Homework 1 **Anirudh Ganesh**
CSE 5523: Statistical Machine Learning Due Date: 03/06/18
Prof. Mikhail Belkin
TA: Siyuan Ma

1 Question 1:

The feature space consists of three possible points (events) A, B, C, which occur with probability 0.1,0.4,0.5, respectively, For each event there are two possible labels +1 or -1, which happen with probability 0.9,0.3 and 0.8 respectively (that is, $P(1|A) = 0.9, P(1|B) = 0.3$, etc.). Determine the Bayes optimal classifier. What is the expected loss of the Bayes optimal classifier?

Soln

$P(A) = 0.1$
$P(B) = 0.4$
$P(C) = 0.5$
$P(1|A) = 0.9$ and $P(-1|A) = 0.1$
$P(1|B) = 0.3$ and $P(-1|B) = 0.7$
$P(1|C) = 0.8$ and $P(-1|C) = 0.2$
From Bayes Rule we have,

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

Thus,

$$P(x|1)P(1) = P(1|x)P(x) \forall x = \{A, B, C\}$$

Hence $C^*(x) = 1$ if $P(1|x)P(x) > P(-1|x)P(x), -1$ otherwise
Thus $C*(A) = 1, C*(B) = -1, C*(C) = 1$
Expected Loss = $0.01 + 0.12 + 0.10 = 0.23$

2 Question 2:

A probability distribution on the real line is a mixture of two classes +1 and -1 with density N(1,2) (normal distribution with mean 1 and variance 2) and N(4,1), with prior probabilities 0.4 and 0.6 respectively. What is the Bayes decision rule? Give an estimate for the Bayes risk.

Soln

For $N(1,2)$, $P(X|Y=1)P(Y=1)$ For $N(4,1)$, $P(X|Y=-1)P(Y=-1)$

$$\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

For $N(1,2)$, $\frac{1}{2\sqrt{2\pi}}e^{\frac{(x-1)^2}{8}}$

For $ N(4,1)$, $\frac{1}{\sqrt{2\pi}}e^{\frac{(x-4)^2}{2}}$

Solving for x such that x in N(1,2) = x in N(4,1) we get $x = 2.42$

Thus, we have $P(X > 2.42) = 1 - z(\frac{2.42-1}{\sqrt{2}}) = 1 - z(0.71) = 1 - 0.8413 = 0.1587$

and $P(X \leq 2.42) = z(\frac{2.42-4}{1}) = z(-1.58) = 1 - 0.0571$

Hence total error = $0.1587 + 0.0571 = 0.2158$

Bayes Decision Rule = 1 if X $\leq$ 2.42 and -1 otherwise

```
In [14]: import matplotlib.mlab as mlab
         import math

         mu1 = 1
         variance1 = 2
         sigma1 = math.sqrt(variance)
         x1 = np.linspace(mu1 - 3*sigma1, mu1 + 3*sigma1, 100)
         plt.plot(x1,mlab.normpdf(x1, mu1, sigma1))

         mu2 = 4
         variance1 = 1
         sigma2 = math.sqrt(variance)
         x2 = np.linspace(mu2 - 3*sigma2, mu2 + 3*sigma2, 100)
         plt.plot(x2,mlab.normpdf(x2, mu2, sigma2))
         plt.title('Illustration of the two classes in Q2')
         plt.show()
```


Illustration of the two classes in Q2
0.25
0.20
0.15
0.10
0.05
0.00
−2
0
2
4
6
8

3 Question 3:

Consider a k-NN classifier for a 2-class problem. What is its expected (classification) loss and how does it compare to the Bayes optimal, when k = 3, assuming you have sufficiently many data points? How does the empirical loss of 3-NN compare to the Bayes optimal? (Recall that the empirical loss of 1-NN is zero).

Soln

Let $P(Y = 1|X) = p_x$

Let $min(p_x, 1 - p_x) = \eta$

The k-NN loss is given by,

$$L(k) = E[\eta] + E[(1 - 2\eta)Pr\{B(k, p_x) > \frac{k}{2}|X\}]$$

Where $Pr\{X = k\} = \begin{pmatrix} n \\ k \end{pmatrix} p_k(1 - p)^{n-k}$ (Binomial Distribution)

And, $B(n, p)$ denotes the Binomial Probability Mass Function.

We can see from this that the empirical loss of k-NN denoted by $R^{(3)}$ is still bounded by R^* and $2R^*(1 - R^*)$, i.e.,

$$R^* \leq R^{(3)} \leq 2R^*(1 - R^*)$$

Where R^* is the bayes optimal.

4 Question 4:

Generate 2000 points from two equally weighted spherical Gaussians $N(0, I), N((3, 0, 0, ...), I)$ in $\mathbb{R}^p, p = 1, 11, 21..101$ (note, you have to first flip a coin to decide from which Gaussian to sample), where I is the identity matrix and the centers of Gaussians are distance 3 apart. Implement 1-NN and 3-NN classifiers. Test the resulting classifier on a separately generated dataset with 1000 pts. Plot the error rate as a function of p. Observations?

Soln

For $p = 1$, sigma = 1, mean = 0 for Distribution 1 and sigma = 1, mean = 3 for Distribution 2

```
In [1]: import numpy as np
        from sklearn import neighbors
        import matplotlib.pyplot as plt

        %matplotlib inline
```

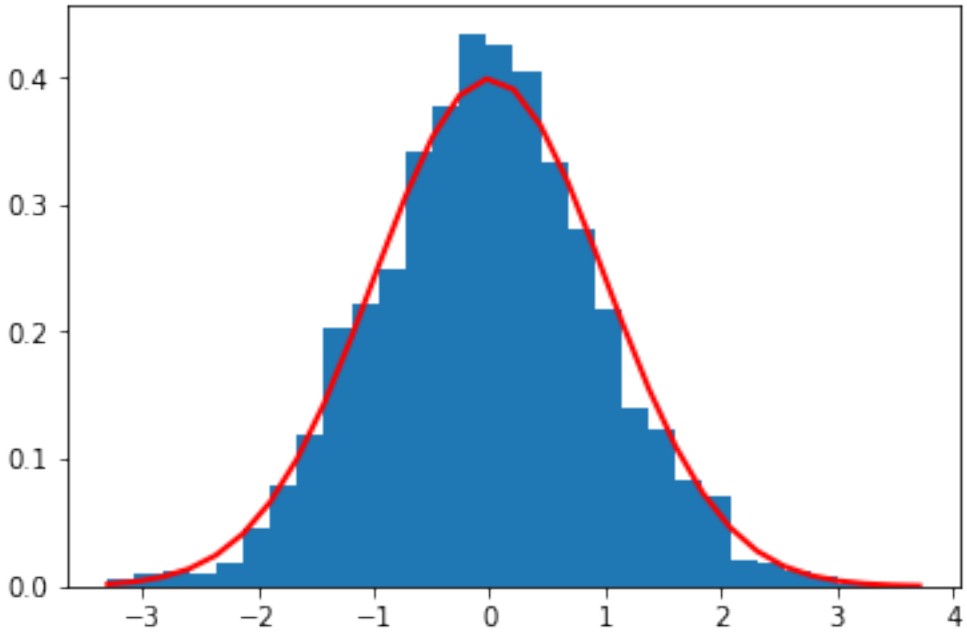
```
In [2]: NUMBER_OF_POINTS = 2000
```

```
In [3]: mu1 = 0
        mu2 = 3

        sigma = 1

        S1 = np.random.normal(mu1, sigma, NUMBER_OF_POINTS)
        S2 = np.random.normal(mu2, sigma, NUMBER_OF_POINTS)
```

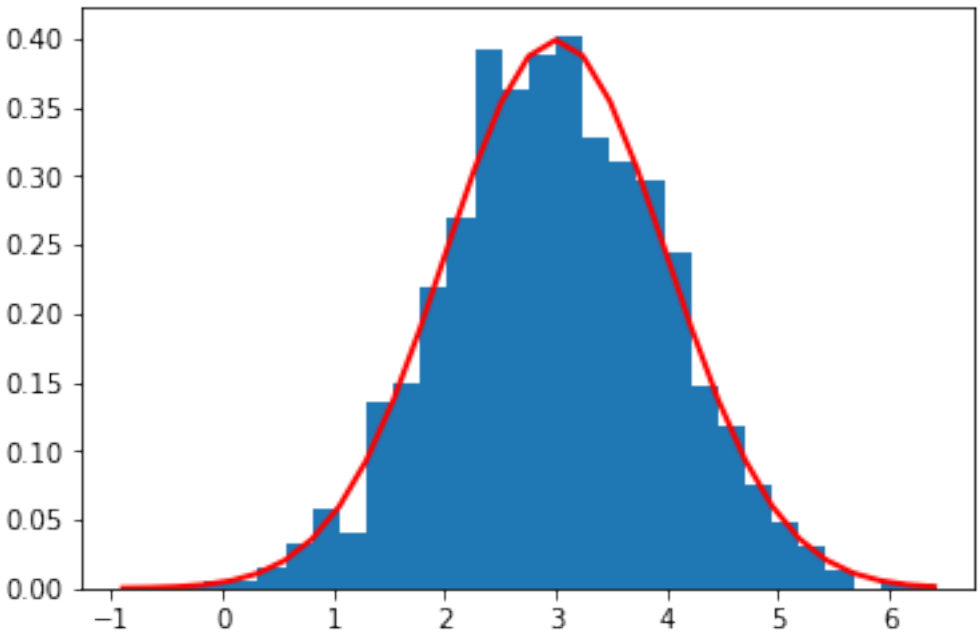
```
In [4]: print(S1)
        print(S2)
```

```
[-0.18297939 -1.24099397  0.64318915 ...  0.94583058  0.8705611
 -0.39329526]
[3.364064   2.35427502 2.88816673 ... 1.85067063 2.5260621  2.18671577]
```

```
In [5]: count, bins, ignored = plt.hist(S1, 30, normed=True)
        plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (bins - mu1)**2 / (2 * sigma**
        plt.show()
```



In [6]:
```
count, bins, ignored = plt.hist(S2, 30, normed=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (bins - mu2)**2 / (2 * sigma**
plt.show()
```



```
In [7]: sample = np.zeros(NUMBER_OF_POINTS)

        for i in range(0, 2000):
            r = np.random.random_sample()
            sample[i] = S2[i] if r >= 0.5 else S1[i]

In [8]: from sklearn.neighbors import NearestNeighbors

        sample = sample.reshape(-1, 1)

        nbrs = NearestNeighbors(n_neighbors=3, algorithm='ball_tree').fit(sample)

        distances, indices = nbrs.kneighbors(sample)

        print(indices)
```

```
[[   0 1779 1728]
 [   1  860 1630]
 [   2  820 1596]
 ...
 [1997  829  495]
 [1998 1093 1694]
 [1999  662  560]]
```

Function to calculate 1NN and 3NN errors for higher dimensions (p = 11, 21 ... 110)

```
In [18]: def test_nn(p, N_train, N_test):
             cov = np.identity(p)
             mean1 = np.zeros(p,)
             mean2 = mean1
             mean2[0] = 3

             sample1 = np.random.multivariate_normal(mean1, cov, N_train).T
             sample2 = np.random.multivariate_normal(mean2, cov, N_train).T

             X_train = np.zeros((p, N_train))
             y_train = np.zeros(N_train)

             for i in range(0, N_train):
                 r = np.random.random_sample()
                 X_train[:, i] = sample2[:, i] if r >= 0.5 else sample1[:, i]
                 y_train[i] = 1 if r >= 0.5 else 0

             nbr1 = neighbors.KNeighborsClassifier(n_neighbors=1, algorithm='ball_tree').fit(X_t
             nbr3 = neighbors.KNeighborsClassifier(n_neighbors=3, algorithm='ball_tree').fit(X_t

             sample1 = np.random.multivariate_normal(mean1, cov, N_test).T
```

```
    sample2 = np.random.multivariate_normal(mean2, cov, N_test).T

    X_test = np.zeros((p, N_test))
    y_test = np.zeros(N_test)

    for i in range(0, N_test):
        r = np.random.random_sample()
        X_test[:, i] = sample2[:, i] if r >= 0.5 else sample1[:, i]
        y_test[i] = 1 if r >= 0.5 else 0

    error_nn1 = 0
    error_nn3 = 0


    for i in range(0, N_test):
        y_1 = nbr1.predict(X_test[:,i].reshape(-1,1).T).item()
        y_3 = nbr3.predict(X_test[:,i].reshape(-1,1).T).item()

        if y_1 != y_test[i]:
            error_nn1 += 1

        if y_3 != y_test[i]:
            error_nn3 += 1

    return (error_nn1*100.0/N_test), (error_nn3*100.0/N_test)
```

In [25]:
```
y1 = []
y3 = []

x = []

for i in range(11, 110, 10):
    x.append(i)
    a, b = test_nn(i, 2000, 1000)
    y1.append(a)
    y3.append(b)

plt.plot(x, y1)
plt.plot(x, y3)
plt.legend(['1-NN', '3-NN'])
plt.title('Plot of error rate vs p')
plt.show()
```

4.0.1 Observation

We can observe the *Curse of dimensionality* firsthand here, increasing the dimensions causes no significant change in terms of its error rate. Even the starkest contrast is within a few percent of each other, and pretty much both of them are around random guess level of performance, which is pretty bad. This is due to the points in higher dimensions being extremely sparse, this causes them to loose their properties that tie them to the distribution. Hence both our classifiers perform very poorly.

5 Question 5:

Soln

What is the VC-dimension of the set of indicator functions of disks in $\mathbb{R}^2$ (i.e., functions which are 1 inside a circle -1 outside (but not the other way around!))? What about the indicator functions of rectangular boxes with sides parallel to the axes? You need to explain why.

5.0.1 For function of disks

For 1 point, it is trivial, we can always draw a disk of a fixed radius with that point at the center, if the point is +1 and have it outside a circle if the point is -1.

For 2 points, also we see that its trivial, as we have only 4 combinations, +1, +1, in which case we can draw a circle with the two points diametrically opposite. For -1, +1 and +1, -1, we can draw a circle centered at +1 and radius such that -1 falls outside. for -1, -1 we can draw any arbitary circle such that the two points remain outside. Thus we can always shatter two points.

For 3 points, we can see that we can always have a circle that passes through 3 points, i.e. the circumcircle of the triangle formed by the points. For the other cases, we can degenrate into 2 point and 1 point. Thus we can see that we can always shatter 3 points.

For 4 points, we have that the 4 points form a quadrilateral, if not then it degenerates into the previous cases. Now in-order to see if the quadrilateral can be shattered, we need to check if the quadrilateral is concave. If it is, then it cannot be shattered as the circle cannot contain the three outside points without containing the one inside point. If we pair up the points as diagonals, then we see that if one pair of the diagonal points need to be separated from the other pair, then this pair needs to remain outside the circle formed our initial pair. But if the second pair is indeed outside the circle formed by our initial pair, then the initial pair will also lie outside the circle formed by our second pair. Thus we have a contradiction. Which means that we cannot shatter 4 points with a circle. Figure 1 below illustrates this.

Hence VC dimension is 3.

Note: But, in the question it was given that we are not able to invert the classifier, i.e., make the classifier classify -1 inside and +1 outside. In such a scenario, we can see that this set cannot shatter even 3 points. As if we take 3 co-linear points with -1 between two +1s then we are unable to form a circle that is able to shatter it. Figure 2 illustrates this.

In this scenario, **our VC Dimension is 2**.

5.0.2 For function of rectangular boxes

For 1 point, it is easy to prove as there is only two subsets that we can split the plane into. Thus its trivial to shatter this scenario.

For 2 points, we can always find an axis aligned rectangle such that all combination of two points can be accounted for.

For 3 points, we can come up with a set of axis aligned box given by the bounding the area of the triangle formed by the three point such that there exists an axis aligned box that contains none of the points, one point individually, two points and all three.

For 4 points, we can again come up with a set of axis aligned boxes. We divide the 4 points into 4 categories, top, bottom, left and right according to their extremity. Now we know that any combination of one, two or three points can be shattered by previous step. For the condition of including none of the points is also trivial. In order to contain all four points we can draw a

rectangle with width boudned by left and right and height bounded by top and bottom. Thus we can also shatter 4 points.

For 5 points, we see that one point will always remain inside the rectangle bounded by the extreme points and thus we cannot shatter.

Hence VC dimension is 4.

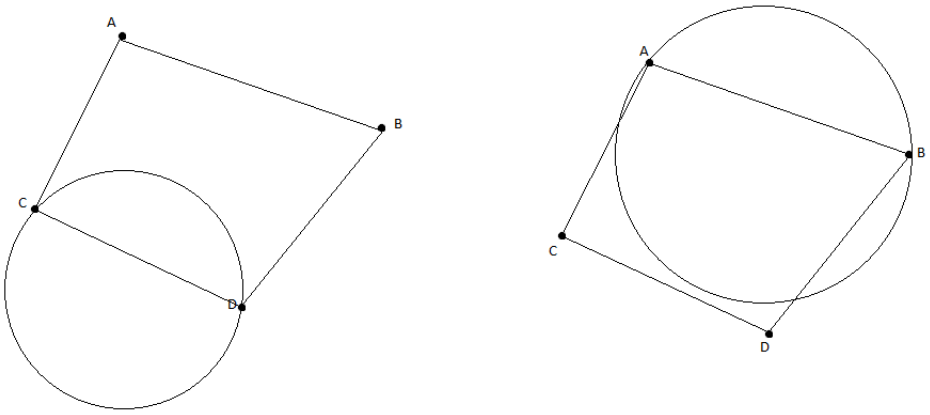


Figure 1: For describing 4 point problem

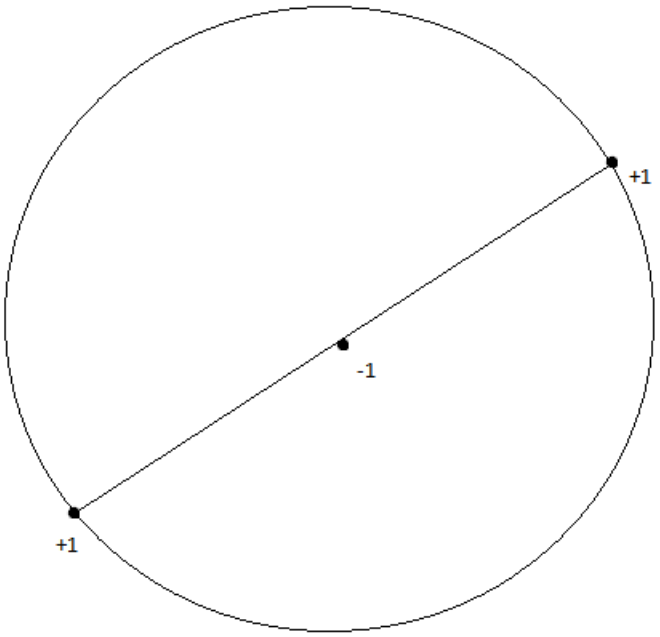


Figure 2: Explaining a special case for our question.