IFT 6269: Probabilistic Graphical Models **Fall 2018**

Lecture 9 — October 2

Lecturer: Simon Lacoste-Julien *Scribe: Eeshan Gunesh Dhekane*

Disclaimer: These notes have only been lightly proofread.

9.1 (Fisher) Linear Discriminant Analysis

Let us consider the problem of binary classification! In the last class, we considered **Logistic Regression**, which is a **discriminative** (conditional) approach to binary classification. In this class, we consider **Linear Discriminant Analysis**[1] (LDA), which is a **Generative** approach to binary classification. Recall that a conditional approach models only the predictive distribution $p(Y \mid X)$ of labels given the input data. On the other hand, the generative approach of LDA models the input data as well. Specifically, LDA models the entire dataset distribution $p(X, Y)$ through Gaussian **Class-Conditional Distributions** $p(X \mid Y)$ and Bernoulli **Prior Distribution** $p(Y)$. The parameters involved in this modeling can be easily learned from a training dataset using closed-form **Maximum Likelihood Estimates**. Having learned the parameters, predictions can be easily made for test dataset through predictive distribution $p(Y \mid X)$. In the following sections, we will consider the LDA approach in detail. However, we begin with a brief introduction to **Vector/Matrix Calculus** followed by the **Plate Notation** and the Graphical Representation of Probabilistic Approaches.

9.2 Vector and Matrix Calculus

9.2.1 Motivation

Recall that the definition of the derivative of a real-valued function of a real-valued variable can be given as follows :

A function $f : \mathbb{R} \to \mathbb{R}$ is said to be differentiable at $x_0 \in \mathbb{R}$

$$\iff \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \text{ exists (and has a finite value).} \quad (9.1)$$

The value of the limit $\lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$ is often denoted by $f'(x_0)$ and is called the **Derivative** of f at x_0. The notion of derivative of a function is extremely useful for modeling its rate of change, points of extrema, linear approximation etc. Thus, we want to extend

[1] Note that **Linear Discriminative Analysis** generalizes the Fisher's Linear Discriminant method, which uses a linear combination of features for data classification. For simplicity, we will denote the (Fisher) Linear Discriminative Analysis by **LDA** throughout the notes.

this notion to a broader class of functions. Specifically, we want to define the derivative of vector/matrix valued functions of vector/matrix argument.

It is clear from [9.1] that the same definition can not be used directly for generalization because it involves the limit of a fraction. For vectors and matrices, the notion of fraction and division is not always well-defined. However, we can rewrite [9.1] as follows :

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h} = f'(x_0) \text{ is equivalent to: } f(x+h)-f(x) = f'(x_0)\cdot h + e(h),$$
$$\text{where } e(h) \text{ is an error function that satisfies } \lim_{h\to 0}\frac{e(h)}{h} = 0 \quad (9.2)$$

This form of the definition of derivative is easier to generalize to the desired broader category of functions. It involves writing the change in the value of the function f as the sum of a linear operator[2] $f'(x_0)$ acting on the change h in the argument of the function and an error function $e(h)$. Note that the linear operator $f'(x_0)$ is defined entirely in terms of x_0 and it acts on the change h in the argument. Also, as $h \to 0$, the error function must satisfy $\lim_{h\to 0}\frac{e(h)}{h}$. With these observations, we can generalize the definition of derivative.

9.2.2 Generalizing Differentiability

We first consider the vector-valued functions of vectors.

Definition [Differentiability] : Consider function f such that $f : \mathbb{R}^m \to \mathbb{R}^n$.
f is **Differentiable** at $\mathbf{x_0} \in \mathbb{R}^m \iff \exists$ a **Linear Operator** $\mathrm{d}f_{\mathbf{x_0}} : \mathbb{R}^m \to \mathbb{R}^n$
such that $\forall\ \Delta \in \mathbb{R}^m, f(\mathbf{x_0}+\Delta) - f(\mathbf{x_0}) = \mathrm{d}f_{\mathbf{x_0}}(\Delta) + o(\|\Delta\|)$ (9.3)

Here, $o(\|\Delta\|)$ represents **Error Function** $h(\|\Delta\|)$ such that $\lim_{\|\Delta\|\to 0}\frac{h(\|\Delta\|)}{\|\Delta\|} = 0$.
The term $o(\|\Delta\|)$ is usually called and is read as the "little oh" of Δ.

The linear operator $\mathrm{d}f_{\mathbf{x_0}}$ is called the **Differential** of f at $\mathbf{x_0}$.
The differential is a linear operator
$\iff \mathrm{d}f_{\mathbf{x_0}}(\Delta_1 + b\cdot\Delta_2) = \mathrm{d}f_{\mathbf{x_0}}(\Delta_1) + b\cdot\mathrm{d}f_{\mathbf{x_0}}(\Delta_2)\ \ \forall\ \Delta_1, \Delta_2 \in \mathbb{R}^m,\ \forall\ b \in \mathbb{R}$

Remark : The differential operator $\mathrm{d}f_{\mathbf{x_0}}$ can be thought of as a "machine" or a "processor" which inputs vectors from $\mathbb{R}^m$, processes them and generates an output vector in $\mathbb{R}^n$. It should be noted that this operator is entirely defined in terms of $\mathbf{x_0}$ and it should be linear.

[2] An operator can be thought of as a "machine" that inputs a variable from the domain space, processes it and yields an output in the target space. We will briefly consider the exact meaning of an operator in the next subsection. However, we will describe the form of these operators for the specific cases of interest.

Remark : In the case of $f : \mathbb{R}^m \to \mathbb{R}^n$, the differential $\mathrm{d}f_{\mathbf{x_0}}$ takes the form of a matrix with order $n \times m$ and the operation $\mathrm{d}f_{\mathbf{x_0}}$ becomes the matrix multiplication of $\mathrm{d}f_{\mathbf{x_0}}$ and Δ. The details are as given below :

If $f : \mathbb{R}^m \to \mathbb{R}^n$, then $\mathrm{d}f_{\mathbf{x_0}} \in \mathbb{R}^{n \times m}$ and $\mathrm{d}f_{\mathbf{x_0}}(\Delta) = \mathrm{d}f_{\mathbf{x_0}} \cdot \Delta \ (\forall\ \Delta \in \mathbb{R}^m)$
The differential represented as a matrix of order $n \times m$ is called the **Jacobian Matrix**.
Let $(\mathrm{d}f_{\mathbf{x_0}})_{i,j}$ denote the $(i,j)-$th component of $\mathrm{d}f_{\mathbf{x_0}}$. Then, $(\mathrm{d}f_{\mathbf{x_0}})_{i,j} = \frac{\partial f_i}{\partial \mathbf{x}_j}\Big|_{\mathbf{x_0}}$ (9.4)
where f_i is $i-$th component of f and $\mathbf{x}_j$ is $j-$th component of $\mathbf{x}$ $(1 \le i \le n, 1 \le j \le m)$.

Remark : This definition of differentiability (and of differential) from [9.3] gives a way to define the derivatives not only for the cases of vectors but also for matrices and tensors. It is easy to see that exactly the same definition will continue to hold for the more general cases of matrices and tensors. However, one needs to be careful about the form of the differential!

Remark : Another important case to consider is that of real-valued functions of square matrices. It is important as it is required a lot for MLE and MAP estimations corresponding to Gaussian distributions. In the case of $f : \mathbb{R}^{n \times n} \to \mathbb{R}$, the differential $\mathrm{d}f_{\mathbf{x_0}}$ takes the form of a matrix with order $n \times n$ and the operation $\mathrm{d}f_{\mathbf{x_0}}$ becomes the **Trace** of matrix multiplication of $\mathrm{d}f_{\mathbf{x_0}}$ and Δ. The details are as given below :

If $f : \mathbb{R}^{n \times n} \to \mathbb{R}$, then $\mathrm{d}f_{\mathbf{x_0}} \in \mathbb{R}^{n \times n}$ and $\mathrm{d}f_{\mathbf{x_0}}(\Delta) = \mathrm{tr}\left(\mathrm{d}f_{\mathbf{x_0}}{}^{\top}\Delta\right)\ (\forall\ \Delta \in \mathbb{R}^{n \times n})$

9.2.3 Chain Rule

One of the most important and frequently used formula for evaluation of differentials of composition of function is the **Chain Rule**. It expresses the differential of the composition of functions in terms of the differentials of the individual functions as follows :

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^q$. Then, $\mathrm{d}(g \circ f)_{\mathbf{x_0}} = \mathrm{d}g_{f(\mathbf{x_0})} \cdot \mathrm{d}f_{\mathbf{x_0}}$ (9.5)
The term $\mathrm{d}g_{f(\mathbf{x_0})} \cdot \mathrm{d}f_{\mathbf{x_0}}$ is the product of the Jacobians $\mathrm{d}g_{f(\mathbf{x_0})} \in \mathbb{R}^{q \times n}$ and $\mathrm{d}f_{\mathbf{x_0}} \in \mathbb{R}^{n \times m}$.

Remark : It is easy to extend the chain rule to composition of more than two functions (Exercise!). Also, note that the resultant product of Jacobians will always be well-defined.

9.2.4 Important Examples

In this subsection, we consider some examples that not only help demonstrate the use of the definition from [9.3] in order to calculate the differentials of certain important functions, but build the required tool-kit for the analysis of LDA in the subsequent sections!

Differential of Squared Mahalanobis Distance : Let $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $\forall\ \mathbf{x} \in \mathbb{R}^d$, we have : $f(\mathbf{x}) = \mathbf{x} - \boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is a constant vector in $\mathbb{R}^d$. Now, in order to evaluate the differential of f at $\mathbf{x}$, consider the following manipulations for any $\mathbf{h} \in \mathbb{R}^d$:

$$f(\mathbf{x}+\mathbf{h}) - (\mathbf{x}) = ((\mathbf{x}+\mathbf{h}) - \boldsymbol{\mu}) - (\mathbf{x} - \boldsymbol{\mu}) = \mathbf{h} = I_{d\times d} \cdot \mathbf{h} + \mathbf{0}$$

where $\mathbf{0}$ is the $d-$dimensional zero vector. Clearly, $\lim_{\|\mathbf{h}\|\to 0} \frac{\mathbf{0}}{\|\mathbf{h}\|} = 0$.

Thus, by definition, we have $\mathrm{d}f_{\mathbf{x}} = I_{d\times d}\ \forall\ \mathbf{x} \in \mathbb{R}^d$ (9.6)

where $I_{d\times d}$ is the $d-$dimensional identity matrix.

Now, consider $g : \mathbb{R}^d \to \mathbb{R}$ such that $\forall\ \mathbf{x} \in \mathbb{R}^d$, we have : $g(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$, where $A \in \mathbb{R}^{d\times d}$ is a fixed matrix of order $d \times d$. Now, in order to find the differential of g at $\mathbf{x}$, we consider the following manipulations for any $\mathbf{h} \in \mathbb{R}^d$:

$$g(\mathbf{x}+\mathbf{h}) - g(\mathbf{x}) = (\mathbf{x}+\mathbf{h})^\top A(\mathbf{x}+\mathbf{h}) - \mathbf{x}^\top A\mathbf{x} = \left(\mathbf{x}^\top A\mathbf{h} + \mathbf{h}^\top A\mathbf{x}\right) + \mathbf{h}^\top A\mathbf{h}$$

Now, $\mathbf{h}^\top A\mathbf{x} \in \mathbb{R} \Rightarrow \left(\mathbf{h}^\top A\mathbf{x}\right)^\top = \mathbf{x}^\top A^\top \mathbf{h}$. Also, $\mathbf{x}^\top A\mathbf{h} \in \mathbb{R}$.

$$\Rightarrow g(\mathbf{x}+\mathbf{h}) - g(\mathbf{x}) = \mathbf{x}^\top \left(A + A^\top\right)\mathbf{h} + \mathbf{h}^\top A\mathbf{h}$$

Now, it is easy to prove that $\lim_{\|\mathbf{h}\|\to 0} \dfrac{\mathbf{h}^\top A\mathbf{h}}{\|\mathbf{h}\|} = 0$. (Try this as an Exercise!

Thus, by definition, we have $\mathrm{d}g_{\mathbf{x}} = \mathbf{x}^\top \left(A + A^\top\right)\ \forall\ \mathbf{x} \in \mathbb{R}^d$ (9.7)

Now, we consider the matrix A to be the inverse of **Covariance Matrix** Σ of some **Gaussian Distribution** $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, with $\boldsymbol{\mu} \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d\times d}$. Then, the Squared **Mahalanobis Distance**[3] with respect to the given Gaussian Distribution, denoted by $\ell : \mathbb{R}^d \to \mathbb{R}$, is defined as follows :

$$\ell_{\boldsymbol{\mu},\Sigma}(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\ \ \forall\ \mathbf{x} \in \mathbb{R}^d \tag{9.8}$$

Note that the term $\ell_{\boldsymbol{\mu},\Sigma}(\mathbf{x})$ appears in the log-likelihood expressions involving Gaussian distributions and hence, we need to evaluate its differential in order to solve for MLE, MAP estimates! Consider the following manipulations for getting the required differential :

[3]For extra information, please refer to the Wikipedia page on [**Mahalanobis Distance**].

Define $\ell : \mathbb{R}^d \to \mathbb{R}$ such that $\ell(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^\top A (\mathbf{x} - \boldsymbol{\mu}) \ \forall \ \mathbf{x} \in \mathbb{R}^d$

$\forall \ \mathbf{x} \in \mathbb{R}^d$, we have : $\ell(\mathbf{x}) = (g \circ f)(\mathbf{x})$ where, $\circ$ denotes function composition.

Here, f and g are the functions defined above.

Thus, the differential for ℓ can be computed using the chain rule as follows :

$$\mathrm{d}\ell_{\mathbf{x}} = \mathrm{d}g_{f(\mathbf{x})} \cdot \mathrm{d}f_{\mathbf{x}} = \left((\mathbf{x} - \boldsymbol{\mu})^\top \left(A + A^\top\right)\right) \cdot I_{d\times d}$$

Now, since Σ^{-1} is symmetric, we have $\Sigma^{-1} = (\Sigma^{-1})^\top$. Thus, we get :

$$\mathrm{d}\ell_{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})^\top \left(A + A^\top\right) \text{ and } \mathrm{d}(\ell_{\boldsymbol{\mu},\Sigma})_{\mathbf{x}} = 2\,(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} \tag{9.9}$$

Remark : In any calculation of differential, we must ensure that the error function (denoted by $h(\|\Delta\|)$) tends to 0 strictly faster than $\|\Delta\|$. In almost all our cases, we will deal with vector, matrix (or tensor) values of Δ. Thus, by default, we will consider $\|\Delta\|$ to be the **Frobenius Norm**[4] of Δ. For any matrix (including a vector), it is defined as follows :

The Frobenius norm of a vector or a matrix T is denoted by $\|T\|_F$.

Frobenius norm of vector $\mathbf{v} \in \mathbb{R}^d$ is defined as : $\|\mathbf{v}\|_F = \sqrt{\sum_{i=1}^{d} |\mathbf{v}_i|^2}$ (9.10)

Frobenius norm of matrix $M \in \mathbb{R}^{n\times m}$ is defined as : $\|M\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |M_{i,j}|^2}$

Differential of Determinant of a Square Matrix : Another important function that appears whenever we consider the log-likelihood involving Gaussian Distributions is the logarithm of the determinant of the covariance matrix. Thus, it is important to consider the differential of the log-determinant of a matrix.

Let function $f : \mathbb{R}^{d\times d} \to \mathbb{R}$ be defined as : $f(A) = \log|A| \ \forall \ A \in \mathbb{R}^{d\times d}$. Consider any matrix $\Delta \in \mathbb{R}^{d\times d}$. Then, $f(A+\Delta) - f(A) = \log|A+\Delta| - \log|A|$. For simplicity, we restrict our proof to symmetric matrix $A \in \mathbb{R}^{d\times d}$ that is strictly positive definite i.e. $A \succ 0$. This implies that A is invertible and that its matrix square roots exists (by the spectral theorem): i.e. $\exists \ B \in \mathbb{R}^{d\times d}$ such that $B \cdot B = A$. We denote B by $A^{\frac{1}{2}}$ and call it **Square Root of Matrix**[5] A. Note that we can always find the square root for a real-valued symmetric matrix and hence, our proof *will* work for the cases when A equals the covariance matrix of a Gaussian distribution or its inverse. Now, for a matrix $M \in \mathbb{R}^{d\times d}$, let $\lambda_i(M)$ $(i \in \{1, \ldots, d\})$ denote the d eigenvalues of M, in decreasing order, with multiplicity counted. Then, we will

[4]For extra information on Frobenius Norm, please refer to the Wikipedia pages on [**Matrix Norm**] and [**Frobenius Inner Product**].

[5]For more information on the Square Roots of Matrices, their existence and construction, please refer to the Wikipedia page on [**Square Root of a Matrix**].

use the following standard linear algebra properties in our derivation :

For matrices $B, C \in \mathbb{R}^{d\times d}$, we have : $|B \cdot C| = |B| \cdot |C|$.
For matrix $M \in \mathbb{R}^{d\times d}$ that is diagonalizable, we have : $\text{tr}\,(M) = \sum_{i=1}^{d} \lambda_i(M)$ and $|M| = \prod_{i=1}^{d} \lambda_i(M)$.
For any matrices $B, C, D \in \mathbb{R}^{d\times d}$, we have : $\text{tr}\,(B \cdot C \cdot D) = \text{tr}\,(D \cdot B \cdot C)$.
For matrix $M \in \mathbb{R}^{d\times d}$, we have : $M \to 0 \Rightarrow \lambda_i(M) \to 0 \;\forall\; i \in \{1, \ldots, d\}$.
For $x \in \mathbb{R}, |x| < 1 \Rightarrow \log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}$ (9.11)

Now, we consider the derivation to obtain the differential of log-determinant of f :

$$\begin{aligned}
\log|A+\Delta| - \log|A| &= \log\left|A^{\frac{1}{2}}\left(I + A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)A^{\frac{1}{2}}\right| - \log\left|A^{\frac{1}{2}}A^{\frac{1}{2}}\right| \\
&= \log\left(\left|A^{\frac{1}{2}}\right|\left|I + A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right|\left|A^{\frac{1}{2}}\right|\right) - \log\left(\left|A^{\frac{1}{2}}\right|\left|A^{\frac{1}{2}}\right|\right) \\
&= \log\left|I + A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right| = \log\left(\prod_{i=1}^{d} \lambda_i\left(I + A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)\right) \\
&= \sum_{i=1}^{d} \log\left(\lambda_i\left(I + A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)\right) = \sum_{i=1}^{d} \log\left(1 + \lambda_i\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)\right) \\
&= \sum_{i=1}^{d} \lambda_i\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right) + \sum_{k=2}^{\infty}\sum_{i=1}^{d} (-1)^{k+1}\frac{\lambda_i^k\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)}{k} \\
&= \sum_{i=1}^{d} \lambda_i\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right) + o\left(\|\Delta\|\right) = \text{tr}\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right) + o\left(\|\Delta\|\right) \\
&= \text{tr}\left(A^{-\frac{1}{2}}A^{-\frac{1}{2}}\Delta\right) + o\left(\|\Delta\|\right) = \text{tr}\left(A^{-1}\Delta\right) + o\left(\|\Delta\|\right)
\end{aligned}$$

Thus, by definition, the differential of log-determinant of A is A^{-1}.
Using the standard notation, we have : $\frac{\mathrm{d}\log|A|}{\mathrm{d}A} = A^{-1}$ (9.12)

Remark [Optional] : The proof above has a small jump! After expanding the terms of $\log(1+\lambda_i(M))$, we get an error function $e(\|\Delta\|)$ in terms of $\lambda_i^k(M)$, where $M = A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}$ and $k > 1$. We need to prove that $\lim_{\|\Delta\|\to 0} \frac{e(\|\Delta\|)}{\|\Delta\|} = 0$. We can prove this as follows :

Observe that λ_i^k are eigenvalues of M^k. Thus, $\sum_{i=1}^{d} \lambda_i^k = \text{tr}\left(M^k\right) = \text{tr}\left(\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)^k\right)$.

Now, for any square matrix $B \in \mathbb{R}^{d\times d}$, $|\text{tr}\,(B)| = \left|\sum_{i=1}^{d} B_{i,i}\right| \le \sum_{i=1}^{d} |B_{i,i}| \le \sqrt{d}\sqrt{\sum_{i=1}^{d} B_{i,i}^2} \le \sqrt{d}\|B\|_F$. The second-last inequality follows from the **Root-Mean-Squares Inequality**[6].

For matrices $B, C \in \mathbb{R}^{d\times d}$, **Cauchy-Schwarz Inequality** gives : $\|B \cdot C\|_F \le \|B\|_F \cdot \|C\|_F$.

[6]The so-called Root-Mean-Squares Inequality forms a special case of a broad category of important inequalities involving the **Generalized Mean**. A concise reference for these inequalities can be found at the Wikipedia page on [**Generalized Means Inequality**]

$$\therefore 0 \leq \left|\sum_{i=1}^{d} \lambda_i^k\right| = \left|\text{tr}\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)^k\right| \leq \sqrt{d}\left\|\left(A^{-\frac{1}{2}}\Delta A^{-\frac{1}{2}}\right)^k\right\|_F \leq \sqrt{d}\left\|A^{-\frac{1}{2}}\right\|_F^k \cdot \|\Delta\|_F^k \cdot \left\|A^{-\frac{1}{2}}\right\|_F^k.$$

This gives us the required limit for every $k > 1$ by using **Squeeze Theorem** of limits. We have : $0 \leq \left|\sum_{i=1}^{d} \lambda_i^k\right| \leq \sqrt{d}\left\|A^{-\frac{1}{2}}\right\|_F^k \cdot \|\Delta\|_F^k \cdot \left\|A^{-\frac{1}{2}}\right\|_F^k$. Since $k > 1$ from the derivation, we have :

$0 \leq \lim_{\|\Delta\|_F \to 0} \frac{\left|\sum_{i=1}^{d} \lambda_i^k\right|}{\|\Delta\|_F} \leq \lim_{\|\Delta\|_F \to 0} \sqrt{d}\left\|A^{-\frac{1}{2}}\right\|_F^k \cdot \|\Delta\|_F^{k-1} \cdot \left\|A^{-\frac{1}{2}}\right\|_F^k = 0$. Thus, we get $\lim_{\|\Delta\|_F \to 0} \sum_{i=1}^{k} \frac{\lambda_i^k}{\|\Delta\|_F} = 0 \ \forall \ k > 1$. This completes the proof that the error function indeed tends to zero strictly faster than $\|\Delta\|$ and we can simply replace the *ugly* expression of the error function by $o\left(\|\Delta\|\right)$.

References : For details related to matrix calculus, please refer to the book [**Matrix Differential Calculus with Applications in Statistics and Econometrics**] by Neudecker and Magnus[7].

9.3 Plate Notation and Graphical Representation

9.3.1 Motivation

Before we begin with the analysis of LDA, let us take a look at the method for **Graphical Representation** of probabilistic approaches and the so-called **Plate Notation**[8]. Here, our aim is to represent any probabilistic approach in a graphical format so that it is easy to visualize ("A picture is worth a thousand words!"). In any probabilistic model, we want to model the uncertainty in some of random variables/vectors and parameters, and then learn the corresponding underlying distributions. We assume that the rest of the random variables/vectors and parameters are fully known and hence, we do not want to model uncertainty in these variables. In our graphical model, we need rules to represent these different sets of variables clearly. Further, the dependencies between various random variables and parameters under consideration should be clearly represented. In addition, it might happen that the scenario has a huge number of random variables or parameters that need to be considered (e.g, dataset with $N = 1$M samples). Then, we need to represent the repeated variables in our diagram in a concise as well as precise manner. Towards this, we consider the following set of rules to define our graphical representation method[9].

[7] Note that there are multiple conventions for setting the dimensions of the derivatives and it is imperative that we stick to one particular convention in order to get consistent results. The definition from [9.3] results in the answers that abide the so-called **Numerator Layout**. In this context, a good reference point for checking the final answers for expressions of derivatives is the Wikipedia page for [**Matrix Calculus**].

[8] For more information, please refer to the Wikipedia page on [**Plate Notation**].

[9] Note that the rules for graphical representation for probabilistic models vary a lot from different sources. The Wikipedia reference on [**Plate Notation**] gives one of such sets of rules for graphical representation.

9.3.2 Rules for Graphical Representation and Plate Notation

1. Random variables and parameters for which we want to model the uncertainty are represented by circular nodes with the variable names.
2. Random variables and parameters that are assumed to be known and for which we do not model the uncertainty are represented by square blocks with the variable names.
3. The dependencies in between various random variables and parameters are represented using directed arrows.
4. If a random variable is observed, then the circular node corresponding to it is shaded. If it is not observed, then the circular node corresponding to it is not shaded.

Note that the rules above help fix a convention for graphical representation of probabilistic approaches. However, we need to cater for cases where variables repeat. The so-called **Plate Notation** help fix conventions for concisely representing the repeated variables in a model.

1. We use a rectangle (also called a **Plate**) to group together inside it all the random variables and parameters that repeat together.
2. Each of the variables in a plate is indexed and the range of the index is mentioned on the plate. In order to expand the representation, we repeated the contents of the plate.
3. Arrows that cross the plate represent one directed arrow per repetition of the plate.

The Figure [9.1] illustrates these rules with the help of an example.

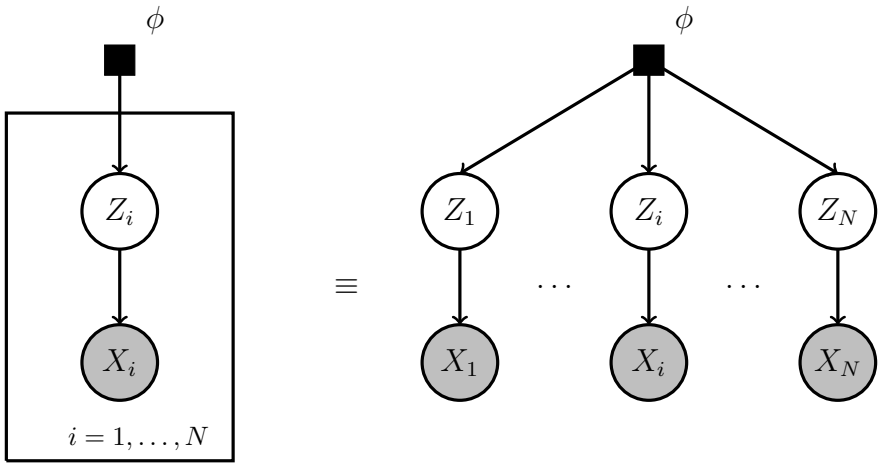

Figure 9.1: An example of a graphical representation involving the plate notation. Note that the parameter ϕ is assumed to be known and the uncertainty in it is not modeled. There are N random variables Z_i, X_i ($i \in \{1, \ldots, N\}$) for which we want to model uncertainty. X_i is dependent only on Z_i. The random variables Z_i is dependent on parameter ϕ. The only observed variables are X_i; Z_i are not observed. Note that the figure to the left represents this model concisely using the rules of graphical representation and plate notation. An equivalent expanded graphical representation is shown on the right.

9.4 Analysis of Fisher LDA

9.4.1 Formulation

We now consider the analysis of the generative (Fisher) linear discriminant analysis (LDA) model for binary classification. Let X denote the random vector corresponding to the input data such that $X \in \mathbb{R}^d$. Let Y denote the random variable corresponding to the binary label of input X. We represent the two labels by $0, 1$ and thus, $Y \in \{0, 1\}$. We model the input data by modeling the entire dataset $p(X, Y)$ in terms of **Class-Conditional Distributions** $p(X \mid Y)$ and **Prior Distribution** $P(Y)$. Now, we assume the following forms of distributions for $p(X \mid Y)$ and $p(Y)$:

$$Y \sim \text{Bernoulli}\,(\pi) \text{ for some } \pi \in [0,1] \Rightarrow p\,(y \mid \pi) = \pi^y (1-\pi)^{1-y} \;\forall\; y \in \{0,1\}$$
$$X \mid Y = j \sim \mathcal{N}\left(\boldsymbol{\mu}_j, \Sigma\right) \Rightarrow p\,(x \mid y = j) = \frac{1}{\sqrt{2\pi}^d \sqrt{|\Sigma|}} e^{-\frac{1}{2}\left(x-\boldsymbol{\mu}_j\right)^\top \Sigma^{-1}\left(x-\boldsymbol{\mu}_j\right)} \tag{9.13}$$

Here, $\boldsymbol{\mu}_j \in \mathbb{R}^d$ is the mean of the $j-$th class $\forall\; j \in \{0,1\}$ and $x \in \mathbb{R}^d$ and Σ represents the covariance matrix with $\Sigma \in \mathbb{R}^{d \times d}$.

Notice that LDA assumes that the covariance matrix Σ is the same for both classes.

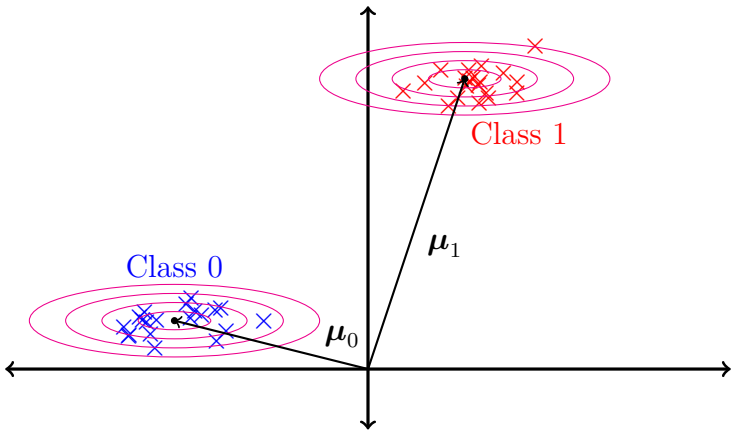

Figure 9.2: The schematics of a scenario which can be best modeled by LDA. Note that the two class-conditional distributions have different means but the same covariance matrix.

Now, let $D = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \mid \mathbf{x}^{(i)} \in \mathbb{R}^d \text{ and } y^{(i)} \in \{0,1\}\,,\; \forall\; i \in \{1, \ldots, n\}\right\}$ be the input dataset. Then, the problem of modeling the dataset becomes the problem of estimating the parameters $\theta = (\pi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \Sigma)$ that define the class conditional and prior distributions. We will use joint **Maximum Likelihood Estimation** to estimate these desired parameters (exercise

in homework 2). As shown in the previous lecture with the general exponential family, one can show that $p(y|x;\theta) = \sigma(w^\top \phi(x))$ where $\sigma(\cdot)$ is the sigmoid function and the parameter w can be expressed as a function of the parameters $(\pi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \Sigma)$. Here $\phi(x) = (\mathbf{x}, 1)$, and thus the decision boundary is linear. If instead we model each class with different covariance matrices Σ_0 and Σ_1, then one can show that $\phi(x)$ is a quadratic function of x and one obtains quadratic discriminant analysis (QDA) – see homework 2.

Remark : The graphical representation for LDA can be given as done in Figure [9.3], where $\theta = (\pi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \Sigma)$. (As an exercise, try to find graphical representations for Linear and Logistic Regression approaches!)

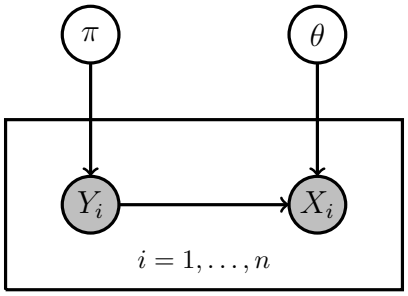

Figure 9.3: The graphical representation of LDA using plate notation.

9.5 Unsupervised Learning

9.5.1 Two Views for Unlabeled Data

Till now, we have consider the scenarios for modeling datasets of input data points and the corresponding labels. However, there are numerous real-life problem settings where we do not have access to the labels corresponding to data. Thus, without any labels, we want to model the data. There are two ways to consider the unlabeled data : *i.* **Mixture Distribution Approach** and *ii.* **Latent Variable Approach**.

In order to understand these approaches better, consider the example of unlabeled data in Figure [9.4]. The given data can be viewed as a mixture of several component distributions. For instance, the data distribution in the figure can be viewed as a mixture of two Gaussian distribution. However, we can easily visualize the data to be coming from two groups, or **Clusters**, such that the points from the same cluster are very similar to one another and those from different are different from one another. Thus, we can understand the structure in the data in a better manner by trying to model these clusters. However, since the cluster assignment of data points are not available in the unsupervised problem, we call the clusters **Latent Variables** and aim to learn the latent variables from the available data. The difference in the two approaches can also be seen from their plate diagrams, which are given in Figure [9.5]. In this scribe, we will only consider the latent variable approach.

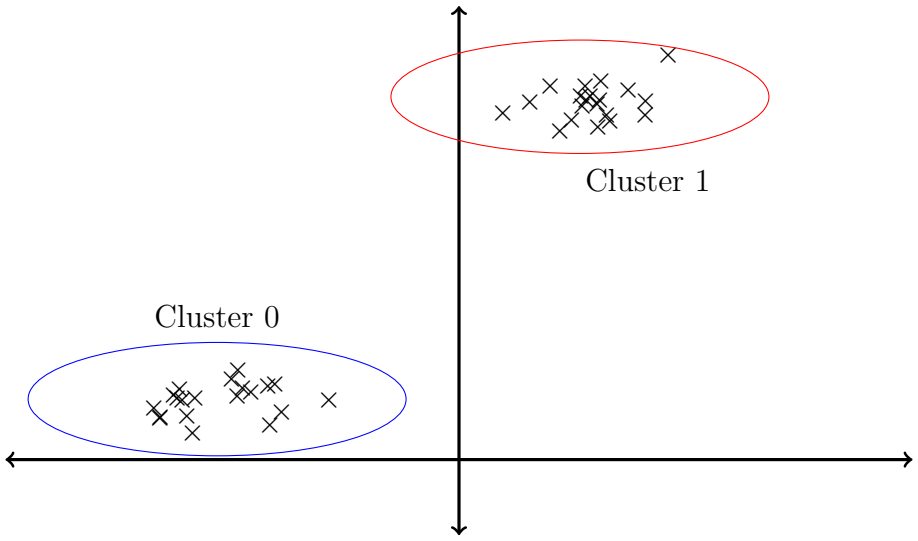

Figure 9.4: There are two views of unlabeled data. We can view it as a mixture distribution or consider the data in terms of latent variables. In the example, it appears as though the data is coming from two different "groups" or "clusters" and hence, we can aim at finding the characteristics of these structures in order to learn the underlying structure.

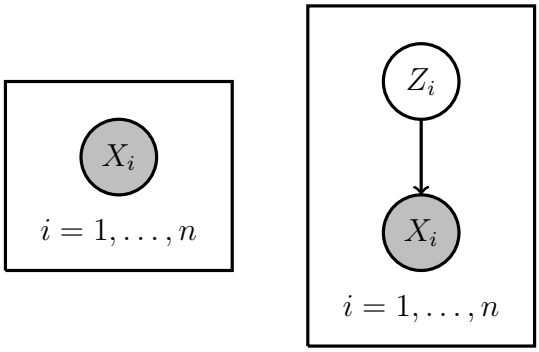

Figure 9.5: Graphical Models for mixture distribution approach (*on the left*) and latent variable approach (*on the right*). Here X_i represents the unlabeled data and Z_i represents the assumed latent variables for the corresponding data point X_i.

9.6 K-Means Algorithm

9.6.1 Motivation

Let us consider the problem of clustering the given unlabeled data. We want to learn a **Cluster Assignment Function** that predicts the cluster to which each data point is mapped. We assume that there are K clusters in the data, which are labeled $\{1, \ldots, K\}$. We represent each of the clusters i ($i \in \{1, \ldots, K\}$) by its representative **Cluster Center** μ_i. The idea is that the data points that belong to a particular cluster center should not *differ*

too much from the corresponding cluster center. We measure this extent of difference using a **Distortion Function**, which is defined in terms of a chosen **Distance Function**. Since we do not have any information about the cluster centers, we initialize them randomly. Then, we perform an iterative algorithm starting with a guess of the cluster assignment function which maps each data point to some cluster center. Then, we update the cluster centers so that the distortion measure is minimized. Intuitively, this step makes our guess of cluster centers better. However, now with the better guess for cluster centers, we can get a better cluster assignment function! Thus, we repeat these two steps to decrease the distortion function until we converge to the best cluster centers and cluster assignment function.

9.6.2 Formulation

We will use the following notations: $x_i \in \mathbb{R}^p, i \in \{1, \dots, n\}$ are the observations we want to partition. $\mu_k \in \mathbb{R}^p, k \in \{1, \dots, K\}$ are the means where μ_k is the center of the cluster k. We will denote μ the associated matrix. $z_{i,k}$ are indicator variables associated to x_i such that $z_{i,k} = 1$ iff x_i belongs to the cluster k, $z_{i,k}$ otherwise. z is the matrix which components are equal to $z_{i,k}$.

Distortion Function : we define the distortion $J(\mu, z)$ by: $J(\mu, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|_2^2$

Algorithm : The aim of the algorithm is to minimize $J(\mu, z)$. To do so we proceed with **Alternating Minimization** or **Block Coordinate Minimization** :

- Initialize cluster centers μ.
- we minimize J with respect to z : $z_{i,k} = 1$ if $\|x_i - \mu_k\|^2 = \min_s \|x_i - \mu_s\|^2$, in other words we associate to x_i the nearest center μ_k.
- we minimize J with respect to μ : $\mu_k = \frac{\sum_i z_{i,k} x_i}{\sum_i z_{i,k}}$.
- we come back to step 1 until convergence.

Remark : The step of minimization with respect to z is equivalent to allocating the x_i in the Voronoi cells which centers are the μ_k.

Remark : During the step of minimization with respect to μ, μ_k is obtained by setting to zero the $k-$th coordinate of the gradient of J with respect to μ. Indeed we can easily see that : $\nabla_{\mu_k} J = -2 \sum_i z_{i,k}(x_i - \mu_k)$

9.6.3 Properties of $k-$means

- It converges in finite number of iterations to a local minimum. However, it is just a local min. In general, it is NP-hard to find the best cluster assignment. In general implies that there are cases which require time exponential in input size. There are certain cases where we get there is very easy solutions.

- It is very fast and requires lesser number of iterations.
- Initialization is very important for $k-$means. There is an algorithm $k-$means++, which gives a clever initialization scheme that guarantees that objective is within $\log k$ of the global optimum with high probability (w.h.p.). (There is a theoretical guarantee). Spread out the initial mean points as much as possible. This avoids the wrong clustering (image in class). We select new means as per the inverse of their distance from previous means.
- **Choice of K :** One of the heuristics is–

$$J(\mu, z, K) = \sum_{i=1}^{n} \sum_{j=1}^{K} z_{i,j} \left\| x_i - \boldsymbol{\mu}_j \right\|^2 + \underbrace{\lambda \cdot K}_{\text{Regularization Term}} \tag{9.14}$$

  λ is the hyperparameter. We need to experiment with λ to fix its value. (Later in the class, we will see Non-Parameteric Models, where K is basically infinite and we can get $p(K \mid data)$. An example is Dirichlet Process Mixture Model). λ has an effect on the optimal value of K.
- $K-$means is very sensitive to the distance measure used. When we are using $\mathcal{L}_2$, we are getting spherical clusters. Also, the choice of clustering depends on the problem itself. The different objectives will have different best choices of clustering, which will be decided by different distance measures. (Figure in class). The "bad clustering" is actually a "good clustering" for the mail-box problem. The "problem" in the previous figure is fixed by Gaussian Mixture Model.
- **Convergence and Initialization :** We can show that this algorithm converges in a finite number of iterations. Therefore the convergence could be local, thus it introduces the problem of initialization. A classic method is use of random restarts. It consists in choosing several random vectors μ, computing the algorithm for each case and finally keeping the partition which minimizes the distortion. Thus we hope that at least one of the local minimum is close enough to a global minimum. One other well known method is the K-means++ algorithm, which aims at correcting a major theoretic shortcomings of the K-means algorithm : the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering. The K-means++ algorithm addresses this obstacles by specifying a procedure to initialize the cluster centers before proceeding with the standard K-means optimization iterations. With the K-means ++ initialization, the algorithm is guaranteed to find a solution that is $\mathcal{O}(\log K)$ competitive to the optimal K-means solution. The intuition behind this approach is that it is a clever thing to well spread out the K initial cluster centers. At each iteration of the algorithm we will build a new center. We will repeat the algorithm until we have K centers. Here are the steps of the algorithm :
  - First initiate the algorithm by choosing the first center uniformly at random among the data points.

- For each data point x_i of your data set, compute the distance between x_i and the nearest center that has already been chosen. We denote this distance $D_{\mu_t}(x_i)$ where μ_t is specified to recall that we are minimizing over the current chosen centers.
- Choose one new data point at random as a new center, but now using a weighted probability distribution where a point x_i is chosen with probability proportional to $D_{\mu_t}(x_i)^2$.
- Repeat Step 1 and Step 2 until K centers have been chosen.

We see that we have now built K vectors with respect to our first intuition which was to well spread out the centers (because we used a well chosen weighted probability). We can now use those vectors as the initialization of our standard K-means algorithm. More details can be found on the K-means++ algorithm in [A]. [A] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.