Introduction to Neural Networks
U. Minn. Psy 5038

Gaussian generative models and inference

■ **Initialize standard library files:**

In[1]:=
```
Off[General::spell1];
```

In[2]:=
```
<< Statistics`MultinormalDistribution`
<< Statistics`DataManipulation`
```

In[4]:=
```
<< Graphics`Graphics`
```

Generative modeling: Drawing univariate samples

Consider the gaussian distribution. The density is proportional to:

In[221]:=
```
Exp[-(x - x0)^2/(2*σ^2)]
```

Out[221]= $e^{-\frac{(x-x0)^2}{2\sigma^2}}$

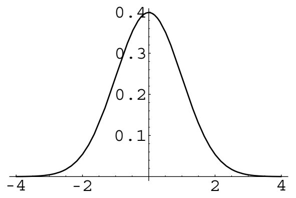
But to make sure that the total probabilty (area) is one, we can derive the normalizing constant by finding the area under the curve:

```
Integrate[Exp[-(x - x0)^2/(2*σ^2)], {x, -Infinity, Infinity}]
```

$\text{If}\left[\text{Re}[\sigma^2] > 0, \sqrt{2\pi}\sqrt{\sigma^2}, \int_{-\infty}^{\infty} e^{-\frac{(x-x0)^2}{2\sigma^2}}\, dx\right]$

And the standard normal distribution is given by letting x0=0 and σ=1:

```
Plot[Exp[-(x1^2)/2]/(Sqrt[2*Pi]), {x1, -4, 4}];
```



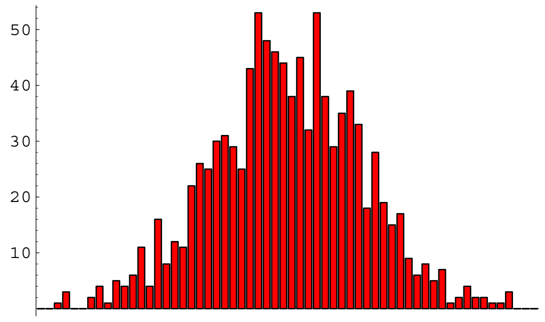
Plot[PDF[NormalDistribution[0,1],x1],{x1,-4,4}]; gives the same thing using the add-on normal distribution function.

If we have a formula like that above that specifies a probability distribution,how can we draw samples from it?

Just for Gaussian. Use Central Limit Theorem

If all we want to do is make a Gaussian random number generator from a uniformly distributed generator, one method doesn't even use the above formula. Instead we can use a theorem that we encountered earlier: the Central Limit Theorem. Just add a bunch of uniform random variables together and:

In[89]:=
```
nusamples = 10;
z1 = Table[Sum[Random[], {i, nusamples}] - nusamples/2, {1000}];
binsize = .1;
midpoints = Table[i + binsize/2, {i, -3, 3 - binsize, binsize}];
freq = BinCounts[z1, {-3, 3, binsize}];
BarChart[Transpose[{freq, midpoints}], BarLabels → None];
```



Looks bell-shaped. And as nusamples->∞, the distribution does approach the Gaussian.

Use Density Mapping theorem. More general.

But what if we don't want a Gaussian generator, but some other kind of random number? We can use the density mapping theorem which can be applied to arbitrary distributions, including gaussian.

We'll use the density mapping theorem to turn uniformly distributed random numbers **Random[]** into gaussian distributed random numbers with mean =0 and standard deviation =1.

$$p_Y(y)\,\delta y = p_X(x)\,\delta x$$

$$p_Y(y)\,\frac{\delta y}{\delta x} = p_X(x)$$

Suppose $p_Y(y)$ =1 (over the unit interval, but zero elsewhere). Then

$\frac{\delta y}{\delta x} = p_X(x)$, and integrating :

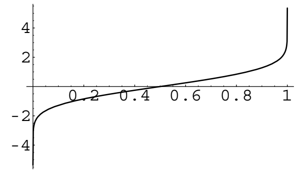
$$y(x) = \int_{-\infty}^{x} p_x(x')\,dx' = P(x) \qquad (1)$$

Thus if we sample from the uniform distribution to get y, x should be distributed according to $p_X(x)$. To do this, we need a mapping from y->x. This is given by the inverse cumulative distribution, i.e. $P^{-1}(y)$.

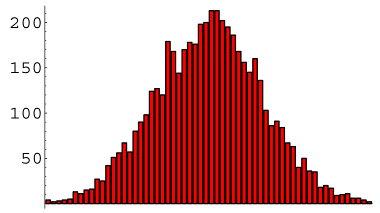
A quick way to implement this for the Gaussian case is to useMathematica's built-in function to get the inverse cumulative normal. InverseErf[] is the inverse of:

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$$

In[95]:=
```
z[p_] := -Sqrt[2] InverseErf[1 - 2 p];
Plot[z[y], {y, 0, 1}];
```



In[36]:=
```
binsize = .1;
midpoints = Table[i + binsize / 2, {i, -3, 3 - binsize, binsize}];
zl = Table[z[Random[]], {5000}];
freq = BinCounts[zl, {-3, 3, binsize}];
BarChart[Transpose[{freq, midpoints}], BarLabels → None];
```



Example of look-up-table method that is fast and works for almost any distribution.

Illustrate with the Gaussian case for a third time

■ Cumulative gaussian

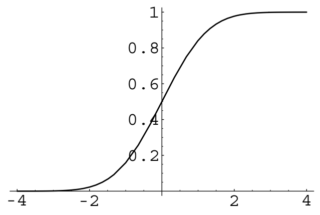
To illustrate the basic common principles, we'll start from scratch (rather than using the add-on function CDF[]).

In[104]:=
```
Clear[cumulgauss, x, x1];
cumulgauss[x_] :=
 Integrate[Exp[-(x1^2) / 2] / (Sqrt[2 * Pi]), {x1, -Infinity, x}]
```

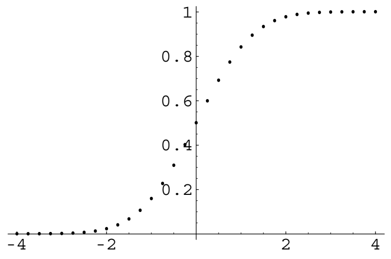
In[106]:=
```
cumulgauss[Infinity]
```

Out[106]=
```
1
```

In[107]:=
```
Plot[cumulgauss[x], {x, -4, 4}];
```



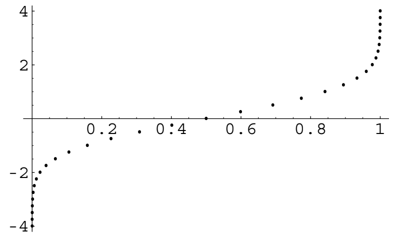
In[108]:=
```
lcumulgauss = Table[{x, cumulgauss[x]}, {x, -4.0, 4.0, .25}];
ListPlot[lcumulgauss];
```



■ Make inverse cumulative gaussian table

In[112]:=
```
invlcumulgauss = RotateLeft[lcumulgauss, {1, 1}];
```

In[113]:=
```
ListPlot[invlcumulgauss];
```

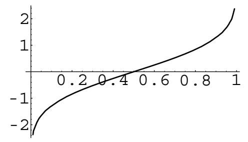


■ Make interpolated function of the inverse cumulative

If we pick Random[], this gives us a point on the x-axis, but it will almost certainly fall between the cracks. So we interpolation between the discrete points to get a continous function:

In[114]:=
```
interinvlcumulgauss = Interpolation[invlcumulgauss];
```

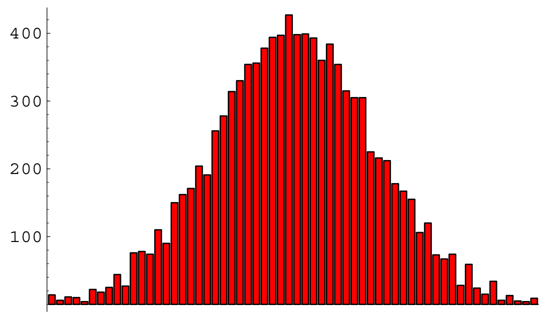
In[115]:=
```
Plot[interinvlcumulgauss[x], {x, .01, .99}];
```



■ Draw a bunch of samples, and plot up histogram

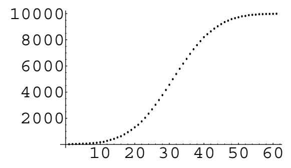
Now we are ready to draw 10,000 samples.

In[121]:=
```
binsize = .1;
midpoints = Table[i + binsize / 2, {i, -3, 3 - binsize, binsize}];
zl = Table[interinvlcumulgauss[Random[]], {10000}];
freq = BinCounts[zl, {-3, 3, binsize}];
BarChart[Transpose[{freq, midpoints}], BarLabels → None];
```



■ Plot up cumulative histogram

In[130]:=
```
CumFreq = FoldList[Plus, 0, freq];
ListPlot[CumFreq];
```



■ A non-Gaussian example: The von Mises distribution, with Matlab code

(courtesy, Paul Schrater)

```
function pofx = vonMisespdf(x,mu,sigma)
% For -pi <= x <= pi
% force x-mu within -pi to pi
y = angle(exp(i*(x-mu)));
kappa = 1/(sigma)^2;
%kappa = sigma;
pofx = exp(kappa*cos(y))/(2*pi*besseli(0,kappa));
function vonrand = vonMisesrand(nrand,mu,sigma)
% inverse cumulative method, executed by table lookup with
% linear interpolation
% build sampled cdf
x = (-pi:2*pi/(2e3):pi);
pofx = vonMisespdf(x,0,sigma);
cofx = cumsum(pofx/sum(pofx));
u = rand(1,nrand);
vonrand = interp1(cofx,x,u)+mu;
```

Generative modeling: Multivariate gaussian, mixtures

Gaussian multivariate distributions

■ Define multivariate gaussian probability density

An n-variate multivariate gaussian (multinormal) distribution with mean vector μ and covariance matrix Σ is denoted $N_n(\mu, \Sigma)$. The density is:

$$p(x) = \frac{1}{(2\pi)^{n/2}\,\mathrm{Det}[\Sigma]^{1/2}}\,\mathrm{Exp}\left[-\frac{1}{2}\,(x-\mu)^T\,\Sigma^{-1}\,(x-\mu)\right] \qquad (2)$$

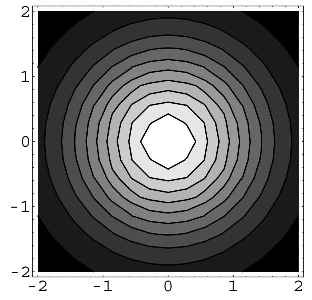
We define a 2-variate density:

```
multigaus[x_,m_,cov_]:=
    Module[{IC,detCov,norm,p},
    IC = Inverse[cov];
    detCov = Abs[Det[cov]];
    norm = N[Sqrt[(2Pi)^2 detCov]];
    p = Exp[-0.5 (x-m).IC.(x-m)]/norm;
    Return[p];
];
```

Two variable examples

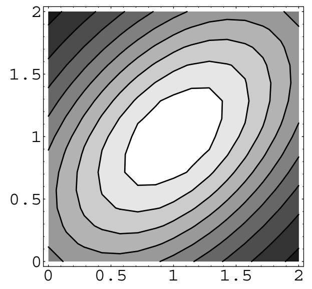
Zero mean, zero correlation

```
m1 = {0,0};
Cov = {{1,0},{0,1}};
ContourPlot[multigaus[{x1,x2},m1,Cov],{x1,-2,2}, {x2,-2,2}];
```



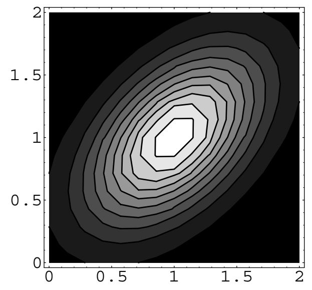
Mean = {1,1}, positive correlation

```
m1 = {1,1};
Cov = {{1,.5},{.5,1}};
ContourPlot[multigaus[{x1,x2},m1,Cov],{x1,0,2}, {x2,0,2}];
```



Mean = {1,1}, positive correlation, small variance

```
m1 = {1,1};
Cov = 0.2{{1,.5},{.5,1}};
ContourPlot[multigaus[{x1,x2},m1,Cov],{x1,0,2}, {x2,0,2}];
```



Mixture of gaussians

α is a mixing parameter

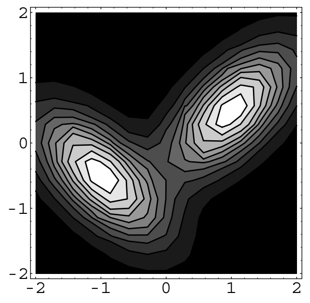
$$p(x) = \alpha p_1(x) + (1-\alpha)\, p_2(x) \text{ where } 0 \le \alpha \le 1 \tag{3}$$

α can be interpreted in terms of a prior probability of choosing which of two distributions a sample will be drawn from.

Starting with p(x,a) use the sum and product rules to derive the above mixture density where p(a=a_1) = α, and p(a = a_2)=(1-α).

```
m1 = {1,.5}; m2 = {-1,-.5};
Cov1 = 0.4*{{1,.6},{.6,1}};
Cov2 = 0.4*{{1,-.6},{-.6,1}};
mix[x_] := 0.5 (multigaus[x,m1,Cov1] + multigaus[x,m2,Cov2]);
```

```
ContourPlot[mix[{x1,x2}],{x1,-2,2}, {x2,-2,2}];
```



■ **Drawing samples from the density--draw from a hat method**

■ **We'll simulate the process of filling a hat with slips of paper, where the number of slips is proportional to the probability the number being in some range (dx1,dx2)**

```
m1 = {0,0};
Cov = {{1,.8},{.8,1}};
dx1 = 0.1;
dx2 = dx1;

Nslips=100;
hat = {};
For[x1=-2,x1<=2,x1=x1+dx1,
    For[x2=-2,x2<=2,x2=x2+dx2,
        np = Nslips*multigaus[{x1,x2},m1,Cov];
        For[i=1,i<np,i=i+1,
            hat = Append[hat,{x1,x2}];
        ];
    ];
];
```

hat is a list of pairs of numbers for which the frequency of occurence of pairs is determined by multigauss.

```
Dimensions[hat]
```

```
{8698, 2}
```

■ **Now let's do a check, where we compile a histogram representing the frequencies of each slip**

First, define the "bins" in domain, that we'll use to check for matches:

```
domain = {};
For[x1=-2,x1<=2,x1=x1+dx1,
    For[x2=-2,x2<=2,x2=x2+dx2,
            domain = Append[domain,{x1,x2}];
    ];
];
```

An alternate way of specifying the domain using Outer[], and Range[]:

```
domain2 = Flatten[Outer[List,Range[-2,2,dx1],Range[-2,2,dx1]],1];
```

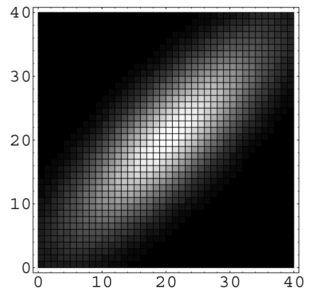
Now we'll count how many times we find that an element of hat matches a domain element:

```
Freq = Map[Count[hat,#]&,domain];
```

```
width = Sqrt[Dimensions[Freq]]
```

```
{40}
```

```
ListDensityPlot[Partition[Freq,width]];
```

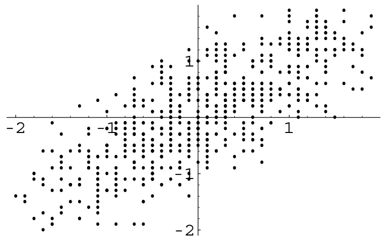


■ **Now draw a sample--simulate pulling a slip from hat**

```
rv:=hat[[Random[Integer,{1,Length[hat]}]]];
```

```
test = Table[hat[[Random[Integer,{1,Length[hat]}]]],{600}];
```

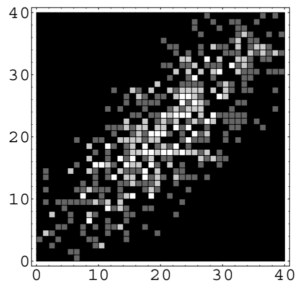
```
g1=ListPlot[test];
```



Of course, we can't see the frequency of draws in this plot, so let's count up the number of occurences per bin, and plot up the results as we did above.

```
Freq2 = Map[Count[test,#]&,domain];
width = Sqrt[Dimensions[Freq2]];
```

```
ListDensityPlot[Partition[Freq2,width]];
```



Drawing multivariate samples from the density -- use the inverse cumulative distribution

Gaussian multivariate mixtures & exploring marginals

This time we'll use that Add-on *Mathematica* functions for multivariate gaussians.

■ Define PDF, CDF

```
m1 = {1,.5};
r=0.4*{{1,.6},{.6,4}};
ndist = MultinormalDistribution[m1, r];
```

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

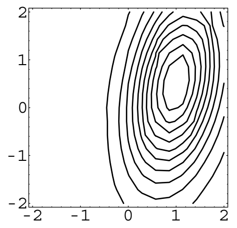
$$0.20855\, e^{\frac{1}{2}(-(-1+x1)(2.74725(-1+x1)-0.412088(-0.5+x2))-(-0.412088(-1+x1)+0.686813(-0.5+x2))(-0.5+x2))}$$

What is the probability of the distribution in the region $x_1 < -2 \bigcap x_2 < 1$.

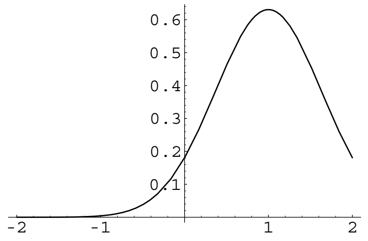
```
CDF[ndist, {-2, 1}]
```

1.02471×10^{-6}

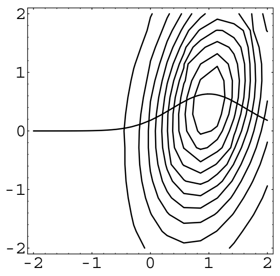
```
g1=ContourPlot[PDF[ndist, {x1, x2}],{x1,-2,2}, {x2,-2,2},ContourShading->False];
```



```
marginal[x1_] := Integrate[PDF[ndist, {x1, x2}], {x2, -Infinity, Infinity}] ;
g2 = Plot[marginal[x1], {x1, -2, 2}];
```



```
Show[{g1, g2}];
```



■ Drawing samples

As we've used in earlier lectures, drawing samples is done by:

```
Random[ndist]
```

```
{-0.52553, -2.54926}
```

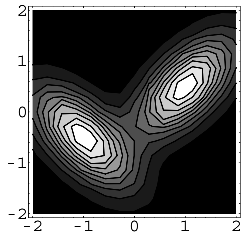
■ Mixtures of gaussians

```
Clear[mix];
```

```
r1=0.4*{{1,.6},{.6,1}};
r2=0.4*{{1,-.6},{-.6,1}};
m1 = {1,.5}; m2 = {-1,-.5};
ndist1 = MultinormalDistribution[m1, r1];
ndist2 = MultinormalDistribution[m2, r2];
```

```
mix[x_] := 0.5 (PDF[ndist1, x] + PDF[ndist2, x]);
```

```
ContourPlot[mix[{x1,x2}],{x1,-2,2}, {x2,-2,2}];
```

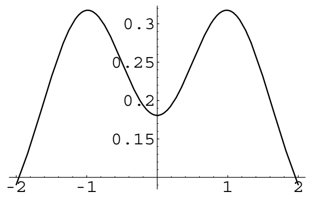


■ Marginals for mixture

marginal[x1_] := Integrate[mix[{x1, x2}], {x2, -Infinity, Infinity}] (4)

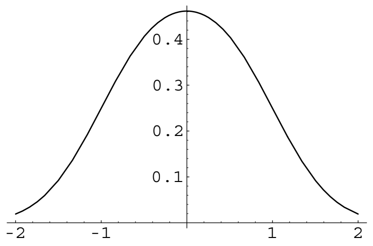
```
Clear[marginal];
marginal[x1_] :=
  0.5 * (Integrate[PDF[ndist1, {x1, x2}], {x2, -Infinity, Infinity}] +
     Integrate[PDF[ndist2, {x1, x2}], {x2, -Infinity, Infinity}]);
```

```
Plot[marginal[x1], {x1, -2, 2}];
```



```
Clear[marginal];
marginal[x2_] :=
  0.5 * (Integrate[PDF[ndist1, {x1, x2}], {x1, -Infinity, Infinity}] +
     Integrate[PDF[ndist2, {x1, x2}], {x1, -Infinity, Infinity}]);
```

```
Plot[marginal[x2], {x2, -2, 2}];
```



Side comments & where we'll see this again

■ Projection pursuit

Which projection (marginal) is more "interesting"--the one onto x1 or onto x2?

Exploratory projection pursuit. (e.g. Intrator, 1993).

■ Inference: Learning parameters of mixture distributions

Return later to the inference problem: Given data, estimate the mixing parameters, means and covariances. EM algorithm.

Bayesian learning of univariate Gaussian mean: MAP

■ Suppose we know the data comes from a Gaussian generative process, but we don't know the mean?

From a statistical point of view, one form of learning is "density estimation" from histogram measurements. In high dimensions this is hard, unless we have a low-dimensional parametric model for the density--i.e. the density is modeled in terms of a few parameters. So for example. The 1D Gaussian could be approximated by a huge list of numbers--one for each bin, each number is an estimate of the probabily of the random variable being in that bin. But because it is Gaussian, we can be more efficient by representing the density in terms of just two numbers (mean and variance), and a formula.

In this context, learning becomes parameter estimation.

Suppose we have a set of samples that come from a Gaussian distribution with known variance σ^2 , but unknown mean μ.

$$x_i = \text{noise, where noise} \sim N[\mu, \sigma], \text{ or equivalently} \quad (5)$$
$$x_i = \mu + \text{noise, where noise} \sim N[0, \sigma]$$

```
ndist0 = NormalDistribution[μ, σ];
```

Although we don't know the mean, we assume a Gaussian prior on the mean:

$$\mu \sim N[\mu 0, \sigma 0] \quad (6)$$

I.e. we assume we know the mean's mean and variance. But we are willing to change our estimate of the mean given new data--i.e. given the posterior.

In[211]:=
```
ndistμ = NormalDistribution[μ0, σ0];
PDF[ndistμ, μ]
```

Out[212]=
$$\frac{e^{-\frac{(\mu-\mu 0)^2}{2\,\sigma 0^2}}}{\sqrt{2\pi}\,\sigma 0}$$

Suppose the generative model N[μ, σ] produces three i.i.d. (independent, identically distributed) samples x_1, x_2, x_3. What is the MAP estimate of μ? Which values of μ make the posterior biggest? We use Bayes rule:

$$p(\mu \mid x_1, x_2, x_3) = \frac{p(x_1, x_2, x_3 \mid \mu)\, p(\mu)}{p(x_1, x_2, x_3)} \quad (7)$$

$p(x_1 \mid \mu)$ is given by :

In[165]:=
```
PDF[ndist0, x1]
```

Out[165]=
$$\frac{e^{-\frac{(-\mu+x_1)^2}{2\,\sigma^2}}}{\sqrt{2\pi}\,\sigma}$$

■ Calculating the MAP estimate of mean

In[213]:=
```
g = PDF[ndist0, x1] * PDF[ndist0, x2] * PDF[ndist0, x3] * PDF[ndistμ, μ];
t = Log[g];
t = PowerExpand[t]
t = D[t, μ]
Solve[-t == 0, μ]
```

Out[215]=
$$-\frac{(\mu-\mu 0)^2}{2\,\sigma 0^2} - \mathrm{Log}[4] - 2\,\mathrm{Log}[\pi] - 3\,\mathrm{Log}[\sigma] - \mathrm{Log}[\sigma 0] - \frac{(-\mu+x_1)^2}{2\,\sigma^2} - \frac{(-\mu+x_2)^2}{2\,\sigma^2} - \frac{(-\mu+x_3)^2}{2\,\sigma^2}$$

Out[216]=
$$-\frac{\mu-\mu 0}{\sigma 0^2} + \frac{-\mu+x_1}{\sigma^2} + \frac{-\mu+x_2}{\sigma^2} + \frac{-\mu+x_3}{\sigma^2}$$

Out[217]=
$$\left\{\left\{\mu \to \frac{\frac{\mu 0}{\sigma 0^2} + \frac{x_1}{\sigma^2} + \frac{x_2}{\sigma^2} + \frac{x_3}{\sigma^2}}{\frac{3}{\sigma^2} + \frac{1}{\sigma 0^2}}\right\}\right\}$$

In general, one can update from n samples in batch mode:

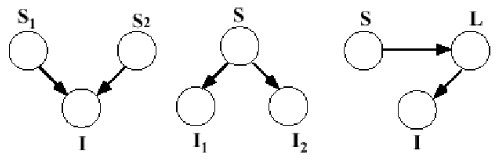
$$\left\{\left\{\mu \to \frac{\frac{\mu 0}{\sigma 0^2} + \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i}{\frac{n}{\sigma^2} + \frac{1}{\sigma 0^2}}\right\}\right\} \qquad (8)$$

For the multi-variate case, see Duda and Hart.

Graphical Models of dependence

■ Graphs: causal structure and conditional independence

The idea is to represent the probabilistic structure of the joint distribution P(S,L,I) by a Bayes net (e.g. Ripley, 1996}, which is a graphical model that expresses how variables influence each other. There are just three basic building blocks: converging, diverging, and intermediate nodes. For example, multiple causal variables causing a given measurement, a single variable producing multiple measurements, or a cause indirectly influencing a measurement through an intermediate variable. These types of influence provide a first step towards modeling the joint distribution and the means to compute probabilities of the unknown variables given known values.



Components of the generative structure for data patterns involve converging, diverging,and intermediate nodes. For example,these could correspond to:multiple (scene) causes {shape S1, illumination S2 giving rise to the same image measure-ment, I ; one cause, S influencing more than one image measurement, {color, I1, brightness, I2}; a scene (or other) cause S, {object identity, S} influencing an image measurement (image contour) through an intermediate variable L (3D shape) .

The arrows tell us how to factor the joint probability into conditionals. So for the three examples above, we have:

p(S1,S2,I)=p(I|S1,S2)p(S1)p(S2)

p(S,I1,I2)=p(I1|S)p(I2|S)p(S)

p(S,L,I)=p(I|L)p(L|S)p(S)

■ Primary, secondary variables.

We can interpret the causal structure in terms of conditional probability.

The data measurements (x) are determined by a typically non-linear function (ϕ) of primary signal variables (S_e) and confounding secondary variables (S_g). Knowledge is represented by the joint probability p(x,S_e,S_g). In general, the causal structure of natural data (e.g. image or speech) patterns is more complex and consequently requires elaboration of its graphical representation. For pattern inference theory, the task is to make a decision about the signal hypotheses or primary signal variables, while discounting the noise or secondary variables. Thus optimal perceptual decisions are determined by p(x,S_e), which is derived by summing over the secondary variables (i.e. marginalizing with respect to the secondary variables): $\int_{S\_g} p(x, S\_e, S\_g)\, dS\_g$.

Influences between variables are represented by conditioning, and a graphical model expresses the conditional independencies between variables. Two random variables may only become independent, however, once the value of some third variable is known. This is called conditional independence.Recall from above that two random variables are independent if and only if their joint probability is equal to the product of their individual probabilities. Thus, if p(A,B) = p(A)p(B), then A and B are independent. If p(A,B|C) = p(A|C)p(B|C), then A and B are conditionally independent.

When corn prices drop in the summer, hay fever incidence goes up. However, if the joint on corn price and hay fever is conditioned on ``ideal weather for corn and ragweed", the correlation between corn prices and hay fever drops. This is because corn price and hay fever symptoms are conditionally independent.

There is a correlation between eating ice cream and drowning. Why? What event should you condition on to make the dependence go away?

■ What is noise? Primary and secondary variables

Noise is whatever you don't care to estimate, but contributes to the data. The secondary variables are noise.

Optimal Inference and task dependence: Fruit example

(due to James Coughlan; in Yuille, Coughlan, Kersten & Schrater).

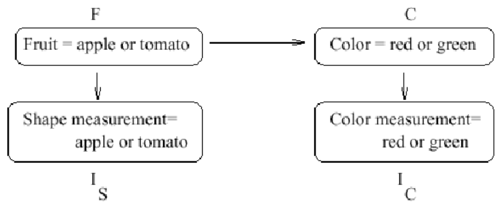

F
Fruit = apple or tomato
C
Color = red or green
Shape measurement= apple or tomato
Color measurement= red or green
I S
I C

Figure from Yuille, Coughlan, Kersten & Schrater.

The the graph specifies how to decompose the joint probability:

p[F, C, Is, Ic] = p[Ic | C] p[C | F] p[Is | F] p[F]

The prior model on hypotheses, F & C

More apples (F=1) than tomatoes (F=2), and:

```
ppF[F_] := If[F == 1, 9 / 16, 7 / 16];
TableForm[Table[ppF[F], {F, 1, 2}], TableHeadings -> {{"F=a", "F=t"}}]
```

```
F=a     9/16
F=t     7/16
```

The conditional probability **cpCF[C|F]**:

```
cpCF[F_, C_] := Which[F == 1 && C == 1, 5 / 9,
   F == 1 && C == 2, 4 / 9, F == 2 && C == 1, 6 / 7, F == 2 && C == 2, 1 / 7];
TableForm[Table[cpCF[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

```
        C=r     C=g
F=a     5/9     4/9
F=t     6/7     1/7
```

So the joint is:

```
jpFC[F_, C_] := cpCF[F, C] ppF[F];
TableForm[Table[jpFC[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

```
        C=r     C=g
F=a     5/16    1/4
F=t     3/8     1/16
```

We can marginalize to get the prior probability on color alone is:

```
ppC[C_] := Sum[jpFC[F, C], {F, 1, 2}]
```

$$\text{ppC[C\_]} := \sum_{F=1}^{2} \text{jpFC[F, C]}$$

Question: Is fruit identity independent of material color--i.e. is F independent of C?

■ Answer

No.

```
TableForm[Table[jpFC[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
TableForm[Table[ppF[F] ppC[C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

```
        C=r      C=g
F=a     5/16     1/4
F=t     3/8      1/16
```

```
        C=r      C=g
F=a     99/256   45/256
F=t     77/256   35/256
```

The generative model: Imaging probabilities

Analogous to collecting histograms for the two switch positions in the SDT experiment, suppose that we have gathered some "image statistics" which provides us knowledge of how the image measurements for shape Is, and for color Ic depend on the type of fruit F, and material color, C. For simplicity, our measurements are discrete and binary (a more realistic case, they would have continuous values), say Is = {am, tm}, and Ic = {rm, gm}.

P(I_S=am,tm | F=a) = {11/16, 5/16}

P(I_S=am,tm | F=t) = {5/8, 3/8}

P(I_C=rm,gm | C=r) = {9/16, 7/16}

P(I_C=rm,gm | C=g) = {1/2, 1/2}

We use the notation am, tm, rm, gm because the measurements are already suggestive of the likely cause. So there is a correlation between apple and apple-like shapes, am; and between red material, and "red" measurements. In general, there may not be an obvious correlation like this.

We define a function for the probability of Ic given C, **cpIcC[Ic | C]**:

```
cpIcC[Ic_, C_] := Which[Ic == 1 && C == 1, 9 / 16,
  Ic == 1 && C == 2, 7 / 16, Ic == 2 && C == 1, 1 / 2, Ic == 2 && C == 2, 1 / 2];
TableForm[Table[cpIcC[Ic, C], {Ic, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"Ic=rm", "Ic=gm"}, {"C=r", "C=g"}}]
```

```
          C=r      C=g
Ic=rm     9/16     7/16
Ic=gm     1/2      1/2
```

The probability of Is conditional on F is **cpIsF[Is | F]**:

```
cpIsF[Is_, F_] := Which[Is == 1 && F == 1, 11 / 16,
   Is == 1 && F == 2, 5 / 8, Is == 2 && F == 1, 5 / 16, Is == 2 && F == 2, 3 / 8];
TableForm[Table[cpIsF[Is, F], {Is, 1, 2}, {F, 1, 2}],
 TableHeadings -> {{"Is=am", "Is=tm"}, {"F=a", "F=t"}}]
```

```
          F=a      F=t
Is=am     11/16    5/8
Is=tm     5/16     3/8
```

The total joint probability

We now have enough information to put probabilities on the 2x2x2 "universe" of possibilities, i.e. all possible combinations of fruit, color, and image measurements. Looking at the graphical model makes it easy to use the product rule to construct the total joint, which is:

p[F, C, Is, Ic] = p[Ic | C] p[C | F] p[Is | F] p[F]:

```
jpFCIsIc[F_, C_, Is_, Ic_ ] := cpIcC[ Ic, C ] cpCF[F, C ] cpIsF[Is, F ] ppF[F ]
```

Usually, we don't need the probabilities of the image measurements (because once the measurements are made, they are fixed and we want to compare the probabilities of the hypotheses. But in our simple case here, once we have the joint, we can calculate the probabilities of the image measurements through marginalization p(Is,Ic)=$\sum_C \sum_F p(F, C, \text{Is}, \text{Ic})$, too:

```
                      2    2
jpIsIc[Is_, Ic_] :=   ∑    ∑   jpFCIsIc[F, C, Is, Ic ]
                     C=1  F=1
```

Three MAP tasks

Suppose that we measure Is=am, and Is = rm. The measurements suggest "red apple", but to find the most probable, we need to take into account the priors too.

■ Define argmax[] function:

```
argmax[x_] := Position[x, Max[x]];
```

■ Pick most probable fruit AND color--Answer "red tomato"

Using the total joint, p(F,C | Is, Ic) = $\frac{p(F,C,\text{Is},\text{Ic})}{p(\text{Is},\text{Ic})} \propto$ p(F,C,Is,Ic)

```
TableForm[jpFCIsIcTable = Table[jpFCIsIc[F, C, 1, 1], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
Max[jpFCIsIcTable]
argmax[jpFCIsIcTable]
```

	C=r	C=g
F=a	$\frac{495}{4096}$	$\frac{77}{1024}$
F=t	$\frac{135}{1024}$	$\frac{35}{2048}$

$\frac{135}{1024}$

```
{{2, 1}}
```

"Red tomato" is the most probable once we take into account the difference in priors.

Calculating p(F,C | Is, Ic). We didn't actually need p(F,C | Is, Ic), but we can calculate it by conditioning the total joint on the probability of the measurments:

```
jpFCcIsIc[F_, C_, Is_, Ic_] := jpFCIsIc[F, C, Is, Ic] / jpIsIc[Is, Ic]
```

```
TableForm[jpFCcIsIcTable = Table[jpFCcIsIc[F, C, 1, 1], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
Max[jpFCcIsIcTable]
argmax[jpFCcIsIcTable]
```

	C=r	C=g
F=a	$\frac{55}{157}$	$\frac{308}{1413}$
F=t	$\frac{60}{157}$	$\frac{70}{1413}$

$\frac{60}{157}$

```
{{2, 1}}
```

■ Pick most probable color--Answer "red"

In this case, we want maximize the posterior:

p(C | Is, Ic)=$\sum_{F=1}^{2} p(F, C \mid \text{Is}, \text{Ic})$

```
pC[C_, Is_, Ic_] := Sum[jpFCcIsIc[F, C, Is, Ic], {F, 1, 2}]
```

$$\text{pC[C\_, Is\_, Ic\_]} := \sum_{F=1}^{2} \text{jpFCcIsIc[F, C, Is, Ic]}$$

```
TableForm[pCTable = Table[pC[C, 1, 1], {C, 1, 2}],
 TableHeadings -> {{"C=r", "C=g"}}]
Max[pCTable]
argmax[pCTable]
```

C=r	$\frac{115}{157}$
C=g	$\frac{42}{157}$

$\frac{115}{157}$

```
{{1}}
```

Answer is that the most probable material color is C = r, "red".

■ Pick most probable fruit--Answer "apple"

p(F | Is, Ic)

$$\text{pF[F\_, Is\_, Ic\_]} := \sum_{C=1}^{2} \text{jpFCcIsIc[F, C, Is, Ic]}$$

```
TableForm[pFTable = Table[pF[F, 1, 1 ], {F, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}}]
Max[pFTable]
argmax[pFTable]
```

F=a $\frac{803}{1413}$

F=t $\frac{610}{1413}$

$\frac{803}{1413}$

```
{{1}}
```

The answer is "apple"

■ Moral of the story: Optimal inference depends on the precise definition of the task

Appendices

Using *Mathematica* lists to manipulate discrete priors, likelihoods, and posteriors

■ A note on list arithmetic

We haven't done standard matrix/vector operations above to do conditioning. We've take advantage of how *Mathematica* divides a 2x3 array by a 2-element vector:

```
M=Array[m,{2,3}]
X = Array[x,{2}]
```

$$\begin{pmatrix} m(1,1) & m(1,2) & m(1,3) \\ m(2,1) & m(2,2) & m(2,3) \end{pmatrix}$$

$\{x(1), x(2)\}$

```
M/X
```

$$\begin{pmatrix} \frac{m(1,1)}{x(1)} & \frac{m(1,2)}{x(1)} & \frac{m(1,3)}{x(1)} \\ \frac{m(2,1)}{x(2)} & \frac{m(2,2)}{x(2)} & \frac{m(2,3)}{x(2)} \end{pmatrix}$$

■ Putting the probabilities back together again to get the joint

```
Transpose[Transpose[pHx] px]
```

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

```
pxH pH
```

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

■ Getting the posterior from the priors and likelihoods:

One reason Bayes' theorem is so useful is that it is often easier to formulate the likelihoods (e.g. from a causal or generative-model of how the data could have occurred), and the priors (often from heuristics, or in computational vision empirically testable models of the external visual world). So let's use *Mathematica* to derive **p(H|x)** from **p(x|H)** and **p(H)** , (i.e. pHx from pxH and pH).

```
px2 = Plus @@ (pxH pH)
```

$\left\{\frac{5}{12}, \frac{1}{4}, \frac{1}{3}\right\}$

```
Transpose[Transpose[(pxH pH)] / Plus @@ (pxH pH)]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

■ Show that this joint probability has a uniform prior (i.e. both priors equal).

```
p = {{1 / 8, 1 / 8, 1 / 4}, {1 / 4, 1 / 8, 1 / 8}}
```

$$\left\{\left\{\frac{1}{8},\frac{1}{8},\frac{1}{4}\right\},\left\{\frac{1}{4},\frac{1}{8},\frac{1}{8}\right\}\right\}$$

Marginalization and conditioning: A small dimensional example using list manipulation in *Mathematica*

■ A discrete joint probability

All of our knowledge regarding the signal discrimination problem can be described in terms of the joint probability of the hypotheses, **H** and the possible data measurements, **x.** The probability function assigns a number to all possible combinations:

p[H, x]

That is, we are assuming that both the hypotheses and the data are discrete random variables.

$$H = \left\{\begin{array}{l} S1 \\ S2 \end{array}\right.$$

$$x \in \{1, 2, \ldots\}$$

Let's assume that x can only take on one of three values, 1, 2, or 3. And suppose the joint probability is:

$$p = \left\{\left\{\frac{1}{12},\frac{1}{12},\frac{1}{6}\right\},\left\{\frac{1}{3},\frac{1}{6},\frac{1}{6}\right\}\right\}$$

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

```
TableForm[p, TableHeadings -> {{"H=S1", "H=S2"}, {"x=1", "x=2", "x=3"}}]
```

	x=1	x=2	x=3
H=S1	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$
H=S2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$

The total probability should sum up to one. Let's test to make sure. We first turn the list of lists into a singel list of scalars using **Flatten[]**. And then we can sum either with **Apply[Plus,Flatten[p]].**

```
Plus @@ Flatten[p]
```

```
1
```

We can pull out the first row of p like this:

```
p[[1]]
```

$$\left\{\frac{1}{12},\frac{1}{12},\frac{1}{6}\right\}$$

Is this the probability of x? No. For a start, the numbers don't sum to one. But we can get it through the two processes of marginalization and conditioning.

■ Marginalizing

What are the probabilities of the data, p(x)? To find out, we use the *sum rule* to sum over the columns:

```
px = Apply[Plus, p]
```

$$\left\{\frac{5}{12},\frac{1}{4},\frac{1}{3}\right\}$$

"Summing over "is also called **marginalization** or **"integrating out".** Note that marginalization turns a probability function with higher degrees of freedom into one of lower degrees of freedom.

What are the prior probabilities? p(H)? To find out, we sum over the rows:

```
pH = Apply[Plus, Transpose[p]]
```

$\{\frac{1}{3}, \frac{2}{3}\}$

■ Conditioning

Now that we have the marginals, we can get use the *product rule* to obtain the conditional probability through conditioning of the joint:

$$p[x \mid H] = \frac{p[H,\ x]}{p[H]}$$

In the Exercises, you can see how to use *Mathematica* to do the division for conditioning. The syntax is simple:

```
pxH = p / pH
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Note that the probability of x conditional on H sums up to 1 over x, i.e. each row adds up to 1. But, the columns do not. **p[x|H]** is a **probability** function of x, but a **likelihood** function of H. The posterior probability is obtained by conditioning on x:

$$p[H \mid x] = \frac{p[H,\ x]}{p[x]}$$

Syntax here is a bit more complicated, because the number of columns of px don't match the number of rows of p. We use Transpose[] to exchange the columns and rows of p before dividing, and then use Transpose again to get back the 2x3 form:

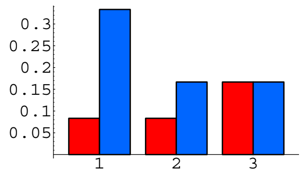
```
pHx = Transpose[Transpose[p] / px]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

Plotting the joint

The following BarChart[] graphics function requires in add-in package (**<< Graphics`Graphics`**), which is specified at the top of the notebook. You could also use **ListDensityPlot[]**.

```
BarChart[p[[1]], p[[2]]];
```



Marginalization and conditioning: An example using *Mathematica* functions

■ A discrete joint probability

All of our knowledge regarding the signal discrimination problem can be described in terms of the joint probability of the hypotheses, **H** and the possible data measurements, **x.** The probability function assigns a number to all possible combinations:

p[H, x]

That is, we are assuming that both the hypotheses and the data are discrete random variables.

$$H = \begin{cases} S1 \\ S2 \end{cases}$$

$x \in \{1, 2, \ldots\}$

Let's assume that x can only take on one of three values, 1, 2, or 3. And suppose the joint probability is:

```
p[H_, x_] := Which[H == 1 && x == 1, 1 / 12, H == 1 && x == 2, 1 / 12, H == 1 && x == 3,
   1 / 6, H == 2 && x == 1, 1 / 3, H == 2 && x == 2, 1 / 6, H == 2 && x == 3, 1 / 6];
```

```
TableForm[Table[p[H, x], {H, 1, 2}, {x, 1, 3}],
 TableHeadings -> {{"H=s1", "H=s2"}, {"X=1", "X=2", "X=3"}}]
```

	X=1	X=2	X=3
H=s1	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$
H=s2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$

The total probability should sum up to one. Let's test to make sure. We first turn the list of lists into a singel list of scalars using **Flatten[]**. And then we can sum either with **Apply[Plus,Flatten[p]].**

```
Sum[p[H, x], {H, 1, 2}, {x, 1, 3}]
```

```
1
```

We can pull out the first row of p like this:

```
Table[p[1, x], {x, 1, 3}]
```

$$\left\{\frac{1}{12}, \frac{1}{12}, \frac{1}{6}\right\}$$

Is this the probability of x? No. For a start, the numbers don't sum to one. But we can get it through the two processes of marginalization and conditioning.

■ Marginalizing

What are the probabilities of the data, p(x)? To find out, we use the *sum rule* to sum over the columns:

```
px[x_] := Sum[p[H, x], {H, 1, 2}];
```

```
Table[px[x], {x, 1, 3}]
```

$$\left\{\frac{5}{12}, \frac{1}{4}, \frac{1}{3}\right\}$$

"Summing over "is also called **marginalization** or **"integrating out".** Note that marginalization turns a probability function with higher degrees of freedom into one of lower degrees of freedom.

What are the prior probabilities? p(H)? To find out, we sum over the rows:

```
pH[H_] := Sum[p[H, x], {x, 1, 3}];
```

```
Table[pH[H], {H, 1, 2}]
```

$$\left\{\frac{1}{3}, \frac{2}{3}\right\}$$

■ Conditioning

Now that we have the marginals, we can get use the *product rule* to obtain the conditional probability through conditioning of the joint:

$$p[x \mid H] = \frac{p[H,\ x]}{p[H]}$$

We use function definition in *Mathematica* to do the division for conditioning. The syntax is simple:

```
pxH[H_, x_] := p[H, x] / pH[H];
```

```
Table[pxH[H, x], {H, 1, 2}, {x, 1, 3}]
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Note that the probability of x conditional on H sums up to 1 over x, i.e. each row adds up to 1. But, the columns do not. **p[x|H]** is a **probability** function of x, but a **likelihood** function of H. The posterior probability is obtained by conditioning on x:

$$p[H \mid x] = \frac{p[H,\ x]}{p[x]}$$

```
pHx[H_, x_] := p[H, x] / px[x];
```

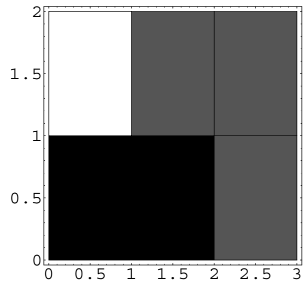
```
Table[pHx [H, x], {H, 1, 2}, {x, 1, 3}]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

Plotting the joint

We use **ListDensityPlot[]**.

```
ListDensityPlot[Table[p[H, x], {H, 1, 2}, {x, 1, 3}]];
```



References

Applebaum, D. (1996). Probability and Information . Cambridge, UK: Cambridge University Press.

Cover, T. M., & Joy, A. T. (1991). *Elements of Information Theory*. New York: John Wiley & Sons, Inc.

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

Intrator, N. Combining Exploratory Projection Pursuit and Projection Pursuit Regression. Neural Computation (5):443-455, 1993. http://www.physics.brown.edu/users/faculty/intrator/papers/epp-ppr.ps.gz

Kersten, D. and P.W. Schrater (2000), *Pattern Inference Theory: A Probabilistic Approach to Vision*, in *Perception and the Physical World*, R. Mausfeld and D. Heyer, Editors. , John Wiley & Sons, Ltd.: Chichester. (pdf)

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

Yuille, A., Coughlan J., Kersten D.(1998) (pdf)