ENGR-E 511; ENGR-E 399

Machine Learning for Signal Processing

Module 05:

Bayesian Classification

Minje Kim

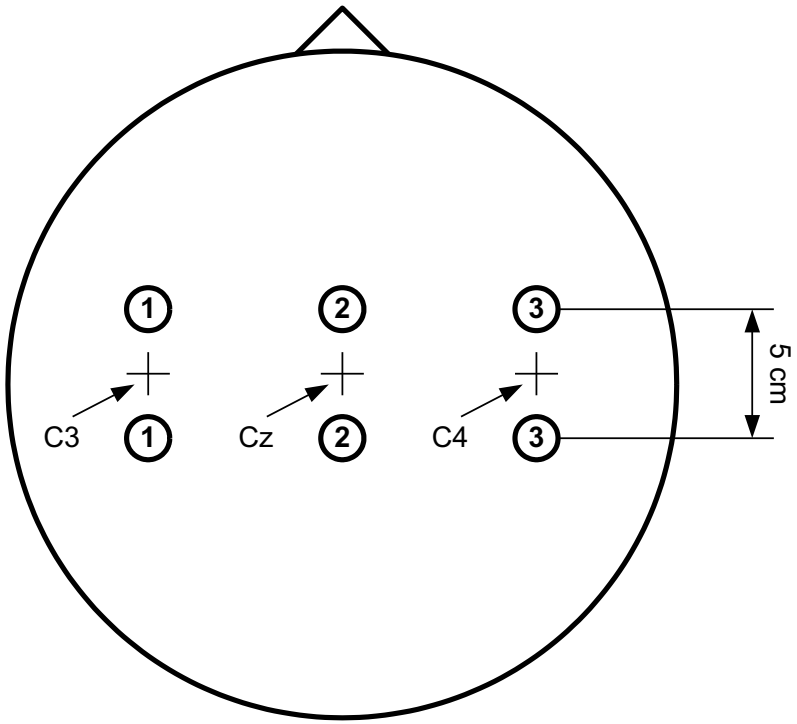
Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje



INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

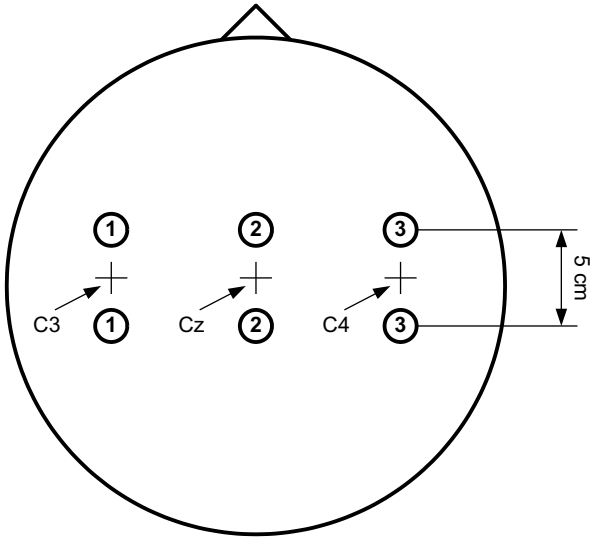
Brain Computer Interface

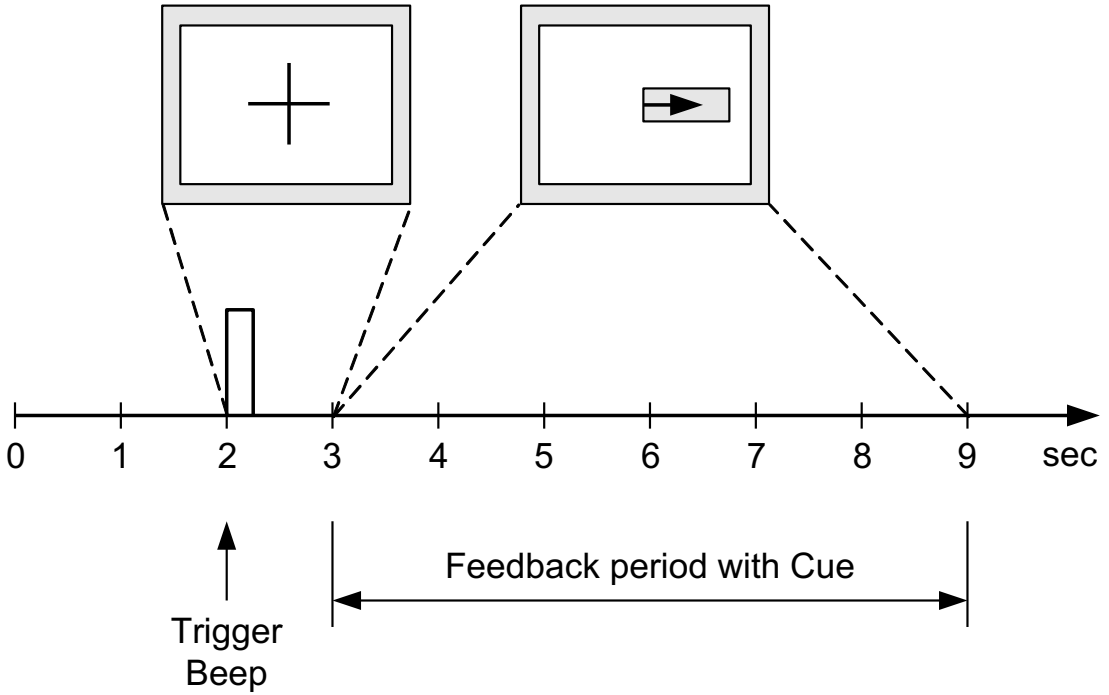
- https://youtu.be/7t84lGE5TXA

Brain Computer Interface

- Motor Imagery

- 112 recordings of 3-channel ElectroEncephaloGram (EEG)
- Each recording is 9-second long (128Hz)
  - After the triggers, from 3 to 9 sec the subject was asked to “imagine” the movement
  - Left or right
- If I keep only the last 6 sec after the trigger I have
  - 6 sec X 128 samples X 3 channels





Brain Computer Interface

- Motor Imagery

- There are different brain waves with different frequencies
  - Delta wave – (0.1 – 3 Hz)
  - Theta wave – (4 – 7 Hz)
  - Alpha wave – (8 – 13 Hz)
  - **Mu wave – (7.5 – 12.5 Hz)**
  - SMR wave – (12.5 – 15.5 Hz)
  - Beta wave – (16 – 31 Hz)
  - Gamma wave – (32 – 100 Hz)

- I care about Mu wave as it's known to be relevant to motor imagery
  - How to extract Mu wave?
  - STFT!

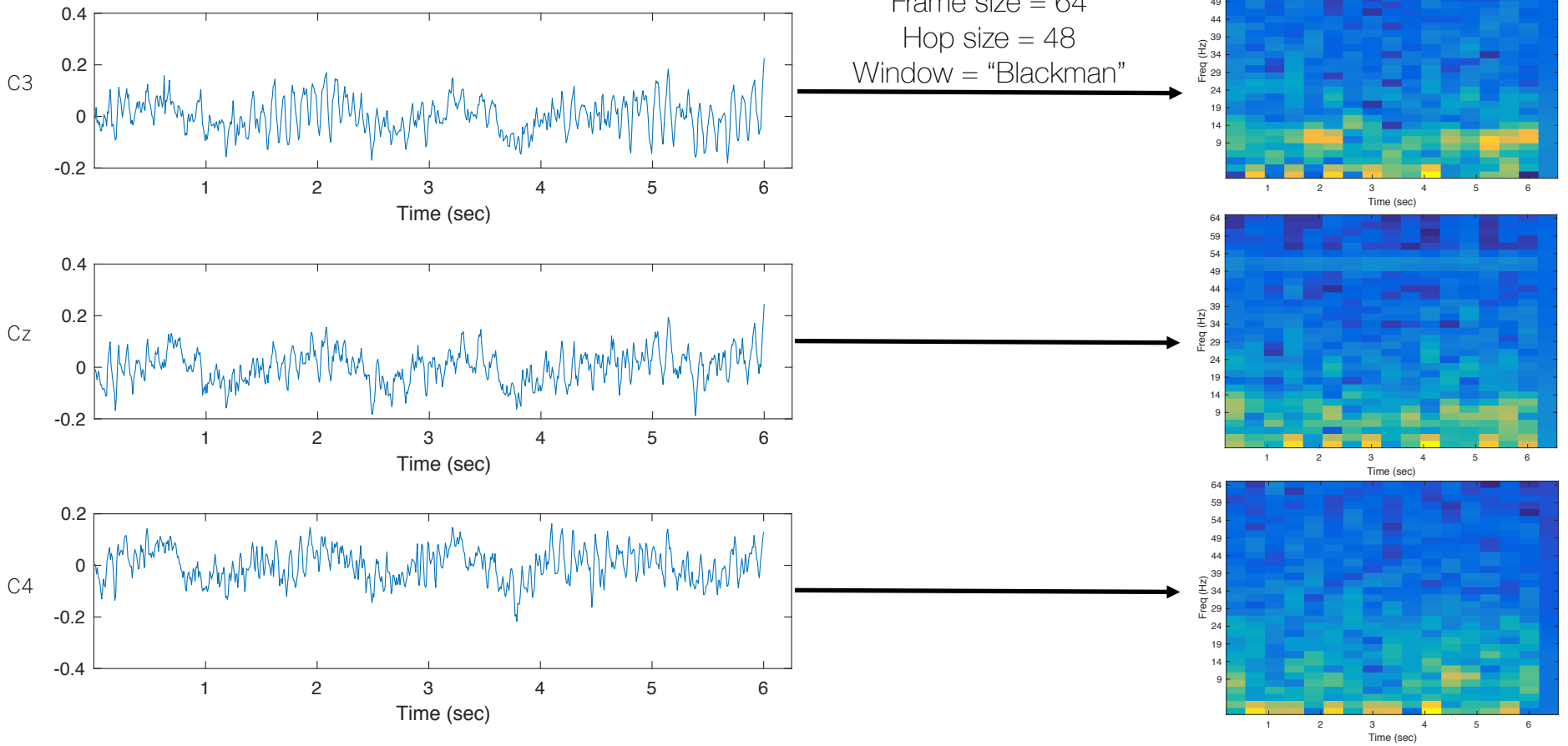
Meditation

http://healthcare.joins.com/img/2003/0530/gra_0530_02.jpg

Brain Computer Interface

- Motor Imagery

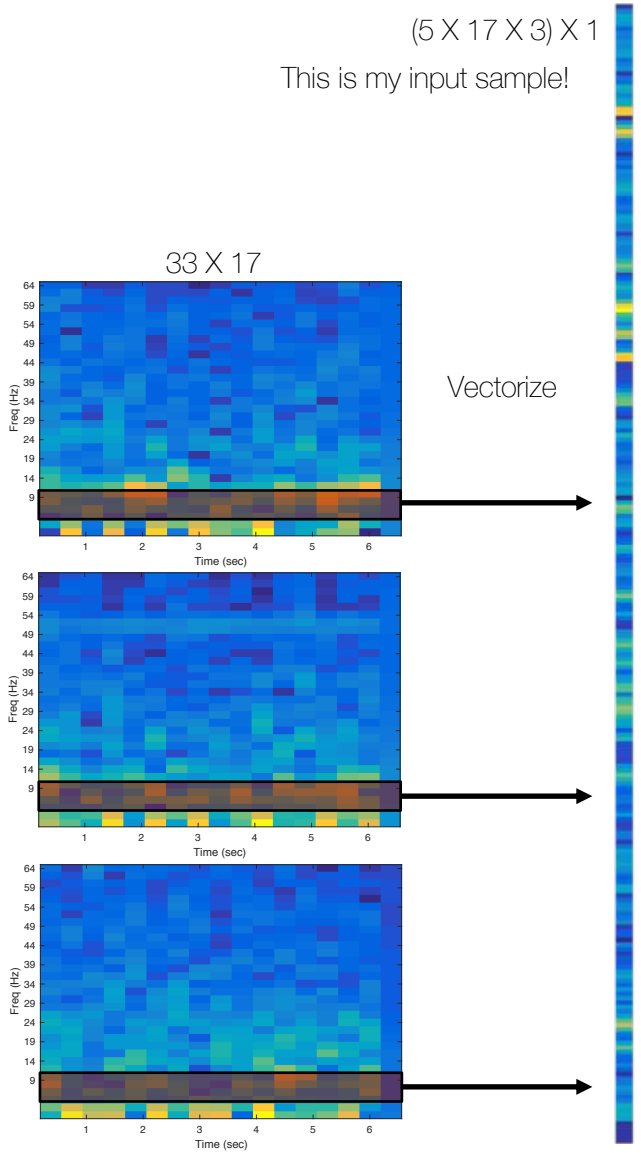
- STFT of the first trial



Brain Computer Interface

- Motor Imagery

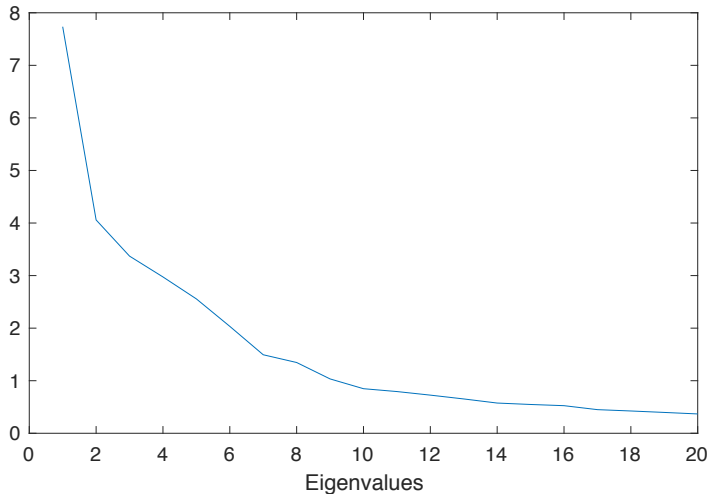
- Because I care about the Mu wave – (7.5 – 12.5 Hz)
  - It's known to be relevant to motor imagery
- I only keep from 3rd to 7th rows
  - 64Hz/32*3=6Hz
  - 64Hz/32*7=14Hz
- Since I've got 112 trials like that
  - My data matrix is 255 X 112
- Now I'm going to use them to train a classifier
- What's wrong with my data set?
  - Too many dimensions!
  - Too small data (might be expensive)

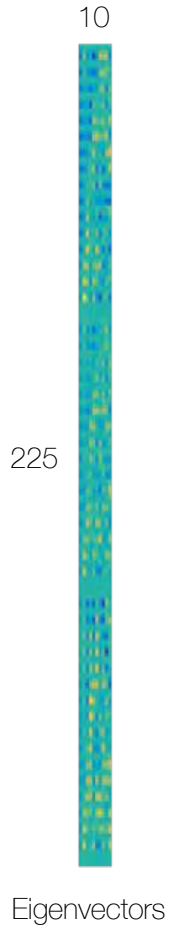


Brain Computer Interface

- Motor Imagery

- Now that we know how to reduce the dimension…
- I applied PCA on the data
- If you remember
  - Centering the data
  - Calculate the covariance matrix
  - Apply eigendecomposition
- I got a few eigenvectors and chose ten out of them
  - Why ten?
  - Based on the elbow of the eigenvalue graph

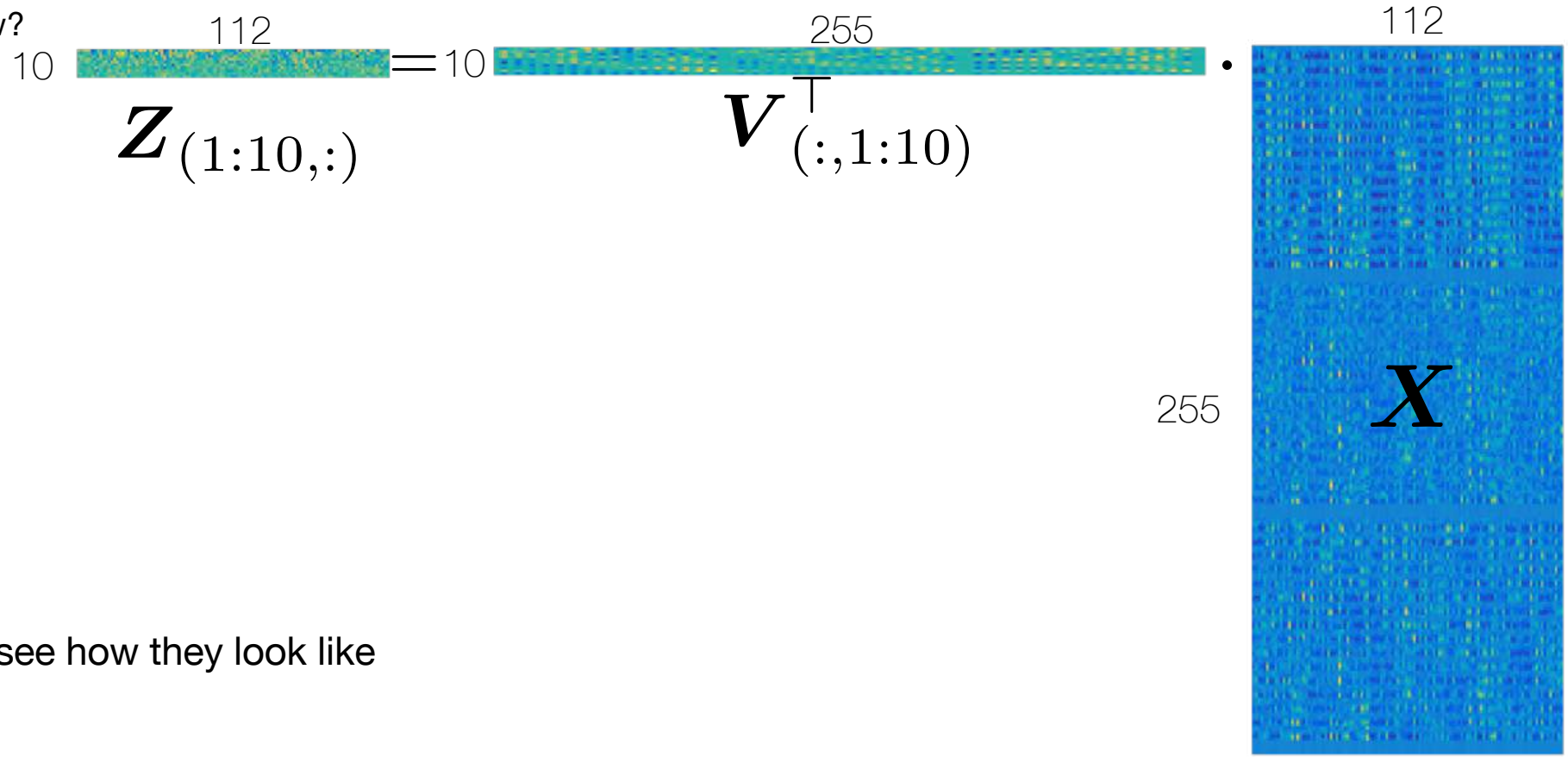




Brain Computer Interface

- Motor Imagery

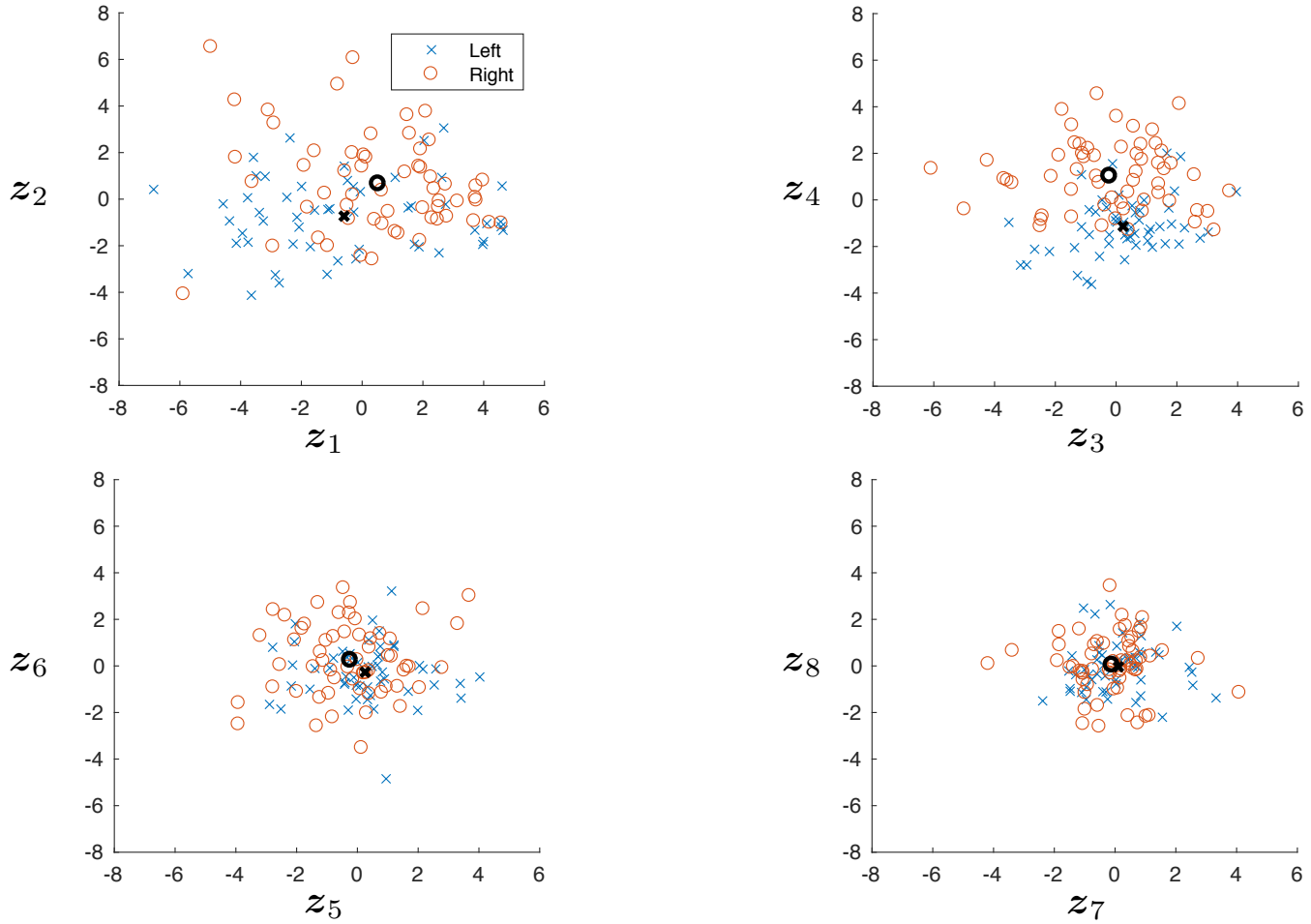
- Now I can reduce the dimensionality from 225 down to 10
  - How?



- Let's see how they look like

Brain Computer Interface

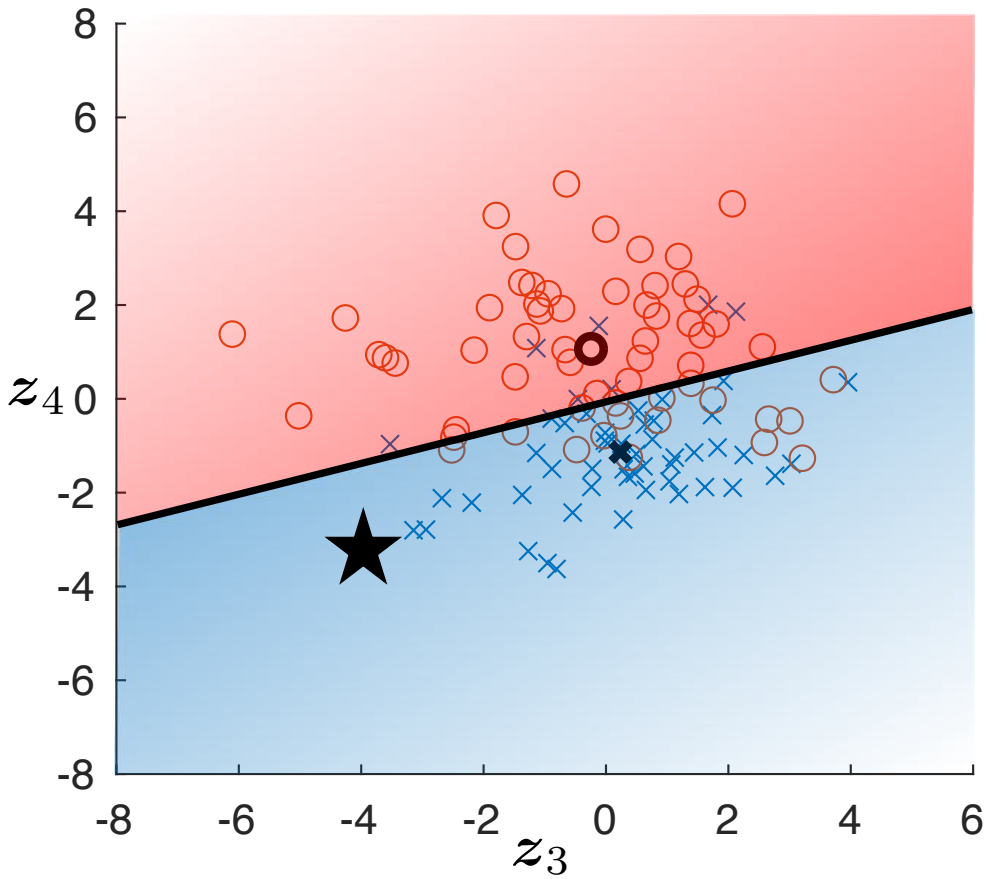
- Motor Imagery



Brain Computer Interface

- Motor Imagery

- For the convenience let's focus on z_3 and z_4
- I feel like drawing a line that can separate the two regions
  - If there's a test sample in the blue side, it might be from the "left" imagery
- How can I find the line?
  - There are many different ways!



Naïve Bayes Classifier

- Decision Surface

- I'll assume that each of the two classes follows a unique Gaussian dist.
- In other words, we need to *model* the training data
  - i.e. What kind of Gaussian should the underlying distribution be?
  - Any ideas?
- We do maximum likelihood estimation per class

$$\arg\max_{\boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L} \sum_{i \in \mathcal{C}_L} \log \mathcal{N}(\boldsymbol{z}_{(:,i)}; \boldsymbol{\mu}_L, \boldsymbol{\Sigma}_L) \qquad \arg\max_{\boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R} \sum_{i \in \mathcal{C}_R} \log \mathcal{N}(\boldsymbol{z}_{(:,i)}; \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R)$$

  - What would the estimation be?

$$\boldsymbol{\mu}_L^* = \frac{1}{|\mathcal{C}_L|} \sum_{i \in \mathcal{C}_L} \boldsymbol{z}_{(:,i)} \qquad \boldsymbol{\mu}_R^* = \frac{1}{|\mathcal{C}_R|} \sum_{i \in \mathcal{C}_R} \boldsymbol{z}_{(:,i)}$$

$$\boldsymbol{\Sigma}_L^* = \frac{1}{|\mathcal{C}_L|} \sum_{i \in \mathcal{C}_L} (\boldsymbol{z}_{(:,i)} - \boldsymbol{\mu}_L^*)(\boldsymbol{z}_{(:,i)} - \boldsymbol{\mu}_L^*)^\top \qquad \boldsymbol{\Sigma}_R^* = \frac{1}{|\mathcal{C}_R|} \sum_{i \in \mathcal{C}_R} (\boldsymbol{z}_{(:,i)} - \boldsymbol{\mu}_R^*)(\boldsymbol{z}_{(:,i)} - \boldsymbol{\mu}_R^*)^\top$$

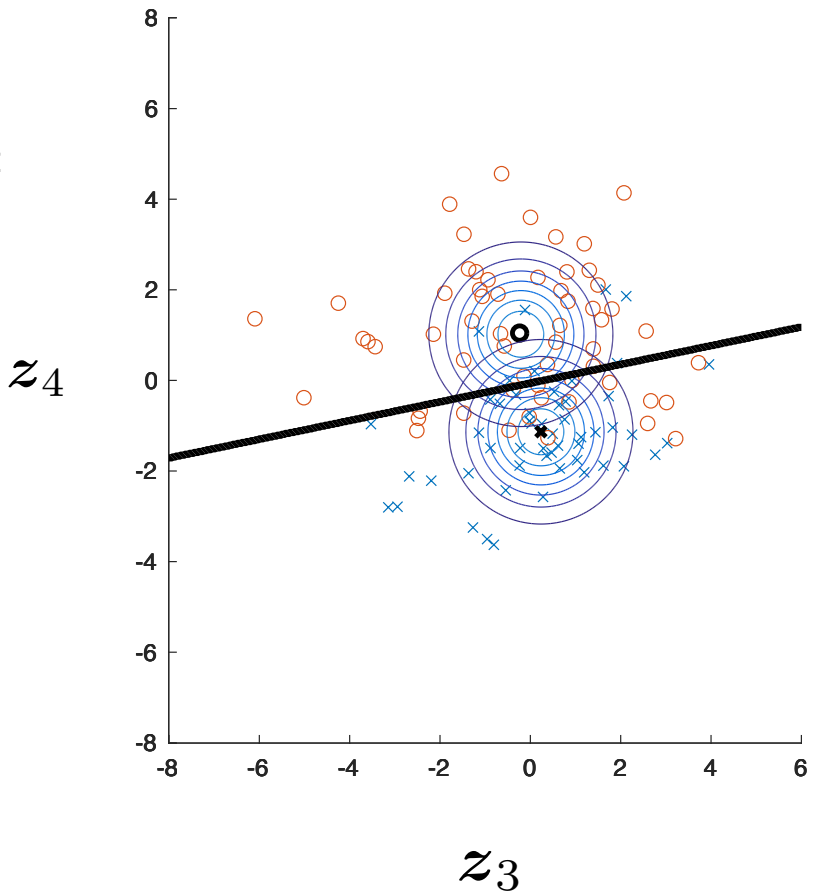
Naïve Bayes Classifier

- Decision Surface

- What would the hyperplane be?
- The Gaussians on the right are not optimal
  - Because I skipped the sample covariance matrix estimation part
  - Instead, I assumed that they are identity matrices

$$\Sigma_L^* = \Sigma_R^* = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

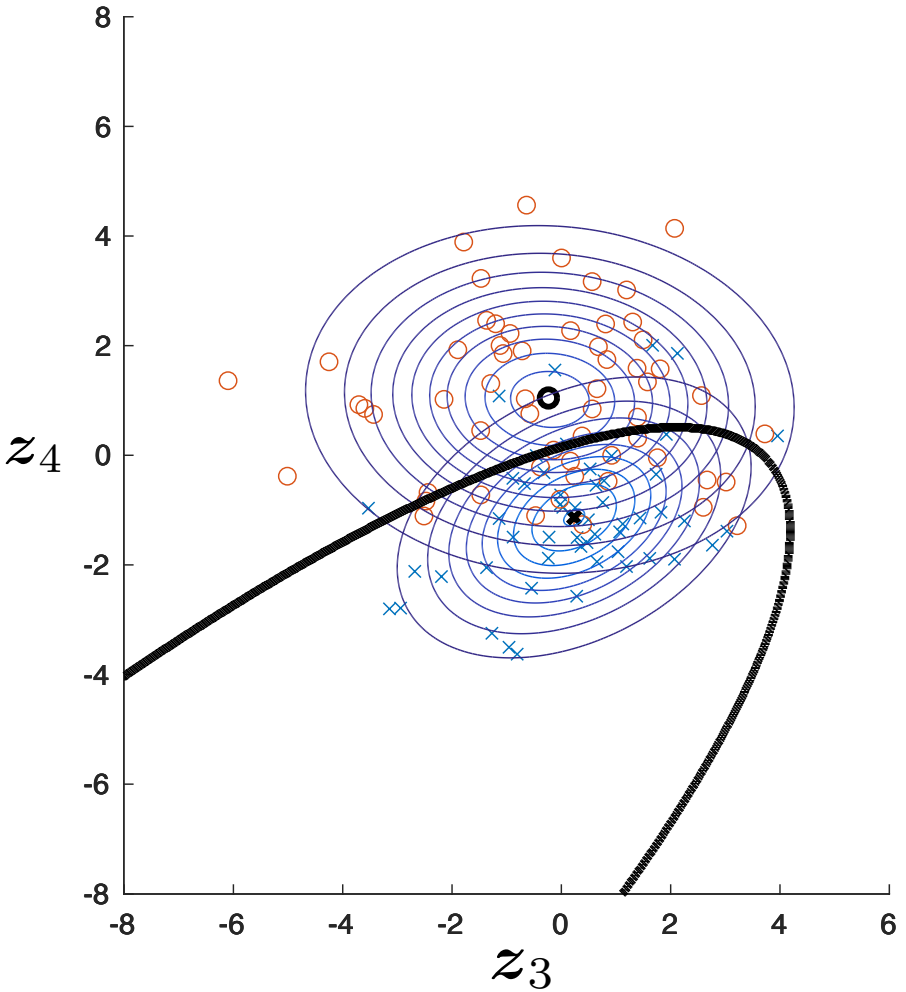
  - Why would I do that?
    - Sometimes sample covariance matrix can be noisy
    - Sometimes? When?
      - If we don't have enough data samples
      - While their dimension is high



Naïve Bayes Classifier

- Decision Surface

- Let's do it with the full covariance estimation
- Now we see that the decision surface is not linear
- Seems more flexible
  - More suitable for complicated classification problems
- Okay, modeling was easy,
  but how do we find the hyperplane
  - And how do we make the decision
    for the new unseen test samples?

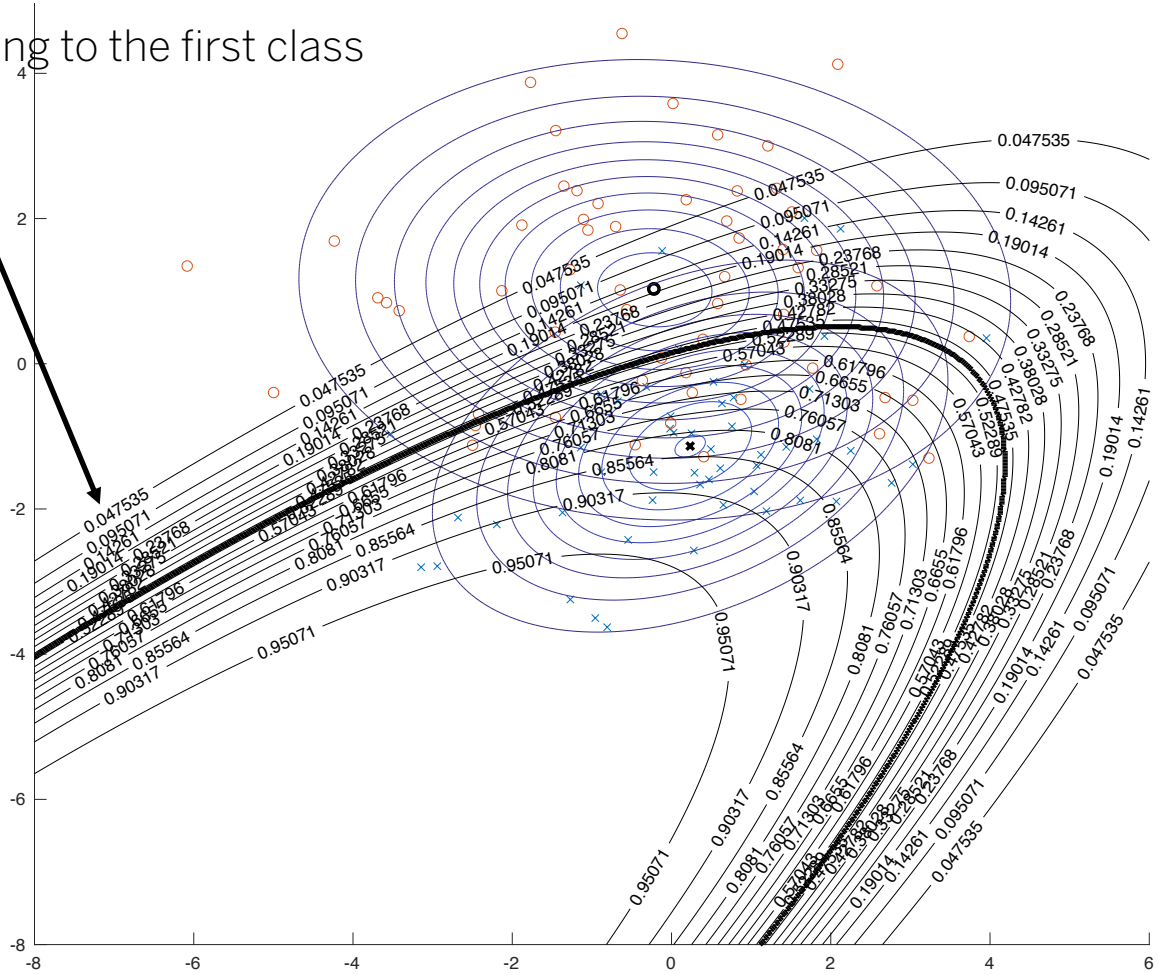


Naïve Bayes Classifier

- Decision Surface

Probability of $\boldsymbol{z}_{(:,i)}$ belonging to the first class

- Once again, how do we find these probabilities?



Naïve Bayes Classifier

- Decision Surface

- Now we need to use Bayes' rule
  - That's where the name is from
- We want to know the **posterior** probabilities of the classes given the data point

$$P(j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|j)P(j)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|j)P(j)}{\sum_j P(\boldsymbol{x}|j)P(j)}$$

- Or, for our EEG example

$$P(L|\boldsymbol{z}_{(:,i)}) = \frac{\mathcal{N}(\boldsymbol{z}_{(:,i)}; \boldsymbol{\mu}_L^*, \boldsymbol{\Sigma}_L^*)P(L)}{\mathcal{N}(\boldsymbol{z}_{(:,i)}; \boldsymbol{\mu}_L^*, \boldsymbol{\Sigma}_L^*)P(L) + \mathcal{N}(\boldsymbol{z}_{(:,i)}; \boldsymbol{\mu}_R^*, \boldsymbol{\Sigma}_R^*)P(R)}$$

Prior probability of choosing "left"

Prior probability of choosing "right"

$$P(j) = \frac{|\mathcal{C}_j|}{\sum_{j'=1}^{J} |\mathcal{C}_{j'}|}$$

Naïve Bayes Classifier

- Decision Surface

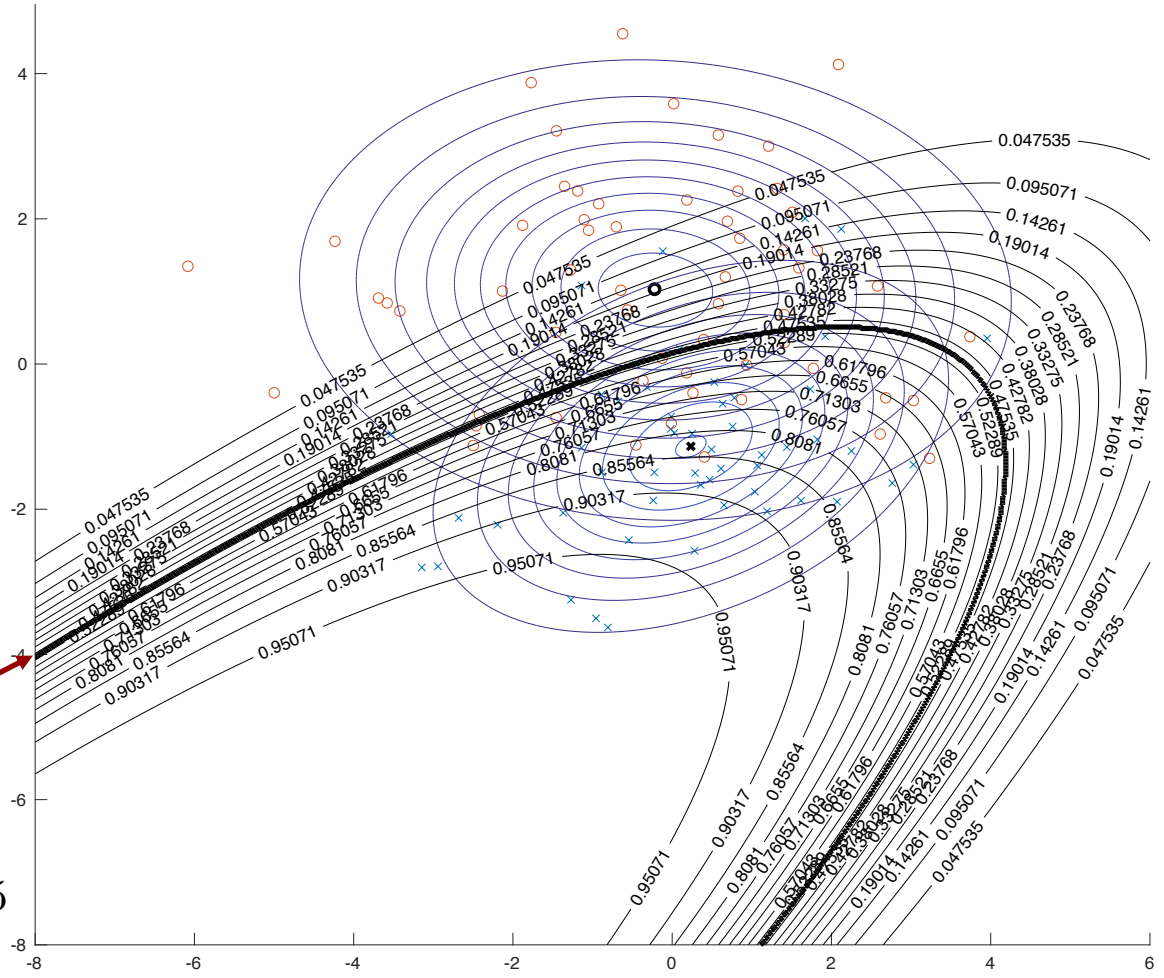
- Let's define the decision surface now

$$P(L|\boldsymbol{z}_{(:,i)}) = P(R|\boldsymbol{z}_{(:,i)})$$

  - Or, if you like logarithm

$$\log P(L|\boldsymbol{z}_{(:,i)}) = \log P(R|\boldsymbol{z}_{(:,i)})$$

$$P(L|z_{(3,i)}, z_{(4,i)}) = P(R|z_{(3,i)}, z_{(4,i)}) = 0.5$$



Naïve Bayes Classifier

- Decision Surface

- In general for the naïve Bayes, the decision boundary between the class *j* and *k* meets this condition:

$$\log P(j|\boldsymbol{x}) = \log P(k|\boldsymbol{x})$$

$$\log P(j)\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) = \log P(k)\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

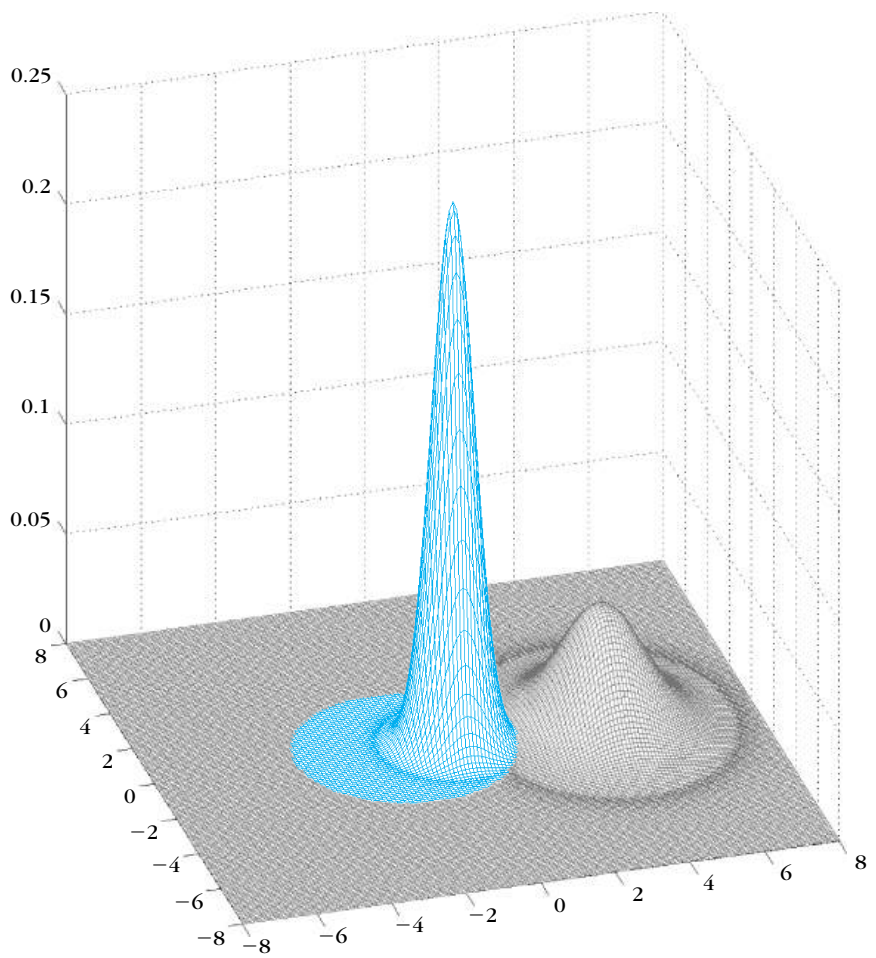
$$\log P(j) + (\boldsymbol{x}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) + const.$$
$$= \log P(k) + (\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + const.$$

$$\log P(j) + \cancel{\boldsymbol{x}^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{x}} + 2\boldsymbol{x}^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j)$$
$$= \log P(k) + \cancel{\boldsymbol{x}^\top \boldsymbol{\Sigma}_k^{-1}\boldsymbol{x}} + 2\boldsymbol{x}^\top \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k)$$

- When is the decision boundary linear?
  - When there's no quadratic terms in the equation
  - i.e. when the two covariance matrices are same

Naïve Bayes Classifier

- Decision Surface



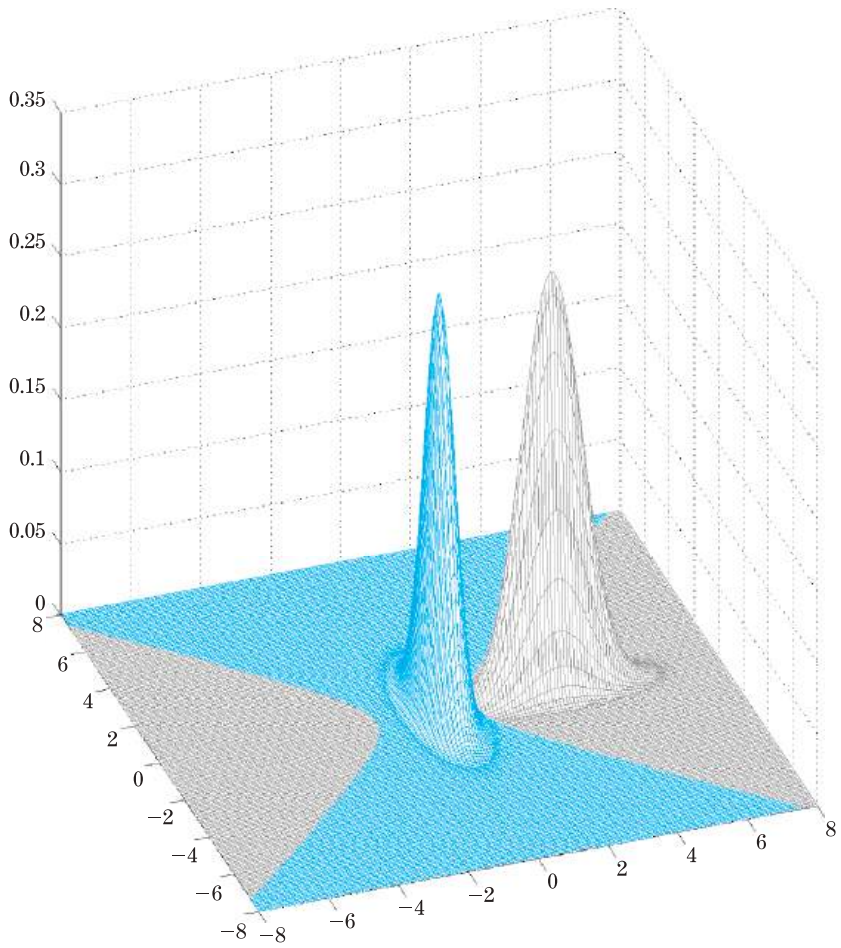
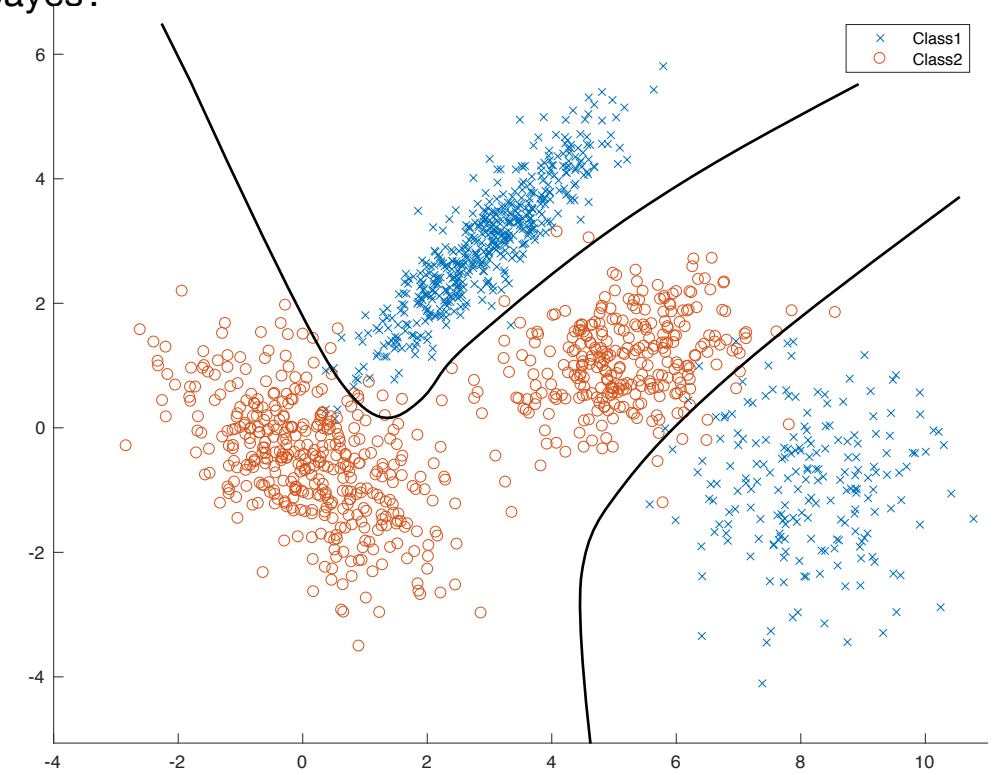


Figure 2.8 and 2.9

GMM Classifier

- Not for clustering, but for classification!

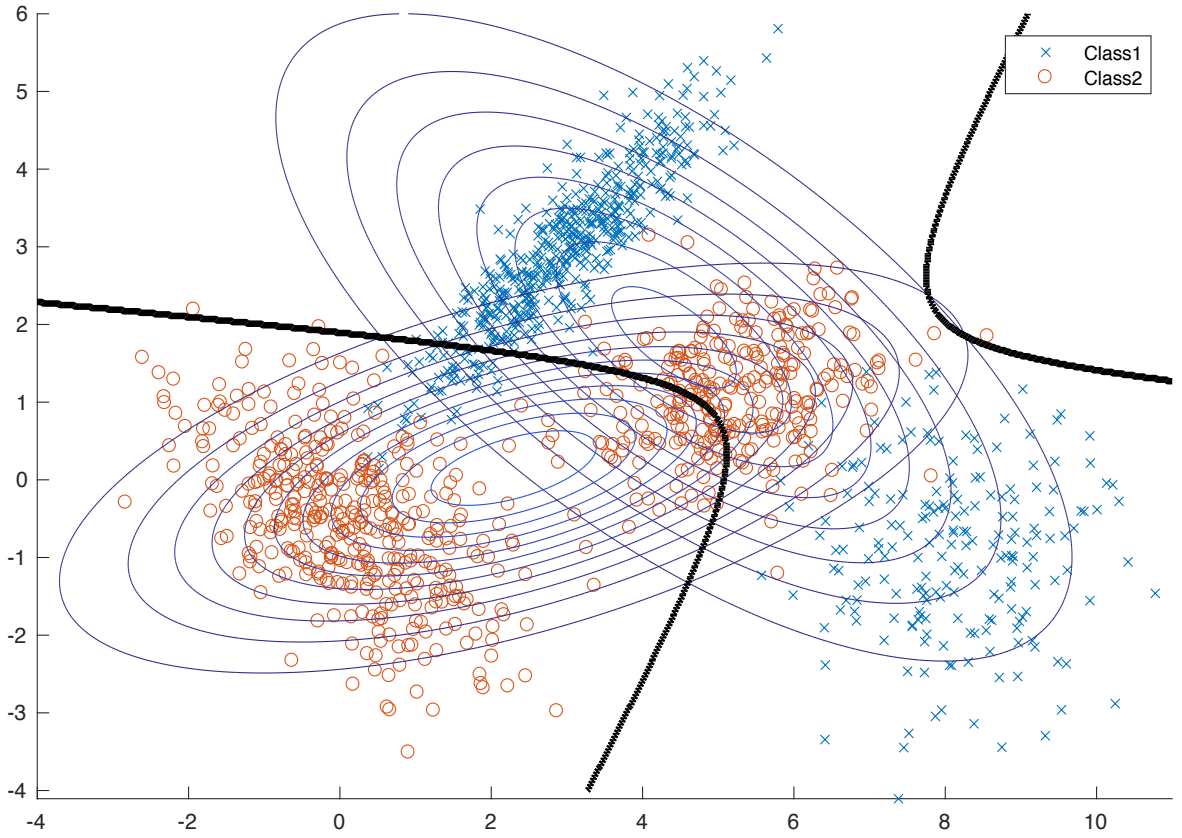
- What are we going to do with this?
  - I feel like drawing a nonlinear decision boundary
  - Can we do this with naïve Bayes?



GMM Classifier

- Not for clustering, but for classification!

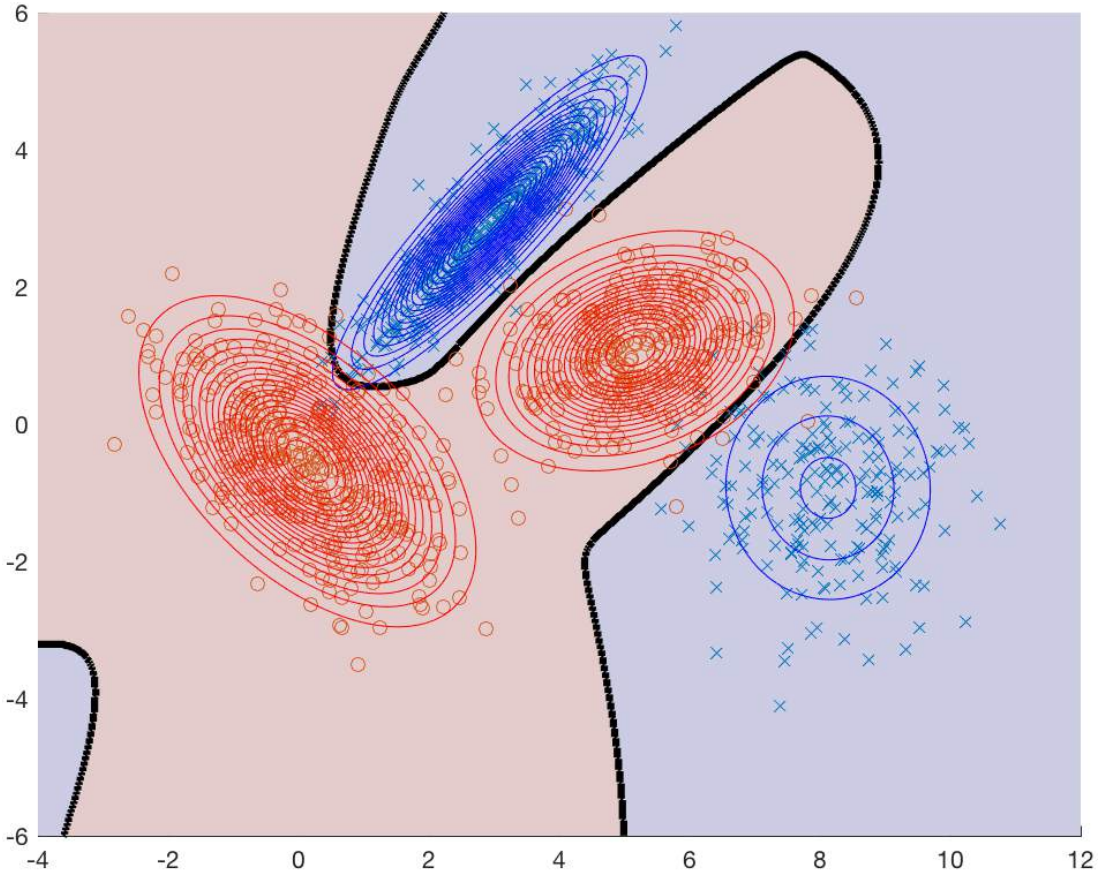
- Naïve Bayes results are not what we want



GMM Classifier

- Not for clustering, but for classification!

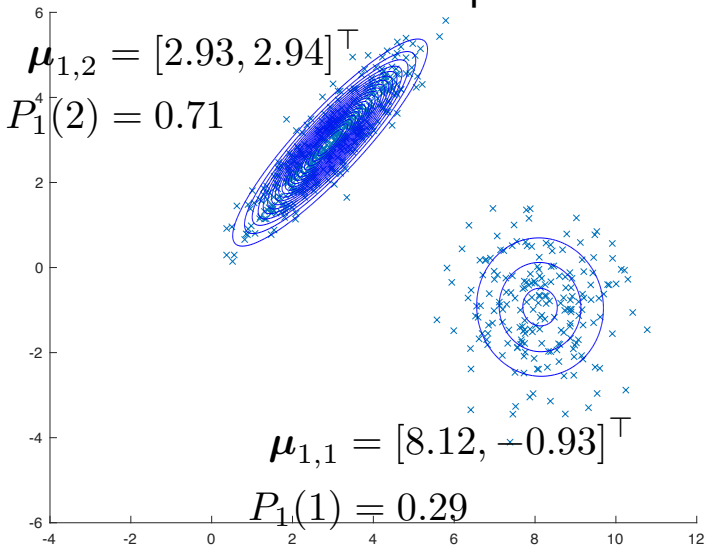
- GMM can define highly non-linear decision boundaries

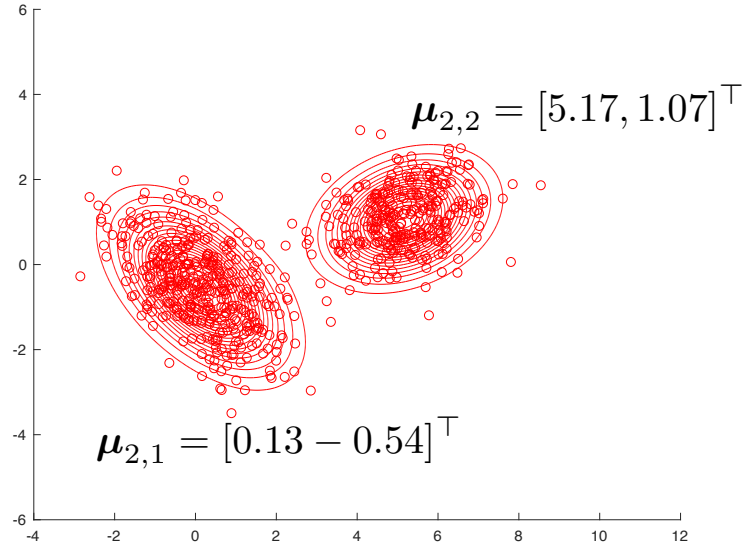


GMM Classifier

- Not for clustering, but for classification!

- Math?
- First we need to learn a GMM per class





- This will give me *J* sets of parameters
  - *J* is the number of classes; L_j is the number of Gaussians in the j-th class

$$j = 1: \quad \{\boldsymbol{\mu}_{1,1}, \boldsymbol{\Sigma}_{1,1}), P_1(1)\}, \{\boldsymbol{\mu}_{1,2}, \boldsymbol{\Sigma}_{1,2}), P_1(2)\}, \cdots, \{\boldsymbol{\mu}_{1,L_1}, \boldsymbol{\Sigma}_{1,L_1}), P_1(L_1)\}$$
$$j = 2: \quad \{\boldsymbol{\mu}_{2,1}, \boldsymbol{\Sigma}_{2,1}), P_2(1)\}, \{\boldsymbol{\mu}_{2,2}, \boldsymbol{\Sigma}_{2,2}), P_2(2)\}, \cdots, \{\boldsymbol{\mu}_{2,L_2}, \boldsymbol{\Sigma}_{2,L_2}), P_2(L_2)\}$$

GMM Classifier

- Not for clustering, but for classification!

- During the test time, since we know all the parameters we need
- We're going to apply the same Bayes rule

$$P(j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|j)P(j)}{\sum_j P(\boldsymbol{x}|j)P(j)}$$

- Some additional summations to take care of GMM inside the classifier

i.e. $P(j) = \frac{|\mathcal{C}_j|}{\sum_{j'=1}^{J} |\mathcal{C}_{j'}|}$

Prior for the j-th class

The summation over l_j is for this j-th class only

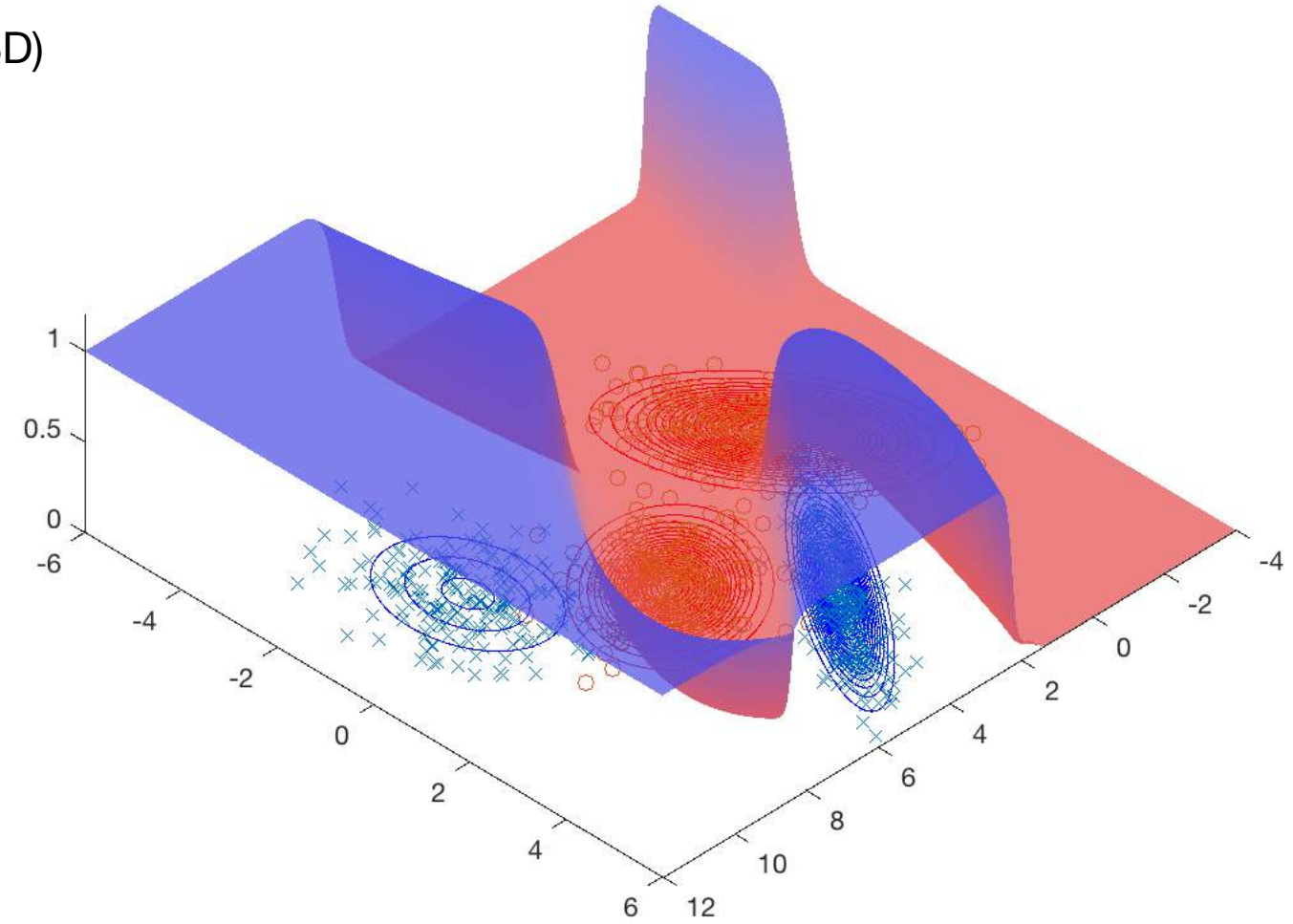
GMM for the j-th class

$$P(j|\boldsymbol{x}) = \frac{P(j)\sum_{l_j=1}^{L_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{j,l_j}, \boldsymbol{\Sigma}_{j,l_j}) P_j(l_j)}{\sum_j P(j) \sum_{l_j=1}^{L_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{j,l_j}, \boldsymbol{\Sigma}_{j,l_j}) P_j(l_j)}$$

GMM Classifier

- Not for clustering, but for classification!

- GMM classifier (3D)



Non-parametric Classifier

- Parzen-Rosenblatt windows for kernel density estimation: let's go finer

- What if we assume much more Gaussians per class?
  - Like, $L_j = |\mathcal{C}_j|$ Gaussians?

- Too many parameters to estimate during training
  - No way to calculate the covariance matrix

- Let's forget about the covariance matrices

$$\boldsymbol{\Sigma}_{j,l} = \sigma^2 \boldsymbol{I}, \quad \forall j, l$$

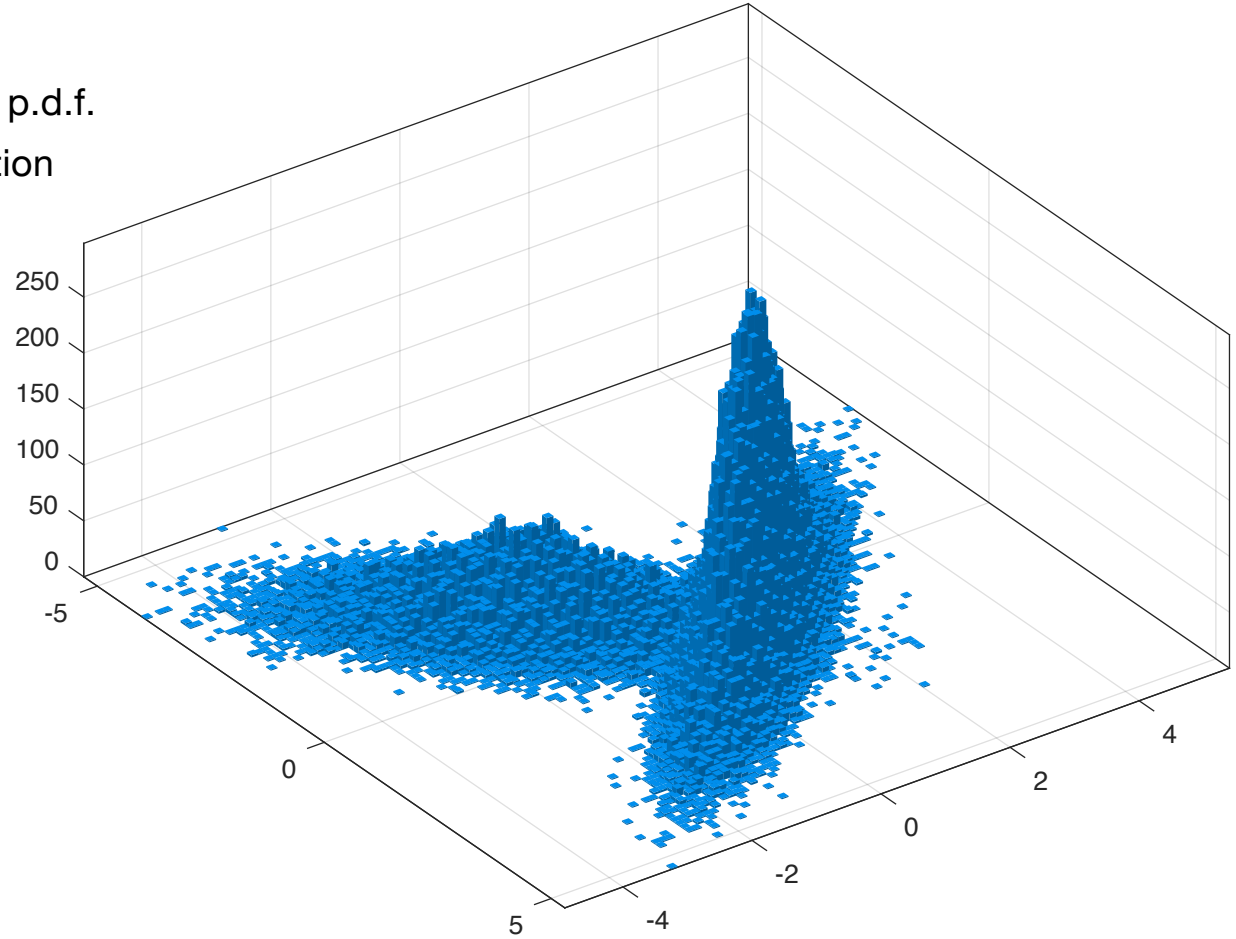
- Let's forget about the means: assume mean is the training samples

$$\mu_{j,l} = \boldsymbol{x}_i, \quad \text{where } i \in \{1, 2, \cdots, L_j\}$$

Non-parametric Classifier

- Parzen-Rosenblatt windows for kernel density estimation: let's go finer

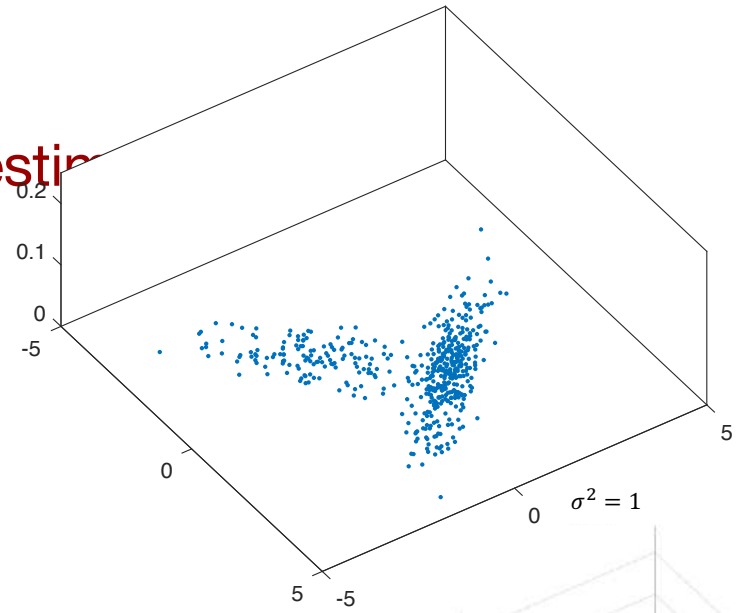
- Non-parametric density estimation
  - Doesn't assume any parametric function for the p.d.f.
  - Data says something about the sample distribution
  - e.g. Histograms

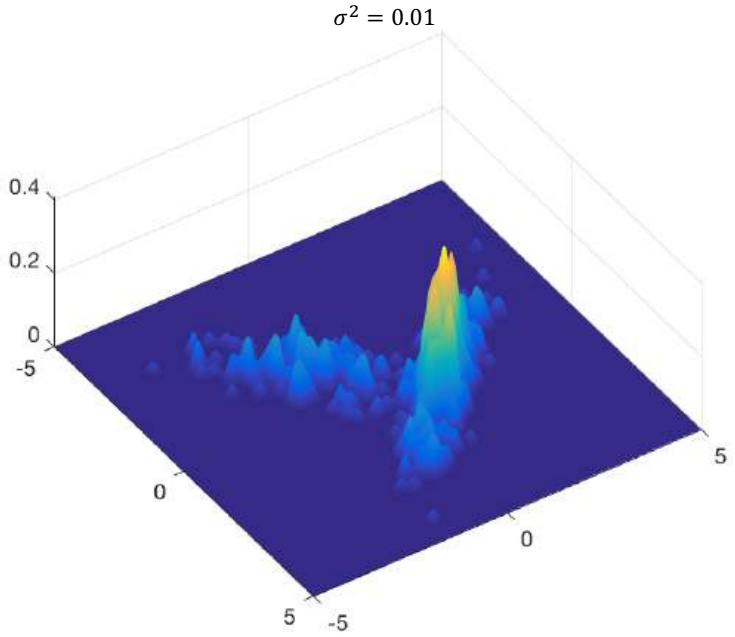


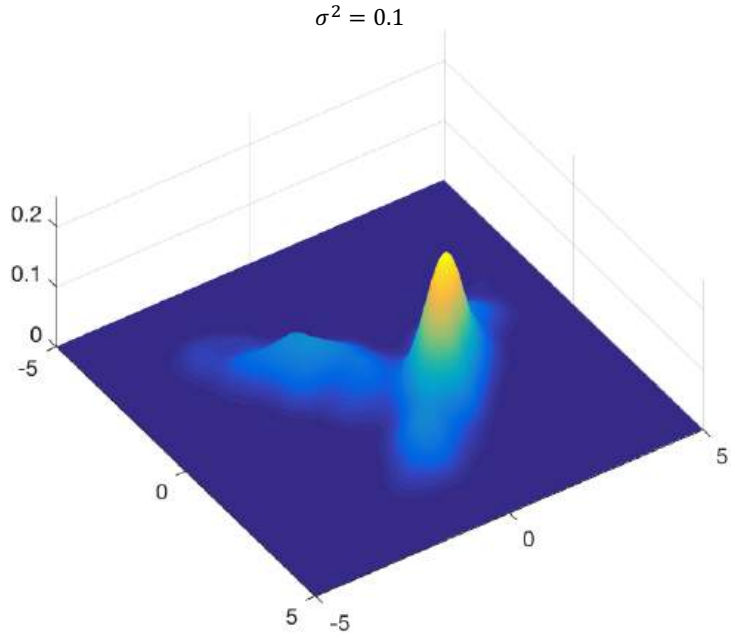
Non-parametric Classifier

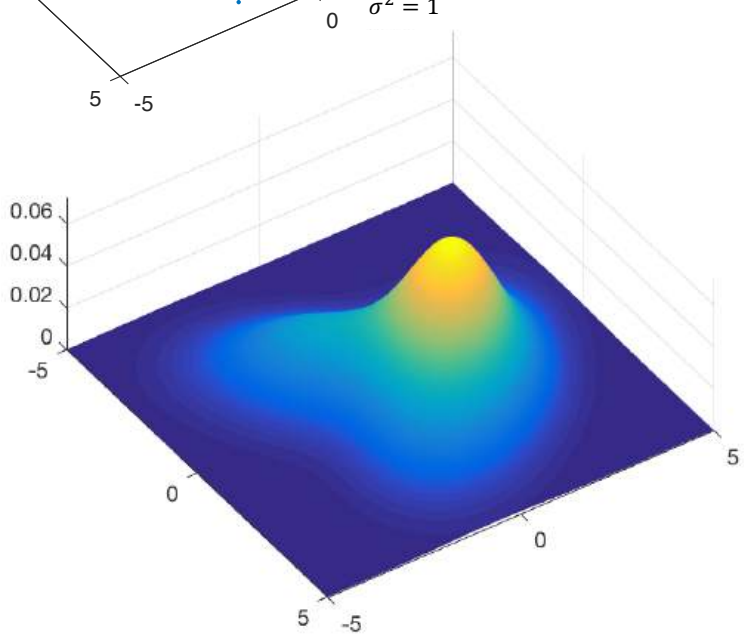
- Parzen-Rosenblatt windows for kernel density estim

- In kernel density estimation
  - Each sample forms a small Gaussian with an isotropic covariance









Non-parametric Classifier

- Parzen-Rosenblatt windows for kernel density estimation: let's go finer

- In k-means clustering what we assume about the covariance is
  - Isotropic (spherical) $\sigma_j \boldsymbol{I}$
  - $\sigma_1 = \sigma_2 = \cdots = \sigma_J$
- On top of clustering, another purpose of GMM was
  - Density estimation
- KDE using Gaussian kernels

$$p(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I})$$

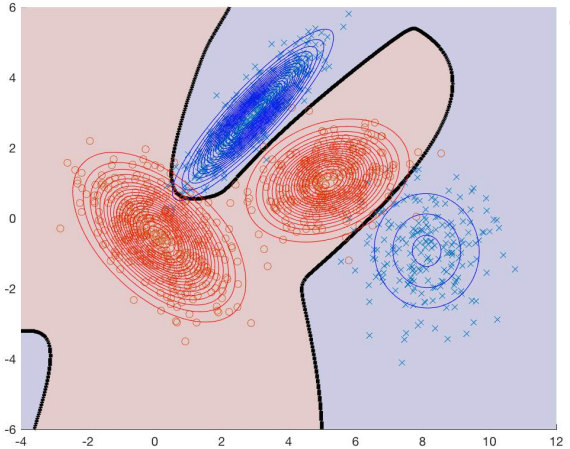
- In general

$$p(\boldsymbol{x}) = \frac{1}{N\sigma} \sum_{i=1}^{N} \mathcal{K}\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{\sigma} \right)$$

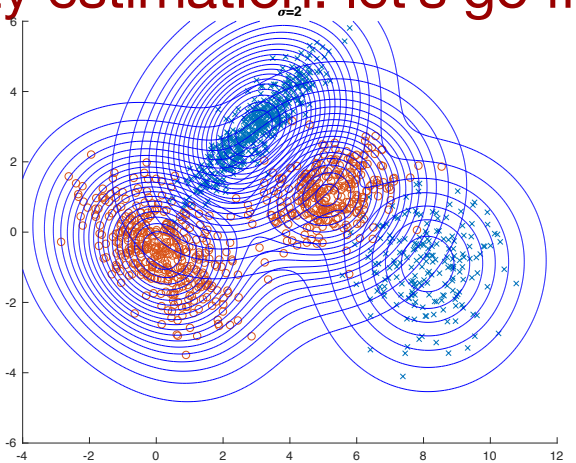
  - Integral of the kernel function $\mathcal{K}$ should be 1

Non-parametric Classifier

- Parzen-Rosenblatt windows for kernel density estimation: let's go finer

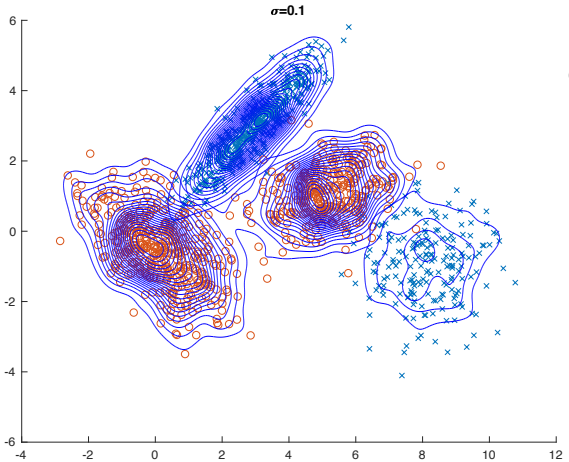


GMM Classifier



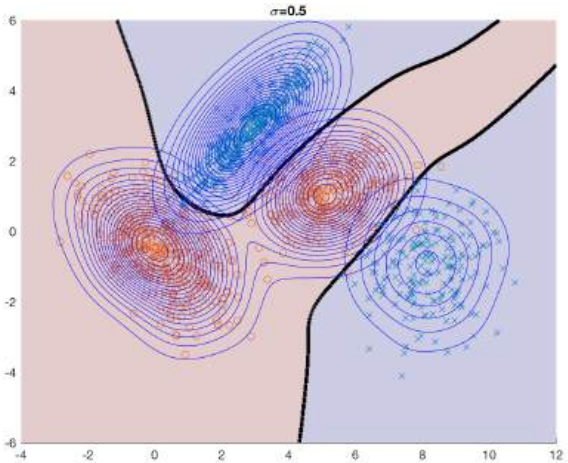
KDE Classifier

$\sigma^2 = 2$



KDE Classifier

$\sigma^2 = 0.1$



KDE Classifier

$\sigma^2 = 0.5$

Non-parametric Classifier

- Parzen-Rosenblatt windows for kernel density estimation: let's go finer

- I don't like it, because
  - Too much fuss
  - For a test sample, its posterior probability involves all training samples

$$P(j|\boldsymbol{x}) = \frac{\cancel{\frac{|\mathcal{C}_j|}{N}} \sum_{i \in \mathcal{C}_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}) / \cancel{|\mathcal{C}_j|}}{\sum_j \cancel{\frac{|\mathcal{C}_j|}{N}} \sum_{i \in \mathcal{C}_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}) / \cancel{|\mathcal{C}_j|}}$$

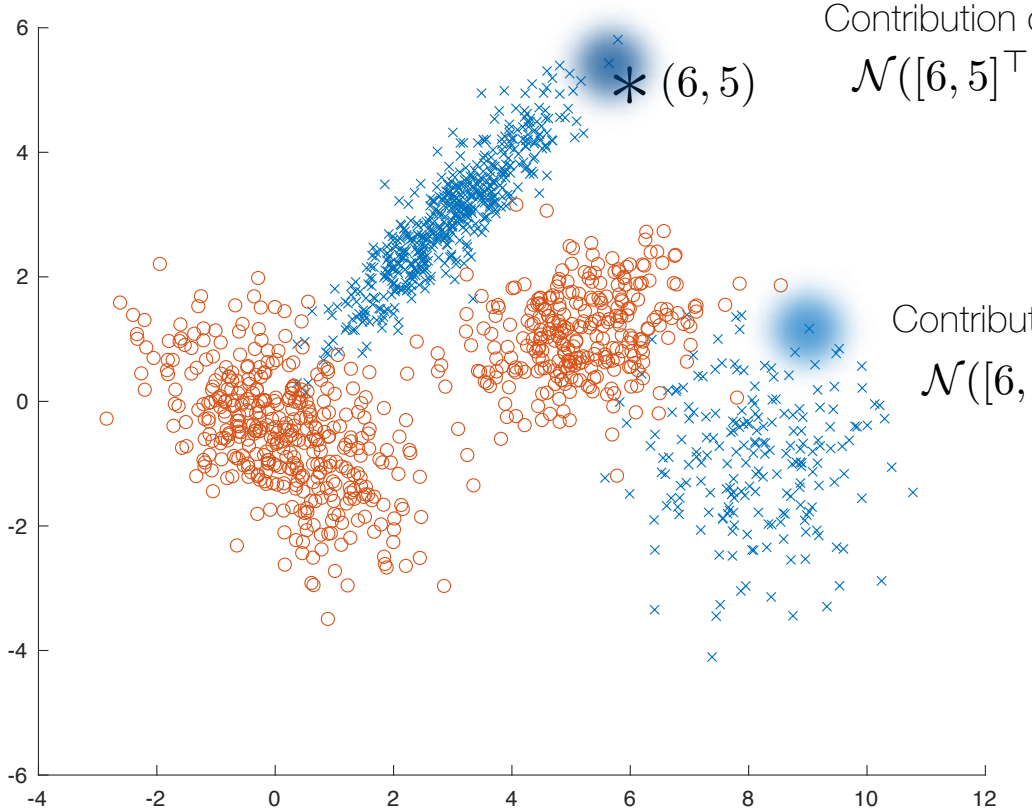
$$= \frac{\sum_{i \in \mathcal{C}_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I})}{\sum_j \sum_{i \in \mathcal{C}_j} \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_i, \sigma^2 \boldsymbol{I})}$$

  - [Sum of all the pdf defined by all the training samples of that class] / [Sum of all the pdf defined by all the training samples]
  - Not a very attractive feature as a classifier (e.g. for a real-time application)

Non-parametric Classifier

- k-Nearest Neighbors

- Why do we care about the small Gaussian distribution that's far away from my current test sample?



Contribution of the closest training sample:

$$\mathcal{N}([6,5]^\top; [5.638, 5.431]^\top, 0.5\boldsymbol{I}) = 0.2309$$

Contribution of an a bit far training sample:

$$\mathcal{N}([6,5]^\top; [9.024, 1.168]^\top, 0.5\boldsymbol{I}) = 0.00000000001426$$

Non-parametric Classifier

- k-Nearest Neighbors

- *k*NN density estimation (for *j*-th class)
  - We assume a region $\mathcal{R}$ tightly defined with the *k* neighbors
  - Its volume V depends on the density of the region
  - This $\mathcal{R}$ is our kernel, whose pdf is uniform (not Gaussian)

$$P(\boldsymbol{x}|l) = \frac{1}{V}$$

  - What's the prior probility?
    - Probability of choosing this particular kernel?

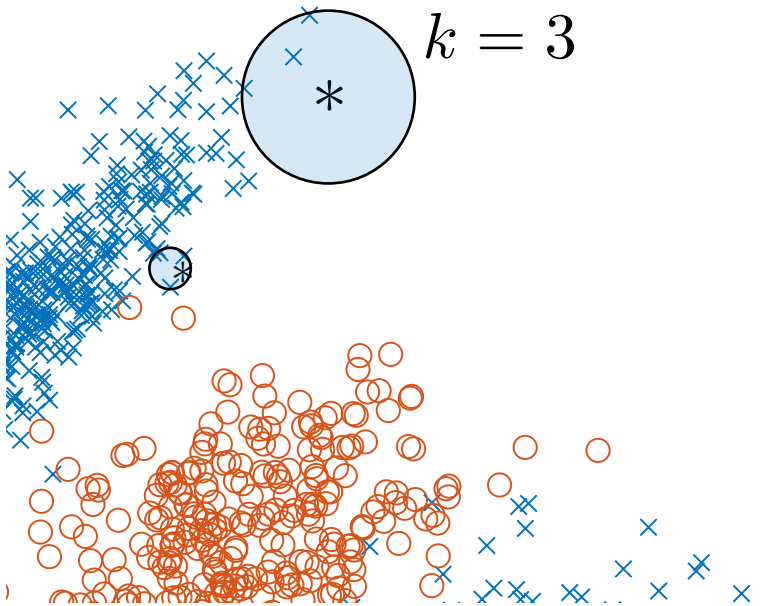
$$P(l) = \frac{k}{N}$$

  - Therefore, the density of *j*-th class

$$P(\boldsymbol{x}|j) = \sum_{l=1}^{L_j} P(\boldsymbol{x}|l)P(l)$$

$$= P(\boldsymbol{x}|l)P(l) \leftarrow$$

because l is the only kernel $\boldsymbol{x}$ belongs to according to the kNN definition



Non-parametric Classifier

- k-Nearest Neighbors

- *k*NN classifier
  - Once again, the same Bayes rule

$$P(j|\boldsymbol{x}) = \frac{P(j)\sum_{l_j=1}^{L_j} P(\boldsymbol{x}|l_j)P(l_j)}{\sum_{j=1}^{J} P(j)\sum_{l_j=1}^{L_j} P(\boldsymbol{x}|l_j)P(l_j)} = \frac{P(j)P(\boldsymbol{x}|l_j)P(l_j)}{\sum_{j=1}^{J} P(j)P(\boldsymbol{x}|l_j)P(l_j)}$$

    - But this time we see only one kernel, formed with the *k*NN
  - Plug in the *k*NN probabilities

$$P(\boldsymbol{x}|l_j) = \frac{1}{V_{l,j}} \qquad P(l_j) = \frac{k}{|\mathcal{C}_j|} \qquad P(j) = \frac{|\mathcal{C}_j|}{N}$$

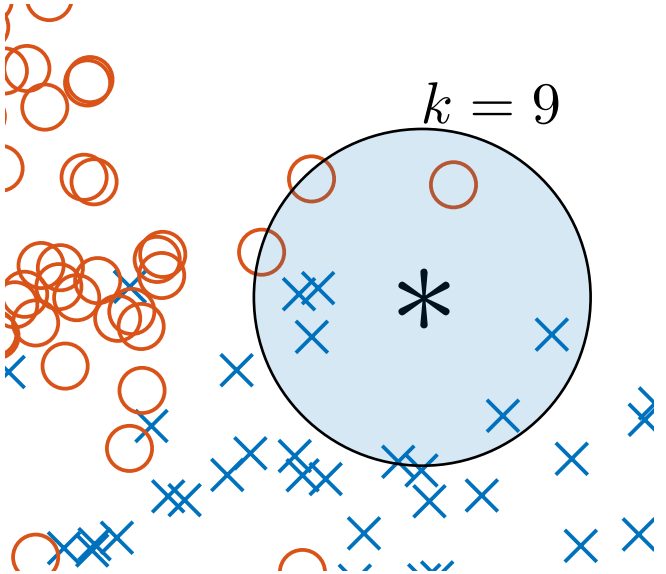
  - Then, $P(j|\boldsymbol{x}) = \frac{\frac{|\mathcal{C}_j|}{N}\frac{k}{V_{l,j}|\mathcal{C}_j|}}{P(\boldsymbol{x})} = \frac{k}{V_{l,j}NP(\boldsymbol{x})} = \frac{Const}{V_{l,j}}$
  - **If the volume of kernel is inverse-proportional to the posterior probability**

$$V_{l,j} > V_{l,j'} \rightarrow \quad P(j'|\boldsymbol{x}) > P(j|\boldsymbol{x})$$

Non-parametric Classifier

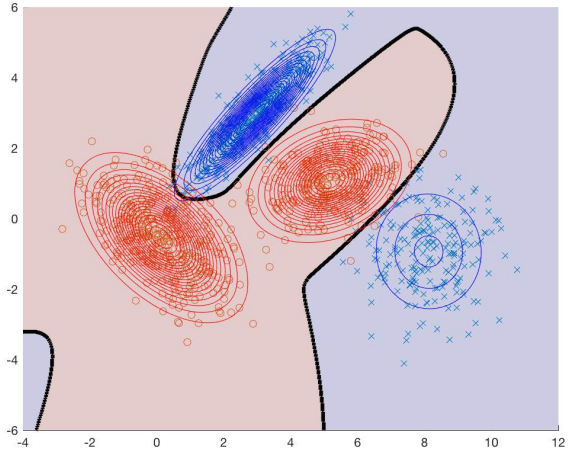
- k-Nearest Neighbors

- So, how do we know the two volumes of the kernels formed from different classes?
- The *k*NN classification algorithm
  1. For a test sample, find its k NN
  2. Count the number of training samples from each class
     - e.g. In the figure,
       6 from class 1,
       3 from class 2
  3. The class with more samples is denser in that area ➔ its kernel volume is smaller
     ➔ **choose the class with more samples**

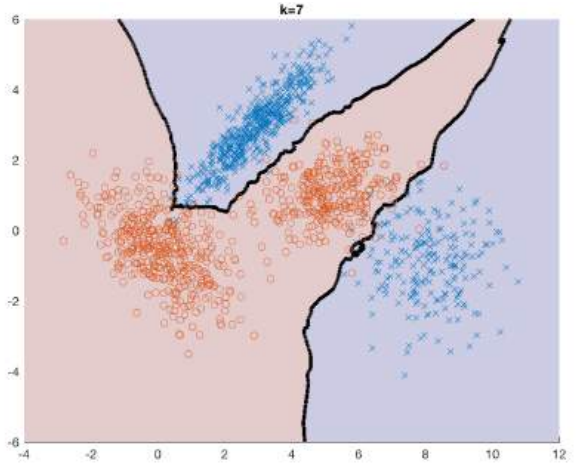


Non-parametric Classifier

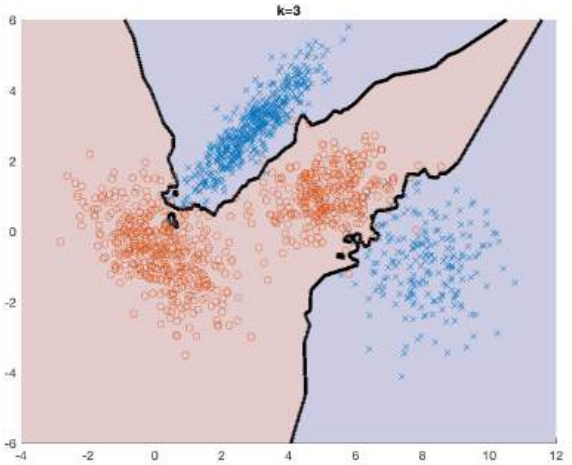
- k-Nearest Neighbors



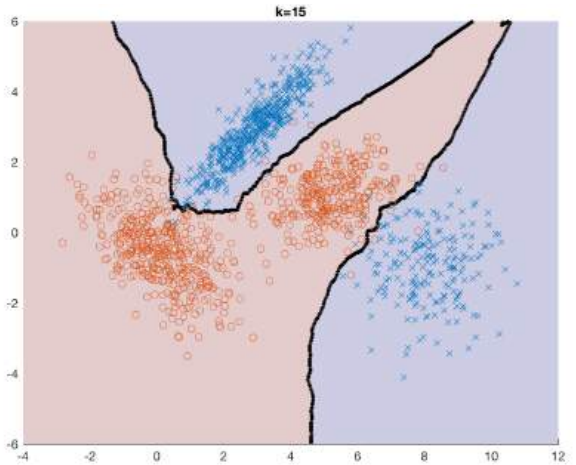
GMM Classifier



7NN Classifier



3NN Classifier



15NN Classifier

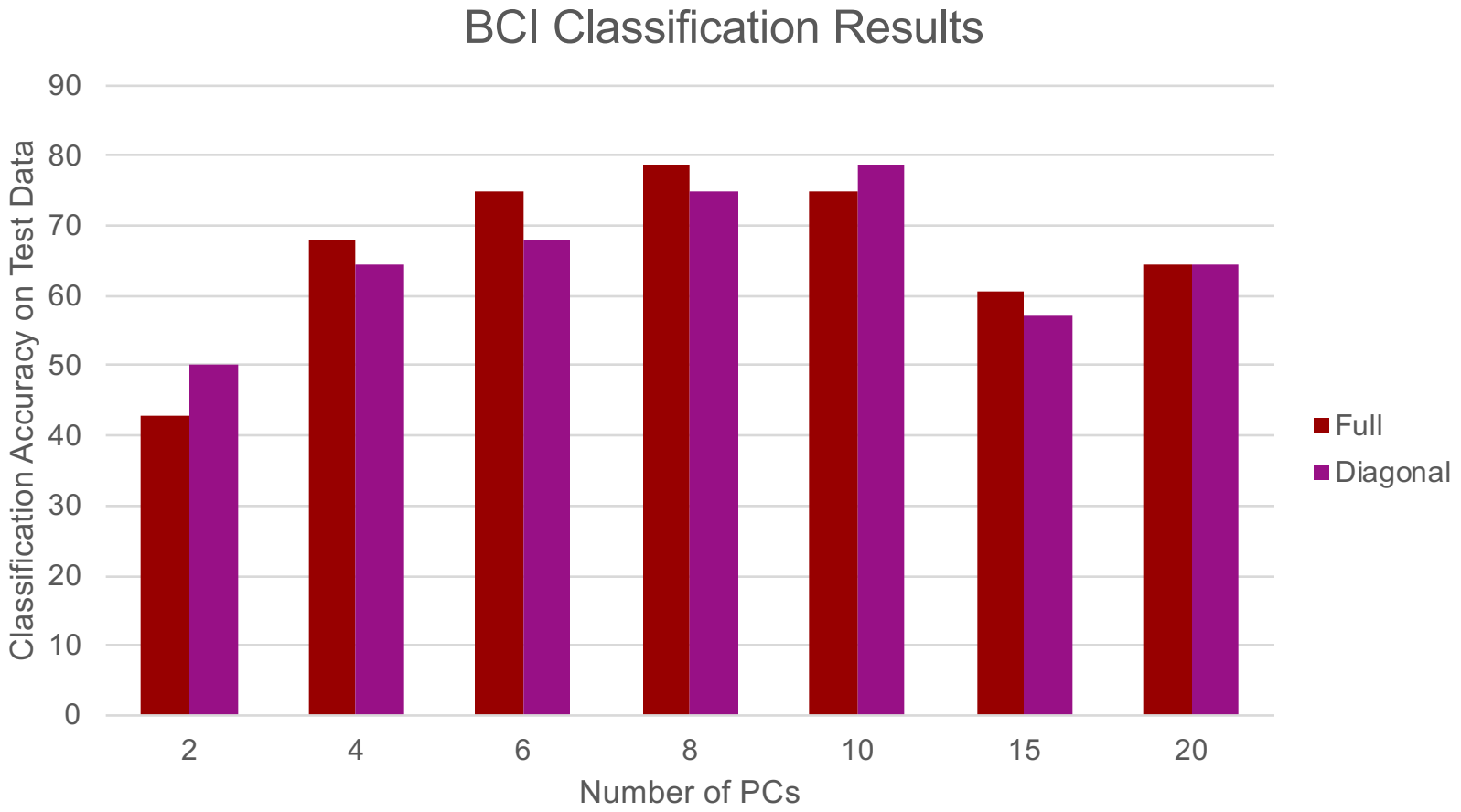
Occam's Razor

- There is a trade-off between the model complexity and the performance of the classifier
  - Naïve Bayes classifier with isotropic Gaussians is easier to train than the one with full covariance matrix
  - But, if we succeed to estimate the full cov matrices, the decision boundaries are quite flexible
    - If you don't have enough training samples and fail to estimate the model parameters correctly, your decision boundaries will be messier
  - Are more complex classifiers always better than simpler ones?
    - Maybe not
    - Overfitting

- Non-parametric methods work well near the well sampled area
  - It's free from parameter estimation, but not entirely parameter free
    - e.g. the number of neighbors
  - Sorting is the bottleneck

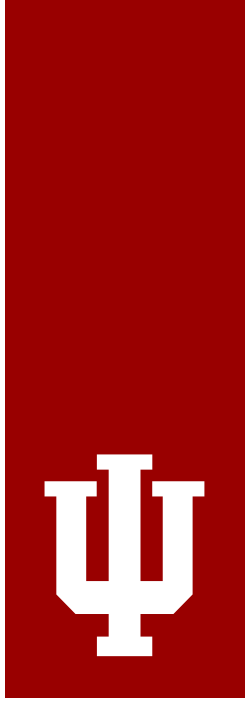
BCI Classification Results

- Left or right



Reading

- Textbook 2.1 – 2.6



Thank You!