Machine Laerning
Part 1

Image Classification

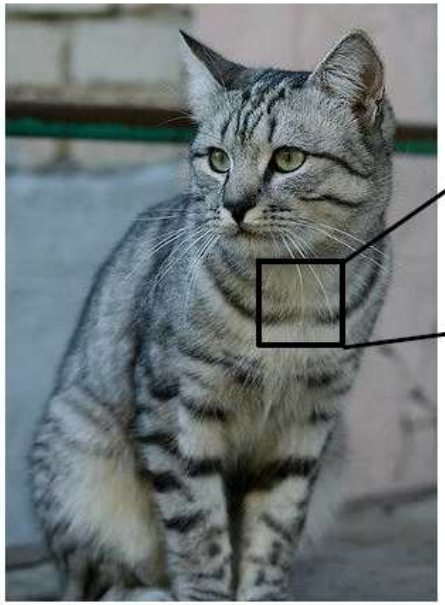
Image Classification: A core task in Computer Vision

(assume given a set of labels)
{dog, cat, truck, plane, ...}

cat

Image Classification

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 120 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 120  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  60  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

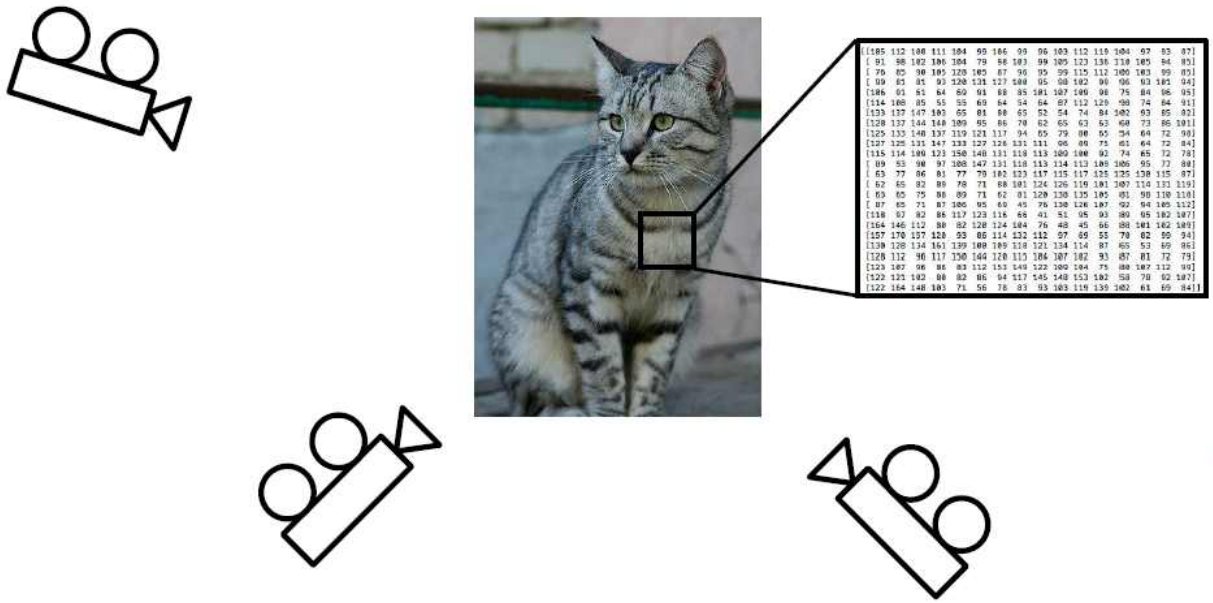
What the computer sees

An image is just a tensor of integers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Image Classification

Challenges: Viewpoint variation



All pixels change when the camera moves!

Image Classification

Challenges: Background Clutter

Image Classification

Challenges: Illumination

Image Classification

Challenges: Occlusion

Image Classification

Challenges: Deformation

Image Classification

Challenges: Intraclass variation

Image Classification

An image classifier

```
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Image Classification

Attempts have been made

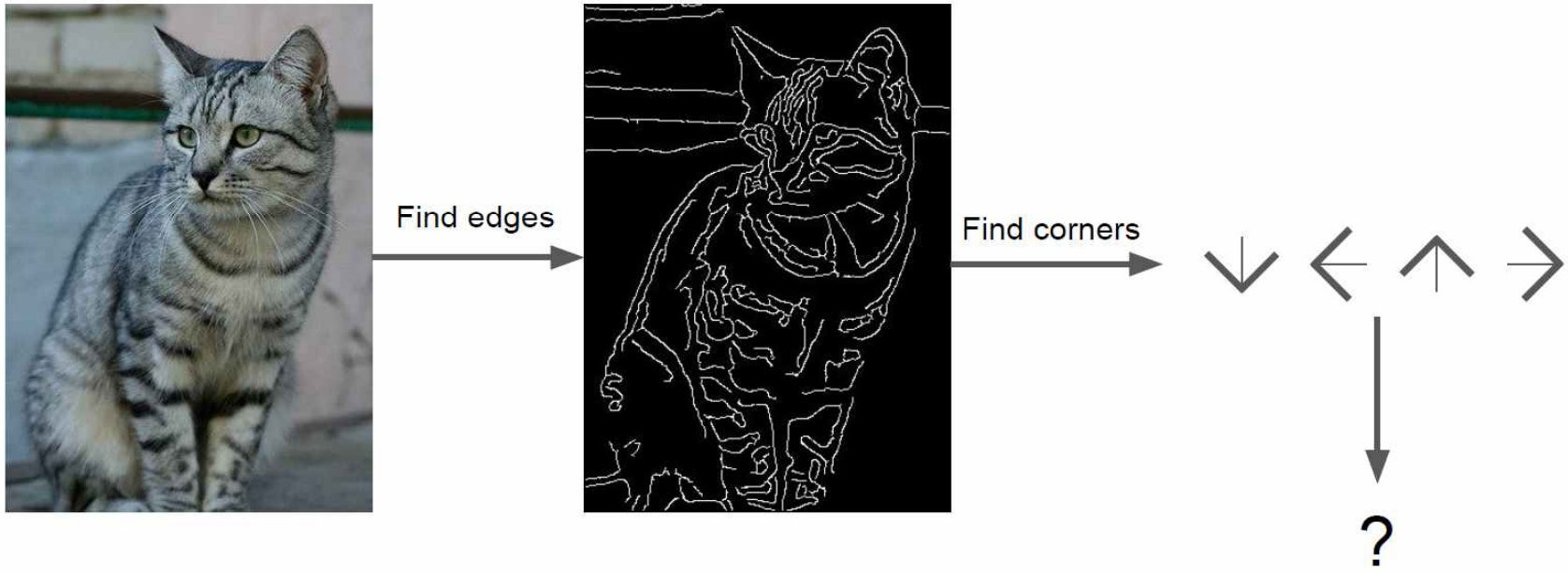


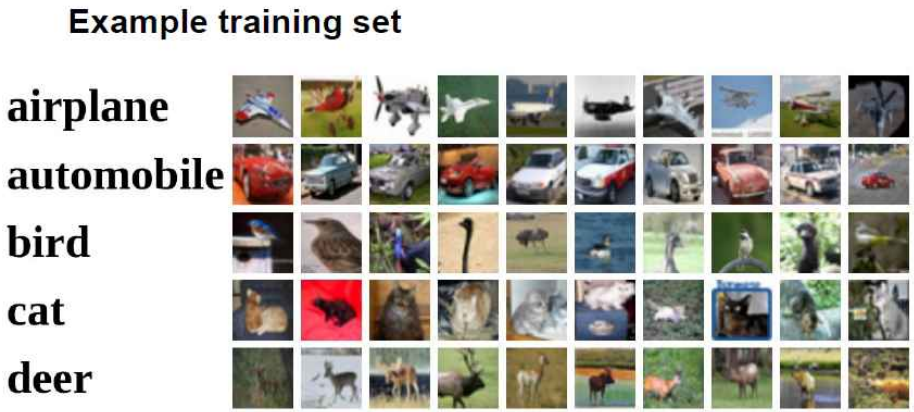
Image Classification

Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning algorithms to train a classifier
3. Evaluate the classifier on new images

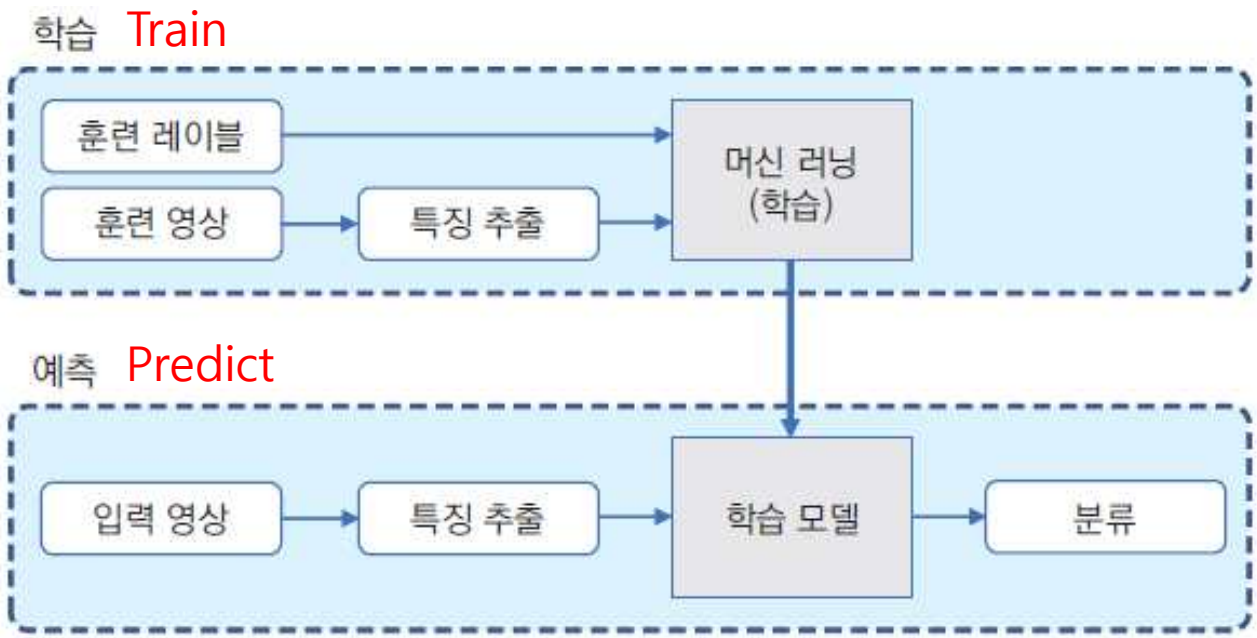
```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```



Machine Learning

- Train & Predict



Machine Learning

- **Regression** (회귀) vs. **Classification** (분류)
  - Regression: 연속적인 수치값 출력
    (ex) (키와 몸무게 상관관계) 키 입력 시 몸무게 예측
  - Classification: 이진 수치값 출력
    (ex) (과일 분류) 0 = 사과, 1 = 바나나

Machine Learning

- **Supervised** (지도학습) vs. **Unsupervised (**비지도학습)
  - Training data 에 정답(참값 or ground truth) 포함 유무

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

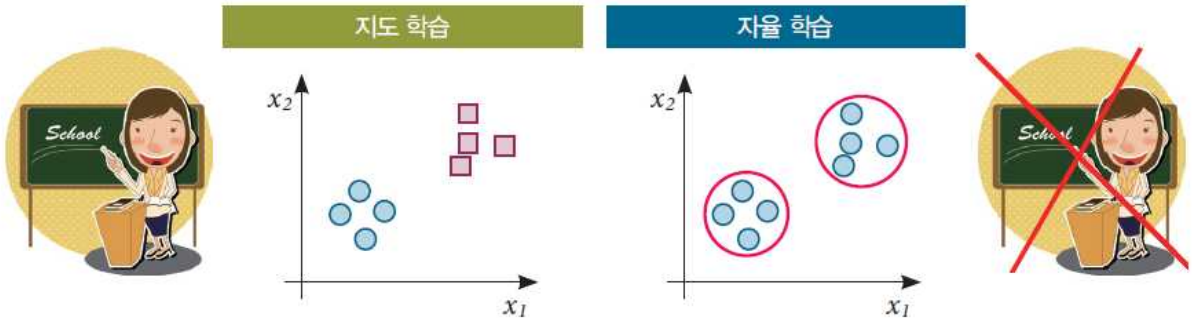
Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

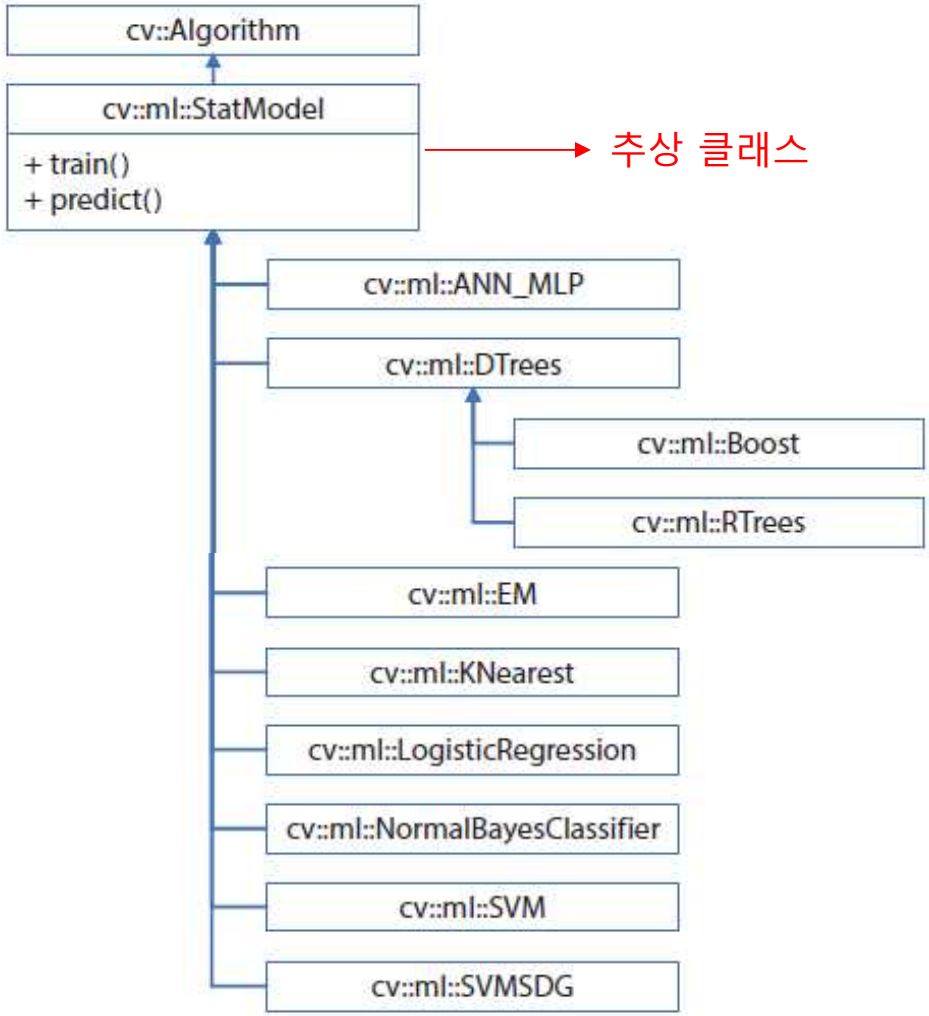
Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, density estimation, etc.



Machine Learning & OpenCV

- OpenCV Class



Machine Learning & OpenCV

- OpenCV Class

클래스 이름	설명
ANN_MLP	인공 신경망(artificial neural network) 다층 퍼셉트론(multi-layer perceptrons). 여러 개의 은닉층을 포함한 신경망을 학습시킬 수 있고, 입력 데이터에 대한 결과를 예측할 수 있습니다.
DTrees	이진 의사 결정 트리(decision trees) 알고리즘. DTrees 클래스는 다시 부스팅 알고리즘을 구현한 ml::Boost 클래스와 랜덤 트리(random tree) 알고리즘을 구현한 ml::RTree 클래스의 부모 클래스 역할을 합니다.
Boost	부스팅(boosting) 알고리즘. 다수의 약한 분류기(weak classifier)에 적절한 가중치를 부여하여 성능이 좋은 분류기를 만드는 방법입니다.
RTrees	랜덤 트리(random tree) 또는 랜덤 포레스트(random forest) 알고리즘. 입력 특징 벡터를 다수의 트리로 예측하고, 그 결과를 취합하여 분류 또는 회귀를 수행합니다.
EM	기댓값 최대화(Expectation Maximization). 가우시안 혼합 모델(Gaussian mixture model)을 이용한 군집화 알고리즘입니다.
KNearest	k 최근접 이웃(k-Nearest Neighbor) 알고리즘. k 최근접 이웃 알고리즘은 샘플 데이터와 인접한 k개의 훈련 데이터를 찾고, 이 중 가장 많은 개수에 해당하는 클래스를 샘플 데이터 클래스로 지정합니다.
LogisticRegression	로지스틱 회귀(logistic regression). 이진 분류 알고리즘의 일종입니다.
NormalBayesClassifier	정규 베이즈 분류기. 정규 베이즈 분류기는 각 클래스의 특징 벡터가 정규 분포를 따른다고 가정합니다. 따라서 전체 데이터 분포는 가우시안 혼합 모델로 표현 가능합니다. 정규 베이즈 분류기는 학습 데이터로부터 각 클래스의 평균 벡터와 공분산 행렬을 계산하고, 이를 예측에 사용합니다.

Machine Learning & OpenCV

- OpenCV Class

```
virtual bool StatModel::train(InputArray samples,
                              int layout,
                              InputArray responses);
```

- samples 훈련 데이터 행렬
- layout 훈련 데이터 배치 방법. ROW_SAMPLE 또는 COL_SAMPLE를 지정합니다.
- responses 각 훈련 데이터에 대응되는 응답(레이블) 행렬
- 반환값 정상적으로 학습이 완료되면 true를 반환합니다.

SampleTypes 열거형 상수	설명
ROW_SAMPLE	각 훈련 데이터가 samples 행렬에 행 단위로 저장되어 있습니다.
COL_SAMPLE	각 훈련 데이터가 samples 행렬에 열 단위로 저장되어 있습니다.

Machine Learning & OpenCV

- OpenCV Class

```
virtual float StatModel::predict(InputArray samples,
                                 OutputArray results = noArray(),
                                 int flags = 0) const;
```

- samples 입력 벡터가 행 단위로 저장된 행렬. CV_32F
- results 각 입력 샘플에 대한 예측 결과가 저장된 행렬
- flags 추가적인 플래그 상수. StatModel::Flags 열거형 상수 중 하나를 지정할 수 있으며, 모델에 따라 사용법이 다릅니다.
- 반환값 입력 벡터가 하나인 경우에 대한 응답이 반환됩니다.

K-Nearest Neighbor

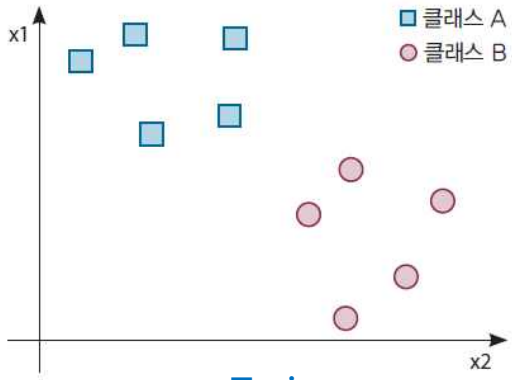
- Algorithm
  - K 개의 최근접 이웃을 찾아 다수의 이웃이 소속된 클래스로 분류함

```
def train(images, labels):
  # Machine learning!
  return model
```

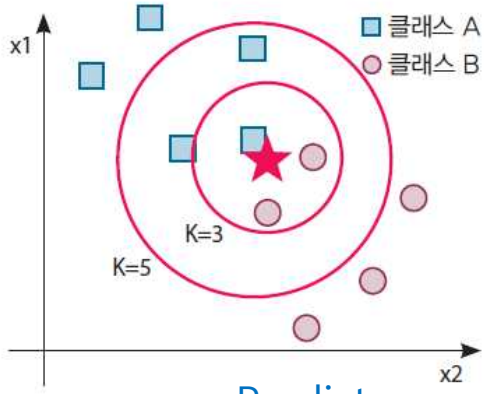
→ Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

→ Predict the label of the most similar training image



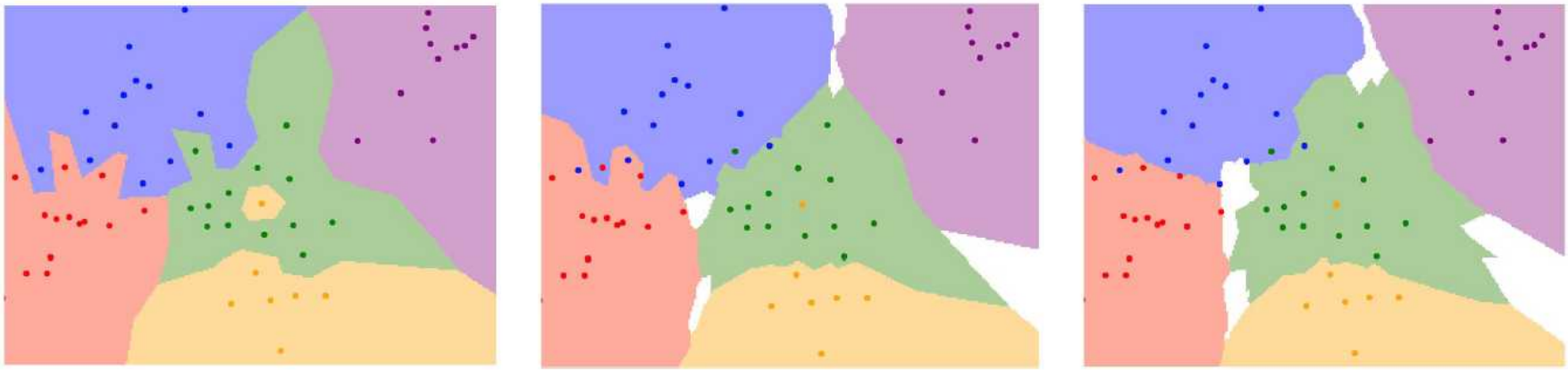
Train



Predict

K-Nearest Neighbor

- Algorithm
  - Take majority vote (다수결 투표) from K closest points



K = 1

(Template Matching)

K = 3

K = 5

K-Nearest Neighbor

- Distance metric

Training data with labels

query data

Distance Metric $\left|\right.$  ,  $\left|\right. \rightarrow \mathbb{R}$

K-Nearest Neighbor

- Distance metric

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image			
56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image			
10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences			
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

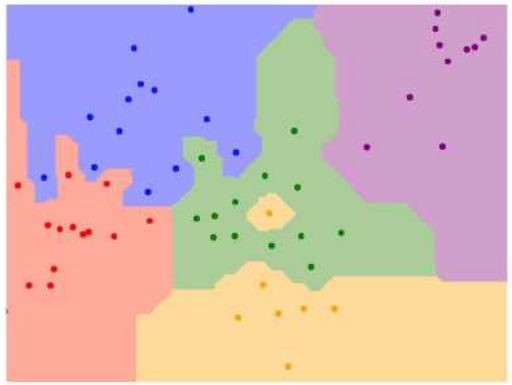
add → 456

K-Nearest Neighbor

- Distance Metric

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K = 1

K-Nearest Neighbor

- Hyperparameters
  - K
  - Distance

- Demo
  - http://vision.stanford.edu/teaching/cs231n-demos/knn/

K-Nearest Neighbor

- Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

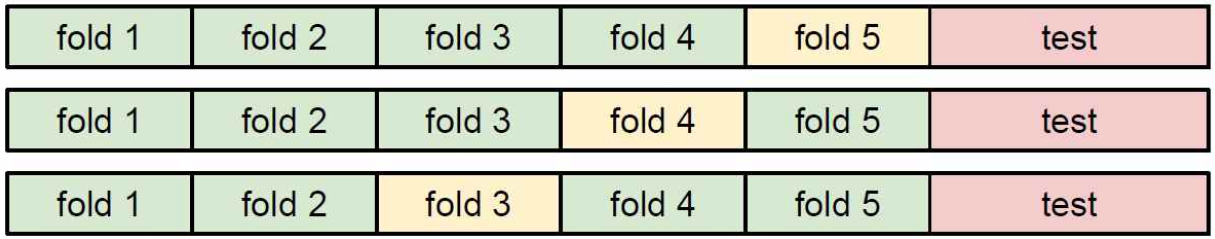
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



Useful for small datasets, but not used too frequently in deep learning

K-Nearest Neighbor

- OpenCV

```
static Ptr<KNearest> KNearest::create();
```

- 반환값 KNearest 객체를 참조하는 Ptr 스마트 포인터 객체

```
virtual void KNearest::setDefaultK(int val);
```

- val kNN 알고리즘에서 사용할 k 값. StatModel::predict() 함수를 사용할 경우 미리 k 값을 적절하게 설정해야 합니다. (default: k=10)

```
virtual void KNearest::setIsClassifier(bool val);
```

- val 이 값이 true이면 분류로 사용하고, false이면 회귀로 사용합니다.

K-Nearest Neighbor

- OpenCV

```
virtual float KNearest::findNearest(InputArray samples,
                                    int k,
                                    OutputArray results,
                                    OutputArray neighborResponses = noArray(),
                                    OutputArray dist = noArray()) const;
```

• samples	테스트 데이터 벡터가 행 단위로 저장된 행렬. 입력 벡터의 차원은 훈련 벡터의 차원과 같아야 하며, 행렬 타입은 CV_32FC1이어야 합니다.
• k	사용할 최근접 이웃 개수. 1보다 같거나 커야 합니다.
• results	각 입력 샘플에 대한 예측(분류 또는 회귀) 결과를 저장한 행렬. samples.rows×1 크기를 갖고, 타입은 CV_32FC1입니다.
• neighborResponses	예측에 사용된 k개의 최근접 이웃 클래스 정보를 담고 있는 행렬. samples.rows×k 크기를 갖고, 타입은 CV_32FC1입니다.
• dist	입력 벡터와 예측에 사용된 k개의 최근접 이웃과의 거리를 저장한 행렬. samples.rows×k 크기를 갖고, 타입은 CV_32FC1입니다.
• 반환값	입력 벡터가 하나인 경우에 대한 응답이 반환됩니다.

StatModel::predict() 함수 보다 많은 정보 반환

K-Nearest Neighbor

코드 15-1 kNN 알고리즘을 이용한 2차원 점 분류 [ch15/knnplane]

```
01  #include "opencv2/opencv.hpp"
02  #include <iostream>
03
04  using namespace cv;
05  using namespace cv::ml;
06  using namespace std;
07
08  Mat img;
09  Mat train, label;
10  Ptr<KNearest> knn;
11  int k_value = 1;
12
13  void on_k_changed(int, void*);
14  void addPoint(const Point& pt, int cls);
15  void trainAndDisplay();
16
17  int main(void)
18  {
19      img = Mat::zeros(Size(500, 500), CV_8UC3);
20      knn = KNearest::create();
21
22      namedWindow("knn");
23      createTrackbar("k", "knn", &k_value, 5, on_k_changed);
24
25      const int NUM = 30;
26      Mat rn(NUM, 2, CV_32SC1);
27
28      randn(rn, 0, 50);
29      for (int i = 0; i < NUM; i++)
30          addPoint(Point(rn.at<int>(i, 0) + 150, rn.at<int>(i, 1) + 150), 0);
31
32      randn(rn, 0, 50);
33      for (int i = 0; i < NUM; i++)
34          addPoint(Point(rn.at<int>(i, 0) + 350, rn.at<int>(i, 1) + 150), 1);
35
36      randn(rn, 0, 70);
37      for (int i = 0; i < NUM; i++)
38          addPoint(Point(rn.at<int>(i, 0) + 250, rn.at<int>(i, 1) + 400), 2);
39
40      trainAndDisplay();
41
42      waitKey();
43      return 0;
44  }
45
46  void on_k_changed(int, void*)
47  {
48      if (k_value < 1) k_value = 1;
49      trainAndDisplay();
50  }
51
```

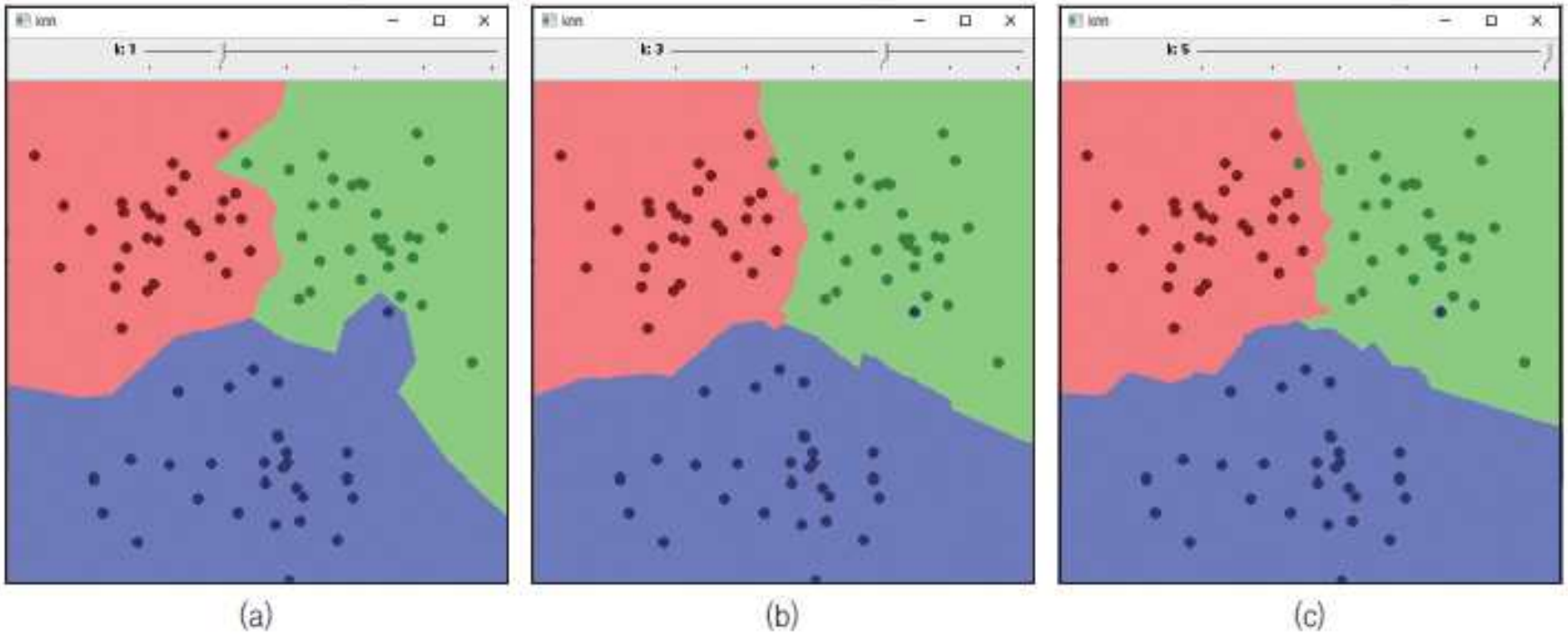
K-Nearest Neighbor

```
52  void addPoint(const Point& pt, int cls)
53  {
54      Mat new_sample = (Mat_<float>(1, 2) << pt.x, pt.y);
55      train.push_back(new_sample);
56
57      Mat new_label = (Mat_<int>(1, 1) << cls);
58      label.push_back(new_label);
59  }
60
61  void trainAndDisplay()
62  {
63      knn->train(train, ROW_SAMPLE, label);
64
65      for (int i = 0; i < img.rows; ++i) {
66          for (int j = 0; j < img.cols; ++j) {
67              Mat sample = (Mat_<float>(1, 2) << j, i);
68
69              Mat res;
70              knn->findNearest(sample, k_value, res);
71
72              int response = cvRound(res.at<float>(0, 0));
73              if (response == 0)
74                  img.at<Vec3b>(i, j) = Vec3b(128, 128, 255); // R
75              else if (response == 1)
76                  img.at<Vec3b>(i, j) = Vec3b(128, 255, 128); // G
77              else if (response == 2)
78                  img.at<Vec3b>(i, j) = Vec3b(255, 128, 128); // B
79          }
80      }
```

```
81
82      for (int i = 0; i < train.rows; i++)
83      {
84          int x = cvRound(train.at<float>(i, 0));
85          int y = cvRound(train.at<float>(i, 1));
86          int l = label.at<int>(i, 0);
87
88          if (l == 0)
89              circle(img, Point(x, y), 5, Scalar(0, 0, 128), -1, LINE_AA);
90          else if (l == 1)
91              circle(img, Point(x, y), 5, Scalar(0, 128, 0), -1, LINE_AA);
92          else if (l == 2)
93              circle(img, Point(x, y), 5, Scalar(128, 0, 0), -1, LINE_AA);
94      }
95
96      imshow("knn", img);
97  }
```

K-Nearest Neighbor

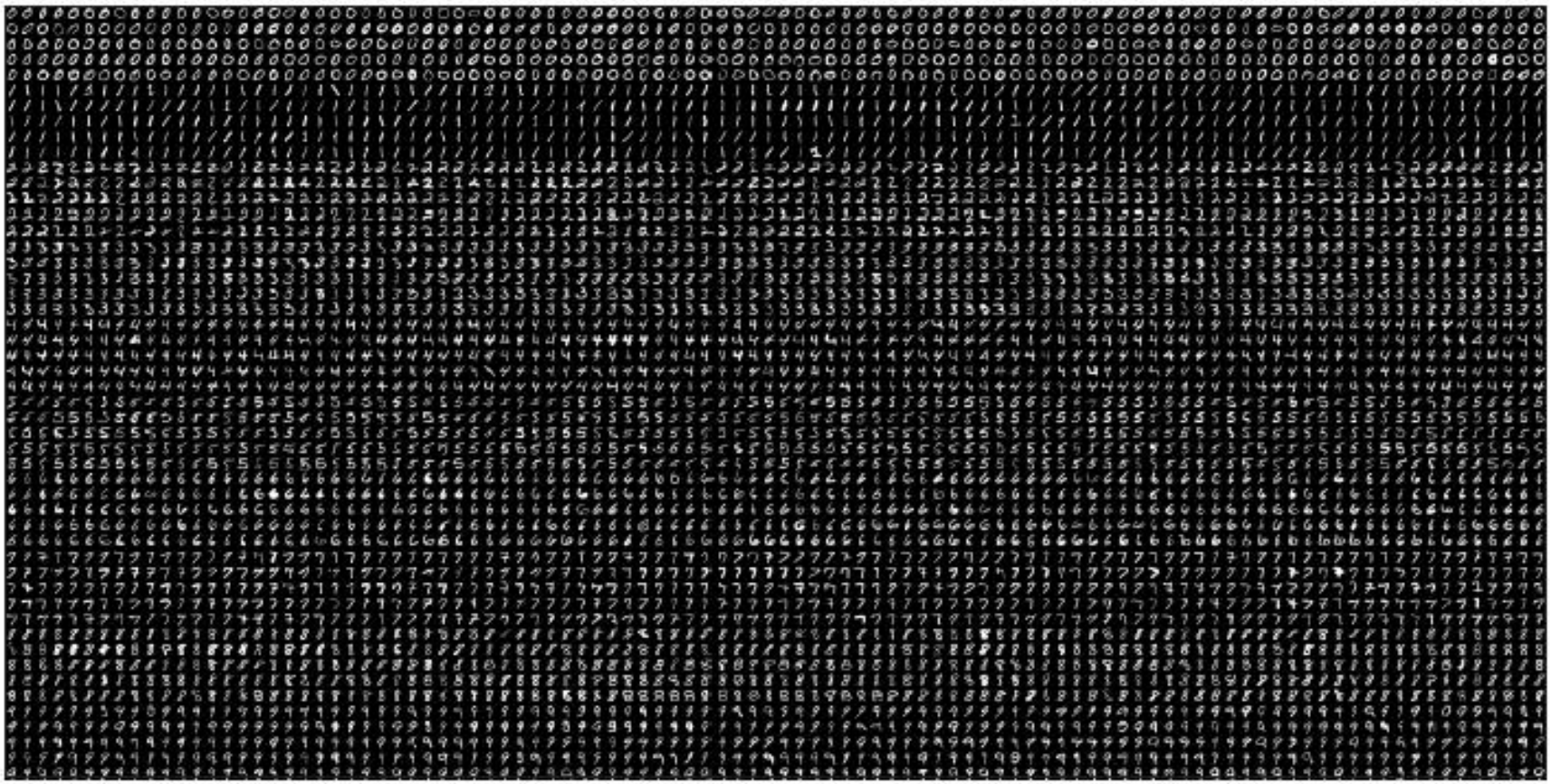
- 출력화면



(a) (b) (c)

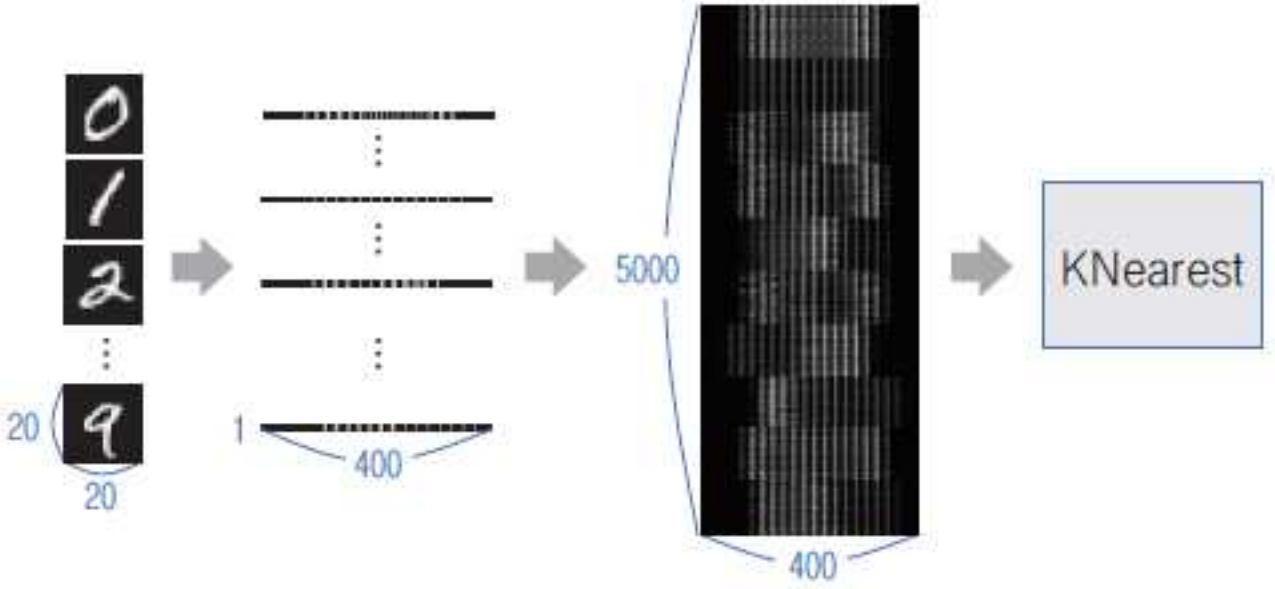
K-Nearest Neighbor

- kNN 을 이용한 필기체 숫자인식
  - Digits.png



K-Nearest Neighbor

- kNN 을 이용한 필기체 숫자인식



K-Nearest Neighbor

코드 15-2 kNN 알고리즘을 이용한 필기체 숫자 학습 [ch15/knndigits]

```
01  Ptr<KNearest> train_knn()
02  {
03      Mat digits = imread("digits.png", IMREAD_GRAYSCALE);
04
05      if (digits.empty()) {
06          cerr << "Image load failed!" << endl;
07          return 0;
08      }
09
10      Mat train_images, train_labels;
11
12      for (int j = 0; j < 50; j++) {
13          for (int i = 0; i < 100; i++) {
14              Mat roi, roi_float, roi_flatten;
15              roi = digits(Rect(i * 20, j * 20, 20, 20));
16              roi.convertTo(roi_float, CV_32F);
17              roi_flatten = roi_float.reshape(1, 1);
18
19              train_images.push_back(roi_flatten);
20              train_labels.push_back(j / 5);
21          }
22      }
23
24      Ptr<KNearest> knn = KNearest::create();
25      knn->train(train_images, ROW_SAMPLE, train_labels);
26
27      return knn;
28  }
```

K-Nearest Neighbor

코드 15-3 마우스로 숫자 그리기 [ch15/knndigits]

```
01  Point ptPrev(-1, -1);
02
03  void on_mouse(int event, int x, int y, int flags, void* userdata)
04  {
05      Mat img = *(Mat*)userdata;
06
07      if (event == EVENT_LBUTTONDOWN) {
08          ptPrev = Point(x, y);
09      } else if (event == EVENT_LBUTTONUP) {
10          ptPrev = Point(-1, -1);
11      } else if (event == EVENT_MOUSEMOVE && (flags & EVENT_FLAG_LBUTTON)) {
12          line(img, ptPrev, Point(x, y), Scalar::all(255), 40, LINE_AA, 0);
13          ptPrev = Point(x, y);
14
15          imshow("img", img);
16      }
17  }
```

K-Nearest Neighbor

코드 15-4 KNearest 클래스를 이용한 필기체 숫자 인식 [ch15/knndigits]

```
11  int main()
12  {
13      Ptr<KNearest> knn = train_knn();
14
15      if (knn.empty()) {
16          cerr << "Training failed!" << endl;
17          return -1;
18      }
19
20      Mat img = Mat::zeros(400, 400, CV_8U);
21
22      imshow("img", img);
23      setMouseCallback("img", on_mouse, (void*)&img);
24
25      while (true) {
26          int c = waitKey(0);
27
28          if (c == 27) {
29              break;
30          } else if (c == ' ') {
31              Mat img_resize, img_float, img_flatten, res;
32
33              resize(img, img_resize, Size(20, 20), 0, 0, INTER_AREA);
34              img_resize.convertTo(img_float, CV_32F);
35              img_flatten = img_float.reshape(1, 1);
36
37              knn->findNearest(img_flatten, 3, res);
38              cout << cvRound(res.at<float>(0, 0)) << endl;
39
40              img.setTo(0);
41              imshow("img", img);
42          }
43      }
44
45      return 0;
46  }
```

K-Nearest Neighbor