In [1]:

```
from hmmlearn import hmm
import os
import collections
import pandas as pd
from sklearn.model_selection import StratifiedShuffleSplit
import glob
import numpy as np
from sklearn import preprocessing
```

In [2]:

```
import collections
```

In [189]:

```
directory = '/Users/alexong/Downloads/Malicia'
os.chdir("/Users/alexong/Downloads/CS131_Midterm")
def length(file):
    with open(file, "r") as file_read:
        for i, l in enumerate(file_read):
            pass
        if i >= 500:
            return True
        else:
            return False

observation = set() # Opcodes
counter = 0 # Amount of lines
test_opcode = []
all_zbot = []
training_opcode = []
all_zAcc = []
test_counter = 0

    for file in zBot:
        if length(file):
            temp_list = []
            with open(file, "r") as f:
                for line in f:
                    line_stripped = line.strip()
                    temp_list.append(line_stripped)
                    observation.add(line_stripped)
                    if counter < 30000:
                        training_opcode.append(line_stripped)
                    counter += 1

            all_zbot.append(temp_list)

        else:
            continue

test_counter = 0
    for file in zAcc:
        if length(file):
            temp_list = []
            with open(file, "r") as f:
                for line in f:
                    line_stripped = line.strip()
                    temp_list.append(line_stripped)
                    observation.add(line_stripped)
            all_zAcc.append(temp_list)

        else:
            continue
```

In [256]:

```
len(observation)
```

Out[256]:

```
420
```

In [247]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(list(observation))
```

Out[247]:

```
▾ LabelEncoder

LabelEncoder()
```

In [264]:

```
encoded_training = le.transform(training_opcode)

encoded_zbot = []
for i in all_zbot:
    encoded_zbot.append(le.transform(i))

encoded_zacc = []
for j in all_zAcc:
    encoded_zacc.append(le.transform(j))
```

In [275]:

```
#len(encoded_zbot)
len(encoded_zacc)
```

Out[275]:

```
1304
```

In [255]:

```
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as plt
```

In [267]:

In [195]:

```
observation_matrix = []
```

In [124]:

In [196]:

```
for line in training_opcode:
    if line in opcode_vocab:
        observation_matrix.append(opcode_vocab[line])
    else:
        observation_matrix.append(M)
```

In [197]:

```
zbotsss = []
for file in all_zbot:
    temp_matrix = []
    for i in file:
        if i in opcode_vocab:
            temp_matrix.append(opcode_vocab[i])
        else:
            temp_matrix.append(M)
    zbotsss.append(temp_matrix)
```

In [198]:

```
zAccss = []
for file in all_zAcc:
    temp_matrix = []
    for i in file:
        if i in opcode_vocab:
            temp_matrix.append(opcode_vocab[i])
        else:
            temp_matrix.append(M)
    zAccss.append(temp_matrix)
```

In [234]:

```
len(zAccss)
```

Out[234]:

```
1304
```

In [151]:

```
len(zbotsss)
```

Out[151]:

```
1929
```

In [257]:

```
len(observation_matrix)
```

Out[257]:

```
30000
```

In [269]:

```
encoded_training
```

Out[269]:

```
array([359, 206,   6, ..., 332, 235, 235])
```

In [270]:

```
reshape = np.array(encoded_training)
array = reshape.reshape(-1,1)
print(array)
```

```
[[359]
 [206]
 [  6]
 ...
 [332]
 [235]
 [235]]
```

In [271]:

```

# Trained without reducing noise
remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(array, len(array))

```

Out[271]:

```
▾                              CategoricalHMM

CategoricalHMM(n_components=2, n_iter=100,
                   random_state=RandomState(MT19937) at 0x7F88F0536240)
```

In [ ]:

```

```

In [226]:

```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.DataFrame()
```

In [279]:
```
x_zbot = []
y_zbot = []
fileCounter = 0

for sequence in encoded_zbot[1000:]:
        if(fileCounter<500):
            array = np.array(sequence[0:500]).reshape(-1,1)
            score = remodel.score(array)
            x_zbot.append(fileCounter)
            y_zbot.append(score)
            fileCounter+=1
        print(f"Score for sequence: {score}")
```

```
Score for sequence: -2030.768604092508
Score for sequence: -1195.420883186615
Score for sequence: -1130.395340003188
Score for sequence: -1087.4833853349687
Score for sequence: -2178.717267614969
Score for sequence: -1163.0470442977105
Score for sequence: -2046.2433006363437
Score for sequence: -1148.411152590163
Score for sequence: -1087.4833853349687
Score for sequence: -inf
Score for sequence: -inf
Score for sequence: -1940.9124571960253
Score for sequence: -1282.087466273797
Score for sequence: -1287.6089193172156
Score for sequence: -1284.9657491436826
Score for sequence: -2217.9015127844496
Score for sequence: -inf
Score for sequence: -1999.4308533850265
Score for sequence: -2173.139911636459
Score for sequence: -1130.3293353972895
```

In [ ]:

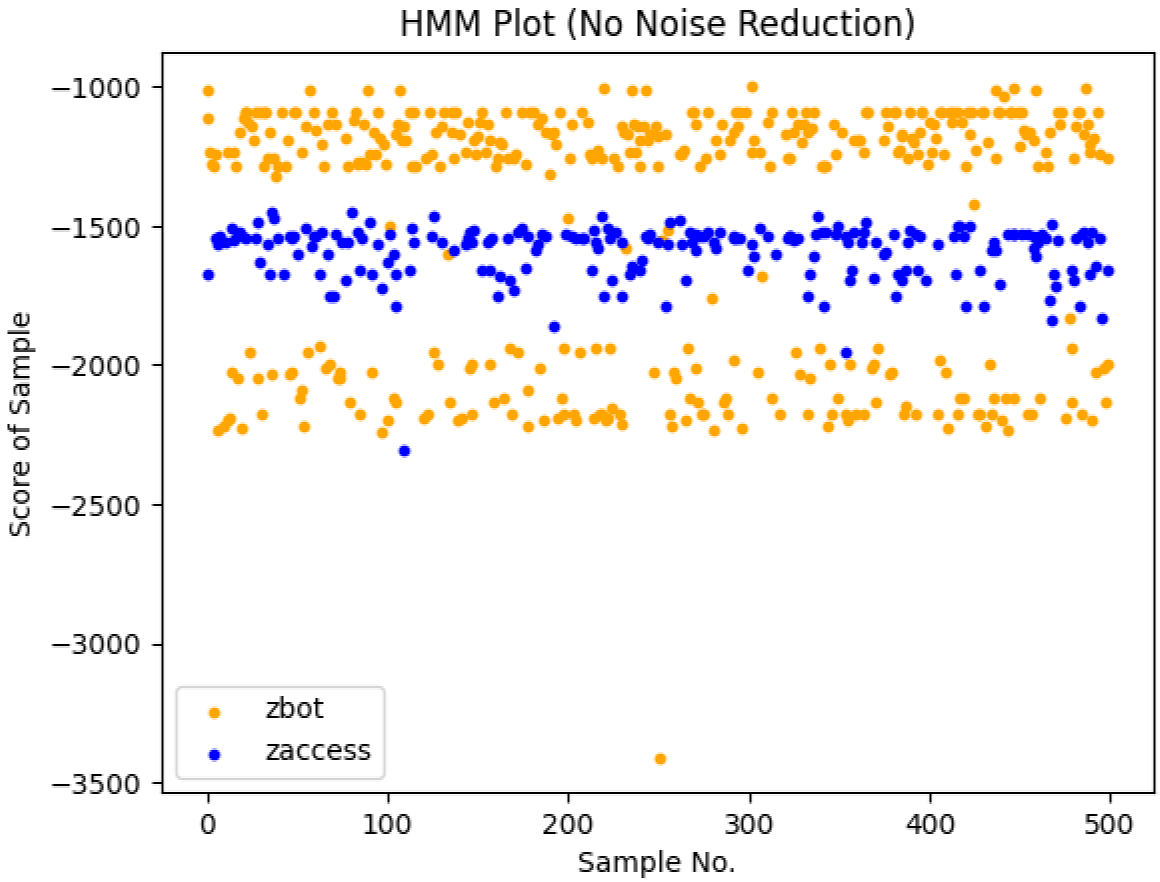
In [ ]:

In [281]:

```
x_zaccess = []
y_zaccess = []
fileCounter = 0
for sequence in encoded_zacc[704:]:
        #temp_matrix = []
        #counter = 0
        #with open(sequence, "r") as f:
            #for line in f:
                #if counter < 500:
                    #stripped_line = line.strip()
                    #counter += 1
                    #if stripped_line in opcode_vocab:
                        #temp_matrix.append(opcode_vocab[stripped_line])
                    #else:
                        #temp_matrix.append(M-1)
                #else:
                    #break
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess.append(fileCounter)
            y_zaccess.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1695.4877908740784
Score for sequence: -1544.2119606787844
Score for sequence: -inf
Score for sequence: -1790.621082758481
Score for sequence: -1528.435348755905
Score for sequence: -1519.4171687809767
Score for sequence: -inf
Score for sequence: -1559.5937758219177
Score for sequence: -inf
Score for sequence: -1673.99451559454
Score for sequence: -1525.2024097242413
Score for sequence: -inf
Score for sequence: -1640.8943489179214
Score for sequence: -inf
Score for sequence: -1544.0752880354216
Score for sequence: -1828.4093534837373
Score for sequence: -inf
Score for sequence: -inf
Score for sequence: -inf
Score for sequence: -1656.1965290039734
```

In [324]:
```
plt.xlabel("Sample No.")
plt.ylabel("Score of Sample")
plt.title("HMM Plot (No Noise Reduction)")
plt.scatter(x_zbot, y_zbot, label = "zbot", color = "orange", s =10)
plt.scatter(x_zaccess, y_zaccess, label = "zaccess", color = "blue", s =10)
plt.legend()
plt.show()
```



In [307]:
```
def opcodeEncoder(M, encoded):
    global opcode_vocab
    opcodeFrequency = {}
    for element in encoded:
        if element not in opcodeFrequency:
            opcodeFrequency[element] = 1
        else:
            opcodeFrequency[element] += 1
    tup = sorted(opcodeFrequency.items(), key=lambda x:x[1], reverse=True)

    for i in range(0, M-1):
        opcode_vocab[tup[i][0]] = i
    #print(opcodeFrequency)
    #print(tup)
    print(opcode_vocab)
```

In [308]:
```
opcodeEncoder(30, encoded_training) # Reducing noise by changing amount of
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28}
```

In [301]:

```
M = 30
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Amount of observables - 1 to combine opcoc
```

In [302]:

```
encoded_noise_reduced
```

```
 30,
 30,
 359,
 332,
 178,
 30,
 7,
 235,
 197,
 417,
 348,
 347,
 348,
 348,
 6,
 30,
 30,
 30,
 348,
 7,
```

In [303]:

```
reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)
print(array)
```

```
[332 235 399 418 332 399 332 418   9 178 332 359 332   6  21 408 235 206
 418 235 197 332 359 235 332   9 332 359  21 235 408 235 202 332   6 332
  21 359 408 202 418 332   6 332 332  21   7 408 235 206 418 235 197 359
 235   9 235 235 399 235 235 399 235 359 211 235   6 211 418 258   9 235
 359 235 235 235 399 399 235 235 235 252   7 235 418   9 235   7 235 235
 235  46 235 202 235 235 235 418   7   7 235   9 197 418   7 178 235  46
 198  46 202 332 332  21  46 202 418   7 178 235 197 235   9 235 178 235
  46 202 332 332 332  21  46 202 235 178 235  46 189   9 418 252 399 235
   7 235  46 206  46 202 332 332  21 235  66 235 197 235 235   9 235 178
 235  46 202 332 332 332  21  46 189 235 235   6 235   9 418 258 178 235
  46 202 332 332 332  21  46 202 418 418 178 235  46 189 235  46 206  46
 202  21 418 258  66 235 197 399 332 258 235   9 235 178 235  46 202  21
  46 202 418 418 178 235  46 189   9 197 235 178 235  46 198  46 202 332
 332 332 332 332 332 332  21 197   6 235 399 235 235  46 206  46 202 332
 332  21  46 202 235  66 235 418 258  66 235 197 235   9 197 418 418 178
 235  46 198  46 202 332 332 332 332  21 197 235 418 235 235 235   9 235
 235 418 258   9 235 359 418   9 418 399   7 359 235 418 418 418 258   7
 211 257 235 235  46 188 197 235 235 399   9 235 235 399 418 211 258 332
 399 235   9 418 418 211 258 332 258 235   9 332 418  21   7 235 258 235
 255 418 399 211 235 418 235 418   7 418   6   7 235 418 211 235 235 235
  46 235 190   9 235 178 235  46 202 332 332  21  46 189 235 359 235   9
 235 178 235  46 202 332 332 332 332 332 332  21  46 189 235 399 235   9
 235 178 235  46 202  21  46 189   7 235 258 235 399 255 418 235 235 235
 235 235 235  46 202 418 211   6 235 418 211 399 235 235 408 235 206 418
 235 235 235   6 235 257 235 235   9 235 178 235  46 202 332  21  46 189
   9 197 235 178 235  46 198  46 202 332 332  21  46 202 235 178 235 197
 235 255 235 359  23   6 235 235  46 206  46 202 332 332 332  21 418 258
  66 235 197 235  46 206  46 202 332 332  21 418 418  66]
```

In [304]:

```
remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))
```

Out[304]:

```
▾                          CategoricalHMM

CategoricalHMM(n_components=2, n_iter=100,
                    random_state=RandomState(MT19937) at 0x7F88F0536840)
```

In [314]:

```python
encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)
encoded_noise_reduced_zbot[1]
```

```
 7,
 235,
 7,
 7,
 235,
 235,
 7,
 235,
 235,
 399,
 235,
 197,
 235,
 332,
 235,
 332,
 21,
 7,
 235,
 235,
```

In [315]:
```
encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)
encoded_noise_reduced_zaccess[2]
```

```
 30,
 235,
 258,
 418,
 258,
 197,
 30,
 46,
 418,
 30,
 300,
 30,
 9,
 235,
 235,
 348,
 178,
 348,
 235,
 235,
In [316]: x_zbot_reduced = []
          y_zbot_reduced = []
          fileCounter = 0

          for sequence in encoded_noise_reduced_zbot[1000:]:
                  if(fileCounter<500):
                      array = np.array(sequence[0:500]).reshape(-1,1)
                      score = remodel.score(array)
                      x_zbot_reduced.append(fileCounter)
                      y_zbot_reduced.append(score)
                      fileCounter+=1
                  print(f"Score for sequence: {score}")
```

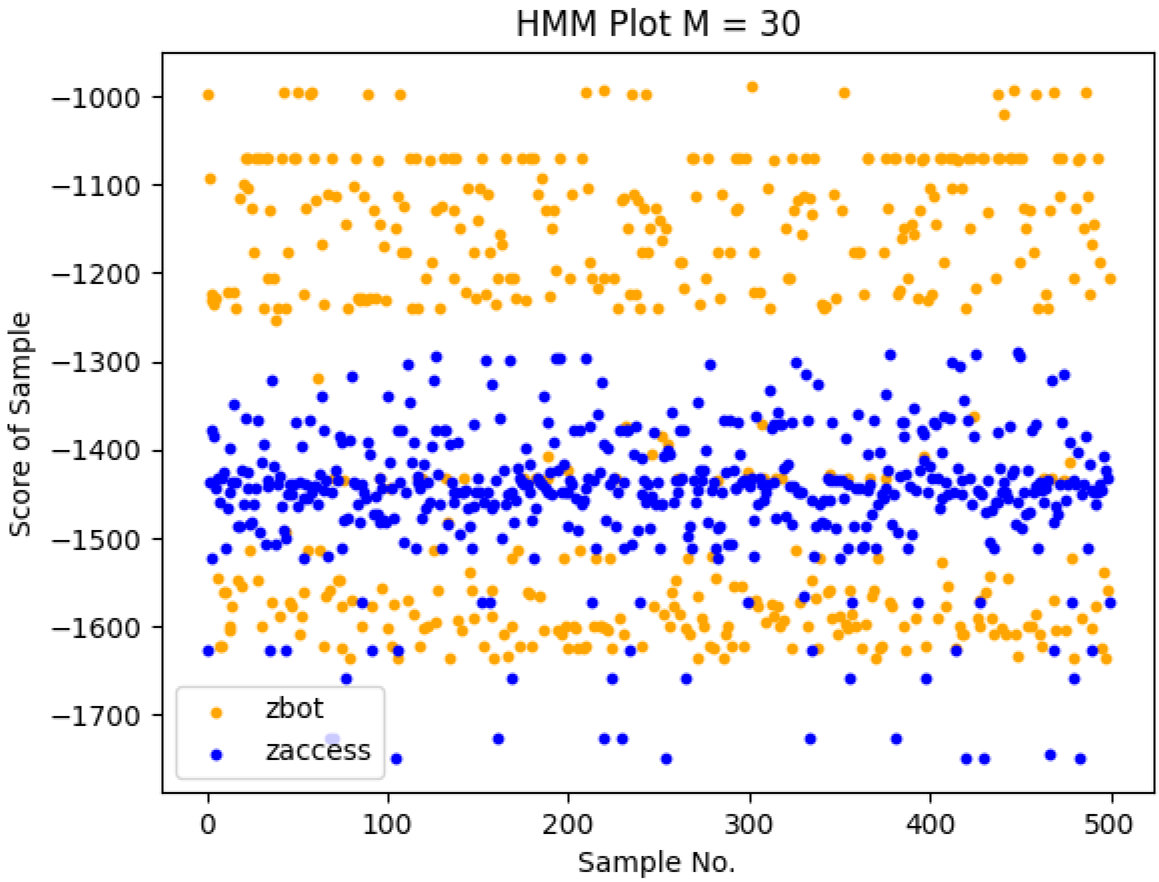
```
Score for sequence: -1600.02420629633
Score for sequence: -1576.3008783240462
Score for sequence: -1222.1928911367268
Score for sequence: -1241.0662839077204
Score for sequence: -1548.0907435631045
Score for sequence: -1116.0196066571418
Score for sequence: -1555.3741359990559
Score for sequence: -1100.2002975804146
Score for sequence: -1069.337601954034
Score for sequence: -1069.337601954034
Score for sequence: -1104.2716494123547
Score for sequence: -1513.657404051661
Score for sequence: -1126.3326316320756
Score for sequence: -1175.5695613832972
Score for sequence: -1069.337601954034
Score for sequence: -1548.0907435631045
Score for sequence: -1069.2934404249709
Score for sequence: -1599.1665588357357
Score for sequence: -1240.1400769210063
Score for sequence: -1069.337601954034
Score for sequence: -1069.2934404249709
```

In [317]:

```python
x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1659.4009590525501
Score for sequence: -1439.0159017034755
Score for sequence: -1404.0223762272553
Score for sequence: -1748.9785086795148
Score for sequence: -1450.2883286765586
Score for sequence: -1438.9469122955788
Score for sequence: -1385.224130577847
Score for sequence: -1511.1887003549673
Score for sequence: -1417.0658213581319
Score for sequence: -1627.2078757126696
Score for sequence: -1446.7332235767662
Score for sequence: -1447.4233881000787
Score for sequence: -1461.8815084492278
Score for sequence: -1448.9570873897312
Score for sequence: -1438.837931767242
Score for sequence: -1445.3038336559632
Score for sequence: -1407.5241019961466
Score for sequence: -1424.3411827081532
Score for sequence: -1434.1535499649865
Score for sequence: -1572.827930437785
```

In [323]:
```
plt.xlabel("Sample No.")
plt.ylabel("Score of Sample")
plt.title("HMM Plot M = 30")
plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
plt.legend()
plt.show()
```



In [345]:

```
M = 31
opcodeEncoder(M, encoded_training) # Reducing noise by changing amount of
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Reduce symbols used

reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)

remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))


encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)


encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)


x_zbot_reduced = []
y_zbot_reduced = []
fileCounter = 0
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28, 25:
29}
```

```
In [346]: for sequence in encoded_noise_reduced_zbot[1000:]:
                  if(fileCounter<500):
                      array = np.array(sequence[0:500]).reshape(-1,1)
                      score = remodel.score(array)
                      x_zbot_reduced.append(fileCounter)
                      y_zbot_reduced.append(score)
                      fileCounter+=1
                  print(f"Score for sequence: {score}")
```

```
Score for sequence: -1561.4627784871209
Score for sequence: -1116.1971670939906
Score for sequence: -1566.8162322594396
Score for sequence: -1100.3078960650905
Score for sequence: -1069.402880549464
Score for sequence: -1069.402880549464
Score for sequence: -1104.2861379994056
Score for sequence: -1519.6395777094458
Score for sequence: -1126.2932548605115
Score for sequence: -1175.5512717571376
Score for sequence: -1069.402880549464
Score for sequence: -1561.4627784871209
Score for sequence: -1069.3645609678356
Score for sequence: -1620.5476615894552
Score for sequence: -1240.5192184043856
Score for sequence: -1069.402880549464
Score for sequence: -1069.3645609678356
Score for sequence: -1205.7452260577688
Score for sequence: -1128.825231487562
Score for sequence: -1579.546438536733
Score for sequence: -1205.7452260577688
```

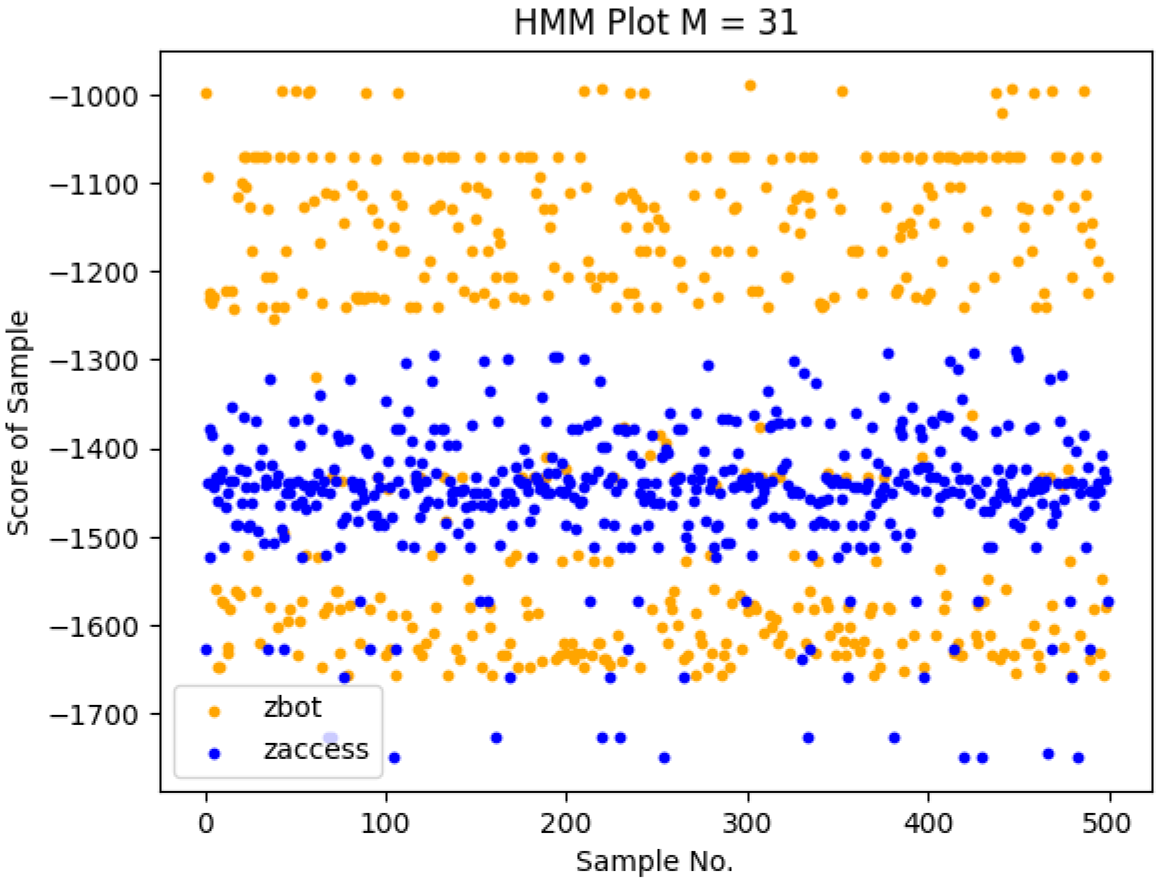
In [347]:

```python

x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1626.3371964813778
Score for sequence: -1438.3450110757558
Score for sequence: -1379.0556521000294
Score for sequence: -1523.0775776863873
Score for sequence: -1385.0780107158337
Score for sequence: -1442.7640538805176
Score for sequence: -1432.5355794935847
Score for sequence: -1458.98725516351
Score for sequence: -1433.8794364700148
Score for sequence: -1425.9106972641412
Score for sequence: -1511.1507934542126
Score for sequence: -1466.520913257413
Score for sequence: -1450.1568281123816
Score for sequence: -1399.845234886161
Score for sequence: -1437.61782475031
Score for sequence: -1353.4963183572734
Score for sequence: -1437.6620101751564
Score for sequence: -1485.676874325994
Score for sequence: -1486.1048835472527
```

In [348]:
```
plt.xlabel("Sample No.")
plt.ylabel("Score of Sample")
plt.title("HMM Plot M = 31")
plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
plt.legend()
plt.show()
```



In [349]:

```python
M = 32
opcodeEncoder(M, encoded_training) # Reducing noise by changing amount of
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Reduce symbols used

reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)

remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))


encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)


encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)


x_zbot_reduced = []
y_zbot_reduced = []
fileCounter = 0
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28, 25:
29, 24: 30}
```

```
In [350]: for sequence in encoded_noise_reduced_zbot[1000:]:
                  if(fileCounter<500):
                      array = np.array(sequence[0:500]).reshape(-1,1)
                      score = remodel.score(array)
                      x_zbot_reduced.append(fileCounter)
                      y_zbot_reduced.append(score)
                      fileCounter+=1
                  print(f"Score for sequence: {score}")
```

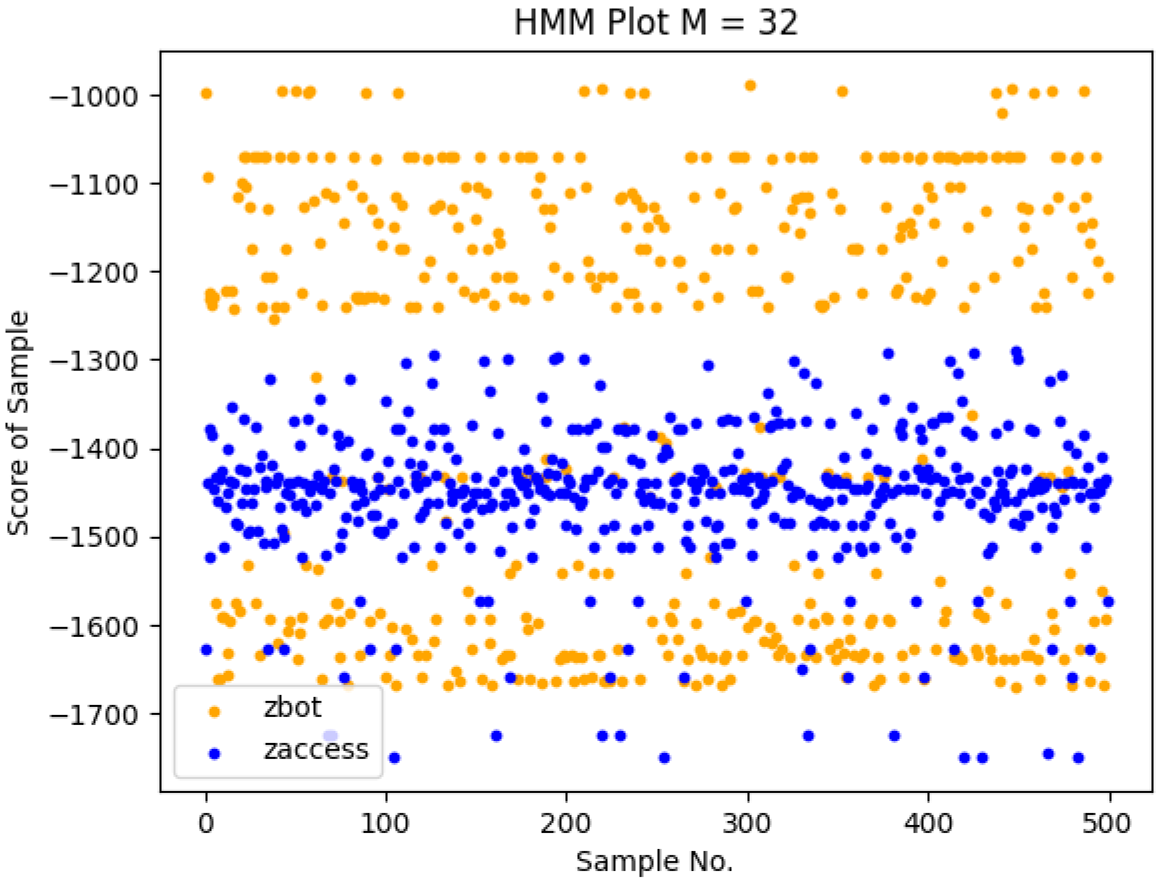
```
Score for sequence: -1241.409347607369
Score for sequence: -1575.1823810334852
Score for sequence: -1116.5153287848877
Score for sequence: -1584.7899556818807
Score for sequence: -1100.4417480329032
Score for sequence: -1069.531013584
Score for sequence: -1069.531013584
Score for sequence: -1104.4343026872075
Score for sequence: -1533.2121466022613
Score for sequence: -1126.4177142822766
Score for sequence: -1175.6004188724073
Score for sequence: -1069.531013584
Score for sequence: -1575.1823810334852
Score for sequence: -1069.5314535341781
Score for sequence: -1634.099240127253
Score for sequence: -1240.8546453607316
Score for sequence: -1069.531013584
Score for sequence: -1069.5314535341781
Score for sequence: -1205.8189634870353
Score for sequence:  1128.9256623741803
```

In [351]:

```
x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1626.6280458572082
Score for sequence: -1439.9737793889572
Score for sequence: -1379.498505924447
Score for sequence: -1523.234936486846
Score for sequence: -1386.148353052543
Score for sequence: -1447.1210926081558
Score for sequence: -1432.6140944885835
Score for sequence: -1459.4297794691074
Score for sequence: -1434.9090621847513
Score for sequence: -1427.0642464801065
Score for sequence: -1511.6032821760548
Score for sequence: -1467.526828251711
Score for sequence: -1451.5131002432963
Score for sequence: -1400.7325199417348
Score for sequence: -1437.847346651355
Score for sequence: -1354.3693404681494
Score for sequence: -1438.5571792617034
Score for sequence: -1485.9135738516109
Score for sequence: -1486.3065129017111
```

In [352]:
```
plt.xlabel("Sample No.")
plt.ylabel("Score of Sample")
plt.title("HMM Plot M = 32")
plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
plt.legend()
plt.show()
```



In [353]:

```
M = 33
opcodeEncoder(M, encoded_training) # Reducing noise by changing amount of
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Reduce symbols used

reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)

remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))


encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)


encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)


x_zbot_reduced = []
y_zbot_reduced = []
fileCounter = 0
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28, 25:
29, 24: 30, 189: 31}
```

```
In [354]: for sequence in encoded_noise_reduced_zbot[1000:]:
                  if(fileCounter<500):
                      array = np.array(sequence[0:500]).reshape(-1,1)
                      score = remodel.score(array)
                      x_zbot_reduced.append(fileCounter)
                      y_zbot_reduced.append(score)
                      fileCounter+=1
                  print(f"Score for sequence: {score}")
```

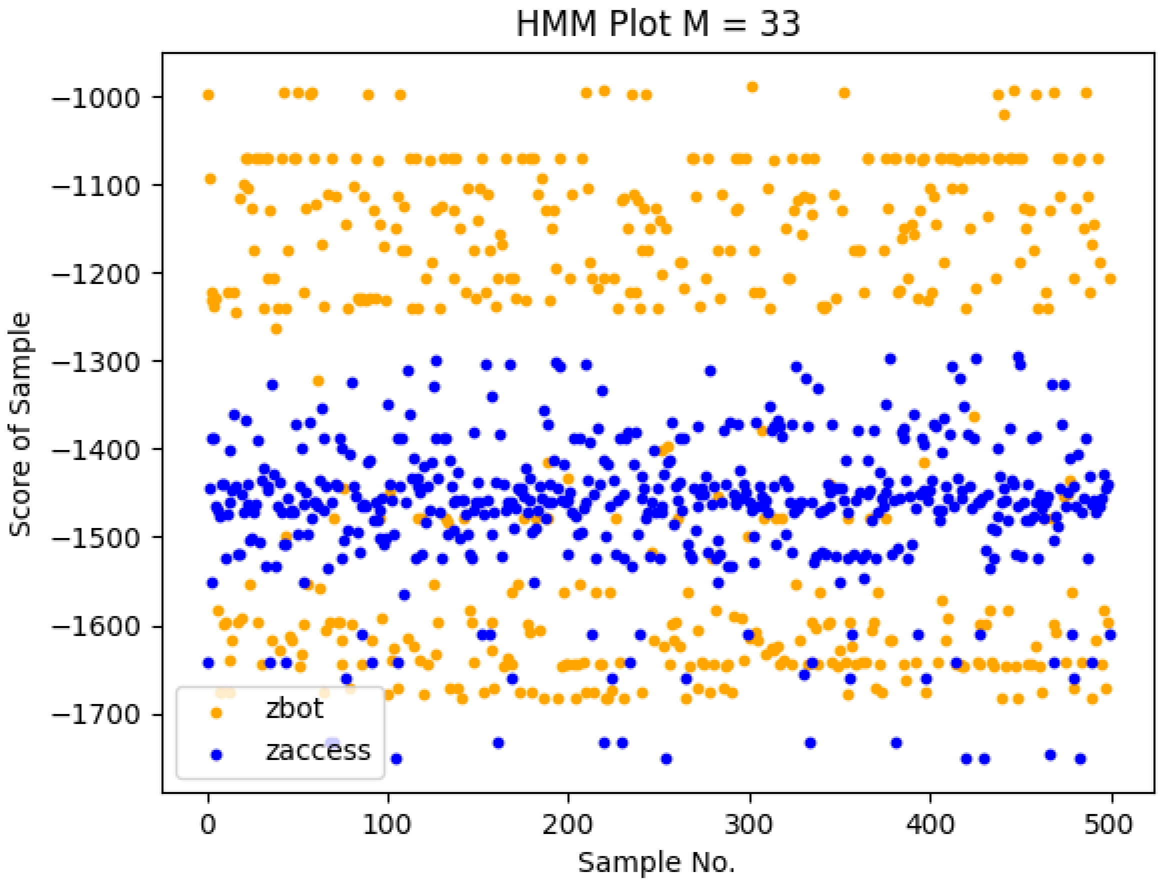
```
Score for sequence: -997.7840342335637
Score for sequence: -1093.3843391826254
Score for sequence: -1223.4692242270442
Score for sequence: -1231.3303788562846
Score for sequence: -1237.3504012460833
Score for sequence: -1228.2556242350142
Score for sequence: -1583.0639651475647
Score for sequence: -1674.9180381190006
Score for sequence: -1674.9180381190006
Score for sequence: -1597.573214771307
Score for sequence: -1595.3831852999722
Score for sequence: -1222.5670734798666
Score for sequence: -1638.6436336968418
Score for sequence: -1676.2282562583177
Score for sequence: -1616.2513051576996
Score for sequence: -1222.5670734798666
Score for sequence: -1245.5763908923461
Score for sequence: -1595.7986804224706
Score for sequence: -1116.896695426055
```

In [355]:

```python
x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1640.8390495273545
Score for sequence: -1445.2260207010772
Score for sequence: -1388.9298585017223
Score for sequence: -1550.0380240039078
Score for sequence: -1387.7365805828395
Score for sequence: -1464.9837030847723
Score for sequence: -1470.1519815355834
Score for sequence: -1476.3328242519856
Score for sequence: -1439.0914619563782
Score for sequence: -1439.9584577283963
Score for sequence: -1524.7217761677255
Score for sequence: -1475.0419173008615
Score for sequence: -1459.8786439897665
Score for sequence: -1401.9813991198102
Score for sequence: -1446.317303172205
Score for sequence: -1360.0262256019237
Score for sequence: -1443.1459438176355
Score for sequence: -1519.1334645577133
Score for sequence: -1519.509948831475
```

```
In [356]: plt.xlabel("Sample No.")
          plt.ylabel("Score of Sample")
          plt.title("HMM Plot M = 33")
          plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
          plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
          plt.legend()
          plt.show()
```



In [357]:

```
M = 29
opcodeEncoder(M, encoded_training) # Reducing noise by changing amount of
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Reduce symbols used

reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)

remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))


encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)


encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)


x_zbot_reduced = []
y_zbot_reduced = []
fileCounter = 0
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28, 25:
29, 24: 30, 189: 31}
```

```
In [358]: for sequence in encoded_noise_reduced_zbot[1000:]:
                  if(fileCounter<500):
                      array = np.array(sequence[0:500]).reshape(-1,1)
                      score = remodel.score(array)
                      x_zbot_reduced.append(fileCounter)
                      y_zbot_reduced.append(score)
                      fileCounter+=1
                  print(f"Score for sequence: {score}")
```

```
Score for sequence: -997.6691613699048
Score for sequence: -1093.3033997365228
Score for sequence: -1223.4834549924017
Score for sequence: -1231.3133281922842
Score for sequence: -1237.5228311534934
Score for sequence: -1228.2703933894413
Score for sequence: -1583.0916730718536
Score for sequence: -1674.78213375462
Score for sequence: -1674.78213375462
Score for sequence: -1597.6383164795004
Score for sequence: -1595.3319908000515
Score for sequence: -1222.569257218104
Score for sequence: -1638.2746190519958
Score for sequence: -1676.1371994110311
Score for sequence: -1615.9573923566488
Score for sequence: -1222.569257218104
Score for sequence: -1245.5132088606574
Score for sequence: -1595.6466881070037
Score for sequence: -1116.7717167034723
```

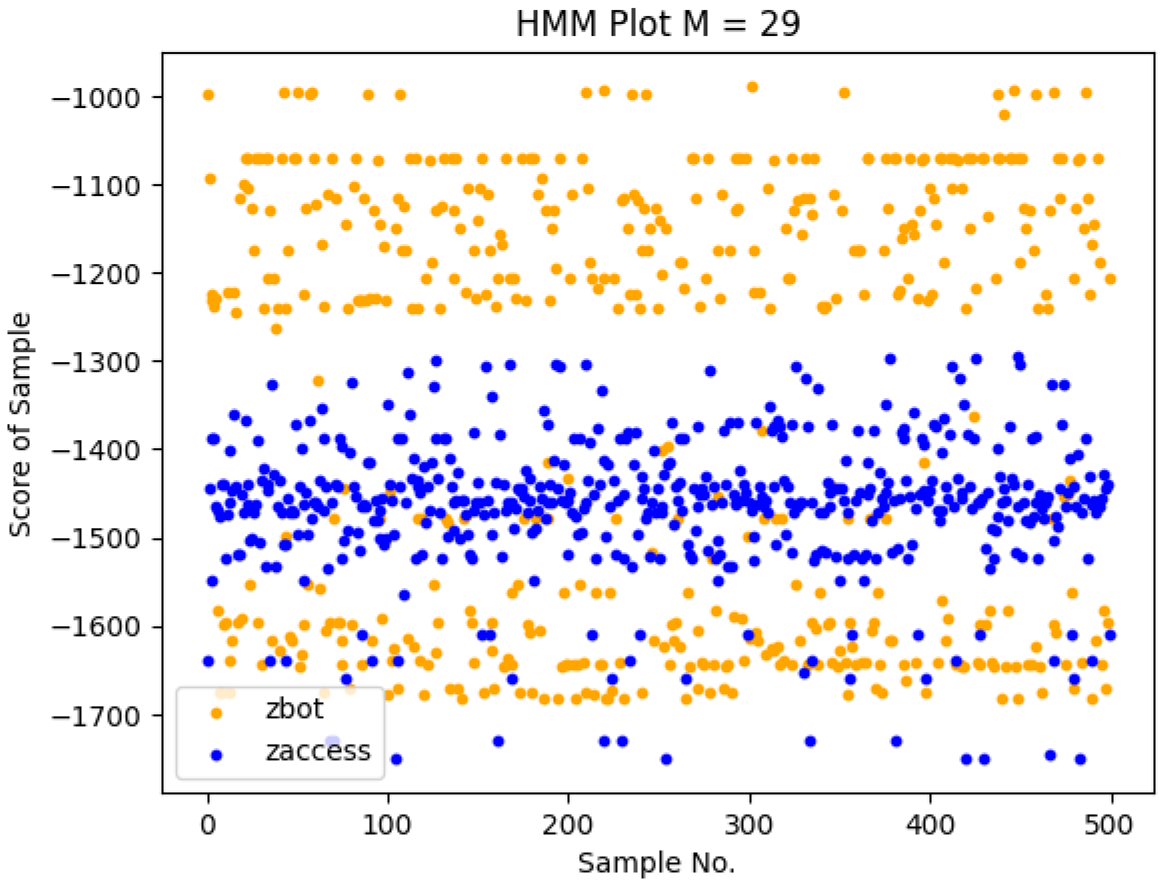
In [359]:

```

x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1388.2188791744459
Score for sequence: -1441.540669881063
Score for sequence: -1535.8982871068838
Score for sequence: -1730.3085371079783
Score for sequence: -1459.7067194926417
Score for sequence: -1730.3085371079783
Score for sequence: -1439.687591223048
Score for sequence: -1441.5165775452276
Score for sequence: -1386.8917589396453
Score for sequence: -1397.8231409485288
Score for sequence: -1524.5780430901664
Score for sequence: -1503.6977089304585
Score for sequence: -1660.5738335030123
Score for sequence: -1491.9419296555097
Score for sequence: -1404.6412265484073
Score for sequence: -1325.0846044177651
Score for sequence: -1442.500212156377
Score for sequence: -1453.0580169442462
Score for sequence: -1495.0316340674876
Score for sequence: -1515.622443266512
```

```
In [360]: plt.xlabel("Sample No.")
          plt.ylabel("Score of Sample")
          plt.title("HMM Plot M = 29")
          plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
          plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
          plt.legend()
          plt.show()
```



In [361]:

```
M = 28
opcodeEncoder(M, encoded_training) # Reducing noise by changing amount of
encoded_noise_reduced = []
for code in encoded_training:
    if code in opcode_vocab:
        encoded_noise_reduced.append(code)
    else:
        encoded_noise_reduced.append(M) # Reduce symbols used

reshape = np.array(encoded_noise_reduced)
encoded_noise_reshaped = reshape.reshape(-1,1)

remodel = hmm.CategoricalHMM(n_components=2,random_state = 42, n_iter=100)
remodel.fit(encoded_noise_reshaped, len(encoded_noise_reshaped))


encoded_noise_reduced_zbot = []
for file in encoded_zbot:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zbot.append(temp_matrix)


encoded_noise_reduced_zaccess = []
for file in encoded_zacc:
    temp_matrix = []
    for code in file:
        if code in opcode_vocab:
            temp_matrix.append(code)
        else:
            temp_matrix.append(M)
    encoded_noise_reduced_zaccess.append(temp_matrix)


x_zbot_reduced = []
y_zbot_reduced = []
fileCounter = 0
```

```
{235: 0, 197: 1, 332: 2, 7: 3, 21: 4, 300: 5, 348: 6, 46: 7, 399: 8, 178: 9, 418:
10, 416: 11, 258: 12, 66: 13, 347: 14, 359: 15, 9: 16, 6: 17, 206: 18, 408: 19, 20
2: 20, 211: 21, 198: 22, 176: 23, 260: 24, 417: 25, 392: 26, 391: 27, 252: 28, 25:
29, 24: 30, 189: 31}
```

In [362]:

```
for sequence in encoded_noise_reduced_zbot[1000:]:
        if(fileCounter<500):
            array = np.array(sequence[0:500]).reshape(-1,1)
            score = remodel.score(array)
            x_zbot_reduced.append(fileCounter)
            y_zbot_reduced.append(score)
            fileCounter+=1
        print(f"Score for sequence: {score}")
```

```
Score for sequence: -1093.353046679329
Score for sequence: -1223.4782353664727
Score for sequence: -1231.3280211205567
Score for sequence: -1237.415392919206
Score for sequence: -1228.2661134947573
Score for sequence: -1583.0601178464774
Score for sequence: -1674.8834366816739
Score for sequence: -1674.8834366816739
Score for sequence: -1597.5807920183577
Score for sequence: -1595.3558769578908
Score for sequence: -1222.5699681988262
Score for sequence: -1638.5127888786096
Score for sequence: -1676.1807235888612
Score for sequence: -1616.1248743919116
Score for sequence: -1222.5699681988262
Score for sequence: -1245.5505637172603
Score for sequence: -1595.734515347709
Score for sequence: -1116.855219921691
Score for sequence: -1591.472464771089
Score for sequence:  1100.5326488377036
```

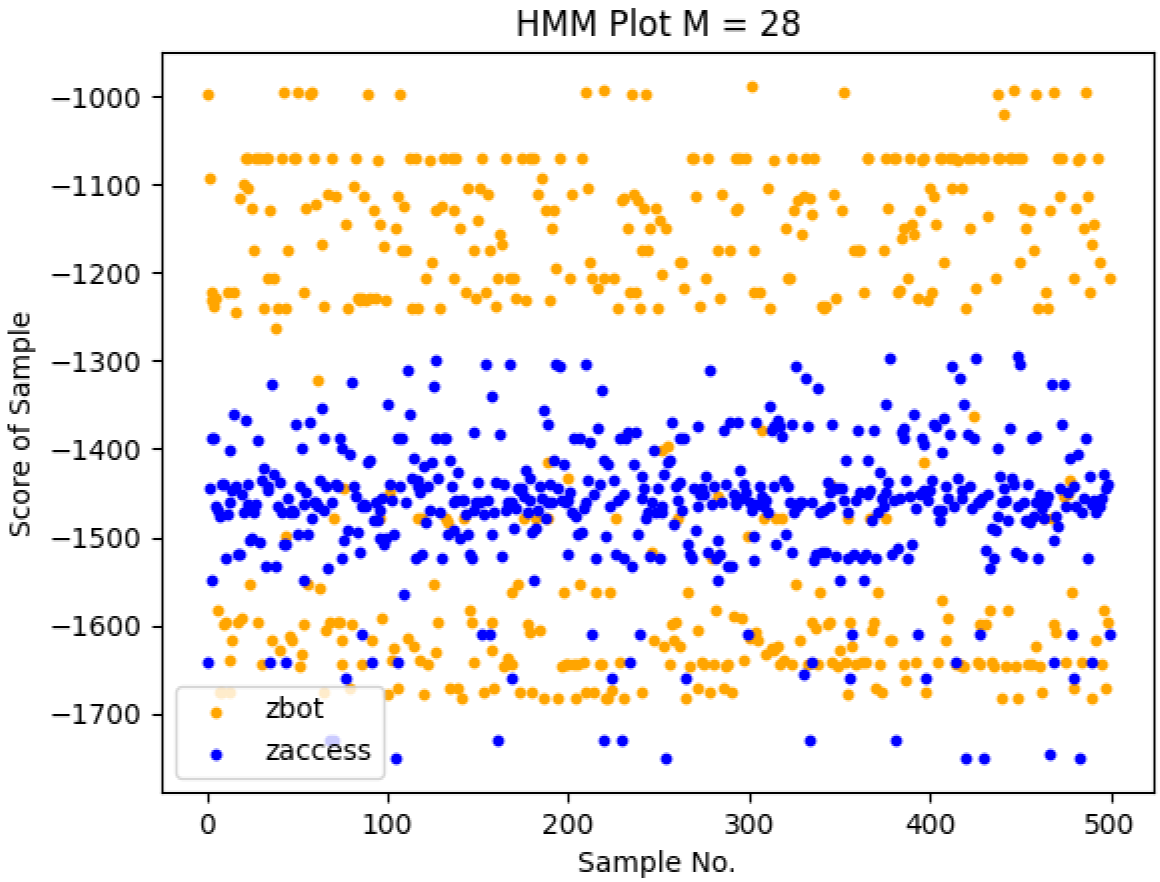
In [363]:

```python

x_zaccess_reduced = []
y_zaccess_reduced = []
fileCounter = 0
for sequence in encoded_noise_reduced_zaccess[704:]:
        if(fileCounter < 500):
            array = np.array(sequence[0:500])
            score = remodel.score(array.reshape(-1,1))
            x_zaccess_reduced.append(fileCounter)
            y_zaccess_reduced.append(score)
            fileCounter += 1
            print(f"Score for sequence: {score}")
```

```
Score for sequence: -1640.5874843123013
Score for sequence: -1444.9404352020738
Score for sequence: -1388.6318215667507
Score for sequence: -1549.6857458024947
Score for sequence: -1387.5272556215818
Score for sequence: -1464.890732981481
Score for sequence: -1469.922561412188
Score for sequence: -1476.295531844242
Score for sequence: -1439.0504374967697
Score for sequence: -1439.7487071728124
Score for sequence: -1524.6677575383642
Score for sequence: -1474.9236482737097
Score for sequence: -1459.8042908888858
Score for sequence: -1401.8464740597967
Score for sequence: -1446.2704596541219
Score for sequence: -1359.869526444512
Score for sequence: -1443.0328856839742
Score for sequence: -1518.9486002762415
Score for sequence: -1519.3287102064335
Score for sequence: -1451.6616042509987
```

```
In [364]: plt.xlabel("Sample No.")
          plt.ylabel("Score of Sample")
          plt.title("HMM Plot M = 28")
          plt.scatter(x_zbot_reduced, y_zbot_reduced, label = "zbot", color = "orange", s =10)
          plt.scatter(x_zaccess_reduced, y_zaccess_reduced, label = "zaccess", color = "blue",
          plt.legend()
          plt.show()
```



```
In [ ]:
```