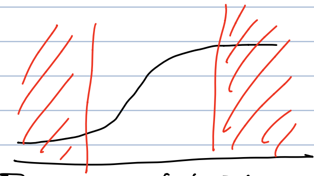
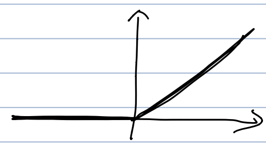
Lab-09-1 ReLU

sigmoid는 gradient를 구할 때



이 부분은 0과 아주 가깝거나 엄청 큰 값이 나와서
back propagation 구할 때 layer가 많으면 앞으로 갈수록
gradient가 거의 소멸한다. ⇒ vanishing gradient

ReLU: f(x)=max(0, x)
x=torch.nn.relu(x)



drop_last: 마지막 batch를 버릴지 말지

torch.nn.init.normal_(linear#.weight) 으로 초기화
마지막 linear는 relu하지 않고 CrossEntropyLoss() 이용
optim.Adam 사용

Lab-09-2 Weight initialization

0으로 초기화하는 것은 안좋은 방식임. back propagation 계산에 지장생김.

* Restricted Boltzmann Machine
  - layer 사이에 connection 없음.
  - 입력 x가 들어가면 y를 만드는 forward가 있고
    y가 들어가면 x를 복원하는 x'이 생김. ⇒ encoding

* Pre-training

  RBM: 두 개의 layer를 학습하고 그 다음 layer를 하나씩
  쌓아서 학습한다.

  Fine-tuning: RBM 마지막 단계에서 back propagation으로
  최종 결과 계산.

* Xavier / He initialization

- Xavier Normal initialization

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

- Xavier Uniform initialization

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, +\sqrt{\frac{6}{n_{in}+n_{out}}}\right)$$

torch.nn.init.xavier_uniform_(linear1.weight)

- He normal initialization

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in}}}$$

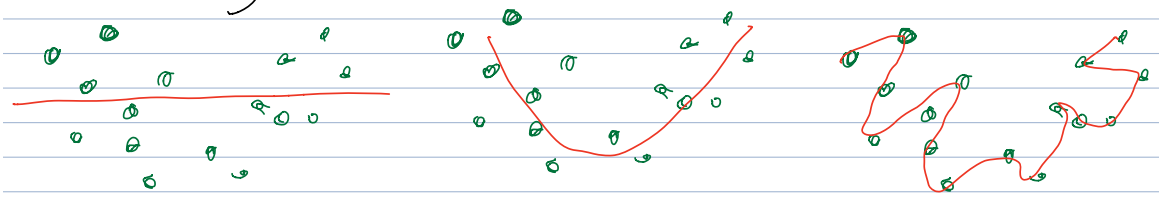
- He Uniform initialization

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}}}, +\sqrt{\frac{6}{n_{in}}}\right)$$

```
std = gain * math.sqrt(2.0/(fan_in+fan_out))
a = math.sqrt(3.0)*std
with torch.no_grad():
    return tensor.uniform(-a, a)
```

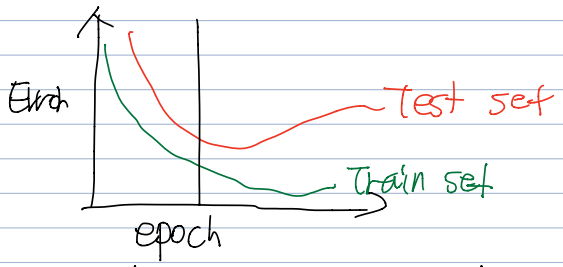
Lab-09-3 Dropout

* Overfitting



Underfitting

Overfitting



training dataset에 대해서는 정확도가 매우 높지만
test dataset에 대해서는 정확도가 낮음.

⇒ training data 더 많게, features 수 줄인다, regularization, dropout을 통해 해결

*Dropout

dropout을 이용한 학습은 layer를 선택하여 전파함.
Overfitting 방지, 성능 향상

dropout = torch.nn.Dropout(p = drop_prob)

* Train & eval mode

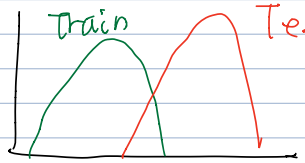
model.train() 버릴 노드와 안버릴 노드를 고려하여 학습한다.
model.eval() 모든 노드를 이용하여 test 한다.
↳ 꼭 명시하자!

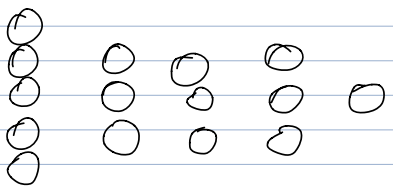
Lab-09-4 Batch Normalization

Gradient Exploding : back propagation 계산에서 너무 큰 값이 나와서 NAN이 뜨는 현상 ↔ Gradient Vanishing

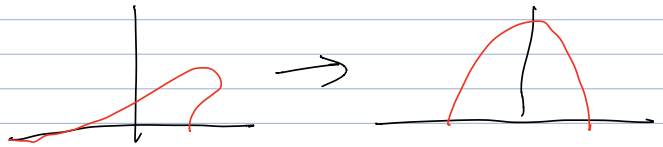
⇒ activation function 바꾸기, initialization 조심하기, 작은 learning rate

*Internal Covariate Shift : Gradient exploding/Vanishing 만들.





* Batch normalization



$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \qquad \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$$

```
bn_prediction = bn_model(X)
bn_loss = criterion(bn_prediction, Y)
```

batch normalization을 쓰면 더 적은 loss,
더 높은 정확도를 얻음.