Machine Learning

Lab 2

Instructions

This document lists the details of what to do and what to include in the written report. Sample code for both MATLAB and Python is provided and can give hints on how to proceed. Follow the code in your chosen language and the comments for tips. The parts in the sample code that you need to fill in is marked with:

```
# ===================== YOUR CODE HERE ====================== #
```

A written report should be handed in before the end of the course. The report should include the answers to all the asked questions in this document, all the plots that are created, as well as the code used.

Tips: Google Colab or Jupyter notebooks for python or Matlab LiveScript are useful tools for facilitating this.

The report is graded with either PASS or PASS WITH DISTINCTION. Complete all the exercises to receive the higher grade.

Preparation

Some useful resources:

- PCA for sci-kit learn
- PCA for Matlab
- t-SNE vs. PCA

Exercises for Pass (G)

1 Dimensionality reduction

In this part you will use three different dimensionality reduction methods, namely PCA, LDA, and t-SNE, to visualize high-dimensional data in two dimensions. These methods are also useful for feature selection. We will use a smaller version of the popular MNIST data set to demonstrate how they work. This data set is used to classify handwritten digits 0-9.

1. Load the smallMNIST.mat data set and plot one of the digits in the data set
2. Normalize the data set using mean normalization.
3. Train a PCA, LDA, and t-SNE model on the data set
4. Plot the data in 2 dimensions using the two most significant dimension from each model. Color the data using 10 different color (one for each class).

1.1 Report

Show the plots of the data in two dimensions using the three different dimensionality reduction methods of PCA, LDA, and t-SNE. Discuss the advantages and disadvantages of each method. Also comment on the results. Which classes are grouped together? Which class stands out the most? Which method performed the best?

2 Clustering

In this part, we will use pre-existing libraries to perform k-means clustering on the data set simplecluster.mat and visualize the results.

1. Load the data simplecluster.mat och plot it
2. Normalize the data to 0 mean and 1 standard deviation
3. Use the k-means algorithm with $n\_clusters = \{2, 3, 5, 10\}$

4. Plot the data points and color them according to their cluster belonging. Also plot the centroid for each cluster in the same plot.

5. (optional) Repeat the process for Gaussian Mixture Model (GMM) and compare the results.

2.1 Report

Show the plots and discuss the results. What seems to be an optimal value for the number of clusters for this data set?

3 Classification

In this section, we explore three fundamental algorithms for classification: Decision Trees (DT), k-Nearest Neighbors (k-NN), and Naive Bayes (NB). We will use pre-existing libraries to train these classifiers on the small MNIST dataset.

1. Load the smallMNIST.mat data set

2. Split the data set into 70% training set and 30% test set

3. Train each of the three classifiers DT, k-NN, and NB and calculate the classification accuracy on the test set

4. Explore how different choices of data pre-processing and hyperparameters effects the classification results

3.1 Report

Compare the results with the different classifiers. Report how the performance changed when using different pre-processing techniques. Try using with and without data normalization. With and without using N number of first components from a PCA. Try different values for hyperparameters, for example, the number of neighbours in the K-NN classifier, number of splits in the decision tree, and the number of PCA components to use. Report the best result you got with each classifier and with parameters you chose.

3.2 Mathematical background

Note: You can skip this section if you do not care about the math for the following algorithms.

3.2.1 k-Nearest Neighbors (k-NN)

The k-NN algorithm classifies an unlabeled sample based on the majority label of its k nearest labeled samples.

Given:

- A set of labeled data points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
- An unlabeled data point x

The steps are: 1. For each labeled data point x_i, compute the distance to x:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{i,j})^2}$$

2. Select the k smallest distances and their corresponding labeled data points.

3. Assign the label y for x based on the majority label among the k nearest labeled data points.

3.3 Decision Tree (DT)

Decision trees partition the feature space into regions. For classification, it assigns a class label to each region. The decision of which feature to split on and where to split is determined based on a criterion like "Gini impurity" or "information gain."

For a binary classification:

- Gini Impurity:

$$Gini(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

- Information Gain:

$$IG(t) = Entropy(t) - \sum_{v \in values(feature)} \frac{|t_v|}{|t|} \times Entropy(t_v)$$

Where:

$$Entropy(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$

3.4 Gaussian Naive Bayes (NB)

Gaussian Naive Bayes is based on the Bayes theorem and assumes that features follow a Gaussian distribution. The Bayes theorem is given by:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In the context of classification:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Where $P(x_i|y)$ is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Here, μ_y and σ_y^2 are the mean and variance of the feature x_i for class y in the training data.

4 Logistic Regression for multi-class classification

In this task, we will implement multi-class classification for our logistic regression classifier that we developed previously for classifying handwritten digits ranging from 0 to 9. While logistic regression was originally designed for binary classification, we can adapt it for multi-class classification using the "one-vs-all" strategy.

Specifically you need to:

1. Load the dataset containing handwritten digits. Split the data into train, val, and test sets.

2. Implement the sigmoid function which will be used to compute the hypothesis, the cost function and the gradient function to compute the gradients for logisitc regression. You can use code that you have written previously
3. Train 10 different logistic regression classifiers using an optimization algorithm (like gradient descent or a pre-built function) to minimize the cost function and learn the parameters θ for each class. Each classifier is trained as a one-vs-all for each class.
4. Evaluate the performance of your classifier on the whole test set. The predicted class is the argmax from all 10 classifiers for each test sample.

4.1 Report

Show the classification accuracy on the train, val, and test set.

5 Logistic Regression with L2-Regularization

In this task, we will implement the L2-regularization in our logistic regression classifier and see if it can improve our classification results on the multi-class smallMNIST.mat dataset.

Specifically you need to:

1. Load the dataset containing handwritten digits. Split the data into train, val, and test sets.
2. Implement the L2 regularization in the cost function of your logistic regression classifier
3. Implement the gradient function to compute the gradients. Make sure you include the term for the L2-regularization. Also remember that the gradient for x_0 should be 0.
4. (optional) Check if your implementation is correct by using gradient checking to see if the difference between the analytical and numerical gradient is small.
5. Train the classifier on the training set and evaluate the performance on the test set.

5.1 Report

Report the results for using the L2 regularization. Try different values for the L2-regularization parameter λ.

5.2 Mathematical Background

Note: You can skip this section if you do not care about the math for the following algorithms.

Given our dataset with m examples and n features, our hypothesis $h_\theta(x)$ for logistic regression is defined as:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where:

- x is our input feature vector.
- θ is the parameter vector.

The cost function $J(\theta)$ for logistic regression with regularization is:

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

where:

- λ is the regularization parameter.
- m is the number of training examples.
- n is the number of features.

note that j starts from 1 and not 0 in the regularization term.

6 Bias-variance analysis

Perform two bias-variance analysis on your logistic regression classifier on the smallMNIST data set by plotting the classification accuracy (or classification error) on the training and validation set as a function of:

1. the training iteration, and
2. the hyperparameter λ used for L2-regularization. Try different suitable values, e.g., $\lambda = [0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$.

6.1 Report

Discuss the two plots. Are the models overfit or underfit? Discuss if more training iterations would improve the performance. What are the best value for λ and explain how you reached this conclusion.

Exercises for Pass with distinction (VG)

7 Reconstructions from dimensionality reduction methods

First, perform PCA-based dimensionality reduction on the dataset smallMNIST.mat, and then study the impact of using varying numbers of principal components, K, on the reconstruction of the original data. Visualize the reconstructed data for different values of K and compare with the original images. (optional: also try if the results are different with a LDA or t-SNE).

7.1 Report

Show a plot of the original image and the reconstruction for K=2, K=50, and K=400 for one digit from the smallMNIST.mat data set from a PCA.

7.2 Mathematical background

Note: You can skip this section if you do not care about the math for the following algorithms.

7.2.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that projects the data onto a lower-dimensional space, preserving as much variance as possible.

Given a data matrix X of size $m \times n$ (where m is the number of samples and n is the number of features):

1. Compute the mean μ of each feature and center the data as:

$$Z = X - \mu$$

2. Compute the covariance matrix Σ of the centered data:

$$\Sigma = \frac{1}{m} Z^T Z$$

3. Compute the eigenvectors U and eigenvalues Λ of Σ.

4. Sort the eigenvectors by decreasing eigenvalues and select the first k eigenvectors to form a $n \times k$ dimensional matrix W.

5. Transform the original n dimensional data samples into k dimensions:

$$Y = ZW$$

7.2.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a technique used to find a linear combination of features that best separate two or more classes in a dataset. The primary goal is to maximize the distance between the means of different classes while minimizing the spread (variance) within each class.

Given a set of data points X with labels y and classes C:

1. Compute the class-specific means μ_i for each class C_i:

$$\mu_i = \frac{1}{n_i} sum_{x \in C_i} x$$

Where n_i is the number of samples in class C_i.

2. Compute the overall mean μ of the data:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

3. Calculate the within-class scatter matrix S_W and the between-class scatter matrix S_B:

$$S_W = \sum_{i=1}^{c} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

$$S_B = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$

4. Solve the generalized eigenvalue problem for the matrix $S_W^{-1} S_B$:

$$S_W^{-1} S_B v = lambdav$$

   Where v are the eigenvectors and λ are the eigenvalues.

5. Order the eigenvectors by decreasing eigenvalues and select the first k eigenvectors to form the transformation matrix W.

6. Project the data into the new space using W:

$$Y = XW$$

The goal of LDA is to project the data into a space where the class separability is maximized.

8 Implementation of k-means

Implement k-means in your own code (do not use any pre-implementation of k-means from a library). Use it on the simplecluster.mat data set with $n\_clusters = \{2, 3, 5, 10\}$. Plot the data in different colors according to their cluster belonging.

8.1 Pseudo-Algorithm

1. Initialization: Randomly select k samples from X as initial centroids.

2. Repeat until convergence:

- Assignment Step: Assign each sample to the nearest centroid.

$$c^{(i)} = \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

- Update Step: Compute new centroids for each cluster based on the mean of samples assigned to that cluster.

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x^{(i)}$$

  Where C_j is the set of samples assigned to the j-th cluster.

3. Return the cluster assignments and centroids.

8.2 Report

Report the centroids, plot the data, and the number of iterations needed for convergence. Now compare them with your results from part 2 and specifically: Compare the plots, number of iterations and centroids. The results will be (slightly) different. Explain why?

9 Plot original images of MNIST from k-means clustering

Perform k-means clustering (using pre-existing libraries or your own implementation) on the smallMNIST.mat data set using 10 clusters. Plot 10 randomly drawn original images from each cluster.

9.1 Report

Show the plot and discuss the results. Do the clusters represent the true classes? How suitable is k-means to be used as a classification method in this case?

10 Plot learning curves

Plot the learning curve of a classifier of your choice on the smallMNIST.mat data set. Remember, a learning curve is a plot of the classification accuracy (y-axis) of the (reduced) training and (complete) validation sets as a function of the training data set size (x-axis).

1. Load the data. Split it into train, val, and test set.
2. Train the model on only 10% of the training data and calculate the classification accuracy on the (reduced) training set and (complete) validation set.
3. Repeat step 2 by adding 10% more to the training set until the full training set has been used as training set
4. Plot the learning curve

10.1 Report

Show a plot of the learning curve and discuss if it would be worth to get more training data. Discuss if the model is underfit or overfit?