

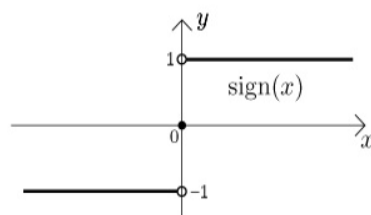
Neural network에서 높은 신뢰도를 가진 모델이 입력을 잘못 분류 하게끔 noise를 더한 이미지를 adversarial example 이라고 한다. 이 현상을 설명하기 위한 많은 시도가 있었는데, 초기에는 비선형성과 overfitting에 초점을 맞춘 연구가 많이 진행되었다.

반면에 이 논문에서는 adversarial perturbation에 대한 neural networks의 취약성의 주요 원인이 선형적인 특성에서 나오며 고차원 공간에서 선형 행동은 적대적 사례를 일으키기에 충분하고 주장한다.

이 논문에서는 이러한 관점을 증명하기 위해 단순한 선형 모델이 적대적 예를 가질 수 있음을 보여준다. Input feature의 정밀도는 제한적이기 때문에 충분히 작은 ϵ 와 noise n 에 대해 $\|n\|_\infty < \epsilon$ 이면 분류기는 x 와 $x+n$ 을 동일한 클래스에 할당한다. Adversarial example인 $\tilde{x} = x+n$ 과 가중치 w 를 내적하면 $w\tilde{x} = wx + wn$ 식이 나온다. 이 식에서 보듯이 noise n 에 의해 활성화가 wn 만큼 성장하게 한다. 여기서 $n = \text{sign}(w)$ 으로 n 에 할당함으로써 n 으로 인한 활성화의 증가를 최대화할 수 있다.

* $\text{sign}(x)$: 부호함수

$$\text{sign}(x) = \begin{cases} 1, & x > 0; \\ 0, & x = 0; \\ -1, & x < 0. \end{cases}$$



w 에 n 차원이 있고 가중치 벡터의 원소의 평균 크기가 m 이면 활성화는 emn 만큼 증가한다.

noise n 이 없을 때

$$w = \begin{bmatrix} -10 \\ -2 \\ 3 \end{bmatrix} \quad \text{input } x = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad \text{일때}$$

$$wx = -20 - 2 + 3 = -19 \quad \text{이다.}$$

noise n 이 있을 때 $n = \text{sign}(w)$ 이고 $\epsilon = 1$ 이라 가정하면

$$\begin{bmatrix} -10 \\ -2 \\ 3 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = wn = 10 + 2 + 3 = 15 \quad \text{이다.}$$

즉 15만큼 증가하게 된다.
고차원일수록 더 많이 증가한다.

이처럼 $\|n\|_\infty$ 는 문제의 차원성에 따라 증가하지 않지만 noise n 에 의한 활성화 변화는 n 에 따라 선형적으로 커질 수 있기 때문에, 고차원 문제의 경우 입력에 대해 아주 작은 변화를 많이 일으켜 출력에 큰 변화를 생기게 할 수 있다. 이를 통해 입력의 치수가 충분한 경우 단순한 선형 모델이 적대적 예를 가질 수 있음을 보여준다.

그 뒤, 이 논문에서는 adversarial examples를 생성하는 빠른 방법을 알려준다.

θ 를 모델의 매개 변수, x 모델에 대한 입력, y 는 x 에 대한 출력, $J(\alpha, x, y)$ 를 신경망 훈련에 사용하는 cost 라고 한다. n 값을 중심으로 비용 함수를 선형화할 수 있으며, $n = \epsilon \text{sign}(\nabla_x J(\alpha, x, y))$ 의 max-norm constrained perturbation 얻을 수 있다. 이를 adversarial example을 생성하는 fast gradient sign method라고 부르며 이때 gradient는 backpropagation를 사용하여 효율적으로 계산할 수 있다.

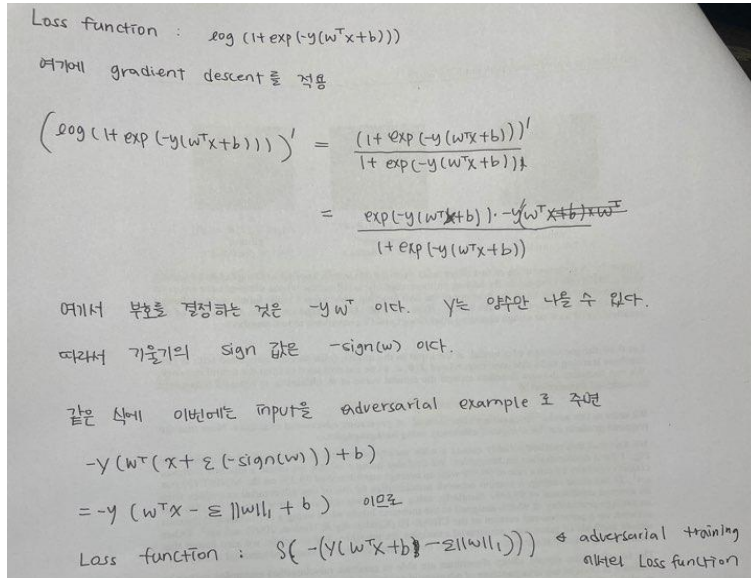
그 뒤, 이 방법이 모델이 입력을 잘못 분류하도록 유발한다는 것을 실험을 통해 발견한다.

$\epsilon = 0.25$ 를 사용하여 adversarial examples를 만들고, MNIST 테스트 세트에서 평균 신뢰도가 79.3%인 softmax 분류기에 입력 시켰을 때 adversarial examples에 대해 99.9%의 오류율을 갖는다. 그리고 같은 설정에서 평균 97.6%의 신뢰도를 갖는 maxout network는 adversarial examples가 입력으로 들어왔을 때 89.4%로 잘못 분류한다. 마찬가지로 $\epsilon = 0.1$ 을 사용하여 CIFAR-10 테스트 세트에서 convolutional maxout network를 사용할 때 평균 96.6%의 오류율을 가진다.

다음으로, 이 논문에서는 이렇게 빠르게 만들어진 adversarial examples를 가지고 adversarial training을 한다. Adversarial example을 만든 다음에 이것을 training 시점에 일반적인 input과 같이 사용을 해서 adversarial example을 training data로써 사용해서 학습시킨다. Logistic regression 예제에 대해 adversarial training을 하고, regularization 기법인 weight decay와 비교한다.

Adversarial training과정은 다음과 같다.

여기서 training을 시킬 때 $-y(wx + b)$ 인 output 값에 대해서 softplus function을 붙인 것을 loss function이라고 설정하고 해당 loss function에 대해 gradient descent를 적용하는 방식을 취한다



Loss function : $\log(1 + \exp(-y(w^T x + b)))$

여기에 gradient descent를 적용

$$\left(\log(1 + \exp(-y(w^T x + b))) \right)' = \frac{(1 + \exp(-y(w^T x + b)))'}{1 + \exp(-y(w^T x + b))}$$
$$= \frac{\exp(-y(w^T x + b)) \cdot (-y(w^T x + b) + w^T)}{1 + \exp(-y(w^T x + b))}$$

여기서 부호를 결정하는 것은 $-y w^T$ 이다. y 는 양수만 나올 수 있다.

따라서 기울기의 sign 값은 $-\text{sign}(w)$ 이다.

같은 식에 이번에는 input을 adversarial example로 주면

$$-y(w^T(x + \epsilon \cdot \text{sign}(w)) + b)$$
$$= -y(w^T x - \epsilon \|w\|_1 + b) \quad \text{이므로}$$

Loss function : $S(-y(w^T x + b) - \epsilon \|w\|_1)$ & adversarial training
이것이 loss function

이렇게 나온 adversarial training에서의 loss function과 L1 regularization의 비슷하지만 다른점은 L1 regularization은 penalty가 training 중에 모델의 활성화에서 빠진다는 것이다. 그래서 penalty는 모델이 학습할수록 사라지기 시작할 수 있다. 따라서 regularization 과정은 adversarial training을 할 때 더 좋은 결과를 얻게 된다. 이는 multiclass softmax regression을 할 때 그 차이가 더 드러난다. 실제로 MNIST에서 maxout networks를 훈련할 때 0.25로 adversarial training을 하여 좋은 결과를 얻어 냈다. L1 weight decay를 적용할 때는 0.0025로 설정해도 너무 컸다.

그 다음으로는, 이 adversarial training을 deep network에 적용시킨다. 이를 통해 fast gradient sign method를 기반으로 한 adversarial training이 효과적인 정규화라는 것을 발견한다.

$J^*(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha)J(\theta, x + \text{sign}(\nabla_x J(\theta, x, y)))$ 식은 일반적인 input에 adversarial examples를 섞어서 실험한다는 것을 나타낸다. 이 논문에서는 모든 실험에서, $\alpha = 0.5$ 를 사용한다. dropout와 함께 regularization된 maxout network를 훈련하기 위해 이 접근 방식을 사용하여, adversarial training이 없을 때 오류율이 0.94%인데 adversarial training으로 0.84%까지 줄일 수 있다.

또한 training set의 adversarial examples에 대해 오류율이 0%에 도달하지 못하고 있다는 것을 관찰했다. 그래서 이 논문에서는 두가지를 수정한다. 먼저, 이 문제에 대해 원래 maxout network에서 사용하는 240대보다 계층당 1600대를 사용하여 모델을 더 크게 만들었다. 그리고 adversarial validation set error에 대한 조기 중단을 사용했다. 그 뒤, training할 epoch 수를 선택한 다음 60,000개의 모든 예제를 재교육했다. 서로 다른 seed를 사용한 5개의 서로 다른 training에서 4개

의 training은 0.77%의 오류율을 보였으며 나머지 하나의 training은 0.83%의 오류율을 보였다. 0.782%의 평균은 dropout일 때의 오류율인 0.79% 와 통계적으로 구분할 수는 없었다. 모델은 또한 adversarial examples에 저항적으로 되었다. Adversarial training이 없는 이 동일한 모델은 fast gradient sign method를 기반으로 한 적대적 사례에 대해 89.4%의 오류율을 보였는데 adversarial training으로 오류율은 17.9%로 떨어졌다.

그리고 adversarial examples는 두 모델 간에 transfer이 가능하지만 adversarial training 이 된 모델은 더 큰 견고성을 보여준다. 원래 모델을 통해 생성된 adversarial examples는 adversarial training이 된 모델에서 19.6%의 오류율을 보이는 반면, 새로운 모델을 통해 생성된 adversarial examples는 원래 모델에서 40.9%의 오류율을 보였다.

하지만 Adversarial training이 된 모델이 adversarial examples를 잘못 분류하는 경우 예측은 여전히 신뢰도가 매우 높았다. 잘못 분류된 examples에 대한 평균 신뢰도는 81.4%였으며 학습된 모델의 가중치가 변경되었다.

그리고 마지막으로 위 논문에서는 dropout, pretraining, 그리고 model averaging 같은 일반적인 regularization 전략은 adversarial examples에 대한 모델의 취약성을 크게 감소시키지 않지만, RBF 네트워크와 같은 비선형 모델에 대해서는 크게 감소시킨다는 것을 보여준다.