Variational Optimization of Neural Networks

Jakob Drachmann Havtorn

Presentation of Master's Thesis Friday 6th July, 2018

DTU Compute
Department of Applied Mathematics and Computer Science

Outline

- Introduction
  - Machine learning
  - Neural networks
  - Evolution strategies
- Variational optimization
  - Definition
  - Search distributions
- Natural gradient
  - Search space distance
  - Steepest descent w.r.t. a distance metric
- Variance reduction
  - Antithetic sampling
  - Local reparameterization

Motivation

Machine learning (ML)

- Explosion in interest in ML and neural networks (NNs)
  - Recognize objects in images
  - Transcribe and understand spoken language
  - Beat champions at games
- Driven by compute power + availability of data

Reinforcement learning (RL)

- Different approaches to RL: Policy gradients and value function methods
  - Distribution of rewards (feedback from game)
  - Action frequency
  - Long time horizons
  - Backpropagation
- variational optimization (VO) as alternative

Introduction

Machine learning

What?

- A subset of **artificial intelligence** in the field of **computer science**
- Use of **statistical techniques** to give computers the ability to **learn from data**
- Goal is solving of complex tasks **without explicit programming**

Supervised learning

- Data is **labelled** in some way $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$
- Learn mapping from $\mathbf{x}$ to y (e.g. handwritten digit $\rightarrow$ numeric value)

Unsupervised learning

- Data is **not labelled**: $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$
- Learn underlying structure or representation of data (e.g. clustering, visualization)

Reinforcement learning (RL)

- Data is **generated** by some complex interaction with an **environment**
- Learn optimal behaviour (e.g. how to play chess) based on maximizing **reward**

Neural networks

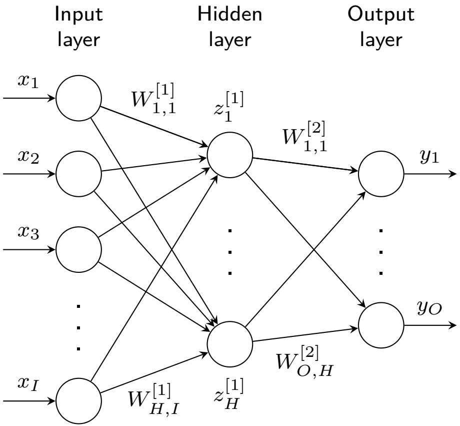


Figure: Feedforward neural network (FNN) with a single hidden layer and an output layer.

Neural networks

Why?

- Huge flexibility and representational power
- Can learn **representations and mappings** from raw data

How?

1. Forward pass: Evaluation of network on task and calculation of error, $f(\mathbf{x}, \mathbf{w})$
2. Backward pass: Change of parameters/weights in direction that reduces error

Backpropagation

$$\frac{\partial f_i}{\partial \mathbf{W}^{[l]}} = \underbrace{\frac{\partial E}{\partial \mathbf{a}^{[L]}} \frac{\partial \mathbf{a}^{[L]}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L-1]}} \frac{\partial \mathbf{a}^{[L-1]]}}{\partial \mathbf{z}^{[L-1]}} \cdots \frac{\partial \mathbf{z}^{[l+1]}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l]}}}_{\boldsymbol{\delta}^{[l]}} \frac{\partial \mathbf{z}^{[l]}}{\partial \mathbf{W}^{[l]}} \tag{1}$$

Neural networks

But!

- Backpropagation requires differentiable architecture
- If **nondifferentiability** is introduced, backpropagation does not work without modification or tricks
  - Nondifferentiable network (discrete latent variables, stochastic elements)
  - Nondifferentiable error (e.g. discrete actions/rewards as in RL)

This work

- How can gradients be estimated when the error (or network) is nondifferentiable?

Evolution strategies

Stochastic estimation of neural network gradient

- Taylor series gives a gradient estimate for any objective/error f dependent on parameters $\mathbf{w}$.
- With $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$f(\mathbf{w}+\boldsymbol{\epsilon}) \approx f(\mathbf{w}) + \boldsymbol{\epsilon}^{\mathrm{T}} \nabla_{\mathbf{w}} f(\mathbf{w}) + \frac{1}{2} \boldsymbol{\epsilon}^{\mathrm{T}} \mathbf{H}(\mathbf{w}) \boldsymbol{\epsilon} \tag{2}$$

$$\mathsf{E}\left[f(\mathbf{w}+\boldsymbol{\epsilon})\boldsymbol{\epsilon}\right] \approx \mathsf{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}}\right] \nabla_{\mathbf{w}}\, f(\mathbf{w}) = \nabla_{\mathbf{w}} f(\mathbf{w})$$

- With isotropic perturbation variance σ^2, the gradient estimator becomes

$$\nabla_{\mathbf{w}} f(\mathbf{w}) \approx \sigma^{-1} \mathsf{E}\left[f(\mathbf{w}+\sigma\boldsymbol{\epsilon})\boldsymbol{\epsilon}\right] \approx \frac{1}{N\sigma} \sum_{n=1}^{N} f(\mathbf{w}+\sigma\boldsymbol{\epsilon}_n)\boldsymbol{\epsilon}_n \tag{3}$$

- As in OpenAI's recent article [1] on evolution strategies (ESs).

Evolution strategies

Algorithm 1 Parallelized evolution strategy

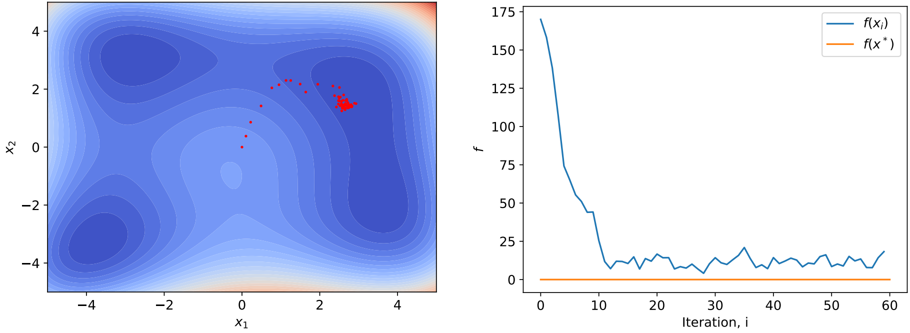
Require: Objective function $f(\mathbf{x})$, learning rate η, perturbation variance σ
Initialize: N workers with known random seeds

repeat
  for each central processing unit (CPU) $i = 1, \ldots, N$ **do** ▷ Parallelized
    Draw random seed s_i
    Sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Evaluate fitness $f(\mathbf{w} + \sigma \boldsymbol{\epsilon}_i)$
  end for
  Share N scalar fitnesses, $f(\mathbf{w} + \sigma \boldsymbol{\epsilon}_i)$ and seeds, s_i, between all CPUs.
  for each worker $i = 1, \ldots, N$ **do** ▷ Parallelized
    Reconstruct all perturbations $\boldsymbol{\epsilon}_j$ for $j = 1, \ldots, N$ using known random seeds.
    Compute gradient

$$\nabla_{\mathbf{w}} f(\mathbf{w}) \approx \frac{1}{N\sigma} \sum_{n=1}^{N} f(\mathbf{w} + \sigma \boldsymbol{\epsilon}_n) \boldsymbol{\epsilon}_n$$

    Update parameters $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f(\mathbf{w})$
  end for
until stopping criteria met

Himmelblau example (#1)



(a) **(b)**

Figure: **(a)** Convergence of the evolutionary strategy used by [1]. **(b)** Objective function value at each iteration of the algorithm. The algorithm finds a minimum but struggles to converge due to the fixed search distribution variance.

Evolution strategies

Gradient variance

- The variance of the gradient estimator is

$$\text{Var}\left[\nabla_{\mathbf{w}} f(\mathbf{w})\right] \approx \text{Var}\left[\frac{1}{N\sigma}\sum_{n=1}^{N} f(\mathbf{w}+\sigma\boldsymbol{\epsilon}_n)\boldsymbol{\epsilon}_n\right] = \frac{1}{N\sigma^2}\text{Var}[f(\mathbf{w}+\sigma\boldsymbol{\epsilon})\boldsymbol{\epsilon}]\ . \tag{4}$$

- In univariate case, for small ϵ or σ, $f(w+\sigma\epsilon) \approx f(w) + \sigma\epsilon f'(w)$ and

$$\begin{aligned} \text{Var}\left[f'(w)\right] &\approx \frac{1}{N\sigma^2}\text{Var}\left[f(w)\epsilon + \sigma\epsilon^2 f'(w)\right] \\ &= \frac{1}{N\sigma^2}\left(f(w)^2 + 2f'(w)^2\sigma^2\right) \\ &= \frac{1}{N\sigma^2} f(w)^2 + \frac{2}{N} f'(w)^2 \end{aligned} \tag{5}$$

Summary

- Variance of gradient explodes as $\sigma \to 0$
- σ must go to zero to precisely locate minima, but how?

Variational optimization

Formulation

Variational upper bound

- VO provides a more rigorous framework for evolutionary strategies [2].

$$f(\mathbf{w}^*) = \min_{\mathbf{w}} f(\mathbf{w}) \leq \mathsf{E}[f(\mathbf{w})]_{p(\mathbf{w}|\boldsymbol{\theta})} \equiv U(\boldsymbol{\theta}) \tag{6}$$

- Minimize the variational upper bound $\min_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$ rather than $f(\mathbf{w})$

Gradient of upper bound

- The VO upper bound is differentiable using the log-derivative trick

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathsf{E}[f(\mathbf{w})]_{p(\mathbf{w}|\boldsymbol{\theta})} \\
&= \nabla_{\boldsymbol{\theta}} \int f(\mathbf{w}) p(\mathbf{w}|\boldsymbol{\theta}) \mathrm{d}\mathbf{w} \\
&= \int f(\mathbf{w}) p(\mathbf{w}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{w}|\boldsymbol{\theta}) \mathrm{d}\mathbf{w} \\
&= \mathsf{E}[f(\mathbf{w}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{w}|\boldsymbol{\theta})]_{p(\mathbf{w}|\boldsymbol{\theta})}
\end{aligned} \tag{7}$$

Formulation

Algorithm 2 Parallelized variational optimization (VO). Adapted from [3]

Require: Objective function $f(\mathbf{x})$, learning rate η, search distribution $p(\mathbf{w}|\boldsymbol{\theta})$
Initialize: N workers with known random seeds

repeat
    for each CPU $i = 1, \dots, N$ **do** ▷ Parallelized
        Draw random seed s_i
        Sample $\mathbf{w}_i \sim p(\mathbf{w}|\boldsymbol{\theta})$
        Evaluate fitness $f(\mathbf{w}_i)$
    end for
    Share N scalar fitnesses, $f(\mathbf{w}_i)$ and seeds, s_i, between all CPUs.
    for each worker $i = 1, \dots, N$ **do** ▷ Parallelized
        Reconstruct all perturbations $\mathbf{w}_j$ for $j = 1, \dots, N$ using known random seeds.
        Compute search distribution and upper bound gradient

$$\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}_i) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{w}_i|\boldsymbol{\theta})$$

        Update search distribution parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$
    end for
until stopping criteria met

Augmentations

Network

- Any architecture (dense, convolutional, recurrent)
- Any training improvement technique (batch normalization, dropout, initialization)

Optimizer

- Any optimizer can be used on the gradients
- Optimizing in $d \gg N$ regime, always moving in subspace of network weight space
- Momentum works by reducing variance and remembering good subspace

VO algorithm

- Natural gradient
- Variance reduction
  - Antithetic sampling
  - Local reparameterization
- Other
  - Rescaling of perturbations by sensitivities
  - Importance mixing
  - Adaptation sampling

Search distributions

Univariate Gaussian

$$p(w|\theta) = \mathcal{N}\left(w|\mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(w-\mu)^2\right) \tag{8}$$

- Take log and derivatives w.r.t. μ and σ with $\epsilon \sim \mathcal{N}(0,1)$,

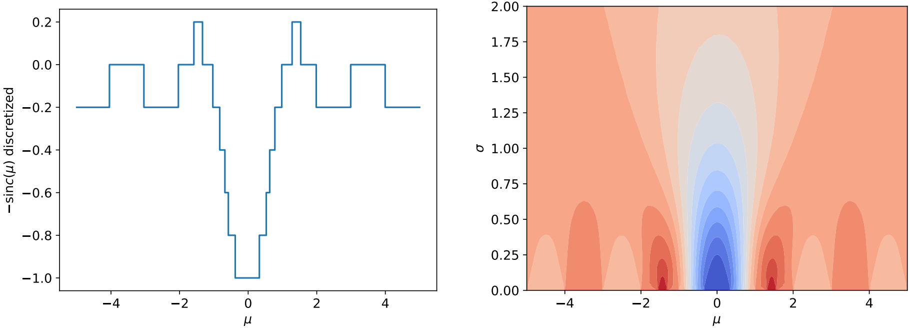
$$\begin{aligned} \frac{\partial}{\partial\mu}\log\mathcal{N}(w|\mu,\sigma^2) &= \frac{1}{\sigma^2}(w-\mu) = \frac{1}{\sigma}\epsilon \\ \frac{\partial}{\partial\sigma^2}\log\mathcal{N}(w|\mu,\sigma^2) &= -\frac{1}{2\sigma^2} + \frac{1}{4\sigma^4}(w-\mu)^2 = \frac{1}{\sigma^2}\left(\epsilon^2 - 1\right). \end{aligned} \tag{9}$$

- Use (7) to obtain the univariate Gaussian search gradient

$$\frac{\partial}{\partial\mu}U(\mu,\sigma^2) = \frac{1}{\sigma}\mathsf{E}[f(\mu+\sigma\epsilon)\epsilon] \approx \frac{1}{N\sigma}\sum_{n=1}^{N} f(\mu+\sigma\epsilon_n)\epsilon_n \tag{10}$$

$$\frac{\partial}{\partial\sigma^2}U(\mu,\sigma^2) = \frac{1}{2\sigma^2}\mathsf{E}\left[f(\mu+\sigma\epsilon)\left(\epsilon^2-1\right)\right] \approx \frac{1}{2N\sigma^2}\sum_{n=1}^{N} f(\mu+\sigma\epsilon_n)\left(\epsilon_n^2-1\right)$$

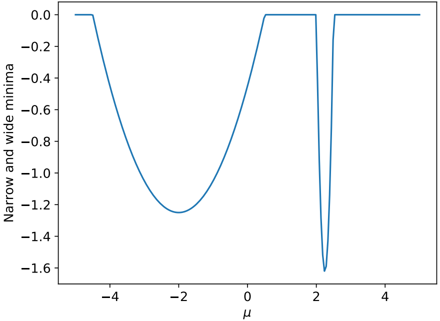
Nondifferentiable objective



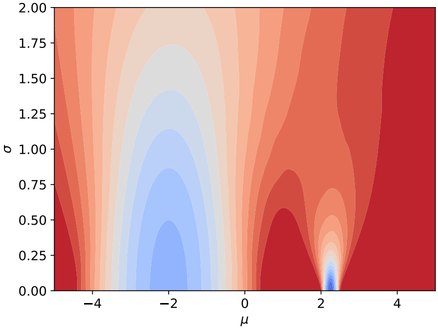
(a) **(b)**

Figure: Using a univariate Gaussian search distribution, the 1-dimensional discretized and nondifferentiable sinc function is turned into a 2-dimensional differentiable variational upper bound. The VO upper bound tends to nondifferentiability for $\sigma \to 0$. Figures inspired by [4].

Shallow and narrow minima



(a)



(b)

Figure: (a) A function with a low curvature local minimum and a high curvature global minimum. **(b)** The contours of the corresponding Gaussian VO objective. This illustrates the tendency of VO to prefer low curvature minima over high curvature minima if situated near each other. Figures inspired by [4].

Search distributions

Isotropic Gaussian

- Let $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Then

$$\nabla_{\boldsymbol{\mu}} U(\boldsymbol{\mu}, \sigma^2) \approx \frac{1}{N\sigma} \sum_{n=1}^{N} f(\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}_n)\boldsymbol{\epsilon}_n \tag{11}$$

$$\nabla_{\boldsymbol{\sigma}^2} U(\boldsymbol{\mu}, \sigma^2) \approx \frac{1}{2N\sigma^2} \sum_{n=1}^{N} f(\boldsymbol{\mu} + \sigma\boldsymbol{\epsilon}_n)\left(\boldsymbol{\epsilon}_n^2 - d\right)$$

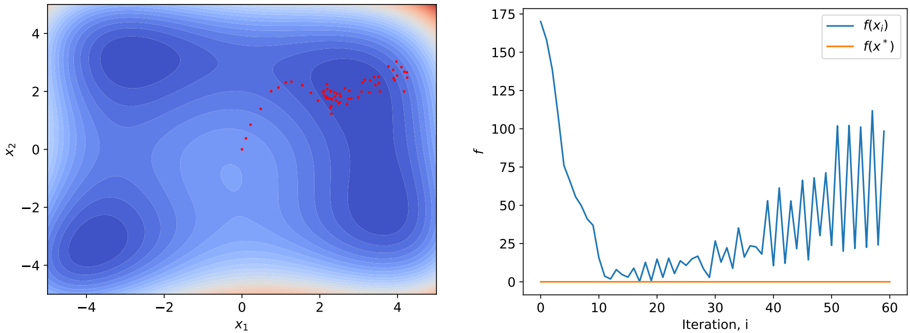
Separable Gaussian

- Let $\boldsymbol{\sigma}^2 = \begin{bmatrix}\sigma_1^2 & \sigma_2^2 & \cdots & \sigma_d^2\end{bmatrix}^{\mathrm{T}}$ and $\boldsymbol{\Sigma} = \mathrm{diag}\left(\boldsymbol{\sigma}^2\right)$. Then

$$\nabla_{\boldsymbol{\mu}} U(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \approx \frac{\boldsymbol{\sigma}^{-1}}{N} \odot \sum_{n=1}^{N} f(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_n)\boldsymbol{\epsilon}_n \tag{12}$$

$$\nabla_{\boldsymbol{\sigma}^2} U(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \approx \frac{\boldsymbol{\sigma}^{-2}}{2N} \odot \sum_{n=1}^{N} f(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_n)\left(\boldsymbol{\epsilon}^2 - 1\right)$$

Himmelblau example (#2)



(a) **(b)**

Figure: **(a)** Convergence of the VO algorithm with isotropic Gaussian search distribution using regular gradients. **(b)** Objective function value at each iteration. Similarly to fixed variance ES, a minimum is found, but the optimization of the variance drives it towards zero, resulting in larger gradients and variance.

Natural gradient

Motivation

Problems with regular search gradients

- **Cannot precisely locate any optimum** since gradient explodes for $\sigma \to 0$.
- Want to make small change to search distribution parameters

$$p(\mathbf{w}|\boldsymbol{\theta}) \leftarrow p(\mathbf{w}|\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \tag{13}$$

but regular gradient defines distance in Euclidean terms, $\sqrt{\Delta\boldsymbol{\theta}^{\mathrm{T}}\Delta\boldsymbol{\theta}}$, which is dependent on search distribution parameterization and inappropriate in high dimensions.

- Is it possible to obtain a gradient that is invariant to parameterization, i.e. **do gradient descent with respect to an invariant measure of the closeness of the current distribution and the updated distribution**?

Distance measure

Kullback-Leibler (KL)

- Measures distance between probability density functions (PDFs) [5]

$$\begin{aligned} \mathsf{KL}(p||q) &\equiv -\int p(\mathbf{w}) \log q(\mathbf{w}) \, \mathbf{dw} + \int p(\mathbf{w}) \log p(\mathbf{w}) \, \mathbf{dw} \\ &= -\int p(\mathbf{w}) \log \left(\frac{q(\mathbf{w})}{p(\mathbf{w})} \right) \mathbf{dw} \end{aligned} \tag{14}$$

- It can be symmetrized and approximated by Taylor series

$$\mathsf{KL}(p(\mathbf{w}|\theta), p(\mathbf{w}|\theta + \Delta\theta)) \approx \frac{1}{2} \Delta\theta^{\mathrm{T}} \mathbf{F}_\theta \Delta\theta \tag{15}$$

Fisher information matrix

$$\mathbf{F}_\theta \equiv \mathsf{E}\left[\nabla_\theta \log p(\mathbf{w}|\theta) \nabla_\theta \log p(\mathbf{w}|\theta)^{\mathrm{T}}\right] \tag{16}$$

Steepest descent w.r.t. a distance metric

Fixing the per iteration search distribution change

- Each minimization step $\Delta\theta$ on the variational upper bound $U(\theta)$ can be written as a Taylor expansion

$$U(\theta + \Delta\theta) \approx U(\theta) + \Delta\theta^{\mathrm{T}} \nabla_{\theta} U(\theta) \tag{17}$$

- The search gradient can then be found by minimizing the variational upper bound while keeping the KL divergence fixed to a small constant, κ.

$$\begin{aligned} &\min_{\theta} U(\theta + \Delta\theta) \approx U(\theta) + \Delta\theta^{\mathrm{T}} \nabla_{\theta} U(\theta) \\ &\text{s.t. } \mathrm{KL}(p(\mathbf{w}|\theta), p(\mathbf{w}|\theta + \Delta\theta)) \approx \frac{1}{2} \Delta\theta^{\mathrm{T}} \mathbf{F}_{\theta} \Delta\theta = \kappa \end{aligned} \tag{18}$$

- This has the so-called natural gradient as solution (search direction)

$$\tilde{\nabla}_{\theta} U(\theta) = \alpha \mathbf{F}_{\theta}^{-1} \nabla_{\theta} U(\theta) \tag{19}$$

Natural gradient for univariate Gaussian

- Fischer information matrix can be found analytically

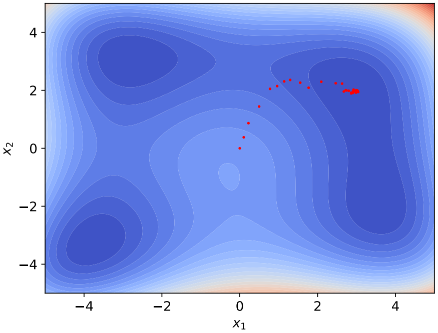
$$\mathbf{F}_{\boldsymbol{\theta}} = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{2\sigma^4} \end{bmatrix} \iff \mathbf{F}_{\boldsymbol{\theta}}^{-1} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & 2\sigma^4 \end{bmatrix} \tag{20}$$

- The gradient is scaled by $\mathbf{F}_{\boldsymbol{\theta}}^{-1}$

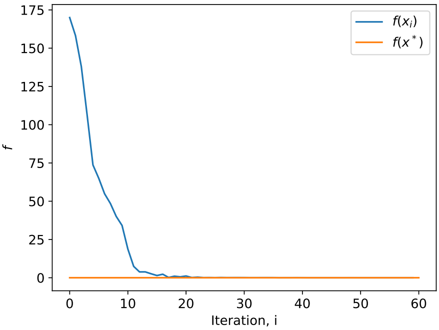
$$\begin{aligned} \sigma^2 \frac{\partial}{\partial \mu} U(\mu, \sigma^2) &= \sigma \mathsf{E}[f(\mu + \sigma\epsilon)\epsilon] \approx \frac{\sigma}{N} \sum_{n=1}^{N} f(\mu + \sigma\epsilon_n)\epsilon_n \\ 2\sigma^4 \frac{\partial}{\partial \sigma^2} U(\mu, \sigma^2) &= \mathsf{E}\left[f(\mu + \sigma^2\sigma\epsilon)\left(\epsilon^2 - 1\right)\right] \approx \frac{\sigma^2}{N} \sum_{n=1}^{N} f(\mu + \sigma\epsilon_n)\left(\epsilon_n^2 - 1\right) \end{aligned} \tag{21}$$

- This can be done similarly for the isotropic and separable Gaussians

Himmelblau example 3



(a)



(b)

Figure: (a) Convergence of the variational optimization algorithm with isotropic Gaussian search distribution using natural gradients. **(b)** Objective function value at each iteration. A minimum is found and the optimization of the variance drives the gradients toward zero resulting in convergence to the optimum.

Variance reduction

Antithetic sampling

Shorthand notation and odd/even decomposition

- Let $g(\mathbf{w}) = f(\mathbf{w})\nabla_{\boldsymbol{\theta}} \log p(\mathbf{w}|\boldsymbol{\theta})$ so $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} g(\mathbf{w})$
- Any function $g(\mathbf{w}) = g_e(\mathbf{w}) + g_o(\mathbf{w})$ where g_e and g_o are even and odd parts

Leveraging covariance between perturbations

- Rather than sampling $\boldsymbol{\epsilon}_n$ IID, take every other to be $-\boldsymbol{\epsilon}_i$ and decompose g

$$\nabla_{\boldsymbol{\theta}} U_a(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N/2} g(\mathbf{w}_n) + g(-\mathbf{w}_n) = \frac{2}{N}\sum_{n=1}^{N/2} g_e(\mathbf{w}_i) \tag{22}$$

- Due to this trick, there is again zero covariance and the variance simplifies

$$\text{Var}[\nabla_{\boldsymbol{\theta}} U_a(\boldsymbol{\theta})] = \frac{2}{N}\text{Var}[g_e(\mathbf{w})] \tag{23}$$

- Compared to the IID sampling result (4), antithetic sampling trades in the variance of g for two times the variance of g_e.
- If g_e is zero everywhere then $\text{Var}[\nabla_{\boldsymbol{\theta}} U_a(\boldsymbol{\theta})] = 0$
- If g_o is zero everywhere then $\text{Var}[\nabla_{\boldsymbol{\theta}} U_a(\boldsymbol{\theta})] = 2\text{Var}[\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})]$

Results for antithetic sampling

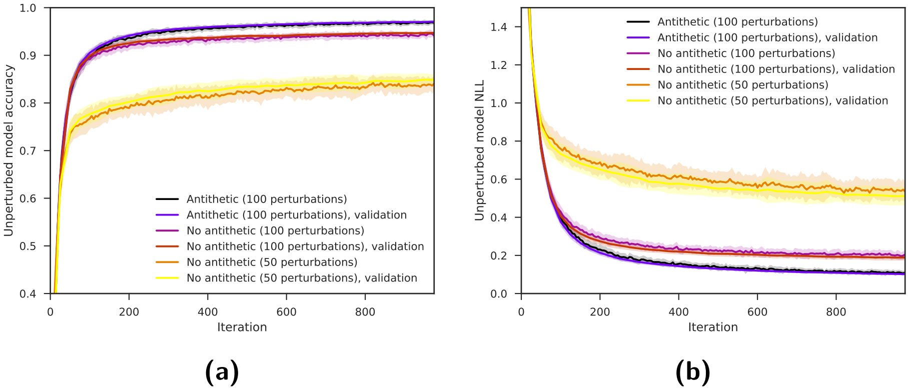


Figure: Results of training a neural network with VO and antithetic sampling on handwritten digit recognition. **(a)** Training and validation set classification accuracy. **(b)** Training and validation set negative log-likelihood (NLL) loss.

Local reparameterization

Introducing the mini-batch

- When the objective/error is a sum of individual terms from a mini-batch $\mathcal{B}$

$$U(\boldsymbol{\theta}) \approx \frac{1}{N|\mathcal{B}|} \sum_{n=1}^{N} \sum_{b \in \mathcal{B}} f_b(\mathbf{w}_n) \tag{24}$$

- This introduces covariance between batch examples (despite IID perturbations)

$$\mathrm{Var}[U(\boldsymbol{\theta})] \approx \frac{1}{N|\mathcal{B}|}\mathrm{Var}[f_b(\mathbf{w})] + \frac{N-1}{N}\frac{|\mathcal{B}|-1}{|\mathcal{B}|}\mathrm{Cov}[f_b(\mathbf{w}), f_{b'}(\mathbf{w})] \tag{25}$$

Removing within-batch covariance

- By sampling new weights for each batch example, $\mathbf{w}_b$, this covariance vanishes [6]

$$\mathrm{Var}\left[\tilde{U}(\boldsymbol{\theta})\right] \approx \frac{1}{N|\mathcal{B}|}\mathrm{Var}[f_b(\mathbf{w}_b)] \tag{26}$$

Local reparameterization

Upper bound gradient

- The gradient is similar to before but now summed over the mini-batch as well

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}) \approx \frac{1}{N|\mathcal{B}|} \sum_{n=1}^{N} \sum_{b \in \mathcal{B}} f_b(\mathbf{w}_{bn}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{w}_{bn}|\boldsymbol{\theta}) \tag{27}$$

- Computationally inefficient due to new weights for each batch example

Propagating a distribution over activations

- Can infer distribution of activations from distribution of weights in FNN

$$q\left(Z_{ib}^{[l]} \,\middle|\, \mathbf{A}^{[l-1]}, \boldsymbol{\theta}\right) = \mathcal{N}\left(Z_{ib}^{[l]} \,\middle|\, \sum_j \mu_{ij}^{[l]} A_{jb}^{[l-1]}, \sum_j {\sigma_{ij}^{[l]}}^2 {A_{jb}^{[l-1]}}^2\right) \tag{28}$$

- Here, $\mathbf{Z}^{[l]} = \mathbf{W}^{[l]}\mathbf{A}^{[l-1]}$, $\mathbf{A}^{[l]} = \varphi(\mathbf{Z}^{[l]})$ and $p(\mathbf{W}|\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{W}|\boldsymbol{\mu}, \sigma^2\mathbf{I}\right)$

Local reparameterization

New gradient

- Gradient is then obtained by perturbing the activation space

$$\nabla_{\boldsymbol{\theta}} \tilde{U}(\theta) \approx \frac{1}{N|\mathcal{B}|} \sum_{n=1}^{N} \sum_{b \in \mathcal{B}} f_b(\mathbf{Z}_{bn}) \nabla_{\boldsymbol{\theta}} \log q(\mathbf{Z}_{bn} | \mathbf{A}_{bn}, \boldsymbol{\theta}) \tag{29}$$

Gradient in FNN

- Specifically, for an FNN

$$\begin{aligned} \frac{\partial \tilde{U}(\boldsymbol{\theta})}{\partial \mu_{ij}^{[l]}} &\approx \frac{1}{N|\mathcal{B}|} \sum_{n=1}^{N} \sum_{b \in \mathcal{B}} f_b(\mathbf{Z}_{bn}) \frac{\xi_{ibn}}{\sqrt{v_{ib}^{[l]}}} A_{jb}^{[l-1]} \\ \frac{\partial \tilde{U}(\boldsymbol{\theta})}{\partial \sigma_{ij}^{[l]^2}} &\approx \frac{1}{N|\mathcal{B}|} \sum_{n=1}^{N} \sum_{b \in \mathcal{B}} f_b(\mathbf{Z}_{bn}) \frac{\xi_{ibn}^2 - 1}{2 v_{ib}^{[l]}} A_{jb}^{[l-1]^2} \quad . \end{aligned} \tag{30}$$

where $Z_{ibn} = m_{ib} + \sqrt{v_{ib}}\, \xi_{ibn}$, with $\xi_{ibn} \sim \mathcal{N}(0, 1)$

Local reparameterization

Forward pass in FNN

- With $\mathbf{A}^{[0]} = \mathbf{X}$ as a batch of inputs, the forward pass in an FNN is

$$\mathbf{m}^{[l]} = \boldsymbol{\mu}^{[l]} \mathbf{A}^{[l-1]} \tag{31a}$$

$$\mathbf{v}^{[l]} = \left(\boldsymbol{\sigma}^{[l]} \mathbf{A}^{[l-1]}\right)^2 \tag{31b}$$

$$\mathbf{Z}^{[l]} = \mathbf{m}^{[l]} + \sqrt{\mathbf{v}^{[l]}} \odot \boldsymbol{\xi}^{[l]} \tag{31c}$$

$$\mathbf{A}^{[l]} = \varphi\left(\mathbf{Z}^{[l]}\right) \tag{31d}$$

- Forward propagation of a distribution over activations

Conclusion

Conclusion

- Variational optimization
- Natural gradient
- Variance reduction

Other topics

- Adapting the variance
- Sensitivity rescaled perturbations
- Reuse of samples (importance mixing)
- Adapting hyperparameters by adaptation sampling
- Fitness transforms

Future work

- How does local reparameterization fare in practice?
- Is there a way around the problems with adapting the variance? Separable Gaussian did not harm performance
- How to compute importance weights in high dimensional spaces in order to use importance mixing and adaptation sampling?

References

- T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning", *arXiv preprint*, 2017. arXiv: 1703.03864. [Online]. Available: http://arxiv.org/abs/1703.03864.
- J. Staines and D. Barber, "Variational Optimization", *arXiv preprint*, Dec. 2012. arXiv: 1212.4507. [Online]. Available: http://arxiv.org/abs/1212.4507.
- D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural Evolution Strategies", *Journal of Machine Learning Research*, vol. 15, pp. 3381–3387, 2008, ISSN: 15337928. DOI: 10.1109/CEC.2008.4631255. arXiv: 1106.4487. [Online]. Available: http://www.jmlr.org/papers/volume15/wierstra14a/wierstra14a.pdf.
- F. Huszár, *Evolution Strategies, Variational Optimisation and Natural ES*, 2017. [Online]. Available: http://www.inference.vc/evolution-strategies-variational-optimisation-and-natural-es-2/ (visited on Mar. 23, 2018).
- S. Kullback and R. A. Leibler, "On Information and Sufficiency", *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951, ISSN: 0003-4851. DOI: 10.1214/aoms/1177729694. arXiv: 1511.00860. [Online]. Available: http://projecteuclid.org/euclid.aoms/1177729694.
- D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick", *arXiv preprint*, Jun. 2015. arXiv: 1506.02557. [Online]. Available: http://arxiv.org/abs/1506.02557.

Abbreviations

CPU central processing unit. 10, 15

ES evolution stategy. 9, 21

FNN feedforward neural network. 6, 32, 33, 34

KL Kullback-Leibler. 24, 25

ML machine learning. 3

NLL negative log-likelihood. 30

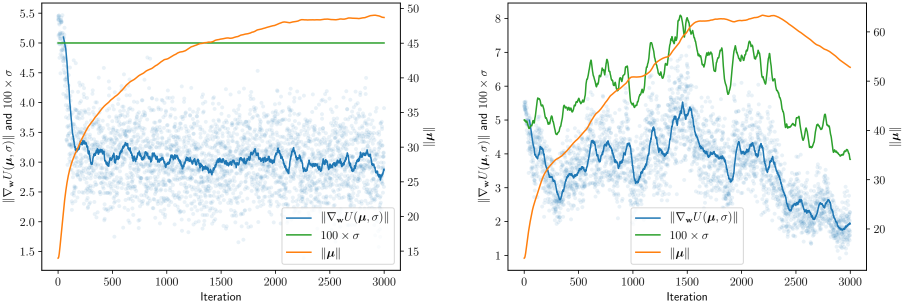
NN neural network. 3, 38, 39

PDF probability density function. 24

RL reinforcement learning. 3, 5, 8

VO variational optimization. 3, 14, 15, 16, 18, 19, 21, 30, 38, 39

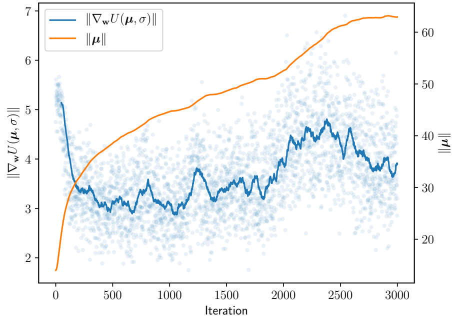
Adapting the variance



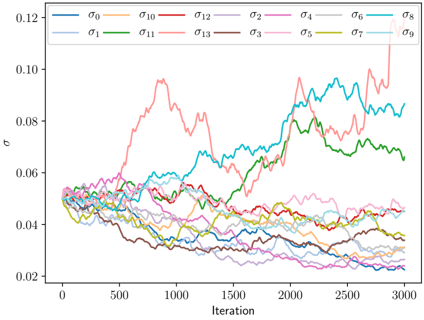
(a) **(b)**

Figure: 2-norms of the NN parameter vector and VO gradient for an isotropic Gaussian search distribution with the variance overlayed (multiplied by 100 for scale). In **(a)** and **(b)**, the fixed and adapted variance versions are shown, respectively. A centered 50 sample moving average is computed for the gradient. It is clear that adapting the variance directly and significantly influences the norm of the gradient and in turn also the norm of the parameter vector.

Adapting the variance



(a)



(b)

Figure: (a) 2-norms of the NN parameter vector and VO gradient for a layer-wise separable Gaussian search distribution with the variances plotted separately in **(b)**. Most variances tend to zero while a few increase. The gradient norm feels the combined effect but generally increases.

Sensitivity rescaled perturbations

Notation

- For a mini-batch of I inputs $\mathbf{X} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_I\right]$ let $\mathbf{Y}(\mathbf{X}, \mathbf{w}) = NN(\mathbf{X}|\mathbf{w})$
- $Y_{ki} = NN(X_{:,i}|\mathbf{w})_k$ equals value of k'th output unit for the i'th batch example

Network divergence

- Divergence in network output units as result of additive perturbation $\boldsymbol{\epsilon}$

$$D(\boldsymbol{\epsilon}|\mathbf{w}) = \frac{1}{I}\sum_{k=1}^{K}\sum_{i=1}^{I}\left(NN(\mathbf{X}|\mathbf{w})_{ki} - NN(\mathbf{X}|\mathbf{w}+\boldsymbol{\epsilon})_{ki}\right)^2 \tag{32}$$

- Perturbations that lead to large divergence risk **catastrophic forgetting**

Rescaled perturbations

- Perturbations can be rescaled to correct for element-wise influence on divergence

$$\boldsymbol{\epsilon}_{\text{safe}} = \frac{\boldsymbol{\epsilon}}{\mathbf{s}}, \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{\theta}) \tag{33}$$

with $\mathbf{s}$ computed in some appropriate way (line search, gradients)

Sensitivity rescaled perturbations

Network output gradients

- By Taylor expansion of outputs around $\mathbf{w}$, $\nabla_{\mathbf{w}} NN(\mathbf{x}_i|\mathbf{w})_k$ can be seen as a point estimate of the sensitivity of the k'th output unit to changes in weights.

$$Y_{ki}(\boldsymbol{\epsilon}|\mathbf{w}) \approx NN(\mathbf{x}_i|\mathbf{w})_k + \boldsymbol{\epsilon}\nabla_{\mathbf{w}} NN(\mathbf{x}_i|\mathbf{w}) \tag{34}$$

- For a single output unit k, these can be averaged over a mini-batch of inputs

$$\frac{1}{I}\sum_{i=1}^{I} |\nabla_{\mathbf{w}} NN(\mathbf{X}|\mathbf{w})_{ki}| \quad \text{or} \quad \frac{1}{I}\sum_{i=1}^{I} \nabla_{\mathbf{w}} NN(\mathbf{X}|\mathbf{w})_{ki}$$

- To handle the K output units, the Euclidean length of the K dimensional "vector" of sensitivities is taken to form $\mathbf{s}_{\text{abs}}$ or $\mathbf{s}_{\text{sum}}$, respectively

$$\sqrt{\sum_{k=1}^{K}\left(\frac{1}{I}\sum_{i=1}^{I} |\nabla_{\mathbf{w}} NN(\mathbf{X}|\mathbf{w})_{ki}|\right)^2} \quad \text{or} \quad \sqrt{\sum_{k=1}^{K}\left(\sum_{i=1}^{I} \nabla_{\mathbf{w}} NN(\mathbf{X}|\mathbf{w})_{ki}\right)^2} \tag{35}$$

- $\mathbf{s}_{\text{abs}}$ is avoids gradient washout (absolute value) but $\mathbf{s}_{\text{sum}}$ is much more efficient