CrossEntropyLoss, KL Divergence, Entropy, and Logits

March 21, 2025

1 Introduction

This document explores CrossEntropyLoss, Kullback-Leibler (KL) Divergence, entropy $H(P)$, and the role of logits in machine learning, with examples for two-class and three-class classification, and an application to LLM distillation including temperature effects.

2 CrossEntropyLoss and KL Divergence

CrossEntropyLoss measures the difference between a true distribution P and a predicted distribution Q:

$$H(P, Q) = -\sum_i P(i) \log Q(i),$$

while KL Divergence quantifies how Q diverges from P:

$$D_{KL}(P||Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right).$$

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

This is a mathematical identity:

$$D_{KL}(P||Q) = \sum_i P(i) \left(\log P(i) - \log Q(i) \right) = -H(P) + H(P, Q),$$

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

where $H(P)$ is the entropy of P, and D_{KL} is the additional cost of using Q.

2.2 Two-Class Example

True label: class 1, $P = [0, 1]$, predicted $Q = [0.3, 0.7]$.

2.2.1 CrossEntropyLoss

$$H(P, Q) = -[0 \cdot \log(0.3) + 1 \cdot \log(0.7)] = -\log(0.7) \approx 0.3567$$

2.2.2 KL Divergence

$$D_{KL}(P||Q) = 1 \cdot \log\left(\frac{1}{0.7}\right) = 0.3567$$

Since $H(P) = 0$, $H(P, Q) = D_{KL}(P||Q)$.

3 Entropy $H(P)$

Entropy measures uncertainty in P:

$$H(P) = -\sum_i P(i) \log P(i)$$

3.1 Two-Class Example

For $P = [0, 1]$:

$$H(P) = -[0 \cdot \log(0) + 1 \cdot \log(1)] = 0$$

(Convention: $0 \log(0) = 0$).

For $P = [0.5, 0.5]$:

$$H(P) = -2 \cdot (0.5 \log(0.5)) \approx 0.6931$$

3.2 Three-Class Example

For $P = [0, 1, 0]$:

$$H(P) = -[0 \cdot \log(0) + 1 \cdot \log(1) + 0 \cdot \log(0)] = 0$$

For $P = [0.33, 0.33, 0.33]$:

$$H(P) = -3 \cdot (0.33 \log(0.33)) \approx 1.0974$$

4 Role of Logits

Logits are raw scores, converted to probabilities via softmax:

$$Q(i) = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

They are used directly in CrossEntropyLoss.

4.1 Two-Class Logit Example

Logits: $z = [1.0, 2.0]$, true label: class 1 ($P = [0, 1]$).

4.1.1 Softmax

$$e^{1.0} \approx 2.718, \quad e^{2.0} \approx 7.389$$

$$\text{Sum} = 10.107$$

$$Q = [0.269, 0.731]$$

4.1.2 CrossEntropyLoss

$$\text{Loss} = -\log(0.731) \approx 0.313$$

If $z = [-1.0, 1.0]$:

$$Q = [0.119, 0.881], \quad \text{Loss} = -\log(0.881) \approx 0.127$$

4.2 Three-Class Logit Example

Logits: $z = [2.0, 1.0, -1.0]$, true label: class 1 ($P = [0, 1, 0]$).

4.2.1 Softmax

$$e^{2.0} \approx 7.389, \quad e^{1.0} \approx 2.718, \quad e^{-1.0} \approx 0.368$$

$$\text{Sum} = 10.475$$

$$Q = [0.705, 0.259, 0.035]$$

4.2.2 CrossEntropyLoss

$$\text{Loss} = -\log(0.259) \approx 1.349$$

If $z = [0.5, 2.5, -0.5]$:

$$Q = [0.114, 0.844, 0.042], \quad \text{Loss} = -\log(0.844) \approx 0.169$$

5 CrossEntropyLoss with Non-Zero Entropy: LLM Distillation Example

In LLM distillation, P is a soft distribution (e.g., from a teacher model), so $H(P) \neq 0$. CrossEntropyLoss remains applicable.

5.1 Three-Class Distillation Example

Teacher's $P = [0.7, 0.2, 0.1]$, student's $Q = [0.6, 0.3, 0.1]$.

5.1.1 Entropy $H(P)$

$$H(P) = -[0.7\log(0.7) + 0.2\log(0.2) + 0.1\log(0.1)]$$
$$\approx -[0.7 \cdot (-0.3567) + 0.2 \cdot (-1.6094) + 0.1 \cdot (-2.3026)]$$
$$\approx 0.8019$$

5.1.2 CrossEntropyLoss

$$H(P, Q) = -[0.7\log(0.6) + 0.2\log(0.3) + 0.1\log(0.1)]$$
$$\approx -[0.7 \cdot (-0.5108) + 0.2 \cdot (-1.2040) + 0.1 \cdot (-2.3026)]$$
$$\approx 0.8287$$

5.1.3 KL Divergence

$$D_{KL}(P||Q) = H(P, Q) - H(P) \approx 0.8287 - 0.8019 = 0.0268$$

Minimizing $H(P, Q)$ reduces D_{KL}.

5.2 Temperature in Distillation

In distillation, a temperature T softens distributions by scaling logits before softmax:

$$P(i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}.$$

Higher T flattens probabilities, emphasizing softer targets.

5.2.1 Example with Temperature

Teacher logits: $z_P = [2.0, 1.0, 0.0]$, student logits: $z_Q = [1.5, 1.0, 0.5]$, $T = 2$.

- Teacher P:

$$e^{2.0/2} = e^1 \approx 2.718, \quad e^{1.0/2} \approx 1.649, \quad e^{0.0/2} = 1$$
$$\text{Sum} = 5.367$$
$$P = [0.506, 0.307, 0.186]$$

- Student Q:

$$e^{1.5/2} \approx 2.117, \quad e^{1.0/2} \approx 1.649, \quad e^{0.5/2} \approx 1.284$$
$$\text{Sum} = 5.050$$
$$Q = [0.419, 0.327, 0.254]$$

5.2.2 CrossEntropyLoss with Temperature

$$\begin{aligned} H(P,Q) &= -[0.506\log(0.419) + 0.307\log(0.327) + 0.186\log(0.254)] \\ &\approx -[0.506\cdot(-0.870) + 0.307\cdot(-1.118) + 0.186\cdot(-1.371)] \\ &\approx 1.038 \end{aligned}$$

At $T = 1$, $P = [0.665, 0.245, 0.090]$, $Q = [0.558, 0.245, 0.197]$, $H(P,Q) \approx 0.842$. Higher T increases loss but smooths learning.

6 Conclusion

The identity $H(P,Q) = H(P) + D_{KL}(P||Q)$ holds universally. For one-hot P, $H(P) = 0$, so $H(P,Q) = D_{KL}(P||Q)$. In distillation, $H(P) \neq 0$, and temperature adjusts softness, yet CrossEntropyLoss remains effective.

1. Classification Label Formats

In multi-class classification, there are two common ways to represent the true class labels:

1. **One-hot encoded vectors:** Each class label is represented by a vector of length C (number of classes), where only one element is 1 and the rest are 0.

   Example: $y = [0, 1, 0]$ for class 1 in 3-class classification

2. **Integer class indices:** The label is a single integer indicating the class index.

   Example: $y = 1$ for class 1

2. Categorical vs. Sparse Categorical Cross-Entropy

- **Categorical Cross-Entropy (CCE)** is used when labels are one-hot encoded.
- **Sparse Categorical Cross-Entropy (SCCE)** is used when labels are integer indices.

CCE Formula (for one-hot labels)

Let $\hat{y}_i$ be the predicted probability for class i, and y_i be the one-hot encoded true label:

$$\mathcal{L}_{\text{CCE}} = -\sum_{i=0}^{C-1} y_i \log(\hat{y}_i)$$

SCCE Formula (for integer labels)

Let y be the correct class index, and $\hat{y}_y$ the predicted probability for that class:

$$\mathcal{L}_{\text{SCCE}} = -\log(\hat{y}_y)$$

3. PyTorch `CrossEntropyLoss`

In PyTorch, `nn.CrossEntropyLoss()` implements Sparse Categorical Cross-Entropy. It:

- Takes raw logits (not softmax probabilities) as input.
- Takes integer class labels (not one-hot).
- Internally applies `log_softmax` + `NLLLoss`.

3.1 Inputs

- $\mathbf{z} = [z_0, z_1, \ldots, z_{C-1}]$: raw logits (model outputs before softmax)
- $y \in \{0, 1, \ldots, C-1\}$: correct class index

3.2 Computation Steps

1. Compute softmax probabilities:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=0}^{C-1} e^{z_j}}$$

2. Compute negative log-likelihood for true class y:

$$\mathcal{L} = -\log(\hat{y}_y) = -\log\left(\frac{e^{z_y}}{\sum_{j=0}^{C-1} e^{z_j}}\right)$$

3. Final simplified form (used internally by PyTorch):

$$\boxed{\mathcal{L} = -z_y + \log\left(\sum_{j=0}^{C-1} e^{z_j}\right)}$$

4. Example

Suppose the model predicts logits for a 3-class problem:

$$\mathbf{z} = [2.0, 1.0, 0.1], \quad \text{and } y = 0$$

Then:

$$\text{Softmax: } \hat{y}_0 = \frac{e^2}{e^2 + e^1 + e^{0.1}} \approx 0.659$$
$$\text{Loss: } \mathcal{L} = -\log(\hat{y}_0) \approx -\log(0.659) \approx 0.417$$

5. Summary Table

Type	**Label Format**	**Used In**
Categorical Cross-Entropy	One-hot (e.g., $[0, 1, 0]$)	TensorFlow, Manual PyTorch
Sparse Categorical Cross-Entropy	Integer index (e.g., $y = 1$)	PyTorch (`CrossEntropyLoss`)

6. How Does the Machine Know the True Label?

In supervised learning, each training sample comes with a known label. For example, in image classification:

Image (Input)	True Label
Cat image	0 (Cat)
Dog image	1 (Dog)
Rabbit image	2 (Rabbit)

These labels are provided during training so the loss function can compare predictions with the correct class. During inference, the model only receives the input and must predict the label.

7. Cross-Entropy vs. KL Divergence

While `CrossEntropyLoss` is mathematically related to KL divergence, it does not use it directly.

KL Divergence Definition:

Let P be the true distribution and Q the predicted distribution:

$$D_{\text{KL}}(P \parallel Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right) = -\sum_i P(i) \log Q(i) + \sum_i P(i) \log P(i)$$

$$D_{\text{KL}}(P \parallel Q) = \text{CrossEntropy}(P, Q) - \text{Entropy}(P)$$

In Supervised Learning:

When P is a one-hot vector, the entropy of P is zero. So:

$$D_{\mathrm{KL}}(P \parallel Q) = \mathrm{CrossEntropy}(P, Q)$$

But PyTorch's `CrossEntropyLoss()` is not computing KL divergence explicitly — it only implements the cross-entropy term:

$$\mathcal{L} = -z_y + \log\left(\sum_{j=0}^{C-1} e^{z_j}\right)$$

Summary:

- **KL divergence is not used directly** in `CrossEntropyLoss`.
- `CrossEntropyLoss` is sufficient when the true labels are deterministic (e.g., one-hot or class indices).