

>> Lectures >> Matlab 10

Navigator



10.1 Probability Distributions

This section discusses two basic probability density functions and probability distributions: *uniform* and *normal*, Gaussian mixture models, and GMM curve fitting.

10.1.1 Common PDFs

Uniform probability density functions can be generated using function `unifpdf`. Given a range x , and a left and right endpoint, `unifpdf` distributes probabilities uniformly over x .

`unifpdf(x, a, b)` : x = vector of range (including granularity), a = left endpoint, b = right endpoint

```
x = -10:10;
pdfUniform = unifpdf(x, -5, 5);
plot(x, pdfUniform);
```

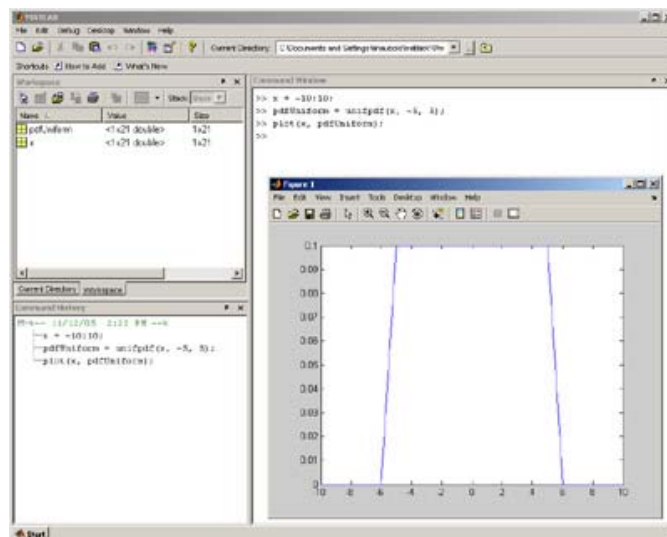


Figure 10.1
[Click to enlarge](#)

Normal probability density functions are generated using function `normpdf`. Characteristic of a normal distribution are mean and standard deviation.

`normpdf(x, mean, std)` : x = vector of range (including granularity)

```
x = -15:0.1:25;
mu = 3;
sigma = 4;
pdfNormal = normpdf(x, mu, sigma);
plot(x, pdfNormal);
```

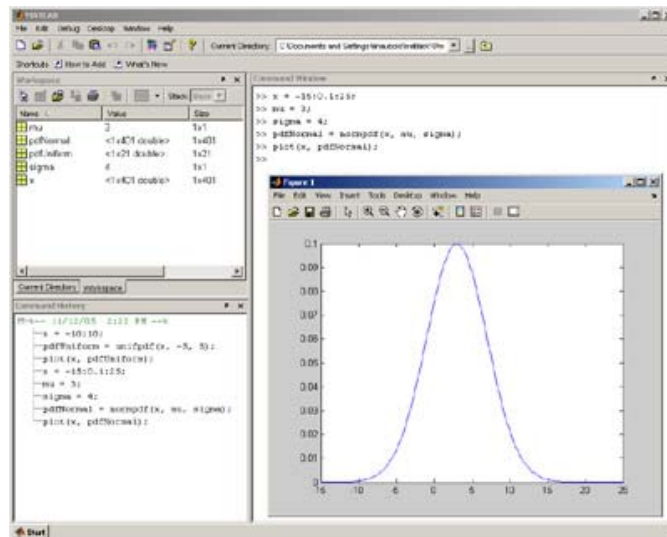


Figure 10.2
[Click to enlarge](#)

10.1.2 Randomly generated PDFs

unifpdf and *normpdf* generate "perfect" densities; however, typical data observations only fit these distributions approximately. To simulate these situations, Matlab offers functions for random number generation for both uniform and normal distributions.

Function `rand` generates uniformly distributed random values between 0 and 1.

`rand(rows, columns)`: matrix with rows and columns of random values.

```
pdfUniform = rand(1, 10000);
subplot(2,1,1), hist(pdfUniform), title('Uniform distribution, histogram');
subplot(2,1,2), hist(pdfUniform, 100), title('Uniform distribution, 100 bin histogram');
```

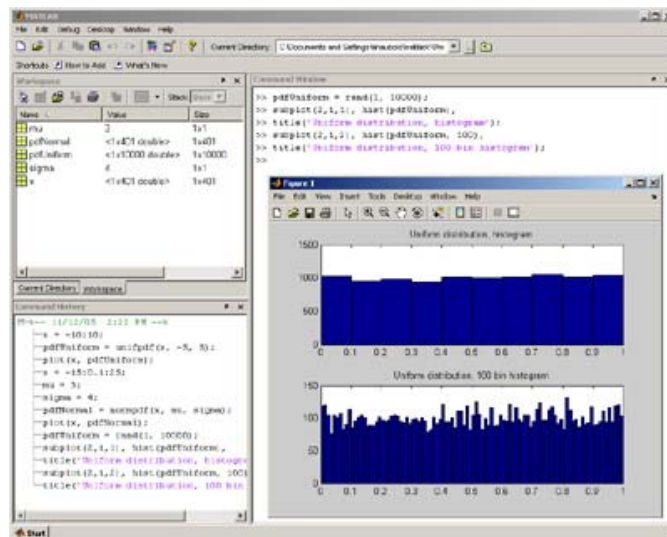


Figure 10.3
[Click to enlarge](#)

To fit a different range of x-values, we can shift and scale the random value matrix accordingly:

```
pdfUniformSS = pdfUniform * 10 + 100;
hist(pdfUniformSS, 100), title('Uniform distribution Shifted and Scaled, 100 bin histogram');
```

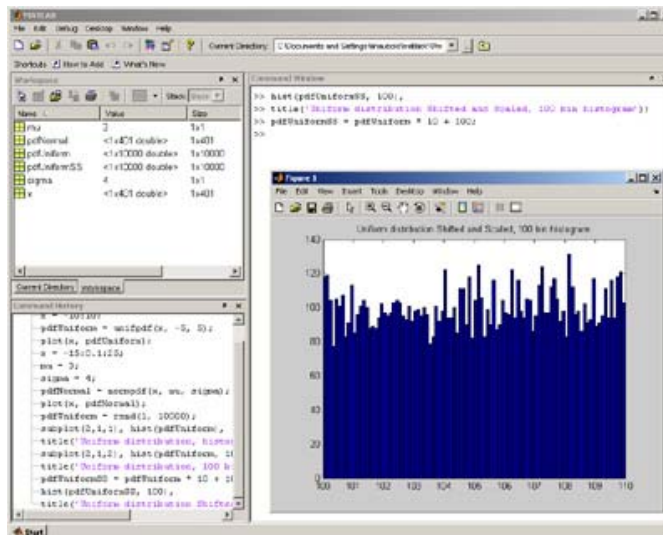


Figure 10.4
[Click to enlarge](#)

Function `randn` generates normally distributed random values with mean=0 and standard deviation=0.

`randn(rows, columns)` : matrix with rows and columns random values.

```

pdfNormal = randn(1, 10000);
subplot(2,1,1), hist(pdfNormal), title('Normal distribution, histogram');
subplot(2,1,2), hist(pdfNormal, 100), title('Normal distribution, 100 bin
histogram');

```

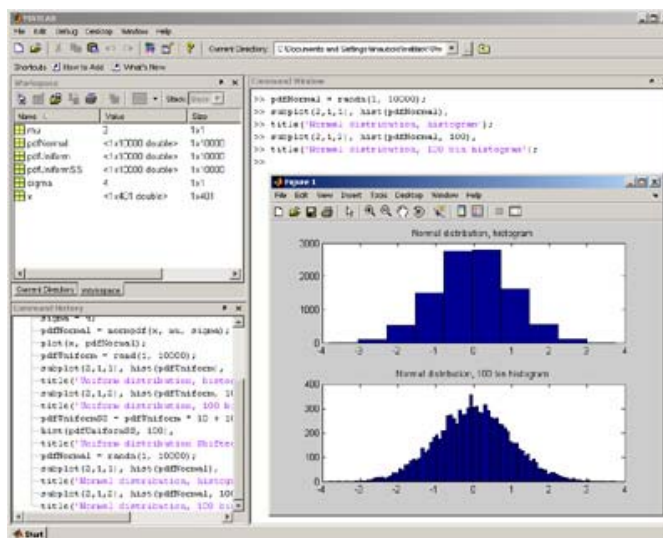


Figure 10.5
[Click to enlarge](#)

A random normal distribution with mean μ and standard deviation σ is obtained by shifting and scaling:

```

pdfMean = 5;
pdfStd = 7;
pdfNormalSS = randn(1, 10000) * pdfStd + pdfMean;
subplot(2,1,1), hist(pdfNormalSS), title('Normal distribution with Mean=5,
Std=7');
subplot(2,1,2), hist(pdfNormalSS, 100), title('Normal distribution with
Mean=5, Std=7');

```

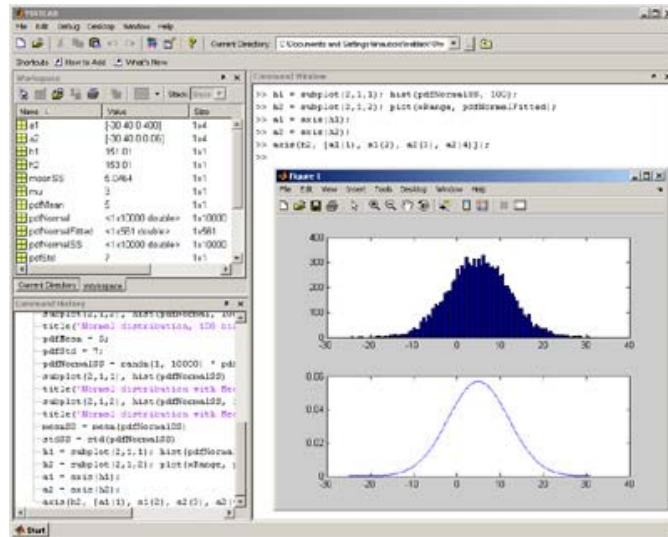



Figure 10.8
[Click to enlarge](#)

For a mixture of different normal distributions (a Gaussian Mixture Model), the correct parameters for mean and standard deviation for each Gaussian cannot be computed by simply taking mean and std of the entire data set. The observed data must be divided into several Gaussians, each of with its own mean and standard deviation. One approach for obtaining the paramaters for each Gaussian is to apply the algorithm for Expectation Maximization:

 [em_1dim.m](#) Expectation Maximization - Approximation of Gaussian Mixture Models (GMM)

This requires an initial guess as to how many Gaussians are hidden in the distribution.

A GMM with 2 Gaussian distributions:

```
pdfGMM = [(randn(1, 10000) * 7 + 3), (randn(1, 10000) * 2 + 9)];
subplot(2,1,1), hist(pdfGMM, 100);
```

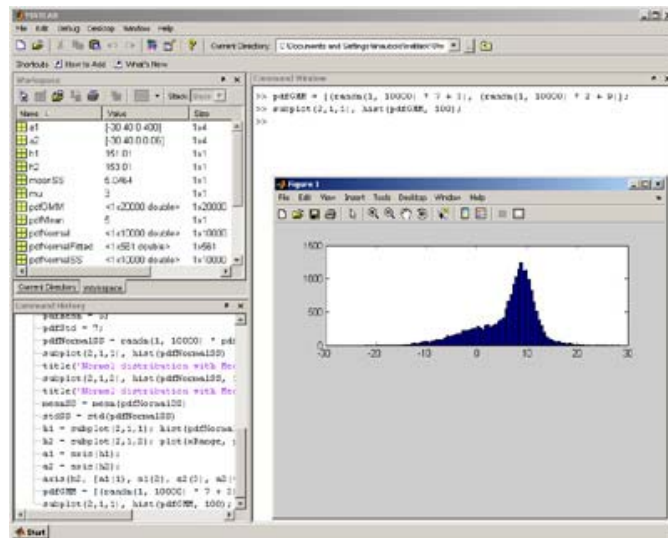


Figure 10.9
[Click to enlarge](#)

Estimate mean and standard deviations for 2 Gaussians:

```
[em_thr, em_thr_behavior, P, meanV, stdV, pdf_x, xx, pdf_xx, cdf_xx] =
em_1dim(pdfGMM, 2);
meanV
stdV
```

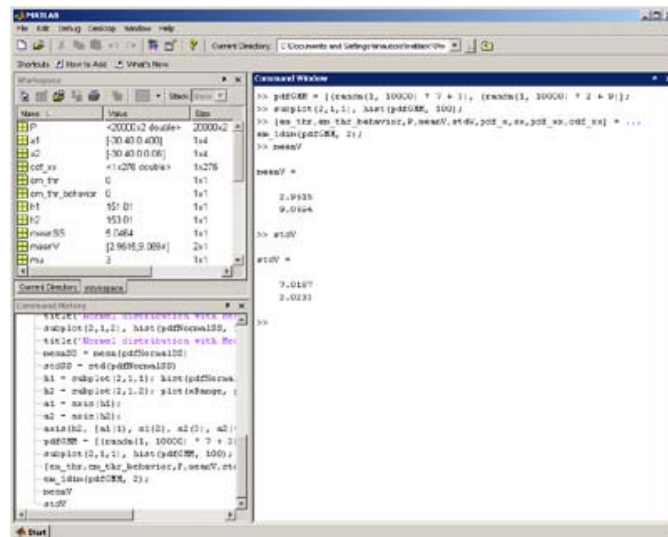


Figure 10.10
[Click to enlarge](#)

To verify, we plot a histogram of the original data and the 2 Gaussian distributions:

```

h1 = subplot(2,1,1); hist(pdfGMM, 100);
h2 = subplot(2,1,2); plot(xx, normpdf(xx, meanV(1), stdV(1)), 'r');
hold on;
subplot(2,1,2), plot(xx, normpdf(xx, meanV(2), stdV(2)), 'g');
hold off;
  
```

```

a1 = axis(h1);
a2 = axis(h2);
axis(h2, [a1(1), a1(2), a2(3), a2(4)]);
  
```

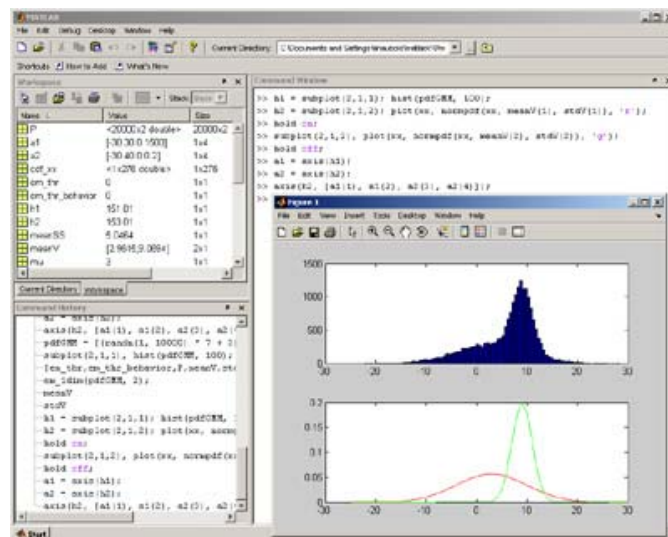


Figure 10.11
[Click to enlarge](#)

