datacamp

COURSE GLOSSARY

Introduction to Deep Learning with PyTorch -learning objectives already created

Activation function: A non-linear function applied to a layer's outputs (pre-activations) to introduce non-linearity, enabling networks to learn complex relationships beyond linear mappings

Backpropagation: The algorithm for computing gradients of the loss with respect to model parameters by propagating error derivatives backward through the network

Cross-entropy loss: A classification loss that measures the difference between the predicted class probability distribution (often raw scores) and the true class labels, penalizing incorrect confident predictions

DataLoader: A PyTorch utility that loads data from a dataset in configurable mini-batches, optionally shuffling and parallelizing data loading for efficient training

Deep learning: A subset of machine learning that trains multi-layered neural networks to automatically learn hierarchical patterns from large amounts of data

Forward pass: The process of passing input data through a neural network layer by layer to compute the final outputs or predictions using the current parameters

Gradient: The derivative of the loss with respect to a model parameter, indicating the direction and magnitude to change that parameter to reduce the loss

Hallucination: When a model produces confident but incorrect or fabricated information, often due to gaps or biases in its training data or reasoning process

Hallucination: When a model produces confident but incorrect or fabricated information, often due to gaps or biases in its training data or reasoning process

Learning rate: A hyperparameter that scales the gradient step size during optimization and determines how quickly or slowly model parameters are updated

Linear layer (nn.Linear): A neural network layer that performs an affine transformation (y = xWˆT + b), mapping inputs of a given size to outputs of a specified size and containing learnable weights and biases

Loss function: A function that quantifies the difference between model predictions and ground-truth targets, producing a scalar value that training seeks to minimize

Mean squared error (MSE) loss: A regression loss equal to the average squared difference between predicted and actual continuous target values, commonly used for continuous outputs

Momentum: An optimizer hyperparameter that accumulates a velocity vector from past gradients to accelerate convergence and help overcome local minima or noisy gradients

Neural network: A computational model composed of interconnected layers of neurons that transforms inputs through learned weights and biases to produce outputs

Neuron: A single computational unit in a neural network that computes a weighted sum of its inputs plus a bias and typically applies an activation function

Optimizer: An algorithm that updates model parameters using computed gradients and hyperparameters (like learning rate and momentum) to minimize the loss during training

Overfitting: A modeling problem where a trained model performs well on training data but poorly on unseen data because it has memorized noise or idiosyncrasies rather than learning generalizable patterns

PyTorch: An open-source deep learning framework for Python that provides tensors, automatic differentiation, neural network building blocks, and optimization utilities

ReLU (Rectified Linear Unit): A widely used hidden-layer activation function that outputs zero for negative inputs and the input itself for positive inputs, helping mitigate vanishing gradients

Sigmoid: An activation function that maps real-valued inputs to the range (0, 1), commonly used for binary classification outputs because it produces values interpretable as probabilities

Softmax: An activation function that converts a vector of real-valued scores into a probability distribution over multiple classes where outputs are non-negative and sum to one

Stochastic Gradient Descent (SGD): An optimizer that updates parameters by taking steps proportional to the negative gradient estimated from mini-batches of data, optionally using momentum to smooth updates

Tensor: A multi-dimensional array (generalization of scalars, vectors, and matrices) that stores data and gradients and is the primary data structure used in PyTorch

TensorDataset: A PyTorch dataset wrapper that groups tensors (e.g., features and labels) into a dataset object so that they can be indexed and used with a DataLoader