به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

شناسائی الگو

تمرین شماره 3

سجاد پاکدامن ساوجی

۸۱۰۱۹۵۵۱۷

فهرست

چکیده

در این تمرین چندین طبقه بند برای حل مسئله داده شده طراحی شد و در پایان این طبقه بند ها با معیار ها مختلفی با یکدیگر مقایسه شدند. در تمرین اول تئوری مربوط به ساخت طبقه بند های naive bayes بررسی شد و در انتها به صورت دستی یک مسئله طبقه بندی با روش naive bayes حل شد. در سوال دوم با مبحث kernel estimation بیش‌تر آشنا شدیم و تاثیر مشابه فیلترینگ را بر روی توزیع احتمال توسط کرنل دیدیم. در سوال ۳ حد بالای واریانس بک تخمین kernel estimation را اثبات کردیم. در سوال ۴ تلاش به پیاده سازی یک bayes optimal classifier کردیم و طی این فرایند با روش های متفاوت رفع مشکل وارون پذیری ماتریس covariance آشنای شدیم. در سوال ۵ طبقه بند سوال پیشین را در بسطر کمینه سازی ریسک بررسی کردیم. در سوال ۶ برای تخمین توزیع های احتمال از تخمین گرهای non-parametric استفاده کردیم. در این سوال هم با پنجره مستطیلی و هم با پنجره گوسی تقریب را انجام دادیم و گذشته از آن تاثیر اسکیل پنجره را نیز بررسی کردیم. در سوال ۷ به جمع بندی و مقایسه طبقه بند ها پرداخته شد.

سوال 1

Pattern Recognition Assignment 3

Problem 1)

1.1 $p(\underline{x}|\omega_i) = p([x_1, \cdots, x_n]^T|\omega_i) \xrightarrow{\text{chain rule}} p(x_1|\omega_i)p(x_2|x_1,\omega_i)\cdots p(x_n|x_{n-1}\cdots x_1,\omega_i)$

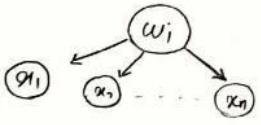
$\xrightarrow[\text{feature vector}]{\text{dimentions of}} p(x_1|\omega_i)p(x_2|\omega_i)\cdots p(x_n|\omega_i) = \prod_{j=1}^{n} p(x_j|\omega_i)$

$$i^* = \underset{i}{\text{argmax}} \prod_{j=1}^{n} p(x_j|\omega_i) \times p(\omega_i) \xrightarrow{Ln} \underset{i}{\text{argmax}} \sum_{j=1}^{n} Ln^{p(x_j|\omega_i)} + Ln^{p(\omega_i)}$$

$$\Rightarrow i^* = \underset{i}{\text{argmax}} \sum_{j=1}^{n} Ln^{p(x_j|\omega_i)} + Ln^{p(\omega_i)}$$

مزیت آن برای تولید این احتمال کردیم در حقیقت یک نوع conditional independ است

اگر bays net مربوط به این مزیت را رسم کنیم به صورت زیر خواهد بود.



1.2 exponential family

- normal
- exponential
- gamma
- bernolli
- beta
- poisson
- chi-squared
- Dirichlet
- categorical
- wishant
- inverse wishart

A distribution from exponential family can generally be represented as:

1. $f_X(x|\theta) = h(x)\exp\{\eta(\theta)\cdot T(x) - A(\theta)\}$
2. $f_X(x|\theta) = h(x)g(\theta)\exp\{\eta(\theta)\cdot T(x)\}$
3. $f_X(x|\theta) = \exp\{\eta(\theta).T(x) - A(\theta) + B(x)\}$

form 1. $i^* = \arg\max_i \sum_{j=1}^{n} \ln f(x_j|\omega_i) + \ln p(\omega_i)$ 2 class problem

$$\sum_{j=1}^{n} \ln f(x_j|\omega_1) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \ln f(x_j|\omega_2) + \ln p(\omega_2)$$

$$\sum_{j=1}^{n} \left(\ln h(x_j) + \eta(\theta_1)\cdot T(x_j) - A(\theta_1) \right) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \left(\ln h(x_j) + \eta(\theta_2)\cdot T(x_j) - A(\theta_2) \right) + \ln p(\omega_2)$$

$$\sum_{j=1}^{n} \ln h(x_j) + \eta(\theta_1) \sum_{j=1}^{n} T(x_j) - nA(\theta_1) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \ln h(x_j) + \eta(\theta_2) \sum_{j=1}^{n} T(x_j) - nA(\theta_2) + \ln p(\omega_2)$$

$$\eta(\theta_1) \sum_{j=1}^{n} T(x_j) - nA(\theta_1) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \eta(\theta_2) \sum_{j=1}^{n} T(x_j) - nA(\theta_2) + \ln p(\omega_2)$$

$\Rightarrow$ decision surface :

$$[\eta(\theta_1) - \eta(\theta_2)] \sum_{j=1}^{n} T(x_j) - n[A(\theta_1) - A(\theta_2)] + \ln \frac{p(\omega_1)}{p(\omega_2)} = 0$$

form 2.

$$i^* = \arg\max_i \sum_{j=1}^{n} \ln f(x_j|\omega_i) + \ln p(\omega_i) \quad \text{2 class problem}$$

$$\sum_{j=1}^{n} \ln f(x_j|\omega_1) + \ln p(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \sum_{j=1}^{n} \ln f(x_j|\omega_2) + \ln p(\omega_2)$$

$$\sum_{j=1}^{n} \ln h(x_j) + \ln g(\theta_1) + \eta(\theta_1)\cdot T(x_j) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \ln h(x_j) + \ln g(\theta_2) + \eta(\theta_2)\cdot T(x_j) + \ln p(\omega_2)$$

$$n \ln g(\theta_1) + \eta(\theta_1) \sum_{j=1}^{n} T(x_j) + \ln p(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} n \ln g(\theta_2) + \eta(\theta_2) \sum_{j=1}^{n} T(x_j) + \ln p(\omega_2)$$

$\Rightarrow$ decision surface :

$$n \ln \frac{g(\theta_1)}{g(\theta_2)} + [\eta(\theta_1) - \eta(\theta_2)] \sum_{j=1}^{n} T(x_j) + \ln \frac{p(\omega_1)}{p(\omega_2)} = 0$$

form 3

$$i^* = \underset{i}{\operatorname{argmax}} \sum_{j=1}^{n} \ln f(x_j|\omega_i) + \ln P(\omega_i) \qquad \text{2-class problem}$$

$$\sum_{j=1}^{n} \ln f(x_j|\omega_1) + \ln P(\omega_1) \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \ln f(x_j|\omega_2) + \ln P(\omega_2)$$

$$\sum_{j=1}^{n} \eta(\theta_1)\cdot T(x_j) - A(\theta_1) + B(x_j) + \ln \frac{P(\omega_1)}{P(\omega_2)} \underset{R_2}{\overset{R_1}{\gtrless}} \sum_{j=1}^{n} \eta(\theta_2)\cdot T(x_j) - A(\theta_2) + B(x_j)$$

$$\eta(\theta_1)\sum_{j=1}^{n} T(x_j) - n A(\theta_1) + \sum_{j=1}^{n} B(x_j) + \ln \frac{P(\omega_1)}{P(\omega_2)} \underset{R_2}{\overset{R_1}{\gtrless}} \eta(\theta_2)\sum_{j=1}^{n} T(x_j) - n A(\theta_2) + \sum_{j=1}^{n} B(x_j)$$

$$\eta(\theta_1)\sum_{j=1}^{n} T(x_j) - n A(\theta_1) + \ln \frac{P(\omega_1)}{P(\omega_2)} \underset{R_2}{\overset{R_1}{\gtrless}} \eta(\theta_2)\sum_{j=1}^{n} T(x_j) - n A(\theta_2) + \ln \frac{P(\omega_2)}{P(\omega_2)}$$

$$[\eta(\theta_1) - \eta(\theta_2)]\sum_{j=1}^{n} T(x_j) - n[A(\theta_1) - A(\theta_2)] + \ln \frac{P(\omega_1)}{P(\omega_2)} = 0 \quad \Longleftarrow \text{ decision surface}$$

1.3

$P(\text{color} \mid \text{poisonous} = \text{YES}) = \{ \text{orange}: \frac{1}{2}, \text{green}: \frac{1}{2} \}$

$P(\text{toughness} \mid \text{poisonous} = \text{yes}) = \{ \text{Hard}: \frac{3}{4}, \text{Soft}: \frac{1}{4} \}$

$P(\text{fungus} \mid \text{poisonous} = \text{YES}) = \{ \text{YES}: \frac{1}{2}, \text{No}: \frac{1}{2} \}$

$P(\text{appearance} \mid \text{poisonous} = \text{YES}) = \{ \text{Smooth}: \frac{1}{4}, \text{wrinkled}: \frac{3}{4} \}$

$P(\text{color} \mid \text{poisonous} = \text{No}) = \{ \text{green}: \frac{2}{3}, \text{Brown}: \frac{1}{3} \}$

$P(\text{toughness} \mid \text{poisonous} = \text{No}) = \{ \text{Hard}: \frac{2}{3}, \text{Soft}: \frac{1}{3} \}$

$P(\text{fungus} \mid \text{poisonous} = \text{No}) = \{ \text{YES}: \frac{1}{3}, \text{No} = \frac{2}{3} \}$

$P(\text{appearance} \mid \text{poisonous} = \text{No}) = \{ \text{Smooth}: \frac{2}{3}, \text{wrinkled}: \frac{1}{3} \}$

$P(\text{YES}) = \frac{4}{7}, \quad P(\text{No}) = \frac{3}{7}$

$i^* = \underset{i}{\operatorname{argmax}} \prod_{j=1}^{n} p(x_j|\omega_i) \times p(\omega_i)$

$\rightarrow p(\text{color}|\text{yes})\,p(\text{toughness}|\text{yes})\,p(\text{fungus}|\text{yes})\,p(\text{appearance}|\text{yes}) \times p(\text{yes})$

$\rightarrow p(\text{green}|\text{yes})\,p(\text{soft}|\text{yes})\,p(\text{yes}|\text{yes})\,p(\text{wrinkled}|\text{yes}) \times p(\text{yes}) =$

$= \frac{1}{2} \times \frac{1}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{4}{7} = 0.0267$

$\rightarrow p(\text{green}|\text{No})\,p(\text{soft}|\text{No})\,p(\text{YES}|\text{No})\,p(\text{wrinkled}|\text{No})\,p(\text{No}) =$

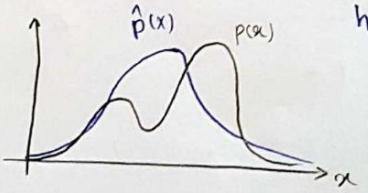
$= \frac{2}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{7} = 0.0105$

$\Rightarrow p(\text{YES}|\chi) > p(\text{No}|\chi) \Rightarrow$ **poisonous = yes**

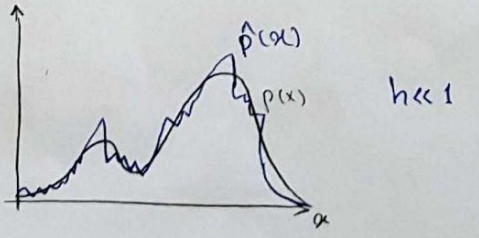
سوال 2

Problem 2

1. as h increases the form of $\hat{p}$ whould be mak smooth.
thus in larg amounts of h $\hat{p}$ whould contain only one maxima and wolld also be extreamly Smooth.



but for Smaller amounts of h the kernel will perfor more locally and whould be able to observe/represent more relative maximas.



قسمت ۲)

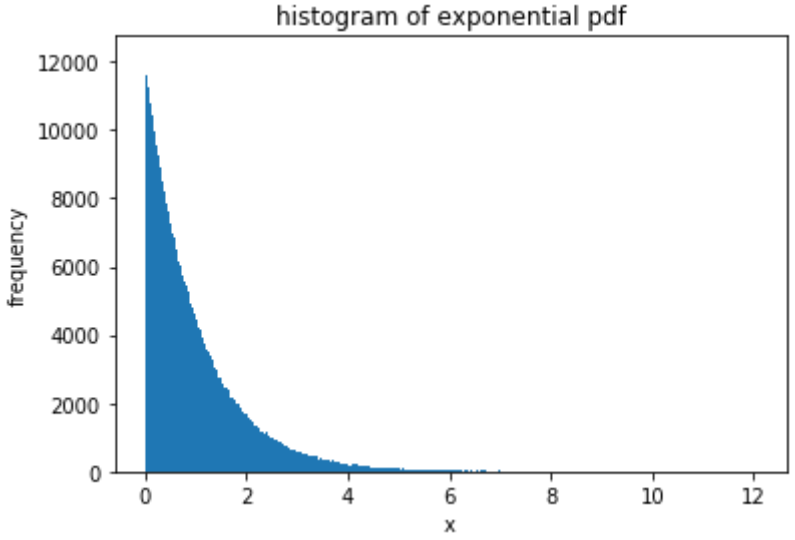
اگر بخواهیم که optimal ترین h یا همان پهنای باند را برای کرنل پیدا کنیم همان طور که در سوال مطرح شده است باید به صورت exhaustive search تمامی مقادیر مختلف را برای h امتحان کنیم و با معیاری همچون MSE (یا مورد مشابه آن) کیفیت تخمین را بررسی کنیم. ولی در حالتی که واقعا مسئله را حل می‌کنیم دسترسی به توزیع احتمال نداریم که آن را برای پیدا کردن پهنای باند بهینه استفاده کنیم از این جهت رابطه بیان شده در حالت عملی کاربرد خیلی خوبی دارد البته ممکن است که برای هر کرنل یا هر مسئله ای خوب عمل نکند با این حال به عنوان a rule of thumb مناسب به نظر می‌رسد.

$$h^* = 1.06\sigma N^{-1/5}$$

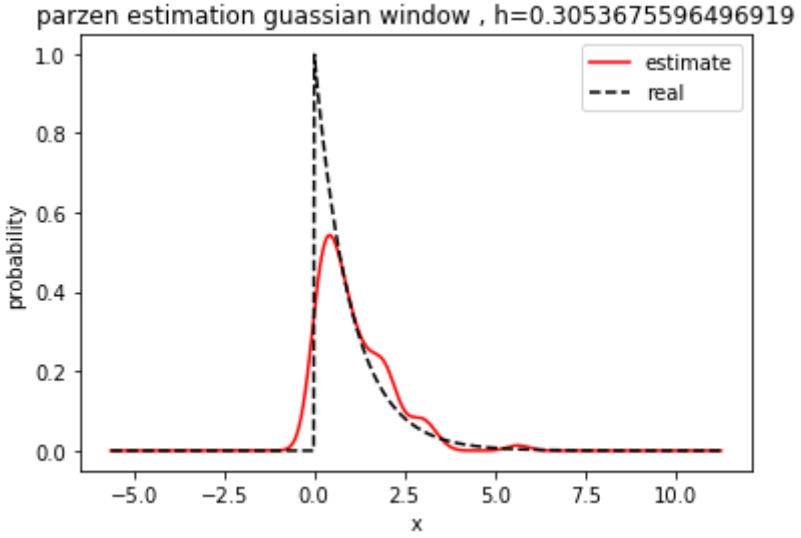
در رابطه ی بالا هرچه تعداد sample هایی که وجود دارد بیشتر باشد مقدار h کمتر خواهد بود که منطقی است زیرا با تعداد بیشتر داده پهنای پنجره لازم نیست که خیلی بزرگ باشد. و همچنین هر چه واریانس داده ها بیشتر باشد

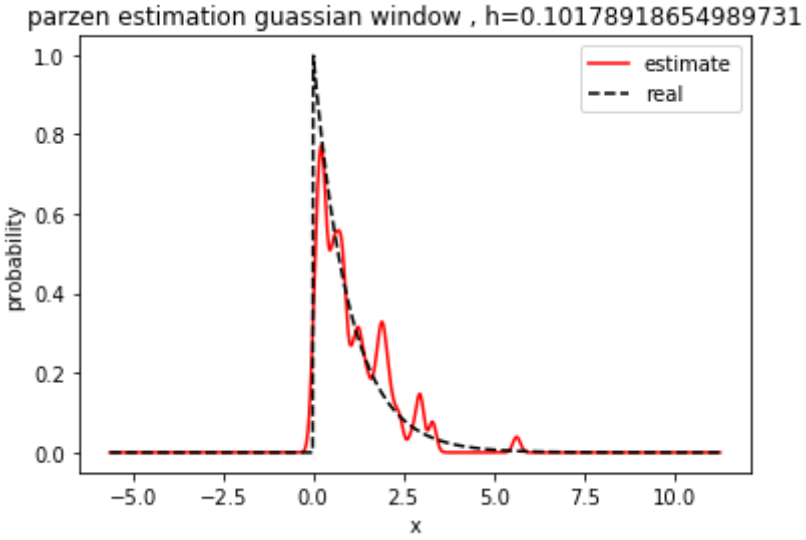
در این صورت مقدار h زیاد تر خواهد بود که باز منطقی است زیرا هرچه پراکندگی داده ها بیشتر باشد نیاز است که تعداد داده های زیادی در پنجره قرار گیرند تا تخمین مناسبی بدست بیاید.

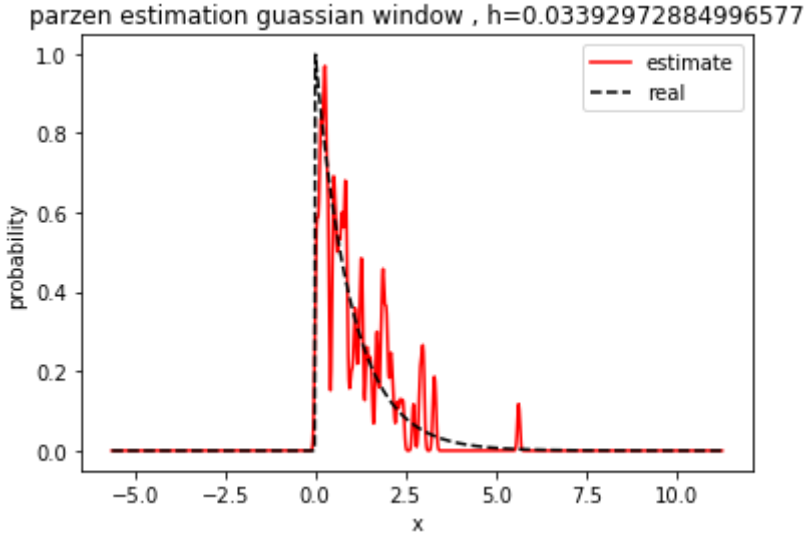
برای بررسی نتایج بدست آمده از این رابطه سعی شد که تخمین یک توزیع نمایی را انجام دهیم در شکل های پایین نتایج را مشاهده می‌کنید



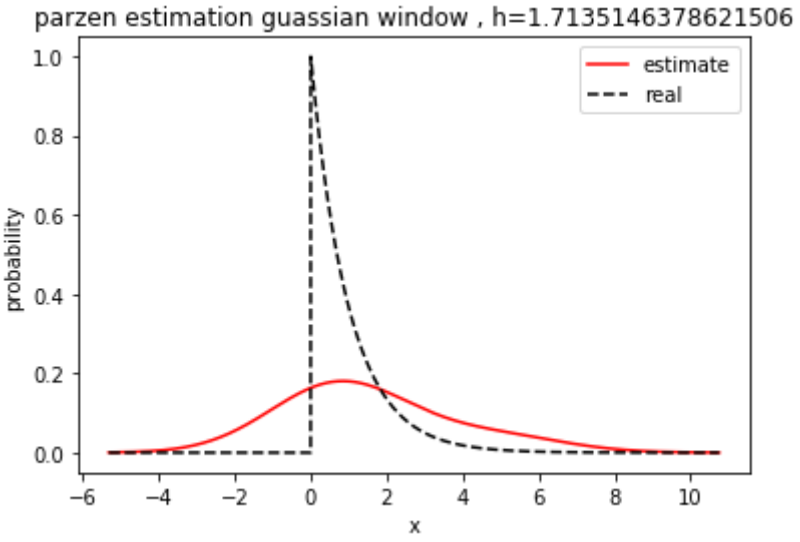
تعداد داده ها ۱۰۰)

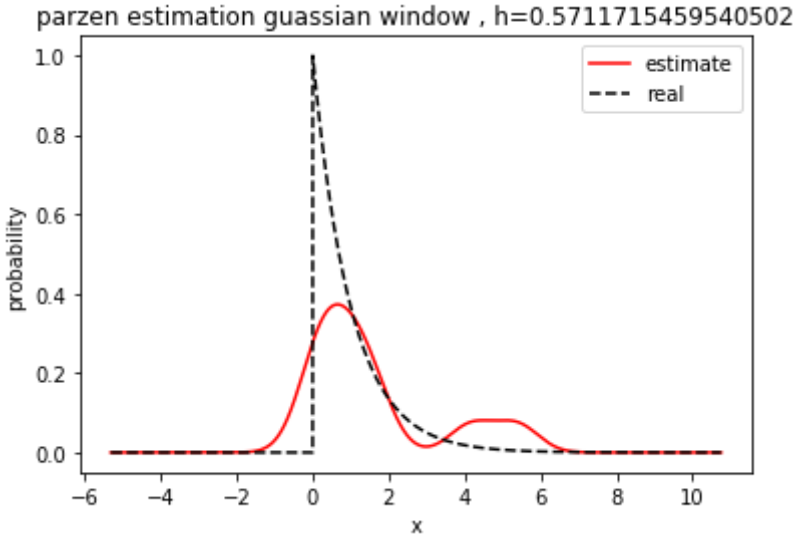


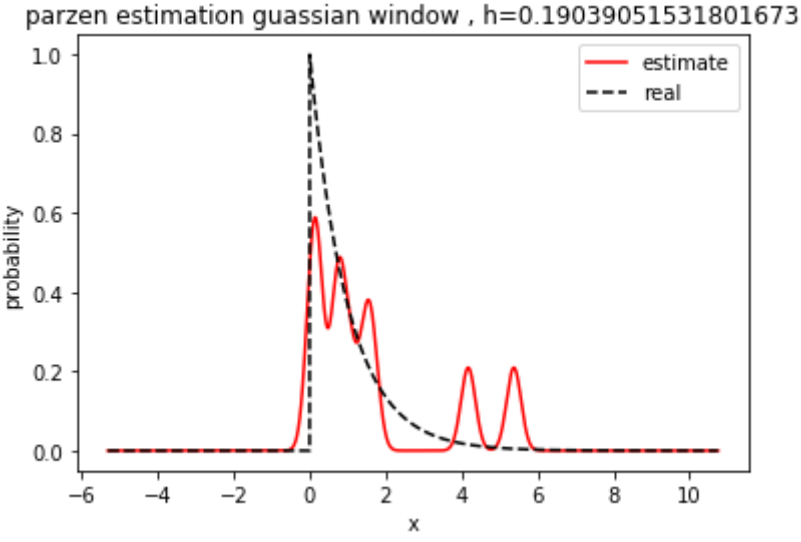




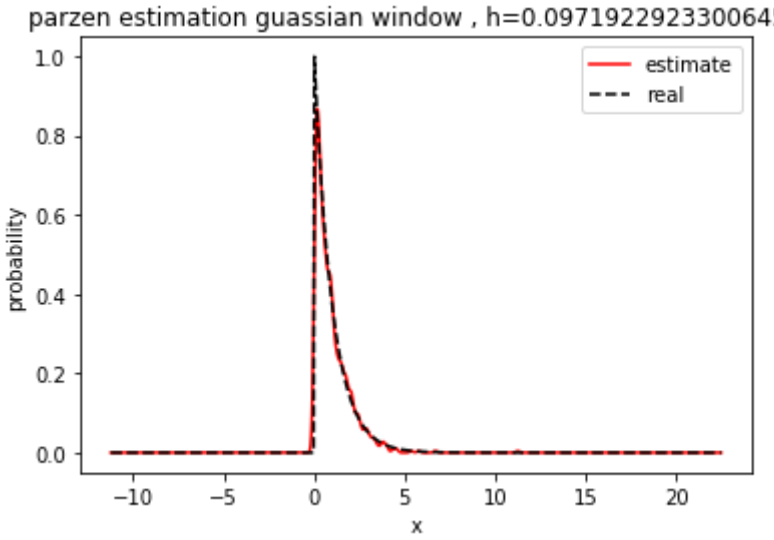
تعداد داده ها ۱۰)

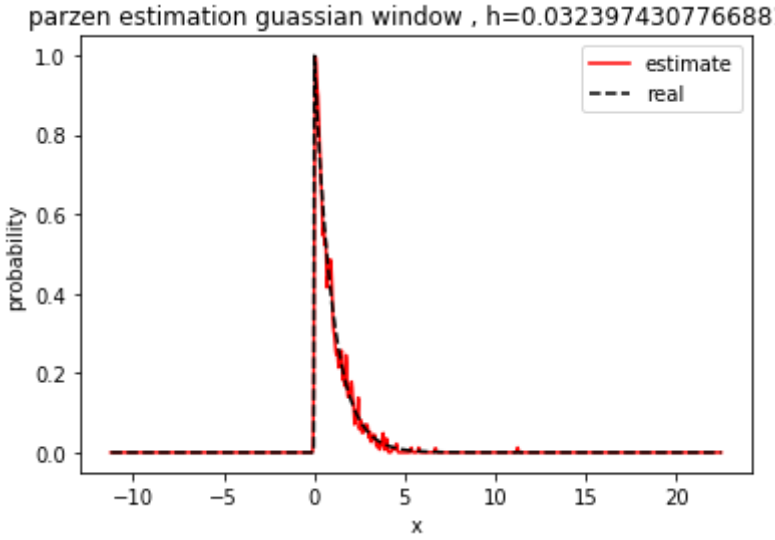


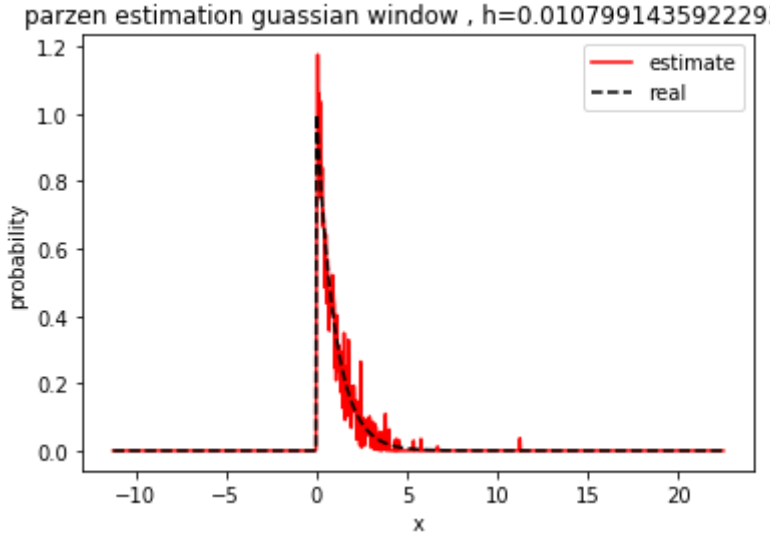




تعداد داده ها ۱۰۰۰)


parzen estimation guassian window , h=0.09719229233006457
estimate
real
probability
x
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
15
20


parzen estimation guassian window , h=0.03239743077668819
estimate
real
probability
x
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
15
20


parzen estimation guassian window , h=0.010799143592229396
estimate
real
probability
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
15
20

قسمت ۳)

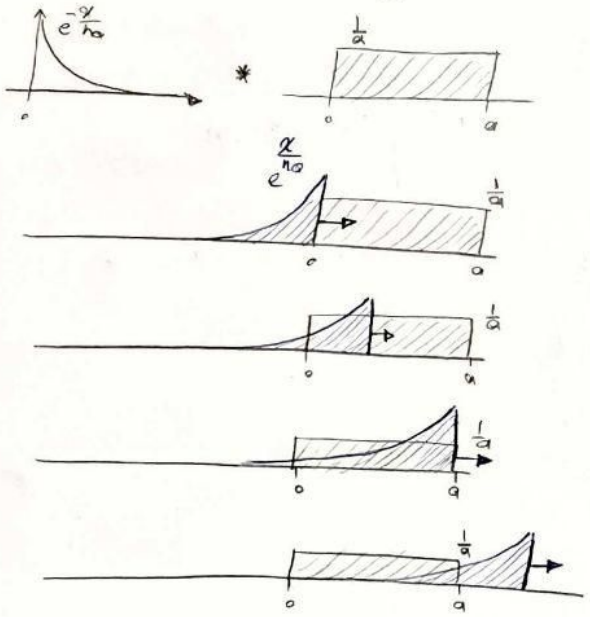
Problem 2)

III. $\hat{p}(x_0) = \frac{k_Q}{QV_Q} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{h_Q}\varphi\left(\frac{x_0 - x_q}{h_Q}\right)$

$$\mathcal{E}\{\hat{p}(x_0)\} = \frac{1}{Q}\sum_{q=1}^{Q}\mathcal{E}\left\{\frac{1}{h_Q}\varphi\left(\frac{x_0 - x_q}{h_Q}\right)\right\} = \frac{1}{Q}\times Q\,\mathcal{E}\left\{\frac{1}{h_Q}\varphi(x_0 - x')\right\}$$

$$= \mathcal{E}\left\{\frac{1}{h_Q}\varphi\left(\frac{x_0 - x'}{h_Q}\right)\right\} = \int\frac{1}{h_Q}\varphi(x_0 - x')f(x')\,dx' = \varphi\left(\frac{x_0}{h_Q}\right) * f(x_0)\times\frac{1}{h_Q}$$

$$\varphi\left(\frac{x}{h_Q}\right) = e^{-\frac{x}{h_Q}}$$



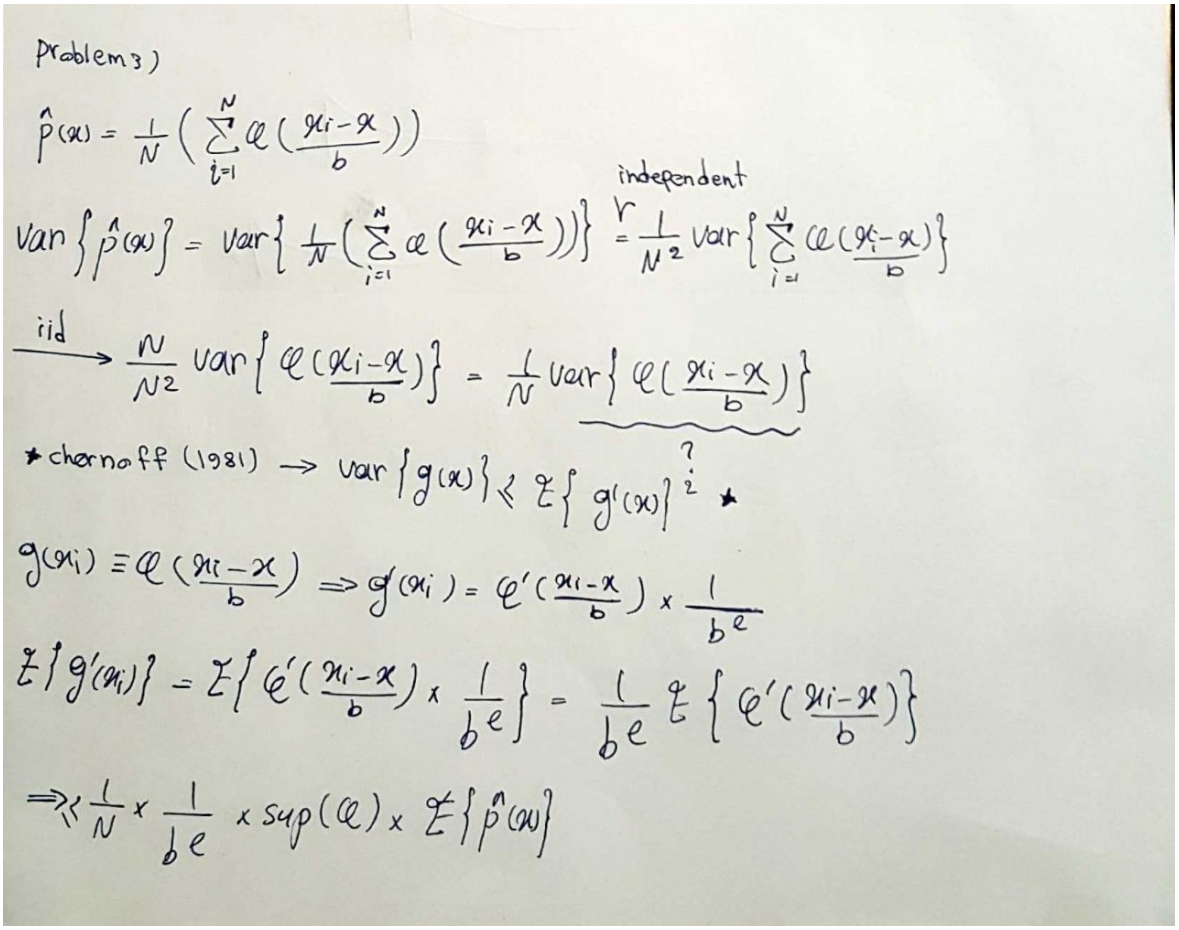
$x < 0 \Rightarrow 0$

$0 < x < a \Rightarrow \int_0^x \frac{1}{a}e^{-\frac{x'}{h_Q}}dx' \times\frac{1}{h_Q} = \frac{1}{a}\times\int_0^x \frac{1}{h_Q}e^{-\frac{x'}{h_Q}} = \frac{-1}{a}e^{-\frac{x'}{h_Q}}\Big]_0^x = \frac{1}{a}\left(1 - e^{-\frac{x}{h_Q}}\right)$

$x > a \Rightarrow \int_{x-a}^{x}\frac{1}{a}\times\frac{1}{h_Q}e^{\frac{-x'}{h_Q}}dx' = \frac{-1}{a}e^{-\frac{x'}{h_Q}}\Big]_{x-a}^{x} = \frac{1}{a}\left(e^{\frac{a}{h_Q}} - 1\right)e^{-\frac{x}{h_Q}}$

$$\bar{P}_n(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{a}\left(1 - e^{-\frac{x}{h_Q}}\right) & 0 < x < a \\ \frac{1}{a}\left(e^{\frac{a}{h_Q}} - 1\right)e^{-\frac{x}{h_Q}} & x > a \end{cases}$$

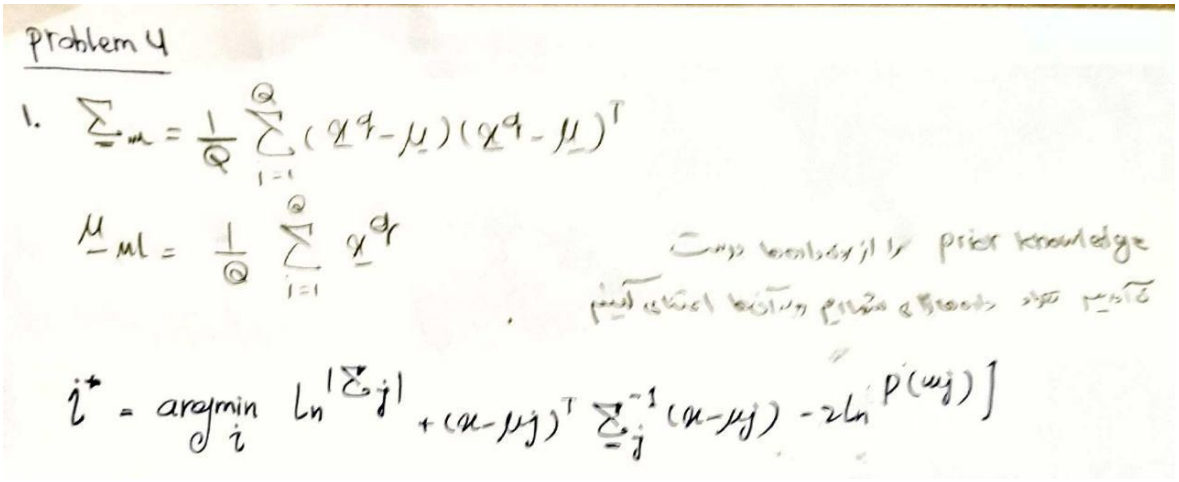
سوال 3



سوال4

قسمت اول)

طبقه بند خواسته شده طراحی شد و برای مشاهده جزئیات طراحی آن به فایل .py که به پیوست گزارش وجود دارد رجوع کنید. روابط برای تخمین بردار میانگین و ماتریس کواریانس نیز در عکس زیر آمده است. برای محاسبه prior knowledge از فرکانس تکرار هر داده در دیتا ست train استفاده شده است.



قسمت دوم)

a)

وارون ناپذیر شدن ماتریس به این علت مشکل ساز خواهد بود که در توزیع گوسی چند بعدی هم وارون آن ماتریس covariance وجود دارد و هم از دترمینان آن در مخرج استفاده شده است بنابراین نیاز داریم که این ماتریس singular نشود

b)

در یکی از تخمین ها دترمینان ماتریس برابر ۰ شد . برای برطرف کردن این مسئله در لینک پیشنهاد شده روش های متعددی پیشنهاد شده بود. برای مثال یک روش خیلی ساده این بود که تنها عناصر قطری ماتریس covariance را در نظر بگیریم . این روش برای حل مشکل ما راهگشا نبود زیرا دترمینان ماتریس به این علت صفر می‌شد که یکی از ابعاد variance برابر ۰ داشت . این اتفاق به این معنی است که در داده های آن کلاس مقدار آن بعد از بردار ویژگی ثابت است و در نتیجه باعث وجود یک مقدار ویژه ۰ خواهد شد. روش دیگری که در لینک پیشنهاد شده بود استفاده از pooled covariance matrix بود و روش دیگر هم میانگین گیری بین دو ماتری covariance و identity بود . روشی که من برای حل مشکل استفاده کردم همان pca استفاده از روشی همانند pca در آن کلاس خاص بود به این صورت که بعد هایی از بردار ویژگی که واریانس آن ها ۰ بود یا به عبارتی ثابت بودند را از فرایند تخمین خارج کردم . این روش همان feature selection بود که در درس آموخته بودیم و پس از انجام روش پیش‌نهاد شده مشکل وارون پذیری ماتریس ها حل شد.

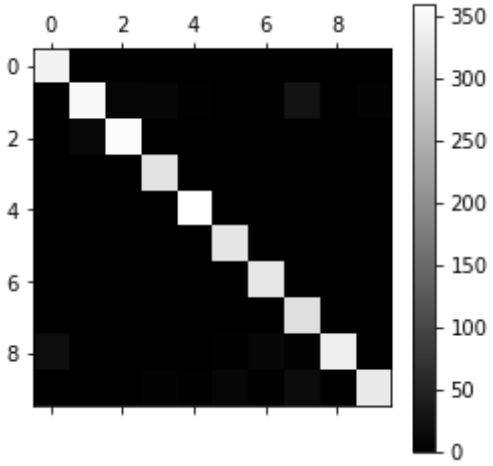
پس از پیاده سازی طبقه بند دقت طبقه بند حدود 95.88 درصد شد.

ماتریس confusion به صورت زیر بدست آمد.

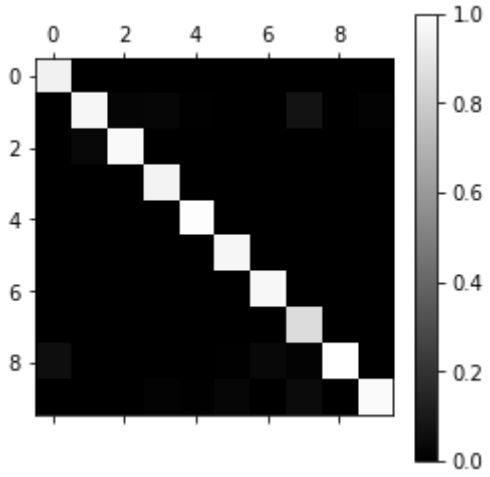
```
conf_mat = confusion_mat(test.labels , pred)
conf_mat
```

```
array([[342,   0,   0,   0,   0,   0,   0,   0,   0,   0],
       [  0, 350,   8,   9,   2,   0,   0,  28,   0,   5],
       [  0,  12, 355,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0, 320,   0,   1,   0,   0,   0,   0],
       [  0,   1,   0,   0, 360,   0,   0,   0,   0,   0],
       [  0,   0,   0,   1,   0, 323,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0, 325,   0,   0,   0],
       [  0,   0,   1,   1,   0,   0,   0, 314,   0,   1],
       [ 21,   1,   0,   0,   0,   2,  11,   5, 336,   1],
       [  0,   0,   0,   5,   2,   9,   0,  17,   0, 329]])
```

ماتریس confusion را نیز به صورت گرافیکی نیز بررسی کردم



می‌دانیم که بهتر از که ماتریس confusion را به صورت نرمال شده بر تعداد داده های هر کلاس نشان دهیم.
در زیر normalized confusion matrix آمده است



این ماتریس نیز به صورت قطری است که نشان دهنده دقت عالی طبقه بندی است.

سوال 5

پیاده سازی طبقه بند risk minimization در فایل .py وجود دارد و نحوه کار با ماژول های پیاده سازی شده نیز در jupyter notebook پیوست شده وجود دارد.

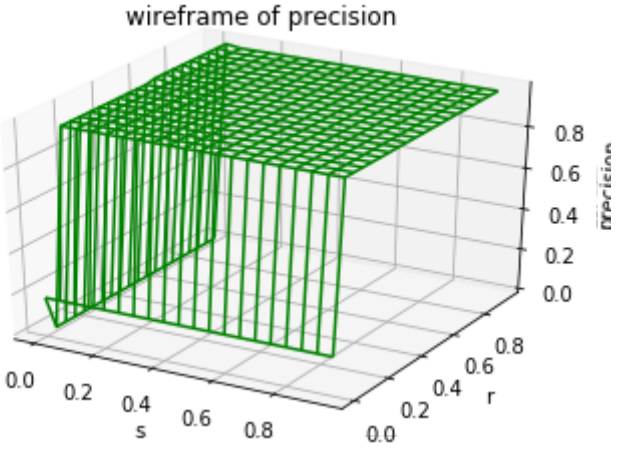
برای بدست آوردن s , r بهینه از exhaustive search استفاده کردیم و ترخ طبقه بندی های درست انجام شده (با در نظر نگرفتن رد شده ها) را به صورت ۳ بعدی رسم کردیم.

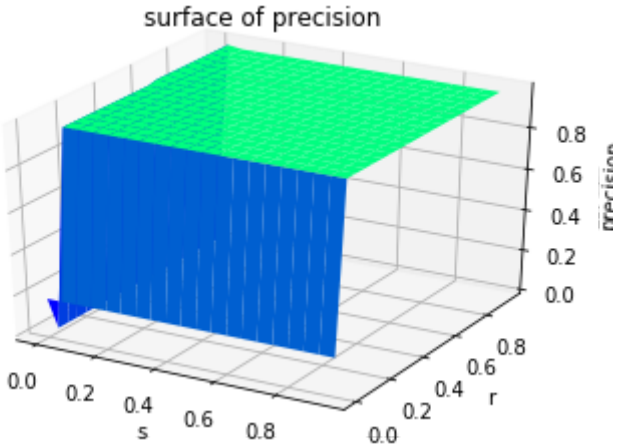
در ابتدا جست و جو را در بازه ۰ تا ۱ با step 0.05 انجام دادیم. و نتایج زیر بدست آمد

optimal s = 0.55

optimal r = 0.05

acc = 0.9896





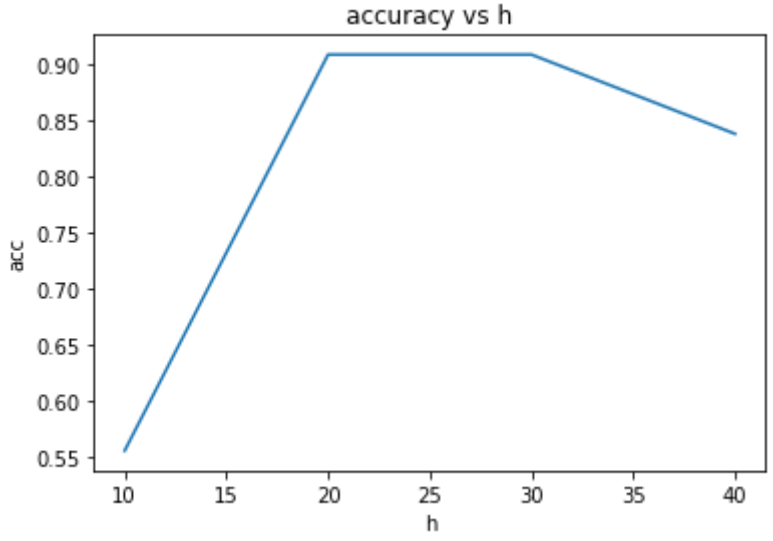
سوال 6

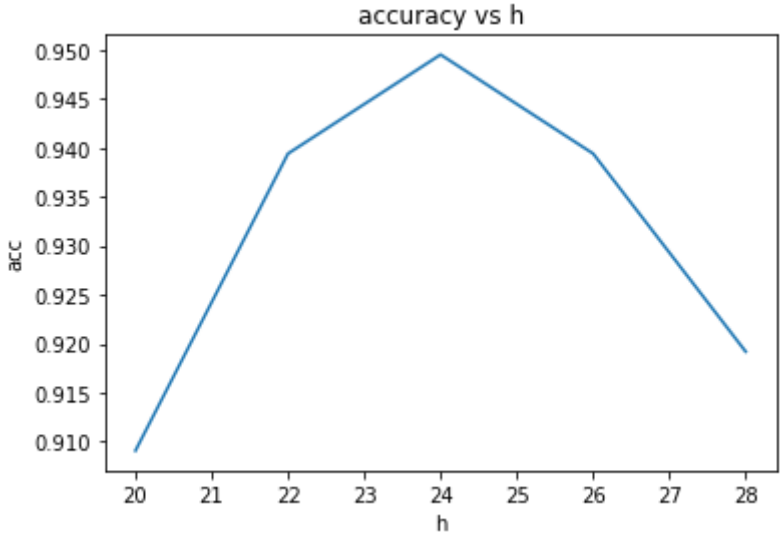
قسمت اول)

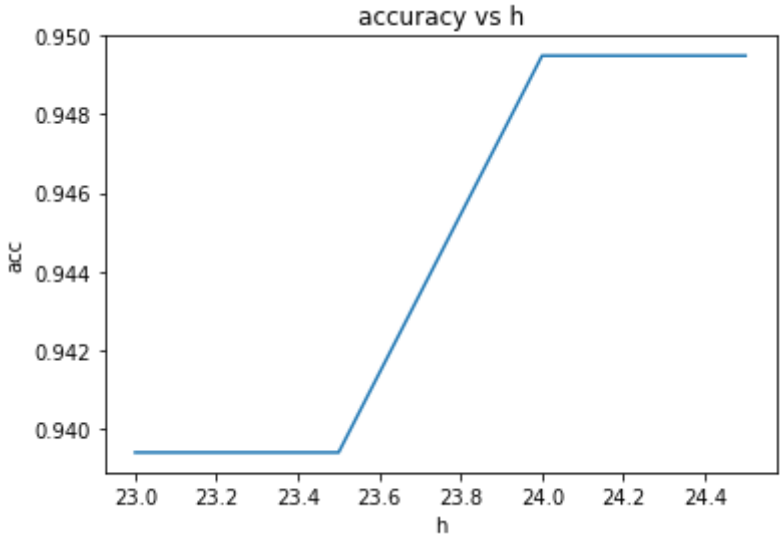
در این قسمت سعی می‌کنیم که بااستفاده از window ها مختلف توزیع هر کلاس را تخمین بزنیم. از ما خواسته شده است که این مسئله را برای دو نوع پنجره حل کنیم. (rectangular window and gaussian window) . یکی از چالش های اصلی در حل اینجور مسائل تعیین hyper parameter است. زیرا این نوع پارامتر ها تنها با exhaustive search بدست می‌آیند و هزینه این نوع جست و جو در تخمین های nonparametric بسیار زیاد است.

پنجره مستطیلی)

برای پنجره مستطیلی باید h مناسب را بدست بیاوریم. در jupyter notebook که پیوست شده است این search انجام شده و نتایج آن در عکس های زیر معلوم است.



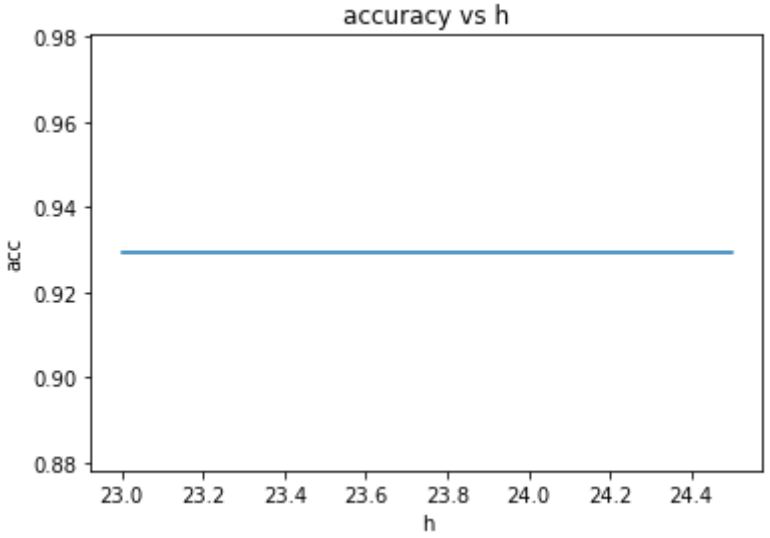


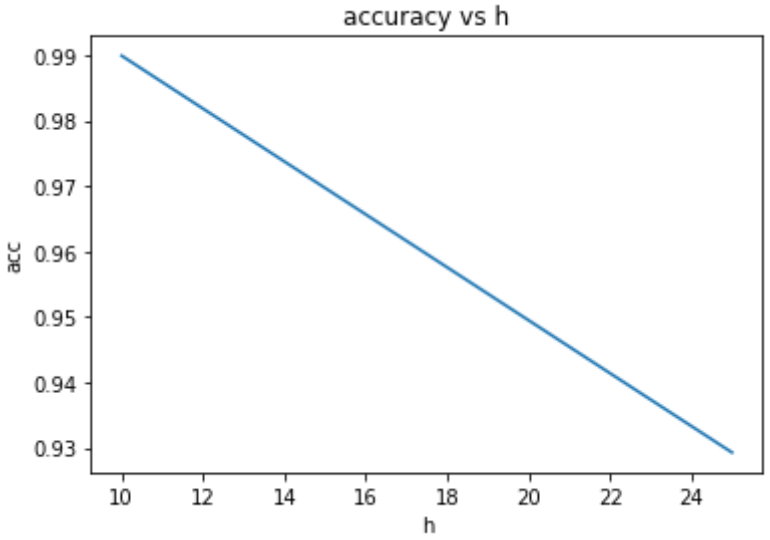


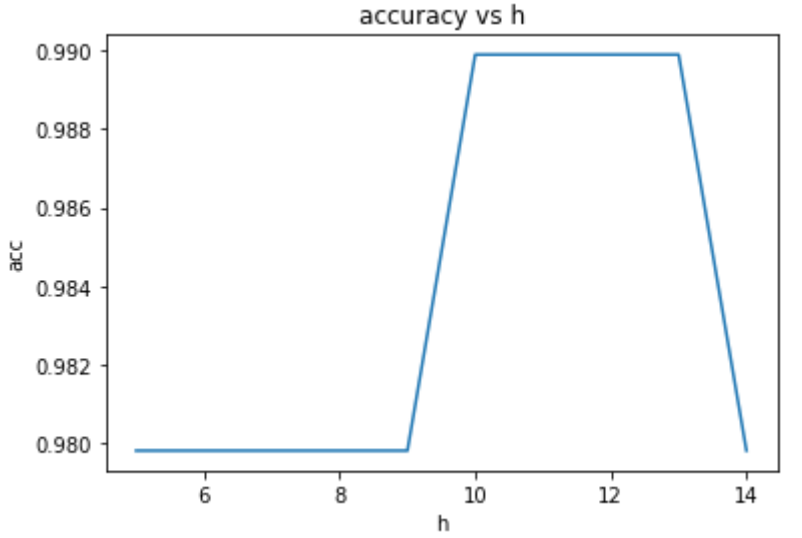
با توجه به نتایج بدست آمده بهترین h برای مسئله ما با توجه به داده های valid عدد ۲۴ است و accuracy طبقه بند در حدود 0.95 می باشد.

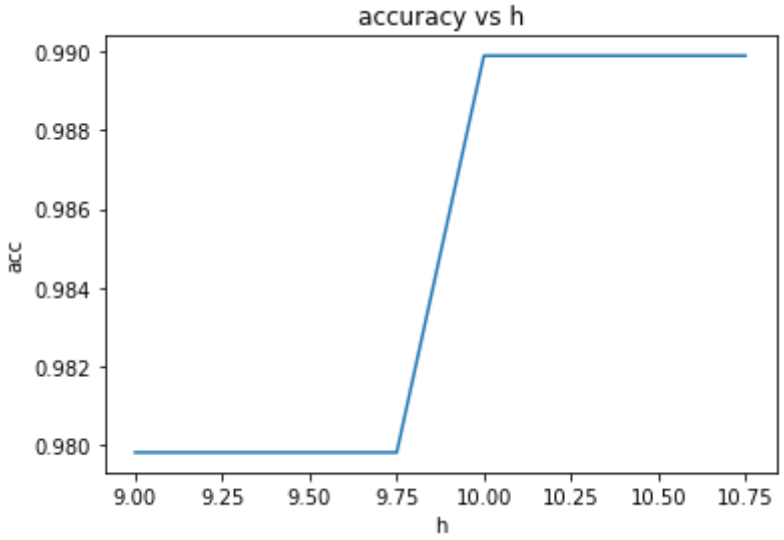
پنجره گوسی)

برای پنجره گوسی باید h مناسب را بدست بیاوریم. در jupyter notebook که پیوست شده است این search انجام شده و نتایج آن در عکس های زیر معلوم است.


accuracy vs h
0.98
0.96
0.94
0.92
0.90
0.88
acc
23.0
23.2
23.4
23.6
23.8
24.0
24.2
24.4
h


accuracy vs h
0.99
0.98
0.97
0.96
0.95
0.94
0.93
acc
10
12
14
16
18
20
22
24
h



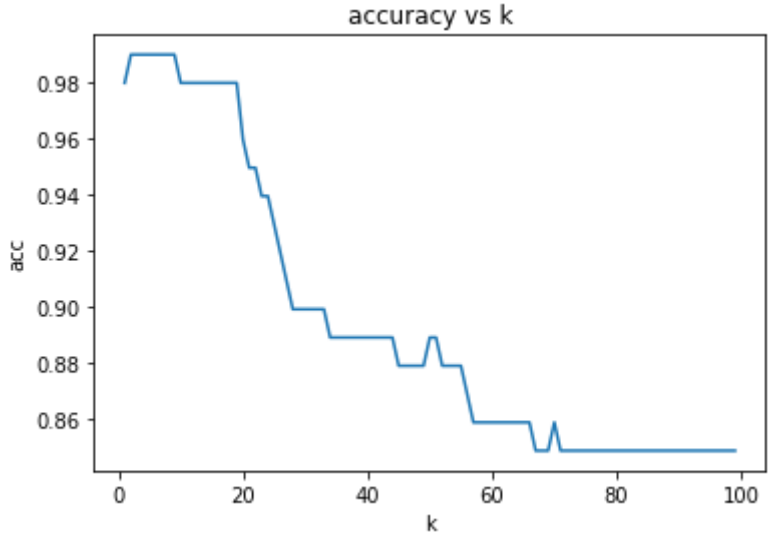


با توجه به نتایج بدست آمده بهترین h برای مسئله ما با توجه به داده های valid عدد 10 است و accuracy طبقه بند در حدود 0.99 می باشد.

قسمت دوم)

در این قسمت قسمت تخمین توزیع با استفاده از روش غیر پارامتری knn انجام شد. یکی از بهترین حالت های این طبقه بند وقتی است که با k=1 تولید شود که همان طبقه بند نزدیکترین همسایه نام دارد. دقت این طبقه بند برابر 0.9797 بدست آمد . پیاده سازی ها این قسمت نیز در فایل .py پیوست شده است و نحوه استفاده از این ماژول ها در jupyter notebook موجود است.

سپس برای پیدا کردن optimal k باید یک exhaustive search انجام دهیم . البته هزینه این نوع سرچ با توجه به پیاده سازی بهینه انجام شده برای knn خیلی کمتر از قسمت های پیشین است. در زیر نمودار حاصل از این search آمده است



با توجه به جست و جو انجام شده ، k بهینه برای این مسئله با توجه به داده های valid ، برابر 2 است و دقت طبقه بندی متناظر با آن برابر 0.98989898989899 می‌باشد

قسمت سوم)

در این قسمت از ما خواسته شده بود که طبقه بند knnرا پیاده سازی کنیم. پیاده سازی این قسمت در فایل .py پیوست شده وجود دارد و نحوه دقیق استفاده از این ماژول در jupyter notebook وجود دارد. در ادامه دقت های بدست آمده برای k های متفاوت آورده شده است.

k = 1 ⟶ acc = 0.9774156660949114

k= 3 ⟶ acc = 0.9779874213836478

k= 5 ⟶ acc = 0.9759862778730704

k= 10 ⟶ acc = 0.9748427672955975

سوال 7

قسمت ۱)

پیاده سازی های خواسته شده همگی در طی پروژه پیاده سازی شده اند و این پیاده سازی ها در فایل .py پیوست شده وجود دارند و همچنین روش دقیق استفاده از این ماژول ها نیز در jupyter notebook آمده است.

قسمت ۲)

a) classification accuracy

optimal bayes classifier with parametric gaussian estimation → 0.9588

optimal risk classifier with parametric gaussian estimation → 0.9896

optimal bayes classifier with non-param parzen estimation (rec window) → 0.95

optimal bayes classifier with non-param parzen estimation (gaussian win) → 0.99

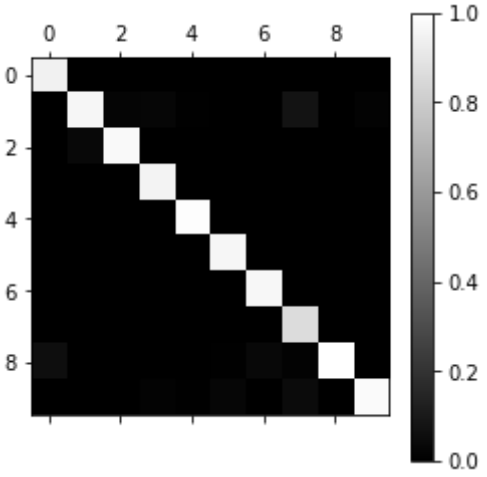
optimal bayes classifier with non-param knn estimation → 0.98

knn classifier → 0.97

تمامی طبقه بند های طراحی شده دقت قابل قبولی از خود نشان می‌دهند با توجه به این که ۱۰ کلاس در data set وجود دارد و اگر طبقه بند تصادفی طراحی می‌کردیم این دقت در حدود 0.1 می‌بود. پس برای مقایسه این طبقه بندها باید از دیگر ویژگی های آن ها استفاده کنیم.

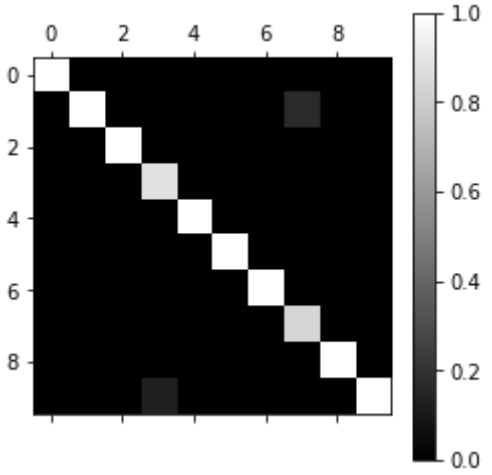
b) confusion matrix

optimal bayes classifier with parametric gaussian estimation :



optimal risk classifier with parametric gaussian estimation :

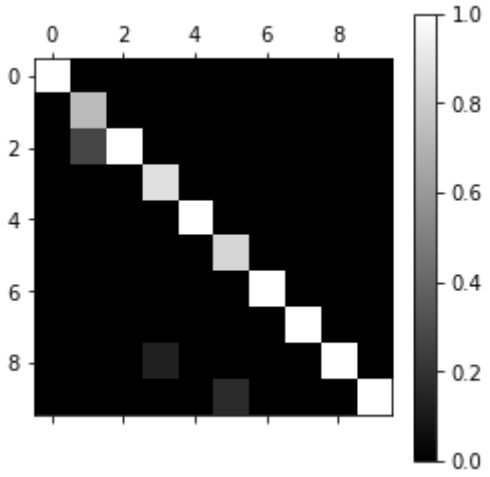
```
array([[16,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 11,  0,  0,  0,  0,  0,  1,  0,  0],
       [ 0,  0,  6,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  7,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  8,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  6,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  6,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  5,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 17,  0],
       [ 0,  0,  0,  1,  0,  0,  0,  0,  0, 15]])
```



در این نوع طبقه بند در حالتی که احتمال خطا را بهینه می‌کردیم این نکته قابل توجه است که چون یک سری از داده ها را reject می‌کنیم در داده هایی که طبقه بندی می‌کنیم دقت طبقه بندی بالاتر می‌رود ولی باید توجه داشت که این داده های reject شده باید در مراحل بعدی توسط طبقه بند های دیگری طبقه بندی شوند

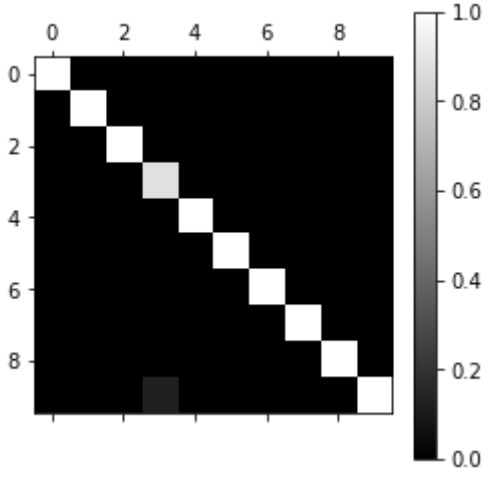
optimal bayes classifier with non-param parzen estimation (rec window) :

```
'([[16,  0,  0,  0,  0,  0,  0,  0,  0,  0],
   [ 0,  8,  0,  0,  0,  0,  0,  0,  0,  0],
   [ 0,  3,  6,  0,  0,  0,  0,  0,  0,  0],
   [ 0,  0,  0,  7,  0,  0,  0,  0,  0,  0],
   [ 0,  0,  0,  0,  8,  0,  0,  0,  0,  0],
   [ 0,  0,  0,  0,  0,  5,  0,  0,  0,  0],
   [ 0,  0,  0,  0,  0,  0,  6,  0,  0,  0],
   [ 0,  0,  0,  0,  0,  0,  0,  6,  0,  0],
   [ 0,  0,  0,  1,  0,  0,  0,  0, 17,  0],
   [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 15]])
```



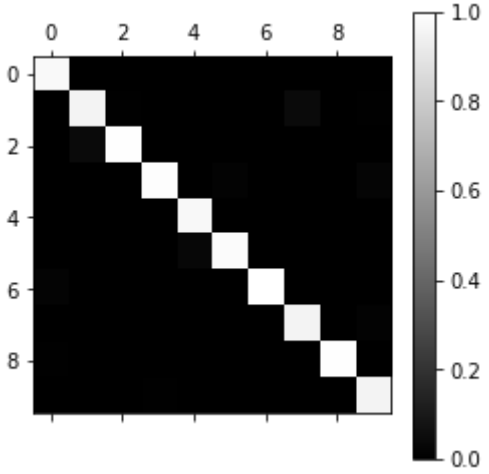
optimal bayes classifier with non-param parzen estimation (gaussian win) :

```
[[16,  0,  0,  0,  0,  0,  0,  0,  0,  0],
 [ 0, 11,  0,  0,  0,  0,  0,  0,  0,  0],
 [ 0,  0,  6,  0,  0,  0,  0,  0,  0,  0],
 [ 0,  0,  0,  7,  0,  0,  0,  0,  0,  0],
 [ 0,  0,  0,  0,  8,  0,  0,  0,  0,  0],
 [ 0,  0,  0,  0,  0,  6,  0,  0,  0,  0],
 [ 0,  0,  0,  0,  0,  0,  6,  0,  0,  0],
 [ 0,  0,  0,  0,  0,  0,  0,  6,  0,  0],
 [ 0,  0,  0,  0,  0,  0,  0,  0, 17,  0],
 [ 0,  0,  0,  1,  0,  0,  0,  0,  0, 15]]
```



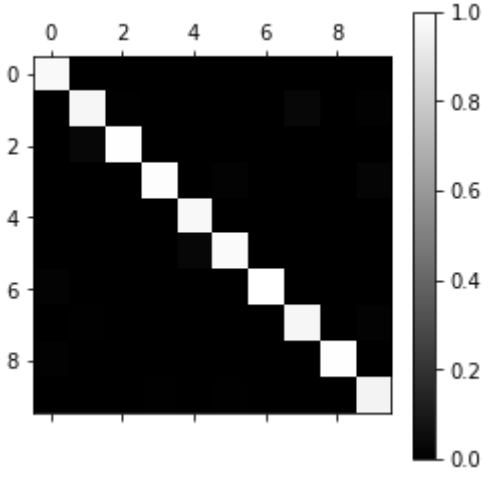
optimal bayes classifier with non-param knn estimation :

```
[[354,   0,   0,   0,   0,   0,   0,   0,   1,   0],
 [  0, 347,   2,   1,   0,   0,   0,  15,   0,   2],
 [  0,  15, 362,   0,   0,   0,   0,   1,   0,   0],
 [  0,   0,   0, 333,   0,   5,   0,   0,   0,   7],
 [  0,   1,   0,   0, 353,   0,   0,   0,   0,   0],
 [  0,   0,   0,   0,  10, 329,   0,   0,   1,   1],
 [  6,   0,   0,   0,   1,   0, 336,   0,   0,   0],
 [  0,   1,   0,   0,   0,   0,   0, 347,   0,   4],
 [  2,   0,   0,   0,   0,   0,   0,   1, 334,   1],
 [  1,   0,   0,   2,   0,   1,   0,   0,   0, 321]]
```



knn classifier :

```
[[354,   0,   0,   0,   0,   0,   0,   0,   1,   0],
 [  0, 350,   2,   1,   0,   0,   0,  10,   0,   3],
 [  0,  11, 362,   0,   0,   0,   0,   1,   0,   0],
 [  0,   0,   0, 333,   0,   5,   0,   0,   0,   7],
 [  0,   1,   0,   0, 354,   0,   0,   0,   0,   1],
 [  0,   0,   0,   0,  10, 328,   0,   0,   1,   1],
 [  5,   0,   0,   0,   0,   0, 336,   1,   0,   0],
 [  0,   2,   0,   0,   0,   0,   0, 351,   0,   4],
 [  3,   0,   0,   0,   0,   0,   0,   1, 334,   1],
 [  1,   0,   0,   2,   0,   2,   0,   0,   0, 319]]
```



هرچه confusion metrics قطری تر باشد بهتر است . از مقایسه confusion metrics های طبقه بند های بالا مهمترین اطلاعاتی که بدست می‌آید عملکرد بهتر gaussian window در مقایسه با rectangular window است.

c) required time for training the algorithm

طبقه بند هایی که به صورت nonparametric عمل می‌کنند هیچ کدام زمان خاصی برای train شدن نیاز ندارند در صورتی که طبقه بند هایی که به صورت parametric عمل می‌کنند زمان زیادی برای train شدن نیاز دارند.

در طبقه بند های این سوال تنها **optimal bayes classifier with parametric gaussian estimation** به مرحله training نیاز دارد. باید دقت شود که طبقه بند های parametric بسته به پیچیدگی تخمین هایی که انجام می‌دهند ممکن است که زمان خیلی زیادی برای train نیاز داشته باشند. در این سوال ما برای هر کلاس تنها یک گوسی تخمین زدیم که آن هم تنها ۲ پارامتر دارد. اگر بخواهیم از mixture density models استفاده کنیم و توزیع پایه در این مدل پیچیده خود توزیع پیچیده ای با چندین پارامتر باشد عملا روش های پارامتریک قابل استفاده نخواهد بود.

d) required time for testing the algorithm

طبقه بند هایی که به صورت nonparametric طبقه بندی را انجام می‌دهند ، برای طبقه بندی هر داده جدید به زمان زیادی نیاز خواهند داشت ، زیرا باید یک epoch از داده ها بگذرند. در مقابل طبقه بند هایی که به صورت parametric کار می‌کنند زمان زیادی برای طبقه بندی هر داده نیاز ندارند.

در پیاده سازی ما از آن جایی که برای تخمین به صورت knn از k های کوچک استفاده کردیم ، سرعت خوبی داشتند ولی با این حال کند تر از حالتی بودندکه به صورت parametric طبقه بندی می‌شدند به همین جهت من از قسمت کمتری از داده های تست برای ارزیابی عملکرد طبقه بند ها استفاده کردم

پیوست 1: روند اجرای برنامه

تمامی پیاده سازی ها در فایل hw3.py قرار دارند و نتایج بدست آمده در فایل assignment3.ipynb قرار دارد.