Lie group techniques for Neural Learning

*Edinburgh June 2004*

Elena Celledoni

SINTEF Applied Mathematics, IMF-NTNU
**Title:** Lie group techniques for Neural Learning

**Author:**

**Performing Organization:**
SINTEF Applied Mathematics, IMF-NTNU

**Performance Period:**
03 JAN 2005

**Abstract:**
See also ADM001749, Lie Group Methods And Control Theory Workshop Held on 28 June 2004 - 1 July 2004., The original document contains color images.
Outline

- Neural Networks
  - a short introduction
Outline

- Neural Networks
  - a short introduction
- Independent Component Analysis
  - Stochastic signal processing
  - Constraint optimization in ICA
Outline

Neural Networks
- a short introduction

Independent Component Analysis
- Stochastic signal processing
- Constraint optimization in ICA

Geometric Integration of Learning equations
- gradient flows and algorithms on manifolds
- MEC learning
- Newton methods
- diffusion algorithms
Neural Networks

Goals:
- Achieve efficient use of machines in tasks currently solved by humans
- Improve computing capabilities looking at the brain as a model
- Understand how the brain works

Applications
- Machine Learning
  1. How can a computer learn from a set of examples?
  2. Constraint optimization
  3. Pattern recognition, classification
  4. Associative memory
- Cognitive science
  1. Models for high level reasoning: language, problem solving
  2. Models for low level reasoning: vision, speech recognition, speech generation
- Neurobiology: find models for how the brain works
List of fields where Neural Networks are used

- Signal processing
- Control
- Robotics (navigation, vision)
- Medicine
- Business and Finance
- Data Compression
The brain as an Information Processing System

- Massively parallel: 10 billion neurons, 10000 synapses per neuron
- Slow hardware: neurons operate at about 100 Hz, while conventional CPUs execute several hundred million machine level operations per second
The brain as an Information Processing System

- Massively parallel: 10 billion neurons, 10,000 synapses per neuron
- Slow hardware: neurons operate at about 100 Hz, while conventional CPUs execute several hundred million machine level operations per second
The brain as an Information Processing System

- Massively parallel: 10 billion neurons, 10000 synapses per neuron
- Slow hardware: neurons operate at about 100 Hz, while conventional CPUs execute several hundred million machine level operations per second

![Motor Neuron Diagram](https://via.placeholder.com/150.png)
The brain as an Information Processing System

- Massively parallel: 10 billion neurons, 10000 synapses per neuron
- Slow hardware: neurons operate at about 100 Hz, while conventional CPUs execute several hundred million machine level operations per second

**Synapse**: transmission of a signal between neurons via a neurotransmitter. **Learning** corresponds to alteration of the strength of the connection between neurons.
A simple model for a neuron

Each node (neuron) receives signal inputs from $n$ neighbor nodes.

$$y_i = f\left( \sum_j w_{i,j} y_j \right)$$

The weighted sum $\sum_j w_{i,j} y_j$ is called the net input. $f$ is the activation function, if $f$ is the identity we have a linear unit. $y_i$ is the output signal.

$$y_i = f(\text{net}_i)$$
Linear Neural Networks

Several inputs one output

http://www.willamette.edu/ gorr
The cocktail-party problem
Suppose you record two time signals $x_1(t)$ and $x_2(t)$ form two different positions in a room. Each recorded signal is a linear mixture of the voices of two speakers which emit two sources $s_1(t)$ and $s_2(t)$

$$
x_1(t) = a_{1,1}s_1(t) + a_{1,2}s_2(t)
$$

$$
x_2(t) = a_{2,1}s_1(t) + a_{2,2}s_2(t)
$$

Estimate $s_1(t)$ and $s_2(t)$ from the sole knowledge of $x_1(t)$ and $x_2(t)$
Independent Component Analysis

The cocktail-party problem
Suppose you record two time signals $x_1(t)$ and $x_2(t)$ form two different positions in a room. Each recorded signal is a linear mixture of the voices of two speakers which emit two sources $s_1(t)$ and $s_2(t)$

$$x_1(t) = a_{1,1}s_1(t) + a_{1,2}s_2(t)$$
$$x_2(t) = a_{2,1}s_1(t) + a_{2,2}s_2(t)$$

Estimate $s_1(t)$ and $s_2(t)$ from the sole knowledge of $x_1(t)$ and $x_2(t)$
Assume the sources and the recorded signals are samples of the zero-mean random variables $x_1, x_2$ (mixtures) and $s_1, s_2$ (independent components).
Assumption $s_1(t)$ and $s_2(t)$ are statistically independent
Independent Component Analysis

The cocktail-party problem
Suppose you record two time signals $x_1(t)$ and $x_2(t)$ form two different positions in a room. Each recorded signal is a linear mixture of the voices of two speakers which emit two sources $s_1(t)$ and $s_2(t)$

$$x_1(t) = a_{1,1}s_1(t) + a_{1,2}s_2(t)$$
$$x_2(t) = a_{2,1}s_1(t) + a_{2,2}s_2(t)$$

Estimate $s_1(t)$ and $s_2(t)$ from the sole knowledge of $x_1(t)$ and $x_2(t)$
Assume the sources and the recorded signals are samples of the zero-mean random variables $x_1, x_2$, (mixtures) and $s_1, s_2$ (independent components).

Assumption $s_1(t)$ and $s_2(t)$ are statistically independent

Unknown source signals $s(t) = [s_1(t), \ldots, s_n(t)]^T$
Given the output signals $x(t) = As(t)$, $x(t) = [x_1(t), \ldots, x_k(t)]^T$
Unknown mixing matrix $A_{p \times n}$
Independent Component Analysis

The cocktail-party problem
Suppose you record two time signals \( x_1(t) \) and \( x_2(t) \) form two different positions in a room. Each recorded signal is a linear mixture of the voices of two speakers which emit two sources \( s_1(t) \) and \( s_2(t) \)

\[
\begin{align*}
  x_1(t) & = a_{1,1}s_1(t) + a_{1,2}s_2(t) \\
  x_2(t) & = a_{2,1}s_1(t) + a_{2,2}s_2(t)
\end{align*}
\]

Estimate \( s_1(t) \) and \( s_2(t) \) from the sole knowledge of \( x_1(t) \) and \( x_2(t) \)
Assume the sources and the recorded signals are samples of the zero-mean random variables \( x_1, x_2, \) (mixtures) and \( s_1, s_2 \) (independent components).
Assumption \( s_1(t) \) and \( s_2(t) \) are statistically independent

Unknown source signals \( s(t) = [s_1(t), \ldots, s_n(t)]^T \)
Given the output signals \( x(t) = As(t), \; x(t) = [x_1(t), \ldots, x_k(t)]^T \)
Unknown mixing matrix \( A \; p \times n \)
Find approximations \( y \) of \( s \) by constructing a de-mixing matrix \( W \) and

\[ y = Wx. \]
Principles for reconstruction

The sum of two independent random variables usually has distribution closer to Gaussian than the two original random variables. (Central Limit Theorem)

\[ x = As \]

Find

\[ y = Wx \approx s \]

maximizing nongaussianity.

A measure of nongaussianity is kurtosis,

\[ \text{kurt}(y) = E\{y^4\} - 3(E\{y^2\})^2, \]

with \( y \) of unit variance \( \text{kurt}(y) = E\{y^4\} - 3. \)
Withening

Preprocessing of the output signals $\mathbf{x} \rightarrow \tilde{\mathbf{x}}$ such that the components of $\tilde{\mathbf{x}}$ are uncorrelated with variances equal to 1

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\} = \mathbf{I}.$$
Preprocessing of the output signals $x \rightarrow \tilde{x}$ such that the components of $\tilde{x}$ are uncorrelated with variances equal to 1

$$E\{\tilde{x}\tilde{x}^T\} = I.$$ 

Use for example $E\{xx^T\} = VDV^T$ and

$$\tilde{x} = VD^{-1/2}V^Tx \quad \Rightarrow \quad E\{\tilde{x}\tilde{x}^T\} = I$$

and $\tilde{x} = VD^{-1/2}V^TAs = \tilde{A}s$, then

$$E\{\tilde{x}\tilde{x}^T\} = \tilde{A}E\{ss^T\}\tilde{A}^T = \tilde{A}\tilde{A}^T = I.$$

---

Lie group techniques for Neural Learning – p.10/24
**Reconstruction**

Reconstruction of \( s \). We can look for a de-mixing matrix \( W \) s.t. \( W^T W = I_p \) and \( y(t) = W x(t) \) solving

\[
\min_{W^T W = I_p} D(W)
\]

\( D(W) \) is the dependency among the components.

**A. Hyvärinen and E. Oja** Independent component analysis: A tutorial, *Neural Networks*. 
Let $\mathcal{M}$ be a Riemannian manifold with metric $m(\cdot, \cdot)$, given $\phi : \mathcal{M} \to \mathbb{R}$ a smooth function the equilibria of

$$\dot{x}(t) = -\nabla \phi(x(t))$$

are the critical points of $\phi$.

$\nabla \phi$ is such that:

- $\nabla \phi(x) \in T_x \mathcal{M}$
- $\phi'(x)(v) = m(\nabla \phi(x), v)$ for all $v \in T_x \mathcal{M}$
Optimizing via gradient flows

Let $\mathcal{M}$ be a Riemannian manifold with metric $m(\cdot, \cdot)$, given $\phi : \mathcal{M} \to \mathbb{R}$ a smooth function the equilibria of

$$\dot{x}(t) = -\nabla \phi(x(t))$$

are the critical points of $\phi$.

$\nabla \phi$ is such that:

- $\nabla \phi(x) \in T_x \mathcal{M}$
- $\phi'(x)(v) = m(\nabla \phi(x), v)$ for all $v \in T_x \mathcal{M}$

U. Helmke and J.B. Moore, Optimization and Dynamical Systems, Springer-Verlag 1994


S.I. Amari, Natural Gradient Works Efficiently in Learning, Neural Computation, 1998

Y. Nishimori, Learning algorithm for ICA by geodesic flows on orthogonal
Proc. IJCNN 99
Consider $S^* = \{[2m_i, w_i]\}$ rigid system of $n$ masses $m_i$ with positions $w_i$ (unitary distance form the origin on mutually orthogonal axis). The masses move in a viscous liquid. No translation.

$$\dot{W} = HW, \quad P = -\mu HW$$

$$\dot{H} = \frac{1}{4}[(F + P)W^T - W(F + P)^T]$$

$\mu$ viscosity parameter

$P$ matrix of the viscosity resistance

$W$ matrix of the positions

$F$ active forces

$H$ angular velocity matrix

$W$ is on $O(n)$ or on the Stiefel manifold
The mechanical system seen as an adapting rule for neural layers with weight matrix $W$. The forces

$$F := -\frac{\partial U}{\partial W}$$

with $U$ a potential energy function. The equilibria of the mechanical systems $S^*$ are at the local minima of $U$. Take $U = J_C$ cost function to be minimized, or $U = -J_O$ objective function to be maximized, $W(t), \ t \to \infty$ approaches the solution of the optimization problem.

S. Fiori, 'Mechanical' Neural Learning for Blind Source Separation, Electronics Letters, 1999
Reformulation of the equations when $n \ll p$

Using the Lie algebra

\[ \dot{W} = HW, \quad P = -\mu HW \]

\[ \dot{H} = \frac{1}{4} [[F + P]W^T - W(F + P)^T] \]

Using the tangent space

\[ \dot{W} = V \]

\[ \dot{V} = g(V, W) \]

where

\[ V = (GW^T - WG^T)W, \quad G = V - W(W^TV/2 + S) \]

and

\[ g(V, W) = (LW^T - WL^T)W + (GW^T - WG^T)V, \quad L = \dot{G} - GW^T G \]
The learning algorithm

\[
\begin{align*}
V_{n+1} &= V_n + hg(V_n, W_n) \\
G_n &= V_n - 1/2W_n(W_n^T V_n) \\
W_{n+1} &= \exp(h(G_n W_n^T - W_n G_n^T)) W_n
\end{align*}
\]

with $W_0 = I_{n \times p}$ and $V_0 = 0_{n \times p}$.

Here

\[
\exp(h(G_n W_n^T - W_n G_n^T)) = [W_n, W_n^\perp] \exp\left(\begin{bmatrix} C - C^T & -R^T \\ R & O \end{bmatrix}\right) [W_n, W_n^\perp]^T
\]

and $C = W_n^T G_n$, and $G_n - W_n C = W_n^\perp R$. We exponentiate matrices of dimension $2p \times 2p$ instead of $n \times n$.

Computational cost

For the exponential $9np^2 + np + \mathcal{O}(p^3)$ flops. For the overall geodesic learning algorithm (one step) $21np^2 + 6np + \mathcal{O}(p^3)$ flops.
Computational gain

Computing the largest eigenvalue of an $n \times n$ matrix $A$ (discretization of the 1-D Laplacian with finite differences).

The potential energy function is $U(w) = -w^T A w$, $p = 1$.

<table>
<thead>
<tr>
<th>SIZE OF $A$</th>
<th>New MEC</th>
<th>Old MEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 32$</td>
<td>$4.72 \times 10^5$</td>
<td>$1.31 \times 10^6$</td>
</tr>
<tr>
<td>$n = 64$</td>
<td>$1.82 \times 10^6$</td>
<td>$5.25 \times 10^6$</td>
</tr>
<tr>
<td>$n = 128$</td>
<td>$7.39 \times 10^6$</td>
<td>$2.10 \times 10^7$</td>
</tr>
<tr>
<td>$n = 256$</td>
<td>$2.49 \times 10^7$</td>
<td>$8.39 \times 10^7$</td>
</tr>
</tbody>
</table>

Floating point operations per iteration versus the size of the problem.
Experiments Blind source separation

Original images, with their kurtosis and their linear mixtures

Kurtosis = 4.981

Kurtosis = 4.699

Kurtosis = 2.157

Kurtosis = 2.871

Kurtosis = 2.953

Kurtosis = 1.329
Source separation

The force \( F(W) = -kE_x[x(x^TW)^3] \).

Recovered image, and potential energy

![Recovered image](image1.png)

![Algorithm: MEC3](image2.png)

![Potential energy](image3.png)

![Kinetic energy](image4.png)
References


Future work

- On the orthogonal group consider quasi-geodesic paths using low-rank splittings
- Other manifolds occur in the case of multi-layer neural networks: Flag manifolds
- comparison with Newton methods
Newton methods, Mahony’s approach

For finding minima or maxima of $\phi : G \to \mathbb{R}$, and $G$ is a Lie group,

- choose an inner product $\langle \cdot, \cdot \rangle$ on the Lie algebra $\mathfrak{g}$ and take an orthonormal basis in the Lie algebra $X_1, \ldots, X_d$, and $\tilde{X}_1, \ldots, \tilde{X}_d$
- the right invariant vector fields

$$\text{grad}\phi = \sum_{i=1}^{d} m(\tilde{X}_i, \text{grad}\phi)\tilde{X}_i = \sum_{i=1}^{d}(\tilde{X}_i\phi)\tilde{X}_i$$

$(m(\tilde{X}, \tilde{Y}) = \langle X, Y \rangle$ (right invariant group metric))

- if $\exp(X)\sigma$ is a critical point of $\phi$, the vector field $\tilde{X}$ satisfies,

$$\text{grad}\phi(\sigma) + \text{grad}(\tilde{X}\phi)(\sigma) = 0$$

Newton methods, Mahony’s approach

For finding minima or maxima of $\phi : G \to \mathbb{R}$, and $G$ is a Lie group,

- choose an inner product $< \cdot, \cdot >$ on the Lie algebra $\mathfrak{g}$ and take an orthonormal basis in the Lie algebra $X_1, \ldots, X_d$, and $\tilde{X}_1, \ldots, \tilde{X}_d$

the right invariant vector fields

\[
\text{grad} \phi = \sum_{i=1}^{d} m(\tilde{X}_i, \text{grad} \phi)\tilde{X}_i = \sum_{i=1}^{d} (\tilde{X}_i \phi)\tilde{X}_i
\]

$$(m(\tilde{X}, \tilde{Y}) = < X, Y > \text{ (right invariant group metric)})$$

- Find $X^k$ such that $\tilde{X}^k$ solves

\[
\text{grad} \phi(\sigma_k) + \text{grad}(\tilde{X}^k \phi)(\sigma_k) = 0
\]

set $\sigma_{k+1} = \exp(X^k)\sigma_k$, $k \leftarrow k + 1$ and continue, (equivalent to Lie Euler for $\dot{\sigma} = X^k \sigma$, $\sigma(0) = \sigma^k$)

Newton methods, other approaches


Diffusion-type algorithms

Perturbation of the standard Reimannian gradient to obtain a randomized gradient. Diffusion-type gradient on $\mathfrak{so}(n)$

\[
V_{\text{diff}}(t) = V(t) + \sqrt{2\theta} \sum_{k=1}^{n(n-1)/2} X_k \frac{d\mathcal{W}_k}{dt}
\]

$V(t)$ deterministic gradient, $X_k$ is a basis of the Lie algebra $\mathfrak{so}(n)$ orthogonal with respect to the chosen metric, and $\mathcal{W}_k(t)$ are real-valued, independent standard Wiener processes i.e. a random variable $\mathcal{W}$ continuous in $t$ s.t.

- $\mathcal{W}(0) = 0$ with probability 1
- for $0 \leq \tau < t$ the random variable $\mathcal{W}(t) - \mathcal{W}(\tau)$ is normally distributed with mean zero and variance $t - \tau$
- for $0 \leq \tau < t < u < v$, the increments $\mathcal{W}(t) - \mathcal{W}(\tau)$ and $\mathcal{W}(v) - \mathcal{W}(u)$ are statistically independent
Diffusion-type algorithms

Perturbation of the standard Reimannian gradient to obtain a randomized gradient. Diffusion-type gradient on $\mathfrak{SO}(n)$

$$V_{\text{diff}}(t) = V(t) + \sqrt{2\theta} \sum_{k=1}^{n(n-1)/2} X_k \frac{dW_k}{dt}$$

$V(t)$ deterministic gradient, $X_k$ is a basis of the Lie algebra $\mathfrak{SO}(n)$ orthogonal with respect to the chosen metric, and $W_k(t)$ are real-valued, independent standard Wiener processes. The learning differential equation is

$$\frac{dW}{dt} = -V_{\text{diff}}(t)W$$

*Langevin-type stochastic differential equation* on the orthogonal group

Conclusion

- Integration of learning equations and gradient flows is achieved with simple first order explicit Lie group integrators.
- Efficient approximation of the matrix exponential from a Lie algebra to a Lie group or the computation of geodesics is crucial.
- Development of methods based on other coordinate maps then the exponential, and quasi-geodesic strategies.
- Geometric integration of stochastic differential equations.