Assingment2_a_b_c

November 20, 2019

2. 아래의 문제를 푸세요.

a. Cross entropy is often used as the objective function when training neural network in classification problems. Suppose the training set includes N traning Pairs $D = \{(x_i^{(train)}, y_i^{(train)})\}_{i=1}^{N}$, when x_i^{train} is a training sample and $y_i^{train} \in \{(1, ..., c\}$ is its class label. z_i is the output of the network given input $x_i^{(train)}$ and the nonlinearity of the output layer is softmax. z_i is a c dimensional vector, $z_{i,k} \in [0, 1]$ and $\sum_{k=1}^{c} z_{i,k} = 1$. Please write the objective function of cross entropy and show that it is equivalent to the negative log-likelihood on the training set, assuming the training samples are indepedent. (10 점)

sol> a "True" probability P and an estimated distribution Q의 두 개의 확률 분포의 Cross-entropy $H(P, Q)$는 다음과 같다.

$$H(P, Q) = -\sum_{x} P(x) log Q(x)$$

k-th class's probability = softmax(z_i) = $z_{i,k} = \frac{e^{z_{i,k}}}{\sum_{k=1}^{c} e^{z_{i,k}}}$ 는 모델의 예측하는 Q(x)이다. P(X)는 correct class에 대한 확률 분포로 항상 correct class의 position만 1인 one-hot vector로 표현된다. (i.e p=[0, 1, , 0] contains a single 1 at the y_i-th position.). 또한 cross-entropy는 다음과 같이 Kullback-Leibler divergence로 쓸 수 있다.

$$H(P, Q) = H(P) + D_{KL}(P||Q)$$

위의 식에서 H(P)는 0 이고 $D_{KL}(P||Q) = \sum_x P(x) log \frac{P(x)}{Q(x)} = \sum_x P(x) log P(x) - P(x) log Q(x)$가 되므로 위의 cross-entropy는 correct class에 대하여 true distribution P(x)와 모델의 예측 distribution Q(x)를 가지고 cross-entropy를 계산하면 다음과 같이 negative log-likelihood가 된다.

$$H(P, Q) = -\sum_{x} log Q(x)$$

b. Desin a three-layer neural network whose decision boundary is as shown in Figure 1. The gray region belongs to class 1 and other region belongs to class 0. Show your network structure, weight and nonlinear activation function. (10점)

(0,0)~(1,1)를 지나는 직선 : $y < x$ 과 (1,1)~(2,0)를 지나는 직선 : $y < -x + 2$의 두 개의 식을 만족하는 아래의 같은 MLP_1

$$MLP_1 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} + \begin{pmatrix} 0 \\ -2 \end{pmatrix}$$

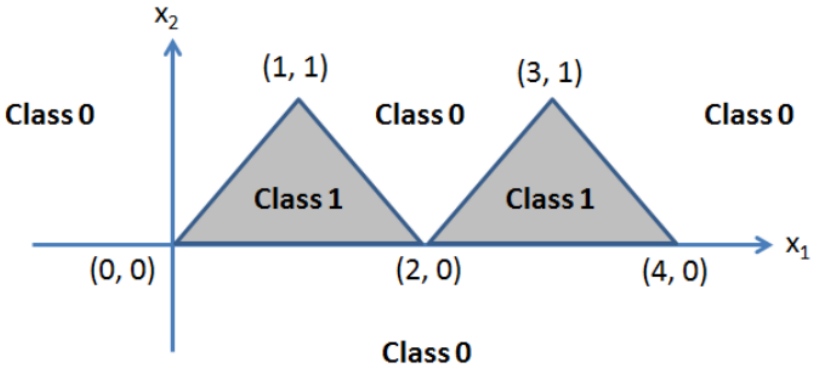

Figure 1

$$layer1 = StepFunction(MLP_1)$$

(0,0)~(2,0)를 지나는 직선 : $y > 0$ 과 (2,0)~(4,0)를 지나는 직선 : $y > 0$의 두 개의 식을 만족하는 아래와 같은 MLP_2

$$MLP_2 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} layer1 + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$layer2 = StepFunction(MLP_2)$$

(2,0)~(3,1)를 지나는 직선 : $y < x - 2$ 과 (3,1)~(4,0)를 지나는 직선 : $y < -x + 4$의 두 개의 식을 만족하는 아래와 같은 MLP_3

$$MLP_3 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} layer2 + \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

$$output = StepFunction(MLP_3)$$

c. Equivariance is an appealing property when design neural network operations. It means that transforming the input image (e.g., translation) will alos transform the output feature maps similarly after certain operations.

Formally, denote the image coordinate by $x \in \mathbb{Z}^2$, and the pixel values at each coordinate by a function $f : \mathbb{Z}^2 \to \mathbb{R}^K$, Where K is the number of image channels. A convolution filter can also be formulated as a function $w : \mathbb{Z}^2 \to \mathbb{R}^K$. Note that f and w are zero outiside the image and filter kernel region, respectively. The convolution operation (correlation indeed for simplicity) is thus defined by $[f * w](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K} f_k(y) W_k(y - x)$.

(c-1) Let Lt be the translation $x \to x + t$ on the image or feature map, i.e., $[L_t f](x) = f(x - t)$. Prove that convolution has equivariance to translation: $[[L_t f] * w](x) = [[L_t [f * w]]](x)$, which

means that first tranlsating the input image then doing the convolution is equivalent to first convolving with the image and then translating the output feature map. (15점).

sol>$[[L_t f] * w](x) = [f(x-t) * w](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K} f_k(f(x-t)) W_k(f(x-t)-x)$ 이고 $[[L_t[f * w]](x) = [L_t * \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K} f_k(y) W_k(y-x)](x) = f(\sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K} f_k(y) W_k(y-x)-t)$ 이므로 위의 식과 유사함으로 translation에 equivariance를 가진다.

(c-2) Let L_R be the 90-rotation on the image or feature map,$R = \begin{pmatrix} cos(\pi/2) & -sin(\pi/2) \\ sin(\pi/2) & cos(\pi/2) \end{pmatrix}$, then $[L_R f](x) = f(R^{-1}x)$. However, convolution is not equivariant to rotations, i.e., $[L_R f] * w \neq L_R[f * w]$. In order to establish the equivalence, the filter also needs to be rotated. Prove that: $[[L_R f] * w](x) = L_R[f * [L_{R^{-1}} w]](x)$. (15 점)

sol> $L_R[f * [L_{R^{-1}} w]](x) = L_R[f * f(w^{-1}Rx)]$ 과 $[[L_R f] * w](x) = [f(R^{-1}x) * w]$ 이므로 convolution is not equivariante to rotation.

Reference

1. Tensorflow-Example of aymericdamien github
2. MahanFathi github
3. jariasf github