Gradient Learning in A Non-Euclidean Space

Tsung-Wei Chiang (Jimmy)

Natinal Taiwan University

d98942029@ntu.edu.tw

September 11, 2018

Motivation:
Parameter Space from Euclidean to Non-Euclidean

- Previous Assumption: Learning takes places in a parameter space which is Euclidean.
- New Assumption: Learning takes places in a parameter space which is Non-Euclidean.
- What if the parameter space is Non-Euclidean [1, 2, 3], e.g., Reimannian?
- (Ordinary) Gradient Descent Methods (including SGD) in deep learning
- Natural Gradient Descent Methods

Overview

- Example: a regression problem,
- input signal: $\mathbf{x}$, generate randomly
- desired response: $f(\mathbf{x})$
- teacher signal: y
- random noise: $\epsilon \sim p_\epsilon(\epsilon)$, e.g., Gaussian
- noisy version of the desired output:

$$y = f(\mathbf{x}) + \epsilon \tag{1}$$

- unknown joint probability distribution: $p(\mathbf{x})$
- The task of a learning machine is to estimate the desired output mapping $f(x)$ by using the available examples of input-output pairs $D = \{(\mathbf{x}_i, y_i), i = 1, \cdots, T\}$ (training examples)

Overview

- $f(\mathbf{x}, \xi)$: a parametrized family of functions as candidates for the desired output
- ξ: a vector parameter. The set of ξ is a parameter space.
- We search for the optimal $\hat{\xi}$ that approximates the ture $f(\xi)$ by using training examples D.
- Regression Problem: when y takes an analog value.
- Pattern Recognition: when y is decrete, say, binary.

Overview

- The instantaneous loss function (of processing $\mathbf{x}$ by machine $f(\mathbf{x}, \xi)$), e.g.:

$$l(\mathbf{x}, y, \xi) = \frac{1}{2} \{y - f(\mathbf{x}, \xi)\}^2 \tag{2}$$

- Generalization error, the loss function function (of machine ξ is the expectation of the instantaneous loss over all possible pairs $(\mathbf{x}, y)$:

$$L(\xi) = E_{p(\mathbf{x}, y)} \{l(\mathbf{x}, y, \xi)\} \tag{3}$$

  where the expectation is taken with respect to the unknown joint probability distribution $p(\mathbf{x}, y)$.

- Training error:

$$L_{\text{train}}(\xi) = \frac{1}{T} \sum_{t=1}^{T} l(\mathbf{x}_t, y_t; \xi_t) \tag{4}$$

  since we do not know $p(\mathbf{x}, y)$, we use the average over the training data.

Overview

- Since we do not know L, we minimize the training error L_{train} to obtain $\hat{\xi}$.
- A regularization term may be added to L_{train}.

On-line Learning

- On-line Learning: Modifying the current candidate $\boldsymbol{\xi}_t$ at time t to obtain $\boldsymbol{\xi}_{t+1}$ at the next time $t+1$ based on the current training example $(\mathbf{x}_t, y_t)$ so as to decrease the instantaneous loss $l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t)$.
- Usually, the negative of the gradient is used to update $\boldsymbol{\xi}_t$:

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \nabla_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t) \tag{5}$$

- ∇: the gradient with respect to $\boldsymbol{\xi}$
- η_t: learning rate, which may depend on t.

On-line Learning

- Since traing data $(\mathbf{x}_t, y_t)$ are given one by one, the change is a random variable,

$$\Delta \boldsymbol{\xi}_t = -\eta_t \nabla_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t) \tag{6}$$

- The expectation of $\nabla_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t)$ equals $\nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}_t)$.
- Stochastic Gradient Descent (SGD) Learning:

$$\Delta \boldsymbol{\xi}_t = -\eta_t \nabla_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t) \tag{7}$$

- Gradient Descent (GD) Learning:

$$E\{\Delta \boldsymbol{\xi}_t\} = -\eta_t E\{\nabla_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t)\} = -\eta_t \nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}_t) \tag{8}$$

- The change of $\Delta \boldsymbol{\xi}_t$ is random but its expectation is in the direction of $-\nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}_t)$.
- Well established as the back-propagation learing method.

On-line Learning

Gradient descent of expected loss L v.s. stochastic gradient descent of l.

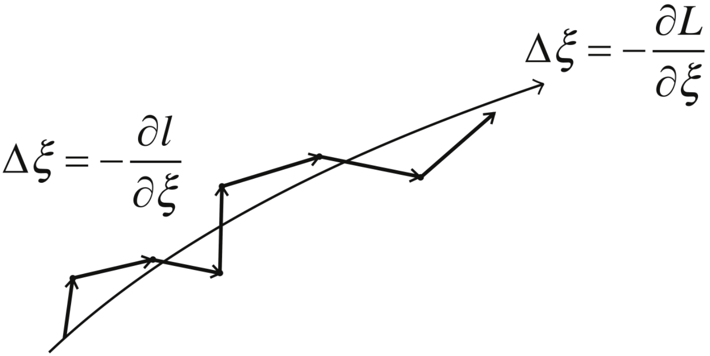

Figure: See [4].

Batch Learning

- Batch Learning: an iterative method which uses all the training data for modifying $\boldsymbol{\xi}_t$ at one step, such that $\boldsymbol{\xi}_t$ is modified to $\boldsymbol{\xi}_{t+1}$ by
- The update

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \frac{1}{T} \sum_{i=1}^{T} \nabla_{\boldsymbol{\xi}} l(\mathbf{x}_i, y_i, \boldsymbol{\xi}_t) \tag{9}$$

What is Natural Gradient

- It is believed that the gradient

$$\nabla_{\xi} L(\xi) = \frac{\partial L(\xi)}{\partial \xi} \tag{10}$$

  is the direction of the steepest change of $L(\xi)$.
- In a geographical map with contour lines, the steepest direction is given by the gradient of the height function $H(\xi)$, i.e. $\nabla_{\xi} H(\xi)$ is orthogonal to contour lines.
- However, this is ture only when an orthonormal coordinate system is used in a Euclidean space.
- Ordinary Gradient: steepest descent direction in Euclidean Manifolds
- Natural Gradient: steepest descent direction in Riemannian Manifolds

What is Natural Gradient

- It is believed that the gradient

$$\nabla_{\xi} L(\xi) = \frac{\partial L(\xi)}{\partial \xi} \tag{11}$$

  is the direction of the steepest change of $L(\xi)$.
- In a geographical map with contour lines, the steepest direction is given by the gradient of the height function $H(\xi)$, i.e. $\nabla_{\xi} H(\xi)$ is orthogonal to contour lines.
- However, this is ture only when an orthonormal coordinate system is used in a Euclidean space.
- Ordinary Gradient: steepest descent direction in Euclidean Manifolds
- Natural Gradient: steepest descent direction in Riemannian Manifolds

A Riemannian Manifold

- A Riemannian manifold can be intuitively seen as one kind of high dimensional ($\geqslant 4D$, or k-manifold, $k \geqslant 3$) surface, a hypersurface.

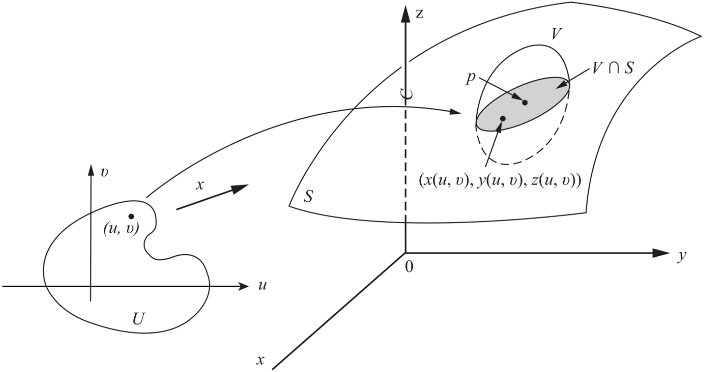

Figure: A 2-manifold, 3D surface. See [5].

A Riemannian Manifold

- In a Riemannian manifold, the square of local distance between two nearby points $\boldsymbol{\xi}$ and $\boldsymbol{\xi} + d\boldsymbol{\xi}$ is given by the quadractic form:

$$ds^2 = g_{ij} d\xi^i \xi^j \tag{12}$$

- Einstein convention is applied here.
- $\mathbf{G} = [g_{ij}]$: a Riemannian metric tensor.
- Let us change the current point $\boldsymbol{\xi}$ to $\boldsymbol{\xi} + d\boldsymbol{\xi}$, and see how the value of $L(\boldsymbol{\xi})$ changes, depending on the direction $d\boldsymbol{\xi}$. We search for the direction in which L changes most rapidly.

Natural Gradient

- In order to make a fair comparison, the step-size of $d\boldsymbol{\xi}$ should have the same magnitude in all directions, so that the length of $d\boldsymbol{\xi}$ should be the same,

$$g_{ij}(\boldsymbol{\xi})d\xi^i d\xi^j = \epsilon^2 \tag{13}$$

- ϵ: a small constant
- We put $d\boldsymbol{\xi} = \epsilon\mathbf{a}$ and require that

$$|\mathbf{a}|^2 = g_{ij}a^i a^j = 1 \tag{14}$$

- The Steepest direction of L is the maximizer of

$$L(\boldsymbol{\xi} + d\boldsymbol{\xi}) - L(\boldsymbol{\xi}) = \epsilon\nabla L(\boldsymbol{\xi}) \cdot \mathbf{a} \tag{15}$$

under the constraint (14)

Natural Gradient

Natural Gradient

- Using the variational method, we obtain:

$$\underset{\mathbf{a}}{\text{maximize}} \nabla_{\xi} L(\xi) \cdot \mathbf{a} - \lambda g_{ij} a^{i} aj \tag{16}$$

- This is a quadratic problem and the steepest direction is obtained as

$$\mathbf{a} \propto \mathbf{G}^{-1} \nabla_{\xi} L(\xi) \tag{17}$$

- The natural gradient or Riemannian gradient of L is

$$\widetilde{\nabla_{\xi}} L(\xi) = \mathbf{G}^{-1}(\xi) \nabla_{\xi} L(\xi) \tag{18}$$

- The natural gradient operator is

$$\widetilde{\nabla_{\xi}} = \mathbf{G}^{-1}(\xi) \nabla_{\xi} \tag{19}$$

Proof

Our goal:

$$\underset{\mathbf{a}}{\text{maximize}} \nabla_\xi L(\xi) \cdot \mathbf{a} - \lambda g_{ij} a^i aj \tag{20}$$

By the Lagrangian method, we take

$$\frac{\partial}{\partial a_i} \left\{ \nabla_\xi L(\xi)^\mathsf{T} \mathbf{a} - \lambda \mathbf{a}^\mathsf{T} \mathbf{G} \mathbf{a} \right\} = 0 \tag{21}$$

Then, we get

$$\nabla_\xi L(\xi) = 2\lambda \mathbf{G} \mathbf{a} \tag{22}$$

$$\mathbf{a} = \frac{1}{2\lambda} \mathbf{G}^{-1} \nabla_\xi L(\xi) \tag{23}$$

Using the constraint $|\mathbf{a}|^2 = g_{ij} a^i a^j = 1$, λ is determined.

Natural Gradient: Discussions

- Special Case: when an orthonoral coordinate system is used in a Euclidean space, we have

$$g_{ij}(\boldsymbol{\xi}) = \delta_{ij} \tag{24}$$

- On-line learning mode, the natural gradient decent method:

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \widetilde{\nabla_{\boldsymbol{\xi}}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t) \tag{25}$$

- Batch learning mode,

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \frac{1}{T} \sum_{i=1}^{T} \widetilde{\nabla_{\boldsymbol{\xi}}} l(\mathbf{x}_i, y_i, \boldsymbol{\xi}_t) \tag{26}$$

Natural Gradient: Discussions

In the case of statistical estimation,

- The loss function and the Riemannian metric **G** are related.
- The Fisher information is a Riemannian metric

$$g_{ij} = E\left[\frac{\partial \log p(\mathbf{x}, \boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \log p(\mathbf{x}, \boldsymbol{\xi})}{\partial \xi_j}\right] \tag{27}$$

- The loss function L uses the same log likelihood $\log p(\mathbf{x}, \boldsymbol{\xi})$.
- The Reimannian **G** uses the same log likelihood $\log p(\mathbf{x}, \boldsymbol{\xi})$.
- The natural gradient method is a version of the Gauss-Newton method.

Natural Gradient: Discussions

In the case of independent component analysis,

- The loss function and the Riemannian metric $\mathbf{G}$ are NOT related.
- The natural gradient learning method is useful in such case, too.
- Parameter space is a set of mixing matrices.
- The loss function L is measured by the degree of independence of unmixed signals.
- The Reimannian $\mathbf{G}$ is measured by the invariant metric of Lie group.

Natural Gradient: Discussions

In the case of independent component analysis,

- Deep learning: [Roux et al. 2007; Ollivier 2015]
- Reinforcement learning: as a policy natual gradient [Kakade 2002; Peters and Schaal 2008; Morimura et al. 2009]
- Optimization: stochastic relaxation technique [Malagò and Pistone 2014; Malagò et al. 2013; Yi et al. 2009; see also Hansen and Ostermeier 2001]

Natural Gradient: Property 1

Natural gradient learning is Fisher efficient.

Theorem

The estimator obtained by on-line natural gradient learning

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \widetilde{\nabla}_{\boldsymbol{\xi}} l(\mathbf{x}_t, y_t, \boldsymbol{\xi}_t) \tag{28}$$

is Fisher efficient, attaining the Cramér-Rao lower bound asymptotically.

Natural Gradient: Property 1, Proof

- Let the error covariance matrix of the estimator at time t be

$$\mathbf{V}_{t+1} = E\left[\left(\xi_{t+1} - \xi_0\right)\left(\xi_{t+1} - \xi_0\right)^{\mathsf{T}}\right] \tag{29}$$

- ξ_0 is the ture value of ξ.
- We expand the loss at ξ_t as

$$\nabla_{\xi} l(\mathbf{x}_t, y_t, \xi_t) = \nabla_{\xi} l(\mathbf{x}_t, y_t, \xi_0) + \nabla_{\xi}\nabla_{\xi} l(\mathbf{x}_t, y_t, \xi_0) \cdot (\xi_t - \xi_0) \tag{30}$$

- Substracting ξ_0 form both sides of (28) and substituting it in (29), we have

$$\mathbf{V}_{t+1} = \mathbf{V}_t - \frac{2}{t}\mathbf{V}_t + \frac{1}{t^2}\mathbf{G}^{-1} + O\left(\frac{1}{t^3}\right) \tag{31}$$

where

$$E\left[\nabla_{\xi} l(\mathbf{x}_t, y_t, \xi_0)\right] = 0 \tag{32}$$

$$E\left[\nabla_{\xi}\nabla_{\xi} l(\mathbf{x}_t, y_t, \xi_0)\right] = \mathbf{G}(\xi_0) \tag{33}$$

- Note that

$$\mathbf{G}(\xi_t) = \mathbf{G}(\xi_0) + O\left(\frac{1}{t}\right) \tag{34}$$

- Then the solution of (31) is asymptotically

$$\mathbf{V}_t = \frac{1}{t}\mathbf{G}^{-1} \tag{35}$$

which prove the theorem.

Natural Gradient: Property 2

- Consider a regression problem, the output is written as
$$y = f(\mathbf{x}, \boldsymbol{\xi}) + \epsilon \tag{36}$$
- First, we consider a simple perceptron, where f is written as
$$f(\mathbf{x}, \boldsymbol{\xi}) = \phi(\mathbf{w} \cdot \mathbf{x}) \tag{37}$$
- Here, we neglect the bias term for simplicity.
- The parameter is a vector $\boldsymbol{\xi} = \mathbf{w}$ and the activation function ϕ is a sigmoid function,
$$\phi(u) = tanh(u) \tag{38}$$
- The gradient is
$$\nabla l(\mathbf{x}, y, \mathbf{w}) = -(y - f)\phi^{'}(\mathbf{w} \cdot \mathbf{x})\mathbf{x} \tag{39}$$
- When the absolute value of $\mathbf{w}$ is large, $\phi(\mathbf{w} \cdot \mathbf{x})$ saturates for most $\mathbf{x}$, becoming nearly equal $+1$ or -1.
- This is the saturation problem, where the gradient almost equal to 0 because $\phi^{'} \approx 0$, and the ordinary SGD learning becomes slow.
- This is not serious in the case of a simple perceptron, but is serious in the case of multilayer perceptrons used in deep learning, where $f(\mathbf{x}, \boldsymbol{\xi})$ is composed of a concatenation of many f. In MLP, We may write the output as
$$f(\mathbf{x}, \boldsymbol{\xi}) = \phi(\mathbf{W}_k\phi(\mathbf{W}_{k-1}\phi \cdots \phi(\mathbf{W}))) \tag{40}$$
where $\boldsymbol{\xi} = (\mathbf{W}_1, \cdots, \mathbf{W}_k)$
- Its derivative with respect to $\mathbf{W}_1$, for example, includes the product of many $\phi^{'}$. Hence, it is almost vanishing in many cases. This is considered as a flaw of back-propagation in deep learning.

Natural Gradient: Property 2

- The natural gradient learning method is free of such a saturation problem. The gradient is written as

$$\nabla l(\mathbf{x}, y, \boldsymbol{\xi}) = -(y - f)\nabla f(\mathbf{x}, \boldsymbol{\xi}) \tag{41}$$

- The magnitude of the ordinary gradient would be very small in many cases but the natural gradient is different.

Natural Gradient: Property 2

Natural gradient is Saturation Free.

Theorem

The magnitude of the natural gradient is given by

$$E\left[\left\|\widetilde{\nabla} l\right\|^2\right] = \mathrm{tr}\left(\overline{\mathbf{G}}(\xi)\overline{\mathbf{G}}^{-1}(\xi)\right) \tag{42}$$

where

$$\overline{\mathbf{G}}(\xi) = E_{p(\mathbf{x},y,\xi_0)}\left[\nabla_\xi l(\mathbf{x},\xi) l(\mathbf{x},\xi)^T\right] \tag{43}$$

It dose not vanish even when ϕ' is small. Moreover,

$$E\left[\left\|\widetilde{\nabla} l\right\|^2\right] \approx k \tag{44}$$

in a neighborhood of the optimal ξ_0, where k is the dimension of ξ.

Natural Gradient: Property 2, Proof

Firstly, from

$$\nabla_{\boldsymbol{\xi}} l(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{G}^{-1}(\mathbf{x}, \boldsymbol{\xi}) \nabla_{\boldsymbol{\xi}} l(\mathbf{x}, \boldsymbol{\xi}) \tag{45}$$

We have

$$E\left[\left\|\widetilde{\nabla} l\right\|^2\right] = E_{p(\mathbf{x}, y, \boldsymbol{\xi}_0)}\left[\mathrm{tr}\mathbf{G}(\boldsymbol{\xi})\mathbf{G}^{-1}(\boldsymbol{\xi})\nabla_{\boldsymbol{\xi}} l(\mathbf{x}, \boldsymbol{\xi}) l(\mathbf{x}, \boldsymbol{\xi})^{\mathsf{T}}\mathbf{G}^{-1}(\boldsymbol{\xi})\right] \tag{46}$$

which completes the proof.
Secondly, when $\boldsymbol{\xi} = \boldsymbol{\xi}_0$, we easily have (44).

References I

S.-i. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998. [Online]. Available: https://doi.org/10.1162/089976698300017746

S. Fiori, "Extended hamiltonian learning on riemannian manifolds: Theoretical aspects," *IEEE Transactions on Neural Networks*, vol. 22, no. 5, pp. 687–700, May 2011.

S. Fiori, "Extended hamiltonian learning on riemannian manifolds: Numerical aspects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 7–21, Jan 2012.

S.-i. Amari, *Information Geometry and Its Applications*, 1st ed. Springer Publishing Company, Incorporated, 2016.

M. do Carmo, *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. [Online]. Available: https://books.google.com.tw/books?id=1v0YAQAAIAAJ

The End