Contents

Introduction

Chapter 1

The Wishart Distribution

1.1 Definition

The *wishart distribution* $W_p(n, \Sigma)$ is a probability distribution of random nonnegative-definite $p \times p$ matrices that is used to model random covariance matrices.

The parameter n is the number of degrees of freedom, and Σ is a nonnegative-definite symmetric $p \times p$ matrix, called the *scale matrix.*

Def. Let $X_1 ... X_n$ be $N_p(0, \Sigma)$ distribuited vectors, forming a data matrix $p \times n$, $X = [X_1 ... X_n]$. The distribution of a $p \times p$, $M = XX' = \Sigma_{i=1}^n X_i X_i'$ *random matrix* is a Wishart distribution. [1]

We have then by definition:

$$M \sim W_p(n, \Sigma) \sim \Sigma_{i=1}^n X_i X_i' \qquad X_i \sim N_p(0, \Sigma) \tag{1.1}$$

so that $M \sim W_p(n, \Sigma)$ is the distribution of a sum of n rank-one matrices defined by independent normal $X_i \in R^p$ with $E(X) = 0$ and $Cov(X) = \Sigma$.

In particular, it holds for the present case:

$$E(M) = nE(X_i X_i') = nCov(X_i) = n\Sigma \tag{1.2}$$

1.2 PDF Computation for Invertible Σ

In general, any $X \sim N(\mu, \Sigma)$ can be represented as

$$X = \mu + AZ, \quad Z \sim N(0, I_p) \tag{1.3}$$

so that

$$\Sigma = Cov(X) = ACov(Z)A' = AA' \tag{1.4}$$

The easiest way to find A in terms of Σ is the LU-decomposition, which finds a unique lower diagonal matrix A with $A_{ii} \geqslant 0$ such that $AA' = \Sigma$.

Then by 1.1 and 1.4, with $\mu = 0$ we have:

$$W_p(n, \Sigma) \sim \sum_{i=1}^{n} (AZ_i)(AZ_i)' \sim A(\sum_{i=1}^{n} Z_i Z_i')A' \sim AW_p(n)A' \qquad (1.5)$$

where $Z_i \sim N(0, I_p)$ and $W_p(n) = W_p(I_p, n)$.

Assuming that $n \geq p$ and Σ is invertible, the density of the random $p \times p$ matrix M in 1.1 can be written [1] :

$$f(M, n, \Sigma) = \frac{1}{2^{\frac{np}{2}} \Gamma_p(\frac{n}{2}) \|\Sigma\|^{\frac{n}{2}}} \|M\|^{\frac{n-p-1}{2}} exp[-\frac{1}{2} trace(\Sigma^{-1} M)] \qquad (1.6)$$

so that $f(M, n, \Sigma) = 0$ unless M is *symmetric and positive-definite.* [2]

Note that in 1.6 we define $\Gamma_p(\alpha)$ as the *generalized gamma function* $\Gamma_p(\alpha) = \pi^{\frac{p(p-1)}{4}} \prod_{i=1}^{p} \Gamma(\frac{2\alpha+1-i}{2})$

1.2.1 Visualizing the Wishart Distribution

The Wishart distribution is a generalization to multiple dimensions of the *chi-squared distribution*, or in the case of non-integer degrees of freedom, of the *gamma distribution.*

We show as a proof in fig.1.1 that for a 1-dimensional and equal to 1 Σ scale matrix, the Wishart distribution $W_1(n, 1)$ collapses to the $\chi^2(n)$ distribution.

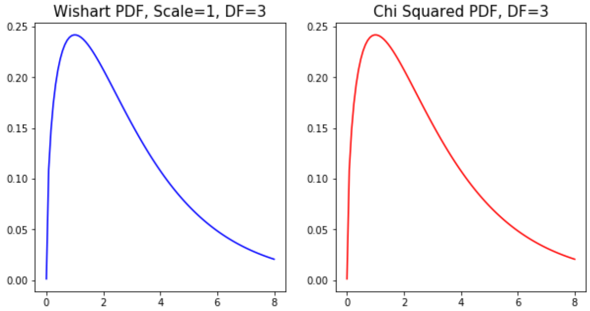

Figure 1.1: *Monodimensional Wishart Distribution and $\chi^2(n)$ distribution comparison*

[1] *Note:* $\|\Sigma, N\| = det(\Sigma, M)$

Save for this simple case, being the Wishart a distribution over matrices, it is a generally hard task to visualize it as a density function.

We can however sample from it and use the eigenvectors and eigenvalues of the resulting sampled matrix to define an ellipse.

An example of this technique is shown in fig.1.2. A set of five sampled matrices is drawn for each plot. While the parameter $n = 2$ (*degrees of freedom*) is the same for both the samplings shown, a different *scale matrix* Σ is used for each plot.

Note that for $\Sigma = I_2$ (left plot in fig.1.2) the sample would look *on average* like circles.

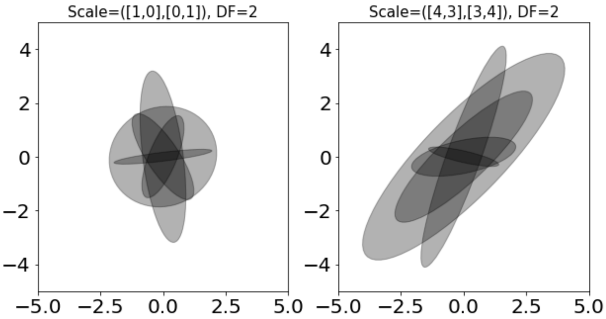

Figure 1.2: *Plot of eigenvalue and eigenvectors defined ellipses, drawn from different scale matrix defined Wishart-sampled distribution.*

1.3 The Wishart Distribution in Bayesian Conjugate Prior Analysis

An important use of the Wishart distribution is as a conjugate prior for *multivariate normal sampling.* We now recall some basics concepts about Bayesian inference and prediction in order to show the application of the Wishart in those fields.

1.3.1 Bayesian Inference and Priors Distributions

The distinctive feature of the Bayesian approach underlies in its way of defining probability.

Probability is treated as *belief* and not as *frequency*, thus introducing a fundamental difference between the Bayesian and the *frequentist* approach and shifting the goal toward the analysis and statement of a *belief* [3].

We can sum up the process of Bayesian inference as follows:

- A probability density called *prior distribution* $\pi(\theta)$ is chosen, expressing the *beliefs* about a parameter θ before any data are seen.
- A statistical model $p(x \mid \theta)$ is chosen, which must reflect the beliefs about x given θ.
- After observing the data $D_n = [X_1 ... X_n]$, the beliefs is updated and the *posterior distribution* $p(\theta \mid D_n)$ is computed.

By Bayes' theorem the posterior distribution can be written as

$$p(\theta \mid X_1 ... X_n) = \frac{p(X_1 ... X_n \mid \theta)\pi(\theta)}{p(X_1 ... X_n)} = \frac{L_n(\theta)\pi(\theta)}{c_n} \propto L_n(\theta)\pi(\theta) \qquad (1.7)$$

where $L_n(\theta) = \prod_{i=1}^{n} p(X_i \mid \theta)$ is the likelihood function and the *normalizing constant* c_n is defined as follows:

$$c_n = p(X_1 ... X_n) = \int p(X_1 ... X_n \mid \theta)\pi(\theta) d\theta = \int L_n(\theta)\pi(\theta) d\theta \qquad (1.8)$$

the normalizing constant is also called the *evidence.*

We now define the general properties of a *conjugate prior.*

If, for a given problem, the posterior distribution $p(\theta \mid X_1 ... X_n)$ and the prior $\pi(\theta)$ belong to the same family of distribution, they're called *conjugated distributions* and the prior is said to be a *conjugate prior* for the given *likelihood function* $L_n = p(X_1 ... X_n \mid \theta)$.

A classical example concerns the Gaussian Distribution: the Gaussian family is conjugate to itself (or self-conjugate) with respect to a Gaussian likelihood function: if the likelihood function is Gaussian, choosing a Gaussian prior over the mean will ensure that the posterior distribution is also Gaussian [4].

Considering the general problem of inferring a distribution for a parameter θ given some observations $D_n = [X_1 ... X_n]$ and referring to theorem 1.7, by which we let the likelihood function be considered fixed as it is usually well-determined from a statement of the data-generating process, it is clear that different choices of the prior distribution $\pi(\theta)$ may make the integral in 1.8 more or less difficult to compute. The product $L_n(\theta)\pi(\theta)$ will also be influenced, gaining the possibility to take one algebraic form or another.

If for certain choices of the prior the posterior has the same algebraic form as the prior, those choices are said to yield a *conjugate prior.*

It is then possible to state that a conjugate prior is an algebraic convenience giving a closed-under-sampling-form expression for the posterior.

1.3.2 The Wishart Conjugate Prior

We now show how the Wishart Distribution is correlated to the*Inverse Gamma Distribution* in a multidimensional setting, by considering a Gaussian model with known mean μ, so that the free parameter is the variance σ^2, as in [3].

The likelihood function is defined as follows:

$$p(X_1...X_n \mid \sigma^2) \propto (\sigma^2)^{-\frac{n}{2}} exp(-\frac{1}{2\sigma^2} n\overline{(X-\mu^2)}), \qquad \overline{(X-\mu^2)} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 \tag{1.9}$$

The conjugate prior is an inverse Gamma distribution. Recall that θ has an inverse Gamma distribution with parameters (α, β) when $\frac{1}{\theta} \sim Gamma(\alpha, \beta)$.

The density is then bound to take the form

$$\pi_{\alpha,\beta}(\theta) \propto \theta^{-(\alpha+1)} e^{-\frac{\beta}{\theta}} \tag{1.10}$$

Using this prior, the posterior distribution of σ^2 is given by

$$p(\sigma^2 \mid X_1...X_n) \sim InvGamma(\alpha + \frac{n}{2}, \beta + \frac{n}{2}\overline{(X-\mu^2)}) \tag{1.11}$$

An alternative way of parameterization of the prior is given by the *Inverse Scaled* χ^2 *Distribution*, whose density is defined as

$$\pi_{\nu_0,\sigma_0^2} \propto \theta^{-(1+\frac{n_0}{2})} exp(-\frac{\nu_0\sigma_0^2}{2\theta}) \tag{1.12}$$

Under this kind of parameterization of the prior, the posterior takes the form

$$p(\sigma^2 \mid X_1...X_n) \sim ScaledInv\chi^2(\nu_0 + n, \frac{\nu_0\sigma_0^2}{\nu_0 + n} + \frac{n(X-\mu^2)}{\nu_0 + n}) \tag{1.13}$$

In the multidimensional setting, the inverse Wishart takes the place of the inverse Gamma.

It has already been stated that the Wishart distribution is a distribution over *symmetric positive semi-definite* $d \times d$ matrices W. A more compact form of the density is given by

$$\pi_{\nu_0,S_0}(W) \propto \mid W \mid^{\frac{(\nu_0-d-1)}{2}} exp(-\frac{1}{2} trace(S_0^{-1}W)), \qquad \mid W \mid = det(W) \tag{1.14}$$

where the parameters are the degrees of freedom ν_0 and the positive-definite *scale matrix* S_0.

If $W^{-1} \sim Wishart(\nu_0, S_0)$ we can then state that W has an *Inverse Wishart Distribution*, whose density has the form

$$\pi_{\nu_0,S_0}(W) \propto \mid W \mid^{-\frac{(\nu_0+d+1)}{2}} exp(-\frac{1}{2}trace(S_0W^{-1})), \qquad \mid W \mid = det(W) \tag{1.15}$$

Let $X_1...X_n$ be $N(0,\Sigma)$ distributed observed data. Then an inverse Wishart prior multiplying the likelihood $p(X_1...X_n \mid \Sigma)$ yields

$$p(X_1...X_n \mid \Sigma)\pi_{\nu_0,S_0}(\Sigma) \propto$$

$$\mid \Sigma \mid^{-\frac{n}{2}} exp(-\frac{n}{2}trace(\overline{S}\Sigma^{-1}) \mid \Sigma \mid^{-\frac{(\nu_0+d+1)}{2}} exp(-\frac{1}{2}trace(S_0\Sigma^{-1})) \tag{1.16}$$

$$= \mid \Sigma \mid^{-\frac{(\nu_0+d+n+1)}{2}} exp(-\frac{1}{2}trace((n\overline{S}+S_0)\Sigma^{-1}))$$

where $\overline{S}$ is the *empirical covariance* $\overline{S} = \frac{1}{n}\sum_{i=1}^{n} X_iX_i^T$.

Thus, a posterior with the form

$$p(\Sigma \mid X_1...X_n) \sim InvWishart(\nu_0 + n, n\overline{S} + S_0) \tag{1.17}$$

is obtained.

Analogally, it can be stated that for the inverse covariance (*precision*) matrix Σ^{-1} the conjugate prior is a Wishart distribution.

Chapter 2

The WISDoM Multiple Order Classification

In this section, the classification method implemented and used on the ADNI2 and ABIDE databases is described, both in an analytical and technical way.

The Wishart Distributed Matrices Multiple Order Classification is a method that allows both classification and *feature selection* for any classification problem whose elements can be tied to a *symmetric positive-definite* matrix representation (i.e. covariance and correlation matrices).

The *"distance"* used to train the classifier is defined as well as the *feature transformation* undergone by the each of the subject analyzed.

The general pipeline and the validation pipeline are then discussed while also introducing an example of possible parallelization for performance enhancing.

2.1 Wishart Sampling

Considering what has been said in the last section, using the Wishart distribution to model and sample the elements of a wide range of problems follows naturally.

As a matter of fact, every calssification problem whose elements take the form of *symmetric positive-definite* matrices can be approached with the method we are about to discuss.

The main idea for the *WISDoM Classifier* is to use the *free parameters* of the Wishart distribution (the scale matrix S_0 and the number n of the degree of freedom, as shown in 1.6) to compute an estimation of the distribution for a certain class of elements, and then assign a single element to a given class by computing some sort of distance between the element being analyzed and the classes.

Furhermore, if we assume that the matrices are somehow representative of the *features* of the system studied (i.e. covariance matrices might be taken into account), a score can be assigned to each feature by estimating the weight of said feature in terms of *Log Likelihood Ratio*.

In other words, a score can be assigned to each feature by analyzing the variation in terms of *LogLikelihood* caused by the deletion of it.

If the deletion of a feature causes significant increase (or decrease) in the *LogLikelihood* computed with respect to the *estimated distributions* for the classes, it can be stated that said feature is highly representative of the system analyzed.

It is now clear that the simplest usable objects to estimate the distribution for a class and to represent its elements is the *covariance matrix*. Further proofs for this statement will be given later on.

Thus, the aim of the WISDoM classifier is not only to assign a given element to the optimal class, but also to identify the features with the highest *"weights"* in the decision process.

2.1.1 Computing the Estimated Distribution

Let us briefly recall the parametrization of the Wishart Distribution in order to clearly define the application conditions for classification problems.

Let $X_1 ... X_n$ be independent $N_p(0, \Sigma)$ distributed vectors, forming a data matrix $p \times n$, $X = [X_1 ... X_n]$. The distribution of a $p \times p$, $M = XX' = \Sigma_{i=1}^{n} X_i X_i'$ *random matrix* is a Wishart distribution with parameters $W_p(n, S_0)$. In the previous chapter (1.6) it has been proved that for normal distributed data, for $S_0 = \Sigma$, a distribution of *random covariance matrices* is obtained.

In a similar fashion, if a good choice for the scale matrix S_0 is made for a given class, a representative distribution for the class can be estimated and samples can be drawn from it.

Covariance matrices are a good choice, although not limiting as long as the matrices are symmetric and positive-definite, both for the way they represent a system and for the property that *the mean of a set of covariance matrices is a covariance matrix.*

If each element of a given class C is represented by a covariance matrix Σ of its features, this property allows us to estimate a distribution for the class by choosing

$$S_0 = \hat{\Sigma}_C = \frac{1}{N}\sum_{i=1}^{N}\Sigma_i \tag{2.1}$$

The other necessary parameter for the estimation is the *degrees of freedom* n.

Assume that an $X_i = (x_1, ..., x_p)$ vector of p features is associated to each element i of a given class, while having n observation of said vector. The covariance matrix Σ_i computed over the n observations will represent the "interactions" between the features of element i.

The degrees of freedom n of the Wishart distribution are then given by the number of times X_i is observed.

Let us introduce an example tied to *functional MR brain imaging* in order to further clarify the concepts being introduced.

An image of patient i's brain is acquired; as usual these images are divided in a certain number p of zones (voxel, pixel etc.), each zone being sampled n times over a given time interval in order to observe a certain type of brain activity and functionality.

It is now clear that the features contained in vector $X_i = (x_1, .., x_p)$ associated to patient i are indeed the zones chosen to divide i's brain image, each zone having been sampled n times during an acquisition interval.

The correlation $p \times p$ matrix Σ_i computed for i's observation is then representative of the functional correlation between the p zones of i's brain.

Repeating this procedure for N patients belonging to a known class C (i.e. a diagnostic group) and computing the $\hat{\Sigma}_C$ scale matrix for the class as stated before, will allow us to estimate a wishart distribution for that class correlation matrices and draw samples from it.

The module used for Wishart generation and sampling by the WISDoM calssifier is the *SciPy.Stats.Wishart* module of the *SciPy* Python3.6 library.

Further details on the generation and sampling algorithm used by the module can be found in [5].

Some samples drawn from Wishart distributions computed with different 5×5 scale matrices and degrees of freedom are shown in fig.2.1.

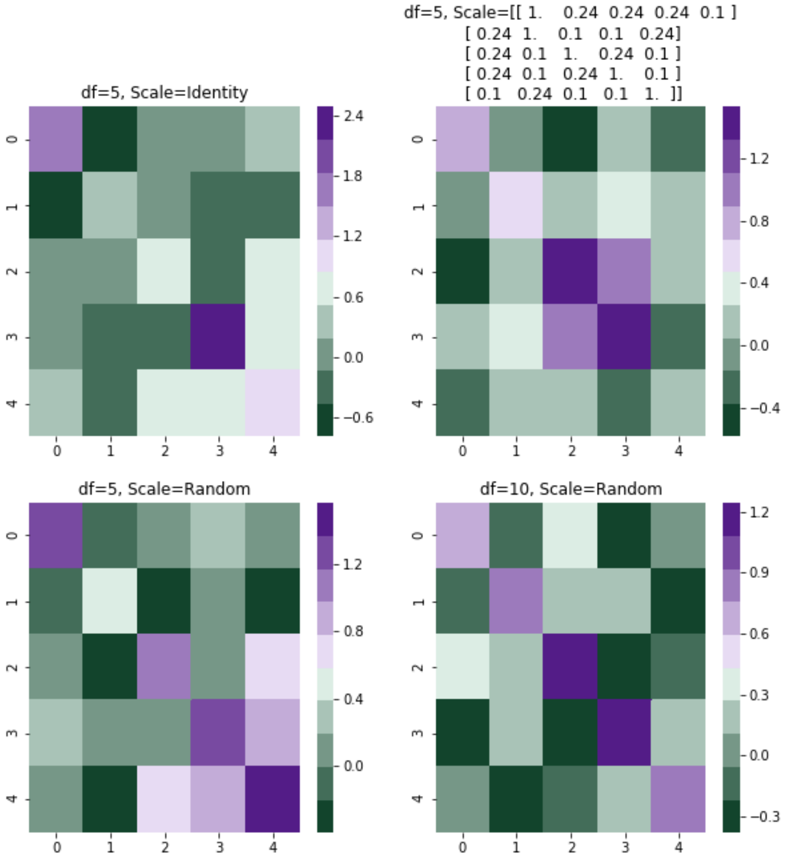

Figure 2.1: *Various sampling from different wishart distribution. A diverging heatmap has been chosen to visualize the values of each sample's elements.*

2.2 Log-Likelihood Ratio Distance

After the definition of the role of the Wishart distribution in symmetric positive definite matrices' modeling, it is necessary to define some sort of distance between the estimated distribution for a class C and its hypotetical elements.

As stated before, this will be done in terms of both entire matrices and *single features*, in order to achieve optimal classification and exract information about a system's most meaningful components.

2.2.1 Complete Matrix Distance

The scoring system used by the WISDoM Classifier relies on the *logpdf* function from the *SciPy.Stats.Wishart* module in order to compute the LogLikelihood of a matrix Σ_i with respect to the Wishart distribution estimated for a class C, using $\hat{\Sigma}_C$ as the scale marix.

If a problem concering two given classes C_A and C_B is taken into account, the score assigned to each Σ_i upon which the classification decision is based, can be defined as follows:

$$score_i = logP_W(\Sigma_i \mid n, \hat{\Sigma}_A) - logP_W(\Sigma_i \mid n, \hat{\Sigma}_B) \tag{2.2}$$

Where $\hat{\Sigma}_{A,B}$ are the scale matrix computed for the classes A, B and $logP_W(\Sigma_i \mid n, \hat{\Sigma}_{A,B})$ can be seen as the logarithm of the probability of Σ_i belonging to the Wishart distribution estimated for one of the two classes A, B.

2.2.2 Single Feature Distance and Multiple Order Reduction

The aim of the WISDoM classifier is to further increase the informations obtained about the system's features during the classification.

To do this it is then necessary to introduce some matemathical properties of the symmetric positive deifnite matrices, upon which the method relies.

It will be shown that it is indeed possible to access different orders of information by scaling a matrix A to its *principal submatrices.*

Def. *Let A be an $n \times n$ matrix. A $k \times k$ submatrix of A formed by deleting $n-k$ rows of A, and the same $n-k$ columns of A, is called principal submatrix of A. The determinant of a principal submatrix of A is called a principal minor of A.*

Note that the definition does not specify which $n - k$ rows and columns to delete, only that their indices must be the same.

Let us introduce a 3×3 example.

For a general matrix $A_{3\times3}$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.3}$$

there are three *first order principal minors*:

- $\mid a_{11} \mid$ formed by deleting the last two rows and columns
- $\mid a_{22} \mid$ formed by deleting the first and third rows and columns
- $\mid a_{33} \mid$ formed by deleting the first two rows and columns

There are three *second order principal minors*:

- $\mid \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \mid$ formed by deleting column 3 and row 3
- $\mid \begin{bmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{bmatrix} \mid$ formed by deleting column 2 and row 2
- $\mid \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix} \mid$ formed by deleting column 1 and row 1

There's one *third order principal minor*, namely $\mid A \mid$.

For the sake of completion, we also recall the following definition.

Def. *Let A by an $n \times n$ matrix. The k^{th} order principal sub-matrix of A obtained by deleting the* ***last*** *$n - k$ rows and columns of A is called the k^{th} order* ***leading principal submatrix*** *of A, and its determinant is called the k^th* ***order leading principal minor*** *of A.*

An imporant property for the principal submatrices of a symmetric positive definite matrix is that *any $(n - k) \times (n - k)$ partition is also symmetric and positive definite.*

It is now clear that such properties can be used to reduce both a class scale matrix $\hat{\Sigma}_C$ and any Σ_i matrix, in order to study its deviation from a class's estimated Wishart distribution derived from the deletion of one of its components (the features conatined in vector $X_i = (x_1, ..., x_p)$ from which the matrix $\Sigma_{i,p\times p}$ is computed).

Iterating this process over all the features, or in other terms analyzing all of the $(p - 1) \times (p - 1)$ principal submatrices of Σ_i and $\hat{\Sigma}_C$, will allow us to assign a score to each feature, representing its weight in the decision for Σ_i to be assigned to one class or another.

Note that for such an order of principal submatrices, the process will reduce the $\Sigma_{i,p\times p}$ matrix to a *score vector* of length p for each element i undergoing the classification.

Let us now introduce the following notation in order to define the score assigned for each of the x_p features of the vector $X_i = (x_1..., x_p)$.

Let Σ_j be a principal submatrix of order $(p-1)$, of the matrix Σ computed on the observation of $X_i = (x_1, ..., x_p)$ for subject i, *obtained by the deletion of the j^{th} row and the j^{th} column*, with $1 \leq j \leq p$.

Let $\hat{\Sigma}_{Cj}$ be a principal submatrix of order $(p-1)$, of the matrix $\hat{\Sigma}_C$ computed for the class C *obtained by the deletion of the j^{th} row and the j^{th} column*, with $1 \leq j \leq p$.

The score assigned to each feature of $X_i = (x_1, ..., x_p)$ is then given by eq.(2.4).

$$Score_j(C) = \Delta logP_{Wj}(C) = logP_W(\Sigma, n \mid \hat{\Sigma}_C, n) - logP_W(\Sigma_j, n \mid \hat{\Sigma}_{Cj}, n) \tag{2.4}$$

In other terms, each partition Σ_j represents the matrix Σ without the elements tied to feature x_j (the elements in row j and column j of Σ). Computing the variation in terms of log-likehood between the estimated wishart distribution for the class and the estimated wishart distribution for the class without component j, allows us to gain informations about which feature weighs more on both subject i's cassification and the general system structure.

Note that this kind of scoring is *class-dependent*. Computing this score vector with respect to all the classes $C_1..C_n$ of a given problem and performing some sort of score ratio will allow the subject i, after a suitable *training*, to be assigned to the most likely of the classes while retaining informations on which features are the most determinant, decision wise.

Let us introduce a *2-classes* example in order to show how this kind of result might be obtained.

Let C_1 and C_2 be the two classes of a given problem.

Let a set of N matrices Σ_i be a set of correlation matrices computed for N subjects i whose class is known.

Let $\hat{\Sigma}_{C1}$ and $\hat{\Sigma}_{C2}$ be the scale matrices computed as seen in eq.(2.1), used to estimate the Whishart distribution for each one of the two classes C_1 and C_2, and $\hat{\Sigma}_{C1j}$ and $\hat{\Sigma}_{C2j}$ their $(p-1)$ order partitions, as in eq.(2.4).

If from each matrix Σ_i the score vector is computed as in eq.(2.4) with respect to each one of the two classes C_1, C_2, an *inter-class log-likelihood ratio* *vector* can be obtained by assigning to each feature a score defined as follows:

$$Ratio_j = \Delta log P_{Wj}(C_1) - \Delta log P_{Wj}(C_2) \tag{2.5}$$

Training a classificator on a set of N subjects whose classes are known, after each matrix Σ_i (and as a consequence each feature vector X_i) has undergone the transformations defined in eq.(2.4) and (2.5), yields a significant improvement in performance for certain classes of problems, as it will be shown later.

A new subject will be, as a matter of fact, classified according to its *transformed ratio vector* given by eq.(2.5), thus simultanesously retaining information about its class's most significant features: *the score assigned to each feature is a measure of how much the deletion of said feature weighs, in terms of log-likelihood variation, on the decision to assign each matrix* Σ_i *to one class or another.*

The entire process can be seen as a *feature transformation*, which leads to a *feature selection*, whose effect is, for certain types of problems, to enhance the classification performance.

2.2.3 Generalizing to $(p - n)$ Order Transformations

As seen in the last section, transforming all the $(p-1) \times (p-1)$ principal submatrices of Σ_i by eq.(2.4), yields a vector of score of length p for each element i.

Anyway, for any $n < p$, a number of principal submatrices of Σ_i can be obtained.

These kind of submatrices can be used to gain informations about the weight of n simoultaneously deleted features on the system structure and classification.

Let us introduce an example for $(p-2)$ order submatrices.

Let Σ_{jk} be a principal submatrix of order $(p-2)$, of the matrix Σ_i computed on the observation of $X_i = (x_1, ..., x_p)$ for subject i, *obtained by the deletion of the* j^{th} *row and the* j^{th} *column and the* k^{th} *row and the* k^{th} *column* , with $1 \leq j, k \leq p$.

Let $\hat{\Sigma}_{Cjk}$ be a principal submatrix of order $(p-2)$, of the matrix $\hat{\Sigma}_{Cjk}$ computed for the class C *obtained by the deletion of the* j^{th} *row and the* j^{th} *column and the* k^{th} *row and the* k^{th} *column* , with $1 \leq j, k \leq p$.

Then, eq.(2.4) becomes:

$$Score_{jk}(C) = \Delta logP_{Wjk}(C) = logP_W(\Sigma, n \mid \hat{\Sigma}_C, n) - logP_W(\Sigma_{jk}, n \mid \hat{\Sigma}_{Cjk}, n) \tag{2.6}$$

in this case, a score is assigned to each coupling of the features j, k, and transformation (2.6) will yield not a vector, but a $p \times p$ matrix with diagonal elements equals to the scores obtained by (2.4), being the iteration with $j = k$ the coupling the j^{th} feature with itself.

Non-diagonal elements represent the score of the coupling of feature j with feature k.

2.3 Pipeline

In this section we discuss each step of the feature transformation and classification process.

Given the recursive nature of the method just described, a crucial issue concerning computational time is the strong dependence between it and the analyzed matrix size.

A rough visualization of the entity of such a dependence can be found in fig.(2.2)

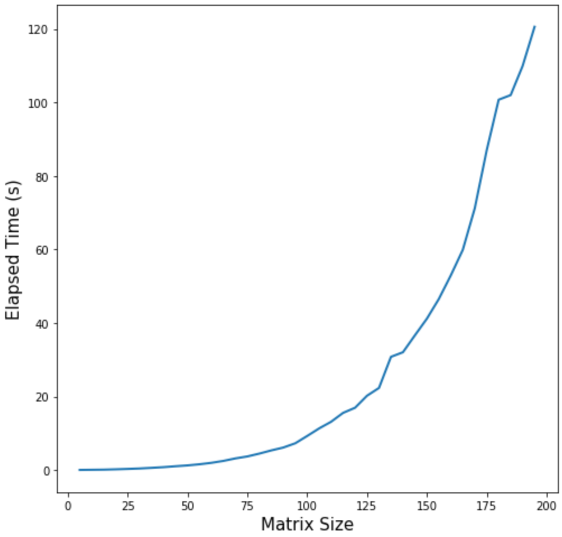

Figure 2.2: *Matrix Size dependence for* $(p-2)$ *order transformations.*

Iterating the $(p-1)$ order transformation described in eq.(2.4) over a large N of observations of size $p \times p$ of a given database, while introducing some kind of *cross-validation* routine may lead to abysmal computational time-wise performances.

A possible solution to this problem is to introduce a high level of automatization for each step, followed by the indroduction of a highly parallelizable overall structure of the pipeline.

2.3.1 The Snakemake Environment

The main tool used to achieve such results is the *Snakemake Workflow Management System*, described in [6], a Python-based interface created to build reproducible and scalable data analyses and machine-learning routines.

To briefly some up the the advantages of using such tools and structures, the *Snakemake Workflow* can be described as rules that denote how to create output files from input files. The workflow is implied by dependencies between the rules that arise from one rule needing an output file of another as an input file [6].

A rule definition specifies :

- *a name*, used by the main rule instance *rule all* and main execution environment for identification
- *any number of input and output files*; tipically one rule's output is another rule input, linking the rules alla the way up to main rule instance.
- *either a shell command or Python code*; containing the creation of the output from the input

Input and output files may contain multiple named *wildcards*, whose values are inferred automatically from the files desired by the user.

To further clarify the role of the wildcards, let us introduce a brief example.

Let's say that our aim is to train a classifier over two classes of elements C_1, C_2. The training part of the database is then divided in two files, each one containing the name of its elements' class in the filename.

Setting a rule to load these files while expecting a wildcard tied to the class name in the filename, will allow the entire set of rules of the pipeline to be executed automatically for class C_1 and class C_2.

Considering this example, the real power of the parallelization capabilities offered by the Snakemake environment are quite clear.

With a simple syntax, looking at the example just proposed, each one of two *cores* of a server where our hypothetical pipeline is running can be set to work indipendently on each subset of data belonging to class C_1 or class C_2.

Building a pipeline whose rules are easily iterable over a set of different *wildcards* will lead to natural and efficient parallelization and automation.

2.3.2 The WISDoM Pipeline

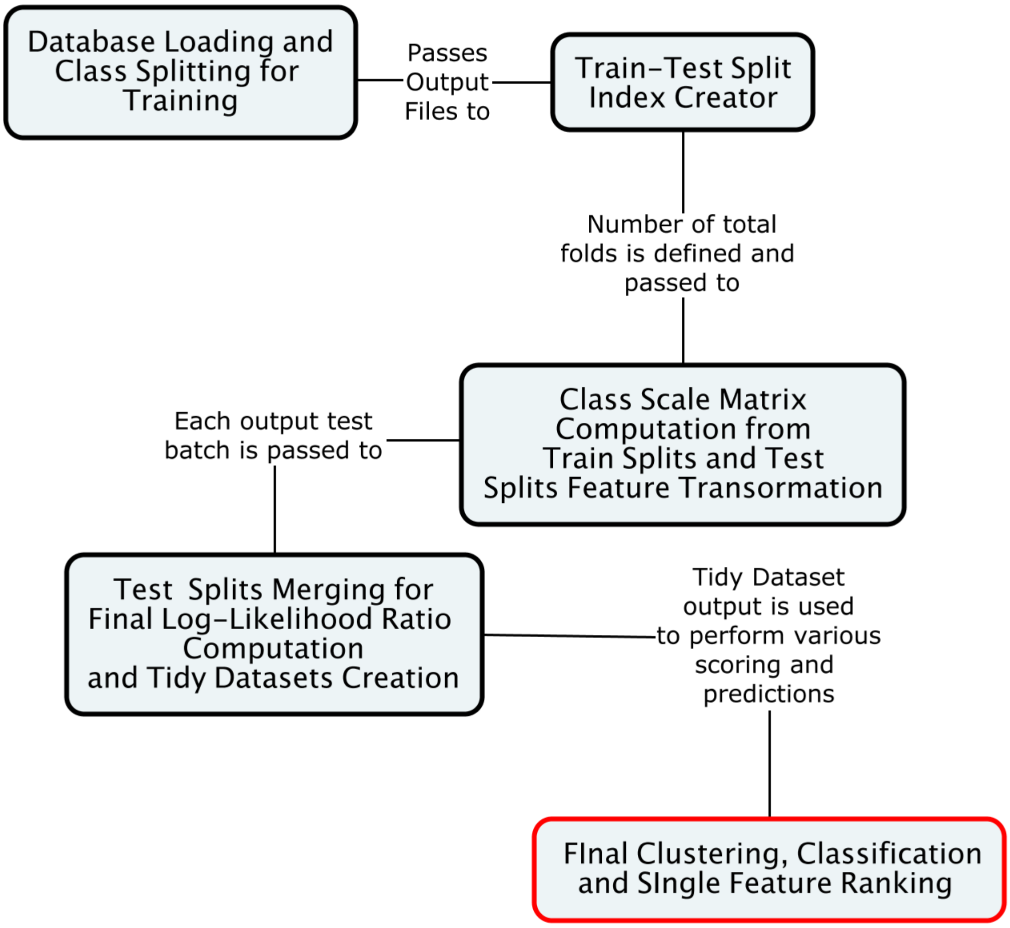
Database Loading and Class Splitting for Training
Passes Output Files to
Train–Test Split Index Creator
Number of total folds is defined and passed to
Class Scale Matrix Computation from Train Splits and Test Splits Feature Transormation
Each output test batch is passed to
Test Splits Merging for Final Log–Likelihood Ratio Computation and Tidy Datasets Creation
Tidy Dataset output is used to perform various scoring and predictions
FInal Clustering, Classification and SIngle Feature Ranking

Figure 2.3: *General pipeline workflow.*

In figure (2.3) the main steps required by the WISDoM Multiple Order Classification are reported.

We will now go through each step in detail in order to show how *train and test splitting* are interpreted for the WISDoM pipeline.

- **Database loading and class splitting**
  rules: case_wrap, seqs_store ; in this section of the pipeline, data are loaded and divided into the classes defined by the wildcards and main rule instance's inputs. An info sheet containing the classification labels for each observation is needed as an *input* for this step.

  In order to achieve fast reading/writing perfomances for big data, the matrices are stored as the sequence of elements belonging to the upper triangle for each matrix (in *.hdf* format). Being symmetric, the entire matrix can be easily reconstructed when needed. At this step, the files containing observations for each class are created.

- **Train-Test split index creator**
  rules: split_gen, tt_gen; in this section of the pipeline, each file containing one class's observations is divided into train-test batches. The total number of train-test folds is defined by wildcards and main rule instance's inputs.

  First, each dataset is divided into sections, then each section is further splitted intoa number of user-specified train-test folds.

- **Class Scale Matrix computation and feature transformation**
  rules: map_gen; this is the core section of the pipeline, where the features are transformed according to eq.(2.4).

  Train-test split files for each class are passed as inputs; the train sets of each batch are used to compute the scale matrix $\hat{\Sigma}_C$ as in eq.(2.1). The estimated Wishart for the class is then computed and the features of each test-set element are transformed.

  In order to compute the Ratio described in eq.(2.5), the above process is repeated for each class with respect to each other. A map containing each transformed feature in term of *quantiles* is also created.

- **Test splits merging and tidy datasets creation**
  rules: t_join, q_join; in the final step of the pipeline, all of the transformed feature test batches are merged into tidy datasets. This type of data sturcture will allow an easy computation of the ratio in eq.(2.5) for each feature; furhtemore, once such dataset is obtained, evertything needed for the transformed observations to undergo any classification pipeline and/or model selection is ready.

A graphic representation of the plan of rules execution can be obtained by using a *directed acyclic graph* (DAG), as shown in figure (2.4).

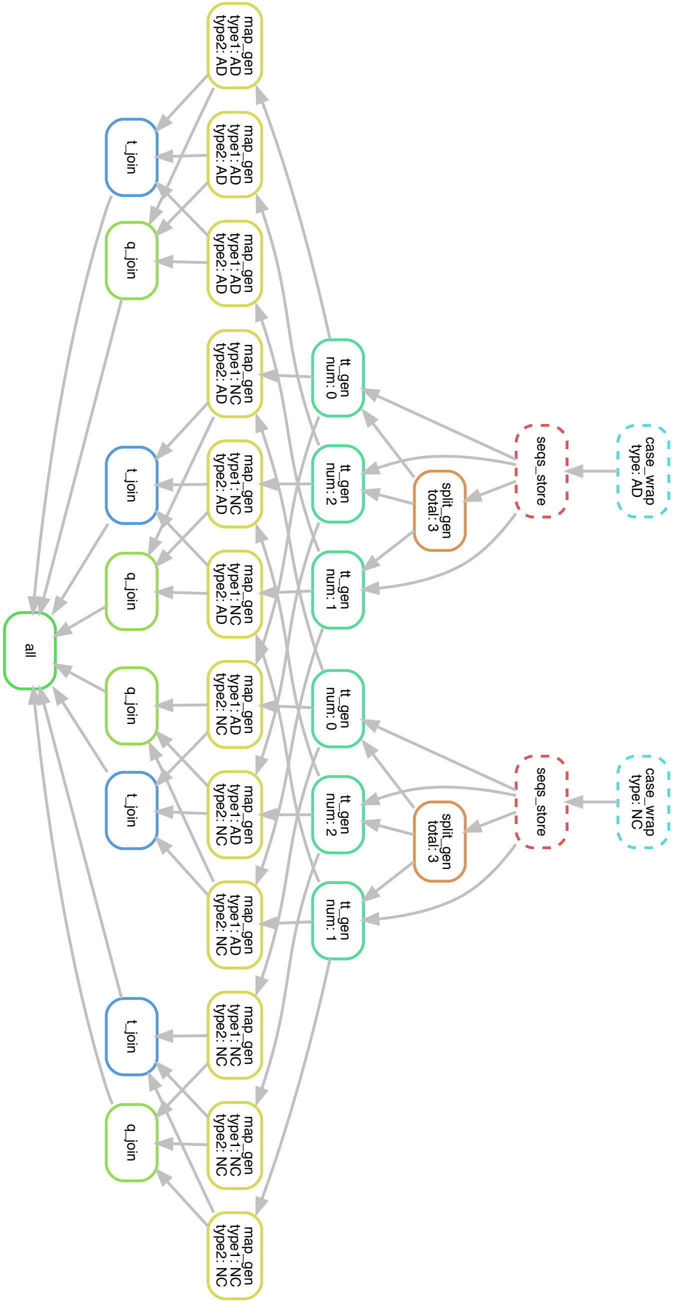
map_gen type1:AD type2:AD
map_gen type1:AD type2:AD
map_gen type1:AD type2:AD
map_gen type1:NC type2:AD
map_gen type1:NC type2:AD
map_gen type1:NC type2:AD
map_gen type1:AD type2:NC
map_gen type1:AD type2:NC
map_gen type1:AD type2:NC
map_gen type1:NC type2:NC
map_gen type1:NC type2:NC
map_gen type1:NC type2:NC
t_join
q_join
t_join
q_join
all
q_join
t_join
t_join
q_join
tt_gen num: 0
tt_gen num: 2
tt_gen num: 1
tt_gen num: 0
tt_gen num: 2
tt_gen num: 1
split_gen total: 3
split_gen total: 3
seqs_store
seqs_store
case_wrap type: AD
case_wrap type: NC

Figure 2.4: *Directed Acyclic Graph representation of the WISDoM pipeline for the observations of ADNI database. Here we have 2 type of subject, labeled AD and NC, undergoing a 3 fold train-test split.*

Bibliography

[1] Hardle, Wolfgang and Leopold Simar
Applied Multivariate Statistical Analysis
Heidelberg: Springer Berlin Heidelberg, 2012

[2] Anderson, T. W.
An Introduction to Multivariate Statistical Analysis
New York: John Wiley and Sons, 2003

[3] Han Liu and Larry Wasserman
Statistical Machine Learning
Pittsburgh: CMU University, 2014

[4] Murphy, Kevin P.
Conjugate Bayesian Analysis of the Gaussian Distribution
Vancouver: University of British Columbia, 2007

[5] W.B. Smith and R.R. Hocking
Algorithm AS 53: Wishart Variate Generator
Applied Statistics, vol. 21, pp. 341-345, 1972.

[6] Köster, Johannes and Rahmann, Sven
Snakemake - A scalable bioinformatics workflow engine
Bioinformatics, Volume 28, Issue 19, Pages 2520–2522,1 October 2012
DOI: https://doi.org/10.1093/bioinformatics/bts480