K-Means

今日の関数一覧

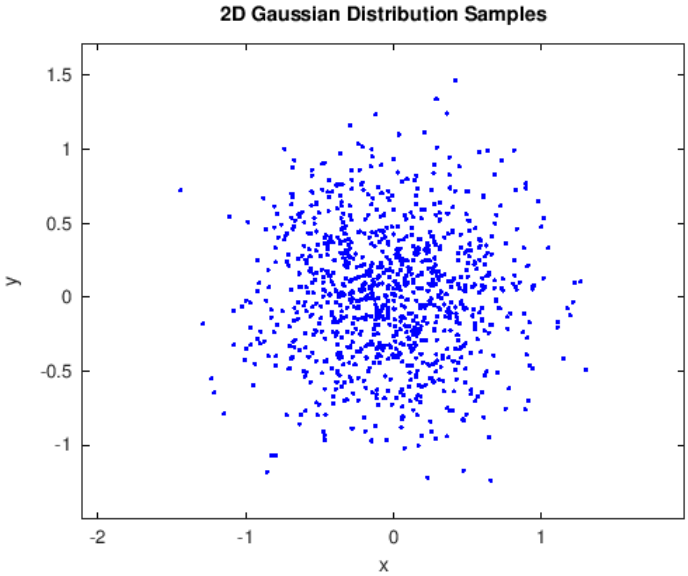
- `sample_gaussian2d.m`
- `rand_mixture_of_gaussians.m`
- `plot_kmeans_results.m` ← スライド中でデモとして記載している内容を関数にまとめたもの
- `kmeans.m`

`sample_gaussian2d.m`

2次元平面上にガウス分布状にサンプル点を生成する関数

In [3]:
```
function samples = sample_gaussian2d(mu, sigma, num_samples)
  samples = mu + sigma * randn(num_samples, 2);
end
```

In [14]:
```
mu = 0;
sigma = sqrt(.2);
num_samples = 1000;

% サンプリング
samples = sample_gaussian2d(mu, sigma, num_samples);

% プロット
figure;
plot(samples(:,1), samples(:,2), 'b.');
xlabel('x');
ylabel('y');
title('2D Gaussian Distribution Samples');
axis equal;
grid off;
```



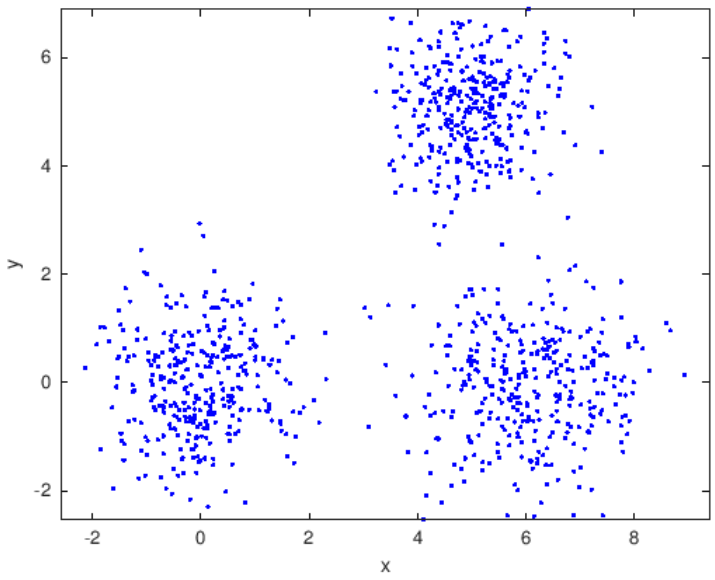
`rand_mixture_of_gaussians.m`

2次元平面上に混合ガウス分布状にサンプル点を生成する関数

In [7]:
```
function samples = rand_mixture_of_gaussians(mu_list, sigma_list, num_samples)
  num_distributions = length(mu_list);
  samples_per_distribution = round(num_samples / num_distributions);

  samples = [];
  for i = 1:num_distributions
    mu = mu_list{i};
    sigma = sigma_list(i);

    % 指定されたガウス分布から2次元のサンプルを生成
    samples_i = bsxfun(@plus, mu, sigma * randn(samples_per_distribution, 2));

    samples = [samples; samples_i];
  end
```

```
  % 必要な数のサンプルに調整
  if size(samples, 1) > num_samples
    samples = samples(1:num_samples, :);
  end
end
```

In [13]:
```
% パラメータ設定
mu_list = {[0, 0], [5, 5], [6, 0]};
sigma_list = [0.9, 0.8, 1.0];
num_samples = 1000;

% ランダムサンプル生成
data = rand_mixture_of_gaussians(mu_list, sigma_list, num_samples);

% プロット設定
figure;
plot(data(:,1), data(:,2), 'b.');
xlabel('x');
ylabel('y');
axis equal;
```



plot_kmeans_results.m

k-meansの結果をわかりやすくプロットするための関数

In [46]:
```
function plot_kmeans_results(data, labels, centroids, obj_values)
  % クラスタ数をラベルから取得
  k = max(labels);

  % プロット設定
  figure;

  % 元のデータプロット
  subplot(2, 2, 1);
  scatter(data(:, 1), data(:, 2), 50, 'b', 'filled');
  title('Original Data Points');
  axis equal;
  set(gca, 'xtick', [], 'ytick', []);  % tickの線を非表示

  % クラスタリング後のデータプロット
  subplot(2, 2, 2);
  hold on;
  colors = hsv(k); % 動的に色を生成

  for i = 1:k
    scatter(data(labels == i, 1), data(labels == i, 2), 50, colors(i, :), 'filled');
  end
  scatter(centroids(:, 1), centroids(:, 2), 100, 'k', 'filled', 'd'); % クラスタ中心をプロット
  title('k-means Clustering');
  axis equal;
  set(gca, 'xtick', [], 'ytick', []);
  hold off;

  % 目的関数の収束の様子をプロット
  subplot(2, 2, [3, 4]);
  plot(obj_values, '-o', 'LineWidth', 6, 'MarkerSize', 10);
  title('Objective Function Convergence');
  xlabel('Iteration', 'fontsize', 12);
  ylabel('Objective Function Value', 'fontsize', 12);
```

```
  grid on;
end
```

kmeans.m

k-means法を実装した関数

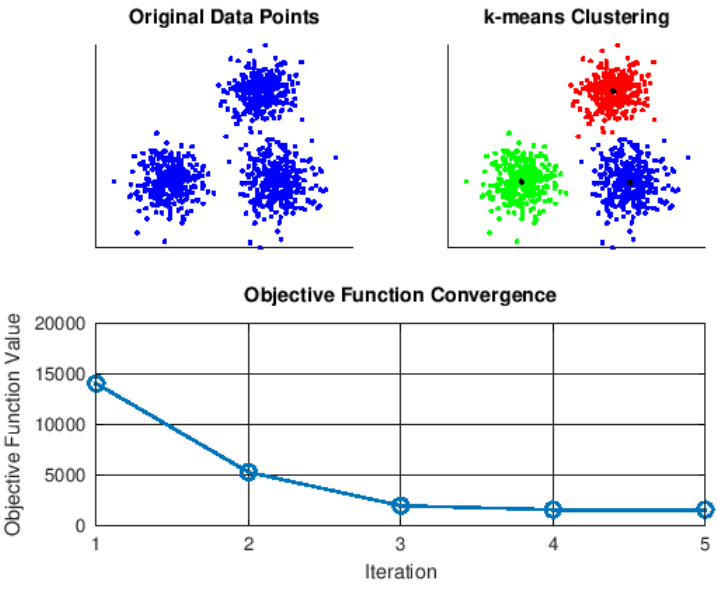
In [15]:
```
function [centroids, labels, obj_values] = kmeans(data, k, max_iter=1000)
  n = size(data, 1);                          % データ点の数
  centroids = data(randperm(n, k), :);        % 初期クラスタ中心の選定
  x2 = sum(data.^2, 2);                       % 各データ点の平方和
  prev_centroids = inf(k, size(data, 2));     % 前回のクラスタ中心
  labels = zeros(n, 1);                       % クラスタラベルの初期化
  obj_values = [];                            % 目的関数の値を記録するリスト

  for iter = 1:max_iter
    m2 = sum(centroids.^2, 2);                % クラスタ中心の平方和
    d = m2 + x2' - 2 * centroids * data';     % 距離行列の計算（ブロードキャスティングを利用）
    [min_d, labels] = min(d, [], 1);          % データ点の所属クラスタを求める

    for t = 1:k
      cluster_points = data(labels == t, :);
      if ~isempty(cluster_points)
        centroids(t, :) = mean(cluster_points, 1);  % 新しいクラスタ中心の計算
      end
    end

    obj_values = [obj_values; sum(min_d)];  % 目的関数の値を記録

    if norm(centroids - prev_centroids) < 0.001  % 収束条件の判定
      break;
    end

    prev_centroids = centroids;
  end
end
```

In [47]:
```
% パラメータ設定
mu_list = {[0, 0], [5, 5], [6, 0]};
sigma_list = [0.9, 0.8, 1.0];
num_samples = 1000;

% ランダムサンプル生成
data = rand_mixture_of_gaussians(mu_list, sigma_list, num_samples);
```

In [48]:
```
%%time

% k-meansクラスタリングの実行
k = 3;
[centroids, labels, obj_values] = kmeans(data, k);
```

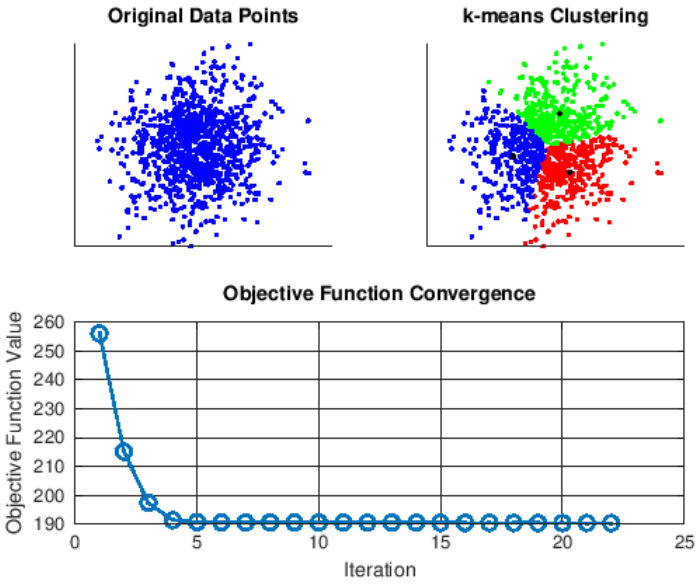
```
Time: 0.01859450340270996 seconds.
```

In [49]:
```
% 結果をプロット
plot_kmeans_results(data, labels, centroids, obj_values)
```



In [50]:
```
% データセットを変更して実験
data = sample_gaussian2d(mu, sigma, num_samples);
```

```
% k-meansクラスタリングの実行
[centroids, labels, obj_values] = kmeans(data, k);

% 結果をプロット
plot_kmeans_results(data, labels, centroids, obj_values)
```



In [ ]: