Setting up your Machine Learning Application

Train / Dev / Test sets: Splitting data into training, development (validation), and test sets to optimize model performance and prevent overfitting.

Bias / Variance: Understanding the trade-off between bias (error from incorrect assumptions) and variance (error from sensitivity to small fluctuations in training data).

Basic Recipe for Machine Learning: A structured approach to improving model performance, including data preprocessing, feature engineering, and hyperparameter tuning.

Regularizing your Neural Network

Regularization: Techniques used to prevent overfitting by adding constraints to the model parameters, such as L1 or L2 regularization.

Why Regularization Reduces Overfitting?: Regularization prevents the model from fitting noise in the training data by penalizing overly complex solutions.

Dropout Regularization: A technique where random neurons are ignored during training, helping prevent reliance on specific neurons and improving generalization.

Understanding Dropout: Dropout is applied during training to reduce co-adaptation between neurons, increasing the robustness of the model.

Other Regularization Methods: Additional techniques such as data augmentation, early stopping, and batch normalization that help improve generalization.

Setting Up your Optimization Problem

Normalizing Inputs: Scaling input features to a common range to improve training stability and convergence speed.

Vanishing / Exploding Gradients: Issues where gradients become too small (vanishing) or too large (exploding), making training deep networks difficult.

Weight Initialization for Deep Networks: Choosing appropriate initial values for network weights to improve training efficiency and prevent vanishing/exploding gradients.

Numerical Approximation of Gradients: Estimating gradients using numerical methods to validate analytical gradient calculations.

Gradient Checking: A technique to verify the correctness of backpropagation implementations by comparing computed gradients with numerical approximations.

Gradient Checking Implementation Notes: Best practices for implementing gradient checking, such as using small epsilon values and ensuring consistency between computed and estimated gradients.

Optimization Algorithms

Mini-batch Gradient Descent: An optimization technique that updates model parameters using small batches of data rather than the entire dataset, balancing efficiency and stability.

Understanding Mini-batch Gradient Descent: A deeper look at how mini-batch gradient descent combines the advantages of both batch and stochastic gradient descent to improve training performance.

Exponentially Weighted Averages: A method to smooth noisy data by giving exponentially decreasing weights to past values, often used in optimization algorithms.

Understanding Exponentially Weighted Averages: Explaining how exponentially weighted averages help track trends in data, particularly in momentum-based optimization.

Bias Correction in Exponentially Weighted Averages: A technique to correct initial bias when using exponentially weighted averages, ensuring accurate trend estimation.

Gradient Descent with Momentum: An optimization technique that accelerates gradient descent by incorporating past gradients, reducing oscillations and improving convergence speed.

RMSprop: An adaptive learning rate optimization algorithm that adjusts the learning rate dynamically based on recent gradient magnitudes, preventing large fluctuations.

Adam Optimization Algorithm: A widely used optimization algorithm that combines momentum and adaptive learning rates to improve training speed and convergence.

Learning Rate Decay: A technique that gradually decreases the learning rate during training to fine-tune model updates and prevent overshooting the optimal solution.

The Problem of Local Optima: An issue in optimization where a model gets stuck in a suboptimal solution instead of finding the global minimum, often mitigated by advanced optimization techniques.

Hyperparameter Tuning

Tuning Process: The process of adjusting hyperparameters to optimize a machine learning model’s performance.

Using an Appropriate Scale to Pick Hyperparameters: Selecting the right scale (linear, logarithmic, etc.) for hyperparameters to ensure effective tuning.

Hyperparameters Tuning in Practice: Pandas vs. Caviar: A comparison of different approaches to hyperparameter tuning, balancing computational cost and performance.

Batch Normalization

Normalizing Activations in a Network: A technique that standardizes inputs to each layer to improve training stability and speed.

Fitting Batch Norm into a Neural Network: Integrating batch normalization layers into a neural network to stabilize learning and mitigate internal covariate shift.

Why Does Batch Norm Work?: Explaining how batch normalization reduces the dependence on initialization and allows for higher learning rates.

Batch Norm at Test Time: Adjustments made during inference to maintain consistency in normalized activations.

Multi-class Classification

Softmax Regression: A generalization of logistic regression that handles multiple classes by converting scores into probabilities.

Training a Softmax Classifier: The process of training a neural network to classify inputs into multiple categories using the softmax function.

Introduction to Programming Frameworks

Deep Learning Frameworks: Software libraries such as TensorFlow and PyTorch that provide tools to build and train deep learning models.

TensorFlow: An open-source deep learning framework that allows users to build and train machine learning models efficiently.

Learn about Gradient Tape and More: A feature in TensorFlow that enables automatic differentiation to compute gradients during training.