Artificial Neural Networks

Teodora Taleska

1. Classification ANN

1.1 Gradient Verification: Procedure and Results

Gradient verification was implemented to check whether the computed gradients during backpropagation were correct. We applied a numerical gradient checking technique, following these steps: (1) Small perturbations were added to each weight individually, (2) the cost function was evaluated at these perturbed weights, and (3) the numerical gradient was estimated using the difference quotient:

$$\frac{\partial J}{\partial \theta} \approx \frac{J(\theta+\varepsilon)-J(\theta-\varepsilon)}{2\varepsilon}$$

where $\varepsilon = 10^{-5}$. (4) This numerical gradient was then compared to the analytically computed gradient from the backpropagation method. The comparison was made using the following formula:

$$\text{difference} = \frac{\|\text{analytical} - \text{numerical}\|}{\|\text{analytical}\| + \|\text{numerical}\|}$$

where a difference smaller than 10^{-7} was considered acceptable.

In our case, the computed difference was below 10^{-7}, confirming that the backpropagation gradients were implemented correctly and that the model updates during training are valid.

1.2 Training Setup and Network Parameters

To successfully fit the *squares* and *doughnut* datasets using the implemented ANN, a specific set of parameters and configurations were chosen. The datasets were split into training and testing subsets, where 70% of the data was used for training and 30% for testing.

The *Squares.tab* dataset was used to test the regression model, while the *Doughnut.tab* dataset was used for classification, even though the data structure was the same. Moreover, the parameters for both models were nearly identical. The classification network configuration was as follows:

- **Network Architecture:** Two hidden layers with 10 neurons each.
- **Activation Functions:** ReLU was used in both hidden layers.
- **Regularization:** L2 regularization with $\lambda = 0.01$.
- **Output Layer:** The output layer utilized an integrated softmax activation function to produce class probabilities suitable for multiclass classification.
- **Evaluation Metric:** Accuracy score was used, measuring the proportion of correctly classified samples.

The training was performed with the following settings for both datasets:

- **Learning Rate:** A learning rate of 0.1 was used. This relatively high learning rate allowed faster convergence without causing instability, as the networks were relatively shallow and well-regularized.
- **Number of Epochs:** Models were trained for $100,000$ epochs to ensure complete convergence, especially important for verifying correct implementation and ensuring stable training without premature stopping.
- **Weight Initialization:** He initialization was employed to initialize the network weights. This method was chosen because it is well-suited for ReLU activations, as it maintains the variance of activations across layers, preventing vanishing or exploding activations, and thereby allowing effective gradient-based optimization.

2. Regression ANN

2.1 Differences in Model Structure

The regression and classification neural networks were both based on a common parent class, *ANNBase*. This base class defined core functionality such as: (1) weight initialization using different strategies (*small* or *He*), (2) forward propagation with bias handling, (3) activation functions and their derivatives (supporting *sigmoid* and *ReLU*), and (4) gradient checking.

The child classes, *ANNClassification* and *ANNRegression*, override specific components to tailor the network behavior to their respective tasks.

For the cost function, *ANNClassification* uses the cross-entropy loss, which is appropriate for classification tasks involving probability outputs and categorical labels. In contrast, *ANNRegression* employs the mean squared error (MSE), which is suited for predicting continuous real-valued targets.

Regarding the output activation, *ANNClassification* applies a softmax function at the output layer to transform the raw outputs into probabilities across the classes. On the other hand, *ANNRegression* leaves the output layer linear, meaning no activation is applied, allowing the network to produce unrestricted real-valued predictions.

In the backward pass, the computation of delta values and gradients differs between the two cases due to the nature of the cost functions and output transformations. In classification, the delta combines the softmax derivative with cross-entropy, whereas in regression, it comes directly from the difference between predicted and actual values using MSE.

2.2 Comparison with Existing Implementations

To validate the models, we compared them against *scikit-learn*'s *MLPRegressor* and *MLPClassifier*.

Both our custom ANNs and *scikit-learn*'s models were trained under identical conditions on the same dataset splits, ensuring that any differences came from the model implementations and not from variations in data or hyperparameters.

For classification, we evaluated performance by comparing the accuracy scores on the test set and calculating the percentage of matching predictions. If more than 90% of the predictions were the same, it was considered a pass. The results showed that the models predictions were identical, confirming that our implementation was correct.

For regression, evaluation focused on the mean squared error (MSE) and the absolute differences between predictions. A pass was declared if the MSE difference was less than 5, and more than 90% of predictions differed by less than 0.5 units. The results indicated that the custom ANN and *scikit-learn*'s MLPRegressor produced similar predictions.

3. Classification of FTIR Spectral Data of Human Tissues using ANN

The goal was to predict tissue classes for a given part of an image based on FTIR spectral data. To simplify the problem, only spectral information was used, treating each pixel as an independent sample without considering spatial context.

3.1 Data Preprocessing

First, the dataset was flattened such that each pixel's spectral information became a single feature vector. The corresponding class labels were also flattened. Samples without annotations (class $= -1$) were filtered out to ensure only valid labeled examples were used for training.

The spectral features were standardized using a *StandardScaler* to have zero mean and unit variance, improving convergence during training.

The processed data was then split into a training set (90%) and a testing set (10%) using a random split.

3.2 Model Architecture

For the classification task, we implemented a fully connected Artificial Neural Network (ANN) using PyTorch. The architecture consisted of the following layers:

- **Input layer:** Dimension equal to the number of spectral features per pixel.
- **First hidden layer:** Fully connected layer with 256 neurons, followed by Batch Normalization and the Tanh activation function.
- **Second hidden layer:** Fully connected layer with 128 neurons, followed by Batch Normalization and the ReLU activation function.
- **Third hidden layer:** Fully connected layer with 64 neurons, followed by the Tanh activation function.
- **Dropout layer:** Dropout with a probability of 0.3 to reduce overfitting.
- **Output layer:** Fully connected layer with 6 neurons (corresponding to the six tissue classes).

The forward pass can be summarized as:

$$\begin{aligned}&\text{Input} \to \text{Linear}(256) \to \text{BatchNorm} \to \text{Tanh}\\&\quad \to \text{Linear}(128) \to \text{BatchNorm} \to \text{ReLU} \to \text{Linear}(64)\\&\quad \to \text{Tanh} \to \text{Dropout} \to \text{Linear}(6)\end{aligned}$$

3.3 Training Details

The model was trained using the AdamW optimizer, with an initial learning rate of 0.001 and a small weight decay of 10^{-5} for regularization.

The loss function used was *CrossEntropyLoss*, suitable for multi-class classification tasks.

The network was trained for 50 epochs, balancing training time and model performance.

3.4 Evaluation

After training, the model was evaluated on the held-out test set, achieving a **test accuracy of 89.22%**. This shows that the model successfully generalized to unseen spectral data while treating each pixel independently.

3.5 Future Work

Future work could involve experimenting with CNNs, hybrid architectures combining spectral and spatial processing, or using data augmentation strategies to further boost performance.