Sparse Variational Learning with Matrix Normal Distributions

Viktor Rudnev [1,2]   Dmitry Kropotov [1,2]

[1]Samsung-HSE Lab

[2]Lomonosov Moscow State University

March 2019

Weight distribution inference

- Posterior distribution $p(\theta, \mathcal{D})$ over DNN weights

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta) p(\theta)}{p(\mathcal{D})}$$

- Training = fitting tractable posterior approximation

$$q(\theta) \approx p(\theta \mid \mathcal{D}_{\text{train}})$$

- Inference = Bayesian model averaging

$$p(\mathcal{D}_{\text{test}} \mid \mathcal{D}_{\text{train}}) = \mathbb{E}_{p(\theta \mid \mathcal{D}_{\text{train}})}[p(\mathcal{D}_{\text{test}} \mid \theta)] \approx \frac{1}{K} \sum_{k=1}^{K} p(\mathcal{D}_{\text{test}} \mid \theta)$$

Posterior approximation families

- $q(\theta) = \mathcal{N}(\mu, \mathrm{diag}(\sigma))$ — too simple,
- Normalizing flows — don't scale well (although FFJORD [3] is nice),
- Implicit — don't scale well,
- Hierarchical models — ok, but still slow to train

Matrix Normal distributions

$$W \in \mathbb{R}^{n\times m} \sim \mathcal{MN}(M, U, V) \Leftrightarrow$$
$$p(W) = \frac{\exp\left(-\frac{1}{2}\mathrm{Tr}\left[U^{-1}(W-M)^T V^{-1}(W-M)\right]\right)}{(2\pi)^{nm/2}|U|^{n/2}|V|^{m/2}},$$

where $M \in \mathbb{R}^{n\times m}$ is the mean matrix, $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are covariance matrices, i.e., symmetric and positive definite ones

$$X \sim \mathcal{MN}(M, U, V) \Leftrightarrow$$
$$\mathrm{vec}(X) \sim \mathcal{N}\left(\mathrm{vec}(M), U \otimes V\right)$$

If $M = 0$, rows and columns of W follow two full normal distributions, that are independent of each other

In other papers

- K-FAC [6]
- Scalable Laplace Approximation [9]
- Second Order SGLD [7]
- Overcoming Catastrophic Forgetting [8]

Motivation

Multivariate regression problem:

$$\mathcal{D} = (x_i, y_i)_{i=1}^{N},\ x_i \in \mathbb{R}^{\mathrm{in}},\ y_i \in \mathbb{R}^{\mathrm{out}}$$

$$p(y_i \mid x_i, W, \beta) = \mathcal{N}\left(y_i \mid Wx_i, \beta^{-1} I\right)$$

$$p(W \mid \alpha) = \mathcal{MN}\left(W \mid 0, I, \mathrm{diag}(\alpha)^{-1}\right)$$

$$p(Y \mid X, W, \alpha, \beta) = P(W \mid \alpha) \prod_{i=1}^{N} p(y_i \mid x_i, W, \beta)$$

Motivation

Exact posterior:

$$p(W \mid Y, X, \alpha, \beta) = \mathcal{MN}(W \mid M, U, V),$$
$$U = I,\ V = \left(\beta X^T X + \mathrm{diag}(\alpha)\right)^{-1},$$
$$M^T = V \beta X^T Y$$

Using variational learning with independent normal distributions would have ignored many correlations between the components of W

Parameterization

- Most papers use low-rank approximation for U and V, e.g.,

$$U = \mathrm{diag}(a) + uv^T; V = \mathrm{diag}(b) + pq^T$$

- We opt to using Cholesky factorization without sacrificing the time complexity:

$$U = AA^T,\ V = BB^T,$$

  where A, B are lower-triangular matrices.
- Also, we optimize log of their diagonal values instead to enforce positive-definiteness
- Benefits: unrestricted, "free" logdet computation, quick reparameterization tricks, Riemannian optimization (WIP)

Variational Learning Checklist

To perform VL one needs to learn how to compute

- Reparameterization trick,
- Local reparameterization trick [4],
- $\mathrm{KL}\left(\mathcal{MN}(M, U, V) \| N\left(0, \mathrm{diag}\left(\sigma^2\right)\right)\right)$,

Reparameterization trick

$$
\begin{gathered}
\mathrm{vec}(X) \sim \mathcal{N}(0, I), \\
W = M + AXB^T \Rightarrow \\
W \sim \mathcal{MN}\left(M, AA^T, BB^T\right)
\end{gathered}
$$

Local reparameterization trick

$$W \sim \mathcal{MN}\left(M, AA^T, BB^T\right),\ x \in \mathbb{R}^m \Rightarrow$$
$$W^T x \sim \mathcal{N}\left(W^T x \mid \mu, ZZ^T\right)$$

Now, if we want to sample $o \sim W^T x$, this is it:

$$\varepsilon \sim \mathcal{N}(0, I)$$
$$o = \mu + Z\varepsilon = M^T x + \varepsilon\sqrt{x^T AA^T x} B^T$$

KL divergence

$$\mathrm{KL}\left(\mathcal{MN}(M, U, V) \| N\left(0, \mathrm{diag}\left(\sigma^2\right)\right)\right) =$$

KL divergence

$$\mathrm{KL}\left(\mathcal{MN}(M, U, V) \| N\left(0, \mathrm{diag}\left(\sigma^2\right)\right)\right) =$$

$$\frac{1}{2}\left(\log \frac{|\mathrm{diag}(\sigma^2)|}{|U \otimes V|} - nm + \mathrm{Tr}\left(\mathrm{diag}(\sigma^2)^{-1}(U \otimes V)\right) + M^T \mathrm{diag}(\sigma^2)^{-1} M\right) =$$

KL divergence

$$\mathrm{KL}\left(\mathcal{MN}(M,U,V) \| N\left(0, \mathrm{diag}\left(\sigma^2\right)\right)\right) =$$

$$\frac{1}{2}\left(\log \frac{|\mathrm{diag}(\sigma^2)|}{|U \otimes V|} - nm + \mathrm{Tr}\left(\mathrm{diag}(\sigma^2)^{-1}(U \otimes V)\right) + M^T \mathrm{diag}(\sigma^2)^{-1} M\right) =$$

$$\frac{1}{2}\left(\sum_{k=1}^{nm} 2\log\sigma_k - \log|U|^m|V|^n - nm + \sum_{k,l=1}^{n,m} \sigma_{km+l}^{-2} U_{kk} V_{ll} + \sum_{k=1}^{nm} \frac{M_{kl}^2}{\sigma_{km+l}^2}\right)$$

KL divergence

$$\mathrm{KL}\left(\mathcal{MN}(M, U, V) \| N\left(0, \mathrm{diag}\left(\sigma^2\right)\right)\right) =$$

$$\frac{1}{2}\left(\log \frac{|\mathrm{diag}(\sigma^2)|}{|U \otimes V|} - nm + \mathrm{Tr}\left(\mathrm{diag}(\sigma^2)^{-1}(U \otimes V)\right) + M^T \mathrm{diag}(\sigma^2)^{-1} M\right) =$$

$$\frac{1}{2}\left(\sum_{k=1}^{nm} 2 \log \sigma_k - \log |U|^m |V|^n - nm + \sum_{k,l=1}^{n,m} \sigma_{km+l}^{-2} U_{kk} V_{ll} + \sum_{k=1}^{nm} \frac{M_{kl}^2}{\sigma_{km+l}^2}\right)$$

If $X = AA^T$, where A is lower triangular, then

$$\log |X| = 2 \log |A| = 2 \sum_{k=1}^{n} \log A_{kk}$$

ARD sparsification

$$\mathcal{D} = (x_i, y_i)_{i=1}^{N}$$

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{N} p(y_i \mid x_i, \theta), \theta \in \mathbb{R}^D$$

$$p(\theta \mid \alpha) \text{ — prior parametrized by } \alpha$$

ARD sparsification

$$\mathcal{D} = (x_i, y_i)_{i=1}^N$$
$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^N p(y_i \mid x_i, \theta), \theta \in \mathbb{R}^D$$
$$p(\theta \mid \alpha) \text{ — prior parametrized by } \alpha$$

We want to find the posterior and optimal α:

$$p(\theta \mid \mathcal{D}, \alpha) = \frac{p(\mathcal{D} \mid \theta) p(\theta \mid \alpha)}{p(\mathcal{D} \mid \alpha)}$$
$$\alpha^* = \arg\max_\alpha p(\mathcal{D} \mid \alpha) = \arg\max_\alpha \int p(\mathcal{D} \mid \theta) p(\theta \mid \alpha) \, \mathrm{d}\theta$$

When $p(\theta \mid \alpha) = \prod_{i=1}^D \mathcal{N}\left(\theta_i \mid 0, \alpha_i^{-1}\right)$, this is called ARD

ARD sparsification

ELBO: $$\log p(\mathcal{D} \mid \alpha) \geq \mathcal{L}(\phi, \alpha) = \mathbb{E}_{q(\theta \mid \phi)}[\log p(\mathcal{D} \mid \theta)] - \\ - D_{\mathrm{KL}}(q(\theta \mid \phi) \| p(\theta \mid \alpha)) \to \max_{\phi, \alpha}$$

ARD sparsification

ELBO: $$\log p(\mathcal{D} \mid \alpha) \geq \mathcal{L}(\phi, \alpha) = \mathbb{E}_{q(\theta|\phi)}[\log p(\mathcal{D} \mid \theta)] - \\ - D_{\mathrm{KL}}(q(\theta \mid \phi) \| p(\theta \mid \alpha)) \to \max_{\phi, \alpha}$$

Now suppose that:

$$p(\theta \mid \alpha) = \prod_{i=1}^{D} \mathcal{N}(\theta_i \mid 0, \alpha_i^{-1}); \;\; q(\theta \mid \mu, \sigma) = \prod_{i=1}^{D} \mathcal{N}\left(\theta_i \mid \mu_i, \sigma_i^2\right)$$

ARD sparsification

ELBO: $$\log p(\mathcal{D} \mid \alpha) \geq \mathcal{L}(\phi, \alpha) = \mathbb{E}_{q(\theta|\phi)}[\log p(\mathcal{D} \mid \theta)] - \\ -D_{\mathrm{KL}}(q(\theta \mid \phi) \| p(\theta \mid \alpha)) \to \max_{\phi, \alpha}$$

Now suppose that:

$$p(\theta \mid \alpha) = \prod_{i=1}^{D} \mathcal{N}(\theta_i \mid 0, \alpha_i^{-1}); \;\; q(\theta \mid \mu, \sigma) = \prod_{i=1}^{D} \mathcal{N}\left(\theta_i \mid \mu_i, \sigma_i^2\right)$$

Then $\alpha_i^* = \left(\mu_i^2 + \sigma_i^2\right)^{-1}$

ARD sparsification

ELBO: $$\log p(\mathcal{D} \mid \alpha) \geq \mathcal{L}(\phi, \alpha) = \mathbb{E}_{q(\theta|\phi)}[\log p(\mathcal{D} \mid \theta)] - \\ - D_{\mathrm{KL}}(q(\theta \mid \phi) \| p(\theta \mid \alpha)) \to \max_{\phi, \alpha}$$

Now suppose that:

$$p(\theta \mid \alpha) = \prod_{i=1}^{D} \mathcal{N}(\theta_i \mid 0, \alpha_i^{-1}); \; q(\theta \mid \mu, \sigma) = \prod_{i=1}^{D} \mathcal{N}\left(\theta_i \mid \mu_i, \sigma_i^2\right)$$

Then $\alpha_i^* = \left(\mu_i^2 + \sigma_i^2\right)^{-1}$. Let's it substitute back:

$$\mathcal{L}_{\mathrm{ARD}}(\mu, \sigma) = \sum_{i=1}^{N} \mathbb{E}_{q(\theta|\mu,\sigma)} \left[\log p(y_i \mid x_i, \theta)\right] - \frac{1}{2} \sum_{j=1}^{D} \log \left(1 + \frac{\mu_j^2}{\sigma_j^2}\right) =$$

$$= \mathcal{L}_{\mathcal{D}}(\mu, \sigma) + \mathbb{R}_{\mathrm{ARD}} \to \max_{\mu, \sigma}$$

Group ARD sparsification

ARD prior:

$$p(\theta \mid \alpha) = \prod_{i=1}^{D} \mathcal{N}\left(\theta_i \mid 0, \alpha_i^{-1}\right)$$

Group ARD prior [5]:

$$p(W \mid \tau, \gamma) = \prod_{i=1}^{n} \prod_{j=1}^{m} \mathcal{N}\left(W_{ij} \mid 0, \tau_i^{-1} \gamma_j^{-1}\right)$$
$$= \mathcal{MN}\left(W, \mathrm{diag}(\tau), \mathrm{diag}(\gamma)\right)$$

Equivalent to a regular ARD with a particular structure:

$$\alpha_{im+j} = \tau_i \gamma_j$$

But one can't obtain τ_i^* and γ_j^* in closed form [2], so we optimize τ_i and γ_j with other parameters

MNIST

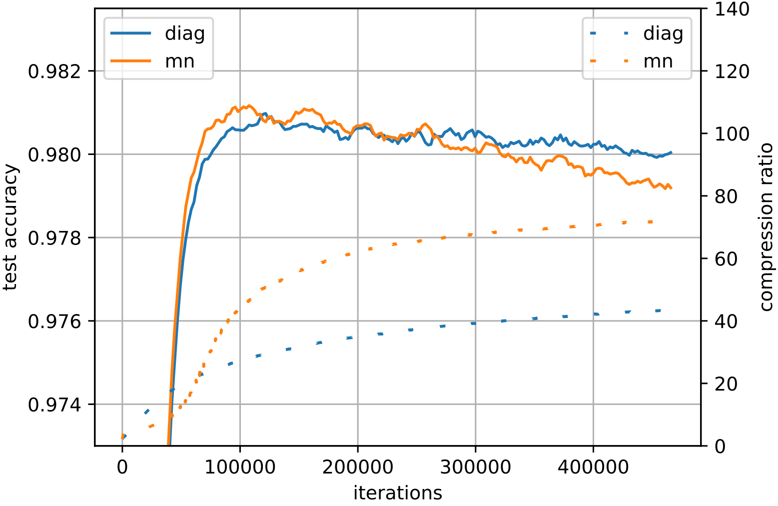


Figure: Learning a sparse fully-connected neural network on MNIST

Fashion MNIST

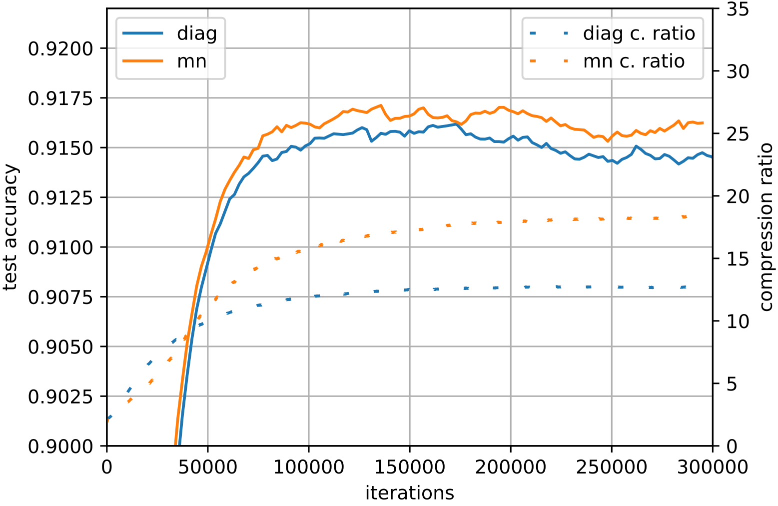


Figure: Learning a sparse convolutional neural network on Fashion MNIST

Fashion MNIST

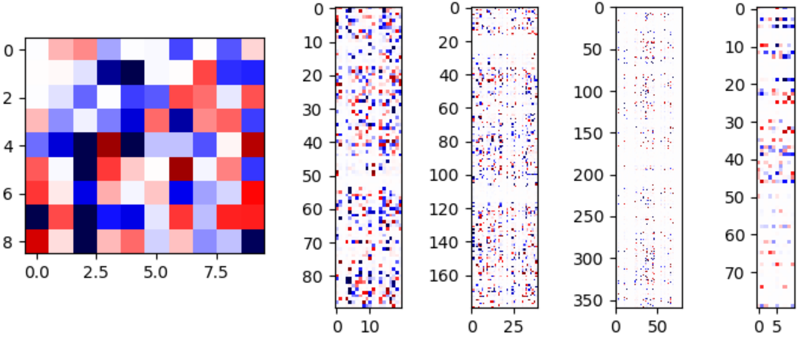


Figure: Learning a sparse convolutional neural network on Fashion MNIST. Weight matrices after learning

CIFAR10 Bayesian Learning

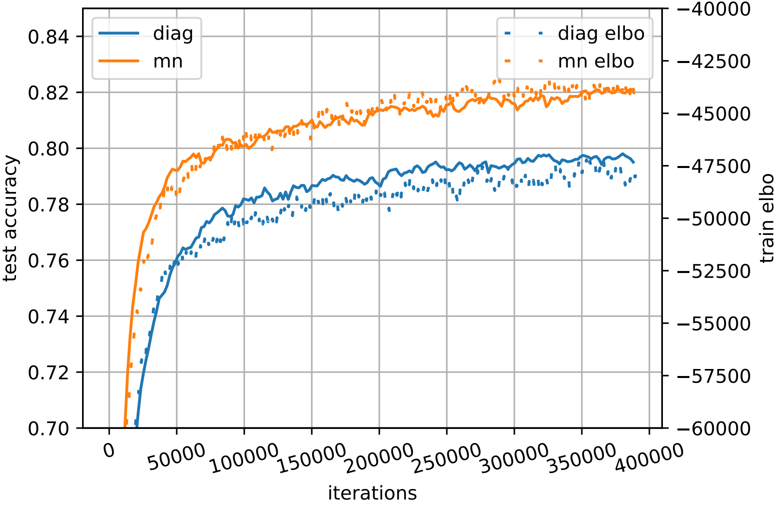


Figure: Learning a bayesian VGG-like network on CIFAR-10

CIFAR10 group sparsification

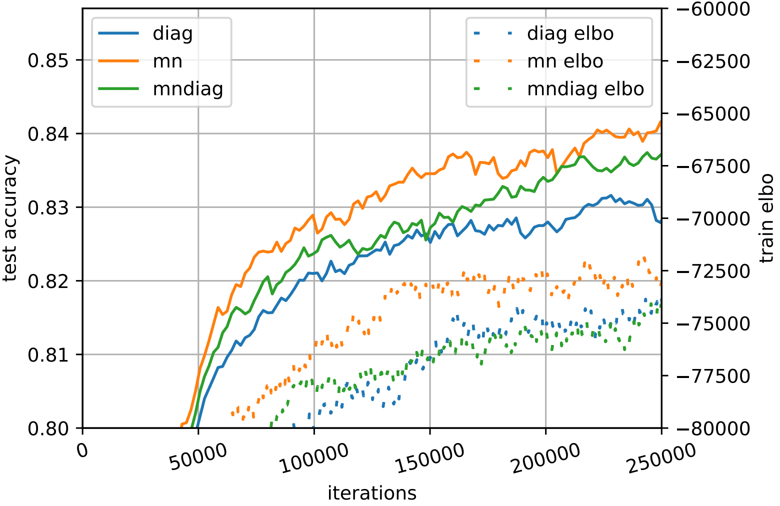


Figure: Test accuracy on CIFAR-10 with the VGG-like network and different posterior approximation families in the group-sparse learning task

CIFAR10 group sparsification

Diag (18610 weights left of 62270)												
A	0.3	0.9	0.9	1.8	3.6	3.6	7.2	14.4	14.4	7.2	3.6	3.6
W	0.2	0.4	0.5	0.9	1.8	2.3	3.1	4.4	2.3	1.4	0.4	0.4
%	30	61	40	52	49	35	57	69	84	80	89	90
MN (27764 weights left of 62270)												
A	0.3	0.9	0.9	1.8	3.6	3.6	7.2	14.4	14.4	7.2	3.6	3.6
W	0.2	0.5	0.6	1.2	1.4	2.4	5.7	6.7	5.2	0.6	0.2	2.5
%	31	41	29	36	62	33	21	54	64	92	94	29
MN+Diagonal (19936 weights left of 62270)												
A	0.3	0.9	0.9	1.8	3.6	3.6	7.2	14.4	14.4	7.2	3.6	3.6
W	0.2	0.3	0.6	1.5	1.7	2	3.7	3.8	3.1	1.2	0.7	0.4
%	37	65	32	18	51	44	48	74	79	84	80	88

Table: Group-sparse learning of the VGG-like network with different posterior approximation families. A – #weights in layers (in thousands), W – #weights left after deleting zeroed weights (in thousands), % – fraction of weights dropped (in percents)

CIFAR10 group sparsification x4

Free zeroing (120884 weights left of 246940)												
A	0.5	3.6	3.6	7.2	14.4	14.4	28.8	57.6	57.6	28.8	14.4	14.4
W	0.4	1.8	2.0	4.8	9.7	10.3	21.8	33.3	25.5	7.6	1.3	1.4
%	22	50	44	32	33	29	24	42	56	74	91	90
Row-only (3120 rows left of 4527)												
A	0.3	1.8	1.8	1.8	3.6	3.6	3.6	7.2	7.2	7.2	3.6	3.6
W	0.3	1.5	1.5	1.8	3.4	3.6	3.6	7.0	4.9	2.9	0.3	0.5
%	0	18	19	1	4	1	1	4	32	59	91	85
Column-only (413 columns left of 540)												
A	20	20	20	40	40	40	80	80	80	40	40	40
W	20	16	20	37	37	40	74	61	48	14	6	40
%	0	20	0	7	7	0	7	24	40	65	85	0

Table: Group-sparse learning of the 4x version of the VGG-like network with MN posterior. Free zeroing: A – #weights in layers (in thousands), W – #weights left after pruning (in thousands), % – fraction of weights dropped (in percents). Row-only: same, but A and W is in hundreds of rows. Column-only: same, but A and W are just #columns

Introduction to Riemannian Optimization

Taskynov Anuar

Moscow State University

taskynov.anuar@mail.ru

March 29, 2019

Overview

1. Manifolds
   - Definition
   - Tangent space
   - Riemannian Manifold
   - Riemannian gradient
2. Riemannian Optimization
   - Geodesic, exponential map
   - Retraction
   - Vector transport
3. Conclusion

Manifold

Definition (manifold)

Manifold $\mathcal{M}$ is a set which looks like Euclidean space around every point. Let $\mathcal{U}_x$ — neighborhood at the point $x \in \mathcal{M}$. Formally $\mathcal{M}$ is a d-**dimensional manifold** if $\forall x$, $\exists$ bijective function ϕ_x: $\mathcal{U}_x \to \mathbb{R}^d$, such that for neighborhoods at x and y $(\mathcal{U}_x \cap \mathcal{U}_y \neq \emptyset)$ the change of coordinates is smooth: $\phi_x \circ \phi_y^{-1}, \phi_y \circ \phi_x^{-1} \in C^\infty(\mathbb{R}^d)$.

- $\phi_x(x) \in \mathbb{R}^d$ is called the local (intrinsic) coordinates of point x.
- If $\mathcal{M} \subset \mathbb{R}^n$, then the point x has global (extrinsic) coordinates $(\in \mathbb{R}^n)$ ang local (intrinsic) coordinates $(\in \mathbb{R}^d)$. For example, $\mathcal{M} = \{x = (x_1, x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 = 1\}$, $x \in \mathbb{R}^2$ — extrinsic coordinates, $t \in [0, 2\pi)$ — intrinsic coordinates.

Manifold (Examples)

Examples:

- $\mathbb{R}^d$.
- Circle.
- Real projective $\mathbb{RP}^{n-1}$ is the set of all directions in $\mathbb{R}^n$.
- Grassman Manifold Grass(p, n) is the set, which parametrizes all p-dimensional linear subspaces of the n-dimensional vector space $\mathbb{R}^n$.
- Stiefel Manifold $\mathrm{St}(p, n) = \{X \in \mathbb{R}^{n \times p} : X^T X = I_p\}$.
- r-rank matrices $\mathcal{M}_r = \{X \in \mathbb{R}^{m \times p} : \mathrm{rank}(X) = r\}$.

Tangent space

Let

$$C_x := \left\{\gamma : \mathcal{I} \to \mathcal{M} : \gamma \in C^1(\mathcal{I}), 0 \in \mathcal{I} \text{ an open interval in } \mathbb{R}, \gamma(0) = x\right\}$$

– set of smooth curves γ on manifold $\mathcal{M}$.

Definition (tangent space for $\mathcal{M} \subset \mathbb{R}^n$)

The tangent space at $x \in \mathcal{M}$, noted $T_x\mathcal{M}$, is the linear subspace of $\mathbb{R}^n$ defined by:

$$T_x\mathcal{M} = \left\{v \in \mathbb{R}^n : v = \gamma'(0), \gamma \in C_x\right\}, \dim(T_x\mathcal{M}) = \dim(\mathcal{M}).$$

Example:

- $T_X\mathrm{St}(p, n) = \left\{Z \in \mathbb{R}^{n\times p} : X^T Z + Z^T X = 0\right\}$

$$= \left\{X\Omega + X_\perp K : \Omega^T = -\Omega, K \in \mathbb{R}^{(n-p)\times p}, X_\perp \in \mathbb{R}^{n\times(n-p)}, X_\perp^T X = 0\right\}.$$

Definition (tangent bundle)

The tangent bundle, noted $T\mathcal{M}$, is the set $T\mathcal{M} = \bigsqcup_{x\in\mathcal{M}} T_x\mathcal{M}$, where $\bigsqcup$ stands for disjoint union. The projection π extracts the root of a vector, that is, $\pi(\xi) = x$ if and only if $\xi \in T_x\mathcal{M}$.

Definition (vector field on $\mathcal{M}$)

A vector field $\mathbf{X} : \mathcal{M} \to T\mathcal{M}$ — smooth mapping, such that $(\pi \circ \mathbf{X})(x) = x$. The vector at x is written $\mathbf{X}_x = \mathbf{X}(x) \in T_x\mathcal{M}$. $\mathcal{X}(\mathcal{M})$ — set of all vector fields.

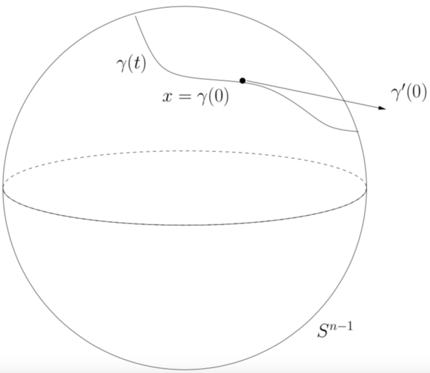


Figure: Tangent space.

Riemannian manifold

Definition (riemannian manifold)

A manifold whose tangent spaces are endowed with a smoothly varying inner product $g_x(\cdot,\cdot) = \langle \cdot,\cdot \rangle_x$ is called a **Riemannian manifold**. Smoothly varying can be understood in the following sense: for all vector fields $\mathbf{X}, \mathbf{Y} \in \mathcal{X}(\mathcal{M})$, the function $x \to g_x(\mathbf{X}_x, \mathbf{Y}_x)$ is a smooth function from $\mathcal{M}$ to $\mathbb{R}$.

- Inner product can be represented as:

$$g_x(\xi_x, \eta_x) = \hat{\xi}_{\hat{x}}^T G_{\hat{x}} \hat{\eta}_{\hat{x}},$$

where $G_{\hat{x}} \in \mathbb{R}^{d \times d}$ — symmetric, positive definite matrix, $\hat{\xi}_{\hat{x}}, \hat{\eta}_{\hat{x}}, \in \mathbb{R}^d$ is coordinate representation of tangent vectors, $\hat{x} \in \mathbb{R}^d$ — local coordinates of x.

- $$\|\xi\|_x := \sqrt{\langle \xi, \xi \rangle_x}, \quad \xi \in T_x\mathcal{M}$$

Riemannian gradient

- Given a smooth scalar field $f : \mathcal{M} \to \mathbb{R}$ on a Riemannian manifold, **the gradient** of f at x, denoted by $\mathrm{grad} f(x)$, is defined as the unique element of $\mathcal{T}_x\mathcal{M}$ that satisfies:

$$\langle \mathrm{grad} f(x), \xi \rangle_x = Df(x)[\xi] := \frac{d}{dt} f(\gamma(t))\Big|_{t=0}, \forall \xi = \gamma'(0) \in \mathcal{T}_x\mathcal{M}.$$

  Thus $\mathrm{grad} f : \mathcal{M} \to T\mathcal{M}$ is a vector field on $\mathcal{M}$.
- Coordinate expression: $\widehat{\mathrm{grad}} f(x) = G_{\hat{x}}^{-1} \mathrm{Grad} \hat{f}(\hat{x})$, where $\mathrm{Grad} \hat{f}(\hat{x})$ is a vector of partial derivatives.
- $$\frac{\mathrm{grad} f(x)}{\|\mathrm{grad} f(x)\|_x} = \underset{\xi \in T_x\mathcal{M}, \|\xi\|_x = 1}{\arg\max} Df(x)[\xi].$$
- Riemannian gradient for $\mathcal{M} \subset \mathbb{R}^n$:

$$\mathrm{grad} f(x) = \mathrm{Proj}_{T_x\mathcal{M}} \nabla \overline{f}(x),$$

  where $\overline{f} : \mathbb{R}^n \to \mathbb{R}$ such that f is a restriction of $\overline{f}$.

Riemannian optimization

- Optimization problem: $f(x) \to \min_{x \in \mathcal{M}}$, where $\mathcal{M}$ is a riemannian manifold.
- How can you optimize this function?
- Usual gradient descent step: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.
- For manifolds: $x_{k+1} = x_k - \alpha_k \mathrm{grad} f(x)$.

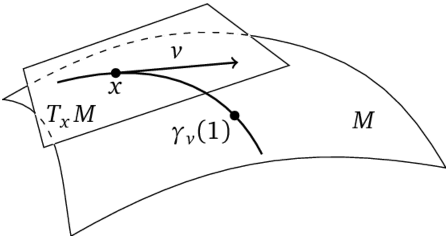


Figure: Mapping from tangent space $T_x\mathcal{M}$ to $\mathcal{M}$.

Geodesic

- Let $\mathbf{Y}, \mathbf{X} \in \mathcal{X}(\mathcal{M})$ — vector fields on $\mathcal{M} \subset \mathbb{R}^n$. **Covariant derivative** $\nabla_{\mathbf{X}}\mathbf{Y}$ of $\mathbf{Y}$ with respect to $\mathbf{X}$ is a vector field:

$$(\nabla_{\mathbf{X}}\mathbf{Y})_x = \lim_{t \to 0} \frac{\mathbf{Y}(x + t\mathbf{X}_x) - \mathbf{Y}(x)}{t} = D\mathbf{Y}(x)[\mathbf{X}_x].$$

Definition (geodesic)

A curve $\gamma : \mathcal{I} \to \mathcal{M}$ with $\mathcal{I}$ an open interval of $\mathbb{R}$ containing 0 is a **geodesic** if and only if $\nabla_{\gamma'(t)}\gamma'(t) = 0, \forall t \in \mathcal{I}$.

- Another definition of geodesic is a curve $\gamma : [a, b] \to \mathcal{M}$ which minimize:

$$\mathrm{L}(\gamma) := \int_a^b \sqrt{< \gamma'(t), \gamma'(t) >_{\gamma(t)}} dt$$

- Riemannian distance of two points $x, y \in \mathcal{M}$, $(\gamma(0) = x, \gamma(1) = y)$:

$$\mathrm{dist} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+ : (x, y) \to \mathrm{dist}(x, y) := \inf_{\gamma \in C^1([0,1] \to \mathcal{M})} \mathrm{L}(\gamma)$$

Definition (exponential map)

Let $\mathcal{M} \subset \mathbb{R}^n$ — riemannian manifold and $x \in \mathcal{M}$. For every $\xi \in T_x\mathcal{M}$, there exists an open interval $\mathcal{I}$ (which contains 0) and a unique geodesic $\gamma(t; x, \xi) : \mathcal{I} \to \mathcal{M}$ such that $\gamma(0) = x$ and $\gamma'(0) = \xi$. The mapping

$$\text{Exp}_x : T_x\mathcal{M} \to \mathcal{M} : \xi \to \text{Exp}_x(\xi) = \gamma(1; x, \xi)$$

is called **exponential map** at x. In particular, $\gamma(0; x, \xi) = x, \forall x \in \mathcal{M}$.

Optimization step: $x_{k+1} = \text{Exp}_{x_k}(-\alpha_k \text{grad} f(x_k))$.
Hard to compute! Because you should solve DE:

$$\begin{cases} \nabla_{\gamma'(t)}\gamma'(t) = 0, \quad t \in (0, 1] \\ \gamma(0) = x, \\ \gamma'(0) = \xi. \end{cases}$$

Retraction

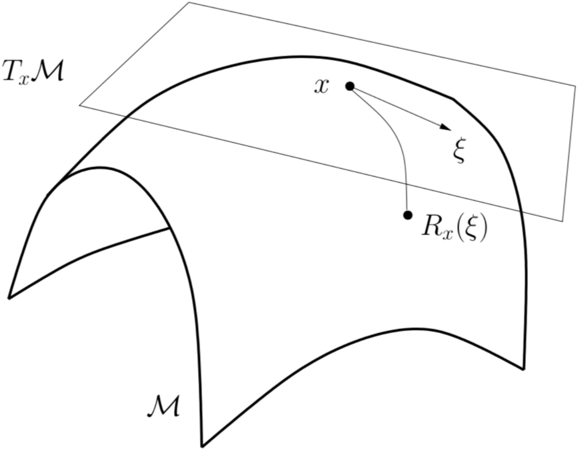


Figure: Retraction

Retraction

Exponential maps can be expensive to compute.

Definition (Retraction)

A **retraction** on a manifold $\mathcal{M}$ is a smooth mapping $R : \mathcal{TM} \to \mathcal{M}$, $\mathcal{M} \subset \mathbb{R}^n$ with the following properties. Let R_x denote the restriction of R to $\mathcal{T}_x\mathcal{M}$.

- $R_x(0) = x$, $0 \in \mathcal{T}_x\mathcal{M}$.
- $DR_x(0) = \mathrm{id}_{\mathcal{T}_x\mathcal{M}}$ – identity mapping, where $DR_x(0) : \mathcal{T}_x\mathcal{M} \to \mathcal{TM}$. Equivalently $\forall \xi \in \mathcal{T}_x\mathcal{M}$, the curve $\gamma_\xi(t) = R_x(t\xi)$ satisfies $DR_x(0)[\xi] = \gamma'_\xi(t)|_{t=0} = \lim_{\tau\to 0} \frac{\gamma_\xi(\tau)-\gamma_\xi}{\tau} = \xi$.

We may do the following step: $x_{k+1} = R_{x_k}(\alpha_k \eta_k)$, $\eta_k \in T_{x_k}\mathcal{M}$. If $\eta_k = -\mathrm{grad} f(x_k)$, then this step is one GD step.

Retraction on $\mathrm{St}(p, n)$

Retractions on $\mathrm{St}(p, n)$:

- $R_X(\xi) = \mathrm{Proj}_{\mathrm{St}(p,n)}(X + \xi)$.
- $R_X(\xi) = (X + \xi)(I_p + \xi^T \xi)^{-1/2}$.
- $R_X(\xi) = \mathrm{QR}(X + \xi)$, where $\mathrm{QR}(\cdot)$ — return orthogonal matrix from QR decomposition.
- $R_X(\xi) = \mathrm{Exp}_X(\xi)$.

Gradient Descent with Momentum

Optimization step on $\mathbb{R}^n$:

$$\begin{cases} d_k = \beta d_{k-1} + \alpha_k \nabla f(x_k); \\ x_{k+1} = x_k - d_k. \end{cases}$$

Back to the manifold $\mathcal{M}$:

$$\begin{cases} d_k = \underbrace{\beta d_{k-1}}_{\in T_{x_{k-1}}\mathcal{M}} + \underbrace{\alpha_k \mathrm{grad} f(x_k)}_{\in T_{x_k}\mathcal{M}}; \\ x_{k+1} = R_{x_k}(-d_k); \end{cases}$$

Vector Transport

Definition (vector transport)

A **vector transport** on a manifold M is a smooth mapping:

$\text{Transp} : \mathcal{TM} \times \mathcal{TM} \to \mathcal{TM}$,

satisfying the following properties for all $x \in \mathcal{M}$:

- $\exists R_x$, called the retraction associated with Transp: $\text{Transp}_\eta(\xi) = \mathcal{T}_{R_x(\eta)}\mathcal{M}$.
- $Transp_0(\xi) = \xi$, $\forall \xi \in \mathcal{T}_x\mathcal{M}$.
- $\text{Transp}_\eta(a\xi + b\zeta) = a\text{Transp}_\eta(\xi) + b\text{Transp}_\eta(\zeta)$.

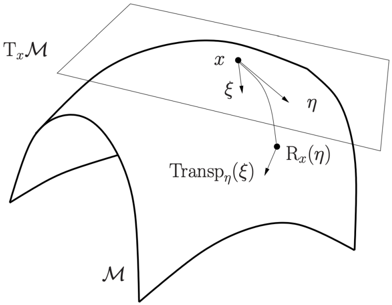


Figure: Vector transport

Riemannian GD with Momentum

Optimization step:

$$\begin{cases} d_k = \text{Transp}_{T_{x_{k-1} \to x_k} \mathcal{M}}(\beta d_{k-1}) + \alpha_k \text{grad} f(x_k); \\ x_{k+1} = R_{x\,k}(-d_k) \end{cases}$$

Conclusion

What do you need to optimize $f(x)$ on riemannian manifold $\mathcal{M}$?

- if you have intrinsic parametrization:
  - $\hat{\text{grad}} f(x) = G_{\hat{x}}^{-1} \nabla_{\hat{x}} f(\hat{x})$.
- if you have extrinsic parametrization:
  - Define tangent space: $T_x\mathcal{M}$;
  - Riemannian gradient: $\text{grad} f(x) = \text{Proj}_{T_x\mathcal{M}} \nabla \bar{f}(x)$, for $\mathcal{M} \subset \mathbb{R}^n$;
  - Retraction operation: $R_x(\xi), \xi \in T_x\mathcal{M}$.
  - Vector transport operation: $\text{Transp}_{T_{x \to y}\mathcal{M}}$.

Riemannian Optimization

Let's have a look at the covariance of MN again:

$$\Sigma = U \otimes V = AA^T \otimes BB^T$$

This is a rank-1 TT tensor, with positive-semidefinite cores (PSD)
So there's a large room for applying RO, since both TT and PSD are manifolds even by themselves.

PSD optimization

Let's find the tangent space of the PSD manifold:

$$U(t) = A(t)A(t)^T$$
$$\mathrm{d}U(t) = A(t)\,\mathrm{d}A(t)^T + \mathrm{d}A(t)A(t)^T$$
$$\delta U \in \mathcal{T}_{AA^T}^{\mathrm{PSD}} \Leftrightarrow \exists \delta A : \delta U = A\delta A^T + \delta A A^T$$

PSD optimization

Let's obtain a retraction:

$$R(U, Z, \varepsilon) = (A + \varepsilon Z)(A + \varepsilon Z)^T$$
$$R'_\varepsilon(U, Z, 0) = ZA^T + AZ^T$$

So, if we have

$$\delta U = A\delta A^T + \delta A A^T \in \mathcal{T}^{\mathrm{PSD}}_{AA^T},$$

then $R(U, \delta A, \varepsilon)$ is a valid retraction

PSD optimization

Suppose we have $U = AA^T$, $Z = \nabla_U f(U)$ for some $f(U)$
We want to project Z onto $\mathcal{T}_U^{\mathrm{PSD}}$, i.e., to solve

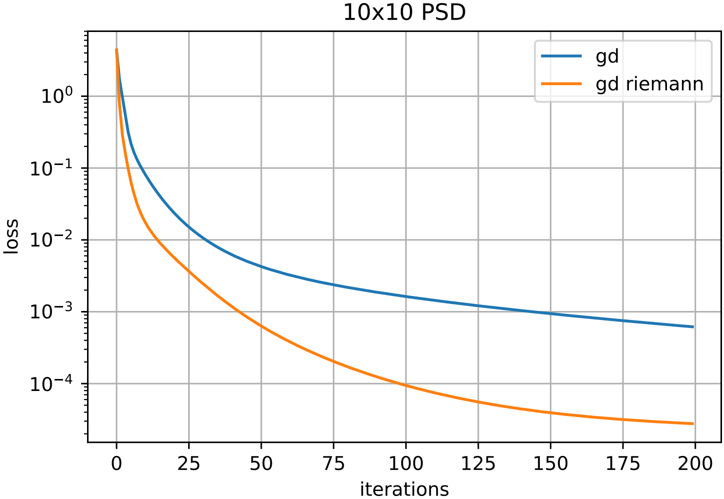
$$\|\delta U - Z\|^2 = \left\| A\delta A^T + \delta A A^T - Z \right\|^2 \to \min_{\delta A}$$

Solutions for intrinsic and extrinsic parameterizations:

$$\delta A = \frac{1}{4}\left(Z + Z^T\right) A^{-T} = \frac{1}{4}\left(\nabla_A f\right)\left(A^T A\right)^{-1}$$
$$\delta U = \delta A A^T + A\delta A^T = \frac{1}{2}\left(Z + Z^T\right)$$

PSD optimization

If we're optimizing U, we need retraction to stay in PSD manifold
If we're optimizing A, we need to regularize $\left(A^T A\right)^{-1}$
But it works:



Kronecker product optimization

Let's find the tangent space of the Kronecker product manifold:

$$U(t) = X(t) \otimes Y(t)$$
$$\mathrm{d}U(t) = \mathrm{d}X(t) \otimes Y(t) + X(t) \otimes \mathrm{d}Y(t)$$
$$\delta U \in \mathcal{T}^{\mathrm{KP}}_{X \otimes Y} \Leftrightarrow \exists \delta X, \delta Y : \delta U = \delta X \otimes Y + X \otimes \delta Y$$

Kronecker product optimization

Let's obtain a retraction:

$$R(U, A, B, \varepsilon) = (X + \varepsilon A) \otimes (Y + \varepsilon B)$$
$$R'_\varepsilon(U, A, B, 0) = A \otimes Y + X \otimes B$$

So, if we have

$$\delta U = \delta X \otimes Y + X \otimes \delta Y \in \mathcal{T}^{\mathrm{KP}}_{X \otimes Y},$$

then $R(U, \delta X, \delta Y, \varepsilon)$ is a valid retraction

Kronecker product optimization

Suppose we have $U = X \otimes Y$, $Z = \nabla_U f(U)$ for some $f(U)$
We want to project Z onto $\mathcal{T}_U^{\mathrm{KP}}$, i.e., to solve

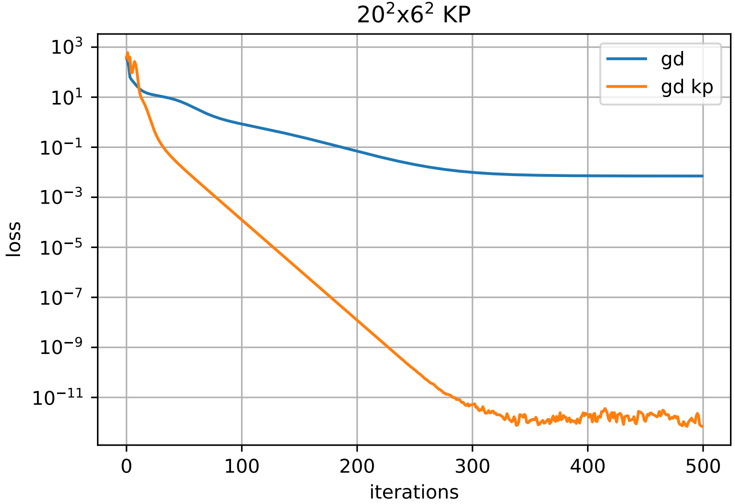
$$\|\delta U - Z\|^2 = \|\delta X \otimes Y + X \otimes \delta Y - Z\|^2 \to \min_{\delta X, \delta Y}$$

Solution:

$$\delta X = \frac{1}{\langle Y, Y \rangle}(\nabla_X f - X \langle Y, \delta Y \rangle)$$
$$\delta Y = \frac{1}{\langle X, X \rangle} \nabla_Y f$$

Kronecker product optimization

It works well when cores have different dimensions:



PSD Kronecker product optimization

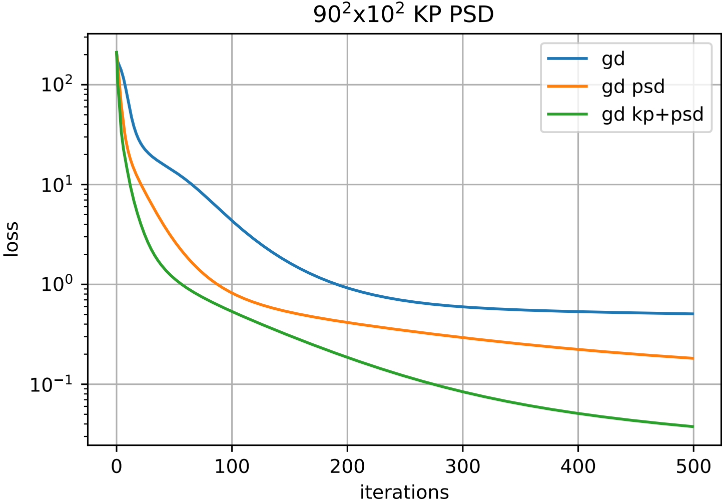
$$L = \|\delta U - Z\|^2 = \|\delta X \otimes Y + X \otimes \delta Y - Z\|^2 =$$
$$\|(\delta A A^T + A \delta A^T) \otimes BB^T + AA^T \otimes (\delta B B^T + B \delta B^T) - Z\|^2 \to \min_{\delta A, \delta B}$$

$$L = \langle \delta X, \langle Y, Y \rangle \delta X + X \langle Y, \delta Y \rangle - \nabla_X f \rangle + \mathrm{Const}_{\delta X} =$$
$$\left\langle \delta A A^T + A \delta A^T, \langle Y, Y \rangle \left(\delta A A^T + A \delta A^T \right) + AA^T \langle Y, \delta Y \rangle - \nabla_X f \right\rangle =$$
$$\left\langle \delta A A^T, \langle Y, Y \rangle \left(\delta A A^T + A \delta A^T \right) + AA^T \langle Y, \delta Y \rangle - \frac{1}{2} \left(\nabla_X f + \nabla_X f^T \right) \right\rangle$$

$$\delta A = \frac{1}{2 \langle Y, Y \rangle} \left(\frac{1}{2} \left(\nabla_A f + \nabla_A f^T \right) A^{-1} - AA^T \langle Y, \delta Y \rangle \right) A^{-T}$$
$$\delta B = \frac{1}{2 \langle X, X \rangle} \left(\frac{1}{2} \left(\nabla_B f + \nabla_B f^T \right) B^{-1} \right) B^{-T}$$

PSD Kronecker product optimization

After some quality time spent on obtaining the solution to the original problem, it works too:



References I

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] Pierre Dutilleul. The mle algorithm for the matrix normal distribution. *Journal of statistical computation and simulation*, 64(2):105–123, 1999.

[3] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[4] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

References II

[5] Dmitry Kropotov, Dmitry Vetrov, Lior Wolf, and Tal Hassner. Variational relevance vector machine for tabular data. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 79–94, 2010.

[6] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

[7] Zachary Nado, Jasper Snoek, Roger Grosse, David Duvenaud, Bowen Xu, and James Martens. Stochastic gradient langevin dynamics that exploit neural network structure. 2018.

[8] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*, 2018.

[9] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. 2018.