Homework 6

Written Due: May 1, 2018 at 7:00PM
Coding Due: May 4, 2018 at 7:00PM

Written Questions

The *Stranger Things* squad discovered that the Dig Dug game in the Palace randomly changes places with its exact duplicate in the Upside Down. Each time you put in a quarter, you are playing the real world version or the Upside Down version with probability $1/2$. The two games are identical, except Max doesn't do quite as well with the version from the Upside Down. We want to help them figure out how the two games differ.

The squad pools their money and Max plays 100 games of Dig Dug. Of the three lives in the game, let $w(i) \in \{0, 1, 2, 3\}$ be the number of times Max wins on quarter i. Assume that her probability of winning on each life is iid.

We want to devise an EM algorithm to find the maximum likelihood estimate for r and u, the probability of Max winning on any given life on the Dig Dug from the real world (r) and on the Dig Dug from the Upside Down (u).

1. Write and justify an expression for $f(w|p)$, the probability of w wins (out of 3) given that the probability of a win is p.
2. Let r_t, u_t denote our estimates for r, u after t iterations of EM. Derive the probability $l_t(i)$ that outcome $w(i)$ is from the real game. Express using f.
3. Write and justify an expression for the log likelihood of the data with respect to a choice of r and u.
4. Using the previous expression, derive the MLE for r_{t+1} and u_{t+1} given $l_t(i)$. Express using $l_t(i)$.
5. Could we skip EM and optimize r and u directly via SGD? Why or why not?

Programming Assignment

Introduction

In this assignment, you will be using the EM algorithm to train a more flexible Naive Bayes classifier. To test the performance of the classifier, you will once again be using the famous MNIST handwritten digits dataset. However, for this assignment, you will be training your model on a different classification task. Instead of predicting the exact digit (0 to 9), you will train a classifier that can predict whether a given digit is even or odd.

Note: You may find this programming assignment conceptually more difficult than the previous assignments. We encourage you to thoroughly read through this handout before writing any code. Additionally, you are encouraged to come to *clinic hours* for high-level conceptual questions (but not debugging help) related to the programming assignment.

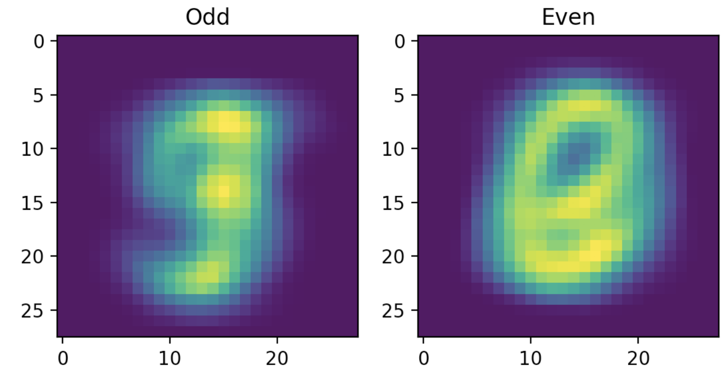
Motivation

In this assignment, you will be training a classifier to predict whether a given handwritten digit is even or odd. Given that you already wrote a Naive Bayes classifier in hw2, you can easily try it out on this problem and see how well it performs.

If you revisit your solution to hw2, and add the following code to your `main.py` file, you should see that the classifier obtains an accuracy of around 80%.

```
data_train_even_odd = Dataset(inputs, (np.mod(labels, 2) == 0).astype(np.uint8))
data_test_even_odd = Dataset(inputs, (np.mod(labels, 2) == 0).astype(np.uint8))
eo_model = NaiveBayes(2)
eo_model.train(data_train_even_odd)
print('Train accuracy', eo_model.accuracy(data_train_even_odd))
```

For such a simple binary classification task, we would expect much better than 80% accuracy! To debug why our classifier is performing so poorly, we can visualize the results of the trained classifier. Plotting the learned parameters, $P(x_{ij} = 1|y = \text{“even”})$ and $P(x_{ij} = 1|y = \text{“odd”})$ for $0 \leq i \leq 27, 0 \leq j \leq 27$, you would see the following results:



What do we notice? The visualization of the parameters for the odd classifier appear to be all of the odd numbers $1, 3, 5, 7, 9$ blurred together, while the even classifier appears to be all of the even numbers $0, 2, 4, 6, 8$ blurred together. This observation highlights that there are *multiple* ways for the class “odd” to be instantiated in the features. (Ex: you can write 1, 3, 5, 7, 9 and they should all be classified as odd). We would like our model to be flexible enough to represent the multiple ways in which the class "odd" can be instantiated in the features. To do this, we will need to add latent variables to our model.

The Assignment

Probabilistic Model

Let $\boldsymbol{x}$ be the input vector, h^j for $j \in \{1, \ldots, k\}$ represent the latent states and $y \in \{0, 1\}$ be the class label. Our classifier should output the label $\hat{y}$ that maximizes $P(\widehat{y}|\boldsymbol{x})$, that is,

$$\widehat{y} = \operatorname*{argmax}_y P(y|\boldsymbol{x}) = \operatorname*{argmax}_y P(y)P(\boldsymbol{x}|y).$$

We can write $P(\boldsymbol{x}|y)$ in terms of the hidden states h^j using the law of total probability,

$$P(\boldsymbol{x}|y) = \sum_{j=1}^{k} P(h^j|y)P(\boldsymbol{x}|y, h^j).$$

Finally, recall that the Naive Bayes assumption is that the features are conditionally independent given the label. Here, we assume that features are conditionally independent given the label *and* hidden state h^j.

$$P(\boldsymbol{x}|y, h^j) = \prod_i P(x_i|y, h^j).$$

Combining the expressions above,

$$\hat{y} = \operatorname*{argmax}_y P(y) \sum_{j=1}^{k} \left(P(h^j|y) \prod_i P(x_i|y, h^j) \right).$$

If we would like to make predictions using the probabilistic model defined above, we need to learn the following parameters:

$$b^y = P(y) \text{ for all } y \text{ (via MLE)},$$
$$b^{jy} = P(h^j|y) \text{ for all } y, j \text{ (via EM), and}$$
$$b^{ij} = P(x_i = 1|y, h^j) \text{ for all } i, y, j \text{ (via EM)}.$$

Maximum Likelihood Estimation

You can learn $P(y)$ using Maximum Likelihood Estimation. That is, you can estimate b^y by counting the fraction of examples with label y:

$$\widehat{b^y} = \frac{\sum_{(\boldsymbol{x},\tilde{y}) \in S} (\tilde{y} = y)}{|S|}.$$

The EM Algorithm

The quantities b^{jy} and b^{ij} cannot be easily approximated using maximum likelihood estimation because our dataset does not include the hidden states h^j. Instead, we will use the EM algorithm.

First, partition the dataset S based on the class labels. Let $S_y = \{(\boldsymbol{x}, \tilde{y}) \in S : \tilde{y} = y\}$. Run the EM algorithm on each subset S_y:

- The parameters of the model are $\theta = \{b^{jy}, b^{ij}\}$. Initialize these parameters randomly from the Uniform Distribution on $(0, 1)$.

- For each data point $\boldsymbol{x} \in S_y$, initialize $Q_{\boldsymbol{x},h^j}(1) = P(h^j|\boldsymbol{x}, \theta) = \frac{1}{k}$ (a uniform distribution over the hidden states).
- Starting with $t = 1$ up to the maximum number of iterations (or until the parameters converge):
  - **E-step**

$$\begin{aligned} \text{Update } Q_{\boldsymbol{x},h^j}(t+1) &= P(h^j|\boldsymbol{x}, y, \theta(t)) \\ &= \frac{P(h^j|y,\theta(t)) \prod_i P(x_i|h^j, y, \theta(t))}{\sum_{j=1}^{J} P(h^j|y,\theta(t)) \prod_i P(x_i|h^j, y, \theta(t))} \\ &= \frac{b^{jy}(t) \prod_i \left(b^{ij}(t)\right)^{x_i} \left(1 - b^{ij}(t)\right)^{1-x_i}}{\sum_{j=1}^{k} b^{jy}(t) \prod_i \left(b^{ij}(t)\right)^{x_i} \left(1 - b^{ij}(t)\right)^{1-x_i}}. \end{aligned}$$

  - **M-step**
    Update the parameters $\theta(t+1) = \{b^{jy}(t+1), b^{ij}(t+1)\}$:

$$b^{jy}(t+1) = \frac{\sum_{\boldsymbol{x} \in S_y} Q_{\boldsymbol{x},h^j}(t+1)}{|S_y|} \text{ for } j = 1, \ldots, k$$

$$b^{ij}(t+1) = \frac{\sum_{\boldsymbol{x} \in S_y} Q_{\boldsymbol{x},h^j}(t+1) x_i}{\sum_{\boldsymbol{x} \in S_y} Q_{\boldsymbol{x},h^j}(t+1)} \text{ for } j = 1, \ldots, k.$$

- Return $\theta(T)$.

Important Note: You should implement the EM algorithm using only Python and Numpy. You should not use any libraries that make the implementation trivial.

Coding Incrementally

We recommend that you complete this assignment by following the following steps:

1. Implement the EM algorithm described above.
2. We have provided a `generate_data` function that generates a very simple dataset assuming the probabilistic model above. Run your EM algorithm on this dataset and ensure that it correctly recovers the parameters of the generative model.
3. Change your probabilistic computations so that they are computed in the log-space.
   To avoid overflow issues, you will most likely need to use the following trick:

$$\log \sum_{i=1}^{N} \exp(x_i) = a + \log \sum_{i=1}^{N} \exp(x_i - a) \text{ for any constant } a.$$

   We recommend that you set $a = \max_i x_i$ to ensure that there is no overflow.
4. Run your algorithm on the MNIST dataset. Print out the parameters at each iteration and ensure that none are overflowing or underflowing to `inf`, `-inf` or `nan`.
5. Print out the training accuracy of your model. It should be around 85%-90% when $k = 5$.
6. Plot the hidden states using the `plot_parameters` function provided.

Stencil Code & Data

You can find the stencil code for this assignment on the course website. We have provided the following two files:

- `main.py` is the entry point of your program which will read in the data, run the classifier and print the results.
- `models.py` contains the `EMNaiveBayesClassifier` model which you will be implementing.

You should *not* modify any code in the `main.py`. All the functions you need to fill in reside in `models.py`, marked by `TODO`s.
To feed the data into the program successfully, please do *not* rename the data files and also make sure all the data files are in a directory named `data` located in the same directory as `main.py`. To run the program, run `python main.py` in a terminal.

MNIST Dataset

You will be using the well-known MNIST Dataset, which includes 60,000 examples for training and 10,000 for testing. All images are size normalized to fit in a 20×20 pixel box and this box is centered in a 28×28 image based on the center of mass of its pixels. The full description of this dataset is available at `http://yann.lecun.com/exdb/mnist/`. There are four files:

- `train-images-idx3-ubyte.gz`: images of 60,000 training examples,
- `train-labels-idx1-ubyte.gz`: labels of 60,000 training examples,
- `t10k-images-idx3-ubyte.gz`: images of 10,000 testing examples,
- `t10k-labels-idx1-ubyte.gz`: labels of 10,000 testing examples.

On a department machine, you can copy the files to your working directory by running the command

```
cp /course/cs1420/data/hw6/* <DEST DIRECTORY>
```

where `<DEST DIRECTORY>` is the directory where you would like to copy the four data files. If you are working locally, you will need to use the `scp` command to copy the files to your computer over `ssh`:

```
scp <login>@ssh.cs.brown.edu:/course/cs1420/data/hw6/* <DEST DIRECTORY>
```

Data Format

The original feature values in this dataset are integers ranging from 0 to 255 (encoding grey scale values), but we have thresholded those features to make them binary (1 for values higher than 99, 0 otherwise). The labels have been modified so that 0 represents an even number and 1 represents an odd number.
As always, we have written all the preprocessing code for you. The final dataset is represented by a `namedtuple` with two fields:

- `data.inputs` is a $m \times 784$ `NumPy` array that contains the binary features of the m examples.
- `data.labels` is a m dimensional `NumPy` array that contains the labels of the m examples.

You can find more infomation on `namedtuple` **here**.
Your program assumes the data is formatted as follows: The first column of data in each file is the dependent variable (the observations y) and all other columns are the independent input variables $(x_1, x_2, \ldots, x_n)$. We have taken care of all data preprocessing, as usual.

If you're curious and would like to read about the datatset, you can find more information **here**, but it is strongly recommended that you use the versions that we've provided in the course directory to maintain consistent formatting.

Report

- Discuss the performance of your classifier. If their are any issues with your implementation, discuss them here.
- Using the `plot_parameters` function provided, plot the learned parameters of your model with $k = 5$. Discuss the results. Include the figure in your report.
- (Extra Credit) Run your classifier on MNIST with $k = 1, 5, 10, 20$. Record the accuracies, create plots of the parameters (such as a visualization similar to those we've included in this handout) and interpret the results.
- (Extra Credit) Discuss the following: Can you approach this task with Gradient Descent? If so, how would you do it? What would be the advantages/disadvantages of doing so? If not, what is your reasoning?

Grading

Loss Targets

Your classifier should score an accuracy between 85-90% using 5 hidden states for each class ($k = 5$). Results will depend on random initialization. We will run your code multiple times to ensure that it is working correctly.

As always, we grade primarily based on code correctness not based on accuracy.

Breakdown

Written Questions	30%
EM Classifier (Fake Data)	40%
EM Classifier (MNIST)	20%
Report	10%
Total	100%

Handing in

Written Questions

Answers to the written questions should be printed, stapled and labeled with the date, homework number and your unique course ID. You should place your handin in the CS1420 handin box located on the second floor of the CIT.

Programming Assignment

To hand in the programming component of this assignment, first ensure that your code runs on *Python 3* using our course `virtualenv`. You can activate the `virtualenv` on a department machine by running the following command in a Terminal:

```
source /course/cs1420/cs142_env/bin/activate
```

Once the `virtualenv` is activated, run your program and ensure that there are no errors. We will be using this `virtualenv` to grade all programming assignments in this course so we recommend testing your code on a department machine each time before you hand in. Note that handing in code that does not run may result in a significant loss of credit.

To hand in the coding portion of the assignment, run `cs142_handin hw6` from the directory containing all of your source code and your report in a file named `report.pdf`.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin. Instead, you should use the course ID that you generated when filling out the collaboration policy form. If you do not have a course ID, you should email the HTAs as soon as possible.

Obligatory Note on Academic Integrity

Plagiarism—don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and, if you have any questions, please contact a member of the course staff.

Feedback

Many aspects of this course have been radically redesigned compared to previous iterations of the course. To continually improve this class and future iterations of the class, we are offering an online **anonymous feedback form**. If you have comments, we would love to hear your constructive feedback on this assignment and on any other aspects of the course.