CSC246 Machine Learning
Project 1: Classification

Overview

The purpose of this assignment is to build your understanding of fundamental classification techniques in machine learning. You will implement two classifiers based on logistic regression and a multilayer perceptron. They will be connected by optimizing cross-entropy loss and using softmax outputs. In addition to implementing these classifiers, you will investigate one or more questions of interest about the models, and collect your findings into a well-written report.

1 Program Requirements

You will implement two different classification techniques, based on logistic regression and multilayer perceptrons, and apply them to two different datasets:

- water – a small dataset with many features and approximately 152 samples obtained via sonar responses to categorize objects underwater as harmless (rocks) or dangerous (mines)
- loans – a medium-sized dataset with fewer features but approximately 1.6k samples obtained from a financial institution. The features correspond to financial and personal attributes, and the target variable corresponds to whether or not a person will default on their debts in the next two years.

Note: These datasets have wildly ranging feature values – you are encouraged to consider rescaling them for more efficient learning.

Logistic Regression

- You must implement a LogisticRegression class in a file named logreg.py following our format below.
- You should implement multivariate logistic regression with softmax outputs, and weights should be initalized using zero-mean gaussians.
- It should be trained using (stochastic) gradient descent, and you should analyze the results when applied to both datasets.

Multilayer Perceptron

- You must implement a MultilayerPerceptron class in a file named mlp.py following our format below.
- You should implement a multilayer perceptron with 10 tanh units and softmax outputs, and initialize layer weights with zero mean unit variance gaussians. Biases can be initialized to zero.
- It should be trained with (stochastic) gradient descent, and you should analyze the results when applied both datasets
- In addition to your program, you must also train (and save) your best overall MLP model for each of the datasets. Use filenames "water.model" and "loans.model". You can train these models as long as you like, using as many hidden variables or layers as you like, with the goal of good **predictive** performance on the held out dataset.

API and Formatting

To facilitate automatic testing, you must follow our format exactly.

- To train a model, use a no-argument constructor followed by a call to fit(X,T), where X is a numpy matrix holding the training data, and T is a one-hot encoding of the target variables.
- To apply a trained model, use only the method predict(X), which should return the model outputs (classifications) for the whole dataset.
- To save a model to a file, call save(path).

- To load a saved model, use a static method named LogisticRegression.load(path), (similar for MultilayerPerceptron), which loads and returns a trained model.
- You are free (and encouraged) to create additional arguments to control hyperparameters (such as learning rate, number of epochs, batch size, regularization, number of hidden units, and initialization scheme); however, your program MUST work using the format below. Therefore you are required to establish (and document) reasonable default values for all your hyperparameters. To assist with data loading and evaluation, you are provided with a utility file that implements these functions (as well as an efficient softmax implementation.)
- In summary, you are writing two classes, both of which should support methods for fit, predict, save, and load. An example usage is given below.

2 Writeup

An important aspect of this course is for students to develop a deep understanding of how machine learning can be applied to real problems. In order to grow (and assess) your understanding, you are required to write a short report that responds to three general prompts corresponding to analysis of your two models, and your response to an open-ended research question.

Logistic Regression Analysis

Your writeup should describe how well logistic regression solves the classification problems posed by both datasets. Because the test data is hidden, you should use the validation data to estimate the models generalization performance. You are not required to use regularization, but you are encouraged to attempt to solve the problem of obtaining a good predictive model as well possible. Your writeup should include narrative text that describes the quality of your model in terms of accuracy, generalization, compute requirements, and reliability. You should include a plot of accuracy vs compute (epochs, time) for your best configuration. This section should be approximately one page of text, maybe less.

Multilayer Perceptron Analysis

Similarly, you should describe how well a basic MLP with 10 tanh units solves the same classification problems; however, using a MLP should obtain different results. They have more theoretical power than logistic regression, but greater computational demands and more subtle training requirements. Be sure to describe how you trained your predictive models that you are submitting with this project. This section should be approximately one page of text.

Research Question

In addition to a basic writeup validating the correctness of your implementation, you should approach this assignment as a mini research project by answering one of the following questions. Begin by thinking about the question, and design an experiment using your code. You should also research the general topic to identify relevant papers from top-tier conferences (such as ICLR, NeurIPS, IJCAI, AAAI, etc.) and relate those papers to your work. This is a very open-ended requirement; all technically strong and good faith attempts will receive full credit, regardless of result. The total length of your response to the question is expected to be approximately one page of text, maybe more.

1. What's the best MLP you can fit in under 1k params? (Compare wide vs deep networks.)
2. How important are good initialization? (Compare different initialization schemes.)
3. How do weight space symmetries affect explainability? (Inspect feature weights obtained by logistic regression, identify important features, inspect weights obtained by MLP, determine if the same features have similar importance. (Start by defining the "weight" of a feature as the sum of all outgoing connections.)
4. What is "double descent", and can you recreate it? (See if you can build an MLP large enough that solves its own overfitting problem.)
5. What are "neural scaling laws", and can you fit one? (See if you can quantify the rate at which your network trains, based on the number of parameters, size of the dataset, compute directed at training, and final network performance.)

3 Grading

Your project will be graded according to these criteria:

- **learning** – 40pts – your program will be autograded using the fit methods for both models on both datasets. Each of the four combinations will be tested five times, and your final predictive accuracy (on the training data) will be compared to a threshold (60%). Your score on each test case will be the number of times your model passes the threshold. Your computation will be limited to approximately one minute per configuration.
- **models** – 10pts – in addition to writing programs, you should submit your best trained MLP models for both datasets, which will be evaluated on held-out test data and compared to a threshold (70%).
- **writeup** – 50 pts – the two basic sections are worth 15 points each, and the research question response is worth 20 points

Please note: In order to streamline grading, please ensure that your full name and UR email are listed at the top of your writeup and your source files.

4 FAQ

- **How long does my write-up need to be?**

  However long you need to fully address all of the points to include.

- **What do I need to show for my results? Do I need graphs/tables/etc?**

  Like in any scientific paper, results need to be more than just the numbers you find in the end. You need to include the evidence from your experiments that shows your result is correct, and qualitative claims should be supported by quantitative evidence whenever possible. Figures such as graphs and tables can be excellent ways to present your findings, as long as they are appropriately explained and discussed in the text of your write-up.

- **How should I structure my write-up? Do I need sections for each experiment?**

  This is one way you could do it, but you are free to structure your write-up however you see fit, so long as it is well organized and includes everything it needs to.

- **Do I need to talk about X in my write-up?**

  If you are asked to investigate it as part of the assignment then you should discuss it in your write-up.

- **I did Y in addition to the assignment requirements. Can I include it?**

  Go for it!

- **Can I use an LLM (like ChatGPT) to write my code? Or my write-up?**
  No. For this course, we ask that all code and writing you submit be written by you. For your write-ups, you may use an LLM-based tool to work on your wording for small pieces of text, but the structure, voice, and ideas must be your own. We ask that you refrain from using LLM code generation tools such as Copilot.

- **I have a question that wasn't answered here.**

  Your TA team is here to help! Ask your question on Discord, send an email, or come to office hours. Don't be afraid to reach out; if you have a question, chances are others are wondering the same thing.

Example Usage

```
from util import *
from logreg import LogisticRegression
from mlp import MultilayerPerceptron
X,T = loadDataset(path)
T = toHotEncoding(T, 2)

m = LogisticRegression()
m.fit(X,T)
Y = m.predict(X)
acc_train = accuracy(Y, T)


m2 = MultilayerPerceptron()
m2.fit(X,T)
Y = m2.predict(X)
acc_train2 = accuracy(Y, T)
m2.save("mlp.model")


m3 = MultilayerPerceptron.load("mlp.model")
Y = m3.predict(X)
acc_train3 = accuracy(Y, T)
```

Utility Functions

```
import numpy as np

def softmax(a):
    ea = np.exp(a - np.max(a, axis=1, keepdims=True))
    return ea / np.sum(ea, axis=1, keepdims=True)

def accuracy(y_true, y_pred):
    return np.sum(np.argmax(y_true,axis=1) == np.argmax(y_pred, axis=1)) / y_true.shape[0]

def toHotEncoding(t, k):
    ans = np.zeros((t.shape[0], k))
    ans[np.arange(t.shape[0]), np.reshape(t, t.shape[0]).astype(int)] = 1
    return ans


def loadDataset(filename):
    """ Read CSV data from file.
    Returns X, Y values with hot targets. """
    labels=open(filename).readline().split(',')
    data=np.loadtxt(filename, delimiter=',', skiprows=1)
    X=data[:,:-1] # observations
    T=data[:,-1]  # targets, discrete categorical features (integers)
    K=1 + np.max(T).astype(int) # number of categories
    N=X.shape[0]  # number of observations
    D=X.shape[1]  # number of features per observation
    return X, T
```