CS-473
Pattern Recognition

Tutorial for Assignment 3

T.As.: Myrto Villia , Despina - Ekaterini Argiropoulos, Michalis Savorianakis

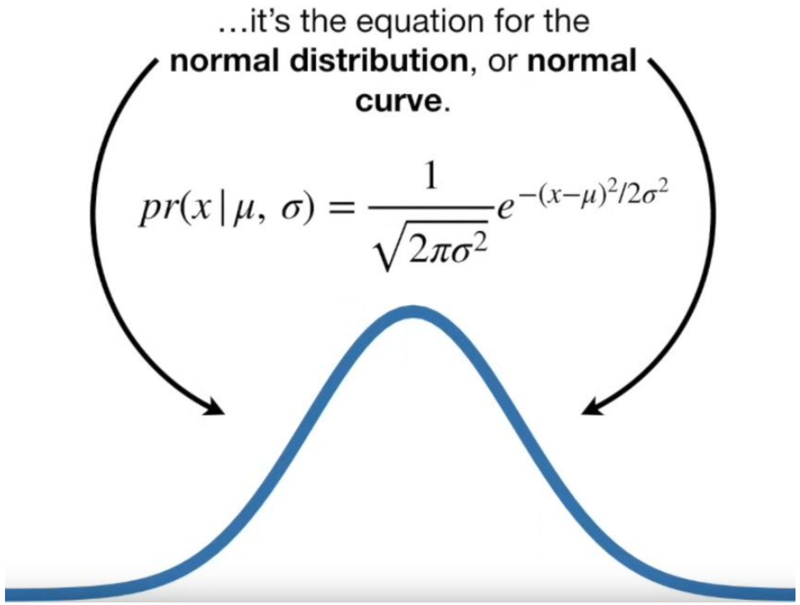
Tutor: Panos Trahanias

April, 2025

Outline of the presentation

- Maximum Likelihood Estimation for Parameter Estimation - **PART A**
- Parzen Windows - **PART B**
- K-Nearest Neighbors (KNN) Classification - **PART C**

Maximum Likelihood Estimation for Parameter Estimation - **PART A**



...it's the equation for the **normal distribution**, or **normal curve**.

$$pr(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

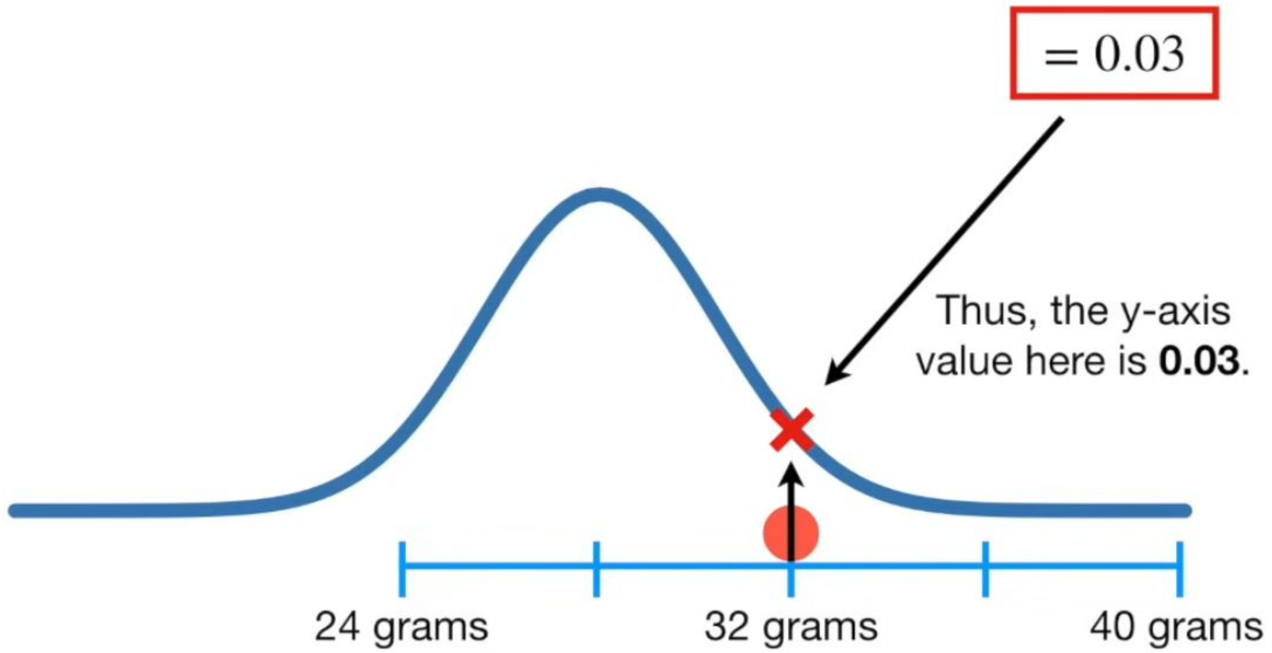
…to find the optimal values for ***μ*** (the mean) and ***σ*** (the standard deviation) given some data, ***x***

$$pr(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

$$L\boxed{(\mu, \sigma \mid x)} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

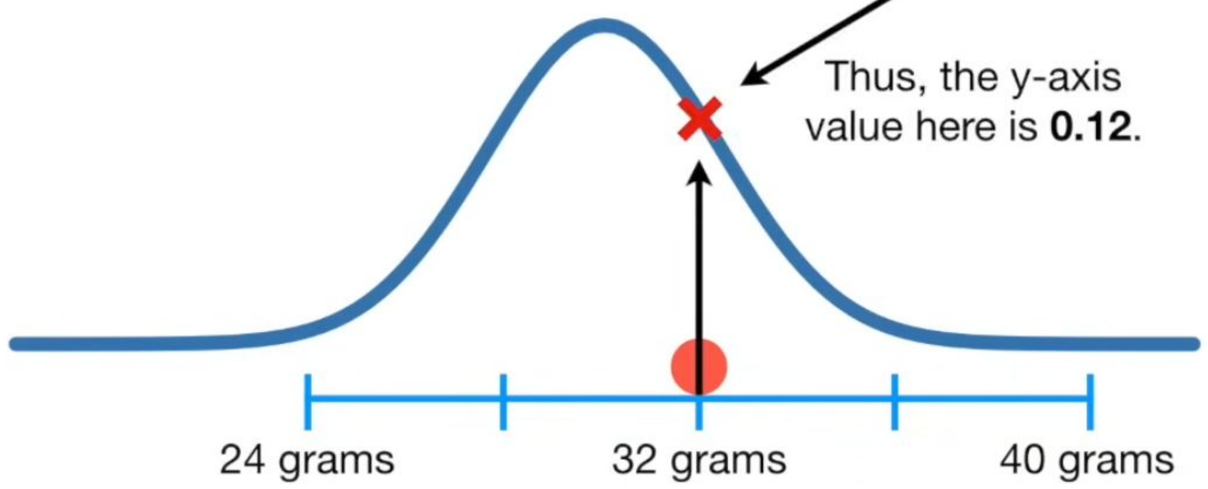
$$L(\mu = 28,\ \sigma = 2 \,|\, x = 32) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} = \frac{1}{\sqrt{2\pi 2^2}} e^{-(32-28)^2/2\times 2^2}$$



Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$L(\mu = 30,\ \sigma = 2 \,|\, x = 32) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} = \frac{1}{\sqrt{2\pi 2^2}} e^{-(32-30)^2/2\times 2^2}$$

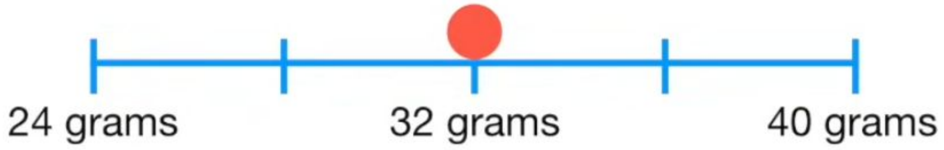
$$\boxed{= 0.12}$$



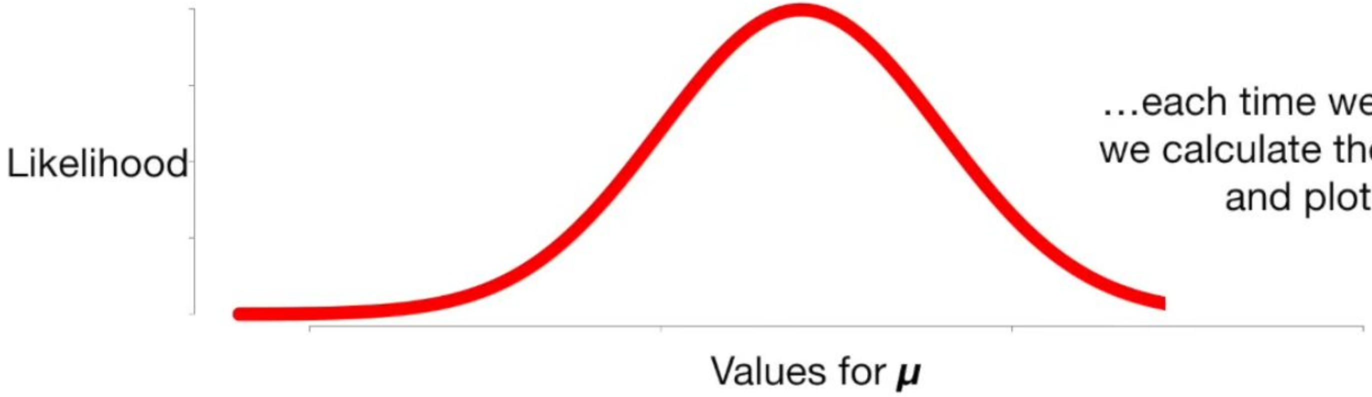
Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\boxed{L(\mu}\, x = 32,\ \sigma = 2) = \frac{1}{\sqrt{2\pi 2^2}} e^{-(32-\mu)^2/2\times 2^2}$$

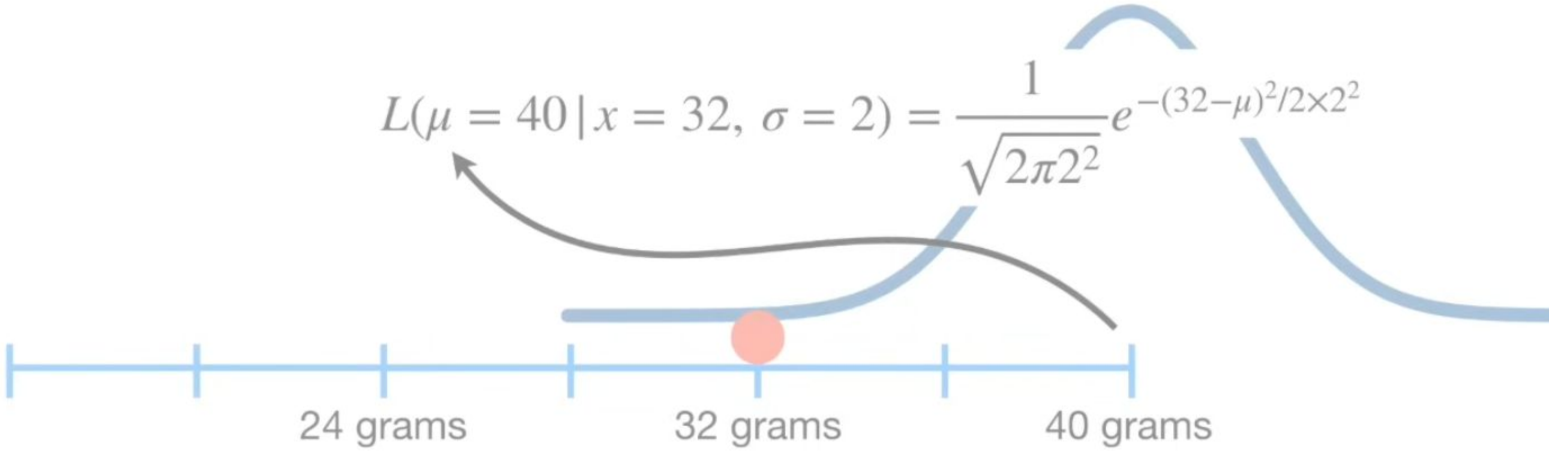
…then we can plug in a whole bunch of values for ***μ*** and see which one gives the maximum likelihood.



Maximum Likelihood Estimation for Parameter Estimation - **PART A**

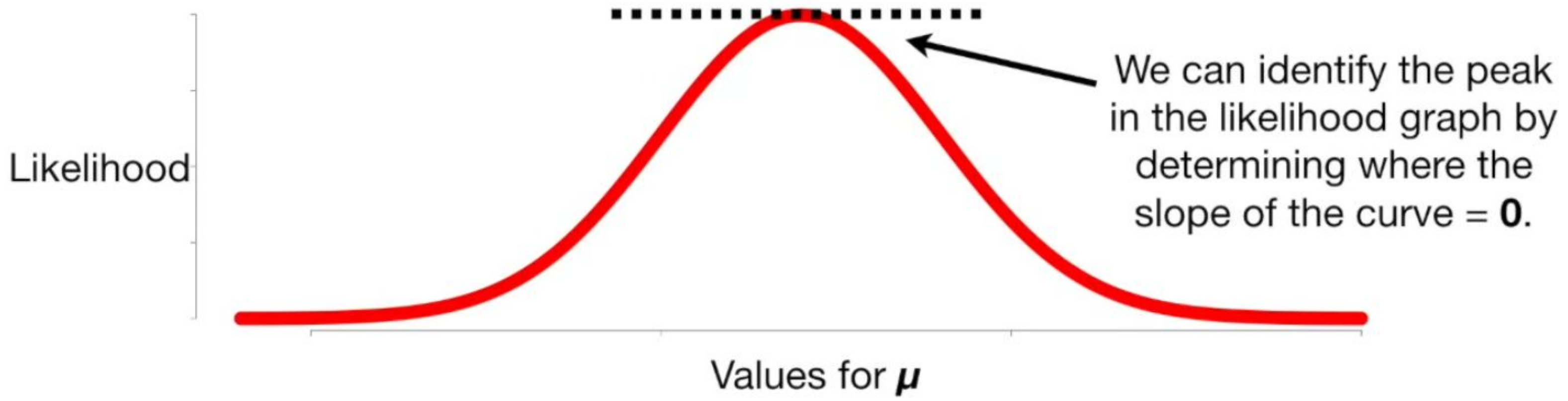


…each time we change $\boldsymbol{\mu}$, we calculate the likelihood and plot it…

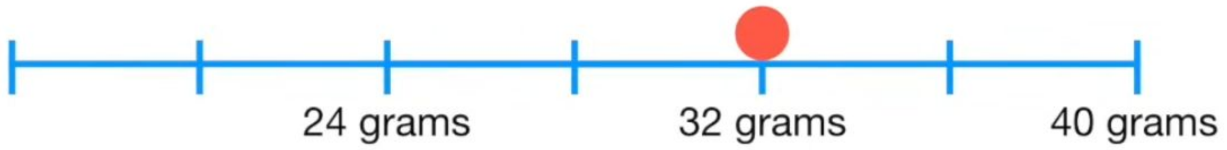


$$L(\mu = 40 \,|\, x = 32,\ \sigma = 2) = \frac{1}{\sqrt{2\pi 2^2}} e^{-(32-\mu)^2/2\times 2^2}$$

Maximum Likelihood Estimation for Parameter Estimation - **PART A**



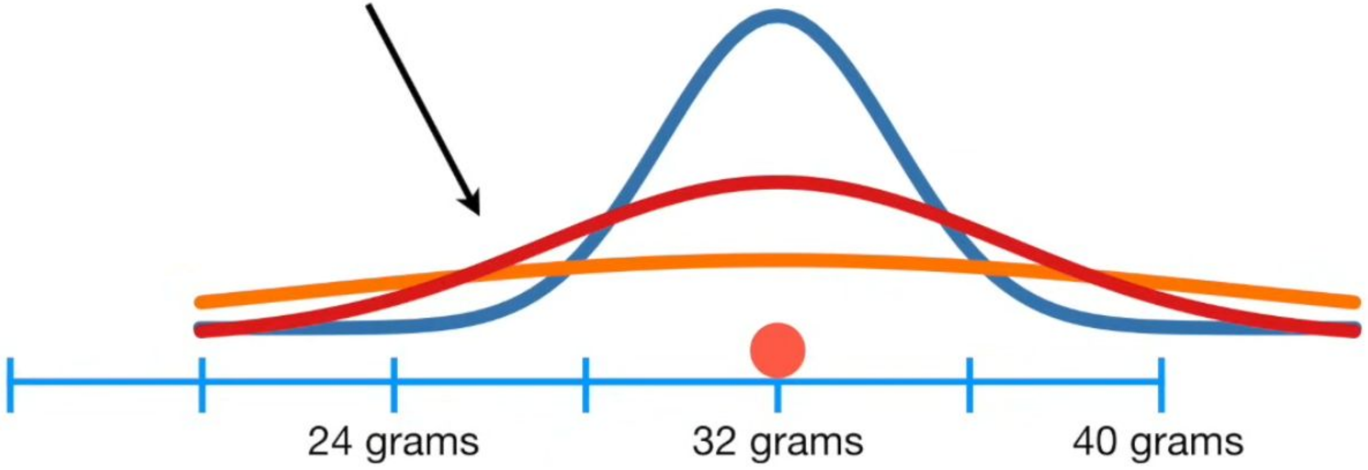
We can identify the peak in the likelihood graph by determining where the slope of the curve = **0**.



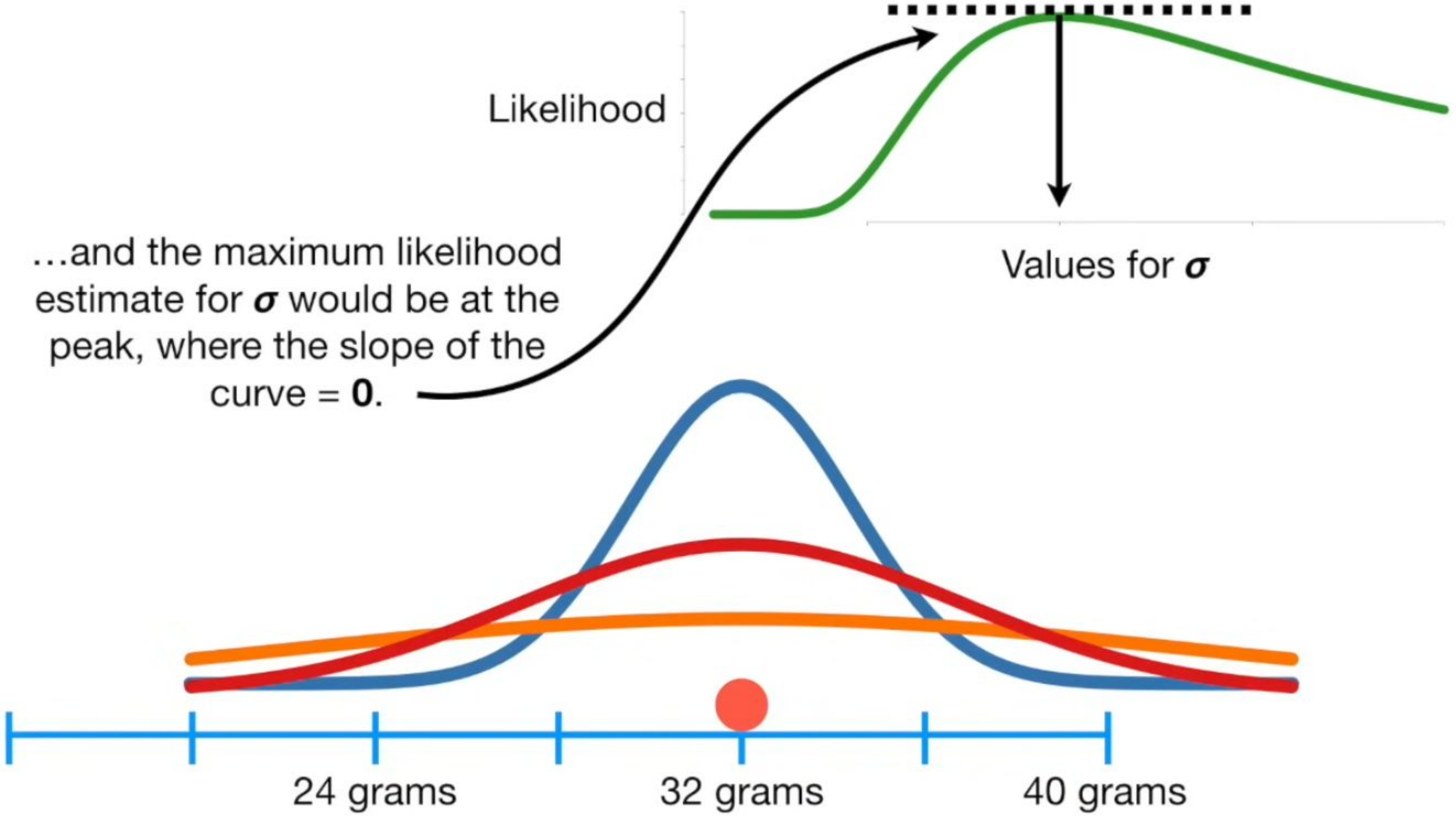
Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\boxed{L(\sigma} \mid x = 32, \mu = 32) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(32-32)^2/2\sigma^2}$$

…and we can plug in different values for ***σ*** to find the one that gives the maximum likelihood.



Maximum Likelihood Estimation for Parameter Estimation - **PART A**

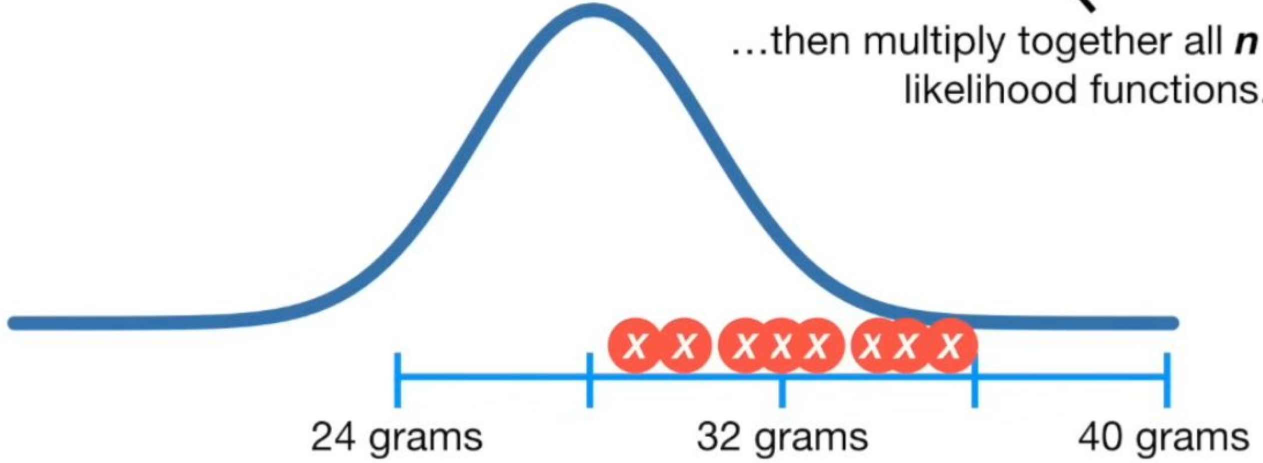

Likelihood
Values for σ
…and the maximum likelihood estimate for σ would be at the peak, where the slope of the curve = 0.
24 grams
32 grams
40 grams

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$L(\mu, \sigma | x_1, x_2, \ldots, x_n) = L(\mu, \sigma | x_1) \times \ldots \times L(\mu, \sigma | x_n)$$

$$\boxed{= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \ldots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2}}$$

…then multiply together all ***n*** individual likelihood functions.

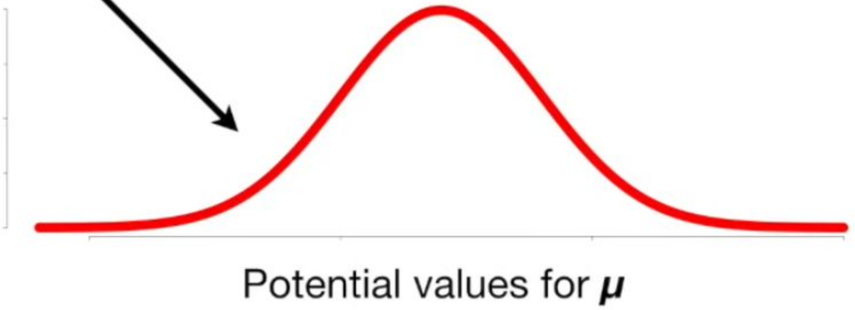


Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$L(\mu, \sigma | x_1, x_2, \dots, x_n) = L(\mu, \sigma | x_1) \times \dots \times L(\mu, \sigma | x_n)$$

$$\boxed{= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \dots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2}}$$

One derivative will be with respect ***μ***, when we treat ***σ*** like it's a constant…

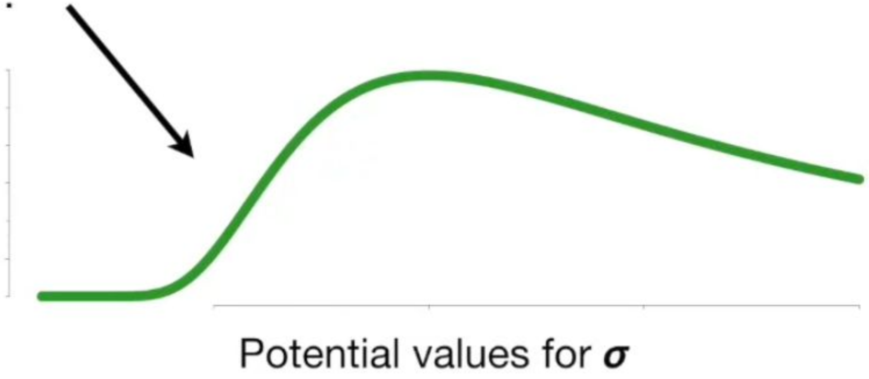


Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$L(\mu, \sigma | x_1, x_2, \ldots, x_n) = L(\mu, \sigma | x_1) \times \ldots \times L(\mu, \sigma | x_n)$$

$$\boxed{= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \ldots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2}}$$

The other derivative will be with respect ***σ***, when we treat ***μ*** like it's a constant…



Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$L(\mu, \sigma | x_1, x_2, \ldots, x_n) = L(\mu, \sigma | x_1) \times \ldots \times L(\mu, \sigma | x_n)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \ldots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2}$$

$$\boxed{\begin{aligned} &\ln\left[L(\mu, \sigma | x_1, \ \ldots, x_n)\right] \\ &\qquad = \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_1-\mu)^2/2\sigma^2} \times \ldots \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_n-\mu)^2/2\sigma^2} \right) \end{aligned}}$$

We do this because it makes taking the derivative way, way easier…

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\ln\left[L(\mu, \sigma | x_1, \ \ldots, x_n)\right]$$

$$= -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2} - \ldots - \frac{(x_n-\mu)^2}{2\sigma^2}$$

$$\frac{\partial}{\partial\mu}\ln\left[L(\mu, \sigma | x_1, \ \ldots, x_n)\right]$$

$$\longleftrightarrow$$

Thus, this is the derivative of the log-likelihood function with respect to $\boldsymbol{\mu}$.

$$\boxed{= \frac{1}{\sigma^2}\left[(x_1 + \ldots + x_n) - n\mu\right]}$$

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\ln\left[L(\mu, \sigma | x_1, \ \ldots, x_n)\right]$$

$$= -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{(x_1-\mu)^2}{2\sigma^2} - \ldots - \frac{(x_n-\mu)^2}{2\sigma^2}$$

$$\frac{\partial}{\partial\sigma}\ln\left[L(\mu, \sigma | x_1, \ \ldots, x_n)\right]$$

…and add up the remaining terms.

$$= 0 - \frac{n}{\sigma} + \boxed{\frac{(x_1-\mu)^2}{\sigma^3} + \quad + \frac{(x_n-\mu)^2}{\sigma^3}}$$

$$= -\frac{n}{\sigma} + \frac{1}{\sigma^3}\left[(x_1-\mu)^2 + \ldots + (x_n-\mu)^2\right]$$

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\frac{\partial}{\partial \mu} \ln\left[L(\mu, \sigma \mid x_1, \ \ldots, x_n)\right] = \frac{1}{\sigma^2}\left[(x_1 + \ldots + x_n) - n\mu\right]$$

$$\frac{\partial}{\partial \sigma} \ln\left[L(\mu, \sigma \mid x_1, \ \ldots, x_n)\right] = -\frac{n}{\sigma} + \frac{1}{\sigma^3}\left[(x_1 - \mu)^2 + \ldots + (x_n - \mu)^2\right]$$

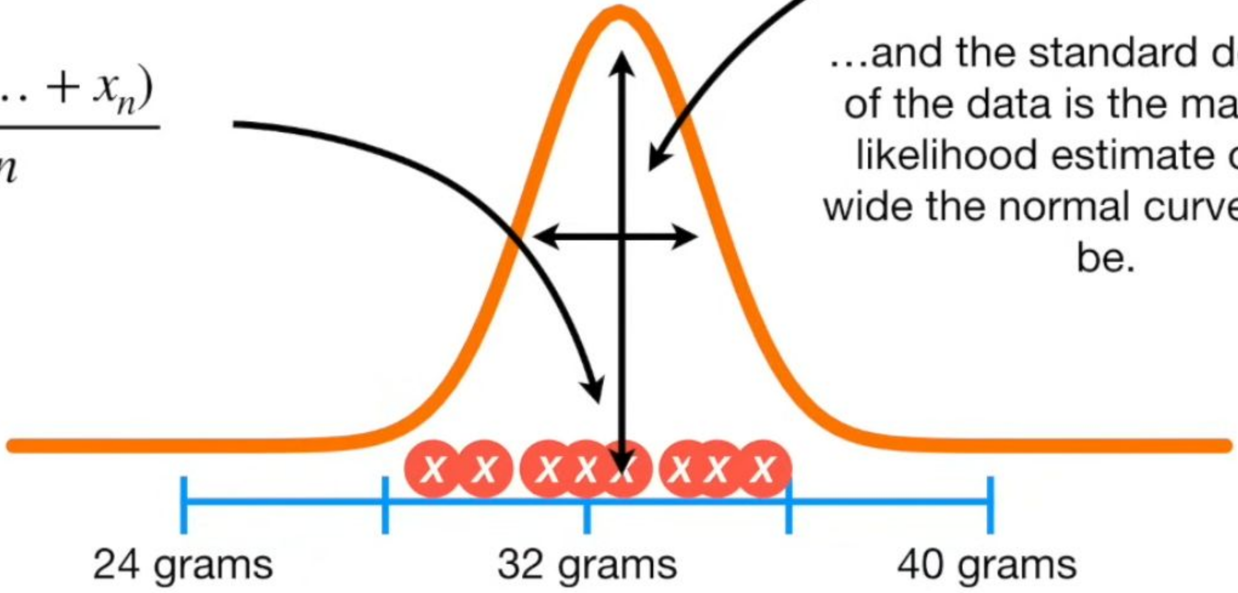
At long last, here are the two derivatives!!!

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

$$\sigma = \sqrt{\frac{(x_1 - \mu)^2 + \ldots + (x_n - \mu)^2}{n}}$$

$$\mu = \frac{(x_1 + \ldots + x_n)}{n}$$

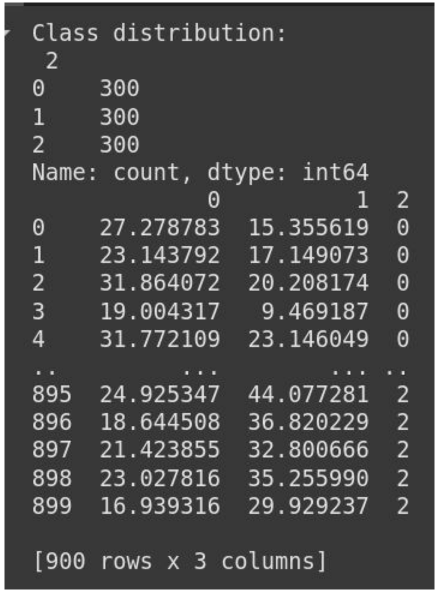
…and the standard deviation of the data is the maximum likelihood estimate of how wide the normal curve should be.



Maximum Likelihood Estimation for Parameter Estimation - **PART A**

1. Implement a function which takes as input an array of samples and returns the mean and the covariance matrix of those samples.
2. Print the mean and the covariance matrix for each one of the 3 classes, using the function from Question 1.

Example dataset



```
Class distribution:
 2
0    300
1    300
2    300
Name: count, dtype: int64
             0          1  2
0    27.278783  15.355619  0
1    23.143792  17.149073  0
2    31.864072  20.208174  0
3    19.004317   9.469187  0
4    31.772109  23.146049  0
..         ...        ... ..
895  24.925347  44.077281  2
896  18.644508  36.820229  2
897  21.423855  32.800666  2
898  23.027816  35.255990  2
899  16.939316  29.929237  2

[900 rows x 3 columns]
```

For each class

- When $\boldsymbol{x}$ is a 2D vector
- Mean: $\boldsymbol{\mu}$ (vector 2x1)
- Covariance: $\boldsymbol{\Sigma}$ (matrix 2x2)

Maximum Likelihood Estimation for Parameter Estimation - **PART A**

3. Considering the function of Question 1, plot each class distribution in one single 3D plot.

$$f(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k * det(\boldsymbol{\Sigma})}} * e^{-\frac{1}{2} * ((\boldsymbol{x}-\boldsymbol{\mu})^T \cdot inv(\boldsymbol{\Sigma}) \cdot (\boldsymbol{x}-\boldsymbol{\mu}))}$$

k = dim, **x** = input vector, **μ** = dist. means, **Σ** = covariance matrix

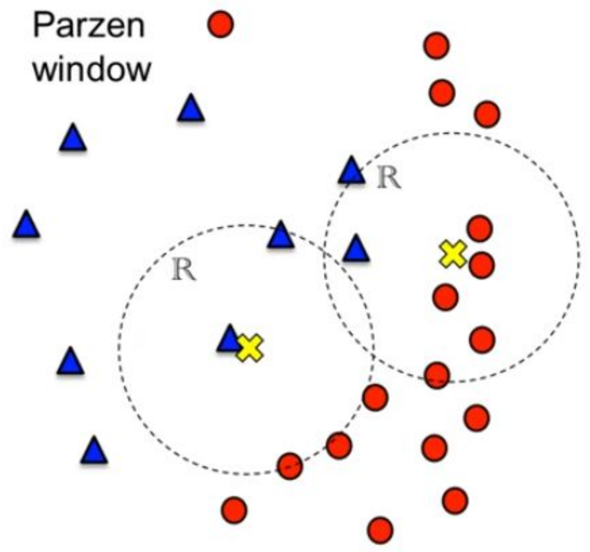
where
k=2
x=[,] -> 2 x 1
μ=[,] -> 2 x 1
Σ=[[,][,]] -> 2 x 2

Parzen Windows - **PART B**

Dataset in two dimensions, 2 classes, 2 testing points (yellow).

Looking for a region of radius R and using a region with fixed radius for all the testing instances.

The right testing point gets a lot more training examples to base its decision.



Parzen Windows - **PART B**

1. Implement the window function $\phi(u)$, when window is a hypercube.

$$\varphi(u) = \begin{cases} 1 & |u_j| \leq \frac{1}{2} \quad j = 1, \dots, d \\ 0 & \text{otherwise} \end{cases}$$

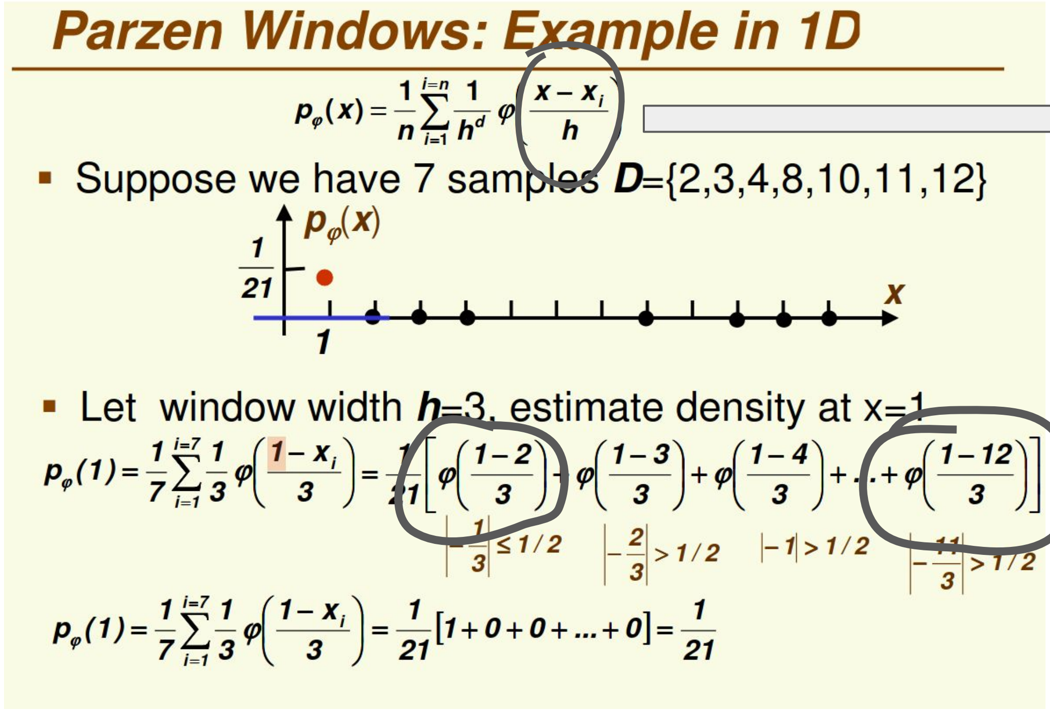
2. Implement the window function $\phi(u)$, when window is a Gaussian kernel.

$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$

Used to determine the weight given to each sample point in the density estimation process. It is a function that assigns higher weights to points closer to the estimation point and lower weights to points farther away.

Parzen Windows - **PART B**

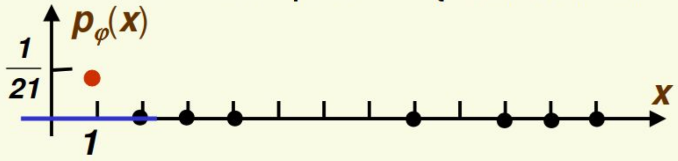
3. Implement a function which takes as input a single point x_i, the center of the window as a single point c, the width h of the window and the *kernel* type (*'hypercube'* or *'Gaussian'*) of the window. The function calls one of the above implemented functions (hypercube window or Gaussian window), with the appropriate input, and returns the result.



Parzen Windows: Example in 1D

$$p_\varphi(x) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{h^d}\varphi\left(\frac{x-x_i}{h}\right) \xrightarrow{} u$$

- Suppose we have 7 samples D={2,3,4,8,10,11,12}



- Let window width h=3, estimate density at x=1

$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1-x_i}{3}\right) = \frac{1}{21}\left[\varphi\left(\frac{1-2}{3}\right) + \varphi\left(\frac{1-3}{3}\right) + \varphi\left(\frac{1-4}{3}\right) + \dots + \varphi\left(\frac{1-12}{3}\right)\right]$$

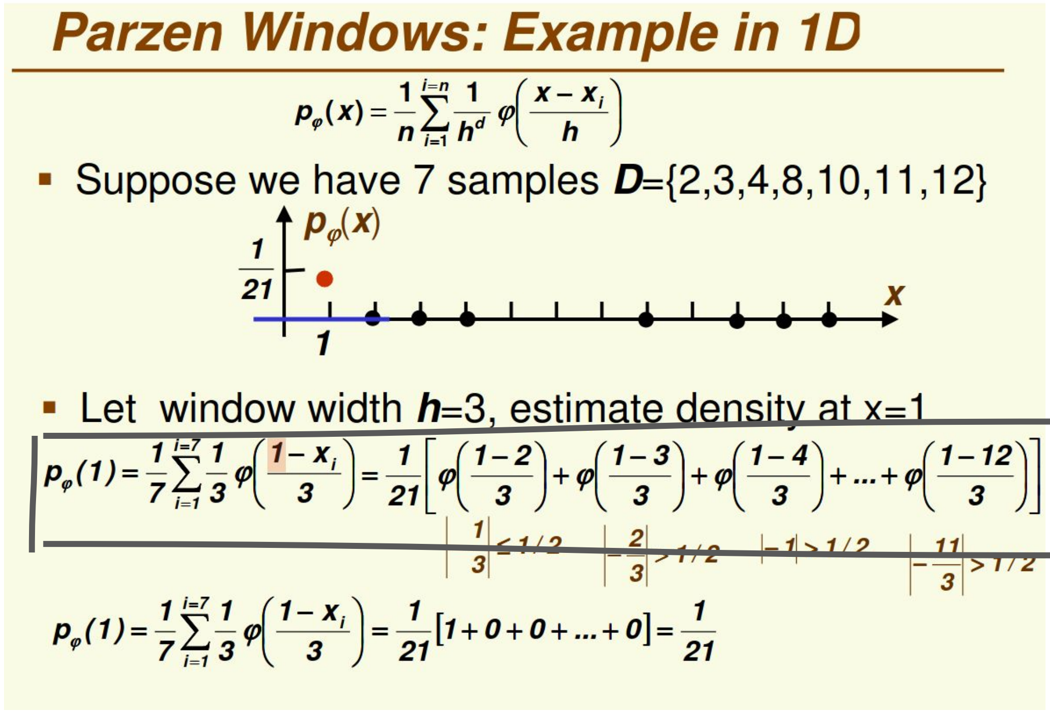
$$\left|-\frac{1}{3}\right| \le 1/2 \qquad \left|-\frac{2}{3}\right| > 1/2 \qquad |-1| > 1/2 \qquad \left|-\frac{11}{3}\right| > 1/2$$

$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1-x_i}{3}\right) = \frac{1}{21}[1+0+0+\dots+0] = \frac{1}{21}$$

x in the example represents c in question 3.

Parzen Windows - **PART B**

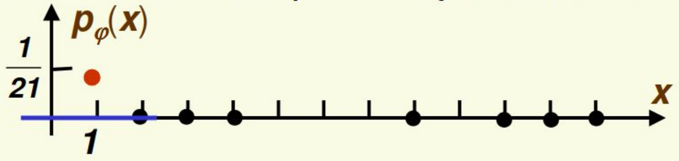
4. In this question, you are asked to develop the Density Function of Parzen Window. Implement a function which takes as input an array of 1-d points *data*, a single point x which represents the center, the width h of the window and the kernel type of the window. The function should return the likelihood of the center x, given the other inputs.



Parzen Windows: Example in 1D

$$p_\varphi(x) = \frac{1}{n}\sum_{i=1}^{i=n} \frac{1}{h^d}\varphi\left(\frac{x - x_i}{h}\right)$$

- Suppose we have 7 samples D={2,3,4,8,10,11,12}



- Let window width h=3, estimate density at x=1

$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1 - x_i}{3}\right) = \frac{1}{21}\left[\varphi\left(\frac{1-2}{3}\right) + \varphi\left(\frac{1-3}{3}\right) + \varphi\left(\frac{1-4}{3}\right) + \dots + \varphi\left(\frac{1-12}{3}\right)\right]$$

$$\left|\frac{1}{3}\right| < 1/2 \qquad \left|-\frac{2}{3}\right| > 1/2 \qquad \left|-1\right| > 1/2 \qquad \left|-\frac{11}{3}\right| > 1/2$$

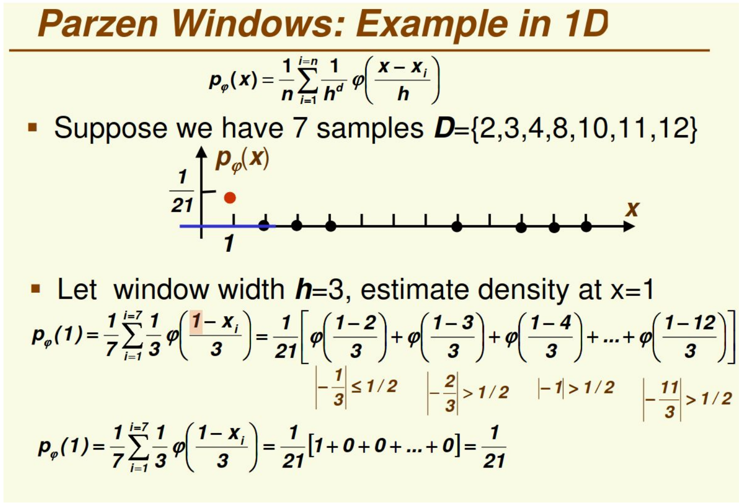
$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1 - x_i}{3}\right) = \frac{1}{21}\left[1 + 0 + 0 + \dots + 0\right] = \frac{1}{21}$$

Parzen Windows - **PART B**

5. What's the best value for the width of the window h? To find this, assume that the dataset you have comes from the normal distribution N(1, 1) (this is a univariate normal distribution). Find the most suitable value for h based on that knowledge.

   (a) Create a histogram of the data to convince yourselves that they come from the aforementioned distribution.

   (b) For every h in the range [0.05, 5] with step = 0.1 calculate 1) the predicted likelihood for every point in the data, 2) their true likelihood (you can use the function norm.pdf(data, loc=1, scale=1)), and 3) the Mean Square Error of the two likelihoods (predicted and true). Repeat this process for both kernels (hypercube and Gaussian). What's the most suitable value for h for each kernel? Print your answer and create a plot which shows the values of h on the x-axis and their MSE on the y-axis (for both kernels).



Parzen Windows: Example in 1D

$$p_\varphi(x) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{h^d}\varphi\left(\frac{x-x_i}{h}\right)$$

- Suppose we have 7 samples D={2,3,4,8,10,11,12}

$p_\varphi(x)$
$\frac{1}{21}$
1
x

- Let window width h=3, estimate density at x=1

$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1-x_i}{3}\right) = \frac{1}{21}\left[\varphi\left(\frac{1-2}{3}\right)+\varphi\left(\frac{1-3}{3}\right)+\varphi\left(\frac{1-4}{3}\right)+\dots+\varphi\left(\frac{1-12}{3}\right)\right]$$

$$\left|-\frac{1}{3}\right| \le 1/2 \quad \left|-\frac{2}{3}\right| > 1/2 \quad |-1| > 1/2 \quad \left|-\frac{11}{3}\right| > 1/2$$

$$p_\varphi(1) = \frac{1}{7}\sum_{i=1}^{i=7}\frac{1}{3}\varphi\left(\frac{1-x_i}{3}\right) = \frac{1}{21}[1+0+0+\dots+0] = \frac{1}{21}$$

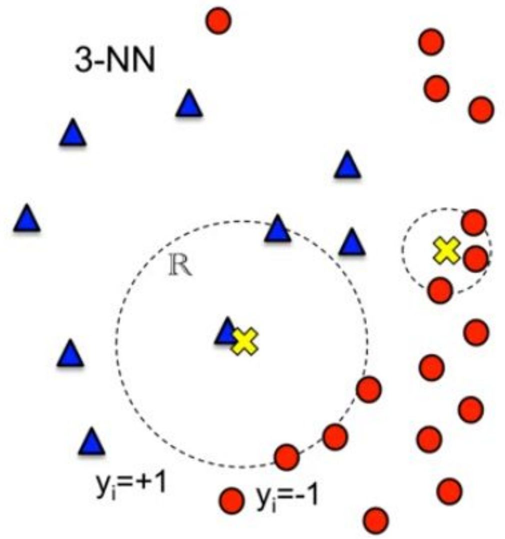
Not only for x = 1, but for all the data points.

K-Nearest Neighbors (KNN) Classification - **PART C**

Dataset in two dimensions, 2 classes, 2 testing points (yellow).

Suppose a 3 nearest neighbors algorithm - it will base the prediction on the region R which has exactly 3 training examples.

Note the difference in the regions for the 2 testing points - always looks for a fixed number of closest neighbors.



K-Nearest Neighbors (KNN) Classification - **PART C**

1. Implement a function which has as input a 2D point x and a NumPy array *train_data* of 2D points, and it computes the Euclidean distances of that point x to all points in the given array. The function should return that NumPy array of the Euclidean distances.

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

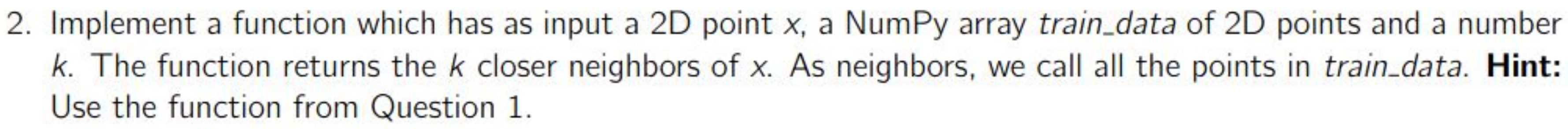
x = [,] -> (2,)

train_data = [[,] , [,] ,[,] ,[,] ,[,] ,[,] ,...[,]]->(N,2)

resulted distances -> (N,)

=

K-Nearest Neighbors (KNN) Classification - **PART C**


2. Implement a function which has as input a 2D point *x*, a NumPy array *train_data* of 2D points and a number *k*. The function returns the *k* closer neighbors of *x*. As neighbors, we call all the points in *train_data*. **Hint:** Use the function from Question 1.

Simple Example

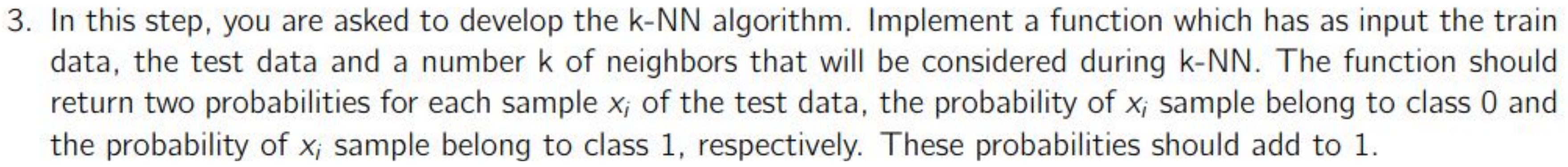
```
distances = [3, 1, 2]
indices = np.argsort(distances)

array([1, 2, 0])
```

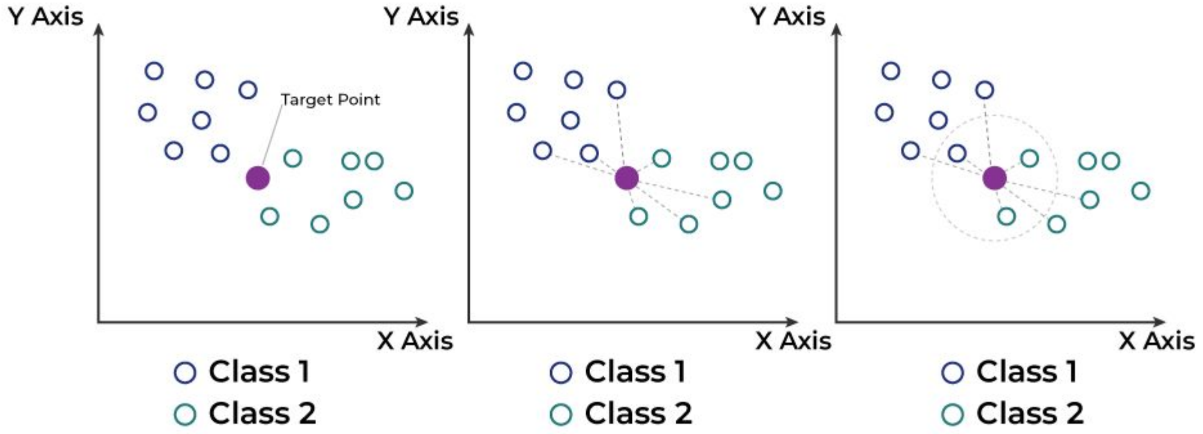
In the assignment

```
np.argsort(distances)[:k]
```

K-Nearest Neighbors (KNN) Classification - **PART C**

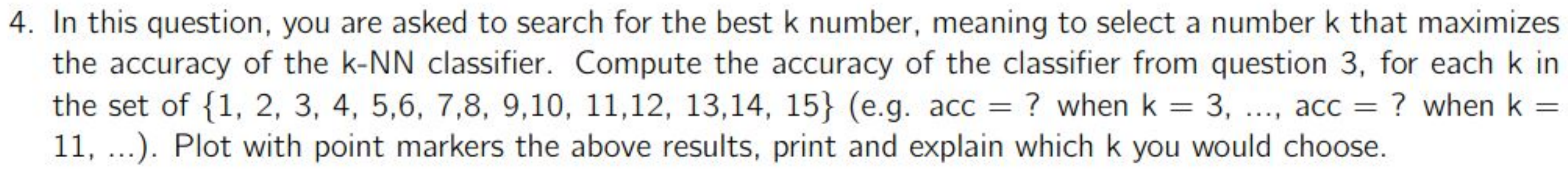


3. In this step, you are asked to develop the k-NN algorithm. Implement a function which has as input the train data, the test data and a number k of neighbors that will be considered during k-NN. The function should return two probabilities for each sample x_i of the test data, the probability of x_i sample belong to class 0 and the probability of x_i sample belong to class 1, respectively. These probabilities should add to 1.



Purple point -> test point we want to classify
For each test point, we find the nearest neighbors from the training samples!
Find their labels!
Two probabilities for each test point -> slide 24, non parametric techniques

K-Nearest Neighbors (KNN) Classification - **PART C**



4. In this question, you are asked to search for the best k number, meaning to select a number k that maximizes the accuracy of the k-NN classifier. Compute the accuracy of the classifier from question 3, for each k in the set of {1, 2, 3, 4, 5,6, 7,8, 9,10, 11,12, 13,14, 15} (e.g. acc = ? when k = 3, ..., acc = ? when k = 11, ...). Plot with point markers the above results, print and explain which k you would choose.

Find the probabilities for each test sample!

What is the prediction of a test sample?

Do we have the true labels of the test samples?

Compute accuracy for each k!

Which k gives the best accuracy?

Thank you!