LAB 2 - HYPERPARAMETER TUNING AND REGULARIZATION

Dr. Pandiyaraju V
Shravan Venkatraman

July 26, 2024

TABLE OF CONTENTS

01 INTRODUCTION

02 MLP RECAP

03 MINI BATCH GRADIENT DESCENT

04 HYPERPARAMETER TUNING

05 TUNING TECHNIQUES

01 INTRODUCTION

02 MLP RECAP

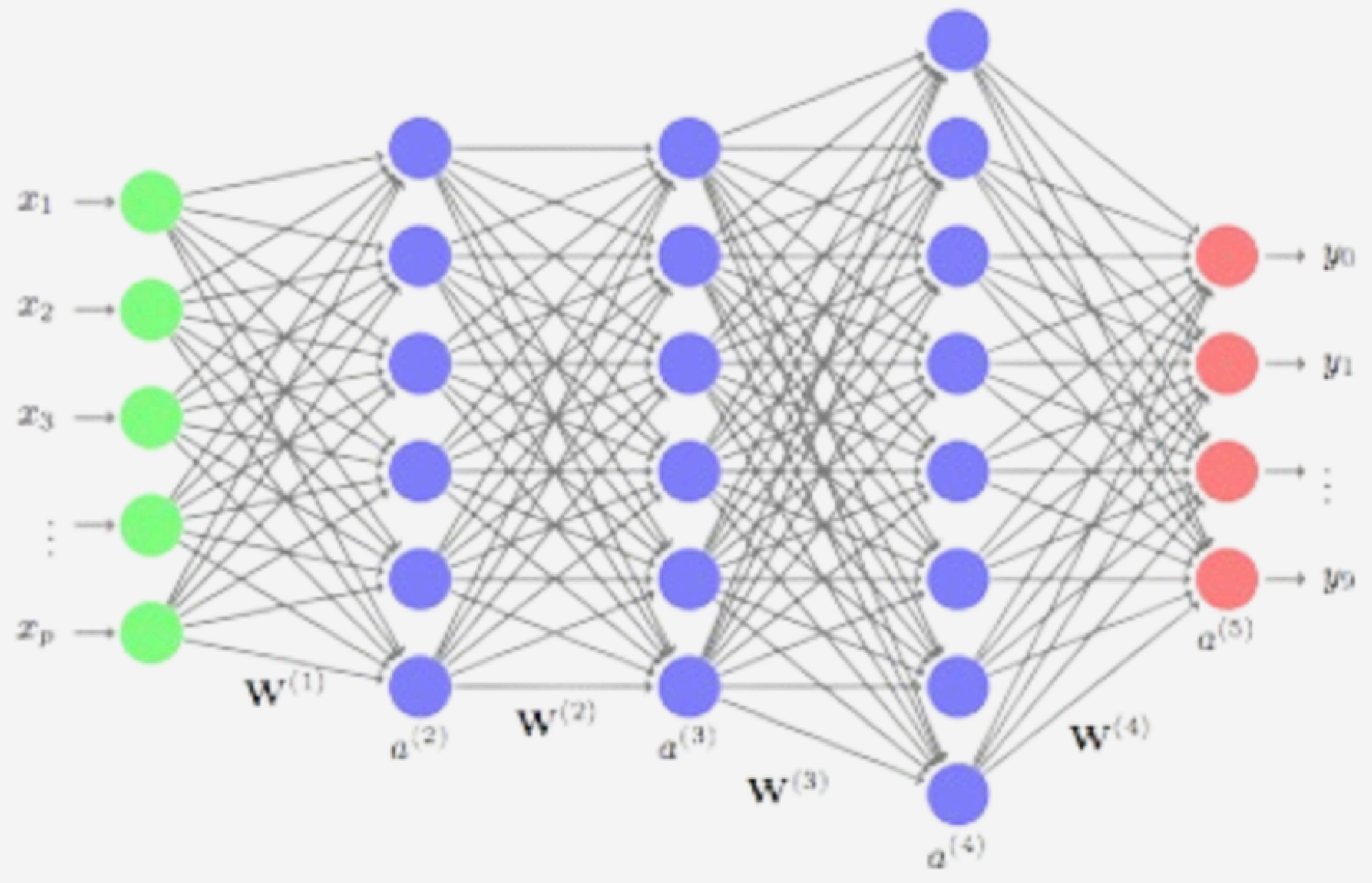
03 MINI BATCH GRADIENT DESCENT

04 HYPERPARAMETER TUNING

05 TUNING TECHNIQUES

MLP RECAP

- **Architecture:** Consists of input layer, hidden layers, and output layer.
- **Activation Functions:** Common functions include ReLU, Sigmoid, and Tanh.
- **Backpropagation:** Algorithm for training MLPs, involves calculating gradients and updating weights.



INTRODUCTION

- **Mini-Batch Gradient Descent:** An optimization technique that splits the training dataset into small batches, allowing the model to update its weights more frequently and improve convergence.
- **Hyperparameters:** Parameters whose values are set before the learning process begins.
- **Regularization:** Techniques to prevent overfitting by adding additional information to constrain the model.
- **Importance:** Improves model performance and generalization.

MINI-BATCH GRADIENT DESCENT

- **Gradient Descent:** Optimization algorithm for finding the minimum of a function.
- **Mini-batch Gradient Descent (MBGD):** Updates model parameters using gradients calculated from a small subset of the training dataset, known as a mini-batch.
- **Why MBGD:** Provides balance between computation efficiency and model performance.
- **Batch size selection:** Smaller sizes introduce noise aiding escape from local minima, while larger sizes offer more stable gradient estimates.

HYPERPARAMETER TUNING

- **Learning Rate:** Controls how much to adjust the model in response to the estimated error each time the model weights are updated.
- **Batch Size:** Number of samples processed before the model is updated.
- **Number of Epochs:** The number of complete passes through the training dataset.
- **Activation Functions:** Determines the output of a neuron given an input or set of inputs.

TUNING TECHNIQUES

- **Grid Search Cross-Validation (CV):** Systematically explores a predefined subset of hyperparameters by exhaustively training models for every combination of hyperparameters.
- **Random Search CV:** Selects hyperparameters randomly from a specified distribution or range, training a fixed number of models.
- **Learning Rate Scheduling:** Dynamically adjusts the learning rate during training to improve convergence.
- **Early Stopping:** Stops training when a monitored metric has stopped improving, preventing overfitting.

REGULARIZATION

- **Fitting**: Aims to reduce errors by fitting the function appropriately on the given training set and avoiding overfitting, thereby improving model generalization.
- **Penalty Term**: Prevents overfitting by adding a penalty term to the loss function, discouraging the model from assigning too much importance to individual features or coefficients.
- **Consistency**: Promotes consistent model performance across different datasets, reducing the risk of dramatic performance changes when encountering new data.

REGULARIZATION TECHNIQUES

- **Dropout:** Randomly disables some percent of neurons in each layer during training to extract robust features and prevent overfitting. Not applied during inference phase.

- **L1/Lasso Regularization:** Simplifies models and improves interpretability by reducing coefficients of less important features to zero.

- **L2 Regularization:** Penalizes large coefficients and constrains their magnitudes, preventing models from becoming overly complex.

https://colab.research.google.com/drive/14CKItbXtE3FprIXAEaDP38H6LMwgOx3t?usp=sharing

DROPOUT

- During training, dropout randomly sets a fraction of the neurons' output to zero in each layer, excluding the output layer.
- This "dropout" happens independently for each neuron on each forward pass.
- Prevents neurons from co-adapting too much, which promotes the development of more robust and diverse feature representations.
- Common dropout values - typically between 0.2 to 0.5

L1 REGULARIZATION (LASSO)

- L1 regularization adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function.

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} |W_i|$$

  where λ is the regularization parameter and Wi are the model weights

- Encourages sparsity in the model weights, leading to feature selection by shrinking less important features' coefficients to zero.

- Helps in reducing overfitting, especially when the number of features is large.

L2 REGULARIZATION (RIDGE)

- L2 regularization involves adding a penalty equal to the squared magnitude of coefficients to the loss function.

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} |W_i|$$

  where λ is the regularization parameter and Wi are the model weights

- Unlike L1, L2 regularization shrinks all coefficients by the same factor but doesn’t set any coefficients to zero.

- Helps in reducing overfitting by discouraging large weights, leading to a smoother and more generalized model.

HANDS ON

MBGD and Hyperparameter Tuning:
https://colab.research.google.com/drive/14CKItbXtE3FprIXAEaDP38H6LMwgOx3t?usp=sharing&authuser=2#scrollTo=YbLarQ5d6yXb

Regularization:
https://colab.research.google.com/drive/1sarHQQiqBN2clJ2QEU4-hFfAcYF7Pxqi?usp=sharing

CHALLENGE TASK

Task 1: Image Classification with CIFAR-10 Dataset
Train two artificial neural networks (ANNs) on the CIFAR-10 dataset using mini-batch gradient descent. Apply hyperparameter tuning to both models, using different regularization techniques for each. Evaluate the model's performance with visualizations of loss and accuracy, and display with reasoning as to which model performed better.

Task 2: Predicting House Prices with the Boston Housing Dataset
Implement an artificial neural network (ANN) for regression on the Boston Housing dataset, applying minibatch gradient descent, hyperparameter tuning, and various regularization techniques. Assess the model using Mean Squared Error and visualize training progress.
Dataset link: https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset

You can load the datasets via TensorFlow/Kaggle API.

REFERENCES

- https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/
- https://www.geeksforgeeks.org/ml-mini-batch-gradient-descent-with-python/
- https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/
- https://www.geeksforgeeks.org/regularization-in-machine-learning/
- https://theaisummer.com/regularization/
- https://medium.com/data-science-365/how-to-apply-l1-and-l2-regularization-techniques-to-keras-models-da6249d8a469