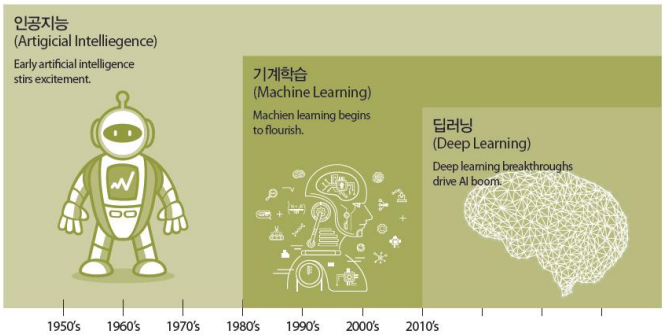
14장 기계학습과 딥러닝

1

기계학습

- 기계 학습(machine learning)은 인공지능의 한 분야로, 컴퓨터에 학습 기능을 부여하기 위한 연구 분야이다.



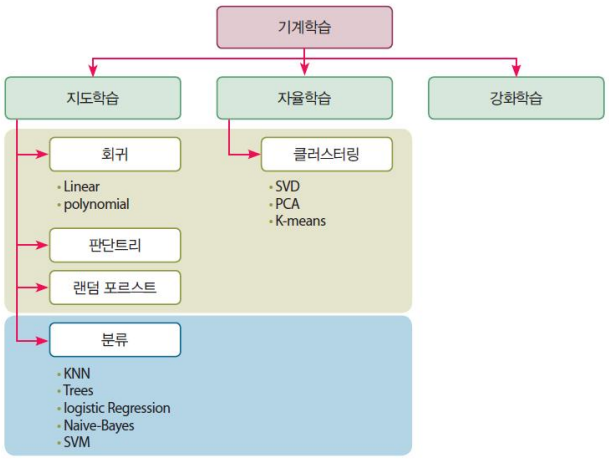
 © 인피니티북스 2019

2

기계 학습이 중요하게 사용되는 분야

3

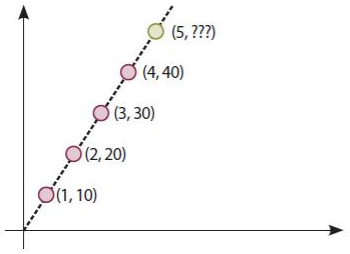
기계학습의 분류



4

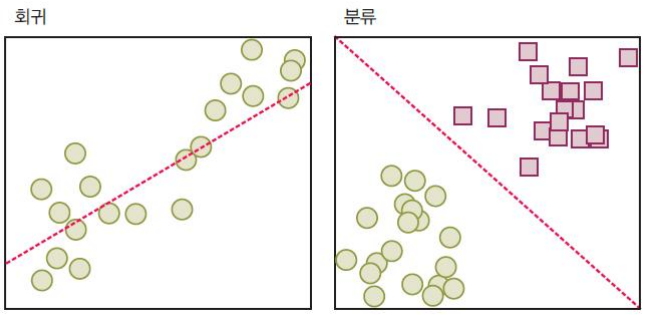
지도학습

- 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것이다



5

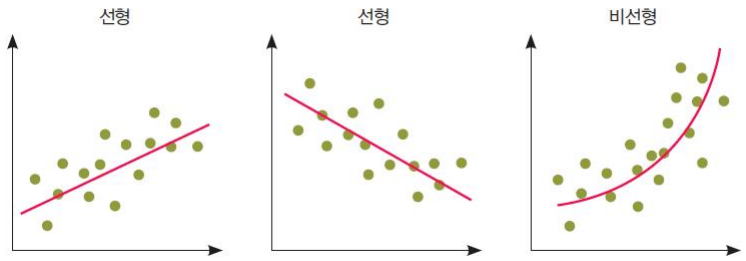
회귀regression 와 분류



6

회귀

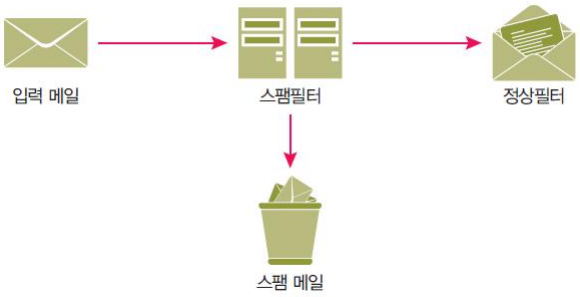
- 회귀regression 란 일반적으로 데이터들을 2차원 공간에 찍은 후에 이들 데이터들을 가장 잘 설명하는 직선이나 곡선을 찾는 문제



7

분류

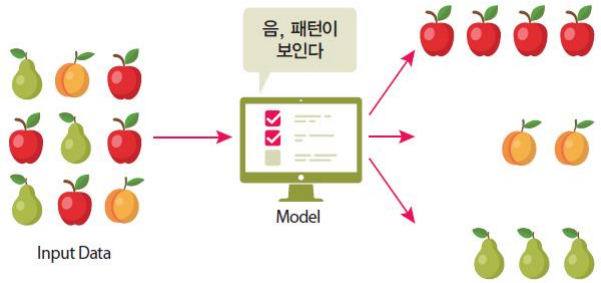
- 식 $y = f(x)$에서 출력 y가 이산적discrete 인 경우에 이것을 분류 classification 문제(또는 인식 문제)라고 부른다.



8

자율학습

- "교사" 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다.
- 식 $y = f(x)$에서 레이블 y가 주어지지 않는 것이다.

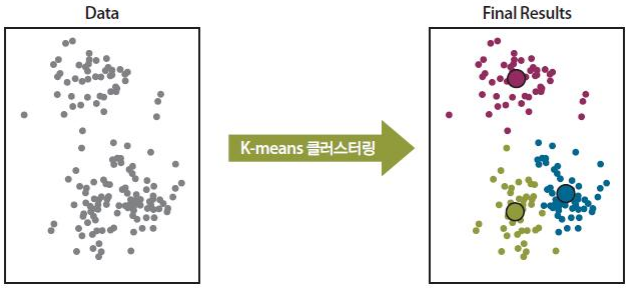


 © 인피니티북스 2019

9

자율학습

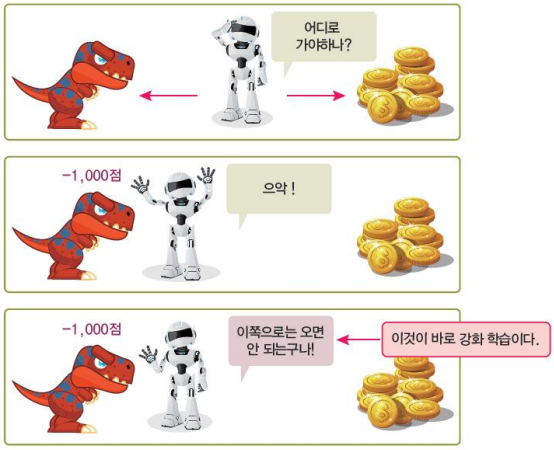
- 가장 대표적인 자율 학습이 클러스터링clustering (군집화)이다.
- K-means 클러스터링 알고리즘



 © 인피니티북스 2019

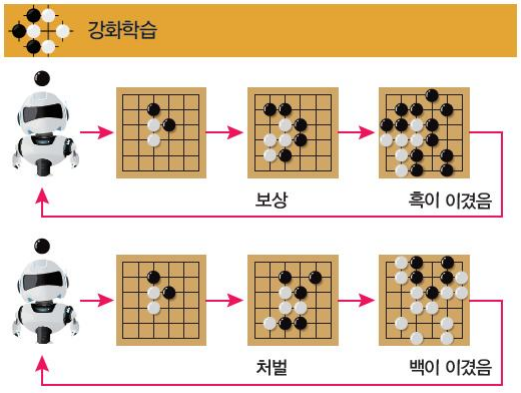
10

강화학습



11

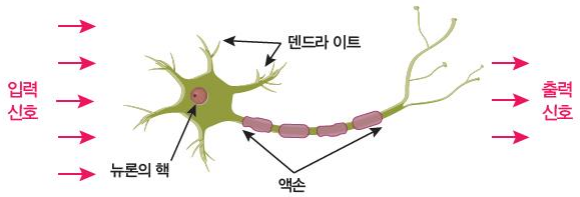
강화학습



12

신경회로망

- 인공신경망 neural network 은 1950년대부터 연구되어 온 연구 주제였다. "생각하는 기계"는 항상 인간의 꿈이었고 사람들은 인간의 두뇌를 본떠서 기계로 만들려는 시도



생물학적 뉴론의 구조

13

퍼셉트론

$$y = \begin{cases} 1 & (w_1x_1 + w_2x_2 \geq b) \\ 0 & (w_1x_1 + w_2x_2 < b) \end{cases}$$

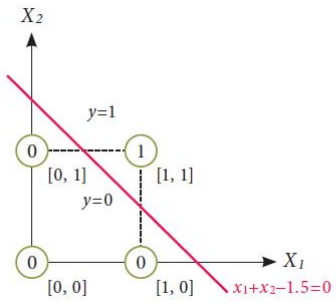
14

논리적인 AND 학습

x_1	x_2	$w_1x_1 + w_2x_2$	b	출력
0	0	0×1+0×1=0	1.5	0
1	0	1×1+0×1=1	1.5	0
0	1	0×1+1×1=1	1.5	0
1	1	1×1+1×1=2	1.5	1

15

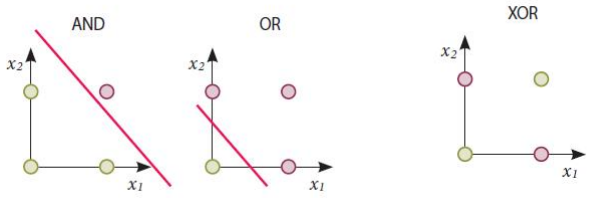
논리적인 AND 학습



16

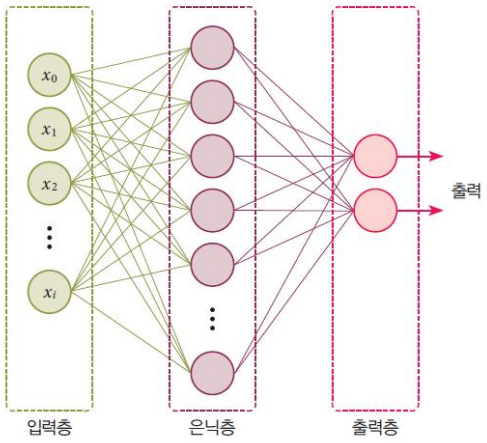
퍼셉트론의 문제점

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



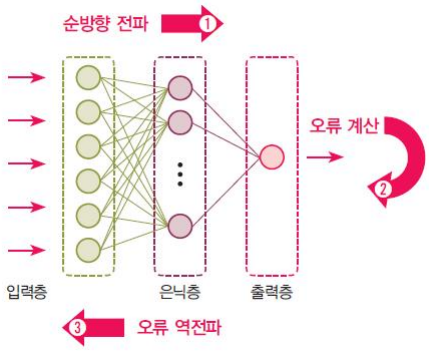
17

다층 퍼셉트론



18

역전파 알고리즘



19

역전파 알고리즘

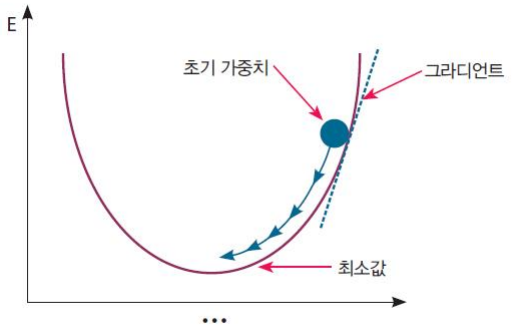
Algorithm 14.1

인공신경망의 가중치를 작은 난수로 초기화한다.
    do 각 학습샘플 sample에 대하여 다음을 반복한다.
        prediction = calculate_network(sample) // 순방향 패스
        actual = desired_output(sample)
        각 출력 노드에서 오류(prediction-actual)를 계산한다.
        은익층에서 출력층으로의 가중치 변경값을 계산한다. // 역방향 패스
        입력층에서 은닉층으로의 가중치 변경값을 계산한다. // 역방향 패스
        전체 가중치를 업데이트한다.
until 모든 샘플이 올바르게 분류될 때까지

20

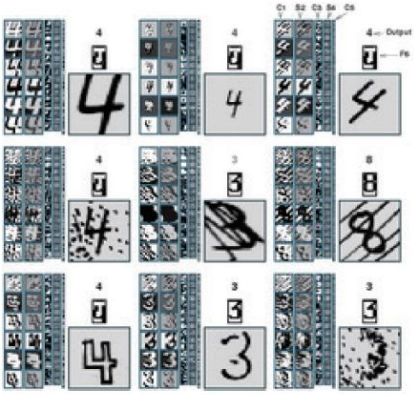
그라디언트 강하 방법

$$E = \frac{1}{2}(t-y)^2$$



21

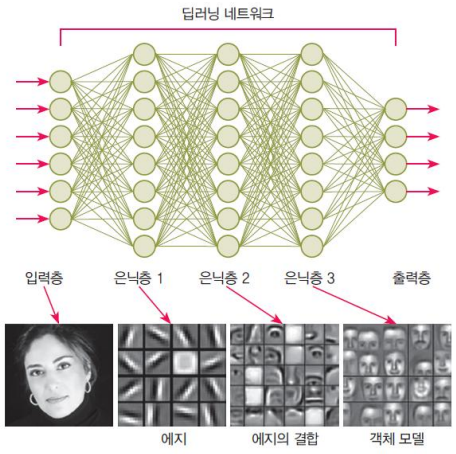
인공신경망



출처: http://yann.lecun.com/ex/research/

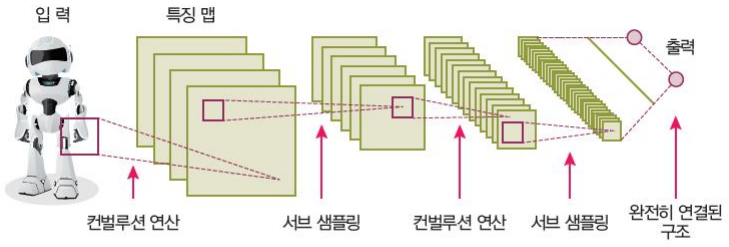
22

딥러닝



23

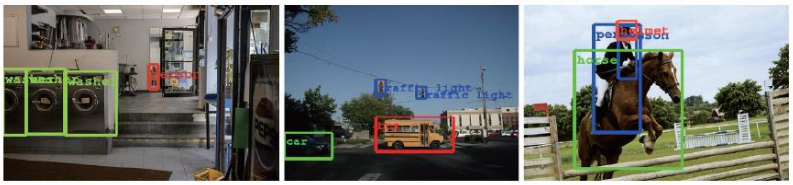
컨벌루션 신경망



*출처: 위키미디어

24

컨볼루션 신경망



출처: Google research

25

XOR 학습시키기

train_features

0.0	0.0
1.0	0.0
0.0	1.0
1.1	1.1

0.0
1.0
1.0
0.0

labels

26

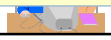
```
int main()
{
	const int hiddenLayerSize = 4;
	float trainingInput[4][2] = {
		{ 0.0, 0.0 },
	{ 0.0, 1.0 },
	{ 1.0, 0.0 },
	{ 1.0, 1.0 }
	};
	Mat trainingInputData = Mat(4, 2, CV_32F, trainingInput);

	float trainingOutput[4][1] = {
		{ 0.0 },
	{ 1.0 },
	{ 1.0 },
	{ 0.0 }
	};
	Mat trainingOutputData = Mat(4, 1, CV_32F, trainingOutput);
```


27

```
	Ptr<ANN_MLP> mlp = ANN_MLP::create();

	Mat layersSize = Mat(3, 1, CV_16U);
	layersSize.row(0) = Scalar(trainingInputData.cols);
	layersSize.row(1) = Scalar(hiddenLayerSize);
	layersSize.row(2) = Scalar(trainingOutputData.cols);

	mlp->setLayerSizes(layersSize);
	mlp-
>setActivationFunction(ANN_MLP::ActivationFunctions::SIGMOID_SYM);

	TermCriteria term = TermCriteria(
		TermCriteria::Type::COUNT + TermCriteria::Type::EPS,
		100000000,
		0.000000000000000001
	);
	mlp->setTermCriteria(term);
	mlp->setTrainMethod(ANN_MLP::TrainingMethods::BACKPROP);
```


28

```
        Ptr<TrainData> trainingData = TrainData::create(
                trainingInputData,
                SampleTypes::ROW_SAMPLE,
                trainingOutputData
        );

        mlp->train(trainingData);

        for (int i = 0; i < trainingInputData.rows; i++) {
                Mat sample = Mat(1, trainingInputData.cols, CV_32F,
trainingInput[i]);
                Mat result;
                mlp->predict(sample, result);
                cout << sample << " -> ";
                cout << result << endl;
        }

        return 0;
}
```

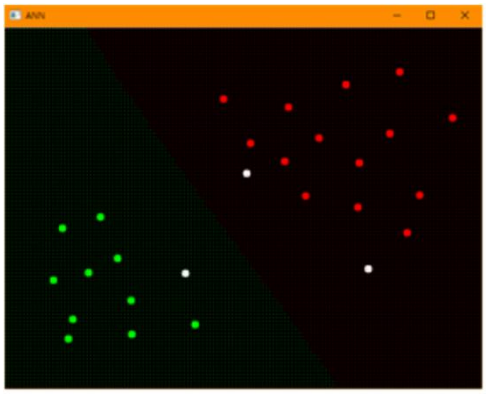



29

실행결과

30

2차원 점들의 분류

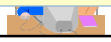


31

```
const Scalar WHITE_COLOR = Scalar(255, 255, 255);
const string winName = "points";
const int testStep = 5;

Mat img, imgDst;
RNG rng;

vector<Point>  trainedPoints;         // 학습된 점들이 저장된다.
vector<int>    trainedPointsMarkers; // 학습된 점들이 클래스가 저장된다.
const int MAX_CLASSES = 2;
vector<Vec3b>  classColors(MAX_CLASSES);
int currentClass = 0;
vector<int> classCounters(MAX_CLASSES);
...
```

32

```
static void on_mouse(int event, int x, int y, int /*flags*/, void*)
{
        if (img.empty())            return;

        int updateFlag = 0;
        // 버튼이 눌리면 현재 위치를 벡터에 저장한다.
        if (event == EVENT_LBUTTONUP)    {
                trainedPoints.push_back(Point(x, y));
                trainedPointsMarkers.push_back(currentClass);
                classCounters[currentClass]++;
                updateFlag = true;
        }
        // 점을 영상위에 그린다.
        if (updateFlag)    {
                img = Scalar::all(0);
                for (size_t i = 0; i < trainedPoints.size(); i++)          {
                        Vec3b c = classColors[trainedPointsMarkers[i]];
                        circle(img, trainedPoints[i], 5, Scalar(c), -1);
                }
                imshow(winName, img);
        }
}
```

33

```
// 입력된 점들을 행벡터 형식의 샘플 데이터로 만든다.
static Mat prepare_train_samples(const vector<Point>& pts)
{
        Mat samples;
        Mat(pts).reshape(1, (int)pts.size()).convertTo(samples, CV_32F);
        return samples;
}

// 입력된 점들과 점들의 레이블을 묶어서 학습 데이터를 생성한다.
static Ptr<TrainData> prepare_train_data()
{
        Mat samples = prepare_train_samples(trainedPoints);
        return TrainData::create(samples, ROW_SAMPLE,
Mat(trainedPointsMarkers));
}
```


34

```
// 영상 안의 모든 위치를 생성하여서 테스트 데이터로 사용한다.
// 예측된 레이블을 화면에 색상으로 표시한다.
static void predict_and_paint(const Ptr<StatModel>& model, Mat& dst)
{
	Mat testSample(1, 2, CV_32FC1);
	for (int y = 0; y < img.rows; y += testStep)  {
		for (int x = 0; x < img.cols; x += testStep)		{
			testSample.at<float>(0) = (float)x;
			testSample.at<float>(1) = (float)y;

			int response = (int)model->predict(testSample);
			dst.at<Vec3b>(y, x) = classColors[response];
		}
	}
}
```

35

```
// 인공신경망을 사용하는 코드
#if _ANN_
static void find_decision_boundary_ANN(const Mat&  layer_sizes)
{
	// 학습 데이터에 대한 레이블을 0으로 초기화한다.
	Mat trainClasses = Mat::zeros((int)trainedPoints.size(), (int)classColors.size(),
CV_32FC1);

	// trainedPointsMarkers[i]의 값이 0이면 첫 번째 화소가 1.0이 된다.
	// trainedPointsMarkers[i]의 값이 1이면 두 번째 화소가 1.0이 된다.
	for (int i = 0; i < trainClasses.rows; i++) {
		trainClasses.at<float>(i, trainedPointsMarkers[i]) = 1.f;
	}

	Mat samples = prepare_train_samples(trainedPoints);
	Ptr<TrainData> tdata = TrainData::create(samples, ROW_SAMPLE, trainClasses);

	Ptr<ANN_MLP> ann = ANN_MLP::create();
	ann->setLayerSizes(layer_sizes);
	ann->setActivationFunction(ANN_MLP::SIGMOID_SYM, 1, 1);
	ann->setTermCriteria(TermCriteria(TermCriteria::MAX_ITER + TermCriteria::EPS,
300, FLT_EPSILON));
	ann->setTrainMethod(ANN_MLP::BACKPROP, 0.001);
	ann->train(tdata);
	predict_and_paint(ann, imgDst);
}
#endif
```

36

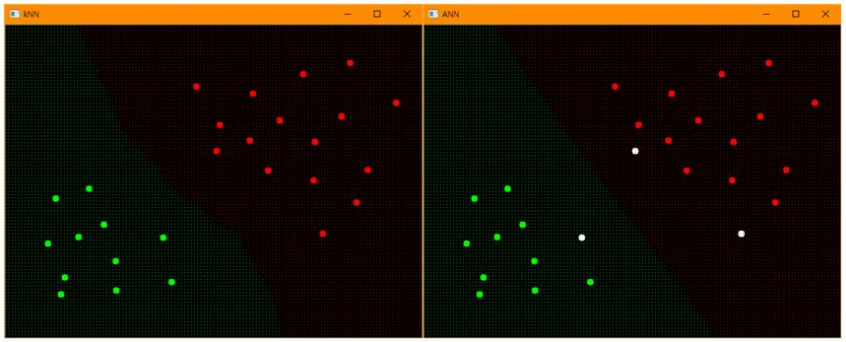
```
int main()
{
	cout << "Use:" << endl
		<< "  key '0' .. '1' - switch to class #n" << endl
		<< "  left mouse button - to add new point;" << endl
		<< "  key 'r' - to run the ML model;" << endl
		<< "  key 'i' - to init (clear) the data." << endl << endl;

	cv::namedWindow("points", 1);
	img.create(480, 640, CV_8UC3);
	imgDst.create(480, 640, CV_8UC3);

	imshow("points", img);
	setMouseCallback("points", on_mouse);
```


```
	classColors[0] = Vec3b(0, 255, 0);
	classColors[1] = Vec3b(0, 0, 255);

	for (;;)  {
		char key = (char)waitKey();
		if (key == 27) break;
		if (key == 'i') {
			img = Scalar::all(0);
			trainedPoints.clear();
			trainedPointsMarkers.clear();
			classCounters.assign(MAX_CLASSES, 0);
			imshow(winName, img);
		}

		if (key == '0' || key == '1') {
			currentClass = key - '0';
		}
```

```
                    if (key == 'r') {
                            double minVal = 0;
                            minMaxLoc(classCounters, &minVal, 0, 0, 0);
                            if (minVal == 0)    {
                                    printf("each class should have at least 1
point\n");
                                    continue;
                            }
                            img.copyTo(imgDst);
#if _ANN_
                            Mat layer_sizes1(1, 3, CV_32SC1);
                            layer_sizes1.at<int>(0) = 2;            // 입력층의 개수: 2
                            layer_sizes1.at<int>(1) = 5;            // 은닉층의 개수: 5
                            layer_sizes1.at<int>(2) = (int)classColors.size(); // 출력
층의 개수: 2
                            find_decision_boundary_ANN(layer_sizes1);
                            imshow("ANN", imgDst);
#endif

                    }
            }

            return 0;
}
```

39

실행결과



40

Q & A