Loss Functions

Loss Function	Use Case	Pros	Cons
Mean Squared Error (MSE)	Regression	Simple to compute and differentiate.	Sensitive to outliers, can lead to slow convergence.
Mean Absolute Error (MAE)	Regression	Robust to outliers.	Less smooth gradients compared to MSE, can be slower to converge.
Huber Loss	Regression (robust to outliers)	Balances sensitivity and robustness to outliers.	Requires tuning of the hyperparameter δ\deltaδ.
Cross-Entropy Loss	Classification (binary and multiclass)	Effective for classification tasks, especially with softmax output.	Can be sensitive to class imbalance.
Binary Cross-Entropy	Binary Classification	Suitable for binary classification, handles probabilities well.	Can suffer from vanishing gradients for extreme predictions.
Categorical Cross-Entropy	Multiclass Classification	Standard for multiclass classification with one-hot encoded labels.	Assumes mutually exclusive classes, not suitable for multi-label classification.
Sparse Categorical Cross-Entropy	Multiclass Classification with integer labels	Efficient for large number of classes, avoids one-hot encoding.	Similar issues as categorical cross-entropy with class imbalance.
Hinge Loss	Support Vector Machines (SVMs)	Good for maximum margin classifiers,	Not differentiable at the margin, less commonly used in deep learning.

		promotes clear class separation.	
Poisson Loss	Poisson regression and count data	Suitable for modeling count data, consistent with Poisson-distributed targets.	Assumes Poisson distribution, not suitable for other types of data.
Cosine Proximity	Similarity Learning	Effective for tasks focusing on similarity (e.g., embeddings).	May not work well for tasks requiring absolute value prediction.
Focal Loss	Object detection and classification (imbalanced data)	Addresses class imbalance by focusing on hard examples.	Introduces an additional hyperparameter γ\gammaγ, requires tuning.

• Mean Squared Error (MSE): Measures the average squared differences between predicted and actual values. Widely used in regression tasks.

- PyTorch Syntax: `torch.nn.MSELoss()`

• Mean Absolute Error (MAE): Measures the average absolute differences between predicted and actual values. Less sensitive to outliers compared to MSE.

- PyTorch Syntax: `torch.nn.L1Loss()`

• Huber Loss: A combination of MSE and MAE, providing robustness to outliers and smooth gradients.

- PyTorch Syntax: `torch.nn.SmoothL1Loss()`

• Cross-Entropy Loss: Measures the difference between two probability distributions, commonly used for classification tasks.

- PyTorch Syntax: `torch.nn.CrossEntropyLoss()`

• Binary Cross-Entropy: A special case of cross-entropy for binary classification problems.

- PyTorch Syntax: `torch.nn.BCELoss()`

• Categorical Cross-Entropy: Used for multiclass classification where each output class is one-hot encoded.

- PyTorch Syntax: `torch.nn.CrossEntropyLoss()`

• Sparse Categorical Cross-Entropy: Similar to categorical cross-entropy but uses integer labels for classes.

- PyTorch Syntax: `torch.nn.CrossEntropyLoss() (with integer labels)`

• Hinge Loss: Used primarily for training Support Vector Machines (SVMs), promoting a large margin between classes.

- PyTorch Syntax: `torch.nn.HingeEmbeddingLoss()`

• Poisson Loss: Suitable for count data and Poisson regression, assuming targets follow a Poisson distribution.

- PyTorch Syntax: `torch.nn.PoissonNLLLoss()`

• Cosine Proximity: Measures the cosine similarity between predicted and actual vectors, useful in similarity learning tasks.

- PyTorch Syntax: `torch.nn.CosineEmbeddingLoss()`

• Focal Loss: Modifies cross-entropy to focus on hard-to-classify examples, useful for imbalanced classification tasks. This loss function is not built-in to PyTorch but can be implemented manually. Here is an example implementation: