

# CS440/ECE448 Fall 2016

## Assignment 3: Naive Bayes Classification

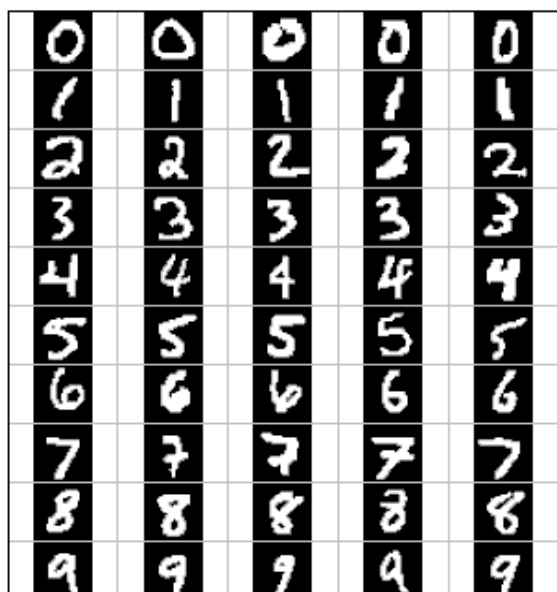
**Due date: Monday, November 14, 11:59:59 PM**

The goal of this assignment is to implement Naive Bayes classifiers as described in [this lecture](#) and to apply it to the task of classifying visual patterns and text documents. As before, you can work in teams of up to three people (three-unit students with three-unit students, four-unit students with four-unit students).

### Contents

- Part 1: [Digit classification](#)
  - Part 1.1 (for everybody): [Single pixels as features](#)
  - Part 1.2 (for four-unit students): [Pixel groups as features](#)
- Part 2: [Text document classification](#)
  - Part 2.1 (for everybody): [Sentiment analysis of movie reviews and binary conversation topic labeling](#)
  - Part 2.2 (for four-unit students): [Conversation topic labels identification from a 40-topic dataset](#)
- [Report checklist](#)
- [Submission instructions](#)

### Part 1: Digit classification



[Berkeley CS 188 project 5](#))

**Data:** [This file](#) is a zip archive containing training and test digits, together with their ground truth labels (see `readme.txt` in the zip archive for an explanation of the data format). There are 5000 training exemplars (roughly 500 per class) and 1000 test exemplars (roughly 100 per class).

#### Part 1.1 (for everybody): Single pixels as features

- **Features:** The basic feature set consists of a single binary indicator feature for each pixel. Specifically, the feature  $F_{i,j}$  indicates the status of the  $(i, j)$ -th pixel. Its value is 1 if the pixel is foreground (no need to distinguish between the two different foreground values), and 0 if it is background. The images are of size  $28 \times 28$ , so there are 784 features in total.

- **Training:** The goal of the training stage is to estimate the **likelihoods**  $P(F_{ij} \mid \text{class})$  for every pixel location  $(i, j)$  and for every digit class from 0 to 9. The likelihood estimate is defined as

$$P(F_{ij} = f \mid \text{class}) = (\text{\# of times pixel } (i, j) \text{ has value } f \text{ in training examples from this class}) / (\text{Total \# of training examples from this class})$$

In addition, as discussed in the lecture, you have to **smooth** the likelihoods to ensure that there are no zero counts. *Laplace smoothing* is a very simple method that increases the observation count of every value  $f$  by some constant  $\kappa$ . This corresponds to adding  $\kappa$  to the numerator above, and  $\kappa \cdot v$  to the denominator (where  $V$  is the number of possible values the feature can take on). The higher the value of  $\kappa$ , the stronger the smoothing. Experiment with different integer values of  $\kappa$  (say, from 1 to 50) and find the one that gives the highest classification accuracy.

You should also estimate the **priors**  $P(\text{class})$  by the empirical frequencies of different classes in the training set.

- **Testing:** You will perform **maximum a posteriori (MAP)** classification of test digits according to the learned Naive Bayes model. Suppose a test image has feature values  $f_{1,1}, f_{1,2}, \dots, f_{28,28}$ . According to this model, the posterior probability (up to scale) of each class given the digit is given by

$$P(\text{class}) \cdot P(f_{1,1} \mid \text{class}) \cdot P(f_{1,2} \mid \text{class}) \cdot \dots \cdot P(f_{28,28} \mid \text{class})$$

Note that in order to avoid underflow, it is standard to work with the log of the above quantity:

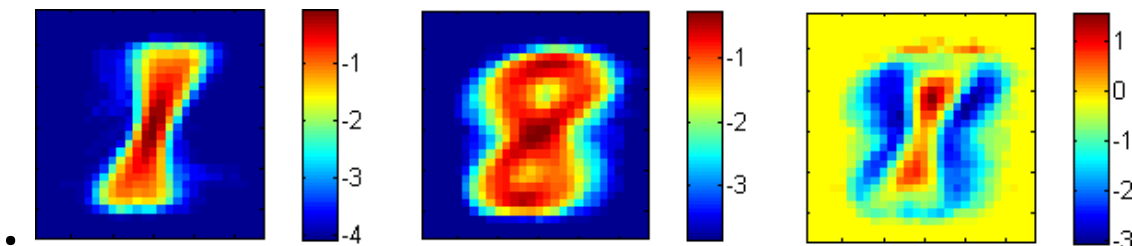
$$\log P(\text{class}) + \log P(f_{1,1} \mid \text{class}) + \log P(f_{1,2} \mid \text{class}) + \dots + \log P(f_{28,28} \mid \text{class})$$

After you compute the above decision function values for all ten classes for every test image, you will use them for MAP classification.

- **Evaluation:** Use the true class labels of the test images from the `testLabels` file to check the correctness of the estimated label for each test digit. Report your performance in terms of the **classification rate for each digit** (percentage of all test images of a given digit correctly classified). Also report your **confusion matrix**. This is a 10x10 matrix whose entry in row  $r$  and column  $c$  is the percentage of test images from class  $r$  that are classified as class  $c$ . In addition, for each digit class, show the test examples from that class that have the highest and the lowest posterior probabilities according to your classifier. You can think of these as the most and least "prototypical" instances of each digit class (and the least "prototypical" one is probably misclassified).

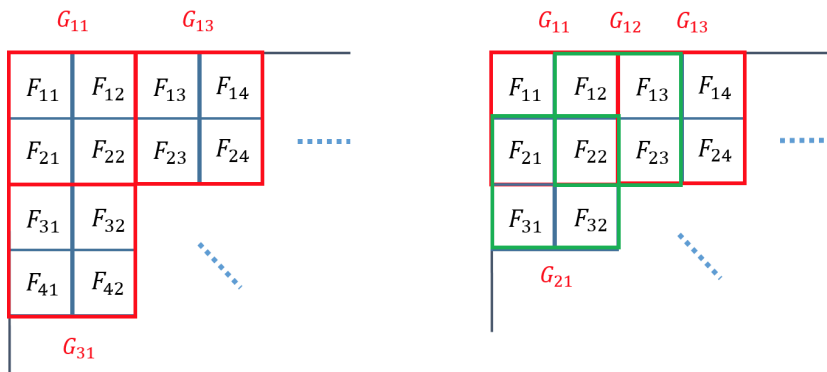
**Important:** The ground truth labels of test images should be used *only* to evaluate classification accuracy. They should not be used in any way during the decision process.

**Tip:** You should be able to achieve at least 70% accuracy on the test set. One "warning sign" that you have a bug in your implementation is if some digit gets 100% or 0% classification accuracy (that is, your system either labels all the test images as the same class, or never wants to label any test images as some particular class).



## Part 1.2 (for four-unit students): Pixel groups as features

Credit: Yanglei Song



In this part, you should build Naive Bayes classifiers for feature sets of  $n \times m$  disjoint/overlapping pixel patches and report the following:

- Test set accuracies for disjoint patches of size  $2 \times 2, 2 \times 4, 4 \times 2, 4 \times 4$ .
- Test set accuracies for overlapping patches of size  $2 \times 2, 2 \times 4, 4 \times 2, 4 \times 4, 2 \times 3, 3 \times 2, 3 \times 3$ .
- Discussion of the trends you have observed for the different feature sets (including single pixels), in particular, why certain features work better than others for this task.
- Brief discussion of running time for training and testing for the different feature sets (which ones are faster and why, and how does the running time scale with feature set size).

**Tip:** You should be able to achieve over 80% accuracy with your best feature set.

## Part 1 Extra Credit

- Experiment with yet more features to improve the accuracy of the Naive Bayes model. For example, instead of using binary pixel values, implement ternary features.
- Apply your Naive Bayes classifier with various features to this [face data](#). It is in a similar format to that of the digit data, and contains training and test images and binary labels, where 0 corresponds to 'non-face' and 1 corresponds to 'face'. The images themselves are higher-resolution than the digit images, and each pixel value is either '#', corresponding to an edge being found at that location, or '.', corresponding to a non-edge pixel.

## Part 2: Text Document Classification

Created by Xuesong Yang based on the corpora of [Fisher English Transcripts Part 1](#), [Part 2](#), and [Movie Review Data](#).

### Part 2.1: For everybody

The goal of this part aims to apply Naive Bayes classifiers to solve basic binary classification problems in the area of natural language processing, e.g., sentiment analysis of movie reviews, conversation topic identification. You need to implement two models on two separate corpora as shown in the following:

**Models:** For each dataset, you will build two different Naive Bayes classifiers--one based on a multinomial distribution and one based on a Bernoulli distribution. Parameters in these two models are estimated on the training data, and the performance of each classifier should be tested on the withheld test data. In other words, the class labels of the test documents should be predicted based on your well-trained models. You should implement the following two Naive Bayes models:

- **Multinomial Naive Bayes:** This is the model described in [the lecture](#), where a document of length  $n$  has variables  $w_1, \dots, w_N$  and each variable  $w_i$  takes on values from 1 to  $v$ , where  $v$  is the size of the vocabulary. You can estimate the likelihoods  $P(w_i = k \mid \text{class})$  by frequency counts, as explained in the lecture.
- **Bernoulli Naive Bayes:** In this model, every document is described by  $v$  binary variables  $w_1, \dots, w_v$ , and  $w_i = 1$  if the  $i$ -th word appears *at least once* in the document, and 0 otherwise. You can estimate the likelihoods  $P(w_i = 1$

| class) as the proportion of documents from that class that feature the  $i$ th word.

### [Implementation Hints]

1. create dictionaries consisting of all unique words occurring in the training documents.
2. estimate conditional probability tables over these dictionaries for each class. Be sure to use Laplace smoothing. Note that there will be words in the test documents that do not occur in the dictionary; simply ignore those.
3. conditional probability may not be the only parameters to estimate. To obtain a better performance, you may also think about how to estimate the prior distribution of class labels, e.g. are they uniformly distributed or some else?

**Preprocessed Corpora:** Train and test data are both preprocessed. Words that carry little meaning, like "the" and "and", have been removed (these are called "stop words"); the remaining words are sorted, counted, and presented in the following format:

```
label word_1:count_1 word_2:count_2 ... word_n:count_n
```

where `label` denotes the target topic label, `word_n:count_n` pair denotes the form of a word, and the number of times that it occurred in the document. The `word:count` features are each separated by a single space. You will implement each classifier for each of the two following corpora:

- **Sentiment analysis of movie reviews:** This corpus has 4000 training documents (2000 positive and 2000 negative), and 1000 test documents (500 positive and 500 negative), where -1 denotes a negative review, and 1 denotes a positive review.
- **Binary conversation topic classification:** This corpus is generated from transcripts of two-person telephone conversations. Before the start of each conversation, two participants are assigned with a specific topic, for example, "Minimum Wage", "Reality TV Shows", "Pets", etc. In this binary classification task, the corpus has been filtered so that it only contains conversations on two topics: "Minimum Wage" and "Life Partners". The corpus has 878 training conversations (440 for Minimum Wage, 438 for Life Partners), and 98 test conversations (49 for Minimum Wage, 49 for Life Partners). For the target labels, -1 denotes Life Partners, and 1 denotes a topic relevant to Minimum Wage.

**Report:** For each of the two models on two datasets, please report the following:

- The overall accuracy and confusion matrix as your model performance on the test data. You should be able to achieve at least 87% accuracy on the topical theme classification task for both models, and at least 70% accuracy on sentiment analysis of movie review task.
- Scores of precision and recall for each class on the test data.

**Precision** = # correct predictions for one class / # all predictions (including both correct and wrong predictions) for that class

**Recall** = # correct predictions for one class / # all target labels for that class

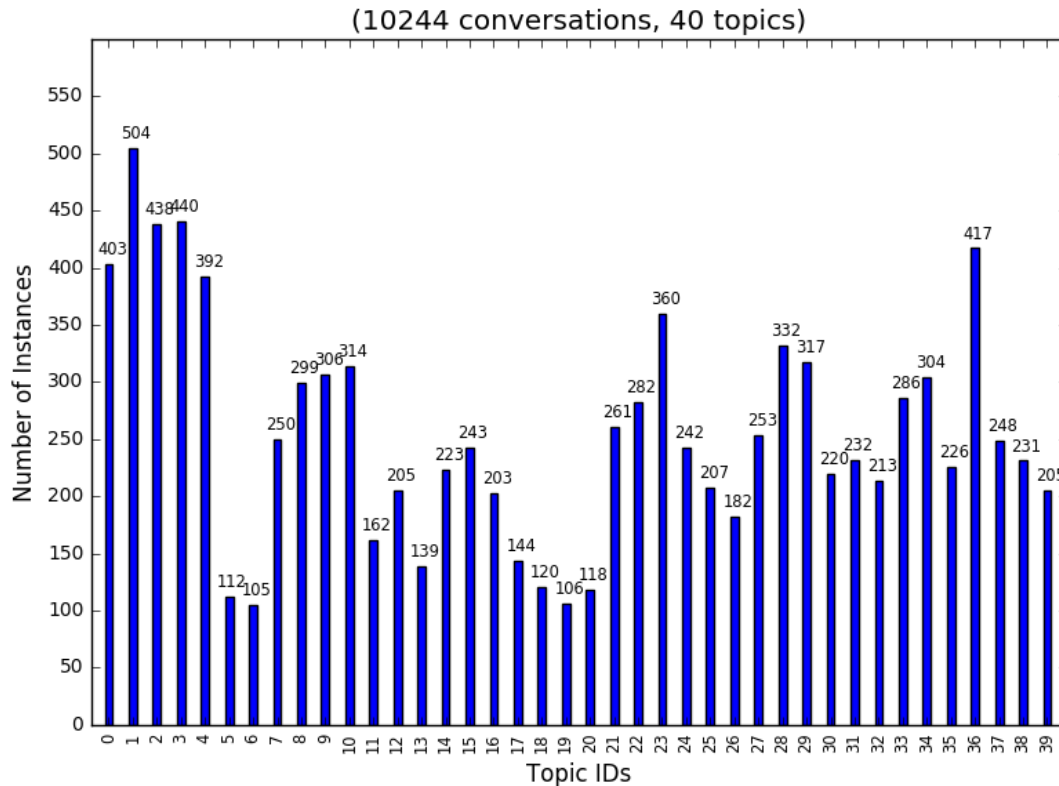
- The top 20 words with the highest likelihood for each class.

## Part 2.2: For four-credit students

Besides the experiments in Part 2.1, you should continue working with conversation topic classification using the [full 40-topic corpus](#). This data contains 40 topic themes as shown in [topic description](#), and has 10244 training conversations, 1154 test conversations. The histograms of class labels on training data is shown in the following. For 40 topic classification task, you should achieve at least 80% accuracy using Multinomial Naive Bayes, and at least 58% accuracy using Bernoulli Naive Bayes.

**Report:** For each of the two models, please report the following:

- Overall accuracy score and confusion matrix of your classifier.
- Precision and recall for each topic separately (40 different precision measurements, 40 different recall measurements).
- For each topic: with what other topic was it most commonly confused?



## Extra Credit for Part 2

- Experiment with advanced techniques for improving performance of Naive Bayes on this dataset, such as lemmatization and tf-idf weighting (not covered in class).
- Experiments using other distribution for conditional probability distribution assumption.
- Visualize the bag-of-words representations of the documents using word cloud maps.
- Visualize the confusion matrix of your results using heatmaps for example.

## Report Checklist

### Part 1:

1. For everybody:
  - Briefly discuss your implementation, especially the choice of the smoothing constant.
  - Report classification rate for each digit and confusion matrix.
  - For each digit, show the test examples from that class that have the highest and lowest posterior probabilities according to your classifier.
  - Take four pairs of digits that have the highest confusion rates, and for each pair, display feature likelihoods and odds ratio.
2. For four-unit students:
  - Report test set accuracies for disjoint patches of size  $2 \times 2$ ,  $2 \times 4$ ,  $4 \times 2$ ,  $4 \times 4$ , and for overlapping patches of size  $2 \times 2$ ,  $2 \times 4$ ,  $4 \times 2$ ,  $4 \times 4$ ,  $2 \times 3$ ,  $3 \times 2$ ,  $3 \times 3$ .
  - Discuss trends for the different feature sets.
  - Discuss training and testing running time for different feature sets.

### Part 2:

## 1. For everybody:

- For multinomial and Bernoulli models, and for both datasets, report the classification rate for each class and show the confusion matrix.
- Precision and recall for each class on the test data.
- For each class, give the top 20 words with the highest likelihood.

## 2. For four-unit students:

- report the same items as for Part 2.1.

**Extra credit:**

- We reserve the right to give bonus points for any advanced exploration or especially challenging or creative solutions that you implement. Three-unit students always get extra credit for submitting solutions to four-unit problems. If you submit any work for bonus points, be sure it is clearly indicated in your report.

**Statement of individual contribution:**

- All group reports need to include a brief summary of which group member was responsible for which parts of the solution and submitted material. We reserve the right to contact group members individually to verify this information.

**WARNING:** *You will not get credit for any solutions that you have obtained, but not included in your report!* For example, if your code prints out path cost and number of nodes expanded on each input, but you do not put down the actual numbers in your report, or if you include pictures/files of your output solutions in the zip file but not in your PDF. The only exception is animated paths (videos or animated gifs).

**Submission Instructions**

As before, **one designated person from the group** will need to submit on [Compass 2g](#) by the deadline. Three-unit students must upload under **Assignment 3 (three units)** and four-unit students must upload under **Assignment 3 (four units)**. Each submission must consist of the following two attachments:

1. A **report** in **PDF format**. As before, the report should briefly describe your implemented solution and fully answer all the questions posed above. **Remember: you will not get credit for any solutions you have obtained, but not included in the report.**

All group reports need to include a brief **statement of individual contribution**, i.e., which group member was responsible for which parts of the solution and submitted material.

The name of the report file should be **lastname\_firstname\_assignment3.pdf**. Don't forget to include the names of all group members and the number of credit units at the top of the report.

2. Your **source code** compressed to a **single ZIP file**. The code should be well commented, and it should be easy to see the correspondence between what's in the code and what's in the report. You don't need to include executables or various supporting files (e.g., utility libraries) whose content is irrelevant to the assignment. If we find it necessary to run your code in order to evaluate your solution, we will get in touch with you.

The name of the code archive should be **lastname\_firstname\_assignment3.zip**.

Multiple attempts will be allowed but in most circumstances, only the last submission will be graded. **We reserve the right to take off points for not following directions.**

**Late policy:** For every day that your assignment is late, your score gets multiplied by 0.75. The penalty gets saturated after four days, that is, you can still get up to about 32% of the original points by turning in the assignment at all. If you have a compelling reason for not being able to submit the assignment on time and would like to make a special arrangement, you

must send me email **at least a week before the due date** (any genuine emergency situations will be handled on an individual basis).

Be sure to also refer to [course policies](#) on academic integrity, etc.