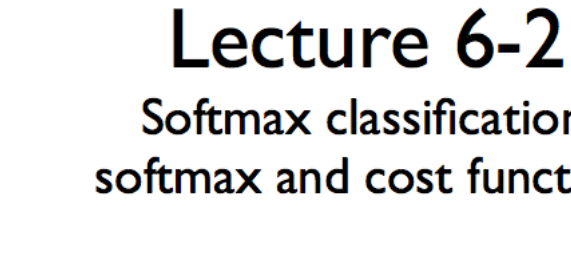
Softmax classifier 의 cost함수 (lec 06-2)



동영상이 2개로 되어 있고, 이전 글에서 sigmoid는 어디에 있는가,까지 진행했다.

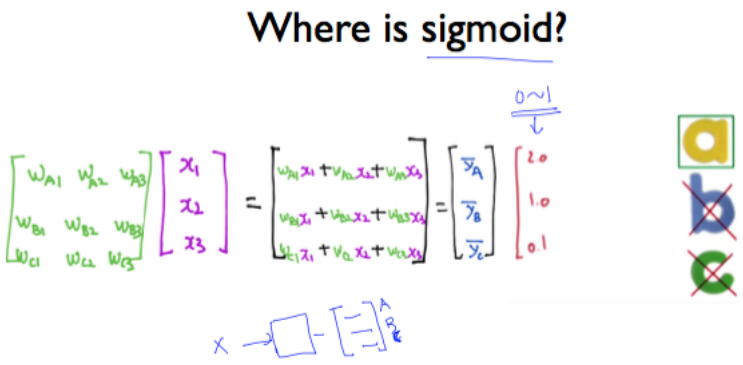
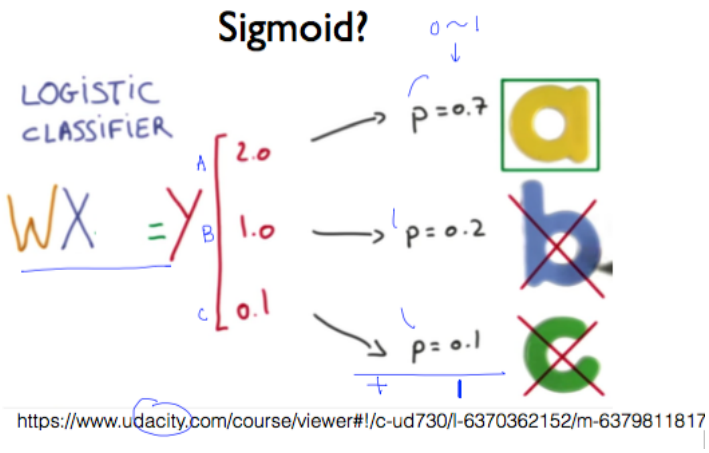


그림 중간에 빨강색으로 표시된 2.0, 1.0, 0.1이 예측된 Y의 값이다. 이것을 Y hat이라고 부른다고 했다. 이 값은 W에 X를 곱하기 때문에 굉장히 크거나 작은 값일 수 있다. 그래서, 이 부분 뒤쪽에 sigmoid가 들어가서 값을 0과 1 사이로 조정하게 된다.



공지사항

파이썬 동영상 (/notice/
머신러닝 동영상 (/notice/76)
머신러닝 목차 (/notice/

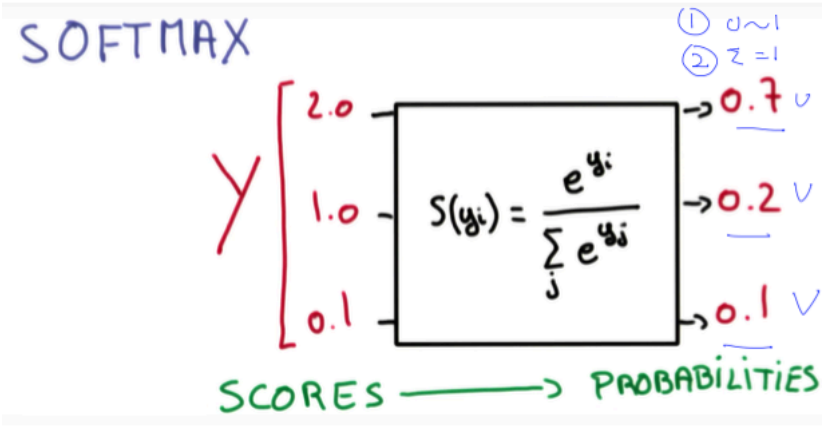
카테고리

분류 전체보기 (87) (/category)
프로필 (0) (/category/%ED%94%8
파이썬 (12) (/category/%ED%8C%8
머신러닝 (6) (/category/%EB%A8%B
머신러닝_김성훈교수님 (45) (/category/%EB%A8%B
텐서플로우 (12) (/category/%ED%85%9
챗봇 (2) (/category/%EC%B1%9
이것저것 (10) (/category/%EC%9D%B

얼떨결에 붙인 파이썬_킴

그림에서 예측한 결과 y가 1개가 아니라 3개라는 점은 진짜 중요하다. 선택 가능한 옵션이 a, b, c의 3개가 있어서 binary classification을 3개 사용했고 각각의 결과를 저장해야 하므로 3개가 된다. binary classification을 세 번에 걸쳐 적용하고 있다는 것을 기억하자.

이들 값을 0과 1 사이의 값으로 바꾸니까, 각각 0.7, 0.2, 0.1이 됐다. 이들을 모두 더하면 1이 된다. a, b, c 중에서 하나를 고르라면 a를 선택하게 된다.

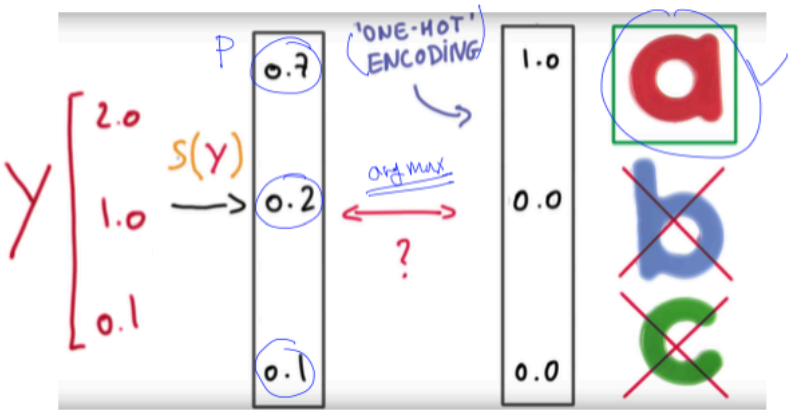


https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

softmax는 점수로 나온 결과를 전체 합계가 1이 되는 0과 1 사이의 값으로 변경해 준다. 전체를 더하면 1이 되기 때문에 확률(probabilites)이라고 부르면 의미가 더욱 분명해진다. 0.7이라는 뜻은 70%의 확률로 a가 될 수 있다는 뜻이다. 검정 상자 안에는 softmax를 구현하는 공식이 표시되어 있는데, 텐서플로우에서는 softmax 함수가 있어서 그냥 호출하면 끝난다.

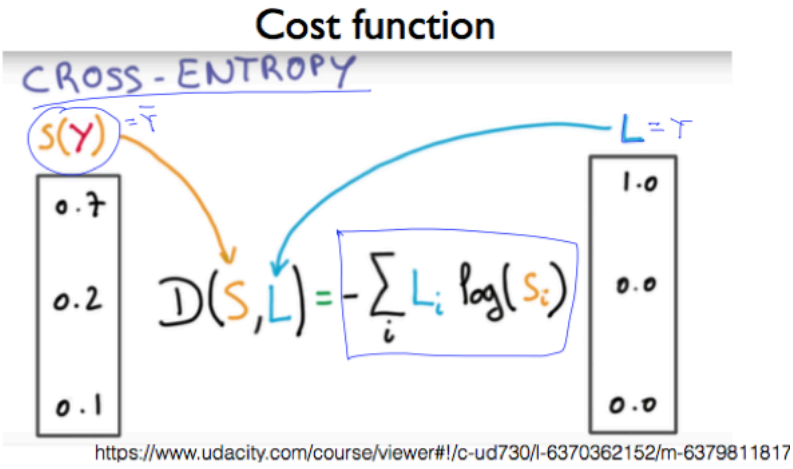
교수님께서는 이 부분을 가볍게 말씀하셨는데, 공식이 어렵지 않고, 코드로 구현하는 것도 어렵지 않다. 확률이기 때문에 전체 합계는 1이 되어야 한다. 다시 말해, 나의 크기가 전체 크기 중에서 어느 정도인지를 판단하면 되기 때문에 (내 크기/전체 합계) 공식으로 표현할 수 있다. 분모에는 시그마(Σ)가 있고, 분자에는 없는 것을 볼 수 있다.

softmax는 두 가지 역할을 수행한다.
1. 입력을 sigmoid와 마찬가지로 0과 1 사이의 값으로 변환한다.
2. 변환된 결과에 대한 합계가 1이 되도록 만들어 준다.



https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

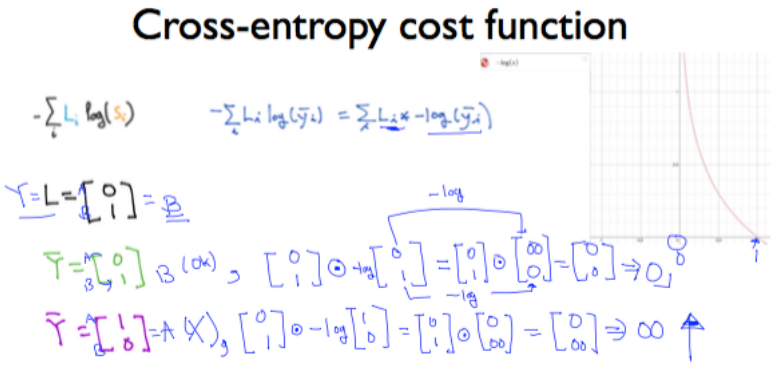
y를 예측한 이후부터의 과정을 알려주는 그림이다. one-hot encoding은 softmax로 구한 값 중에서 가장 큰 값을 1로, 나머지를 0으로 만든다. 어떤 것을 선택할지를 확실하게 정리해 준다. one-hot encoding은 설명한 것처럼 매우 간단하기 때문에 직접 구현할 수도 있지만, 텐서플로우에서는 argmax 함수라는 이름으로 제공하고 있다.



지금까지 글을 작성하면서 가장 어려웠던 부분이 cost 함수였던 것 같다. 이번에도 여지없이 비용을 측정하는 cost 함수가 나왔고, 색깔이 그다지 좋지 않다.

entropy는 열역학에서 사용하는 전문 용어로 복잡도 내지는 무질서량을 의미한다. 엔트로피가 크다는 것은 복잡하다는 뜻이다. cross-entropy는 통계학 용어로, 두 확률 분포 p와 q 사이에 존재하는 정보량을 계산하는 방법을 말한다. 다행스럽게 cross-entropy라는 용어에 엄청나게 심오한 이론이 숨어있는 것 같지는 않다.

S(Y)는 softmax가 예측한 값이고, L(Y)는 실제 Y의 값으로 L은 label을 의미한다. cost 함수는 예측한 값과 실제 값의 거리(distance, D)를 계산하는 함수로, 이 값이 줄어드는 방향으로, 즉 entropy가 감소하는 방향으로 진행하다 보면 최저점을 만나게 된다.



cross-entropy cost 함수가 제대로 동작한다는 것을 풀어서 설명하고 있다. 공식의 오른쪽에 나타난 log는 logistic regression에서 이미 봤다. 그림 오른쪽에 있는 것처럼 1을 전달하면 y는 0이 되고, 0을 전달하면 y는 무한대가 된다. 이때, 비용이 최소로 나오는 것을 선택해야 하므로, 전체 결과가 무한대가 나온다면 선택할 수 없다는 것을 뜻한다.

지금 그림은 두 가지 중에서 한 가지를 선택하는 것을 보여주고 있다. 이전 그림은 a, b, c 중에서 선택을 하는 것이기 때문에 이번 그림과는 일부 맞지 않는다. 세 가지라면 label에 들어가는 값이 [0, 1]이 아니라 a(1,0,0)나 b(0,1,0)처럼 세 개의 값이 들어가는 부분이 다르다. 추가적으로 y hat 또한 3개가 될 것이고, 계산은 조금 더 복잡해질 것이다.

그런데, 처음 공식의 log에는 S가 들어가 있었는데, 실제 계산은 y hat으로 하고 계신다. 0과 1로 log를 취하면 결과가 분명하기 때문에 그렇게 하신 것이지만, 의미가 분명하지 않을 수도 있다. 가령, S는 sigmoid를 가리키고 0부터 1 사이의 실수이므로 실제 결과는 0 또는 무한대가 아니라 작은 값이나 큰 값이 나오게 된다. 이들 값을 취합해서 cost를 계산하고 그에 따라 weight를 계산하면서 조절해서 최저점을 찾게 된다.

$$-\sum_i L_i \log(S_i) \quad => \quad -\sum_i L_i \log(\hat{y}_i) \quad => \quad \sum_i L_i * -\log(\hat{y}_i)$$

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, \quad cost=0$$

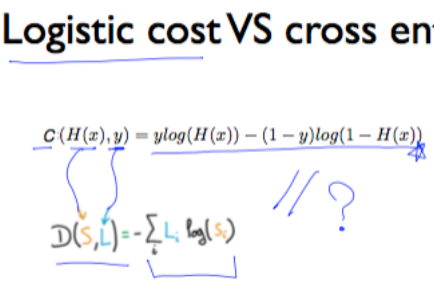
$$\hat{Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty, \quad cost=\infty$$

교수님께서 보여주신 식 전개를 다시 썼다. L은 label의 약자로 Y의 실제 값이다. Y hat은 Y를 예측한 값으로, 맞게 예측했을 때와 틀리게 예측했을 때를 보여주고 있다. 여기서 왜 L이 1개의 값이 아니라 2개의 값을 갖는지 이해를 못했었다. 처음 들었을 때는 그냥 그러려니 했는데, 생각할수록 이해가 되지 않았던 부분이다. 다음 글에 나오는 텐서플로우 예제를 보면, 실제 L의 값은 3개짜리 배열로 표현된다. A, B, C 중에서 고르니까.

결론부터 말하면 binary classification을 두 번 진행한다는 뜻이다. 이전 그림처럼 3개 중에서 하나를 고른다면, L의 크기는 3이 되어야 한다. 각각의 요소는 binary classification의 결과다. 공식에서 보면 L의 i번째라고 표현하는 것은, 여기서는 두 개 있으니까 두 번 반복하게 된다. Y를 예측한 값이 0 또는 1일 수밖에 없는 이유는 one-hot encoding을 거쳤기 때문이다.

위의 공식이 A와 B 중에서 하나를 선택해야 한다면, 현재 L에 들어있는 값은 B에 해당하는 요소가 1이므로 B를 가리킨다. 첫 번째 Y hat은 B를 가리키니까 맞게 예측했고, 두 번째 Y hat은 A를 가리키므로 잘못 예측했다. element-wise 곱셈을 적용한 최종 결과를 보면, 맞게 예측했을 때는 0이 나오고, 잘못 예측했을 때는 무한대가 나왔다. 잘 동작하는 cost 함수임을 알 수 있다.

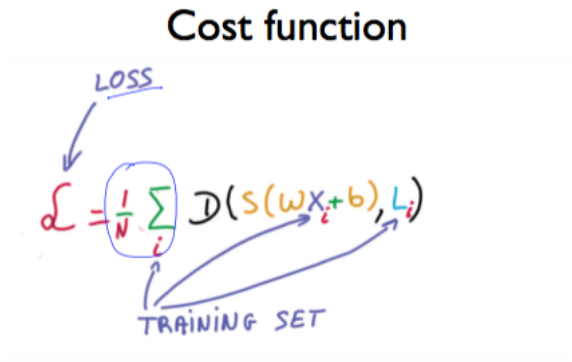
동영상에서는 L의 값이 (1,0)일 때, 즉 A일 때에 대해서도 보여주지만, 똑같은 설명의 반복이므로 생략한다. 그러나, 이 글을 보는 사람은 직접 이 부분에 대해 손으로 직접 써봐야 한다. 나는 그렇게 이해했다.



여기서 교수님께서 과제를 내주셨다. logistic regression에서 사용했던 cost 함수와 multinomial classification의 cross-entropy cost 함수가 똑같다고 말씀하셨다. 같은 이유에 대해 살짝 설명하셨는데, 나에게는 정말 살짝이었다.

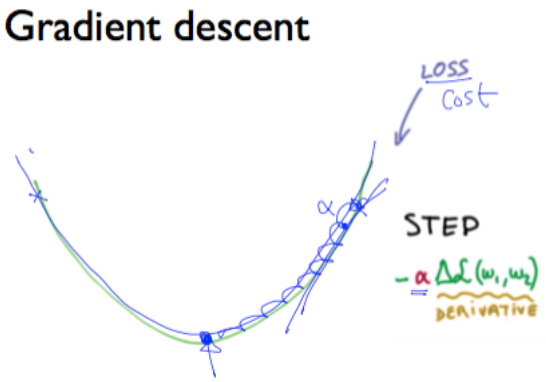
왜 두 개의 cost 함수가 같은 것일까? 동영상을 다섯 번쯤 보고 앤드류 교수님 수업도 듣고 하면서 답을 찾은 것 같다.

cost 함수의 목적은 틀렸을 때 벌을 주어서 비용을 크게 만들어야 하는데, 양쪽 모두 무한대라는 벌칙을 적용한다. 다만 logistic regression에서는 2개의 log식을 연결해서 사용하지만 cross-entropy에서는 행렬로 한 번에 계산하는 방식을 취할 뿐이다. 즉, logistoic regression을 cross-entropy로 처리할 수 있다. 바로 앞에 나온 2개 중에 하나를 선택하는 그림이 logistoic regression의 cross-entropy 버전이다. cross entropy 하나로 logistic regression까지 처리할 수 있으므로, 포함 관계로 생각할 수도 있을 것 같다. 어찌 됐든 cross entropy 하나만 하면 된다는 뜻이다.



https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

cost 함수를 다시 한번 강조하셨다. WX + b는 지겹도록 본 공식으로 y를 예측한다. 왼쪽에 있는 L은 loss의 약자로 다른 말로는 cost 또는 error라고 한다. 비용이라고 하는 것은 결국 잘못 예측했을 때의 값, 즉 에러(error)라고 볼 수 있다. training set을 갖고 작업해야 한다고도 얘기한다. training set으로 학습하고 validation set으로 검증하고, test set으로 최종 확인까지 한다고 나중에 나온다.



https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811827

cross-entropy cost 함수를 만들었다면, gradient descent 알고리듬에 적용해서 최소 비용을 찾아야 한다. 역시 이때 중요한 것은 그림에 표현된 w1, w2의 값이다. 알파(a)는 learning rate로 어느 정도로 이동할 것인지를 알려주고, 알파 오른쪽의 삼각형은 미분을 한다는 뜻이다.

예전 글에서 gradient descent 알고리듬을 구현하기 위해서는 cost 함수와 gradient를 계산하는 함수가 모두 필요하다고 얘기했었다. 여기서 보여주려는 것이 learning rate에 미분 결과를 곱한 값을 빼야 한다는 점이다. 앞에 음수 기호(-)가 있다.

김성훈 교수님께서는 이걸 미분하는 것은 너무 복잡하기 때문에 생략한다고 말씀하셨다. 앤드류 교수님도 이 부분에서 octave에서 제공하는 함수를 써서 처리하고 실제 구현은 보여주지 않으셨다.

15

머신러닝_김성훈교수님 (/category/머신러닝_김성훈교수님) 카테고리의 다른 글

17. 학습 rate, Overfitting, 그리고 일반화 (Regularization) (lec 07-1) (/23?category=573319) (0) 2016.07.28
16. TensorFlow로 Softmax Classification의 구현하기 (lab 06) (/21?category=573319) (0) 2016.07.27
15. Softmax classifier 의 cost함수 (lec 06-2) (/20?category=573319) (0) 2016.07.27
14. Softmax Regression- 기본 개념 소개 (lec 06-1) (/19?category=573319) (0) 2016.07.27
13. 파이썬으로 Logistic Regression 직접 구현 (/18?category=573319) (0) 2016.07.27
12. TensorFlow로 Logistic Classification의 구현하기 (lab 05) (/17?category=573319) (0) 2016.07.27