multiclass classification theoretical questions

Omar Ayman Bakr

June 2025

1 soft max input shifting

Let the input vector be:

$$\mathbf{z} = [z_1, z_2, \ldots, z_n]$$

The softmax function is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Now suppose we add a constant C to every element of $\mathbf{z}$, creating a new vector:

$$\mathbf{z}' = [z_1 + C, z_2 + C, \ldots, z_n + C]$$

Then the softmax becomes:

$$\text{softmax}(z_i + C) = \frac{e^{z_i + C}}{\sum_{j=1}^{n} e^{z_j + C}} = \frac{e^{z_i} e^{C}}{\sum_{j=1}^{n} e^{z_j} e^{C}} = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Thus,

$$\text{softmax}(z_i + C) = \text{softmax}(z_i)$$

Conclusion: Adding a constant to all inputs does not change the softmax output. This is useful for numerical stability, e.g., by subtracting $\max(z)$ from all elements.

2 Softmax vs Sigmoid

Let the input vector be:

$$\mathbf{x} = [x_0, x_1]$$

The softmax function for class i is defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{e^{x_0} + e^{x_1}}$$

We examine the probability assigned to class 1:

$$\text{softmax}(x_1) = \frac{e^{x_1}}{e^{x_0} + e^{x_1}}$$

Now divide numerator and denominator by e^{x_1}:

$$\text{softmax}(x_1) = \frac{1}{\frac{e^{x_0}}{e^{x_1}} + 1} = \frac{1}{e^{x_0 - x_1} + 1}$$

This is the definition of the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

So,

$$\text{softmax}(x_1) = \sigma(x_1 - x_0)$$

Conclusion: In the binary case, softmax applied to two values is equivalent to applying the sigmoid function to the **difference between the two logits**. This confirms the equivalence:

$$\text{softmax}(x_1) = \sigma(x_1 - x_0)$$

3 Softmax Derivative

We want to prove the following formula for the derivative of the softmax function:

$$\frac{\partial s(x)_i}{\partial x_j} = \begin{cases} s(x)_i(1 - s(x)_i) & \text{if } i = j \\ -s(x)_i s(x)_j & \text{if } i \neq j \end{cases}$$

To make the proof concrete and beginner-friendly, we consider a simple case where the input vector is:

$$x = [x_1, x_2, x_3]$$

and the softmax output is:

$$z_i = \frac{e^{x_i}}{e^{x_1} + e^{x_2} + e^{x_3}} \quad \text{for } i = 1, 2, 3$$

Let $D = e^{x_1} + e^{x_2} + e^{x_3}$ be the denominator. Then:

$$z_3 = \frac{e^{x_3}}{D}, \quad z_1 = \frac{e^{x_1}}{D}$$

Case 1: $\frac{\partial z_3}{\partial x_3}$

Using the quotient rule:

$$\frac{\partial z_3}{\partial x_3} = \frac{e^{x_3} \cdot D - e^{x_3} \cdot e^{x_3}}{D^2} = \frac{e^{x_3}(D - e^{x_3})}{D^2}$$

Factoring:

$$= \frac{e^{x_3}}{D}\left(1 - \frac{e^{x_3}}{D}\right) = z_3(1 - z_3)$$

This matches the formula for $i = j$.

Case 2: $\frac{\partial z_3}{\partial x_1}$

Since e^{x_3} does not depend on x_1, we have:

$$\frac{\partial z_3}{\partial x_1} = \frac{0 \cdot D - e^{x_3} \cdot e^{x_1}}{D^2} = -\frac{e^{x_3} e^{x_1}}{D^2}$$

But:

$$z_3 = \frac{e^{x_3}}{D}, \quad z_1 = \frac{e^{x_1}}{D} \quad \Rightarrow \quad \frac{\partial z_3}{\partial x_1} = -z_3 z_1$$

This matches the formula for $i \neq j$.

Conclusion

In both cases, we verified that:

$$\frac{\partial s(x)_i}{\partial x_j} = \begin{cases} s(x)_i(1 - s(x)_i) & \text{if } i = j \\ -s(x)_i s(x)_j & \text{if } i \neq j \end{cases}$$

holds true for the softmax function.

4 Softmax with Cross Entropy

Let:

- z_i: the logit for class i
- $\hat{y}_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$: the predicted probability for class i using the softmax function
- y_i: the ground truth label for class i, one-hot encoded (i.e., $y_i = 1$ for the correct class, otherwise 0)
- $\mathcal{L}$: the cross-entropy loss, defined as:

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

We want to compute the derivative of the loss with respect to the logit z_k for some class k. Using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial z_k} = \sum_i \frac{\partial \mathcal{L}}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_k}$$

First, we compute:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

Second, we use the derivative of the softmax function:

$$\frac{\partial \hat{y}_i}{\partial z_k} = \begin{cases} \hat{y}_k(1 - \hat{y}_k) & \text{if } i = k \\ -\hat{y}_i \hat{y}_k & \text{if } i \neq k \end{cases}$$

Plugging into the chain rule:

$$\frac{\partial \mathcal{L}}{\partial z_k} = \sum_i \left(-\frac{y_i}{\hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_k} \right)$$

Split into two cases:
Case 1: $i = k$**:**

$$-\frac{y_k}{\hat{y}_k} \cdot \hat{y}_k(1 - \hat{y}_k) = -y_k(1 - \hat{y}_k)$$

Case 2: $i \neq k$**:**

$$\sum_{i \neq k} \left(-\frac{y_i}{\hat{y}_i} \cdot (-\hat{y}_i \hat{y}_k) \right) = \sum_{i \neq k} y_i \hat{y}_k$$

So we get:

$$\frac{\partial \mathcal{L}}{\partial z_k} = -y_k(1 - \hat{y}_k) + \sum_{i \neq k} y_i \hat{y}_k$$

Since the label vector y is one-hot encoded (i.e., $\sum_i y_i = 1$), the full expression simplifies to:

$$\frac{\partial \mathcal{L}}{\partial z_k} = \hat{y}_k - y_k$$

Kullback-Leibler (KL)-Divergence

Let:

- $P(x)$: the true distribution (ground truth labels)
- $Q(x)$: the predicted probability distribution

1. Cross-Entropy

The cross-entropy between $P(x)$ and $Q(x)$ is defined as:

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

2. KL Divergence (as in the image)

The KL divergence from $P(x)$ to $Q(x)$ is defined as:

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

3. Simplify Using Log Rules

Apply the identity $\log \frac{a}{b} = \log a - \log b$:

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

The first term is the (negative) entropy of P:

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

So:

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

4. Conclusion

Since $H(P)$ is constant (the true distribution doesn't change during training), minimizing $D_{\mathrm{KL}}(P \,||\, Q)$ is equivalent to minimizing cross-entropy $H(P, Q)$.

Therefore, minimizing KL divergence is equivalent to minimizing cross-entropy when $P(x)$ is fixed.