Derivation of Cross-Entropy Loss Function

Definition of Cross-Entropy

The cross-entropy between the true distribution P and the predicted distribution Q is defined as:

$$H(P, Q) = -\sum_x p(x) \log q(x)$$

where:

- $p(x)$ is the probability of event x under the true distribution P.
- $q(x)$ is the probability of event x under the predicted distribution Q.

Definition of Information Entropy

The information entropy of the true distribution P is given by:

$$H(P) = -\sum_x p(x) \log p(x)$$

Definition of Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence from the true distribution P to the predicted distribution Q is defined as:

$$\mathrm{KL}(P\|Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Relationship Between Cross-Entropy and KL Divergence

Expanding the KL divergence:

$$\mathrm{KL}(P\|Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

This can be separated into:

$$\mathrm{KL}(P\|Q) = \sum_x p(x) [\log p(x) - \log q(x)]$$

Which simplifies to:

$$\mathrm{KL}(P\|Q) = \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x)$$

Notice that the first term is the entropy of the true distribution P, and the second term is the cross-entropy:

$$\mathrm{KL}(P\|Q) = -H(P) + H(P,Q)$$

Thus, the cross-entropy $H(P,Q)$ can be expressed as:

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

Cross-Entropy Loss Function

In machine learning, the cross-entropy loss function is used to measure the difference between the predicted distribution Q and the true distribution P. Minimizing this loss function optimizes the model's performance. The loss function can be written as:

$$\mathrm{Loss} = -\sum_x p(x)\log q(x)$$

This demonstrates that the cross-entropy loss function combines the information entropy of the true distribution and the KL divergence between the true distribution and the predicted distribution.

Relationship Between Information Entropy and Softmax Classifier Cross-Entropy Loss

Softmax Function and Probability Distribution

The Softmax function transforms the raw model outputs (logits) into a probability distribution. Given a vector of raw outputs z, the Softmax function's output $\hat{y}_i$ for the i-th class is:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Here, $\hat{y}_i$ represents the predicted probability for the i-th class.

Information Entropy

Information entropy measures the uncertainty of a probability distribution p. For a probability distribution p where p_i is the probability of the i-th class, the entropy $H(p)$ is defined as:

$$H(p) = -\sum_i p_i \log p_i$$

Information entropy quantifies the uncertainty or randomness inherent in the distribution.

Cross-Entropy Loss

In classification tasks, the cross-entropy loss function measures the difference between the predicted probability distribution and the true label distribution. Given a true label one-hot vector y and the predicted probability distribution $\hat{y}$, the cross-entropy loss L is defined as:

$$L = -\sum_i y_i \log \hat{y}_i$$

Here:

- y_i is the element of the true label vector y (typically 0 or 1).
- $\hat{y}_i$ is the predicted probability for the i-th class.

Relationship Between Cross-Entropy and KL Divergence

The relationship between cross-entropy $H(P, Q)$ and Kullback-Leibler (KL) divergence can be expressed as:

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

where:

- $H(P)$ is the entropy of the true distribution P.
- $\mathrm{KL}(P \| Q)$ is the KL divergence from the true distribution P to the predicted distribution Q.

In classification tasks, if the true label y is a one-hot vector, the cross-entropy loss function L is a special case of the KL divergence $\mathrm{KL}(P \| \hat{y})$:

$$L = -\sum_i y_i \log \hat{y}_i = \mathrm{KL}(P \| \hat{y})$$

where P is a one-hot vector and $\hat{y}$ is the model's predicted probability distribution.

Summing the Loss Across All Samples

In training, the loss for each image is computed as described above. To get the overall loss for the batch, the losses for all images are summed (or averaged):

$$L_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} L_i$$

where:

- N is the total number of samples.
- L_i is the cross-entropy loss for the i-th image.

Training Process

During training, the average loss L_{avg} is used to compute gradients and update the model parameters through backpropagation. This ensures that the model is optimized to minimize the loss across all samples, making the training process more stable and effective.

Gradient of the Total Loss with Respect to Weights

Given:

- N is the total number of training samples.
- C is the total number of classes.
- W is the weight matrix with a number of rows C*D, where D is the dimension of input X.
- L_i is the loss for the i-th training sample.
- L is the total loss.
- R is the regularization term.
- λ is the hyperparameter for regularization.
- $f_{j,i}$ is the score for class j for the i-th sample.
- $\hat{y}_{j,i}$ is the probability for class j for the i-th sample.
- $x_{d,n}$ is the d-th feature of the n-th sample.

The total loss L is:

$$L_{\text{total}} = \frac{1}{N}\sum_{n=1}^{N} L_n + \lambda \sum_{c=1}^{C}\sum_{d} W_{c,d}^2$$

where the individual loss L_i is:

$$L_n = -f_{k,n} + \log\left(\sum_{c=1}^{C} e^{f_{c,n}}\right)$$

For a single training sample n, the cross-entropy loss L_n is given by:

$$L_n = -\log(\hat{y}_{k,n})$$

where:

- $\hat{y}_{k,n}$ is the predicted probability of the true class k for the n-th sample.

- k is the true class label for the n-th sample.

The predicted probability $\hat{y}_{k,n}$ is computed using the softmax function applied to the scores (logits) produced by the model:

$$\hat{y}_{k,n} = \frac{e^{f_{k,n}}}{\sum_j e^{f_{j,n}}}$$

Define the probability for class j:

$$\hat{y}_{j,i} = \frac{e^{f_{j,i}}}{\sum_{c=1}^{C} e^{f_{c,i}}}$$

The gradient of L_n with respect to $f_{j,n}$ is:

$$\frac{\partial L_n}{\partial f_{j,n}} = \hat{y}_{j,n} - \delta_{j,k}$$

where $\delta_{j,k}$ is the Kronecker delta, which is 1 if $j = k$ and 0 otherwise.
The score $f_{j,n}$ is given by:

$$f_{j,n} = W_{j,d} x_{d,n}$$

The gradient of $f_{j,n}$ with respect to $W_{c,d}$ is:

$$\frac{\partial f_{j,n}}{\partial W_{c,d}} = x_{d,n} \cdot \delta_{j,c}$$

Thus, the gradient of L_n with respect to $W_{c,d}$ is:

$$\frac{\partial L_n}{\partial W_{c,d}} = \frac{\partial L_n}{\partial f_{j,n}} \cdot \frac{\partial f_{j,n}}{\partial W_{c,d}} = (\hat{y}_{j,n} - \delta_{j,k}) \cdot (x_{d,n} \cdot \delta_{j,c}) = (\hat{y}_{c,n} - \delta_{c,k}) \cdot x_{d,n}$$

Finally, the total loss L_{total} is:

$$L_{\text{total}} = \frac{1}{N} \sum_{n=1}^{N} L_n + \lambda \sum_{c=1}^{C} \sum_{d} W_{c,d}^2$$

The gradient of the total loss L_{total} with respect to $W_{m,n}$ is:

$$\frac{\partial L_{\text{total}}}{\partial W_{c,d}} = \frac{1}{N} \sum_{n=1}^{N} (\hat{y}_{c,n} - \delta_{c,k}) \cdot x_{d,n} + 2\lambda W_{c,d}$$

Question 1:

Why do we expect our loss to be close to -log(0.1)? Explain briefly.

Answer:
This is because:

- Number of Classes: With 10 classes, if the model is uncertain or performing poorly, it might predict each class with roughly equal probability. For a uniform distribution, the probability for each class would be $\frac{1}{10} = 0.1$.
- Cross-Entropy Loss: The loss is given by the negative logarithm of the predicted probability for the true class. If the probability assigned to the true class is 0.1, the loss for that sample will be:

$$L = -\log(0.1) \approx 2.3026$$

- Total Loss: If the model's predictions are consistently poor across samples, the average loss for the entire dataset will be close to this value, reflecting the low confidence and poor performance of the model.

In summary, the total loss being close to $-log(0.1)$ indicates that the model's average predicted probability for the true class is low, which is expected when the model is not performing well.

Question 2:

Suppose the overall training loss is defined as the sum of the per-datapoint loss over all training examples. It is possible to add a new datapoint to a training set that would leave the SVM loss unchanged, but this is not the case with the Softmax classifier loss.

Answer:
Yes
Explanation::
For an SVM, the overall training loss is determined by the sum of hinge losses across all training examples. The hinge loss for each example depends on whether the example is correctly classified and how far it is from the margin. If you add a new datapoint that does not affect the margin of existing datapoints or if it lies within the margin (i.e., the hinge loss for this new point is zero), the overall training loss can remain unchanged.
In contrast, for a Softmax classifier, the loss function is based on cross-entropy. Adding a new datapoint to the training set will typically increase the overall loss because it introduces an additional term to the loss function, which increases the total cross-entropy loss. This is because each new datapoint contributes positively to the overall loss calculation, reflecting its contribution to the classification error. Thus, the loss for the Softmax classifier is more directly affected by the addition of new datapoints.