머신러닝 용어집

이 용어집에서는 머신러닝과 관련된 일반 용어 및 텐서플로우에서만 사용하는 용어를 정의합니다.

A

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 (#classification_model)의 예측이 얼마나 정확한지를 의미합니다. **다중 클래스 분류** (#multi-class)에서 정확성의 정의는 다음과 같습니다.

$$\text{정확성} = \frac{\text{정확한 예측}}{\text{총 예시 수}}$$

이진 분류 (#binary_classification)에서 정확성의 정의는 다음과 같습니다.

$$\text{정확성} = \frac{\text{참양성} + \text{참음성}}{\text{총 예시 수}}$$

참양성 (#TP) 및 **참음성** (#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** (#ReLU), **시그모이드** (#sigmoid_function) 등의 함수입니다.

AdaGrad

각 매개변수의 경사를 재조정하여 사실상 각 매개변수에 독립적인 **학습률** (#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** (#classification_threshold)을 고려하는 평가 측정항목입니다.

AUC(ROC 곡선 (#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 (#neural_network)에서 **경사하강법** (#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** (#model) 또는 휴리스틱입니다. 기준은 모델 개발자가 특정 문제에 예상되는 최소 성능을 산정하는 데 도움을 줍니다.

배치(batch)

모델 학습 (#model_training)의 **반복** (#iteration) 1회, 즉 **경사** (#gradient) 업데이트 1회에 사용되는 예의 집합입니다.

배치 크기 (#batch_size)를 참조하세요.

배치 크기(batch size)

배치 (#batch) 하나에 포함되는 예의 개수입니다. 예를 들어 **SGD** (#SGD)의 배치 크기는 1이고, **미니 배치** (#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 w_0으로 표현됩니다. 예를 들어 다음 수식에서 편향은 b입니다.

$$y' = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

예측 편향 (#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 (#bucketing)을 참조하세요.

버케팅(bucketing)

하나의 특성(일반적으로 **연속** (#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** (#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** (#positive_class)가 항상 적절한 긍정 강화를 받는 한 **네거티브 클래스** (#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** (#feature)입니다. 예를 들어 `Tudor, ranch, colonial`이라는 3가지 가능한 값으로 이루어진 불연속 집합을 갖는 `house style`이라는 범주형 특성이 있다고 가정해 보겠습니다. `house style`을 범주형 데이터로 표현하면 `Tudor`, `ranch`, `colonial`이 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: `red`와 `white`)을 갖도록 허용할 수 있습니다.

범주형 특성을 **불연속 특성** (#discrete_feature)이라고도 합니다.

수치 데이터 (#numerical_data)와 대비되는 개념입니다.

중심(centroid)

k-평균 (#k-means) 또는 **k-중앙값** (#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** (#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** (#graph) 자체는 체크포인트에 포함되지 않습니다.

클래스(class)

열거형 목표값 집합 중 하나로서 라벨로 쓰입니다. 예를 들어 스팸을 감지하는 **이진 분류** (#binary_classification) 모델의 두 클래스는 *스팸* 및 *스팸 아님*입니다. 개의 품종을 식별하는 **다중 클래스 분류** (#multi_class_classification) 모델의 클래스는 *푸들*, *비글*, *퍼그* 등입니다.

클래스 불균형 데이터 세트(class-imbalanced data set)

두 클래스의 **라벨** (#label)이 서로 크게 다른 빈도를 보이는 **이진 분류** (#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** (#regression_model)과 비교되는 개념입니다.

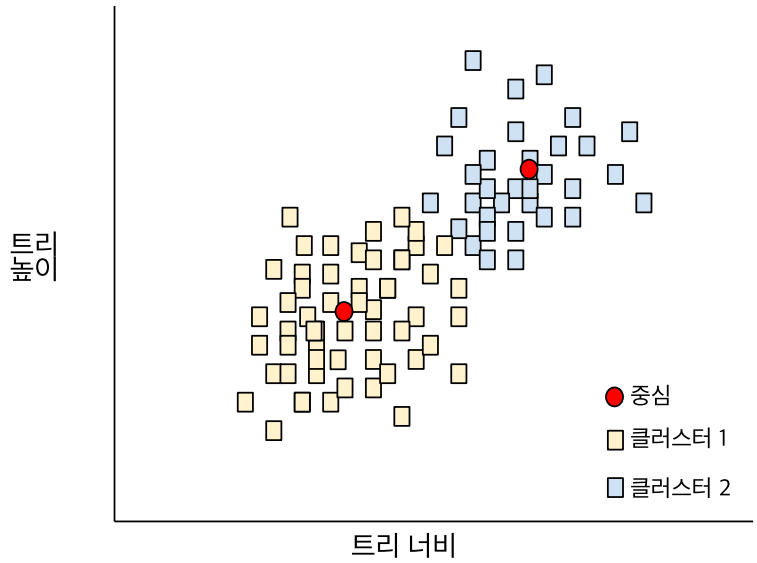
분류 임계값(classification threshold)

포지티브 클래스 (#positive_class)와 **네거티브 클래스** (#negative_class)를 구분 짓기 위한 모델의 예측 점수에 적용되는 스칼라값 기준입니다. **로지스틱 회귀** (#logistic_regression) 결과를 **이진 분류** (#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)

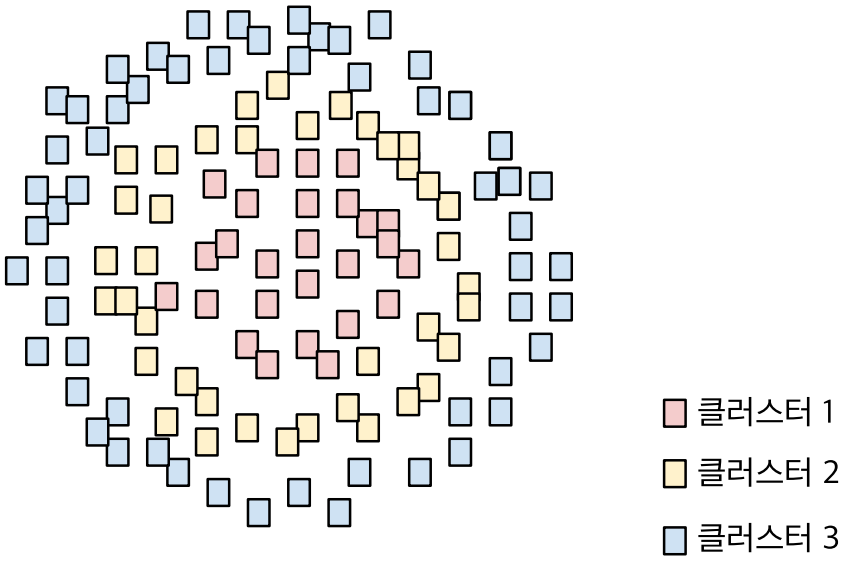
예 (#example)와 관련된 그룹화이며 특히 **비지도 학습** (#unsupervised_machine_learning)에서 사용됩니다. 모든 예가 그룹으로 묶이고 나면 사람이 선택적으로 각 클러스터에 의미를 부여할 수 있습니다.

클러스터링에는 여러 가지 알고리즘이 사용됩니다. 예를 들어 **k-평균** (#k-means) 알고리즘에서는 다음 다이어그램과 같이 각 예의 **중심** (#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 (#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** (#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 있습니

다.

혼동행렬은 **정밀도** (#precision), **재현율** (#recall) 등의 다양한 성능 측정항목을 계산하는 데 충분한 정보를 포함합니다.

연속 특성(continuous feature)

가능한 값이 무한한 범위를 갖는 부동 소수점 특성입니다. **불연속 특성** (#discrete_feature)과 대비되는 개념입니다.

수렴(convergence)

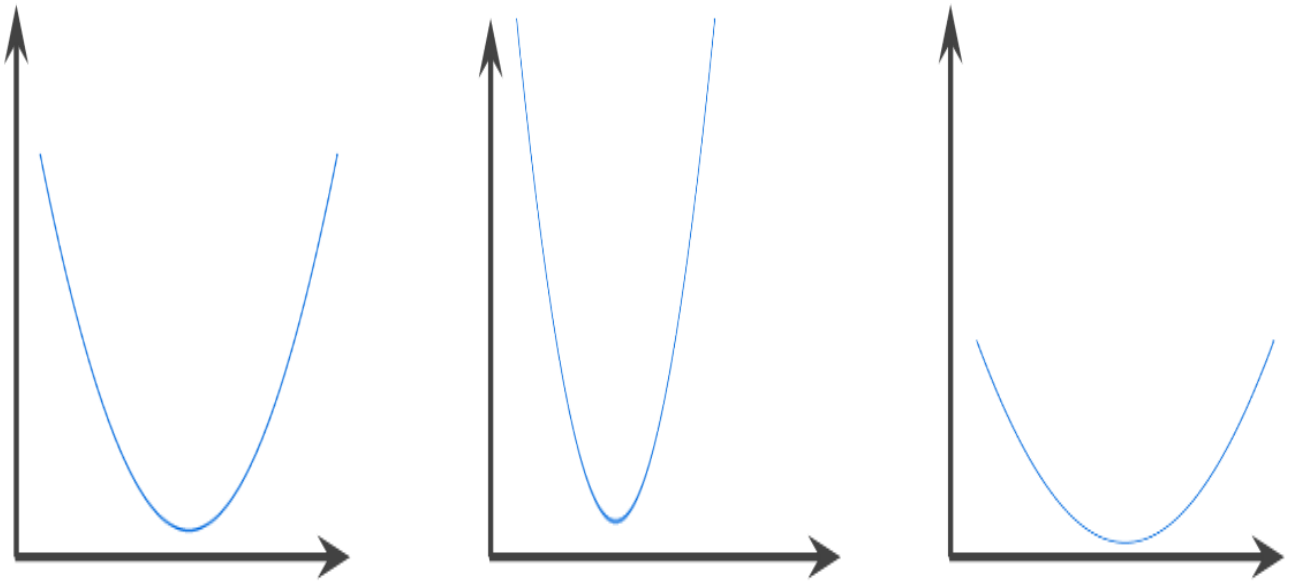
학습 중에 어느 정도 이상의 반복 후 학습 **손실** (#loss) 및 검증 손실이 거의 또는 전혀 변화하지 않는 상태를 비공식적으로 일컫는 용어입니다. 즉, 현재 데이터로 더 학습해도 모델이 개선되지 못할 때 모델이 수렴되었다고 말합니다. 딥 러닝의 경우 손실 값이 거의 일정하게 유지하며 수많은 반복을 하여 일시적으로 모델이 수렴한다고 착각을 할 수 있으나 결국 하강을 하는 경우도 있습니다.

조기 중단 (#early_stopping)을 참조하세요.

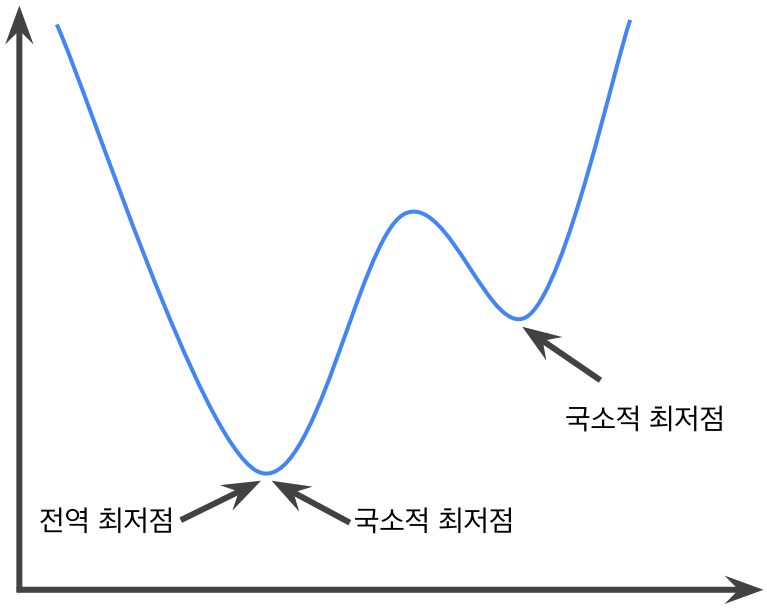
Boyd와 Vandenberghe의 볼록 최적화 (https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)를 참조하세요.

볼록 함수(convex function)

함수 그래프의 위쪽 영역이 **볼록 집합** (#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) 함수는 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** (#loss_functions) 중 다음을 비롯한 다수는 볼록 함수입니다.

- **L_2 손실(L2 loss)** (#L2_loss)
- **로그 손실(Log Loss)** (#Log_Loss)
- **L_1 정규화(L1 regularization)** (#L1_regularization)
- **L_2 정규화(L2 regularization)** (#L2_regularization)

경사하강법 (#gradient_descent)의 여러 가지 변형은 순볼록 함수의 최저점에 가까운 점을 찾도록 보장합니다. 마찬가지로, **확률적 경사하강법** (#SGD)의 여러 변형은 순볼록 함수의 최저점에 가까운 점을 찾을 가능성이 높지만 항상 보장되지는 않습니다.

두 볼록 함수의 합(예: L_2 손실 + L_1 정규화)은 볼록 함수입니다.

심층 모델 (#deep_model)은 어떠한 경우에도 볼록 함수가 아닙니다. 그럼에도 불구하고 **볼록 최적화** (#convex_optimization)를 위해 설계된 알고리즘은 심층 네트워크에서 비교적 양호한 해를 구할 가능성이 높지만, 이러한 해가 전역 최저점이라는 보장은 없습니다.

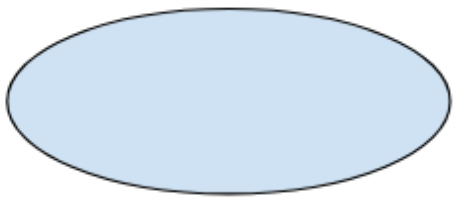
볼록 최적화(convex optimization)

경사하강법 (#gradient_descent) 등의 수학적 기법을 사용하여 **볼록 함수** (#convex_function)의 최저점을 찾는 과정입니다. 머신러닝에 대한 연구 중 상당한 비중이 볼록 최적화와 같은 다양한 문제를 고안하고 효과적인 해법을 찾는 데 집중되었습니다.

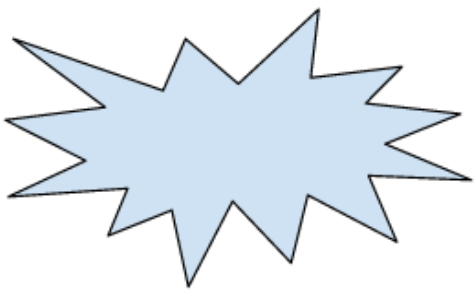
자세한 내용은 Boyd와 Vandenberghe의 볼록 최적화 (https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)를 참조하세요.

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** (#convolutional_operation) 또는 **컨볼루셔널 레이어** (#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** (#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 (#convolutional_operation)에서 사용되는 두 가지 중 하나입니다. 다른 하나는 입력 행렬의 슬라이스입니다. 컨볼루셔널 필터는 입력 행렬과 **순위(차원 수)** (#rank)는 동일하지만 모양은 더 작은 행렬입니다. 예를 들어 입력 행렬이 28x28인 경우 컨볼루셔널 필터는 이보다 작은 2차원 행렬이 됩니다.

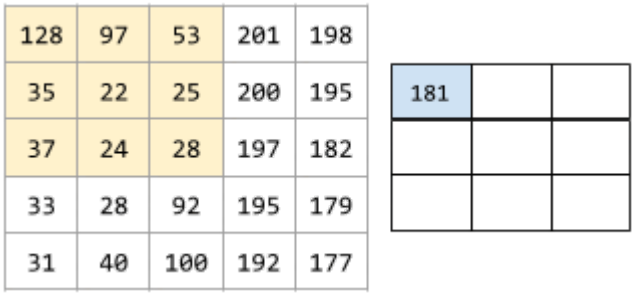
사진 조작에서 사용되는 컨볼루셔널 필터는 일반적으로 1과 0으로 구성된 일정한 패턴으로 설정됩니다. 머신러닝에서 컨볼루셔널 필터는 일반적으로 난수로 채워지며 네트워크가 이상적인 값을 학습시킵니다.

컨볼루셔널 레이어(convolutional layer)

심층신경망의 한 레이어로, 입력 행렬에 **컨볼루셔널 필터** (#convolutional_filter)를 적용합니다. 예를 들어 다음과 같은 3x3 **컨볼루셔널 필터** (#convolutional_filter)가 있다고 생각해 보세요.

0	1	0
1	0	1
0	1	0

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** (#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** (#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.

128	97	53	201	198
35	22	25	200	195
37	24	28	197	182
33	28	92	195	179
31	40	100	192	177

이제 다음과 같은 2x2 컨볼루셔널 필터가 있다고 생각해 보세요.

1	0
0	1

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.

128	97	x	1	0	=>	128	0	=	128+22=150
35	22		0	1		0	22		

컨볼루셔널 레이어 (#convolutional_layer)는 일련의 컨볼루셔널 연산으로 이루어지며, 각 연산은 입력 행렬의 서로 다른 슬라이스에 적용됩니다.

비용(cost)

손실 (#loss)의 동의어입니다.

교차 엔트로피(cross-entropy)

다중 클래스 분류 문제 (#multi-class)로 일반화한 **로그 손실** (#Log_Loss)입니다. 교차 엔트로피는 두 확률 분포 간의 차이를 계량합니다. **퍼플렉시티** (#perplexity)를 참조하세요.

맞춤 에스티메이터(custom Estimator)

이러한 방향 (https://www.tensorflow.org/extend/estimators)을 따라 직접 작성하는 **에스티메이터** (#Estimators)입니다.

사전 제작된 에스티메이터 (#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 (#example)를 모아 놓은 집합입니다.

Dataset API(tf.data)

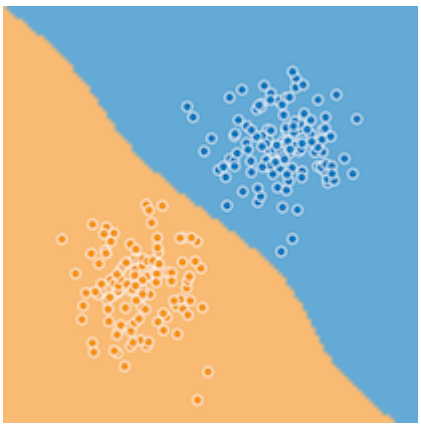
데이터를 읽고 머신러닝 알고리즘이 요구하는 형태로 변환하는 상위레벨의 TensorFlow API입니다. `tf.data.Dataset` 개체는 일련의 요소를 나타내며, 각 요소는 하나 이상의 **텐서** (#tensor)를 포함합니

다. `tf.data.Iterator` 개체는 `Dataset`의 요소에 대한 액세스를 제공합니다.

Dataset API에 관한 자세한 내용은 텐서플로우 프로그래머 가이드에서 데이터 가져오기 (https://www.tensorflow.org/programmers_guide/datasets)를 참조하세요.

결정 경계(decision boundary)

이진 클래스 (#binary_classification) 또는 **다중 클래스 분류 문제** (#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 (#fully_connected_layer)의 동의어입니다.

심층 모델(deep model)

여러 **히든 레이어** (#hidden_layer)를 포함하는 **신경망** (#neural_network) 유형입니다. 심층 모델은 학습 가능한 비선형성에 의존합니다.

와이드 모델 (#wide_model)과 대비되는 개념입니다.

밀집 특성(dense feature)

대부분의 값이 0이 아니고 일반적으로 부동 소수점 값의 **텐서** (#tensor)로 이루어진 **특성** (#feature)입니다. **희소 특성** (#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** (#feature)입니다. 예를 들어 값이 *동물*, *식물*, *광물* 중 하나여야 하는 특성은 불연속 또는 범주형 특성입니다. **연속 특성** (#continuous_feature)과 대비되는 개념입니다.

드롭아웃 정규화(dropout regularization)

신경망 (#neural_network)을 학습시키는 데 유용한 **정규화** (#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** (#model)입니다. 즉, 데이터가 끊임없이 모델에 유입됩니다.

E

조기 중단(early stopping)

학습 손실이 완전히 감소되기 **전에** (#regularization) 모델 학습이 종료되는 *정규화* 방식입니다. 조기 중단에서는 **검증 데이터 세트** (#validation_set)의 손실이 증가하기 시작할 때, 즉 **일반화** (#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** (#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** (#dense_feature)로 표현합니다. 각 요소는 0~1 범위의 부동 소수점 값을 갖습니다. 이는 임베딩의 예입니다.

텐서플로우에서는 **신경망** (#neural_network)의 다른 매개변수와 마찬가지로 **역전파** (#backpropagation) **손실** (#loss)을 통해 임베딩을 학습합니다.

경험적 위험 최소화(ERM, empirical risk minimization)

학습 세트에서 손실을 최소화하는 함수를 선택함을 의미합니다. **구조적 위험 최소화** (#SRM)와 대비되는 개념입니다.

앙상블(ensemble)

여러 **모델** (#model)의 예측을 병합한 결과입니다. 다음 중 하나 이상을 통해 앙상블을 만들 수 있습니다.

- 서로 다른 초기화
- 서로 다른 **초매개변수** (#hyperparameter)
- 서로 다른 전체 구조

심층 모델 및 와이드 모델 (https://www.tensorflow.org/tutorials/wide_and_deep)도 앙상블의 일종입니다.

세대(epoch)

전체 데이터 세트의 각 예를 한 번씩 확인한 전체 학습 단계입니다. 따라서 한 세대(이폭)는 **N**/**배치 크기** (#batch_size) 학습 **반복** (#iteration)을 나타내며, 여기에서 **N**은 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)에서 설명한 것과 같이 나만의 **맞춤 에스티메이터** (#custom_estimator)를 만들거나 다른 사용자가 작성한 **사전 제작된 에스티메이터** (#pre-made_Estimator)를 인스턴스화할 수 있습니다.

예(example)

데이터 세트의 한 행입니다. 예는 하나 이상의 **특성** (#feature)을 포함하며, **라벨** (#label)을 포함할 수도 있습니다. **라벨이 있는 예** (#labeled_example) 및 **라벨이 없는 예** (#unlabeled_example)를 참조하세요.

F

거짓부정(FN, false negative)

모델에서 **네거티브 클래스** (#negative_class)로 잘못 예측한 예입니다. 예를 들어 모델에서 특정 이메일 메시지가 스팸이 아닌 것으로(네거티브 클래스) 추론했지만 실제로는 스팸인 경우가 여기에 해당합니다.

거짓긍정(FP, false positive)

모델에서 **포지티브 클래스** (#positive_class)로 잘못 예측한 예입니다. 예를 들어 모델에서 특정 이메일 메시지가 스팸인 것으로(포지티브 클래스) 추론했지만 실제로는 스팸이 아닌 경우가 여기에 해당합니다.

거짓긍정률(FP rate, false positive rate)

ROC 곡선 (#ROC)의 x축입니다. FP 비율은 다음과 같이 정의됩니다.

$$\text{거짓긍정률} = \frac{\text{거짓긍정}}{\text{거짓긍정} + \text{참부정}}$$

특성(feature)

예측 (#prediction)을 수행하는 데 사용되는 입력 변수입니다.

특성 열(Feature column, tf.feature_column)

모델에서 특정한 특성을 어떻게 해석해야 하는지 지정하는 함수입니다. 특성 함수 호출에 반환된 출력을 수집하는 목록은 모든 **에스티메이터** (#Estimators) 생성자에게 필수 매개변수입니다.

모델에서 `tf.feature_column` 함수를 사용하면 입력 특성의 여러 가지 표현을 간편하게 실험할 수 있습니다. 자세한 내용은 텐서플로우 프로그래머 가이드의 특성 열 챕터 (https://www.tensorflow.org/get_started/feature_columns)를 참조하세요.

'특성 열'은 Google에서만 쓰는 용어입니다. Yahoo/Microsoft의 VW (https://en.wikipedia.org/wiki/Vowpal_Wabbit) 시스템에서는 특성 열을 '네임스페이스'라고 하며, 필드 (https://www.csie.ntu.edu.tw/~cjlin/libffm/)라고 지칭하는 경우도 있습니다.

특성 교차(feature cross)

범주형 데이터나 버케팅을 통한 연속 특성으로 부터 얻어진 개별 이진 특성들을 (데카르트 곱을 취해) 교차해 얻은 **합성 특성** (#synthetic_feature)입니다. 특성 교차는 비선형 관계를 표현하는 데 도움이 됩니다.

특성 추출(feature engineering)

모델을 학습시키는 데 유용할 **특성** (#feature)이 무엇인지 판단하고 로그 파일 및 기타 소스의 원시 데이터를 해당 특성으로 변환하는 과정입니다. 텐서플로우에서 특성 추출은 일반적으로 원시 로그 파일 항목을 **tf.Example** (#tf.Example) 프로토콜 버퍼로 변환하는 작업을 의미합니다. tf.Transform (https://github.com/tensorflow/transform)도 참조하세요.

특성 추출을 **특성 뽑아내기(extraction)**라고도 합니다.

특성 세트(feature set)

머신러닝 모델에서 학습에 사용하는 **특성** (#feature) 그룹입니다. 예를 들어 우편번호, 택지 규모, 입지 조건으로 단일 특성 세트를 구성하여 주택 가격을 예측하는 모델에 사용할 수 있습니다.

특성 사양(feature spec)

특성 (#feature) 데이터를 **tf.Example** (#tf.Example) 프로토콜 버퍼에서 추출하는 데 필요한 정보를 기술합니다. tf.Example 프로토콜 버퍼는 데이터의 컨테이너에 불과하므로 다음 정보를 지정해야 합니다.

- 추출할 데이터(특성의 키)
- 데이터 유형(예: float 또는 int)
- 길이(고정 또는 가변)

Estimator API (#Estimators)는 **FeatureColumns** (#feature_columns) 목록으로부터 특성 사양을 생성하는 기능을 제공합니다.

극소수 학습(few-shot learning)

객체 분류에 자주 사용되는 머신러닝 접근방식으로서 학습 예제 데이터가 적은 경우에서도 분류기를 효과적으로 학습하는 것을 목적으로 합니다.

원샷 학습 (#one-shot_learning)을 참조하세요.

전체 소프트맥스(full softmax)

소프트맥스 (#softmax)를 참조하세요. **후보 샘플링** (#candidate_sampling)과 대비되는 개념입니다.

완전 연결 레이어(fully connected layer)

각 **노드** (#node)가 후속 히든 레이어의 *모든* 노드에 연결된 **히든 레이어** (#hidden_layer)입니다.

완전 연결 레이어를 **밀집 레이어** (#dense_layer)라고도 합니다.

G

일반화(generalization)

모델에서 학습에 사용된 데이터가 아닌 이전에 접하지 못한 새로운 데이터에 대해 올바른 예측을 수행하는 능력을 의미합니다.

일반화 선형 모형(generalized linear model)

가우스 잡음 (https://en.wikipedia.org/wiki/Gaussian_noise)에 기초한 **최소 제곱 회귀** (#least_squares_regression)모델을 푸아송 잡음 (https://en.wikipedia.org/wiki/Shot_noise), 범주형 잡음 등의 여타 잡음 유형에 기초한 다른 모델 유형으로 일반화한 결과물입니다. 일반화 선형 모델의 예는 다음과 같습니다.

- **로지스틱 회귀** (#logistic_regression)
- 다중 클래스 회귀
- 최소 제곱 회귀

볼록 최적화 (https://en.wikipedia.org/wiki/Convex_optimization)를 통해 일반화 선형 모형의 매개변수를 구할 수 있습니다.

일반화 선형 모델에는 다음과 같은 속성이 있습니다.

- 최적화된 최소 제곱 회귀 모델의 평균적인 예측은 학습 데이터의 평균 라벨과 동일합니다.
- 최적화된 로지스틱 회귀 모델이 예측하는 평균적인 확률은 학습 데이터의 평균 라벨과 동일합니다.

일반화 선형 모형의 성능은 특성에 따라 제한됩니다. 일반화 선형 모형은 심층 모델과 달리 '새로운 특성을 학습'하지 못합니다.

경사(gradient)

모든 독립 변수를 기준으로 한 **편미분** (#partial_derivative)의 벡터입니다. 머신러닝에서 경사는 모델 함수의 편미분의 벡터입니다. 경사는 가장 급격한 상승 방향을 가리킵니다.

경사 제한(gradient clipping)

경사 (#gradient) 값을 적용하기 전에 제한을 둡니다. 경사 제한은 수치적 안정성을 보장하고 경사 발산 (http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf)
을 예방하는 데 도움이 됩니다.

경사하강법(gradient descent)

학습 데이터의 조건에 따라 모델의 매개변수를 기준으로 손실의 경사를 계산하여 **손실** (#loss)을 최소화하는 기법입니다. 쉽게 설명하면, 경사하강법은 매개변수를 반복적으로 조정하면서 손실을 최소화하는 **가중치** (#weight)와 편향의 가장 적절한 조합을 점진적으로 찾는 방식입니다.

그래프(graph)

텐서플로우에서는 계산 사양을 의미합니다. 그래프의 노드는 연산을 의미합니다. 엣지는 방향성을 가지며, 연산의 결과(텐서 (https://www.tensorflow.org/api_docs/python/tf/Tensor))를 다른 연산의 피연산자로 전달함을 의미합니다. **텐서보드** (#TensorBoard)를 사용하여 그래프를 시각화할 수 있습니다.

H

휴리스틱(heuristic)

문제에 대해 이상적이지는 않지만 진전을 이루거나 교훈을 얻기에는 충분한 실용적인 해법입니다.

히든 레이어(hidden layer)

신경망 (#neural_network)에서 **입력 레이어** (#input_layer)(특성)와 **출력 레이어** (#output_layer)(예측) 사이에 위치하는 합성 레이어입니다. 신경망에 하나 이상의 히든 레이어가 포함될 수 있습니다.

힌지 손실(hinge loss)

각 학습 예에서 최대한 멀리 떨어진 **결정 경계** (#loss)를 구하여 예와 경계 사이의 간격을 최대화하도록 고안된 **분류** (#classification_model)의 **손실** (#decision_boundary) 함수군입니다. **KSVM** (#KSVMs)은 힌지 손실을 사용하거나 제곱 힌지 손실 등의 관련 함수를 사용합니다. 이진 분류에서는 힌지 손실 함수가 다음과 같이 정의됩니다.

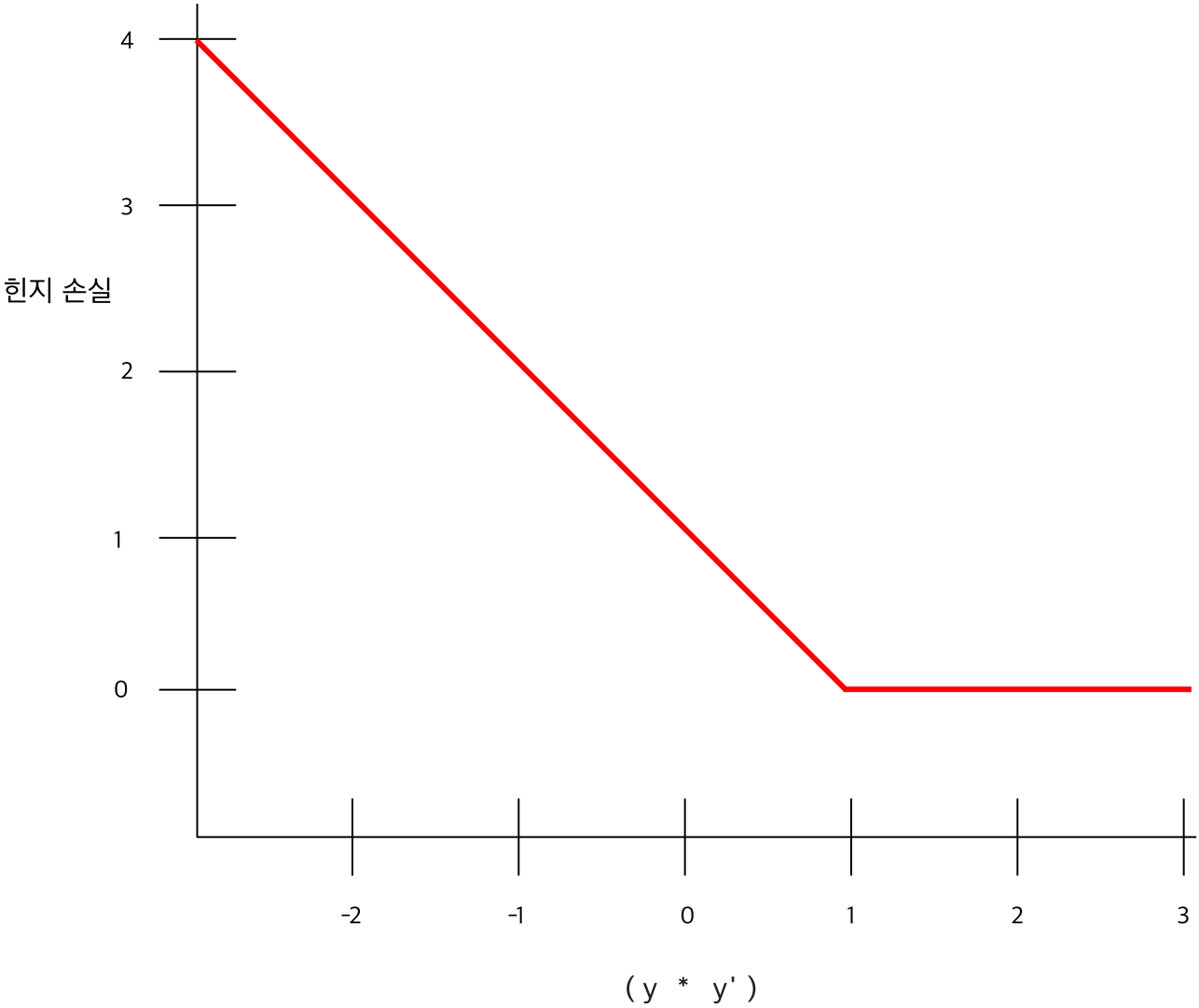
$$\text{손실} = \text{최대값}(0, 1 - (y' * y))$$

여기에서 *y'*는 분류자 모델의 원시 출력입니다.

$$y' = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

또한 *y*는 참 라벨(-1 또는 +1)입니다.

따라서 힌지 손실과 (y * y')로 그래프를 그리면 다음과 같습니다.



홀드아웃 데이터(holdout data)

학습 중에 의도적으로 사용하지 않은('홀드아웃된') **예** (#example)입니다. **검증 데이터 세트** (#validation_set)와 **테스트 데이터 세트** (#test_set)는 홀드아웃 데이터의 예입니다. 홀드아웃 데이터는 학습에 사용하지 않은 데이터에 대한 모델의 일반화 능력을 평가하는 데 도움이 됩니다. 홀드아웃 세트의 손실은 학습 데이터의 손실보다 이전에 접하지 못한 데이터의 손실을 더 우수하게 예측합니다.

초매개변수(hyperparameter)

모델 학습을 연속적으로 실행하는 중에 사용자 본인 에 의해 조작되는 '손잡이'입니다. 예를 들어 **학습률** (#learning_rate)은 초매개변수 중 하나입니다.

매개변수 (#parameter)와 대비되는 개념입니다.

초평면(hyperplane)

한 공간을 두 부분공간으로 나누는 경계입니다. 예를 들어 직선은 2차원의 초평면이고 평면은 3차원의 초평면입니다. 머신러닝에서 통용되는 초평면의 의미는 고차원 공간을 나누는 경계입니다. **커널 서포트 벡터 머신** (#KSVMs)은 일반적으로 초고차원 공간에서 초평면을 사용하여 포지티브 클래스와 네거티브 클래스를 구분합니다.

I

독립적이고 동일한 분포(i.i.d, independently and identically distributed)

변화가 없는 분포에서 각 값이 이전에 추출된 값에 의존하지 않도록 추출한 데이터입니다. i.i.d.는 유용한 수학적 구조이지만 현실에서는 거의 찾아볼 수 없다는 점에서 머신러닝의 이상기체 (https://en.wikipedia.org/wiki/Ideal_gas)로 비유할 수 있습니다. 예를 들어 웹페이지의 방문자 분포는 짧은 기간에는 i.i.d.일 수 있습니다. 즉, 짧은 기간에는 분포가 변하지 않으며 각 사용자의 방문은 일반적으로 서로 독립적입니다. 그러나 기간을 확대하면 웹페이지 방문자 수에 계절적인 편차가 나타날 수 있습니다.

추론(inference)

머신러닝에서는 학습된 모델을 **라벨이 없는 예** (#unlabeled_example)에 적용하여 예측을 수행하는 과정을 의미할 때가 많습니다. 통계학에서는 특정한 관찰 데이터에 맞게 분포의 매개변수를 조정하는 과정을 의미합니다. 통계적 추론에 대한 위키백과 문서 (https://en.wikipedia.org/wiki/Statistical_inference)를 참조하세요.

입력 함수(input function)

텐서플로우에서 입력 데이터를 **에스티메이터** (#Estimators)의 학습, 평가 또는 예측 방법으로 반환하는 함수입니다. 예를 들어 학습 입력 함수는 **학습 세트** (#batch)의 특성 및 라벨로 이루어진 **배치** (#training_set)를 반환합니다.

입력 레이어(input layer)

신경망 (#neural_network)의 첫 번째 레이어로서 입력 데이터를 수신합니다.

인스턴스(instance)

예 (#example)의 동의어입니다.

해석 가능성(interpretability)

모델의 예측을 사람이 어느 정도까지 설명할 수 있는지를 나타냅니다. 심층 모델은 해석할 수 없는 경우가 많습니다. 즉, 심층 모델의 여러 레이어를 사람이 알아보기는 어려울 수 있습니다. 반면, 선형 회귀 모형 및 **와이드 모델** (#wide_model)은 일반적으로 해석 가능성이 훨씬 더 높습니다.

평가자 간 동의(inter-rater agreement)

작업을 수행할 때 인간 평가자들이 서로 동의하는 빈도를 나타냅니다. 평가자들이 동의하지 않는 경우 작업 지시를 개선해야 할 수 있습니다. **평정자 간 동의** 또는 **평가자 간 신뢰성**이라고도 합니다. 가장 널리 사용되는 평가자 간 동의 측정 방식 중 하나인 Cohen's kappa (https://en.wikipedia.org/wiki/Cohen%27s_kappa)를 참조하세요.

반복(iteration)

학습 중에 모델의 가중치를 한 번 업데이트하는 작업입니다. 반복은 데이터 **배치** (#batch) 하나의 손실을 기준으로 매개변수의 경사를 계산하는 작업으로 이루어집니다.

K

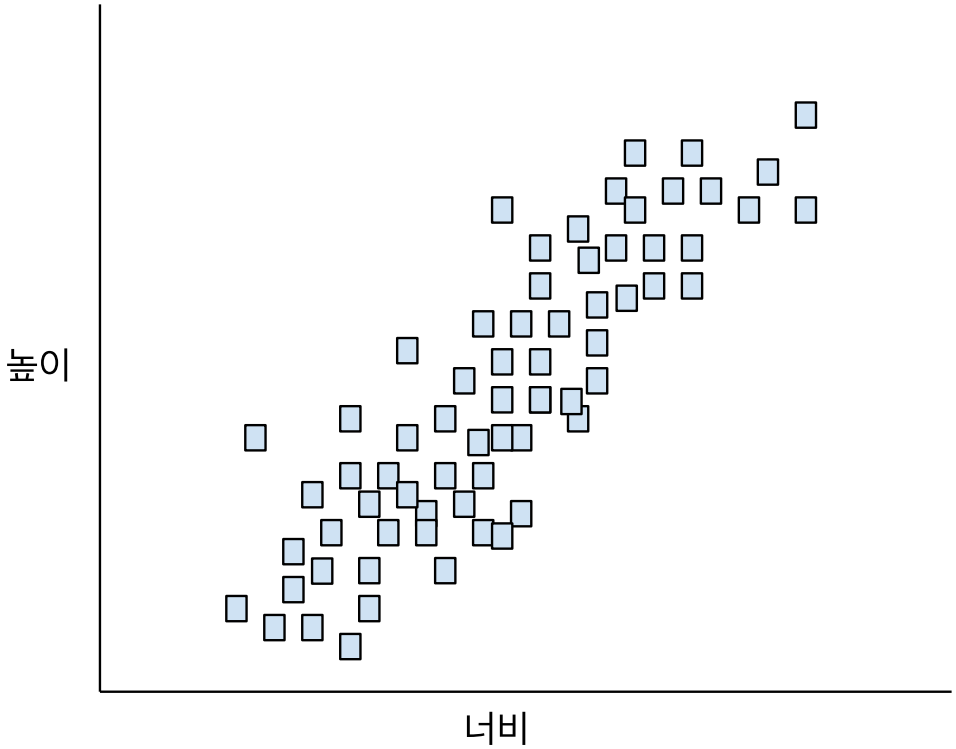
k-평균(k-means)

비지도 학습의 한 방법으로 데이터를 그룹화하는 데 널리 사용되는 **클러스터링** (#clustering) 알고리즘입니다. k-평균 알고리즘은 기본적으로 다음과 같은 일을 합니다.

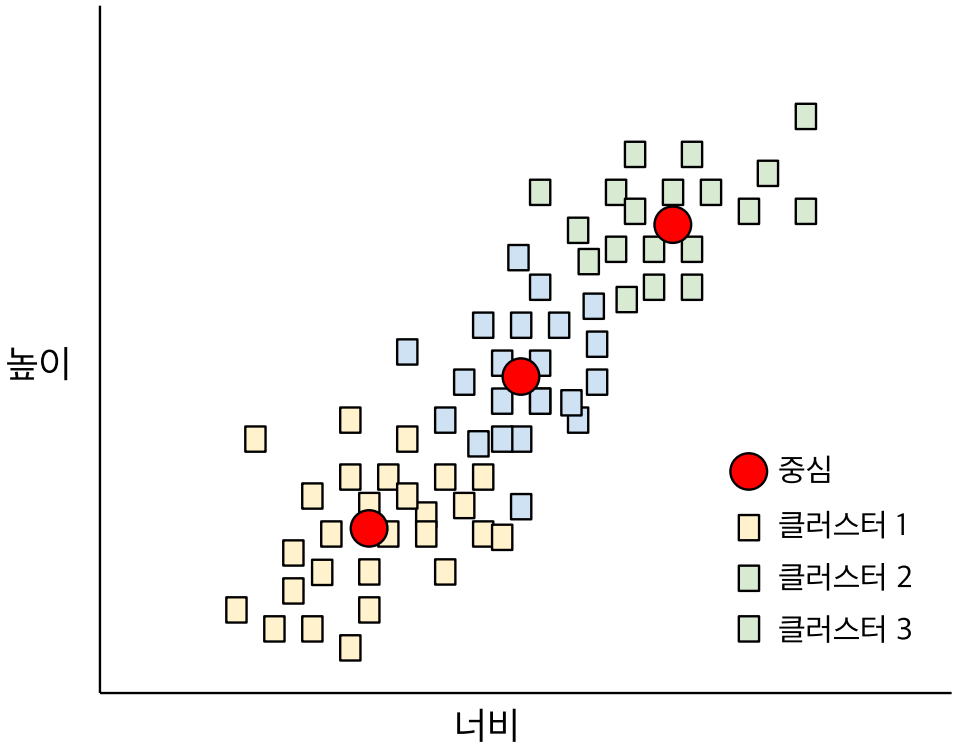
- 최고의 중심점(**중심** (#centroid)이라고 함)들을 반복적으로 결정합니다.
- 각 예를 가장 가까운 중심에 배정합니다. 같은 중심에 가장 가까운 예는 같은 그룹에 속합니다.

k-평균 알고리즘은 각 예가 가장 가까운 중심과 각 예 사이 거리의 누적 *제곱*을 최소화할 수 있는 중심의 위치를 선택합니다.

개의 키와 몸길이를 나타낸 다음 그래프를 예로 들어보겠습니다.



k=3인 경우 k-평균 알고리즘에서는 3개의 중심을 결정합니다. 각 예는 가장 가까운 중심에 배정되며, 그 결과 세 개의 그룹이 만들어집니다.



한 제조업체에서 애견용 스웨터의 S, M, L 사이즈의 이상적인 크기를 결정하고 싶어 한다고 생각해 보겠습니다. 세 개의 중심은 클러스터에 속한 각 개의 평균 키와 평균 몸길이를 나타냅니다. 따라서 제조업체에서는 이 세 개의 중심을 기준으로 하여 스웨터의 크기를 설정할 수 있습니다. 일반적으로 한 클러스터의 중심은 해당 클러스터의 예가 *아닙니다*.

앞의 그림은 키와 몸길이라는 두 개의 특성만 있는 예의 k-평균을 보여줍니다. k-평균을 사용하면 여러 가지 특성을 기준으로 예를 그룹화할 수 있습니다.

k-중앙값(k-median)

k-평균 (#k-means)과 밀접한 관련이 있는 클러스터링 알고리즘입니다. 이 두 알고리즘의 실질적인 차이는 다음과 같습니다.

- k-평균 알고리즘에서는 중심이 될 수 있는 위치와 각 예의 위치 사이 거리를 *제곱*한 값의 합계를 최소화하는 방식으로 중심을 결정합니다.
- k-중앙값 알고리즘에서는 중심이 될 수 있는 위치와 각 예의 위치 사이의 거리 값의 합계를 최소화하는 방식으로 중심을 결정합니다.

거리의 정의도 다릅니다.

- k-평균 알고리즘에서는 중심에서 예까지의 유클리드 거리 (https://en.wikipedia.org/wiki/Euclidean_distance)를 사용합니다. 2차원에서 유클리드 거리는 피타고라스 정리를 사용하여 빗변의 길이를 계산하는 것을 의미합니다. 예를 들어 (2,2)와 (5, -2) 사이의 k-평균 거리는 다음과 같습니다.

$$\text{유클리드 거리} = \sqrt{(2-5)^2 + (2--2)^2} = 5$$

- k-중앙값 알고리즘에서는 중심에서 예까지의 맨해튼 거리 (https://en.wikipedia.org/wiki/Taxicab_geometry)를 사용합니다. 이 거리는 각 차원 값 차의 절대값의 합입니다. 예를 들어 (2,2)와 (5, -2) 사이의 k-중앙값 거리는 다음과 같습니다.

$$\text{맨하탄 거리} = |2-5| + |2--2| = 7$$

Keras

널리 사용되는 Python 머신러닝 API입니다. Keras (https://keras.io)는 텐서플로우를 비롯한 여러 딥 러닝 프레임워크에서 실행되며, 텐서플로우에서는 **tf.keras** (https://www.tensorflow.org/api_docs/python/tf/keras)로 제공됩니다.

커널 서포트 벡터 머신(KSVM, Kernel Support Vector Machine)

입력 데이터 벡터를 상위 차원 공간에 매핑하여 **포지티브 클래스** (#positive_class)와 **네거티브 클래스** (#negative_class) 사이의 간격을 최대화하는 것을 목표로 하는 분류 알고리즘입니다. 예를 들어 입력 데이터 세트가 특성 100개로 이루어진 분류 문제를 생각해 보겠습니다. KSVM은 포지티브 클래스와 네거티브 클래스 사이의 간격을 최대화하기 위해 내부적으로 이러한 특성을 백만차원 공간에 매핑할 수 있습니다. KSVM은 힌지 손실 (#hinge-loss)이라는 손실 함수를 사용합니다.

L

L_1 손실(L1 loss)

모델이 예측하는 값과 **라벨** (#loss)의 실제 값 차이의 절대값에 기초한 **손실** (#label) 함수입니다. L_1 손실은 **L_2 손실** (#squared_loss)보다 이상점에 둔감합니다.

L_1 정규화(L1 regularization)

가중치의 절대값 합에 비례하여 가중치에 페널티를 주는 **정규화** (#regularization) 유형입니다. **희소 특성** (#sparse_features)에 의존하는 모델에서 L_1 정규화는 관련성이 없거나 매우 낮은 특성의 가중치를 정확히 0으로 유도하여 모델에서 해당 특성을 배제하는 데 도움이 됩니다. **L_2 정규화** (#L2_regularization)와 대비되는 개념입니다.

L_2 손실(L2 loss)

제곱 손실 (#squared_loss)을 참조하세요.

L_2 정규화(L2 regularization)

가중치 **제곱** (#regularization)의 합에 비례하여 가중치에 페널티를 주는 *정규화* 유형입니다. L_2 정규화는 높은 긍정 값 또는 낮은 부정 값을 갖는 이상점 가중치를 0은 아니지만 0에 가깝게 유도하는 데 도움이 됩니다. **L1 정규화** (#L1_regularization)와 대비되는 개념입니다. L_2 정규화는 선형 모델의 일반화를 항상 개선합니다.

라벨(label)

지도 학습에서 **예** (#example)의 '답' 또는 '결과' 부분을 의미합니다. 라벨이 있는 데이터 세트의 각 예는 하나 이상의 특성과 하나의 라벨로 구성됩니다. 예를 들어 주택 데이터 세트의 경우 특성은 침실 수, 화장실 수, 주택의 연령일 수 있고 라벨은 주택의 가격일 수 있습니다. 스팸 감지 데이터 세트의 경우 특성은 제목, 보낸 사람, 이메일 메시지 자체일 수 있고 라벨은 '스팸' 또는 '스팸 아님'일 가능성이 높습니다.

라벨이 있는 예(labeled example)

특성 (#feature)과 **라벨** (#label)을 포함하는 예입니다. 지도 학습에서 모델은 라벨이 있는 예를 학습합니다.

람다(lambda)

정규화율 (#regularization_rate)의 동의어입니다.

중복으로 정의된 용어입니다. 여기에서는 **정규화** (#regularization) 맥락의 용어 정의에 집중합니다.

레이어(layer)

신경망 (#neural_network)에서 입력 특성의 집합 또는 뉴런의 출력을 처리하는 **뉴런** (#neuron) 집합입니다.

텐서플로우에서는 추상적인 의미로 사용되기도 합니다. 레이어는 **텐서** (#tensor) 및 구성 옵션을 입력으로 취하고 다른 텐서를 출력하는 Python 함수입니다. 필요한 텐서가 작성되면 사용자는 **모델 함수** (#model_function)를 통해 결과를 **에스티메이터** (#Estimators)로 변환할 수 있습니다.

Layers API(tf.layers)

여러 레이어를 조합하여 **심층** (#deep_model)신경망을 구축하는 텐서플로우 API입니다. Layers API를 통해 다음과 같은 다양한 유형의 **레이어** (#layer)를 만들 수 있습니다.

- `tf.layers.Dense` - **완전 연결 레이어** (#fully_connected_layer)
- `tf.layers.Conv2D` - 컨볼루셔널 레이어

맞춤 에스티메이터 (#custom_estimator)를 만들 때는 레이어 개체를 작성하여 모든 **히든 레이어** (#hidden_layers)의 특징을 정의합니다.

Layers API는 **Keras** (#Keras) 레이어 API 규약을 따릅니다. 즉, Layers API의 모든 함수는 Keras layers API의 해당 함수와 접두사는 다를 수 있으나 이름 및 서명이 동일합니다.

학습률(learning rate)

경사하강법을 통해 모델을 학습시키는 데 사용되는 스칼라값입니다. 각 반복에서 **경사하강법** (#gradient_descent) 알고리즘은 학습률을 경사에 곱합니다. 이 곱셈의 결과를 **경사 스텝**이라고 합니다.

학습률은 핵심적인 **초매개변수** (#hyperparameter)입니다.

최소 제곱 회귀(least squares regression)

L_2 손실 (#L2_loss)을 최소화하면서 학습시킨 선형 회귀 모형입니다.

선형 회귀(linear regression)

회귀 모형 (#regression_model)의 한 유형으로서, 입력 특성의 선형 조합으로부터 연속 값을 출력합니다.

로지스틱 회귀(logistic regression)

분류 문제에서 선형 예측에 **시그모이드 함수** (#sigmoid_function)를 적용하여 가능한 각 불연속 라벨값에 대한 확률을 생성하는 모델입니다. 로지스틱 회귀는 **이진 분류** (#binary_classification) 문제에 흔히 사용되지만 **다중 클래스** (#multi-class) 분류 문제에도 사용될 수 있습니다. 이러한 경우를 **다중 클래스 로지스틱 회귀** 또는 **다항 회귀**라고 합니다.

로지트(logit)

분류 모델에서 생성되는 원시(정규화되지 않음) 예측 벡터로, 대개는 정규화 함수로 전달됩니다. 모델에서 다중 클래스 분류 문제를 해결하고 있는 경우 로지트는 소프트맥스 함수 (https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits_v2)의 입력으로 사용되는 것이 일반적입니다. 그런 다음 소프트맥스 함수에서 가능한 클래스별로 하나의 값을 갖는 (정규화된) 확률 벡터를 생성합니다.

또한 로지트는 **시그모이드 함수** (#sigmoid_function)의 요소별 역을 지칭할 때도 있습니다. 자세한 내용은 tf.nn.sigmoid_cross_entropy_with_logits (https://www.tensorflow.org/api_docs/python/tf/nn/sigmoid_cross_entropy_with_logits)를 참조하세요.

로그 손실(Log Loss)

이진 **로지스틱 회귀** (#logistic_regression)에 사용되는 **손실** (#loss) 함수입니다.

로그 오즈(log-odds)

어떤 이벤트가 일어날 가능성의 로그입니다.

이벤트가 이진 확률을 의미하는 경우의 **가능성**은 성공 확률(p) 대 실패 확률(1-p)의 비율을 의미합니다. 특정 이벤트의 성공 확률이 90%, 실패 확률이 10%라고 가정해 보겠습니다. 이 경우의 가능성은 다음과 같이 계산됩니다.

$$\text{가능성} = \frac{\text{p}}{\text{(1-p)}} = \frac{.9}{.1} = 9$$

로그 오즈는 가능성의 로그입니다. 관례에 따르면 '로그'란 자연 로그를 나타내지만 로그의 밑은 사실 1보다 큰 임의의 수가 될 수 있습니다. 관례를 따르면, 앞에서 든 예의 로그 오즈는 다음과 같이 나타낼 수 있습니다.

$$\text{로그 오즈} = ln(9) = 2.2$$

로그 오즈는 **시그모이드 함수** (#sigmoid_function)의 역함수입니다.

손실(loss)

모델의 **예측** (#prediction)이 **라벨** (#label)과 얼마나 차이가 나는지를 측정합니다. 다르게 표현하면, 모델이 얼마나 부정확한지를 나타냅니다. 이 값을 판단하려면 모델에서 손실 함수를 정의해야 합니다. 예를 들어 선형 회귀 모형은 일반적으로 **평균 제곱 오차** (#MSE)를, 로지스틱 회귀 모형은 **로그 손실** (#Log_Loss)을 손실 함수로 사용합니다.

M

머신러닝(machine learning)

입력 데이터를 바탕으로 예측 모델을 구축(학습)하는 프로그램 또는 시스템입니다. 학습된 모델은 시스템에서 모델을 학습시키는 데 사용한 데이터와 동일한 분포에서 추출되지만 이전에 나타나지 않은 새로운 데이터에 대해 유용한 예측을 수행하는 데 사용됩니다. 머신러닝은 이러한 프로그램 또는 시스템과 관련된 학문 분야를 가리키는 용어이기도 합니다.

평균 제곱 오차(MSE, Mean Squared Error)

예시당 평균 제곱 손실입니다. MSE는 **제곱 손실** (#squared_loss)을 **예시** (#example)의 개수로 나누어 계산합니다. **텐서플로우 플레이그라운드** (#TensorFlow_Playground)에서 '학습 손실' 및 '테스트 손실'로 표시하는 값이 MSE입니다.

측정항목(metric)

중요한 의미가 있는 수치입니다. 머신러닝 시스템에서 직접 최적화될 수도 있고, 그렇지 않을 수도 있습니다. 시스템에서 최적화를 시도하는 측정항목을 **목표** (#objective)라고 합니다.

Metrics API(tf.metrics)

모델을 평가하는 텐서플로우 API입니다. 예를 들어 `tf.metrics.accuracy`는 모델의 예측이 라벨과 일치하는 빈도를 판단합니다. **맞춤 에스티메이터** (#custom_estimator)를 작성할 때는 Metrics API 함수를 호출하여 모델 평가 방법을 지정합니다.

미니 배치(mini-batch)

학습 또는 추론의 단일 반복에서 함께 실행되는 **예** (#example)의 전체 배치 중에서 무작위로 선택한 소규모 부분집합입니다. 미니 배치의 **배치 크기** (#batch_size)는 일반적으로 10~1,000입니다. 전체 학습 데이터가 아닌 미니 배치의 손실을 계산하면 효율성이 크게 향상됩니다.

미니 배치 확률적 경사하강법(SGD, mini-batch stochastic gradient descent)

미니 배치 (#mini-batch)를 사용하는 **경사하강법** (#gradient_descent) 알고리즘입니다. 즉, 미니 배치 SGD는 학습 데이터 중 작은 부분집합을 기반으로 경사를 예측합니다. **기본적인 SGD** (#SGD)에서는 크기가 1인 미니 배치를 사용합니다.

ML

머신러닝 (#machine_learning)의 약어입니다.

모델(model)

ML 시스템이 학습 데이터로부터 학습한 내용을 표현합니다. 중복으로 정의된 용어로서, 다음과 같은 서로 관련된 두 가지 의미 중 하나를 나타낼 수 있습니다.

- 예측이 계산되는 방식의 구조를 표현하는 **텐서플로우** (#TensorFlow) 그래프
- 해당 텐서플로우 그래프에서 **학습** (#model_training)에 의해 결정되는 특정 가중치 및 편향

모델 함수(model function)

머신러닝 학습, 평가, 추론을 구현하는 **에스티메이터** (#Estimators) 내의 함수입니다. 예를 들어 모델 함수 안에서 학습을 구현한 부분에서는 심층신경망의 토폴로지를 정의하고 **옵티마이저** (#optimizer) 함수를 정하는 등의 작업을 처리할 수 있습니다. **사전 제작된 에스티메이터** (#pre-made_Estimator)를 사용하면 다른 누군가가 작성한 모델 함수를 쓸 수 있습니다. **맞춤 에스티메이터** (#custom_estimator)를 사용하면 모델 함수를 직접 만들어야 합니다.

모델 함수 작성에 관한 자세한 내용은 맞춤 에스티메이터 작성 (https://www.tensorflow.org/get_started/custom_estimators)을 참조하세요.

모델 학습(model training)

최상의 **모델** (#model)을 결정하는 과정입니다.

모멘텀(Momentum)

학습 단계가 현재 단계의 도함수뿐 아니라 바로 앞 단계의 도함수에도 의존하는 정교한 경사하강법 알고리즘입니다. 모멘텀에서는 물리학의 모멘텀(운동량) 계산과 마찬가지로 시간에 따른 경사의 지수가중이동평균을 계산합니다. 모멘텀은 학습이 국소 최저점에서 정체되는 현상을 방지하는 데 도움이 될 수 있습니다.

다중 클래스 분류(multi-class classification)

셋 이상의 클래스 사이에서 구분하는 분류 문제입니다. 예를 들어 단풍나무에는 약 128개 종이 있으므로 단풍나무 종을 분류하는 모델은 다중 클래스입니다. 반대로, 이메일을 두 가지 범주(*스팸*, *스팸 아님*)로만 구분하는 모델은 **이진 분류 모델** (#binary_classification)입니다.

다항 분류(multinomial classification)

다중 클래스 분류 (#multi-class)의 동의어입니다.

N

NaN 트랩(NaN trap)

모델의 숫자 중 하나가 학습 중에 NaN (https://en.wikipedia.org/wiki/NaN)이 됨으로 인해 모델의 다른 여러 숫자 또는 모든 숫자가 결국 NaN이 되는 상황입니다.

NaN은 'Not a Number(숫자 아님)'의 약어입니다.

네거티브 클래스(negative class)

이진 분류 (#binary_classification)에서는 클래스 중 하나는 포지티브로, 다른 하나는 네거티브로 규정됩니다. 포지티브 클래스는 모델에서 찾으려는 대상이고, 네거티브 클래스는 그와 다른 가능성입니다. 예를 들어 의료 검사의 네거티브 클래스는 '종양 아님'일 수 있습니다. 이메일 분류기의 네거티브 클래스는 '스팸 아님'일 수 있습니다. **포지티브 클래스** (#positive_class)를 참조하세요.

신경망(neural network)

사람의 두뇌를 본뜬 모델로서, 단순 연결 유닛 또는 **뉴런** (#neuron)으로 이루어지며 비선형성을 갖는 여러 레이어로 구성됩니다. 하나 이상의 레이어는 **히든 레이어** (#hidden_layer)입니다.

뉴런(neuron)

신경망 (#neural_network)의 노드로서, 일반적으로 여러 입력 값을 취하여 하나의 출력 값을 생성합니다. 뉴런은 입력 값의 가중 합에 **활성화 함수** (#activation_function)(비선형 변환)를 적용하여 출력 값을 계산합니다.

노드(node)

중복으로 정의된 용어로서 다음 중 하나를 의미합니다.

- **히든 레이어** (#hidden_layer)의 뉴런
- 텐서플로우 **그래프** (#graph)의 연산

정규화(normalization)

실제 값 범위를 표준 값 범위(일반적으로 -1~+1 또는 0~1)로 변환하는 과정입니다. 예를 들어 어떤 특성의 원래 범위가 800~6,000인 경우, 뺄셈과 나눗셈을 거쳐 값 범위를 -1~+1로 정규화할 수 있습니다.

조정 (#scaling)을 참조하세요.

수치 데이터(numerical data)

정수 또는 실수로 나타낸 **특성** (#feature)입니다. 예를 들어 부동산 모델에서는 주택의 규모(제곱피트 또는 제곱미터)를 수치 데이터로 표현할 수 있습니다. 특성을 수치 데이터로 표현한다는 것은 특성 값 사이에 또는 특성 값과 라벨 간에 *수학적* 관계가 있다는 의미입니다. 예를 들어 주택의 크기를 수치 데이터로 표현한다는 것은 넓이가 200제곱미터인 주택은 넓이가 100제곱미터인 주택보다 두 배 넓다는 의미입니다. 또한, 주택의 제곱미터 넓이는 주택 가격과 수학적 관계를 가질 가능성이 높습니다.

모든 정수 데이터를 수치 데이터로 표현해야 하는 것은 아닙니다. 예를 들어 정수를 우편번호로 사용하는 국가가 있더라도 모델에서는 정수 우편번호를 수치 데이터로 표현해서는 안 됩니다. 그 이유는 우편번호가 `20000`이라고 해서 우편번호가 10000인 경우보다 두 배 또는 절반의 속성을 갖지는 않기 때문입니다. 또한, 우편번호는 부동산의 가치와 상관관계가 *있긴 하지만* 우편번호가 20000인 부동산 가치가 우편번호가 10000인 부동산 가치의 두 배라고 가정할 수는 없습니다. 따라서 우편번호는 **범주형 데이터** (#categorical_data)로 표현되어야 합니다.

수치 특성을 **연속 특성** (#continuous_feature)이라고도 합니다.

Numpy

Python에서 효율적인 배열 작업을 제공하는 오픈소스 수학 라이브러리 (http://www.numpy.org/)입니다. **Pandas** (#pandas)는 Numpy를 기반으로 합니다.

O

목표(objective)

알고리즘에서 최적화하려는 측정항목입니다.

오프라인 추론(offline inference)

예측 (#prediction) 그룹을 생성하여 저장해 두고 요청에 따라 예측을 검색합니다. **온라인 추론** (#online_inference)과 대비되는 개념입니다.

원-핫 인코딩(one-hot encoding)

다음과 같은 특징을 갖는 희소 벡터입니다.

- 요소 중 하나가 1로 설정됩니다.
- 다른 요소는 모두 0으로 설정됩니다.

원-핫 인코딩은 가능한 값의 유한집합을 갖는 문자열 또는 식별자를 표현하는 데 널리 사용됩니다. 예를 들어 식물학 데이터 세트에 15,000가지 종이 수록되어 있으며 각 종이 고유한 문자열 식별자로 표기되어 있다고 가정해 보겠습니다. 특성 추출 시에는 이러한 문자열 식별자를 크기가 15,000인 원-핫 벡터로 인코딩할 수 있습니다.

원샷 학습(one-shot learning)

객체 분류에 자주 사용되는 머신러닝 접근방식으로서 하나의 학습 예에서 효과적인 분류자를 학습하는 것을 목적으로 합니다.

극소수 학습 (#few-shot_learning)을 참조하세요.

일대다(one-vs.-all)

가능한 해가 N개인 분류 문제에서, 일대다 솔루션은 가능한 각 결과에 하나씩 서로 다른 N개의 **이진 분류자** (#binary_classification)로 구성됩니다. 가령, 예를 동물, 식물, 광물 중 하나로 분류하는 모델에서는 일대다 솔루션이 다음과 같은 3가지 이진 분류자를 제공할 수 있습니다.

- 동물 또는 동물 아님
- 식물 또는 식물 아님
- 광물 또는 광물 아님

온라인 추론(online inference)

요청에 따라 **예측** (#prediction)을 생성합니다. **오프라인 추론** (#offline_inference)과 대비되는 개념입니다.

작업(Operation, op)

텐서플로우 그래프의 노드입니다. 텐서플로우에서는 **텐서** (#tensor)를 만들거나 조작하거나 삭제하는 모든 절차를 작업으로 간주합니다. 예를 들어 행렬 곱셈은 텐서 2개를 입력으로 취하고 텐서 하나를 출력으로 생성하는 작업입니다.

옵티마이저(optimizer)

경사하강법 (#gradient_descent) 알고리즘의 구체적인 구현입니다. 옵티마이저에 대한 텐서플로우의 기본 클래스는 tf.train.Optimizer (https://www.tensorflow.org/api_docs/python/tf/train/Optimizer)입니다. 다른 옵티마이저는 다음 중 하나 이상의 개념을 활용하여 주어진 **학습 세트** (#training_set)에서 경사하강법의 효과를 강화할 수 있습니다.

- momentum (https://www.tensorflow.org/api_docs/python/tf/train/MomentumOptimizer) (모멘텀)
- 업데이트 빈도 (AdaGrad (https://www.tensorflow.org/api_docs/python/tf/train/AdagradOptimizer) = ADAptive GRADient descent; Adam (https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer) = ADAptive with Momentum; RMSProp)
- 희소성/정규화 (Ftrl (https://www.tensorflow.org/api_docs/python/tf/train/FtrlOptimizer))

- 복잡한 수학적 개념 (Proximal (https://www.tensorflow.org/api_docs/python/tf/train/ProximalGradientDescentOptimizer) 등)

NN 구동 옵티마이저 (https://arxiv.org/abs/1606.04474)도 생각해 볼 수 있습니다.

이상점(outliers)

다른 대부분의 값과 동떨어진 값입니다. 머신러닝에서는 다음과 같은 경우가 이상점에 해당합니다.

- **가중치** (#weight)의 절대값이 높은 경우
- 예측된 값이 실제 값과 비교적 멀리 떨어진 경우
- 입력 데이터의 값이 평균에서 대략적으로 표준편차 3만큼 떨어진 경우

이상점은 모델 학습에서 문제를 일으키는 경우가 많습니다.

출력 레이어(output layer)

신경망의 '최종' 레이어입니다. 이 레이어에 답이 포함됩니다.

과적합(overfitting)

생성된 모델이 **학습 데이터** (#training_set)와 지나치게 일치하여 새 데이터를 올바르게 예측하지 못하는 경우입니다.

P

Pandas

열 중심의 데이터 분석 API입니다. 텐서플로우를 비롯하여 다양한 ML 프레임워크에서 pandas 데이터 구조 입력을 지원합니다. pandas 문서 (http://pandas.pydata.org/)를 참조하세요.

매개변수(parameter)

ML 시스템에서 스스로 학습하는 모델의 변수입니다. 예를 들어 ML 시스템에서 학습이 반복됨에 따라 **가중치** (#weight) 매개변수의 값이 서서히 학습됩니다. **초매개변수** (#hyperparameter)와 대비되는 개념입니다.

매개변수 서버(PS, Parameter Server)

모델의 **매개변수** (#parameter)를 분산형 환경에서 추적하는 작업입니다.

매개변수 업데이트(parameter update)

학습 중에 모델의 **매개변수** (#parameter)를 조정하는 작업으로서, 일반적으로 **경사하강법** (#gradient_descent)의 단일 반복 내에서 이루어집니다.

편미분(partial derivative)

하나를 제외한 모든 변수를 상수로 간주하고 구한 도함수입니다. 예를 들어 *f(x, y)*를 *x*로 미분한 편도함수는 *y*를 상수로 두고 *f*를 *x*만의 함수로 간주한 도함수입니다. *f*를 *x*로 미분한 편도함수는 방정식의 다른 변수를 모두 무시하고 *x*의 변화에만 집중합니다.

분할 전략(partitioning strategy)

매개변수 서버 (#Parameter_Server) 전반에서 여러 변수를 분할하는 알고리즘입니다.

성능(performance)

다음과 같은 의미로 중복으로 정의된 용어입니다.

- 소프트웨어 공학에서 보편적으로 사용되는 의미입니다. 즉, 특정 소프트웨어가 얼마나 빠르게 또는 효율적으로 실행되는지를 의미합니다.

- ML의 맥락에서는 특정 **모델** (#model)이 얼마나 정확한지를 성능으로 정의합니다. 즉, 모델의 예측이 얼마나 효과적인지입니다.

퍼플렉시티(perplexity)

모델 (#model)의 작업 수행 능력을 나타내는 척도입니다. 예를 들어 사용자가 스마트폰 키보드에 입력하는 단어의 처음 몇 글자를 읽고 가능한 완성 단어의 목록을 제안하는 작업을 수행한다고 가정해 보겠습니다. 이 작업의 대략적인 퍼플렉시티 P는 사용자가 실제로 입력하려는 단어가 목록에 포함될 때까지 제안해야 하는 추측 단어 수입니다.

퍼플렉시티와 **교차 엔트로피** (#cross-entropy)의 관계는 다음과 같습니다.

$$P = 2^{-\text{교차 엔트로피}}$$

파이프라인(pipeline)

머신러닝 알고리즘의 기반이 되는 인프라입니다. 파이프라인에는 데이터 수집, 학습 데이터 파일에 데이터 넣기, 하나 이상의 모델 학습, 프로덕션 환경으로 모델 내보내기 등이 포함됩니다.

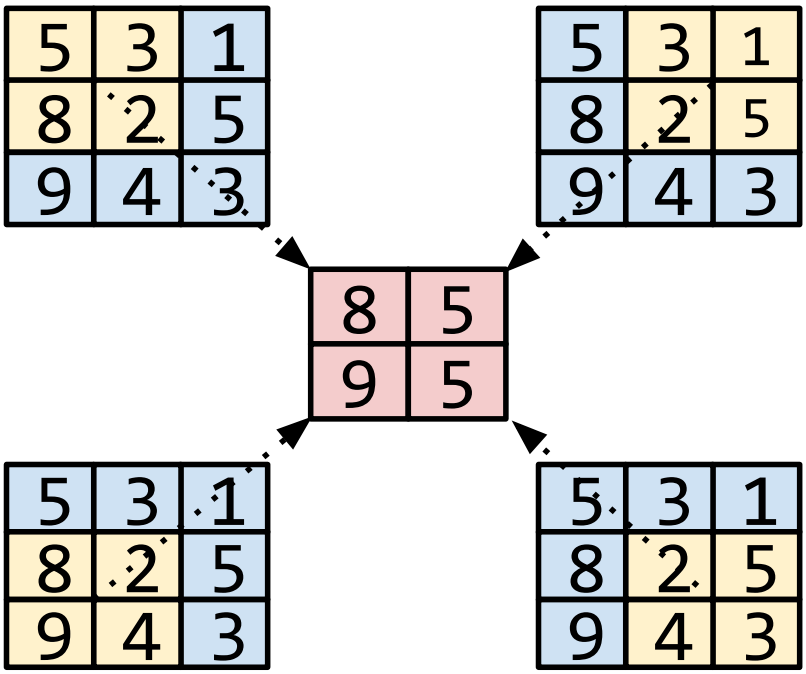
풀링(pooling)

이전의 **컨볼루셔널 레이어** (#convolutional_layer)에서 생성된 행렬을 작은 행렬로 줄이는 과정입니다. 풀링을 하면 보통 풀링된 영역에서 최대값 또는 평균값을 취하게 됩니다. 예를 들어 다음과 같은 3x3 행렬이 있다고 가정해 보겠습니다.

5	3	1
8	2	5
9	4	3

풀링 연산에서는 컨볼루셔널 연산에서와 마찬가지로 행렬을 슬라이스로 나눈 다음 **스트라이드** (#stride) 보폭으로 슬라이드하면서 컨볼루셔널 연산을 수행합니다. 예를 들어 풀링 연산에서 컨볼루셔널 행렬을 1x1 스트라이드로 2x2 슬라이스가 되게 나눈다고 가정해 보겠습니다. 다음 다이어그램에서 확인할 수 있

듯이 풀링 연산이 네 번 수행됩니다. 각 풀링 연산에서 슬라이스에서 얻은 네 개의 값 중 최대값을 취한다고 생각해 보세요.



풀링을 사용하면 입력 행렬에서 **병진 불변** (#translational_invariance)을 사용할 수 있습니다.

시각적으로 활용되는 풀링은 공식적으로 **공간 풀링**으로 더 잘 알려져 있습니다. 시계열에서 활용되는 풀링은 **일시적 풀링**이라고 합니다. 비공식적으로 풀링을 **서브샘플링** 또는 **다운샘플링**이라고 부르는 경우도 있습니다.

포지티브 클래스(positive class)

이진 분류 (#binary_classification)에서 가능한 두 가지 클래스에 포지티브 및 네거티브 라벨이 지정됩니다. 모델을 테스트하는 목적은 포지티브한 결과를 찾는 것입니다. 사실은 두 가지 결과를 동시에 테스트하는 셈이지만 우선은 이렇게 이해해도 무방합니다. 예를 들어 의료 검사의 포지티브 클래스는 '종양'일 수 있습니다. 이메일 분류기의 포지티브 클래스는 '스팸'일 수 있습니다.

네거티브 클래스 (#negative_class)와 대비되는 개념입니다.

정밀도(precision)

분류 모델 (#classification_model)의 측정항목입니다. 정밀도는 모델이 **포지티브 클래스** (#positive_class)를 정확히 예측한 빈도를 나타냅니다. 즉 다음과 같습니다.

$$\text{정밀도} = \frac{\text{참긍정}}{\text{참긍정} + \text{거짓긍정}}$$

예측(prediction)

모델에 입력 **예** (#example)를 제공하여 생성된 출력입니다.

예측 편향(prediction bias)

예측 (#prediction)의 평균이 데이터 세트의 **라벨** (#label) 평균과 얼마나 차이가 나는지 나타내는 값입니다.

사전 제작된 에스티메이터(pre-made Estimator)

다른 사람이 제작한 **에스티메이터** (#Estimator)입니다. 텐서플로우는 `DNNClassifier`, `DNNRegressor`, `LinearClassifier` 등의 몇 가지 사전 제작된 에스티메이터를 제공합니다. 여기의 안내 (https://www.tensorflow.org/extend/estimators)에 따라 사전 제작된 에스티메이터를 직접 만들 수 있습니다.

선행 학습된 모델(pre-trained model)

이미 학습이 끝난 모델 또는 모델 구성요소(예: **임베딩** (#embeddings))입니다. 선행 학습된 임베딩을 **신경망** (#neural_network)에 입력하는 경우도 있습니다. 모델에서 선행 학습된 임베딩(embeddings)에 의존하지 않고 임베딩(embeddings) 자체를 학습하는 경우도 있습니다.

사전 믿음(prior belief)

학습을 시작하기 전에 데이터에 대해 갖는 견해입니다. 예를 들어 **L_2 정규화** (#L2_regularization)는 **가중치** (#weight)가 작으며 보통 0 근처에 분포한다는 사전 믿음에 의존합니다.

Q

대기열(queue)

대기열 데이터 구조를 구현하는 텐서플로우 **작업** (#Operation)입니다. 일반적으로 I/O에 사용됩니다.

R

순위(rank)

ML에서 중복으로 정의된 용어로서 다음 중 하나를 의미할 수 있습니다.

- **텐서** (#tensor)의 차원 수입니다. 예를 들어 스칼라의 순위는 0이고, 벡터의 순위는 1이고, 행렬의 순위는 2입니다.
- 클래스를 오름차순으로 분류하는 ML 문제에서 클래스의 서수 위치입니다. 예를 들어 행동 순위 시스템은 강아지의 보상에 가장 높은 순위(스테이크)부터 가장 낮은 순위(시든 양배추)까지 매길 수 있습니다.

평가자(rater)

예 (#example)에 **라벨** (#label)을 제공하는 사람입니다. '평정자'라고도 합니다.

재현율(recall)

분류 모델 (#classification_model)에 대해 다음과 같은 의문에 답하는 측정항목입니다. 가능한 모든 긍정 라벨 중에서 모델이 올바르게 식별한 것은 몇 개일까요? 즉 다음과 같습니다.

$$\text{재현율} = \frac{\text{참긍정}}{\text{참긍정} + \text{거짓부정}}$$

정류 선형 유닛(ReLU, Rectified Linear Unit)

다음 규칙을 따르는 **활성화 함수** (#activation_function)입니다.

- 입력이 음수 또는 0이면 출력은 0입니다.
- 입력이 양수이면 출력은 입력과 같습니다.

회귀 모형(regression model)

연속(일반적으로 부동 소수점) 값을 출력하는 모델 유형입니다. 개나리', '진달래' 같은 불연속 값을 출력하는 **분류 모델** (#classification_model)과 비교되는 개념입니다.

정규화(regularization)

모델의 복잡도에 페널티를 부여합니다. 정규화는 **과적합** (#overfitting)을 방지하는 데 도움이 됩니다. 정규화의 여러 가지 종류는 다음과 같습니다.

- **L_1 정규화** (#L1_regularization)
- **L_2 정규화** (#L2_regularization)
- **드롭아웃 정규화** (#dropout_regularization)
- **조기 중단** (#early_stopping)(정식으로 인정되는 정규화 방식은 아니지만 과적합을 효과적으로 제한할 수 있음)

정규화율(regularization rate)

람다로 표현되는 스칼라값으로서, 정규화 함수의 상대적 중요도를 지정합니다. 다음은 정규화율의 영향을 보여주는 단순화된 **손실** (#loss) 방정식입니다.

$$\text{최소화}(\text{손실 함수} + \lambda(\text{정규화 함수}))$$

정규화율을 높이면 **과적합** (#overfitting)이 감소하지만 모델의 **정확성** (#accuracy)이 떨어질 수 있습니다.

표현(representation)

데이터를 유용한 **특성** (#feature)에 매핑하는 과정입니다.

수신자 조작 특성 곡선(ROC curve, Receiver Operating Characteristic curve)

다양한 **분류 임계값** (#classification_threshold)에서 **참양성률** (#TP_rate)과 **거짓양성률** (#FP_rate)이 이루는 곡선입니다. **AUC** (#AUC)를 참조하세요.

루트 디렉토리(root directory)

여러 모델의 텐서플로우 체크포인트 및 이벤트 파일의 하위 디렉토리를 호스팅하도록 지정된 디렉토리입니다.

평균 제곱근 오차(RMSE, Root Mean Squared Error)

평균 제곱 오차 (#MSE)의 제곱근입니다.

회전 불변(rotational invariance)

이미지 분류 문제에서 알고리즘이 이미지의 방향이 바뀌더라도 이미지를 분류해 낼 수 있는 능력을 의미합니다. 예를 들어 이러한 능력을 갖춘 알고리즘은 위쪽, 옆쪽, 아래쪽 등 어느 방향을 향하고 있건 테니스 라켓을 식별해 낼 수 있습니다. 회전 불변이 항상 바람직한 것은 아닙니다. 예를 들어 거꾸로 된 9는 9로 분류되어서는 안 됩니다.

병진 불변 (#translational_invariance) 및 **크기 불변** (#size_invariance)을 참조하세요.

S

SavedModel

텐서플로우 모델을 저장하고 복구하는 데 권장되는 형식입니다. SavedModel은 언어 중립적이며 복구 가능한 직렬화 형식으로서 상위레벨 시스템 및 도구에서 텐서플로우 모델을 생성, 사용 및 변환하도록 지원합니다.

자세한 내용은 텐서플로우 프로그래머 가이드에서 저장 및 복원 (https://www.tensorflow.org/programmers_guide/saved_model)을 참조하세요.

Saver

모델 체크포인트 저장을 담당하는 텐서플로우 개체 (https://www.tensorflow.org/api_docs/python/tf/train/Saver)입니다.

조정(scaling)

특성 추출 (#feature_engineering)에서 널리 사용되는 방식으로서 특성 값 범위를 데이터 세트의 다른 특성 범위와 일치하도록 맞춥니다. 예를 들어 데이터 세트의 모든 부동 소수점 특성을 0~1 범위로 맞출 수 있습니다. 어떤 특성의 범위가 0~500이라면 각 값을 500으로 나누어 특성을 조정할 수 있습니다.

정규화 (#normalization)를 참조하세요.

scikit-learn

널리 사용되는 오픈소스 ML 플랫폼입니다. www.scikit-learn.org (http://www.scikit-learn.org/)를 참조하세요.

준지도 학습(semi-supervised learning)

학습 예 중 일부에는 라벨이 있으며 일부에는 없는 데이터로 모델을 학습시킵니다. 준지도 학습의 기법 중 하나는 라벨이 없는 예의 라벨을 추론한 후 해당 라벨로 학습하여 새 모델을 만드는 것입니다. 준지도 학습은 라벨이 없는 예가 풍부하지만 라벨을 획득하는 비용이 많이 드는 경우에 유용할 수 있습니다.

시퀀스 모델(sequence model)

입력에 순서 종속성이 있는 모델입니다. 예를 들면 이전에 시청한 동영상의 순서를 바탕으로 다음에 시청할 동영상을 예측하는 경우입니다.

세션(session, tf.session)

텐서플로우 런타임 상태를 캡슐화하고 **그래프** (#graph)의 전부 또는 일부를 실행하는 개체입니다. 낮은 수준의 TensorFlow API를 사용하면 하나 이상의 `tf.session` 개체를 직접 인스턴스화하고 관리합니다. Estimators API를 사용하면 에스티메이터가 세션 개체를 인스턴스화합니다.

시그모이드 함수(sigmoid function)

로지스틱 또는 다항 회귀 출력(로그 확률)을 확률에 매핑하여 0~1 사이의 값을 반환하는 함수입니다. 시그모이드 함수의 공식은 다음과 같습니다.

$$y = \frac{1}{1 + e^{-\sigma}}$$

여기에서 **로지스틱 회귀** (#logistic_regression) 문제의 σ는 다음과 같습니다.

$$\sigma = b + w_1 x_1 + w_2 x_2 + \ldots w_n x_n$$

즉, 시그모이드 함수는 σ를 0~1 사이의 확률로 변환합니다.

일부 **신경망** (#neural_network)에서는 시그모이드 함수가 **활성화 함수** (#activation_function) 역할을 합니다.

크기 불변(size invariance)

이미지 분류 문제에서 알고리즘이 이미지의 크기가 바뀌더라도 이미지를 분류해 낼 수 있는 능력을 의미합니다. 예를 들어 이러한 능력을 갖춘 알고리즘은 픽셀이 2백만 개든 2십만 개든 간에 고양이를 식별해 낼 수 있습니다. 가장 우수한 이미지 분류 알고리즘도 크기 불변에 있어서 실질적인 제약이 있습니다. 예를 들어 고작 20픽셀만 사용하는 고양이 이미지를 알고리즘 또는 인간이 올바르게 분류해 낼 가능성은 낮습니다.

병진 불변 (#translational_invariance) 및 **회전 불변** (#rotational_invariance)을 참조하세요.

소프트맥스(softmax)

다중 클래스 분류 모델 (#multi-class)에서 가능한 각 클래스의 확률을 구하는 함수입니다. 확률의 합은 정확히 1.0입니다. 예를 들어 소프트맥스는 특정 이미지가 강아지일 확률을 0.9로, 고양이일 확률을 0.08로, 말일 확률을 0.02로 판단할 수 있습니다. **전체 소프트맥스**라고도 합니다.

후보 샘플링 (#candidate_sampling)과 대비되는 개념입니다.

희소 특성(sparse feature)

대부분의 값이 0이거나 비어 있는 **특성** (#feature) 벡터입니다. 예를 들어 1 값 하나와 0 값 백만 개를 포함하는 벡터는 희소 벡터입니다. 또 다른 예로서, 검색어의 단어는 희소 특성일 수 있습니다. 특정 언어에서 가능한 단어는 무수히 많지만 특정 검색어에는 몇 개의 단어만 나오기 때문입니다.

밀집 특성 (#dense_feature)과 대비되는 개념입니다.

희소 표현(sparse representation)

값이 0이 아닌 요소만 저장하는 텐서의 **표현** (#representation)입니다.

예를 들어 영어 어휘에는 약 백만 개의 단어들이 있습니다. 하나의 영어 문장에서 어떤 단어가 몇 번 사용되었는지 나타내주는 다음의 두 가지 방법을 생각해 봅시다.

- 이 문장을 **밀집 표현** 방식으로 나타내면 백만 개의 셀들로 구성된 정수 백터를 정의한 다음 벡터의 셀 대부분에는 0을 할당하고, 극히 일부 셀에는 작은 정수값을 할당해야 합니다.
- 이 문장을 희소 표현 방식으로 나타내면 실제로 문장에 있는 단어를 나타내는 셀만 저장합니다. 그러므로 20개의 고유한 단어가 포함된 문장을 희소 표현 방식으로 나타내면 문장에 포함된 단어에 해당하는 20개의 셀만 저장하면 됩니다.

'개가 꼬리를 흔든다.'라는 문장을 두 가지 방법으로 나타낼 수 있습니다. 다음 표에서 보듯이 밀집 표현 방식으로 나타내려면 거의 백만 개에 달하는 셀을 사용해야 하지만, 희소 표현 방식에서는 세 개의 셀만 사용하면 됩니다.

밀집 표현

셀 개수	단어	발생 횟수
0	가	0

셀 개수	단어	발생 횟수
1	갸	0
2	거	0
3	겨	0
	… 발생 횟수가 **0**인 단어가 **140,391**개 더 있음	
140395	개가	1
	… 발생 횟수가 **0**인 단어가 **633,062**개 더 있음	
773458	꼬리를	1
	… 발생 횟수가 **0**인 단어가 **189,136**개 더 있음	
962594	흔든다	1

… 발생 횟수가 **0**인 단어가 매우 많음

희소 표현

셀 개수	단어	발생 횟수
140395	개가	1
773458	꼬리를	1
962594	흔든다	1

희소성(sparsity)

벡터 또는 행렬에서 0 또는 null로 설정된 요소의 개수를 벡터 또는 행렬의 총 항목 개수로 나눈 값입니다. 예를 들어 10x10 행렬이 있고 이 중 98개의 셀에 0이 들어 있다고 생각해 보겠습니다. 이때 희소성은 다음과 같이 계산됩니다.

$$\text{희소성} = \frac{98}{100} = 0.98$$

특성 희소성은 특성 벡터의 희소성을 의미하며 **모델 희소성**은 모델 가중치의 희소성을 의미합니다.

공간 풀링(spatial pooling)

풀링 (#pooling)을 참조하세요.

제곱 힌지 손실(squared hinge loss)

힌지 손실 (#hinge-loss)의 제곱입니다. 제곱 힌지 손실은 일반 힌지 손실보다 이상점에 더 많은 페널티를 가합니다.

제곱 손실(squared loss)

선형 회귀 (#linear_regression)에 사용되는 **손실** (#loss) 함수입니다. **L_2 손실**이라고도 합니다. 이 함수는 라벨이 있는 **예** (#example)에 대한 모델의 예측 값과 **라벨** (#label)의 실제 값 차이의 제곱을 계산합니다. 이 손실 함수는 제곱을 구하므로 부정확한 예측에 더 큰 영향을 줍니다. 즉, 제곱 손실은 **L_1 손실** (#L1_loss)보다 이상점에 민감하게 반응합니다.

정적 모델(static model)

오프라인으로 학습되는 모델입니다.

정상성(stationarity)

데이터 세트의 데이터가 갖는 속성으로서, 하나 이상의 차원에서 데이터 분포가 일정하게 유지되는 경우입니다. 이 차원은 대부분의 경우 시간으로서, 시간에 따라 변화하지 않는 데이터는 정상성을 갖는 것입니다. 예를 들어 9월에서 12월까지 변화가 없는 데이터는 정상성을 갖습니다.

단계(step)

배치 (#batch) 하나에 대한 정방향 및 역방향 평가입니다.

보폭(step size)

학습률 (#learning_rate)의 동의어입니다.

확률적 경사하강법(SGD, stochastic gradient descent)

배치 크기가 1인 **경사하강법** (#gradient_descent) 알고리즘입니다. 즉, 확률적 경사하강법은 데이터 세트에서 무작위로 균일하게 선택한 하나의 예에 의존하여 각 단계의 예측 경사를 계산합니다.

구조적 위험 최소화(SRM, structural risk minimization)

다음 두 목표 사이에서 균형을 찾는 알고리즘입니다.

- 예측이 가장 정확한 모델을 만듭니다. 예를 들어 손실을 최소화합니다.
- 모델을 최대한 단순하게 유지합니다. 예를 들어 강력한 정규화를 적용합니다.

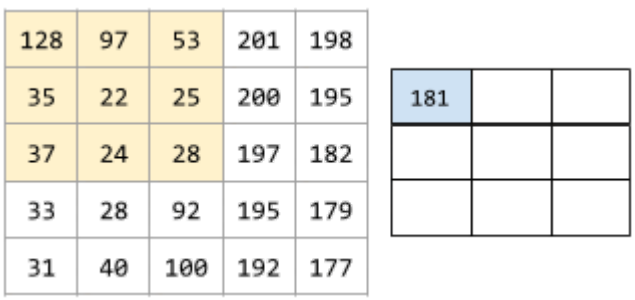
예를 들어 손실을 최소화하면서 학습 세트에 정규화를 적용하는 함수는 구조적 위험 최소화 알고리즘입니다.

자세한 내용은 http://www.svms.org/srm/ (http://www.svms.org/srm/)을 참조하세요.

경험적 위험 최소화 (#ERM)와 대비되는 개념입니다.

스트라이드(stride)

스트라이드란 컨볼루셔널 연산이나 풀링에서 다음번에 오는 일련의 입력 슬라이스의 차원별 델타 값을 말합니다. 예를 들어 다음의 애니메이션은 컨볼루셔널 연산의 (1, 1) 스트라이드를 보여줍니다. 따라서 다음 입력 슬라이스는 이전 입력 슬라이스로부터 오른쪽으로 한 칸 이동한 위치에서 시작합니다. 연산이 오른쪽 가장자리에 다다르면 다음 슬라이스는 한 줄 아래 왼쪽 가장자리에서 시작합니다.



앞의 예는 2차원 스트라이드를 보여줍니다. 입력 행렬이 3차원인 경우 스트라이드도 3차원이 됩니다.

서브 샘플링(subsampling)

풀링 (#pooling)을 참조하세요.

요약(summary)

텐서플로우에서는 특정 **단계** (#step)에 계산된 값 또는 값 집합을 의미하며, 일반적으로 학습 중에 모델 측정항목을 추적하는 데 사용됩니다.

지도 머신러닝(supervised machine learning)

입력 데이터 및 해당 **라벨** (#label)을 사용하여 **모델** (#model)을 학습시킵니다. 지도 머신러닝은 학생이 특정 과목을 배우기 위해 일련의 문제와 정답을 공부하는 것과 같습니다. 기존 문제의 정답을 맞힐 수 있게 된 학생은 같은 과목에서 처음 보는 새로운 문제를 풀 수 있습니다. **비지도 머신러닝** (#unsupervised_machine_learning)과 비교되는 개념입니다.

합성 특성(synthetic feature)

입력 특성 중에는 없지만 하나 이상의 입력 특성으로부터 생성되는 **특성** (#feature)입니다. 합성 특성에는 다음과 같은 종류가 있습니다.

- 연속 특성을 범위 빈으로 **버케팅** (#bucketing)합니다.
- 하나의 특성 값에 다른 특성 값이나 해당 특성 값을 곱하거나 나눕니다.
- **특성 교차** (#feature_cross)를 생성합니다.

정규화 (#normalization) 또는 **조정** (#scaling)만으로 생성한 특성은 합성 특성에 해당하지 않습니다.

T

타겟(target)

라벨 (#label)의 동의어입니다.

시계열 데이터(temporal data)

서로 다른 여러 시점에 기록된 데이터입니다. 예를 들어 겨울 코트 매출액을 날짜별로 기록한 데이터는 시계열 데이터입니다.

텐서(Tensor)

텐서플로우 프로그램의 기본 데이터 구조입니다. 텐서는 대부분 스칼라, 벡터 또는 행렬로 이루어진 N차원 데이터 구조이며, N은 매우 큰 수일 수 있습니다. 텐서의 요소는 정수, 부동 소수점 또는 문자열 값을 포함할 수 있습니다.

텐서 처리 장치(TPU, Tensor Processing Unit)

텐서플로우 프로그램의 성능을 최적화하는 ASIC(Application-Specific Integrated Circuit)입니다.

텐서 차수(Tensor rank)

순위 (#rank)를 참조하세요.

텐서 형태(Tensor shape)

텐서 (#tensor)가 여러 차원에 포함하는 요소 수입니다. 예를 들어 [5, 10] 텐서의 형태는 1차원에서 5, 2차원에서 10입니다.

텐서 크기(Tensor size)

텐서 (#tensor)가 포함하는 스칼라의 총 개수입니다. 예를 들어 [5, 10] 텐서의 크기는 50입니다.

텐서보드(TensorBoard)

하나 이상의 텐서플로우 프로그램을 실행하는 중에 저장된 요약을 표시하는 대시보드입니다.

텐서플로우(TensorFlow)

대규모 분산형 머신러닝 플랫폼입니다. 이 용어는 데이터 흐름 그래프의 일반적인 계산을 지원하는 텐서플로우 스택의 기본 API 레이어를 지칭하기도 합니다.

텐서플로우는 머신러닝에 주로 사용되지만, ML과 관계가 없더라도 데이터 흐름 그래프를 사용한 수치 연산을 필요로 하는 다른 작업에도 텐서플로우를 사용할 수 있습니다.

텐서플로우 플레이그라운드(TensorFlow Playground)

다양한 **초매개변수** (#hyperparameters)가 모델(주로 신경망) 학습에 주는 영향을 시각적으로 보여 주는 프로그램입니다. 텐서플로우 플레이그라운드로 실험해 보려면 http://playground.tensorflow.org (http://playground.tensorflow.org)로 이동하세요.

텐서플로우 서빙(TensorFlow Serving)

학습된 모델을 프로덕션 환경에 배포하는 플랫폼입니다.

테스트 세트(test set)

데이터 세트 중에서 검증세트를 통한 초기 검증이 완료된 **모델** (#model)을 테스트하는 데 사용되는 부분집합입니다.

학습 세트 (#training_set) 및 **검증세트** (#validation_set)와 대비되는 개념입니다.

tf.Example

머신러닝 모델 학습 또는 추론을 위한 입력 데이터를 기술하는 표준 프로토콜 버퍼 (https://developers.google.com/protocol-buffers/)입니다.

시계열 분석(time series analysis)

머신러닝 및 통계학에서 **시계열 데이터** (#temporal_data)를 분석하는 하위 분야입니다. 분류, 클러스터링, 예측, 이상 감지 등 다양한 유형의 머신러닝 문제에 시계열 분석이 요구됩니다. 예를 들어 시계열 분석을 사용하여 과거 매출 데이터를 근거로 겨울 코트의 향후 월별 매출을 예측할 수 있습니다.

학습(training)

모델을 구성하는 이상적인 **매개변수** (#parameter)를 결정하는 과정입니다.

학습 세트(training set)

데이터 세트 중에서 모델 학습에 사용되는 부분집합입니다.

검증세트 (#validation_set) 및 **테스트 세트** (#test_set)와 대비되는 개념입니다.

전이 학습(transfer learning)

머신러닝 작업 간에 정보를 전송합니다. 예를 들어 다중 작업 학습에서는 단일 모델이 여러 작업을 해결합니다. 예를 들어 **심층 모델** (#deep_model)은 다양한 작업에 대해 서로 다른 출력 노드를 가질 수 있습니다. 전이 학습에서는 더 단순한 작업의 솔루션 지식을 보다 복잡한 작업으로 전송하거나, 데이터가 더 많은 작업의 지식을 데이터가 더 적은 작업으로 전송합니다.

대부분의 머신러닝 시스템에서는 *단일* 작업을 해결합니다. 전이 학습은 하나의 프로그램으로 *여러* 작업을 해결할 수 있는 인공 지능 개발의 초석이 됩니다.

병진 불변(translational invariance)

이미지 분류 문제에서 알고리즘이 이미지에 있는 개체의 위치가 바뀌더라도 이미지를 분류해 낼 수 있는 능력을 의미합니다. 예를 들어 이러한 능력을 갖춘 알고리즘은 강아지가 프레임의 중앙에 있든 왼쪽 끝에 있든 간에 강아지를 식별해 낼 수 있습니다.

크기 불변 (#size_invariance) 및 **회전 불변** (#rotational_invariance)을 참조하세요.

참음성(TN, true negative)

모델에서 *네거티브 클래스*로 **올바르게** (#negative_class) 예측한 예입니다. 예를 들어 모델에서 특정 이메일 메시지가 스팸이 아닌 것으로 추론했으며 실제로도 스팸이 아니었던 경우가 여기에 해당합니다.

참긍정(TP, true positive)

모델에서 *포지티브 클래스*로 **올바르게** (#positive_class) 예측한 예입니다. 예를 들어 모델에서 특정 이메일 메시지가 스팸인 것으로 추론했으며 실제로도 스팸이었던 경우가 여기에 해당합니다.

참긍정률(TP rate, true positive rate)

재현율 (#recall)의 동의어입니다. 즉 다음과 같습니다.

$$\text{참긍정률} = \frac{\text{참긍정}}{\text{참긍정} + \text{거짓부정}}$$

참양성률은 **ROC 곡선** (#ROC)의 y축입니다.

U

라벨이 없는 예(unlabeled example)

특성 (#feature)은 있지만 **라벨** (#label)이 없는 예입니다. 라벨이 없는 예는 **추론** (#inference)에 대한 입력입니다. **준지도** (#semi-supervised_learning) 및 **비지도** (#unsupervised_machine_learning) 학습에서는 학습 중에 라벨이 없는 예가 사용됩니다.

비지도 머신러닝(unsupervised machine learning)

일반적으로 라벨이 없는 데이터 세트에서 패턴을 찾도록 **모델** (#model)을 학습시킵니다.

비지도 머신러닝의 가장 일반적인 용도는 데이터를 서로 비슷한 예의 그룹으로 클러스터링하는 것입니다. 예를 들어 비지도 머신러닝 알고리즘은 음악의 다양한 속성을 기반으로 곡을 서로 클러스터링할 수 있습니다. 결과 클러스터는 음악 추천 서비스 등의 다른 머신러닝 알고리즘에 입력으로 사용될 수 있습니다. 클러스터링은 정확한 라벨을 확보하기 어려운 분야에서 유용합니다. 예를 들어 악용 및 사기 행위 방지와 같은 분야에서 클러스터는 사람이 데이터를 이해하는 데 도움을 줄 수 있습니다.

비지도 머신러닝의 또 다른 예는 **주성분 분석(PCA)** (https://en.wikipedia.org/wiki/Principal_component_analysis)입니다. 예를 들어 장바구니 수백만 개의 내용을 포함하는 데이터 세트에 PCA를 적용하면 레몬이 들어있는 장바구니에 제산제가 같이 들어있는 경우가 많다는 사실이 드러날 수 있습니다.

지도 머신러닝 (#supervised_machine_learning)과 비교되는 개념입니다.

V

검증세트(validation set)

데이터 세트 중에서 학습 세트와 별도로 **초매개변수** (#hyperparameter)를 조정하는 데 사용하는 부분집합입니다.

학습 세트 (#training_set) 및 **테스트 세트** (#test_set)와 대비되는 개념입니다.

W

가중치(weight)

선형 모델에서 **특성** (#feature)의 계수 또는 심층 네트워크의 엣지입니다. 선형 모델 학습의 목표는 각 특성의 이상적인 가중치를 결정하는 것입니다. 가중치가 0인 특성은 모델에 영향을 주지 못합니다.

와이드 모델(wide model)

일반적으로 많은 **희소 입력 특성** (#sparse_features)을 갖는 선형 모델입니다. 이러한 모델은 출력 노드에 직접 연결되는 많은 수의 입력을 갖는 특수한 유형의 **신경망** (#neural_network)이므로 '와이드'로 지칭됩니다. 와이드 모델은 심층 모델보다 디버그 및 조사가 더 쉬운 경우가 많습니다. 와이드 모델은 **히든 레이어** (#hidden_layer)를 통한 비선형성 표현이 불가능하지만 **특성 교차** (#feature_cross), **버킷화** (#bucketing) 등의 변환을 사용하여 다른 방식으로 비선형성을 모델링할 수 있습니다.

심층 모델 (#deep_model)과 대비되는 개념입니다.

9월 7, 2018에 마지막으로 업데이트되었습니다.