HOMEWORK 7: GRAPHICAL MODELS

10-301/10-601 Introduction to Machine Learning (Spring 2021)
https://www.cs.cmu.edu/~10601/
DUE: Friday, April 30, 2021 11:59 PM

Summary In this assignment you will go through exercises on MAP/MLE and learning graphical models with and without missing data. Finally, you will implement Gaussian Naive Bayes to predict a word category given the real-valued voxels of a human fMRI.

START HERE: Instructions

- **Collaboration Policy**: Please read the collaboration policy here: https://www.cs.cmu.edu/~10601

- **Late Submission Policy:** See the late submission policy here: https://www.cs.cmu.edu/~10601

- **Submitting your work:** You will use Gradescope to submit answers to all questions and code. Please follow instructions at the end of this PDF to correctly submit all your code to Gradescope.

    - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. Submissions must be written in LaTeX. Each derivation/proof should be completed in the boxes provided. If you do not follow the template, your assignment may not be graded correctly by our AI assisted grader.

    - **Programming:** You will submit your code for programming questions on the homework to Gradescope (https://gradescope.com). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.6.9, OpenJDK 11.0.5, g++ 7.4.0) and versions of permitted libraries (e.g. `numpy` 1.17.0 and `scipy` 1.4.1) match those used on Gradescope. You have unlimited Gradescope programming submissions. However, we recommend debugging your implementation on your local machine (or the Linux servers) and making sure your code is running correctly first before submitting you code to Gradescope.

- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on Piazza.

Linear Algebra Libraries When implementing machine learning algorithms, it is often convenient to have a linear algebra library at your disposal. In this assignment, Java users may use EJML[a] or ND4J[b] and C++ users Eigen[c]. Details below. (As usual, Python users have NumPy.)

EJML for Java EJML is a pure Java linear algebra package with three interfaces. We strongly recommend using the SimpleMatrix interface. The autograder will use EJML version 0.38. When compiling and running your code, we will add the additional command line argument `-cp "linalg_lib/ejml-v0.38-libs/*:linalg_lib/nd4j-v1.0.0-beta7-libs/*:./"` to ensure that all the EJML jars are on the classpath as well as your code.

ND4J for Java ND4J is a library for multidimensional tensors with an interface akin to Python's NumPy. The autograder will use ND4J version 1.0.0-beta7. When compiling and running your code, we will add the additional command line argument `-cp "linalg_lib/ejml-v0.38-libs/*:linalg_lib/nd4j-v1.0.0-beta7-libs/*:./"` to ensure that all the ND4J jars are on the classpath as well as your code.

Eigen for C++ Eigen is a header-only library, so there is no linking to worry about—just `#include` whatever components you need. The autograder will use Eigen version 3.3.7. The command line arguments above demonstrate how we will call you code. When compiling your code we will include, the argument `-I./linalg_lib` in order to include the `linalg_lib/Eigen` subdirectory, which contains all the headers.

We have included the correct versions of EJML/ND4J/Eigen in the `linalg_lib.zip` posted on the Piazza Resources page for your convenience. It contains the same `linalg_lib/` directory that we will include in the current working directory when running your tests. Do **not** include EJML, ND4J, or Eigen in your homework submission; the autograder will ensure that they are in place.

---

[a] https://ejml.org
[b] https://deeplearning4j.org/docs/latest/nd4j-overview
[c] http://eigen.tuxfamily.org/

Written Questions (60 points)

1 Short Questions

X_1	X_2	X_3	Probability
0	0	0	0.15
1	0	0	0.05
0	1	0	0.15
1	1	0	0.05
0	0	1	0.1
1	0	1	0.3
0	1	1	0.05
1	1	1	0.15

Table 1: Joint Probability Table

1. (2 points) What's the value of $P(X_1 = 1)$?

> Your answer:
>
> 0.55

2. (2 points) What's the value of $P(X_1 = 0|X_3 = 1)$?

> Your answer:
>
> 0.25

3. (3 points) Is $(X_1 \perp\!\!\!\perp X_2)|X_3$? (i.e., is X_1 conditionally independent of X_2 given X_3?)

   Select one:

   ○ True

   √ **False**

4. (3 points) Is $X_1 \perp\!\!\!\perp X_2$?

   Select one:

   ○ True

   √ **False**

5. (4 points) Consider the plot which shows MLE and MAP estimates of θ, the probability of a particular coin coming up heads, as the number of coin flips grows. This plot is taken from the reading available at http://www.cs.cmu.edu/~tom/mlbook/Joint_MLE_MAP.pdf

In this plot, the true probability is $\theta = 0.3$. Imagine that you plot the same figure, but in a new setting where the true value of θ is 0.25 instead of 0.3. Assume you use the same MAP priors as in the current plot (e.g., $\gamma_0 = 36, \gamma_1 = 24$).

true θ=0.30, γ_0=36, γ_1=24
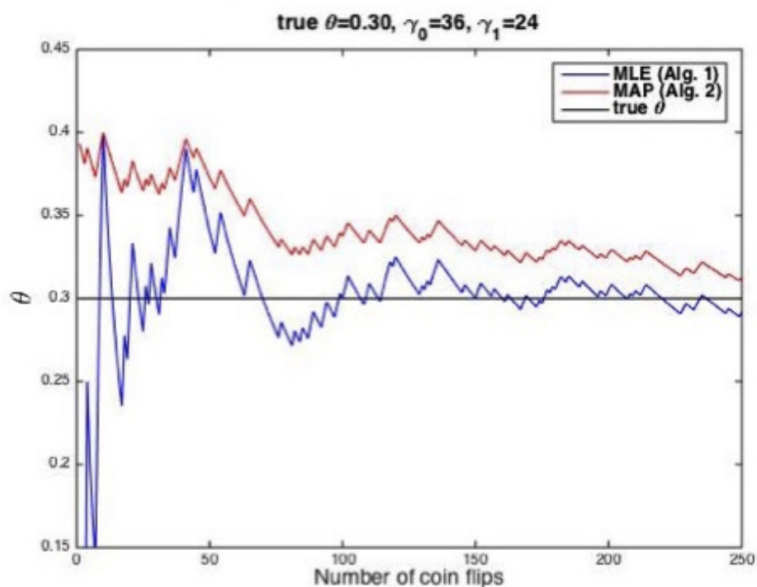
Will the red line change? Will the blue line change? If so, how? Will the distance between these two lines change? Will the starting or ending points of these lines move up or down? If so, how?

Your answer:

The red line will change. The blue line will also change. Both the line will come down. The distance between Red line and blue line will increase slightly as the MAP prior is more inaccurate than before. The end point of the red line will move little down. The end point of the blue line will move more down than the red line.

2 Graphical Models: Representations

Consider the graphical model below over 5 boolean random variables:
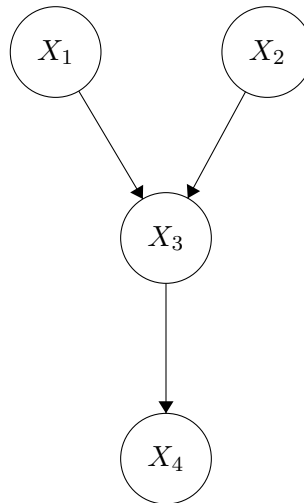
We also have the associated conditional probability tables (as an example the top left element of table 3 reads as $P(X_3 = 0|X_1 = 0, X_2 = 0) = 0.4$):

$X_1 = 0$	0.3
$X_1 = 1$	0.7

$X_2 = 0$	0.5
$X_2 = 1$	0.5

	$X_1 = 0, X_2 = 0$	$X_1 = 0, X_2 = 1$	$X_1 = 1, X_2 = 0$	$X_1 = 1, X_2 = 1$
$X_3 = 0$	0.4	0.7	0.8	0.5
$X_3 = 1$	0.6	0.3	0.2	0.5

	$X_3 = 0$	$X_3 = 1$
$X_4 = 0$	0.8	0.25
$X_4 = 1$	0.2	0.75

Table 2: Conditional Probability tables

In this section, we will test your understanding of several aspects of directed graphical models. For each question below, either write your answer as a fraction or write your answer to 5 decimal places (if needed).

1. (2 points) What is $P(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0)$?

> Your Answer:
>
> 0.0175

2. (2 points) What is the value of $P(X_1 = 1)$?

> Your Answer:
>
> 0.7

3. (2 points) What is the value of $P(X_4 = 1)$?

> Your Answer:
>
> 0.394

4. (2 points) What is $P(X_1 = 1, X_2 = 1, X_4 = 1)$?

> Your Answer:
>
> 0.16625

5. (2 points) What is $P(X_2 = 1 | X_4 = 1, X_3 = 0)$?

> Your Answer:
>
> 0.45161

6. (3 points) $(X_1 \perp\!\!\!\perp X_2)|X_3$

  ○ True

  $\checkmark$ **False**

7. (3 points) $(X_1 \perp\!\!\!\perp X_4)|X_3$

  $\checkmark$ **True**

  ○ False

8. (2 points) What is the minimum number of parameters we must estimate in order to learn this graphical model?
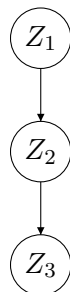
> Your Answer:
>
> 8

9. (2 points) If we made no assumptions about dependencies among random variables $X_1, X_2, X_3,$ and X_4 how many parameters would we need to estimate?

> Your Answer:
>
> 15

10. (2 points) Given random variables as Z_1, Z_2, Z_3, write a graphical model that reflects the following conditional independence assumption: $(Z_1 \perp\!\!\!\perp Z_3)|Z_2$. You need only to draw the corresponding DAG of the model. You do not need to give the parameters of the model.

> Your Answer:
>
> Z_1
>
> $\downarrow$
>
> Z_2
>
> $\downarrow$
>
> Z_3

11. (2 points) Given random variables as Z_1, Z_2, Z_3, write a graphical model that reflects NO conditional independencies among the variables. You may write the joint distribution of the model, or draw the corresponding DAG of the model.
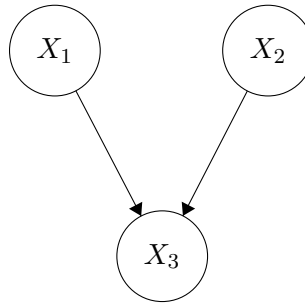
Your Answer:

$$P(Z_1, Z_2, Z_3) = P(Z_3|Z_2, Z_1) * P(Z_2|Z_1) * P(Z_1)$$

3 Graphical Models: Learning Models

3.1 MLE and MAP

Now consider the following graphical model:

We have the following observed data:

X_1	X_2	X_3
1	0	1
0	0	0
1	1	0
1	1	1
0	1	0
1	0	0
1	0	0
1	1	1
0	0	1
0	0	1

1. (2 points) Given the data and the graphical model above, we would like to learn the parameters of the model using Maximum Likelihood Estimation. Write the Conditional Probability Distribution associated with X_3, use MLE estimates of the parameters based on this data.

> **Your Answer:**
>
> | X_1 | X_2 | $P(X_3 = 0|X_1, X_2)$ | $P(X_3 = 1|X_1, X_2)$ |
> |---|---|---|---|
> | 0 | 0 | 1/3 | 2/3 |
> | 0 | 1 | 1/1 | 0/1 |
> | 1 | 0 | 2/3 | 1/3 |
> | 1 | 1 | 1/3 | 2/3 |

2. (2 points) Now write a second set of parameter values, again for the Conditional Probability Distribution associated with X_3, but this time use MAP estimates with a Beta(2,2) prior.

> **Your Answer:**
>
> | X_1 | X_2 | $P(X_3 = 0 \mid X_1, X_2)$ | $P(X_3 = 1 \mid X_1, X_2)$ |
> |---|---|---|---|
> | 0 | 0 | 2/5 | 3/5 |
> | 0 | 1 | 2/3 | 1/3 |
> | 1 | 0 | 3/5 | 2/5 |
> | 1 | 1 | 2/5 | 3/5 |

3.2 EM: Learning with Missing Data

Now suppose we have the same graphical model, but the data is sometimes missing the value of X_3. This gives the following data:

X_1	X_2	X_3
1	0	*
0	0	*
1	1	0
1	1	1
0	1	0
1	0	0
1	0	0
1	1	1
0	0	1
0	0	1

There are now several parameters we cannot estimate directly since X_3 is missing:

$P(X_3 = 1 \mid X_1 = 0, X_2 = 0)$
$P(X_3 = 1 \mid X_1 = 1, X_2 = 0)$

To handle this missing data, we would like to use the EM algorithm as follows for boolean data:

1. E-step: For each row (sample) x_n that contains a missing value, use the observed features of row $\mathbf{x}_n$ and the current parameters θ to calculate $\mathbb{E}(\mathbf{z}_n \mid \mathbf{x}_n), \theta$, where $\mathbf{z}_n$ is the missing value(s) in that row.

2. M-step: Re-estimate the parameters θ in a similar procedure to MLE on the fully observed data, but instead of counts of the unobserved variable use expected counts.

3. Iterate until convergence, i.e. model likelihood has converged.

More explicitly in the boolean case for the model given, in the E-step for a given sample n we calculate

$\mathbb{E}(X_{n,3}) = \mathbb{E}(X_{n,3}|x_{n,1}, x_{n,2}, \theta)$, where $x_{n,i}$ denotes the ith variable in the nth sample.

Then, in the M-step, we re-estimate the parameters θ with the expected counts:

$$\theta_{x_3|i,j} = \frac{\sum_{n=1}^{N} I(x_{n,1} = i, x_{n,2} = j) * \mathbb{E}(x_{n,3})}{\sum_{n=1}^{N} I(x_{n,1} = i, x_{n,2} = j)}$$

where $\theta_{x_3|i,j} = P(x_3 = 1|x_1 = i, x_2 = j)$. Here $I(a)$ an "indicator" function, whose value is 1 if a is true, and 0 otherwise.

1. (2 points) Execute the first E-step of the EM algorithm. More precisely, assume we initialize each unknown parameter to 0.5, and other parameters to their MLE estimates. Give the expectations of the missing X_3 variables for row 1 and for row 2 in the data:

$\mathbb{E}(X_{1,3}|x_{1,1}, x_{1,2}, \theta)$:

> Your Answer:
>
> 0.5

$\mathbb{E}(X_{2,3}|x_{2,1}, x_{2,2}, \theta)$:

> Your Answer:
>
> 0.5

2. (6 points) Now execute the first M-step. List the estimated values of the unknown model parameters we obtain in this M-step. (Note that we use the expected count only when the variable is unobserved in an example).

θ_{x_1}:

> Your Answer:
>
> 0.6

θ_{x_2}:

> Your Answer:
>
> 0.4

$\theta_{x_3|0,0}$:

> Your Answer:
>
> $\frac{5}{6}$

$\theta_{x_3|1,1}$:

> Your Answer:
>
> $\frac{2}{3}$

$\theta_{x_3|0,1}$:

> Your Answer:
>
> 0

$\theta_{x_3|1,0}$:

> **Your Answer:**
>
> $\frac{1}{6}$

3. (2 points) Last, let's simulate the second E-step. List the actual values for all the expectations we calculate in this E-step.

$\mathbb{E}(X_{1,3}|x_{1,1}, x_{1,2}, \theta)$:

> **Your Answer:**
>
> $\frac{1}{6}$

$\mathbb{E}(X_{2,3}|x_{2,1}, x_{2,2}, \theta)$:

> **Your Answer:**
>
> $\frac{5}{6}$

4 Programming Empirical Questions

The following questions should be completed as you work through the programming component of this assignment.

1. (3 points) Using the data provided, plot the test accuracy (vertical axis) of the classifier versus the number of training examples used (horizontal axis) when using all 21,764 voxels. The data consists of 42 samples, and for each $i \in [10, ..., 42]$, plot the test accuracy of the classifier after training on the first i samples in the train dataset.
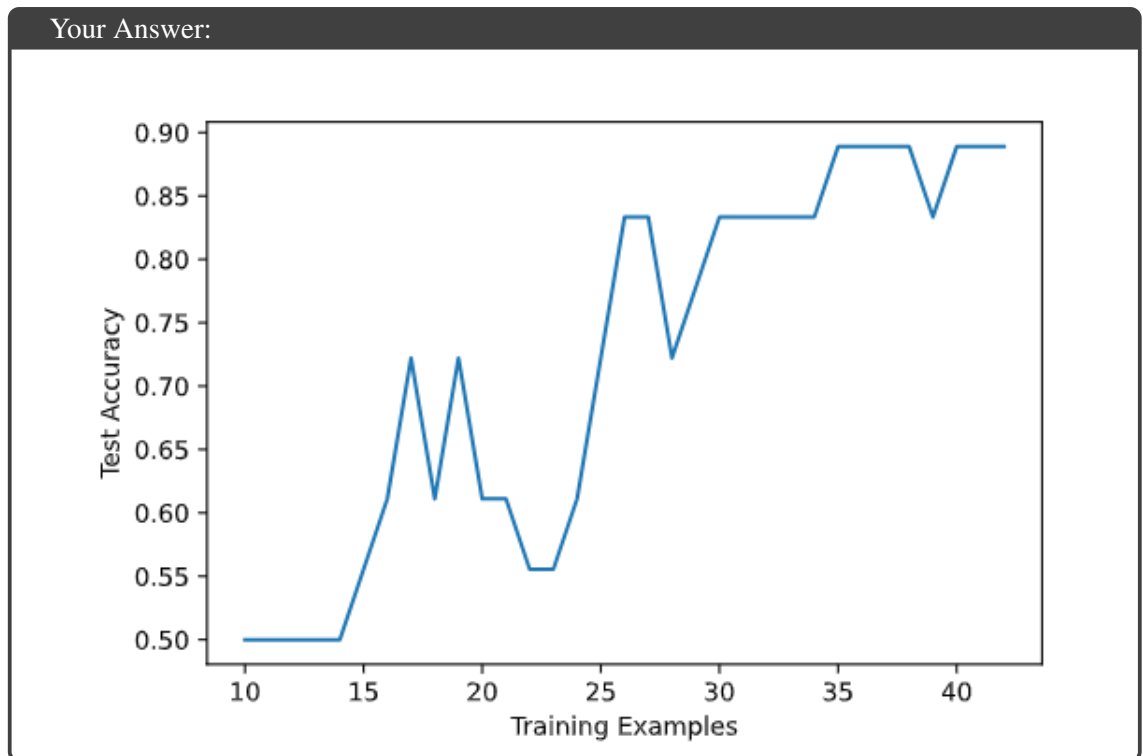
2. (3 points) Using the data provided, plot the test accuracy of your classifier (vertical axis) when training on all available training data, versus the top k number of features you select (horizontal axis). Instead of plotting a point for each of the 21,764 values, plot a point for every 200 voxels starting at 50, so k is in the set $[50, 250, 450, ..., 21650]$.

Your Answer:
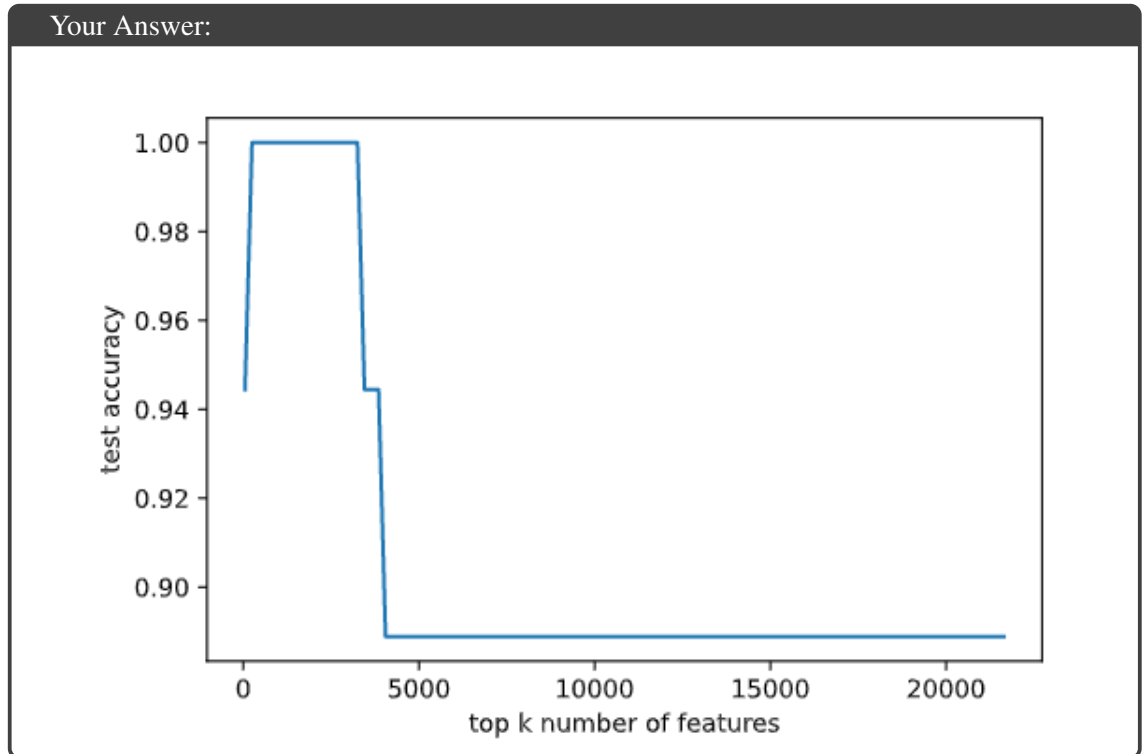
3. (2 points) Using the visualization tool, submit two slices of your choice, one for the building class and one for the tool class, that show a visual difference in the neural activation patterns. Comment briefly on your observations. See 12 for details on generating the output.

Your Answer:

Programming (40 points)

5 The Task

In this task, you will be using Gaussian Naive Bayes to predict the word stimulus given to a human subject based on their observed neural activity measured by functional magnetic resonance imaging (fMRI).

Studies have shown that thinking about different semantic categories of words (for example, tools, buildings, and animals) activates different spatial patterns of neural activation in the brain. A study conducted in 2008 generated a computational model which predicts the fMRI neural activation associated with thinking about arbitrary concrete nouns.

In this homework you will use the data from that study for a different purpose: to train a Naive Bayes classifier to predict which category of words (e.g., tools, buildings, vehicles) the person is reading, based on their fMRI neural image. For this assignment you will only be distinguishing between two categories: tools and buildings.

The programs you write will be automatically graded using the Gradescope system. You may write your programs in **Python, Java, or C++**. However, you should use the same language for all parts below.

6 The Datasets

Datasets The fMRI dataset used in this experiment was originally collected by Marcel Just and his colleagues in Carnegie Mellon University's CCBI. Seventeen 5-mm thick oblique-axial slices were imaged with a gap of 1-mm between slices, normalized, and resampled to 3x3x6 mm3 voxels. Thus, each fMRI image in the dataset is represented as 21,764 voxels (3D pixels).

File Format The dataset consists of two files: train_data.csv and test_data.csv. Each row in the file represents one example: a brain image, and the label to be predicted (the category of the word the person was reading when this brain image was collected). Each row contains 21,765 columns separated by commas. Columns 1 through 21,764 represent the voxels making up the fMRI image. The last column in each row contains the label, which can be one of two categories: "tool" or "building".

7 Model Definition

To predict the word stimulus category given the fMRI voxels, you will use a Gaussain Naive Bayes model. Whereas in class we discussed Bernoulli Naive Bayes when the input features were binary, here we are using Gaussian Naive Bayes since the voxel values for each feature are continuous.

Formally, if we have input features $\mathbf{x} \in \mathbb{R}^M$ and labels $y \in \{0, 1\}$, the "Gaussian" aspect of Gaussian Naive Bayes assumes that

$$P(x_m|y = k) \sim \mathcal{N}(\mu_{m,k}, \sigma_{m,k}) \quad \forall m \in [1, ..., M], k \in \{0, 1\}$$

i.e. the probability of a feature m given the class k comes from a normal distribution with parameters $\mu_{m,k}, \sigma_{m,k}$.

So, we have

$$P(X_m = v|y = k) = \frac{1}{\sqrt{2\pi\sigma_{m,k}^2}} \exp\left(\frac{-(v - \mu_{m,k})^2}{2\sigma_{m,k}^2}\right)$$

which is the Gaussian PDF parameterized by $\mu_{m,k}$ and $\sigma_{m,k}$.

Under this Gaussian assumption along with the Naive Bayes assumption, we have the following generative model for the data with binary labels and continuous features:

$$p(\mathbf{x}, y) = p(y) \prod_{m=1}^{M} p(x_m|y)$$

where $p(x_m|y)$ is a normal distribution as defined above.

Classification:

After learning a generative model, we would like to be able to perform classification for our binary labelling task. In order to do so for input $\mathbf{x}$ we compute our prediction

$$\hat{y} = \underset{y}{\operatorname{argmax}} \, p(y|\mathbf{x})$$

.

8 MLE Estimation

To learn the model, we would first like to use MLE to estimate the model parameters. Assume we have a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

To estimate $P(Y = 1)$ using MLE, since Y is a binary value we end up with

$$P(Y = 1) = \frac{\#D\{Y = 1\}}{|D|}$$

where $\#D\{Y = 1\}$ is the count of items in dataset D where $Y = 1$.

We also need to estimate $P(X_m = v|Y = k)$ for each feature i. To do so, for a given class and since each feature is continuous, we need to estimate a class-conditional Gaussian distribution with parameters μ and σ as explained in 7

Thus, for feature m and class k and, we estimate $\hat{\mu}_{m,k}$ as

$$\frac{1}{\sum_{i=1}^{N} I(Y^{(i)} = k)} \sum_{i=1}^{N} \mathbf{x}_m^{(i)} * I(Y^{(i)} = k)$$

We also estimate $\hat{\sigma}_{m,k}^2$ as

$$\frac{1}{\sum_{i=1}^{N} I(Y^{(i)} = k)} \sum_{i=1}^{N} (\mathbf{x}_m^{(i)} - \hat{\mu}_{m,k})^2 * I(Y^{(i)} = k)$$

9 Log-Space Arithmetic

In computing $P(Y|X)$ via Bayes' Rule, we end up with

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \propto P(X|Y)P(Y)$$

. If X is a vector of size M where M is very large, we end up with a large product of probabilities:

$$\prod_{i=1}^{M} P(X_i|Y)P(Y)$$

.

To avoid underflow problems (reaching an incorrect value of 0 when multiplying many probabilities that are ≤ 1), we can convert the computation to log-space.

As a result, instead of computing $P(Y|X)$, you should compute

$$\log(P(Y|X)) = \log(P(Y)) + \sum_{i=1}^{M} \log(P(X_i|Y))$$

where we use the natural logarithm base.

10 Feature Selection

After building the Gaussian Naive Bayes classifier, we would like to optimize our performance by extracting only the top k useful features.

This can be done by selecting features whose mean given the label is 'tool' is very different from the mean given the label is 'building'. For example, the 'best' feature would maximize

$$|\mu_{m,tool} - \mu_{m,building}|$$

11 Implementation

Write a program gnb.{py|java|cpp|m} that implements a binary classifier using Gaussian Naive Bayes. Your program should output the predictions and error rates on the train and test datasets with training done on the top k voxels, where k is specified from the command line parameter *num_voxels*.

Note: Because computing the joint probability involves repeated multiplications, you will need to work in *log space* to prevent underflow.

11.1 Command Line Arguments

The autograder runs and evaluates the output from the files generated, using the following command:

For Python:
```
$ python3 gnb.py [args...]
```

For Java:
```
$ javac -cp "./lib/ejml-v0.38-libs/*:./" gnb.java
$ java -cp "./lib/ejml-v0.38-libs/*:./" gnb [args...]
```

For C++:
```
$ g++ -g -std=c++11 -I./lib gnb.cpp; ./a.out [args...]
```

Where above [args...] is a placeholder for six command-line arguments. These arguments are described in detail below:

1. <train_input>: path to the training input .csv file (see Section 6)

2. <test_input>: path to the test input .csv file (see Section 6)

3. <train_out>: path to output .labels file to which the prediction on the *train* data should be written (see Section 11.2)

4. `<test_out>`: path to output `.labels` file to which the prediction on the *test* data should be written (see Section 11.2)

5. `<metrics_out>`: path of the output `.txt` file to which metrics such as train and validation error should be written (see Section 11.3)

6. `<num_voxels>`: an integer denoting that the top `num_voxels` found via the feature selection method described should be used for training the gnb classifer (see Section 10)

As an example, if you implemented your program in Python, the following command line would run your program.

```
$ python3 gnb.py train_data.csv test_data.csv train_out.labels
\ test_out.labels metrics_out.txt 21764
```

11.2 Output: Labels Files

Your program should output a `.labels` file containing the predictions of your model on training data (`<train_out>`) and test data (`<test_out>`) with training done on the top *num_voxels* voxels as specified from the command line input. The file should contain the predicted labels for each example printed on a new line. Use `\n` to create a new line.

Your labels should exactly match those of a reference implementation – this will be checked by the auto-grader by running your program and evaluating your output file against the reference solution.

Note: You should output your predicted labels using the same *string* identifiers as the original training data: either 'tool' or 'building'.

A few lines of sample output is given below.

```
building
building
tool
```

11.3 Output Metrics

Generate the train and test errors from your Gaussian Naive Bayes classifier. You must output a metric file containing the train and test errors as shown below.

A sample output is given below.

```
error(train): 0.000000
error(test): 0.000000
```

Take care that your output has the exact same format as shown above. Each line should be terminated by a Unix line ending `\n`.

12 Visualization

We have provided for you a visualization tool for you to view horizontal slices of the brain from the fMRI scans.

The script takes in two arguments: 1) the path to the dataset of your choice, and 2) a row index into the dataset corresponding to the image you would like to visualize.

Here is an example of how to run the tool:

```
$ python3 visualize.py <path_to_dataset> <row_index>
```

Below is an example of what an output slice should look like. Note the back of the head corresponds to $x = 0$ in this plot, and the front of the head to $x = 59$. The bright yellow (high positive activation) region around $(x = 11, y = 25)$ is part of visual cortex. Dark blue regions indicate lower than average activation. The empty regions inside the brain are the white matter of the brain (the axons, but not the firing cortical regions, so we mask out fMRI activity in these regions).
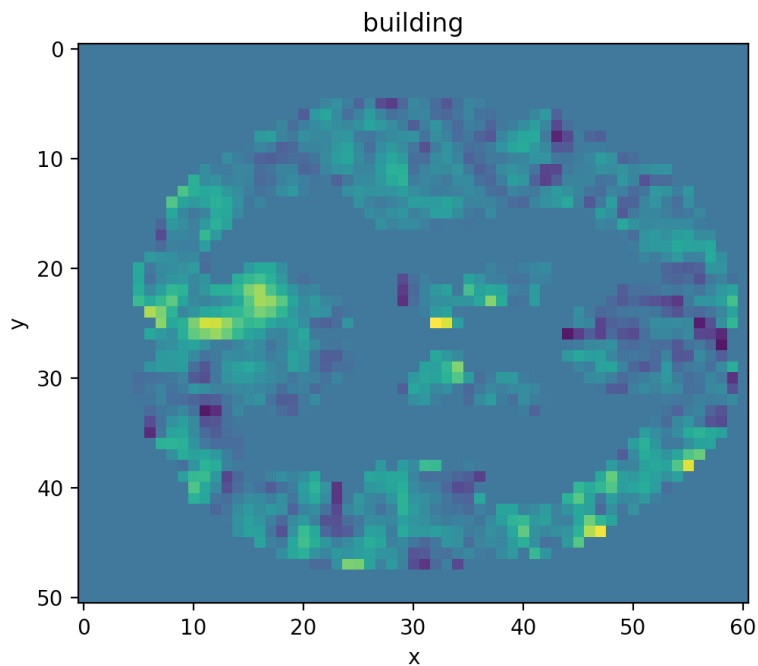
13 Gradescope Submission

You should submit your `gnb.{py|java|cpp}` and a to Gradescope. Please do not use any other file name for your implementation. This will cause problems for the autograder to correctly detect and run your code.