EECS 545 Homework3

Hui Cai

February 26, 2018

1.

(a)

The error rate on test data is 0.125%

(b)

Follow the notation in lecture slides. Let's consider for some new data $x = (x_1, x_2, ..., x_d)$, we would classify it based on

$$\begin{aligned} y &= \text{argmax} P(y = c|X = x) \\ &= \text{argmax} P(X = x|y = c)P(y = c) \\ &= \text{argmax} P(y = c) \prod_{d=1}^{D} P(X_d = x_d|y = c) \\ &= \text{argmax}\ \pi_c \prod_{d=1}^{D} \theta_{cdx_d} \\ &= \text{argmax}\ \pi_c \prod_{d=1}^{D} \prod_{m=0}^{M} \theta_{cdm} 1_{\{x_d=m\}} \end{aligned}$$

where $\theta_{cdx_d} = P(X_d = m|c)$, $m \in \{0, 1, ..., M\}$. Since it's a two-class problem with binary features. So $c \in \{0, 1\}$ and $m \in \{0, 1\}$. Then $y = 1$ if

$$\begin{aligned} & P(y = 1|X = x) - P(y = 0|X = x) \geq 0 \\ \Rightarrow & \log(P(y = 1|X = x)) - \log(P(y = 0|X = x)) \geq 0 \\ \Rightarrow & \log(\pi_1) + \sum_{d=1}^{D} \log(\theta_{1d1}) 1_{\{x_d=1\}} + \sum_{d=1}^{D} \log(1 - \theta_{1d1}) 1_{\{x_d=0\}} - \log(\pi_0) - \sum_{d=1}^{D} \log(\theta_{0d1}) 1_{\{x_d=1\}} - \sum_{d=1}^{D} \log(1 - \theta_{0d1}) 1_{\{x_d=0\}} \geq 0 \\ \Rightarrow & \log(\pi_1) + \sum_{d=1}^{D} \log(\theta_{1d1}) x_d + \sum_{d=1}^{D} \log(1 - \theta_{1d1})(1 - x_d) - \log(\pi_0) - \sum_{d=1}^{D} \log(\theta_{0d1}) x_d - \sum_{d=1}^{D} \log(1 - \theta_{0d1})(1 - x_d) \geq 0 \\ \Rightarrow & \log(\pi_1) + \sum_{d=1}^{D} \log(\frac{\theta_{1d1}}{1 - \theta_{1d1}}) x_d + \sum_{d=1}^{D} \log(1 - \theta_{1d1}) - \log(\pi_0) - \sum_{d=1}^{D} \log(\frac{\theta_{0d1}}{1 - \theta_{0d1}}) x_d - \sum_{d=1}^{D} \log(1 - \theta_{0d1}) \geq 0 \end{aligned}$$

We define $\omega = \omega_1 - \omega_0$, where $\omega_c = (\log \frac{\theta_{c11}}{1-\theta_{c11}}, \log \frac{\theta_{c21}}{1-\theta_{c21}}, ..., \log \frac{\theta_{cD1}}{1-\theta_{cD1}}, \log \pi_c + \sum_{d=1}^{D} \log(1 - \theta_{cd1}))^T$.
Then the above fomula is equivalent to $\omega^T \phi(x) \geq 0$, where $\phi(x) = [x_1, x_2, ..., x_D, 1]^T$.
So, a Naive Bayes binary classifier is a linear classifier.

2.

(a)

We could prove a kernel is valid by finding an implicit feature mapping $\phi(x)$ such that $\kappa(x, x') = \phi(x)^T \phi(x')$
Let κ_1, κ_2: positive-definite functions defined on $R^s \times R^s$, then they can be written as $\phi(x)^T \phi(x')$,

where $\phi(x) = [\phi_1(x), ..., \phi_s(x)]^T$

i.

$$\begin{aligned}\kappa(x, x') &= a\kappa_1(x, x') \\ &= a\phi(x)^T\phi(x') \\ &= \sqrt{a}\phi(x)^T\sqrt{a}\phi(x') \\ &= \phi'(x)^T\phi'(x')\end{aligned}$$

where $\phi'(x) := \sqrt{a}\phi(x)$.

ii.

$$\begin{aligned}\kappa(x, x') &= \kappa_1(x, x') + \kappa_2(x, x') \\ &= \phi^1(x)^T\phi^1(x') + \phi^2(x)^T\phi^2(x') \\ &= \phi'(x)^T\phi'(x')\end{aligned}$$

where $\phi'(x) := [\phi^1(x), \phi^2(x)]^T$.

iii.

$$\begin{aligned}\kappa(x, x') &= \kappa_1(x, x')\kappa_2(x, x') \\ &= \sum_{i=1}^{s} \phi_i^1(x)^T\phi_i^1(x') \sum_{i=1}^{s} \phi_i^2(x)^T\phi_i^2(x') \\ &= \sum_{i,j} \phi_i^1(x)^T\phi_i^1(x')\phi_j^2(x)^T\phi_j^2(x') \\ &= \phi'(x)^T\phi'(x')\end{aligned}$$

where $\phi'(x) := [\phi_1^1(x)\phi_1^2(x), ..., \phi_s^1(x)\phi_s^2(x)]^T$, it's a $s^2 \times 1$ vector.

iv. It's obvious that $\kappa(x, x') = f(x)f(x')$ is valid since it follows that $\phi(x) = [f(x)]$.

v.
Follows property iii, $\kappa(x, x') = \kappa_1(x, x')\kappa_2(x, x')$ is valid.
Then, $\kappa(x, x') = \kappa_1(x, x')^d = \kappa_1(x, x')\kappa_1(x, x')\kappa_1(x, x')^{d-2}$ =valid kernel$\times\kappa_1(x, x')^{d-2}$ = ... =valid kernel$\times$valid kernel
So, $\kappa(x, x')$ is valid kernel.

vi.
Suppose $p(x) = a_n x^n + ... + a_1 x^1 + a_0$. Then, $\kappa(x, x') = p(\kappa(x, x')) = a_n\kappa(x, x')^n + ... + a_1\kappa(x, x')^1 + a_0$,
with property i, ii, v, it's obvious that it's a valid kernel.

(b)

Expand the Gaussian Kernel:

$$\begin{aligned}\kappa(x, x') &= \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)\exp\left(-\frac{x^Tx'}{2\sigma^2}\right)\exp\left(-\frac{\|x'\|^2}{2\sigma^2}\right)\end{aligned}$$

and by Taylor's theorem,

$$\exp\left(-\frac{x^Tx'}{2\sigma^2}\right) = \sum_{n=0}^{\infty} \frac{1}{n!}\left(\frac{x^Tx'}{\sigma^2}\right)^n$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!\sigma^{2n}} (x^T x')^n$$

We know that $(x^T x')^n$ is a valid kernel. So, let $\psi_n(x)$ denote the feature mapping of $(x^T x')^n$, then

$$\begin{aligned}
\exp(-\frac{x^T x'}{2\sigma^2}) &= \sum_{n=0}^{\infty} \frac{1}{n!\sigma^{2n}} \psi_n(x)^T \psi_n(x) \\
&= (\sqrt{\frac{1}{0!}}\psi_0(x), ..., \sqrt{\frac{1}{n!\sigma^{2n}}}\psi_n(x))^T (\sqrt{\frac{1}{0!}}\psi_0(x'), ..., \sqrt{\frac{1}{n!\sigma^{2n}}}\psi_n(x')) \\
&= \eta(x)^T \eta(x')
\end{aligned}$$

of which $\eta(x)$ is a new feature mapping with infinite dimension. The second line of the above formula is a valid kernel because of the propery we showed above. So,

$$\begin{aligned}
\kappa(x, x') &= \exp(-\frac{\|x\|^2}{2\sigma^2}) \eta(x)^T \eta(x') \exp(-\frac{\|x'\|^2}{2\sigma^2}) \\
&= \phi(x)^T \phi(x')
\end{aligned}$$

of which $\phi(x) = \exp(-\frac{\|x'\|^2}{2\sigma^2}) \eta(x)$ is the infinite dimension feature mapping of $\kappa(x, x')$.

3.

(a)

In Kernel Percepton, ω is usually defined as $\omega = \sum_i \alpha_i y_i \phi(x_i)$, where α_i's are coefficients, $\phi(\cdot)$ is the implicit feature map corresponding to the chosen kernel, and training examples $x_1, x_2, .., x_n$, their labels $y_1, y_2, .., y_n$ where $y_i \in \{-1, 1\}$.
For certain data point x', our predictor is

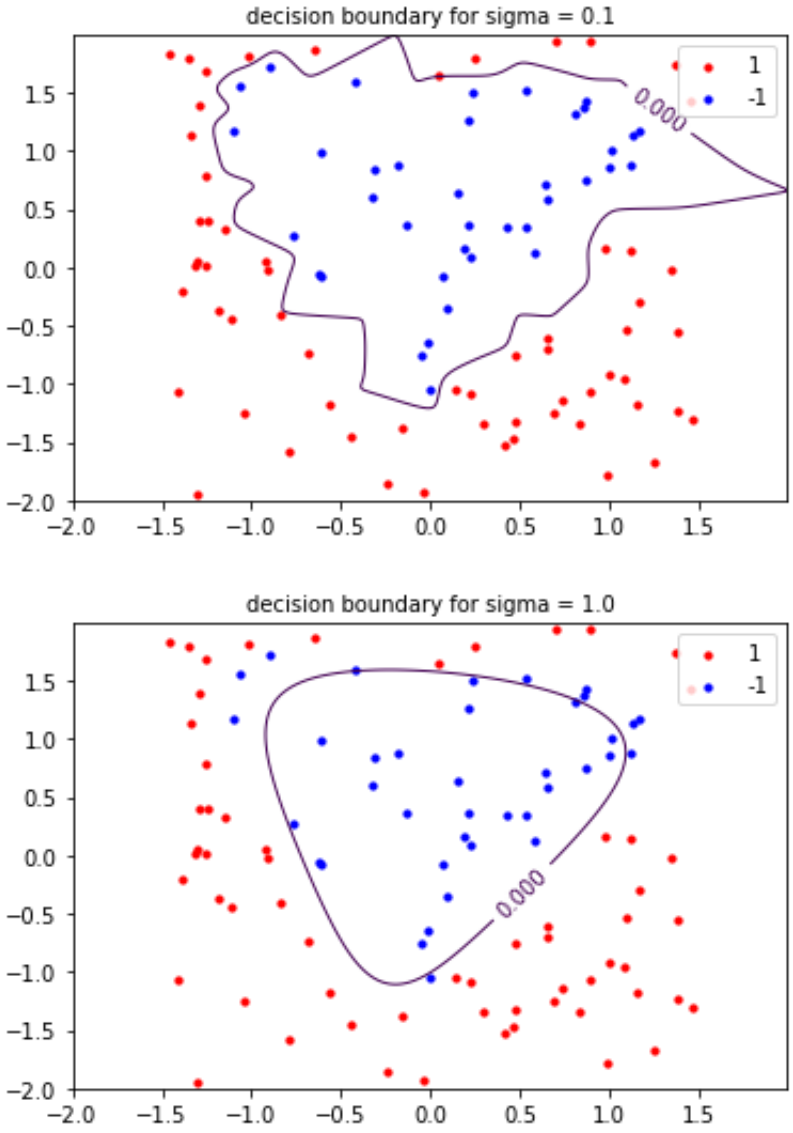
$$\begin{aligned}
y(x) &= \text{sgn}(\omega^T \phi(x')) \\
&= \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \phi(x')) \\
&= \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i \kappa(x_i, x'))
\end{aligned}$$

Here, what we need to train is α, a $n \times 1$ vector.
Given the data stated above, here is the algorithm:
Initialize: Set $\alpha = [0, 0, ..., 0]^T$
For j = 1,...,n
Observe x_j, predict $y = \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i \kappa(x_i, x_j))$
Receive $y_j \in \{-1, 1\}$, update:

$$\alpha = \begin{cases} \alpha & \text{if} \quad y_j \sum_{i=1}^{n} \alpha_i y_i \kappa(x_i, x_j) > 0 \\ \alpha + y_j (y \circ \kappa(x, x_j)) & \text{otherwise} \end{cases}$$

where y, x denotes the whole training data, and $\circ$ denotes the Hadamard product. So $y \circ \kappa(x, x_j)$ is a vector.
End
The above algorithm can loop for several times. And to some sense like SGD, the term $y_j(y \circ \kappa(x, x_j))$ is derived from derivative against α with certain loss function.

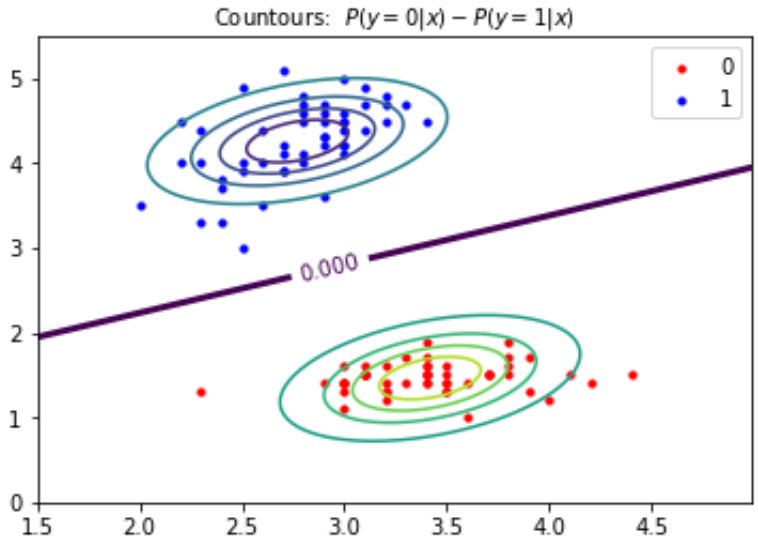
(b)



We can see that when $\sigma = 0.1$, the decision boundary managed to separate all points correctly, but there's high variance. When $\sigma = 1.0$, the variance will be reduced but there will be more bias, so some of the points are not correctly classified.

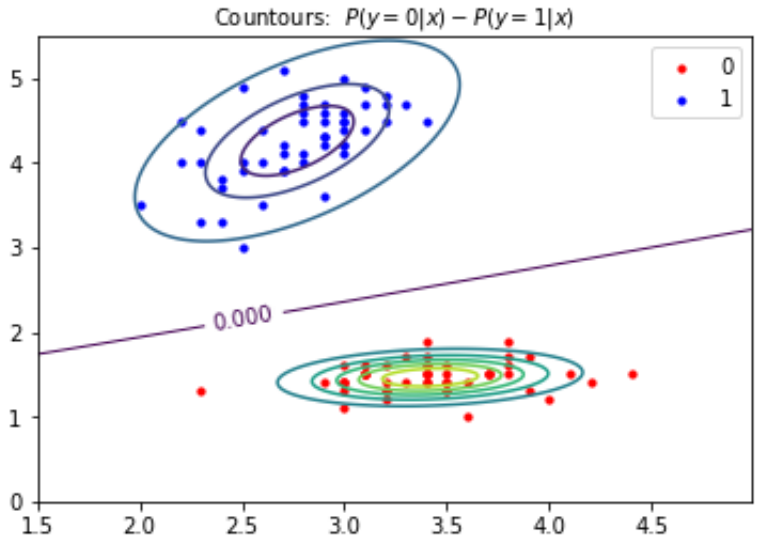
4.

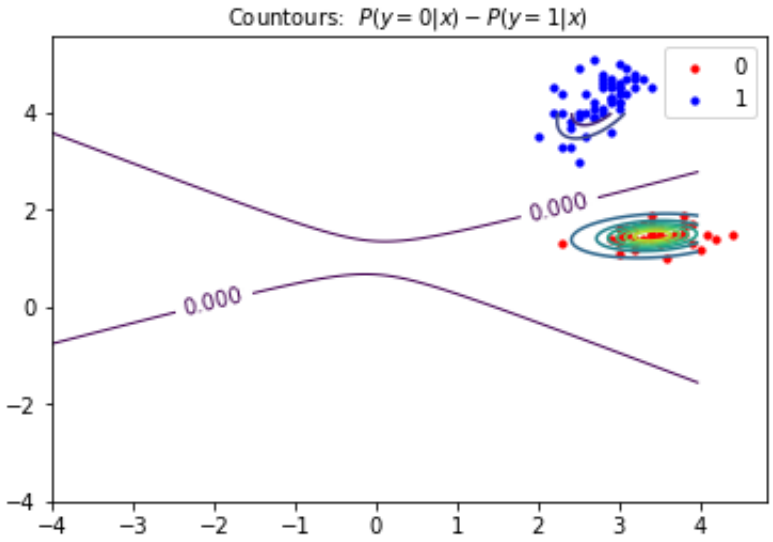
(a)



Since the two set has the same covariance matrix, the boundary would be linear.

(b)





We can see that the decision boundary found by QDA is quadratic. It seems linear in the first plot of (b), so I slight change the axis to have a full view. Besides, it also moves closer to the red points because of the smaller variance of red points.