Question 1

a) Conditional probability of discrete random variable X given a discrete random variable Y is the probability distribution of X for any particular value of that of , i.e. Y = y, for any given value y. From here, we can write the formula for the conditional probability mass function of X given Y=y accordingly, as:

$$p_{X|Y}(x|y) = P(X = x|Y = y) = \frac{P(Y = y \cap X = x)}{P(Y = y)}$$

Note that for this expression to be valid, P(Y=y) must not be 0. The numerator is the intersection of two discrete random variables (the collection of elements that both X and Y have).

b) P(X: x = h) = 2/3, P(X: x = t) = 1/3
For three tosses, there are 8 combinations of results (h as 'head', t as 'tail'):

h-h-h
h-h-t
h-t-h
h-t-t
t-h-h
t-h-t
t-t-h
t-t-h

We know that the first one is a 'head' and there are two 'head's, so the two possible combinations are: h-h-t and h-t-h. Therefore:

$$P(\sum X = 2 : x_i = h | X : x_1 = h)$$
$$= (P(X : x_1 = h) \times P(X : x_2 = h) \times P(X : x_3 = t)) + (P(X : x_1 = h) \times P(X : x_2 = t) \times P(X : x_3 = h))$$
$$= (\frac{2}{3}) \times (\frac{2}{3}) \times (\frac{1}{3}) + (\frac{2}{3}) \times (\frac{1}{3}) \times (\frac{2}{3})$$
$$= \frac{8}{27}$$

c)
(i) P(X, Y) in terms of P(X) and P(Y|X) (joint probability in terms of conditional probability)

$$P(X \cap Y) = P(X)\ P(Y|X)$$

(ii) P(X, Y) = P(Y), P(X|Y)

$$P(X \cap Y) = P(Y)\ P(X|Y)$$

d) Bayes' Theorem Proof

The probability of two events X and Y happening, $P(X \cap Y)$, is the probability of X, P(X), times that of Y given that X has occurred, P(Y|X)

$$P(X \cap Y) = P(X)\, P(Y|X) \ldots\ldots (1)$$

Additionally, the probability of Y and X happening could also be expressed as the probability of Y, P(Y), times that of X given Y has occurred, P(X|Y)

$$P(Y \cap X) = P(Y)\, P(X|Y) \ldots\ldots (2)$$

(1) and (2) refer to the same quantity (probability of two same events), just with different wording,

$$P(X \cap Y) = P(Y \cap X) = P(X)\, P(Y|X) = P(Y)\, P(X|Y) \ldots\ldots (3)$$

Rearranging (3),

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

Which is the Bayes' Theorem

e)
i). We define the two discrete random variables as: student – S {Udem, McGill}, and billingualism – B {true, false}. This problem is solved under the assumption that students belong to either McGill or UdeM. From problem statement, we have: McGill 55% (P(S: s = McGill) = 0.55), then for UdeM:

$$P(S: s = UdeM) = 1 - 0.55 = 0.45$$

So the probability for a student to be from UdeM is 45%

ii). Billingualism: UdeM students 80%, McGill students 50%. We can re-write that as:

$$P(B: b = true \mid S: s = UdeM) = 0.8$$
$$P(B: b = true \mid S: s = McGill) = 0.5$$

Then the probability of this student being McGill student is:

$$P(B: b = true, S: s = McGill)$$
$$= P(B: b = true \mid S: s = McGill) \times P(S: s = McGill)$$
$$= 0.5 \times 0.55$$
$$= 0.275$$

Which is 27.5%

Question 2

a) From the table, we can see that:

$$P(word = \text{“}goal\text{”} | topic = \text{“}politics\text{”}) = \frac{7}{1000}$$

Or, in percentage terms, 0.7%

b) Again, refer to the table given:

$$P(word = \text{“}goal\text{”} | topic = \text{“}sports\text{”}) = \frac{1}{100}$$

Given the sample size (whose population contain solely of words of topic ‘sports’), the total expected appearance of word ‘goal’ is:

$$200 \times \frac{1}{100} = 2$$

The word ‘goal’ is expected to appear 2 times (twice) in this document.

c) Now we draw from the corpus, where we know a priori:

$$P(topic = \text{“}sports\text{”}) = \frac{2}{3}$$
$$P(topic = \text{“}politics\text{”}) = \frac{1}{3}$$

Both topics contain the word ‘goal’, where:

$$P(word = \text{“}goal\text{”} | topic = \text{“}sports\text{”}) = \frac{1}{100}$$
$$P(word = \text{“}goal\text{”} | topic = \text{“}politics\text{”}) = \frac{7}{1000}$$

Therefore, the probability of having ‘goal’ across the entire corpus is:

$$\begin{aligned} & P(word = \text{“}goal\text{”}) = \sum P(word = \text{“}goal\text{”} | topics) \times P(topics) \\ = \; & P(word = \text{“}goal\text{”} | topic = \text{“}sports\text{”}) \times P(topic = \text{“}sports\text{”}) + \dots \\ & P(word = \text{“}goal\text{”} | topic = \text{“}politics\text{”}) \times P(topic = \text{“}politics\text{”}) \\ = \; & \frac{1}{100} \times \frac{2}{3} + \frac{7}{1000} \times \frac{1}{3} = \frac{9}{1000} \end{aligned}$$

d) Now we draw from the corpus, where we know a priori:

$$P(topic = \text{“}sports\text{”}) = \frac{2}{3}$$
$$P(topic = \text{“}politics\text{”}) = \frac{1}{3}$$

Both topics contain the word ‘kick’, where:

$$P(word = \text{“}kick\text{”} | topic = \text{“}sports\text{”}) = \frac{1}{200}$$
$$P(word = \text{“}kick\text{”} | topic = \text{“}politics\text{”}) = \frac{3}{1000}$$

The probability of word ‘kick’ appearing in documents of topic ‘sports’ is:

$$P(topic = \text{“}sports\text{”} | word = \text{“}kick\text{”})$$

$$= \frac{P(word = \text{“}kick\text{”} | topic = \text{“}sports\text{”})}{\sum P(word = \text{“}kick\text{”} | topics)} = \ldots$$

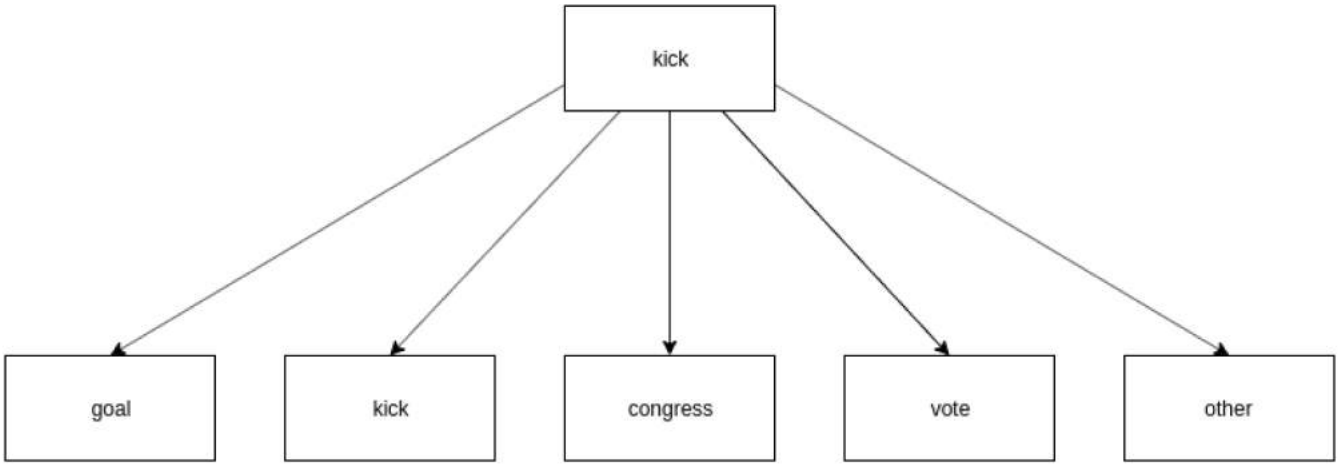
$$\frac{P(word = \text{“}kick\text{”} | topic = \text{“}sports\text{”}) \times P(topic = \text{“}sports\text{”})}{P(word = kick | topic = sports) \times P(topic = sports) + P(word = kick | topic = politics) \times P(topic = politics)}$$

$$= \frac{\frac{1}{200} \times \frac{2}{3}}{\frac{1}{200} \times \frac{2}{3} + \frac{3}{1000} \times \frac{1}{3}}$$

$$= \frac{\frac{1}{300}}{\frac{13}{3000}}$$

$$= \frac{10}{13}$$

e) We draw the following chart to help visualize:



There are two levels which we need to traverse to reach the stated objective. Firstly, to have the first word as ‘kick’, the probability is:

P(word = “kick” | topics) = $\sum P(word = "kick" \mid topics)\, P(topics)$
= P(word = “kick” | topic = “politics”) P(topic = “politics”) + P(word = “kick” | topic = “sports”) P(topic = “sports”)
$= \frac{3}{1000} \times \frac{1}{3} + \frac{1}{200} \times \frac{2}{3}$
$= \frac{13}{3000}$

This is the probability to reach the first level of selection of ‘kicks’. For the second level, we are to find the probability of ‘goal’:

P(word = “goal” | topics) = $\sum P(word = "goal" \mid topics)\, P(topics)$
= P(word = “goal” | topic = “politics”) P(topic = “politics”) + P(word = “goal” | topic = “sports”) P(topic = “sports”)
$= \frac{7}{100} \times \frac{1}{3} + \frac{1}{100} \times \frac{2}{3}$
$= \frac{3}{100}$

The overall probability is then the product of the two, where:

$$\frac{P(word = \text{“}goal\text{”} \mid word = \text{“}kick\text{”})}{P(word = \text{“}goal\text{”} \mid topics)} = P(word = \text{“}kick\text{”} \mid topics)$$

which:

P(word = “goal” | word = “kick”) = P(word = “kick” | topics) × P(word = “goal” | topics)
$= \frac{13}{3000} \times \frac{3}{100}$
$= \frac{13}{100000}$

Or, in percentage, 0.013%

f) We could construct a data set expressed in the form similar to the ‘iris’ example. The idea is to have *d* columns of features, where each ‘feature’ is simply the word we are interested in, say ‘kick’, ‘goal’, ‘congress’. If a word is ‘kick’, we put ‘1’ and ‘0’ on all other feature columns, and vice versa.

We already know that the labels: politics, sports. We can then traverse through the documents and fill in the corresponding value. An example is theSn:

Feature					Label
kick	goal	congress	vote	others	(topic)
1	0	0	0	0	sports
0	1	0	0	0	sports
0	0	1	0	0	politics
0	0	0	1	0	sports
0	0	1	0	0	politics
0	0	0	0	1	politics

$$
\begin{array}{c}
10000s \\
01000s \\
00100p \\
00010s \\
00100s \\
00001p \\
\ldots
\end{array}
$$

We could then manipulate to get some values. For example, by sorting and splitting based on labels, we could determine topic probabilities. The same sort/split operations could be used to further understand the conditional topical probabilities, where by we sort and split certain feature (column) within one label (politics or sports).

To achieve that, the smallest value of θ is then the maximum value of x, given the relation $0 \leq x_i \leq \theta$, therefore, we have:

$$\hat{\theta} = max(x_1, x_2, \dots, x_n)$$

Question 4

$$f_\theta = 2\theta x e^{-\theta x^2}$$

$$lik(\theta) = (2\theta x_1 e^{-\theta {x_1}^2})(2\theta x_2 e^{-\theta {x_2}^2})\dots(2\theta x_n e^{-\theta {x_n}^2}) = \prod_{i=1}^{n} 2\theta x_i e^{-\theta {x_i}^2}$$

Use the monotonicity of logarithmic functions, and given the fact that x is strictly positive for all samples, we can simplify the logarithmic of the likelihood function. If we take the logarithmic, we get:

$$log(lik(\theta)) = \sum_{i=1}^{n} log(2\theta x_i e^{-\theta {x_i}^2}) = \sum_{i=1}^{n} log(2\theta x_i) + log(e^{-\theta {x_i}^2}) = \sum_{i=1}^{n} log(2\theta x_i) - \theta {x_i}^2$$

To find the maximum, we take the derivative and set it to zero

$$\underset{\theta}{argmax}(log(lik(\theta))) = \underset{\theta}{argmax} \sum_{i=1}^{n} (log(2\theta x_i) - \theta {x_i}^2)$$

$$\frac{d(log(lik(\theta)))}{d\theta} = 0$$

$$\frac{d(\sum_{i=1}^{n} log(2\theta x_i) - \theta {x_i}^2)}{d\theta} = 0$$

$$\sum_{i=1}^{n} \frac{1}{\theta} - {x_i}^2 = \frac{n}{\theta} - \sum_{i=1}^{n} {x_i}^2 = 0$$

The MLE for θ is then,

$$\hat{\theta} = \frac{n}{\sum_{i=1}^{n} {x_i}^2}$$

Question 5

a) $P_n(e) = P_n(e|x)\, P_n(x)$

from question, we know that

$P(S^+) = P(S^-) = 50\% = 1/2$

$\therefore P_n(x) = (\frac{1}{2})^n = \prod_{i=0}^{n} (\frac{1}{2})$ ... ①

n as n-points

for k-neighbours, we have $\frac{k-1}{2}$ classification tasks:

$P_n(e|x) = C_0^n + C_1^n + C_2^n + C_3^n + \cdots C_{\frac{k-1}{2}}^n$

$= \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j} = \sum_{j=0}^{\frac{k-1}{2}} C_j^n$ ... ②

① × ②:

$P_n(e) = P_n(e|x) P_n(x)$

$= (\frac{1}{2})^n \cdot \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j}$

b) $P_n(e) = \frac{1}{2^n} \sum_{j=0}^{(k-1)/2} \binom{n}{j}$

$k=1$,

$P_n(e) = \frac{1}{2^n} \sum_{j=0}^{0} \binom{n}{j} = \frac{1}{2^n} \binom{n}{0} = \frac{1}{2^n} \left(\frac{n!}{0!\,n!}\right)$

$k=3$.

$P_n(e) = \frac{1}{2^n} \sum_{j=0}^{1} \binom{n}{j} = \frac{1}{2^n}\left[\binom{n}{0} + \binom{n}{1}\right]$

$= \frac{1}{2^n}\left[\frac{n!}{0!\,n!} + \frac{n!}{1!\,(n-1)!}\right]$

$k=5$.

$P_n(e) = \frac{1}{2^n}\left[\frac{n!}{0!\,n!} + \underbrace{\frac{n!}{1!\,(n-1)!} + \frac{n!}{2!\,(n-2)!}}_{①}\right]$

① >0 for $n>0$, which means $\sum_{j=0}^{(k-1)/2}$ has smallest value of 1 when $k=1$. As k increases, more terms will make the series evaluate to larger values.

$j=0$	$\frac{n!}{0!\,n!} = 1$	>0
$j=1$	$\frac{n!}{1!\,(n-1)!} = n$	>0
$j=2$	$\frac{n!}{2!\,(n-2)!} = \frac{n(n-1)}{2}$	>0 for $n>1$
$j=3$	$\frac{n!}{3!\,(n-3)!} = \frac{n(n-1)(n-2)}{6}$	>0 for $n>2$

$\therefore$ for $k=1$, $P_n(e) = \frac{1}{2^n} < P_n(e)$ where $k>1$ $(3, 5, \ldots)$

c) $P_n(e) = \sum_{j=0}^{k-1/2} \frac{1}{2^n}\binom{n}{j}$

$= \frac{1}{2^n}\left[\frac{n!}{0!n!} + \frac{n!}{1!(n-1)!} + \cdots + \frac{n!}{(\frac{k-1}{2})!(n-\frac{k-1}{2})!}\right]$

as n increases. we need to make sure largest term converge. by ratio test:

$L\left|\frac{a_{n+1}}{a_n}\right| = \frac{(n+1)(n)(n-1)\cdots(n-\frac{k-1}{2})}{n(n-1)(n-2)\cdots(n-\frac{k-1}{2}+1)} \cdot \frac{(\frac{k-1}{2})!}{(\frac{k-1}{2}+1)!} \cdot \frac{2^n}{2^{n+1}}$

$= \left(\frac{n+1}{n-\frac{k-1}{2}-1}\right)\left(\frac{1}{2}\right)\frac{1}{(\frac{k-1}{2}+1)}$

$= \left(\frac{2n+2}{2n-(k-1)-2}\right)\left(\frac{1}{2}\right)\left(\frac{2}{k-1+2}\right)$

$= \left(\frac{2n+2}{2n-k-1}\right)\left(\frac{1}{2}\right)\left(\frac{2}{k+1}\right)$

$= \frac{2(n+1)}{(2n-k-1)(k+1)} < 1$

$2n+2 < 2nk+2n-k^2-2k-1$

$k^2 - 2nk + 2k + 3 < 0$

~~$(k^2-2nk+1)+(2k+2)<0$~~

~~$(k-1)^2$~~

$(k^2+2k+1) + 2 - 2nk < 0$

$(k+1)^2 + 2 < 2nk \quad \cdots$ ①

now, if $k \le a\sqrt{n}$, ① would satisfy for most'd as constant

$(k+1)^2 < 2nk - 2$

$k+1 < \sqrt{2nk-2}$

$k < \sqrt{2nk-2} - 1$

it would still be the same order. Then

$\lim_{n\to\infty} P_n(e) = 0$

Question 6

Since it's a fair coin:

$$P(Y=0) = P(Y=1) = 0.5$$

To get $P(Y=0 \mid X=x)$:

$$\frac{P(X=x \mid Y=0)\,P(Y=0)}{\sum_{i=1}^{n} P(X=x \mid Y=y)\,P(Y=y)}$$

$$= \frac{P(X=x \mid Y=0)\,P(Y=0)}{P(X=x \mid Y=0)\,P(Y=0) + P(X=x \mid Y=1)\,P(Y=1)} = \frac{①}{① + ②}$$

① $P(X=1 \mid Y=0)\,P(Y=0) = f_{\mu_1,\Sigma_1}(x)\left(\frac{1}{2}\right)$

② $P(X=1 \mid Y=1)\,P(Y=1) = f_{\mu_2,\Sigma_2}(x)\left(\frac{1}{2}\right)$

$$\therefore P(Y=0 \mid X=x) = \frac{f_{\mu_1,\Sigma_1}(x)}{f_{\mu_1,\Sigma_1}(x) + f_{\mu_2,\Sigma_2}(x)}$$

$$= \frac{\left(\frac{1}{2\pi}\right)\left(\det(\Sigma_1)\right)^{-1/2} \cdot \exp\left[-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)\right]}{\left(\frac{1}{2\pi}\right)\left[\det(\Sigma_1)^{-1/2} \cdot \exp\left[-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] + \det(\Sigma_2)^{-1/2} \cdot \exp\left[-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]\right]}$$

$$= \frac{\text{top} \cdot \left(\det(\Sigma_1)\right)^{1/2} \cdot \exp\left[+\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right]}{\text{bottom} \cdot \left(\det(\Sigma_1)\right)^{1/2} \cdot \exp\left[+\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right]}$$

$$= \frac{1}{1 + \sqrt{\frac{\det(\Sigma_1)}{\det(\Sigma_2)}} \cdot \exp\left[\frac{1}{2}\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]\right]}$$

$$= \frac{1}{1 + \left[\left(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\right) \cdot e^{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}\right]^{1/2}}$$

$$= \frac{1 - \sqrt{\frac{\det(\Sigma_1)}{\det(\Sigma_2)}} \cdot e^{\frac{1}{2}\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]}}{1 - \frac{\det(\Sigma_1)}{\det(\Sigma_2)} \cdot e^{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}$$

$$= \frac{\det(\Sigma_2) - \sqrt{\det(\Sigma_1)\det(\Sigma_2)} \cdot e^{\frac{1}{2}\left[(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right]}}{\det(\Sigma_2) - \det(\Sigma_1) \cdot e^{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}$$

This is the most simplified fraction free form.

We can also write:

$$P(Y=0 \mid X=x) = \frac{\det(\Sigma_2) - \sqrt{\det(\Sigma_1)\det(\Sigma_2)}\, e^{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}{\det(\Sigma_2) - \det(\Sigma_1) \cdot e^{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}$$

Parzen

September 28, 2019

1 Q5

1.1 Code

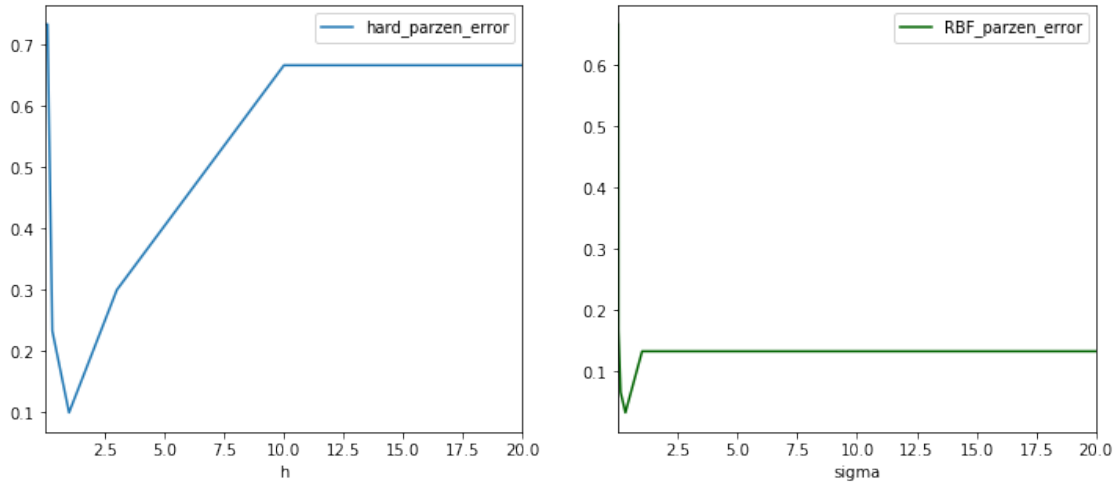
[208]:
```
from solution import *

import pandas as pd
import matplotlib.pyplot as plt
```

[209]:
```
def get_test_errors2(iris):
    hs = [0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0]
    sigmas = [0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0]

    (training_set, validation_set, test_set) = split_dataset(iris)

    tr_x, tr_y = training_set[:, :-1], training_set[:, -1]
    val_x, val_y = validation_set[:, :-1], validation_set[:, -1]
    te_x, te_y = test_set[:, :-1], test_set[:, -1]

    er = ErrorRate(tr_x, tr_y, val_x, val_y)

    er_hp = np.array([er.hard_parzen(h) for h in hs])
    er_sp = np.array([er.soft_parzen(sigma) for sigma in sigmas])

    return hs, sigmas, er_hp, er_sp


hs, sigmas, er_hp, er_sp = get_test_errors2(iris)
```

1.2 Results

[213]:
```
df_q5 = pd.DataFrame(np.array([hs, sigmas, er_hp, er_sp]).T, columns=['h',␣
 ↪'sigma', 'hard_parzen_error', 'RBF_parzen_error'])
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
df_q5[['h','hard_parzen_error']].plot(x='h', y='hard_parzen_error', ax=axes[0])
df_q5[['sigma','RBF_parzen_error']].plot(x='sigma', y='RBF_parzen_error',␣
 ↪ax=axes[1], color='DarkGreen')
```

[213]: `<matplotlib.axes._subplots.AxesSubplot at 0x19f5808e9e8>`



1.3 Analysis

The results of Q5 are shown in above figure. It is obvious that the error rates of the two methods dropped rapaidly when their parameters were under 1. The figures then gradually rose to 0.67 and 0.13, respectively. After that, both two figures maintained stably.

However, soft RBF Parzen performed much better than hard Parzen: (1) the minimal error rate for soft RBF Parzen is 0.03, only one-third of that of hard Parzen (0.1). (2) soft RBF Parzen had much better results even that it had improper parameter settings.

2 Q7

2.1 Complexity Analysis

The complexity of those two methods are clear.

For each test example, both two methods need to calculate the standard Euclidean distance with each sample in training data. This calculation requires $O(nd)$, where n denotes the number of training examples and d is the dimension of each example.

For hard Parzen, it then need to campare the distance with h to filter out training examples out of the window, and it can also count the votes when meeting examples in window. So, this step requires $O(n)$. Finally, obtaining the label with most votes requires $O(m)$, where m denotes the number of different classes. Therefore, the complexity of hard Parzen for each test example is

$O(nd + n + m)$,

and it does not change when h varies. This is because that this method have to calculate distances with all training examples.

However, soft RBF Parzen does not need to check whether each training example is in the window, instead it directly calculate KDE. Hence, its complexity is

$O(nd + m)$,

and it also does not change when σ varies, as σ only participates in the calculation of KDE.

3 Q9

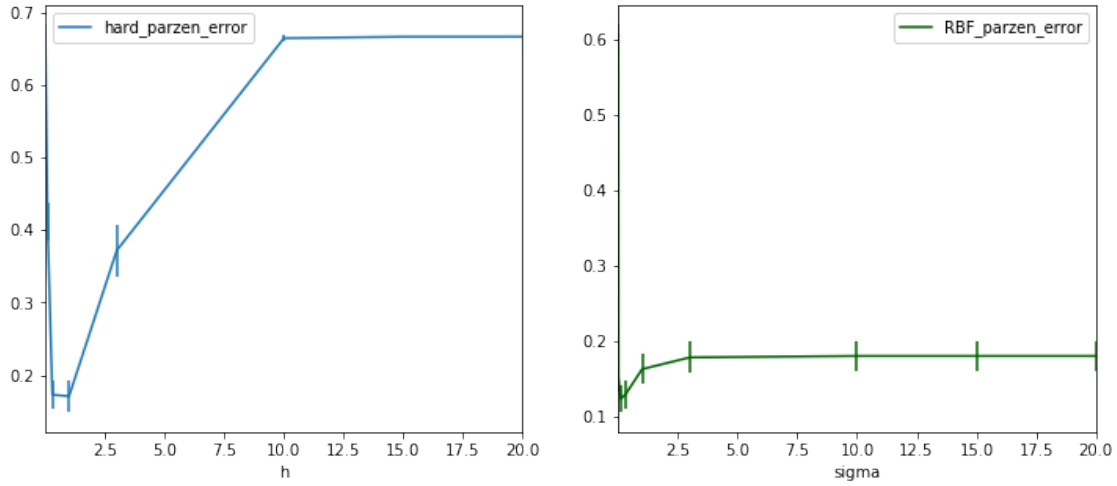
3.1 Code

[ ]:
```
def get_val_errors(iris, A):


    hs = [0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0]
    sigmas = [0.001, 0.01, 0.1, 0.3, 1.0, 3.0, 10.0, 15.0, 20.0]

    (training_set, validation_set, test_set) = split_dataset(iris)

    def project_data(data):
        projected_x = random_projections(data[:, :-1], A)
        return np.concatenate([projected_x, np.reshape(data[:, -1],[-1,1])],␣
 ↪axis=-1)



    training_set = project_data(training_set)
    validation_set = project_data(validation_set)
    test_set = project_data(test_set)

    tr_x, tr_y = training_set[:, :-1], training_set[:, -1]
    val_x, val_y = validation_set[:, :-1], validation_set[:, -1]
    te_x, te_y = test_set[:, :-1], test_set[:, -1]

    er = ErrorRate(tr_x, tr_y, val_x, val_y)

    er_hp = np.array([er.hard_parzen(h) for h in hs])
    er_sp = np.array([er.soft_parzen(sigma) for sigma in sigmas])

    return (er_hp, er_sp)

def get_q9_data(iris):
    results = []
    for i in range(500):
        A = np.random.normal(0,1, 8).reshape([-1, 2])
        results.append(get_val_errors(iris.copy(), A))
        if i % 10 == 0 :
            print('%i of %i' % (i+1, 500), end='\r')

    return results

results = get_q9_data(iris)
```

3.2 Results

[204]:
```
mean_results = np.array(results).mean(axis=0)
std_results = np.array(results).std(axis=0)
df_q9 = pd.DataFrame(np.array([hs, sigmas, mean_results[0], mean_results[1],␣
 ↪std_results[0]*0.2, std_results[1]*0.2]).T,
                     columns=['h', 'sigma', 'hard_parzen_error',␣
 ↪'RBF_parzen_error', 'hard_std', 'RBF_std'])


fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))

df_q9[['h','hard_parzen_error', 'hard_std']].plot(x='h', y='hard_parzen_error',␣
 ↪yerr='hard_std', ax=axes[0])
df_q9[['sigma','RBF_parzen_error', 'RBF_std']].plot(x='sigma',␣
 ↪y='RBF_parzen_error', yerr='RBF_std', ax=axes[1], color='DarkGreen')
```

[204]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19f58625080>
```



3.3 Analysis

The results of Q9 are shown in the above figure.

Comparing with the figure of Q5, the average error rates of Q9 were always higher. For example, the minimal error rates for hard Parzen and soft RBF Parzen were 0.17 and 0.12 respectively, where the figures in Q5 were only 0.10 and 0.03 respectively. Therefore, the performance of random projections dropped.

However, on the other hand, the curve was clearly gentler than that of Q5, which may be more helpful to find h^* or σ^*.