**A/B 테스트(A/B testing)**

둘 이상의 기법을 통계적으로 비교하는 방법으로서, 일반적으로 기존 기법과 새로운 기법을 서로 비교합니다. A/B 테스트의 목표는 더 우수한 기법을 찾는 것뿐만 아니라 그 차이가 통계적 유의성을 갖는지 여부를 파악하는 것입니다. A/B 테스트에서는 일반적으로 단일 측정항목을 사용하여 두 기법을 비교하지만, 적용 가능한 기법 및 측정항목의 수에는 유한성의 범위 내에서 제한이 없습니다.

**정확성(accuracy)**

[**분류 모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#classification_model)의 예측이 얼마나 정확한지를 의미합니다. [**다중 클래스 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#multi-class)에서 정확성의 정의는 다음과 같습니다.

정확성 = 정확한예측 / 총예시수

[**이진 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification)에서 정확성의 정의는 다음과 같습니다.

정확성 = ( 참양성 + 참음성 ) / 총예시수

[**참양성**](https://developers.google.com/machine-learning/glossary/?hl=ko#TP) 및 [**참음성**](https://developers.google.com/machine-learning/glossary/?hl=ko#TN)을 참조하세요.

**활성화 함수(activation function)**

이전 레이어의 모든 입력에 대한 가중 합을 취하고 출력 값(일반적으로 비선형)을 생성하여 다음 레이어로 전달하는 [**ReLU**](https://developers.google.com/machine-learning/glossary/?hl=ko#ReLU), [**시그모이드**](https://developers.google.com/machine-learning/glossary/?hl=ko#sigmoid_function) 등의 함수입니다.

**AdaGrad**

각 매개변수의 경사를 재조정하여 사실상 각 매개변수에 독립적인 [**학습률**](https://developers.google.com/machine-learning/glossary/?hl=ko#learning_rate)을 부여하는 정교한 경사하강법 알고리즘입니다. 자세한 설명은 [이 논문](http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf)을 참조하세요.

**AUC(ROC 곡선 아래 영역)**

가능한 모든 [**분류 임계값**](https://developers.google.com/machine-learning/glossary/?hl=ko#classification_threshold)을 고려하는 평가 측정항목입니다.

AUC([ROC 곡선](https://developers.google.com/machine-learning/glossary/?hl=ko#ROC) 아래 영역)는 무작위로 선택한 긍정 예가 실제로 긍정일 가능성이 무작위로 선택한 부정 예가 긍정일 가능성보다 높다고 분류자가 신뢰할 확률입니다.

**B**

**역전파(backpropagation)**

[**신경망**](https://developers.google.com/machine-learning/glossary/?hl=ko#neural_network)에서 [**경사하강법**](https://developers.google.com/machine-learning/glossary/?hl=ko#gradient_descent)을 수행하는 기본 알고리즘입니다. 우선, 정방향 단계에서 각 노드의 출력 값을 계산하고 캐시합니다. 그런 다음 역방향 단계에서 그래프를 통과하며 각 매개변수를 기준으로 오차의 [편미분](https://en.wikipedia.org/wiki/Partial_derivative)을 계산합니다.

**기준(baseline)**

모델의 성능을 비교하는 참조 지점으로 사용되는 단순한 [**모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#model) 또는 휴리스틱입니다. 기준은 모델 개발자가 특정 문제에 예상되는 최소 성능을 산정하는 데 도움을 줍니다.

**배치(batch)**

[**모델 학습**](https://developers.google.com/machine-learning/glossary/?hl=ko#model_training)의 [**반복**](https://developers.google.com/machine-learning/glossary/?hl=ko#iteration) 1회, 즉 [**경사**](https://developers.google.com/machine-learning/glossary/?hl=ko#gradient) 업데이트 1회에 사용되는 예의 집합입니다.

[**배치 크기**](https://developers.google.com/machine-learning/glossary/?hl=ko#batch_size)를 참조하세요.

**배치 크기(batch size)**

[**배치**](https://developers.google.com/machine-learning/glossary/?hl=ko#batch) 하나에 포함되는 예의 개수입니다. 예를 들어 [**SGD**](https://developers.google.com/machine-learning/glossary/?hl=ko#SGD)의 배치 크기는 1이고, [**미니 배치**](https://developers.google.com/machine-learning/glossary/?hl=ko#mini-batch)의 배치 크기는 일반적으로 10~1,000입니다. 학습 및 추론 중에 배치 크기는 일반적으로 고정되지만, 텐서플로우는 동적 배치 크기를 허용합니다.

**편향(bias)**

원점을 기준으로 한 절편 또는 오프셋입니다. 편향 또는 **바이어스 항**은 머신러닝 모델에서 *b* 또는 *w0*으로 표현됩니다. 예를 들어 다음 수식에서 편향은 *b*입니다.

[**예측 편향**](https://developers.google.com/machine-learning/glossary/?hl=ko#prediction_bias)과 혼동하지 마시기 바랍니다.

**이진 분류(binary classification)**

상호 배타적인 두 클래스 중 하나를 출력하는 분류 작업 유형입니다. 예를 들어 이메일 메시지를 평가하고 '스팸' 또는 '스팸 아님'을 출력하는 머신러닝 모델은 이진 분류자입니다.

**비닝(binning)**

[**버케팅**](https://developers.google.com/machine-learning/glossary/?hl=ko#bucketing)을 참조하세요.

**버케팅(bucketing)**

하나의 특성(일반적으로 [**연속**](https://developers.google.com/machine-learning/glossary/?hl=ko#continuous_feature))을 버킷(bucket) 또는 빈(bin)이라고 하는 여러 이진 특성으로 변환하는 작업으로서, 일반적으로 값 범위를 기준으로 합니다. 예를 들어 온도를 하나의 부동 소수점 연속 특성으로 표현하는 대신 온도 범위를 불연속 빈으로 나눌 수 있습니다. 민감도가 1/10도인 온도 데이터가 있다면 0.0~15.0도 범위의 모든 온도를 1번 빈에, 15.1~30.0도 범위를 2번 빈에, 30.1~50.0도 범위를 3번 빈에 넣을 수 있습니다.

**C**

**캘리브레이션 레이어(calibration layer)**

일반적으로 [**예측 편향**](https://developers.google.com/machine-learning/glossary/?hl=ko#prediction_bias)을 보정하기 위한 예측 후 조정입니다. 조정된 예측 및 확률은 관찰된 라벨 집합의 분포와 일치해야 합니다.

**후보 샘플링(candidate sampling)**

학습 도중 소프트맥스등을 사용하여 모든 긍정 라벨에 대한 확률을 계산하는 최적화입니다. 부정 라벨의 경우 무작위 샘플에 대해서만 계산합니다. 예를 들어 라벨이 *beagle* 및 *dog*인 예가 있으면 후보 샘플링에서 *beagle* 및 *dog* 클래스 출력에 대해 예측되는 확률과 해당 손실 항을 계산하고 나머지 클래스(*cat*, *lollipop*, *fence*)의 무작위 부분집합에 대해서도 계산합니다. 이 방식이 성립하는 이유는 [**포지티브 클래스**](https://developers.google.com/machine-learning/glossary/?hl=ko#positive_class)가 항상 적절한 긍정 강화를 받는 한 [**네거티브 클래스**](https://developers.google.com/machine-learning/glossary/?hl=ko#negative_class)는 빈도가 적은 부정 강화로부터 학습할 수 있기 때문이며, 이는 실제로 경험적으로 관찰되는 사실입니다. 후보 샘플링의 장점은 모든 부정에 대한 예측을 일일이 계산하지 않으므로 연산 효율이 높다는 것입니다.

**범주형 데이터(categorical data)**

가능한 값의 불연속 집합을 갖는 [**특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#feature)입니다. 예를 들어 Tudor, ranch, colonial이라는 3가지 가능한 값으로 이루어진 불연속 집합을 갖는 house style이라는 범주형 특성이 있다고 가정해 보겠습니다. house style을 범주형 데이터로 표현하면 Tudor, ranch, colonial이 주택 가격에 주는 영향을 모델이 개별적으로 학습할 수 있습니다.

불연속 집합의 값은 상호 배타적일 수 있으며, 특정 예에 하나의 값만 적용할 수도 있습니다. 예를 들어 car maker 범주형 특성은 예마다 하나의 값(Toyota)만 허용할 가능성이 높습니다. 둘 이상의 값을 적용할 수 있는 경우도 있습니다. 자동차 하나를 여러 색으로 도색할 수도 있으므로 car color 범주형 특성은 하나의 예가 여러 값(예: red와 white)을 갖도록 허용할 수 있습니다.

범주형 특성을 [**불연속 특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#discrete_feature)이라고도 합니다.

[**수치 데이터**](https://developers.google.com/machine-learning/glossary/?hl=ko#numerical_data)와 대비되는 개념입니다.

**중심(centroid)**

[**k-평균**](https://developers.google.com/machine-learning/glossary/?hl=ko#k-means) 또는 [**k-중앙값**](https://developers.google.com/machine-learning/glossary/?hl=ko#k-median) 알고리즘에 의해 결정되는 클러스터의 중심입니다. 예를 들어 k가 3인 경우 k-평균 또는 k-중앙값 알고리즘에서는 3개의 중심을 찾아냅니다.

**체크포인트(checkpoint)**

특정 시점에 모델 변수의 상태를 포착한 데이터입니다. 체크포인트를 통해 모델 [**가중치**](https://developers.google.com/machine-learning/glossary/?hl=ko#weight)를 내보내고 여러 세션(session)을 넘나들며 학습을 수행할 수 있습니다. 또한 체크포인트로부터 과거의 오류(예: 작업 선점)를 이어받아 학습을 할 수 있습니다. 단, [**그래프**](https://developers.google.com/machine-learning/glossary/?hl=ko#graph) 자체는 체크포인트에 포함되지 않습니다.

**클래스(class)**

열거형 목표값 집합 중 하나로서 라벨로 쓰입니다. 예를 들어 스팸을 감지하는 [**이진 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification) 모델의 두 클래스는 *스팸* 및 *스팸 아님*입니다. 개의 품종을 식별하는 [**다중 클래스 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#multi_class_classification) 모델의 클래스는 *푸들*, *비글*, *퍼그* 등입니다.

**클래스 불균형 데이터 세트(class-imbalanced data set)**

두 클래스의 [**라벨**](https://developers.google.com/machine-learning/glossary/?hl=ko#label)이 서로 크게 다른 빈도를 보이는 [**이진 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification) 문제입니다. 예를 들어 질병 데이터 세트에서 0.0001의 예가 긍정 라벨을 가지고 0.9999의 예가 부정 라벨을 가진다면 클래스 불균형 문제에 해당하지만, 축구 시합 예측에서 예 중 0.51은 한 팀이 이길 것으로, 0.49는 다른 팀이 이길 것으로 라벨이 지정되었다면 클래스 불균형 문제가 *아닙니다*.

**분류 모델(classification model)**

둘 이상의 불연속 클래스를 구분 짓는 데 사용되는 머신러닝 모델 유형입니다. 예를 들어 자연어 처리 분류 모델은 입력 문장이 프랑스어인지, 스페인어인지, 이탈리아어인지 구분할 수 있습니다. [**회귀 모형**](https://developers.google.com/machine-learning/glossary/?hl=ko#regression_model)과 비교되는 개념입니다.

**분류 임계값(classification threshold)**

[**포지티브 클래스**](https://developers.google.com/machine-learning/glossary/?hl=ko#positive_class)와 [**네거티브 클래스**](https://developers.google.com/machine-learning/glossary/?hl=ko#negative_class)를 구분 짓기 위한 모델의 예측 점수에 적용되는 스칼라값 기준입니다. [**로지스틱 회귀**](https://developers.google.com/machine-learning/glossary/?hl=ko#logistic_regression) 결과를 [**이진 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification)에 매핑하는 데 사용됩니다. 예를 들어 특정 이메일 메시지가 스팸일 확률을 판단하는 로지스틱 회귀 모형이 있다고 가정해 보겠습니다. 분류 임계값이 0.9인 경우 로지스틱 회귀 값이 0.9를 넘으면 *스팸*으로, 0.9 미만이면 *스팸 아님*으로 분류됩니다.

**클러스터링(clustering)**

[**예**](https://developers.google.com/machine-learning/glossary/?hl=ko#example)와 관련된 그룹화이며 특히 [**비지도 학습**](https://developers.google.com/machine-learning/glossary/?hl=ko#unsupervised_machine_learning)에서 사용됩니다. 모든 예가 그룹으로 묶이고 나면 사람이 선택적으로 각 클러스터에 의미를 부여할 수 있습니다.

클러스터링에는 여러 가지 알고리즘이 사용됩니다. 예를 들어 [**k-평균**](https://developers.google.com/machine-learning/glossary/?hl=ko#k-means) 알고리즘에서는 다음 다이어그램과 같이 각 예의 [**중심**](https://developers.google.com/machine-learning/glossary/?hl=ko#centroid) 근접도를 기준으로 클러스터링합니다.

그런 다음 연구원이 클러스터를 검토하고 클러스터 1에는 '난쟁이 나무', 클러스터 2에는 '완전한 크기의 나무'와 같이 이름을 붙입니다.

다음에서 확인할 수 있는 것처럼 중심점에서 예가 얼마나 떨어져 있는지를 바탕으로 한 클러스터링 알고리즘도 있을 수 있습니다.

**협업 필터링(collaborative filtering)**

여러 다른 사용자의 관심분야를 기반으로 특정 사용자의 관심분야를 예측하는 방식입니다. 협업 필터링은 추천 시스템에 자주 사용됩니다.

**혼동행렬(confusion matrix)**

[**분류 모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#classification_model)의 예측 성공률, 즉 라벨과 모델의 분류 사이의 상관관계를 요약한 NxN 표입니다. 혼동행렬의 축 중 하나는 모델이 예측한 라벨이고, 다른 축은 실제 라벨입니다. N은 클래스 수를 나타냅니다. [**이진 분류**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification) 문제에서는 N=2입니다. 예를 들어 다음은 이진 분류 문제에 대한 샘플 혼동행렬입니다.

|  | **종양(예측)** | **비종양(예측)** |
| --- | --- | --- |
| 종양(실제) | 18 | 1 |
| 비종양(실제) | 6 | 452 |

위 혼동행렬에서는 실제로 종양이 있었던 샘플 19개 중 18개는 모델이 정확히 분류(참긍정 18개)했고, 1개는 종양이 없는 것으로 잘못 분류(거짓부정 1개)했습니다. 마찬가지로, 실제로 종양이 없었던 샘플 458개 중 452개는 정확히 분류(참음성 452개)되었고 6개는 잘못 분류(거짓양성 6개)되었습니다.

다중 클래스 분류 문제인 경우 혼동행렬로 착오 패턴을 파악할 수 있습니다. 예를 들어 혼동행렬은 필기 숫자를 인식하도록 학습된 모델이 4를 9로, 아니면 7을 1로 잘못 예측하는 경향이 있음을 드러낼 수 있습니다.

혼동행렬은 [**정밀도**](https://developers.google.com/machine-learning/glossary/?hl=ko#precision), [**재현율**](https://developers.google.com/machine-learning/glossary/?hl=ko#recall) 등의 다양한 성능 측정항목을 계산하는 데 충분한 정보를 포함합니다.

**연속 특성(continuous feature)**

가능한 값이 무한한 범위를 갖는 부동 소수점 특성입니다. [**불연속 특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#discrete_feature)과 대비되는 개념입니다.

**수렴(convergence)**

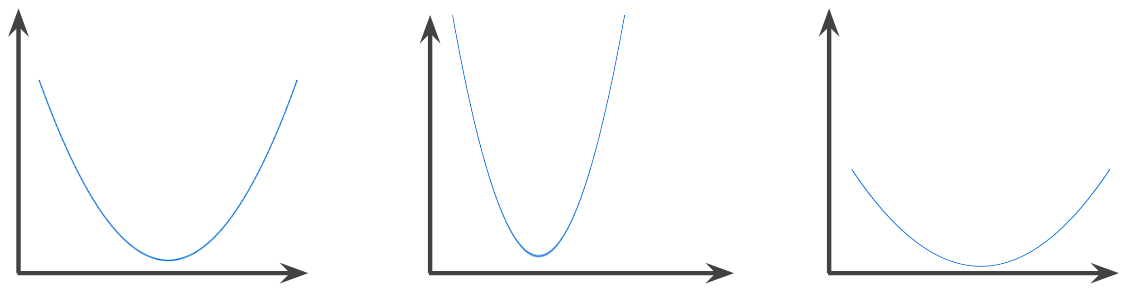
학습 중에 어느 정도 이상의 반복 후 학습 [**손실**](https://developers.google.com/machine-learning/glossary/?hl=ko#loss) 및 검증 손실이 거의 또는 전혀 변화하지 않는 상태를 비공식적으로 일컫는 용어입니다. 즉, 현재 데이터로 더 학습해도 모델이 개선되지 못할 때 모델이 수렴되었다고 말합니다. 딥 러닝의 경우 손실 값이 거의 일정하게 유지하며 수많은 반복을 하여 일시적으로 모델이 수렴한다고 착각을 할 수 있으나 결국 하강을 하는 경우도 있습니다.

[**조기 중단**](https://developers.google.com/machine-learning/glossary/?hl=ko#early_stopping)을 참조하세요.

Boyd와 Vandenberghe의 [볼록 최적화](https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)를 참조하세요.

**볼록 함수(convex function)**

함수 그래프의 위쪽 영역이 [**볼록 집합**](https://developers.google.com/machine-learning/glossary/?hl=ko#convex_set)인 함수입니다. 볼록 함수의 전형적인 예는 **U**자 모양의 함수입니다. 예를 들어 다음은 모두 볼록 함수입니다.



그러나 다음 함수는 볼록 함수가 아닙니다. 그래프의 위쪽 영역이 볼록 집합이 아닌 것을 볼 수 있습니다.

**순볼록(strictly convex) 함수**는 국소 최저점이 정확히 하나이며, 이 점은 전역 최저점과 일치합니다. 고전적인 U자형 함수는 순볼록 함수입니다. 그러나 직선과 같은 볼록 함수는 순볼록 함수가 아닙니다.

일반적인 [**손실 함수**](https://developers.google.com/machine-learning/glossary/?hl=ko#loss_functions) 중 다음을 비롯한 다수는 볼록 함수입니다.

* [**L2 손실(L2 loss)**](https://developers.google.com/machine-learning/glossary/?hl=ko#L2_loss)
* [**로그 손실(Log Loss)**](https://developers.google.com/machine-learning/glossary/?hl=ko#Log_Loss)
* [**L1 정규화(L1 regularization)**](https://developers.google.com/machine-learning/glossary/?hl=ko#L1_regularization)
* [**L2 정규화(L2 regularization)**](https://developers.google.com/machine-learning/glossary/?hl=ko#L2_regularization)

[**경사하강법**](https://developers.google.com/machine-learning/glossary/?hl=ko#gradient_descent)의 여러 가지 변형은 순볼록 함수의 최저점  
에 가까운 점을 찾도록 보장합니다. 마찬가지로, [**확률적 경사하강법**](https://developers.google.com/machine-learning/glossary/?hl=ko#SGD)의 여러 변형은 순볼록 함수의 최저점에 가까운 점을 찾을 가능성이 높지만 항상 보장되지는 않습니다.

두 볼록 함수의 합(예: L2 손실 + L1 정규화)은 볼록 함수입니다.

[**심층 모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#deep_model)은 어떠한 경우에도 볼록 함수가 아닙니다. 그럼에도 불구하고 [**볼록 최적화**](https://developers.google.com/machine-learning/glossary/?hl=ko#convex_optimization)를 위해 설계된 알고리즘은 심층 네트워크에서 비교적 양호한 해를 구할 가능성이 높지만, 이러한 해가 전역 최저점이라는 보장은 없습니다.

**볼록 최적화(convex optimization)**

[**경사하강법**](https://developers.google.com/machine-learning/glossary/?hl=ko#gradient_descent) 등의 수학적 기법을 사용하여 [**볼록 함수**](https://developers.google.com/machine-learning/glossary/?hl=ko#convex_function)의 최저점을 찾는 과정입니다. 머신러닝에 대한 연구 중 상당한 비중이 볼록 최적화와 같은 다양한 문제를 고안하고 효과적인 해법을 찾는 데 집중되었습니다.

자세한 내용은 Boyd와 Vandenberghe의 [볼록 최적화](https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)를 참조하세요.

**볼록 집합(convex set)**

유클리드 공간에서 부분집합에 속한 임의의 두 점을 잇는 선이 해당 부분집합에 완전히 포함되는 성질을 갖는 부분집합입니다. 예를 들어 다음 두 도형은 볼록 집합입니다.



그러나 다음 두 도형은 볼록 집합이 아닙니다.



**컨볼루션(convolution)**

수학적으로 간단히 말하면 두 가지 함수가 섞인 것입니다. 머신러닝에서 컨볼루션은 가중치를 학습시키기 위해 컨볼루셔널 필터와 입력 행렬을 혼합합니다.

머신러닝에서 '컨볼루션'이라는 용어는 종종 [**컨볼루셔널 연산**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_operation) 또는 [**컨볼루셔널 레이어**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_layer)를 짧게 지칭할 때 사용됩니다.

컨볼루션이 없으면 머신러닝 알고리즘이 큰 텐서의 모든 셀에 있어서 별도의 가중치를 학습해야 합니다. 예를 들어 2,000x2,000 크기의 이미지를 학습하는 머신러닝 알고리즘은 4백만 개의 개별적인 가중치를 찾아야 됩니다. 컨볼루션이 있기 때문에 머신러닝 알고리즘은 [**컨볼루셔널 필터**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_filter)에 있는 모든 셀의 가중치만 찾아도 되고, 이로 인해 모델 학습에 필요한 메모리가 크게 줄어듭니다. 컨볼루셔널 필터가 적용되는 경우 모든 셀에 같은 필터가 적용되며, 각 셀에 필터가 곱해집니다.

**컨볼루셔널 필터(convolutional filter)**

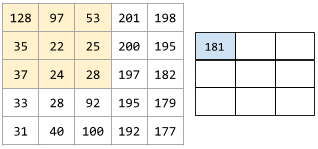
[**컨볼루셔널 연산**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_operation)에서 사용되는 두 가지 중 하나입니다. 다른 하나는 입력 행렬의 슬라이스입니다. 컨볼루셔널 필터는 입력 행렬과 [**순위(차원 수)**](https://developers.google.com/machine-learning/glossary/?hl=ko#rank)는 동일하지만 모양은 더 작은 행렬입니다. 예를 들어 입력 행렬이 28x28인 경우 컨볼루셔널 필터는 이보다 작은 2차원 행렬이 됩니다.

사진 조작에서 사용되는 컨볼루셔널 필터는 일반적으로 1과 0으로 구성된 일정한 패턴으로 설정됩니다. 머신러닝에서 컨볼루셔널 필터는 일반적으로 난수로 채워지며 네트워크가 이상적인 값을 학습시킵니다.

**컨볼루셔널 레이어(convolutional layer)**

심층신경망의 한 레이어로, 입력 행렬에 [**컨볼루셔널 필터**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_filter)를 적용합니다. 예를 들어 다음과 같은 3x3 [**컨볼루셔널 필터**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_filter)가 있다고 생각해 보세요.

다음의 애니메이션은 5x5 입력 행렬을 가지고 있고 9개의 컨볼루셔널 연산으로 구성되는 컨볼루셔널 레이어를 보여줍니다. 각 컨볼루셔널 연산은 입력 행렬의 서로 다른 3x3 슬라이스에서 이루어집니다. 그 결과 생성되는 3x3 행렬(오른쪽)은 9개의 컨볼루셔널 연산의 결과로 구성됩니다.



**컨볼루셔널 신경망(convolutional neural network)**

적어도 하나의 레이어가 [**컨볼루셔널 레이어**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_layer)인 신경망입니다. 일반적으로 컨볼루셔널 신경망은 다음과 같은 레이어의 조합으로 구성됩니다.

* 컨볼루셔널 레이어
* 풀링 레이어
* 밀집 레이어

컨볼루셔널 신경망은 이미지 인식과 같은 특정 종류의 문제에서 큰 성공을 거두었습니다.

**컨볼루셔널 연산(convolutional operation)**

컨볼루셔널 연산은 다음과 같은 2단계 수학 연산입니다.

1. [**컨볼루셔널 필터**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_filter) 및 입력 행렬의 슬라이스 등 요소별 곱셈 (입력 행렬의 슬라이스는 컨벌루셔널 필터와 순위 및 크기가 동일함)
2. 곱셈의 결과로 얻어지는 행렬 내 모든 값의 합계

다음과 같은 5x5 입력 행렬을 예로 들어보겠습니다.

이제 다음과 같은 2x2 컨볼루셔널 필터가 있다고 생각해 보세요.

각 컨볼루셔널 연산은 입력 행렬의 단일 2x2 슬라이스와 연관됩니다. 예를 들어 입력 행렬의 왼쪽 상단에 있는 2x2 슬라이스를 사용해 보겠습니다. 이 슬라이스의 컨볼루션 연산은 다음과 같습니다.

[**컨볼루셔널 레이어**](https://developers.google.com/machine-learning/glossary/?hl=ko#convolutional_layer)는 일련의 컨볼루셔널 연산으로 이루어지며, 각 연산은 입력 행렬의 서로 다른 슬라이스에 적용됩니다.

**비용(cost)**

[**손실**](https://developers.google.com/machine-learning/glossary/?hl=ko#loss)의 동의어입니다.

**교차 엔트로피(cross-entropy)**

[**다중 클래스 분류 문제**](https://developers.google.com/machine-learning/glossary/?hl=ko#multi-class)로 일반화한 [**로그 손실**](https://developers.google.com/machine-learning/glossary/?hl=ko#Log_Loss)입니다. 교차 엔트로피는 두 확률 분포 간의 차이를 계량합니다. [**퍼플렉시티**](https://developers.google.com/machine-learning/glossary/?hl=ko#perplexity)를 참조하세요.

**맞춤 에스티메이터(custom Estimator)**

[이러한 방향](https://www.tensorflow.org/extend/estimators?hl=ko)을 따라 직접 작성하는 [**에스티메이터**](https://developers.google.com/machine-learning/glossary/?hl=ko#Estimators)입니다.

[**사전 제작된 에스티메이터**](https://developers.google.com/machine-learning/glossary/?hl=ko#pre-made_Estimator)와 대비되는 개념입니다.

**D**

**데이터 분석(data analysis)**

데이터 분석이란 샘플, 측정치, 시각화를 고려하여 데이터를 이해하는 작업입니다. 처음으로 데이터 세트를 받은 직후, 모델을 개발하기 전에 특히 데이터 분석이 유용합니다. 또한 실험을 이해하고 시스템의 문제를 디버깅하는 데에도 중요합니다.

**DataFrame**

Pandas에서 데이터 세트를 표현하는 데 널리 사용되는 데이터 유형입니다. DataFrame은 표와 비슷합니다. DataFrame의 각 열에는 이름(헤더)이 있으며 각 행은 숫자로 식별됩니다.

**데이터 세트(data set)**

[**예시**](https://developers.google.com/machine-learning/glossary/?hl=ko#example)를 모아 놓은 집합입니다.

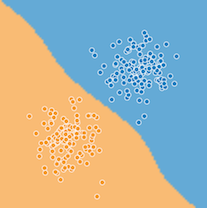
**Dataset API(tf.data)**

데이터를 읽고 머신러닝 알고리즘이 요구하는 형태로 변환하는 상위레벨의 TensorFlow API입니다. tf.data.Dataset 개체는 일련의 요소를 나타내며, 각 요소는 하나 이상의 [**텐서**](https://developers.google.com/machine-learning/glossary/?hl=ko#tensor)를 포함합니다. tf.data.Iterator 개체는 Dataset의 요소에 대한 액세스를 제공합니다.

Dataset API에 관한 자세한 내용은 텐서플로우 프로그래머 가이드에서 [데이터 가져오기](https://www.tensorflow.org/programmers_guide/datasets?hl=ko)를 참조하세요.

**결정 경계(decision boundary)**

[**이진 클래스**](https://developers.google.com/machine-learning/glossary/?hl=ko#binary_classification) 또는 [**다중 클래스 분류 문제**](https://developers.google.com/machine-learning/glossary/?hl=ko#multi-class)에서 모델이 학습한 클래스 사이의 구분선입니다. 예를 들어 아래 그림과 같은 이진 분류 문제의 경우 결정 경계는 주황색 클래스와 파란색 클래스 사이의 경계선입니다.



**밀집 레이어(dense layer)**

[**완전 연결 레이어**](https://developers.google.com/machine-learning/glossary/?hl=ko#fully_connected_layer)의 동의어입니다.

**심층 모델(deep model)**

여러 [**히든 레이어**](https://developers.google.com/machine-learning/glossary/?hl=ko#hidden_layer)를 포함하는 [**신경망**](https://developers.google.com/machine-learning/glossary/?hl=ko#neural_network) 유형입니다. 심층 모델은 학습 가능한 비선형성에 의존합니다.

[**와이드 모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#wide_model)과 대비되는 개념입니다.

**밀집 특성(dense feature)**

대부분의 값이 0이 아니고 일반적으로 부동 소수점 값의 [**텐서**](https://developers.google.com/machine-learning/glossary/?hl=ko#tensor)로 이루어진 [**특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#feature)입니다. [**희소 특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#sparse_features)과 대비되는 개념입니다.

**기기(device)**

CPU, GPU, TPU 등 텐서플로우 세션을 실행할 수 있는 하드웨어 카테고리입니다.

**불연속 특성(discrete feature)**

가능한 값의 유한 집합을 갖는 [**특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#feature)입니다. 예를 들어 값이 *동물*, *식물*, *광물* 중 하나여야 하는 특성은 불연속 또는 범주형 특성입니다. [**연속 특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#continuous_feature)과 대비되는 개념입니다.

**드롭아웃 정규화(dropout regularization)**

[**신경망**](https://developers.google.com/machine-learning/glossary/?hl=ko#neural_network)을 학습시키는 데 유용한 [**정규화**](https://developers.google.com/machine-learning/glossary/?hl=ko#regularization) 형태입니다. 드롭아웃 정규화를 사용하면 단일 경사 스텝이 일어날 때마다 특정 네트워크 레이어의 유닛을 고정된 개수만큼 무작위로 선택하여 삭제합니다. 드롭아웃하는 유닛이 많을수록 정규화가 강력해집니다. 이 방식은 네트워크를 학습시켜 더 작은 네트워크로 이루어진 대규모 앙상블을 모방하도록 하는 방식과 비슷합니다. 자세한 내용은 [드롭아웃: 신경망의 과적합을 방지하는 간단한 방법](http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf)을 참조하세요.

**동적 모델(dynamic model)**

온라인으로 학습되며 지속적으로 업데이트되는 [**모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#model)입니다. 즉, 데이터가 끊임없이 모델에 유입됩니다.

**E**

**조기 중단(early stopping)**

학습 손실이 완전히 감소되기 [**전에**](https://developers.google.com/machine-learning/glossary/?hl=ko#regularization) 모델 학습이 종료되는 *정규화* 방식입니다. 조기 중단에서는 [**검증 데이터 세트**](https://developers.google.com/machine-learning/glossary/?hl=ko#validation_set)의 손실이 증가하기 시작할 때, 즉 [**일반화**](https://developers.google.com/machine-learning/glossary/?hl=ko#generalization) 성능이 악화될 때 모델 학습을 종료합니다.

**임베딩(embeddings)**

연속 값 특성으로 표현된 범주형 특성입니다. 일반적으로 임베딩(embeddings)은 고차원 벡터를 저차원 공간으로 변환한 결과입니다. 예를 들어 영어 문장의 단어를 다음 두 가지 방법 중 하나로 표현할 수 있습니다.

* 백만 개의 요소를 갖는(고차원) [**희소 벡터**](https://developers.google.com/machine-learning/glossary/?hl=ko#sparse_features)로 표현합니다. 모든 요소는 정수입니다. 벡터의 각 셀은 서로 다른 영어 단어를 나타내며, 셀 값은 해당 단어가 문장에서 나오는 횟수를 나타냅니다. 영어 문장 하나에 포함되는 단어 수는 대개 50개 이하이므로 벡터의 거의 모든 셀에 0이 포함됩니다. 0이 아닌 소수의 셀은 해당 단어가 문장에서 나오는 횟수를 나타내는 낮은 정수(일반적으로 1)를 포함합니다.
* 수백 개의 요소를 갖는(저차원) [**밀집 벡터**](https://developers.google.com/machine-learning/glossary/?hl=ko#dense_feature)로 표현합니다. 각 요소는 0~1 범위의 부동 소수점 값을 갖습니다. 이는 임베딩의 예입니다.

텐서플로우에서는 [**신경망**](https://developers.google.com/machine-learning/glossary/?hl=ko#neural_network)의 다른 매개변수와 마찬가지로 [**역전파**](https://developers.google.com/machine-learning/glossary/?hl=ko#backpropagation) [**손실**](https://developers.google.com/machine-learning/glossary/?hl=ko#loss)을 통해 임베딩을 학습합니다.

**경험적 위험 최소화(ERM, empirical risk minimization)**

학습 세트에서 손실을 최소화하는 함수를 선택함을 의미합니다. [**구조적 위험 최소화**](https://developers.google.com/machine-learning/glossary/?hl=ko#SRM)와 대비되는 개념입니다.

**앙상블(ensemble)**

여러 [**모델**](https://developers.google.com/machine-learning/glossary/?hl=ko#model)의 예측을 병합한 결과입니다. 다음 중 하나 이상을 통해 앙상블을 만들 수 있습니다.

* 서로 다른 초기화
* 서로 다른 [**초매개변수**](https://developers.google.com/machine-learning/glossary/?hl=ko#hyperparameter)
* 서로 다른 전체 구조

[심층 모델 및 와이드 모델](https://www.tensorflow.org/tutorials/wide_and_deep?hl=ko)도 앙상블의 일종입니다.

**세대(epoch)**

전체 데이터 세트의 각 예를 한 번씩 확인한 전체 학습 단계입니다. 따라서 한 세대(이폭)는 N/[**배치 크기**](https://developers.google.com/machine-learning/glossary/?hl=ko#batch_size) 학습 [**반복**](https://developers.google.com/machine-learning/glossary/?hl=ko#iteration)을 나타내며, 여기에서 N은 총 예시 수입니다.

**에스티메이터(Estimator)**

tf.Estimator 클래스의 인스턴스로서, 텐서플로우 그래프를 작성하고 텐서플로우 세션(session)을 실행하는 로직을 캡슐화합니다. [여기](https://www.tensorflow.org/extend/estimators?hl=ko)에서 설명한 것과 같이 나만의 [**맞춤 에스티메이터**](https://developers.google.com/machine-learning/glossary/?hl=ko#custom_estimator)를 만들거나 다른 사용자가 작성한 [**사전 제작된 에스티메이터**](https://developers.google.com/machine-learning/glossary/?hl=ko#pre-made_Estimator)를 인스턴스화할 수 있습니다.

**예(example)**

데이터 세트의 한 행입니다. 예는 하나 이상의 [**특성**](https://developers.google.com/machine-learning/glossary/?hl=ko#feature)을 포함하며, [**라벨**](https://developers.google.com/machine-learning/glossary/?hl=ko#label)을 포함할 수도 있습니다. [**라벨이 있는 예**](https://developers.google.com/machine-learning/glossary/?hl=ko#labeled_example) 및 [**라벨이 없는 예**](https://developers.google.com/machine-learning/glossary/?hl=ko#unlabeled_example)를 참조하세요.