UNIUNEA EUROPEANĂ

Instrumente Structurale 2014-2020

“Metode de optimizare Riemanniene pentru învățare profundă”
Proiect cofinanțat din Fondul European de Dezvoltare Regională prin
Programul Operațional Competitivitate 2014-2020

On the Geometry of Neural Networks

Luigi Malagò

Romanian Institute of Science and Technology

Algebraic Statistics Workshop, Genova January 23, 2017

Romanian Institute of Science and Technology

A private no-profit research institute founded in 2009 in Cluj-Napoca, RO

Currently employs ~20 researchers

Performs research in

- Computational and Experimental Neuroscience
- Computational Intelligence
- Machine Learning and Optimization
- Dynamical Systems

RIST is undergoing a phase of significant and sustained growth supported by EU and RO structural funds (up to 4M Euro)

15 open positions by end 2017: calls at www.rist.ro

Outline

- The mathematical model of the Perceptron
- Training of a Neural Network
- The Information Geometry of statistical models
- The natural gradient
- Current and future lines of research

"One geometry cannot be more true than another; it can only be more convenient". Henri Poincaré, Science and Hypothesis, 1902.

The Mathematical Model of the Perceptron

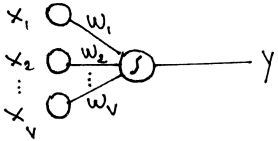
An artificial Neural Network is a computational model made of interconnected units, called neurons, which process input data and generates outputs, similarly to a non linear function

The Mathematical Model of the Perceptron

An artificial Neural Network is a computational model made of interconnected units, called neurons, which process input data and generates outputs, similarly to a non linear function

The simplest network is made of a single perceptron which takes as input a linear combination of the input $\boldsymbol{x} = (x_1, \ldots, x_V)$ and generates an output y according to the model

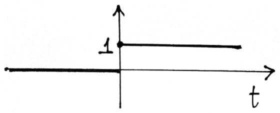
$$y = \varphi\left(\sum_{i=1}^{V} w_i x_i + h\right) ,$$

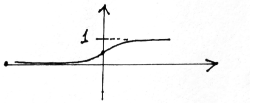


where $\boldsymbol{w} = (x_1, \ldots, x_N) \in \mathbb{R}^V$ are the connection weights, and $\varphi(t)$ is the activation function, defined over $\mathbb{R}$

Geometrical Interpretation of the Perceptron

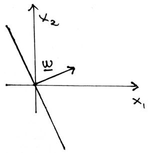
Activations functions are usually monotonic and bounded





$\text{threshold}(t) = \mathbf{1}_{t \geq 0}$ $\qquad$ $\text{sig}(t) = \dfrac{1}{1 + e^{-t}}$

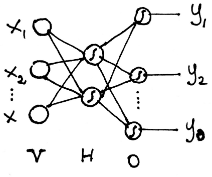
In case of $\text{threshold}(t) = \mathbf{1}_{t \geq 0}$, the weight vector is the normal vector to the decision hyperplane



i.e., a single perceptron can only learn linearly separable datasets

The Three-Layer Perceptron

Multiple units can be combined to generate complex behaviors



For instance, consider a hidden layer of H units

$$y_k = \varphi_k \left(\sum_{j=1}^{H} v_{kj} \varphi_j \left(\sum_{i=1}^{V} w_{ji} x_i \right) \right) = f(\boldsymbol{x}; \boldsymbol{\theta}) \ ,$$

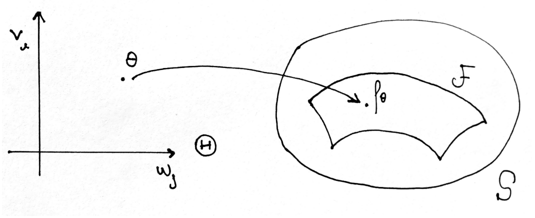
where $\boldsymbol{\theta} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_H, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_O)$ is the vector of parameters which describes the network

The Parameter Space and the Functional Space

A multilayer Neural Network defines a non-linear function

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$$

from the space $\mathcal{X} \in \mathbb{R}^V$ to $\mathcal{Y} \in \mathbb{R}^O$, parametrized by $\boldsymbol{\theta} \in \Theta$



Consider the infinite-dimensional functional space $\mathcal{S}$ of all functions from $\mathbb{R}^V$ to $\mathbb{R}^O$, the weights $\boldsymbol{\theta}$ define a finite-dimensional manifold $\mathcal{F} \ni f$, identified by the topology of the network

Training of a Neural Network

A Neural Network is a parametric model, where the connection weights $\boldsymbol{w}$ correspond to the parameters of the model

Learning is achieved by adjusting the parameters to realize the function between $\boldsymbol{x}$ and $\boldsymbol{y}$ expressed by the training set $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \dots, (\boldsymbol{x}^{(M)}, \boldsymbol{y}^{(M)})$

Given a loss function ℓ which measures how different is the prediction $f(\boldsymbol{x}; \boldsymbol{\theta})$ from the true outcome y, the training can be performed by minimizing the risk

$$R(\boldsymbol{\theta}) = \mathbb{E}\left[\ell(\boldsymbol{y}, f(\boldsymbol{x}; \boldsymbol{\theta}))\right]$$

Since $p(\boldsymbol{x}, \boldsymbol{y})$ is unknown in general and can only be estimated, the risk is replaced by the empirical risk of the training set

$$R_{\mathsf{err}}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} \ell(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}))$$

A Probabilistic Model for Neural Networks

The training set is usually supposed to be noisy, i.e., to express a probabilistic relationship

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}) + \boldsymbol{\epsilon} \ ,$$

where $\boldsymbol{\epsilon}$ represents some independent random noise

Thus, it becomes natural to define the network as a probabilistic model which implements a conditional probability density

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = r(\boldsymbol{y}|f(\boldsymbol{x}; \boldsymbol{\theta}))$$

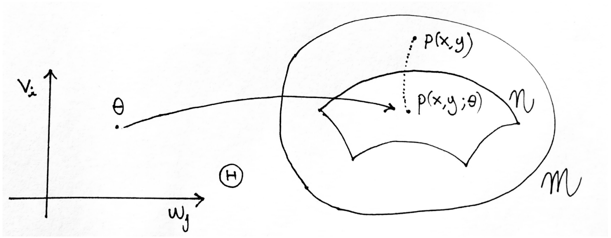
By assuming a true conditional probability density, when the input is generated by a probability density $q(\boldsymbol{x})$, the joint distribution is

$$p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) q(\boldsymbol{x})$$

The Statistical Manifold

The set of all joint probability distributions $p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$ parametrized by $\boldsymbol{w}$ identifies a finite-dimensional manifold $\mathcal{N}$ in the infinite dimensional space of integrable densities $\mathcal{M}$

The parameters $\boldsymbol{\theta}$ act as a coordinate system over $\mathcal{N}$



Training Neural Networks

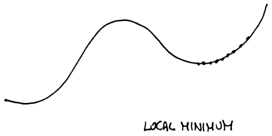
The optimum of the empirical risk is usually computed iteratively, by gradient descent

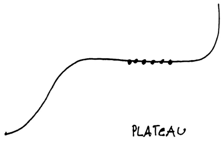
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda \nabla R_{\mathsf{emp}}(\boldsymbol{\theta}_t)$$

where $\lambda > 0$ is the learning rate, where ∇ denotes the vector of partial derivatives with respect to $\boldsymbol{\theta}$

Gradient descent can be implemented as batch/mini-batch learning or as online learning

The gradient may converge to local minima and slow down in presence of saddle points or plateaux





Single-Layer Backpropagation

Consider a perceptron, let the loss function be the square loss,

$$\ell\left(\boldsymbol{y}^{(j)}, f(\boldsymbol{x}^{(j)}; \boldsymbol{\theta})\right) = \frac{1}{2}\left(\boldsymbol{y}^{(j)} - \varphi\left(\sum_{i=1}^{V} w_i x_i^{(j)}\right)\right)^2$$

By evaluating the gradient w.r.t. $\boldsymbol{\theta} = (w_1, \dots, w_V)$ we obtain

$$\nabla R_{\mathsf{emp}}(\boldsymbol{\theta}_t) = -\left(\boldsymbol{y}^{(t)} - \varphi\left(\sum_{i=1}^{V} w_i x_i^{(t)}\right)\right)\varphi'\left(\sum_{i=1}^{V} w_i x_i^{(t)}\right)\boldsymbol{x}$$

In case $\varphi(s) = \mathsf{sig}(s)$, then $\varphi(s)' = \frac{\partial}{\partial s}\varphi(s) = \mathsf{sig}(s)(1 - \mathsf{sig}(s))$

For multilayer networks, a formula for the gradients can be efficiently obtained by backpropagation, i.e., by the chain rule

$$(g \circ h)'(s) = g(h(s))' = g(h(s))'\ h'(s)$$

Multi-Layer Backpropagation

Let H_l the number of neurons for layer l, with $l = 1, \dots, L$ and $H_1 = V$ and $H_L = O$

The multi-layer backpropagation algorithm becomes

1 . Feedforward pass: computation of $\boldsymbol{h}^{(1)}, \dots, \boldsymbol{h}^{(L)}$

2 . For the output layer compute, for $i = 1, \dots, O$

$$\delta_i^{(L)} = \left(\boldsymbol{h}^{(L)} - \boldsymbol{y}\right) \varphi' \left(\sum_{j=1}^{H_{L-1}} w_{ij}^{(L-1)} h_j^{(L-1)} \right)$$

3 . Perform a backward pass for $l = L - 1, \dots, 2$ and $i = 1, \dots, O$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{H_{l+1}} w_{ij}^{(l)} \delta_j^{(l+1)} \right) \varphi' \left(\sum_{j=1}^{H_{l-1}} w_{ij}^{(l-1)} h_j^{(l-1)} \right)$$

4 . Compute $\nabla_{w_{ij}^{(l)}} R_{\text{emp}}(\boldsymbol{\theta}_t) = h_j^{(l)} \delta_j^{(l+1)}$

Gradient Descent Over Statistical Models

A natural approach to optimize a function $F(\boldsymbol{\theta}) : \mathcal{N} \to \mathbb{R}$ is given by a naive gradient descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda \nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)$$

- ∇ is shorthand for $\frac{\partial}{\partial \boldsymbol{\theta}}$
- $\lambda > 0$ step size

However a series of issues may arise:

- dependence on the parameterization
- slow convergence over plateaux
- (target distribution may not be a critical point)
- (gradient may point outside of the domain of Θ)

Many of these issues are consequence of the choice of a Euclidean geometry for $\mathcal{M}$

Information Geometry

Euclidean geometry is not the most convenient geometry for statistical models, as (probably) first remarked by Hotelling (1930) and Rao (1945)

Information Geometry

Euclidean geometry is not the most convenient geometry for statistical models, as (probably) first remarked by Hotelling (1930) and Rao (1945)

Information Geometry follows a different geometric approach, given by the representation of statistical models as Riemannian statistical manifolds, endowed with the Fisher information metric

Besides the Riemannian one, Information Geometry also studies other non-Euclidean geometries for statistical models, based on the notion of dual affine manifolds

Information Geometry

Euclidean geometry is not the most convenient geometry for statistical models, as (probably) first remarked by Hotelling (1930) and Rao (1945)

Information Geometry follows a different geometric approach, given by the representation of statistical models as Riemannian statistical manifolds, endowed with the Fisher information metric

Besides the Riemannian one, Information Geometry also studies other non-Euclidean geometries for statistical models, based on the notion of dual affine manifolds

The research in Information Geometry has started in the 80's, with the pioneer work of Amari (1982,1985), Barndorff-Nielsen (1978), Cencov (1982), Lauritzen (1987), Pistone and Sempi (1995) and colleagues

Standard References

Three monographs by Amari, who is considered the founder of Information Geometry

- S.-I. Amari. *Differential-geometrical methods in statistics.* Lecture notes in statistics, Springer-Verlag, Berlin, 1985.
- S.-I. Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. AMS, Oxford University Press, 2000. Translated from the 1993 Japanese original by Daishi Harada.
- S.-I. Amari. *Information Geometry and Its Applications.* Springer, 2016.

Other standard references are

- M. Murray and J. Rice. *Differential geometry and statistics.* Monographs on Statistics and Applied Probability 48. Chapman and Hall, 1993.
- R. E. Kass and P. W. Vos. *Geometrical Foundations of Asymptotic Inference.* Series in Probability and Statistics, Wiley, 1997.

Geometry Derived by the KL Divergence

An alternative geometry for a statistical model can be defined by measuring infinitesimal distances using the Kullback-Leibler divergence

$$D_{\mathsf{KL}}(p||q) = \int_\Omega p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$$

Geometry Derived by the KL Divergence

An alternative geometry for a statistical model can be defined by measuring infinitesimal distances using the Kullback-Leibler divergence

$$D_{\mathsf{KL}}(p\|q) = \int_\Omega p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$$

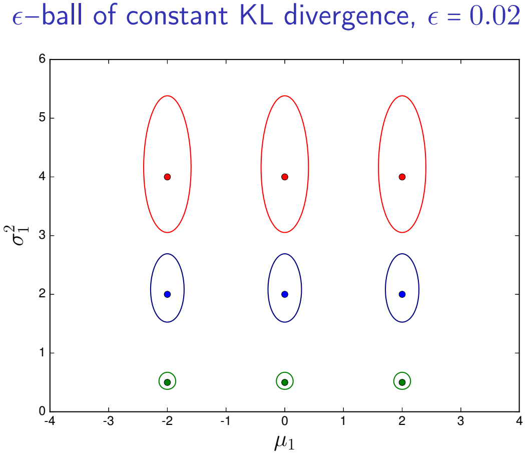
It can be proved that such choice determines a Riemannian structure for $\mathcal{M}$, where the Fisher Information matrix plays the role of metric tensor

The direction of steepest ascent $\Delta\boldsymbol{\theta}$ in a Euclidean space for $F_{\boldsymbol{\theta}}$ can then be evaluated by minimizing $F_{\boldsymbol{\theta}}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})$ with $\|\Delta\boldsymbol{\theta}\| = 1$

Amari replaces this contraint with the KL divergence

$$\begin{aligned} &\arg\min_{\Delta\boldsymbol{\theta}} \; F_{\boldsymbol{\theta}}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \\ &\text{s.t. } D_{\mathsf{KL}}(p_{\boldsymbol{\theta}} \| p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}) = \epsilon \end{aligned}$$

Example: The Gaussian Distribution



Let $p_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$, and $p_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$,

$$D_{\mathsf{KL}}(p_0 \| p_1) = \log \frac{\sigma_1}{\sigma_0} + \frac{\sigma_0^2 + (\mu_0 - \mu_1)^2}{2\sigma_1^2} - \frac{1}{2}$$

Amari's Natural Gradient (1998) 1/2

By taking the second-order Taylor approximation of the KL divergence in $\boldsymbol{\theta}$ we get

$$\begin{aligned} D_{\mathsf{KL}}(p_{\boldsymbol{\theta}} \| p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}) &= \mathbb{E}_{\boldsymbol{\theta}}[\log p_{\boldsymbol{\theta}}] - \mathbb{E}_{\boldsymbol{\theta}}[\log p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}] \\ &\approx \mathbb{E}_{\boldsymbol{\theta}}[\log p_{\boldsymbol{\theta}}] - \mathbb{E}_{\boldsymbol{\theta}}[\log p_{\boldsymbol{\theta}}] - \mathbb{E}_{\boldsymbol{\theta}}[\nabla \log p_{\boldsymbol{\theta}}]^{\mathrm{T}} \boldsymbol{\Delta\theta} + \\ &\quad - \frac{1}{2} \Delta\boldsymbol{\theta}^{\mathrm{T}} \mathbb{E}_{\boldsymbol{\theta}} \left[\nabla^2 \log p_{\boldsymbol{\theta}} \right] \Delta\boldsymbol{\theta} \\ &= \frac{1}{2} \Delta\boldsymbol{\theta}^{\mathrm{T}} I(\boldsymbol{\theta}) \Delta\boldsymbol{\theta}, \end{aligned}$$

where $I_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ is the Fisher Information matrix

$$\begin{aligned} I_{\boldsymbol{\theta}}(\boldsymbol{\theta}) &= -\mathbb{E}_{\boldsymbol{\theta}} \left[\nabla^2 \log p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}} \right] \\ &= \mathbb{E}_{\boldsymbol{\theta}} \left[\nabla \log p(\boldsymbol{\theta}) \nabla \log p(\boldsymbol{\theta})^{\mathrm{T}} \right] \end{aligned}$$

Amari's Natural Gradient (1998) 2/2

We proceed by taking the first-order approximation of $F_{\boldsymbol{\theta}}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})$

$$\begin{gathered}\arg\min_{\Delta\boldsymbol{\theta}} \; F_{\boldsymbol{\theta}}(\boldsymbol{\theta}) + \nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta})^{\mathrm{T}} \Delta\boldsymbol{\theta} \\ \text{s.t. } \frac{1}{2}\Delta\boldsymbol{\theta}^{\mathrm{T}} I_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} = \epsilon\end{gathered}$$

We apply the Lagrangian method, and solve for $\Delta\boldsymbol{\theta}$

$$\begin{aligned}&\nabla_{\Delta\boldsymbol{\theta}} \left(F_{\boldsymbol{\theta}}(\boldsymbol{\theta}) + \nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta})^{\mathrm{T}} \Delta\boldsymbol{\theta} - \lambda \frac{1}{2} \Delta\boldsymbol{\theta}^{\mathrm{T}} I_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} \right) = 0 \\ &\nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta}) - \lambda I_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} = 0 \\ &\Delta\boldsymbol{\theta} = \lambda I_{\boldsymbol{\theta}}(\boldsymbol{\theta})^{-1} \nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta})\end{aligned}$$

Such derivations lead to the natural gradient (Amari, 1998)

$$\widetilde{\nabla} F_{\boldsymbol{\theta}}(\boldsymbol{\theta}) = I_{\boldsymbol{\theta}}(\boldsymbol{\theta})^{-1} \nabla F_{\boldsymbol{\theta}}(\boldsymbol{\theta})$$

Training Neural Networks by Natural Gradient

Neural Networks can be traing by natural gradient

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda I(\boldsymbol{\theta}_t)^{-1} \nabla R_{\mathsf{emp}}(\boldsymbol{\theta}_t)$$

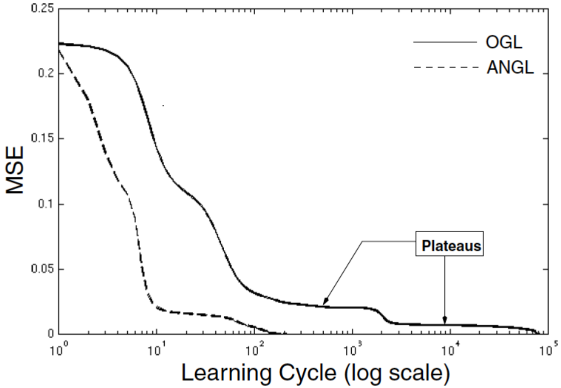
where $\lambda > 0$ is the learning rate, ∇ denotes the vector of partial derivatives with respect to $\boldsymbol{\theta}$ and I is the Fisher Information matrix

Natural gradient has better convergence properties and is less likely to get stuck in plateaux

However, natural gradient requires to solve a linear system at each iteration, which poses computational issues for large networks

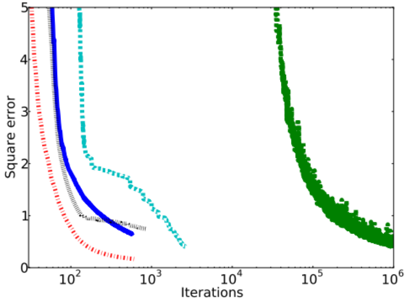
The research on natural gradient is mainly focused on finding efficient approximations for the Fisher Information matrix

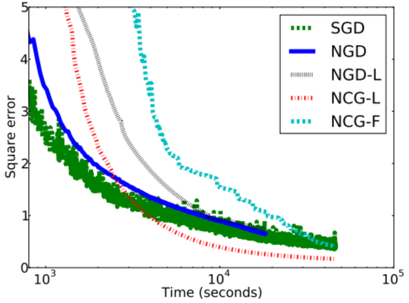
Experimental Results 1/2



Park, Amari, and Fukumizu (2000). IRIS flower classification dataset, 150 training points.

Experimental Results 1/2





Pascanu and Bengio (2014). Curves dataset, 6 layers deep auto encoders. 20k training samples 784 dimensions.

Natural Gradient in Machine Learning

Natural gradient (Amari, 1998) methods are becoming constantly popular in machine learning, e.g.,

- Training of Neural Networks (Amari, 1997) and recently Deep Learning (Ollivier et. al., 2014; Pascanu and Bengio, 2014; Desjardins et. al., 2014; Martens et. al., 2015; Ollivier, 2015)
- Reinforcement learning and Markov Decision Processes (Kakade, 2001; Peters and Schaal, 2008)
- Stochastic Relaxation and Evolutionary Optimization (i.e., black-box derivative-free methods) (Wiestra et. al., 2008-14; Malagò et. al., 2011; Ollivier et. al., 2011; Akimoto et. al., 2012)
- Bayesian variational inference (Honkela et. al., 2008)
- Bayesian optimization
- and many others

Take Home Message and Current Research

- The geometry of statistical models is much richer than one could expect
- The theory is beautiful and the number of possible applications of natural gradient methods is large in machine learning
- Natural gradient shows superior performance compared to the vanilla gradient
- The efficient computation of the natural gradient is probably the biggest issue, unless some special cases

Take Home Message and Current Research

- The geometry of statistical models is much richer than one could expect
- The theory is beautiful and the number of possible applications of natural gradient methods is large in machine learning
- Natural gradient shows superior performance compared to the vanilla gradient
- The efficient computation of the natural gradient is probably the biggest issue, unless some special cases

- Currently a lot of research is focused on approximations and decompositions for large dimensional settings
- An emerging line of research is the design of second-order methods in for the optimization over statistical manifolds

Open Postdoc Positions at RIST

RIST has multiple positions inq Information Geometry, Riemannian Optimization and Deep Learning, funded by a 4-years EU starting grant "DeepRiemann - Riemannian Optimization Methods for Deep Learning"

I will be happy to meet you soon in Cluj-Napoca ;-)