Brno University of Technology
Faculty of Information Technology

Soft Computing Project

02 - Demonstrating Learning with Backpropagation: Basic Algorithm Enhanced with Adam Optimizer

Author:
Elena Carasec (xcaras00)

Brno, Czech Republic
Monday 27th November, 2023

1 Solved Problem

1.1 Project's Aim

The problem that this project addresses is to demonstrate learning with backpropagation algorithm with an optimiser. In this context, the Adam optimizer has been selected due to its widespread adoption, known for delivering precise results, rapid computation, and a manageable set of parameters for fine-tuning [1].

1.2 Machine Learning Task

The objective of the Neural Network is to address a regression task, specifically predicting house prices.

1.3 Dataset

The dataset employed for this regression task is the "Boston House Prices" dataset obtained from Kaggle [2]. Comprising 506 samples, the dataset encompasses 10 feature columns and one target column representing the predicted price. The data have been scaled using the MinMaxScaler from scikit-learn.

1.4 Model Architecture

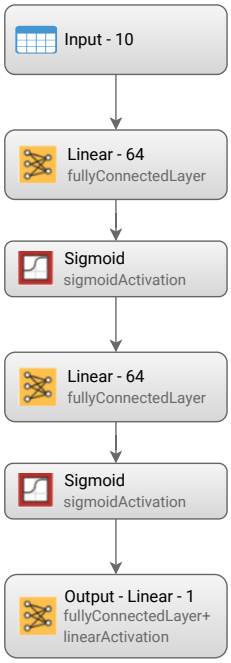

Figure 1: Model Architecture.

The architecture of the neural network is illustrated in Figure 1. It begins with an Input Layer featuring 10 neurons, followed by two hidden fully-connected Linear layers with Sigmoid activation functions. The output layer consists of a Linear layer with a single neuron utilizing a Linear (Identity) activation function. The Mean Squared Error loss function is employed to compute the loss.

1.5 Training and Testing

To train the model the Backpropagation algorithm is used, which includes the following steps:

1. **Initialization**: Initialize the neural network with random weights.
2. **Forward Pass**: Input data is fed forward through the network layer by layer. Each layer performs a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.
3. **Compute Loss**: Compare the predicted output (result of the forward pass) with the actual target output using a loss function.
4. **Backward Pass**: The algorithm starts from the output layer and works backward to calculate the gradient of the loss with respect to each weight and bias in the network. This is done using the chain rule from calculus.
5. **Gradient Descent**: The gradients calculated during the backward pass indicate the direction and magnitude of the steepest increase in the loss. The weights are adjusted in the opposite direction to reduce the loss.
6. **Repeat**: Steps 2 to 5 are repeated for multiple epochs. This allows the model to gradually learn and improve its performance on the training data.

Forward Pass equations: X - input, W - weights, B - Bias.

- Linear layer: $Y = XW + B$
- Sigmoid activation function: $Y = \frac{1}{1+e^{-X}}$
- Linear (Identity) activation function: $Y = X$
- Mean Squared Error: $MSE = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Backward Pass equations:

- Linear layer: $\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \mathbf{W}^T \frac{\partial \mathcal{L}}{\partial \mathbf{Y}}$
  $\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} \mathbf{X}^T$
  $\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{Y}_i}$

- Sigmoid activation function: $\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} \odot \sigma(\mathbf{X}) \odot (1 - \sigma(\mathbf{X}))$, where $\odot$ denotes element-wise multiplication.
- Linear (Identity) activation function: $\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = 1$
- MSE: $\frac{\partial MSE}{\partial \hat{\mathbf{Y}}} = \frac{1}{n}(\hat{\mathbf{Y}} - \mathbf{Y})$

Adaptive Movement Estimation algorithm, or **Adam**, is an optimization algorithm, which combines the ideas from Momentum and RMSProp optimizers. The Adam optimization algorithm maintains two moving averages: the first moment (mean) and the second moment (uncentered variance). These moving averages are calculated during each iteration using exponential decay. It updates the moving averages and computes the biased-corrected estimates. Adam has hyperparameters such as the learning rate α, decay rates for the first and second moment estimates β_1 and β_2, and a small constant ϵ to prevent division by zero. Adam is known for its robustness and effectiveness in a wide range of settings.

The implementation of backpropagation and Adam optimizer was inspired by An Intuitive Guide to Back Propagation Algorithm with Example and Code Adam Optimization Algorithm From Scratch

2 User's Guide

The application is written in Python3 and requires the following libraries:

- `PyQt5` - GUI for the application.
- `matplotlib` - Plots the learning curve.
- `numpy` - Computaions for backpropagation.
- `pandas` - Retrieves the data and converts them to a DataFrame.
- `scikit_learn` - Splits the dataset into training and testing. Scales the data.

2.1 Application Launching

The application is easily launched using the script: `./launch.sh`. This script installs all the necessary requirements and starts the application from `main_gui.py`. The remaining files consist of scripts handling the application's logic, including data preparation, model training and testing, and the UI form.

2.2 User's Manual

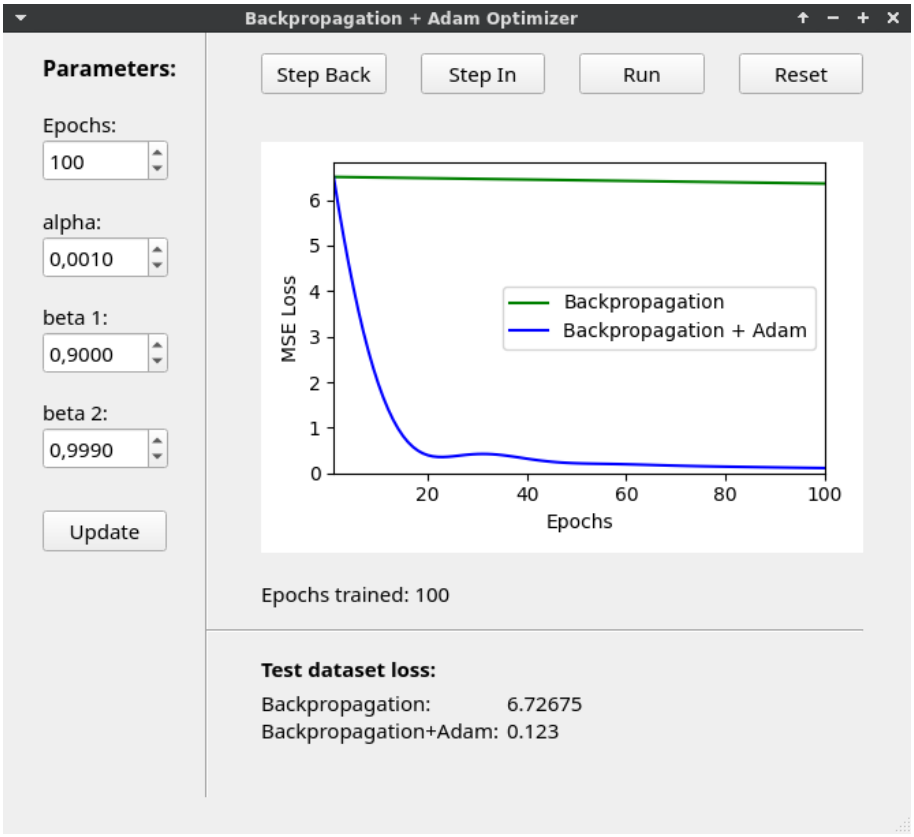

Figure 2: Application.

The application illustrates the differences in learning curves between a model using Backpropagation without any optimizers and a model using Backpropagation with the Adam optimizer. Upon launching, the application appears as shown in Figure 2.

To update the parameters, navigate to the **"Parameters"** section. Adjust the *"alpha"* parameter for the learning rate, which impacts both networks. The *"beta 1"* and *"beta 2"* parameters affect only the network with the Adam optimizer. After selecting the desired parameters, click **"Update"** to reinitialize the models.

To train the models for the specified number of epochs, click **"Run"**.

For training the models for only one epoch, click **"Step In"**.

To revert to the state of the models from the previous epoch, click **"Step Back"**. The graph displaying the learning curve is automatically updated.

The **Test dataset loss** shows the Mean Squared Error for the testing dataset using the model from the last epoch.

To reinitialize the models, click **"Reset"**. Note that the weights are randomly initialized, resulting in different learning curves after each reinitialization.

References

[1] Ayush Gupta. *A comprehensive guide on Optimizers in deep learning*. Sept. 2023. URL: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=The%20results%20of%20the%20Adam,for%20most%20of%20the%20applications..

[2] Manimala. *Boston House Prices*. Aug. 2017. URL: https://www.kaggle.com/datasets/vikrishnan/boston-house-prices/.