ECE4950 HW2

Yi Chen yc2329 https://github.com/agraynel

March 7, 2017

1 Problem1

1.1 class probabilities

$P_r(Gender = M) = \frac{4}{9}$

$P_r(Gender = F) = \frac{5}{9}$

1.2 Estimate means and variances

Means:

$\mu_{\mathcal{H}|\mathcal{M}}$ = (72 + 68 + 75 + 64) / 4 = 69.75

$\mu_{\mathcal{S}|\mathcal{M}}$ = (12 + 9 + 11 + 10.5) / 4 = 10.625

$\mu_{\mathcal{H}|\mathcal{F}}$ = (65 + 67 + 62 + 70 + 64) / 5 = 65.6

$\mu_{\mathcal{S}|\mathcal{F}}$ = (8 + 7.5 + 6 + 8.5 + 8) / 5 = 7.6

Variances:

$\sigma^2_{\mathcal{H}|\mathcal{M}} = \frac{1}{N}\sum_{i=1}^{N}(x(i) - \mu)^2$

$= ((72 - 69.75)^2 + (68 - 69.75)^2 + (75 - 69.75)^2 + (64 - 69.75)^2)/4$

$= 17.1875$

$\sigma^2_{\mathcal{S}|\mathcal{M}} = ((12-10.625)^2 + (9-10.625)^2 + (11-10.625)^2 + (10.5-10.625)^2)/4$
$= 1.172$
$\sigma^2_{\mathcal{H}|\mathcal{F}} = ((65-65.6)^2 + (67-65.6)^2 + (62-65.6)^2 + (70-65.6)^2 + (64-65.6)^2)/5$
$= 7.44$
$\sigma^2_{\mathcal{S}|\mathcal{F}} = ((8-7.6)^2 + (7.5-7.6)^2 + (6-7.6)^2 + (8.5-7.6)^2 + (8-7.6)^2)/5 = 0.74$

1.3 Gausian Naive Bayes

Suppose height is 68 inches, and foot size is 9.5:
$p(H=68|M) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} = \frac{1}{10.39} e^{-\frac{3.0625}{34.375}} = 0.088$
$p(S=9.5|M) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} = \frac{1}{2.714} e^{-\frac{1.266}{2.344}} = 0.215$
$p(H=68|F) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} = \frac{1}{6.837} e^{-\frac{5.76}{14.88}} = 0.099$
$p(S=9.5|F) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} = \frac{1}{2.156} e^{-\frac{3.61}{1.48}} = 0.040$

1.4 Classification

$P_r(Gender = M) = \frac{4}{9}$
$P_r(Gender = F) = \frac{5}{9}$
$p(H,S|M) = p(H|M) * p(S|M) = 0.01892$
$p(M)p(H,S|M) = \frac{4}{9} * 0.01892 = 0.00841$
$p(H,S|F) = p(H|F) * p(S|F) = 0.00396$
$p(F)p(H,S|F) = \frac{5}{9} * 0.00396 = 0.0022$
Because $p(M)p(H,S|class = M)$ is larger, so we would classify the above sample as male.

2 Problem2

2.1 Gaussian Naive Bayes

See appendix and attached files for codes.

2.2 Accuracy

-50.csv: 0.890173410405

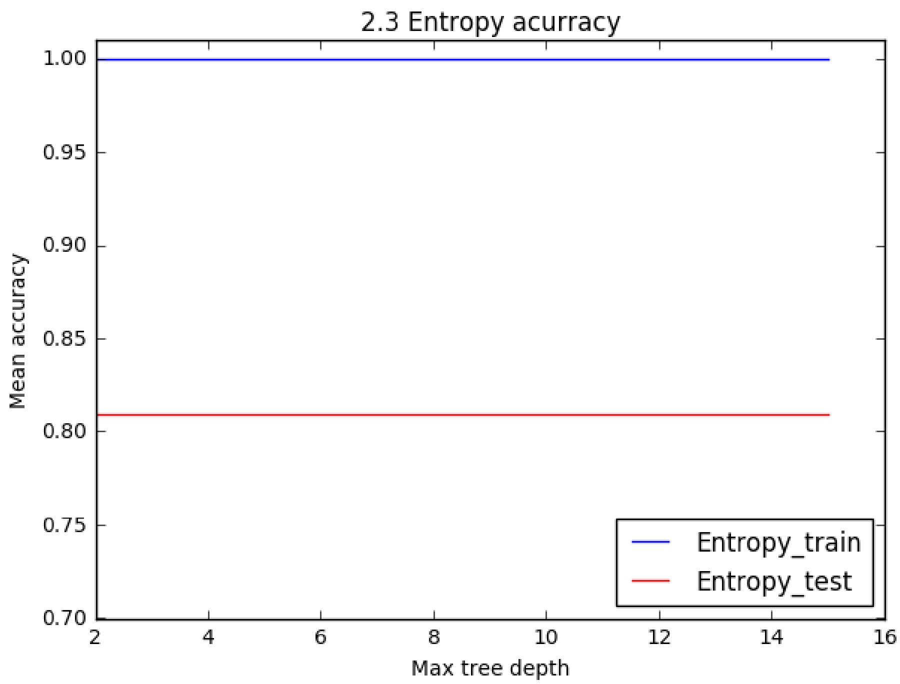
-200.csv: 0.964769647696

-400.csv: 0.96449704142

2.3 Decision tree learning

See appendix and attached files for codes.

The figure of accuracy of -50.csv using decision tree learning methods:



The test accuracy of decision tree with best depth is around 0.81 to 0.82, while the accuracy of Naive Bayes is 0.89. Obviously, Naive Bayes has a better test error.

2.4 Decision tree using minimum leaf size criterion

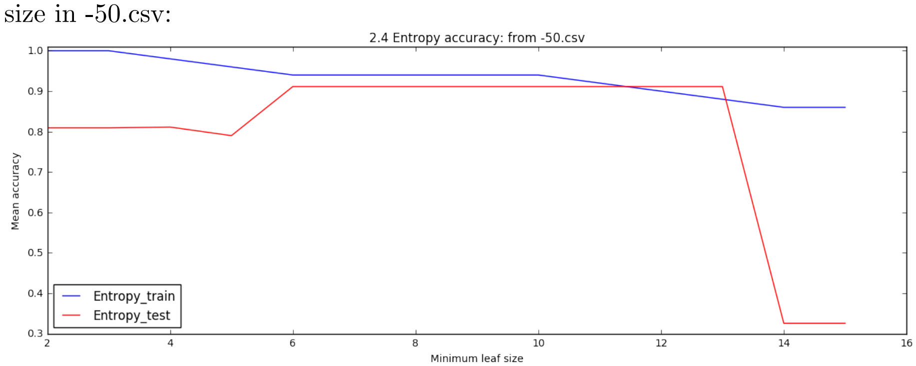
See appendix and attached files for codes, and figures are at the end of this section.

In -50.csv, compared to maximum depth criterion, when minimum leaf size is between 6 and 13(both sides included), the accuracy is larger, reaching 0.9. While the accuracy of decision tree classifier using maximum depth criterion is always above 0.8. When minimum leaf size is between 6 and 13, the accuracy is also a little larger than the accuracy of Naive Bayes.

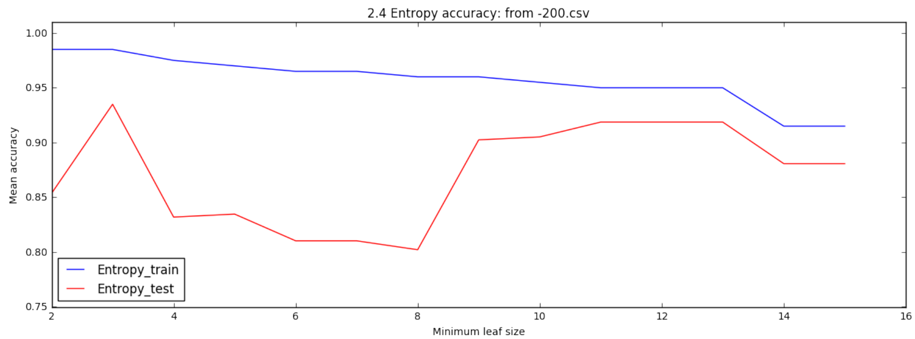
In -200.csv, compared to maximum depth criterion, when minimum leaf size is 2, 3 or between 9 and 13, the accuracy is larger. While the accuracy of decision tree classifier using maximum depth criterion is always around 0.8 when the depth is larger than 3. The accuracy is less than the accuracy of Naive Bayes.

In -400.csv, compared to maximum depth criterion, the maximum and average accuracy are smaller in using minimum leaf size. The accuracy is also less than the accuracy of Naive Bayes.

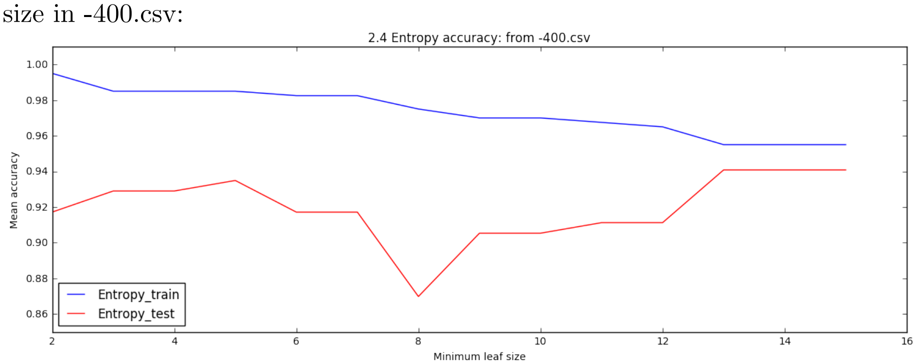
The figure of test and training accuracy with respect to different minimum leaf size in -50.csv:



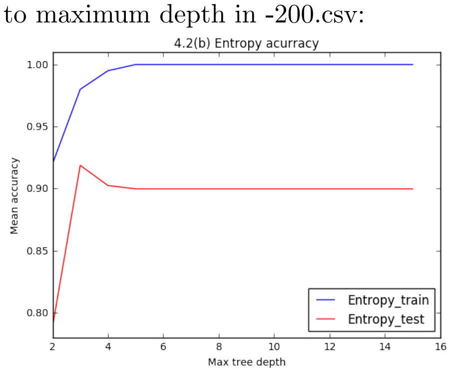
The figure of test and training accuracy with respect to different minimum leaf size in -200.csv:

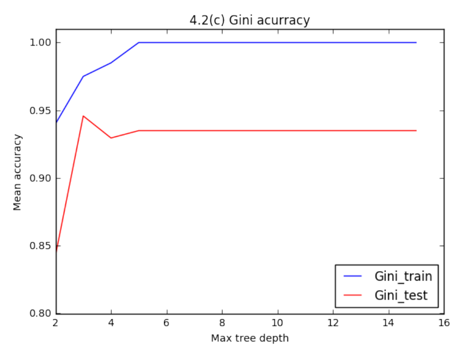


The figure of test and training accuracy with respect to different minimum leaf size in -400.csv:

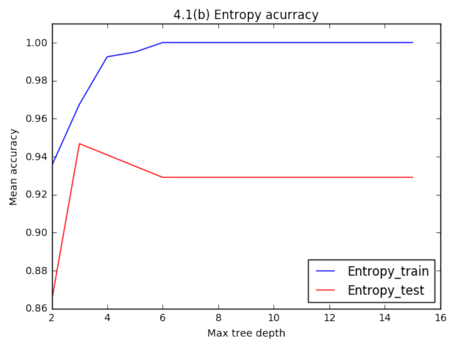


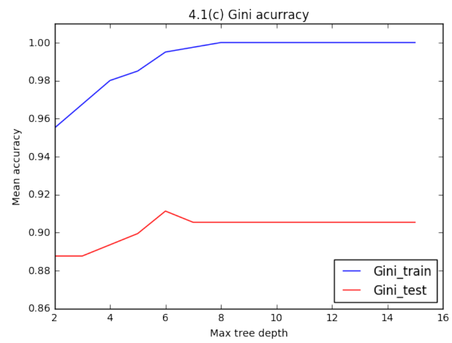
From homework 1, we have the figure of test and training accuracy with respect to maximum depth in -200.csv:





From homework 1, we have the figure of test and training accuracy with respect to maximum depth in -400.csv:





3 Problem3

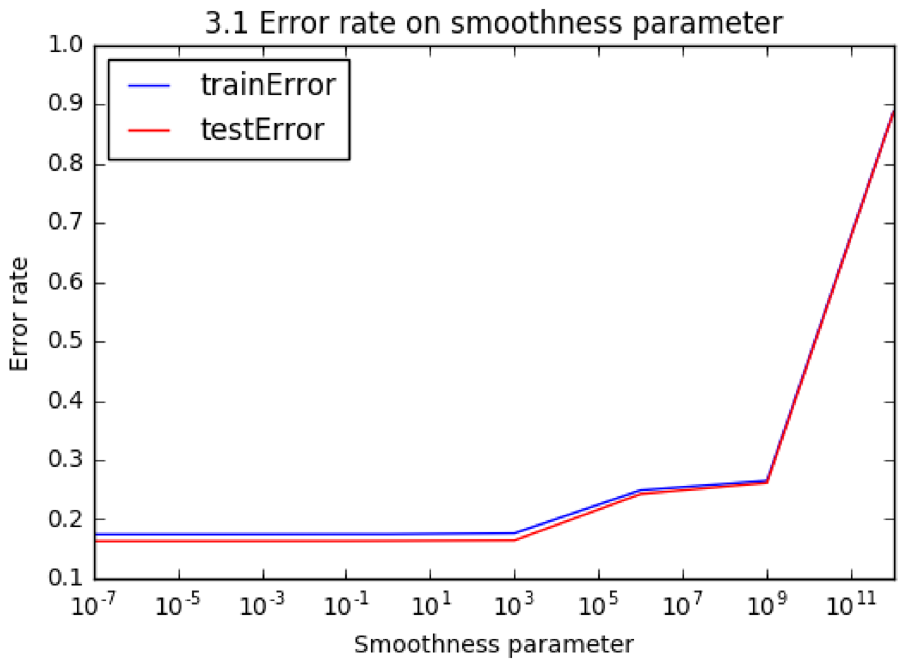
3.1 Multinomial Naive Bayes

See appendix and attached files for codes.

Error rate and smoothness parameter:

β	Train error rate	Test error rate
1e-07	0.174333333333	0.163
0.0001	0.17445	0.1631
0.1	0.174683333333	0.1633
1	0.174716666667	0.1635
1000	0.176466666667	0.1644
1000000	0.249066666667	0.2426
1000000000	0.26525	0.2616
1000000000000	0.887633333333	0.8865

Figure of error rate as a function of smoothness parameter β:



3.2 $\beta \rightarrow \infty$

See appendix and attached files for codes.

$Add - \beta Prob(x) = \frac{n_x+\beta}{N+D*\beta}$

As $\beta \rightarrow \infty$, error rate also $\rightarrow 0.9. Whensmoothing, someofthefeatureswillnotpresentinthethelearningsample$

β	Train error rate	Test error rate
inf	0.901283333333	0.902

4 Appendix

4.1 Code for 2.1

```
#2.1 Gaussian Naive Bayes
#This is the code for all the 3 sets of data. Only need
    to change the file names in the code! On piazza, TA
```

```
    said only using entropy criterion is enough.
#CREDITS: http://scikit-learn.org/stable/modules/
    generated/sklearn.naive_bayes.GaussianNB.html#sklearn.
    naive_bayes.GaussianNB.partial_fit

import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

#this is the path of dataset
Xtrn = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/X-trn-50.csv', header = None)
Xtst = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/X-tst-50.csv', header = None)
Ytrn = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/Y-trn-50.csv', header = None)
Ytst = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/Y-tst-50.csv', header = None)
#NOTICE: change the file name to -50.csv, -200.csv, -400.
    csv respectively.

clf = GaussianNB()
clf.fit(Xtrn, Ytrn.values.ravel())
#CREDITS: http://stackoverflow.com/questions/34165731/a-
    column-vector-y-was-passed-when-a-1d-array-was-
    expected
#GaussianNB(priors=None)
# train with Gaussian
```

```
test = clf.predict(Xtst)
train_accuracy = clf.score(Xtrn, Ytrn)
test_accuracy = accuracy_score(Ytst, test)

print(test_accuracy)
```

4.2 Code for 2.3

```
#2.3 read 50 files decision tree classifier with maximum
    depth

import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

#this is the path of dataset
Xtrn50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/X-trn-50.csv', header = None)
Xtst50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/X-tst-50.csv', header = None)
Ytrn50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/Y-trn-50.csv', header = None)
Ytst50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/
    homework/hw2/dataset/Y-tst-50.csv', header = None)

entropy_50_train = []
entropy_50_test = []

```

```
for i in range(2, 16):
    clf_entropy = DecisionTreeClassifier(criterion = 'entropy', max_depth= i, random_state = 0)
    clf_entropy = clf_entropy.fit(Xtrn50, Ytrn50)
    entropy_test = clf_entropy.predict(Xtst50)
    entropy_50_train.append(clf_entropy.score(Xtrn50, Ytrn50))
    entropy_50_test.append(accuracy_score(Ytst50, entropy_test))

# Plotting decision regions

plt.figure(figsize=(15, 5))
plt.plot(range(2, 16), entropy_50_train, c='blue', label='Entropy train')
plt.plot(range(2, 16), entropy_50_test, c='red', label='Entropy test')
plt.legend(loc=4)
plt.ylim(0.7, 1.01)
plt.ylabel('Mean accuracy')
plt.xlabel('Max tree depth')
plt.title('2.3 Entropy accuracy')

plt.show()
```

4.3 Code for 2.4

```
#2.4 decision tree classifier with minimum leaf size i criterion.
```

```python
#this is one typical code for one situation. jusr change the path and get other datasets.

import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

#this is the path of dataset
Xtrn50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/homework/hw2/dataset/X-trn-50.csv', header = None)
Xtst50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/homework/hw2/dataset/X-tst-50.csv', header = None)
Ytrn50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/homework/hw2/dataset/Y-trn-50.csv', header = None)
Ytst50 = pd.read_csv('/Users/Agraynel/Desktop/ECE4950/homework/hw2/dataset/Y-tst-50.csv', header = None)

entropy_50_train = []
entropy_50_test = []
#change 50 to 200, 400, then we can acquire all the figures of 3 conditions.

for i in range(2, 16):
    clf_entropy = DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf = i, random_state = 0)
    clf_entropy = clf_entropy.fit(Xtrn50, Ytrn50)
    entropy_test = clf_entropy.predict(Xtst50)
```

```
    entropy_50_train.append(clf_entropy.score(Xtrn50,
        Ytrn50))
    entropy_50_test.append(accuracy_score(Ytst50,
        entropy_test))

# Plotting decision regions

plt.figure(figsize=(15, 5))
plt.plot(range(2, 16), entropy_50_train, c='blue', label=
    'Entropy_train')
plt.plot(range(2, 16), entropy_50_test, c='red', label='
    Entropy_test')
plt.legend(loc = 3)
plt.ylim(0.3, 1.01)
plt.ylabel('Mean accuracy')
plt.xlabel('Minimum leaf size')
plt.title('2.4 Entropy accuracy')

plt.show()
```

4.4 Code for 3.1

```
#3.1 Error rate on smoothness parameter
import csv
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn import datasets
```

```python
from sklearn.datasets import fetch_mldata

beta = [0.0000001, 0.0001, 0.1, 1, 1000, 1000000, 1000000000, 1000000000000]
## Loading data
mnist = fetch_mldata('MNIST-original')
train, test = mnist.data[0:60000, :], mnist.data[60000:, :]
X_trn, y_trn = mnist.data[0:60000, :], mnist.target[0:60000]
X_tst, y_tst = mnist.data[60000:, :], mnist.target[60000:]

trainError = [];
testError = [];
for i in range(0, 8):
    clf = MultinomialNB(alpha = beta[i], class_prior = None, fit_prior = True)
    clf = clf.fit(X_trn,y_trn)
    tst = clf.predict(X_tst)
    accuracy = clf.score(X_trn, y_trn)
    testAccuracy = accuracy_score(y_tst, tst)
    trainError.append(1 - accuracy)
    testError.append(1 - testAccuracy)
    print(beta[i], 1 - accuracy, 1 - testAccuracy)

# Plotting error rate as a function of smoothness parameter

plt.figure()
```

```
plt.plot(beta, trainError, c='blue', label='trainError')
plt.plot(beta, testError, c='red', label='testError')
plt.xscale('log')
plt.legend(loc = 0)
plt.ylim(0.1, 1)
plt.ylabel('Error rate')
plt.xlabel('Smoothness parameter')
plt.title('3.1 Error rate on smoothness parameter')

plt.show()
```

4.5 Code for 3.2

```
#3.2 smoothness parameter is infinity
import csv
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn import datasets
from sklearn.datasets import fetch_mldata

## Loading data
mnist = fetch_mldata('MNIST-original')
train, test = mnist.data[0:60000, :], mnist.data[60000:,
    :]
X_trn, y_trn = mnist.data[0:60000, :], mnist.target
    [0:60000]
```

```
X_tst, y_tst = mnist.data[60000:, :], mnist.target
    [60000:]

trainError = [];
testError = [];
clf = MultinomialNB(alpha = float("inf"), class_prior =
    None, fit_prior = True)
clf = clf.fit(X_trn,y_trn)
tst = clf.predict(X_tst)
accuracy = clf.score(X_trn, y_trn)
testAccuracy = accuracy_score(y_tst, tst)
trainError.append(1 - accuracy)
testError.append(1 - testAccuracy)
print(float("inf"), 1 - accuracy, 1 - testAccuracy)
```