パターン認識
Pattern Recognition

佐藤真一
Shin'ichi Satoh

国立情報学研究所
National Institute of Informatics

Apr 25, 2023

Course Information

- Course web page: http://research.nii.ac.jp/~satoh/utpr/
- Credits will be given based on final report (mandatory) and assignments (3 out of 7 are mandatory)
- Attendance record will NOT be taken
- If you fail to submit minimum 3 assignments and final report, you will not obtain credits.

Schedule (subject to change)

4/11 Orientation, Bayes decision theory, probability distribution

4/18 Random variable, random vector, normal distributions

4/25 Parametric density estimation, discriminant function

5/2 Nonparametric density estimation, Parzen windows, k-nearest neighbor estimate

5/9 k-nearest neighbor classification, classification error estimation

5/16 Bayes error estimation, classification error estimation, cross-validation, bootstrap

5/23 Linear classifier, perceptron, MSE classifier, Widrow-Hoff rule

5/30 neural network, deep learning

6/6 all about SVM

6/13 Orthogonal expansions, Eigenvalue decomposition

6/20 no class

6/27 Clustering, dendrogram, aggromerative clustering, k-means

7/4 Graphs, normalized cut, spectral clustering, Laplacian Eigenmaps

7/11 extra (if needed)

Review of last class

- Normal distributions (Gaussian distributions)
- Sufficient statistics of normal distributions: mean and covariance matrix
- Sample estimates: sample mean and sample covariance matrix

Roadmap of Today

- Estimation of parametric probability distribution
- Design of classifier based on Bayes decision theory

Discriminant Functions: General, Multicategory Case

Let's consider the problem to classify given observation into one of c classes.
We can formalize this problem using discriminant functions

$$g_i(x), \ i = 1, \ldots, c.$$

The classifier is then assign a feature vector x to class ω_i if

$$g_i(x) > g_j(x) \text{ for all } j \neq i.$$

Discriminant Functions: General, Multicategory Case

Recall Bayes decsion Theory:

Decide ω_i *if* $P(\omega_i|x) > P(\omega_j|x)$ *for all* $j \neq i$

we can use:

$g_i(x) = P(\omega_i|x)$.

With risk:

Risk: $R(\alpha_i|x) = \sum_j \lambda_{ij} P(\omega_j|x)$

and

Decide ω_i *if* $R(\alpha_i|x) < R(\alpha_j|x)$ *for all* $j \neq i$

we can use:

$g_i(x) = -R(\alpha_i|x)$.

Discriminant Functions: General, Multicategory Case

In the case of Bayes decsion rule (minimum error rate)

$$g_i(x) = P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x|\omega_j)P(\omega_j)}$$

$$g_i(x) = p(x|\omega_i)P(\omega_i)$$

$$g_i(x) = \log p(x|\omega_i) + \log P(\omega_i)$$

Discriminant Functions: Two Category Case

We can use one dichotomizer instead of c discriminant functions:

$$g(x) \stackrel{\text{def}}{=} g_1(x) - g_2(x)$$

with the rule: Decide ω_1 if $g(x) > 0$ otherwise ω_2.
Other forms:

$$g(x) = P(\omega_1|x) - P(\omega_2|x)$$
$$g(x) = \log \frac{p(x|\omega_1)}{p(x|\omega_2)} + \log \frac{P(\omega_1)}{P(\omega_2)}.$$

Discriminant Functions for the Normal Density

Recall the normal density:

$$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

and discriminant function:

$$g_i(x) = \log p(x|\omega_i) + \log P(\omega_i).$$

The discriminant function for normal density is then:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| + \log P(\omega_i).$$

Case 1: $\Sigma_i = \sigma^2 I$

The simplest case: features are statistically independent and each feature has the same variance σ^2.
Particularly $|\Sigma_i| = \sigma^{2d}$ and $\Sigma_i^{-1} = (1/\sigma^2)I$.
Then the discriminant functions are:

$$\begin{aligned} g_i(x) &= -\frac{||x - \mu_i||^2}{2\sigma^2} + \log P(\omega_i) \\ &= -\frac{1}{2\sigma^2}[x^T x - 2\mu_i^T x + \mu_i^T \mu_i] + \log P(\omega_i). \end{aligned}$$

By ommitting common terms we obtain linear discriminant functions:

$$g_i(x) = w_i^T x + w_{i0}$$

where

$$w_i = \frac{1}{\sigma^2}\mu_i \text{ and } w_{i0} = -\frac{1}{2\sigma^2}\mu_i^T \mu_i + \log P(\omega_i).$$

Case 1: $\Sigma_i = \sigma^2 I$

Two-category case:

$$
\begin{aligned}
g(x) &= w^T(x - x_0) \\
w &= \mu_1 - \mu_2 \\
x_0 &= \frac{1}{2}(\mu_1 + \mu_2) - \frac{\sigma^2}{||\mu_1 - \mu_2||^2} \log \frac{P(\omega_1)}{P(\omega_2)}(\mu_1 - \mu_2).
\end{aligned}
$$

The decision surfaces are then hyperplanes perpendicular to the line connecting μ_1 and μ_2.
If $P(\omega_1) = P(\omega_2)$, the hyperplanes go through the middle point of the line.
This finally results in nearest neighbor classifier.

Case 1: $\Sigma_i = \sigma^2 I$

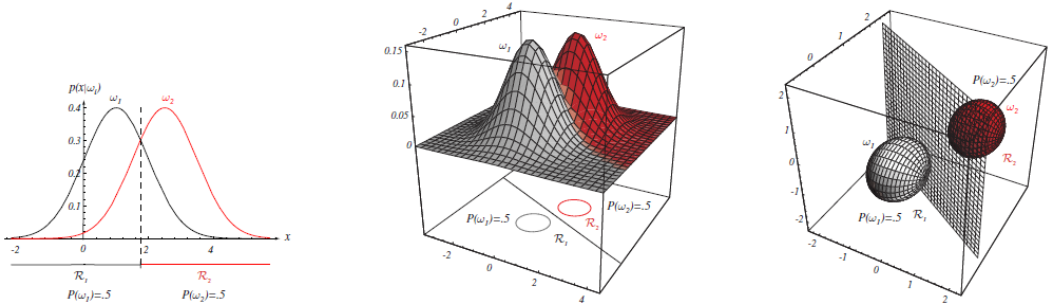


FIGURE 2.10. If the covariance matrices for two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these one-, two-, and three-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the three-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Case 1: $\Sigma_i = \sigma^2 I$

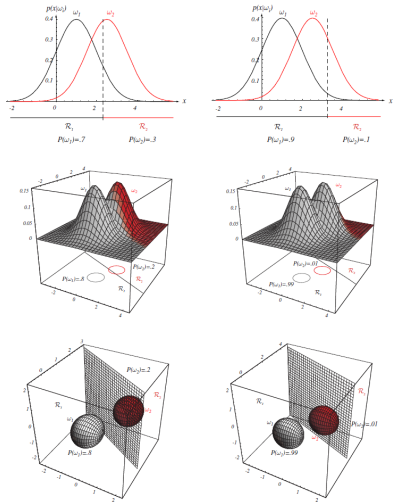


FIGURE 2.11. As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these one-, two- and three-dimensional spherical Gaussian distributions. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley &

Case 2: $\Sigma_i = \Sigma$

Another simple case: the covariance matrices are identical.
Then the discriminant functions can be simplified as

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma^{-1}(x - \mu_i) + \log P(\omega_i).$$

By expanding quadratic form $(x - \mu_i)^T \Sigma^{-1}(x - \mu_i)$ we can further simplify by dropping common terms:

$$g_i(x) = w_i^T x + w_{i0}$$

where

$$w_i = \Sigma^{-1}\mu_i \text{ and } w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log P(\omega_i).$$

Case 2: $\Sigma_i = \Sigma$

Two-category case:

$$
\begin{aligned}
g(x) &= w^T(x - x_0) \\
w &= \Sigma^{-1}(\mu_1 - \mu_2) \\
x_0 &= \frac{1}{2}(\mu_1 + \mu_2) - \frac{1}{(\mu_1 - \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2)} \log \frac{P(\omega_1)}{P(\omega_2)}(\mu_1 - \mu_2).
\end{aligned}
$$

Case 2: $\Sigma_i = \Sigma$

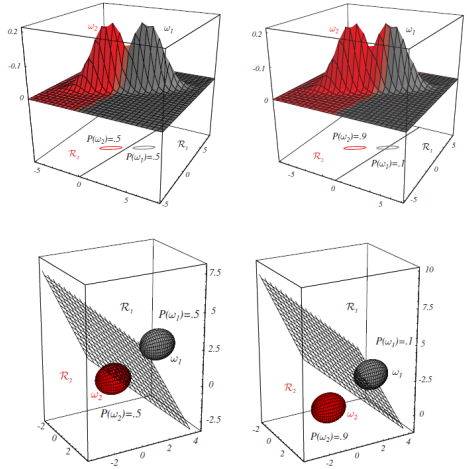


FIGURE 2.12. Probability densities (indicated by the surfaces in two dimensions and ellipsoidal surfaces in three dimensions) and decision regions for equal but asymmetric Gaussian distributions. The decision hyperplanes need not be perpendicular to the line connecting the means. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Case 3: Σ_i = arbitrary

We can drop only $(d/2)\log 2\pi$.
The discriminant functions are then quadratic:

$$g_i(x) = x^T W_i x + w_i^T x + w_{i0}$$
$$W_i = -\frac{1}{2}\Sigma_i^{-1}$$
$$w_i = \Sigma_i^{-1}\mu_i$$
$$w_{i0} = \mu_i^T W_i \mu_i - \frac{1}{2}\log|\Sigma_i| + \log P(\omega_i).$$

Case 3: Σ_i = arbitrary

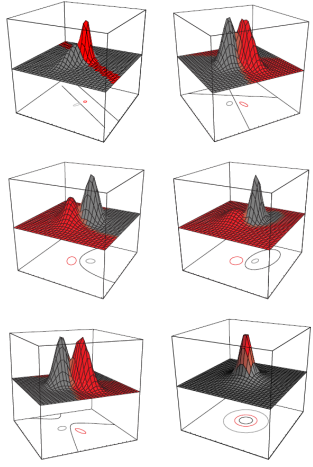
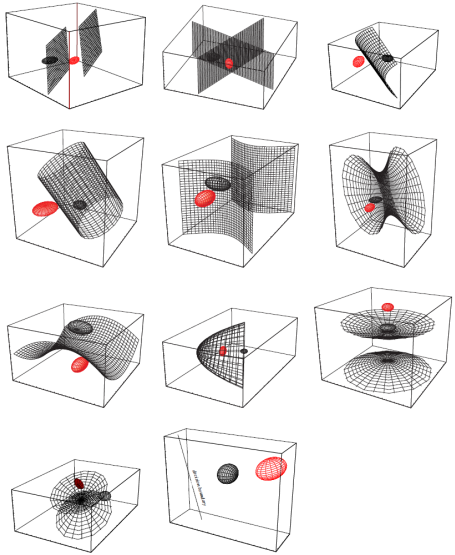


FIGURE 2.14. Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general hyperquadrics. Conversely, given any hyperquadric, one can find two Gaussian distributions whose Bayes decision boundary is that hyperquadric. These variances are indicated by the contours of constant probability density. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Case 3: Σ_i = arbitrary



Assignment 1

Generate two sets of 2-D points yielding two normal distributions.

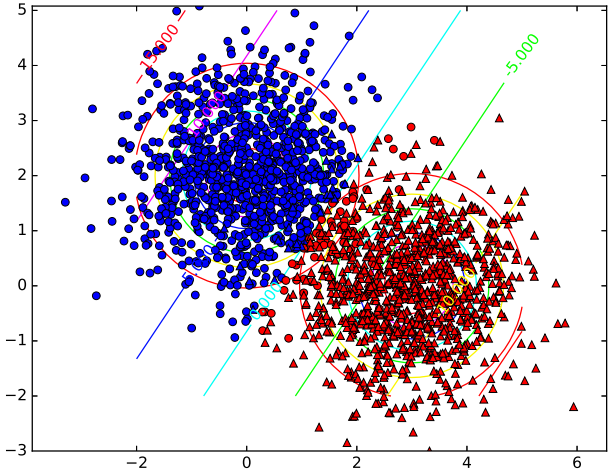
Case 1 $\Sigma_i = \sigma^2 I$

Case 2 $\Sigma_i = \Sigma$

Case 3 $\Sigma_i =$ arbitrary

Classify points by discriminant functions knowing parameters of normal distributions.
Plot scatter of points (different colors for classes and different shapes for correct/error classification), contour maps of distributions, and discrimination boundary.
Observe the phenomena by changing the parameters and priors.

Assignment 1 (cov_diag.py)



Assignment 1 (cov_diag.py)

```
import numpy as np
import matplotlib.pyplot as plt


def gausscontour(c, m, xx, yy):
    xt = xx - m[0, 0]
    yt = yy - m[1, 0]
    ic = np.linalg.inv(c)
    p = 1. / (2. * np.pi * np.sqrt(np.linalg.det(c))) \
        * np.exp(-1. / 2. * \
                 (ic[0, 0] * xt * xt \
                  + (ic[0, 1] + ic[1, 0]) * xt * yt \
                  + ic[1, 1] * yt * yt))
    return p
```

Assignment 1 (cov_diag.py)

```
d = 2
n = 1000
m1 = np.array([0., 2.])[:, np.newaxis]
m2 = np.array([3., 0.])[:, np.newaxis]
p1 = 0.3
p2 = 1 - p1
cov1 = np.eye(2)
cov2 = cov1
x1 = np.random.randn(d, n) + m1.dot(np.ones([1, n]))
x2 = np.random.randn(d, n) + m2.dot(np.ones([1, n]))
w = m1 - m2
x0 = 1. / 2. * (m1 + m2) - 1. / np.linalg.norm(m1 - \
                m2)**2. * np.log(p1 / p2) * (m1 - m2)
l1 = (w.T.dot(x1 - x0) > 0)[-1]
l2 = (w.T.dot(x2 - x0) > 0)[-1]
```

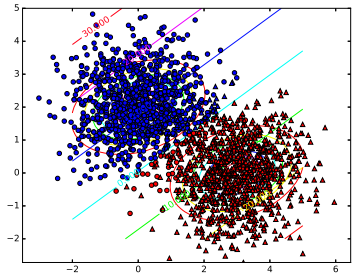
Assignment 1 (cov_diag.py)

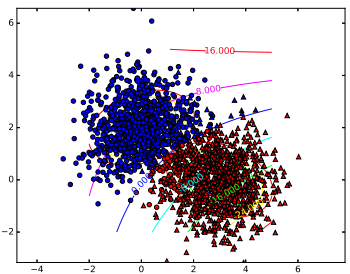
```
[xx,yy]=np.meshgrid(np.linspace(-2,5),np.linspace(-2,5))
plt.figure()
plt.axis('equal')
p1=gausscontour(cov1,m1,xx,yy)
plt.contour(xx,yy,p1,cmap='hsv')
p2=gausscontour(cov2,m2,xx,yy)
plt.contour(xx,yy,p2,cmap='hsv')
# correct x1
plt.plot(x1[0,np.where(l1)],x1[1,np.where(l1)],'bo')
# wrong x1
plt.plot(x1[0,np.where(~l1)],x1[1,np.where(~l1)],'ro')
# correct x2
plt.plot(x2[0,np.where(1-l2)],x2[1,np.where(1-l2)],'r^')
# wrong x2
plt.plot(x2[0,np.where(l2)],x2[1,np.where(l2)],'b^')
```

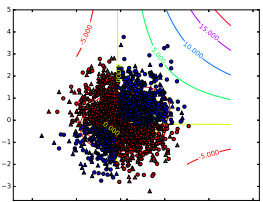
Assignment 1 (cov_diag.py)

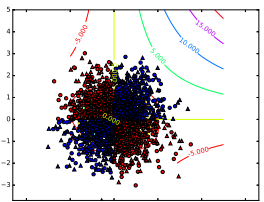
```
xxyy = np.c_[np.reshape(xx, -1), np.reshape(yy, -1)].T
pp = w.T.dot(xxyy - x0 * np.ones([1, xxyy.shape[1]]))
pp = np.reshape(pp, xx.shape)
cs = plt.contour(xx, yy, pp, cmap='hsv')
plt.clabel(cs)
# plt.savefig('cov_diag.eps')
plt.show()
```

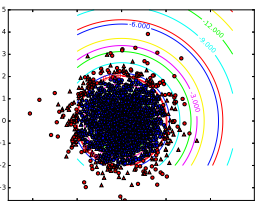
Assignment 1





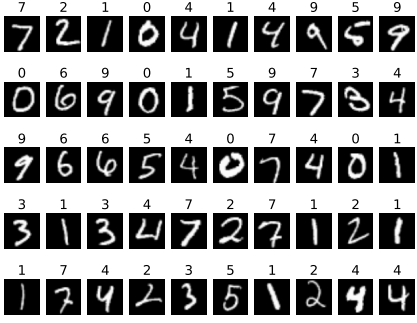






The MNIST Data

The MNIST data is consisted of images of handwritten digits with groundtruth.
Each image is the size of 28 pixel by 28 pixel with gray scale value for each pixel.
Training data: 60,000 images, Test data: 10,000 images.



MNIST preparation

Download MNIST data from `http://yann.lecun.com/exdb/mnist/`.

train-images-idx3-ubyte.gz	training set images	(9912422 bytes)
train-labels-idx1-ubyte.gz	training set labels	(28881 bytes)
t10k-images-idx3-ubyte.gz	test set images	(1648877 bytes)
t10k-labels-idx1-ubyte.gz	test set labels	(4542 bytes)

mnread.py is prepared (find in the program package).
To properly use mnread.py prepare 'mnist' directory under your working directory and put MNIST data there.

mnread.readim : read MNIST images
mnread.readlabel : read MNIST labels

Assignment 2 (MNIST)

Train classifiers using MNIST training data and classify MNIST test data.
Assume each image of is 784 ($= 28 \times 28$) dimensional vector.
Further assume that each digit follows normal distribution:

Case 1 $\Sigma_i = \sigma^2 I$

Case 2 $\Sigma_i = \Sigma$

Case 3 $\Sigma_i =$ arbitrary

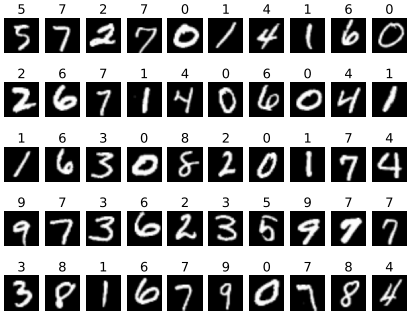
Estimate parameters of normal distributions by MNIST training data.
Classify test data by discriminant functions using the estimated parameters.

Intrinsic problems? Consider regularization, pseudo inverse, etc.

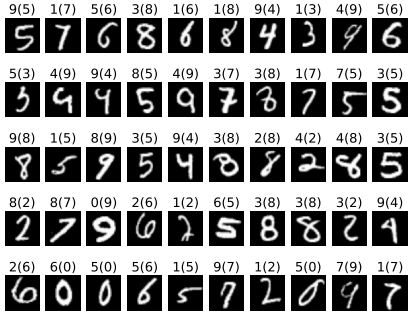
Assignment 2 (mntest.py)

Classification results assuming identical isotropic normal distributions ($\Sigma_i = \sigma^2 I$): reported accuracy is 82%.

goods



bads



Assignment 2 (mntest.py)

```
import numpy as np
import matplotlib.pyplot as plt
import mnread


def train(label, data):
    data = np.reshape(data, [data.shape[0], -1])
    lset = set(label)
    model = np.empty((len(lset), data.shape[1]), dtype=float)
    for x in lset:
        model[x, :] = np.mean(data[np.where(label == x), :], axis=1)
    return model


def classify(data, model):
    data = np.reshape(data, [data.shape[0], -1])
    label = np.empty(data.shape[0], dtype=int)
    for i in range(data.shape[0]):
```

Assignment 2 (mntest.py)

```
trlabel = mnread.readlabel(mnread.trlabelfz)
trdata = mnread.readim(mnread.trdatafz)
tstlabel = mnread.readlabel(mnread.tstlabelfz)
tstdata = mnread.readim(mnread.tstdatafz)

model = train(trlabel, trdata)
estlabel = classify(tstdata, model)
print('accuracy: %g' % (float(sum(estlabel == tstlabel)) / len(tstlabel)))
```

Assignment 2 (mntest.py)

```
plt.figure()
plt.suptitle('goods')
goods = np.random.permutation(np.where(estlabel == tstlabel)[-1])[range(50)]
for i, good in enumerate(goods):
    plt.subplot(5, 10, i + 1)
    plt.axis('off')
    plt.imshow(tstdata[good, :, :], cmap='gray')
    plt.title(estlabel[good])
plt.savefig('good.eps')
plt.figure()
plt.suptitle('bads')
bads = np.random.permutation(np.where(~(estlabel == tstlabel))[-1])[range(50)]
for i, bad in enumerate(bads):
    plt.subplot(5, 10, i + 1)
    plt.axis('off')
    plt.imshow(tstdata[bad, :, :], cmap='gray')
    plt.title('%s(%s)' % (estlabel[bad], tstlabel[bad]))
plt.savefig('bad.eps')
```

Hint of Assignment 2

Case 1 As mntest.py shows, simply compute mean vector for each digit in training, and the classify each digit to the class of the nearest mean.

Case 2 As in case 1, you compute mean vector for each digit in training.
Then in computing common covariance matrix, compute residuals from corresponding means then compute covariance matrix.

Case 3 Rather simple: compute mean and covariance matrix for each digit.
95% accuracy was achieved in case 3.

Hint of Assignment 2

You might face difficulty in obtaining the inverse of covariance matrices because of the singularity.
One reason is because some locations have zero intensities for all training data.
This can be remedied by using generalized inverse (aka pseudoinverse, Moore-Penrose pseudoinverse).
Available as Matlab pinv, Python numpy.linalg.pinv.
Inverse:

$$
\begin{aligned}
\Sigma\Phi &= \Phi\Lambda \text{ (eigenvalue decomposition of } \Sigma) \\
\Sigma &= \Phi\Lambda\Phi^T \\
&= \sum \lambda_i \phi_i \phi_i^T \\
\Sigma^{-1} &= \sum \frac{1}{\lambda_i} \phi_i \phi_i^T
\end{aligned}
$$

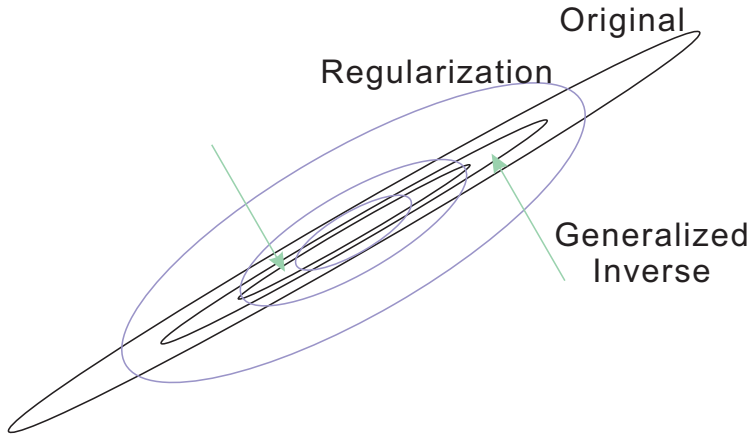
Generalized inverse ignores all (close to) zero eigenvalues in obtaining the inverse.
This equivalently ignores all-zero pixels (and linearly dependent pixels if any).

Hint of Assignment 2

You might also consider regularization.
Regularization: Instead of computing Σ^{-1} when Σ is singular, compute $(\Sigma + \alpha I)^{-1}$ with some constant α.
How it works?

$$
\begin{aligned}
(\Sigma + \alpha I)^{-1} &= (\Phi \Lambda \Phi^T + \alpha I)^{-1} \\
&= [\Phi \Lambda \Phi^T + \Phi(\alpha I)\Phi^T]^{-1} \\
&= [\Phi(\Lambda + \alpha I)\Phi^T]^{-1} \\
&= [\sum(\lambda_i + \alpha)\phi_i \phi_i^T]^{-1} \\
&= \sum \frac{1}{\lambda_i + \alpha} \phi_i \phi_i^T
\end{aligned}
$$

Hint of Assignment 2



Hint of Assignment 2

In the case when Σ_i are singular, the handling of $\log |\Sigma_i|$ should also be considered.
Please consider why this term is needed.
Then you may know how to modify this term.