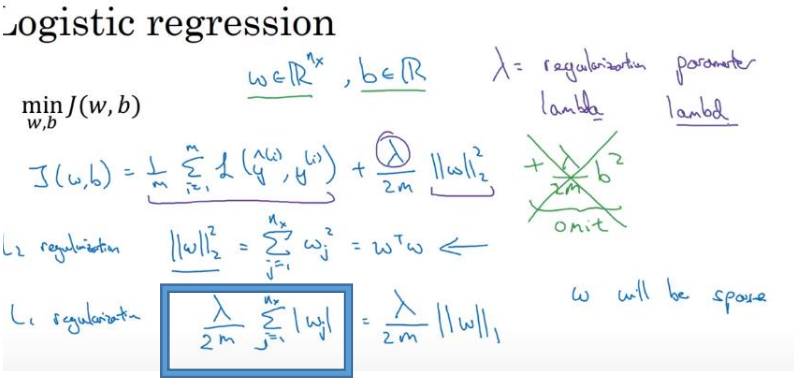
<Regularization>

- 오버피팅, 높은 분산 문제 해결 방법 => 정규화하기 or 더 많은 train data 얻기
- 정규화 : 과대적합을 막고 분산을 줄이는데 도움을 준다.

1. 로지스틱 회귀



- 로지스틱 회귀는 비용함수 J (개별 예측 손실에 관한 샘플)를 최소화한다.
- 정규화 추가하기 위해 람다를 추가한다.
- **L2norm** : w제곱의 norm (j 의 1부터 nx까지 wj^2 의 값을 더한 것 = wTw) 추가
- 가장 일반적인 정규화이다.

- b에 대한 norm은 생략 가능하다

⇒ w는 high dimension vector이고 모든 매개변수는 w에 있고 b는 하나의 숫자이기 때문이다.

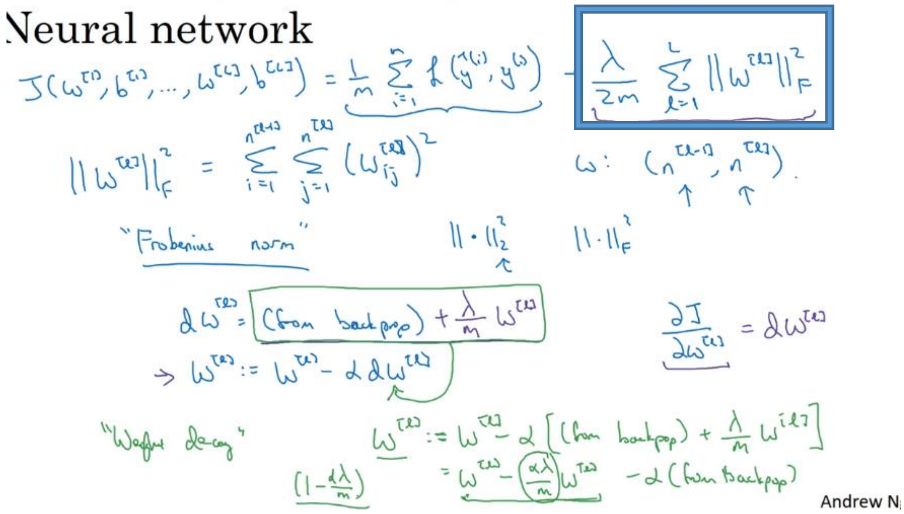
- **L1norm**
- 파란색 네모 앞에 람다/2m 은 스케일링 한 것을 의미한다.
- L1 쓰면 w는 희소해진다. (w벡터 안에 0 이 많아진다)

  ⇒ 모델을 압축, 메모리 적게 사용한다.

<람다>

- 람다는 정규화 매개변수 교차검증 세트에서 주로 사용한다.
- 훈련세트에 잘 맞으면서 두 매개변수의 norm 잘 설정해 과대 적합 막을 수 있는 최적의 값을 찾는 것이다.
- 설정이 필요한 하이퍼파라미터이다.

2. 신경망



- w의 dimension (n^[l-1],n^[l]) : 해당층 l-1과 l의 은닉 유닛 개수를 나타낸다.
- **frobenious norm** : 행렬 원소제곱 합
- 역전파에 정규화 항 더하기 여전히 비용함수의 미분의 의미한다.
- L2 정규화는 가중치 감소라고도 불린다.

< 왜 정규화는 과대적합을 줄일 수 있을까?>

- 비용함수 J 는 sum of losses 함수이다.
- 정규화에서 람다를 크게 하면 가중치 행렬 w를 0에 가깝게 만드는 것이다.
- 많은 은닉 유닛을 0 에 가까운 값으로 설정하여 은닉 유닛의 영향력 줄여서 더 간단하고 작은 신경망으로 만든다. (로지스틱 회귀에 가까워진다)
- 은닉 유닛을 0으로 만드는 것 아니다! 영향력이 작아지는 것이다

<tanh 활성화 함수를 사용할 때>

- z가 작으면 선형 영역만 사용하게 된다.
- z가 커지거나 작아지면 활성화 함수는 선형부분을 벗어난다.
- 가중치가 작으면 z=wa+b이므로 z도 작아진다.
- 모든 층이 선형회귀처럼된다

⇒ 정규화 매개변수가 크면 w는 작아지고, b를 무시하면 z는 상대적으로 작고, 작은 범위의 값을 가지기 때문에 활성화 함수가 선형이 된다. 전체 신경망도 선형을 띄게 되어 과적합 가능성이 줄어든다!

- J에 가중치가 너무 커지는 것을 막기위해 추가적인 항 추가
- 경사하강법 반복의 수에 대한 함수로 비용함수를 설정

드롭아웃 정규화

- 과적합한 신경망에서 신경망의 각각의 층에 대해 노드를 삭제하는 확률을 설정한다.

ex) 동전을 던진 후(0.5의 확률) 삭제하기.

- 노드를 삭제할 때 연결된 링크도 삭제한다.

1. 하나의 샘플을 역전파로 훈련
2. 다시 동전 던지고 다른 세트의 노드 삭제

=> 각 샘플마다 감소된 신경망 사용

<드롭아웃을 구현하는 방법>

역 드롭아웃

ex) 층이 3개인 layer

벡터 d3=np.random.rand(a3.shape[0],a3.shape[1])<keep.prob

 # 은닉 유닛이 유지될 확률 : keep.prob

a3=np.multiply(a3,d3) # 대응되는 a3의 원소를 0으로 만들게 된다.

a3/=keep.prob # 50개의 유닛(뉴런) (50,m) dim. 일 때 평균적으로 10개의 유닛 삭제되는것이다

z4=w4*z3+b4

a3의 원소의 20%가 0이 된다.

z4의 기대값 낮추기 않기 위해 0.8로 나누기

- **역드롭아웃 keep_prob 상관없이 a3의 기대값을 같게 유지한다**
- d 벡터를 사용해 서로 다른 훈련 샘플다마 다른 은닉 유닛들을 0으로 만든다
- 여러 번 반복하면 0으로 되는 은닉 유닛은 랜덤하게 바뀐다
- d3 는 어떤 노드를 0으로 할지 결정한다

테스트 시간에는 테스트 샘플 a0=X

- 테스트에서는 드롭아웃을 사용하지 않는다.
- 예측을 하는 것이기 때문에 결과가 무작위로 나오면 안되기 때문이다.
- 역드롭아웃의 효과는 테스트에서 드롭아웃을 하지 않아도 활성화 기댓값이 변하지 않는다.

드롭아웃의 이해

- 단일 유닛 관점에서 보면 입력을 받았을 때 의미있는 출력이 나와야한다.
- 유닛은 계속 무작위로 바뀌니까 어떤 특성에도 의존할 수 없다.
- 각각에 가중치를 분산시켜야한다 => 가중치의 norm 의 제곱값이 줄어든다.
- 가중치를 줄이고 L2정규화처럼 과적합을 막는다.

- keep_probs : 해당 유닛을 유지할 확률 => 층마다 다르게 할 수 있다.
- 과대적합 우려가 적은 층에는 큰 keep_probs 가져도 된다.
- 입력층에 대해서는 1 or 0.9 를 사용한다.

⇨ 다른 층보다 과대적합의 우려가 큰 층에 대해서는 keep_probs를 낮게 설정한다.

(더 많은 하이퍼 파라미터가 생긴다.)

⇨ 드롭아웃은 정규화 기법, 오버피팅을 막는다.

단점 : 비용함수가 잘 정의되지 않는다

비용함수가 하강하는지 확인하기 어렵다

다른 정규화 방법들

<데이터 증식 Data augmentation>

고양이 분류기 학습

- 수평 방향으로 뒤집은 이미지도 샘플에 추가해 훈련 샘플을 늘릴 수 있다.
- 이미지의 무작위적인 변형으로도 훈련 샘플 만들 수 있다.

<Early stopping>

- 훈련 오차나 J 단조 감소한다.
- 개발 세트 오차 개발세트에서의 로지스틱 손실 함수=> 아래로 내려가다가 증가한다.
- 그래프에서 나타나는 값의 iteraton 을 선택한다.
- 반복을 중간에 멈추면 w가 중간 크기의 값을 가진다. => 과대적합을 막는다.

- 단점 : optimal cost J 찾는 것과 과적합을 막는 것을 독립적으로 하지 못하게 된다.

⇨ L2를 쓰는 것