CS 454

Introduction to Machine Learning and Artificial Neural Networks

Prof. Dr. Ethem Alpaydın

Fall 2021

Homework #1

Parametric Classification

by

Abdullah Saydemir

S014646

October 17, 2021

Part 1

1. I calculated the priors, mean, standard deviation using the training dataset using formulas. These variables are calculated once using the training dataset and used also in the testing dataset. Results are shown below.

```
PS C:\Users\abdul\source\CS454> & C:/Users/abdul/AppData/Local/Programs/
Priors    :  0.3333333333333333 0.3333333333333333 0.3333333333333333
Averages  :  24.48 34.12 49.44
Std dev.  :  1.992385504865963 4.1358916813669095 5.091797325110258
```

Figure 1

2. Then, assuming that the data follows Gaussian distribution, I used the Gaussian distribution formula to calculate the likelihood of an instance to be in a class for each class. In other words, $P(X = x_i \mid C = 1)$, $P(X = x_i \mid C = 2)$ and $P(X = x_i \mid C = 3)$ are calculated for $i = 1, 2, \ldots, 150$ where x_i is the i^{th} instance in the dataset. Results for the first five instances are shown below.

```
PS C:\Users\abdul\source\CS454> & C:/Users/abdul/AppData/Local/Programs/Python/Python39/python.exe c:/Users/abdul/sourc
Likelihoods [:5]
[0.19450609952109907, 0.04355790019882406, 0.14967591498900223, 0.08998158135579994, 0.04204847229238516]
[0.004833259363706552, 0.0006297653163912096, 0.014038990858896434, 0.021917966106191084, 0.03227569383782517]
[2.9747607451003595e-07, 1.3171401742376461e-08, 1.9599431638633926e-06, 4.748020843991977e-06, 1.106702023310843e-05]
```

Figure 2

3. In the last step, posteriors are calculated using prior, likelihood and marginal probabilities. Results for the first five instances are shown below.

```
PS C:\Users\abdul\source\CS454> & C:/Users/abdul/AppData/Local/Programs/Python/Python39/python.exe c:/Users/abdul/source
Posteriors [:5]
[0.975752156254195, 0.9857476448975879, 0.9142363816078697, 0.8040940784940842, 0.5656600579604737]
[0.02424635143825334, 0.014252057023804348, 0.08575164685117516, 0.19586349220663365, 0.43419106216434367]
[1.4923075515465512e-06, 2.980786077446889e-07, 1.1971540955131012e-05, 4.2429299282084083e-05, 0.0001488798751826 1264]
```

Figure 3

Above outputs can be printed to the console by running *main.py*. Resulting plots showing the instances, likelihoods and posterior probabilities are shown below. These plots are drawn in another python program, *plot.py*.

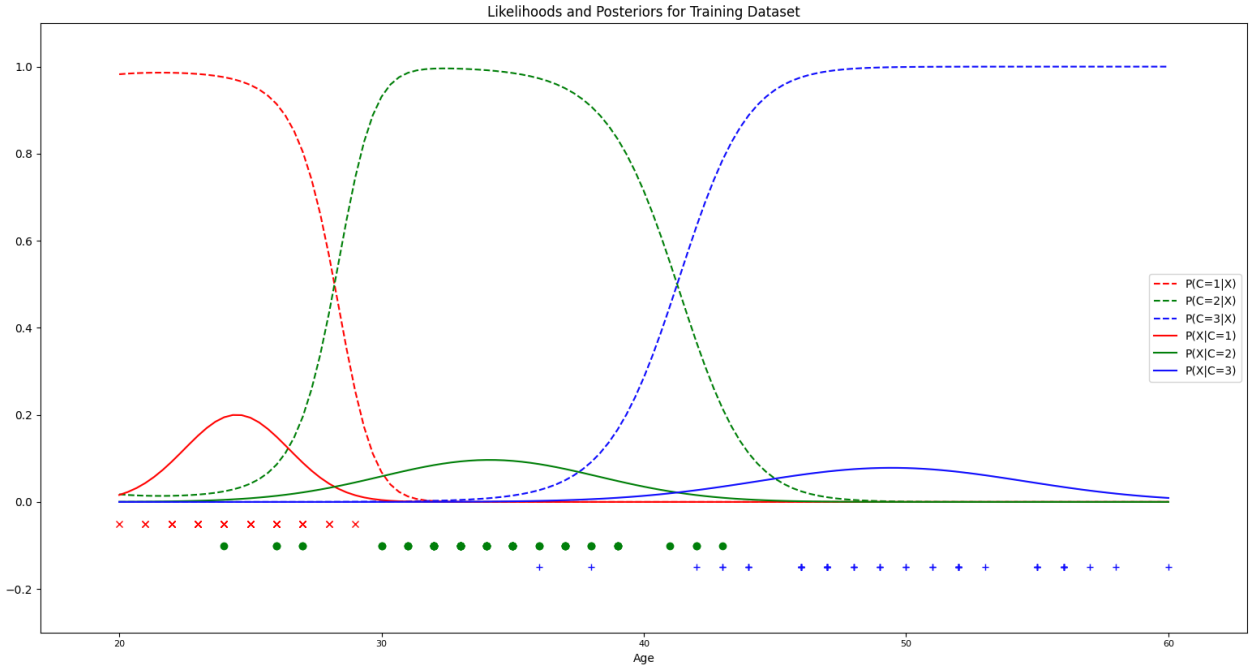

Figure 4

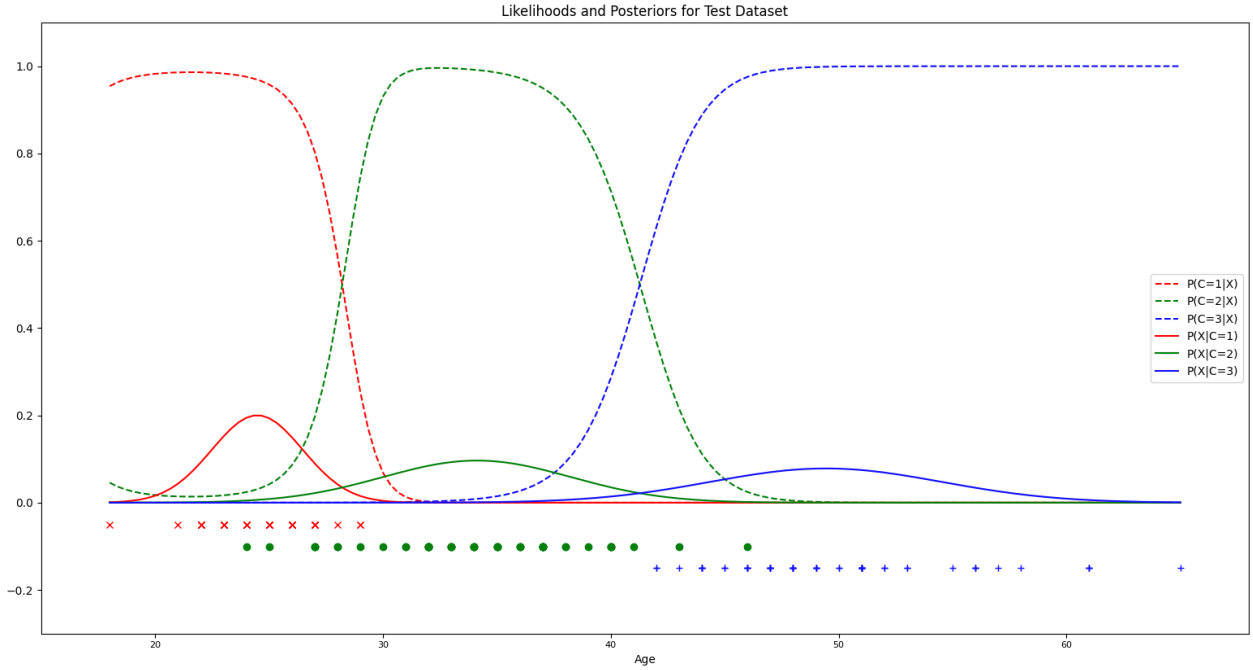

Figure 5

From the graphs, we can see that posterior probabilities are always between zero and one, and summation of the posterior probabilities are equal to one. There are 2 intersections on each graph. These points show the assignments of the instances to the classes. In other words, instances to the left of the first intersection are assigned to class one, instances between two intersections are assigned to class 2 and others are assigned to class 3, assuming that rejection does not occur.

Part 2

Now we have the posteriors, we can use them to assign an instance to the class with the highest posterior.

a) In case of zero/one loss, we just look at the highest posterior and assign the class. Confusion matrices below show the number of correspondences and discrepancies between actual classes and assigned classes.

0-1 Loss Training		Actual		
		C1	C2	C3
Action	C1	49	5	0
	C2	1	42	2
	C3	0	3	48

Figure 6

0-1 Loss Test		Actual		
		C1	C2	C3
Action	C1	48	7	0
	C2	2	41	0
	C3	0	2	50

Figure 7

b) If we can reject predicting a class for an instance, we should calculate the threshold for decision first.

$$R_1 = 0\,x\,P(C=1|X) + 4\,x\,P(C=2|X) + 4\,x\,P(C=3|X) = 4\,(P(C=2|X) + P(C=3|X))$$
$$R_1 = 4\,(1 - P(C=1|X))$$

$$R_2 = 4\,x\,P(C=1|X) + 0\,x\,P(C=2|X) + 4\,x\,P(C=3|X) = 4\,(P(C=1|X) + P(C=3|X))$$
$$R_2 = 4\,(1 - P(C=2|X))$$

$$R_3 = 4\,x\,P(C=1|X) + 4\,x\,P(C=2|X) + 0\,x\,P(C=3|X) = 4\,(P(C=1|X) + P(C=2|X))$$
$$R_3 = 4\,(1 - P(C=3|X))$$

$$R_4 = 1\,x\,P(C=1|X) + 1\,x\,P(C=2|X) + 1\,x\,P(C=3|X) = 1$$

To assign an instance to C1, $R_1 < R_2$, $R_1 < R_3$, and $R_1 < R_4$ must hold true. Then,

$$R_1 < R_2 : 4(1 - P(C = 1|X)) < 4(1 - P(C = 2|X))$$
$$: P(C = 1|X) > P(C = 2|X)$$

$$R_1 < R_3 : 4(1 - P(C = 1|X)) < 4(1 - P(C = 3|X))$$
$$: P(C = 1|X) > P(C = 3|X)$$

$$R_1 < R_4 : 4(1 - P(C = 1|X)) < 1$$
$$: P(C = 1|X) > 3/4$$

Similar equations can be written for other cases (i.e. assigning an instance to C2). However, it is important to note that since the sum of posteriors must be equal to one, if one of them is greater than ¾ then other posteriors are guaranteed to be less than ¾. Therefore, we can assign an instance to a class if the posterior is greater than ¾. If none of the posteriors are greater than ¾, we reject.

Confusion matrices below show the number of correspondences and discrepancies between actual classes and assigned classes.

Reject Training		Actual C1	Actual C2	Actual C3
Action	C1	47	5	0
	C2	0	41	2
	C3	0	1	47
	Reject	3	3	1

Figure 8

Reject Test		Actual C1	Actual C2	Actual C3
Action	C1	47	5	0
	C2	0	35	0
	C3	0	2	48
	Reject	3	8	2

Figure 9

Explanations

1. *We assumed that the distributions of the samples (instances, observations) are Gaussian. When you plot the instances together with the likelihood functions does it seem to fit that distribution?*

   Yes. We can observe from Figure 4 and 5 that we can fit a model to predictions. Likelihood distributions curves are pretty much similar to bell-shaped curves which is a typical curve for Gaussian distribution.

2. *You may have observed that the model made misclassifications during prediction. What can be the cause of that?*

   Because in all cases, we have some outliers who don't obey the generalization. In Figure 4 and 5, we can see that some instances from different classes collide. However, since the posterior of one class is greater than the other, all of the instances are assigned to the same class. Therefore, some of them are misclassified. If the instances of the classes were completely discrete, then the number of misclassifications would be very low. However, we have collisions in our data.

3. *You may have observed some rejected instances during classification in Part 2.b of the homework. Why are those instances rejected?*

   In Part 2.b, we are looking for some *confidence* to take a specific action, i.e. decision threshold. I calculated the decision threshold and it is ¾. If an instance is rejected, that means none of the posteriors are greater than ¾ and we are not confident enough to put that instance to a class. When we are not confident enough, we are trying to avoid having much larger costs by simply rejecting it.

What I learned

- Before the homework, I falsely thought that marginal probability was equal to one. I learned that, instead, the sum of the posteriors is equal to one.
- From confusion matrices, if the threshold for decision is high, then we reject more cases. On the other hand, it lowers the possibility of misclassification.

```
import csv
import math
import matplotlib.pyplot as plt

def read_csv_file(file_name):
    """
    Reads a csv file after second line and returns list of dictionaries with the dat
a. Initial values of predictions are -1
    """
    numbers = []
    with open(file_name, 'r') as csv_file:
        csv_reader = csv.reader(csv_file)
        next(csv_reader)
        for item in list(csv_reader):
            my_dict = { 'value': int(item[0]), 'class': int(item[1]) , 'prediction_z
ero_one' : -1, 'prediction_reject' : -1}
            numbers.append(my_dict)
    return numbers


def calculate_appearances(numbers, class_id):
    """
    Calculates how many times an item that belongs to a class appears in a list
    """
    appear = 0.0
    for dict in numbers:
        if dict['class'] == class_id:
            appear += 1.0
    return appear


def sum_numbers(numbers, class_id = None):
    """
    Sums all the numbers in a list
    """
    sum = 0.0
    if class_id == None:
        for number in numbers:
            sum += float(number)
    else:
        interested_class = [x for x in numbers if x['class'] == class_id]
        for number in interested_class:
            sum += float(number['value'])
    return sum


def my_mean(numbers, class_id = None):
    """
    Calculates the mean of a list of numbers
    """
    if class_id == None:
        return sum_numbers(numbers) / float(len(numbers))
    else:
        return sum_numbers(numbers, class_id) / calculate_appearances(numbers, class
_id)


def standard_deviation(numbers, class_id = None):
    """
    Calculates the standard deviation of a list of numbers
    """
    average = 0.0
    variance = 0.0
    if class_id == None:
        average = my_mean(numbers)
        variance = sum_numbers([float((x['value']) - average) ** 2 for x in numbers]
) / float(len(numbers))
    else:
        average = my_mean(numbers, class_id)
        variance = sum_numbers([(float(x['value']) - average) ** 2 for x in numbers
if x['class'] == class_id]) / calculate_appearances(numbers, class_id)
    return variance ** 0.5
```

```
def calculate_likelihood(number, mean, std_dev):
    """
    Calculates the likelihood
    """
    likelihood = 1.0 / (std_dev * (2.0 * math.pi) ** 0.5) * (math.exp(-((number - me
an) ** 2) / (2.0 * std_dev ** 2)))
    return likelihood


numbers = read_csv_file("training.csv")
numbers_one   = [x['value'] for x in numbers if x['class'] == 1]
numbers_two   = [x['value'] for x in numbers if x['class'] == 2]
numbers_three = [x['value'] for x in numbers if x['class'] == 3]

count = float(len(numbers))

prior_class_1 = calculate_appearances(numbers, 1) / count
prior_class_2 = calculate_appearances(numbers, 2) / count
prior_class_3 = calculate_appearances(numbers, 3) / count

print("Priors    : ",prior_class_1, prior_class_2, prior_class_3)

mean_class_1 = my_mean(numbers, 1)
mean_class_2 = my_mean(numbers, 2)
mean_class_3 = my_mean(numbers, 3)

print("Averages  : ",mean_class_1,mean_class_2,mean_class_3)

std_class_1 = standard_deviation(numbers, 1)
std_class_2 = standard_deviation(numbers, 2)
std_class_3 = standard_deviation(numbers, 3)

print("Std dev.  : ",std_class_1,std_class_2,std_class_3, end='\n\n')

likelihoods_class_1 = [calculate_likelihood(numbers[i]['value'], mean_class_1, std_c
lass_1) for i in range(len(numbers))]
likelihoods_class_2 = [calculate_likelihood(numbers[i]['value'], mean_class_2, std_c
lass_2) for i in range(len(numbers))]
likelihoods_class_3 = [calculate_likelihood(numbers[i]['value'], mean_class_3, std_c
lass_3) for i in range(len(numbers))]

print("Likelihoods [:5]",likelihoods_class_1[:5],likelihoods_class_2[:5],likelihoods
_class_3[:5], sep='\n',end='\n\n')

posterior_ones      = [prior_class_1 * likelihoods_class_1[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]
posterior_twos      = [prior_class_2 * likelihoods_class_2[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]
posterior_threes    = [prior_class_3 * likelihoods_class_3[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]

print("Posteriors [:5]", posterior_ones[:5], posterior_twos[:5], posterior_threes[:5
], sep='\n',end='\n\n')

for i in range(len(numbers)):

    if posterior_ones[i] > posterior_twos[i] and posterior_ones[i] > posterior_three
s[i]:
        numbers[i]['prediction_zero_one'] = 1
    elif posterior_twos[i] > posterior_ones[i] and posterior_twos[i] > posterior_thr
ees[i]:
        numbers[i]['prediction_zero_one'] = 2
    elif posterior_threes[i] > posterior_ones[i] and posterior_threes[i] > posterior
_twos[i]:
        numbers[i]['prediction_zero_one'] = 3

    if posterior_ones[i] > 0.75:
        numbers[i]['prediction_reject'] = 1
    elif posterior_twos[i] > 0.75:
```

```
        numbers[i]['prediction_reject'] = 2
    elif posterior_threes[i] > 0.75:
        numbers[i]['prediction_reject'] = 3
    else:
        numbers[i]['prediction_reject'] = 4


confusion_matrix = [[0,0,0],[0,0,0],[0,0,0]]
confusion_matrix_r = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]
for item in numbers:
    confusion_matrix[item['prediction_zero_one']-1][item['class']-1] += 1
    confusion_matrix_r[item['prediction_reject']-1][item['class']-1] += 1

print("Confusion Matrix Training 0-1",*confusion_matrix, sep='\n', end='\n\n')
print("Confusion Matrix Training w/ Rejection",*confusion_matrix_r, sep='\n', end='\
n\n')



test_list = read_csv_file("testing.csv")
test_list_one   = [x['value'] for x in test_list if x['class'] == 1]
test_list_two   = [x['value'] for x in test_list if x['class'] == 2]
test_list_three = [x['value'] for x in test_list if x['class'] == 3]

likelihoods_class_1_test = [calculate_likelihood(test_list[i]['value'], mean_class_1
, std_class_1) for i in range(len(test_list))]
likelihoods_class_2_test = [calculate_likelihood(test_list[i]['value'], mean_class_2
, std_class_2) for i in range(len(test_list))]
likelihoods_class_3_test = [calculate_likelihood(test_list[i]['value'], mean_class_3
, std_class_3) for i in range(len(test_list))]

posterior_ones_test      = [prior_class_1 * likelihoods_class_1_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]
posterior_twos_test      = [prior_class_2 * likelihoods_class_2_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]
posterior_threes_test    = [prior_class_3 * likelihoods_class_3_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]

for i in range(len(test_list)):
    if posterior_ones_test[i] > posterior_twos_test[i] and posterior_ones_test[i] >
posterior_threes_test[i]:
        test_list[i]['prediction_zero_one'] = 1
    elif posterior_twos_test[i] > posterior_ones_test[i] and posterior_twos_test[i]
> posterior_threes_test[i]:
        test_list[i]['prediction_zero_one'] = 2
    elif posterior_threes_test[i] > posterior_ones_test[i] and posterior_threes_test
[i] > posterior_twos_test[i]:
        test_list[i]['prediction_zero_one'] = 3

    if posterior_ones_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 1
    elif posterior_twos_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 2
    elif posterior_threes_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 3
    else :
        test_list[i]['prediction_reject'] = 4

confusion_matrix_test = [[0,0,0],[0,0,0],[0,0,0]]
confusion_matrix_test_r = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]

for item in test_list:
    confusion_matrix_test[item['prediction_zero_one']-1][item['class']-1] += 1
    confusion_matrix_test_r[item['prediction_reject']-1][item['class']-1] += 1

print("Confusion Matrix Test 0-1",*confusion_matrix_test, sep='\n', end='\n\n')
print("Confusion Matrix Test w/ Rejection",*confusion_matrix_test_r, sep='\n', end='
\n\n')
```

```
import csv
import math
import matplotlib.pyplot as plt
import numpy as np

def read_csv_file(file_name):
    """
    Reads a csv file after second line and returns list of dictionaries with the dat
a. Initial values of predictions are -1
    """
    numbers = []
    with open(file_name, 'r') as csv_file:
        csv_reader = csv.reader(csv_file)
        next(csv_reader)
        for item in list(csv_reader):
            my_dict = { 'value': int(item[0]), 'class': int(item[1]) , 'prediction_z
ero_one' : -1, 'prediction_reject' : -1}
            numbers.append(my_dict)
    return numbers


def calculate_likelihood(number, mean, std_dev):
    """
    Calculates the likelihood
    """
    likelihood = 1.0 / (std_dev * (2.0 * math.pi) ** 0.5) * (math.exp(-((number - me
an) ** 2) / (2.0 * std_dev ** 2)))
    return likelihood

'''
Below values are gathered from the training data.
'''
prior_class_1 = 1/3
prior_class_2 = 1/3
prior_class_3 = 1/3

mean_class_1 = 24.48
mean_class_2 = 34.12
mean_class_3 = 49.44

std_class_1 = 1.992385504865963
std_class_2 = 4.1358916813669095
std_class_3 = 5.091797325110258

numbers = read_csv_file("training.csv")
numbers_one   = [x['value'] for x in numbers if x['class'] == 1]
numbers_two   = [x['value'] for x in numbers if x['class'] == 2]
numbers_three = [x['value'] for x in numbers if x['class'] == 3]
minimum = min([x['value'] for x in numbers])
maximum = max([x['value'] for x in numbers])

plot_nums =  np.linspace(minimum, maximum, (maximum-minimum)*3+1)

likelihoods_class_1 = [calculate_likelihood(plot_nums[i], mean_class_1, std_class_1)
 for i in range(len(plot_nums))]
likelihoods_class_2 = [calculate_likelihood(plot_nums[i], mean_class_2, std_class_2)
 for i in range(len(plot_nums))]
likelihoods_class_3 = [calculate_likelihood(plot_nums[i], mean_class_3, std_class_3)
 for i in range(len(plot_nums))]

posterior_ones      = [prior_class_1 * likelihoods_class_1[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums))]
posterior_twos      = [prior_class_2 * likelihoods_class_2[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums))]
posterior_threes    = [prior_class_3 * likelihoods_class_3[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums))]


plt.title("Likelihoods and Posteriors for Training Dataset")
plt.plot(numbers_one  , [-0.05 for i in numbers_one], 'rx')
```

```
plt.plot(numbers_two  , [-0.1 for i in numbers_two] , 'go')
plt.plot(numbers_three, [-0.15 for i in numbers_three], 'b+')

plt.plot(plot_nums, posterior_ones, 'r--', label='P(C=1|X)')
plt.plot(plot_nums, posterior_twos, 'g--', label='P(C=2|X)')
plt.plot(plot_nums, posterior_threes, 'b--', label='P(C=3|X)')

plt.plot(plot_nums, likelihoods_class_1, 'r-', label='P(X|C=1)')
plt.plot(plot_nums, likelihoods_class_2, 'g-', label='P(X|C=2)')
plt.plot(plot_nums, likelihoods_class_3, 'b-', label='P(X|C=3)')

plt.legend(loc='center right')

plt.axis([minimum-3, maximum+3, -0.3, 1.1])
plt.xlabel('Age')
plt.show()



test_list = read_csv_file("testing.csv")
test_list_one   = [x['value'] for x in test_list if x['class'] == 1]
test_list_two   = [x['value'] for x in test_list if x['class'] == 2]
test_list_three = [x['value'] for x in test_list if x['class'] == 3]

minimum = min([x['value'] for x in test_list])
maximum = max([x['value'] for x in test_list])

plot_nums_test =  np.linspace(minimum, maximum, (maximum-minimum)*3+1)

likelihoods_class_1 = [calculate_likelihood(plot_nums_test[i], mean_class_1, std_cla
ss_1) for i in range(len(plot_nums_test))]
likelihoods_class_2 = [calculate_likelihood(plot_nums_test[i], mean_class_2, std_cla
ss_2) for i in range(len(plot_nums_test))]
likelihoods_class_3 = [calculate_likelihood(plot_nums_test[i], mean_class_3, std_cla
ss_3) for i in range(len(plot_nums_test))]

posterior_ones      = [prior_class_1 * likelihoods_class_1[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums_test))]
posterior_twos      = [prior_class_2 * likelihoods_class_2[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums_test))]
posterior_threes    = [prior_class_3 * likelihoods_class_3[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(plot_nums_test))]


plt.title("Likelihoods and Posteriors for Test Dataset")
plt.plot(test_list_one  , [-0.05 for i in test_list_one], 'rx')
plt.plot(test_list_two  , [-0.1 for i in test_list_two] , 'go')
plt.plot(test_list_three, [-0.15 for i in test_list_three], 'b+')

plt.plot(plot_nums_test,posterior_ones, 'r--', label='P(C=1|X)')
plt.plot(plot_nums_test,posterior_twos, 'g--', label='P(C=2|X)')
plt.plot(plot_nums_test,posterior_threes, 'b--', label='P(C=3|X)')

plt.plot(plot_nums_test,likelihoods_class_1, 'r-', label='P(X|C=1)')
plt.plot(plot_nums_test,likelihoods_class_2, 'g-', label='P(X|C=2)')
plt.plot(plot_nums_test,likelihoods_class_3, 'b-', label='P(X|C=3)')

plt.legend(loc='center right')

plt.axis([minimum-3, maximum+3, -0.3, 1.1])
plt.xlabel('Age')
plt.show()
```