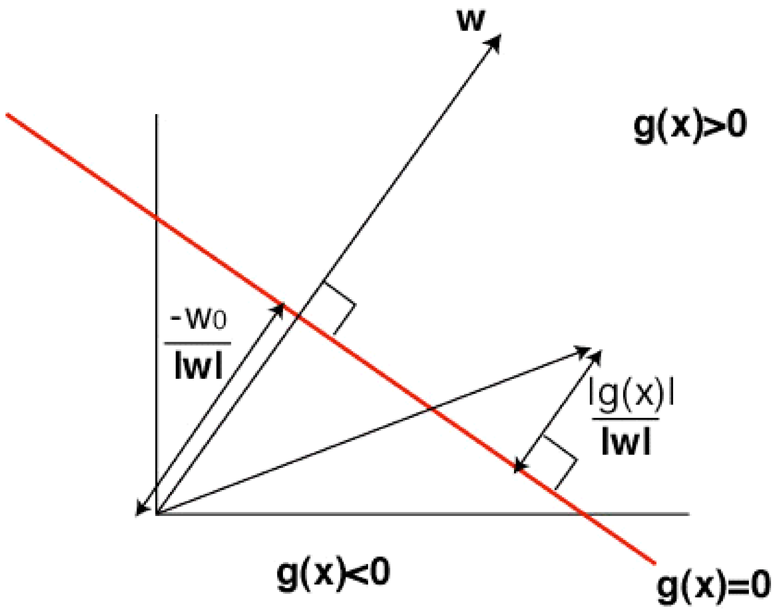
$$w.x / |w| = -w_0 / |w| \qquad (4)$$

Thus, the threshold determines the position of the hyperplane.

One can also show that the normal distance of x to the hyperplane is given by:

$$g(x) / |w| \qquad (5)$$

So we've seen that: 1) disriminant function divides the input space by a hyperplane decision surface; 2) The orientation of the surface is determined by the weight vector w; 3) the location is determined by the threshold w_0; 4) the discriminant function gives a measure of how far in input vector is from the hyperplane.

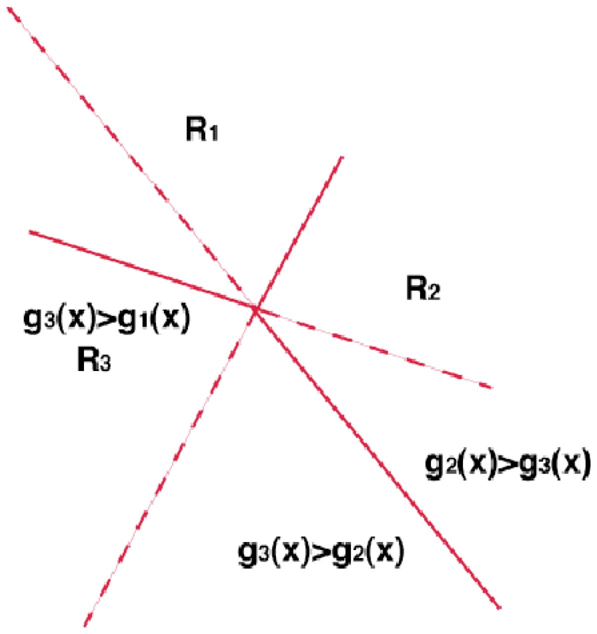


Multiple classes

Suppose there are c classes. There are a number of ways to define multiple class discriminant rules. One way that avoids undefined regions is:

$$g_i(x) = w_i.x + w_{i0}, \quad i = 1, \ldots, c \qquad (6)$$

Assign x to the ith class if : $g_i(x) > g_j(x)$ for all $j \neq i$. (7)



It can be shown that this classifier partitions the input space into simply connected convex regions. This means that if you connect any two feature vectors belonging to the same class by a line, all points on the line are in the same class. Thus this linear classifier won't be able to handle problems for which there are disconnected clusters of features that all belong to the same class. Also, from a probabilistic perspective, if the underlying generative probability model for a given class has multiple modes, this linear classifier won't do a good job either.

Task-dependent Dimensionality reduction

Fisher's linear "discriminant"

The idea is that the original input space may be impractically huge, but if we can find a subspace (hyperplane) that preserves the distinctions between categories as well as possible, we can make our decisions in smaller space. We will derive the Fisher linear "discriminant".

This is closely related to the psychology idea of finding "distinctive" features. E.g. consider bird identification. If I want to discriminate cardinals from other birds in my backyard, I can make use of the fact that (male) cardinals may be the only birds that are red. So even tho' the image of a bird can have lots of dimensions, if I project the image on to the "red" axis, I can do fairly well with just one number. How about male vs. female human faces?

■ Generative model: two nearby gaussian classes

Define two bivariate base distributions

```
(ar = {{1, 0.99}, {0.99, 1}};
ndista = MultinormalDistribution[{0, -1}, ar];)
(br = {{1, .9}, {.9, 2}};
ndistb = MultinormalDistribution[{0, 1}, br];)
```

Find the expression for the probability distribution function of ndista

```
pdf = PDF[ndista, {x1, x2}]
```

$$1.12822\, e^{\frac{1}{2}\left(-x1\,(50.2513\,x1-49.7487\,(x2+1))-(x2+1)\,(50.2513\,(x2+1)-49.7487\,x1)\right)}$$

Use Mean[ ] and CovarianceMatrix[ndista] to verify the population mean and the covariance matrix of ndistb

```
Mean[ndistb]
```

```
{0, 1}
```

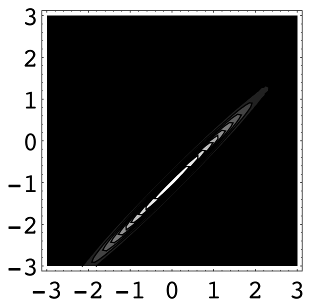
```
Covariance[ndista]
```

$$\begin{pmatrix} 1 & 0.99 \\ 0.99 & 1 \end{pmatrix}$$

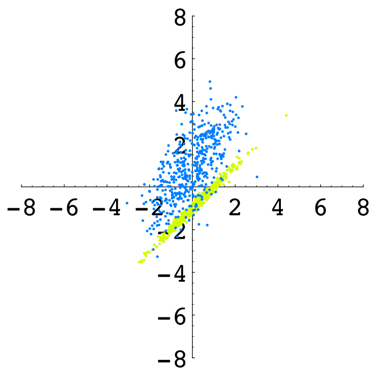
Try different covariant matrices. Should they be symmetric? Constraints on the determinant of ar, br?

Make a contour plot of the PDF ndista

```
pdfa = PDF[ndista, {x1, x2}];
ContourPlot[pdfa, {x1, -3, 3}, {x2, -3, 3}, PlotPoints → 64,
 PlotRange → All]
```



```
nsamples = 500;
a = Table[Random[ndista], {nsamples}];
ga = ListPlot[a, PlotRange → {{-8, 8}, {-8, 8}}, AspectRatio → 1,
   PlotStyle → Hue[0.2`], DisplayFunction → Identity];
b = Table[Random[ndistb], {nsamples}];
gb = ListPlot[b, PlotRange → {{-8, 8}, {-8, 8}}, AspectRatio → 1,
   PlotStyle → Hue[0.6`], DisplayFunction → Identity];
Show[ga, gb, DisplayFunction → $DisplayFunction]
```



Use Mean[ ] to find the *sample* mean of b. Whats is the sample *covariance* of b?

```
Mean[b]
```

```
{-0.0471119, 0.925014}
```

```
Covariance[b]
```

$$\begin{pmatrix} 0.995891 & 0.933731 \\ 0.933731 & 2.04603 \end{pmatrix}$$

■ Try out different projections of the data by varying the slope (m) of the discriminant line

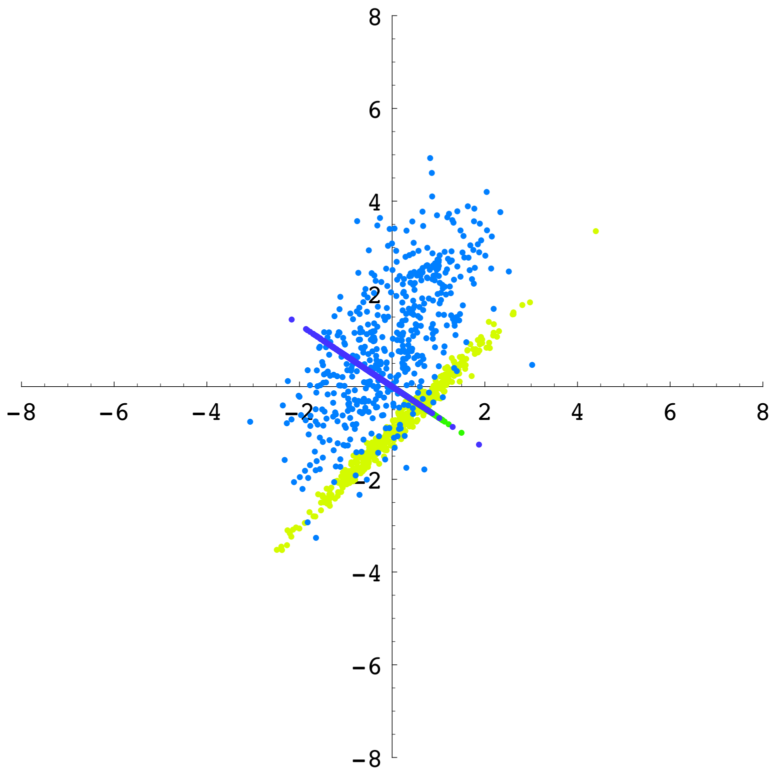
```
m = -2 / 3;
wnvec = {1, m} / Sqrt[1 + m^2];
```

```
{{x, y}}.{n1, n2}
Map[#1 * {n1, n2} &, {{x, y}}.{n1, n2}]
```

$\{\text{n1}\, x + \text{n2}\, y\}$

$(\,\text{n1}\,(\text{n1}\, x + \text{n2}\, y) \quad \text{n2}\,(\text{n1}\, x + \text{n2}\, y)\,)$

```
aproj = (#1 wnvec &) /@ (a.wnvec);
gaproj = ListPlot[aproj, AspectRatio → 1, PlotStyle → Hue[0.3`],
   DisplayFunction → Identity];
bproj = (#1 wnvec &) /@ (b.wnvec);
gbproj = ListPlot[bproj, AspectRatio → 1, PlotStyle → Hue[0.7`],
   DisplayFunction → Identity];
Show[ga, gb, gaproj, gbproj, DisplayFunction → $DisplayFunction,
 AspectRatio → Automatic]
```



By trial and error, find a value of m that separates the classes well along the projection line

Calculate the "signal-to-noise" ratio along the projection line: difference between the means divided by the square root of the product of the standard deviations along the line

```
Mean[aproj]
```

```
{0.488142, −0.325428}
```

■ Theory for simple 2-class case

(see Duda and Hart for general case)

A measure of the separation between the projections is the difference between the means:

$$| w.(m_a - m_b) |$$

and

$$m_a = \frac{1}{N}\sum_{i=1}^{N} x, \text{ summed over the N x's from class a} \tag{8}$$

$$m_b = \frac{1}{M}\sum x, \text{ summed over the M x's from class b}$$

where w (**wnvec**) is the unknown unit vector along the discriminant line .

In our case above, the vector difference between the means is:

```
Mean[a] - Mean[b]
```

```
{−0.0923584, −2.07925}
```

and the difference between the means projected onto a discriminant line is:

```
wnvec.(Mean[a] - Mean[b])
```

```
0.847261
```

To improve separation, we can't just scale w, because the noise scales too.

We'd like the difference between the means to be large relative to the variation for each class. We can define a measure of the scatter for the projected samples in say class a (a==1), by:

$$\sum_{y\in\text{class a}} \left(y - \tilde{m}_a\right)^2 \quad (9)$$

where $\tilde{m}_a$is the sample mean of the points from class a projected onto discriminant line and y=**w**.***x***

Or in terms of the Mathematic example:

```
Apply[Plus, aproj - wnvec.Mean[a]];
```

The total scatter S is defined by the sum of the scatters for both classes (a and b).

$$S = \sum_{y\in\text{class a}} \left(y - \tilde{m}_a\right)^2 + \sum_{y\in\text{class b}} \left(y - \tilde{m}_b\right)^2$$

If we divide the above number by the total number of points, we have an estimate of the variance of the combined data along the projected axis.

We now have the basic ingredients behind intuition for the Fisher linear discriminant. We'd like to find that **w** for which J:

$$J\,(w) = \frac{\left| \tilde{m}_a - \tilde{m}_b \right|^2}{S} = \frac{\left| \mathbf{w}.(m_a - m_b) \right|^2}{S}$$

is biggest. We want to maximize the difference between the projected class means, while minimizing the dispersion of the data on the projected line.

One can show that S = $\mathbf{w}^T.\mathbf{S_W}.\mathbf{w}$, where

$S_W$is measure of within-class variation called the *within-class scatter matrix:*

$$S_W = \sum_{i=1}^{2} \sum_{x\in\text{Class } i} (x - m_i)\,(x - m_i)^T \quad (10)$$

For the numerator, a measure of between class variation is the *between-class scatter matrix*:

$$S_B = (m_1 - m_2).(m_1 - m_2)^T \quad (11)$$

and the difference between the projected means can be show to be:

$$\left| \tilde{m}_a - \tilde{m}_b \right|^2 = w^T.S_B.w$$

Find **w** (corresponding to slope) to maximize the criterion function

$$J\,(w) = \frac{w^T.S_B.w}{w^T.S_W.w} \quad (12)$$

Answer:

$$w = S_W^{-1}.(m_a - m_b) \quad (13)$$

■ Demo: Finding Fisher's linear discriminant

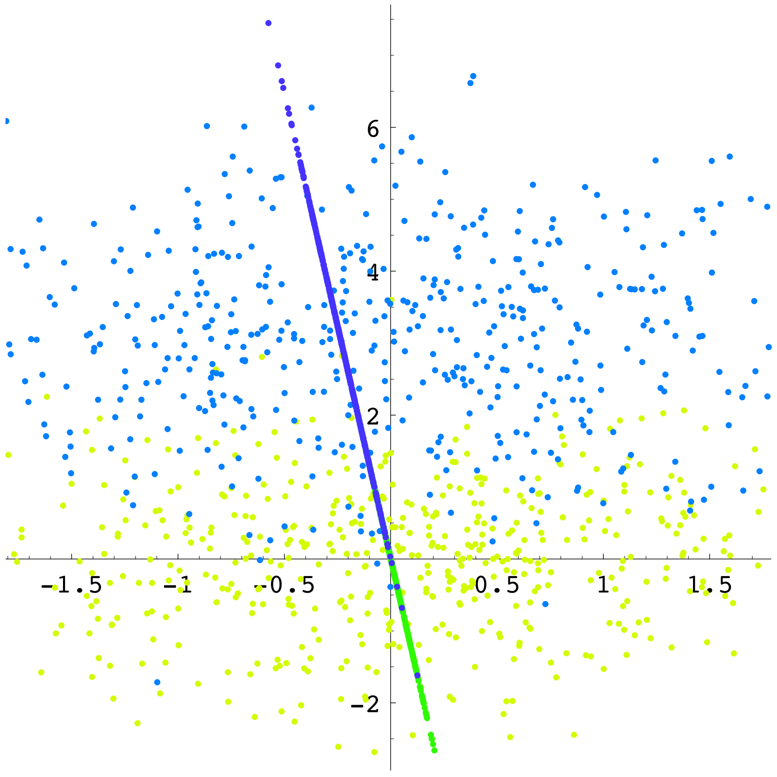
```
normalize[x_] := x / Sqrt[x.x];
```

```
ma = Mean[a];
mb = Mean[b];
```

```
Sa = Sum[Outer[Times, a[[i]] - ma, a[[i]] - ma], {i, 1, nsamples}];
Sb = Sum[Outer[Times, b[[i]] - mb, b[[i]] - mb], {i, 1, nsamples}];
Sw = Sa + Sb;
wldf = normalize[Inverse[Sw].(ma - mb)]
```

```
{0.0769866, -0.997032}
```

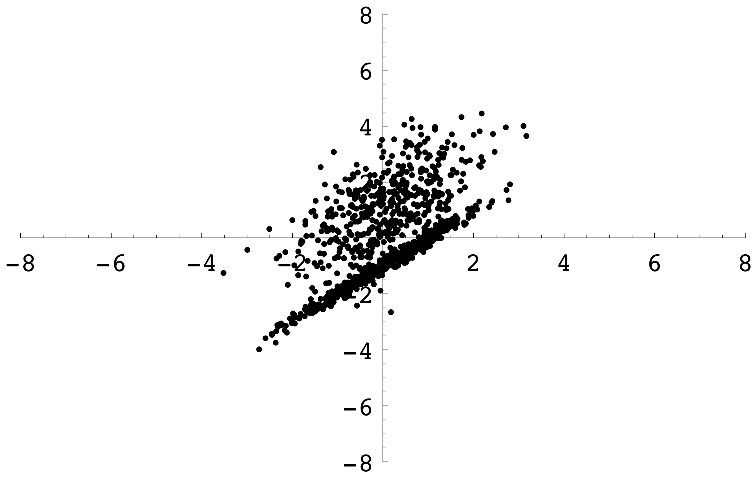
```
aproj = (#1 wldf &) /@ (a.wldf);
gaproj = ListPlot[aproj, AspectRatio → 1, PlotStyle → Hue[0.3`],
   DisplayFunction → Identity];
bproj = (#1 wldf &) /@ (b.wldf);
gbproj = ListPlot[bproj, AspectRatio → 1, PlotStyle → Hue[0.7`],
   DisplayFunction → Identity];
Show[ga, gb, gaproj, gbproj, DisplayFunction → $DisplayFunction]
```



We started off with a 2-dimensional input problem and turned it into a 1-D problem. For the n-dimensional case, see Duda and Hart.

■ Compare with the principal component axes

```
c = Join[a, b];
ListPlot[c, PlotRange → {{-8, 8}, {-8, 8}}]
```



```
g1 = ListPlot[c, PlotRange → {{-8, 8}, {-8, 8}}, AspectRatio → 1,
   DisplayFunction → Identity];
```

```
auto = Covariance[c]
eigvalues = Eigenvalues[auto]
eigauto = Eigenvectors[auto]
```

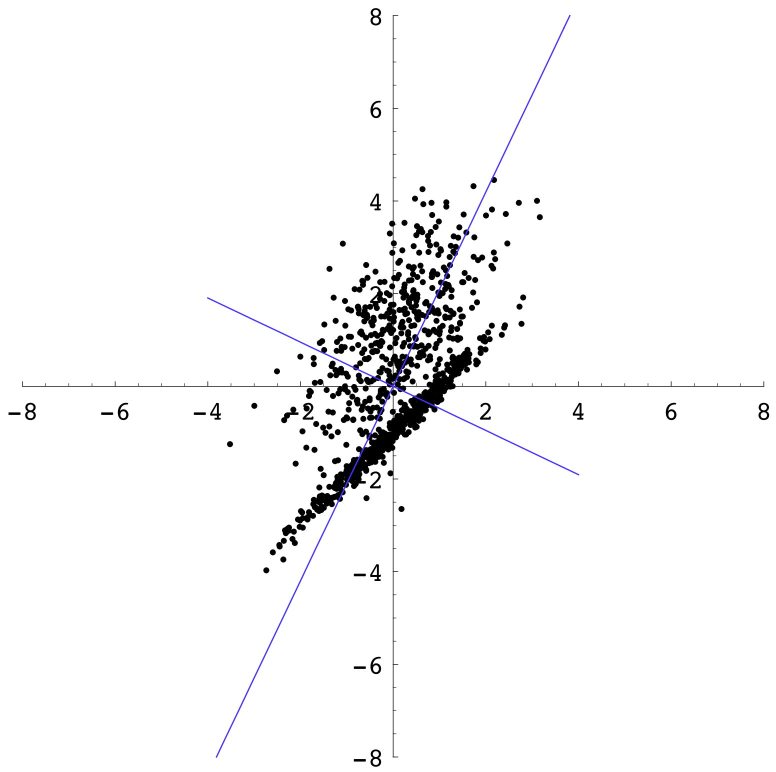
$$\begin{pmatrix} 0.970786 & 0.928494 \\ 0.928494 & 2.47561 \end{pmatrix}$$

```
{2.91828, 0.528115}
```

$$\begin{pmatrix} 0.430355 & 0.90266 \\ 0.90266 & -0.430355 \end{pmatrix}$$

```
gPCA = Plot[{eigauto[[1,2]]/eigauto[[1,1]] x,
    eigauto[[2,2]]/eigauto[[2,1]] x},
        {x,-4,4}, AspectRatio->1,
        DisplayFunction->Identity,
        PlotStyle->{RGBColor[.2,0,1]}];
```

```
Show[g1, gPCA, DisplayFunction → $DisplayFunction]
```


8
6
4
2
-2
-4
-6
-8
-8
-6
-4
-2
2
4
6
8

How does the principal component (biggest variance) compare with the Fisher discriminant line?

References

Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis . New York.: John Wiley & Sons.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: Wiley.

(Amazon.com)

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. Neural Computation, 4(1), 1-58.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation, 4*(3), 415-447.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge University Press.

Vapnik, V. N. (1995). *The nature of statistical learning*. New York: Springer-Verlag.

http://neuron.eng.wayne.edu/software.html