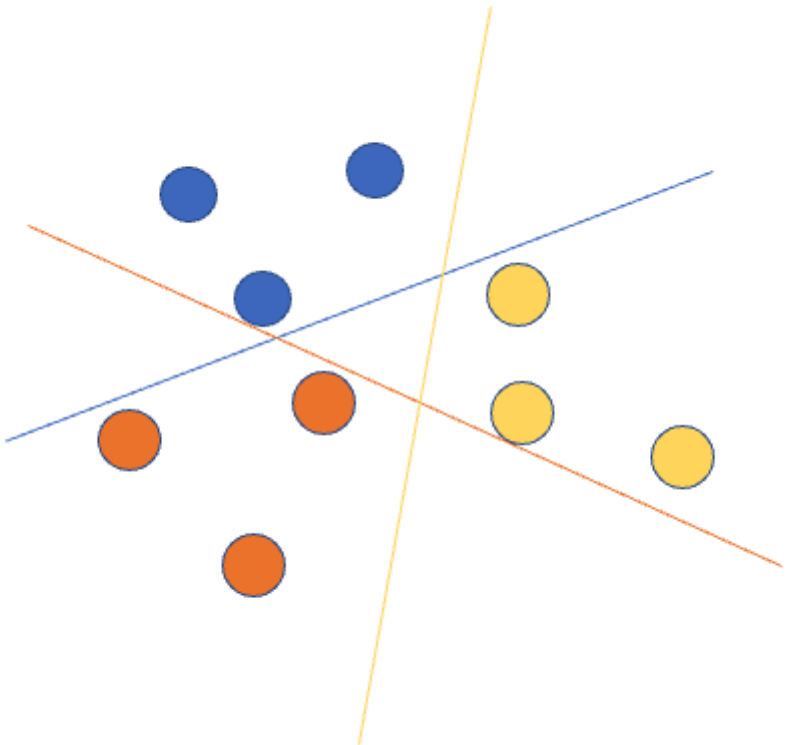
Multi-Class SVM

SVM 은 Support Vector Machine

- 기본적으로 SVM 은 binary class classifier 이다. 이 SVM 을 multi-class 를 다루기 위해 다양한 방법론이 논의되었다.

방법 1. One vs Rest

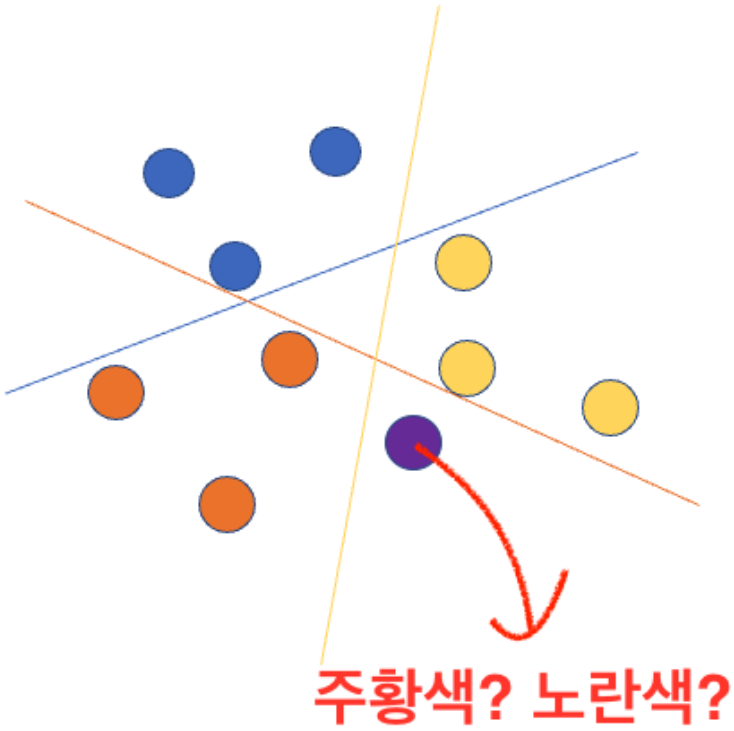
- K 개의 서로 다른 SVM 을 만드는 것
  - k 번째 모델 $y_k(x)$ 는 클래스 C_k 로 부터 데이터를 양의 예시로, 나머지를 $K-1$ 개의 클래스에서의 데이터를 음의 예시로 학습함.
  - 즉, 위와 같이 binary 화 시켜 이 작업을 K 번 반복한다는 뜻이다. 그렇다면 처음 말했던것과 같이 K개의 서로 다른 SVM 이 만들어지게 된다는 것.
    - k 번째 모델 $y_k(x)$ 는 +1 의 label 을 붙이고, 나머지 $K-1$ 개의 class 를 -1 의 label 로 설정하여 training set 을 만들게되고, SVM 의 결정 초평면도 K개가 생성되게 된다.



- test 를 진행할 때에는 K 개의 초평면에 모두 test 하게 되는데 K 번의 분류에서 1가지만 양수를 출력하게 되고 나머지는 모두 음수를 출력한다면 괜찮

지만, 여러개의 다른 class 에 속하는 결과값이 나올 수도 있다는 것이다.

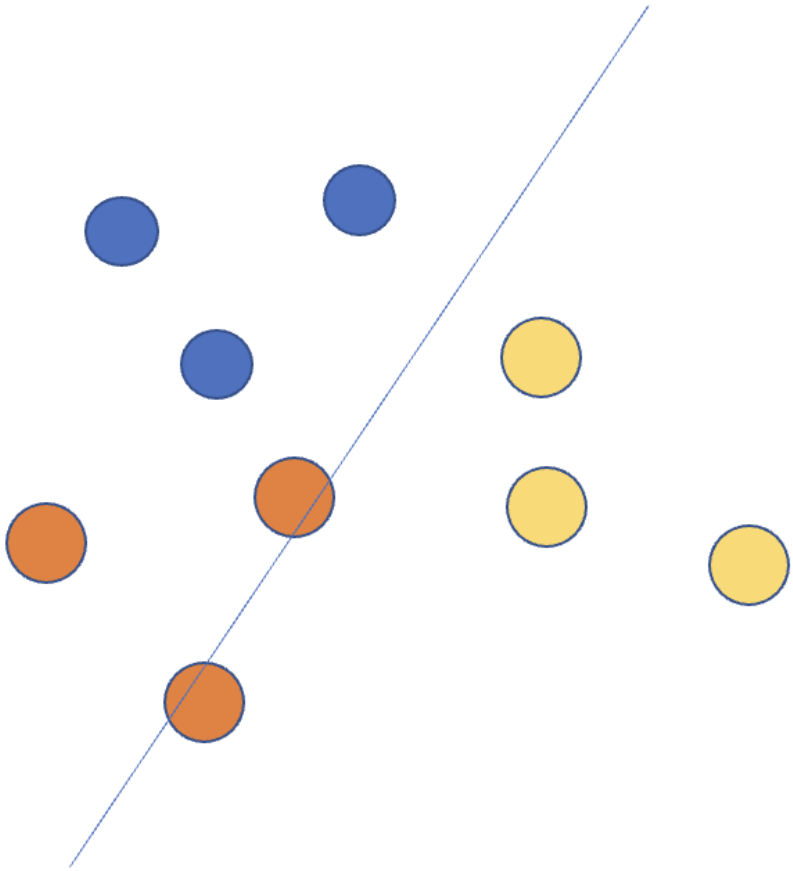
- 또한, 개별 분류기를 사용하게 되면 일관성이 없는 결과를 얻게될 수 있다.
  - 하나의 input value 가 여러개의 다른 class 들에 속하게 되는 결과를 내놓을 수 있다는 것.



- 따라서, K 번의 test 중에서 가장 큰 $y_k(x)$ 를 출력하는 값을 class 로 예측
  - $y_k(x) = \underset{k}{max} y_k(x)$
  - 하지만....
    - 문제점 1 : 각 분류기에서 출력하는 $y_k(x)$ 들이 비교가능한 척도를 가지고 있다는 보장이 없음.
      - 서로 다르게 학습이 되었기 때문에
    - 문제점 2 : binary classifier 의 training set 이 많은 불균형을 이루게 됨.

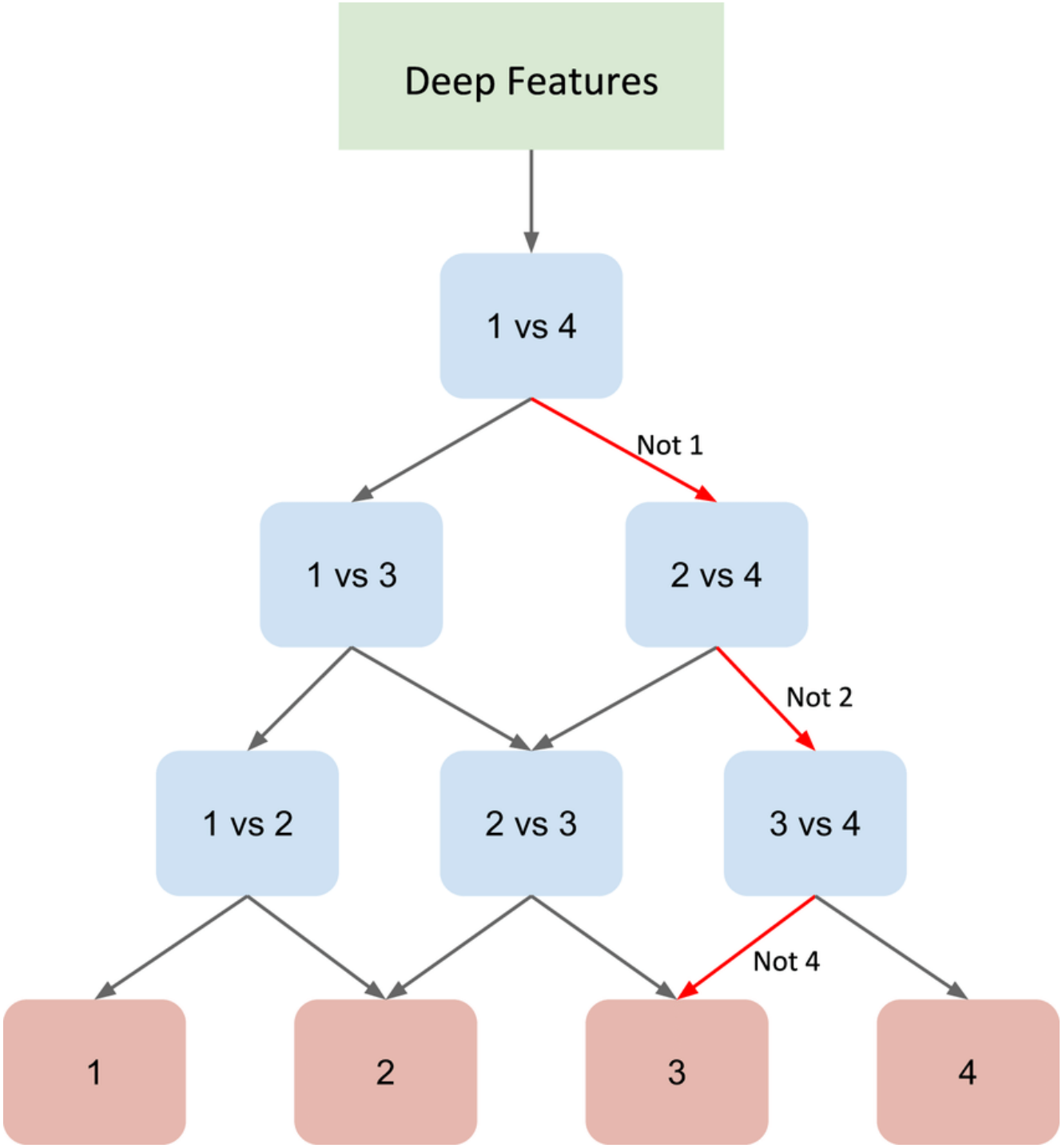
방법 2. One vs One

- K 개의 class 중에서 2개를 선택하여 2 class 에 대한 결정 초평면을 생성한다.



- 이렇게 되면 ${}_K\mathrm{C}_2$ 개의 classifier 가 생성되기 때문에, $\frac{K(K-1)}{2}$ 개의 classifier 가 생성되는 것이다.
  - $y_{i,j}(x)$ 는 c_i, c_j 를 분류하는 decision boundary 인데, 새로 들어오는 input 인 x 가 c_i 로 분류하면 +1 를 얻게 되고, 반대라고 하면 c_j 가 +1 를 얻게 된다.
    - 위 방법을 ${}_K\mathrm{C}_2$ 번 반복하면 가장 많은 값을 얻게된 class 로 x 를 분류하게 되면 된다.
    - 이렇게 되면 K 가 커질 수록 training 시간이 오래걸리고, test 또한 오래걸리게 된다.
      - $O(K^2N^2)$
- 이러한 문제를 개선하기 위하여 나온 것이 **DAGSVM**

DAGSVM



- 총 $\frac{K(K-1)}{2}$ 개의 classifier 가 생성되게 되고, 새로운 test point 를 classifier 하기 위해서는 $K-1$ 개 쌍만 classifier 가 계산하면 되고, 어떤걸 사용하는지는 어떤 경롤르 통해 그래프를 통과하는 지에 따라 달려있음.

결론

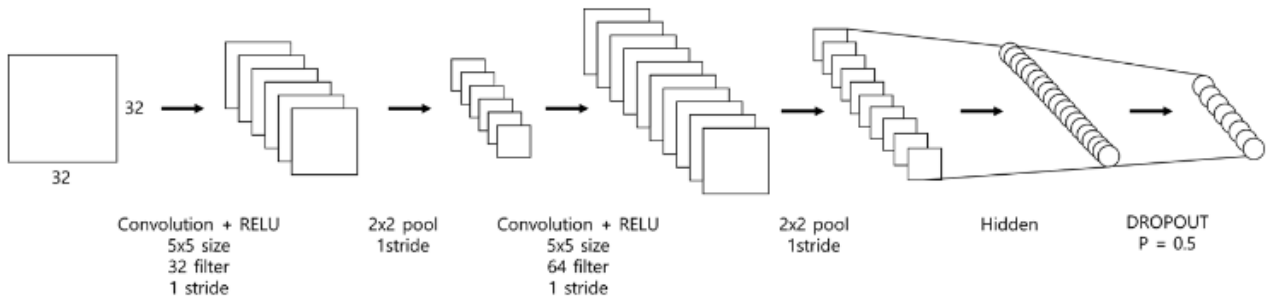
- 사실 이런 다양한 SVM 이 있지만, 요즘은 CNN 에 다 먹힌 상황.... SVM 을 사용한다고 하면 **One vs Rest 를 많이 사용하고 있음**

- 추가로, CNN-SVM 논문이 나왔는데, Softmax Classifier 를 SVM 으로 변경한 것.

Further...

CNN-SVM

- paper : An Architecture Combining Convolutional Neural Network(CNN) and Support Vector Machine(SVM) for Image Classification
- 의의 : 본 연구 결과는 "Deep Learning using Linear Support Vector Machines"의 제안된 CNN-SVM에 대한 검토를 더욱 검증하기 위한 방법론의 개선을 보증하는데 의의를 둔다.



- 마지막 layer단에서 convolutional softmax 대신 **L2-SVM**을 이용

> (1) INPUT : 32 x 32 x 1
> (2) Conv : 5x5, 32, 1
> (3) ReLU
> (4) Pool : 2x2, 1
> (5) Conv : 5x5, 64, 1
> (6) ReLU
> (7) Pool : 2x2, 1
> (8) FC : 1024
> (9) Dropout(0.5)
> (10) FC : 10 // Output Class

Make Model for MNIST Data (CNN)

```
# MNIST CNN 모델 정의
mnist_model = CNN().to(device)

criterion = torch.nn.CrossEntropyLoss().to(device)
optimizer = torch.optim.Adam(mnist_model.parameters(), lr=learning_rate)

total_batch = len(mnist_trainloader)
print('총 배치의 수 : {}'.format(total_batch))
```

Make Model for MNIST Data (CNN + SVM)

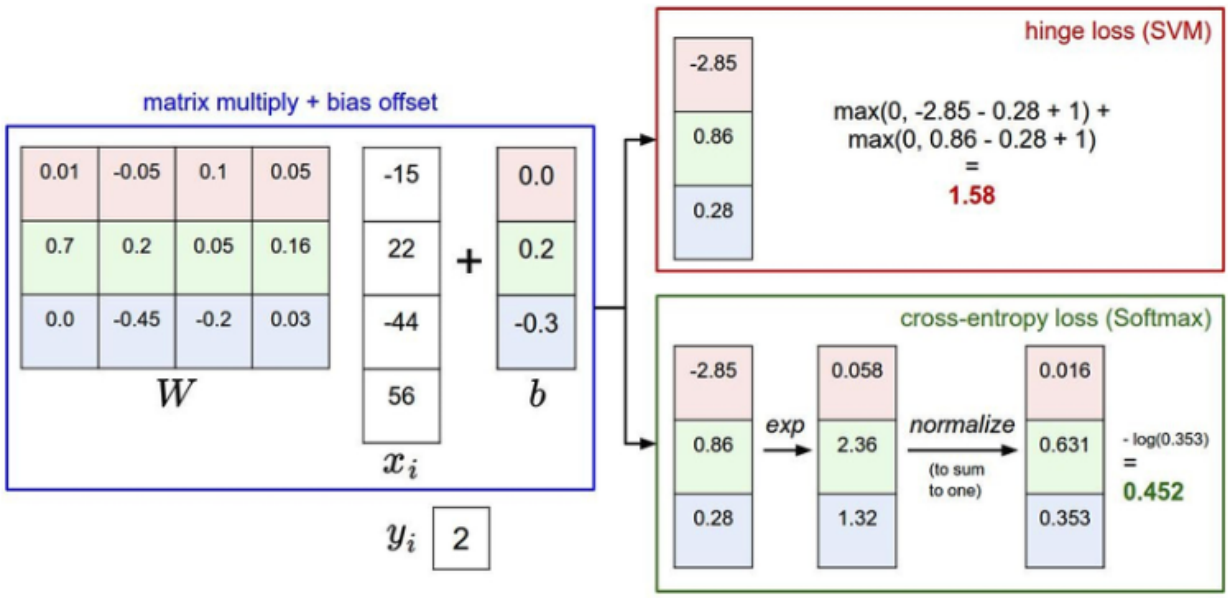
```
# MNIST CNN+SVM 모델 정의
minst_SVM_model = CNN().to(device)

criterion = multiClassHingeLoss().to(device)
optimizer = torch.optim.Adam(minst_SVM_model.parameters(), lr=learning_rate)

total_batch = len(mnist_trainloader)
print('총 배치의 수 : {}'.format(total_batch))
```

Dataset	CNN-Softmax	CNN-SVM
MNIST	99.23%	99.04%
Fashion-MNIST	91.86%	90.72%

Hinge Loss vs Cross-Entropy Loss

- https://lsjsj92.tistory.com/391 굉장히 잘 설명되어있음!
- Hinge Loss
  - $L_i = \sum_{j \neq y_i} \begin{cases} 0, & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1, & \text{otherwise} \end{cases}$
    $= \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$
  - 여기서 s_j 는 정답이 아닌 클래스 score, s_{y_i} 는 정답인 클래스 score 이다.
    - 만약 정답 클래스(s_{y_i})가 정답이 아닌 클래스(s_j) + safety margin(= 1) 보다 크면 loss 는 0
    - 만약 위에 조건이 아니라고 하면 정답이 아닌 클래스(s_j) - 정답 클래스(s_{y_i}) + safety margin(= 1) 를 loss 로 갖는다는 뜻.

    - 즉, 정답 클래스가 정답이 아닌 클래스 + safety margin 보다 크면 그냥 loss 가 0이 되는 것이고, loss 0은 매우 좋다는 뜻이다!
    - 그렇기 때문에 $L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$ 와 같은 식이 나올 수 있다.
- Cross-Entropy Loss
    - $L_i = -logP(Y = y_i|X = x_i) = -log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$



Reference

- PRML 7.1.3
- https://ddiri01.tistory.com/207
- https://velog.io/@euisuk-chung/Paper-Review-An-Architecture-Combining-Convolutional-Neural-NetworkCNN-and-Support-Vector-Machine-SVM-for-Image-Classification
- https://lsjsj92.tistory.com/391
- https://cs.brown.edu/courses/csci1430/proj3/