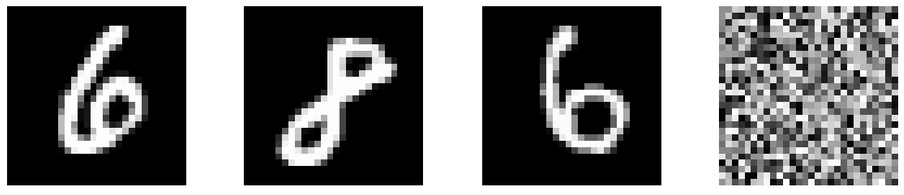
1. Introduction

Alice, Bob, and Daniel are friends learning machine learning together. After watching a few lectures, they are very proud of having learned many useful tools, including linear and logistic regression, non-linear features, regularization, and kernel tricks. To see how these methods can be used to solve a real life problem, they decide to get their hands dirty with the famous digit recognition problem using the MNIST (Mixed National Institute of Standards and Technology) database.

Hearing that you are an excellent student in the MITx machine learning class with solid understanding of the material and great coding ability in Python, they decide to invite you to their team and help them with implementing these different algorithms.

The MNIST database contains binary images of handwritten digits commonly used to train image processing systems. The digits were collected from among Census Bureau employees and high school students. The database contains 60,000 training digits and 10,000 testing digits, all of which have been size-normalized and centered in a fixed-size image of 28 × 28 pixels. Many methods have been tested with this dataset and in this project, you will get a chance to experiment with the task of classifying these images into the correct digit using some of the methods you have learned so far.



Setup:

As with the last project, please use Python's **NumPy** numerical library for handling arrays and array operations; use **matplotlib** for producing figures and plots.

This project will be split in two parts. Project 2 (this project) consists in the first part and project 3 will cover the second part.

1. *Note on software: For all the projects, we will use python 3.6 augmented with the **NumPy** numerical toolbox, the **matplotlib** plotting toolbox.*

2. Download mnist.tar.gz and untar it in to a working directory. The archive contains the various data files in the Dataset directory, along with the following python files:

   - `part1/linear_regression.py` where you will implement linear regression
   - `part1/svm.py` where you will implement support vector machine
   - `part1/softmax.py` where you will implement multinomial regression
   - `part1/cubic.py` where you will implement a cubic features mapping
   - `part1/kernel.py` where you will implement polynomial and Gaussian RBF kernels
   - `part1/main.py` where you will use the the code you write for this part

Important: The archive also contains files for the second part of the MNIST project. For this project, you will only work with the `part1` folder.

To get warmed up to the MNIST data set run `python main.py`. This file provides code that reads the data from **mnist.pkl.gz** by calling the function `get_MNIST_data` that is provided for you in **utils.py**. The call to `get_MNIST_data` returns Numpy arrays:

1. `train_x` : A matrix of the training data. Each row of `train_x` contains the features of one image, which are simply the raw pixel values flattened out into a vector of length $784 = 28^2$. The pixel values are float values between 0 and 1 (0 stands for black, 1 for white, and various shades of gray in-between).

2. `train_y` : The labels for each training datapoint, aka the digit shown in the corresponding image (a number between 0-9).
3. `test_x` : A matrix of the test data, formatted like `train_x`.

4. `test_y` : The labels for the test data, which should only be used to evaluate the accuracy of different classifiers in your report.

Next, we call the function `plot_images` to display the first 20 images of the training set. Look at these images and get a feel for the data (don't include these in your write-up).

Tip: Throughout the whole online grading system, you can assume the NumPy python library is already imported as `np`. In some problems you will also have access to python's `random` library, and other functions you've already implemented. Look out for the "Available Functions" Tip before the codebox, as you did in the last project.

This project will unfold both on MITx and on your local machine. However, we encourage you to first implement the functions locally and run the test scripts to validate basic funcitonality. Think of the online graders as a submission box to submit your code when it is ready. You should not have to use the online graders to debug your code.

Discussion

Hide Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Project 2: Digit recognition (Part 1) / 1. Introduction

Add a Post

Show all posts ▼ by recent activity ▼

Project2 setup
Project2 needs sklearn package. conda activate 6.86x conda install scikit-learn
Community TA
4

[Staff] File names in tar ball
Hi, I think there is a typo in listed file names (list item #2). I downloaded the tar ball and after untar, I get these as listed files under 'part1' directory. cubic_features_checker.py...
2

Do we get the answers for home work 1 probelems?
will the results be published for homework 0 and 1.
2

Learn About Verified Certificates