CS303 Project2 Report

- StudentId: 12110517
- Name: 钟志源

Subtask 1

1. Introduction

We are required to independently train a model using the provided training set. This model should then be utilized to predict labels for the test set. The predicted labels are to be saved in `classification_results.pkl` file. The baseline model is Softmax Regression.

2. Methodology

Algorithm/model design.

- Baseline model: Softmax Regression:

SoftmaxRegression Class:

```
- Constructor:
    - num_classes: Number of classes for classification
    - learning_rate: Learning rate for gradient descent (default = 0.01)
    - num_iterations: Number of training iterations (default = 100)
    - random_seed: Random seed for reproducibility (default = None)

- fit method:
    - X_train: Training feature data
    - y_train: Training labels
    - X_val: Validation feature data (optional)
    - y_val: Validation labels (optional)
    - decay_rate: Rate of learning rate decay (default = 0.9995)

    - Initialize weights randomly
    - Initialize lists to store training and validation losses, and
accuracies
    - Loop over iterations:
        - Calculate logits and softmax probabilities
        - Compute cross-entropy loss
        - Compute gradient and update weights using gradient descent
        - Decay learning rate after specific iterations
        - Calculate training accuracy and append to list
        - If validation data provided:
            - Calculate validation loss
            - Calculate validation accuracy and append to list
    - Return lists of training and validation losses and accuracies
```

```
- predict method:
    - X: Feature data for prediction

    - Calculate logits
    - Return predicted class labels
```

- Improved model 1: Softmax Regression with decay learning rate: During gradient descent, the learning rate is decayed by a factor of *0.9995* after every *100* iterations. The model is trained with a high learning rate of 0.15 and then gradually decreases to around 0.01. This helps the model to converge faster and also prevents it from overshooting the minima. Finally fine-tune the weights with a lower learning rate.
- Improved model 2: The network architecture comprises a simple Multi-layer Perceptron (MLP) with two fully connected layers:
    - Input to First Hidden Layer: Input features are linearly transformed using the first weight matrix and bias, followed by a ReLU activation.
    - First Hidden Layer to Output: Activations from the first hidden layer undergo linear transformation using the second weight matrix and bias, followed by a softmax function for multi-class classification, yielding class probability distributions.

NeuralNetwork Class:

```
- initialize_weights method:
    - input_size: Size of input features
    - Initialize weights and biases for layers

- build_model method:
    - input_size: Size of input features
    - Call initialize_weights method

- forward_pass method:
    - X: Input data
    - training: Flag indicating training/testing mode
    - Calculate forward pass through the network

- backward_pass method:
    - X: Input data
    - y: Actual labels
    - a1, a2: Activation outputs from forward pass
    - Compute gradients and update weights

- fit method:
    - X_train, y_train: Training data and labels
    - X_val, y_val: Validation data and labels (optional)
    - Train the neural network and monitor performance

    - Initialize model
```

```
    - Loop over epochs:
        - Loop over batches:
            - Forward pass
            - Compute loss and accuracy
            - Backward pass
        - Calculate average loss and accuracy for the epoch
        - Perform validation if validation data provided
        - Print and store epoch metrics
    - Return training and validation loss and accuracy histories
```

Analysis

The deciding factor of its performance is the learning rate. If the learning rate is too small, the model will converge slowly. If the learning rate is too large, the model will not converge. The learning rate should be adjusted according to the specific problem. For network architecture, the number of layers and the number of neurons in each layer will affect the performance of the model. The more layers and neurons, the more complex the model, and the better the performance, but too many layers and neurons will lead to overfitting.

3. Experiments

1. Metrics: classification accuracy.
2. Experimental results:
    - Baseline model: Softmax Regression
        - Training accuracy: 0.509
        - Validation accuracy: 0.505
    - Improved model 1: Softmax Regression with decay learning rate
        - Training accuracy: 0.512
        - Validation accuracy: 0.511
    - Improved model 2: MLP
        - Training accuracy: 0.572
        - Validation accuracy: 0.515

4. Further thoughts

Using different optimizer like Adam, SGD with momentum, etc. can improve the performance of the model.

Subtask 2

1. Introduction

This task is to implement a nearest neighbor search (NNS) model to find the most similar images in the image repository for each image in the test set. This model should then be utilized to respond to image queries from the test set. For each image in the test set, should find 5 similar images in the image repository. The baseline model is KNN.

2. Methodology

Algorithm/model design.

- baseline: KNN using Euclidean distance.
- improved model: KNN using Manhattan distance.

Analysis

Manhattan distance is less sensitive to outliers compared to Euclidean distance. Since it calculates distances along orthogonal axes, outliers have less influence, making it more robust in the presence of outliers or noise in the data.

In some datasets, certain features might have more significance than others in determining similarity. Manhattan distance considers each feature's linear contribution equally, making it potentially more suitable when all features should be weighted equally.

3. Experiments

1. Metrics: For each test image, suppose you submit n similar images(In this subtask, n = 5) and there are m images that are considered to be similar by the evaluation process, the accuracy should be m/n×100%. The test accuracy is the average accuracy of the whole test set.
2. Experimental results: 0.05

4. Further thoughts

Additional distance metrics that could be explored for capturing image similarities: e.g. Cosine Similarity: Measures the cosine of the angle between two vectors, disregarding their magnitude. Effective when focusing on the orientation of features rather than their magnitudes, suitable for text-based or high-dimensional data representations.

Subtask 3

1. Introduction

Write an algorithm to select no more than 30 features from the image features of the classification validation set. The task is to generate a mask code that only preserves the most important features.

2. Methodology

Algorithm/model design.

Baseline: Random selection of features with seed 42.

Improved model 1: Selection of features with seed 666.

Improved model 2: Using RandomForestClassifier to select features. Train the model and then sorted the feature importances in descending order and extracted the indices of the top 30 features.

Analysis

Using RandomForestClassifier, aims to iteratively select the most important features from the image features of the validation set. Compared to the baseline (random selection with seed 42), this method ensures feature selection based on their contribution to the model's predictive performance.

3. Experiments

1. Metrics: classification accuracy using the selected features.
2. Experimental results:
    - Baseline: 0.150
    - Improved model 1: 0.235
    - Improved model 2: 0.45

4. Further thoughts

Alternative Feature Selection Methods: Explore other feature selection techniques, like using heuristic function to iteratively select features, as the subtask 2 in project 1.