# Applied Data Mining Midterm Practice

*Xun Zhao, xz2827*

## Problem 1

- No, the Bayes-optimal classifier is $f(\vec{x}) = \underset{y \in Y}{arg\,max} P(Y = y | X = \vec{x})$, which does not depends on loss function.
- No, it is likely to overfitting for small $k$ value. For example, assuming $k = 1$, the decision boundary can be highly non-linear.
- The data points have labels, such as discrete $y$ (classification) or continuous $y$ (regression).

## Problem 2

### Problem 2.1: Naive Bayes

#### 1.

The naive Bayes classifier is based on an assumption that all the features are independent from each other. Meanwhile, in the case of this problem, it is mentioned that the input variables $\vec{x}_i, i = 1, 2, ..., N$, which contains 5 features $(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(5)})$, has a spherical Guassian distribution. According to the defination of spherical Guassian, for all observation $\vec{x}_i$, $(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(5)})$should be independent from each other. Thus, the training data set fits the requirement of naive Bayes classifier.

Estimation formula is as follows,

$$\begin{aligned}
\hat{y}_{new} &= f(\vec{x}_{new}) \\
&= \underset{c_k}{argmax} P(Y \in c_k) \cdot P(X = \vec{x}_{new} | Y \in c_k) \\
&= \underset{c_k}{argmax} P(Y \in c_k) \cdot \prod_{i=1}^{5} P(X^{(i)} = \vec{x}_{new}^{(i)} | Y \in c_k) \\
&= \underset{c_k}{argmax} \frac{N_{c_k}}{N_{c_1} + N_{c_2} + N_{c_3}} \cdot \frac{1}{\sqrt{(2\pi)^5 \prod_{i=1}^{5} \sigma_i^2}} exp[-\frac{1}{2} \sum_{i=1}^{5} \frac{(\vec{x}_{new}^{(i)} - \vec{\mu}^{(i)})^2}{\sigma_i^2}]
\end{aligned}$$

**2.**

In the estimation function given above, $\vec{\mu}$ is the mean vector of all $\vec{x}$ that belong to different classes $c_k$,

$$\vec{\mu}_k = (\frac{\Sigma\vec{x}^{(1)}}{N_{c_k}}, ..., \frac{\Sigma\vec{x}^{(5)}}{N_{c_k}}),\ \vec{x} \in c_k$$

and $\sigma^2$ is the estimated variance of different classes and different features,

$$\sigma^{2}{}_{k}^{(i)} = \frac{1}{N_{c_k}} \sum (\vec{x}^{(i)} - \vec{\mu_k}^{(i)})^2,\ \vec{x} \in c_k$$

where $\vec{x}^{(i)}$ is the $i$-th feature of $\vec{x}$ that belongs to class $c_k$.

Prior $P(Y \in c_k)$ is estimated as

$$P(Y \in c_k) = \frac{N_{c_k}}{N_{c_1} + N_{c_2} + N_{c_3}}$$

**3.**

I think the classifier works well under the independence assumption, because the distribution model is well defined and parameters can be estimated easily.

However in some ways, the behavior depends on how the data set is distributed and how the new data is given.

For example, if three classes are highly overlapped, the estimation function can get three high probabilities in all classes, and choose the largest one. In this case, it is more likely to draw a wrong conclusion because the difference among three probabilities is small and misleading.

For another example, even three classes are well divided, if the new data is far away from most training data points, it can also generate three low but similar probabilities that gives little information.

## Problem 2.2: Perceptron

**1.**

The minimum empirical risk is $\frac{1}{22}$, for the boundary can classify all the solid circles to one class excpet one open circle ($\tilde{x}_2$), and other open circles into another class. Then there is only one misclassification, namely:

$$R = \frac{1}{n}\Sigma_{i=1}^{n} L(f(x_i), y_i) = \frac{1}{22}$$

**2.**

To classify, we need to compute the value of $\vec{v}_H \cdot \vec{x} + c$.

$$\begin{aligned}\vec{v}_H \cdot \vec{x}_1 + c &= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} -3 \\ 0 \end{pmatrix} + \frac{1}{2\sqrt{2}} \\ &= -\frac{5}{4}\sqrt{2} < 0\end{aligned}$$

$$\begin{aligned}\vec{v}_H \cdot \vec{x}_2 + c &= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} + \frac{1}{2\sqrt{2}} \\ &= \frac{\sqrt{2}}{4} > 0\end{aligned}$$

Thus, $\vec{x}_1$ is classifed to the class that is below the boundary (class $-1$), and $\vec{x}_2$ is classifed to the class above the boundary (class $+1$).

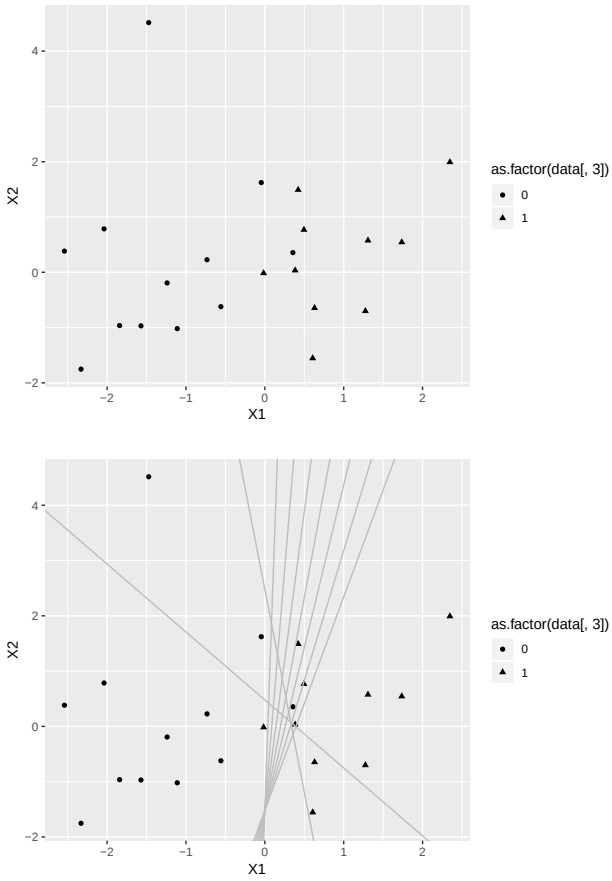
**3.**

The algorithm will not have a solution, the value of $\vec{v}_H$ will fluctuate in a small range. It is because every time it iterate through all the points, there will always be some point(s) that is misclassified, and the $\vec{v}_H$ changes every time.

$$\vec{v}_{H,new} = \vec{v}_H + \alpha L(\hat{y}_i, y_i) \cdot \vec{x}_i$$

Basically, it is because the data points are linearly non-separable. Under the condition that learning rate $\alpha$ is constant, the result is not convergent.

The result of perceptron algorithm with $\alpha = 1$ is as follow, and the boundary lines are shown as black lines after refreshing $\vec{v}_H$ for 10 times,

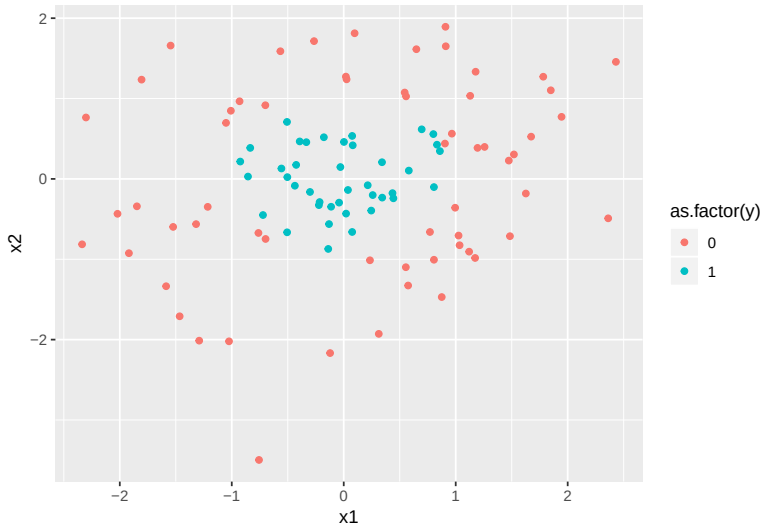



# Problem 3

When the decision boundary is not linear, the maximum margin classifier is not optimal.

For example,

```
x1 = rnorm(100, 0, 1)
x2 = rnorm(100, 0, 1)
y = ifelse(x1 ^ 2 + x2 ^ 2 < 1, 1, 0)
d = data.frame(x1, x2, y)
library(ggplot2)
ggplot(d, aes(x = x1, y = x2, color = as.factor(y))) + geom_point()
```




In this case, it is hard to define margin if the classifier use a linear function, although using *kernel function* can solve it.

# Problem 4

## Fig 1.

Would: Linear classifier with gradient descent learning

Why: It is because that the linear classifier can finally give a straignt line as the boundary. With the gradient descent and squared-error, we can learn the slope and intercept to minimize the loss.

Would not: Perceptron classifier

Why: It is because the perceptron cannot separate non-linearly separatable data. No matter what boundary it learnt, there would always be some points that are misclassified, which change the slope and intercept again.

## Fig 2.

Would: Naive Bayes

Why: The boundary is very smooth curve, so the boundary may be well defined by a function, which can be derived from the $P(class1) = P(class2)$ equation. Plus, the distribution model of data points have to be complex enough to draw such a complex boundary.

Would not: Linear classifier

Why: the boundary is not linear.

### Fig 3.

Would: K-nearest classifier

Why: the boundary is not smooth, but is connected by many line segments, which can be seen as the edges of K-nearest polygons of those data points.

Would not: Linear classifier

Why: the boundary is not linear.

## Problem 5

### 1.

The sample space is $\mathbb{R}^{k+m}$, whose first k elements are from $\vec{a}$ and last m elements are from $\vec{b}$. The labels are in two classes: 1 (match) and 0 (not match).

### 2.

The loss function's penalty is too large, making the classifier is highly overfitting the "match" data. In this case, only new data that fits the training data can get "match".

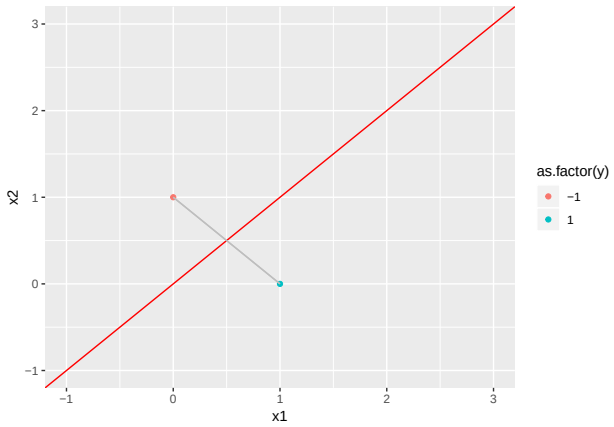
### 3.

Divide the training data into two parts, which are not overlapping with each other. Then use the first part, namely, training set to training the classifier as usual. After that, apply classifier on second part, namely, test set to culculate the loss.

Repeat the process several times for different training and test sets. Then choose the classifier that has the lowest loss on test set.

This process means to generalize the classifier to fits the new data. In the learning process, the test set can be seen as new data to classifier trained from training set. If the classifier is overfitting with training set, the loss on test will be large. Otherwise, the loss is low and the classifier can be well generalized to new data.

# Problem 6




# Problem 7

## 1.

$$< \vec{v_H}, \vec{x_1} > -c = (\frac{1}{\sqrt{2}} \times (-3) + \frac{1}{\sqrt{2}} \times 0) - \frac{1}{2\sqrt{2}}$$
$$= -\frac{7}{2\sqrt{2}} < 0$$

$$< \vec{v_H}, \vec{x_2} > -c = (\frac{1}{\sqrt{2}} \times \frac{1}{2} + \frac{1}{\sqrt{2}} \times \frac{1}{2}) - \frac{1}{2\sqrt{2}}$$
$$= \frac{1}{2\sqrt{2}} > 0$$

So $\vec{x_1}$ is classified to class $-1$, and $\vec{x_2}$ is classified to class $+1$.

## 2.

No, SVM is the improvement of linear classifier. It maximize the margin which makes sure that all the data points are away from the boundary as far as possible. In this case, SVM can better fits the new data than simple linear classifier.

So, same as linear classifier, SVM also calculates the $< \vec{v_H}, \vec{x} > -c$ and use the sign as the class label.

## 3.

Sigmoid function is the approximate of $0-1$ loss function. We approximate it because the sigmoid function is continuous, and we can calculate its derivative and use the gradient descent learning.