**MNIST 초급**

**MNIST란?**

* 간단한 컴퓨터 비전 데이터셋, 머신러닝의 가장 처음 단계
* 손으로 쓰여진 이미지(숫자로 구성), 각 이미지가 어떤 정보(숫자)인지에 관한 라벨을 포함

**MNIST 실습으로 하는일**

* 모델(머신러닝 모델)이 이미지를 보고 어떤 숫자인지 예측하는 모델을 훈련시키는 작업
* 텐서플로우를 살짝 맛보는 단계, 소프트맥스 회귀라고 하는 아주 간단한 모델로 시작할 것

**MNIST데이터셋**

* MNIST데이터셋 : 파이썬에서 코드로써 다운받고 설치함.

from tensorflow.examples.tutorials.mnist import input\_data

mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)

* + 데이터

- 55,000개 학습데이터(training set, mnist.train)

- 10,000개의 테스트 데이터(test set, mnist.test)

- 5,000개의 검증데이터(validation set, mnist.validation)

=> 우리가 학습한 것이 정말로 일반화되었다고 확신할 수 있는 근거가 됨.

\* 일반화 : 모델이 새 데이터에 대해 정확하게 예측할 수 있으면 이를 training set 에서

test set으로 ‘일반화’ 되었다 라고 한다.

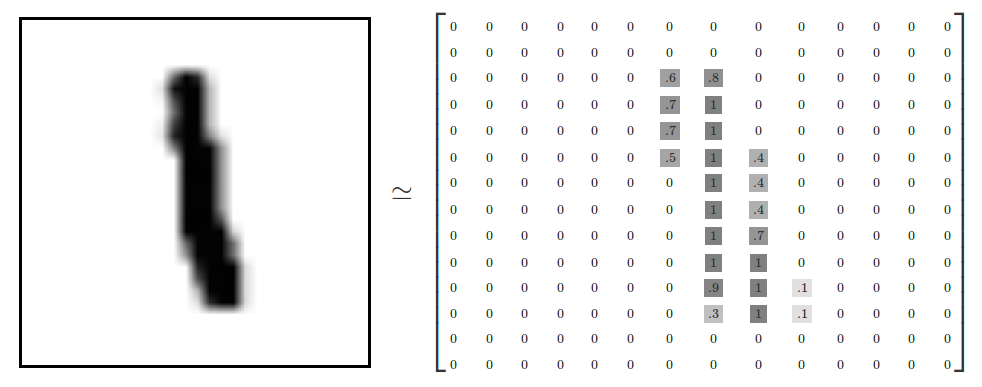
**---------------------------------------------training set---------------------------------------------------**

\* MNIST 데이터셋 = 두부분

- 손으로 쓴 숫자 이미지(image, 데이터/ xs, mnist.train.images)

- 라벨(정답 / ys, mnist.train.labels)

- 각 이미지는 28 x 28 pixel이고 이것을 큰 배열로 해석할 수 있음.

(오른쪽은 28x28의 2차원 배열)

* + - 이 배열을 28x28=784개의 벡터로 만든다. (데이터를 펼침)
    - 데이터를 펼쳐서 mnist.train.images는 [55000, 784]의 형태의 텐서(n차원 배열)가 됨.

|  |  |
| --- | --- |
| mnist.train.xs : 이미지 들어있음. | Mnist.train.ys : 이미지에 대한 라벨 들어있음. |
| https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/images/mnist-train-xs.png | https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/images/mnist-train-ys.png   * 각 라벨 : 0~9 사이 숫자, 각 이미지가 어떤 숫자인지 말해줌. * n번째 숫자 : n번째 차원이 1인 벡터로 표현됨. * Mnist.train.labels = [55000, 10] 모양 가진 실수배열 됨.(소프트맥스 회귀 결과가 정수형 아닌 실수형으로 산출되기 때문) |

--------------------------------------------------------------------------------------------------------------

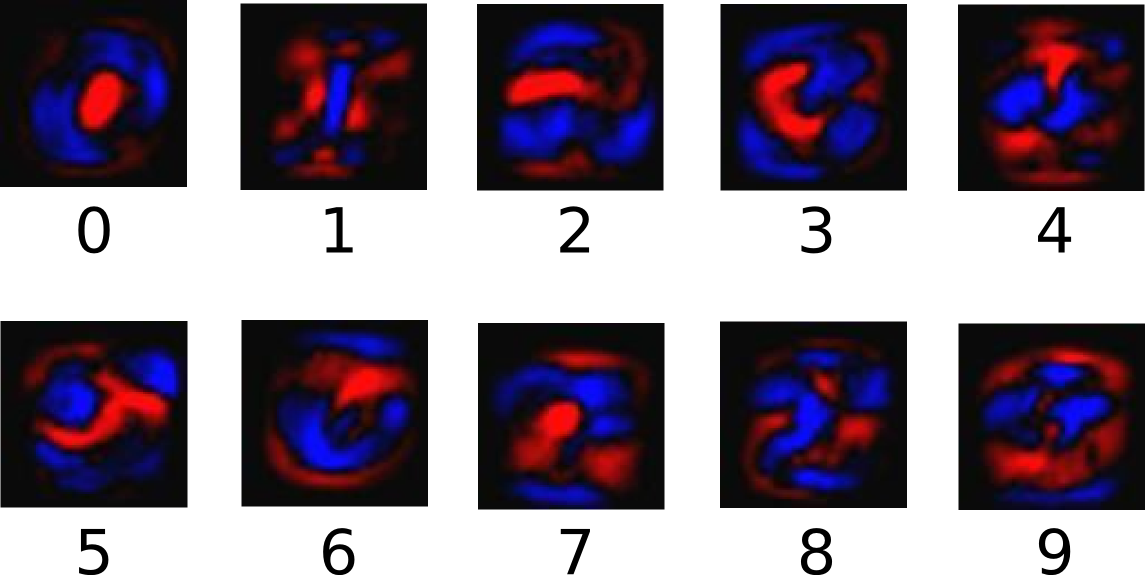
**소프트맥스 회귀(softmax regression)**

소프트맥스 회귀는 판단해야되는 값이 서로 다른 여러 항목 중 하나일 확률을 계산하고자 할 때 사용하기 적절하다. 소프트맥스는 각 값이 0과 1사이의 값으로 이루어 지고, 각 값을 모두 합하면 1이되는 목록을 제공하기 때문이다.

소프트맥스 회기의 단계는 2단계이다.

1. 입력한 데이터가 각 클래스에 속한다는 증거(evidence)를 수치적으로 계산
2. 계산한 값을 확률로 반환

한 이미지가 특정 클래스에 속하는지 계산하기 위해 각 픽셀의 어두운 정도(intensity)를 가중치합(서로 다른 계수를 곱해 합하는 계산, weighted sum)을 한다. 여기서 가중치는 해당 픽셀이 진하다는 것이 특정 클래스에 속한다는 것에 반하는 내용이라면 음(-)값, 특정클래스에 속하면 양(+)값을 가지게 된다.



(빨간부분은 음의 가중치, 파란부분은 양의 가중치)

또한 여기에 바이어스(bias, 결과값의 일부는 입력된 데이터와는 독립적일 수 있다는 것을 고려하기 위함)라는 추가적 항을 더하게 된다. 이를 수식으로 표현하면(입력값 x가 주어졌을 때의 클래스 i에 대한 증거값)

evidencei=∑jWi, jxj+bi,( Wi는 가중치, bi,는 클래스 i에 대한 bias, j는 입력 데이터로 사용한 이미지 x의 픽셀값을 합하기 위한 인덱스)이며 각 클래스에 대해 계산한 증거값들을 “소프트맥스”함수를 활용해 예측 확률 y로 변환한다.

y=softmax(evidence)=softmax(∑jWi, jxj+bi)

소프트 맥스는 우리가 계산한 선형 함수를 우리가 원하는 형태(이 경우 10가지 경우에 대한 확률 분포)로 변환하는데 사용하는 “활성화” 또는 “링크”함수의 역할을 한다. 이러한 계산(계산한 증거 값들을 입력된 데이터 값이 각 클래스에 속할 확률로 변환하는 것)을 정의하면

softmax(x)=normalize(exp(x))

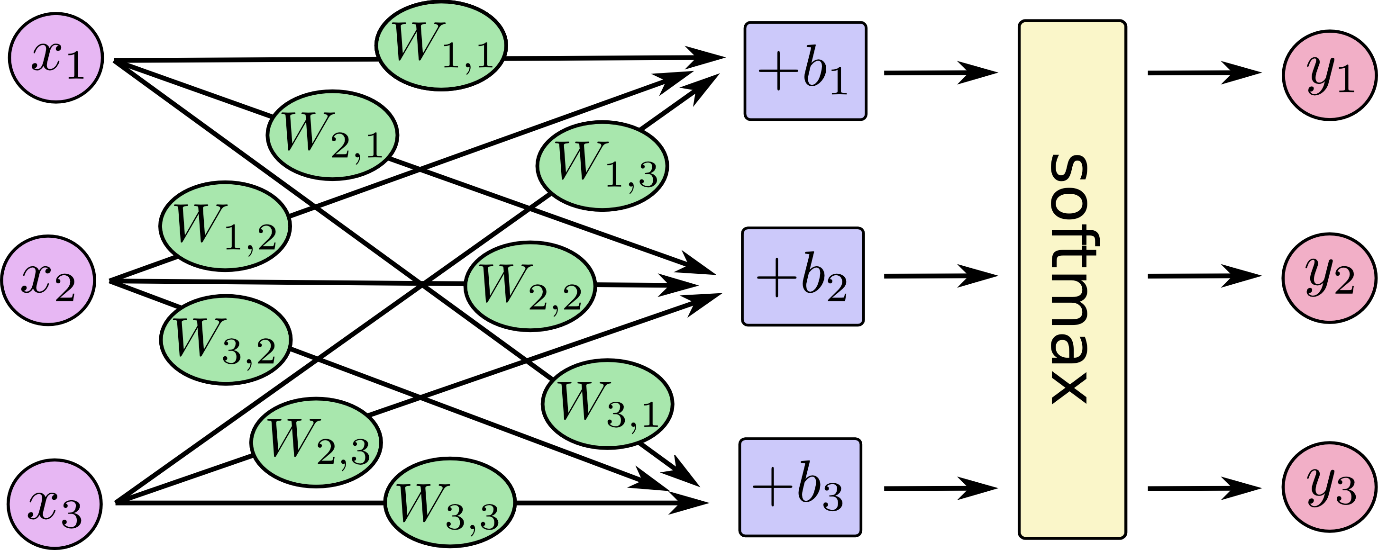
이 식을 전개하면

softmax(x)i=exp(xi) / ∑jexp(xj) 값을 얻는다.

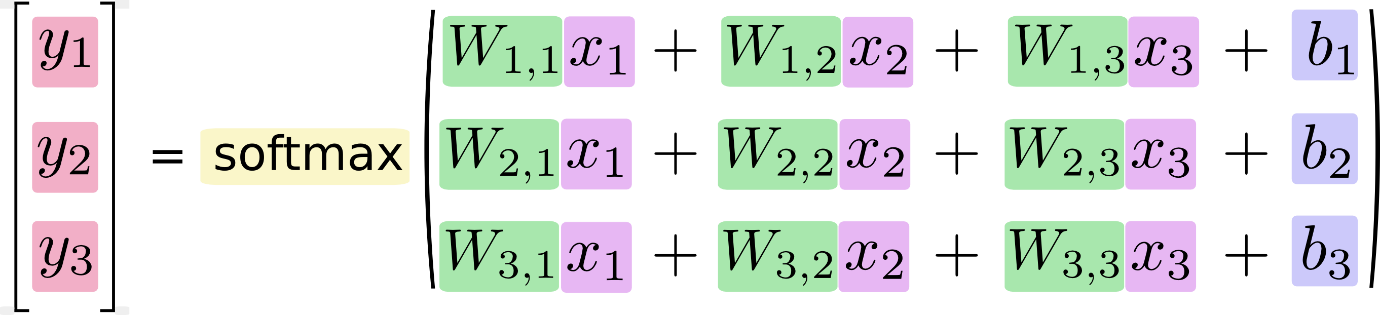
대다수의 경우에서, 일단 소프트맥스를 입력값을 지수화(증거값을 하나 더 추가하면 어떤 가설에대해서 주어진 가중치를 곱으로 증가시키는 것)한 뒤 정규화 하는 과정이라고 생각하면 편하다.

반대로 증거값의 개수가 하나 줄어든다는 것은 가설의 가중치가 기존 가중치의 분수비로 줄어들게 된다는 것을 뜻한다. 어떤 가설도 0 또는 음의 가중치를 가질 순 없다. 그런 뒤 소프트맥스는 가중치를 정규화한 후 모두 합하면 1이 되는 확률 분포를 만든다.

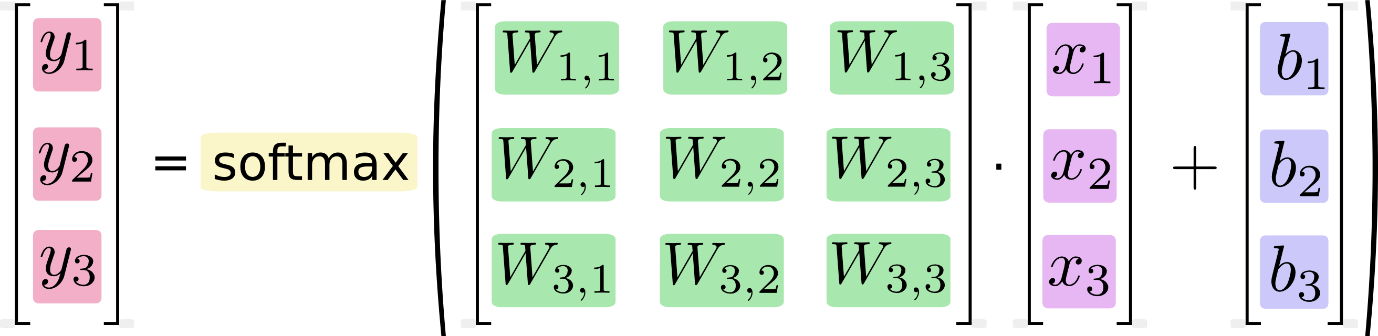
소프트 맥스 회귀는 각각의 출력값에 대해 가중치 합을 계산하고 바이어스를 더한 뒤 소프트맥스를 적용하는 다음과 같은 형태를 가지게 된다.



이를 수식으로 표현하면



이를 다시 행렬곱과 백터합으로 변경하여 “백터화(계산의 효율화에 도움이 됨)”할 수 있다.



즉, y=softmax(Wx+b)

라고 할 수 있다.

회귀구현

파이썬에서는 오버헤드를 피하기 위해 하나의 무거운 작업을 독립적으로 실행하는 대신, 텐서플로우는 서로 상호작용을 하는 연산간의 그래프를 유저가 기술하도록 하고 그 연산 모두가 파이썬 밖에서 동작하는 방식으로 오버헤드를 피함.

구현

1. 텐서플로우를 사용하기위해 임포트  
   import tensorflow as tf
2. 상호작용하는 연산들을 심볼릭 변수를 활용해 기술함.  
   x = tf.placeholder(tf.float32, [None, 784])

X에 특정값이 주어지는 것은 아님, 이는 ‘placeholder’로, 우리가 텐서플로우에서 연산을 실행할 때 값을 입력할 자리임(위에서 말한 784차원의 벡터로 변형된 MNIST이미지 데이터가 들어감)->[None, 784]의 형태를 갖고 부동소수점으로 이루어진 2차원 텐서로 표현함.(None은 해당 차원의 길이가 어떤 길이든 될 수 있음을 의미함)

1. 가중치와 바이어스는 텐서플로우에서 Variable이라는 방법으로 연산함(Variable은 서로 상호작용하는 연산으로 이루어진 텐서플로우 그래프 안에 존재하는 수정가능한 텐서이다.)

W = tf.Variable(tf.zeros([784, 10]))

b = tf.Variable(tf.zeros([10]))

W와 b는 각각 0으로 이루어진 텐서로 초기화 하며, 이를 학습해 나갈 것이다. 여기서 W의 형태가 [784, 10]인 이유는 W에 784차원의 이미지 벡터를 곱해서 각 클래스에 대한 증거값을 나타내는 10차원 벡터를 얻고자 하기 때문이다. B는 그 10차원 벡터에 더하기 위해 [10]의 형태를 가진다.

1. 모델을 구현한다.

y = tf.nn.softmax(tf.matmul(x, W) + b)

여기서 본 수식에서 곱했던 순서인 Wx 와 반대인데, 그 이유는 x가 여러 입력값을 ㄱ자는 2차원 텐서인 경우에도 대응하기위한 작은 트릭이다. 그 다음 b를 더하고, 마지막으로 tf.nn.softmax를 적용한다.

학습

모델이 안좋다는 것은, 비용(cost) 또는 손실(loss)라 부르며 이것들은 우리가 원하는 결과에서 얼마나 떨어져있는지를 보여주는 값으로, 이 격차가 크면클수록 좋지않은 모델이고, 적으면 적을수록 좋은 모델이며 우리가 원하는 모델은 격차가 좋은 모델이다.

모델의 손실을 정의하기 위해 사용되는 함수로써 “크로스 엔트로피(원랜 정보 이론 분야에서 정보를 압축하는 방법으로 고안됨)”가 있다. 크로스 엔트로피의 정의는

Hy′(y)=−∑iy′ilog(yi)

Y는 우리가 예측한 확률분포이며, y’은 실제 분포(우리가 입력하는 원-핫 벡터)이다. 크로스 엔트로피는 우리의 예측이 실제 값을 설명하기에 얼마나 비효율적인지를 측정하는 함수이다.

1. 크로스엔트로피를 구현하기 위해서는 올바른 답을 넣기 위한 새로운 placeholder를 추가하는 것부터 시작함.

y\_ = tf.placeholder(tf.float32, [None, 10])

1. 크로스 엔트로피를 구현할 수 있다.

cross\_entropy = tf.reduce\_mean(-tf.reduce\_sum(y\_ \* tf.log(y), reduction\_indices=[1]))

1. 텐서플로우에게 학습 비율 0.5로 경사 하강법(gradient decent algorithm, 텐서플로우가 각각의 변수를 비용을 줄이는 방향으로 조금씩 이동시키는 매우 단순한 방법)을 적용하여 크로스 엔트로피를 최소화하도록 지시한다.

train\_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross\_entropy)

여기서 텐서플로우는 역전파와 경사하강이라는 새로운 작업을 당신의 그래프에 추가하는 일을 한다.

1. 작성한 변수들을 초기화 하는 작업을 추가한다.

init = tf.global\_variables\_initializer()

1. 이제 session에서 모델을 실행시키고, 변수들을 초기화 하는 작업을 실행시킨다.

sess = tf.Session()

sess.run(init)

1. 학습을 시킨다.

for i in range(1000):

batch\_xs, batch\_ys = mnist.train.next\_batch(100)

sess.run(train\_step, feed\_dict={x: batch\_xs, y\_: batch\_ys})

반복되는 루프의 각 단계마다, 우리는 학습 데이터셋에서 무작위로 선택된 100개의 데이터로 구성된 “배치(batch)”를 가져온다. 그 다음엔 placeholder의 자리에 데이터를 넣을 수 있도록 train\_step을 실행하여 배치 데이터를 넘긴다.

이러한 무작위 데이터의 작은 배치를 사용하는 방법을 확률적 학습(stochastic training)이라고 부른다.(이번에는 확률적 경사 하강법)

모델평가

1. 모델의 성능을 확인해본다.(tf.equal )

correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1))

tf.argmax는 텐서 안에서 특정 축을 따라 가장 큰 값의 인덱스를 찾기에 매우 유용한 함수이다.(tf.argmax(y,1)는 우리의 모델이 생각하기에 각 데이터에 가장 적합하다고 판단한 라벨이며, tf.argmax(y\_, 1)는 실제 라벨이다.

1. 우리의 테스트 데이터를 대상으로 정확도를 계산한다.

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

print(sess.run(accuracy, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels}))