13장 영상 분류
(Image Classification)

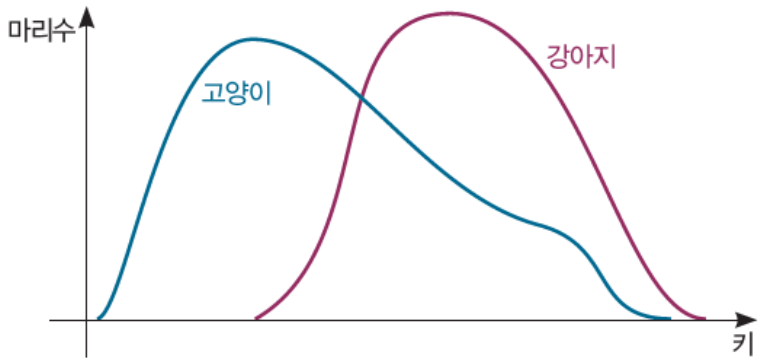
영상 분류


전처리
(Preprocessing)
특징추출
(feature extraction)
분류
(Classification)
강아지
고양이

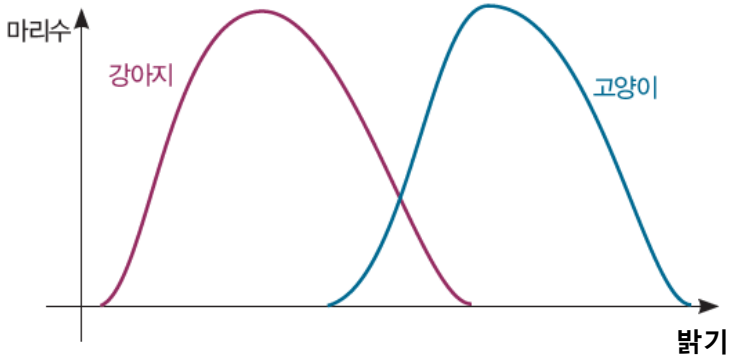

영상 분류기
● 고양이 : 83%
● 강아지 : 10%
● 모피 : 5%
● ...

특징 공간 (feature space)

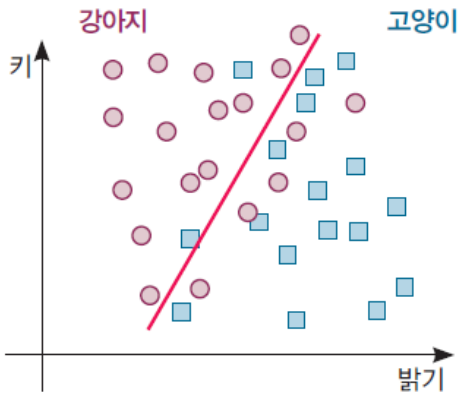
특징 1: 키



특징 2: 밝기



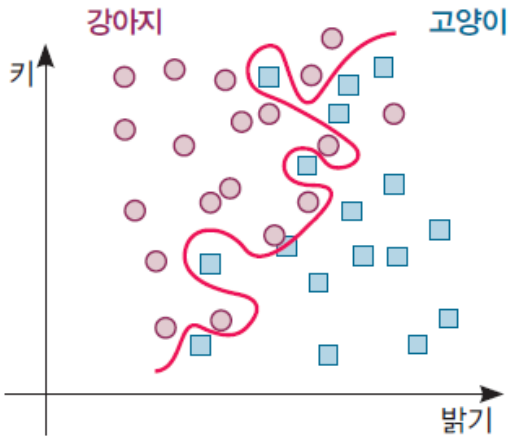
2개의 특징 사용 시



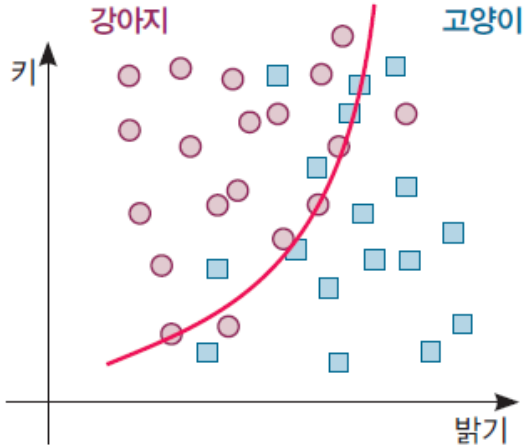
특징 공간 (feature space)

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$: 특징벡터 (feature vector)

분류기의 판단 경계선



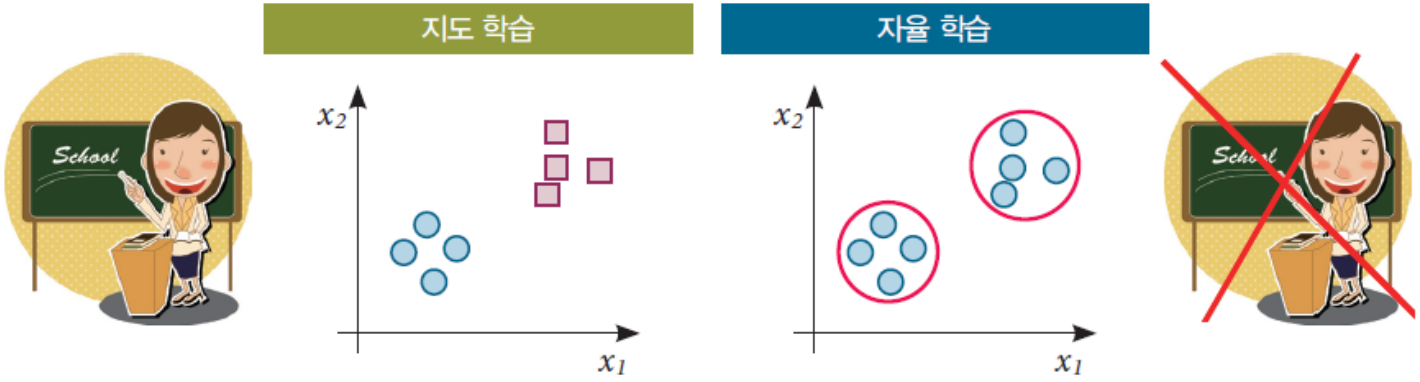
- 복잡
- 샘플데이터 (학습데이터) 에는 완벽하게 적용
- 실제데이터에도 잘 적용 ??



- 단순
- 실제데이터에도 적절하게 적용 가능

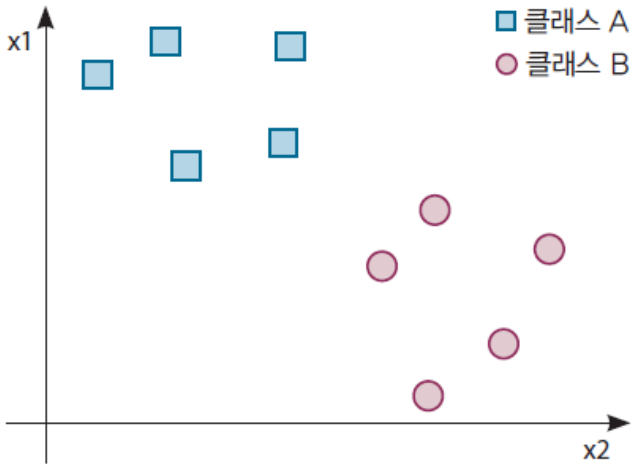
분류기의 학습방법

- 지도학습 vs. 비지도학습
  - 지도학습 (supervised learning)
    - 학습 샘플에 대한 label (참값, Ground Truth) 부여
    - kNN
  - 비지도학습 (자율학습, unsupervised learning)
    - 학습 샘플에 대한 label 없음
    - K-means

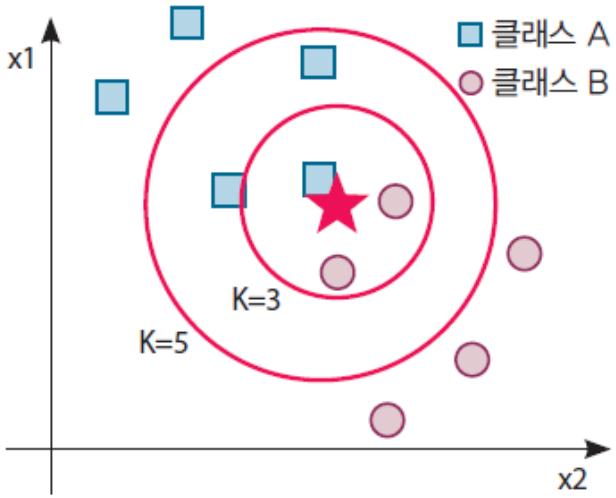


kNN 알고리즘

- K-Nearest Neighbor 알고리즘
  - Feature space 에서 임의의 feature vector 에 대한 클래스 분류
  - K 개의 최근접 이웃을 찾아 다수의 이웃이 소속된 클래스로 분류함



학습 데이터



테스트 데이터

kNN 알고리즘

- OpenCV 함수
  - 학습 단계

```
Ptr <ml::KNearest> knn = ml::KNearest::create();
Ptr <ml::TrainData> trainData = ml::TrainData::create(train_features, ROW_SAMPLE, labels);
knn->train (trainData);
```

  - 테스트 단계

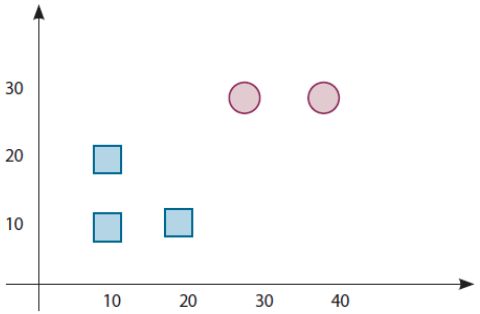
```
Mat predictedLabels;
knn->findNearest(sample, K, predictedLabels);
```

kNN 알고리즘

- 학습 단계

```
Ptr <ml::KNearest> knn = ml::KNearest::create();
Ptr <ml::TrainData> trainData = ml::TrainData::create(train_features, ROW_SAMPLE, labels);
knn->train (trainData);
```

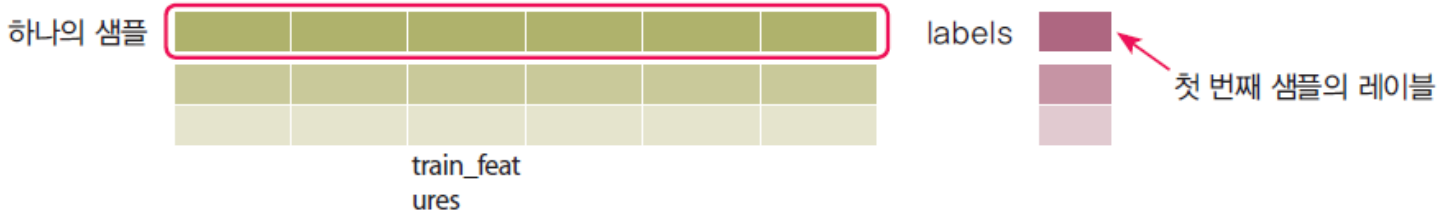
영상의 행에 샘플이 저장됨



train_features

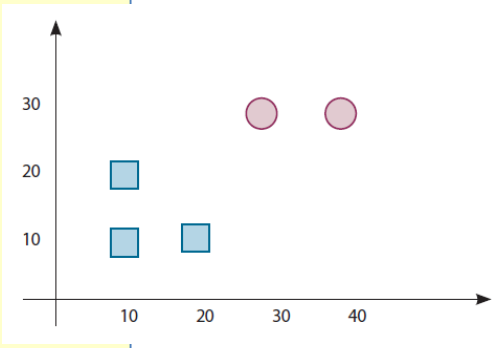
10	10
10	20
20	10
30	30
40	30

1
1
1
2
2



실습 1

```
int main()
{
    Mat train_features(5, 2, CV_32FC1);
    Mat labels(5, 1, CV_32FC1);

    // 점의 좌표를 train_features에 입력한다.
    train_features.at<float>(0, 0) = 10, train_features.at<float>(0, 1) = 10;
    train_features.at<float>(1, 0) = 10, train_features.at<float>(1, 1) = 20;
    train_features.at<float>(2, 0) = 20, train_features.at<float>(2, 1) = 10;
    train_features.at<float>(3, 0) = 30, train_features.at<float>(3, 1) = 30;
    train_features.at<float>(4, 0) = 40, train_features.at<float>(4, 1) = 30;

    // 원하는 레이블을 labels에 입력한다.
    labels.at<float>(0, 0) = 1;
    labels.at<float>(1, 0) = 1;
    labels.at<float>(2, 0) = 1;
    labels.at<float>(3, 0) = 2;
    labels.at<float>(4, 0) = 2;

    // 학습 과정
    Ptr<ml::KNearest> knn = ml::KNearest::create();
    Ptr<ml::TrainData> trainData = ml::TrainData::create(train_features, ROW_SAMPLE, labels);
    knn->train(trainData);
```

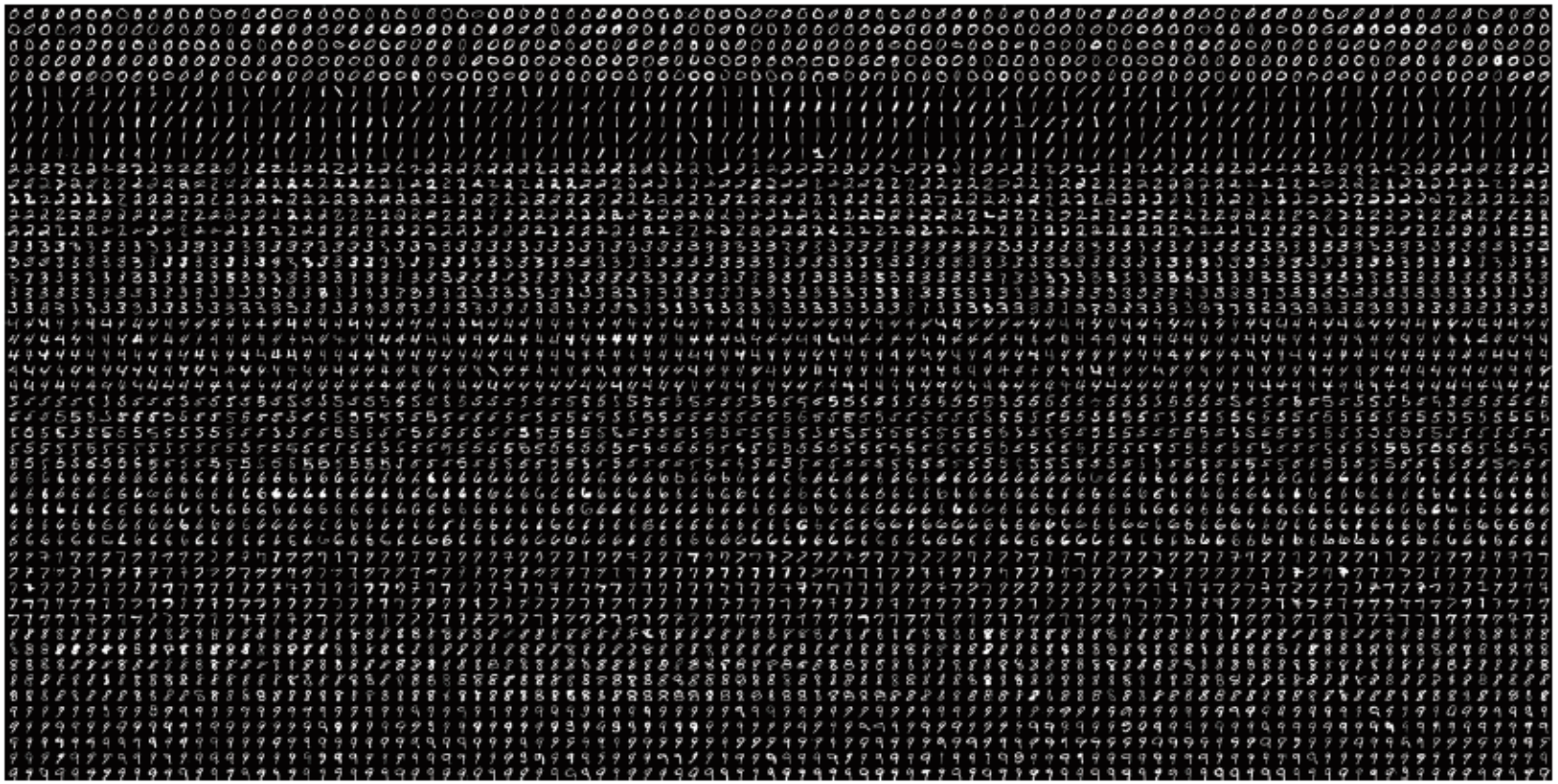


실습 1

```
    // 테스트 과정
    Mat sample(1, 2, CV_32FC1);
    Mat predictedLabels;

    // 테스트 데이터를 입력한다.
    sample.at<float>(0, 0) = 28, sample.at<float>(0, 1) = 28;
    knn->findNearest(sample, 2, predictedLabels);

    float prediction = predictedLabels.at<float>(0, 0);
    cout << "테스트 샘플의 라벨 = " << prediction << endl;
    return 0;
}
```

실습 2

- kNN을 이용한 숫자 인식



실습 2

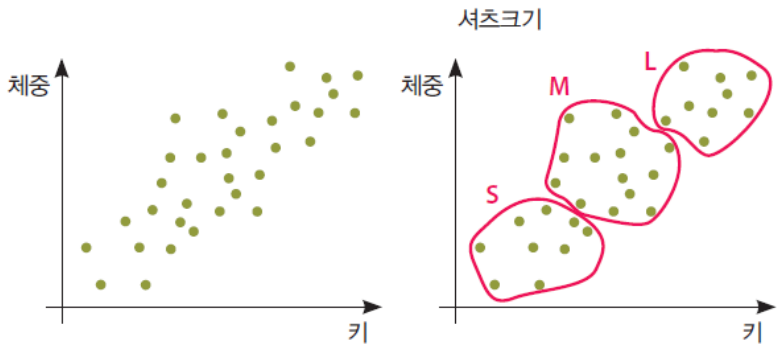
```
int main()
{
    Mat img;
    img = imread("d:/digits.png", IMREAD_GRAYSCALE);
    namedWindow("original", WINDOW_AUTOSIZE);
    imshow("original", img);
    waitKey(0);

    Mat train_features(5000, 400, CV_32FC1);
    Mat labels(5000, 1, CV_32FC1);

    // 각 숫자 영상을 행 벡터로 만들어서 train_feature에 저장한다.
    for (int r = 0; r < 50; r++) {
        for (int c = 0; c < 100; c++) {
            int i = 0;
            for (int y = 0; y < 20; y++) {
                for (int x = 0; x < 20; x++) {
                    train_features.at<float>(r * 100 + c, i++) = img.at<uchar>(r * 20 + y, c * 20 + x);
                }
            }
        }
    }

   // 각 숫자 영상에 대한 레이블을 저장한다.
   for (int I = 0; I < 5000; i++) {
        labels.at<float>(I, 0) = (I / 500);
    }
```

실습 2

```
    // 학습 과정
    Ptr<ml::KNearest> knn = ml::KNearest::create();
    Ptr<ml::TrainData> trainData = ml::TrainData::create(train_features, ROW_SAMPLE, labels);
    knn->train(trainData);

    // 테스트 과정
    Mat predictedLabels;
    for (int i = 0; i < 5000; i++) {
        Mat test = train_features.row(i);
        knn->findNearest(test, 3, predictedLabels);
        float prediction = predictedLabels.at<float>(0);
        cout << "테스트 샘플" << i << "의 라벨 = " << prediction << '₩n';
    }
}
```

K-Means 알고리즘

- 목적
  - 주어진 n 개의 관측값을 k 개의 클러스터 (그룹) 로 분할
  - 관측값들의 거리가 최소인 클러스터로 분류



K=3

K-Means 알고리즘

Algorithm 13.1

입력값

① *k*: 클러스터 수

② *D*: *n*개의 데이터를 포함하는 집합

출력값: *k*개의 클러스터

알고리즘

① 집합 *D*에서 *k*개의 데이터를 임의로 추출하고, 이 데이터들을 각 클러스터의 중심centroid으로 설정한다(초기값 설정).

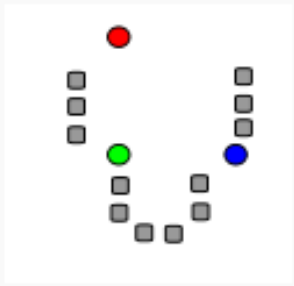
② 집합 *D*의 각 데이터에 대해 *k*개의 클러스터 중심과의 거리를 계산하고, 각 데이터가 어느 중심점centroid과 가장 유사도가 높은지 알아낸다. 그리고 그렇게 찾아낸 중심점으로 각 데이터들을 할당한다.

③ 클러스터의 중심점을 다시 계산한다. 즉, 2에서 재할당 된 클러스터들을 기준으로 중심점을 다시 계산한다.

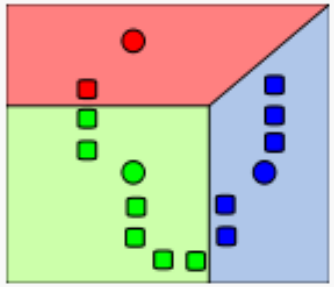
④ 각 데이터의 소속 클러스터가 바뀌지 않을 때까지 ②, ③ 과정을 반복한다.

K-means 알고리즘

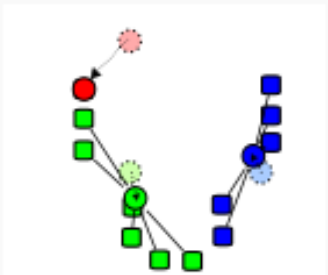
Demonstration of the standard algorithm



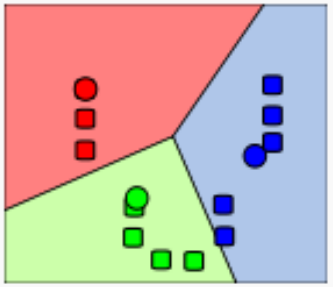
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

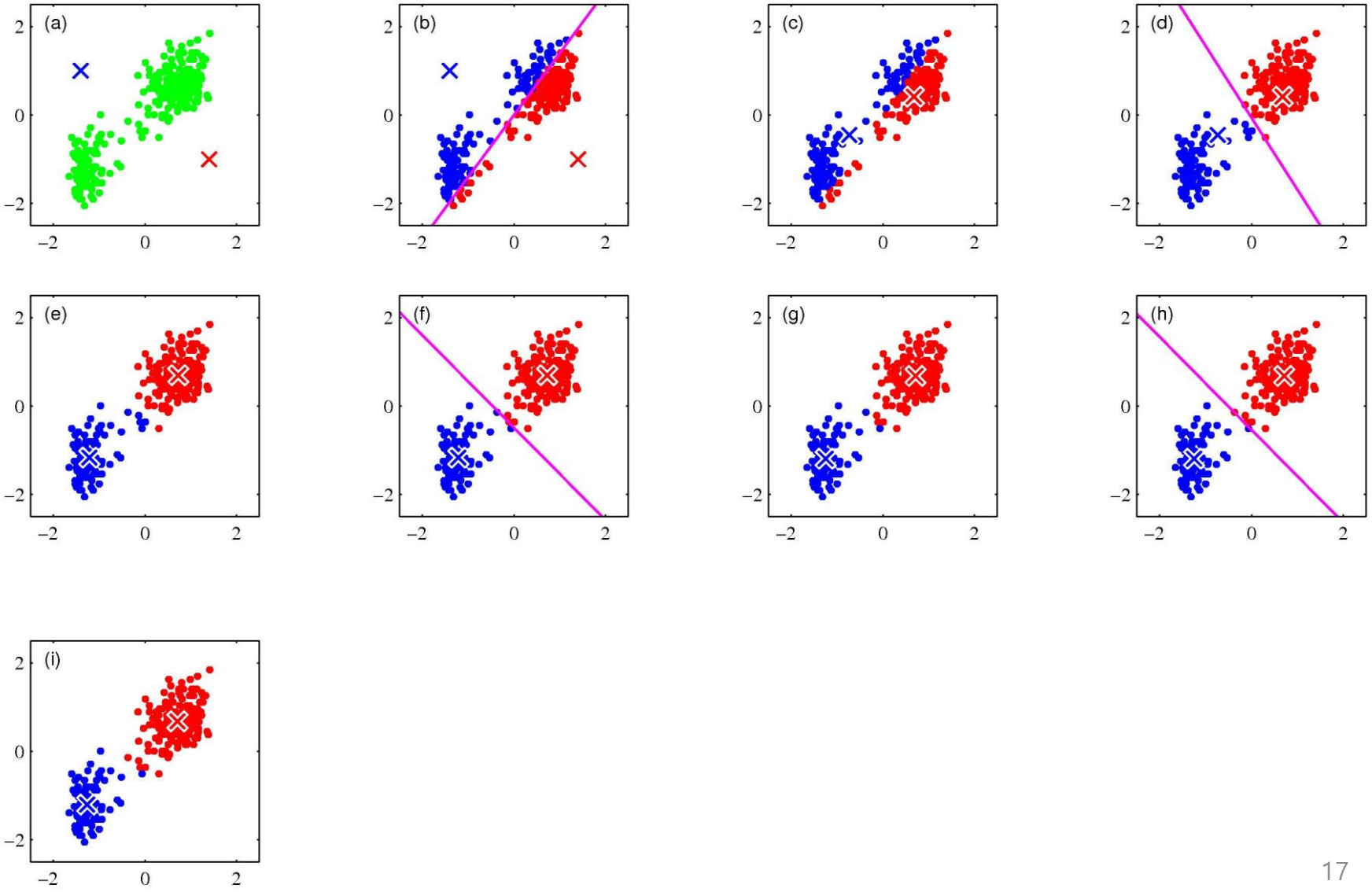


3) The centroid of each of the *k* clusters becomes the new means.



4) Steps 2 and 3 are repeated until convergence has been reached.

K-means 알고리즘



K-Means 알고리즘

- OpenCV 함수

```
double kmeans(InputArray samples, int K, InputOutputArray labels,
        TermCriteria criteria, int attempts, int flags,
        OutputArray centers=noArray())
```

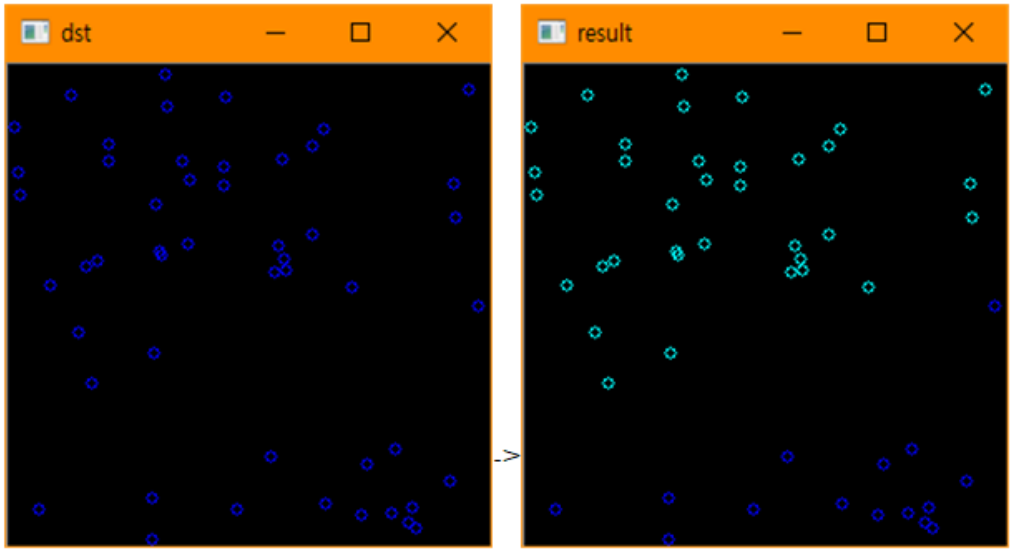
매개 변수	설명
samples	샘플 데이터 행렬. 행렬의 행에 샘플이 저장되어 있다.
K	우리가 원하는 클러스터의 개수
labels	각 클러스터의 레이블이 저장될 행렬
criteria	알고리즘을 종료하는 조건을 기술한다.
attempts	알고리즘이 수행되는 횟수
flags	알고리즘 초기화 플래그. KMEANS_PP_CENTERS는 Arthur and Vassilvitskii이 주장한 클러스터 초기화 방법을 의미한다. KMEANS_RANDOM_CENTERS은 랜덤으로 클러스터의 중심점을 잡는 것이다.
centers	클러스터의 중심이 저장된 출력 행렬

실습 3

```
int main()
{
    Mat samples(50, 2, CV_32F);

    for (int y = 0; y < samples.rows; y++) {
                samples.at<float>(y, 0) = (rand() % 255);
                samples.at<float>(y, 1) = (rand() % 255);
    }
    Mat dst(256, 256, CV_8UC3);

    for (int y = 0; y < samples.rows; y++) {
                float x1 = samples.at<float>(y, 0);
                float x2 = samples.at<float>(y, 1);
                circle(dst, Point(x1, x2), 3, Scalar(255, 0, 0));
    }
    imshow("dst", dst);
    Mat result;
    Mat labels(50, 1, CV_8UC1);

    Mat centers;
    result = Mat::zeros(Size(256, 256), CV_8UC3);
    kmeans(samples, 2, labels, TermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 10000,
    0.0001),3, KMEANS_PP_CENTERS, centers);
```

실습 3

```
    for (int y = 0; y < samples.rows; y++) {
                float x1 = samples.at<float>(y, 0);
                float x2 = samples.at<float>(y, 1);
                int cluster_idx = labels.at<int>(y, 0);
                if (cluster_idx == 0)
                    circle(result, Point(x1, x2), 3, Scalar(255, 0, 0));
                else
                    circle(result, Point(x1, x2), 3, Scalar(255, 255, 0));
    }
    imshow("result", result);
    waitKey(0);
    return(0);
}
```



실습 4: 색상 개수 줄이기

```
int main()
{
    Mat src = imread("d:/lenna.jpg", 1);

    // 학습 데이터를 만든다.
    Mat samples(src.rows * src.cols, 3, CV_32F);
    for (int y = 0; y < src.rows; y++)
       for (int x = 0; x < src.cols; x++)
          for (int z = 0; z < 3; z++)
             samples.at<float>(y + x * src.rows, z) = src.at<Vec3b>(y, x)[z];

    // 클러스터의 개수는 15가 된다.
    int clusterCount = 15;
    Mat labels;
    int attempts = 5;
    Mat centers;
    kmeans(samples, clusterCount, labels, TermCriteria(CV_TERMCRIT_ITER | CV_TERMCRIT_EPS, 10000, 0.0001), attempts,
    KMEANS_PP_CENTERS, centers);

    Mat new_image(src.size(), src.type());
    for (int y = 0; y < src.rows; y++)
       for (int x = 0; x < src.cols; x++)
       {
          int cluster_idx = labels.at<int>(y + x * src.rows, 0);
          new_image.at<Vec3b>(y, x)[0] = centers.at<float>(cluster_idx, 0);
          new_image.at<Vec3b>(y, x)[1] = centers.at<float>(cluster_idx, 1);
          new_image.at<Vec3b>(y, x)[2] = centers.at<float>(cluster_idx, 2);
       }
    imshow("clustered image", new_image);
    waitKey(0);
}
```

성능 평가

- 인식 성능 측정

$$\text{정인식률} = \frac{c}{N},\ \text{기각률} = \frac{r}{N},\ \text{오류율} = \frac{e}{N}$$
$$\text{이때 } c = \text{맞는 샘플수},\ r = \text{기각한 샘플수},\ e = \text{틀린 샘플수} \qquad (1.1)$$
$$(N = c + r + e)$$

  - (예) 자동차 인식 문제
    - 입력: 자동차를 찍은 영상
    - 출력: 5 class (세단, RV, 버스, 트럭, 트레일러)
      - 맞는 샘플(c): 세단->세단, 버스->버스 …
      - 틀린 샘플 (e): 세단->버스, 버스-> 트럭, ……
      - 기각 샘플 (r): 분류포기

  - 정인식률 correct recognition rate
  - 기각률 rejection rate
  - 오류율 error rate

성능 평가

- 혼동 행렬 confusion matrix
  - 오류 경향을 세밀하게 분석하는데 사용

표 1-1 부류가 두 개인 경우의 혼동 행렬

참 부류 \ 분류 결과	ω_1	ω_2
ω_1	n_{11}(TP)	n_{12}(FN)
ω_2	n_{21}(FP)	n_{22}(TN)

(예) 불량검출문제: ω_1 = 정상 (Positive), ω_2 = 불량 (Negative)

TP: True Positive (참 긍정)
TN: True Negative (참 부정)
FP: False Positive (거짓 긍정)
FN: False Negative (거짓 부정)

성능 평가

- 참/거짓 긍정률, 참/거짓 부정률, 재현률과 정확률, *F* 측정

참 부류 \ 분류 결과	ω_1	ω_2
ω_1	n_{11}(TP)	n_{12}(FN)
ω_2	n_{21}(FP)	n_{22}(TN)

$$
\begin{aligned}
\text{거짓 긍정률}\left(\mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}\right)&=\frac{n_{21}}{(n_{21}+n_{22})}\\
\text{거짓 부정률}\left(\mathrm{FNR}=\frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}\right)&=\frac{n_{12}}{(n_{11}+n_{12})}\\
\text{참 긍정률}\left(\mathrm{TPR}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\right)&=\frac{n_{11}}{(n_{11}+n_{12})}\\
\text{참 부정률}\left(\mathrm{TNR}=\frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}\right)&=\frac{n_{22}}{(n_{21}+n_{22})}
\end{aligned}
\tag{1.2}
$$

(Precision) $$\text{정확률}=\frac{n_{11}}{(n_{11}+n_{21})}$$

(Recall) $$\text{재현율}=\frac{n_{11}}{(n_{11}+n_{12})}$$ (1.3)

(F measure) $$F_\beta=(1+\beta^2)\frac{\text{정확률}\times\text{재현율}}{\beta^2\times\text{정확률}+\text{재현율}}$$

$$F_1=\frac{2\times\text{정확률}\times\text{재현율}}{\text{정확률}+\text{재현율}}$$ (1.4)

성능 평가

예제 1-1 **얼굴 검출의 성능 측정**

[그림 1-10]은 세 개의 영상을 가진 데이터베이스에서 얼굴을 검출한 결과를 보여준다. 앞에서 다룬 정확률, 재현율, F_1 측정을 구해보자.

그림 1-10 얼굴 검출 성능

총 15개의 얼굴 중 12개를 옳게 검출했으므로 참 긍정 $n_{11}=12$, 세 개의 얼굴을 못 찾았으므로 거짓 부정 $n_{12}=3$, 그리고 얼굴 아닌 곳을 얼굴로 검출한 것이 두 개이므로 거짓 긍정 $n_{21}=2$이다. 따라서 정확률은 12/14이고 재현율은 12/15이다. F_1 측정은 24/29이다.

참 \ 분류	얼굴	얼굴아님
얼굴	12	3
얼굴아님	2	

$$\text{Precision} = \frac{12}{12+2} = 0.857$$

$$\text{Recall} = \frac{12}{12+3} = 0.800$$

$$\text{F1} = \frac{2 \times 0.857 \times 0.800}{0.867+0.800} = 0.827$$

HW

1. 교재 13장 Exercise : 1, 6, 7번 (문제풀이)

2. 실습 4 는 lenna 영상의 색상을 RGB 기준으로 15개로 줄이는 예제이다. HSI 변환의 H (색상) 값 기준으로 10 개로 줄이는 프로그램을 작성하라.