CS303 Project2 Report

Name: Tan Hao Yang

SID: 12212027

Subtask 1

1. Introduction

1. Sub-task 1 focuses on Supervised Learning for Image Classification, using preprocessed image vectors to train a model for predicting labels. The baseline model, Softmax Regression, transforms input vectors into class probabilities and optimizes performance using cross-entropy loss.
2. The objective is to train an accurate classification model using the provided dataset and optionally improve upon the baseline. This task provides a hands-on opportunity to apply supervised learning techniques and explore custom approaches to enhance performance.

2. Methodology

1. Algorithm/Model Design

The **Softmax Regression Model** used as the baseline algorithm is a linear classifier optimized using the cross-entropy loss function. The algorithm predicts the probabilities of each class and assigns the label with the highest probability. Below is an outline of the algorithm design with a flow chart and pseudocode.

Pseudocode:

```
Input: Feature vectors X, labels Y, learning_rate α, num_iterations T
Output: Optimized weights W

1. Add bias term to X
2. Initialize W randomly
3. For iteration = 1 to T:
   a. Compute logits: o = XW
   b. Apply softmax: P = exp(o) / sum(exp(o), axis=1)
   c. Compute loss: L = -mean(Y * log(P))
   d. Compute gradient: grad = X^T (P - Y) / N
   e. Update weights: W = W - α * grad
4. Return W
```

2. Analysis

Optimality

- **Advantages**: The model is simple and computationally efficient. The softmax regression algorithm ensures that the output probabilities for all classes sum to 1, making it interpretable.
- **Limitations**: As a linear model, it assumes a linear relationship between input features and class labels. It might not perform well on non-linear or complex data distributions.

Complexity

- **Time Complexity**: The primary cost is matrix multiplication for logits computation and gradient updates. For N samples, d-dimensional features, and q classes:
  - Forward pass: $O(N \cdot d \cdot q)$.
  - Backward pass (gradient computation): $O(N \cdot d \cdot q)$.
    Thus, the overall complexity per iteration is $O(N \cdot d \cdot q)$.

Deciding Factors of Performance

- **Learning Rate**: A well-tuned learning rate is critical for convergence and avoiding overshooting or slow optimization.
- **Number of Iterations**: Insufficient iterations might result in underfitting, while excessive iterations could lead to overfitting.

3. Experiments

1. Metrics

To evaluate the model's performance, the following metrics are used:

- **Accuracy**: Measures the proportion of correctly predicted labels over the total number of samples.

2. Experimental Results

- **Baseline Results**: Using the default settings of the Softmax Regression model:
  - Training accuracy: 47.77%
- **Effect of Hyperparameters**:
  - **Number of Iterations**:
    - Increasing iterations from 10000 to 20000 improved accuracy by approximately 0.1%, indicating the need for more training epochs for this dataset.

4. Further Thoughts

If more time and resources were available, the following improvements could be implemented:

1. **Advanced Models**:
   - Incorporate a neural network with non-linear activation functions to capture complex patterns in the data.
   - Explore tree-based models like Random Forest or Gradient Boosting for comparison.
2. **Hyperparameter Tuning**:
   - Use grid search or Bayesian optimization to fine-tune parameters such as learning rate, batch size, and iterations.

Subtask 2

1. Introduction

1. Subtask 2 focuses on **unsupervised learning for image retrieval**, where the goal is to find similar images in a repository using a test image vector. The image repository serves as the training set, and **Nearest Neighbor Search (NNS)** with **Euclidean distance** is the baseline method.
2. The purpose is to develop methods for efficient, unsupervised image retrieval, enabling applications like **content-based image search** in fields such as e-commerce and visual search engines.

2. Methodology

1. Algorithm/Model Design

Baseline: Nearest Neighbor Search (NNS)

The baseline uses the **Nearest Neighbor Search (NNS)** algorithm to find the most similar images in the repository based on the **Euclidean distance** between feature vectors. The key components are:

1. **Feature Normalization**: Input vectors are normalized to ensure consistent scale for distance calculations.
2. **Training**: The image repository's feature vectors are stored in memory and used as the reference set.
3. **Querying**: For each test vector, the algorithm retrieves the top k closest vectors from the repository using Euclidean distance.

Pseudo-code:

```
1. Normalize all feature vectors in the repository (training set).
2. Build a nearest neighbor model with the repository data.
3. For each test feature vector:
    • Normalize the test vector.
    • Calculate the Euclidean distances between the test vector and all repository vectors.
    • Return the IDs of the top $k$ closest repository vectors.
```

2. Analysis

1. **Optimality**:
    - The algorithm guarantees optimal retrieval based on the defined similarity measure (Euclidean distance).
2. **Complexity**:
    - **Training Time**: $O(1)$ for storing repository data (no model learning involved).
    - **Query Time**: $O(n \cdot d)$, where n is the number of repository vectors, and d is the feature dimensionality.
3. **Deciding Factors of Performance**:
    - **Distance Metric**: Euclidean distance may not be optimal for all tasks, and alternative metrics (e.g., cosine similarity) could improve performance.
    - **Repository Size**: As the size of the repository grows, query time increases, requiring optimizations like approximate search.

3. Experiments

1. Metrics

To measure the performance of the image retrieval system, the following metrics are used:

- **Precision@K**: Measures the proportion of relevant images among the top K retrieved results.

2. Experimental Results

- **Baseline (NNS with Euclidean Distance)**:
  - **Precision@5**: 0.0486
- **Alternative Similarity Metric (Cosine Similarity)**:
  - **Precision@5**: 0.0494

3. Observations from Experiments

1. **Effect of Similarity Metrics**:
   - Cosine similarity performed slightly better than Euclidean distance in precision, likely due to its better handling of normalized feature vectors.
   - Euclidean distance was faster for smaller repositories but scaled poorly as the repository size increased.

4. Further Thoughts

Potential Improvements

1. **Alternative Distance Metrics**:
   - Investigate learned similarity metrics using Siamese networks or contrastive learning to better align with human-perceived similarity.
2. **Hyperparameter Optimization**:
   - Conduct a grid search or Bayesian optimization for parameters like k, distance metric, and dimensionality reduction to fine-tune performance.

Subtask 3

1. Introduction

1. General Introduction to Subtask 3
   Subtask 3 focuses on **feature selection** to reduce the dimensionality of input image vectors. The goal is to identify the **30 most relevant dimensions** from the original high-dimensional feature space. The feature selection process is critical to improve model efficiency and performance, as it reduces noise and computational complexity while retaining the most useful information for classification.
2. The purpose of this subtask is to:
   Evaluate the effectiveness of feature selection techniques by comparing the classification accuracy achieved with the selected features. As well as establishing a baseline (random selection) to analyze improvements from more advanced feature selection methods.

2. Methodology

1. Problem Representation

The task is to select **30 features** from the high-dimensional input data to optimize the classification process. The training and testing steps remain unchanged, and the effectiveness of feature selection is measured by classification accuracy. The input data consists of:

- **Validation Data**: A matrix where rows represent samples and columns represent features.
- **Validation Labels**: A vector of corresponding class labels.

The goal is to find the subset of features that maximizes classification accuracy, while reducing dimensionality and computational cost.

2. Algorithm/Model Design

Baseline: Random Selection

- Randomly selects 30 features using a fixed random seed.
- This establishes a naive baseline for comparison.

Proposed Method: SelectKBest with ANOVA F-test

- Uses **ANOVA F-value** to rank features based on their statistical significance in differentiating classes.
- Selects the top 30 features that are most relevant for classification.

Pseudo-code for SelectKBest Method:

```
1. Input: Validation data $X$, labels $Y$, and the number of features to select $k$.
2. Preprocess the data to ensure correct dimensions.
3. Compute the ANOVA F-values for each feature:
    • For each feature $f_i$, calculate the variance between and within classes.
4. Rank features by their F-values.
5. Select the top $k$ features based on their rankings.
6. Generate a binary mask indicating the selected features.
7. Output: Feature mask.
```

3. Analysis

1. **Optimality**:
    - **Random Selection**: Suboptimal, as it does not use data-driven criteria for feature selection.
    - **SelectKBest with F-test**: Selects features based on their statistical contribution to class separation, making it more optimal for classification tasks.
2. **Complexity**:
    - **Random Selection**: $O(1)$, as it simply selects features randomly.
    - **SelectKBest**: $O(m \cdot n)$, where m is the number of samples and n is the total number of features (due to the calculation of F-values).
3. **Deciding Factors of Performance**:
    - **Feature Relevance**: The F-test assumes linear relationships and may not work well if the data is non-linear or noisy.
    - **Number of Features (k)**: The choice of k affects the balance between dimensionality reduction and retaining enough information for classification.

3. Experiments

1. Metrics and Test Flow

Metrics:

- **Classification Accuracy**: Measures how accurately the selected features improve model performance. Higher accuracy indicates better feature selection.

2. Experimental Results

Baseline (Random Selection):

- **Classification Accuracy**: 41.29%

SelectKBest (ANOVA F-test):

- **Classification Accuracy**: 40.33%

Effect of Different Algorithms:

- **Random Selection** performed poorly due to lack of data-driven criteria.
- **SelectKBest** significantly outperformed Random Selection by focusing on the most relevant features, improving the accuracy.

3. Observations

- **Effect of Algorithms**: Algorithms like SelectKBest that use statistical measures outperform naive methods like Random Selection by identifying relevant features effectively.

4. Further Thoughts

Improvements with More Time:

1. **Advanced Feature Selection**:
   - Implement recursive feature elimination (RFE) or LASSO for feature selection to handle non-linear relationships and feature redundancy.
2. **Hyperparameter Optimization**:
   - Use grid search or Bayesian optimization to find the optimal k.