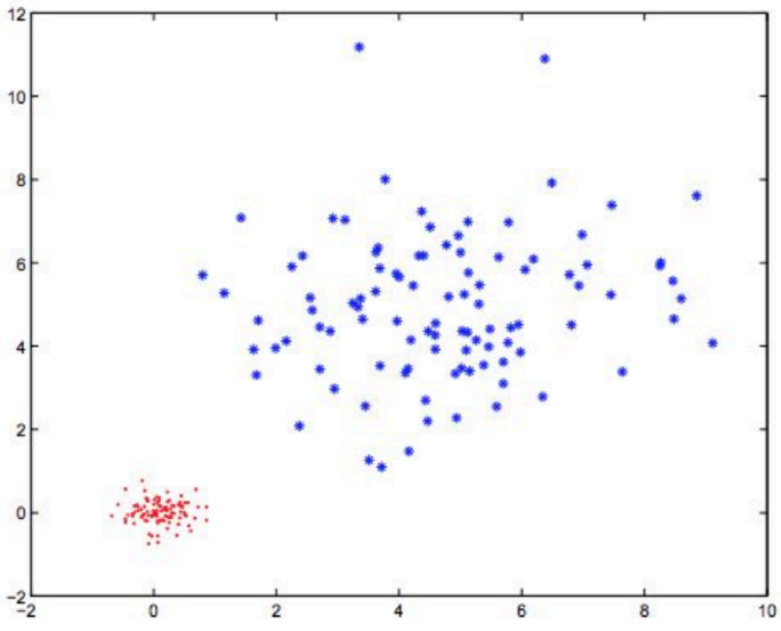
数据科学家直通车项目考试

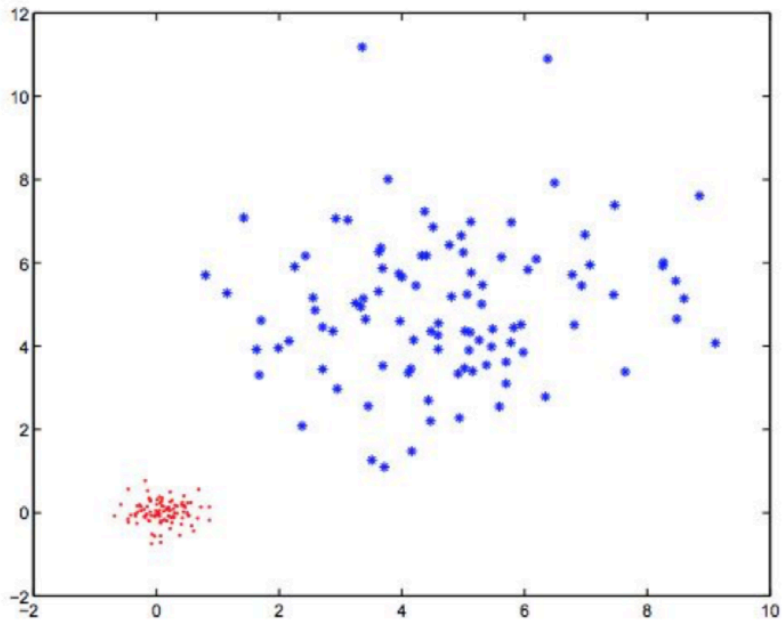
Phase 2 : Yelp Business推荐系统项目

Meina Wang

Q1 (10pts):

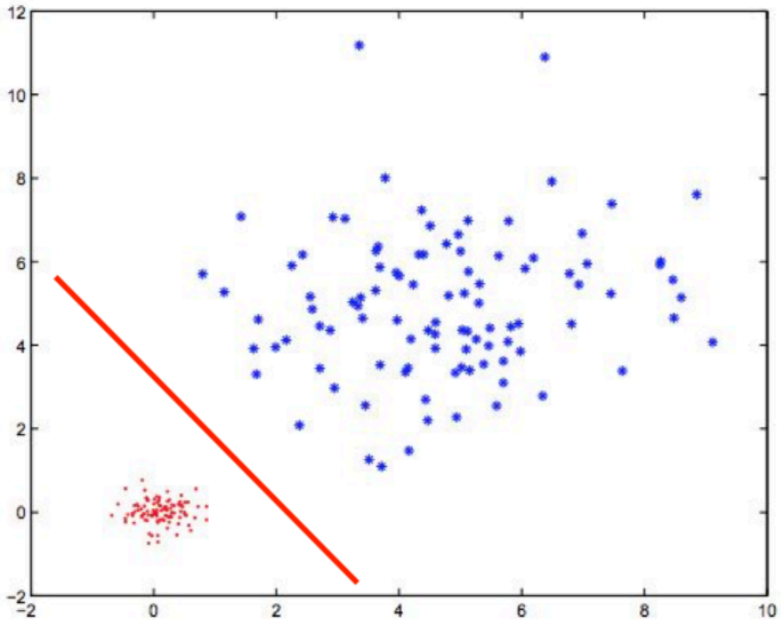
Let’s train a Gaussian Naive Bayes (GNB) model on the training set shown below. Assume that the dataset satisfies Gaussian Naive Bayes assumptions, and also assume that the variance is independent of instances but dependent on classes, i.e. σ ik = σ k where i represents the i-th instance X(i) and k ∈ 1, 2 indexes classes. Draw the decision boundaries when you train GNB:
(a: left figure) using the same variance for both classes, σ 1 = σ 2.
(b: right figure) using separate variance for each class σ 1 != σ 2

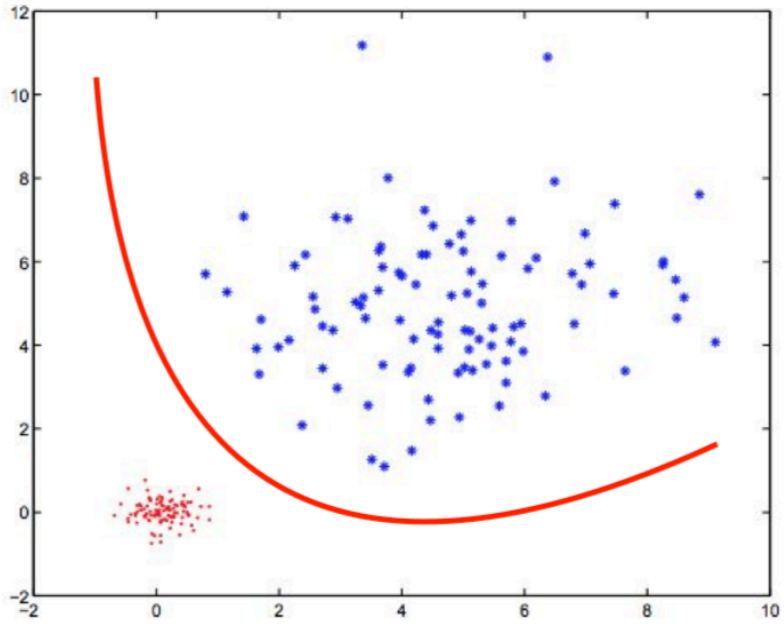




A1:

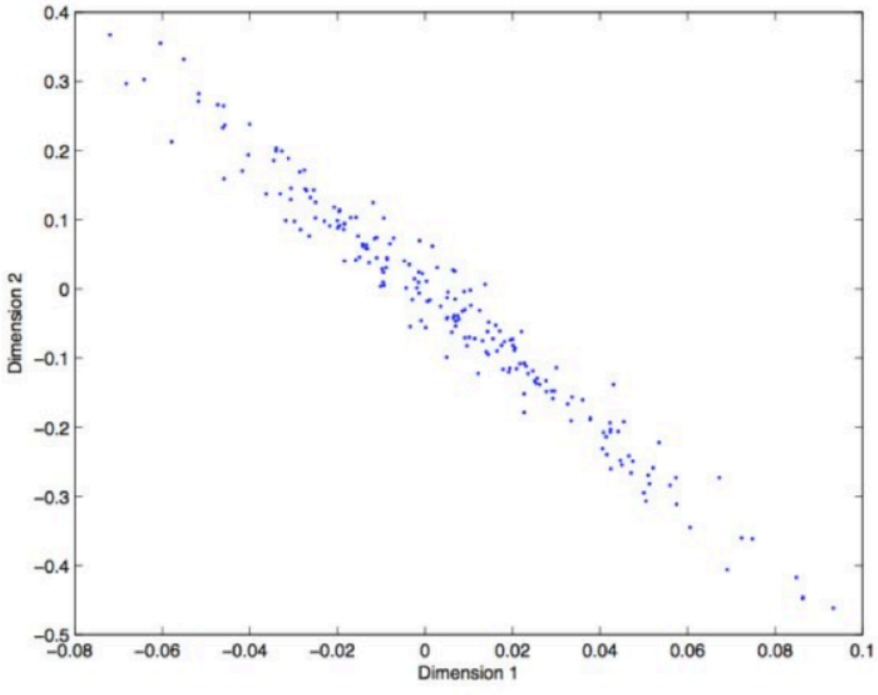
a: The decision boundary will be linear.
b: The decision boundary will be quadratic.

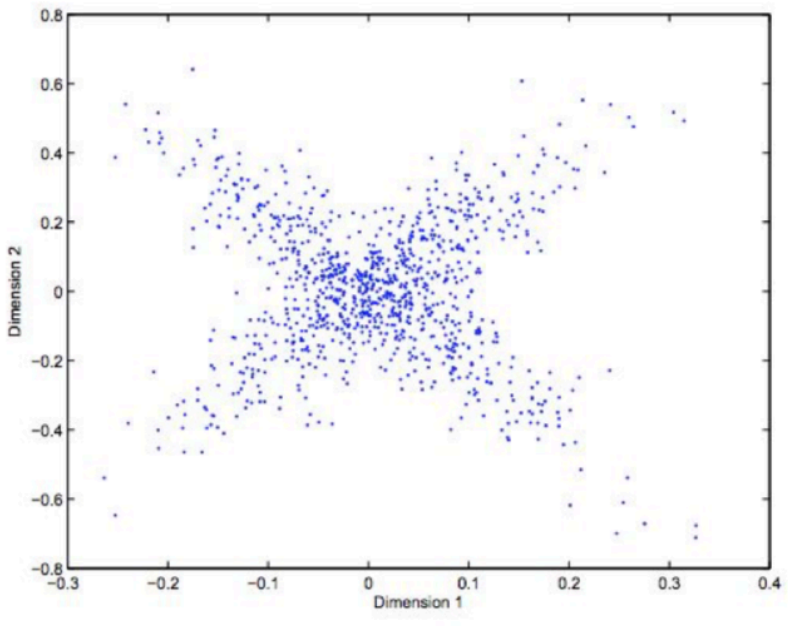




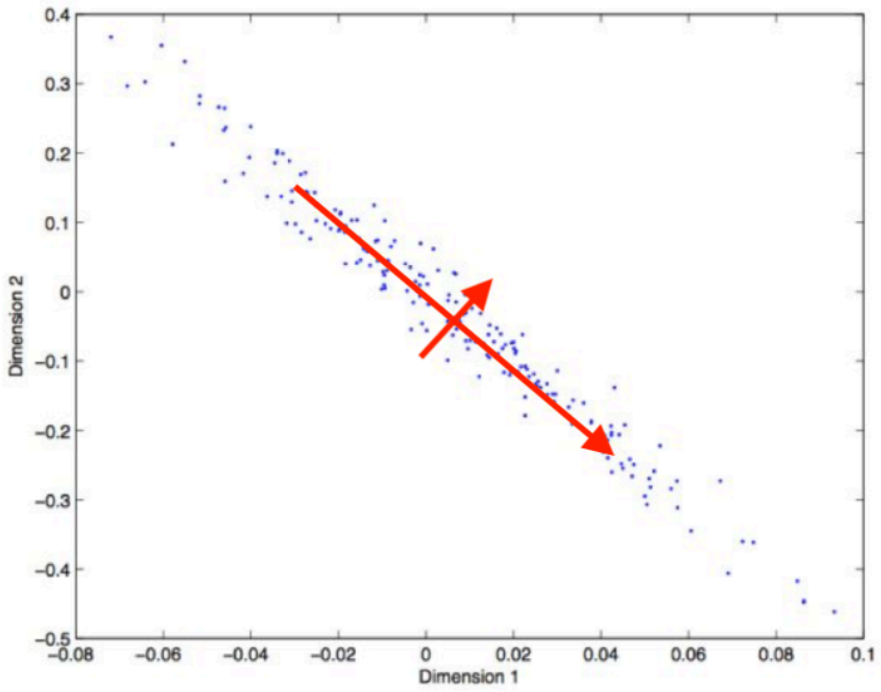
Q2 (10pts):

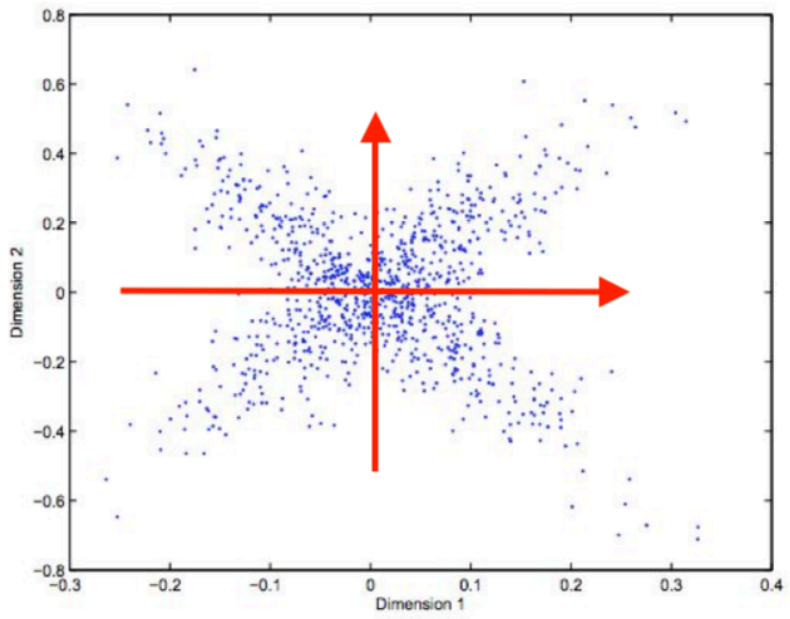
Principal component analysis is a dimensionality reduction method that projects a dataset into its most variable components. You are given the following 2D datasets, draw the first and second principle components on each plot.





A2:





Q3 (5pts):

For data D and hypothesis H , determine if the following equations must always be true.

(a)	$\sum_h P(H = h \mid D = d) = 1$
(b)	$\sum_h P(D = d \mid H = h) = 1$
(c)	$\sum_h P(D = d \mid H = h) P(H = h) = 1$

A3:

(a):Yes. The sum of probabilities given data d for all the hypothesis should always add up to 1.
(b):No.
(c):No.

Q4 (5 pts each, 10 pts total):

Assume we are trying to learn a decision tree. Our input data consists of N samples, each with k attributes (N >> k). The depth of a tree is defined as the maximum number of nodes between the root and any of the leaf nodes (including the leaf, not the root).
(a) If all attributes are binary, what is the maximal number of leaf (decision) nodes that we can have in a decision tree for this data? What is the maximum possible depth of a decision tree for this data (in Big-O notation)?
(b) If all attributes are continuous, what is the maximum number of leaf nodes that we can have in a decision tree for this data? What is the maximum possible depth for a decision tree for this data?

A4:

(a): 2^{k-1}. The maximum possible depth of a decision tree for this data is $O(k)$.
(b): If all attributes are continuous, the maximum number of leaf nodes that we can have in a decision tree for this data can be the same number as the data, which is N. The maximum depth for a decision tree for this data is also N.

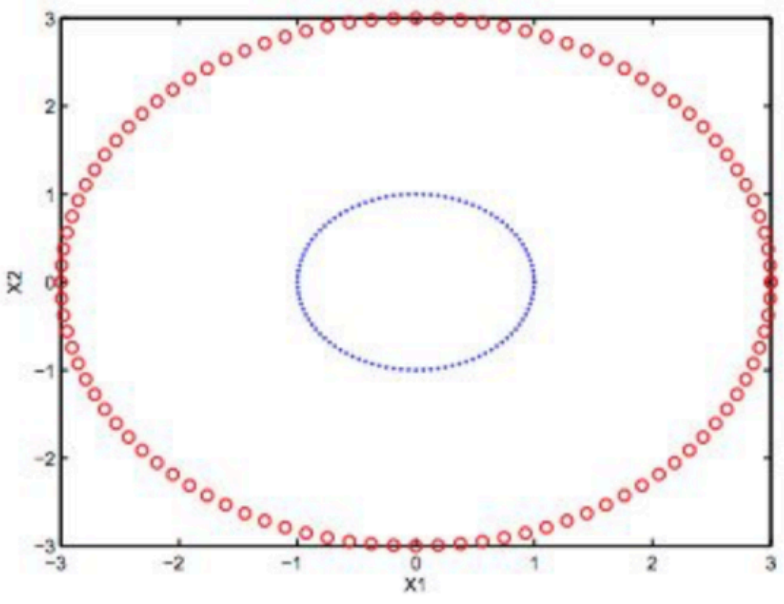
Q5 (10 pts each, 30 pts total):

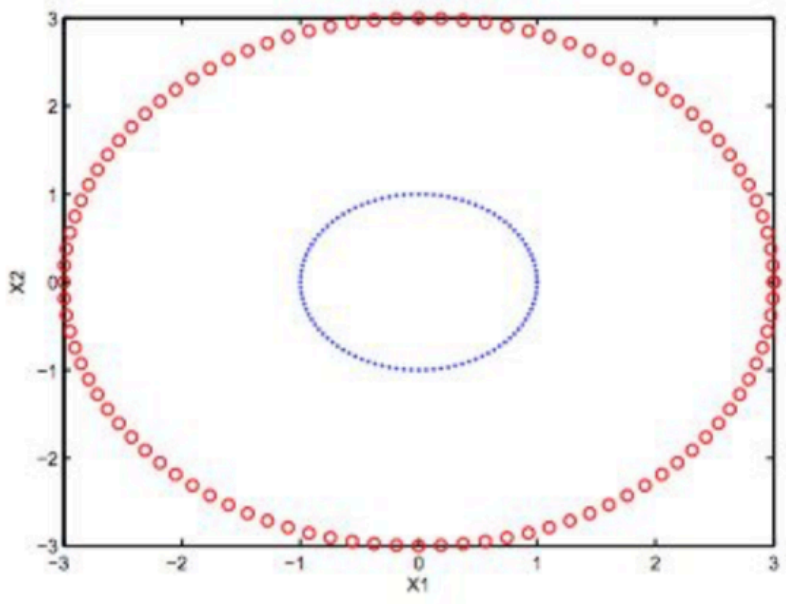
SVM and Kernels
(a) One of the most commonly used kernels in SVM is the Gaussian RBF kernel: k(x i, x j) =

$exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma}\right)$. Suppose we have three points, z_1, z_2, and x. z_1 is geometrically very close to x, and z_2 is geometrically far away from x. What is the value of $k(z_1, x)$ and $k(z_2, x)$?. Choose one of the following:

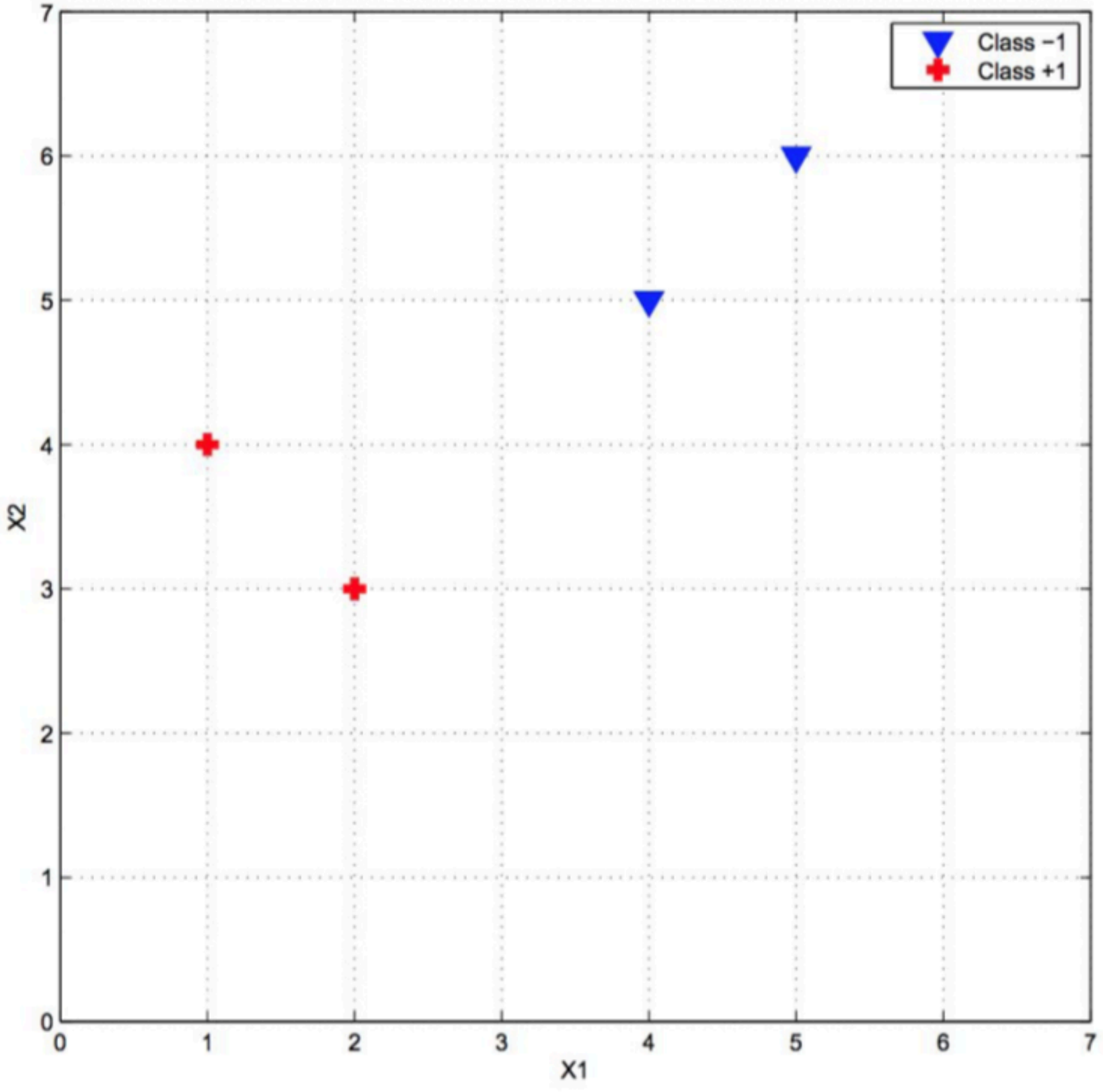
a. $k(z_1, x)$ will be close to 1 and $k(z_2, x)$ will be close to 0.
b. $k(z_1, x)$ will be close to 0 and $k(z_2, x)$ will be close to 1.
c. $k(z_1, x)$ will be close to c_1, $c_1 \gg 1$ and $k(z_2, x)$ will be close to c_2, $c_2 \ll 0$, where $c_1, c_2 \in R$
d. $k(z_1, x)$ will be close to c_1, $c_1 \ll 0$ and $k(z_2, x)$ will be close to c_2, $c_2 \gg 1$, where $c_1, c_2 \in R$

(b) Given the following 2 plots, which illustrates a dataset with two classes. Draw the decision boundary when an SVM is trained with linear & polynomial (order 2) kernels respectively. Classes have equal number of instances.



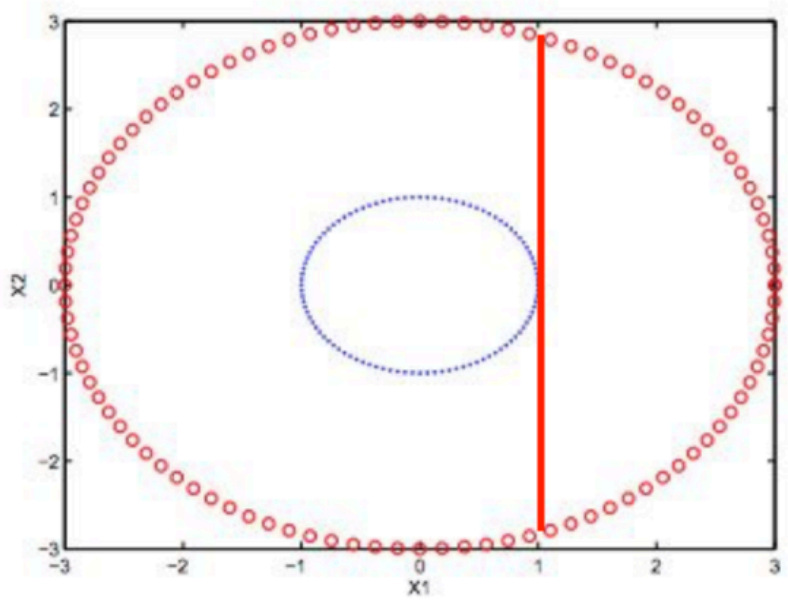


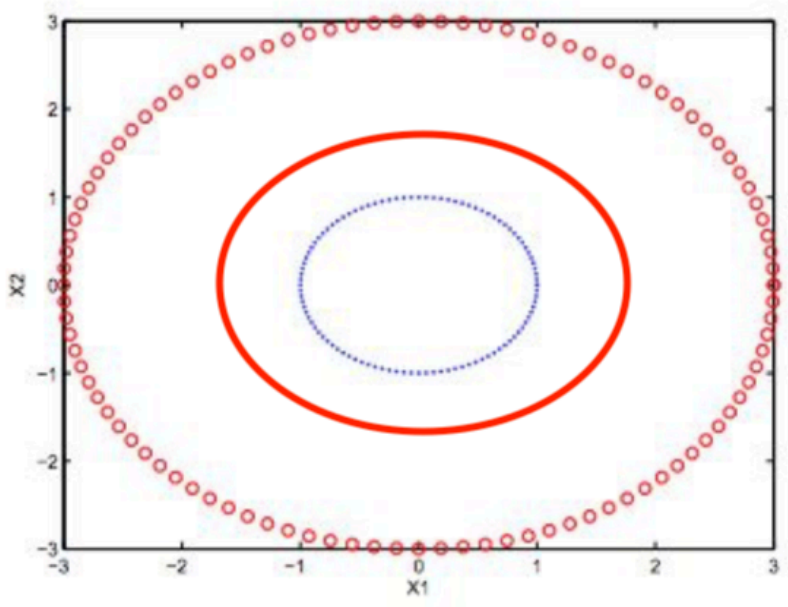
(c) Hard Margin SVM. Support vector machines learn a decision boundary leading to the largest margin from both classes. This time a SVM is trained on a tiny dataset with 4 points shown below. This dataset consists of two examples with class label -1 (denoted with plus), and two examples with class label +1 (denoted with triangles). Circle the support vectors and draw the decision boundary.



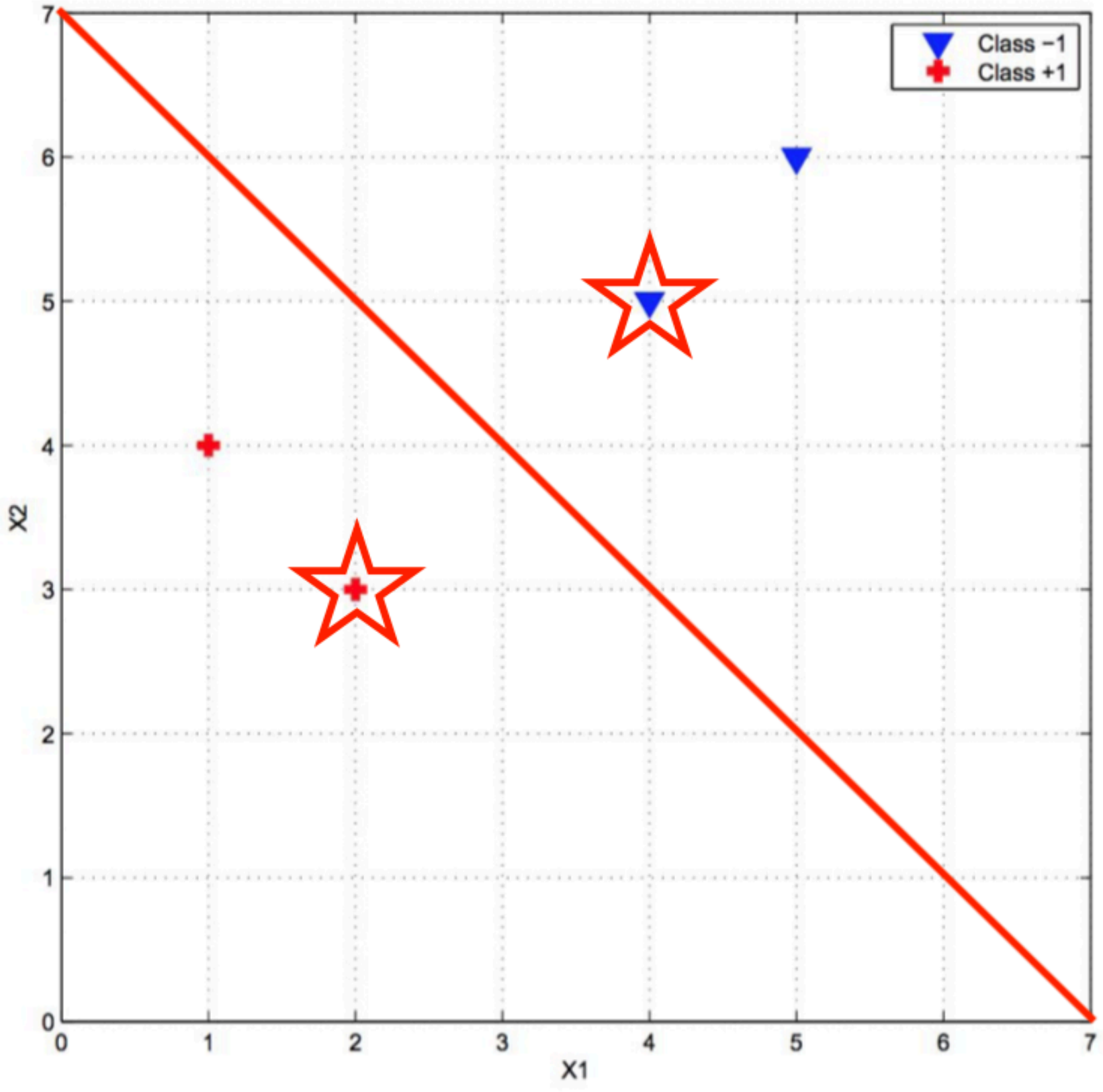
A5:

(a): a. Since k(z1, x) is closer to x, it will be close to 1. While k(z2, x) is far, it will be close to 0.
(b):





(c): Two stars are the two support vectors, and the red line is the decision boundary.


Class −1
Class +1
X2
X1
0
1
2
3
4
5
6
7

Q6 (5 pts each, 15 pts total):

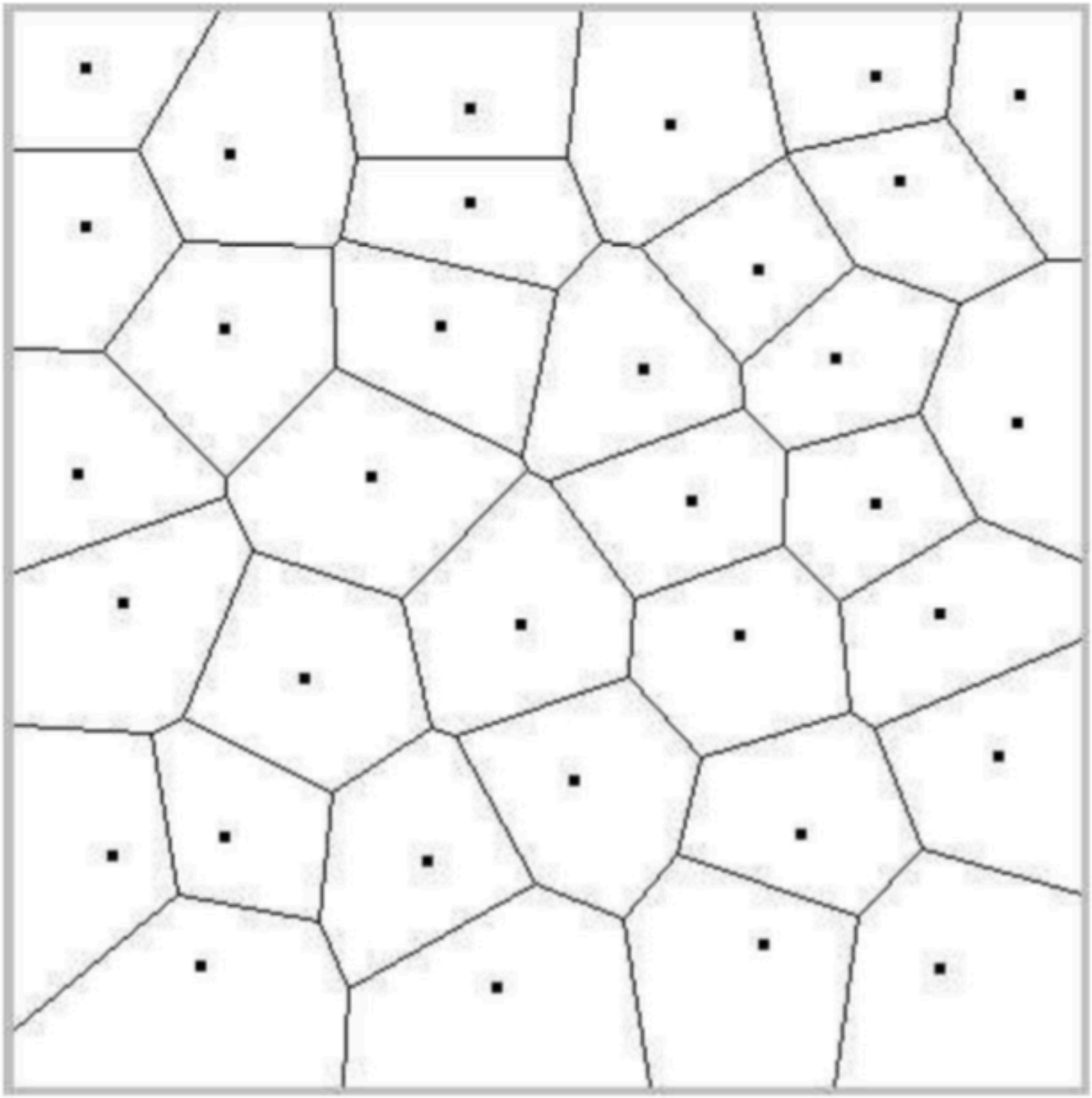
Feature and model selection.
(a) Consider learning a classifier in a situation with 1000 features total. 50 of them are truly informative about class. Another 50 features are direct copies of the first 50 features. The final 900 features are not informative. Assume there is enough data to reliably assess how useful features are, and the feature selection methods are using good thresholds. How many features will be selected by mutual information?
(b) Consider k-fold cross-validation. Let's consider the tradeoffs of larger or smaller k (the number of folds). With a higher number of folds, the estimated error will be, on average,
(i) Higher,
(ii) Lower,
(iii) Same or
(iv) Can't tell
(c) We are trying to learn regression parameters for a dataset which we know was generated from a polynomial of a certain degree, but we do not know what this degree is. Assume the data was actually generated from a polynomial of degree 5 with some added Gaussian noise (that is $y = w + wx + wx^2 + wx^3 + wx^4 + wx^5 + \varepsilon, \varepsilon \sim N(0, 1)$). For training we have 100 {x,y} pairs and for testing we are using an additional set of 100 {x,y} pairs. Since we do not know the degree of the polynomial we learn two models from the data. Model A learns parameters for a polynomial of degree 4 and model B learns parameters for a polynomial of degree 6. Which of these two models is likely to fit the test data better?

A6:

(a): ~100 features.
(b): (ii) Lower. We have more training data with more folds.
(c): Model A is likely to fit the test data better, since it is simplier than model B, it may generalize better to test data.

Q7 (5pts):

Let us try and classify data points in 2D Euclidean space. We are given n instances of such points $P_1, P_2, \ldots, P_n$ and the corresponding category for each point C_1, C_2, C_n[where $C_1, C_2, .. C_n$ take values from the set of all possible class labels]. Under the k nearest neighbors classification scheme, each new element Q is simply categorized by a majority vote among its k nearest neighbors in instance space. The 1-NN is a simple variant of this which divides up the input space for classification purposes into a convex region (see figure below for the 1NN decision boundaries under the Euclidean distance measure), each corresponding to a point in the instance set.



Is it possible to build a decision tree (with decisions at each node of the form “is x > a”, “is x < b”, “is y > c”, or “is y < d” for any real constants a,b,c,d) which classifies exactly according to the 1-NN scheme using the Euclidean distance measure ? If so, explain how. If not, explain why not.

A7:

No.
The boundaries from decision tree are always parallel to x axis and y axis. However, for this case, the decision boundaries for 1 - NN of each data point are not necessarily parallel to the x or y axis. Therefore, the answer is it is not possible to build a decision tree.

Q8 (10pts):

K-mean clustering & visualization. Start from the following Python codes and visualize K-mean clustering results where K = 3 & 8.

A8:

In [76]:

```
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
# Though the following import is not directly being used, it is required # for 3D pl
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn import datasets
np.random.seed(5)
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

In [84]:

```
X.shape
```

Out[84]:

```
(150, 4)
```

In [85]:

```
y.shape
```

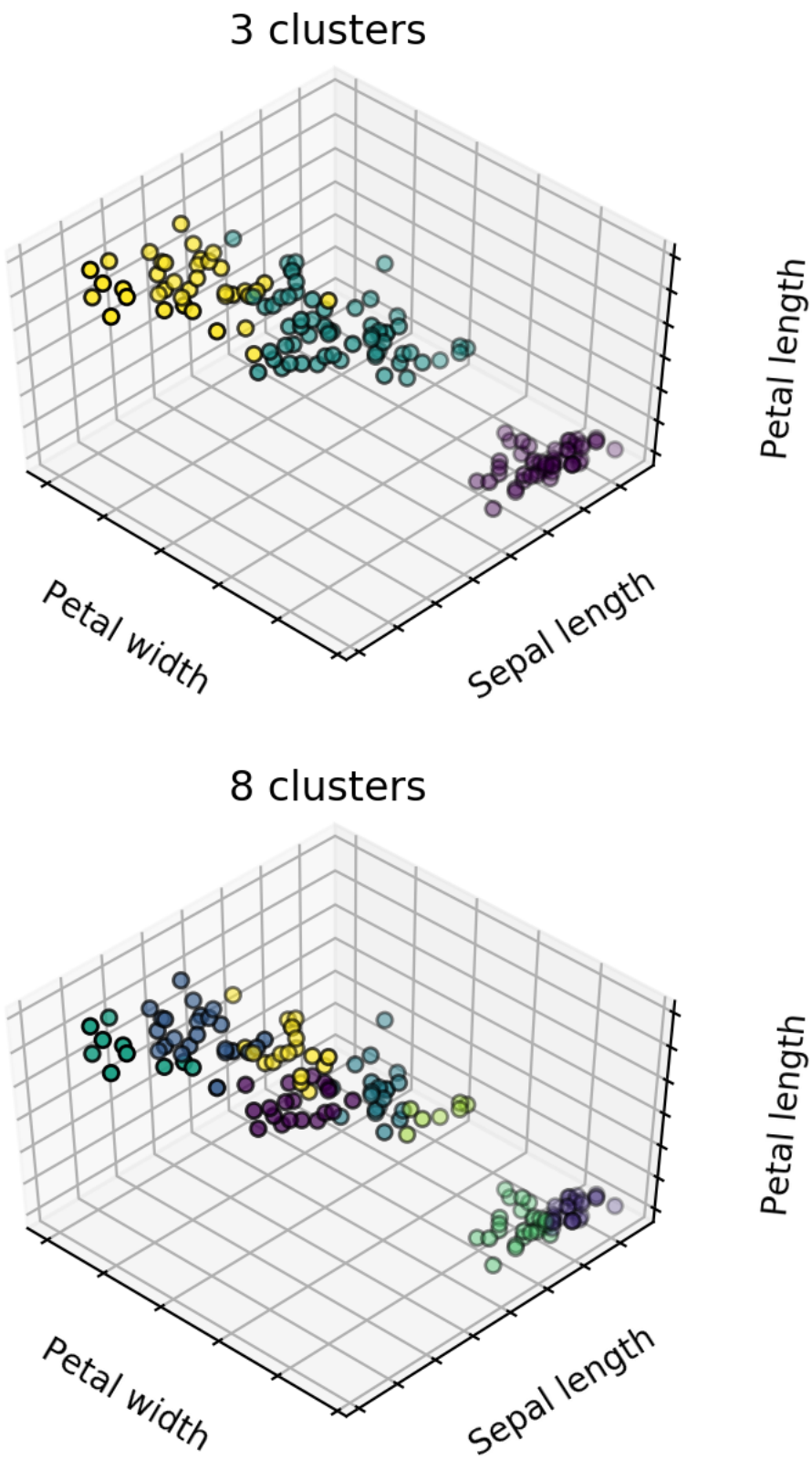
Out[85]:

```
(150,)
```

In [87]:

```
estimators = [('k_means_iris_3', KMeans(n_clusters=3)),
              ('k_means_iris_8', KMeans(n_clusters=8))]

fignum = 1
titles = ['3 clusters', '8 clusters']
for name, est in estimators:
    fig = plt.figure(fignum, figsize=(4, 3))
    ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=50, azim=134)
    est.fit(X)
    labels = est.labels_

    ax.scatter(X[:, 3], X[:, 0], X[:, 2],
```

```
                c=labels.astype(np.float), edgecolor='k')

    ax.w_xaxis.set_ticklabels([])
    ax.w_yaxis.set_ticklabels([])
    ax.w_zaxis.set_ticklabels([])
    ax.set_xlabel('Petal width')
    ax.set_ylabel('Sepal length')
    ax.set_zlabel('Petal length')
    ax.set_title(titles[fignum - 1])
    ax.dist = 12
    fignum = fignum + 1

fig.show()
```



Q9 (5pts):

Make prediction with SVM. Start from the following Python codes and configure a SVM to achieve test accuracy > 0.96

A9:

In [10]:

```
import numpy as np
from sklearn import datasets
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
d = datasets.load_digits()
X = d.data
y = d.target
```

In [12]:

```
X.shape
```

Out[12]:

```
(1797, 64)
```

In [13]:

```
y.shape
```

Out[13]:

```
(1797,)
```

In [65]:

```
svc = svm.SVC(C=1, kernel='linear')
```

In [66]:

```
X_folds = np.array_split(X, 10)
y_folds = np.array_split(y, 10)
```

In [67]:

```
scores = list()
```

In [68]:

```
for k in range(10):
    X_train = list(X_folds)
    X_test = X_train.pop(k)
    X_train = np.concatenate(X_train)
    y_train = list(y_folds)
    y_test = y_train.pop(k)
    y_train = np.concatenate(y_train)
    scores.append(svc.fit(X_train, y_train).score(X_test, y_test))
print(scores)
```

```
[0.9388888888888889, 0.9944444444444445, 0.9333333333333333, 0.9666666
666666667, 0.9611111111111111, 0.9888888888888889, 0.9666666666666667,
0.9888268156424581, 0.9329608938547486, 0.9664804469273743]
```

In [69]:

```
print(np.mean(scores))
```

```
0.963826815642458
```