

CSC 411
Machine Learning and Data Mining
Assignment 2
Out: Oct. 6
Due: Oct. 17 [noon]

Overview

In this assignment, you will build a decision tree, and experiment with a neural network and a naive Bayes model. Some code that implements a neural network with one hidden layer, and the naive Bayes model will be provided for you (both MATLAB and Python).

You will be working with the following dataset:

Digits: The file `digits.mat` contains 6 sets of 16×16 greyscale images in vector format (the pixel intensities are between 0 and 1 and were read into the vectors in a raster-scan manner). The images contain centered, handwritten 2's and 3's, scanned from postal envelopes. `train2` and `train3` contain examples of 2's and 3's respectively to be used for training. There are 300 examples of each digit, stored as 256×300 matrices. Note that each data vector is a column of the matrix. `valid2` and `valid3` contain data to be used for validation (100 examples of each digit) and `test2` and `test3` contain test data to be used for final evaluation **only** (200 examples of each digit).

1 Decision Trees (20 points)

The Department of Computer Science used to have a lot of free food events for students. However, recently the free food supply has dropped quite a lot, so the admin staff has to decide which students will get it. As the number of students interested in free food is large, you are asked to build a machine learning system to automate the decision process.

You are given some past data containing student information and the amount of free food given to each of the students, shown in Table 1. Each row in the table corresponds to one student and each student is described by 4 attributes (the first 4 columns): grad or undergrad; taken CSC 411 or not; level of hunger; and level of interest in free food. The amount of free food (the last column) is measured in three different levels: large, medium and small.

Build a decision tree by hand on this data set using the greedy algorithm described in class. Show the steps involved in constructing the tree.

<i>status</i>	<i>'411'</i>	<i>hunger</i>	<i>interest</i>	food-serving
undergrad	yes	high	high	large
undergrad	yes	medium	medium	medium
grad	no	high	high	medium
grad	no	low	low	small
grad	no	medium	medium	small
grad	yes	medium	medium	small
undergrad	no	high	medium	medium
undergrad	yes	high	high	large
undergrad	no	low	medium	small
undergrad	yes	high	high	large
grad	no	medium	high	medium
grad	no	medium	medium	small
undergrad	yes	high	medium	large
grad	yes	medium	high	medium

Table 1: Training data.

2 Neural Networks (36 points)

Code for training a neural network with one hidden layer of logistic units, logistic output units and a cross entropy error function is included. The main components are:

MATLAB

- `init_nn.m`: initializes the weights and loads the training, validation and test data.
- `train_nn.m`: runs `num_epochs` of backprop learning.
- `test_nn.m`: Evaluates the network on the test set.

Python

- `nn.py` : Methods to perform initialization, backprop learning and testing.

a) Basic generalization [12 points]

Train a neural network with 10 hidden units. You should first use `init_nn` to initialize the net, and then execute `train_nn` repeatedly (more than 5 times). Note that `train_nn` runs 100 epochs each time and will output the statistics and plot the error curves. Alternatively, if you wish to use Python, set the appropriate number of epochs in `nn.py` and run it. Examine the statistics and plots of training error and validation error (generalization). How does the network's performance differ on the training set versus the validation set during learning? Show a plot of error curves (training and validation) to support your argument.

b) Classification error [8 points]

You should implement an alternative performance measure to the cross entropy, the mean classification error. You can consider the output correct if the correct label is given a higher probability than the incorrect label. You should then count up the total number of examples that are classified incorrectly according to this criterion for training and validation respectively, and maintain this statistic at the end of each epoch. Plot the classification error vs. number of epochs, for both training and validation.

c) Learning rate [8 points]

Try different values of the learning rate ϵ (“eps”) defined in `init_nn.m` (and in `nn.py`). You should reduce it to .01, and increase it to 0.2 and 0.5. What happens to the convergence properties of the algorithm (looking at both cross entropy and %Correct)? Try momentum of {0.0, 0.5, 0.9}. How does momentum affect convergence rate? How would you choose the best value of these parameters?

d) Number of hidden units [8 points]

Set the learning rate ϵ to .02, momentum to 0.5 and try different numbers of hidden units on this problem (you might also need to adjust `num_epochs` accordingly in `init_nn.m`). You should use two values {2, 5}, which are smaller than the original and two others {30, 100}, which are larger. Describe the effect of this modification on the convergence properties, and the generalization of the network.

3 Naive Bayes (28 points)

In this question you will experiment with a binary naive Bayes classifier for the Digits dataset. In a naive Bayes classifier, the conditional distribution for example $\mathbf{x} \in \mathbb{R}^d$ to take on class c (out of K different classes) is defined by

$$p(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)p(c)}{\sum_{k=1}^K p(\mathbf{x}|k)p(k)}$$

where $p(\mathbf{x}|c) = \prod_{i=1}^d p(x_i|c)$ according to the naive Bayes assumption. In this question, we model $p(x_i|c)$ as a Gaussian for each i as

$$p(x_i|c) = \mathcal{N}(x_i|\mu_{ic}, \sigma_{ic}^2) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} \exp\left(-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}\right)$$

The prior distribution $p(c)$ and parameters $\mu_c = (\mu_{1c}, \dots, \mu_{dc})^\top$, $\sigma_c^2 = (\sigma_{1c}^2, \dots, \sigma_{dc}^2)^\top$ for all c are learned on a training set using maximum likelihood estimation.

Code for training this binary naive Bayes classifier is included. The main components are: MATLAB

- `train_nb.m`: trains a naive Bayes classifier given some data.
- `test_nb.m`: tests a trained naive Bayes classifier on some test data.
- `visualize_digits.m`: visualizes a data matrix that contains digits.

Python

- `nb.py`: includes code to train and test naive Bayes classifiers, as well as code for visualization.

You are required to fill in `run_nb.m` in MATLAB or the `main` method of `nb.py` in Python to complete the pipeline of training, testing a naive Bayes classifier and visualize learned models. The code you need to fill in should be less than 10 lines.

Report the training and test accuracy using the naive Bayes model, and show the visualization of the mean and variance vectors μ_c and σ_c^2 for both classes. Briefly comment on the visualization results.

4 Compare k -NN, Neural Networks and Naive Bayes (16 points)

Try k -NN on this digit classification task using the code provided in the first assignment, and compare the results with those you got using neural networks and naive Bayes. Briefly comment on the differences between these classifiers.

Write up

Hand in answers to all the questions in the parts above. The goal of your write-up is to document the experiments you've done and your main findings. So be sure to explain the results.

It is up to you whether you would like to do the write-up of Part 1 (Decision Trees) by hand or on the computer. In either case you should turn in a hardcopy of your answer to Part I. The answers to the rest of the questions should be in pdf form, and turned in along with your code.

Package your code and a copy of the write-up pdf document using `zip` or `tar.gz` in a file called `CSC411-A2-*your-student-id*.[zip|tar.gz]`. Only include functions and scripts that you modified. Submit this on MarkUs by October 17, 2014, noon. Turn in a hard copy of the write-up of Part 1 before class on October 17 as well.