Natural Evolution Strategies

0.1 摘要

本文介绍了自然进化策略（NES），这是一个最新的黑盒优化算法家族，它使用自然梯度沿更高期望适应度的方向更新参数化搜索分布。我们介绍了一系列解决收敛性、鲁棒性、样本复杂性、计算复杂性和超参数敏感性问题的技术。本文探讨了NES系列的一些实现，例如针对高维空间搜索的通用多变量正态分布和可分离分布。实验结果显示，在各种标准基准上，发布的性能最好，在其他基准上，也有竞争性的性能

关键词：natural gradient, stochastic search, evolution strategies, black-box optimization, sampling

1. Introduction

许多现实世界的优化问题过于困难或复杂，无法直接建模。因此，最好以“黑盒子”的方式解决这些问题，除了在参数空间的某些点进行适应性评估外，黑盒优化是不需要关于需要优化的目标函数（即“适应性”或“成本”）的额外信息的。属于这一类别的问题很多，从健康和科学应用（Winter等人，2005年；Shir和Bäck，2007年；Jebalia等人，2007年）到航空设计（Hasenjäger等人，2005年;Klockgetherand Schwefel，1970年）和控制（Hansen等人，2009年）。

在过去的五十年中，已经开发并应用了许多这方面的算法，在许多情况下，这些算法为艰巨的任务提供了良好甚至接近最优的解决方案，没有他们，这些问题就需要领域专家以巨大的成本手工制作解决方案，而且往往会产生更糟糕的结果。由于几乎不可能找到全局最优解，在黑箱优化算法中产生了大量的启发式算法，从而导致了复杂但经常具有高性能的方法的普及。

在本文中，我们介绍了自然进化策略（NES），这是一种新的黑盒优化框架，它具有相对干净的推导过程，但实现了最先进的性能（借助于精心选择的启发式方法）。与分布估计算法（EDA）的框架（Mühlenbein和Paass，1996；Larrãnaga，2002；Pelikan等人，2000）和许多进化策略方法（如Os termeier等人，1994）类似，核心思想是维护和迭代更新搜索分布，从中提取搜索点并随后进行评估。然而，NES使用自然梯度沿更高期望适应度的方向更新搜索分布（例如，EDAs【分布估计算法 】通常使用最大似然法来拟合搜索点的分布）

1.1 连续黑盒优化

黑盒优化问题产生了各种各样的方法。第一类方法受到经典优化方法的启发，包括单纯形方法，如Nelder-Mead（Nelder和Mead，1965），以及拟牛顿算法族的成员。模拟退火（Kirkpatrick et al.，1983）是1983年引入的一种流行方法，其灵感来自热力学，实际上是Metropolis-Hastings算法的一种改编。其他方法，如受进化启发的方法，从20世纪50年代初

开始发展起来。其中包括广泛的遗传算法（Holland，1975；Goldberg，1989）、差分进化（Storn和Price，1997）、分布估计算法（Larrãnaga，2002；Pelikan等人，2000；Bosman和Thierens，2000；Bosman等人，2007；Pelikan等人，2006）、粒子群优化（Kennedy andEberhart，2001），以及交叉熵方法（Rubinstein和Kroese，2004）

【上面这些是常见的黑盒优化算法】

【下面介绍一个不典型的后面会与NES比较的黑盒优化算法】

20世纪60年代和70年代，Ingo-Rechenberg和Hans-Paul Schwefel引入了进化策略（Rechenberg和Eigen，1973；Schwefel，1977），旨在应对高维连续值域，四十多年来一直是一个活跃的研究领域（Beyer和Schwefel，2002）。ESs涉及评估批次（“世代”）中真实值基因型的适合度，之后保留最佳基因型，而丢弃其他基因型。幸存者然后繁殖（通过稍微改变他们所有的基因）以产生下一批后代。这一过程经过几代人的发展，对于许多困难的优化问题，都可以得到从合理到优秀的结果。多年来，该算法框架得到了广泛的发展，通过使用全协方差矩阵来表示相关突变。【上面介绍ES的一个过程】这就使得框架能够利用协方差来捕获相互关联的依赖关系，与此同时，为下一代产生变异的个体。【ES效果】 最终的算法，协方差矩阵适应进化策略（**CMA-ES**；Hansen和Ostermeier，2001），在许多研究中取得了成功（例如，Friedrichs和Igel，2005；Muller等人，2002；Shepherdet等人，2006）。虽然进化策略在黑箱优化方面已被证明是有效的，但分析程序的实际动态结果却很困难，尽管有各种研究人员的不懈努力（Beyer，2001；Jagerskupper，2007；Jebaliaet al.，2010；Auger，2005；Schaul，2012f）。

1.2 NES族

自然进化策略（NES）是一类进化策略，它通过估计分布参数的梯度来迭代更新搜索分布。

一般步骤如下：使用参数化搜索分布生成一批搜索点，并在每个搜索点处计算适应度函数。分布参数（包括策略参数）允许算法自适应地捕获适应度函数的（局部）结构。例如，在高斯分布的情况下，这包括平均值和协方差矩阵。从样本中，NES估计参数上的搜索梯度，以达到更高的预期适应度。然后，NES沿着自然梯度执行梯度上升步骤，这是一种二阶方法，与普通梯度不同，它将重新规范化关于不确定性的更新。这一步骤至关重要，因为它可以防止给定参数化带来的振荡、过早收敛和不希望出现的效果（参见第2.3节和图2，了解自然梯度如何解决这些问题的概述）。整个过程重复进行，直到达到惊人的标准。

“NES族”的所有成员都基于相同的原则运作。它们在分布类型和使用的梯度近似方法上有所不同。不同的搜索空间需要不同的搜索分布；例如，在低维情况下，对全协方差矩阵建模可能非常有益。另一方面，在高维中，一种更具可伸缩性的替代方法是将协方差仅限制在对角线上。此外，高度多模态的搜索空间可能受益于更多的重尾分布（如Cauchy分布，与高斯分布相反）。最后一个区别在于，我们可以通过分析计算自然梯度的分布，和我们需要从样本中估计自然梯度的更一般的分布。

1.3 下文思路

本文基于并扩展了我们之前对自然进化策略的研究（Wierstra et al.，2008；Sun et al.，2009 a，b；Glasmachers et al.，2010 a，b；Schaul et al.，2011），其结构如下：第2节介绍了Wierstra et al.（2008）中描述的搜索梯度的一般思想，使用参数化分布解释随机搜索，同时向更高的期望适应度梯度上升。第2.2节揭示了朴素梯度的局限性，随后通过引入自然梯度（第2.3节）解决了该局限性，从而得出了规范的NES算法。

然后，第3节重新组合了一系列增强NES性能和可靠性的技术。这包括适应度整形（设计用于呈现关于保序适应度变换的算法不变性（Wierstra et al.，2008），第3.1节），以及自适应采样，这是一种在线调整学习率的新技术（第3.2节）。我们为具有旋转对称性的全类多变量分布提供了一种新的NES公式（第3.3节）。作为特例，我们总结了多元高斯搜索分布的技术，构成了最常见的情况（第3.4节）。最后，在第3.5节中，我们开发了框架的广度，激发了它的有用性，并推导出了许多具有不同搜索分布的NES变体。

随后的实验调查表明，该方法在国外基准范围内具有竞争力（第5节）。本文最后讨论了不同技术和分布类型的有效性，并对未来的发展进行了展望（第6节）。

2. 搜索梯度

自然进化策略的核心思想是使用搜索梯度（首次引入Berny，2000，2001）来更新搜索分布的参数。我们将搜索梯度定义为样本中 期望适应度 的梯度。搜索分布可以被视为一个多正态分布，但原则上可以是我们可以找到其log密度关于其参数导数的任何分布。例如，有用的分布包括高斯混合模型和带有重尾的柯西分布。

如果我们用θ表示密度π（z |θ）的参数，用f（z）表示样本z的适应度函数，我们可以将搜索分布下的期望拟合度写为

$$J(\theta) = \mathbb{E}_\theta[f(\mathbf{z})] = \int f(\mathbf{z})\, \pi(\mathbf{z}\,|\,\theta)\, d\mathbf{z}. \qquad (1)$$

The so-called 'log-likelihood trick' enables us to write

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int f(\mathbf{z})\, \pi(\mathbf{z}\,|\,\theta)\, d\mathbf{z} \\
&= \int f(\mathbf{z})\, \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\, d\mathbf{z} \\
&= \int f(\mathbf{z})\, \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\, \frac{\pi(\mathbf{z}\,|\,\theta)}{\pi(\mathbf{z}\,|\,\theta)}\, d\mathbf{z} \\
&= \int \Big[f(\mathbf{z})\, \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)\Big]\, \pi(\mathbf{z}\,|\,\theta)\, d\mathbf{z} \\
&= \mathbb{E}_\theta \left[f(\mathbf{z})\, \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)\right].
\end{aligned}$$

【这边有这样的一个逻辑：所谓的搜索分布下的对适合度函数的期望，这个期望的梯度方向就是我们更新的方向，期望这个点怎么求？用样本来近似期望（涉及到定义）】

我们找到了更新方向，利用梯度下降法更新即可

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta),$$

where η is a learning rate parameter. Algorithm 1 provides the pseudocode for this very general approach to black-box optimization by using a search gradient on search distributions.

Algorithm 1: Canonical Search Gradient algorithm

input: f, θ_{init}
repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$
    evaluate the fitness $f(\mathbf{z}_k)$
    calculate log-derivatives $\nabla_\theta \log \pi(\mathbf{z}_k|\theta)$
  end
  $\nabla_\theta J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\theta \leftarrow \theta + \eta \cdot \nabla_\theta J$
until *stopping criterion is met*

在这个框架中使用搜索梯度类似于进化策略，因为它以迭代方式生成ES所谓的候选者解决方案中批量向量值样本的fitnesses。然而，它是不同的，因为它将这个“总体”表示为一个参数化分布，并且它使用搜索梯度来更新这个分布的参数，搜索梯度是使用fitnesses计算的

2.1 Search Gradient for Gaussian Distributions

在“default”d-维多变量正态分布的情况下，高斯分布的参数是平均值$\mu \in R^d$（候选解中心）和协方差$\sum \in R^{d \times d}$（突变矩阵）。让θ表示这些参数：θ=〈μ，∑〉。为了有效地从这个分布中取样，我们需要协方差矩阵的平方根，balabala,就是把分布标准化一下，（或者说把样本标准化一下）

$$\begin{aligned} \pi(\mathbf{z}\,|\,\theta) &= \frac{1}{(\sqrt{2\pi})^d |\det(\mathbf{A})|} \cdot \exp\left(-\frac{1}{2}\left\|\mathbf{A}^{-1} \cdot (\mathbf{z} - \boldsymbol{\mu})\right\|^2\right) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \cdot \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu})\right) \end{aligned}$$

denote the density of the multinormal search distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

标准化是为了方便后求相关的梯度啥的应该，其实多元Gauss分布的例子没什么不同的地方，Algorithm 2只是Algorithm 1的一个特例罢了。

Algorithm 2: Search Gradient algorithm: Multinormal distribution

input: f, $\boldsymbol{\mu}_{init}$,$\boldsymbol{\Sigma}_{init}$
repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{z}_k \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
    evaluate the fitness $f(\mathbf{z}_k)$
    calculate log-derivatives:
    $\nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}_k|\theta) = \boldsymbol{\Sigma}^{-1}(\mathbf{z}_k - \boldsymbol{\mu})$
    $\nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}_k|\theta) = -\frac{1}{2}\boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{z}_k - \boldsymbol{\mu})(\mathbf{z}_k - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}$
  end
  $\nabla_{\boldsymbol{\mu}} J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\nabla_{\boldsymbol{\Sigma}} J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta \cdot \nabla_{\boldsymbol{\mu}} J$
  $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \eta \cdot \nabla_{\boldsymbol{\Sigma}} J$
until *stopping criterion is met*

2.2 Limitations of Plain Search Gradients

Plain的搜索梯度法是有很大缺陷的，显然，即使在一维情况下，也不可能准确地定位（二次）最优值。【至少我还没感觉出来很显然 =（】又把上面Gauss的例子一维化说了下，【没看懂说实话】

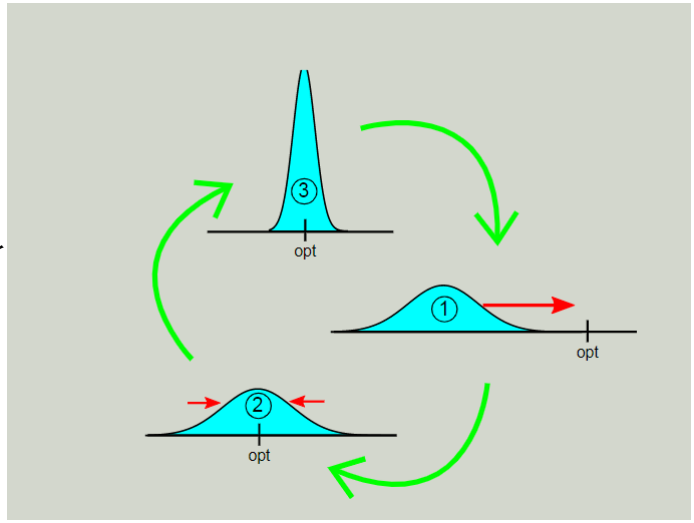
$$\nabla_\mu J = \frac{z-\mu}{\sigma^2},$$
$$\nabla_\sigma J = \frac{(z-\mu)^2-\sigma^2}{\sigma^3},$$

and the updates, assuming simple hill-climbing (i.e., a population size $\lambda = 1$) read:

$$\mu \leftarrow \mu + \eta\frac{z-\mu}{\sigma^2},$$
$$\sigma \leftarrow \sigma + \eta\frac{(z-\mu)^2-\sigma^2}{\sigma^3}.$$

反正就是说，以正态分布为例，需要更新的参数是σ, μ,但是$\Delta\mu, \Delta\sigma$都是反比于σ的，这个从上面一维的更新公式可以看出，这就说明了几点问题：

1.当更新一段时间后，σ会小于1，这样很容易导致在更新一次，就会更新极大一步，导致更新过头，就是下图3到1的过程了



这样的更新是不稳定的，更来更去白费功夫，虽然我们可以通过降低lr，增加size of samples，但这也只是减缓，不能avoid。

2、当一开始的σ很大，远大于一的时候，每次都更新的很慢

对于一维的高斯分布，梯度控制着分布在同一搜索空间维度上的位置和方差，【多维可能也是】，虽然这是一种特例，但是仍然是很普遍的，对于所有平移和缩放不变的搜索分布族都是通用的。

这种不稳定性造成 search gradient法不普遍被使用。效果不好肯定没人开发使用啊。

【接下来就自然而然的引出NES了。。。】

首先，natural gradient 法 通过使 关于所使用的特定参数的 更新 保持不变 来将search gradient 变成一个可行的优化问题

2.3 Using the Natural Gradient

最后得到的就是一个自然梯度的算法

2.3.1 【讲Fisher Information Matrix的，应该不错】

链接：https://agustinus.kristia.de/techblog/2018/03/11/fisher-information/

- 我们会用最大似然估计来估计 $p(x|\theta)$ 中的 θ，估计嘛，就要评价估计估计值 θ 的好坏，然后定义最大似然估计函数的梯度为score function

$$s(\theta) = \nabla_\theta \log p(x|\theta)\,,$$

易证$E_{p(x|\theta)}(s(\theta)) = 0$

$$\begin{aligned}
\mathop{\mathbb{E}}_{p(x|\theta)}[s(\theta)] &= \mathop{\mathbb{E}}_{p(x|\theta)}[\nabla \log p(x|\theta)] \\
&= \int \nabla \log p(x|\theta)\, p(x|\theta)\, \mathrm{d}x \\
&= \int \frac{\nabla p(x|\theta)}{p(x|\theta)} p(x|\theta)\, \mathrm{d}x \\
&= \int \nabla p(x|\theta)\, \mathrm{d}x \\
&= \nabla \int p(x|\theta)\, \mathrm{d}x \\
&= \nabla 1 \\
&= 0
\end{aligned}$$

- 那我们对我们的估计有多确定呢？ 定义一个关于期望估计的不确定性衡量【就是score function的方差】

$$\mathop{\mathbb{E}}_{p(x|\theta)}\left[(s(\theta) - 0)\,(s(\theta) - 0)^{\mathrm{T}}\right].$$

θ是一个向量，其是这就是Fisher Information Matrix

Fisher Information Matrix:

$$\mathbf{F} = \mathop{\mathbb{E}}_{p(x|\theta)}\left[\nabla \log p(x|\theta)\, \nabla \log p(x|\theta)^{\mathrm{T}}\right].$$

考虑到最大似然函数很复杂，计算其梯度也是很棘手，所以我们用经验分布代替，通俗地讲，就是从训练数据中来计算这个F

which is given by our training data $X = \{x_1, x_2, \cdots, x_N\}$. In this form, $\mathbf{F}$ is called Empirical Fisher:

$$\mathbf{F} = \frac{1}{N} \sum_{i=1}^{N} \nabla \log p(x_i|\theta) \nabla \log p(x_i|\theta)^{\mathrm{T}} .$$

F还有个性质 $\mathbf{F} = -\mathbb{E}_{p(x|\theta)}\left[\mathbf{H}_{\log p(x|\theta)}\right]$.

推导过程如下：【没看懂，第一步就凉凉】

Proof. The Hessian of the log likelihood is given by the Jacobian of its gradient:

$$\begin{aligned}
\mathbf{H}_{\log p(x|\theta)} &= \mathrm{J}\left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right) \\
&= \frac{\mathbf{H}_{p(x|\theta)}\, p(x|\theta) - \nabla p(x|\theta)\, \nabla p(x|\theta)^{\mathrm{T}}}{p(x|\theta)\, p(x|\theta)} \\
&= \frac{\mathbf{H}_{p(x|\theta)}\, p(x|\theta)}{p(x|\theta)\, p(x|\theta)} - \frac{\nabla p(x|\theta)\, \nabla p(x|\theta)^{\mathrm{T}}}{p(x|\theta)\, p(x|\theta)} \\
&= \frac{\mathbf{H}_{p(x|\theta)}}{p(x|\theta)} - \left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)\left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)^{\mathrm{T}},
\end{aligned}$$

where the second line is a result of applying quotient rule of derivative. Taking expectation wrt. our model, we have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{p(x|\theta)}\left[\mathbf{H}_{\log p(x|\theta)}\right] &= \mathop{\mathbb{E}}_{p(x|\theta)}\left[\frac{\mathbf{H}_{p(x|\theta)}}{p(x|\theta)} - \left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)\left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)^{\mathrm{T}}\right] \\
&= \mathop{\mathbb{E}}_{p(x|\theta)}\left[\frac{\mathbf{H}_{p(x|\theta)}}{p(x|\theta)}\right] - \mathop{\mathbb{E}}_{p(x|\theta)}\left[\left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)\left(\frac{\nabla p(x|\theta)}{p(x|\theta)}\right)^{\mathrm{T}}\right] \\
&= \int \frac{\mathbf{H}_{p(x|\theta)}}{p(x|\theta)} p(x|\theta)\, \mathrm{d}x - \mathop{\mathbb{E}}_{p(x|\theta)}\left[\nabla \log p(x|\theta)\, \nabla \log p(x|\theta)^{\mathrm{T}}\right] \\
&= \mathbf{H}_{\int p(x|\theta)\, \mathrm{d}x} - \mathbf{F} \\
&= \mathbf{H}_1 - \mathbf{F} \\
&= -\mathbf{F} .
\end{aligned}$$

Thus we have $\mathbf{F} = -\mathbb{E}_{p(x|\theta)}\left[\mathbf{H}_{\log p(x|\theta)}\right]$. □

不过我们单看这个结论，可以看到F的作用其实是对对数似然函数的曲率的分析衡量【Hessian矩阵看成二阶导数】

事实上，我们对Fisher Information Matrix 的使用，是一个简单的替代对于Hessian矩阵，在二次优化问题上。

链接：https://agustinus.kristia.de/techblog/2018/03/14/natural-gradient/

2.3.2 【Natural Gradient Descent】

$$\frac{-\nabla_\theta \mathcal{L}(\theta)}{\|\nabla_\theta \mathcal{L}(\theta)\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{d \text{ s.t. } \|d\| \leq \epsilon}{\arg\min} \mathcal{L}(\theta + d)\,.$$

> The above expression is saying that the steepest descent direction in parameter space is to pick a vector d, such that the new parameter $\theta + d$ is within the ϵ-neighbourhood of the current parameter θ, and we pick d that minimize the loss. Notice the way we express this neighbourhood is by the means of Euclidean norm. Thus, the optimization in gradient descent is dependent to the Euclidean geometry of the parameter space.
>
> Meanwhile, if our objective is to minimize the loss function (maximizing the likelihood), then it is natural that we taking step in the space of all possible likelihood, realizable by parameter θ. As the likelihood function itself is a probability distribution, we call this space distribution space. Thus it makes sense to take the steepest descent direction in this distribution space instead of parameter space.
>
> Which metric/distance then do we need to use in this space? A popular choice would be KL-divergence. KL-divergence measure the "closeness" of two distributions. Although as KL-divergence is non-symmetric and thus not a true metric, we can use it anyway. This is because as d goes to zero, KL-divergence is asymptotically symmetric. So, within a local neighbourhood, KL-divergence is approximately symmetric [1].

2.3.2.1 Distribution Space

要最大化似然函数来找$p(x|\theta)$中的参数θ，这等价于找最小损失函数$\mathcal{L}(\theta) = -\log p(x|\theta)$单纯沿着负梯度方向用最速下降法，

$$\frac{-\nabla_\theta \mathcal{L}(\theta)}{\|\nabla_\theta \mathcal{L}(\theta)\|} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{d \text{ s.t. } \|d\| \leq \epsilon}{\arg\min} \mathcal{L}(\theta + d)\,.$$

This is the steepest descent direction around the local neighbourhood of the current value of θ in the parameter space.【浅浅解释一下:我们要做的是在以θ为中心的ϵ小领域内找到一个最好的d（loss function下降最多的），但是我们注意到，这里的距离使用的是Euclidean norm，Thus, the optimization in gradient descent is dependent to the Euclidean geometry of the parameter space.】

一段不懂的逻辑：

如果我们的目标是最小化损失函数（最大化似然），那么我们自然可以在所有可能由参数θ实现的似然函数所构成的空间中进行移动。由于似然函数本身是一个概率分布，我们称之为分布空间（distribution space）。因此，在该分布空间而不是参数空间（parameter space)中采用最陡下降方向是有意义的。【反正就是说我们要在分布空间找min最好 WHY？ I DON'T UNDERSTAND】

之后，在分布空间里距离的衡量就是靠KL散度了{尽管KL散度是非对称的，因此不是真正的度量，但我们还是可以使用它。这是因为当d为零时，KL散度是渐近对称的。因此，在局部邻域内，KL散度近似对称}

2.3.2.2 Fisher Information and KL-divergence

证明过程不贴了，

$$\begin{aligned}\mathbf{H}_{\mathrm{KL}[p(x|\theta)\,\|\,p(x|\theta')]} &= -\int p(x|\theta)\,\nabla^2_{\theta'}\log p(x|\theta')\big|_{\theta'=\theta}\,\mathrm{d}x \\ &= -\int p(x|\theta)\,\mathbf{H}_{\log p(x|\theta)}\,\mathrm{d}x \\ &= -\mathop{\mathbb{E}}_{p(x|\theta)}\left[\mathbf{H}_{\log p(x|\theta)}\right] \\ &= \mathbf{F}\,.\end{aligned}$$

F是KL散度的Hessian Matrix

2.3.2.3 Steepest Descent in Distribution Space

Claim: Let $d \to 0$. The second order Taylor series expansion of KL-divergence is $\mathrm{KL}[p(x|\theta)\,\|\,p(x|\theta+d)] \approx \frac{1}{2}d^{\mathrm{T}}\mathrm{F}d$.

对于论文中

$$\lim \delta\theta \to 0,$$
$$D\left(\theta+\delta\theta||\theta\right)=\frac{1}{2}\delta\theta^{\top}\mathbf{F}\left(\theta\right)\delta\theta,$$

使用拉格朗日乘子法，Loss function使用Taylor一阶近似，KL散度使用Taylor二阶近似，可以得到

$$d=-\frac{1}{\lambda}\mathbf{F}^{-1}\nabla_{\theta}\mathcal{L}(\theta)$$

其中$\frac{1}{\lambda}$可以被lr吸收，就可以得到论文中 $\widetilde{\nabla}_{\theta}J=\mathbf{F}^{-1}\nabla_{\theta}J(\theta).$

文章中使用的是少量参数，少量样本来优化，考虑到实际中参数量，样本数过大问题，就应运而生了近似估计Fisher Information Matrix或者Hessian Matrix的算法，比较流行的是ADAM，大大降低了存储，计算的时间复杂度。貌似后面论文讲的也是这个算法。

最后贴一个知乎理解，链接：https://zhuanlan.zhihu.com/p/228099600 有选择地借鉴里面的某些翻译说法。

【有点小疑问：

即 $$\frac{\partial \log l(\theta)}{\partial \theta} = \sum_{i=1}^{n} \nabla_\theta \log p(x_i|\theta) = 0$$

这个是对数似然函数的梯度；

论文中是

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda} \sum_{k=1}^{\lambda} f(\mathbf{z}_k)\, \nabla_\theta \log \pi(\mathbf{z}_k \,|\, \theta),$$

这样定义的，

纯理论优化natural gradient时，就是单纯优化对数似然函数，使其最大化，或者最小化loss function（这个就是对数似然函数加个负号），但是论文似乎是加了个适应度函数fitness function

$f(z)$，其实最后的推导什么的应该没什么影响

论文佐证：

left, the solid (black) arrows indicate the gradient samples $\nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)$, while the dotted (blue) arrows correspond to $f(\mathbf{z}) \cdot \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)$, that is, the same gradient estimates, but scaled with fitness. Combining these, the bold (green)

】

3. Performance and Robustness Techniques

在下文中，我们将介绍并引入关键的启发式算法，以提高NES的性能和鲁棒性。适应度整形（Wierstra et al.，2008）旨在使算法对任意保序适应度变换保持不变（第3.1节）。第3.2节介绍了一种在线调整学习率的新技术——自适应采样。在第3.3节和第3.4节中，我们描述了提高应用于多正态分布的NES算法性能的两个关键技术：指数参数化确保协方差矩阵保持正定；其次，提出了一种将坐标系转换为“自然”坐标系的新方法，这使得算法计算效率更高

3.1 Fitness Shaping

NES使用基于秩的适应度成形，以使算法在适应度函数的单调递增（即，保持秩）变换下保持不变。为了达到这个目的，用utility value 替代fitness function 即用u_i替代$f(z_i)$（其中我们认为u_1是最好的，就像我们认为$f(z_1)$是最好的一样。）

$$\nabla_\theta J(\theta) = \sum_{k=1}^{\lambda} u_k \, \nabla_\theta \log \pi(\mathbf{z}_k \,|\, \theta).$$

$$u_k = \frac{\max\left(0, \log(\frac{\lambda}{2}+1) - \log(k)\right)}{\sum_{j=1}^{\lambda} \max\left(0, \log(\frac{\lambda}{2}+1) - \log(j)\right)} - \frac{1}{\lambda},$$

这样做就可以把u_i看成一个自由参数。此外，这和下面会比较的CMA-ES算法采用的方法相关，我们这样做是便于下面的比较？

但是，根据我们的经验，只要它是单调的，并且基于rank而不是原始适应度，这种选择【用u_i替代$f(z_i)$】对性能并不是至关重要的（例如，一个简单地随等级线性增加的函数）

3.2 Adaptation Sampling

自适应采样，用来在线调参的

这样的一个逻辑：$z \sim \pi_\theta$,同时我们有一个衡量样本质量的函数$\phi(z)$【可以理解为fitness function吗略有差别：下面高亮处】我们的目标就是调整θ使得$\phi(z)$最大。

一个很直观的想法就是一点一点的调整θ,只要比之前好，就采纳【进化算法？】

此过程类似于NES算法本身，但在元级别应用于算法参数，而不是搜索分布。

这个adaptation的目标是最大化 progress的pace 这和最大化fitness function 本身略有差别