< Linear Classifier >

① 데이터 가져오기

(1) Normalize

X_train : [50000, 3, 32, 32]

mean_image = X_train.mean(dim=0, keepdim=True).mean(dim=2, keepdim=True).mean(dim=3, keepdim=True)

X_train -= mean_image

X_test -= mean_image

(2) Reshape

X_train = X_train.reshape(X_train.shape[0], -1)

(3) Add bias

$y = xW + b =$ 

ones_train = torch.ones(X_train.shape[0], 1, device = X_train.device)

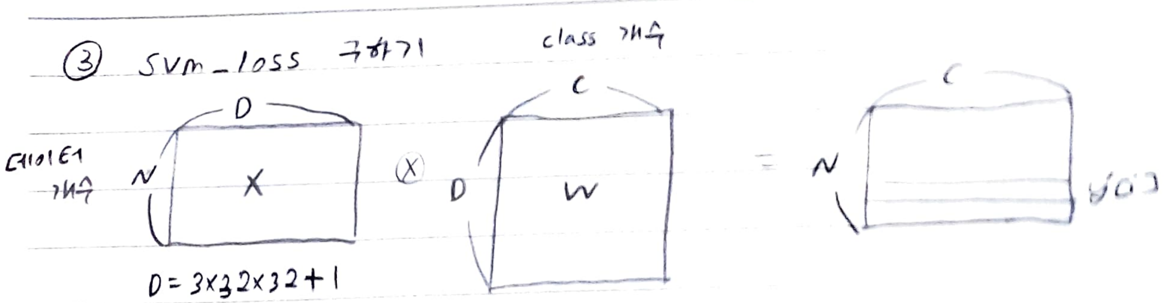
X_train = torch.cat([X_train, ones_train], dim = 1)

② gradient of W

$y = Wx$ 에서 $\frac{dy}{dW}$ 값을 확인 → 이를 바탕으로, W 값을 조정

$$\frac{dy}{dW_{ij}} = \frac{y(W + hM_{ij}) - y(W - hM_{ij})}{2h} \quad \left(M_{ij} = \begin{array}{c} \quad\; j \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \right)$$

③ svm_loss 구하기

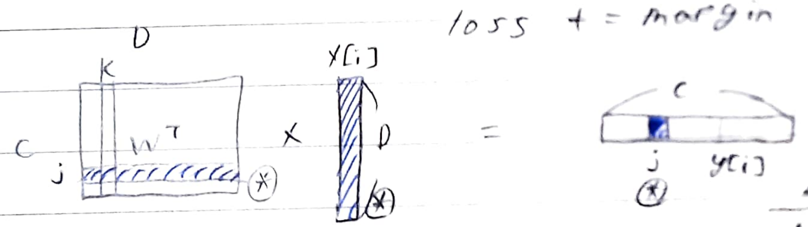


D = 3x32x32+1

ⓥ loss, grad 계산 과정

```
for i in range(N):
    scores = W.t().mv(X[i])
    correct = scores[y[i]]

    for j in range(C):
        if j != y[i]:
            margin = scores[j] - correct + 1
            if margin > 0:
                loss += margin
```



(*)끼리 곱해져서 scores[j] 계산 됨

$$\frac{\Delta scores[j]}{\Delta W^T[j][k]} = X[i][k]$$

⇒ ① $$\frac{\Delta scores[j]}{\Delta W[k][j]} = X[i][k]$$

② $$\frac{\Delta correct}{\Delta W^T[y[i]][k]} = X[i][k]$$

⇒ $$\frac{\Delta correct}{\Delta W[k][y[i]]} = X[i][k]$$

①, ② 에 의해

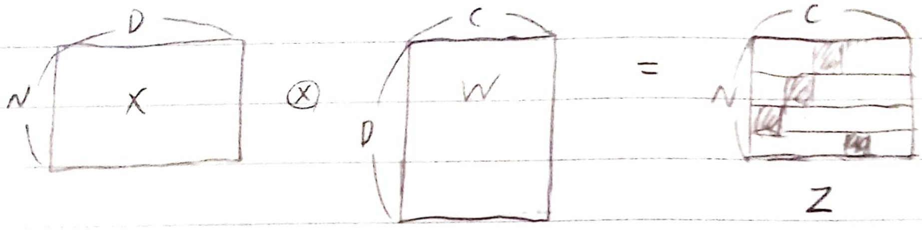
```
dW[:, j] += X[i]
dW[:, y[i]] -= X[i]
```

```
loss = loss / N + reg * torch.sum(W*W)
dW = dW / N + reg * 2 * W
```

④ svm_loss를 for문 없이 구하기 (W, X, y, reg)

y[i] : i번째 data의 class 정보

Z = torch.mm(X, W)



correct_label = torch.arange(N), y

Z[correct_label] = Z[0][2], Z[1][1], Z[2][0], Z[3][3] (ex) y = [2, 1, 0, 3]

margin = torch.max(zero, Z - Z[correct_label] + 1)

broadcasting

정답값에 해당하는 margin은 계산하지 않으므로

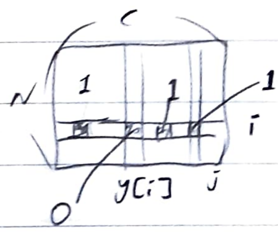
⇒ margin[correct_label] = 0 (j != y[i] 고려해줌)

∴ loss = torch.sum(margin) / N + reg * torch.sum(W * W)

dW은 margin > 0 인 경우만 계산하므로

⇒ ~~mask = torch.zeros_like(margin)~~

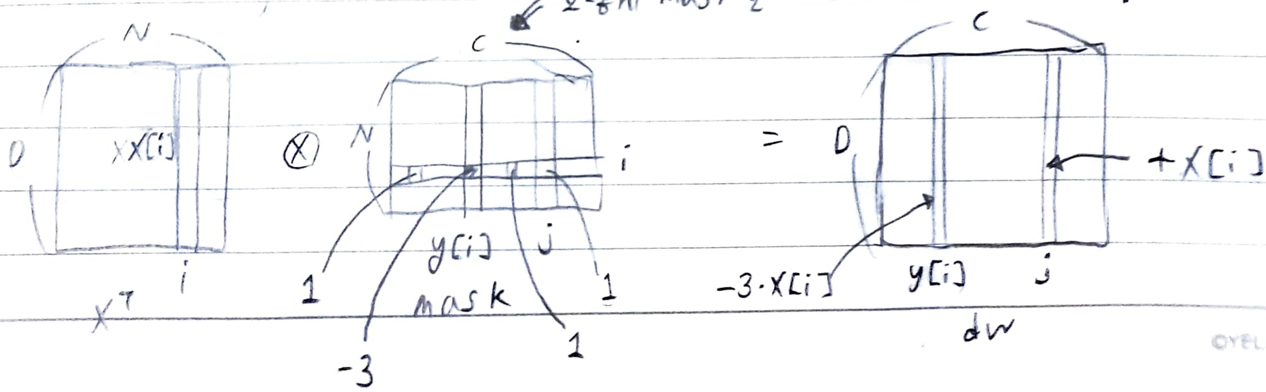
mask[margin > 0] = 1 하고 mask 완성



count = torch.sum(mask, dim=1)
mask[correct_label] = - count

∴ correct 값은 mask[i] 행 중 양수 값 개수 만큼 계산되므로 (그것이 3이라고 하자)

요렇게 mask를 설정하면 된다



<softmax>

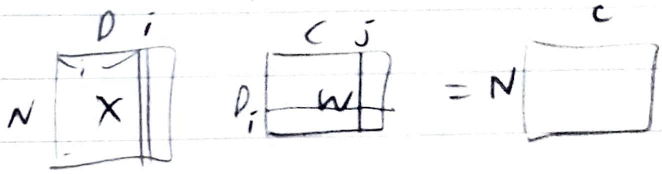
$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

이때, e^f 값이 너무 커질수 있으므로 분모 분자를 큰 값 C로 나누자.

$C = \max(e^{f_i})$ 로 두면 적절하다.

즉, 각 f_i 들에서 $\max(f)$를 뺀 값으로 계산해도 무방하다.

dW 계산의 과정을 살펴보자.



$$\frac{dL}{dW_{ij}} = \sum_{k=1}^{N} \frac{dL_k}{dW_{ij}} = \sum_k \frac{df_{kj}}{dW_{ij}} \cdot \frac{dL_k}{df_{kj}} = \sum_k X[k][i] \cdot \frac{dL_k}{df_{kj}}$$

① $j = y[k]$ 일 때 ($y[k]$: k번째 sample의 정답 class)

$$\frac{dL_k}{df_{kj}} = \frac{d\left(-\log \frac{e^{f_{y_k}}}{\sum_j e^{f_j}}\right)}{df_{y_k}} = \frac{de^{f_{y_k}}}{df_{y_k}} \cdot \frac{d\left(-\log \frac{e^{f_{y_k}}}{\sum_j e^{f_j}}\right)}{de^{f_{y_k}}}$$

$$= e^{f_{y_k}} \cdot \left(\frac{d\left(-\log \frac{x}{x+\square}\right)}{dx}\right)$$

$$= e^{f_{y_k}} \cdot \left(-\frac{1}{x} + \frac{1}{x+\square}\right) = \left(-1 + \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

② $j \neq y[i]$

$$\frac{dL_i}{df_j} = \frac{e^{f_j}}{\sum_j e^{f_j}}$$