Introduction to Neural Networks
U. Minn. Psy 5038

More probability

■ **Initialize standard library files:**

In[249]:=
```
Off[General::spell1];
SetOptions[ContourPlot, ImageSize → Small];
SetOptions[Plot, ImageSize → Small];
SetOptions[ListPlot, ImageSize → Small];
```

```
<< "BarCharts`";
```

In[254]:=
```
SetOptions[BarChart, ImageSize → Small];
```

Goals

Review the basics of probability distributions and statistics

More on generative modeling: drawing samples

Graphical models for inference

Optimal inference and Task dependence

Probability overview

Random variables, discrete probabilities, probability densities, cumulative distributions

■ **Discrete: random variable X can take on a finite set of discrete values**

X = {x(1),...,x(N)]

$$\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} p(X = x(i)) = 1$$

■ Densities: X takes on continuous values, x, in some range.

Density : $p(x)$

Analogous to material mass,
we can think of the probability over some small domain of the random variable as " probability mass " :

$$\text{prob}(x < X < \text{dx} + x) = \int_{X}^{\text{dX}+X} p(x)\, dx$$

$$\text{prob}(x < X < \text{dx} + x) \simeq p(x)\, \text{dx}$$

With the mass analogy, however, an object (event space) always " weighs " 1 :

$$\int_{-\infty}^{\infty} p(x)\, dx = 1$$

Cumulative distribution:

$$\text{prob}\,(X < x) = \int_{-\infty}^{x} p(X)\, dX$$

■ Densities of discrete random variables

The Dirac Delta function, $\delta[\bullet]$, allows us to use the mathematics of continuous distributions for discrete ones, by defining the density as:

$\text{p[x]} = \sum_{i=1}^{N} p_i \delta[\text{x - x[i]}]$, where $\delta[\text{x - x[i]}] = \begin{cases} \infty & \text{for } x = x[i] \\ 0 & \text{for } x \neq x[i] \end{cases}$

Think of the delta function, $\delta[\bullet]$, as ϵ wide and $1/\epsilon$ tall, and then let $\epsilon \to 0$, so that:

$$\int_{-\infty}^{\infty} \delta(y)\, dy = 1$$

The density, p[x], is a series of spikes. It is infinitely high only at those points for which x = x[i], and zero elsewhere. But "infinity" is scaled so that the local mass or area around each point x[i], is p_i.

■ Joint probabilities

Prob (X AND Y) = $p(X, Y)$
Joint density : $p(x, y)$

Three basic rules of probability

Suppose we know everything there is to know about a set of variables (A,B,C,D,E). What does this mean in terms of probability? It means that we know the joint distribution, p(A,B,C,D,E). In other words, for any particular combination of values (A=a,B=b, C=c, D=d,E=e), we can calculate, look up in a table, or determine some way or another the number p(A=a,B=b, C=c, D=d,E=e).

Deterministic relationships are special cases.

■ Rule 1: Conditional probabilities from joints: The product rule

Probability about an event changes when new information is gained.

Prob(X given Y) = p(X|Y)

$$p(X \mid Y) = \frac{p(X, Y)}{p(Y)}$$

$$p(X, Y) = p(X \mid Y)\, p(Y)$$

The form of the product rule is the same for densities as for probabilities.

■ Rule 2: Lower dimensional probabilities from joints: The sum rule (marginalization)

$$p(X) = \sum_{i=1}^{N} p(X, Y(i))$$

$$p(x) = \int_{-\infty}^{\infty} p(x, y)\, dy$$

■ Rule 3: Bayes' rule

From the product rule, and since p[X,Y] = p[Y,X], we have:

$$p(Y \mid X) = \frac{p(X \mid Y)\, p(Y)}{p(X)}, \text{ and using the sum rule, } p(Y \mid X) = \frac{p(X \mid Y)\, p(Y)}{\sum_Y p(X, Y)}$$

Bayes Terminology in inference

Suppose we have some partial data (see half of someone's face), and we want to recall or complete the whole. Or suppose that we hear a voice, and from that visualize the face. These are both problems of statistical inference. We've already studied how to complete a partial pattern using energy minimization, and how energy minimization can be viewed as probability maximization.

We typically think of the **Y** term as a random variable over the hypothesis space (a face), and **X** as data or a stimulus (partial face, or sound). So for recalling a pattern **Y** from an input stimulus **X,** We'd like to have a function that tells us:

p(Y | X) which is called the **posterior** probability of the hypothesis (face) given the stimulus (partial face or sound).

-- i.e. what you get when you condition the joint by the stimulus data. The posterior is often what we'd like to base our decisions on, because it can be proved that picking the hypothesis **Y** which maximizes the posterior (i.e. maximum a posteriori or **MAP** estimation) minimizes the average probability of error.

p(Y) is the **prior** probability of the hypothesis. Some hypotheses are "a priori" more likely than others. But even if it isn't made explicit, a model prior usually implicitly assumes conditions. Given a context, such as your room, some faces are more likely than others. For me an image patch stimulating my retina in my kitchen is much more likely to be my wife's than my brother's (who lives in another state). Priors are contingent, i.e. conditional on context, **p(Y| context)**, even if the context is not made explicity.

p(X|Y) is the **likelihood** of the hypothesis. Note this is a probability of **X**, but not of **Y**.(The sum over X is one, but the sum over Y isn't necessarily one.)

Independence

Knowledge of one event doesn't change the probability of another event.

p(X)=p(X|Y)

p(X,Y)=p(X)p(Y)

Density mapping theorem

Suppose we have a change of variables that maps a discrete set of x's uniquely to y's: X->Y.

Discrete random variables

No change to probability function. The mapping just corresponds to a change of labels, so the probabilities p(X)=p(Y).

Continuous random variables

Form of probability density function does change because we require the probability "mass" to be unchanged: p(x)dx = p(y)dy

Suppose, y=f(x)

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

In higher dimensions, the transformation is done by multiplying the density by the Jacobian, the determinant of the matrix of partial derivatives of the change of coordinates.

One can express the density mapping theorem as:

$$p_Y(y) = \int \delta(y - f(x))\, f^{-1}(x)\, p_X(x)\, dx$$

over each monotonic part of f.

Convolution theorem for adding rvs

Let x be distributed as g(x), and y as h(x). Then the probability density for z=x+y is, f(z):

$$f(z) = \int g(s)\, h(z - s)\, ds \qquad (1)$$

Statistics

■ Expectation & variance

Analogous to center of mass:

Definition of expectation or average:

$$\text{Average[X]} = \bar{X} = E[X] = \Sigma\, x[i]\, p[x[i]] \sim \sum_{i=1}^{N} x_i / N$$

$$\mu = E[X] = \int x\, p(x)\, dx$$

Some rules:

E[X+Y]=E[X]+E[Y]

E[aX]=aE[X]

E[X+a]=a+E[X]

Definition of variance:

$$\sigma^2 = \text{Var[X]} = \text{E[[X-}\mu\text{]^2]} = \sum_{j=1}^{N} ((p(x(j)))\,(x(j) - \mu))^2 = \sum_{j=1}^{N} p_j\left(x_j - \mu\right)^2$$

$$\text{Var}[X] = \int (x - \mu)^2\, p(x)\, dx \sim \sum_{i=1}^{N} (x_i - \mu)^2 \Big/ N$$

Standard deviation:

$\sigma = \sqrt{\text{Var}[X]}$

Some rules:

$\text{Var}[X] = E\left[X^2\right] - E[X]^2$
$\text{Var}[aX] = a^2\, \text{Var}[X]$

■ Covariance & Correlation

Covariance:

Cov[X,Y] =E[[X - μ_X] [Y - μ_Y]]

Correlation coefficient:

$$\rho[X, Y] = \frac{\text{Cov}[X, Y]}{\sigma_X\, \sigma_Y}$$

■ Covariance matrix

Suppose now that X is a vector: $\{X_1, X_2,...\}$

Then we can describe the covariance between pairs of elements of X:

Σ_{ij}= cov[X_i,X_j] =E[[X_i - μ_{X_i}] [X_j - μ_{X_j}]] ~ $\sum_{n=1}^{N} \left(x_i^n - \mu_{X_i}\right)\left(x_j^n - \mu_{X_i}\right)^T / N$

In matrix form, the covariance can be written:

Σ = cov[X]= E[(X-E[X])$(X - E[X])^T$]

In other words, the covariance matrix can be approximated by the average outer product. In the language of neural networks, it is a Hebbian matrix memory of pair-wise relationships.

■ Independent random variables

If p(X,Y)=p(X)p(Y), then

E[X Y] = E[X] E[Y] (uncorrelated)
Cov[X, Y] = ρ[X, Y] = 0
Var[X + Y] = Var[X] + Var[Y]

If two random variables are uncorrelated, they are not necessarily independent.

Two random variables are said to be orthogonal if their correlation is zero.

Degree of belief vs., relative frequency

What is the probability that the Vikings will win the Superbowl in 2008? The Packers? Assigning a number between 0 and 1 is assigning a degree of belief. These probabilities are also called subjective probabilities. "Odds" determine subjective probabilities, where the "odds of **x** to **y**" means probability = **x/(x+y)**.

What is the probability that a coin will come up heads? In this case, we can do an experiment. Flip the coin n times, and count the number of heads, say h[n], and then set the probability, p = h[n]/n -- the relative frequency . Of course, if we did it again, we may not get the same estimate of p. One solution often given is:

$$p = \lim_{n\to\infty} \frac{h(n)}{n}$$

A problem with this, is that there is no guarantee that a well – defined limit exists.

In some domains we can measure statistics, and model probabilities of both inputs and outputs. So the relative frequency interpretation seems reasonable. In practice, the dimensions of many problems in perception, cognition, language, and memory are so high, that it is impractical to do this. Suppose you wanted to estimate the joint probabilities of all 6 letter english words (or worse yet, 8x8 pixel images). There are over 300 million possible combinations of 26 letters--i.e. over 300 million "bins" for your word counts. Most of these would have zero or near-zero entries, and it would be hard to get good estimates of most of the joint probabilities of 6 letter combinations. Although there are ways to estimate "objective priors" in high-dimensional spaces (see below), once we use the statistical framework to model perception, say of a particular cue (say), then probabilities can become more like "subjective unconscious beliefs".

Principle of insufficient reason

■ Principle of symmetry

Suppose we have N events, x[1],x[2],x[3],...,x[N] that are all physically identical except for the label. Then assume that

$$\text{prob}(x(1)) = \text{prob}(x(2)) = \text{prob}(x(3)) = \text{prob}(x(N)) = \frac{1}{N}$$

In other words, if we have no additional information about the events,we should assume that they are uniformly distributed. I.e., assume a *uniform prior*.

What about the continous case where there is no reason to assume any particular value at all between -∞ and +∞?

Improper priors.

■ Information theory and Maximum entropy

Information theory provides a powerful extension to the principle of symmetry. Information of event (or signal) X is:

$$\text{Information}[X] = -\log_2(p(X))$$

The more improbable an event, the more suprising it is, and the more information it provides.

Using the definition of expectation above, we can specify the expectation of information (or average information), which is called entropy. Entropy of a random variable X with probability distribution p[X] is:

$$H(X) = \text{Average}(\text{Information}[X]) = -\sum_X p(X) \log_2(p(X))$$

It can be shown that out of all possible probability distributions, H(X) is biggest for the uniform distribution, p(X)=1/N. Maximum entropy is looking like the symmetry principle.

It turns out that a more powerful formulation of the principle of symmetry is maximum entropy. For example, out of all possible probability distributions of a random variable with infinity range, but with a specific mean and standard deviation, the Gaussian is unique in having the largest entropy. If the range goes from zero to infinity, and we know the mean, the maximum entropy distribution is an exponential (Cover and Thomas).

An interesting application of the maximum entropy principle is to learning image textures joint probabilities: p(I[1],...,-I[N]), where N is very big, but where one has only a relatively small number of measured statistics relative to the number of possible images (which is really huge). The measurements underdetermine the dimesionality of the probability space--i.e. there are many different probability distributions which give the same statistics. So the principle of symmetry, or insufficient reason, says to choose the one with the maximum entropy. This provides a way to model high dimensional priors. And in fact, this has been done for texture models described in a previous lecture (Zhu et al.).

Some examples of generative modeling: Multivariate gaussian, mixtures

Making a univariate (scalar) gaussian random number generator: Method 1: Just for Gaussian

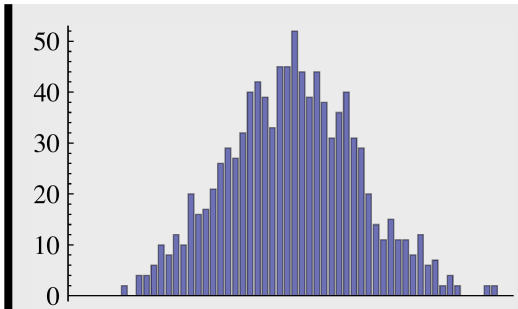
■ Use Central Limit Theorem

If all we want to do is make a Gaussian random number generator from a uniformly distributed generator, we can use the Central Limit Theorem.

Try the cell below with nusamples = 1, 2, .., 10,...

In[255]:=
```
nusamples = 10;
zl = Table[ Sum[RandomReal[], {i, 1, nusamples}] - nusamples/2, {1000}];
binsize = 0.1;
freq = BinCounts[zl, {-3, 3, binsize}];
BarChart[freq, BarLabels → None]
```

Out[259]=



Method 2: Use Density Mapping theorem. More general.

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

$$y(x) = \int_{-\infty}^{x} p_x(x')\,dx' = P(x) \qquad (2)$$

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. $P^{-1}(y)$.

Let's implement this. The quick way is to use *Mathematica*'s built-in function to get the inverse cumulative.

Method 2a: Applied to Gaussian

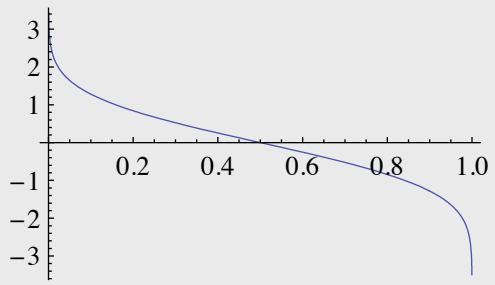
InverseErf[ ] is the inverse of :

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

We can use this to define a function for the inverse cumulative of a gaussian:

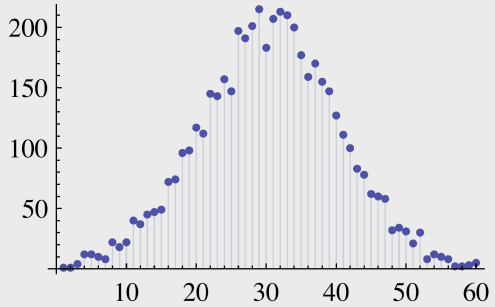
In[260]:=
```
Clear[z];
z[p_] := √2 InverseErf[1 - 2 p];
Plot[z[y], {y, 0, 1}]
```

Out[262]=



In[263]:=
```
binsize = 0.1;
zl = Table[z[RandomReal[]], {5000}];
freq = BinCounts[zl, {-3, 3, binsize}];
ListPlot[freq, Filling → Axis]
```

Out[266]=



Method 2b: From scratch: Works for almost any distribution. Do the Gaussian case for a third time

Suppose we have a discrete representation of any cumulative distribution. How can we generate samples? We'll illustrate the method with a discretization of the Gaussian. Our immediate goal is to produce a discrete approximation to the cumulative gaussian. To review where things come from, we'll start with the definition of a Gaussian, and make sure it is normalized.

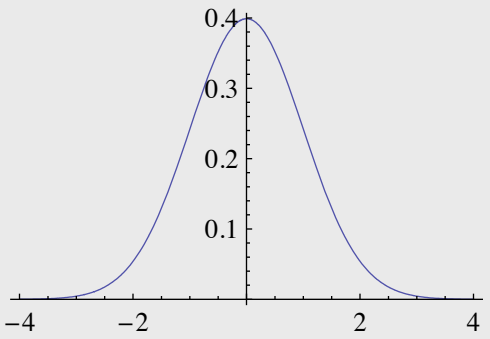
In[267]:=
```
Integrate[Exp[-(x - x0) ^ 2 / (2 * σ^ 2) ], {x, -Infinity, Infinity}]
```

Out[267]=
$$\text{If}\left[\text{Re}\left(\sigma^2\right) > 0,\ \frac{\sqrt{2\,\pi}}{\sqrt{\frac{1}{\sigma^2}}},\ \text{Integrate}\left[e^{-\frac{(x-\text{x0})^2}{2\,\sigma^2}},\ \{x,\ -\infty,\ \infty\},\ \text{Assumptions} \to \text{Re}\left(\sigma^2\right) \le 0\right]\right]$$

Let x0=0 and σ=1:

In[268]:=
```
Plot[E^(-x1^2/2)/Sqrt[2 π], {x1, -4, 4}]
```

Out[268]=



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

■ Cumulative gaussian

In[269]:=
```
Clear[cumulgauss, x, x1];
cumulgauss[x_] := NIntegrate[Exp[-(x1^2) / 2] / (Sqrt[2 * Pi]),
  {x1, -Infinity, x}]
```

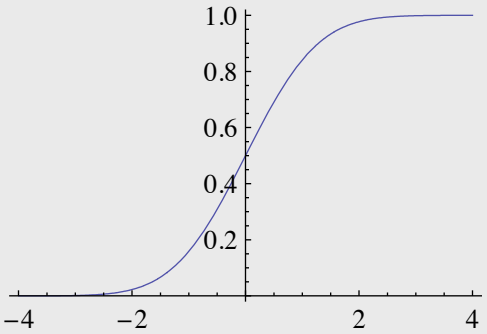
In[271]:=
```
cumulgauss[Infinity]
```

Out[271]=
```
1.
```

We can plot up cumulgauss (not very efficiently):

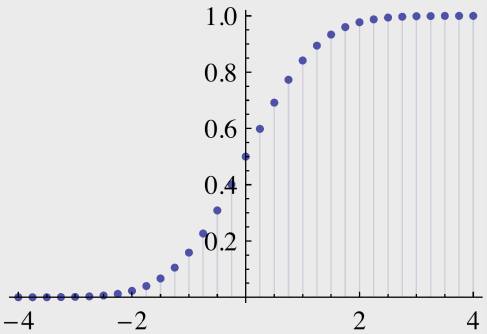
In[272]:=
```
Plot[cumulgauss[x], {x, -4, 4}]
```

Out[272]=



In[273]:=
```
lcumulgauss = Table[{x, cumulgauss[x]}, {x, -4.`, 4.`, 0.25`}];
ListPlot[lcumulgauss, Filling → Axis]
```

Out[274]=



■ Make inverse cumulative gaussian table

In[275]:=
```
invlcumulgauss = RotateLeft[lcumulgauss, {0, 1}];
```

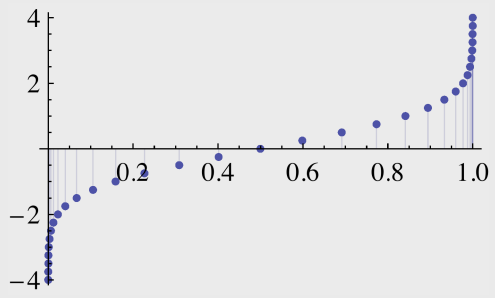
To see what this does, evaluate:

In[276]:=
```
{{x1, y1}, {x2, y2}, {x3, y3}}
RotateLeft[{{x1, y1}, {x2, y2}, {x3, y3}}, {0, 1}]
```

Out[276]=
$$\begin{pmatrix} x1 & y1 \\ x2 & y2 \\ x3 & y3 \end{pmatrix}$$

Out[277]=
$$\begin{pmatrix} y1 & x1 \\ y2 & x2 \\ y3 & x3 \end{pmatrix}$$

In[278]:=
```
ListPlot[invlcumulgauss, Filling → Axis]
```

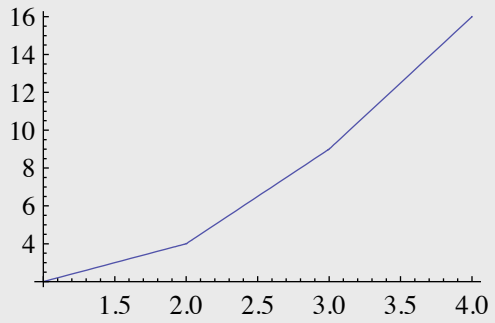
Out[278]=


■ Make interpolated function of the inverse cumulative

Interpolation works by fitting polynomial curves to the data. Try the test below with various interpolation orders (the default is 3)

In[279]:=
```
test = Interpolation[{{1, 2.`}, {2, 4}, {3, 9}, {4, 16.`}},
   InterpolationOrder → 1];
Plot[test[x], {x, 1, 4}]
```

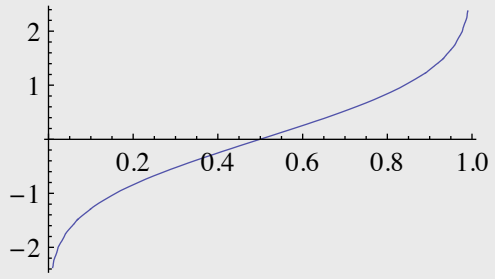
Out[280]=



In[281]:=
```
interinvlcumulgauss = Interpolation[invlcumulgauss];
```

In[282]:=
```
Plot[interinvlcumulgauss[x], {x, 0.01`, 0.99`}]
```

Out[282]=



■ Draw samples with a standard deviation of Sqrt[10]

In[283]:=
```
Round[10 interinvlcumulgauss[RandomReal[]]]
```

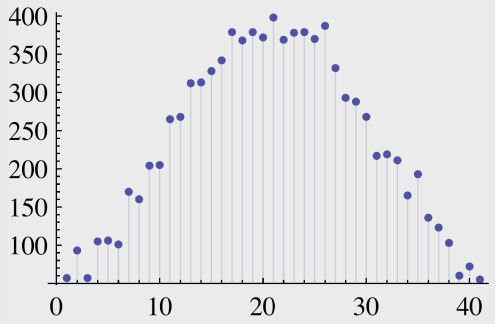
Out[283]=
```
-12
```

■ Draw a bunch of samples, and plot up histogram

In[284]:=
```
z = Table[Round[10 interinvlcumulgauss[RandomReal[]]], {10 000}];
domain = Range[-20, 20];
Freq = (Count[z, #1] &) /@ domain;
```

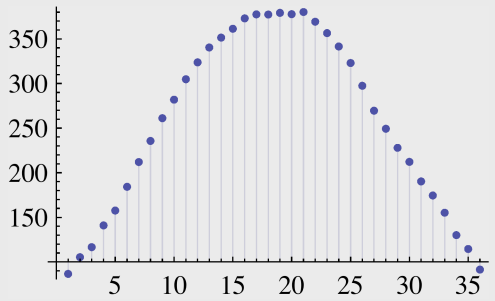
In[287]:=
```
ListPlot[Freq, Filling → Axis]
```

Out[287]=



In[288]:=
```
ListPlot[MovingAverage[Freq, 6], Filling → Axis]
```

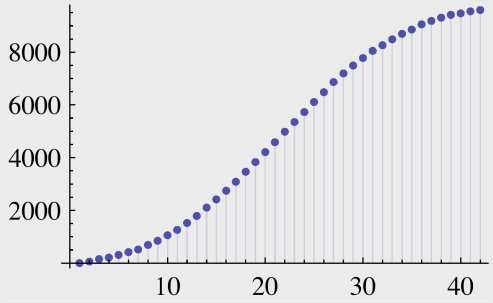
Out[288]=



■ Plot up cumulative histogram

In[289]:=
```
CumFreq = FoldList[Plus, 0, Freq];
ListPlot[CumFreq, Filling → Axis]
```

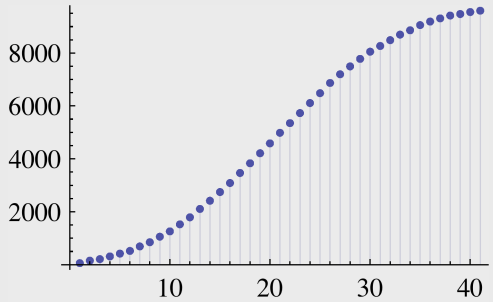
Out[290]=



Same thing, with the new *Mathematica* 6 function, Accumulate[]:

In[291]:=
```
CumFreq = Accumulate[Freq];
ListPlot[CumFreq, Filling → Axis]
```

Out[292]=



Multivariate (vector) gaussian 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:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\,\mathrm{Det}[\Sigma]^{1/2}}\,\mathrm{Exp}\left[-\frac{1}{2}(\mathbf{x}-\mu)^{T}\,\Sigma^{-1}\,(\mathbf{x}-\mu)\right] \qquad (3)$$

We define a 2-variate density:

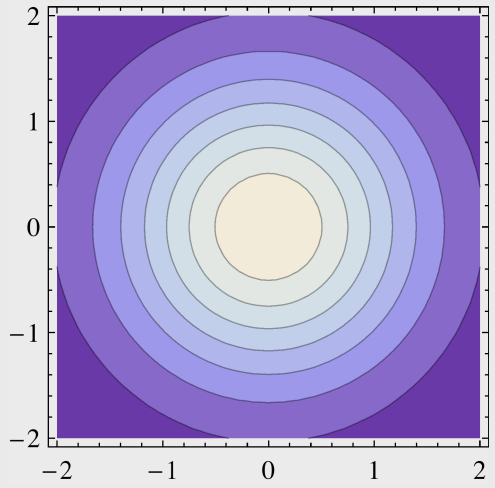
In[293]:=
```
multigaus[x_,m_,cov_]:=
    Module[{IC,detCov,norm,p},
    IC = Inverse[cov];
    detCov = Abs[Det[cov]];
    norm = N[Sqrt[(2Pi)^2 detCov]];
    p = Exp[-0.5 (x-m).IC.(x-m)]/norm;
    Return[p];
];
```

■ Two variable examples

■ Zero mean, zero correlation

In[294]:=
```
m1 = {0, 0}; Cov = {{1, 0}, {0, 1}};
ContourPlot[multigaus[{x1, x2}, m1, Cov], {x1, -2, 2}, {x2, -2, 2}]
```

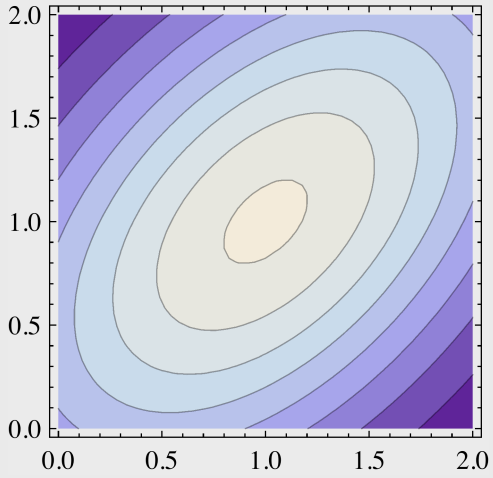
Out[294]=



■ Mean = {1,1}, positive correlation

In[295]:=
```
m1 = {1, 1}; Cov = {{1, 0.5`}, {0.5`, 1}};
ContourPlot[multigaus[{x1, x2}, m1, Cov], {x1, 0, 2}, {x2, 0, 2}]
```

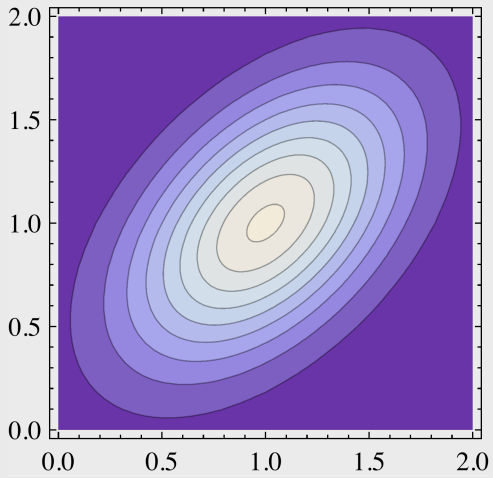
Out[295]=



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

In[296]:=
```
m1 = {1, 1}; Cov = 0.2 {{1, 0.5}, {0.5, 1}};
ContourPlot[multigaus[{x1, x2}, m1, Cov], {x1, 0, 2}, {x2, 0, 2}]
```

Out[296]=



■ Mixture of gaussians

α is a mixing parameter

$$p(x) = \alpha p_1(x) + (1-\alpha)\, p_2(x) \quad \text{where } 0 \le \alpha \le 1 \tag{4}$$

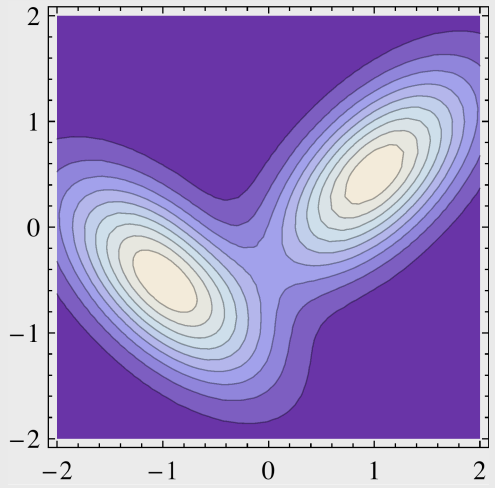
α can be interpreted in terms of a prior probability of choosing which of two distributions a sample will be drawn from.

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).

In[297]:=
```
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]);
```

In[301]:=
```
ContourPlot[mix[{x1, x2}], {x1, -2, 2}, {x2, -2, 2}]
```

Out[301]=



■ 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)

In[302]:=
```
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.

In[309]:=
```
Dimensions[hat]
```

Out[309]= {8760, 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:

In[310]:=
```
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[]:

In[312]:=
```
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:

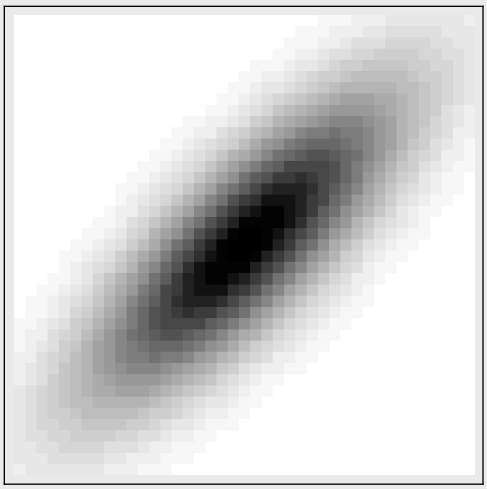
In[313]:=
```
Freq = Map[Count[hat,#]&,domain];
```

In[314]:=
```
width = Sqrt[Dimensions[Freq]]
```

Out[314]=
```
{41}
```

In[315]:=
```
ArrayPlot[Partition[Freq, width], ImageSize → Small, DataReversed → True]
```

Out[315]=



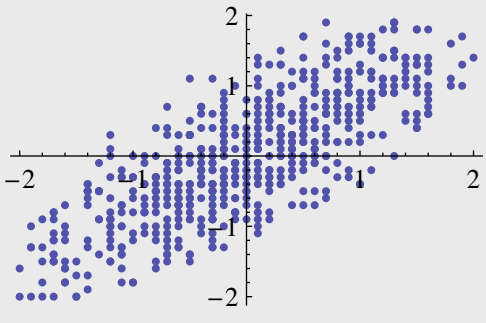
■ Now draw a sample--simulate pulling a slip from hat

In[316]:=
```
rv := hat[[RandomInteger[{1, Length[hat]}]]];
```

In[317]:=
```
test = Table[hat[[RandomInteger[{1, Length[hat]}]]], {600}];
```

In[318]:=
```
g1 = ListPlot[test]
```

Out[318]=



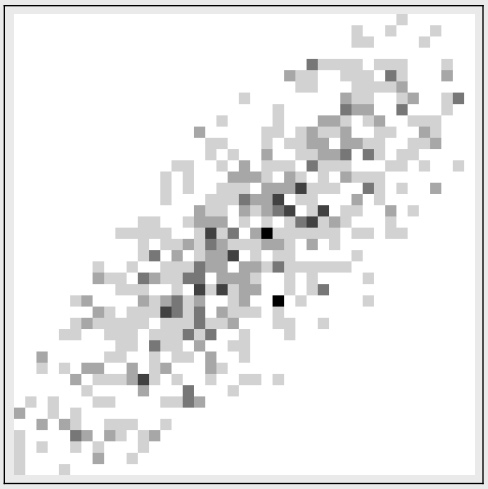
Of course, we can't see the frequency of draws in this plot, so let's count up the number of occurences per bin, and plot up

the results as we did above.

In[319]:=
```
Freq2 = Map[Count[test,#]&,domain];
width = Sqrt[Dimensions[Freq2]];
```

In[321]:=
```
ArrayPlot[Partition[Freq2, width], ImageSize → Small, DataReversed → True]
```

Out[321]=



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

---

Add-on *Mathematica* functions for gaussian multivariates & exploring marginals

■ Define PDF, CDF

In[322]:=
```
Needs["MultivariateStatistics`"]
```

The add – on function,
`MultinormalDistribution[` *mu*, *sigma* `]` specifies *a* multinormal (multivariate Gaussian) distribution with mea

In[323]:=
```
m1 = {1,.5};
r=0.4*{{1,.6},{.6,4}};
ndist = MultinormalDistribution[m1, r];
```

In[326]:=
```
pdf = PDF[ndist, {x1, x2}]
```

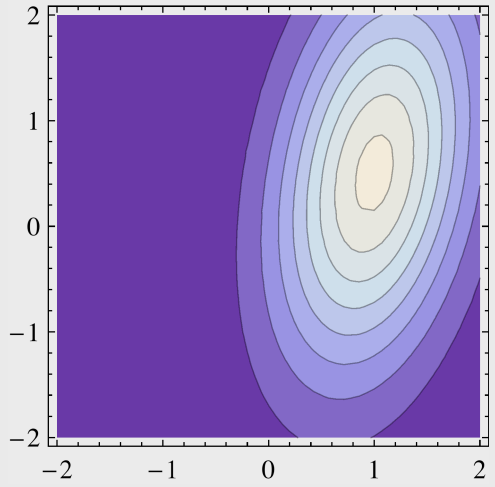
Out[326]=
```
0.0339281
```

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

In[327]:=
```
CDF[ndist, {-2, 1}]
```

Out[327]=
$$1.02471 \times 10^{-6}$$

In[328]:=
```
g1 = ContourPlot[PDF[ndist, {x1, x2}], {x1, -2, 2}, {x2, -2, 2}]
```

Out[328]=


In[329]:=
```
Clear[x1, x2];
marginal[x1_] := ∫_-∞^∞ PDF[ndist, {x1, x2}] dx2;
marginal2[x2_] := ∫_-∞^∞ PDF[ndist, {x1, x2}] dx1;
```

In[332]:=
```
mt = Table[{x1, marginal[x1]}, {x1, -2, 2, .2}];

g2 = ListPlot[mt, Joined → True, PlotStyle → {Red, Thick}];
```

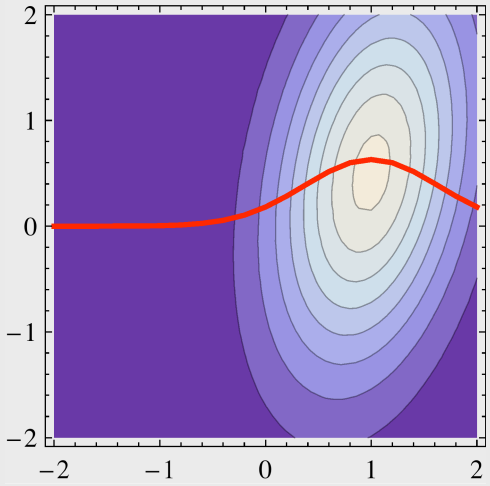
In[334]:=
```
marginal2[-1]
```

Out[334]=
0.15613

In[335]:=
```
mt2 = Table[{x2, marginal2[x2]}, {x2, -2, 2, .2}];
g3 = ListPlot[mt2, Joined → True, PlotStyle → {Green, Thick}];
```

In[337]:=
```
Show[{g1, g2}]
```

Out[337]=



■ Drawing samples

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

In[338]:=
```
RandomReal[ndist]
```

Out[338]= {0.653079, −0.226794}

■ Mixtures of gaussians revisited with MultinormalDistribution[]

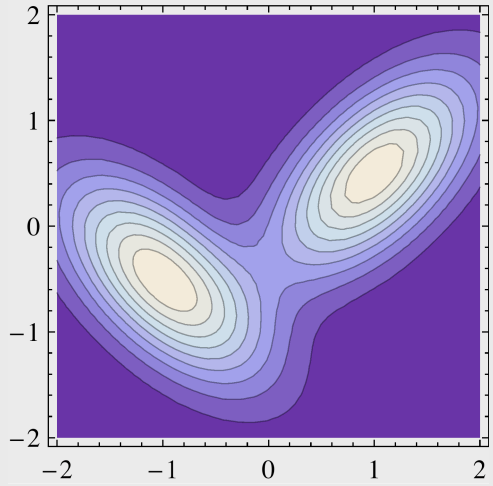
In[339]:=
```
Clear[mix];
```

In[340]:=
```
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];
```

In[345]:=
```
mix[x_] := 0.5 (PDF[ndist1, x] + PDF[ndist2, x]);
```

In[346]:=
```
ContourPlot[mix[{x1, x2}], {x1, -2, 2}, {x2, -2, 2}]
```

Out[346]=



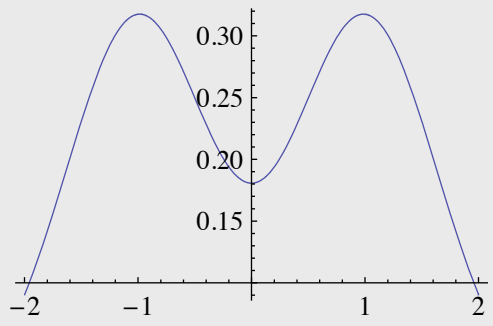
■ Marginals for mixture

```
marginal[x1_] := Integrate[mix[{x1, x2}], {x2, -Infinity, Infinity}]        (5)
```

In[347]:=
```
Clear[marginal];
marginal[x1_] :=
  0.5 * (NIntegrate[PDF[ndist1, {x1, x2}], {x2, -Infinity, Infinity}] +
     NIntegrate[PDF[ndist2, {x1, x2}], {x2, -Infinity, Infinity}]);
```

In[349]:=
```
Plot[marginal[x1], {x1, -2, 2}]
```

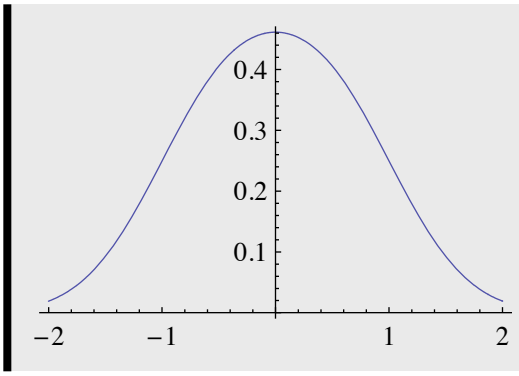
Out[349]=



In[350]:=
```
Clear[marginal];
marginal[x2_] :=
  0.5 * (NIntegrate[PDF[ndist1, {x1, x2}], {x1, -Infinity, Infinity}] +
     NIntegrate[PDF[ndist2, {x1, x2}], {x1, -Infinity, Infinity}]);
```

In[352]:=
```
Plot[marginal[x2], {x2, -2, 2}]
```

Out[352]=



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

Exploratory projection pursuit

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).

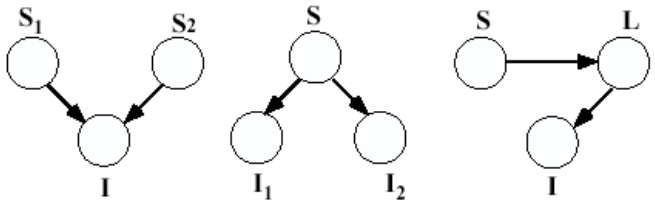
■ Inference: Learning parameters of mixture distributions

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

Graphical Models of dependence

■ Graphs: causal structure and conditional independence

In general, natural image pattern formation is specified by a high-dimensional joint probability, requiring an elaboration of the causal structure that is more complex than the simple SDT model. 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 (e.g. scene) variables causing a given image measurement, a single variable producing multiple image measurements, or a cause indirectly influencing an image 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 image 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 measurement, 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:

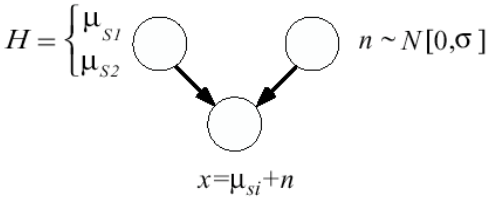
p(S1,S2,I)=p(I|S1,S2)p(S1)p(S2)

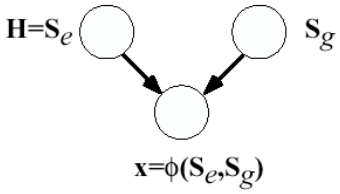
p(S,I1,I2)=p(I1|S)p(I2|S)p(S)

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

■ Primary, secondary variables.

The following figure draws a parallel between the causal structure, as determined by the generative model, for signal detection theory (as in the light detection problem), and the general problem of visual inference.





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

The top panel shows one possible generative graph structure for an ideal observer problem in classical signal detection theory (SDT). The data are determined by the signal hypotheses plus (additive gaussian) noise. Knowledge is represented by the joint probability p(x,H,n)=p(x|H,n)p(H)p(n). The lower panel shows a simplified example of the generative structure for perceptual inference from a pattern inference theory perspective. The image measurements (x) are determined by a typically non-linear function (\phi) 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). Both scene and image variables can be high dimensional vectors. In general, the causal structure of natural image patterns is more complex and consequently requires elaboration

of its graphical representation. For SDT and 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): $\int_{S\_g} p(x, S\_e, S\_g)\, d S\_g$.

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 in SDT and in pattern inference theory

Noise is whatever you don't care to estimate, but contributes to the data.

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.

Golden, R. (1988). A unified framework for connectionist systems. Biological Cybernetics, 59, 109-120.

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)

Kersten, D., Mamassian P & Yuille A (in press) Object perception as Bayesian inference. Annual Review of Psychology. (pdf, http://arjournals.annualreviews.org/doi/pdf/10.1146/annurev.psych.55.090902.142005)

Kersten, D., & Yuille, A. (2003). Bayesian models of object perception. Current Opinion in Neurobiology, 13(2) http://gandalf.psych.umn.edu/~kersten/kersten-lab/papers/KerstenYuilleApr2003.pdf

Knill, D. C., & Richards, W. (1996). *Perception as Bayesian Inference*. Cambridge: Cambridge University Press.

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

Van Trees, H. L. (1968). Detection, Estimation and Modulation Theory . New York: John Wiley and Sons.

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