THESIS

DEPARTMENT OF COMPUTING

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

Information Geometry For Machine Learning

Author:
Adrian Millea

Supervisor:
Marc Deisenroth

November 4, 2015

Submitted in partial fulfillment of the requirements for Master of Research

Abstract

Machine learning is concerned with finding an appropriate representation of some given data $\mathcal{X}$, and then learning the underlying structure present, with the goal of making predictions for unseen data. The representation is of paramount importance as it can ultimately drive the learning algorithm. One of the goals of this thesis is to look exactly at the representation problem from the perspective of differential geometry (DG). We mostly deal with parametric models where the data is assumed to depend on some unknown (to be learned) parameters Θ. DG is a set of powerful mathematical tools that can deal naturally with different representations, and often the change between representations provides insights into some characteristics of the problem. We will look at multiple machine learning algorithms and models, and see how DG concepts and techniques can sometimes improve learning speed, accuracy and provide deeper insights into existing learning paradigms. DG is used widely in many fields of science and engineering, whereas the statistical context that we are interested in, is referred to as Information Geometry.

Contents

1 Introduction **1**

2 Background **3**

2.1 Affine connection . . . 9
2.2 The α-connection . . . 13
2.3 Exponential family . . . 14
2.4 Dual connections . . . 16
2.4.1 Divergence . . . 17
2.5 Conclusion . . . 19

3 Natural gradient **20**

3.1 Relation to the KL divergence . . . 21
3.2 Stochastic variational inference . . . 24
3.2.1 Mean-field approximate inference . . . 26
3.2.2 Coordinate ascent and natural gradient . . . 27
3.2.3 Experimental results . . . 29
3.3 Deep neural networks . . . 29
3.3.1 Natural neural networks . . . 30
3.4 The manifold tangent classifier . . . 33
3.4.1 The tangent bundle . . . 34
3.4.2 Obtaining an atlas . . . 34
3.4.3 Classification . . . 35
3.4.4 The Manifold Tangent Classifier . . . 35
3.5 Reinforcement learning . . . 36
3.5.1 Natural policy gradient . . . 38
3.5.2 Natural actor-critic . . . 39
3.6 Natural evolution strategies . . . 40

4 The volume element **45**

4.1 Metric Learning . . . 45
4.1.1 Overview . . . 47
4.2 Manifold Learning . . . 48
4.2.1 Locally Accumulated Information . . . 50
4.2.2 A gap between two models . . . 51
4.3 Conclusion . . . 52

5 Pattern recognition 53
5.1 Hashing for neighbor search 55
5.1.1 Hashing covariance matrices 56
5.2 Linear dynamical models 57
5.3 Non-parametric Fisher 58
5.4 Facial analysis and domain adaptation 59

6 Contribution 62
6.1 Bayesian metric learning 62
6.2 Supervised pretraining 64
6.3 Accelerated learning 65

Chapter 1

Introduction

Machine learning has as a main object of study a given data set $\mathcal{X} = \{x_i, ..., x_n\}$. The goal of machine learning algorithms is to catch some characteristics of the data such that they are able to generalize to yet unseen data. This process of generalization is referred to as learning or inference (even though in the statistical machine learning community these are differentiated in some sense). There are two main opposed learning paradigms, parametric and non-parametric. For parametric models, the main assumption is that the underlying structure in the data can be described by a finite-set of parameters Θ. As data arrives, the parameters are learned (modified after model-specific learning rules) but their number does not change. In non-parametric approaches, the complexity of the model (the actual number of parameters) increases as more data is available, incurring additional computational cost to the overall learning algorithm. There is a direct relation between the model and the data, and any assumption for each space is critical for successful modeling. Usually, neither of the two spaces have an Euclidean geometry and one of the main points of this thesis will be to advocate for finding the appropriate geometry for each problem at hand. For data space, a simple example is considering the task of object recognition from images. We need to extract some representative features from objects, some higher-order structures, which define the distance measure used. If we would use raw pixel data, Euclidean distances would be irrelevant for this task, yielding, for example, similar objects for similar colors in the images. For model space, even the normal family of probability distribution, has a hyperbolic geometry, overall in very few cases the Euclidean geometry makes sense, that is accurately describes relevant distances between data points. We argue for finding the geometry of the combination between the data space and model space as the data induces a geometry in the model. For accurate and efficient learning both of these spaces need to be taken into account and especially the interaction between them, how the actual process of learning depends on them. Information geometry refers exactly to these dependencies and it requires a more flexible metric that is able to take into account the different geometries of data and model spaces.

For this reasons, we will investigate the insights that the wide set of mathematical tools present in Differential Geometry (the purely mathematical field on which Information Geometry is based) can give to Machine Learning when considering the data and the model geometries. The relation between the data and the model has been investigated in statistics through the Fisher information; for statistical models it all started with the fundamental work of Rao in 1945, which proved that an estimator is bounded from below by the inverse of the Fisher information matrix (FIM). To define the FIM we need to consider the probability function of X (observed) that depends on some parameter θ (unknown) and taking the partial derivative of the natural logarithm of $p(x;\theta)$ with respect to θ gives what is referred to as *the score*, $\frac{\partial \log p(x;\theta)}{\partial\theta}$. The first moment of the score with respect to $p(x;\theta)$ is 0 ($\mathbb{E}\left[\frac{\partial \log p(x;\theta)}{\partial\theta}\right] = 0$), while the second moment is the FIM and is given by:

$$\mathcal{F} = \mathbb{E}\left[\frac{\partial \log p(x;\theta)}{\partial\theta}\,\frac{\partial \log p(x;\theta)}{\partial\theta}\right]$$

where the expectation is taken with respect to $p(x;\theta)$. The Cramer-Rao bound says that the precision with which we can estimate the parameters θ is bounded by the Fisher, i.e.:

$$Var(\widehat{\theta}) \geq \mathcal{F}^{-1}$$

Obviously if the true probability function is known, and the FIM can be computed, an efficient estimator can be found. However, as it often happens in machine learning, we don't know the model and if we assume a complex enough model then the Fisher is not tractable anymore. For example, neural networks have too many parameters for the Fisher to be tractable, and thus different approximations schemes will be used. We will see how knowing the Fisher, or even approximating it, can lead to significant improvements in learning performance (with additional computational cost) in a wide variety of models. We will then see how considering particular aspects of the geometry induced by the data can have beneficial aspects in parametric and non-parametric settings. The geometries considered reside on differentiable manifolds, the main object of study in differential geometry. We will investigate properties of these manifolds and how their analytical properties can help in reasoning about machine learning problems.

The thesis is structured as follows: In Chapter I we provide a short introduction to the field of Differential Geometry with its main objects of study and their properties. In Chapter II we describe a widely used application of Information Geometry to Machine Learning, the Natural Gradient and its variations. We will see how the natural gradient can be employed in a wide variety of settings. In Chapter III we describe two purely geometrical approaches to optimization for machine learning making use of the volume element, and we will see the profound insights that such modeling can bring to learning. In Chapter IV we show some applications of Information Geometric concepts to pattern recognition for vision, one of the fields of machine learning where IG was most fruitful.

Chapter 2

Background

Most of the following concepts are introduced as in [1] and [2]. As we mentioned in the introduction, we are interested in exploring properties of different geometries that can be associated with parametric model spaces as well as data spaces. These geometries can be associated with a set with specific properties, called a differentiable manifold. Information geometry uses methods from differential geometry, in particular Riemannian geometry to reason about these non-Euclidean manifolds that parametric models, and in particular, statistical models can be described as. Informally, a differentiable manifold is a generalization of smooth curves or smooth surfaces in higher dimensions. It is a set S of elements, or points, associated with a coordinate system. A *coordinate system* is a one-to-one mapping from S (or more specifically from on open subset of S) to $\mathbb{R}^n$. This n is called the dimension of the manifold. This means that each point on the manifold can be described as a set of n real numbers (note that each point is described by its n coordinates given by n coordinate functions). Each local neighborhood around a point is homeomorphic to a subset of $\mathbb{R}^n$, so we could say the manifold is locally Euclidean, while the map (called transition map) that takes us from one chart (from one neighborhood) to another, needs to be smooth. A set of all charts on the manifold is called an *atlas* and the equivalence class of atlases defines the type of manifold, for differentiable maps, this gives differentiable manifolds. For statistical manifolds each point on the manifold is a probability distribution (the formal definition of a statistical manifold is quite involved and not really relevant for the current work, thus we omit it here; the interested reader can see, for example [3]) . The coordinate system in such a manifold is exactly the set of parameters of the family of distributions $S = \{p(x, \xi)\}$ where x is a random variable and ξ is the set of parameters of the distributions. We give next the properties of a manifold. In this background chapter we use the Einstein summation convention, that is, indices that we find as superscripts *and* subscripts in the same formula are summed over. This is the general methodology in differential geometry.

Let S be a manifold and $\phi : S \to \mathbb{R}^n$ a coordinate system for S. Then ϕ maps each $p \in S$ to n real numbers, i.e.: $\phi(p) = [\xi^1(p), ..., \xi^n(p)] = [\xi^1, ..., \xi^n]$. These are called the coordinates of the point p. Each ξ^i is then a function that maps the point p to what is called its i^{th} coordinate, thus we can call ξ^i *coordinate functions*; this is just the component version of ϕ. The functions ξ^i are C^∞ (infinitely many times

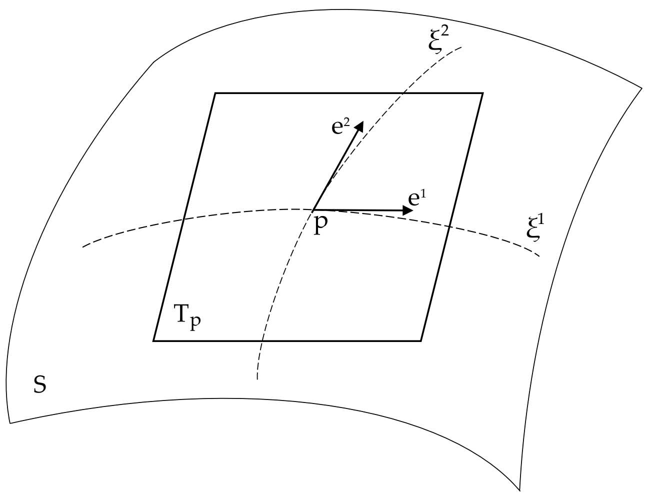


Figure 2.1: Tangent space. Figure from [1].

differentiable). Furthermore, in this context, noting that the order of integration and differentiation does not matter is of importance such that we can write:

$$\int \partial_i p(x;\xi)dx = \partial_i \int p(x;\xi)dx = \partial_i 1 = 0 \tag{2.1}$$

Instead of the ϕ notation we will use $[\xi^i]$. Many coordinates systems exist and usually we are interested in the transformation between different coordinate systems. An example of a transformation between two coordinate systems is considering the Cartesian and the polar coordinates in $\mathbb{R}^2$. Transforming from the polar (r,ϕ) to Cartesian coordinates (x,y) is given by the $sine$ and $cosine$ functions, i.e. $x = rcos\phi, y = rsin\phi$. A general transformation is trivial as follows (assume we want to transform from the representation in the $[\xi_i]$ system to a new representation in let's say $[\rho^i]$): apply the inverse mapping ϕ^{-1} (it is assumed that the inverse exists) to get the point p in S, and then apply the new transformation, say $\psi : S \to \mathbb{R}^n$. The the overall transformation looks like:

$$\psi \circ \phi^{-1} : [\xi^1,...,\xi^n] \to [\rho^1,...,\rho^n].$$

Formally, a C^∞ differentiable manifold has the following definition:

Definition 1
Let S be a set of elements or points. We say S is a C^∞ differentiable manifold if there is a set of coordinate systems $\mathcal{A}$ for S such that the following conditions are satisfied:

- $\forall \phi \in \mathcal{A}$ *is a one-to-one mapping from S to a open subset in $\mathbb{R}^n$*
- $\forall \phi \in \mathcal{A}$*, given any one-to-one mapping from S to $\mathbb{R}^n$ the following is true:*

$$\psi \in \mathcal{A} \Leftrightarrow \psi \circ \phi^{-1} \textit{is a } C^\infty \textit{ diffeomorphism.}$$

We consider a C^∞ *diffeomorphism* to be an isomorphism for which has the property that both the map and its inverse are C^∞ differentiable. Assume $\mathcal{A}$ contains a collection of coordinate charts whose domains cover S, then $\mathcal{A}$ is called an *atlas* for S.

Examples: (examples are taken from [4]).
Euclidean space: For each non-negative integer n, the Euclidean space $\mathbb{R}^n$ is an n-dimensional smooth manifold with the smooth structure given by the atlas with a single chart $(\mathbb{R}^n, Id_{\mathbb{R}^n})$. This is called the standard smooth structure on $\mathbb{R}^n$ and the coordinates are called the standard coordinates. The coordinate charts are given by (U,ϕ) with ϕ a diffeomorphism from U to another open subset $\widehat{U} \subseteq \mathbb{R}^n$.
Finite dimensional vector spaces: Let V be a finite dimensional vector space. Any norm on V determines a topology, which is independent of the choice of norm, in the sense that equivalent norms determine the same topology. The smooth structure defined on V is natural in the following sense. Let $(E_1,...,E_n)$ be an ordered basis for V, this defines an basis isomorphism $E : \mathbb{R}^n \to V$ by:

$$E(x) = \sum_{i-1}^{n} x^i E_i, \forall x \in \mathbb{R}^n$$

The map is a homeomorphism (invertible with continuous inverse, preserves the topological properties, is the equivalent of an isomorphism for topological spaces) so (V, E^{-1}) is a chart. Assume $(\widehat{E_1},...,\widehat{E_n})$ is another basis with $\widehat{E}(x) = \sum_j x^j \widehat{E_j}$ is the corresponding isomorphism, then there exists some invertible matrix (A_i^j) such that $E_i = \sum_j A_i^j \widehat{E_j}$ for each i. The transition map between the two charts is given by $\widehat{E}^{-1} \circ E(x) = \widehat{x}$ with $\widehat{x} = (\widehat{x}^1,...,\widehat{x}^n)$ given by:

$$\sum_{j=1}^{n} \widehat{x}^j \widehat{E_j} = \sum_{i=1}^{n} x^i E_i = \sum_{i,j=1} x^i A_i^j \widehat{E_j}$$

which gives $\widehat{x}^j = \sum_i A_i^j x^i$. This means that the map that sends x to $\widehat{x}$ is an invertible, linear map and thus a diffeomorphism, which means any two such charts are smoothly compatible. The collection of such charts is called the standard smooth structure on V.
We continue with an additional set of related notions refers to tangent vectors and tangent spaces which describe the idea of linear approximation for smooth manifolds. Just as a function of one variable can be approximated by its tangent line, or a surface in $\mathbb{R}^3$ by its tangent plane, or a parametrized curve in $\mathbb{R}^n$ by its velocity vector, the manifold can be approximated at a point p by its tangent space at point p, or a curve on the manifold passing through point p by its tangent vector at point p. A smooth coordinate chart (U,ϕ) gives a natural isomorphism from the space of tangent vectors (the collection of tangent vectors at a point forms the tangent space) to the manifold S at p to the space of tangent vectors to $\mathbb{R}^n$ at $\phi(p)$ which is isomorphic to the space of geometric tangent vectors at $\phi(p)$ which means that any smooth coordinate chart gives a basis for each tangent space. The disjoint union of all tangent spaces of the manifold forms the tangent bundle. The tangent bundle

is a fundamental concept in differential geometry and is itself a manifold of dimension $2n$ where n is the dimension of the original manifold. The differential of a map between manifolds, induces a bundle map between the tangent bundles of the manifolds. The tangent bundle is a very important concept in differential geometry. Formally, let $[\xi_i]$ be the coordinate system of a manifold S and let p be the point in S for which we want to define the tangent vector and the tangent space. We call $\mathbf{e}_i$ a *tangent vector*, the vector which is parallel to the i^{th} coordinate curve and goes through p. A coordinate curve is a curve defined by varying just one of the n numbers that define the coordinate system and keeping all other fixed. When considering all the $\mathbf{e}_i$ vectors at the point p, we see that they form an n-dimensional space, which is called the *tangent space* and which we denote by T_p (see Figure (2.1)).
Examples:(taken from [5])
Tangent vectors to points in $\mathbb{R}^n$: The standard coordinates on $\mathbb{R}^n$ yield standard coordinates on $T_p\mathbb{R}^n$. Let $e_i = (0, ..., 0, 1, 0, ..., 0)$ be the i-th canonical vector, and let $\alpha_i(t) = te_i + p$ be a path in $\mathbb{R}^n$, with $\alpha_i(0) = p$. Its equivalence class $[\alpha_i]$ is a vector in $T_p\mathbb{R}^n$ which is denoted by $\frac{\partial}{\partial x_i}\Big|_p$. In Calculus the ordered basis $\frac{\partial}{\partial x_1}\Big|_p$, ..., $\frac{\partial}{\partial x_n}\Big|_p$ is the basis in which the Jacobian is written and defines a natural isomorphism $T_p\mathbb{R}^n \cong \mathbb{R}^n$. This isomorphism is natural because $\mathbb{R}^n$ has a natural basis and is not any vector space. So if ρ is a path in $\mathbb{R}^n$, then $\rho'(0) \in T_{\rho(0)}\mathbb{R}^n$ via this isomorphism.

One important property of the tangent space follows (given as proposition 3.10 in [4]):

Proposition 1
If S is an n-dimensional smooth manifold, then for each $p \in S$, the tangent space T_pS is an n-dimensional vector space.

Until now we have been talking about notions that apply to general differential manifolds. For a manifold to be a Riemannian manifold, the metric defined on the manifold should be a Riemannian metric. For this we first need the following definitions. We first define what a tensor field is. For this we need the definition of multilinearity. Let $F : V_1 \times V_2 \times ... \times V_r \to W$ with $V_1, ..., V_r, W$ being linear spaces (also known as vector spaces). We say that F is a multilinear mapping if the following is true: for each i when fixing the value of all but one variable in the function F, we get this new function $\widehat{F_i} : V_i \to W$ which is a linear mapping from V_i to W. Then, for each point $p \in S$, we denote by $[T_p]_r^0$ be the family of multilinear mappings of the form $\underbrace{T_p \times ... \times T_p}_{\text{r direct products}} \to \mathbb{R}$ and we denote by $[T_p]_r^1$ be the family of multilinear mappings of the form $\underbrace{T_p \times ... \times T_p}_{\text{r direct products}} \to T_p$. A *tensor field* of covariant degree r and contravariant degree q is denoted by $[T_p]_r^q$ (we can also say that it is a tensor field of type (q, r)) and is a mapping as $H : p \to H_p$ and it maps each $p \in S$ to an element in $H_p \in [T_p]_r^q$. A vector field is then simply a tensor field of type (1,0). We can finally define what a Riemannian metric is. Let S be a manifold and $\langle , \rangle_p$ the inner product defined on the tangent space $T_p(S)$ (for all points $p \in S$). Let D and D' be any two tangent vectors in $T_p(S)$. Then the following is true: $\langle D, D' \rangle_p \in \mathbb{R}$. In addition, the following are satisfied:

$$\begin{aligned} \textit{linearity:} \quad & \langle aD + bD', D'' \rangle_p = a\langle D, D'' \rangle_p + b\langle D', D'' \rangle_p \forall a, b \in \mathbb{R} \\ \textit{symmetry:} \quad & \langle D, D' \rangle_p = \langle D', D \rangle_p \\ \textit{positive-definiteness:} \quad & \text{If } \; D \neq 0 \; \text{ then } \; \langle D, D \rangle_p > 0 \end{aligned}$$

The inner product defined on S is a tensor field of covariant degree 2, we denote this by $\langle , \rangle_p \in [T_p(S)]^0_2$. This inner product is known as the *Riemannian metric*. The Riemannian metric is not unique, there is an infinite number of Riemannian metrics on a manifold. To define a length on this manifold then we let $\gamma : [a,b] \to S, \gamma(a) = p_1, \gamma(b) = p_2$ be a curve in S (a differential path) then the length of this curve is given by:

$$\|\gamma\| = \int_a^b \left\| \frac{d\gamma}{dt} \right\| dt = \int_a^b \sqrt{g_{ij}\dot{\gamma^i}\dot{\gamma^j}} dt$$

where $\dot{\gamma^i}$ is the derivative of $\gamma^i = \xi^i \circ \gamma$. The tangent vector $\dot{\gamma}$ can be seen as an operator given by $\frac{d\gamma}{dt}$, where t is the variable which parametrizes the curve γ. A minimizer of this distance is called a geodesic path and is given by:

$$\gamma^* = \arg \min_{\gamma[0,1] \to S} \|\gamma\|$$

One can find such a minimum length curve by minimizing an energy function, which has the same minima as the original function, having the integrand squared:

$$E[\gamma] = \int_a^b g_{ij}\dot{\gamma^i}(t)\dot{\gamma^j}(t) dt$$

Solving this problem makes use of calculus of variations, and the Euler-Lagrange equation is derived that is satisfied by the minimizer.
In statistical modeling, an important property of tangent spaces is the fact that they approximate quantities on the manifold and it enables us to work in the tangent-space where we can use traditional vector-space techniques. But to map from the manifold to the tangent space we also need a well-posed mapping from the tangent space to the manifold. Such mappings are called retractions, and the exponential map and its inverse, the logarithmic map, are natural retractions on Riemannian manifolds. If we consider the set of all $n \times n$ orthogonal matrices with determinant 1 (this forms a Lie group [6]). If we consider a point $I \in SO(n)$ and then the tangent space $T_I(SO(n))$ is the space of all $n \times n$ skew-symmetric matrices, and considering the tangent space at any other point on the manifold $O \in SO(n)$, this is given by $OT_I(SO(n))$. Now we define the exponential of an $n \times n$ matrix by the following infinite series:

$$exp(V) = I + V + \frac{V^2}{2!} + \frac{V^3}{3!} + \ldots$$

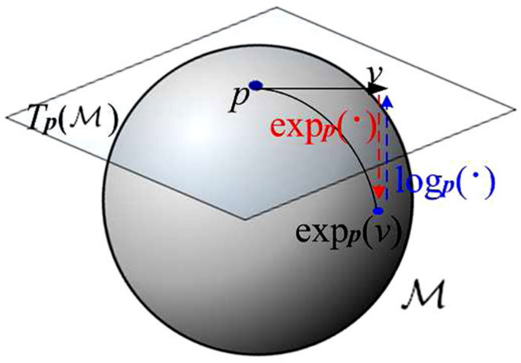


Figure 2.2: Exponential and logarithmic maps on $\mathbb{S}^2$. Figure from [7].

and taking the standard Euclidean inner product on the tangent space given by: $\langle V_1, V_2\rangle = trace(V_1 V_2^T)$ then a geodesic path is given by: $t \mapsto \alpha(t) = O\exp(tO^{-1}V)$ which is a constant speed geodesic with initial point $\alpha(0) = O$ and initial velocity $\dot{\alpha}(0) = V$. For a fixed V, the mapping forms a group action of $\mathbb{R}$ on $SO(n)$. This gives us the exponential map $\exp_O(V) : T_O SO(n) \to SO(n)$, which is given by the point reached by a constant speed geodesic with the specified initial condition; this gives $\alpha(1) = O\exp(tO^{-1}V)$. The inverse, the logarithmic map is given by: $\log_{O_1}(O_2) = O_1\log(O_1^{-1}O_2)$. In this case the mapping are given by matrix exponential and logarithmic.
Example: (taken from [7])
Unit sphere $\mathbb{S}^2$: Let $p \in \mathbb{S}^2$, then for every $v \in T_p(\mathbb{S}^2)$ we can define the exponential map as $\exp_p(v)$ which is given by projecting on the sphere the vector v, that is we measure a length $\|v\|$ along the geodesic, which is actually the great circle cut by the plane defined by v and the normal vector at p, starting from p in the direction of v. The point defined in this way on $\mathbb{S}^2$ is referred to as $\exp_p(v)$. We show the exponential and logarithmic maps in Figure (2.2).
TO DO - [formal definition of statistical manifold]

For a statistical manifold a valid Riemannian metric is considered to be the *Fisher metric* (we will see later a fundamental property of the Fisher metric on statistical manifolds), defined as (we write it as a matrix):

$$g_{ij} = \mathbb{E}_{p(x,\xi)}\left[\frac{\partial \log p(x,\xi)}{\partial \xi_i}\frac{\partial \log p(x,\xi)}{\partial \xi_j}\right]$$

Firstly, the squared distance between two infinitesimally close distributions $p(x,\xi)$ and $p(x,\xi + d\xi)$ is given by the double of the KL divergence:

$$ds^2 = \sum g_{ij}\partial\xi_i\partial\xi_j = 2KL(p(x,\xi)\|p(x,\xi + d\xi))$$

Then, the following observations are in order (first one according to Equation (2.1)):

$$E_\xi[\partial_i l_\xi] = 0 \tag{2.2}$$

$$g_{ij} = -E_\xi[\partial_i \partial_j l_\xi]$$

with $l_\xi = \log p(x;\xi)$.
Example: *The normal family of distributions.* The coordinates that completely define a Gaussian distribution are the mean and variance (we consider a one dimensional distribution), i.e. μ and σ^2, and when taking the derivatives of the log probability with respect to each of the parameters (or in the differential geometric perspective, coordinates): $\frac{\partial l}{\partial \mu} = \frac{x-\mu}{\sigma}$ and $\frac{\partial l}{\partial \sigma} = \frac{(x-\mu)^2}{\sigma^3} - \frac{1}{\sigma}$. Thus, the metric is given by:

$$g(\frac{\partial}{\partial \mu}, \frac{\partial}{\partial \sigma}) = 0$$
$$g(\frac{\partial}{\partial \mu}, \frac{\partial}{\partial \mu}) = \frac{1}{\sigma^2}$$
$$g(\frac{\partial}{\partial \sigma}, \frac{\partial}{\partial \sigma}) = \frac{2}{\sigma^2}$$

which gives an interesting geometric result, i.e. the normal family can be identified with the upper half plane, i.e.: $H = (x,y)|y > 0$ and substituting the metric ($x = \mu$ and $y = \sigma^2$) shows that the geometry of this Riemannian manifold is the hyperbolic geometry. Thus, the normal family manifold (one of the simplest family of distributions, but yet non-trivial) with the associated Fisher metric, gives rise to one of the simplest non-euclidean geometries. When considering the unit normal, i.e. $l(\mu)(x) = \frac{1}{2}\sum_i (x-\mu)^2$ we see that the Fisher is in this case the standard metric on $\mathbb{R}^n$, i.e.: $g_{ij} = \delta_{ij}$, which is the Euclidean metric: $g(u,v) = \sum_i u^i v^i$. We show in Figure (2.3) how geodesics look like in classic (a), source (b), natural (c) and expectation (d) parameters for the normal family. An interesting property of the invariance of the metric with respect to the choice of coordinate system, gives rise to *Jeffrey's prior* in Bayesian statistics, an often used uninformative prior. Let $S = \left\{p_\xi | \xi = [\xi^1, ..., \xi^n] \in \Theta\right\}$ be a statistical model and let $G(\xi)$ denote the Fisher information matrix at point ξ. Now, if we assume that the volume $V \triangleq \int_\Theta \sqrt{\det G(\xi)} d\xi$ is finite with respect to the Fisher metric (note that the integral is n-fold), then we can define $Q(\xi) \triangleq \frac{1}{V}\sqrt{\det G(\xi)}$ which is a probability density function on Θ. Because of the invariance over the choice of coordinate system, we can consider it as a probability density function on S. This prior has been shown to be significant in universal data compression [9].

2.1 Affine connection

Affine connections are an augmentation of a manifold such that there is a relation between two points p and q on the manifold through their tangent spaces T_p and T_q. For any Riemannian manifold there exists a unique affine connection with certain properties, that we will see later. Between others, it enables parallelism between

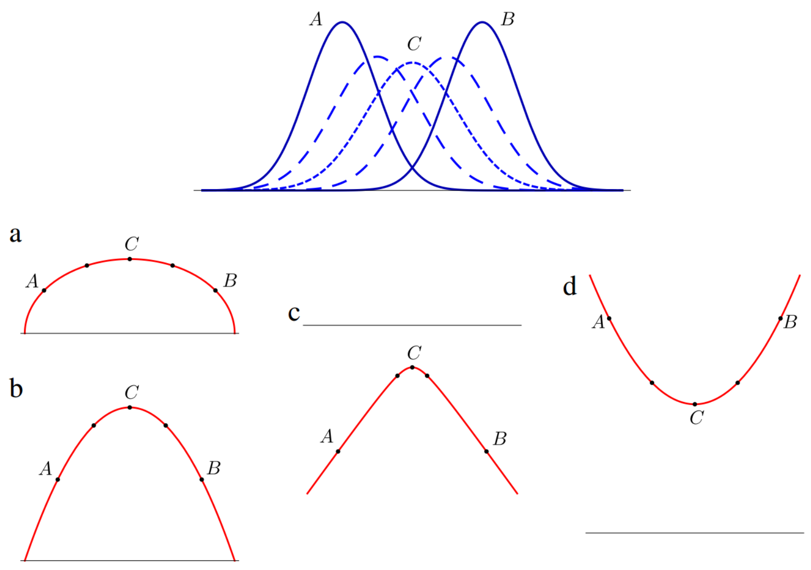


Figure 2.3: Shortest paths between normal distributions A and B in the corresponding half-planes: (a) classic parameters (μ, σ);(b) source parameters (μ, σ^2); (c) natural parameters $(\theta_1, \theta_2) = (\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2})$ and (d) expectation parameters $(\eta_1, \eta_2) = (\mu, \mu^2 + \sigma^2)$. Figure from [8].

different points on the manifold, or along a curve on a manifold. We will try to give next a short intuition about the need for connections on a manifold. The velocities which a point describes in its movement along a path of motion constitutes a vector field, referred to as the velocity field along the curve. If the path of motion is a straight line this means that the velocity is constant and thus the vector field associated with it. We can say that this vector field has a rate of change 0. When considering different geometries on the plane, not the Euclidean one, the straight lines now become curved lines according to the geometry, however the velocity vector fields associated with the new type of straight line should remain constant. This means that the description of the rate of change of vector fields should be modified

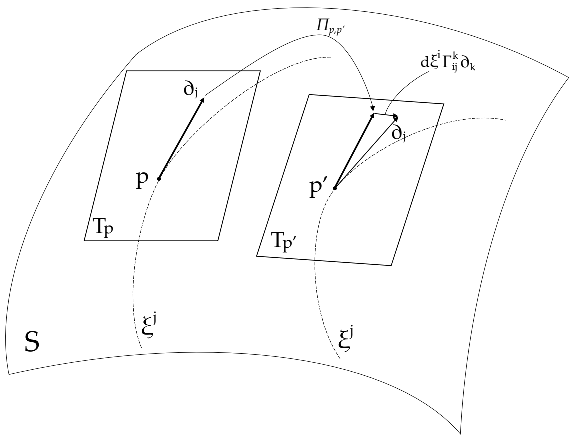


Figure 2.4: Affine connection (an infinitesimal translation). Figure from [1].

to take into account the new geometry. Other vectorial quantities related to vector fields need to be changed to accommodate for the new type of straight lines, for example differentials of functions need to be described in such a way as to maintain the natural relationships. [3] In short, given any two neighboring points (infinitesimally close to each other), an affine connection describes a linear one-to-one mapping between the tangent spaces of the two points. For statistical models a special type of affine connection is associated with a canonical divergence, a distance measure for statistical families. It also unveils some particularities of statistical manifolds with respect to the Fisher metric (we will show in Section ... how through Cencov theorem the Fisher metric is the only metric that is preserved under probabilistic meaningful transformations). Cencov introduced the α-connection and then Amari showed how special values for α give rise to the unique metric connection ($\alpha = 0$), while for the values $\alpha = 1$ and $\alpha = -1$, the exponential family and the mixture family respectively have 0 curvature, so that the connection is said to be flat. Because the α connection plays such in important role for statistical manifolds, giving rise to the Kullback-Leibler divergence and its variants, which are widely used in the machine learning community, moreover it is the means through which the theorem of Cencov shows the essential characteristic of the Fisher metric, we choose to define next the α-connection and from it derive the α-divergences. But first, we need the definition of an affine connection. Formally, given $p, p' \in S$ and $d\xi^i \triangleq \xi^i(p') - \xi^i(p)$, where $[\xi^i]$ is a coordinate system of S, we say that the two points are infinitesimally close to each other if we can ignore the second order infinitesimal, i.e. $(d\xi^i)(d\xi^j)$. To have a linear mapping $\pi_{p,p'}$ between T_p and T'_p we need to be able to express $\pi_{p,p'}((\partial_j)_p)$ as a linear combination of $\left\{(\partial_1)_{p'}, ..., (\partial_n)_{p'}\right\}$ where we denote by $\partial_j = \frac{\partial}{\partial \xi^j}$ the tangent vectors corresponding to each of the coordinate curves . So then the mapping we want is given by:

$$\pi_{p,p'}((\partial_j)_p) = (\partial_j)_{p'} - d\xi^i (\Gamma_{ij}^k)_p (\partial_k)_{p'}$$

where $\left\{(\Gamma_{ij}^k)_p; i, j, k = 1, ..., n\right\}$ are n^3 real numbers that depend on point p. To conclude, if for each pair of infinitesimally close points p and p', there exists such a linear mapping as defined above, (with $\pi_{p,p'} : T_p \rightarrow T_{p'}$) and if the Γ functions are C^∞, then we say we have an *affine connection* on S. The Γ functions are called the *connection coefficients* with respect to the coordinate system $[\xi^i]$. Through an affine connection we can describe the relation between two tangent spaces T_p and $T_{p'}$ of two infinitesimally close points p and p' in S, and to establish a relation between two farther away points we can consider such relations in sequence, however the relation depends also on the curve between the two points. Let us consider such a translation of tangent vectors along a curve. Let $\gamma : [a, b] \rightarrow S$ be a curve on the manifold S with $\gamma(a) = p$ and $\gamma(b) = q$. We define a *vector field along* γ to be a mapping from each $\gamma(t)$ to a tangent vector $X(t) \in T_{r(t)}$ (with $X : t \mapsto X(t)$). Now, if $\forall t \in [a, b]$ and the respective dt, the associated tangent vectors are linearly related as defined by the connection:

$$X(t + dt) = \pi_{\gamma(t),\gamma(t+dt)}(X(t))$$

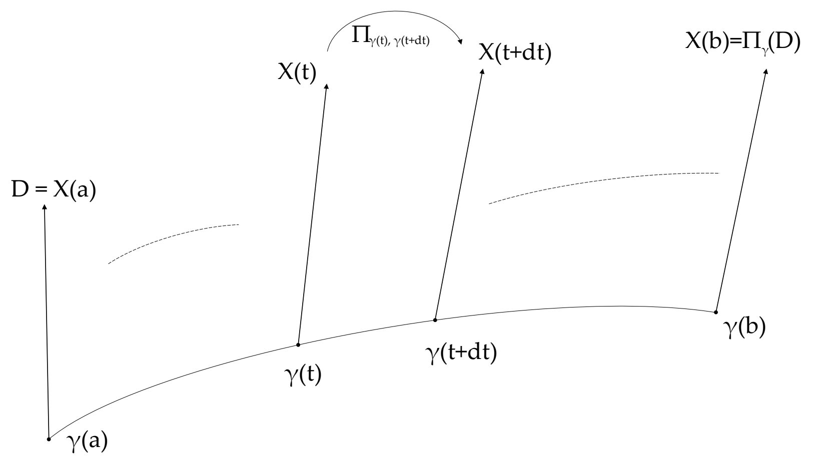


Figure 2.5: Parallel translation of a vector along a curve. Figure from [1].

then we say X is *parallel along* γ. We next need the notion of covariant derivative which can be easily defined considering first the usual derivative:

$$\frac{dX(t)}{dt} = \lim_{h\to 0}\frac{X(t+h)-X(t)}{h}$$

and then just replacing $X(t+h)$ with $\pi_{\gamma(t+h),\gamma(t)}(X(t+h))$ which gives us the *covariant derivative* denoted by $\frac{\delta X(t)}{dt}$ with $\delta X(t)$ given by (Figure (2.5)):

$$\delta X(t) = \pi_{\gamma(t+dt),\gamma(t)}(X(t+dt)) - X(t)$$

Now we define the directional derivative of a vector field $X = X^i\partial_i \in \mathcal{T}$ on S along a tangent vector $D = D^i(\partial_i)_p \in T_p$ (if the components of a vector field are C^∞ with respect to some system of coordinates then they are C^∞ with respect to any other system of coordinates, we denote this by $\mathcal{T}$). Let there be a curve on S whose tangent vector at point p is D, taking the covariant derivative of X along this curve we get:

$$\nabla_D X = D^i\left\{(\partial_i X^k)_p + X^j_p(\Gamma^k_{ij})_p\right\}(\partial_k)_p \in T_p(S)$$

We can define for each $X, Y \in \mathcal{T}(S)$ the vector field $\nabla_X Y \in \mathcal{T}(S)$ by $(\nabla_X Y)_p = \nabla_{X_p} Y \in T_p(S)$. This is called the *covariant derivative* of Y with respect to X. Replacing X with $X^i\partial_i$ and Y with $Y^i\partial_i$ we have:

$$\nabla_X Y = X^i\left\{\partial_i Y^k + Y^j\Gamma^k_{ij}\right\}\partial_k$$

And when considering $X = \partial_i$ and $Y = \partial_j$ we have the component version of the covariant derivative:

$$\nabla_{\partial_i}\partial_j = \Gamma^k_{ij}\partial_k \tag{2.3}$$

This can be verbalized as the vector field that describes the change in the basis vector ∂_j as it is moved in the direction of the basis vector ∂_i. We will define here, another set of functions, which can be considered a different set of coefficients for the same connection ∇:

$$\Gamma_{ij,k} = \langle \nabla_{\partial_i} \partial_j, \partial_k \rangle = \Gamma_{ij}^{h} g_{hk}$$

where g is the Riemannian metric associated with the manifold S. The operator that we defined by $\nabla : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$ that maps (X,Y) to $\nabla_X Y$ satisfies the following: $\forall X, Y, Z \in \mathcal{T}$ and $f \in \mathcal{F}$ (the set of C^∞ on S):

- $\nabla_{X+Y} Z = \nabla_X Z + \nabla_Y Z$
- $\nabla_X (Y + Z) = \nabla_X Y + \nabla_X Z$
- $\nabla_{fX} Y = f \nabla_X Y$
- $\nabla_X (fY) = f \nabla_X Y + (Xf) Y$

Because $\nabla_X Y$ is multilinear with respect to X but not with respect to Y, ∇ is not a tensor field. In short, an affine connection on S is a mapping $\nabla : \mathcal{T}(S) \times \mathcal{T}(S) \to \mathcal{T}(S)$ that satisfies the above conditions and the connection coefficients are given by Equation (2.3). We next define what a flat connection is, but first we need the notion of parallelism on a manifold. Let $X \in \mathcal{T}(S)$ be a vector field on S and γ be a curve on S. Then, $\forall \gamma$ if $X_\gamma : t \to X_{\gamma(t)}$ is parallel along γ with respect to the affine connection ∇, then we can say that X is *parallel on S* with respect to the connection ∇. For $X = X^i \partial_i$ to be parallel on S it is necessary and sufficient that $\nabla_Y X = 0 \ \ \forall Y \in \mathcal{T}(S)$ or:

$$\partial_i X^k + X^j \Gamma_{ij}^k = 0$$

Now consider a manifold S and a coordinate system on S, say $[\xi^i]$. If the n basis vector fields, i.e. $\partial_i = \frac{\partial}{\partial \xi^i}$ with $i = 1, \dots, n$ are all parallel on S then we can say $[\xi_i]$ is an *affine coordinate system* for ∇. This is equivalent to $\nabla_{\partial_i} \partial_j = 0$ and also with all the connection coefficients of ∇ with respect to $[\xi_i]$, i.e. $\left\{\Gamma_{ij}^k\right\}$ being all identically equal to 0. Now we can state the definition of a flat connection: if an affine coordinate system exists for ∇ then we say ∇ is flat for we can say that S is flat with respect to ∇.

2.2 The α-connection

As we mentioned in the previous section, the α-connection plays an important role for statistical manifolds, through the divergences it defines. Moreover, one of the fundamental results in information geometry, Cencov theorem, shows how the Fisher metric is invariant to certain transformations and the α-connection is uniquely associated with the Fisher metric on the manifold. We proceed to the definition of the α-connection and its implications. Let $S = p_\xi$ be an n-dimensional model (or manifold) and consider again the set of n^3 functions (similar to the connection coefficients) which map each point on the manifold to the following value:

$$(\Gamma^{(\alpha)}_{ij,k})_\xi = E_\xi \left[\left(\partial_i \partial_j l_\xi + \frac{1-\alpha}{2} \partial_i l_\xi \partial_j l_{xi} \right) (\partial_k l_\xi) \right]$$

where α can be any real number. This gives an affine connection ∇^α on S given by:

$$\langle \nabla^\alpha_{\partial_i} \partial_j, \partial_k \rangle = \Gamma^\alpha_{ij,k}$$

where the metric $\langle , \rangle$ is given by the Fisher metric. This ∇^α is called the *α-connection*. ∇^α is a symmetric connection. We note that ∇^α satisfies:

$$\begin{aligned} \nabla^\alpha &= (1-\alpha)\nabla^0 + \alpha \nabla^1 \\ &= \frac{1+\alpha}{2}\nabla^1 + \frac{1-\alpha}{2}\nabla^{-1} \end{aligned}$$

To note here is that in general, when $\alpha \neq 0$, ∇^α is not a metric connection. We define next the metric connection: let ∇ be an affine connection on the Riemannian manifold $(S, g = \langle , \rangle)$, and for all vector fields $X, Y, Z \in \mathcal{T}(S)$, the following is always true:

$$Z\langle X, Y \rangle = \langle \nabla_Z X, Y + \langle X, \nabla_Z Y \rangle$$

In this case, ∇ is called a *metric connection* on S with respect to the metric g. Rewriting this using the component expressions, we get:

$$\partial_k g_{ij} = \Gamma_{ki,j} + \Gamma_{kj,i}$$

A connection, which is both metric and symmetric (i.e. $\Gamma_{ij,k} = \Gamma_{ji,k}$) is called a *Riemannian connection* or *Levi-Civita connection*. To note here is, that for any valid Riemannian metric g, such a connection is unique. The following theorem is in order, considering $\alpha = 0$.

Theorem 1
The 0-connection is the Riemannian connection with respect to the Fisher metric.

2.3 Exponential family

The exponential family, probably the most used family of probability distributions, has also some special properties when it comes to considering the α-connection on the manifold of exponential probability distributions. We will show next, how, if a family is exponential, the manifold that it defines admits a connection with $\alpha = 1$ which is flat, i.e. the connection vanishes everywhere, that is the coefficients of the connection are all identically 0. The divergence associated with such a manifold is the well-known KL-divergence, which is used, for example, in (among many others learning methods) the maximum likelihood estimation for exponential models. Minimizing the KL-divergence in variational methods, is equivalent to maximizing the lower bound as we will see in Section (3.2). We now introduce the exponential family which, we said, is related to ∇^1. If a statistical model can be expressed in terms of functions $C, F_1, \ldots, F_n$ where C and F_i are defined on $\mathcal{X}$ and ψ is defined on Θ, as follows:

$$p(x;\theta) = \exp\left[C(x) + \sum_{i=1}^{n} \theta^i F_i(x) - \psi(\theta)\right] \tag{2.4}$$

then we say that the set of models S is an *exponential family* and $[\theta^i]$ are its *natural parameters*. The following relations are in order (recall that we denote $\partial_i = \frac{\partial}{\partial \theta^i}$):

$$\partial_i l(x;\theta) = F_i(x) - \partial_i \psi(\theta)$$
$$\partial_i \partial_j l(x;\theta) = -\partial_i \partial_j \psi(\theta)$$

This results in $\Gamma^1_{ij,k} = -\partial_i \partial_j \psi(\theta) E_\theta[\partial_k l_\theta]$ which we know is 0 from Equation (2.2). Differently put, we see that $[\theta^i]$ is a 1-affine coordinate system for S, thus, we can say that S is 1-flat. Thus, the connection given by ∇^1 is called the *exponential connection*.

A dual notion (we will see shortly what we mean by dual in this context), is the *mixture connection*. Given a similar set of functions C, F_i, if a point on the manifold (or a member of the family described by S) can be written as:

$$p(x;\theta) = C(x) + \sum_{i=1}^{n} \theta^i F_i(x)$$

then we say that S is a mixture family with mixture parameters $[\theta^i]$. A well-known form for such a distribution is given by:

$$p(x,\theta) = \sum \theta^i p_i(x) + (1 - \sum \theta^i) p_0(x)$$

with $p_i(x)$ being the components of the mixture distributions and, $0 < \theta^i < 1$ and $\sum \theta^i < 1$. The exponential-analogue observations follow:

$$\partial_i l(x;\theta) = \frac{F_i(x)}{p(x;\theta)}$$
$$\partial_i \partial_j l(x;\theta) = \frac{-F_i(x) F_j(x)}{p(x;\theta)^2}$$

which gives that $\partial_i \partial_j l + \partial_i l \partial_j l = 0$ which results in $\Gamma^{-1}_{ij,k}$ being all 0. As mentioned, this makes S to be -1-flat. To summarize:

Theorem 2
An exponential family (mixture family) is 1-flat or e-flat (-1-flat or m-flat) and its natural parameters (mixture parameters) form an e-affine (m-affine) coordinate system.

It turns out that an exponential family is also m-flat, while a mixture family is also e-flat. For details see [1]. In general, a model which is α-flat need not be an exponential family or a mixture family, however if it is α-flat for all α this means it is an

Euclidean space with respect to the Fisher metric. We talked about the α-connection and its role for exponential families, however the α-connection plays a central role in defining the invariance properties of the Fisher, regardless of the type of statistical manifold in question. We proceed with the preamble for giving Cencov theorem, a turning point for information geometry.
One of the first fundamental results in statistics with respect to the Riemannian geometry is given by [10] and refers to the unique properties of the Fisher metric as the Riemannian metric for statistical manifolds and also to the respective α-connection on the manifold. As we said in the definition of the Riemannian metric, there can be an infinite number of Riemannian metrics associated with a manifold, as well as an infinite number of affine connections. The most important property of the Fisher metric and of the α-connection is the invariance property with respect to the sufficient statistics: let $S = \{p(x;\xi)\}$ be a model on $\mathcal{X}$ as before and let $F : \mathcal{X} \to \mathcal{Y}$ induce another model given by $S_F = \{q(y;\xi)\}$. If F is the sufficient statistics for S then the following is true:

$$\partial_i \log p(x;\xi) = \partial_i \log q(F(x);\xi) \tag{2.5}$$

and thus g_{ij} and the $\Gamma^{\alpha}_{ij,k}$ are the same on both S and S_F. We now can say that the Fisher metric and the α-connection are *invariant with respect to the sufficient statistics* F. To formally state Cencov theorem we first need some preliminaries. We consider a manifold which has as points, probability distributions on a finite set. Let $\{(g_n, \nabla_n)\}_{n=1}^{\infty}$ be a sequence of Riemannian metrics and affine connections on $\mathcal{P}_n$ (with $\mathcal{P}_n = \{\mathcal{X}_n\}$ and with $\mathcal{X}_n = \{0, 1, ..., n\}$). If S is a model on $\mathcal{X}_n$ and $F : \mathcal{X}_n \to \mathcal{X}_m$ $(n \geq m)$ is a surjective function then we know that (g_n, ∇_n) and (g_m, ∇_m) are valid metrics and connections on $S \subset \mathcal{P}_n$ and $S_F \subset \mathcal{P}_m$. The theorem follows from [1]:

Theorem 3 (Cencov theorem)
We make the assumption that $\{(g_n, \nabla_n)\}_{n=1}^{\infty}$ is invariant with respect to the sufficient statistics $F : \mathcal{X}_n \to \mathcal{X}_m$ such that the metrics and connections on S and S_F are assumed to be invariant. Then there exists a positive number c and a real number α, such that $\forall n$, g_n is the Fisher metric on $\mathcal{P}_n$ scaled by a factor of c and ∇_n is the α-connection on $\mathcal{P}_n$.

This is the fundamental result that shows that the Fisher metric and the α-connection are invariant with respect to the sufficient statistics. This property is uniquely met by the Fisher and the α-connection. However this is true for finite sets, for infinite sets $\mathcal{X}$, the discussion is a bit different and uses a kind of limiting procedure, it takes into account the limits of the Fisher and the α-connection, which under some regularity conditions coincide with the actual Fisher and α-connection.

2.4 Dual connections

[1] dedicate a whole chapter to the study and properties of duality of the α-connections on statistical manifolds with their respective Fisher metric. In particular, they consider the triple $(g, \nabla^{\alpha}, \nabla^{-\alpha})$ and state that this duality is of fundamental significance

when studying the geometric structure of statistical models. We give next, the definition of dual connection (or conjugate connection). Given a manifold S and an associated Riemannian metric $g = \langle , \rangle$ on S and two affine connections ∇ and ∇^*, then if for any vector fields $X, Y, Z \in \mathcal{T}(S)$ the following holds:

$$Z\langle X, Y\rangle = \langle \nabla_Z X, Y\rangle + \langle X, \nabla^*_Z, Y\rangle$$

then we call ∇ and ∇^* *duals* with respect to the metric g. When considering the system of coordinates $[\xi^i]$ and the coordinate expressions $g_{ij}, \Gamma_{ij,k}$ and Γ_{ij,k^*} of the triplet, then the following is true:

$$\partial_k g_{ij} = \Gamma_{ki,j} + \Gamma^*_{kj,i}$$

To note here is that $(\nabla^*)^* = \nabla$ and that even though each connection by itself is not metric, $\frac{(\nabla+\nabla^*)}{2}$ is a metric connection. The following theorem is in order:

Theorem 4
For any statistical model, or any manifold of finite measure, the α-connection and the $(-\alpha)$-connection are dual with respect to the Fisher metric.

2.4.1 Divergence

We give next the definition of an important class of divergences on statistical models, the so called *f-divergence*. Let $f(z)$ be a convex function on $z > 0$. For two probability distributions p, q the f-divergence is defined as:

$$D_f(p|q) = \int p(x) f(\frac{q(x)}{p(x)})$$

We see that the α-divergences are given by a special class of the f-divergences. In particular:

$$f^\alpha(z) = \begin{cases} \frac{4}{1-\alpha^2}\left\{1 - z^{(1+\alpha)/2}\right\} & \text{if } \alpha \neq \pm 1 \\ z \log z & \text{if } \alpha = 1 \\ -\log z & \text{if } \alpha = -1 \end{cases}$$

For $\alpha = 0$ we get a valid distance measure, known as the *Hellinger distance*, given by:

$$D^0 = 2\int (\sqrt{p(x)} - \sqrt{q(x)})^2 dx$$

whereas for $\alpha = \pm 1$ we get the *Kullback-Leibler* divergence $KL(p||q) = \int p(x) \log(\frac{q(x)}{p(x)})$ and its inverse $KL^{-1}(p||q) = \int q(x) \log(\frac{p(x)}{q(x)})$. When considering two distributions $p(x)$ and $q(x)$, the geodesics between them induced by the ∇^{-1} (mixture) and ∇^1 (exponential) are given by:

$$\gamma_\lambda(x) = (1-\lambda)p(x) + \lambda q(x)$$

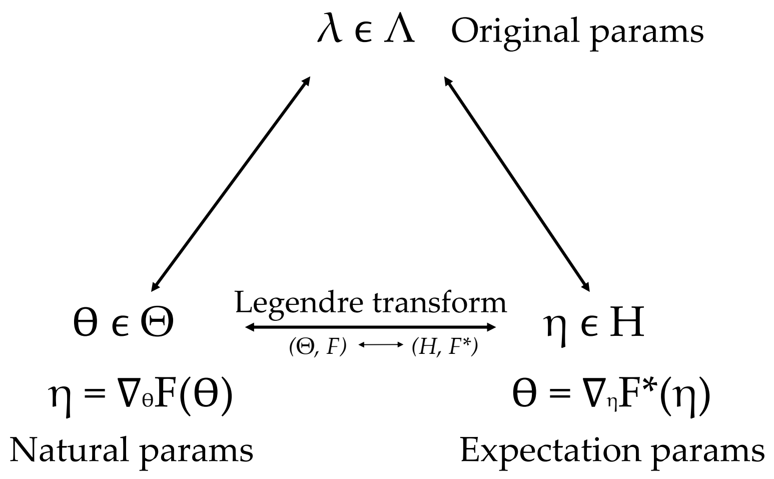


Figure 2.6: Dual parametrization of exponential families from Legendre transformation. Figure from [11]

$$\log \gamma_\lambda(x) = (1-\lambda)\log p(x) + \lambda \log q(x) - \log Z_\lambda$$

with $Z_\lambda = \int f^{1-\lambda}(x) g^\lambda(x) dx$ being the normalization coefficient or partition function. The canonical divergence between two exponential families turns out to be the Kullback-Leibler divergence, given by:

$$d(p(x;\theta_1)||p(x;\theta_2)) = F(\theta_1) + F^*(\eta_2) - \langle \theta_1, \eta_2 \rangle$$

with F^* being the Legendre-Fenchel dual of F given by:

$$F^*(\eta) = \sup_{\theta \in \mathcal{P}_\Theta} \langle \theta, \eta \rangle - F(\theta)$$

Some interesting properties follow. Taking the gradient of F gives the moment parameter η: $\nabla F(\theta) = \eta$ while taking the gradient of the dual gives the natural parameter: $\nabla_\eta F^*(\eta) = \theta$. The diagram in Figure (2.6) describes the dual parametrization of the exponential family. Another important aspect of the divergence between two exponential family distributions is the fact that the canonical divergence can be written as the Bregman divergence on the natural parameter space:

$$d(p(x;\theta_1)||p(x;\theta_2)) = B_F(\theta_1||\theta_2) = F(\theta_1) + F^*(\theta_2) - \langle \theta_1 - \theta_2, \nabla F(\theta 2)$$

Moreover, the following is true: $B_F(\theta_1||\theta_2) = B_{F^*}(\eta_2||\eta_1)$. The duality of the exponential family and the peculiar relation between the different parametrizations is used fruitfully throughout machine learning, for example in Section (3.2) we use the fact that the expectation of the sufficient statistics is the gradient of the log-normalizer and also the fact that the Hessian of the log-normalizer is the covariance matrix of the sufficient statistics if the family is a minimal exponential family, that is the functions F in Equation (2.4) are linearly independent . In [12] an important result is achieved for geodesic paths making use of this duality. For more details regarding the exponential family and the relations between different parametrization can be found for example in [11] or [1].

2.5 Conclusion

There are many statistical concepts, like divergence for example, that have a special geometric meaning. In different parametrization, the divergence function has very interesting properties that can shed light on the nature of the problem and the particularities that a certain parametrization can have. The relatively new field of information geometry is gaining ground as more concepts from differential geometry gain meaning in statistics. The field of differential geometry encompasses notions from many mathematical fields, like topology, calculus, algebra and geometry. It is an old mathematical discipline that is used for example in General and Special Relativity and has a wide range of complex tools available.

Chapter 3

Natural gradient

On of the most important concepts that comes from differential geometry to machine learning and optimization is the natural gradient. Consider the following problem: we search for the minimum of a function, which is dependent on some parameters (the parameters of the statistical model). When using the conventional gradient, the assumption is that the function depends in the same way on each of the parameters, but this is almost never the case, as the data induces a geometry in the model, a particular dependency, but also the model itself usually has a different geometry than the Euclidean one, which is the one assumed for the standard gradient. Thus, to be able to take into account the existing dependency of the function on the parameters we need a more complex gradient definition. The original work of [13] on the natural gradient dealt with the optimization of the multilayer perceptron in the narrow case where we assume a Gaussian input (he also assumes that the neurons have on average a Gaussian activation, but this is not so constraining as the first assumption). The natural gradient has at its roots the seminal work of [14], saying that the Riemannian structure of a statistical model is defined by the Fisher information. Among the properties of the Fisher information, we mentioned earlier the Chentsov theorem stating that this is the only invariant metric of a statistical model. The work of Amari starts with the following theorem [13]:W

Theorem 5
The steepest descent direction in a Riemannian manifold, where w, is the parameter vector, G is the associated metric and $\nabla L(w)$ is the normal gradient is given by:

$$-\widehat{\nabla} L = -G^{-1}(w)\nabla L(w)$$

and $\nabla L(w)$ is given by:

$$\nabla L(w) = \left(\frac{\partial}{\partial w_1} L(w), \ldots, \frac{\partial}{\partial w_n} L(w) \right)^T$$

The proof is quite simple for such a significant result. $L(w)$ is the function that we want to minimize at $L(w + dw)$ where $||dw||^2 = \epsilon^2$ is fixed for a small ϵ. We want to find this dw that minimizes the function $L(w)$.

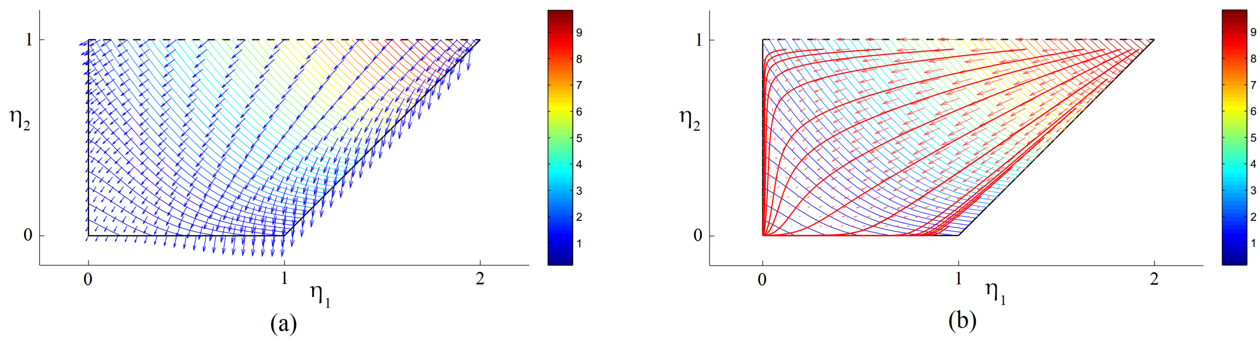


Figure 3.1: Vanilla gradient field (a) and natural gradient field (b) together with level lines over the marginal polytope of a discrete exponential distribution. The axes represent the expectation parameters. Figure from [15].

Proof. Let $dw = \epsilon a$, we search for a that minimizes $L(w + dw) = L(w) + \epsilon \nabla L(w) \cdot a$ under the following constraint: $||a||^2 = \sum g_{ij} a_i a_j = 1$. Forming the Lagrangian we then have:

$$\frac{\partial}{\partial a_i} \left\{ \nabla L(w)^T a - \lambda a^T G a \right\} = 0$$

After differentiating we get:

$$\nabla L(w) = 2\lambda G a$$

And then we get:

$$a = \frac{1}{2\lambda} G^{-1} \nabla L(w)$$

The natural gradient is then defined to be: $\widehat{\nabla} L(w) = G^{-1} \nabla L(w)$. If we work in Euclidean space and also in an orthonormal coordinate system, the two gradients are equal, i.e. $\widehat{\nabla} L = \nabla L$. The update of the weight vector w is given by:

$$w_{t+1} = w_t - \eta_t \widehat{\nabla} L(w_t)$$

where η_t is the learning step dependent on time. We show in Figure (3.1) an example of vanilla and natural gradient field on the marginal polytope of $F(p) = \mathbb{E}_p[f]$ with $f = c_0 + c_1 x_1 + c_2 x_2 + c_{12} x_1 x_2$ and $c_0 = 0,_1 = 1, c_2 = 2, c_{12} = 3$. The marginal polytope is the convex span of the values of the sufficient statistics of the distribution p which is a discrete exponential family distribution. For details see [15].

3.1 Relation to the KL divergence

The natural gradient can also be defined as the direction of descent that keeps the change in KL-divergence constant [16; 17], some small value. Let us assume the functions that map the parameters $w \in \mathbb{R}^m$ to probability density functions $p(x)$, with $p : \mathbb{R}^n \to [0, \infty)$ with $x \in \mathbb{R}^n$. We again look at the same function $L(w + dw)$ where we try to find dw such that we minimize $L(w + dw)$, formally:

$$\operatorname*{argmin}_{dw} L(w + dw) \text{ such that } KL(p_w \| p_{w+dw}) = constant \tag{3.1}$$

This means that we will move along the statistical manifold of the probability distributions defined by parameters w with constant speed, in the KL-sense. (KL has been shown to behave like a distance measure locally, see [1]). Because we want a local infinitesimal movement, we can safely assume that $dw \to 0$. Approximating the KL divergence by the second-order Taylor expansion we get:

$$\begin{aligned} KL(p_w \| p_{w+dw}) &\approx (\mathbb{E}_x[\log p_w] - \mathbb{E}_x[\log p_w]) \\ &\quad - \mathbb{E}_x[\nabla \log p_w(x)]\Delta w - \frac{1}{2}\Delta w^T \mathbb{E}_x[\nabla^2 \log p_w]\Delta w \\ &= \frac{1}{2}\Delta w^T \mathbb{E}_x[-\nabla^2 \log p_w(x)]\Delta w \\ &= \frac{1}{2}\Delta w^T \mathbf{F} \Delta w \end{aligned}$$

where $\mathbf{F}$ is the Fisher matrix. We saw in Equation (2.2) that $\mathbb{E}_x[\nabla \log p_w(x)] = 0$ and so just the last term remains in the equation above. Using this approximation for the KL divergence we write the Lagrangian of Equation (3.1), approximating $L(w + dw)$ by its first-order Taylor expansion:

$$L(w) + \nabla L(w)\Delta w + \frac{1}{2}\lambda dw^T \mathbf{F} \Delta w$$

Solving this equation to get dw, we get the natural gradient. To keep in mind here is that these approximation make sense just around w. [18] shows that taking larger steps harms convergence. We can get the Fisher information matrix $\mathbf{F}$ by taking the expectation of the negative Hessian, as shown in [16]:

$$\begin{aligned} \mathbb{E}_x\left[-\frac{\partial^2 \log p_w}{\partial w}\right] &= \mathbb{E}_x\left[-\frac{\partial \frac{1}{p_w}\frac{\partial p_w}{\partial w}}{\partial w}\right] \\ &= \mathbb{E}_x\left[-\frac{1}{p_w(x)}\frac{\partial^2 p_w}{\partial w^2} + \left(\frac{1}{p_w}\frac{\partial p_w}{\partial w}\right)^T \left(\frac{1}{p_w}\frac{\partial p_w}{\partial w}\right)\right] \\ &= -\frac{\partial^2}{\partial w^2}\left(\sum_x p_w(x)\right) + \mathbb{E}_x\left[\left(\frac{\partial \log p_w(x)}{\partial w}\right)^T \left(\frac{\partial \log p_w(x)}{\partial w}\right)\right] \\ &= \mathbb{E}_x\left[\left(\frac{\partial \log p_w(x)}{\partial w}\right)^T \left(\frac{\partial \log p_w(x)}{\partial w}\right)\right] \end{aligned}$$

Intuitively, as the local KL-divergence is invariant to reparametrization, because it measures a difference between two probability density functions, and because the natural gradient is strongly related to it, the natural gradient is also invariant to

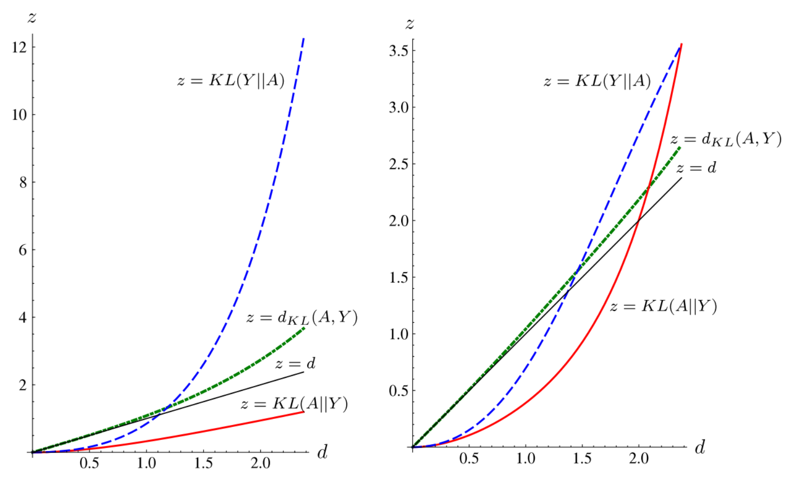


Figure 3.2: Kullback-Leibler divergence compared to the Fisher distance along geodesics.

reparametrization, but we will discuss more about this in later sections. Another perspective of this is considering the natural gradient as doing whitening in parameter space [19]. From the geometric perspective, the natural gradient gives the geodesic path from the current distribution to the target distribution. We show in Figure (3.2) an example for the univariate normal showing $KL(A\|Y)$ (red continuous curve), $KL(Y\|A)$ (blue dashed curve) and the symmetrized $d_{KL}(A,Y)$ (green dot-dashed curve), compared to the Fisher distance d (black line) varying in the interval $[0, 2.3769]$. For more details see [8].

In the following sections of this chapter, we will show a few applications of the natural gradient to machine learning problems. We will first describe the similarity of the natural gradient with the coordinate ascent updates in the context of stochastic variational inference (SVI), a relatively new stochastic version of variational inference, that is a highly scalable version of the standard mean-field variational inference. We will then turn to deep architectures and we will describe shortly how the natural gradient is seen in the context of deep networks, in particular we will see the similarities and differences compared to other optimization algorithms, like the Gauss-Newton or Krylov subspace methods. After this short perspective, we will then describe in more detail a new type of deep architecture that forces the Fisher to be identity through whitening. Because we are interested also in how the general information geometric perspective can be used for deep learning, we show a simple application to classification with contractive autoencoders, when using the main tangent directions defined by the encoder with respect to the input. Even though it is not a natural gradient problem, it is very related to it and offers a different perspective on the directions of interest in the learning space, more specifically we see how the data induces a geometry in the model and how we can use this for efficient learning. After this, we turn to another application of the natural gradient, this time to reinforcement learning and, in particular, to actor-critic methods. All these applications are meant to show the reader the wide applicability of the natural gradient to machine learning problems, and the improvements that can be obtained when considering a non-Euclidean geometry in the learning space, which is defined by the data space combined with the model space. We advocate that, even though learning can be significantly improved when considering the particular model geometry, it can be further enhanced when considering the geometry that the data induces into the model, which is a further particularization of the overall model geometry.

3.2 Stochastic variational inference

The research described next can be found in [20]. When considering large datasets, for example documents, and we want to make some sense out of them, find structure and be able to order them according to topics, themes, etc. we need an appropriate framework that is both scalable and is able to explore the dataset in a consistent manner. In the language of statistical machine learning, we need a graphical model with an efficient posterior inference algorithm that is suitable for finding structure in the documents. In a more formal description, assume we have the observations x_n, which we assume depend on some local hidden variables, z_n and some global hidden variables β. The local variables also depend on the global variables, which themselves depend on a hyperparameter α. The graphical model representing this description is in Figure (3.3). The graphical model depicted represents a general class of graphical models where we have observations and local and global hidden variables. The model is important because each observation depends on its local hidden variable and on the global hidden variables. Moreover, each observation and local hidden variable associated are conditionally independent given the global hidden variables, of all the other local hidden variables and other observations. This makes the joint distribution of all the variables, factorizable into a global term and a product of local terms which usually makes inference tractable. This type of model is often used in Bayesian inference as it captures the local structure into the local hidden variables but keeps a global hidden structure common to all the observations. Particularizing this model gives rise to numerous applications in machine learning, like mixture of Gaussians (the global variables would be the mixture proportions and the means and variances of the Gaussians, while the local variable would be the cluster label for an observation) , and as the authors show LDA, HDP, etc. The joint distribution governing the variables in the graphical model can be written as:

$$p(x,z,\beta|\alpha) = p(\beta|\alpha)\prod_{n=1}^{N} p(x_n, z_n|\beta)$$

The goal in this context is to know more about the hidden variables, i.e. compute the posterior given the data: $p(\beta, z|x)$. The difference between local and global hidden variables lies in the conditional dependencies in the graphical model, in this case the observation x_n and local hidden variable z_n are conditionally independent given the global variables, of all other local hidden variables and observations. Formally this is written as:

$$p(x_n, z_n|x_{-n}, z_{-n}, \beta) = p(x_n, z_n|\beta)$$

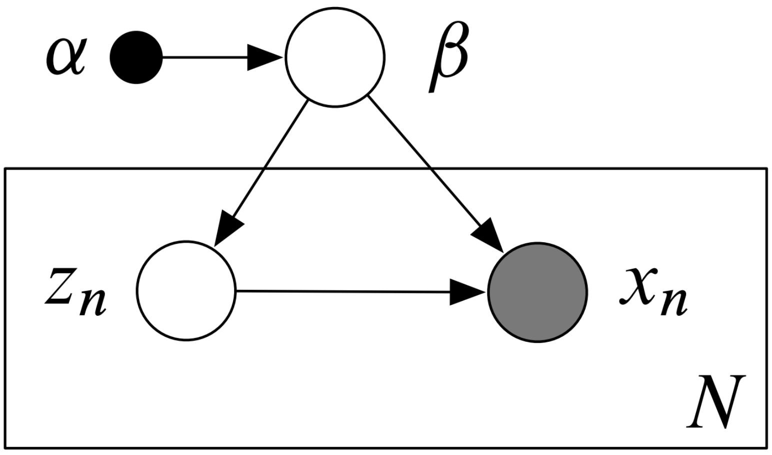


Figure 3.3: General graphical model example.

where we have denoted by x_{-n}, z_{-n} all the variables (observed and local hidden) excluding the nth one. The authors assume the conditionals are in the exponential family, we show the form here to also define the notation used:

$$p(\beta|x,z,\alpha) = h(\beta)\exp\left\{\eta_g(x,z\alpha)^T t(\beta) - a_g(\eta_g(x,z,\alpha))\right\}$$
$$p(z_{nj}|x_n, z_{n,-j}, \beta) = h(z_{nj})\exp\left\{\eta_l(x_n, z_{n,-j}, \beta)^T t(z_{nj}) - a_l(\eta_l(x_n, z_{n,-j}, \beta))\right\}$$

The functions $h(\cdot)$ and $a(\cdot)$ are the base-measure and the log-normalizer of the distribution and are scalar functions, while $\eta(\cdot)$ and $t(\cdot)$ are vector functions and are the natural parameters and the sufficient statistics of the distribution. The sufficient statistics of a distribution completely summarizes that distribution. If the family is a minimal exponential family (i.e. the functions F are linearly independent), the expectation of the sufficient statistics gives the moments of the distribution and there exists a one to one mapping between the moments and the natural parameters [1]. The natural parameter space for an exponential family is always open convex, so it is desirable to usually work in this space when trying to estimate a density function. Moreover, the gradient of the log normalizer with respect to the natural parameters give the expected sufficient statistics of the same order as the derivative. For a second order derivative a second order moment is obtained. The sufficient statistics of an exponential family is sufficient for estimating the natural parameters. Being conditional distributions, the natural parameter vector depends on the variables conditioned on. The local variables z_{nj} depend on β and on the other variables in the nth context (x_n and local variables $z_{n,-j}$). Making this assumption about the conditionals allows to have a conjugacy relation between β and the (z_n, x_n) which makes the following conditional being in the exponential family (assuming the prior for β is also in the exponential family):

$$p(x_n, z_n|\beta) = h(x_n, z_n)\exp\left\{\beta^T t(x_n, z_n) - a_l(\beta)\right\}$$
$$p(\beta) = h(\beta)\exp\left\{\alpha^T t(\beta) - a_g(\alpha)\right\}$$

This ensures that the conditional for the global variable is in the exponential family with the natural parameter given by:

$$\eta_g(x, z, \alpha) = (\alpha_1 + \sum_{n=1}^{N} t(z_n, x_n), \alpha_2 + N)$$

This conjugate exponential model is highly used in statistical models for machine learning. In the end we want to compute the posterior of all the hidden variables, given the observations we have, i.e.:

$$p(z, \beta|x) = \frac{p(x, z, \beta)}{\int p(x, z, \beta) dz d\beta}$$

However, the denominator is intractable to compute. This is where mean-field inference can be used to compute an approximation of the true posterior by considering the model is fully factorizable and the factors are themselves in the exponential family. Through this assumptions, posterior inference becomes tractable. We will give next a short introduction to mean-field variational inference.

3.2.1 Mean-field approximate inference

The main idea in mean-field approximation is to consider the variables independent (so the posterior is fully factorizable) and then approximate one variable as a function of the expectations of all the other variables through a process which resembles the expectation-maximization algorithm. The mean-field optimization technique has been shown to optimize the Kullback-Leibler divergence $KL(q||p)$ where q is the variational distribution, part of a tractable family and p is the unknown true distribution of interest. Using Jensen's inequality for expectations, and making use of the fact that the logarithm is a concave function we have $\mathbb{E}[f(y)] \geq \mathbb{E}[\log(f(y))]$; in this case the optimization has been shown to maximize what is called the evidence lower bound or ELBO:

$$\begin{aligned}
\log p(x) &= \log \int p(x, z, \beta) dz d\beta \\
&= \log \int p(x, z, \beta) \frac{q(z, \beta)}{q(z, \beta)} dz d\beta \\
&= \log \left(\mathbb{E}_q \left[\frac{p(x, z, \beta)}{q(z, \beta)} \right] \right) \\
&\geq \mathbb{E}_q[\log p(x, z, \beta)] - \mathbb{E}_q[\log q(z, \beta)] \text{ Jensen's inequality} \\
&\triangleq \mathcal{L}(q)
\end{aligned}$$

where we have introduced a distribution over the local and global hidden variables z and β, $q(z, \beta)$. $\mathcal{L}(q)$ is the ELBO and is constituted of the expected log joint of all the variables including the data with respect to the approximating distribution q and the entropy of the approximating (or variational) distribution q. As we said earlier, the mean-field approach assumes the variables are independent:

$$q(z,\beta) = q(\beta|\lambda) \prod_{n=1}^{N} \prod_{j=1}^{J} q(z_{nj}|\phi_{nj})$$

where λ are the parameters of the distribution of the global hidden variables β and ϕ_n are the parameters of the distribution of the local hidden variables in the nth context, z_n. As we said, it is needed that $q(\beta|\lambda)$ and $q(z_{nj}|\phi_{nj})$ be in a family of distributions for which posterior inference is tractable (that is, the expectations in the ELBO can be efficiently computed), so the authors choose the exponential family with λ and ϕ being the natural parameters of the distributions. We will give next the update formulas for the parameters of the global and local hidden variables.

3.2.2 Coordinate ascent and natural gradient

As a function of λ the objective function becomes:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\beta|\lambda)}[\log p(\beta|x,z)] - \mathbb{E}_{q(\beta|\lambda)}[\log q(\beta)] + constant$$

The equation looks a bit different from the original objective, but this is because we are interested in terms that depend on $q(\beta|\lambda)$, while the other terms are absorbed in the constant. For details see the original paper [20]. When plugging in the distribution $q(\beta|\lambda)$ into the objective and making use of the fact that for the exponential family, the expectation of the sufficient statistics is the gradient of the log normalizer ($\mathbb{E}_q[t(\beta)] = \nabla_\lambda a_g(\lambda)$), we get the following form for the ELBO:

$$\mathcal{L}(\lambda) = \mathbb{E}_q[\eta_g(x,z,\alpha)]^T \nabla_\lambda a_g(\lambda) - \lambda^T \nabla_\lambda a_g(\lambda) + a_g(\lambda) + constant.$$

Usually in this case, coordinate updates are used, meaning taking the gradient of the ELBO with respect to one of the hidden parameters ($\lambda or \phi$), equating to 0 such that we can get a local optimum and then updating the parameter of interest from this. Formally:

$$\nabla_\lambda \mathcal{L} = \nabla_\lambda^2 a_g(\lambda)(\mathbb{E}_q[\eta_g(x,z,\alpha)] - \lambda) \tag{3.2}$$

Setting this to zero gives the following update for λ.

$$\lambda = \mathbb{E}_q[\eta_g(x,z,\alpha)]$$

Doing the same for the parameters of the local variables ϕ we get an update similar to the one for λ:

$$\phi_{nj} = \mathbb{E}_q[\eta_l(x_n, z_{n,-j}, \beta)]$$

This is, in short, how variational inference works using coordinate ascent. The authors notice that, when taking the natural gradient with respect to the parameters of the global and local hidden variables, the update equations resemble the coordinate ascent updates. We saw before that the natural gradient minimizes the KL divergence. However, we saw that the KL is not symmetric, $KL(p||q) \neq KL(q||p)$. One way around this is to consider the symmetrized KL, i.e.:

$$KL^{sym}(\lambda, \lambda') = \mathbb{E}_\lambda\left[\log \frac{q(\beta|\lambda)}{q(\beta|\lambda')}\right] + \mathbb{E}_{\lambda'}\left[\log \frac{q(\beta|\lambda')}{q(\beta|\lambda)}\right]$$

for the optimization problem that gives the steepest ascent:

$$\underset{d\lambda}{\operatorname{argmax}} f(\lambda + d\lambda) \text{ subject to } KL^{sym}(\lambda, \lambda + d\lambda) < \epsilon$$

As we saw before, the KL is invariant to reparametrization and that the Fisher metric is the Riemannian metric associated with a statistical manifold. We saw also that following the natural gradient is equivalent to minimizing the KL distance between $q(\beta|\lambda)$ and $q(\beta|\lambda + d\lambda)$ with small enough $d\lambda$. By approximating $q(\beta|\lambda + d\lambda)$ with its first-order Taylor expansion about λ, we get:

$$\begin{aligned}\log q(\beta|\lambda + d\lambda) &= O(d\lambda^2) + \log q(\beta|\lambda) + d\lambda^T \nabla_\lambda \log q(\beta|\lambda)\\ q(\beta|\lambda + d\lambda) &= O(d\lambda^2) + q(\beta|\lambda) + q(\beta|\lambda) d\lambda^T \nabla_\lambda \log q(\beta|\lambda)\end{aligned}$$

Plugging in the equation of the symmetrized KL, we get:

$$\begin{aligned}KL^{sym}(\lambda, \lambda + d\lambda) &= \int_\beta (q(\beta|\lambda + d\lambda) - q(\beta|\lambda))(\log q(\beta|\lambda + d\lambda) - \log q(\beta|\lambda))\\ &= O(d\lambda^3) + \int_\beta q(\beta|\lambda)(d\lambda^T \nabla_\lambda \log q(\beta|\lambda))^2 d\beta\\ &= O(d\lambda^3) + \mathbb{E}_q[(d\lambda^T \nabla_\lambda \log q(\beta|\lambda))^2]\\ &= O(d\lambda^3) + d\lambda^T G(\lambda) d\lambda\end{aligned}$$

When $q(\beta|\lambda)$ is in the exponential family the metric G is the second derivative of the log normalizer: $G(\lambda) = \nabla^2_\lambda a_g(\lambda)$. This is because for the exponential family the Hessian of the log normalizer with respect to the natural parameter λ is the covariance matrix of the sufficient statistic vector $t(\beta)$. Now, considering the natural gradient with respect to λ, we know from before that we should multiply the normal gradient with the inverse Fisher information matrix. This means, multiplying Equation (3.2) with G^{-1}, giving the following expression for the natural gradient:

$$\widehat{\nabla_\lambda}\mathcal{L} = (\nabla^2_\lambda a_g)^{-1} \nabla^2_\lambda a_g(\lambda)(\mathbb{E}_q[\eta_g(x, z, \alpha)] - \lambda)$$

We see a very interesting thing. The terms related to the log normalizer have canceled out, leaving:

$$\widehat{\nabla_\lambda}\mathcal{L} = \mathbb{E}_q[\eta_g(x, z, \alpha)] - \lambda$$

which is very similar to the coordinate update expression for λ. The same happens also for the local parameters ϕ. Thus, the coordinate ascent algorithm can be interpreted as a projected natural gradient algorithm [21]. Thus, *a coordinate update step* *is equivalent to taking a step of length one in the direction of the natural gradient*. In this paper using the natural gradient instead of the normal gradient makes the computation more efficient because the multiplication with the Fisher matrix disappears from the final parameter update, whereas for the normal gradient this multiplication is still necessary.

3.2.3 Experimental results

In the seminal research presented here, the authors experiment with SVI on a large corpus of documents. They make use of two widely used topic models, one parametric, latent Dirichlet allocation (LDA) [22] and one nonparametric, hierarchical Dirichlet process (HDP) [23]. They perform comparisons between these models, but also comparisons between standard variational inference and the stochastic version presented above. The data consists of three collection of documents and for each collection, the authors removed stop words, rare words and very frequent words. The collection are:

- *Nature*: This collection contains 350,000 documents from the journal Nature between years 1869 and 2008. After preprocessing the data, the vocabulary had 58 million words from a dictionary of 4,200 terms.
- *New York Times*: This collection has 1.8 million documents from the newspaper New York Times between years 1987 and 2007. Again, after preprocessing the collection has 461 million words from a dictionary of 8,000 terms.
- *Wikipedia*: This collection has 3.8 million articles from Wikipedia which after preprocessing turn into 482 million words from a dictionary of 7,700 terms.

For all collections, a set of 10,000 documents was not included in training as to evaluate model fitness on it, so a test set. To evaluate how good a model is, the authors employ the predictive distribution, first by estimating the corpus' topics and then given a test document (or part of it), they estimate the topic proportions in the document. After this, they combine the topic proportions with the topics to get a predictive distribution over the dictionary. And when holding out a set of words when performing inference, then the better predictive model is the one that assigns higher probability to the set of held out words. There are three parameters that affect convergence speed and performance: forgetting rate k, which controls how the fast old information is forgotten, τ de-emphasizes earlier iterations and the minibatch size is the number of documents that are analyzed in every iteration (subsampling). For more details of the values used see [20].

3.3 Deep neural networks

The following research described is based on [16]. As first described in [13] it is possible to make use of the natural gradient when optimizing neural networks. The output of a MLP with a linear activation can be seen as the mean of a Gaussian with

Table 3.1: Comparison of predictive performance with batch size of 500 documents and forgetting rate $k = 0.9$. LDA is sensitive to the number of topics (25, 50, 100, 200, 300). HDP gives better performance on all 3 collections of documents. Standard variational inference gave worse results than SVI. Results taken from [20].

Model	Nature	New York Times	Wikipedia
LDA 25	-7.24	-7.73	-7.44
LDA 50	-7.23	-7.68	-7.43
LDA 100	-7.26	-7.66	-7.41
LDA 200	-7.50	-7.78	-7.64
LDA 300	-7.86	-7.98	-7.74
HDP	**-6.97**	**-7.38**	**-7.07**

fixed variance, conditioned on the input. As we mentioned earlier, finding a solution that minimizes the squared error loss, under these assumptions, is equivalent to the maximum likelihood solution. In [24] it is shown that the Fisher matrix and the Gauss-Newton matrix are equivalent for the squared error loss. In fact, as we mentioned earlier in Section (3.1), because the natural gradient is minimizing the (local) distance function given by the KL-divergence, an argument can be made that it can be applied to any model that possesses such a distance function, being squared error or something else. In [16] they show that Hessian free optimization and Krylov Subspace Descent are implementations of the natural gradient by considering matching pairs of sigmoid and cross-entropy objective, softmax and negative log-likelihood respectively. However, we have to keep in mind that the equivalence is not both ways, in the sense that the natural gradient can also be applied to models where such a distance function does not exist, but the structure of a Riemannian manifold does, for example natural gradient works for Independent Component Analysis [25]. An interesting interpretation of the natural gradient is derived in [16] giving rise to the natural conjugate gradient, which becomes the normal conjugate gradient when Euclidean space and a symmetric Hessian in the second order Taylor expansion are considered.

3.3.1 Natural neural networks

Very recent work in deep neural networks [26] looked at the natural gradient from a different perspective. They started from wondering if there can be a certain parametrization of the neural network such that the Fisher information is identity. If this could be done, this would mean that the natural gradient descent and stochastic gradient descent would be equivalent, finding the same solution. They do this by constructing the so called whitened neural network (WNN), and then employing the so called projected natural gradient descent (PRONG). Let x be the observation vector, and y its associated label, $\pi(x, y)$ the empirical distribution under the log-loss and the model is given by $p(y|x;\theta)$. The problem is fitting the parameters $\theta \in \mathbb{R}^n$. We proceed by shortly describing the functioning of such a network. Considering a

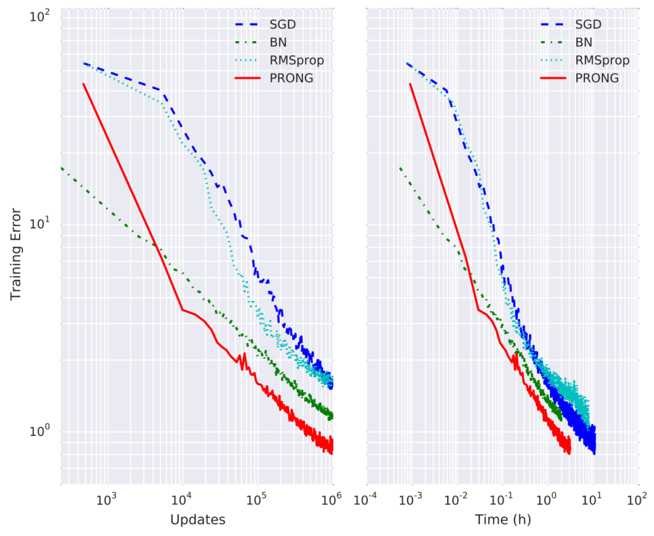


Figure 3.4: Training error versus number of updates (left). Training error versus cpu-time (right). Figure taken from [26].

deep network for binary classification with L layers:

$$p(y=1|x) = h_L = f_L(W_L h_{L-1} + b_L)$$
$$\vdots$$
$$h_1 = f_1(W_1 x + b_1)$$

where the parameters of the network are: $W_1, b_1, ..., W_L, b_L$ (denoted by θ) are the weights and biases and f_i is a non-linear function. Defining the error that is back-propagated through the ith nonlinearity as δ_i and ignoring the off block-diagonal components (the off block-diagonal components being ignored means that they are assumed to be 0, which means that the parameters involved in the matrix entry, e.g. i and j for g_{ij}, are assumed to be orthogonal, which would mean that the maximum likelihood estimated are assumed to be independent) of the Fisher matrix, the block F_{W_i} corresponding to interactions in layer i is given by:

$$F_{W_i} = \mathbb{E}_{x\sim\pi, y\sim p}\left[vec(\delta_i h_{i-1}^T) vec(\delta_i h_{i-1}^T)^T\right]$$

where $vec(X)$ returns a column vector from all the rows of the matrix X. When considering just the block-diagonal components this means we are approximating the initial matrix, an approximation which is better than just a simple diagonal approximation. This would incur some error, however, the overall process is iterative and such an approximation works well. When assuming δ_i and activations h_{i-1} are independent random variables (this assumption is not really valid, as the δ are usually correlated) , the above expression can be rewritten as:

$$F_{W_i}(km, ln) \approx \mathbb{E}_{x\sim\pi, y\sim p}\left[\delta_i(k)\delta_i(l)\right] \mathbb{E}_\pi\left[h_{i-1}(m) h_{i-1}(n)\right]$$

and is thus the Fisher block, which captures interactions between $W_i(k,m)$ and $W_j(l,n)$. The critical assumption made by the authors is that they can *enforce* $\mathbb{E}_\pi[h_i h_i^T] =$

I for all layers (for more details on the exact procedure see the original work [26]). In addition they consider the statistics fixed, i.e. $\mu_i(\theta) = \mathbb{E}[h_i]$ and $\Sigma_i(\theta) = \mathbb{E}[h_i h_i^T]$. The activation of the WNN is given by:

$$h_i = f_i(V_i U_{i-1}(h_{i-1} - c_i) + d_i)$$

where we can see the centering parameter $c_i = \mu_i$ gives zero mean in expectation of the dot product, similar to the centering reparametrization of the Deep Boltzmann Machines [27]. The matrix $U_{i-1} \in \mathbb{R}^{N-1 \times N-1}$ is a ZCA-whitening matrix for each layer, and the rows of which are given by the eigen-decomposition of Σ_{i-1}:

$$\Sigma_i = \widehat{U}_i \cdot diag(\lambda_i) \cdot \widehat{U}_i^T \Rightarrow U_i = diag(\lambda_i + \epsilon)^{-\frac{1}{2}} \cdot \widehat{U}_i^T$$

where ϵ controls the maximum multiplier on the learning rate and the V_i and d_i are analogous to the θ parameters of the normal neural network, i.e. W_i and b_i but they work in the whitened space $U_i(h_i - c_i)$. Many layers can be stacked onto one another forming a deep neural network with parameters $\omega = \{V_1, d_1, ..., V_L, d_L\}$ and whitening coefficients $\phi = \{U_0, c_0, ..., U_{L-1}, c_{L-1}\}$. The important thing to observe here is that the whitening coefficients are not learned through loss minimization (in which case the model could be considered over-parametrized), but they are estimated from the model statistics. The whitening coefficients are estimated from model statistics because the data has already been incorporated into the model, and thus the whitening coefficients reflect the structure of the data in the model. However, this would change the network function from one step to the next, thus the product $V_i U_{i-1}$ is preserved after the whitening coefficients update. So the optimizer sees these coefficients as constants, although they are used to improve conditioning of the Fisher matrix with respect to ω, and now it can be seen that the block diagonal terms of the Fisher depends on terms $\mathbb{E}[(U_i h_i)(U_i h_i)^T]$ which is identity by construction. The updating of the coefficients has to leave the actual objective function unchanged and thus the authors preserve the product $V_i U_{i-1}$ before and after each update of the whitening coefficients. To compute the whitening matrix U_i the cost is cubic in the layer size, so the authors find a way to amortize this cost over T consecutive updates by making use of the smoothness of gradient descent. Thus, the SGD in the whitened network will be close to the NGD immediately after reparametrization. The authors also find an improvement to this algorithm by using a diagonal scaling of U_i similar to the diagonal natural gradient. For an interesting discussion regarding duality of the whitened space and the normal space of parameters and the relation with mirror descent, see [26]. We show in Figure (3.4) a performance comparison of the PRONG algorithm for natural neural network with the standard stochastic gradient descent, the newly discovered batch normalization technique [28] and another adaptive algorithm called rmsprop [29], which in fact resembles PRONG as the normalization terms both contain the square root of the Fisher. The performance is tested on the well-known MNIST dataset. The authors investigated the approximation of PRONG further, by constructing a 3 layer MLP (such that the Fisher is tractable) and feeding a subsampled version of MNIST. In Figure (3.5) we show the FIM of the middle layer before and after whitening the activations. The critical difference between the tested algorithms is the difference in the condition number of the FIM relative to the initial value.

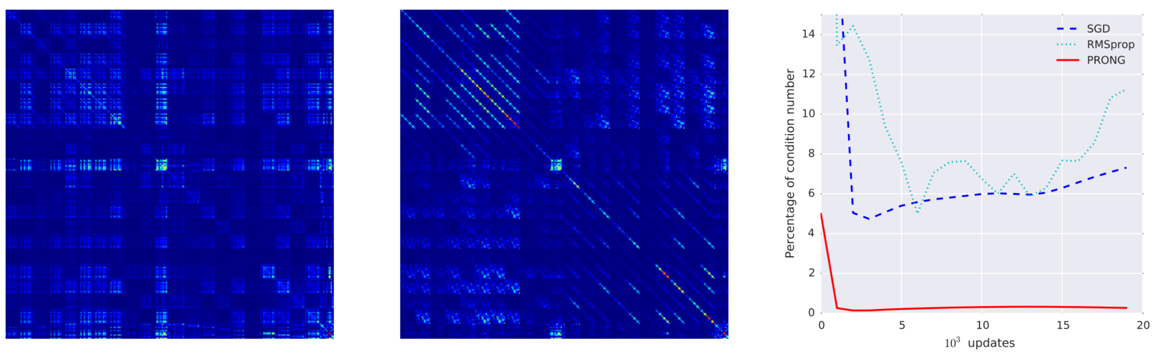


Figure 3.5: Fisher matrix for a 3-layer MLP before (left) and after (middle) first reparametrization. The condition number of the Fisher relative to the initial conditioning (right). Figure taken from [26].

3.4 The manifold tangent classifier

In this section we present a very interesting association between deep architectures, in particular, deep autoencoders and differential geometry, specifically, the tangent bundle. The original research can be found in [30]. We proceed by describing what a contractive auto-encoder is. The auto-encoder has been used in machine learning for learning feature representation of high-dimensional data. Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$ be the set of data points we want to classify. Then an auto-encoder finds a low dimensional representation by mapping first x to $h(x)$, called the hidden representation and then mapping $h(x) \in \mathbb{R}^h$ back to the input space $g(h(x))$, basically reconstructing the input from the hidden representation. The $h(x)$ function is called the encoder and the $g(h(x))$ function is called the decoder. The goal is to minimize the reconstruction error $L(x, g(h(x)))$ for the all data x_i. The functions h and g are the usual functions used in deep networks, i.e. $\sigma(Wx + b)$ where σ can be the element-wise logistic sigmoid for h and for g x is replaced with $h(x)$, W is the matrix of weights (these are shared between the encoder and decoder) and b is the vector of biases, that different for encoder and decoder. For the decoder an identity function can be used instead of the logistic sigmoid. The loss function can be squared error: $L(x, r) = ||x - r||^2$ or the Bernoulli cross-entropy: $L(x, r) = -\sum_{i=1}^{d} x_i \log(r_i) + (1 - x_i) \log(1 - r_i)$. The auto-encoder was initially used for dimensionality reduction, producing a lower dimensional representation of the input, where traditional classifiers could be used more efficiently to find a discrimination between classes. However, sometimes, depending on the data, letting the hidden representation have a higher dimension than that of the input is also very useful (this is called over-complete representation), but this is prone to overfitting and thus regularization techniques need to be used such that the auto-encoder extracts relevant features and avoids simple solutions. Often, sparse representation that use L_1 regularization work very well. An interesting recent addition to auto-encoders is the use of the Frobenius norm of the encoder's Jacobian as a regularizer. By keeping this small, this enables robustness of the representation to small differences in the input. This is called the contractive auto-encoder, and its objective is given by:

$$\mathcal{J}_{CAE} = \sum_{x \in \mathcal{D}} L(x, g(h(x))) + \lambda \|J(x)\|^2$$

where λ is a hyper-parameter that controls how strongly this norm affects the overall loss and $J(x) = \frac{\partial h}{\partial x}(x)$. Reasoning in this direction, one can also penalize higher order derivatives. This can be done by using a simple stochastic technique, minimizing the difference between the Jacobian at x and the Jacobian at $\widehat{x} = x + \epsilon$. This is referred to as CAE+H, it can be found in [31] and its loss is given by:

$$\mathcal{J}_{CAE+H} = \sum_{x \in \mathcal{D}} L(x, g(h(x))) + \lambda \|J(x)\|^2 + \gamma \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} \left[\|J(x) - J(x+\epsilon)\|^2 \right]$$

where γ is an additional hyper-parameter controlling how much higher-order derivatives (or specifically, stochastic approximations) affect the loss function.

3.4.1 The tangent bundle

The regularization used in CAE and CAE+H makes $h(x)$ insensitive to most input directions. However, it remains sensitive to the directions which help reconstruct the data, which are the directions that give the discriminability (i.e. being able to distinguish) between similar training points. This has a natural geometric interpretation, these directions span the (local) tangent space of the data manifold. In short, the disjoint reunion of tangent spaces is called the tangent bundle as we mentioned in Chapter (2). Based on this observation, a local coordinate system can be derived, such that is locally Euclidean and that it captures the dependency between the data points. Moreover, it is robust to small changes in the input. To be able to map h to data points in such a way that we can get an atlas for the manifold, h must be locally invertible around each data point . We need to be sure of the fact that h is bijective between the data points $\mathcal{D}$ and $h(\mathcal{D})$. This gives the following conditions for injectivity: $\exists \alpha \in \mathbb{R}^{d_h}$ such that $\Delta_{ij} = \sum_k^{d_h} \alpha_k W_k$, for any i, j where $\Delta_{ij} = x_i - x_j$ and W_k are the rows of W. For surjectivity we just need to limit the domain of h: $h(D) \subset (0,1)^{d_h}$. For more details about this reasoning, see the original research in [30].

3.4.2 Obtaining an atlas

As we said in the Chapter (2) chapter, an atlas is a collection of charts that cover the whole manifold. In our case we will define a local chart around each data point. As we said, the main idea is that $h(x)$ is insensitive to most changes in the input directions, however not when moving from an example x_i to x_j. The authors say this should be reflected in the spectrum of the Jacobian at each training point. A local chart is then defined by decomposing the matrix J(x) using singular value decomposition: $J^T(x) = U(x)S(x)V^T(x)$ where $U(x)$ and $V(x)$ are orthogonal and

$S(x)$ is diagonal. Then the tangent plane $\mathcal{H}_x$ at x is given by the span of the principal singular vectors $\mathcal{B}_x$:

$$\mathcal{B}_x = \{U_{\cdot k}(x) | S_{kk}(x) > \epsilon\} \text{ and } \mathcal{H}_x = \{x + v | v \in span(\mathcal{B}_x)\}$$

where $U_{\cdot k}$ is the kth column of $U(x)$ and $span(\{z_k\}) = \{x | x = \sum_k w_k z_k, w_k \in \mathbb{R}\}$. The atlas $\mathcal{A}$ then can be defined, based on a local linear approximation around each data point:

$$\mathcal{A} = \{(\mathcal{M}_x, \phi_x) | x \in \mathcal{D}, \phi_x(\widehat{x}) = \mathcal{B}_x(\widehat{x} - x)\}$$

This can be applied to all layers of a deep network, in a greedy layer-wise way to initialize a network with CAE as in [32].

3.4.3 Classification

To perform classification in this newly defined space, one can use any nearest neighbor classifier with the distance defined as the shortest distance between hyperplanes [33]. The tangents at each point enable us to shrink the distance between two points when they can be approximated by a linear combination of their local tangents. The tangent distance between two points x, y is defined to be the distance between the hyperplanes $\mathcal{H}_x, \mathcal{H}_y \subset \mathcal{R}^d$ spanned by $\mathcal{B}_x$ and $\mathcal{B}_y$ respectively. The definition of distance between two spaces is given by:

$$d(\mathcal{H}_x, \mathcal{H}_y) = inf \left\{ ||z - w||^2 | (z, w) \in \mathcal{H}_x \times \mathcal{H}_y \right\}$$

This is a convex problem and the solution can be obtained by solving a system of linear equations as in [33]. Informally, this is allowing the points x, y to move in the directions of their local charts and then the distance is defined as the minimum distance between these new coordinates.

Nearest-neighbor classifiers are impractical for large datasets because for each test case the computation scales linearly with n. A neural network classifier is much more efficient after training. The authors employ the tangent propagation as used in [34] to constrain the output o of a neural network classifier to be less sensitive to variations in the directions spanned by the local chart of x by adding to the cost function the following term:

$$\Omega(x) = \sum_{u \in \mathcal{B}_x} || \frac{\partial o}{\partial x}(x) u ||^2$$

This computation can be done linearly in the number of network weights.

3.4.4 The Manifold Tangent Classifier

The overall picture of training a deep network can be described as:

Table 3.2: Classification error on MNIST.

K-NN	NN	SVM	DBN	CAE	DBM	CNN	MTC
3.09%	1.60%	1.40%	1.17%	1.04%	0.95%	0.95%	**0.81%**

- Train a stack of K CAE layers where each one is getting the input from the previous one.
- For all $x_j \in \mathcal{D}$ compute the Jacobian and its SVD. Form B_{x_i} as the set d_M of principal singular vectors.
- Add a classification layer with the appropriate number of outputs (number of classes) and cost function which gets the additional penalty term from the tangent propagation described above.

We show the results obtained in [30] on the MNIST dataset, when compared to other techniques in Table (3.2). We see that the results of the MTC improve compared with the other learning algorithms through the simple propagation of the tangent directions into the cost function. We want to emphasize here that the Jacobian of interest is, in fact, describing the data geometry induced into the model geometry. If we would consider just the model geometry then we would be interested in how the function h changes, independent of the input fed, whereas if we would consider just the data geometry this would be equivalent to reasoning about the structure of the data independent of the model used, the intrinsic particularities of the data. Thus, considering how the model changes with respect to the data is extracting the geometry of interest, i.e. the geometry that the data is inducing into the model. In our opinion this is a very important distinction and we advocate that if we would know the geometry of the data, independent of model and also the geometry of the model independent of the data, we could, in principle, figure out the exact characteristics of the geometry induced by the data into the model, which could then be used for highly efficient learning.

3.5 Reinforcement learning

As we mentioned in the beginning of this chapter, we will show next how the natural gradient can be applied to reinforcement learning algorithms. This is to have a general picture of how the model geometry can drive learning in a better and more efficient way than assuming an Euclidean geometry in different learning contexts. The following research is presented following [35]. Reinforcement learning has developed as a field of machine learning, borrowing concepts from supervised and unsupervised learning, but keeping a particular perspective of the dynamics of learning. It is biologically inspired, resembling the way humans learn, by interacting with the environment and discovering the dynamics of reward getting. Usually, an agent is endowed with such reinforcement learning capabilities to explore the environment (thus, he can act in the world), observe the environment (he has a type of memory from which he can extrapolate, learn) and his goal is to maximize some function (the reward function, he wants to get as many rewards as possible) for which he forms a type of mapping between his actions and the rewards he is getting. More formally, a reinforcement learning agent can solve problems that are modeled as Markov Decision Processes (MDP). If we let X be the state space, U the action space, f is the transition function (this is a probability function) which takes the agent from one state to the next, and ρ is the reward function, then an MDP is a tuple (X, U, f, ρ) with:

$$f : X \times U \times X \rightarrow [0, \infty)$$
$$\rho : X \times U \times X \rightarrow \mathbb{R}$$

Let x_k be the current state of the agent, u_k be the current action the agent is choosing in this state, and x_{k+1} be the next state that the agent is in after he took the action. Then, after a transition to a new state, the agent receives a reward given by:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1})$$

The agent chooses actions according to a policy $\pi : X \times U \rightarrow [0, \infty)$, and its overall goal is to find a policy such that it maximizes the expected value of a function g which is a function of the reward. g can be a discounted (there is a practice in RL, where distant future rewards are worth less than closer ones, this process is referred to as discounted rewards) sum of rewards or the average reward.
However many problems exist for such an agent. How much should he explore and how much should he exploit what he already knows ? Maybe some bigger rewards can be obtained if more exploration is done (the exploration-exploitation dilemma). If after a sequence of actions, some reward is obtained, which action should the agent believe got him the reward ? The last one or the last few ones ? What if some actions are detrimental but can lead to a high expected reward in the long run. Such problems have spanned a multitude of ideas and approaches in reinforcement learning that have been applied to many real-world problems with success. One of the biggest achievement in the machine learning community was an agent that was able to learn to play Atari games (about 50 different games) to a performance level comparable to humans, from just pixel data. The agent was a reinforcement learning based agent, however the underlying learning representation was a deep convolutional network. The approach is called DQN [36] and combines the successful deep learning framework with reinforcement learning and some additional engineering tricks (for example episode replay, where an agent can choose to "remember" some past episodes and relearn from them). In the reinforcement learning (RL) setting, an agent interacts with the environment to get rewards, or optimize its behavior. The learning is done as to maximize the expected reward by choosing proper actions. When in a certain state, if the agent has learned an optimal action for that state, it is called an optimal policy. However until finding the optimal policy, the agent needs to explore the environment to uncover the possible rewards in each state. The underlying representation of the estimation of the reward is called the value-function (usually denoted by V), and in short, it associates an actual numerical value to how good or bad it is to be in a certain state. If we also take into account actions in

each state, then the function that maps actions to states is called action-value function (usually denoted by Q). Many reinforcement learning algorithms exist, we will focus on two main paradigms: policy gradient methods and actor-critic methods.

3.5.1 Natural policy gradient

Policy gradient represents a class of RL algorithms based on a parametrized family of policies (they are also called actor-only) that can be used with optimization methods to yield a gradient direction for improvement. If we let $\theta \in \mathbb{R}^n$ be the parameters of the policy π_θ and let J be the expected return, the function we want to maximize (or minimize, depends on the function is defined) is defined in Equation (3.3). Further assuming that π_θ and J are differentiable with respect to θ, the gradient of J with respect to θ is given by:

$$\nabla_\theta J(\theta) = \frac{\partial J}{\partial \pi_\theta} \frac{\partial \pi_\theta}{\partial \theta}$$

Actor-only algorithms are capable of dealing with continuous actions, however, usually the gradient is estimated with high variance, which then makes the agent learn slowly. The natural gradient has been introduced in such a setting by [37]. Let us consider a parametrized stochastic policy $\pi_\theta(s,a)$ in state s. π_θ denotes a probability distribution over actions in state s, being a conditional probability function we could denote it also as $p(a|s)$, however we follow the notation of the original work in [35]. This policy is a point on the Riemannian manifold, with the associated metric tensor given by the Fisher information matrix, that, for the above policy is given by:

$$\begin{aligned} F(\theta,s) &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T\right] \\ &= \int_A \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T da \end{aligned}$$

where A is the set of all actions. This policy for a single state is related to the immediate reward over a single step from s (through the reward function, taking an action from a state immediately gives a reward to the agent), but it does not give information about the expected return $J(\pi)$, which is obviously defined over complete state trajectories, usually defined as (in the average reward case):

$$J(\pi) = \lim_{n\to\infty} \frac{1}{n} \mathbb{E}\left[\sum_{k=0}^{n-1} r_{k+1} | \pi\right] = \int_S d^\pi(s) \int_A \pi(s,a) \int_S f(s,a,s')\rho(s,a,s') ds' da ds \tag{3.3}$$

where f is the state transition probability density function and ρ is the reward function. In the average reward case, to get an suitable Fisher information matrix, in the average reward case, [37] took the expectation of $F(\theta,s)$ with respect to the stationary state distribution $d^\pi(s)$, i.e.:

$$F(\theta) = \int_S d^\pi(s) F(\theta,s) ds$$

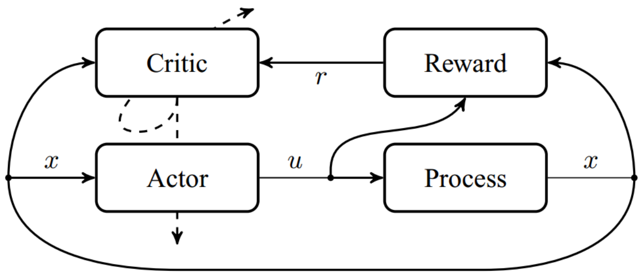


Figure 3.6: Schematic of an actor-critic algorithm. The dashed lines represent the learning dependency, the critic updates itself but also the actor. Figure from [35]

where the stationary state distribution $d^\pi(s)$ is given by $d^\pi = \lim_{k\to\infty} p(s_k = s, \pi)$ and can be described as the long term behavior of the Markov chain induced by the policy π, in short the fraction of times that the Markov chain spends in each state. This was later showed by [38; 39] that it is a valid Fisher for the manifold of probability distributions over trajectories in the Markov Decision Process. Even though the original work was just for the average return case, it was later shown by [38; 39] that it applies to the discounted reward case as well, further showing performance comparisons between normal policy gradient and natural policy gradient.

3.5.2 Natural actor-critic

Critic-only algorithms use a state-action value function but no special function for the policy. A simple way to come up with a policy is to select in each state greedy actions, i.e. actions that have the highest state-action value, which would indicate that the expected reward value is the highest. But this would mean that an procedure that finds this maximum is needed in every state and moreover, in an environment with continuous actions, the computation becomes really expensive. Critic-only methods usually discretise the action space and in this way they lose given data, and thus accuracy of the final solution. Actor-critic methods combine the two methods by making use of continuous actions and still make use of the critic's low-variance knowledge of the agent's performance. Estimating the expected reward enables the actor to make use of low-variance gradients and thus the agent's learning speed is improved. We show an overview of an actor-critic algorithm in Figure (3.6).

Natural actor-critic methods, so actor-critic methods that make use of the natural gradient, were first introduced in [38] to successfully update the policy. The authors gave a proof that the natural gradient $\nabla_\theta J(\theta)$ is the *compatible* feature parameter w of the approximated value function:

$$\widehat{\nabla}_\theta J(\theta) = w$$

where a parametrized (by w) value function approximator is said to be *compatible*to the policy if it satisfies:

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

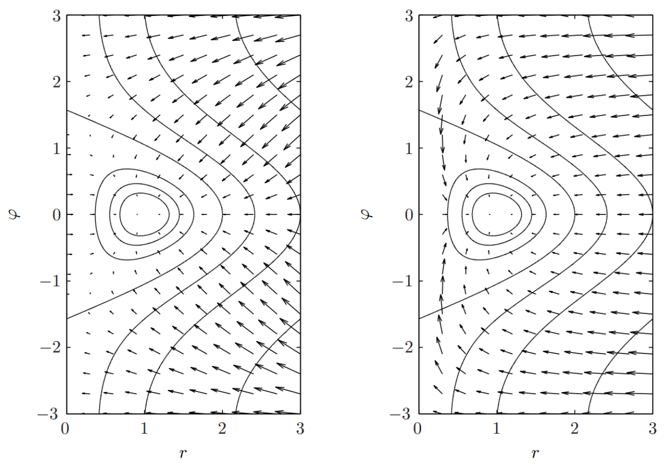


Figure 3.7: Standard (left) and natural (right) gradient of the cost function in polar coordinates. Figure from [35]

Then they used the natural gradient without the need for the explicit calculation of the Fisher. Thus, the policy update step is given by: $\theta_{k+1} = \theta_k + \alpha_a w_{k+1}$. For the policy evaluation, the authors used Least-squared temporal difference - $Q(\lambda)$. They then showed the algorithm performed better than the standard policy gradient on a cart-pole balancing problem. The work was then extended to use a recursive least-squares for the critic calculation by [40]. Both methods were successfully used for controlling robot arms. The work was further extended to use a neural network for the actor which automatically adds hidden layers if necessary [41].

Then the natural actor critic was extended more [42] by employing TD-learning (temporal difference) in the actor and providing convergence proofs for the algorithms. The convergence is important to such algorithms due to the use of function approximation and bootstrapping for the critic (which are both needed for scalable RL). We show an example of the difference between standard and natural gradient in polar coordinates in Figure (3.7). We see how the two vector fields that define the directions of the natural and conventional gradient agree very little overall, even though the directions are not completely different, they differ significantly far from the optimum. Indeed, the two gradients mainly agree just around the optimum, thus a learning trajectory following the natural gradient would be significantly different than a trajectory following the conventional gradient, and we would expect this difference to grow with increasing number of dimensions. For more details see [35].

3.6 Natural evolution strategies

The following research description is based on [43]. The authors describe a family of evolution strategies, which update the so called search distribution in an iterative manner by making use of the gradient with respect to the distribution parameters. Evolution strategies are particularly well suited for high dimensional, continuous domains and it is a prolific field of research in global optimization. In short, they are population-based algorithm that are ranked according to the degree of performance and the best ones are kept and mutated in the next generation of the population. The main idea of which Natural Evolution Strategies make use of is the *search gradient* [44]. A search gradient is a sampled gradient of the expected fitness and is used to update the parameters of the search distribution. The search distribution needs to have the property that the derivatives of the log density function with respect to the parameters can be computed. Some common distributions used are multivariate Gaussians, Gaussian mixture models and Cauchy distribution. The discussion that follows is based on the multivariate Gaussian. Let $\mu \in \mathbb{R}^d$ be the mean of the Gaussian representing the candidate solution center and $\Sigma \in \mathbb{R}^{d\times d}$ representing the mutation matrix. To sample from this distribution we need to consider the square root of the covariance matrix (this is not unique, but this does not matter for this purpose of sampling; many algorithms exist to obtain a square root of a matrix, like the Jordan decomposition or the Cholesky factorization) , $A \in \mathbb{R}^{d\times d}$, with $A^T A = \Sigma$, then $z = \mu + A^T s$ is the operation that transforms a normal vector $s \sim \mathcal{N}(0, I)$ into a sample $z \sim \mathcal{N}(\mu, \Sigma)$, where I is the identity matrix. The search distribution has the density function given by:

$$\begin{aligned}\pi(z|\theta) &= \frac{1}{(\sqrt{2\pi})^d |\det(A)|} \cdot \exp(-\frac{1}{2}\left\|A^{-1}\cdot(z-\mu)\right\|^2) \\ &= \frac{1}{\sqrt{(2\pi)^d |\det(\Sigma)|}} \cdot \exp(-\frac{1}{2}(z-\mu)^T \Sigma^{-1}(z-\mu))\end{aligned}$$

Here π represents the mathematical constant. Then to update the parameters of the search distribution we need the following gradients of the log-density with respect to the parameters:

$$\begin{aligned}\nabla_\mu \log \pi(z|\theta) &= \Sigma^{-1}(z-\mu) \\ \nabla_\Sigma \log \pi(z|\theta) &= \frac{1}{2}\Sigma^{-1}(z-\mu)(z-\mu)^T\Sigma^{-1} - \frac{1}{2}\Sigma^{-1}\end{aligned}$$

The authors then describe how it is impossible to locate a quadratic optimum even in a one-dimensional case, because the updates become numerically unstable as the variance decreases. They point out that this is a special case where the gradient controls both position and variance of a distribution and they show next how the natural gradient tackles the insufficiencies of the normal gradient by making updates invariant with respect to the parametrization. The canonical natural evolution strategy presented by the authors makes use of the natural gradient. Then to improve performance and robustness fitness shaping is employed. Fitness shaping is a technique that makes the algorithm invariant to any order-preserving transformation [43]. Then two reparametrizations are shown, which make the algorithms efficient. We proceed by describing fitness shaping next. Let z_1 be the best individual from the population and subsequent z_i be sorted by fitness until the last, z_λ. Replacing fitness with utility such that the utility values $u_1 \geq ... \geq u_\lambda$ gives the following expression of the gradient:

$$\nabla_\theta J(\theta) = u_k \nabla_\theta \log \pi(z_k|\theta)$$

where the utility is given by:

$$u_k = \frac{\max(0, \log(\frac{\lambda}{2}+1) - \log(k))}{\sum_{j=1}^{\lambda} \max(0, \log(\frac{\lambda}{2}+1) - \log(j))} - \frac{1}{\lambda}$$

which is related to the one used in the seminal algorithm CMA-ES [45], however the authors state that the choice of the utility function is not crucial, as long as it is rank based and monotonous. In the original research the authors present an algorithm for adapting the hyperparameters such as the learning rate called adaptation sampling, but we will skip its presentation here. We proceed by presenting the computation of the natural gradient for multivariate normal distributions derived from linear transformations of rotationally symmetric distributions. The use of such distributions enables the Fisher matrix to be obtained in closed form. Let $z \in \mathbb{R}^d$ be a sample and $r = \|z\|$ be its radial component, with $Q_\tau(z)$ be a family of distributions which are rotationally symmetric, parametrized by the vector τ. From this invariance the density can be written as: $Q_\tau(z) = q_\tau(r^2)$ for a functions $q_\tau : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. The search distribution has the density given by:

$$\begin{aligned}\pi(z|\mu, A, \tau) &= \frac{1}{|\det(A)|} \cdot q_\tau(\left\|(A^{-1})^T(z-\mu)\right\|^2) \\ &= \frac{1}{\sqrt{\det(A^T A)}} \cdot q_\tau((z-\mu)^T(A^T A)^{-1}(z-\mu))\end{aligned}$$

with transformation parameters $\mu \in \mathbb{R}^d$ and invertible matrix $A \in \mathbb{R}^{d\times d}$, but A can be restricted to any continuous subgroup of $GL(n)$. The function q_τ is the density of the random variable given by $s = (A^{-1})^T(z-\mu)$ and τ controls the tail of the distributions, i.e. if large mutations are common or rare. Next, a very important concept is introduced, i.e. local “natural” coordinates, which are coordinates that come up from the rotation invariant normal form (the normal form is a simplified representation which preserves the structure of interest) . These coordinates should be the canonical coordinates in which the normal form is the search distribution. The main idea is to replace the natural gradient steps on the manifold of invertible matrices with a one-to-one vector space representation, given by the matrix exponential, restricted to the vector space of symmetric matrices. The local exponential coordinates become: $(\delta, M) \to (\mu_{new}, A_{new}) = (\mu + A^T\delta, A\exp(\frac{1}{2}M))$. They are said to be local because the current search distribution is encoded as $(\delta, M) = (0,0)$. In these coordinates the Fisher takes the following form:

$$F = \begin{pmatrix} I & v \\ v^T & c \end{pmatrix}$$

$$\text{with } v = \frac{\partial^2 \log \pi(z)}{\partial(\delta, M)\partial\tau} \in \mathbb{R}^{(m-d')\times d'} \text{ and } c = \frac{\partial^2 \log \pi(z)}{\partial \tau^2} \in \mathbb{R}^{d'\times d'}$$

where d' is the dimensionality of τ. For Gaussians, which do not have a radial parameter (a radial parameter is defined for a radial distribution function also known as the pair correlation function, it describes how the density varies as a function of distance from a reference) , the Fisher is given by I, and thus the normal gradient is the same as the natural gradient. For other types of distribution, where τ play a role, using the Woodbury formula gives the following expression for the Fisher:

$$F = \begin{pmatrix} I & v \\ v^T & c \end{pmatrix}^{-1} = \begin{pmatrix} I + Hvv^T & -Hv \\ -Hv^T & H \end{pmatrix}$$

with $H = (c - v^T v)^{-1}$ and making use of the fact that $H^T = H$. When computing the gradient, this is given by:

$$\nabla_{\delta,M,\tau}|_{\delta=0,M=0} \log \pi(z|\mu, A, \tau, \delta, M) = g = (g_\delta, g_M, g_\tau)$$

$$\begin{aligned} \text{with } g_\delta &= -2 \cdot \frac{q'_\tau(||s||^2)}{q_\tau(||s||^2)} \cdot s \\ g_M &= -\frac{1}{2} I - \frac{q'_\tau(||s||^2)}{q_\tau(||s||^2)} \cdot ss^T \\ g_\tau &= \frac{1}{q_\tau(||s||^2)} \cdot \nabla_\tau q_\tau(||s||^2) \end{aligned}$$

where $q'_\tau = \frac{\partial}{\partial(r^2)} q_\tau$ is the derivative of q_τ with respect to r^2 and $\nabla_\tau q_\tau$ is the gradient with respect to τ. Thus, the natural gradient is given by:

$$F^{-1} \cdot g = \begin{pmatrix} (g_\delta, g_M) - Hv(v^T(g_\delta, g_M) - g_\tau) \\ H(v^T(g_\delta, g_M) - g_\tau) \end{pmatrix}$$

This version of the inverse Fisher matrix can be computer in $\mathcal{O}(d^2)$ compared to the naive version of the Fisher which needs $\mathcal{O}(d^6)$. One last ingredient is needed for using this class of radial distributions, that is, sampling from it is needed. Recall that first a sample is drawn from the standard density and then it is transformed according to: $z = A^T s + \mu$. The sampling can be decomposed into sampling the radial component, $r^2 = ||z||^2$ and then a unit vector $v \in \mathcal{R}^d, ||v|| = 1$. The squared radius has the density given by:

$$\widehat{q_\tau}(r^2) = \int_{||z||^2 = r^2} Q_\tau(z) dz = \frac{2\pi^{d/2}}{\Gamma(d/2)} \cdot (r^2)^{(d-1)/2} \cdot q_\tau(r^2)$$

where $\Gamma(\cdot)$ is the gamma function. When considering multivariate Gaussians the natural gradient is given by:

$$\begin{aligned} \nabla_\delta J &= \sum_{k=1}^{\lambda} f(z_k) \cdot s_k \\ \nabla_M J &= \sum_{k=1}^{\lambda} f(z_k) \cdot (s_k s_k^T - I) \end{aligned}$$

where s_k is the k^{th} best sample in the current batch in the above mentioned local coordinate system and z_k is the same sample but transformed according to the rule described above, in other words, in task coordinates, and f is the objective function of interest. The authors call this the exponential NES (xNES)and we show the pseudocode in Algorithm (1). The covariance factor A is decomposed into a scalar $\sigma > 0$ (scale) and a normalized covariance factor satisfying $\det(B) = 1$ (shape). Decoupling the factor A into these two orthogonal components allows independent learning rates for each of them. The complexity of xNES is $\mathcal{O}(d^3)$ but by updating in local non-exponential coordinates this is then just $\mathcal{O}(d^2)$.

Algorithm 1 xNES-multinormal case

```
**Require:** $f, \mu_{init}, \Sigma_{init} = A^T A$
**Ensure:** $\sigma \leftarrow \sqrt[d]{|\det(A)|}; B \leftarrow A/\sigma$
  **repeat**
    **for** $k = 1$ to $\lambda$ **do**
      draw sample $s_k \sim \mathcal{N}(0, I)$
      $z_k \leftarrow \mu + \sigma B^T s_k$
      evaluate the fitness $f(z_k)$
    **end for**
    sort $\{(s_k, z_k)\}$ with respect to $f(z_k)$ and compute utilities $u_k$
    compute gradients $\nabla_\delta J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot s_k$   and   $\nabla_M J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot (s_k s_k^T - I)$
                      $\nabla_\sigma J \leftarrow tr(\nabla_M J)/d$   and   $\nabla_B J \leftarrow \nabla_M J - \nabla_\sigma J \cdot I$
                      $\mu \leftarrow \mu + \eta_\delta \cdot \sigma B \cdot \nabla_\delta J$
    update parameters $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma/2 \cdot \nabla_\sigma J)$
                      $B \leftarrow B \cdot \exp(\eta_B/2 \cdot \nabla_B J)$
  **until** stopping criterion is met
```

Chapter 4

The volume element

As we saw in Chapter (2), the volume element plays an important role in Bayesian statistics, it is known as Jeffrey's prior and it defines an uninformative prior for statistical models. However, there is more to the volume element. We will describe next two approaches present in machine learning that make use of the volume element. The first one is based on [46] and used the inverse volume element when learning the metric over a statistical manifold, while the second one is used by [47] and uses the volume element to define the information density and information capacity for the space of models, in other words, to define model complexity and embedding quality.

4.1 Metric Learning

The metric learning problem is concerned with finding an embedding of the data into a (usually) lower dimensional manifold with an associated metric that best described the distribution of the data. Let the data be given by: $D = \{x_1, ..., x_N\} \subset \mathcal{M}$ with $\mathcal{M}$ a differentiable manifold, we are interested in the metric associated with $\mathcal{M}$. Let this metric g be part of a parametric family of candidate metrics:

$$\mathcal{G} = \left\{ g^\theta : \theta \in \Theta \subset \mathcal{R}^k \right\}$$

In [46] they advocate for the parametric model as they say it generally performs better on high dimensional, sparse data as text documents, which is the application of interest. The metric is chosen such that the objective function $\mathcal{O}(g, D)$ is maximized. This is given by:

$$\mathcal{O}(g, D) = \prod_{i=1}^{N} \frac{(\mathrm{dvol}g(x_i))^{-1}}{\int_{\mathcal{M}} (\mathrm{dvol}g(x_i))^{-1} dx}$$

where $\mathrm{dvol}g(x) = \sqrt{\det G(x)}$ is the differential volume element of the manifold and $G(x)$ is the Gram matrix of the metric g at point x. The determinant is positive since the metric is positive definite. The volume can be said to summarize the size of the metric at x in one scalar. The paths going through areas of the manifold with high

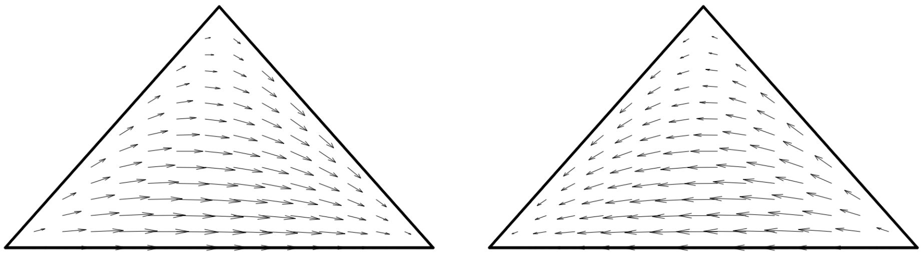


Figure 4.1: F_λ (left) and F_λ^{-1} (right) on $\mathbb{P}_2$ with $\lambda = \frac{2}{10}, \frac{5}{10}, \frac{3}{10}. Figurefrom[46]$.

volume will be longer than paths that go through low volume areas. This makes sense, as maximizing the inverse volume will result in shorter curves. Because of the normalization constraint for the volume, we may see the normalized volume element as a probability distribution over the manifold, thus maximizing the objective can be seen as a maximum likelihood problem. The log of the objective is equivalent to the log likelihood under the model:

$$p(x;\lambda) = \frac{1}{Z}(\sqrt{detG^\lambda(x)})^{-1}$$

If G is the Fisher matrix, the model is the inverse of the Jeffrey's prior $p(x) \propto \sqrt{detG(x)}$. As the authors states, it is hard to grasp the implications of a particular choice for the family of metrics, thus he chooses to specify a parametric family of transformations $\{F_\lambda \mathcal{I} : \lambda \in \Lambda\}$ which will results in the pull-back metrics $\mathcal{G} = \left\{F_\lambda^* \mathcal{I} : \lambda \in \Lambda\right\}$ of the Fisher metric $\mathcal{I}$. By choosing an isometry $F : (\mathcal{M}, g) \rightarrow (\mathcal{N}, \delta)$ where δ is the metric on the Euclidean space, the distances on the original manifold $\mathcal{M}$ which are locally distorted, can be measured on $\mathcal{N}$ using the flat Euclidean metric. The family of transformations $F_\lambda : \mathbb{P}_n \rightarrow \mathbb{P}_n$ is taken to be:

$$F_\lambda(x) = \left\{\frac{x_1\lambda_1}{\langle x, \lambda\rangle}, \ldots, \frac{x_{n+1}\lambda_{n+1}}{\langle x, \lambda\rangle}, \lambda \in \mathbb{P}_n\right\}$$

where $\mathbb{P}_n$ is n-simplex $\mathbb{P}_n = \left\{x \in \mathbb{R}_+^{n+1} : \sum_i x_i = 1\right\}$ and $\langle x, \lambda\rangle$ is the scalar product given by $\sum_{i=1}^{n+1} x_i\lambda_i$. This family of transformations F_λ is a Lie group under composition whose parametric space is $\lambda = \mathbb{P}_n$. The identity element is given by $\left(\frac{1}{n+1}, \ldots, \frac{1}{n+1}\right)$ and the inverse is $F_\lambda^{-1} = F_\eta$ with $\eta_i = \frac{1/\lambda_i}{\sum_k 1/\lambda_k}$. The group acts on the elements $x \in \mathbb{P}_n$ by increasing the ones with high λ. We show the action in Figure (4.1). The same author shows in a previous section how the Fisher is the pull-back metric from the sphere under the square root transformation (on the n-simplex). Based on this fact then $F_\lambda^*\mathcal{I}$ is the pull-back metric of $(\mathbb{S}_+^n, \delta)$ through the following transformation:

$$\widehat{F}_\lambda(x) = \left\{\sqrt{\frac{x_1\lambda_1}{\langle x, \lambda\rangle}}, \ldots, \sqrt{\frac{x_{n+1}\lambda_{n+1}}{\langle x, \lambda\rangle}}, \lambda \in \mathbb{P}_n\right\}$$

This gives a close form for the geodesic:

$$d_{F_\lambda^*\mathcal{I}}(x, y) = acos\left(\sum_{i=1}^{n+1}\sqrt{\frac{x_i\lambda_i}{\langle x, \lambda\rangle}\frac{y_i\lambda_i}{\langle y, \lambda\rangle}}\right)$$

The difference between this formula and the tf-idf cosine similarity [48] is the square root and the choice of λ. Now the volume element needs to be computed where $G = F_\lambda^*$. To compute $[G]_{ij} = F_\lambda^* \mathcal{I}(\partial_i, \partial_j)$ where $\{\partial_i\}_{i=1}^n$ is a basis for the tangent space at x, $T_x\mathbb{P}_n$ given by the rows of the following matrix:

$$U = \begin{bmatrix} 1 & 0 & \dots 0 & -1 \\ 0 & 1 & \dots 0 & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots 1 & -1 \end{bmatrix} \in \mathbb{R}^{n \times n+1}$$

To compute the determinant for the volume, we need the following propositions [46]:

Proposition 2
The matrix $[G]_{ij} = F_\lambda^* \mathcal{I}(\partial_i, \partial_j)$ is given by:

$$G = JJ^T = U(D - \lambda\alpha^T)(D - \lambda\alpha^T)^T U^T$$

where $D \in \mathbb{R}^{n+1 \times n+1}$ is a diagonal matrix whose entries are $[D]_{ii} = \sqrt{\frac{\lambda_i}{x_i}} \frac{1}{2\sqrt{\langle x, \lambda\rangle}}$ and α is a column vector given by $[\alpha]_i = \sqrt{\frac{\lambda_i}{x_i}} \frac{x_i}{2\langle x, \lambda\rangle^{3/2}}$

All vectors are treated as column vectors and for $\lambda, \alpha \in \mathbb{R}^{n+1}, \lambda\alpha^T \in \mathbb{R}^{n+1 \times n+1}$ is the outer product matrix $[\lambda\alpha^T]_{ij} = \lambda_i \alpha_i$. For a proof, see the original research in [46], p 105. The following proposition gives the formula for the determinant (without the normalization).

Proposition 3
The determinant of F_λ^* is proportional to the following term:

$$det F_\lambda^* \propto \frac{\prod_{i=1}^{n+1} (\lambda_i / x_i)}{\langle x, \lambda\rangle^{n+1}}$$

4.1.1 Overview

[46] brings many contributions to information geometry. First he extends Cencov theorem to conditional models. Then he proves an equivalence between the exponential loss for AdaBoost and maximum likelihood for conditional exponential models, gaining some insights from the equivalence and improving on the existing approach. Then he defines the embedding principle that considers each data point as coming from a parametric model and then he looks at the Fisher information on the manifold defined by the estimates of the parameters. Then he describes the multinomial information diffusion kernel that captures the geometry of the data through the embedding into the sphere. The results for using the support vector machines with these kernels for classification are better than linear or Gaussian kernels. The thesis culminates with the metric learning problem that we summarized. The idea of embedding the geometry into some known space through a parametric family of transformations is often used in information geometry and we will see later different approaches to this.

4.2 Manifold Learning

Manifold learning (MAL) is a technique used mostly for nonlinear dimensionality reduction that assumes some given high dimensional data (observations) $y_1, ..., y_n \in \mathbb{R}^D$ lie on a low dimensional submanifold $\left\{\Gamma(z) : z \in \mathbb{R}^d\right\}$ given by a smooth map $\Gamma : \mathbb{R}^d \to \mathbb{R}^D$ with $d \ll D$. It is possible to assume a parametric form for Γ [49], but the majority of MAL techniques use non-parametric methods. Given a family of models $\mathcal{M}$, a model $X_n \in \mathcal{M}$ can be seen as a collection of coordinates $x_1, ..., x_n$ which can be encoded into n distributions over $\{1, 2, ..., n\}$:

$$p_{j|i}(X) = \frac{\exp(-s_{ij}^X)}{\sum_{j \neq i} \exp(-s_{ij}^X)} \forall j \neq i, p_{i|i} = 0 (\forall i) \tag{4.1}$$

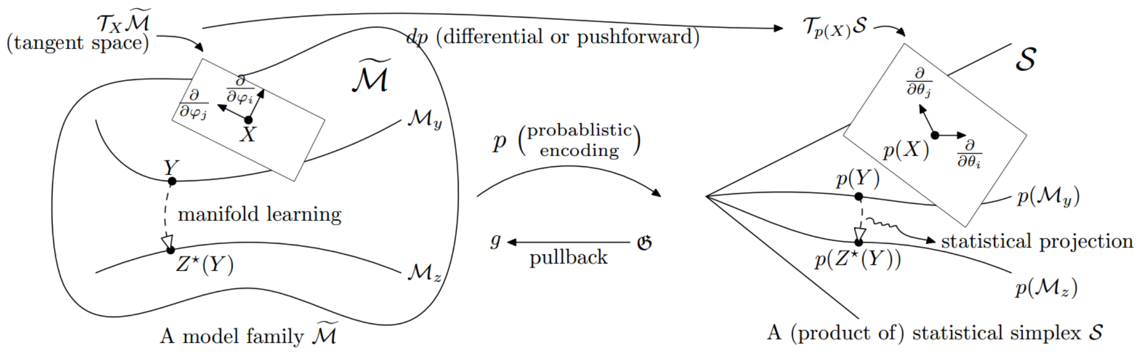
or just one distribution over $\{1, 2, ..., n\}^2$:

$$p_{ij}(X) = \frac{\exp(-s_{ij}^X)}{\sum_{i,j:j \neq i} \exp(-s_{ij}^X)} \forall j \neq i, p_{ii} = 0 (\forall i) \tag{4.2}$$

In both cases s_{ij}^X is a difference measure between x_i and x_j, like a square distance and p is the probability that x_i and x_j are similar, after the appropriate normalization. The first expression represents a conditional probability while the second is a joint probability. The equations above are said to encode distributed local information, because the information in p is distributed sample wise with each sample having limited knowledge, mostly about its neighbors. A widely used MAL technique is SNE and its variations, where the distance function is given by: $s_{ij}^X = \tau_i \left\|x_i - x_j\right\|$ with $\tau_i > 0$ a scalar. One argument for the developments presented next, is the fact that even though the model can have different geometries, after the encoding, $p(X)$ becomes a unified space. For example, $p_{j|i}(X)$ is a point on the product manifold given by $(\mathbb{P}^{n-1})^n$ where $\mathbb{P}^{n-1}$ is the statistical simplex given by $\{(p_1, ..., p_n) | \forall i, p_i > 0, \sum_{i=1}^n p_i = 1\}$. For the second equation $p_{ij}(X)$ is a point on $\mathbb{P}^{n^2-1}$. This perspective enables comparison between different types of models. Any point $(p_1, ..., p_n)$ corresponds uniquely to $(\theta_1, ..., \theta_n)$ through the invertible map $\theta_i = \log(p_i/p_r), \forall i$ with $p_r, 1 \leq r \leq n$ being a reference probability.

Lemma 1
On $\mathbb{P}^{n-1}$ the Riemannian metric is given by: $g(\theta) = p_i(\theta)\delta_{ij} - p_i(\theta)p_j(\theta)$

The Fisher information enables the measuring of information. Through the volume element on the statistical manifold we can measure the information density of a model $\sqrt{\det(g(\theta))}$, it tells us how much information about θ can we get from a single observation. A small volume means a small amount of information, more uncertainty and thus many samples are needed to get a reasonable value for θ. Then if we integrate the volume over a manifold, we get the information capacity of a model family $\mathcal{M} \subset \mathbb{P}^{n-1}$, which is $\int_{\Theta \in \mathcal{M}} \sqrt{\det g(\theta)} d\theta$. The volume is invariant to reparametrization. To be able to reason about any parametric model $\mathcal{M}_r$ on the



$\mathcal{T}_X\widetilde{\mathcal{M}}$ (tangent space)
dp (differential or pushforward)
$\mathcal{T}_{p(X)}\mathcal{S}$
$\mathcal{S}$
$\widetilde{\mathcal{M}}$
$\frac{\partial}{\partial\varphi_j}$
$\frac{\partial}{\partial\varphi_i}$
X
$\mathcal{M}_y$
Y
manifold learning
$Z^\star(Y)$
$\mathcal{M}_z$
p (probablistic encoding)
g pullback $\mathfrak{G}$
$\frac{\partial}{\partial\theta_j}$
$p(X)$
$\frac{\partial}{\partial\theta_i}$
$p(Y)$
$p(\mathcal{M}_y)$
statistical projection
$p(Z^\star(Y))$
$p(\mathcal{M}_z)$
A model family $\widetilde{\mathcal{M}}$
A (product of) statistical simplex $\mathcal{S}$

Figure 4.2: Geometry of manifold learning. Figure from [47].

simplex, we need again the push-forward, which is a mapping from $\mathcal{M}_r$ to $\mathbb{P}^{n-1}$. To get an inner product on $\mathcal{M}_\nabla$ we use the pull-back, the inverse transformation. For this case, the pull-back is given by the following theorem:

Theorem 6
Let $\mathcal{M}_r = \left\{s_{ij}(\phi) | \phi = (\phi_1, ..., \phi_r)^T \in \Phi\right\}$ *be a model family with* $\Phi \in \mathbb{R}^r$. *The pull-back with respect to Equation* (4.1) *is:*

$$g(\phi) = \sum_{k=1}^{n}\left[\sum_{l=1}^{n} p_{l|k}\left(\frac{\partial s_{kl}}{\partial \phi}\right)\left(\frac{\partial s_{kl}}{\partial \phi}\right)^T - \left(\sum_{l=1}^{n} p_{l|k}\frac{\partial s_{kl}}{\partial \phi}\right)\left(\sum_{l=1}^{n} p_{l|k}\frac{\partial s_{kl}}{\partial \phi}\right)^T\right]$$

The metric $g(\phi)$ defined in this way is positive semi-definite, which means the manifold is semi-Riemannian. When considering a model X moving rigidly, meaning it has zero volume, it means it contributes no information. Note that $g(\phi)$ defines a meta-metric. Assuming an ambient space for the manifold of interest $\mathcal{M}$, given by $\widehat{\mathcal{M}}^n D = \left\{X_n = (x_1, ..., x_n)^T | \forall i, x_i \in \mathbb{R}^D\right\}$. The semi-Riemannian metric defines an $(nD \times nD)$ positive semi-definite matrix $\widehat{g}(X)$ for any $X \in \widehat{\mathcal{M}}$. A $D \times D$ sub-block of this matrix can be used to investigate the infinitesimal length $||dx_i||$ on the manifold when x_i is shifted to $x_i + dx_i$.

Theorem 7
$\langle \frac{\partial}{\partial x_i}, \frac{\partial}{\partial x_i} \rangle_{\widehat{g}(X)} = 4\sum_{j=1}^{n} p_{i|j}(1-p_{i|j})(x_j - x_i)(x_j - x_i)^T + 4\sum_{j=1}^{n} p_{j|i}\left(x_j - \sum_{j=1}^{n} p_{j|i}x_j\right)\left(x_j - \sum_{j=1}^{n} p_{j|i}x_j\right)^T$ We see that this looks like a local covariance around x_i with the locality defined by p. If dx_i is orthogonal to the data manifold this would give a small $||dx_i||$ meaning that the samples move along the normal directions, which would give a small information volume. The sample-wise volume element provides a quantification of the information in each sample giving insights into outlier detection or landmark selection for subsampling [50]. This also gives a geometric view of kernel density estimation on a manifold [51], a growing density is equivalent to maximizing the variance of information on $\widehat{\mathcal{M}}$.

4.2.1 Locally Accumulated Information

The complexity of a fixed model is given by a matrix of pairwise differences $s_{ij}(X) = \left\|x_i - x_j\right\|^2$ but many other distance functions can be used. The general idea is to form a model family from X, through variations of X such that the different perspectives help to measure the amount of information. This can be achieved by assuming an isotropic observer on each sample x_i that perceives information encoded by s_{ij} given by:

$$s_{ij}(\tau) = \begin{cases} \tau_i \cdot s^X_{ij} & \text{if } j \in kNN_i \\ +\infty & \text{if otherwise} \end{cases}$$

$\forall i, \tau_i > 0$ looks at samples that are near of far from x_i, in this way the information perceivable at different scales can be incorporated in a consistent manner. k with $2 \leq k \leq n-1$ describes the range as the maximum number of samples that can be observed from any x_i. The goal of such an approach is to ignore some distant relationships in a consistent manner, to also reduce computation. This enables datasets of different sizes to be dealt with in a consistent manner on the same manifold $\mathbb{P}_{k-1}$ and so, they can be compared with each other. If X and k are fixed than the set of $\tau = (\tau_1, ..., \tau_n)$ forms a n-dimensional manifold of models, denoted by $\mathcal{O}^n_{X,k}(\tau)$ that has the following geometry:

Theorem 8
The Riemannian metric of $\mathcal{O}^n_{X,k}(\tau) is g(\tau) = diag(g_1(\tau_1), ..., g_n(\tau_n))$ *with:*

$$\begin{aligned} g_i(\tau_i) &= -\frac{\partial}{\partial \tau_i}\left(\sum_{j \in kNN_i} p_{j|i}(\tau_i) s^X_{ij}\right) \\ &= \sum_{j \in kNN_i} p_{j|i}(\tau_i)(s^X_{ij})^2 - \left(\sum_{j \in kNN_i} p_{j|i}(\tau_i) s^X_{ij}\right)^2 \end{aligned}$$

The scale of this manifold can be computed as in the following definition:

Definition 2
The locally accumulated information (LAI) of X given the visual range k is defined as:

$$|\mathcal{O}_k|(X) = \frac{1}{n}\sum_{i=1}^{n}\left(\int_0^\infty \sqrt{g_i(t)}dt\right)$$

An interesting observation is the fact that $\mathcal{O}^n_{X,k}(\tau)$ resembles a positive orthant $(0, \infty)^n$ and LAI measures its average side length. The LAI equation is a sum over the Fisher information integrated along a curve, which corresponds to a local observations process. LAI effectively measure how "many" distinct distributions are observed during the changing of the observation scale from 0 to ∞. Additional propositions in [47] show how LAI is invariant to scaling and how is always finite and has a lower bound that can be approached by placing X approximately on a regular n-simplex. LAI can be computed with any numerical integrator and it involves n integrations that can be further reduced by subsampling. The cost scales with k ($k \ll n$).

4.2.2 A gap between two models

A major issue in MAL is how the difference between input and output is measured. If such a difference is measured then one can evaluate the embedding quality and MAL can be formulated as an optimization problem. We saw before how each input Y and output Z can be extended to two model families. This section described a way of evaluating the embedding quality by continuously deforming one family into the other along a connecting manifold. Then the volume of this manifold measures their difference. Such a manifold exists for any given Y and Z. Let the model family $\mathcal{H}^{2n}_{Y,Z,k}$ be given by:

$$s_{ij}(c) = \begin{cases} a_i s^Y_{ij} + b_i s^Z_{ij} & \text{if } j \in kNN_i \\ +\infty & \text{if otherwise} \end{cases} \tag{4.3}$$

$\forall i, kNN_i = kNN_i(Y) \cup kNN_i(Z)$ are the input or output neighbors of i, $a_i > 0, b_i > 0$ and $c = (a_1, b_1, ..., a_n, b_n)$ is a global coordinate system. The boundary $b = 0$ reduces to the model family of Y, while the boundary $a = 0$ reduces to the model family of Z. The geometry of the model family $\mathcal{H}^{2n}_{Y,Z,k}$ is given by:

Theorem 9
The Riemannian metric of $\mathcal{H}^{2n}_{Y,Z,k}$ with respect to Equation 4.3 is given by the following block diagonal:

$$g(c) = \begin{bmatrix} g^1_{aa} & g^1_{ab} & & & \\ g^1_{ba} & g^1_{bb} & & & \\ & & \ddots & & \\ & & & g^n_{aa} & g^n_{ab} \\ & & & g^n_{ba} & g^n_{bb} \end{bmatrix}$$

For more details about this metric and an extended discussion, see the original work [47]. The idea is to construct a low dimensional *film* across the manifold connecting Y and Z, and by computing the volume of this film one can then estimate the closeness between input and output boundaries. The authors then align the two lines corresponding to Y and Z as close as possible on each side of the gap. The volume of this film is then an approximation for the minimal effort to shift a continuous spectrum of information from the family of Y to the family of Z. The authors then give the expression for computing this volume and make various arguments about the difference of their technique and other often used in the literature. For example, they say that their MAL technique do not favor a specific type of local information, as do others and by considering the gap between Y and Z an intrinsic difference is revealed. The integration over the continuous spectrum also reveals the true information loss. An important trade-off of this MAL technique is to minimize the volume of the gap between Y and Z, to minimize the lost information, but maximize the volume of Z, meaning to maximize the information retained.

4.3 Conclusion

Lebanon [46] showed through the embedding principle how single data points can be modeled as points on a manifold of probability distributions and in [47] we saw how an entire dataset can be modeled as one single point on a manifold through a probabilistic mapping. Moreover, a geometric perspective on nonlinear embedding and the quality of embedding have been defined. We saw how the volume element [46], a purely geometric measure, can be used to define a simple objective function with the goal of finding a parametric transformation that defines the metric on the manifold, while in [47] the volume is used with two different purposes, first to minimize the lost information and second to maximize the retained information with the purpose of dimensionality reduction. A conclusion that can be drawn from these approaches, is that the modeling of data or pdfs on a manifold is highly flexible, and working with the tools in differential geometry like pullbacks, embedding into a known space through parametric transformations, making use of intrinsic geometric measures like volumes, can greatly enhance the statistical approach and understanding to machine learning.

Chapter 5

Pattern recognition

The following research described is based on [7; 52]. In computer vision, where the dimensionality of the data is usually high (number of pixels) we are interested in finding representations which are meaningful (shape, color histogram, orientation, SIFT, histogram of gradients, linear dynamical systems, covariance matrices, etc.) but also have a lower dimensional representation such that the computation involving such features is efficient, or at least more efficient than in the original pixel space. Obviously such high level features do not live in Euclidean space, and usually the distance function (or a better distance function than the usual L_p norm) is a non-linear function which is usually hard to compute. Shape spaces have been associated with Riemannian manifolds - Kendall's shape space is a complex spherical manifold [53], affine shape spaces are Grassmann manifolds [54], the space of covariance matrices or tensors is a product manifold of the special orthogonal group and the diagonal group, and is not a vector space [55; 56], the space of linear subspaces is a quotient space of the orthogonal group which gives a Grassmann manifold [57], linear dynamical systems can be described as the column space of the observability matrix, which is again a Grassmann manifold [58]. The space of histograms form a simplex in $\mathbb{R}^n$, which is not a vector space. These non-Euclidean spaces have been studied intensively to better understand the geometry of the spaces and be able to define geodesics and adjacent notions to ultimately provide a better understanding for data embedding and better inference and learning [59; 60; 61; 58]. We show in Tables (5.1) the most used manifolds for visual data with their associated distance functions in Table (5.2) and respective applications in Table (5.3). We describe shortly a few manifolds that have been used in vision over the years:

- *Shape features*: Shapes are described by a set of landmarks on the object. When normalizing such that the shape is invariant to translation, scale and rotation, the shapes are shown to reside on a complex spherical manifold [53]. If normalizing again for all the affine transformations, the shapes can be shown to reside on a Grassmann manifold. Recently, shapes were described such that they reside on the manifold of planar curves [62].

- *Covariance features*: When considering a (usually) small region of the image and we take its covariance, this has been shown to encode local shape and texture reasonably well and has been used in human detection [61], object

Table 5.1: Manifolds used for visual data.

Manifold	Numerical representation	Dimension
1. Spherical manifold	n-vector with norm	$n-1$
2. Stiefel manifold	$n \times k$-orthogonal matrix	$nk - \frac{k(k+1)}{2}$
3. Grassmann manifold	$n \times k$-orthogonal matrix	$k(n-k)$
4. Covariance matrices	$n \times n$-symmetric positive definite matrix	$\frac{n(n+1)}{2}$

Table 5.2: Distance functions

1. $d(X_1, X_2) = \cos^{-1}(|x_1^T x_2)$
2. no closed analytical form
3. $d(X_1, X_2) = \sum_i ||\theta_i||^2$ where $\cos(\theta_i)$ are the singular values of $X_1^T X_2$
4. $d(X_1, X_2) = \sqrt{\sum_{i=1}^{N} \log^2 \lambda_i(X_1, X_2)}$, where $\lambda)i$ are generalized eigenvalues of $\lambda X_1 v = X_2 v$

tracking [63] and texture classification [56]. In medical imaging, diffusion tensor MRI produces voxels, each associated with a 3×3 symmetric positive definite matrix [55].

- *Time warps*: A function which is positive and monotonically increasing and maps the unit-interval to itself is called a time-warp function. Their derivatives can be seen as probability density functions, and taking the square roots of pdfs can be seen as describing a sphere in the space of functions. This was used to recognize human activities [64]. Also, sampling close planar curves can be seen as done on a sphere in the space of functions [60].

- *Subspaces*: The set of k-dimensional subspaces of $\mathbb{R}^n$ is referred to as the Grassmann manifold. The Grassmann manifold has been used to describe a linear subspace of 9-dimensions which constitutes the set of images under various illuminations [65]. A related manifold, which is constituted of k basis vectors of $\mathbb{R}^n$ is called the Stiefel manifold (this can be seen also as $n \times k$ tall-thin orthonormal matrices) and has been used in face recognition [66]. A Stiefel manifold can help in finding projection of some data with desired statistical properties like sparsity or discriminability [67]. .

- *Dynamic models*: As we mentioned earlier, the space spanned by the columns of the observability matrix of a dynamic model (be it LDS, ARMA, etc.) can be described as a point on the Grassman manifold [58].

Table 5.3: Applications

1. Kendall's shape space [53], probability density functions [68].
2. Face recognition [66], dimensionality reduction [67].
3. Image set modeling [69; 70; 71; 72], dynamic models [73; 58]
4. Region descriptors [63; 61; 56], diffusion tensor imaging [55]

5.1 Hashing for neighbor search

Consider the problem of searching for visually similar patterns, being videos or images, either for classification or for object or scene recognition. Usually this is done in Euclidean space, however the data as well as distances, are not accurately described in Euclidean space, this is why usually one would like a smarter (data-dependent) embedding of the data. When considering the data lying on a non-Euclidean, differentiable manifold, the distances between data points are computed using geodesic distances. However the same distance computation problem remains, one needs to compute pairwise distances between all data points, which is computationally not feasible. However the fact that we assume the data to be on a manifold enables us to reason differently about this problem. First of all, just considering distances which are data dependent, would require an additional preprocessing cost compared to the Euclidean case, and even though the distances would be more relevant for the problem at hand the computational cost would be too high, especially considering the size of video/image datasets. We will describe next some solutions to geodesic distance computation and in addition, as with most similarity based techniques, we are interested a relevant centroid/mean that describes the data or subsets of the data. A simple solution, but one that provides only approximations is to consider the vector-space defined by the tangent space of the manifold, which can approximately describe the data and distances between points with an associated L_2 norm. However, any exact solution on the tangent space will be an approximation to the original problem, just because the tangent space can be seen as an approximation to the original manifold, approximating geodesics with the distances in the tangent space.
A very interesting approach to similarity search and indexing is based on hashing. Hashing was initially and mostly used in cryptography, where a specific (highly-nonlinear and non-invertible) transformation was critical for the safe keeping of passwords or sensitive-data. Hashing is also used to check for the integrity of some data packet. In short, it is a summarization of the data which has a very low probability of being a description of other data. To keep in mind is that the forward transformation is quite fast (from data to the hash key). When hashing images or videos, one should take into account the semantic meaning and have similar hash keys describe similar videos or images. This is known as locally sensitive hashing (LSH) [74]. Next, we give a short description of the locally sensitive hashing problem. Let $X = x_i \in \mathbb{R}^n$ be a set of data points, and assume we want to search for a similar point (this might be an image) to a query point (q), then assuming there exists $x \in X$ such that $d(x,q) \leq r$ then we want to know with high probability that

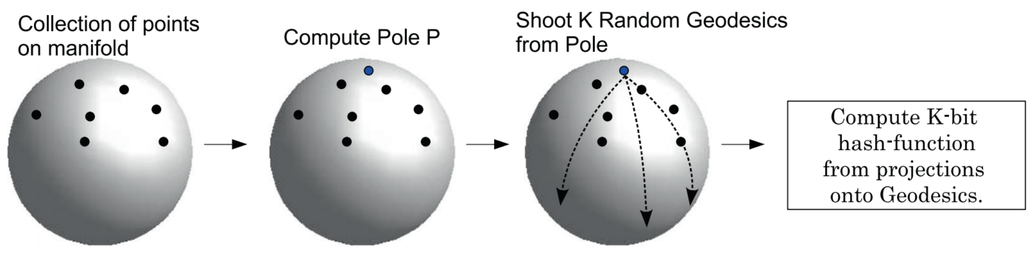

Collection of points on manifold
Compute Pole P
Shoot K Random Geodesics from Pole
Compute K-bit hash-function from projections onto Geodesics.

Figure 5.1: Computing Karcher mean as the pole of the manifold and then shooting K random geodesics from the pole defines K projections that give a K-bit hash function. Figure from [52]

the point we will retrieve (x') will be close to q, i.e. $(x',q) \leq (1+\epsilon)r$. LSH constructs $\mathcal{H}$, a family of hash functions, which are applied to $\dot{X}$ such that for any $u,v \in X$ the following hold:

$$d(u,v) \leq r \Rightarrow Pr(h(u)) = h(v) \geq p_1$$
$$d(u,v) \geq (1+\epsilon)r \Rightarrow Pr(h(u) = h(v)) \leq p_2$$

The function h can be described by random projections in the following way: $h(v) = sgn(v.r)$ where r is a random unit vector and sgn is the sign function, thus h is binary with values in $-1,1$. Another choice that gives integer hash values is given by: $h(v) = [\frac{v.r+b}{w}]$ where r is a direction chosen from a stable distribution and b is a random number between $[0,w]$. Then a $k-bit$ hash function is constructed by concatenating k hash functions in $H(x) = [h_1(x), h_2(x), ..., h_k(x)]$. A hash table L is then constructed by randomly choosing $H_1, H_2, ..., H_L \in \mathcal{H}^k$. Then all data points are hashed into these L tables (i.e. a k-bit hash function is computed for each $l \in L$). The important choices are K and L such that one knows with high probability that $(r,\epsilon)-NN$ of q is found. This was the basic LSH, however when considering the same procedure on a manifold, one needs to find a pole p and then embed all the data points into the tangent space (which is a vector-space) at p. The Karcher mean is usually chosen to represent the pole. Then a hash function can be described as:

$$h_p(x) = sgn(\log_p(x).v)$$

with $v \in T_p\mathcal{M}$ being a random sample direction in the tangent space at p. The analogous to the integer function specified before is given by:

$$h_p(x) \approx \left\lceil \frac{\log_p(x).v + b}{w} \right\rceil$$

where again $v \in T_p\mathcal{M}$ is random direction in the tangent space at p. Choosing K different directions gives a K-bit hash function. The process is depicted in Figure (5.1)

5.1.1 Hashing covariance matrices

When applying the above framework to a texture classification problem, the authors show that with significantly fewer computations they can reach a similar level of accuracy. The texture database used is the Brodatz[1], which contains 111 textures, each of 640x640. The authors split the images in 4 parts of 320x320 each and use 2 in training and 2 in testing. Then they select a rectangular window for each image, with a random size between 16x16 and 128x128 and for each of these they compute the following:

$$F(x,y) = \left[I(x,y), |\frac{\partial I}{\partial x}|, |\frac{\partial I}{\partial y}|, |\frac{\partial^2 I}{\partial x^2}|, |\frac{\partial^2 I}{\partial y^2}| \right]^T.$$

Then the final descriptor for the region is the 5x5 covariance of F. So the set we are interested in is a manifold of covariance matrices. The authors then sample 100 windows for each image and they do the same for testing. For each of these they find the nearest neighbor and assign the image-class to the respective nearest-neighbor image class. Then the they use majority voting to assign the texture class to the image. Now the training set consists of 111x2 images which gives rise to 22,200 covariance matrices for training. Computing the nearest neighbor is computationally expensive, however using the geodesics computations described above requires significantly less computations.

5.2 Linear dynamical models

We now consider data that is endowed with features that evolve over time. There are many such time-series data, like video sequences, or any kind of motion, human or otherwise, dynamics textures, sound sequences, etc. Usually modeling these types of systems makes use of what is called a state space model. That is a parametric model which evolves over time, thus it has also a set of parameters which deals explicitly with time through a state transition function, i.e. describing how the system evolves from one step to the next. When considering such dynamics as being linear the model is given by the linear system:

$$\begin{aligned} f(t) &= Cz(t) + w(t) \text{ with } w(t) \sim \mathcal{N}(0,R) \\ z(t+1) &= Az(t) + v(t) \text{ with } v(t) \sim \mathcal{N}(0,Q) \end{aligned}$$

where $z \in \mathbb{R}^d$ is called the hidden state vector, A is the transition matrix and C is called the measurement matrix. $f \in \mathbb{R}^d$ are the observed features and w, v are some noise associated with the transition function and the measurement function respectively, modeled as normal distributions with means 0 and covariance R, Q. The system starts from the initial condition $z(0)$, and the expected observations are:

$$\begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ . \\ . \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \\ . \\ . \end{pmatrix} z(0) = O_\infty(M) z(0)$$

[1]http://www.ux.uis.no/ tranden/brodatz.html

This is a time invariant model, the observation sequence generated by such a model lies in the column space of the observability matrix given by $O_\infty = [C^T, (CA)^T, ...]^T$, however the fact that human actions are finite allows to consider a finite observation sequence of length n. The subspace spanned by the column of the observability matrix is the Grassmann manifold, and a point on it is a single instantiation of the dynamical model described above. The Grassmann manifold is denoted by $G_{k,m}$ and can be seen as a quotient of the special orthogonal group $SO(m)$ [75]. Geodesic paths on the manifold defined by the orthogonal group are given by one-parameter exponential flows $t \to \exp(tB)$ with $B \in \mathbb{R}^{m\times m}$ is a matrix which is skew-symmetric. The fact that the Grassmann manifold is a quotient group makes the matrix B to have a more particular form given by: $B = \begin{pmatrix} 0 & A^T \\ -A & 0 \end{pmatrix}$ with $A \in \mathbb{R}^{(m-k)\times k}$. The matrix A is a parametrization of the direction and speed of the geodesic flow. Let Y_0 be an orthonormal basis on the manifold, then considering a direction matrix A, the geodesic flow from Y_0 in the direction B is given by: $Y(t) = Q\exp(tB)J$ with $Q \in SO(m)$ and $Q^TY = J$, with $J = [I_K; 0_{m-k,k}]$. Using this geometry parametric and non-parametric classifiers can be employed for robust performance. Unsupervised clustering can also be used making use of this geometry. Considering the task of labeling action performed by humans in videos, one should consider the fact that action sequences can have different lengths, and thus each such segment should be parametrized by a relatively small number of parameters for which the linear dynamical model is a good choice. In this way k-mean clustering of video-segments can be performed successfully [76].

5.3 Non-parametric Fisher

We argued from multiple perspectives for the parametric Fisher as a fundamental role for modeling probability distributions on manifolds. However, there is a non-parametric version of the Fisher that has been shown to be of interest in shape and functional data analysis, which we will describe next [68]. Let P_0 be the set of positive density functions on $[0,1]$. The tangent space at a point p on this manifold is given by the set of all vectors $v : [0,1] \to \mathbb{R} | \int_0^1 v(x)dx = 0$. The nonparametric version of the Fisher is:

$$\langle v_1, v_2 \rangle_p = \int_0^1 v_1(t)v_2(t)\frac{1}{p(t)}dt \tag{5.1}$$

for any two vectors $v_1, v_2 \in T_p(P_0)$. Under the square root transformation $q(t) = \sqrt{p(t)}$ the Fisher transforms into the Euclidean metric. Let a small perturbation of $p(t)$ be given by $\delta p(t)$, this transforms into $\delta q(t) = \frac{1}{q\sqrt{p(t)}}\delta p(t)$ and when substituting for example $v_1(t) = 2\sqrt{p(t)}q(t)$ as the perturbation $\delta p(t)$ in Eq. (5.1) we get the L_2 inner product. Noticing that under this transformation the integral of q^2 is 1 this gives a Hilbert sphere $\mathbb{S}_\infty$ for which we know how to compute geodesics and exponential or their inverse maps. A very noticeable property of the Fisher metric, which we mentioned previously is the invariance to reparametrization, which when considering parametrized curves (for shape analysis for example), this action of reparametrization just changes the coordinates of a curve, but not the actual shape. However, to proceed further one needs to extend the Fisher to the space of parametrized curves, because originally it was defined for the space of positive density functions. The authors propose the following approach: first step is to work with the cumulative distributions function $f : [0,1] \to [0,1]$ which is a CDF for $p \in P_0$, $\dot{f} = p$ with the space of CDFs being $\mathcal{F}_0$. The tangent space for $\mathcal{F}_0$ is given by: $T_f(\mathcal{F}_0) = \left\{ w : [0,1] \to \mathcal{R} | \int_0^1 \dot{w}(t) = 0 \right\}$ with the metric on $\mathcal{F}_0$ given by:

$$\langle w_1, w_2 \rangle_p = \int_0^1 \dot{v}_1(t)\dot{v}_2(t)\frac{1}{\dot{f}(t)}dt$$

Changing again the variable as $q(t) = (\dot{f}(t))\sqrt{|\dot{f}(t)|}$ the Fisher metric can be again converted to the L_2. This function is called the square root slope function and allows to deal with functions that can be zero in the interval $[0,1]$. This transformation is useful for the large class of absolutely-continuous functions on $[0,1]$. Now going back to the initial goal of defining a metric for the class of (absolutely continuous) parametrized curves and considering the magnitude of the velocity vector of a curve as $r(t) = |\dot{\beta}(t)|$ and the direction $\Theta(t) = \frac{\dot{\beta}(t)}{r(t)}$ we now have two functions that define the curve: $r : [0,1] \to \mathcal{R}_+$ and $\Theta : [0,1] \to \mathcal{S}^{n-1}$. This new metric for curves is called the elastic metric and is defined by the following family of metrics:

$$\langle (\delta r_1, \delta\Theta_1), (\delta r_2, \delta\Theta_2) \rangle_{(r,\Theta)} = a \int_0^1 \delta r_1(t)\delta r_2(t)\frac{1}{r(t)}dt + b \times \int_0^1 \langle \delta\Theta_1(t), \delta\Theta_2(t) \rangle r(t)dt$$

with $a, b > 0$. If $n = 1$, Θ vanishes and the formula is now the one for analyzing PDFs. To further simplify computations for the elastic metric a square root velocity function (SRVF) is again defined as $q(t) = \sqrt{r(t)}\Theta(t) = \frac{\dot{\beta}(t)}{\sqrt{|\dot{\beta}(t)|}}$ (see []) and putting $a = \frac{1}{4}$ and $b = 1$ the elastic metric becomes the L_2 metric. Then quantities like geodesics can be computed in the SRVF space for the L_2 metric. We show examples in Figure (5.2) for some geodesics computed in this space of planar, closed curves, under the elastic metric but using the SRVF representation. For further details see [].

5.4 Facial analysis and domain adaptation

When it comes to facial features, manifold based techniques are useful for two related tasks. First is the age estimation problem, which uses landmark information from the face as the age of a person and his/her identity can be encoded in the shape of a face, in addition the texture on the face, both evolving over a significant period of time. These features can be extracted as facial landmark points, a collection which can be modeled as a point on the Grassmann manifold. Consequently, age estimation and face recognition are then a regression problem and a classification problem respectively [77]. As the long term change in the face shape can encode

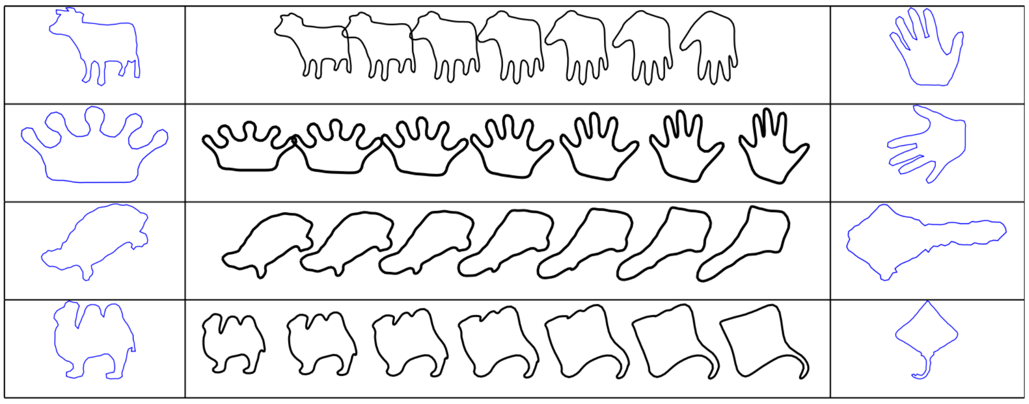


Figure 5.2: Geodesics between shapes using the elastic metric under the SRVF representation. Figure from [7]

age information, likewise short term change in facial features can encode expression information. Expressions are represented as a collection of landmark points in the affine shape-space as an approximation to the projective shape-space. As such, the deformations of a face can be encoded as statistical variations and motions on the affine-shape manifold [78]. A challenge in face recognition problems is to account for the blur of an image, which might be due to low resolution for example. This is done by considering the space spanned by blurred images of the original one [79]. This task is accomplished by first creating a blur kernel (by convoluting an image with a complete set of orthonormal basis functions) and then showing that the subspace created by the original image and the blurred ones are equal under the assumptions of no noise and some properties of the blur kernel. These subspaces can be viewed as points on the Grassmann manifold. In current days there is an emphasis on the 'big data' paradigm, where different types of sources can be used to obtain more and more data of some specific sort. For example when training a classifier a particular data set is used, however we would then like to use our learned model to reason about other sources of data, for example if we're talking about images and object recognition, maybe the new data is subject to different illuminations or viewpoints. This dataset bias has been dealt with by the methodology which is referred to as Domain Adaptation (DA). There are two types of DA, depending on the nature of the test set, if there some partial labels to the new domain we want to adapt our model, then it is called semi-supervised, otherwise, if we have no labels at all, it is called unsupervised. The semi-supervised uses the labeled samples for learning the transformation from the original domain to the new target domain [80], while the unsupervised version assumes a. certain types of transformations between domains [81], b. the existence of features which are common to both domains [82] or c. the existence of a latent space where the distance between the two domains is minimal [83]. The unsupervised version is more realistic and general giving the fact the most of the time there are no labels for the new data domain. A critical problem in such a scenario is the path for such a transformation from the source domain to the target domain. [84] take a geometrical perspective on such a path, by making use of the statistical information in the target and source domains. An adaptation path is computed that generates a number of intermediate representations between the two domains, without the need for the invariance assumption between domains. The idea is, in short, to create generative subspaces of the same dimension thorough PCA, and then model these as points on the Grassman manifold, while geodesics on this manifold now represent domain shifts. When sampling points along geodesics intermediary cross-domain representations can be obtained on which a discriminative classifier can be trained for recognition. This direction was then further explored with respect to better sampling strategies along the geodesic [85], noise or outlier removal [86], or data-dependent regularization for handling different domains [87].

Chapter 6

Contribution

We are interested in making learning in deep networks more efficient, enhance stochastic gradient descent with new insights from information geometry without making use of the Fisher explicitly. We are mostly interested in learning how to learn, reason about the space in which learning happens. To this end, we present the following ideas. We describe a new class of algorithms that are based on the dynamics of learning, this can be done considering the dynamics between the data, in time, given the base learning algorithm, the deep SGD machine. We model learning instances as points on a manifold and thus the meta-learning level requires to consider geometrical properties of the learning dynamics, like geodesics on this manifold. In our framework learning is part of the model itself (given the data), the weight changes are the observations for our learning framework.

6.1 Bayesian metric learning

As we mentioned many times, the tangent space at a point defines a linear approximation to the underlying manifold at that point. When the manifold is considered to support the given data points, even though we consider the decomposition of the Jacobian $\frac{\partial h}{\partial x}$ as unraveling the tangent space at x, it is actually the tangent space at the non-linear transformation of x in the model space (be it a neural network or autoencoder or some smooth non linear transformation, a parametric model). Actually, we note that even if the transformation is not smooth, the manifold described by a parametric model is. By decomposing the Jacobian at each given data point, we find the principal tangent directions of the model given the data $p(\mathcal{M}|x_i)$ with $i = 1,...,n$ where $D = x_1,...,x_n$ is the training data. However, we are interested in the underlying metric which the manifold is endowed with. As we saw in [30], by finding these directions one can then penalize the cost function by the sum of all the derivatives of the output with respect to the input in these tangent directions. However, reasoning further in the same manner, we note that if these directions are important when forward propagating the input, they should be important also when backpropagating, or at least a subset of them. A subset because: the tangent space at a point defines the major directions of change of the model given the input, however when learning we change the h function as a function of the error, thus the Jacobian changes as well. We are interested in finding those directions for which learning (that is, changing the weights) is most relevant. But for this we need an estimation of the metric present on the manifold. The metric is completely described by the inner products $g_x(\partial_i, \partial_j)$ of the basis elements of the tangent space at a point ∂_i with $i = 1, \ldots, n$, i.e.:

$$g_x(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i u_j g_x(\partial_i, \partial_j)$$

The Gram matrix $G(x)_{ij} = g_x(\partial_i, \partial_j)$ completely described the metric on the manifold. From the successive approximations at the individual points, we can approximate the metric of the underlying manifold through a Bayesian approach. We consider a posterior distribution of simple parametrized metrics and as data points arrive and we estimate the tangents to them, we can then incorporate these observations into the parametric model of the metric. Thus, with more data points, our estimate of the metric becomes more accurate. Learning a parametric metric has been done before and it has been shown to be fruitful in [46]. A parametric metric is given by:

$$\mathcal{G} = \left\{ g^{\theta} | \theta \in \Theta \subset \mathbb{R}^k, k \ll n \right\}$$

where θ are the parameters, k is their dimension which is much smaller than the actual manifold we are approximating, i.e. the manifold described by the initial n-dimensional model (if it is neural network, n is the number of weights). We will then try to use this metric to move into the model configuration space (adjust weights) to learn better, more efficient. One of the main points of using a Bayesian approach for estimating the metric is that the procedure of finding a tangent space at each data point is very expensive, thus by knowing the metric, this would not be needed anymore. The Bayesian approach gives a way of calculating the posterior distribution of the parameters of the metric given the observations. However we still need an important ingredient to our approach. Assume we already have an estimated metric on the manifold (but keep in mind that this metric is parametric, which means it has a limited number of parameters, much smaller than the actual number of neurons), and we know in which direction is best to move now to minimize the error, the main problem is, how do we move in weight space such that there is consistency between the learned metric and the weight changes. We need a consistent transformation from the parameters of the metric estimated from the tangent spaces to the model space where many more weights probably correspond to one or few parameters of the metric learned. A similar approach is taken in [88] but there they use a linear activation function, that is why this is possible. We can use for such a transformation a random projection or a similar transformation. This guarantees (based on some properties of the transformation, through the Johnson-Lindenstrauss lemma [89]) especially in high dimensional spaces, that some properties are preserved, that we will catch some meaningful relations between the two manifolds. A random projection is basically a projection of the original data onto a lower k-dimensional subspace (we note here that if we consider many such projections and also take them to be orthogonal this takes us to a Stiefel manifold, where we have closed forms for geodesics, etc.). Let the X be a $d \times N$ matrix consisting of N observations in d dimensions, and a random projection matrix R which is $k \times d$, then the projection is given by:

$$X^{rp}_{k\times N} = R_{k\times d} X_{d\times N}$$

To note here is that random projections are computationally cheap. In our case we are interested in N which is much smaller than d (which is very large for deep networks), so we can change the roles, without affecting the overall goal of the projection. If we are also minimizing the L_1 norm, or if we select the projection matrix to be sparse, this would force sparsity in the representation, that is through the transformation a small number of weights will need to be changed to account for the required direction of movement. This would be similar to the dropout technique where just a subset of weights are changed at each iteration, but we would use the same transformation at each iteration, instead of randomly selecting a subset of the weights to change, and in the end, in the testing phase the need for averaging found in dropout would not be needed anymore. If enforcing orthogonality of the transformation would spread the change almost equally among the weights, and this would be equivalent to the whitening transformation found in the newly discovered Natural Neural Networks presented in Section (3.3.1).

6.2 Supervised pretraining

When considering the original penalty term of :

$$\Omega(x) = \sum_{u\in\mathcal{B}_x} \|\frac{\partial o}{\partial x}(x)u\|^2$$

which when added to the cost function minimizes the sensitivity of the output with respect to the local chart (the principal tangent directions) we note that this does not take into account the actual task of classification. It is indeed true that we want this insensitivity to the local chart at each data point, but this is true only in the general sense. When looking at two different data points we observe that if they belong to the same class we indeed want this, however when considering different classes we would like the sensitivity to be maximum. We consider the term:

$$\Omega(x_i, x_j) = \sum_{u_i\in\mathcal{B}_{x_i}} \sum_{u_j\in\mathcal{B}_{x_j}} \|\frac{\partial o}{\partial x_i}(x_i)u_i \frac{\partial o}{\partial x_j}(x_j)u_j \; \delta_{ij}\|^2$$

with $\delta_{ij} = 1$ if x_i, x_j belong to the same class and $\delta_{ij} = -1$ if x_i, x_j belong to different classes. In this way can also change the metric such that it maximizes distances between data points in different classes and minimizes distances between datapoints in the same class. If we don't consider the above function as a penalizing term and if we consider a family of local charts at each x which is parametrized by some set of parameters λ we can embed the vectors directions into a new function which is a function the chart. If we restrict this function to be positive then we can see it as a probability density function of o, λ and x (because the family of u vectors depend on each x), we can write is as $p(o;\lambda;x)$, and if we leave out the norm and the sign associated with the penalty, i.e. δ_{ij}, we then get:

$$\Omega(x_i, x_j) = \frac{\partial p(o;\lambda;x)}{\partial x_i}(x_i)\frac{\partial p(o;\lambda;x)}{\partial x_j}(x_j)\ p(o;\lambda;x)$$

We see that this looks a lot like the Fisher but where we have two terms, one which depends only on the input and output and can be computed, and the other one that depends on the local chart at each point x. This can give rise to an incremental type of learning where we consider more tangent vectors as learning advances. The first few principal tangents define the main directions of learning, but as we advance we need more minute adjustments so we can consider also smaller directions for learning. This can be seen as greedy learning where each added direction builds up on the existing ones, without affecting the previous learning. This type of greedy incremental learning is welcome in the deep learning paradigm as the huge number of parameters makes many approaches intractable.

6.3 Accelerated learning

When dealing with deep networks a big overhead in their training is the training algorithm. Using SGD works most of the time, but this is due to the huge size of the network that has many local optima (actually in deep nets there are exponentially more local optima than in shallow nets [90]), thus it is easy for SGD to find one of these. The learning rate is also of critical importance and many techniques exist to improve the convergence speed, among which a decaying learning rate schedule or constant are the most often used. However choosing a schedule or a particular magnitude for the learning rate biases the learning such that just certain local optima are reached. We propose next a way of inferring the learning path for a deep neural network based on information geometry. This approach then permits to predict where the network will be (the weight configuration - the actual values of the weights) after a certain number of learning steps with a given learning rate without the need to actually go through all SGD iterations to get there. We call it accelerated learning as it predicts where the learning will be after a period of many steps (this is dependent of how nonlinear the learning process is in the new space) and thus takes the network to that state. We model the weight configuration after each SGD iteration as a point on a manifold. Thus, a move from one iteration to the next is a geodesic on this manifold. The geodesic is a parametrized curve in $\mathbb{R}^n$ governed by n parameters λ, where n is the dimension of this newly defined manifold. Each observation (i.e. SGD iteration) gives new information on the value of the parametrized curve, that can be learned through statistical inference. Now we are already in the realm of Bayesian statistics, we have observations, a choice of the model (this can be anything from simple exponential models to mixtures, etc.) that defines the parameters of the curve and a loss function, which is given by the error predicting the next point on the curve, as given by the next iteration of SGD. We thus have a model that represents the learning curve, effectively, which we can use to predict next values and thus take the network configuration to some future predicted state. In practice, we envision the algorithm performing well for a limited number of predicted states and then requiring retraining. This should be a function of how well the model catches the learning dynamics but also how often the learning dynamics changes. We expect the model to readjust itself at different levels of learning (the curve will probably be different at the start and the end of learning). However a problem remains. How do we connect the actual weight configuration space to the parameter space of the curves (or the model we are using to generate these parameters). This will probably make use of the Frenet frame, which is an orthonormal moving reference frame defined for each point of the curve and is the main method in differential geometry to describe properties of the curve. The Frenet frame is invariant under reparametrization. We could also employ a different approach and assume the learning is optimal and also assuming that minimum geodesics are defined by optimum learning, amounts to considering the geodesic defined by two subsequent iterations of SGD as a minimum geodesic on this manifold. With this assumption we can learn the parameters (direction and speed) of the geodesic flow starting at the first iteration and then simulate the flow such that we don't need to go through each SGD iteration. In particular, instead of using a parametrized curve we can use a linear dynamical system that models the learning dynamics. It has been shown that linear dynamical systems can be modeled as points on a Grassmann manifold, where we have a closed form for the equation of a minimal geodesic. The observations for the LDS will be the weights (or some transformation of them) and the cost function is given by the prediction error at each step of the SGD. When the error is small enough our LDS has caught the dynamics of learning and thus we can predict with some degree of accuracy where the learning will take us after a number of SGD steps. The one parameter geodesic flow on the Grassmann manifold is given by:

$$t \to \exp(tB) \text{ with } B = \begin{pmatrix} 0 & A^T \\ -A & 0 \end{pmatrix}$$

where A parametrizes the speed and direction of the geodesic flow. As we said, by assuming the SGD iterations define this flow, we can learn these parameters such that they accurately catch the time evolution of the flow.

Bibliography

[1] S.-I. Amari and H. Nagaoka, *Methods of information geometry*, vol. 191. American Mathematical Soc., 2007. pages 3, 4, 10, 12, 15, 16, 18, 22, 25

[2] S.-I. Amari, “Information geometry on hierarchy of probability distributions,” *Information Theory, IEEE Transactions on*, vol. 47, no. 5, pp. 1701–1711, 2001. pages 3

[3] M. K. Murray and J. W. Rice, *Differential geometry and statistics*, vol. 48. CRC Press, 1993. pages 3, 11

[4] J. M. Lee, *Smooth manifolds*. Springer New York, 2003. pages 5, 6

[5] J. Lawson, *Differential Geometry*. Course Syllabus, 2006. pages 6

[6] W. Rossmann, *Lie groups: an introduction through linear groups*, vol. 5. Oxford University Press, 2002. pages 7

[7] R. Li, P. Turaga, A. Srivastava, and R. Chellappa, “Differential geometric representations and algorithms for some pattern recognition and computer vision problems,” *Pattern Recognition Letters*, vol. 43, pp. 3–16, 2014. pages 8, 53, 60

[8] S. I. Costa, S. A. Santos, and J. E. Strapasson, “Fisher information distance: a geometrical reading,” *Discrete Applied Mathematics*, 2014. pages 10, 23

[9] B. S. Clarke and A. R. Barron, “Information-theoretic asymptotics of Bayes methods,” *Information Theory, IEEE Transactions on*, vol. 36, no. 3, pp. 453–471, 1990. pages 9

[10] N. Cencov, “Statistical decision rules and optimal inferences, translation of math,” 1982. pages 16

[11] F. Nielsen and V. Garcia, “Statistical exponential families: A digest with flash cards,” *arXiv preprint arXiv:0911.4863*, 2009. pages 18

[12] R. B. Grosse, C. J. Maddison, and R. R. Salakhutdinov, “Annealing between distributions by averaging moments,” in *Advances in Neural Information Processing Systems*, pp. 2769–2777, 2013. pages 18

[13] S.-I. Amari, “Natural gradient works efficiently in learning,” *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998. pages 20, 29

[14] C. Radhakrishna Rao, “Information and accuracy attainable in the estimation of statistical parameters,” *Bulletin of the Calcutta Mathematical Society*, vol. 37, no. 3, pp. 81–91, 1945. pages 20

[15] L. Malagò and G. Pistone, “Natural gradient flow in the mixture geometry of a discrete exponential family,” *Entropy*, vol. 17, no. 6, pp. 4215–4254, 2015. pages 21

[16] R. Pascanu and Y. Bengio, “Revisiting natural gradient for deep networks,” *arXiv preprint arXiv:1301.3584*, 2013. pages 21, 22, 29, 30

[17] J. Martens, “New perspectives on the natural gradient method,” *Computing Research Repository*, vol. abs/1412.1193, 2014. pages 21

[18] T. Schaul, “Natural evolution strategies converge on sphere functions,” in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pp. 329–336, ACM, 2012. pages 22

[19] J. Sohl-Dickstein, “The natural gradient by analogy to signal whitening, and recipes and tricks for its use,” *arXiv preprint arXiv:1205.1828*, 2012. pages 23

[20] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, “Stochastic variational inference,” *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013. pages 24, 27, 29, 30

[21] M.-A. Sato, “Online model selection based on the variational Bayes,” *Neural Computation*, vol. 13, no. 7, pp. 1649–1681, 2001. pages 28

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003. pages 29

[23] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, “Hierarchical dirichlet processes,” *Journal of the american statistical association*, vol. 101, no. 476, 2006. pages 29

[24] T. Heskes, “On “natural” learning and pruning in multilayered perceptrons,” *Neural Computation*, vol. 12, no. 4, pp. 881–901, 2000. pages 30

[25] S.-I. Amari, “Natural gradient learning for over-and under-complete bases in ICA,” *Neural Computation*, vol. 11, no. 8, pp. 1875–1883, 1999. pages 30

[26] G. Desjardins, K. Simonyan, R. Pascanu, and K. Kavukcuoglu, “Natural neural networks,” *arXiv preprint arXiv:1507.00210*, 2015. pages 30, 31, 32, 33

[27] G. Montavon and K.-R. Müller, “Deep Boltzmann machines and the centering trick,” in *Neural Networks: Tricks of the Trade*, pp. 621–637, Springer, 2012. pages 32

[28] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015. pages 32

[29] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012. pages 32

[30] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller, "The manifold tangent classifier," in *Advances in Neural Information Processing Systems*, pp. 2294–2302, 2011. pages 33, 34, 36, 62

[31] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *Machine Learning and Knowledge Discovery in Databases*, pp. 645–660, Springer, 2011. pages 34

[32] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 833–840, 2011. pages 35

[33] P. Simard, Y. LeCun, and J. S. Denker, "Efficient pattern recognition using a new transformation distance," in *Advances in neural information processing systems*, pp. 50–58, 1993. pages 35

[34] P. Simard, B. Victorri, Y. LeCun, and J. Denker, "Tangent prop-a formalism for specifying selected invariances in an adaptive network," in *Advances in neural information processing systems*, pp. 895–903, 1992. pages 35

[35] I. Grondman, L. Buşoniu, G. A. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 1291–1307, 2012. pages 36, 38, 39, 40

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. pages 37

[37] S. KAKADE, "A natural policy gradient," *Advances in Neural Information Processing Systems*, vol. 14, pp. 1531–1538, 2001. pages 38

[38] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pp. 1–20, 2003. pages 39

[39] J. A. Bagnell and J. Schneider, "Covariant policy search," IJCAI, 2003. pages 39

[40] J. Park, J. Kim, and D. Kang, "An RLS-based natural actor-critic algorithm for locomotion of a two-linked robot arm," in *Computational Intelligence and Security*, pp. 65–72, Springer, 2005. pages 40

[41] S. Girgin and P. Preux, “Basis expansion in natural actor critic methods,” in *Recent Advances in Reinforcement Learning*, pp. 110–123, Springer, 2008. pages 40

[42] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, “Natural actor–critic algorithms,” *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009. pages 40

[43] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, “Natural evolution strategies,” in *Evolutionary Computation, 2008.(IEEE World Congress on Computational Intelligence).*, pp. 3381–3387, IEEE, 2008. pages 40, 41

[44] A. Berny, “Selection and reinforcement learning for combinatorial optimization,” in *Parallel Problem Solving from Nature PPSN VI*, pp. 601–610, Springer, 2000. pages 41

[45] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001. pages 42

[46] G. Lebanon, *Riemannian geometry and statistical machine learning*, vol. PhD Thesis. Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005. pages 45, 46, 47, 52, 63

[47] K. Sun and S. Marchand-Maillet, “An information geometry of statistical manifold learning,” in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1–9, 2014. pages 45, 49, 52

[48] G. Salton and J. Michael, “Mcgill,” *Introduction to modern information retrieval*, pp. 24–51, 1983. pages 47

[49] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006. pages 48

[50] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008. pages 49

[51] Y. Bengio, P. Vincent, J.-F. Paiement, O. Delalleau, M. Ouimet, and N. LeRoux, “Learning eigenfunctions of similarity: linking spectral clustering and kernel PCA,” tech. rep., Technical Report 1232, Departement d’Informatique et Recherche Oprationnelle, Universite de Montreal, 2003. pages 49

[52] P. Turaga and R. Chellappa, “Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications,” in *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 282–289, ACM, 2010. pages 53, 56

[53] D. G. Kendall, “Shape manifolds, procrustean metrics, and complex projective spaces,” *Bulletin of the London Mathematical Society*, vol. 16, no. 2, pp. 81–121, 1984. pages 53, 55

[54] E. Begelfor and M. Werman, "Affine Invariance Revisited," in *Computer Vision and Pattern Recognition, Computer Society Conference on*, vol. 2, pp. 2087–2094, IEEE, 2006. pages 53

[55] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006. pages 53, 54, 55

[56] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Computer Vision–ECCV 2006*, pp. 589–600, Springer, 2006. pages 53, 54, 55

[57] J. Hamm and D. D. Lee, "Extended Grassmann kernels for subspace-based learning," in *Advances in Neural Information Processing Systems*, pp. 601–608, 2009. pages 53

[58] P. Turaga, A. Veeraraghavan, and R. Chellappa, "Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008. pages 53, 54, 55

[59] P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi, "Principal geodesic analysis for the study of nonlinear statistics of shape," *Medical Imaging, IEEE Transactions on*, vol. 23, no. 8, pp. 995–1005, 2004. pages 53

[60] A. Srivastava, S. H. Joshi, W. Mio, and X. Liu, "Statistical shape analysis: Clustering, learning, and testing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 4, pp. 590–602, 2005. pages 53, 54

[61] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on Riemannian manifolds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 10, pp. 1713–1727, 2008. pages 53, 55

[62] A. Srivastava, W. Mio, E. Klassen, and S. Joshi, "Geometric analysis of continuous, planar shapes," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 341–356, Springer, 2003. pages 53

[63] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on Lie algebra," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 728–735, IEEE, 2006. pages 54, 55

[64] A. Veeraraghavan, A. Srivastava, A. K. Roy-Chowdhury, and R. Chellappa, "Rate-invariant recognition of humans and their activities," *Image Processing, IEEE Transactions on*, vol. 18, no. 6, pp. 1326–1339, 2009. pages 54

[65] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 684–698, 2005. pages 54

[66] Y. M. Lui, J. R. Beveridge, and M. Kirby, "Canonical Stiefel quotient and its application to generic face recognition in illumination spaces," in *Biometrics: Theory, Applications, and Systems, 2009. IEEE 3rd International Conference on*, pp. 1–8, IEEE, 2009. pages 54, 55

[67] A. Srivastava and X. Liu, "Tools for application-driven linear dimension reduction," *Neurocomputing*, vol. 67, pp. 136–160, 2005. pages 54, 55

[68] A. Srivastava, I. Jermyn, and S. Joshi, "Riemannian analysis of probability density functions with applications in vision," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007. pages 55, 58

[69] T.-K. Kim, J. Kittler, and R. Cipolla, "Discriminative learning and recognition of image set classes using canonical correlations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1005–1018, 2007. pages 55

[70] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: a unifying view on subspace-based learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 376–383, ACM, 2008. pages 55

[71] O. Arandjelović, G. Shakhnarovich, J. Fisher, R. Cipolla, and T. Darrell, "Face recognition with image sets using manifold density divergence," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 581–588, IEEE, 2005. pages 55

[72] Y. M. Lui and J. R. Beveridge, "Grassmann registration manifolds for face recognition," in *Proceedings of the 10th European Conference on Computer Vision: Part II*, pp. 44–57, Springer, 2008. pages 55

[73] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic textures," in *Computer Vision, 2001. Eighth IEEE International Conference on*, vol. 2, pp. 439–446, IEEE, 2001. pages 55

[74] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, vol. 99, pp. 518–529, 1999. pages 55

[75] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, 1998. pages 58

[76] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2273–2286, 2011. pages 58

[77] T. Wu, P. Turaga, and R. Chellappa, "Age estimation and face verification across aging using landmarks," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 6, pp. 1780–1788, 2012. pages 59

[78] S. Taheri, P. Turaga, and R. Chellappa, "Towards view-invariant expression analysis using analytic shape manifolds," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 306–313, IEEE, 2011. pages 60

[79] R. Gopalan, S. Taheri, P. Turaga, and R. Chellappa, "A blur-robust descriptor with applications to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1220–1226, 2012. pages 60

[80] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, pp. 101–126, 2006. pages 60

[81] C. Wang and S. Mahadevan, "Manifold alignment without correspondence.," in *IJCAI*, vol. 2, p. 3, 2009. pages 60

[82] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, "Learning bounds for domain adaptation," in *Advances in neural information processing systems*, pp. 129–136, 2008. pages 60

[83] J. Blitzer, S. Kakade, and D. P. Foster, "Domain adaptation with coupled subspaces," in *International Conference on Artificial Intelligence and Statistics*, pp. 173–181, 2011. pages 60

[84] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 999–1006, IEEE, 2011. pages 60

[85] J. Zheng, M.-Y. Liu, R. Chellappa, and J. Phillips, "A Grassmann manifold-based domain adaptation approach," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 2095–2099, IEEE, 2012. pages 61

[86] L. Duan, D. Xu, and S.-F. Chang, "Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1338–1345, IEEE, 2012. pages 61

[87] I.-H. Jhuo, D. Liu, D. Lee, S.-F. Chang, *et al.*, "Robust visual domain adaptation with low-rank reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2168–2175, IEEE, 2012. pages 61

[88] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013. pages 63

[89] W. B. Johnson and J. Lindenstrauss, “Extensions of Lipschitz mappings into a Hilbert space,” *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984. pages 63

[90] T. Kowaliw, N. Bredeche, and R. Doursat, *Growing Adaptive Machines: Combining Development and Learning in Artificial Neural Networks*, vol. 557. Springer, 2014. pages 65