

# Introduction to Neural Networks

## Probability and neural networks

### Initialize standard library files:

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

---

## Introduction

### Last time

From energy to probability. From Hopfield net to Boltzmann machine. We showed how the Hopfield network, which minimized an energy function, could be viewed as finding states that increase probability.

### Neural network computations from the point of view of statistical inference

By treating neural network learning and dynamics in terms of probability computations, we can begin to see how a common set of tools and concepts can be applied to:

1. **Inference**: as a process which makes the best guesses given data. E.g. find  $H$  such that  $p(H | \text{data})$  is biggest.
2. **Learning**: a process that discovers the parameters of a probability distribution. E.g. find weights such that  $p(\text{weights} | \text{data})$  is biggest
3. **Generative modeling**: a process that generates data from a probability distribution. E.g. draw samples from  $p(\text{data} | H)$

### Today

Boltzmann machine: review  
Probability and statistics overview  
Introduction to drawing samples

---

## Boltzmann Machine: review

### Inference

The network is has the same connectivity structure as the Hopfield net. And like the first Hopfield net-

work, the units are either on or off. But updating is different. A stochastic update rule improves the chances of a network evolving to a global minimum.

Suppose temperature  $T$  is fixed at some value, say  $T=1$ . Then we could update the network and let it settle to thermal equilibrium, a state characterized by some statistical stability, but with occasional jiggles. Let  $V_\alpha$  represent the vector of neural activities, i.e. a pattern of 1's and 0's. The probability of a particular state  $\alpha$  is given by:

$$p(V_\alpha) = \kappa e^{-\frac{E_\alpha}{T}}$$

$$\kappa = \frac{1}{\sum_{\text{all } \alpha} e^{-\frac{E_\alpha}{T}}}$$

(Recall that the second equation is the normalization constant that ensures that the total probability i.e. over all states, is 1.)

The update rule thresholds a weighted sum of inputs, set to 0 or 1 -- i.e. a classic TLU. Nodes (neurons) are updated asynchronously. Temperature  $T$  starts high and is gradually lowered, with the effect that updates that jump up the energy landscape become increasingly rarer.

## Learning

Now let's see how learning weights can be formulated as a statistical problem. We divide up the units into two classes: hidden and visible units. Values of the visible units are determined by the environment. If the visible units are divided up into "stimulus" and "response" units, then the network learns associations (supervised learning). If they are just stimulus units, then the network observes and organizes its interpretation of the stimuli that arrive (unsupervised learning).

Our goal is to have the hidden units discover the structure of the environment. Once learned, if the network was left to run freely using stochastic sampling, the visible units would take on values that reflect the probabilistic structure of the environment they learned. In other words, the network has a generative model of the visible structure.

Consider two probabilities over the visible units,  $V$ :

$P(V)$  - probability of visible units taking on certain values determined by the environment.

$P'(V)$  - probability that the visible units take on certain values while the network is running at thermal equilibrium.

If the hidden units have actually "discovered" the structure of the environment, then the probability  $P$  should match  $P'$ . How can one achieve this goal? Recall that for the Widrow-Hoff and error backpropagation rules, we started from the constraint that the network should minimize the error between the network's prediction of the output, and the actual target values supplied during training. We need some measure of the discrepancy between the desired and target states for the Boltzmann machine. The idea is to construct a measure of how far away two probability distributions are from each other--how far  $P$  is from  $P'$ . One such function is the Kullback-Leibler (KL) measure or relative entropy.

$$G(T_{12}, T_{13}, \dots, T_{ij}, \dots) = \sum_{\substack{\text{all states} \\ \text{over visible units}}} P(V_\alpha) \log \left( \frac{P(V_\alpha)}{P'(V_\alpha)} \right)$$

Then we need a rule to adjust the weights so as to bring  $P' \rightarrow P$  in the sense of reducing the KL measure  $G$ . Ackley et al. derived the following rule for updating the weights so as to bring the probabilities closer together. Make weight changes  $\Delta T_{ij}$  such that:

$$\Delta T_{ij} = \varepsilon (p_{ij} - p'_{ij})$$

where  $p_{ij}$  is the probability of two units,  $V[i]$  and  $V[j]$ , both being 1 when environment is clamping the states at thermal equilibrium averaged over many samples.  $p'_{ij}$  is the probability of  $V[i]$  and  $V[j]$  being 1 when the network is running freely without the environment and sufficiently long to be at equilibrium.

### Pros and cons

In theory, the Boltzmann machine is a very general machine for inference and learning. In practice, it doesn't scale up well to realistic high-dimensional problems, such as image processing and recognition. The **Restricted Boltzmann machine** is a special case that helps with scalability. However, in contrast to backpropagation, the learning rule has biological plausibility, in that updates only depend on the connections between pairs of neurons.

## Probability and statistics overview

*To understand modern neural network theory, we need to understand probabilistic modeling. Why?*

### Basic rules of probability

**Joint probabilities.** 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  $p(A=a,B=b, C=c, D=d,E=e)$ , for any particular instances, a, b, c, d, e.

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

Probability about an event changes when new information is gained.

$$\text{Prob}(X \text{ given } Y) = p(X|Y)$$

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

$$p(X, Y) = p(X | Y) p(Y)$$

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

### Independence

Knowledge of one event doesn't change the probability of another event. This can be expressed in terms of conditional probability as:

$$p(X) = p(X|Y) \text{ which by the product rule is:}$$

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

► 1. Demo using Mathematica.

Consider the joint probability of two events: ( $x < 3$ ) and ( $x > 1$ ), corresponding to  $X$  and  $Y$  above. What is the probability of the first, given that the second is observed (or fixed in advanced)?

```
In[491]:= NProbability[(x < 3) && (x > 1), x ≈ PoissonDistribution[5]] /
  NProbability[(x > 1), x ≈ PoissonDistribution[5]]
```

```
Out[491]= 0.0877728
```

which can also be calculated using Mathematica's expression, `|]`, for conditioning:

```
In[492]:= NProbability[(x < 3) |] (x > 1), x ≈ PoissonDistribution[5]]
```

```
Out[492]= 0.0877728
```

Try replacing the poisson distribution with:  $x \sim \text{NormalDistribution}[0, 1]$

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

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

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

The distributions on the left are called *marginal distributions*. They are obtained by “*summing over*” or “*integrating out*” the variables we are not interested in.

- 2. Can you think of examples in perception of variables that the system isn't interested in? What is “noise”?

## Rule 3: Bayes' rule

Bayes rule allows us to invert probabilities. For example, as the Rev. Bayes himself contemplated, one can calculate the probability that a ball randomly thrown on a billiard table will land on a line drawn say, 1/3 of the way across. The probability is proportional to the ratio of the area on that side to the whole area. But what if you didn't know where the line was, and you were given data as to which side the balls were landing. What is the probability of the line being at some location, e.g. 1/3 of the way across?

Another example is coin flipping. If you knew that a coin was biased to come up with heads with a probability of  $p = 2/3$ rd, you could predict the probability distribution of heads and tails (2/3, 1/3). But what if you didn't know the coin's  $p$ , but had data, say one flip, say  $Y = \{\text{“heads”}\}$ , what is the probability of the coin's bias (i.e. the probability of the probability  $p$ )? We'll look at the coin flipping problem from a Bayesian perspective later.

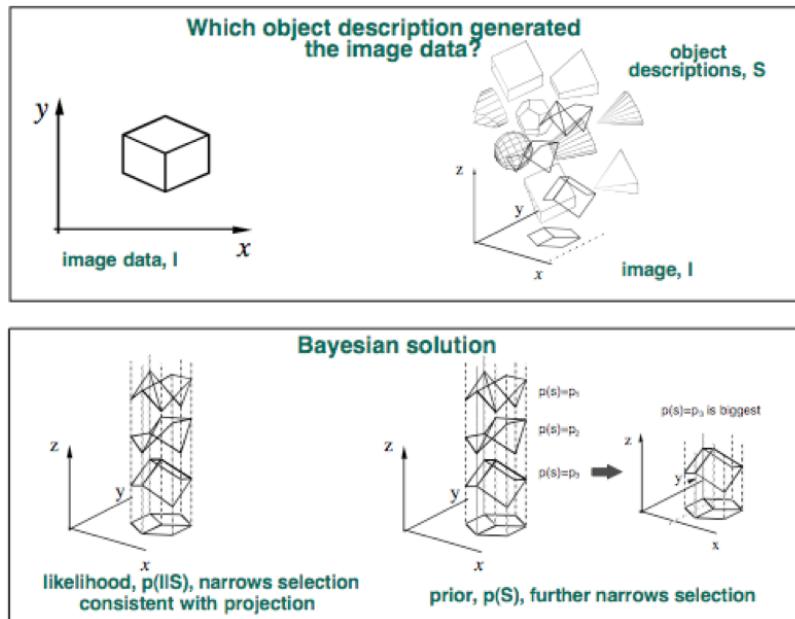
From the product rule, and since  $p[X, Y] = p[Y, X]$ , we have:

$$p(Y | X = \text{data}) = \frac{p(X | Y) p(Y)}{p(X)} \text{ proportional to } p(X | Y) p(Y)$$

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

## Developing intuitions: Bayes inference applied to visual perception

Consider the problem of interpreting the line drawing shown in the upper left part of the figure below.



What is the “data”? What is the perceptual hypothesis? What are the probabilistic relationships between them? If we can answer these questions, we can first pose problems of perception as problems of perceptual inference. Then perhaps later we can investigate the possibility that a neural network is solving the perceptual inference problem.

$$p[\text{hypothesis} \mid \text{data}] = \frac{p[\text{data} \mid \text{hypothesis}] p[\text{hypothesis}]}{p[\text{data}]}$$

$$p[S \mid I] = \frac{p[I \mid S] p[S]}{p[I]}$$

Usually, we will be thinking of the “**hypothesis**” term as a random variable over the hypothesis space, and “**data**” comes from some measurements. So for visual inference, **S = hypothesis** (the scene), and **I = data** (the image data), and **I = f(S)**. In perceptual inference, optimal decisions are based on:

$$p(S|I),$$

the **posterior** probability of the scene given the image, i.e. what you get when you condition the joint by the image data. The posterior is often what we'd like to base our decisions on, because as we discuss below, picking the hypothesis **S** which maximizes the posterior (i.e. maximum a posteriori or **MAP** estimation) minimizes the average probability of error.

**p(S)** is the **prior** probability of the scene.

**p(I|S)** is the **likelihood** of the scene. Note this is a probability over values of **I**, but not of **S**. So  $\int p(I|S) dI = 1$ .

A MAP decision based on  $p(\mathbf{S}|\mathbf{I})$  does not necessarily involve explicit computations of the likelihood and prior as in Bayes rule. Such inferential processes are sometimes called *discriminative*.

*Generative inferences* rely on explicit knowledge of how the data might have been generated, as represented by the likelihood  $p(\mathbf{I} | \mathbf{S})$ , perhaps together with a model of the prior,  $p(\mathbf{S})$ .

In vision,  $\mathbf{S}$  is often a high-dimensional description of scene parameters, and  $\mathbf{I}$  is a vector of image measurements.

One of the challenges of visual inference is that the data by itself is insufficient to decide the value(s) of the hypothesis. In other words, the image measurements  $\mathbf{I}$  don't uniquely determine the scene description  $\mathbf{S}$ . A visual system must make a good guess. To do this well, it needs to take into account all the ways the image might have been generated--i.e. the causes of image measurements, together with a model of the prior probabilities of those causes.

We will return to examples of Bayesian inference later. But first, let's go over some basic definitions in statistics.

## Statistics

The definitions and properties of statistics are important to know about in the context of what it means to have patterns of data, such as an image. Image data can come in many forms, such as the raw intensity values in a picture, or some simple measurement like how far a pixel intensity is from its neighbors. Often one needs to summarize some quantity in a complex pattern, like an image. These summaries take the form of statistics. Statistics like the mean (or expectation) and variance can be sufficient to characterize a whole distribution, such as a gaussian. This assumption is used in parametric statistics. But other times there is no simple distribution to characterize the underlying probabilistic generative process. Then we might represent the distribution in terms of frequencies with respect to some relevant measurements. A histogram of image intensities is an example--i.e. how frequently do graylevels 0, 1, 2, ...255 occur in the image?

### Expectation & variance

Analogous to center of mass (the "balancing" point):

*Definition of expectation or average:*

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

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

Some rules:

$$\begin{aligned} E[X+Y] &= E[X] + E[Y] \\ E[aX] &= a E[X] \\ E[X+a] &= a + E[X] \end{aligned}$$

*Definition of variance:*

$$\sigma^2 = \text{Var}[X] = E[(X-\mu)^2] = \sum_{j=1}^N p(x_j) (x_j - \mu)^2 \sim \sum_{i=1}^N (x_i - \mu)^2 / N$$

$$\text{Var}[X] = \int (x - \mu)^2 p(x) dx$$

Standard deviation:

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

Some rules:

$$\text{Var}[X] = E[X^2] - E[X]^2$$

$$\text{Var}[aX] = a^2 \text{Var}[X]$$

## Covariance & Correlation

Covariance:

$$\text{Cov}[X, Y] = E[(X - \mu_X)(Y - \mu_Y)]$$

Correlation coefficient:

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

## Cross and Autocovariance matrix

Suppose X and Y are vectors:  $\{X_1, X_2, \dots\}$  and  $\{Y_1, Y_2, \dots\}$

$$\text{Cov}[X_i, Y_j] = E[(X_i - \mu_{X_i})(Y_j - \mu_{Y_j})] \sim \sum_{n=1}^N (x_i^n - \mu_{X_i})(y_j^n - \mu_{Y_j})^T / N$$

$$\text{Autocov}[X_i, X_j] = E[(X_i - \mu_{X_i})(X_j - \mu_{X_j})] \sim \sum_{n=1}^N (x_i^n - \mu_{X_i})(x_j^n - \mu_{X_j})^T / N$$

In matrix form, the autocovariance can be written:

$$\Sigma = \text{cov}[X] = E[(X - E[X])(X - E[X])^T]$$

In other words, the covariance matrix can be approximated by the average outer product, which apart from normalization, is exactly how the Hebbian memory matrix was constructed in our first model of unsupervised learning and memory.

## Independent random variables

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

$$E[XY] = E[X]E[Y] \text{ (uncorrelated)}$$

$$\text{Cov}[X, Y] = \rho[X, Y] = 0$$

$$\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y]$$

If two random variables are uncorrelated ( $\rho[X, Y] = 0$ ), they are *not* necessarily independent.

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

Seeking rotations of the coordinates of high-dimensional data for which the data projections are uncorrelated is the basis of Principal Components Analysis (PCA). Seeking dimensions for which the projections are independent is Independent Component Analysis (ICA). These methods are examples of unsupervised learning which we will discuss later in the course.

## Degree of belief vs., relative frequency

What is the probability that it will rain tomorrow? Assigning a number between 0 and 1 is assigning a degree of belief. These probabilities are also called subjective probabilities. It is nonsense to think of “repeating” tomorrow to see how frequently it rains.

What is the probability that a coin will come up heads? In this case, we can imagine actually doing an experiment to find out. Flip the coin  $n$  times, and count the number of heads, say  $h[n]$ , and then set the probability,  $p \approx h[n]/n$  -- the relative frequency (which is a statistic). Of course, if we did it again, we may not get the same estimate of  $p$ . One solution often given is:

$$p = \lim_{n \rightarrow \infty} \frac{h(n)}{n}, \text{ where } h(n) = \# \text{ heads after } n \text{ flips.}$$

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

In some domains we may have enough data that 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. Further, we may want to model “beliefs” as probabilities, even if they can’t be directly measured.

Suppose you wanted to estimate the joint probabilities of 6 letter combinations from a database of 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, then probabilities can become more like "subjective unconscious beliefs". From a modeling perspective, one can treat a specific prior as an assumed ingredient, and test to see how well the model accounts for the data, and how well it predicts new data. If it the model does a poor job empirically, then one can go back and question whether an alternative prior could improve the predictions.

## Principle of insufficient reason

### Principle of symmetry

Suppose we have  $N$  events,  $x[1], x[2], x[3], \dots, x[N]$  that are all physically identical except for the label. In the absence of any other information about these events, it seems reasonable to assume that

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

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

What about the continuous case where there is no reason to assume any particular value at all between  $-\infty$  and  $+\infty$ ? If one assumes the prior density is constant, then it has to approach zero everywhere to



make sure the sum between  $-\infty$  and  $+\infty$  is 1

...mmm...a constant value is often assumed anyway, and this is called an improper prior.

## Information theory and Maximum entropy

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

$$\text{Information}[X] = -\log_2(p(X)), \text{ in bits.}$$

The more improbable an event, the more “surprising” it is in some sense, 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 for our above case,  $H(X)$  is biggest for the uniform distribution,  $p(X)=1/N$ . Maximum entropy is looking like the symmetry principle.

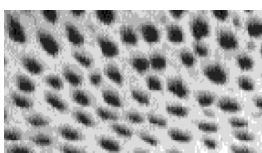
(Note: information and entropy can also be defined using  $\ln = \log_e$ , in which case the unit is nats, rather than bits.)

Indeed it turns out that a useful generalization of the principle of symmetry is maximum entropy. For example, out of all possible probability distributions of a random variable with infinite range from negative to positive real numbers, 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 distribution. (Cover and Thomas). (This is the same calculation that physicists use to predict the density of air as a function of altitude above the earth’s surface -- there is an exponential drop-off with height.)

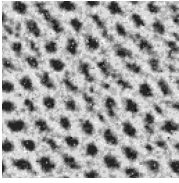
## Application to the perception of textures

An interesting application of the maximum entropy principle is to learning image textures joint probabilities:  $p(I[1], \dots, I[N])$ , 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 dimensionality of the probability space--i.e. there are many different probability distributions that give the same statistics. So the principle of symmetry, (also called insufficient reason, principle of indifference), 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 briefly described in a previous lecture (Zhu et al.). Here is an observed sample:



Here is a synthesized sample after Minimax entropy learning:



In general, it is a hard problem to draw true samples from high dimensional spaces. One needs a quantitative model of the distribution and a method such as *Gibbs sampling* to draw samples.

The previous lecture mentioned an example of an application of texture synthesis to understanding the networks of the brain's visual system, see Freeman et al. (2011), Freeman et al. (2013). (See too: McDermott, J. H., & Simoncelli, 2011, for sound textures.)

## Mathematica functions for multivariate distributions, computing marginals

Later we'll tackle the problem of drawing samples from high-dimensional, but structured distributions. But here we'll use *Mathematica* built-in functions to graphically illustrate some of the above ideas, and help build intuitions.

### Multivariate gaussian probability density

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

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

*Mathematica* has a built-in function, **MultinormalDistribution** $[\mu, \Sigma]$ .

For example, you can define the probability density function for mean vector  $\{\mu_1, \mu_2\}$ , and covariance matrix  $\{\{\sigma_{11}^2, \rho * \sigma_{11} * \sigma_{22}\}, \{\rho * \sigma_{11} * \sigma_{22}, \sigma_{22}^2\}\}$ , where  $\rho$  parameterizes correlation.

```
In[451]:=  $\Sigma = \{\{\sigma_{11}^2, \rho * \sigma_{11} * \sigma_{22}\}, \{\rho * \sigma_{11} * \sigma_{22}, \sigma_{22}^2\}\};$ 
```

```
PDF[MultinormalDistribution[{\mu1, \mu2}, \Sigma], {x, y}]
```

```
Out[452]=  $e^{\frac{1}{2} \left( -\frac{(x-\mu_1)(y\rho\sigma_{11}-\rho\mu_2\sigma_{11}-x\sigma_{22}+\mu_1\sigma_{22})}{(-1+\rho^2)\sigma_{11}^2\sigma_{22}} - \frac{(y-\mu_2)(-y\sigma_{11}+\mu_2\sigma_{11}+x\rho\sigma_{22}-\rho\mu_1\sigma_{22})}{(-1+\rho^2)\sigma_{11}\sigma_{22}^2} \right)} / \left( 2\pi \sqrt{\sigma_{11}^2\sigma_{22}^2 - \rho^2\sigma_{11}^2\sigma_{22}^2} \right)$ 
```

```
In[453]:=  $\Sigma // \text{MatrixForm}$ 
```

```
Out[453]//MatrixForm=
```

$$\begin{pmatrix} \sigma_{11}^2 & \rho \sigma_{11} \sigma_{22} \\ \rho \sigma_{11} \sigma_{22} & \sigma_{22}^2 \end{pmatrix}$$

You can ask for the Covariance to be returned:

```
In[454]:= Covariance[MultinormalDistribution[{μ1, μ2}, Σ]] // MatrixForm
Out[454]/MatrixForm=
```

$$\begin{pmatrix} \sigma_{11}^2 & \rho \sigma_{11} \sigma_{22} \\ \rho \sigma_{11} \sigma_{22} & \sigma_{22}^2 \end{pmatrix}$$

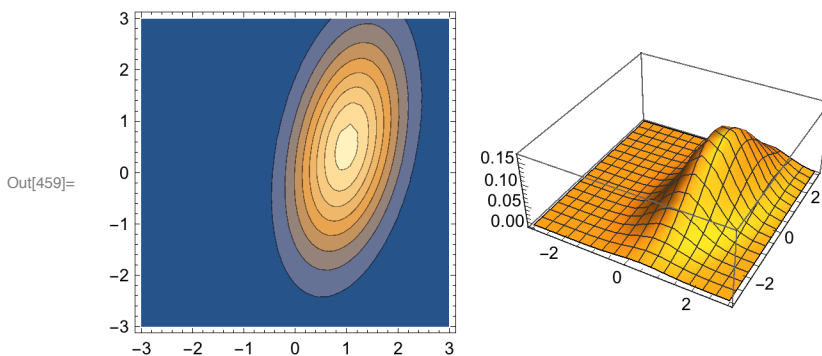
## Examples of PDF, CDF

### A specific example, MultinormalDistribution[μ, Σ]

```
In[455]:= m1 = {1, 1/2};
r = (1/2) * {{1, 2/3}, {2/3, 4}};
ndist = MultinormalDistribution[m1, r];
pdf = PDF[ndist, {x1, x2}]
Out[458]=
```

$$\frac{1}{4 \sqrt{2} \pi} 3 e^{\frac{1}{2} \left( -(-1+x1) \left( \frac{9}{4} (-1+x1) - \frac{3}{8} \left( -\frac{1}{2}+x2 \right) \right) - \left( -\frac{3}{8} (-1+x1) + \frac{9}{16} \left( -\frac{1}{2}+x2 \right) \right) \left( -\frac{1}{2}+x2 \right) \right)}$$

```
In[459]:= GraphicsRow[{g1 = ContourPlot[PDF[ndist, {x1, x2}], {x1, -3, 3}, {x2, -3, 3}],
g13D = Plot3D[PDF[ndist, {x1, x2}], {x1, -3, 3}, {x2, -3, 3}]]]
Out[459]=
```



### Calculating probabilities using the CDF

What is the probability of  $x_1$  and  $x_2$  taking on values in the region  $x_1 < .5 \cap x_2 < 2$ ?

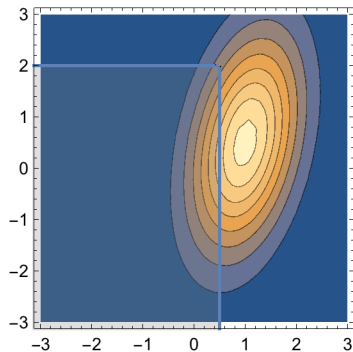
Recall that the cumulative distribution function is given by:

$$\text{CDF}(x_1, x_2) = \int_{-\infty}^{x_2} \int_{-\infty}^{x_1} p(x_1, x_2) dx_1 dx_2$$

So the answer is the area under the PDF shown below.

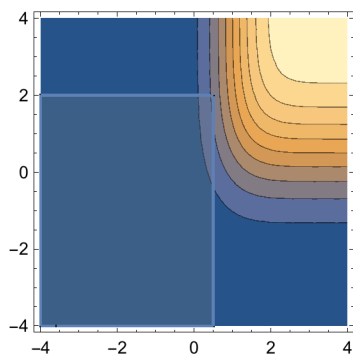
```
grp = RegionPlot[x1 < .5 && x2 < 2, {x1, -4, 4}, {x2, -4, 4},
  PlotStyle → Directive[Opacity[.25], EdgeForm[], FaceForm[Gray]]];
```

```
Show[{ g1, grp}, ImageSize → Small]
```



We could numerically integrate to find the area. Alternatively, the answer can be found from the built-in cumulative distribution function, or CDF. The value corresponds to the height of the contour at  $\{x_1, x_2\} = \{.5, 2.0\}$ . Move your mouse cursor over the contour plot below near the  $\{.5, 2.0\}$  point.

```
gcdf = ContourPlot[CDF[ndist, {x1, x2}], {x1, -4, 4}, {x2, -4, 4}, ImageSize → Small];
Show[{ gcdf, grp}, ImageSize → Small]
```



Move your mouse over the plot to see the value of the CDF at  $\{x_1, x_2\} = \{.5, 2.0\}$ . A more precise answer is:

```
CDF[ndist, {.5, 2.0}]
```

```
0.225562
```

## Finding the mode

The mode is the value of the random variable with the highest probability or probability density. For discrete distributions, think of it as the most frequent value.

(Sometimes the word “mode” is used to refer to a *local* maximum in a density function. Then the distribution is called multimodal. If it has two modes, it is called bimodal.) There may not be a unique mode--the uniform distribution is an extreme case of this.

For the Gaussian case, the mode vector corresponds to the mean vector. But we can pretend we don't know that, and use the **FindMaximum[]** function to find the maximum and the coordinates where the

max occurs:

```
In[460]:= FindMaximum[PDF[ndist, {x1, x2}], {{x1, 0}, {x2, 0}}]
```

```
Out[460]= {0.168809, {x1 → 1., x2 → 0.5}}
```

## Marginals

Suppose we want the distribution for just  $x_1$ ,  $\text{PDF}[x_1] = \text{marginal}[x_1]$ . How do we find it? Integrate  $\text{PDF}[x_1, x_2]$  with respect to  $x_2$ . This is called calculating the marginal distribution of  $x_1$ . It is obtained by integrating out the other variable,  $x_2$ . Similarly, we can calculate  $\text{PDF}[x_2]$ .

```
Clear[x1, x2];
```

```
marginal[x1_] := Integrate[PDF[ndist, {x1, x2}], dx2];
```

```
marginal2[x2_] := Integrate[PDF[ndist, {x1, x2}], dx1];
```

```
ndist
```

```
MultinormalDistribution[{{1, 1/2}, {{1/2, 1/3}, {1/3, 2}}}]
```

What is the mode of  $\text{PDF}[x_2]$ ?

```
FindMaximum[marginal2[zz], {{zz, 0}}]
```

```
{0.282095, {zz → 0.5}}
```

Now let's plot up the marginals on top of the contour plot of the joint distribution.

```
mt = Table[{x1, marginal[x1]}, {x1, -3, 3, .2}];
```

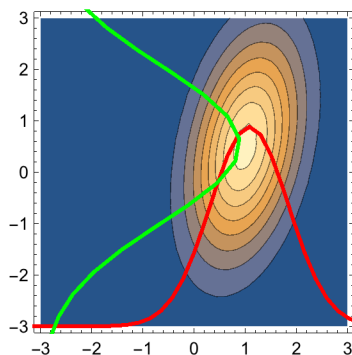
```
g2 = ListPlot[mt, Joined → True, PlotStyle → {Red, Thick}, Axes → False];
```

```
mt2 = Table[{x2, marginal2[x2]}, {x2, -3, 3, .4}];
```

```
g3 = ListPlot[mt2, Joined → True, PlotStyle → {Green, Thick}, Axes → False];
```

```
theta = Pi / 2;
```

```
Show[g1, Epilog → {Inset[g2, {0, -3}, {0, 0}], Inset[g3, {-3, 0}, {0, 0},  
Automatic, {{Cos[theta], Sin[theta]}, {Sin[theta], -Cos[theta]}}}]
```



- ▶ 3. Use the built-in function, `MarginalDistribution[ ]`, to calculate and plot the marginals:

```

marginalM = MarginalDistribution[ndist, 1]
marginal2M = MarginalDistribution[ndist, 2]

NormalDistribution[1,  $\frac{1}{\sqrt{2}}$ ]

NormalDistribution[ $\frac{1}{2}$ ,  $\sqrt{2}$ ]

```

## Drawing samples

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

```

RandomVariate[ndist]
{2.30121, 2.38163}

```

or

```

RandomReal[ndist]
{0.765882, -0.377347}

```

or using the marginal from above,

```
In[461]:= RandomVariate[marginalM]
```

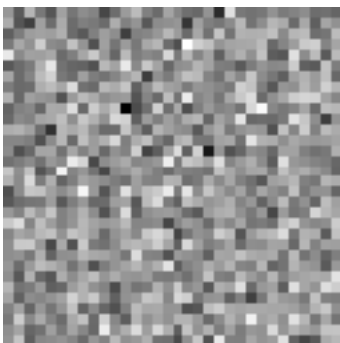
```
Out[461]= 2.73756
```

Here's an example of a "white gaussian noise" image. It is called "white" because there are no correlations between the pixel intensities. Each one is drawn independently of any of the others.

```

width = 32;
data = RandomVariate[NormalDistribution[0, 1], width * width];
Image[Partition[data, width]] // ImageAdjust

```



## Mixtures of gaussians with MultinormalDistribution[ ]

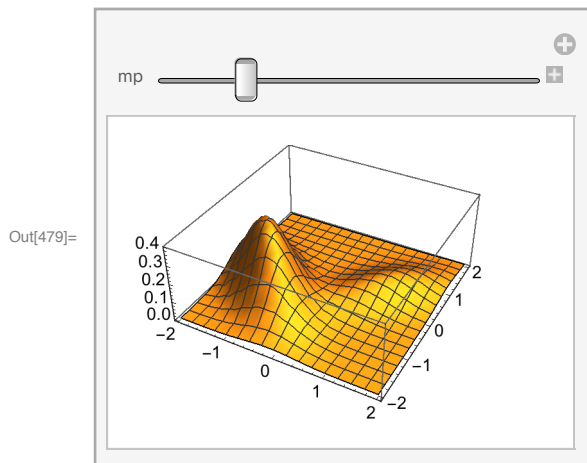
Multivariate gaussian distributions are often inadequate to model real-life problems, that for example might involve more than one mode or might have non-gaussian properties. One solution is to approximate more general distributions by a sum or mixture of gaussians. Mixtures can be used to model clusters of feature values for object categories. The feature values for each category corresponds to a

multivariate gaussian.

```
In[472]:= 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];

In[478]:= mix[x_,mp_] := mp*PDF[ndist1, x] + (1-mp)*PDF[ndist2, x];

In[479]:= Manipulate[gg1 = Plot3D[mix[{x1, x2}, mp], {x1, -2, 2},
  {x2, -2, 2}, PlotRange -> Full, ImageSize -> Small], {{mp, .2}, 0, 1}]
```

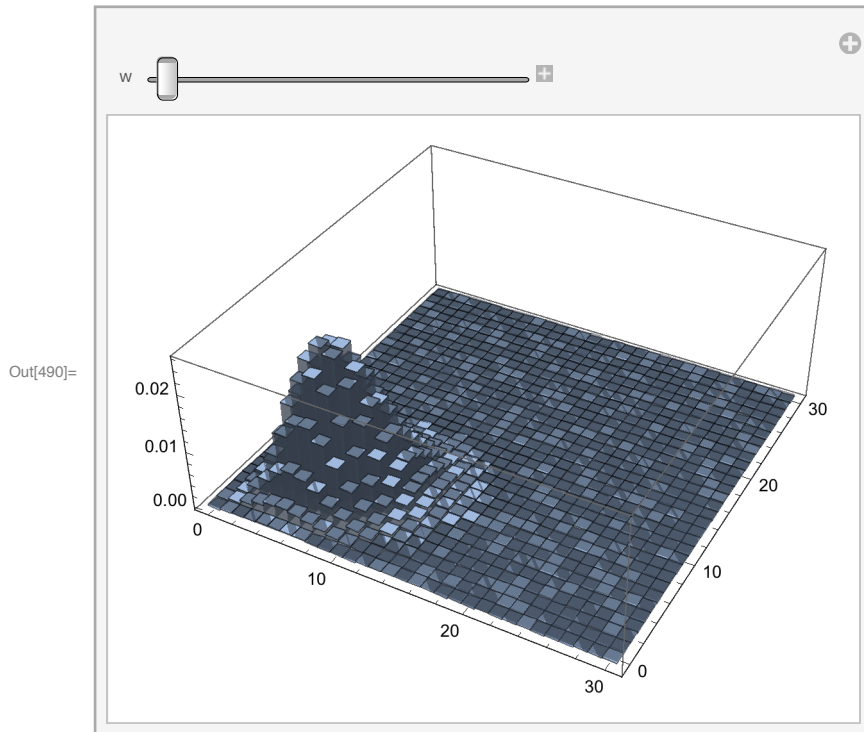


*Mathematica* has a built-in function for defining mixture distributions:

```

In[488]:= Clear[D, w, x, y];
D[w_] =
  MixtureDistribution[{w, 1 - w}, {MultivariatePoissonDistribution[7, {9, 10}],
    MultivariatePoissonDistribution[2, {5, 4}]}];
Manipulate[DiscretePlot3D[PDF[D[w], {x, y}], {x, 0, 30},
  {y, 0, 30}, ExtentSize -> Full], {w, 0, 1}]

```



See Zoran and Weiss (2011) for an interesting application of mixture models to discovering visual features in natural images.

## Sampling in low dimensions

The ability to simulate the process of drawing random samples has become basic to computational modeling of inferential processes. We first go over a few basics to understand how to draw samples from arbitrary univariate distributions. In later lectures, we'll introduce Monte Carlo sampling, and in particular Markov Chain Monte Carlo methods to estimate complex distributions for Bayesian inference.

### Preliminaries: Density mapping theorem

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

#### 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

In this case, the form of probability density function changes because we require the probability "mass" to be unchanged:  $p(x)dx = p(y)dy$

Suppose,  $y=f(x)$

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

Transformation of variables is used in making random number generators for probability densities other than the uniform distribution, such as a Gaussian.

Below we'll need to use the cumulative distribution function:  $CDF(x) = \text{prob}(X < x) = \int_{-\infty}^x p(X) dX$

## Univariate sampling

The easiest way to draw a sample is to use a built-in function:

```
RandomVariate[NormalDistribution[0, 1]]
```

```
1.03062
```

But what if we only have a random number generator that provides uniformly distributed numbers between 0 and 1? How can we get numbers that are Gaussian distributed?

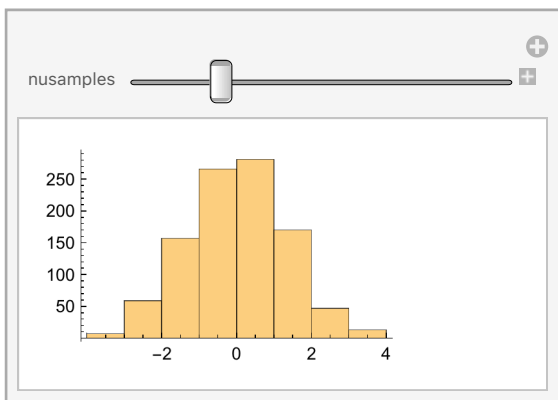
If understand some principles behind generating random numbers for a specified distribution.

## Method 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. The Central Limit Theorem says that the sum of a sufficiently large number of independent random variables drawn from the same underlying distribution (with finite mean and variance), will be approximately normally distributed. The approximation gets better as the number of samples increases.

Try the cell below with  $n_{\text{samples}} = 1, 2, \dots, 10, \dots$

```
Manipulate[
  z1 = Table[
    Sum[RandomInteger[], {i, 1, nusamples}] - 
    nusamples/2, {1000}];
  Histogram[z1, ImageSize -> Small], {nusamples, 1, 30, 1}]
```



Draws that are independently drawn from the same distribution are called i.i.d, for independently and identically distributed.

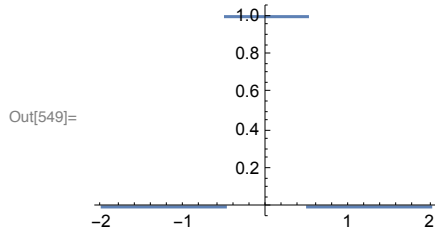
► 4. Convolution theorem for adding random variables.

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

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

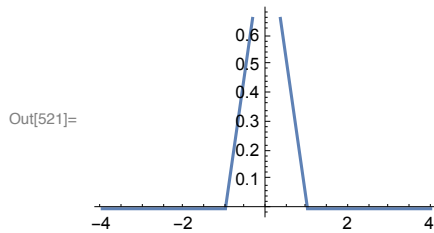
Suppose  $g(x) = h(x) = \text{UnitBox}[x]$ ;

`Plot[UnitBox[x], {x, -2, 2}]`



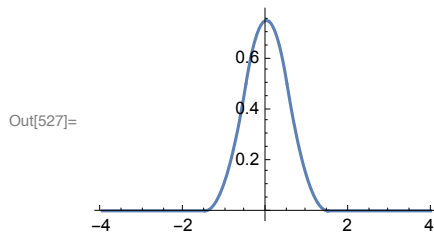
Here's the distribution of the sum of two numbers drawn from `UnitBox[x]`:

```
In[520]:= c1 = Convolve[UnitBox[x], UnitBox[x], x, y];
Plot[c1, {y, -4, 4}]
```



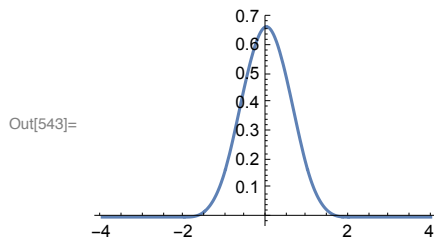
interesting, lets convolve again:

```
In[526]:= c2 = Convolve[c1, UnitBox[y], y, x];
Plot[c2, {x, -4, 4}]
```



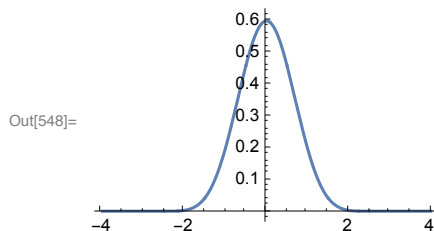
mmm ...interesting, starting to look almost gaussian. lets convolve again

```
In[542]:= c3 = Convolve[c2, UnitBox[x], x, y];
Plot[c3, {y, -4, 4}]
```



one more time:

```
In[547]:= c4 = Convolve[c3, UnitBox[y], y, x];
Plot[c4, {x, -4, 4}]
```



## General method: Use Density Mapping theorem

We'll use the density mapping theorem to turn uniformly distributed random numbers `RandomReal[]` 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  $p_Y(y)$  is a uniform distribution, i.e.  $p_Y(y) = 1$  (over the unit interval, but zero elsewhere). Then

$$y(x) = \int_{-\infty}^x p_X(x') dx' = P(x), \text{ the cumulative.}$$

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  $y \rightarrow x$ . This is given by the inverse cumulative distribution, i.e.  $P^{-1}(y)$ . Let's implement this.

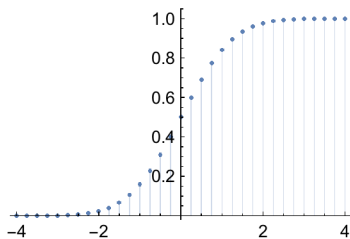
## Example

Suppose we have a discrete representation of some cumulative distribution. How can we generate samples from an assumed underlying probability density?

`lcumulgauss`

```
{{-4., 0.0000316712}, {-3.75, 0.0000884173}, {-3.5, 0.000232629},
{-3.25, 0.000577025}, {-3., 0.0013499}, {-2.75, 0.00297976}, {-2.5, 0.00620967},
{-2.25, 0.0122245}, {-2., 0.0227501}, {-1.75, 0.0400592}, {-1.5, 0.0668072},
{-1.25, 0.10565}, {-1., 0.158655}, {-0.75, 0.226627}, {-0.5, 0.308538},
{-0.25, 0.401294}, {0., 0.5}, {0.25, 0.598706}, {0.5, 0.691462}, {0.75, 0.773373},
{1., 0.841345}, {1.25, 0.89435}, {1.5, 0.933193}, {1.75, 0.959941},
{2., 0.97725}, {2.25, 0.987776}, {2.5, 0.99379}, {2.75, 0.99702}, {3., 0.99865},
{3.25, 0.999423}, {3.5, 0.999767}, {3.75, 0.999912}, {4., 0.999968}}
```

`ListPlot[lcumulgauss, Filling -> Axis]`



We'd like to sample a value on the  $y$ -axis, and then read off the corresponding value on the  $x$ -axis. We'll do this in two stages.

## Make inverse cumulative table

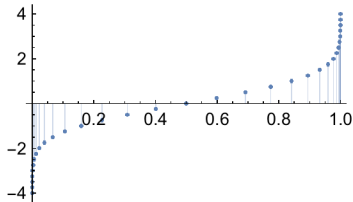
This is a useful trick whenever you want an inverse function, given a discrete representation.

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

To see what this does, evaluate:

```
{x1, y1}, {x2, y2}, {x3, y3}
RotateLeft[{{x1, y1}, {x2, y2}, {x3, y3}}, {0, 1}]
{{x1, y1}, {x2, y2}, {x3, y3}}
{{y1, x1}, {y2, x2}, {y3, x3}}
```

```
ListPlot[invlcumulgauss, Filling -> Axis]
```

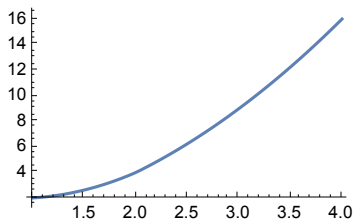


### Make interpolated function of the inverse cumulative

We need to be able to draw a uniformly distributed number on the x-axis and then read out what the y should be. But we can see from the graph above that our list, `invlcumulgauss`, has gaps in it. We'll interpolate using

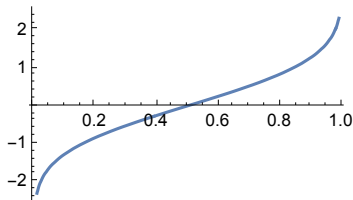
**Interpolation[]**. `Interpolation[]` is a very useful trick worth knowing about, because it turns discrete representations into continuous functions. Here's an example.

```
test = Interpolation[{{1, 2.}, {2, 4}, {3, 9}, {4, 16}}, InterpolationOrder -> 2];
Plot[test[x], {x, 1, 4}]
```



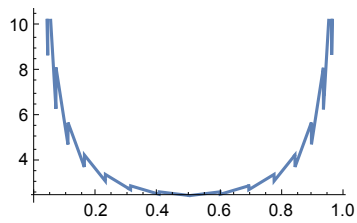
```
interinvlcumulgauss = Interpolation[invlcumulgauss, InterpolationOrder -> 2];
```

```
Plot[interinvlcumulgauss[x], {x, 0.01, 0.99}] (* avoiding x=0 and x=1 *)
```



- ▶ 5. Interpolation works by fitting polynomial curves to the data. Try various interpolation orders (the default is 3), `Interpolation[invlcumulgauss, InterpolationOrder -> 2]`
- ▶ 6. You can take derivatives of Interpolated functions, but they will exaggerate points that were not interpolated smoothly

```
Plot[interinvcumulgauss'[x], {x, 0.01, 0.99}]
```



## Draw samples with a standard deviation of $\text{Sqrt}[10]$

```
Round[10 interinvcumulgauss[RandomReal[]]]
```

```
10
```

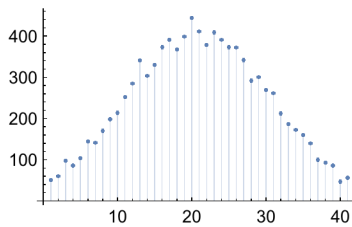
## Draw a bunch of samples, and plot up histogram

```
z = Table[Round[10 interinvcumulgauss[RandomReal[ {.01, .99}]]], {10 000}];
```

```
domain = Range[-20, 20];
```

```
Freq = (Count[z, #1] &) /@ domain;
```

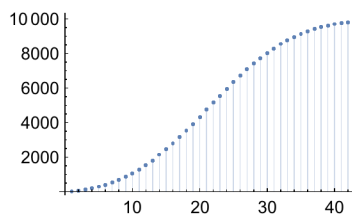
```
ListPlot[Freq, Filling -> Axis]
```



## “Return full circle”: Plot up cumulative histogram

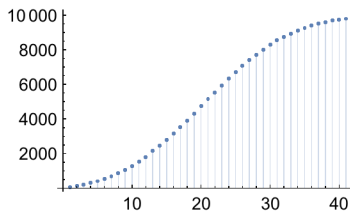
```
CumFreq = FoldList[Plus, 0, Freq];
```

```
ListPlot[CumFreq, Filling -> Axis]
```



Same thing, with `Accumulate[]` :

```
CumFreq = Accumulate[Freq];
ListPlot[CumFreq, Filling -> Axis]
```



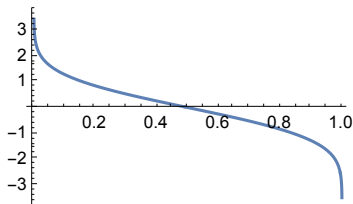
- 7. Method 2 Applied to Gaussian, where a built-in inverse function is available.

InverseErf[] is the inverse of :

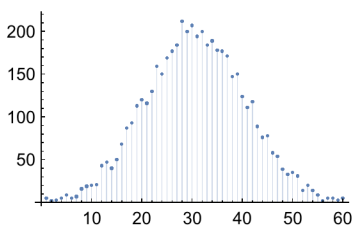
$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

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

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



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




---

## Next time

Graphical models and structured distributions  
Sampling in higher dimensional spaces

---

## 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.
- Berkes P, Orbán G, Lengyel M, and Fiser J. (2011) Spontaneous Cortical Activity Reveals Hallmarks of an Optimal Internal Model of the Environment. *Science* 7 January 2011: 331 (6013), 83-87. [DOI:10.1126/science.1195870]
- Freeman, J., & Simoncelli, E. P. (2011). Metamers of the ventral stream. *Nature Publishing Group*, 14(9), 1195–1201. <http://doi.org/10.1038/nn.2889>
- Freeman, J., Ziemba, C. M., Heeger, D. J., Simoncelli, E. P., & Movshon, J. A. (2013). A functional and perceptual signature of the second visual area in primates. *Nature Publishing Group*, 16(7), 974–981. <http://doi.org/10.1038/nn.3402>
- 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 (2004) Object perception as Bayesian inference. *Annual Review of Psychology*. (pdf, <http://arjournals.annualreviews.org/doi/pdf/10.1146/annurev.psych.55.090902.142005>)
- Knill, D. C., & Richards, W. (1996). *Perception as Bayesian Inference*. Cambridge: Cambridge University Press.
- McDermott, J. H., & Simoncelli, E. P. (2011). Sound Texture Perception via Statistics of the Auditory Periphery: Evidence from Sound Synthesis. *Neuron*, 71(5), 926–940. <http://doi.org/10.1016/j.neuron.2011.06.032>
- 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.
- Zhu, S. C., Wu, Y., & Mumford, D. (1998). Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2), 107–126.