Image Analysis
Exercise - Advanced segmentation
Fisherman's Linear discriminant analysis for segmentation

Introduction

The exercise is to the extent of the pixel-wise classification problem from being based on the intensity histogram of a single image modality to combining multiple image modalities. Hence, here we wish to segment image features into two classes by training a classifier based on the intensity information from multiple image modalities.
Multiple-image modalities just mean a series of images that contain different but complementary image information of the same object. It is assumed that the image modalities have the same size, so we have a pixel-to-pixel correspondence between the two images. An image feature is an identifiable object in the image e.g., of a dog, moon rocket, or brain tissue types that we wish to segment into individual classes.

Here we aim to segment two types of brain tissues into two feature classes. To improve the segmentation, we wish to combine two MRI image modalities instead of a single: one is a "T1 weighted MRI" and the other is a "T2 weighted MRI". Both are acquired at the same time.

Get the data from **courses.compute.dtu.dk/02502**.

Exercise - You simply go step-by-step and fill in the command lines and answer/discuss the questions (Q1-Q12).

Theory

The Linear Discriminate Classifier
As a classifier, we will use a class of linear discriminate functions that aims to place a hyperplane in the multi-dimensional feature space that acts as a decision boundary to segment two features into classes. Since we only look at image intensities of two image modalities our multi-dimensional feature space is a 2D intensity histogram. The linear discriminant classifier is based on the Bayesian theory where the posterior probability is the probability of voxel $\boldsymbol{x}$ belonging to class $\boldsymbol{C_i}$. The voxel x belongs to the class with the highest posterior probability. A class posterior probability is expressed by Bayes:

$$P(Ci|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i},)\boldsymbol{P}(\boldsymbol{C_i})}{P(\boldsymbol{x})} \quad \text{[Eq. 1]}$$

Where

- $P(x|\mu_i, \Sigma_i,) = K_i\ exp((x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i))$ is the conditional probability. We assumes that the distribution of data for a feature can be modeled using a parametric Gaussian model in multiple dimensions i.e.: $\mathcal{N}(\mu_i, \Sigma_i)$
- $P(C_i)$ is the prior probability for each class and $\sum_i P(\boldsymbol{C_i}) = 1$
- $P(x)$ is the marginal probability and is $\sum_i P(\boldsymbol{C_i}|x)$.

In Eq. 1 we need to train the model using training examples representative of the distribution of features to be segmented. The training examples we often draw ourselves or are provided by an expert. Given the training examples, we train the model parameter which are the means, covariances, and prior probabilities for each class.

The linear discriminant function for two classes is based on the log-ratio between the two posterior probabilities:

$$\text{Log}(\frac{P(C2|\boldsymbol{x})}{P(C1|\boldsymbol{x})}) > \log(T) \text{ [Eq. 2]}$$

Note, we take the log() on both sides as a trick to make the expressions easier to realize the model practically but then remember: To calculate the final class posterior probabilities once the classifier model has been trained one needs to account for the log(P(Ci|x)).

We derive the model based on the expression in Eq 2 and its model assumptions.
Firstly, we assume homoscedasticity of variances meaning that the variance of the covariance matrix of each feature is the same for all classes ($\Sigma_1 = \Sigma_2 = \Sigma_0$). Further we assume to have isotropic covariances i.e., no preferred direction or round shape distribution in 2D as illustrated in Figure 1. This means that the covariances in the off diagonal of the covariance matrix is zero and we have equal variances in the diagonal.

With these assumptions, Eq. 2 is a Linear Discriminant Analysis (LDA) classifier and can be expressed by the means and the covariances of the two gaussian distributions and their prior probabilities:

$$\log\left(\frac{P1}{P2}\right) - \frac{1}{2}(\boldsymbol{\mu_2} + \boldsymbol{\mu_1})^T \Sigma_0^{-1}(\boldsymbol{\mu_2} - \boldsymbol{\mu_1}) + \boldsymbol{x}^T \Sigma_0^{-1}(\boldsymbol{\mu_2} - \boldsymbol{\mu_1}) > \log(T) \text{ [Eq. 3]}$$

The expression can look a bit complicated at first glimpse but if we add colors a general structure of the expression appears. The green part $\Sigma_0^{-1}(\boldsymbol{\mu_2} - \boldsymbol{\mu_1})$ we name a weight vector, $\boldsymbol{w}$ which is normal to the hyperplane, and along $\boldsymbol{w}$ the decision boundary separating the two classes is to be placed. We can say that $\boldsymbol{w}$ determines how the hyperplane is orientated in the coordinate system to best separate the two classes. In other words, $\boldsymbol{w}$ is our model like in Eq. 3, and is defined by its assumptions.
The orange part $\log\left(\frac{P1}{P2}\right) - \frac{1}{2}(\boldsymbol{\mu_2} + \boldsymbol{\mu_1})^T$ we call the bias $\boldsymbol{c}$ which describes where the hyperplane hence the decision boundary along the $\boldsymbol{w}$ is placed. Basically, the expression for $\boldsymbol{c}$ says that the decision boundary is placed halfway between the means of the two class distributions weighted with the ratio of the prior probability for each class. So, it is halfway between the class means if the two classes have equal prior probabilities, $P(C_i)$ (i.e., 0.5). In 1D, it is the same as the parametric classifier where we define a threshold where the two distributions cross and have equal probabilities.

We can reformulate Eq. 3 in a general way:

$$\boldsymbol{y_{C\in 2}(x)} = \boldsymbol{x^T w + w_o} \text{ ; where } \boldsymbol{w_o = c\, w} \text{ [Eq. 4]}$$

The weight vector is multiplied both with the vector of a data point $\boldsymbol{x}$ and with the vector of the bias $\boldsymbol{c}$. Multiplying two vectors is the same as the dot-product between two vectors i.e., $\boldsymbol{ab} = \|\boldsymbol{a}\|\|\boldsymbol{b}\| \cos\theta$. Hence, the function of the weight vector is to rotate the hyperplane in feature space with an angle that separates the two classes. Hence, when multiplied with the bias $\boldsymbol{c}$ it is projected along $\boldsymbol{w}$ and defines the decision boundary. When multiplied with a data point vector $\boldsymbol{x}$ it projects the data point along $\boldsymbol{w}$ and whether the projected data point belongs to class 1 or 2 depends on its projected position in relation to the decision boundary. Figure 1 illustrates for the LDA model and two gaussian classes with equal isotropic variances the principle of how the weight-vector projects the hyperplane and decision boundary independent of the orientation between the two class means i.e., Figure 1 (A vs B).

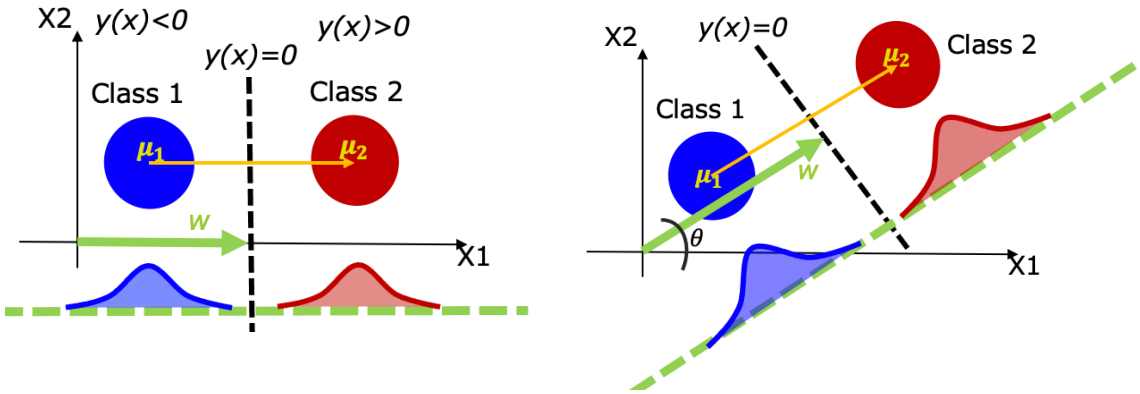


Figure 1 Principle how the LDA model can be explained by using a weight vector $\boldsymbol{w}$ that is normal to the hyperplane. The weight vector projects the hyperplane to define the decision boundary ($\boldsymbol{cw}$) (striped line) that separates the two classes. Data points are likewise projected along $\boldsymbol{w}$. Left: the hyperplane follows the coordinate system. Right: The hyperplane is not aligned with the coordinate system and has a rotation angle.

In summary, to classify if a point $\boldsymbol{x}$ belongs to class 2 using Eq. 4 we multiply it with $\boldsymbol{w}$ which projects the point into the same orientation as the hyperplane and then we subtract the bias. If $\boldsymbol{y_{C\in 2}(x)} > \log(T)$ the point is closer to class 2 than class 1 (Note, $\log(T) = 0$ in Eq 3).

The Fisher's linear discriminant classifier

The LDA classifier model assumptions are not correct if the variances of the two gaussian distributions are not equal or isotropic. In this case the hyperplane of the LDA is not guaranteed to optimally separate the two classes as illustrated in Figure 2A. However, if changing the model to account for non-equal and non-isotropic class variances, the projections of the hyperplane, and decision boundary will ensure a better class-separation as shown in Figure 2B. This will result in more precise classifications.

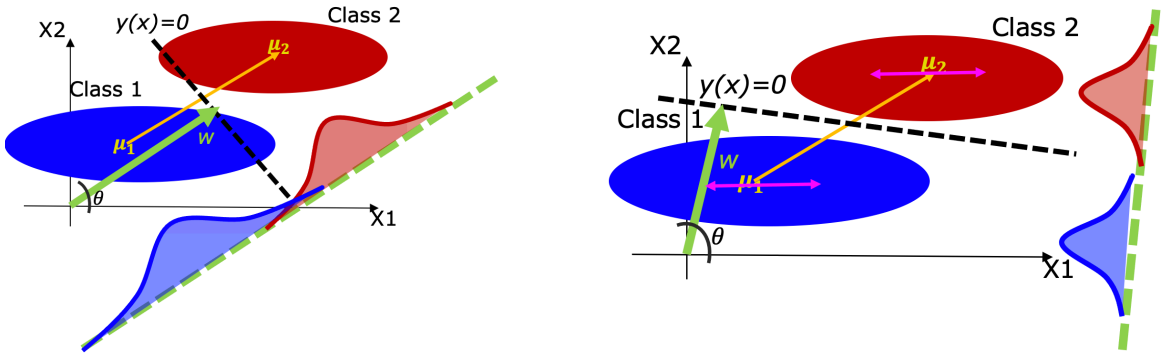


Figure 2 Illustration that is the two class distributions have different covariances and non-isotropic the LDA model can fail to place the hyperplane to optimally separate the two classes (Left). Right: If we make a new model i.e., Fisher's LDA assuming that covariances can be different the hyperplane better separated the two classes.

The Fisher's discriminate classifier accounts for non-equal and non-isotropic class variances by taking the ratio of *between-class covariance* and the *total within covariance*. The weight-vector is to project the hyperplane, so it minimizes the ratio between the two. For this, we can express a cost function to find $\boldsymbol{w}$. The between-class covariance is $S_B = (\boldsymbol{\mu_2} - \boldsymbol{\mu_1})^T(\boldsymbol{\mu_2} - \boldsymbol{\mu_1})$ and the total within-covariance is $S_w = \boldsymbol{\Sigma_1} + \boldsymbol{\Sigma_2}$. **Now** we search for a solution of the weight-vector that minimizes the ratio between the two covariances.

$$\boldsymbol{J(w)} = \frac{\boldsymbol{w^T S_B w}}{\boldsymbol{w^T S_w w}}$$

This defines the cost function that we differentiate and set to zero to find $\boldsymbol{w}$.

$$\frac{\partial \boldsymbol{j}(\boldsymbol{w})}{\boldsymbol{w}} = \mathbf{0}$$

The new weight-vector, $\boldsymbol{w}$, based on Fisher's discriminate classifier model expresses $w = S_w^{-1}(\boldsymbol{\mu_2} - \boldsymbol{\mu_1})$ and used it in Eq 4. The bias, $\boldsymbol{c}$, is the same as for the LDA as we still wish the decision boundary to be placed halfway between the two class means.
When knowing the expression of $\boldsymbol{w}$ and $\boldsymbol{wo}$ we can use Fisher's linear discriminant classifier to classifier if a sample belongs to class 2 or not as well as to estimate the posterior probability in the same way as described for the LDA.

Provided data and functions

ex6_ImgData2Load.mat contains all image and ROI data which are loaded into the variables:

ImgT1 One axial slice of the brain using T1W MRI acquisition

ImgT2 One axial slice of the brain using T2W MRI acquisition

ROI_WM Binary training data for class 1: Expert identification of voxels belonging to tissue type: White Matter

ROI_GM Binary training data for class 2: Expert identification of voxels belonging to tissue type: Grey Matter

LDA.py A Python function that realizes Fisher's linear discriminant analyses as described in Note for lecture 6.

Learning objectives

After completing this exercise, the student should be able to do the following:

1. Implement, train, and evaluate multi-dimensional segmentation using a Linear Discriminate classifier i.e., Fisherman' Linear discriminant analysis
2. To visualize the 1D intensity histograms of two different image modalities that contain different intensity information of the same image features.
3. To identify the expected intensity thresholds in each of the 1D histograms that best segment the same feature in the two image modalities.
4. To visually the 2D histogram of two image modalities that map the same object but with different intensity information.
5. To interpret the 2D histogram information by identifying clusters of 2D intensity distributions and relating these to the images' features.
6. To draw an expected linear hyper plan in the 2D histogram that best segments and features in the two image modalities
7. To extract training data sets and their corresponding class labels from expert-drawn regions-of-interest data, and map their corresponding 2D histogram for visual inspection
8. To relate the Bayesian theory to a linear discriminate analysis classifier for estimating class probabilities of segmented features.

9. To judge if the estimated linear or a non-linear hyper plan is optimally placed for robust segmentation of two classes.

Image Segmentation

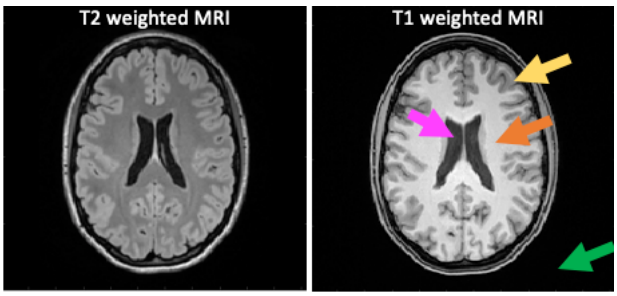
Start by importing some useful functions:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
```

And the data:

```
in_dir = 'data/'
in_file = 'ex6_ImagData2Load.mat'
data = sio.loadmat(in_dir + in_file)
ImgT1 = data['ImgT1']
ImgT2 = data['ImgT2']
ROI_GM = data['ROI_GM'].astype(bool)
ROI_WM = data['ROI_WM'].astype(bool)
```

Exercise 1 *Display both the T1 and T2 images, their 1 and 2D histograms and scatter plots. Tips: Use the plt.imshow(), plt.histogram() and plt.scatter() functions Add relevant title and label for each axis. One can use 'plt.subplots()' to show more subfigures in the same figure. Remove intensities from background voxels for 1D and 2D histograms.*



The two MRI image modalities contain different types of intensity classes:

(I) (Orange): The White Matter (WM) is the tissue type that contains the brain network - like the cables on the internet. The WM ensures the communication flow between functional brain regions.

(II) (yellow): The Grey Matter (GM) is the tissue type that contains the cell bodies at the end of the brain network and are the functional units in the brain. The functional units are like CPUs in the computer. They are processing our sensorial input and are determining a reacting to these. It could be to start running.

(III) (Magenta): Cerebrospinal fluid (CSF) which is the water in the brain

(IV) (green): Background of the image

Q1: What is the intensity threshold that can separate the GM and WM classes (roughly) from the 1D histograms?

Q2: Can the GM and WM intensity classes be observed in the 2D histogram and scatter plot?

Exercise 2 *Place trainings examples i.e. ROI_WM and ROI_GM into variables C1 and C2 representing class 1 and class 2 respectively. Show in a figure the manually expert drawings of the C1 and C2 training examples.*
Tips: use plt.imshow()

Q3: Do the ROI drawings look like what you expect from an expert?

Exercise 3 *For each binary training ROI find the corresponding training examples in ImgT1 and ImgT2. Later these will be extracted for LDA training.*
Tips: You may use the 'np.argwhere()' function which returns the index to voxels in the image full filling e.g. intensity values >0 hence belong to a given class. Name the index variables qC1 and qC2, respectively

Q4: What is the difference between the 1D histogram of the training examples and the 1D histogram of the whole image? Is the difference expected?

Exercise 4 *Make a training data vector (X) and target class vector (T) as input for the 'LDA()' function. T and X should have the same length of data points.*

X *Training data vector should first include all data points for class 1 and then the data points for class 2. Data points are the two input features ImgT1, ImgT2*

T *Target class identifier for X where '0' are Class 1 and a '1' is Class 0.*

Exercise 5 *Make a scatter plot of the training points of the two input features for class 1 and class 2 as green and black circles, respectively. Add relevant titles and labels to the axis*

Q5: How does the class separation appear in the 2D scatter plot compared with the 1D histogram? Is it better?

Exercise 6 *Train the linear discriminant classifier using the Fisher discriminant function and estimate the weight-vector coefficient W (i.e. w0 and w) for classification given X and T by using the 'W=LDA()' function. The LDA function outputs W=[[$w0_1$ w_1]; [$w0_2$ w_2]] for class 1 and 2 respectively.*

Tip: Read the Bishop note in Chapter 4.

W = LDA(X,T);

Exercise 7 *Apply the linear discriminant classifier i.e. perform multi-modal classification using the trained weight-vector W for each class: It calculates the linear score Y for *all* image data points within the brain slice i.e. y(x)= w+w0. Actually, y(x) is the log($P(C_i|x)$).*

```
Xall= np.c_[ImgT1[mask].flatten(), ImgT2[mask].flatten()]
Y = np.c_[np.ones((len(Xall), 1)), Xall] @ W.T
```

Exercise 8 *Perform multi-modal classification: Calculate the posterior probability i.e. P(X|C1) of a data point belonging to class 1*

Note: Using Bayes [Eq 1]: Since y(x) is the log of the posterior probability [Eq2] we take exp(y(x)) to get P(X|C1)=(P(X|mu,var)P(C1))) and divide with the marginal probability (P(X)) as a normalization factor.

```
PosteriorProb = np.clip(np.exp(Y) / np.sum(np.exp(Y),1)[:,np.newaxis], 0, 1)
```

Exercise 9 *Apply segmentation: Find all voxels in the T1w and T2w image with P(X|C1)>0.5 as belonging to Class 1 using the 'find()' function. Similarly, find all voxels belonging to class 2.*

Exercise 10 *Show scatter plot of segmentation results as in 5).*

Q6 Can you identify where the hyperplane is placed i.e. y(x)=0?

Q7 Is the linear hyperplane positioned as you expected, or would a non-linear hyper plan perform better?

Q8 Would segmentation be as good as using a single image modality using thresholding?

Q9 From the scatter plot do the segmentation results make sense? Are the two tissue types segmented correctly.

Exercise 11 *Show segmentation results and training examples of class 1 and 2 in sub-figures.*

Q10 Are the training examples representative of the segmentation results? Are you surprised that so few training examples perform so well? Do you need to be an anatomical expert to draw these?

Q11 Compare the segmentation results with the original image. Is the segmentation results satisfactory? Why not?

Q12 Is one class completely wrongly segmented? What is the problem?