CSCI-GA.2565-001 Machine Learning: Homework 4

Due 11:59pm EST, April 26, 2022 on Gradescope

We encourage LaTeX-typeset submissions but will accept quality scans of hand-written pages. LaTeX-typeset submissions will be good practice for anyone thinking of getting into research.

1 Variational Inference (VI)

(a) Let $p(\mathbf{z}, \mathbf{x}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)$ where $\mathbf{x}_i, \mathbf{z}_i \in \mathbb{R}^D$. In VI, we find an approximation $q_\lambda(\mathbf{z})$ to $p(\mathbf{z}|\mathbf{x})$. Let $V_1(\lambda)$ be the set of variational approximations $\{q_\lambda : q_\lambda(\mathbf{z}) = \prod_{i=1}^{N} q(\mathbf{z}_i; \lambda_i)\}$ where λ_i are parameters learned for each datapoint $\mathbf{x}_i$. Instead, let $f_\lambda(\mathbf{x})$ be a deep neural network and $V_2(\lambda) = \{q_\lambda : q_\lambda(\mathbf{z}) = \prod_{i}^{N} q(\mathbf{z}_i; f_\lambda(\mathbf{x}_i))\}$. Which of the two families is more expressive (i.e. approximates a larger set of distributions)? Prove either $V_1(\lambda) \subseteq V_2(\lambda)$ or $V_2(\lambda) \subseteq V_1(\lambda)$ or both.

For a given parametric family, the free optimization over λ values will be the most general.

For $V_1(\lambda)$, we have the λ_i values optimized for each data point.

For $V_2(\lambda)$, we have the λ values based on a parametric function. Here, as compared to the first case, the constraint is that all the variational parameters lie in the range of the network and the corresponding transformation. Only for a neural network with an infinite capacity, we could represent any possible mapping for x and λ. For an appropriate structure of the distribution in the first case, we will have a corresponding optimization for x_i . In the second case, we are adding the assumption that the parameterized function to represent the mapping between the x space and the λ is the same for different values of x.

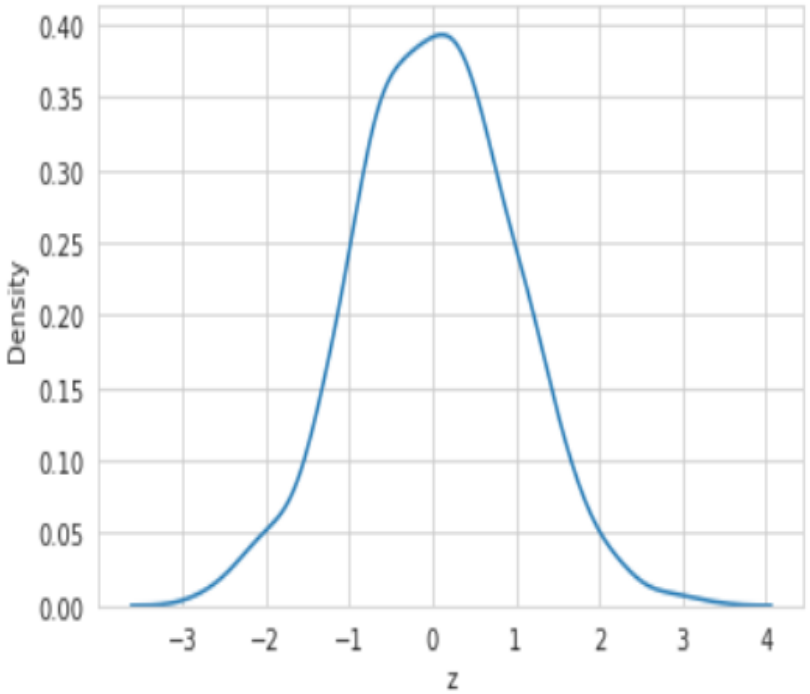
If suppose a set of λ values has been represented by the neural network for V_2. V_1 can also represent these λ values. In V_1, we can set the λ values for each data point as those output by the neural network for each data point.

The first case is more general than the second case.

Thus, $V_2(\lambda) \subseteq V_1(\lambda)$.

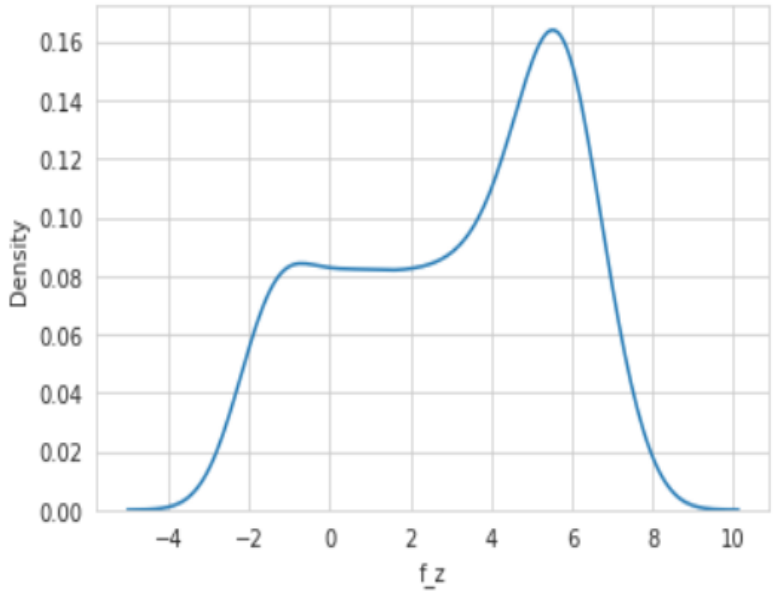
(b) Stepping away from VI, let's review how to use invertible transformations and the change of variables formula to turn simple distributions into complex ones.

(i) Draw $N = 1000$ samples of $z_0 \sim \mathcal{N}(0, 1)$ and produce a density plot of the samples (see starter code).



(ii) Look up "Planar Flow" in eq. 4 of The Expressive Power of a Class of Normalizing Flow Models. For some value of w, b, u, and some invertible non-linearity h, produce a density plot of the samples of $f(z_0)$ where f is the planar flow.

```
sns.kdeplot(data=df_new, x='f_z')
plt.show()
```




```
def f(z):

    z=torch.tensor(z).clone()
    w=3
    w=torch.tensor(w).clone()

    b=1
    u=5
    h= w*z+b
    h=1/(1+torch.exp(-h))
    return z+u*h
```

Note that $d = 1$ so that z and w in the equation are scalars. It is okay if h is only invertible on its output range, e.g. sigmoid on (0,1).

(iii) One reason that flows are useful is that they can map unimodal distributions into multimodal ones, while still allowing for a tractable density. Find values of h, w, b, u such that the plot of $f(z_0)$ has more than one mode.

We have,

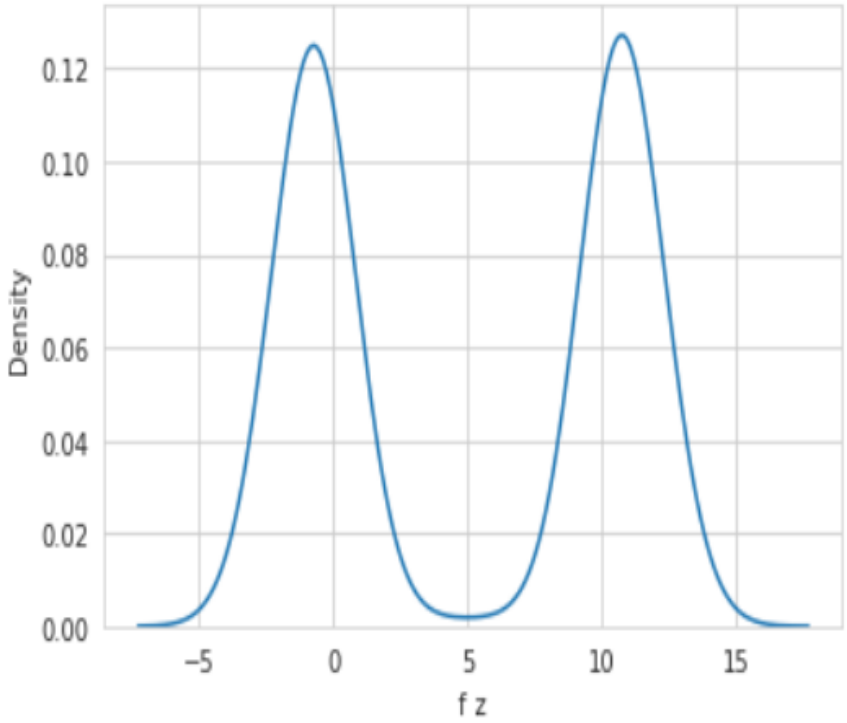
$$f(z) = z + u * h(w * z + b)$$

We define,

$$f(z) = z + 10 * h(100 * z + 0.1)$$

A high value of w denotes a sharp variation of the sigmoid around 0. The Gaussian distribution has a high concentration around 0. positive samples will get tend to be close to sigmoid value 1 and

negative samples will tend to be close to sigmoid value 0. Hence we get two peaks around 0 and 10 as per our transformation defined.



(iv) Use change of variables and write down an explicit expression for the density of $f(z_0)$. This depends on your choice of h.

Let $y = f(z)$ For a transformation of Random Variables, we have,

$$g(y) = p(z) * |J^1|$$

Where the J^1 is the Jacobian,

$$g(y) = p(z) * \left| \frac{\partial g^{-1}(y)}{\partial y} \right|$$

We can write this using the property of Jacobians based on the inverse function theorem as,

$$g(y) = p(z) \left| \frac{\partial f(z)}{\partial z} \right|^{-1}$$

We have,

$$g(y) = \frac{p(z)}{|J^2(f(z))|}$$

J^2 is a Jacobian.

Now,

$$J^2(z) = \frac{\partial f(z)}{\partial z}$$

We have,

$$f(z) = z + uh(w * z + b)$$

$$h = \sigma(w * z + b) = \frac{1}{1 + e^{-(w*z+b)}}$$
$$h' = \sigma(w * z + b)(1 - \sigma(w * z + b))$$

$$f'(z) = 1 + u * w * h'(wz + b)$$

Therefore,

$$g(y) = \frac{p(z)}{1 + u * w * \frac{1}{1+e^{-(w*z+b)}} * \left(1 - \frac{1}{1+e^{-(w*z+b)}}\right)}$$

(c) Let's generalize this to D-dimensional variables and to a sequence of invertible functions f (not necessarily planar flows). We can take a sample $\mathbf{z}^{(0)} \sim \mathcal{N}(\mathbf{z}^{(0)} \mid 0, I)$ and then transform that sample iteratively using a sequence of invertible functions, $f_1, \ldots, f_K$ (each from D to D dimensions), to obtain a sample $\mathbf{z}^{(K)}$

$$\mathbf{z}^{(K)} = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}^{(0)}).$$

Using the change-of-variables formula, write a formula for the log-density of $\mathbf{z}^{(K)}$ using the functions $\{f_k\}$, possibility their inverses, Jacobians, etc, and the log-density of $\mathcal{N}(\mathbf{z}^{(0)} \mid 0, I)$.

We have, for some R.V.s z and z^1, for the transformation from z to z^1, we have,

$$g(z^1) = p(z)\left|\frac{\partial f}{\partial z}\right|^{-1}$$

where,

$$z^1 = f(z)$$

if,

$$z^2 = f'(z^1)$$

then,

$$q(z^2) = g(z^1)\left|\frac{\partial f'}{\partial z^1}\right|^{-1} = p(z)\left|\frac{\partial f}{\partial z}\right|^{-1}\left|\frac{\partial f'}{\partial z^1}\right|^{-1}$$

We can see that there will be a product of the Jacobians in the denominator. This is the case for composite functions. We have,

$$\mathbf{z}^{(K)} = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}^{(0)}).$$

Therefore, for the distribution over z_k,

$$q(z^K) = \frac{q(z^0)}{\prod_{i=1}^{K}\left|det\frac{\partial f^k}{\partial z^{k-1}}\right|}$$

$$ln(q(z^K)) = ln(q(z^0)) - \sum_{i=1}^{K} ln\left|det\frac{\partial f^k}{\partial z^{k-1}}\right|$$

(d) Let's return to VI and how we can improve inference using flows.

(i) How can we define an $q_\lambda(\mathbf{z})$ to $p(\mathbf{z}|\mathbf{x})$ using flows?

We want to model $p(z/x)$.

For amortized variational inference, we construct an inference model using a deep neural network to build a mapping from the observations x to the parameters(mean and variance) of the initial density $q(z_0)$. This will be conditioned on x.

In the non-amortized case, when we are performing the optimization for each data point, we can start with some simple distribution for z_0, as a standard Gaussian R.V. $\epsilon = \mathcal{N}(0, 1)$ of appropriate dimension.

We can then use normalizing flows to transform z_0 to z_1 up to z_K. The distribution $q(z_K$ will be our approximation to $p(z/x)$. The parameters which govern the individual transformations are set in a way that the Jacobian computations are efficient.

(ii) How can we optimize the parameters of these functions to yield a good approximation to $p(\mathbf{z}|\mathbf{x})$? The normalizing flows give us a neat equation for the distribution of $q(z_K/x)$ in terms of the distribution of $q(z_0/x)$ and the log sum of the Jacobians which are efficiently parameterized for smooth computations. Now, we want to estimate p(z/x).

Recall that for our variational estimate of the maximum likelihood, maximizing the ELBO is equivalent to minimizing the KL divergence between the posterior for p and q.

We thus focus on maximizng the elbo. We have,

$$log(p(x)) = E_q[\log p(x, z) - log(q(z; \lambda))] + KL(q(z; \lambda)||p(z/x))$$

$$L(\lambda) = E_q[\log p(x, z) - log(q(z; \lambda))]$$

If we solve this optimization problem, the upper bound for $L(\lambda)$ is the log likelihood, this occurs when the desired KL divergence is minimum i.e. 0. Now, we know our approximation q is $q(z_k)$

$$L(\lambda) = E_q(z_k)[\log p(x, z_k) - \log q(z_k; \lambda)]$$

We can use the reparemeterization trick to transform the expectation wrt to

$$L(\lambda) = E_{q(z_k)}[\log p(x, z_k) - \log q(z_k)]$$

We have,

$$\log(q(z^K)) = \log(q(z^0)) - \sum_{i=1}^{K} ln \left| det \frac{\partial f^k}{\partial z^{k-1}} \right|$$

We can make a transform the expression from an expectation wrt z_0 to an expectation wrt z_k. The law of the unconscious statistician will aid for validiity.

$$L(\lambda) = E_{q(z_0)} \left[\log p(x, f(z_0)) - \log q(z^0) - \sum_{i=1}^{K} ln \left| det \frac{\partial f^k}{\partial z^{k-1}} \right| \right]$$

Notice that in the case of standard variational inference, z_0 starts with the standard normal. We can sample from z_0 and use monte carlo estimation for the expectation and the corresponding gradients. We express z_k as a function of z_0. We then get the $\log p(x, z)$ term in terms of z_0 and we get the elbo term as dependent on z_0 and the parameters. We know how to write the gradient of the expectation in terms an the expectation. Using,

$$\nabla_{lambda} \log q = \frac{\nabla_{lambda} q}{q}$$

This can enable efficient monte carlo estimation.

Consider the use of the variational autoencoder with normalizing flows.

$$L(\lambda) = E_{q(z_k)}[\log p(x/z_k)] - KL(q(z_k)||p(z))$$

We can set up the posterior distribution of q based on a normalizing flow starting with a standard normal. We have a prior for z for the distribution over x .For every x, use the reparameterization trick to obtain z from q. Put this forward into to generator to obtain a distribution of x given z. Compute the variational lower bound of the log likelihood. Perform the optimization.

(iii) Is this more flexible than using a Gaussian approximation for $q_\lambda(\mathbf{z})$?

Yes, a transformation of Gaussian is a more flexible distribution. A transformation of a gaussian R.V. can capture a rich set of distributions not necessarily a Gaussian. This is more flexible than using a Gaussian approximation.

2 Monte Carlo Gradients

Suppose we want to maximize the following expectation with respect to a parameter μ:

$$\max_{\mu} \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{N}(\mu,1)} \left[-\mathbf{z}^2\right] \tag{1}$$

We now explore several ways to do this with gradients.

(a) What is the closed form of the integral Equation (1)?

We can express the expectation as,

$$f(u) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(z-u)^2}[-z^2]dz$$

$$f(u) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(z-u)^2} z^2 dz$$

Considering a change of variable as, z- u = t t = z+u z goes from negative infinity to infinity. Considering a finite mean, t goes from negative infinity to infinity.

$$f(u) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(z-u)^2} z^2 dz$$

$$f(u) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} (t^2 + 2tu + u^2) dt$$

$$f(u) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} t^2 dt + -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} 2tu dt + -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} u^2 dt$$

We express this as,

$$f(u) = a + b + c$$

Consider the following integral,

$$f(u) = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} t^2 dt$$

$$I = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} dt$$

$$I^2 = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x)^2} dx * -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y)^2} dy$$

$$I^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x^2+y^2)} dxdy$$

$$I^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x^2+y^2)} dxdy$$

Changing cartesian coordinates to polar coordinates,

$$I^2 = \int_{0}^{2\pi}\int_{0}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}r^2} r dr d\theta$$

Changing variables,

$$r^2/2 = k$$
$$r.dr = dk$$

$$I^2 = \int_0^{2\pi} \int_0^{\infty} \frac{1}{2\pi} e^{-k} dk d\theta$$

$$I^2 = \int_0^{2\pi} \frac{1}{2\pi} - [e^{-k}]_0^{\infty} d\theta$$

$$I^2 = \int_0^{2\pi} \frac{1}{2\pi} 1 d\theta$$

$$I = 1$$

Therefore,

$$c = -u^2 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} 2tudt + - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} dt$$

$$c = -u^2.I$$
$$c = -u^2$$

$$b = - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} 2tudt$$

$$b = -u \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} 2tdt$$

We look at the integral and realize that it is an odd function. i.e.

$$f(x) = f(-x)$$

For the domain of negative infinity to infinity, the integral will be zero.

$$b = 0$$

$$a = - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} t^2 dt$$

$$a = - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} t^2 dt$$

We know,

$$\int uv' dx = uv - \int u'v dx$$

Let,

$$u = t$$

$$v = -\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2}$$

$$v' = \frac{1}{\sqrt{2\pi}} t e^{-\frac{1}{2}(t)^2}$$

$$-a = \int uv' dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} t^2 dt$$

Using standard integration by parts, we get, Considering n as the limit of t tending to infinity.

$$-a = -ne^{-n^2/2} - ne^{-n^2/2} + \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(t)^2} dt$$

$$a = -1$$

$$f(u) = a + b + c$$

$$f(u) = -u^2 - 1$$

(b) Recall from class the score function gradient (also known as the REINFORCE gradient or log-derivative trick). Write down the score function gradient for this problem.

We find the score function gradients,

$$F(u) = E_{p(z)}[-z^2]$$

$$\nabla_u F(u) = \nabla_u E_{p(z)}[-z^2]$$

$$\nabla_u F(u) = E_{p(z)}[\nabla_u \log p(z)[-z^2]]$$

Using the property for taking gradients inside expectations.

$$p(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(z-u)^2}$$

$$\nabla_u \log p(z) = (z - u)$$

$$\nabla_u F(u) = E_{p(z)}[(z - u)[-z^2]]$$

(c) In words, describe the direction in which the score function $\nabla_\mu \log p_\mu(x)$ of a distribution points.

The direction in which the score function $\nabla_\mu \log p_\mu(x)$ corresponds to the difference between the z and u vectors.

(d) Recall from class that another option is the reparameterization gradient. Which distribution s and mapping g would reparameterize this problem? Write this down for this problem.

The standard normal will give us the appropriate reparameterization.

Let

$$p(\epsilon) = \mathcal{N}(0, 1)$$

$$p(z) = \mathcal{N}(\mu, 1)$$

$$z = t(\epsilon; \mu)$$

Based on transformation properties of Gaussian R.V.s.

$$z = \epsilon + \mu$$

We have an expectation wrt z which we can convert as an expectation wrt to ϵ .

$$p(z) = p(\epsilon).|J|$$

$$f(z) = -z^2$$

$$f(z) = g(\epsilon)$$

$$g(\epsilon) = f(t(\epsilon; \mu))$$

$$g(\epsilon) = -(z + u)^2$$

Using the law of the unconscious statistician, following from the change of variables, we want to find an expecation of g wrt ϵ.

$$E_{p(z)}[-z^2] = E_\epsilon[-(\epsilon + \mu)^2]$$

$$E_\epsilon[-(\epsilon + \mu)^2] = -E_\epsilon[\epsilon^2 + 2\mu\epsilon + \mu^2]$$

$$-E_\epsilon[\epsilon^2 + 2\mu\epsilon + \mu^2] = -E_\epsilon[\epsilon^2] + -2\mu E_\epsilon[\epsilon] + -E_\epsilon[\mu^2]$$

$$E_\epsilon[\epsilon] = 0 \quad (mean)$$
$$E_\epsilon[\epsilon^2] = 1 \quad (\mu^2 + \sigma^2)$$

Therefore,

$$E_{p(z)}[-z^2] = -\mu^2 - 1$$

$$E_\epsilon[-(\epsilon + \mu)^2] = -\mu^2 - 1$$

$$\nabla_u E_\epsilon[-(\epsilon + \mu)^2] = E_\epsilon[\nabla_u \log(p(\epsilon)) - (\epsilon + \mu)^2 + \nabla_u - (\epsilon + \mu)^2]$$

$$\nabla_u E_\epsilon[-(\epsilon + \mu)^2] = E_\epsilon[\nabla_u \log(p(\epsilon)). - (\epsilon + \mu)^2 + \nabla_u - (\epsilon + \mu)^2]$$

Since R.V. ϵ does not depend on μ. get,

$$\nabla_u E_\epsilon[-(\epsilon + \mu)^2] = E_\epsilon[-2(\epsilon + \mu)]$$

(e) What conditions do you require on p_θ and f ($f(z) = -z^2$ in this case) for each way to re-write the gradient above? Do these apply to both continuous and discrete distributions p_θ?

The transformation from ϵ to z must exist. The function mapping ϵ to z must be such that the distribution is well defined. In relation to the properties of the transformation of random variables, if z is a monotonically increasing or deceasing function of ϵ. The corresponding determinant of the Jacobian must be well defined.

The function mapping from ϵ to z must be differentiable.

Similarly, the function mapping from z to f(z) must also be differentiable.

(f) The above gradients are still defined as expectations. Monte-Carlo estimate both gradients $M = 10$ times for each sample size $N = 1, 10, 100, 1000$ and plot the variance (over the M trials). What do you observe?

For the Gradient of the expecation,

After reparameterization, the gradient is given as,

$$\nabla_u E_\epsilon[-(\epsilon + \mu)^2] = E_\epsilon[-2(\epsilon + \mu)]$$

The score function gradient is given as,

$$\nabla_u F(u) = E_{p(z)}[(z - u)[-z^2]]$$

Please see notebook pdf.

(g) Why is the variance of the gradient a concern in practice?

Suppose we do a monte carlo estimate of our gradient. Based on the variance graph we have plotted, we can see that any particular estimate may have a significant deviation from the value of the analytical estimate of the gradient wrt to the parameters. For some loss objective, this analytical estimate would be the gradient wrt to the parameters which we would want to compute.

Now, suppose we use our monte carlo estimate of our gradient. We then update the parameters based on this estimate by gradient descent. We now have a new set of parameter values.

The set of parameter values which we would obtain after updation by the monte carlo estimate of our gradients may be different from the set of parameter values which we would obtain after updation of the analytical gradient. The monte carlo gradient is an approximation to the analytical gradient.

Now, in our optimization process, this may lead us into regions in the parameter space which may not be very useful. Our trajectory may be suboptimal.

3 Causal Inference: Basics

The average treatment effect (ATE) is a measure used to compare treatments in medicine and economic policy. The ATE is the difference in mean potential outcomes of treated units and control units. In this question, we consider the effect of eating a lot of chocolate on the age of retirement. We denote $T = 1$ to mean 'ate a lot of chocolate' and $T = 0$ to mean 'ate very little chocolate'. (Problem Credit: Jaan Altosaar)

The table below contains 10 sampled people. Each sample contains the person's handedness $H \in \{L, R\}$, treatment $T \in \{0, 1\}$, and the observed outcome (age of retirement) $Y(T)$. The empty cells indicate the counterfactual outcomes that are not possible to observe.

Person	Handedness (H)	T	Y(1)	Y(0)
1	L	1	60	
2	L	1	75	
3	L	1	53	
4	L	1	69	
5	L	0		63
6	R	1	50	
7	R	0		42
8	R	0		50
9	R	0		58
10	R	0		59

(a) Estimate the ATE as the difference of the mean outcomes of the $T = 1$ and the $T = 0$ populations:

$$\text{ATE}_1 = \mathbb{E}\big[Y|T = 1\big] - \mathbb{E}\big[Y|T = 0\big]$$

Compute estimates of these expectations using the datapoints in the table and report the difference.

Empirically,

$$E[Y/T = 1] = (60 + 75 + 53 + 69 + 50)/5 = 61.4$$

$$E[Y/T = 0] = (59 + 58 + 50 + 42 + 63)/5 = 54.4$$

$$ATE_1 = 61.4 - 54.4 = 7$$

(b) As we can see from the data, the population that the treatment was given to is mostly left-handed people. Instead, compute the handedness-specific effect and average over handedness:

$$\text{ATE}_2 = \mathbb{E}_{P(H)}\big[\mathbb{E}\big[Y|T = 1, H\big] - \mathbb{E}\big[Y|T = 0, H\big]\big]$$

Compute estimates of these expectations using the data points in the table and report the difference.

$$\begin{aligned} ATE_2 = p(H = L)(E[Y/T = 1, H = L] - (E[Y/T = 0, H = L]) + \\ p(H = R)(E[Y/T = 1, H = R] - (E[Y/T = 0, H = R]) \end{aligned} \tag{2}$$

$$ATE_2 = \frac{1}{2}\left(\frac{60 + 75 + 53 + 69}{4} - \frac{63}{1}\right) + \frac{1}{2}\left(\frac{50}{1} - \frac{42 + 50 + 58 + 59}{4}\right)$$

$$ATE_2 = -0.5$$

(c) Which one of ATE_1 and ATE_2 estimates the ATE better and why? Explain. Is your answer affected by the knowledge of whether handedness is known to affect both treatment (chocolate) allocation and age of retirement?

Consider the factorization which corresponds to our calculation of the following expectation,

$$E[Y/T=1] = \sum_y p(y/t=1)y$$

$$E[Y/T=1] = \sum_y \sum_h p(y,h/t=1)p(y/t)y$$

$$E[Y/T=1] = \sum_y \sum_h p(y/h,t)p(h/t=1)y$$

Here, t is set as 1.

We then get empirically,

$$E[Y/T=1] = \sum_{l=1}^{4} \frac{1}{4}\frac{4}{5}60... + \frac{1}{5}50$$

This is the constitution of our answer for ATE_1

For ATE_2, we have,

$$E_{P(H)}[E[Y/T=1,H]] = \sum_h p(h) \sum_y p(y/t=1,h)y$$

$$E_{P(H)}[E[Y/T=1,H]] = \sum_y \sum_h p(y/t=1,h)p(h)y$$

ATE_2 can better reflect the causal set up of the experiment. Once we set our variable t, we should only consider variations in the y variable wrt variations in t. The variations in the confounder variable after setting t can influence variations in the observed variable, but we primarily want to model the variations in y wrt to the variations in t notwithstanding the distribution of the confounder given the t. This is since the t is already set. So the stochasticity which captures the dependence of h(confounder) given t need not be a part of the expectation.

This follows from the functional view of causal inference.

We want to set the target variable t to some value, based on this setting, we want to observe the changes in y.

If suppose we have the prior knowledge that handedness is known to affect both t and y. We may then want to consider the variations in h as a result of setting t and keep that factor in our expectation. Here, ATE_1 might be better.

4 Causal Inference: Doubly Robust Estimators

Denote unit i's treatment, outcome, and its vector of covariates by T_i, Y_i, and X_i. Let us model propensity scores by $e(x;\hat{\theta})$ and the outcome by $f(x;\hat{\psi})$. Assume strong ignorability and positivity holds. We define the Doubly Robust (DR) Estimator for average outcome when treated by:

$$\frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}-\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]. \tag{3}$$

(a) Suppose the propensity model $e(x;\hat{\theta})$ for $P(T_i=1|X_i=x)$ is correctly specified, i.e., $e(x;\hat{\theta})=P(T_i=1|X_i=x)$. Given any function f, show that DR is an unbiased estimator of the average outcome when treated, $\mathbb{E}[Y^{(1)}]$.

For our doubly robust estimator DR,

$$DR=\frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}-\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]. \tag{4}$$

$$E_{X,T,Y}[DR]=E_{X,T,Y}\left[\frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}-\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right]. \tag{5}$$

Using the linearity of expectation.

$$E_{X,T,Y}[DR]=\frac{1}{n}\sum_{i=1}^{n}\left[E_{X,T,Y}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]-E_{X,T,Y}\left[\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right]. \tag{6}$$

For the expectation, we can write this as, based on the factorization, $p(x,y,t)=p(x,y)p(t/x,y)$, we will have the subsequent summations for the expectations.

For any Y_i,

$$Y_i=Y_{1,i}T+Y_{0,i}(1-T)$$

$$E_{X,T,Y}[DR]=\frac{1}{n}\sum_{i=1}^{n}\left[E_{X,Y}E_{T/(X,Y)}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]-E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right]. \tag{7}$$

Simplifying the inner expectation,

$$E_{X,T,Y}[DR]=\frac{1}{n}\sum_{i=1}^{n}\left[E_{X,Y}\left(p(t=1/x,y)*\frac{1*Y_{1,i}}{e(X_i;\hat{\theta})}+p(t=0/x,y)\frac{0*Y_{0,i}}{e(X_i;\hat{\theta})}\right)\right. \tag{8}$$

$$\left.-E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right] \tag{9}$$

We use strong ignorability,

$$Y_0,Y_1 \perp\!\!\!\perp T/X$$

Based on positivity,

$$0<e<1$$

We get,

$$E[T/X,Y] = E[T/X]$$

.

We also know that,

$$E[T/X] = p(t=1/x) * 1 + p(t=0/x) * 0 = p(t=1/x)$$

$$E_{X,T,Y}[DR] = \frac{1}{n}\sum_{i=1}^{n}\left[E_{X,Y}\left(p(t=1/x) * \frac{1 * Y_{1,i}}{e(X_i;\hat{\theta})}\right)\right. \tag{10}$$

$$\left. -E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right] \tag{11}$$

We know that the propensity score, $e(x;\hat{\theta})$ models $p(t=1/x)$

We can thus write our expression as,

$$E_{X,T,Y}[DR] = \frac{1}{n}\sum_{i=1}^{n}\left[E_{X,Y}\left(Y_{1,i}\right)\right. \tag{12}$$

$$\left. -E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]\right] \tag{13}$$

$$E_{X,T,Y}[DR] = \frac{1}{n}\sum_{i=1}^{n}E_{X,Y}\left(Y_{1,i}\right) \tag{14}$$

$$-\frac{1}{n}\sum_{i=1}^{n}E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] \tag{15}$$

We look at the first part of the equation. The each individual summation will give us the expected value of Y_1, the sum over n terms will put a coeffeicent on n which will be cancelled by the denominator. Thus we get,

$$E_{X,T,Y}[DR] = E[Y_1] \ -\frac{1}{n}\sum_{i=1}^{n}E_{X,Y}E_{T/(X,Y)}\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] \tag{16}$$

We now look at the second part of this equation, using strong ignorability,

$$E_{X,T,Y}[DR] = E[Y_1] \ -\frac{1}{n}\sum_{i=1}^{n}E_{X,Y}\left[p(t=1/x)\frac{1-e}{e}f + p(t=0/x)\frac{-e}{e}f\right] \tag{17}$$

We look at what the propensity score models,

$$E_{X,T,Y}[DR] = E[Y_1] \ -\frac{1}{n}\sum_{i=1}^{n}E_{X,Y}\left[p(t=1/x)\frac{1-p(t=1/x)}{p(t=1/x)}f - p(t=0/x)f\right] \tag{18}$$

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n} E_{X,Y}\Big[(1 - p(t = 1/x))f - p(t = 0/x)f\Big] \tag{19}$$

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n} E_{X,Y}\Big[p(t = 0/x)f - p(t = 0/x)f\Big] \tag{20}$$

Thus we get,

$$E_{X,T,Y}[DR] = E[Y_1] \tag{21}$$

The doubly robust estimator is an unbiased estimate of $E[Y_1]$.

(b) Suppose the model $f(x;\hat{\psi})$ for $\mathbb{E}[Y_i|X_i = x]$ is correctly specified, i.e., $f(x;\hat{\psi}) = \mathbb{E}[Y_i|X_i = x, T = 1] = \mathbb{E}[Y_i^{(1)}|X_i = x]$. Given any function e taking values in $(0, 1)$, show that the estimator in (1) is still unbiased for $\mathbb{E}[Y^{(1)}]$

We start with the equation (18) from our previous section. We start expanding the term which involves f.

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n} E_{X,Y}E_{T/(X,Y)}\Big[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\Big] \tag{22}$$

Here, $f(x;\hat{\psi}) = \mathbb{E}[Y_i|X_i = x, T = 1] = \mathbb{E}[Y_i^{(1)}|X_i = x]$.

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n}\sum_{x,y,t} p(y/x,t)p(t/x)p(x)\Big[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\Big] \tag{23}$$

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n}\sum_{x,y} p(y/x,t)p(x)\Big[p(t = 1/x)\frac{1 - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi}) + \tag{24}$$

$$p(t = 0/x)\frac{-e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\Big] \tag{25}$$

Based on what the propensity score models, we get a similar cancellation, since, based on our model, we set T as 1. The nature of f given x does not change based on the binary variable T. For the case when t is 0 in our summation, we still have that given the x_i value. This enables us to do the cancellation which produces an unbiased estimate of Y_i. This follows from strong ignorability.

$$E_{X,T,Y}[DR] = E[Y_1] \ - \frac{1}{n}\sum_{i=1}^{n}\sum_{x,y} p(y/x,t)p(x)\Big[p(t = 0/x)f(X_i;\hat{\psi}) - \tag{26}$$

$$p(t = 0/x)f(X_i;\hat{\psi})\Big] \tag{27}$$

$$E_{X,T,Y}[DR] = E[Y_1] \tag{28}$$

(c) Lower variance estimators are better in general. Control variates can lower the variance.

Recall that control variates help you estimate $\mathbb{E}[f]$ via monte-carlo by instead defining a new function with the same expectation. This is done by taking a function g with $\mathbb{E}[g(x)] = 0$, then taking $\hat{f}(x) = f(x) - ag(x)$ for some $a \in \mathbb{R}$. Here $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$, but the variances are not equal. The value of a that makes the variance of $\hat{f}$ smallest is $a^* = \frac{Cov(f,g)}{var(g)}$.

When both $e(x;\hat{\theta})$ and $f(x;\hat{\psi})$ are correctly specified, use control variates to justify the use of Doubly Robust Estimators.

We have,

$$DR = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_i Y_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]. \tag{29}$$

Let,

$$h(x) = \frac{TY}{e(X;\hat{\theta})}. \tag{30}$$

$$g(x) = \left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]. \tag{31}$$

First we consider h(x). We take h(x) as our estimate for Y_1.

We have seen,

$$E_{Y,T,X}[h(x)] = E[Y_1] \tag{32}$$

We now have an empirical approximation for h(x).

$$h(x) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_i Y_i}{e(X_i;\hat{\theta})}\right] \tag{33}$$

We have an empirical approximation for g(x).

$$g(x) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right] \tag{34}$$

For this approximation, we know again from the previous questions,

$$E_{Y,T,X}[h(x)] = E[Y_1] \tag{35}$$

This can act an estimator for Y_1.

$$Var[h(x)] = E\left[\frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_i Y_i}{e(X_i;\hat{\theta})}\right] - E[Y_1]\right]^2 \tag{36}$$

$$Var[h(x)] = E\left[\frac{\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right] - nE[Y_1]}{n}\right]^2 \tag{37}$$

$$Var[h(x)] = \frac{1}{n^2}E\left[\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - E[Y_1]\right]\right]^2 \tag{38}$$

$$Var[h(x)] = \frac{1}{n^2}E\left[\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - E[Y_1]\right]^2 + Cov\left(\frac{T_iY_i}{e(X_i;\hat{\theta})} - E[Y_1], \frac{T_jY_j}{e(X_j;\hat{\theta})} - E[Y_1]\right)\right] \tag{39}$$

Here, since are samples are identical and independently distributed,

$$Var[h(x)] = \frac{1}{n^2}\left[\sum_{i=1}^{n}E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - E[Y_1]\right]^2\right] \tag{40}$$

$$Var[h(x)] = \frac{1}{n^2}\left[\sum_{i=1}^{n}E\left[\frac{T_i^2Y_i^2}{e(X_i;\hat{\theta})^2} - 2\frac{T_iY_i}{e(X_i;\hat{\theta})}E[Y_1] + E[Y_1]^2\right]\right] \tag{41}$$

$$Var[h(x)] = \frac{1}{n^2}\left[\sum_{i=1}^{n}\left[E\left[\frac{T_i^2Y_i^2}{e(X_i;\hat{\theta})^2}\right] - 2E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]E[Y_1] + E[Y_1]^2\right]\right] \tag{42}$$

$$Var[h(x)] = \frac{1}{n^2}\left[\sum_{i=1}^{n}\left[E\left[\frac{T_i^2Y_i^2}{e(X_i;\hat{\theta})^2}\right] - E[Y_1]^2\right]\right] \tag{43}$$

$$E_{X,T,Y}\left[\frac{T_i^2Y_i^2}{e(X_i;\hat{\theta})}\right] = \sum_{x,y}p(x)p(y/x,t)p(t=1/x)\frac{Y_{i,1}^2 * 1}{e_i^2} + \sum_{x,y}p(x)p(y/x,t)p(t=0/x)\frac{Y_{i,1}^2 * 0}{e_i^2} \tag{44}$$

$$E_{X,T,Y}\left[\frac{T_i^2Y_i^2}{e(X_i;\hat{\theta})}\right] = E[\frac{Y_1^2}{e}] \tag{45}$$

$$Var[h(x)] = \frac{1}{n^2}\left[\sum_{i=1}^{n}\left[E\left[\frac{Y_1^2}{e(X;\hat{\theta})^2}\right] - E[Y_1]^2\right]\right] \tag{46}$$

$$Var[h(x)] = \frac{1}{n}\left[E\left[\frac{Y_1^2}{e(X;\hat{\theta})^2}\right] - E[Y_1]^2\right] \tag{47}$$

Let,

Now consider the function, $h(x) - g(x)$. We have seen,

$$E_{Y,T,X}[g(x)] = 0 \tag{48}$$

We have seen,

$$E_{Y,T,X}[h(x) - g(x)] = E[Y_1] \tag{49}$$

$$Var[h(x) - g(x)] = E\left[\frac{1}{n}\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{50}$$

$$Var[h(x) - g(x)] = E\left[\frac{\sum_{i=1}^{n}\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - nE[Y_1]}{n}\right]^2 \tag{51}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}E\sum_{i=1}^{n}\left[\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{52}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}E\left[\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{53}$$

Analogous to the discussion on how we computed $h(x)$, dealing with the covariance terms , we get,

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}E\left[\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{54}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}E\left[\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{55}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}E\left[\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] - E[Y_1]\right]^2 \tag{56}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}\left[E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]^2 \tag{57}$$

$$-2E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]E[Y_1] + E[Y_1]^2\Bigg] \tag{58}$$

We know that $h(x)$ and $h(x) - g(x)$ have the same expectations which is $E[Y_1]$, we get,

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}\left[E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})} - \frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]^2\right. \tag{59}$$

$$\left. -E[Y_1^2]\right] \tag{60}$$

$$Var[h(x) - g(x)] = \frac{1}{n^2}\sum_{i=1}^{n}\left[E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]^2 - 2E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] + E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right]^2\right. \tag{61}$$

$$\left. -E[Y_1^2]\right] \tag{62}$$

We can see that here, inside the summation, the first and last terms constitute the variance of h(x).

$$E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] = \sum_{x,y}p(y/x,t)p(x)p(t=1/x)\frac{Y_i}{e_i}\frac{1-e_i}{e_i}f_i \tag{63}$$

$$+\sum_{x,y}p(y/x,t)p(x)p(t=0/x)\frac{0*Y_i}{e_i}\frac{e_i}{e_i}f_i \tag{64}$$

This comes out to be,

$$E\left[\frac{T_iY_i}{e(X_i;\hat{\theta})}\right]\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})}f(X_i;\hat{\psi})\right] = E_{X,Y}\left[\frac{Yf}{e}\right] - E_{X,Y}\left[Yf\right] \tag{65}$$

We know, $f = E[Y/X]$

$$E[Yf] = \sum_{x,y}p(y/x)p(x)y\sum_{y'}p(y'/x)y'$$

$$E[Yf] = \sum_{x}p(x)\sum_{y}p(y/x)p(x)y\sum_{y'}p(y'/x)y'$$

$$E[Yf] = \sum_{x}p(x)E[Y/X]E[Y/X]$$

$$E[Yf] = \sum_{x}p(x)f^2$$

The inner summation and the outer summation will have different indices, assuming independence in Y samples, .

$$E[Yf] = E[f^2]$$

$$Var(f) = Ef^2 = \sum_{x}p(x)E[y/x]^2$$

$$Var(Y) = EY^2 = \sum_y p(y)y^2$$

$$Var(Y) = EY^2 = \sum_x p(x) \sum_y p(y/x)y^2 = Var(f)$$

Since f is an unbiased estimator of Y given X, we can see that the variance of f on its own

$$E\left[\frac{T_i Y_i}{e(X_i;\hat{\theta})}\right]\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right] = E_{X,Y}\left[\frac{Yf}{e}\right] - E_{X,Y}\left[Yf\right] \tag{66}$$

$$-2E\left[\frac{T_i Y_i}{e(X_i;\hat{\theta})}\right]\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right] = -2E\left[\frac{Yf}{e}\right] + 2E\left[f^2\right] \tag{67}$$

$$E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]^2 = E\left[\frac{T_i e_i}{f_i} - f_i\right]^2 \tag{68}$$

$$E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]^2 = \sum p(x) \sum p(y/x)p(t=1/x)\left(\frac{T_i^2 f_i^2}{e_i^2} - 2\frac{T_i f_i}{e_i} + f_i^2\right) + \tag{69}$$

$$\sum p(x) \sum p(y/x)p(t=0/x)\left(\frac{T_i^2 f_i^2}{e_i^2} - 2\frac{T_i f_i}{e_i} + f_i^2\right) \tag{70}$$

$$E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]^2 = \sum p(x) \sum p(y/x)\left(\frac{f_i^2}{e_i} - 2\frac{f_i}{e_i} + p(t=1/x)f_i^2\right) + \tag{71}$$

$$\sum p(x) \sum p(y/x)p(t=0/x)\left(\frac{T_i^2 f_i^2}{e_i^2} - 2\frac{T_i f_i}{e_i} + f_i^2\right) \tag{72}$$

$$E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]^2 = \sum p(x) \sum p(y/x)\left(\frac{f_i^2}{e_i} - 2\frac{f_i}{e_i} + p(t=1/x)f_i^2\right) + \tag{73}$$

$$\sum p(x) \sum p(y/x)(1 - p(t=1/x))f_i^2\Big) \tag{74}$$

$$E\left[\frac{T_i - e(X_i;\hat{\theta})}{e(X_i;\hat{\theta})} f(X_i;\hat{\psi})\right]^2 = \sum p(x) \sum p(y/x)\left(\frac{f_i^2}{e_i} - 2\frac{f_i}{e_i} + p(t=1/x)f_i^2\right) \tag{75}$$

$$-\sum p(x) \sum p(y/x)p(t=1/x)f^2 + \sum p(x) \sum p(y/x)f^2 \tag{76}$$

We look at line 75, the first term in line 75 is $E[f^2/e]$. The second term is - 2E[Y], the expression. The third term and the second will cancel with the first term of line 76. What we will have is the term from 75 will cancel with the first term from 76. The last term will be of 76 will be present in our expression. We compare this with our original expression from 54, 63. We know from positivity, e lies between 0 and 1. Hence the first term above will be greater than the second term. The result will be positive. This is subtracted from the variance of $h(x)$. Therefore, something negative is added to h(x) thus reducing the variance of h(x).

The variance of $h(x) - g(x)$ is thus less. We have used a control variate to reduce the variance of the term for which we want to take an expectation wrt some distribution. The expectation of this extra term to be added is zero. Which enables our aggregate expectation to be the same as our original expectation.

5 Deep Generative Models with f-divergences

f-divergences measure the difference between two given probability distributions. Given two distributions P and Q with density functions p and q, we define the f-divergence,

$$D_f(P\|Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

(a) How can we estimate f-divergences using likelihood ratio estimators as we learned in class? Hint: see lecture slides on GANs.

We set up a generative adversarial network and set up the training procedure with a min max game for the generator and the discriminator. We assign a pseudo label $y = 1$ to x coming from the data generating distribution. i.e. $x \in p_{data}$ and we assign a pseudo label $y = 0$ to $x \in p_{model}$ .i.e. x coming from the distribution corresponding to our model.

$$\frac{p_{data}(x)}{p_{model}(x)} = \frac{p(x/y=1)}{p(x/y=0)}$$

Using Bayes Rule,

$$\frac{p_{data}(x)}{p_{model}(x)} = \frac{p(y=1/x)p(x)p(y=0)}{p(y=0/x)p(x)p(y=1)}$$

Assuming equal samples from p_{data} and p_{model}, we get,

$$\frac{p_{data}(x)}{p_{model}(x)} = \frac{p(y=1/x)}{1-p(y=1/x)}$$

Note that the discriminator will try to predict the probability that the given x is coming from the true data generating distribution .i.e $p(y = 1/x)$. This is the output of the discriminator. We will thus train a GAN to obtain the classifier in form of the discriminator.

Now that we have $p(y = 1/x)$,

Here, we assume access to samples from P (true data) and Q (samples from our model e.g. GAN), but you do not have access to computing the density p(x).

For some x sampled from the distribution of q(x), we can find

$$\frac{p(x)}{q(x)} = \frac{p_{data}(x)}{p_{model}(x)} = (\frac{p(y=1/x)}{1-p(y=1/x)}$$

For our sampled x', we sample from q(x) N times.

$$D_f(P\|Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx = \frac{1}{N}\sum_{i=1}^{N} f\left(\frac{p(x_i')}{q(x_i)}\right)$$

This enables efficient monte carlo estimation of the concerned expectation.

(b) Show how to minimize $D_f(P\|Q_\theta)$ with respect to θ using the estimate from your solution to (1).

The objective which is the f divergence is in terms on the function of the output of the discriminator. We would want to minimize the f divergence. This can help us make efficient backpropagation to achieve the optimization with respect to the parameters of the generator. For some data point x sampled from q(x). We take the monte carlo estimate wrt this data point.

$$D_f(P||Q) = \frac{p(x)}{q(x)} = \frac{p_{data}(x)}{p_{model}(x)} = \frac{p(y=1/x)}{1-p(y=1/x)}$$

$$\log D_f(P||Q) = \frac{p(x)}{q(x)} = \frac{p_{data}(x)}{p_{model}(x)} = \log p(y=1/x) - \log(1-p(y=1/x))$$

This is the loss wrt to the output of the discriminator for some x from the distribution of q. We can now perform the minimization of the divergence but stochastic gradient descent.

(c) Let $\mathcal{Q}$ be the family of distributions that Q_θ lives in: $Q_\theta \in \mathcal{Q}$. Assume that $P \notin \mathcal{Q}$. Compare the KL and Reverse KL. What are properties that each f-divergence imposes its corresponding minimizer, $Q_\theta^\star$, i.e.

$$Q_\theta^\star = \arg\min_{Q_\theta \in \mathcal{Q}} D_f(P \| Q_\theta).$$

Name	$D_f(P\|Q)$	$f(u)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$

Hint: Consider $u = p(x)/q(x)$ very small, near 1, and large.

We first consider the Reverse KL divergence,

$$D_r(P|Q) = \int q(x) \log \frac{q(x)}{p(x)} dx$$

We know the minimization of $D_r(P|Q)$ will occur when p and q are the same distribution. The minimization of the divergence will tend to bring similarity between p and q.

When u is 1, the logarithm is 0. The loss assigned to the objective is 0. As compared to when p and q are different, if q is larger than p, the loss will get some positive number for the loss. If p is larger than q, while we get a negative number here,p can only be larger than q to a certain extent since we are modelling a distribution over x.

Here, when p(x) tends to zero, q(x) is finite ,u is very large. The Divergence value will be very high and thus the penalty imposed on the objective of minimization will be high. Therefore, the parameters θ of q(x) will be learned such that q(x) avoids assigning probability mass where p(x) is very small.

E.g. if p(x) is a Gaussian with some mean and covariance. q(x) is a product of independent gaussians. q(x) could tend to concentrate around the mean of p(x).

We consider the KL divergence,

$$D(P|Q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Here, u will be very large when, for a finite p(x), we have assigned a very small probability mass to q(x).

Thus, the distribution of q(x) and the parameters θ will be learned such that the distribution is relatively spread out over the regions where p has a probability mass, this is as compared to the reverse KL divergence.

6 Reinforcement Learning, Reward design, Curriculum Learning

Suppose that you have a robot that should learn to move from any of a set of starting positions $s_0 \in \mathcal{S}_0$ to a goal position G. The robot can move by performing a sequence of small, low-level continuous actions, such as rotating its joints. However, you do not know how to explicitly program a sequence of actions that will move the robot from any s_0 to G. You decide to use a reinforcement learning algorithm. Your RL algorithm uses the policy gradient to learn a policy $\pi_\theta(a|s)$ that maps from states s to distributions over actions a. You consider learning episodes of a finite number of T steps.

(a) You start by designing the Markov Decision Process (MDP) that defines the robot's learning environment. The robot receives reward $R = 1000$ when it reaches location G and $R = 0$ upon entering any other position. What happens as the robot explores in this environment? What is the Monte Carlo estimate of the policy gradient when G is not reached in T steps in any of the sample trajectories?

Consider the formulation of the gradient for the objective function which we want to maximize. Our goal is to maximize the reward.

We start with our future reward estimate for actor critic methods.

$$L(\theta) = E_{\pi_\theta}[r_\tau]$$

$$r_\tau = \sum_t \gamma_t r_t$$

After some simplification,

$$\nabla_\theta L(\theta) = \sum_{i=1}^{T} E_{\pi_\theta}[(\sum_{t'=t+1}^{t} r_{t'}) \nabla_\theta \log \pi_\theta(a_t/s_t)]$$

Where $\sum_{t'=t+1}^{t} r_{t'}$ is the future reward estimate. We can introduce a function $Q(s_t, a_t)$ which denotes the expected value of the future reward estimate given a state, action at a time step t.

We can then modify our gradient. When we introduce a value function of the state as done in the literature and then formulate our objective in terms of the advantage function which is the Q function minus the value function.

The expected value of the value function is zero. This is similar to the previous question on control variates.

Our iterative formulation would mean that we would only need to find $V_t^\pi(s_t)$

For our gradient computations, when we perform our monte carlo estimation of the quantity which is used to represent the part which captures the future reward estimate, we will sample various trajectories. We then get our monte carlo estimate for our value function for a state vat some time steps in terms of the sums of the rewards at future time steps averaged across the sampled trajectories.

If we are not particular about variance, we can think of a monte carlo estimate of the future reward directly. This again involves sums over rewards obtained at various time steps for the sampled trajectories.

But, for our trajectories, since we don't reach the goal state, the future reward estimate will be 0. There will always be a zero reward value at every time steps.

The monte carlo estimate of the policy gradient will be 0.

(b) Suppose the robot must start in $\mathcal{S}_0$ for all learning trajectories. In which ways could you alter the MDP environment to improve exploration and improve gradients? Note: this is an open-ended question... it is something you would encounter when applying RL in real life and does not have one right answer.

We may have to change our reward scheme to obtain efficient critics. Since this is a robot locomotion problem, if feasible, we can define the notion of spatial similarity between the goal state and other states. We can then adjust the reward function accordingly. Rather than random sampling, we could change the procedure by which we sample in a way that gives importance to trajectories containing the goal state G.

We could work directly in terms of a Q function. Using fitted Q iteration, we can use a parameterized function to learn Q based on the obtained minimization after the approximation on the expected V value in terms of a maximization over the states of Q as given in the literature. This does not require samples from a particular the policy.

We can use online Q learning. It only needs one sample at a time.

In terms of exploitation and exploration, we would want to use an annealing schedule in our problem. e.g. We could have a temperature variable set for our policy distribution where we sample the actions given the states. This can be subject to change over the iterations of our training. We want an efficient schedule. We could use deterministic annealing or we could go for variational tempering. We could set the temperature as a latent variable participating in our sampling distribution. We can assign to it a prior distribution. Such a formulation can enable the introduction of the temperature distribution parameters in our optimization objective.

The key is that we could learn an efficient schedule from the data for the dilemma of exploration vs exploitation rather than setting a heuristic based deterministic schedule. We could do hyperparameter tuning for the temperature parameter.

Based on the set up of the experiment, we will have to check for feasibility and tractability after setting up well defined goals.

hw4_question2

April 24, 2022

[1]:
```
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
zero = torch.zeros(1)
one = torch.ones(1)
```

[2]:
```
def generate_samples(N,u):
  samples={}
  z_samples=torch.normal(mean= u*torch.ones(N),std= torch.ones(N))
  epsilon_samples= torch.normal(mean= torch.zeros(N),std= torch.ones(N))
  samples["z"]=z_samples
  samples["epsilon"]=epsilon_samples

  return samples
```

[3]:
```
generate_samples(10,2)
```

[3]:
```
{'epsilon': tensor([ 0.7554,  0.6631,  0.9468,  1.6838, -0.5048, -0.9781,
-2.1833, -1.0085,
          0.0492, -0.9947]),
 'z': tensor([ 1.8783,  2.0542,  1.0016,  1.1221,  1.0930, -0.0632,  1.7621,
1.2709,
          0.9462,  2.4385])}
```

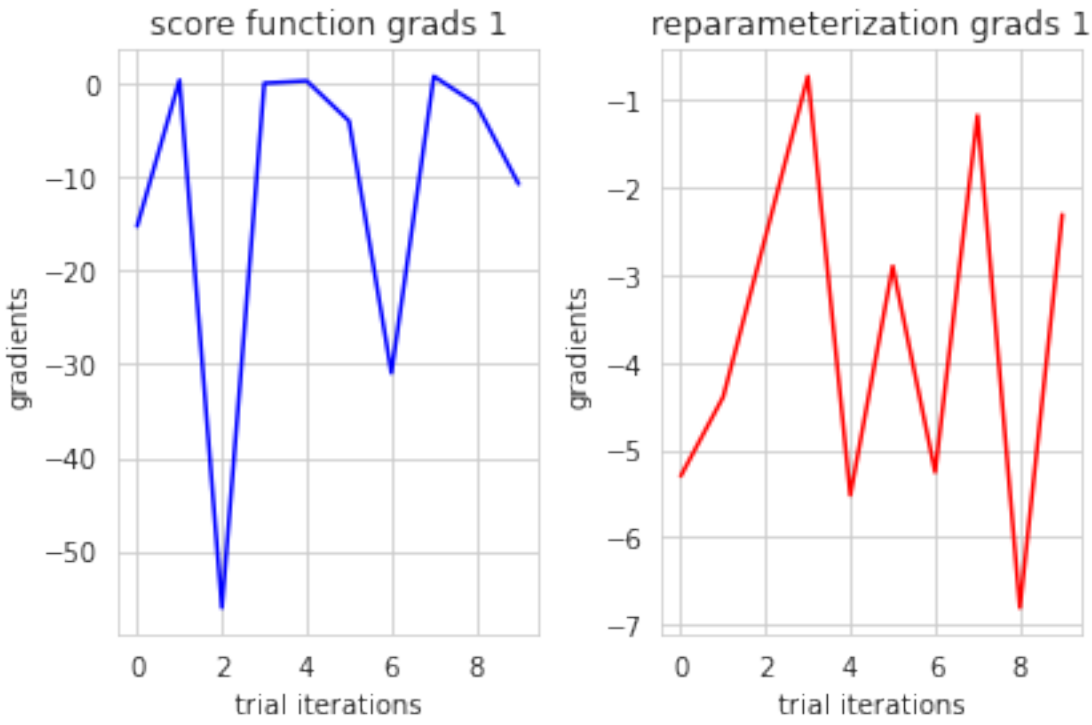
[4]:
```
def find_gradients(M,N,u):
  z_grads={}
  e_grads={}
  for i in range(M):
     samples= generate_samples(N,u)
     s=samples["z"]
     diff= s - u
     f_z = -torch.square(s)
     z_grads[i]= torch.dot(diff,f_z)/N
```

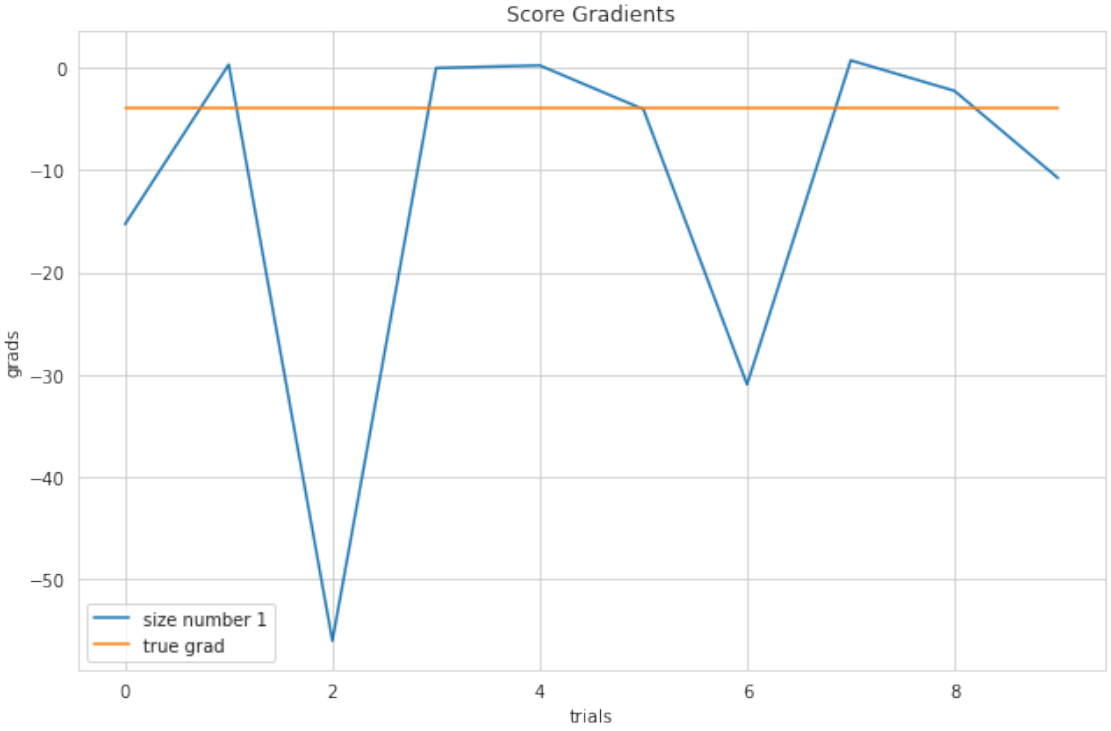
```
    e=samples["epsilon"]
    e_grads[i]= torch.sum(-2*e -2*u*torch.ones(N))/N


  return z_grads,e_grads
```

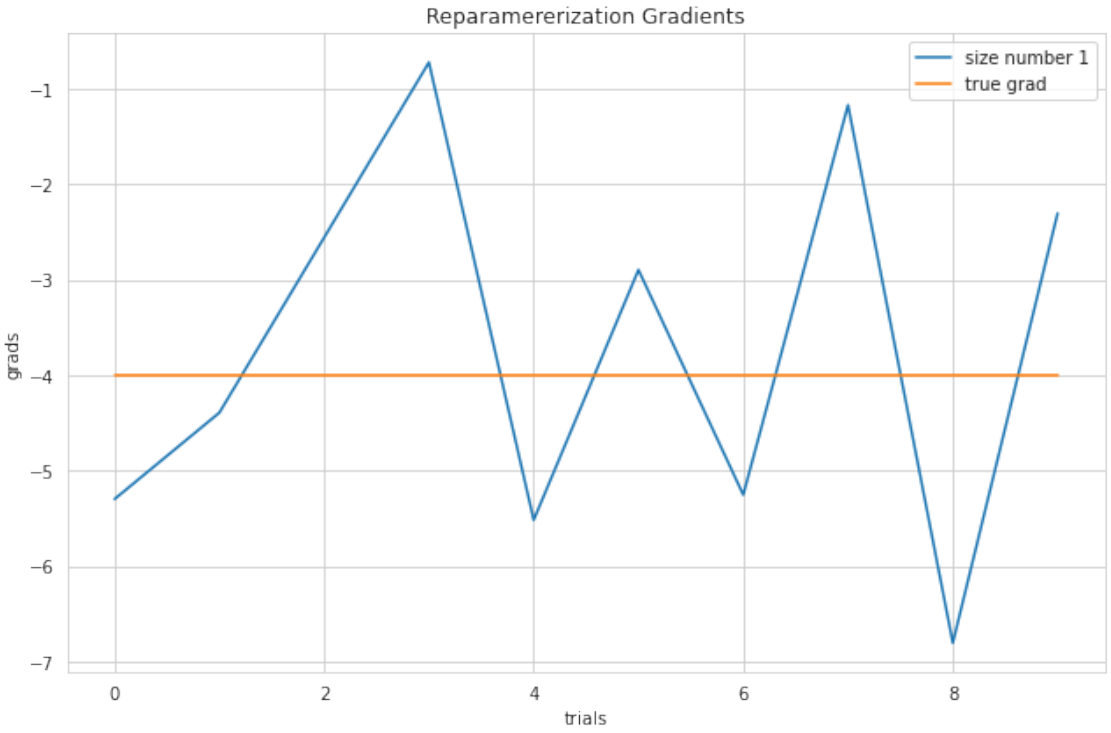
[5]:
```
import matplotlib.pyplot as plt

def plot_gradients(M,sizes,u):

  Y_Z=[]
  Y_E=[]
  for N in sizes:
    z_grads,e_grads=find_gradients(M,N,u)
    x=torch.tensor([i for i in range(M)])

    y_z=torch.tensor([z_grads[i] for i in range(len(z_grads))])
    y_e=torch.tensor([e_grads[i] for i in range(len(e_grads))])
    Y_E.append(y_e)
    Y_Z.append(y_z)

    fig, axes = plt.subplots(nrows=1, ncols=2)

    for i in range(len(axes)):

      if(i%2==0):

            axes[i].plot(x.to('cpu').numpy(), y_z.to('cpu').numpy(), 'b')
            axes[i].set_xlabel('trial iterations')
            axes[i].set_ylabel('gradients')
            axes[i].set_title('score function grads'+ " "+ str(N))




      else:
            axes[i].plot(x.to('cpu').numpy(), y_e.to('cpu').numpy(), 'r')
            axes[i].set_xlabel('trial iterations')
            axes[i].set_ylabel('gradients')
            axes[i].set_title('reparameterization grads'+" "+str(N))



    plt.tight_layout()
    plt.show()

    j=1
```

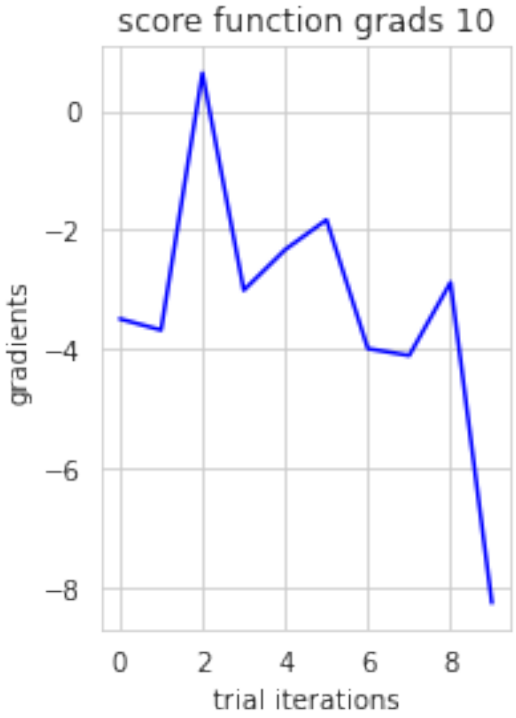
```
    fig, axes = plt.subplots(figsize=(9,6))
    for y in Y_Z:

        axes.plot(x.to('cpu').numpy(), y.to('cpu').numpy(),label="size number"+"␣
↪"+str(j))
        j+=1


    t=-torch.ones(M)*2*u

    axes.plot(x.to('cpu').numpy(),t.to('cpu').numpy(),label="true grad")
    axes.set_xlabel('trials')
    axes.set_ylabel('grads')
    axes.set_title('Score Gradients');
    plt.legend()
    plt.tight_layout()
    plt.show()

    fig, axes = plt.subplots(figsize=(9,6))
    i=1
    for y in Y_E:

        axes.plot(x.to('cpu').numpy(), y.to('cpu').numpy(),label="size number"+"␣
↪"+str(i))
        i+=1


    t=-torch.ones(M)*2*u

    axes.plot(x.to('cpu').numpy(),t.to('cpu').numpy(),label="true grad")
    axes.set_xlabel('trials')
    axes.set_ylabel('grads')
    axes.set_title('Reparamererization Gradients');
    plt.legend()
    plt.tight_layout()
    plt.show()
```

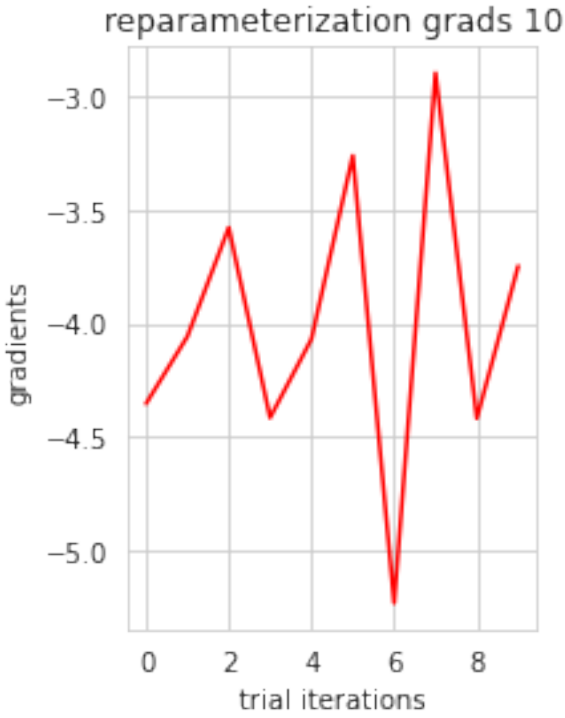
[6]:
```
plot_gradients(10,[1,10,100,1000],2)
```

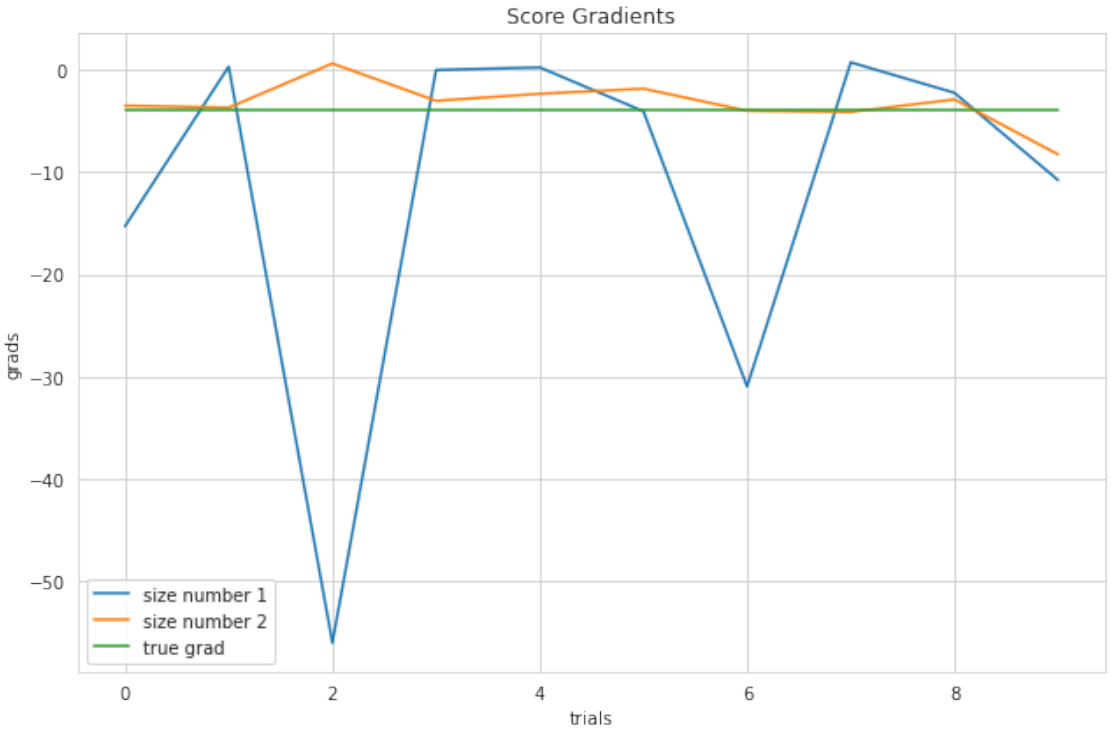

score function grads 1
gradients
trial iterations
reparameterization grads 1
gradients
trial iterations

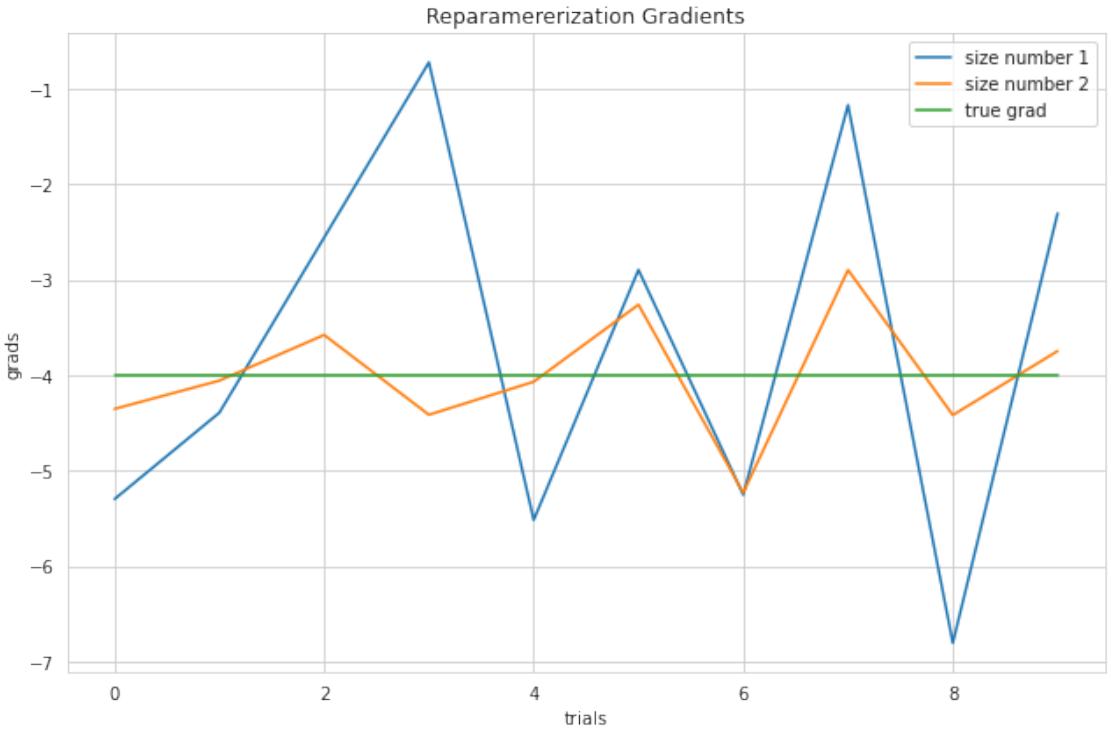

Score Gradients
grads
trials
size number 1
true grad

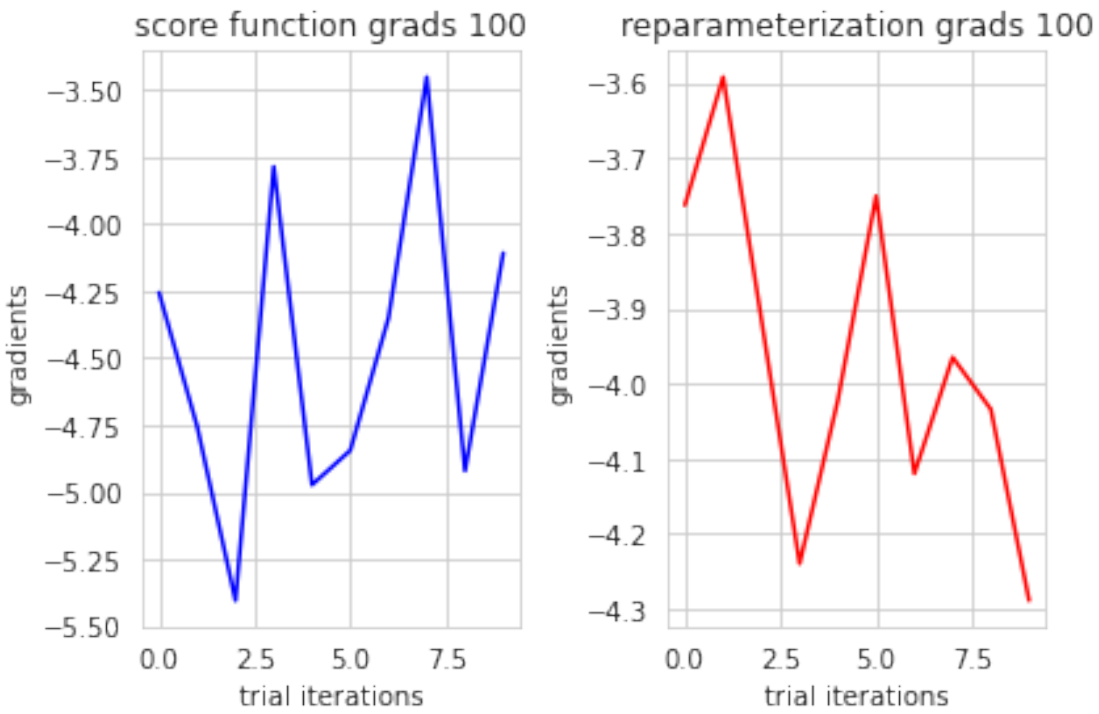

Reparamererization Gradients
size number 1
true grad
grads
trials

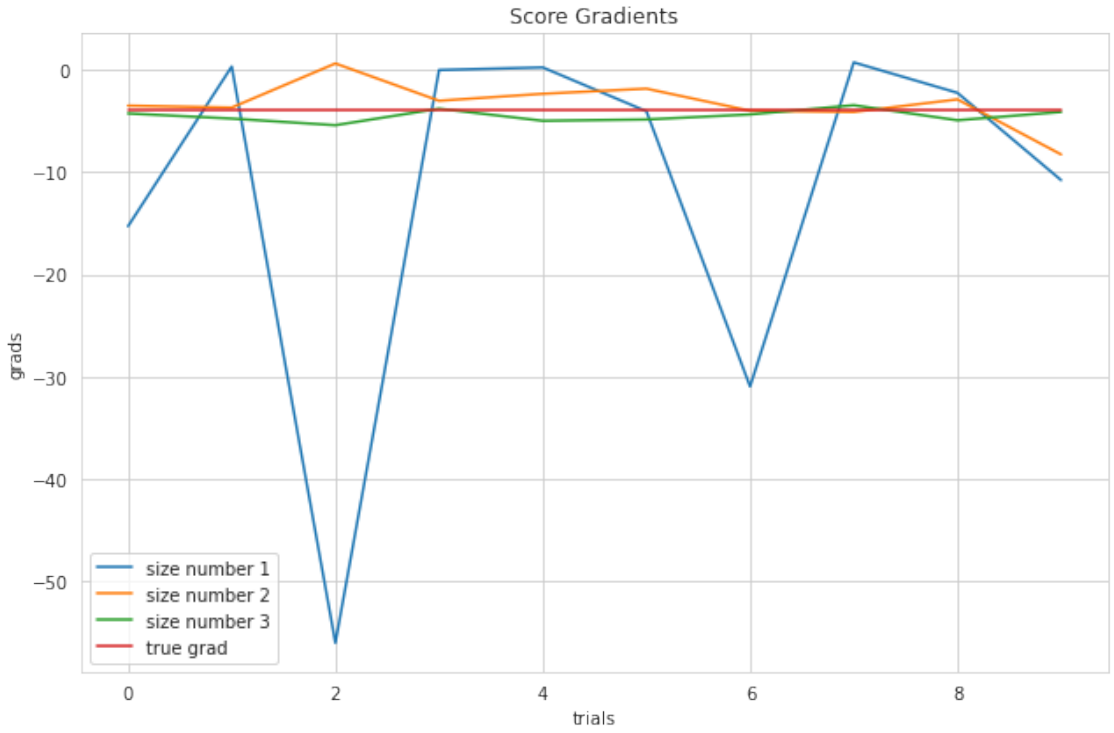

score function grads 10
gradients
trial iterations

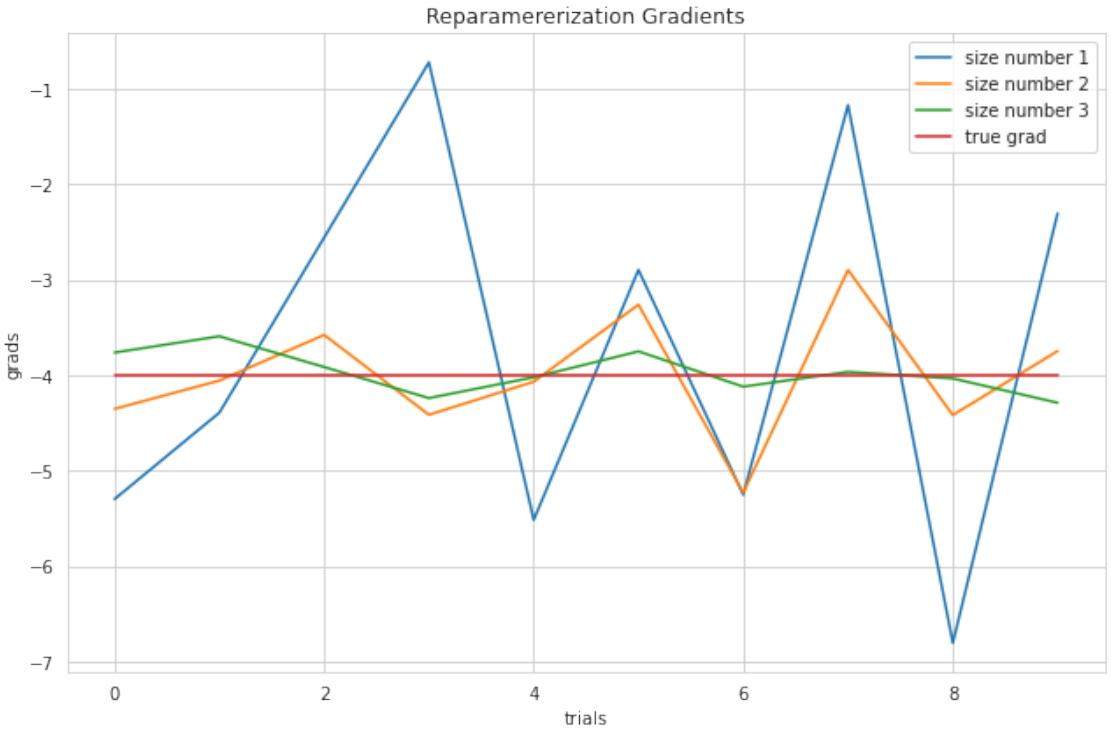

reparameterization grads 10
gradients
trial iterations

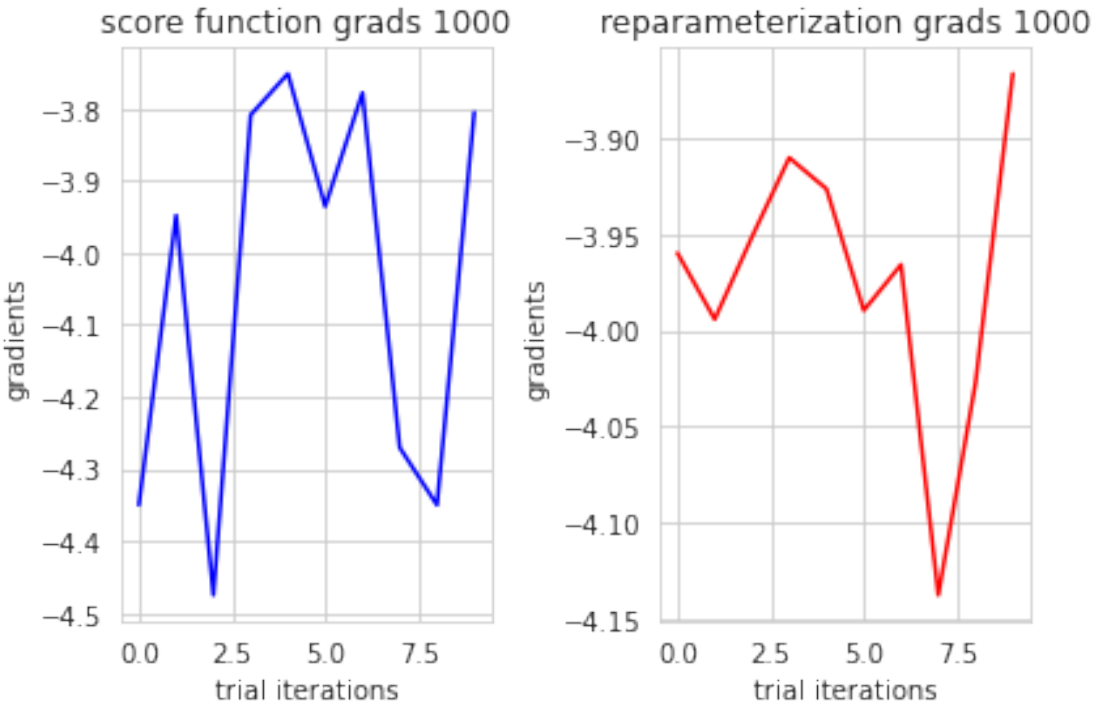

Score Gradients
grads
trials
size number 1
size number 2
true grad
0
−10
−20
−30
−40
−50
0
2
4
6
8

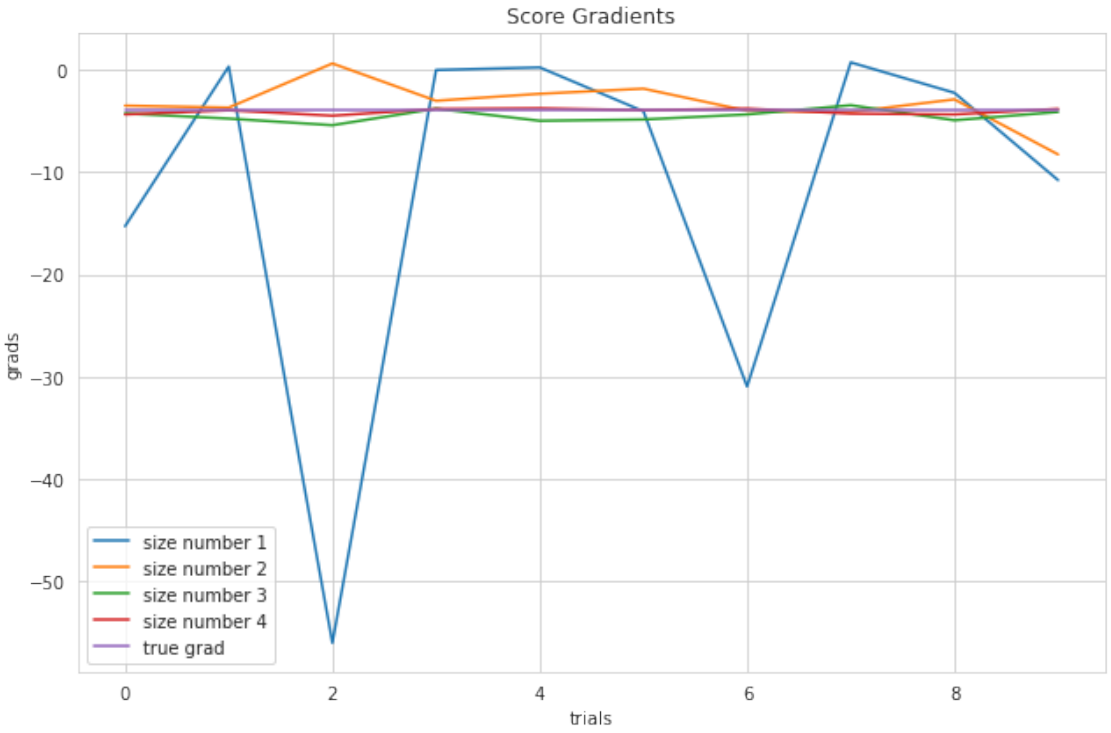

Reparamererization Gradients
grads
trials
size number 1
size number 2
true grad
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8

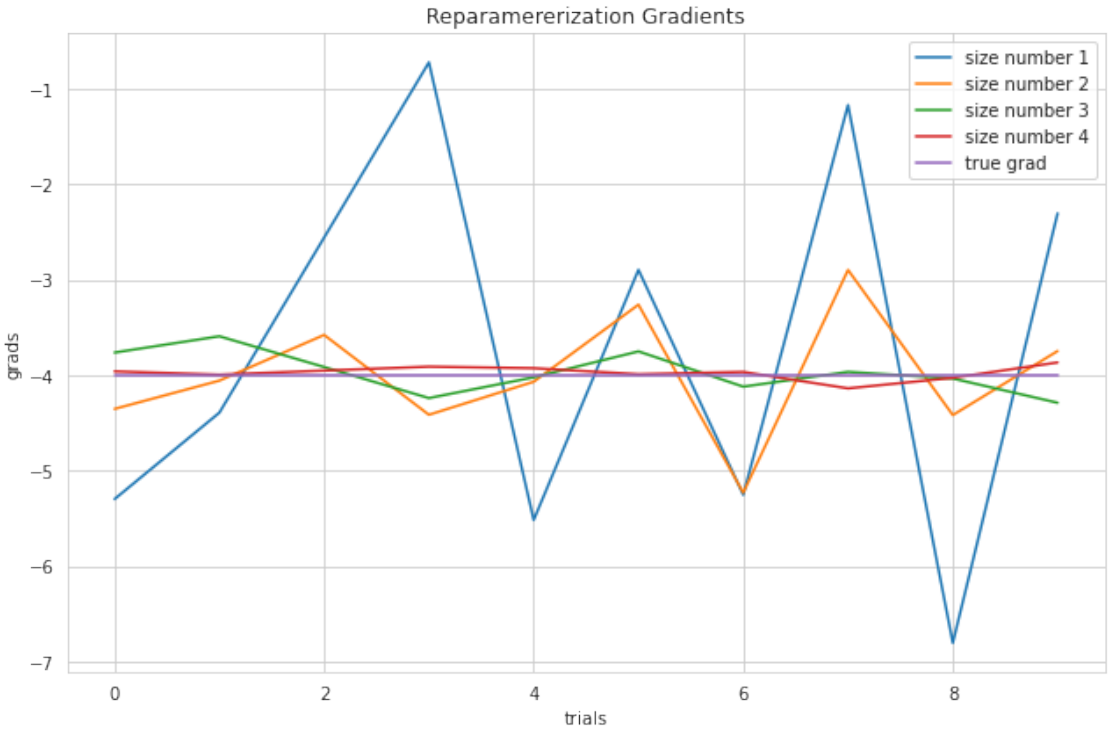

score function grads 100
reparameterization grads 100
gradients
trial iterations
−3.50
−3.75
−4.00
−4.25
−4.50
−4.75
−5.00
−5.25
−5.50
−3.6
−3.7
−3.8
−3.9
−4.0
−4.1
−4.2
−4.3
0.0
2.5
5.0
7.5


Score Gradients
grads
trials
size number 1
size number 2
size number 3
true grad
0
−10
−20
−30
−40
−50
0
2
4
6
8


Reparamererization Gradients
size number 1
size number 2
size number 3
true grad
grads
trials
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8


score function grads 1000
gradients
trial iterations
−3.8
−3.9
−4.0
−4.1
−4.2
−4.3
−4.4
−4.5
0.0
2.5
5.0
7.5
reparameterization grads 1000
gradients
trial iterations
−3.90
−3.95
−4.00
−4.05
−4.10
−4.15
0.0
2.5
5.0
7.5


Score Gradients
grads
trials
size number 1
size number 2
size number 3
size number 4
true grad


Reparamererization Gradients
grads
trials
size number 1
size number 2
size number 3
size number 4
true grad

[7]:
```
def find_variances(M,N,u):

  z_variances=[]
  e_variances=[]
  for i in range(M):

    z_grads, e_grads =find_gradients(M,N,u)
    y_z = torch.tensor([z_grads[i] for i in range(len(z_grads))])

    y_e = torch.tensor([e_grads[i] for i in range(len(z_grads))])

    z_variances.append(torch.var(y_z))
    e_variances.append(torch.var(y_e))

  return torch.tensor(z_variances), torch.tensor(e_variances)
```

[8]:
```
def plot_variances(M,N,u):
  z_variances, e_variances = find_variances(M,N,u)

  fig, axes = plt.subplots(figsize=(9,6))

  avg_z=torch.sum(z_variances)/len(z_variances)
  avg_e=torch.sum(e_variances)/len(e_variances)

  t_z= torch.ones(len(z_variances))*avg_z

  t_e= torch.ones(len(e_variances))*avg_e

  x=torch.tensor([i for i in range(len(z_variances))])

  axes.plot(x.to('cpu').numpy(), z_variances.to('cpu').numpy(),label="z␣
 ↪variances"+" "+str(N))
  axes.plot(x.to('cpu').numpy(), e_variances.to('cpu').numpy(),label="e␣
 ↪variances"+" "+str(N))

  axes.plot(x.to('cpu').numpy(), t_z.to('cpu').numpy(),label="avg z variance")

  axes.plot(x.to('cpu').numpy(), t_e.to('cpu').numpy(),label="avg e variance")

  axes.set_xlabel('trials')
  axes.set_ylabel('variance')
  axes.set_title('Variances for Gradients');
  plt.legend()
  plt.tight_layout()
  plt.show()
```

[9]:
```
def plot_variances_aplha(M,Sizes,u):
  fig, axes = plt.subplots(figsize=(9,6))

  for N in Sizes:
    #print("Size"+"" + str(N))
    #plot_variances(M,N,u)
    z_variances, e_variances = find_variances(M,N,u)



    avg_z=torch.sum(z_variances)/len(z_variances)
    avg_e=torch.sum(e_variances)/len(e_variances)

    t_z= torch.ones(len(z_variances))*avg_z

    t_e= torch.ones(len(e_variances))*avg_e

    x=torch.tensor([i for i in range(len(z_variances))])

    axes.plot(x.to('cpu').numpy(), z_variances.to('cpu').numpy(),label="z␣
↪variances"+" "+str(N))
    axes.plot(x.to('cpu').numpy(), e_variances.to('cpu').numpy(),label="e␣
↪variances"+" "+str(N))


  axes.set_xlabel('trials')
  axes.set_ylabel('variance')
  axes.set_title('Variances for Gradients');
  plt.legend()
  plt.tight_layout()
  plt.show()
```

We look at the average variance per some N as 100, 10 cases. true mean as 2.

[10]:
```
z_variances, e_variances = find_variances(10,100,2)

avg_z=torch.sum(z_variances)/len(z_variances)
avg_e=torch.sum(e_variances)/len(e_variances)
```

[11]:
```
avg_z
```

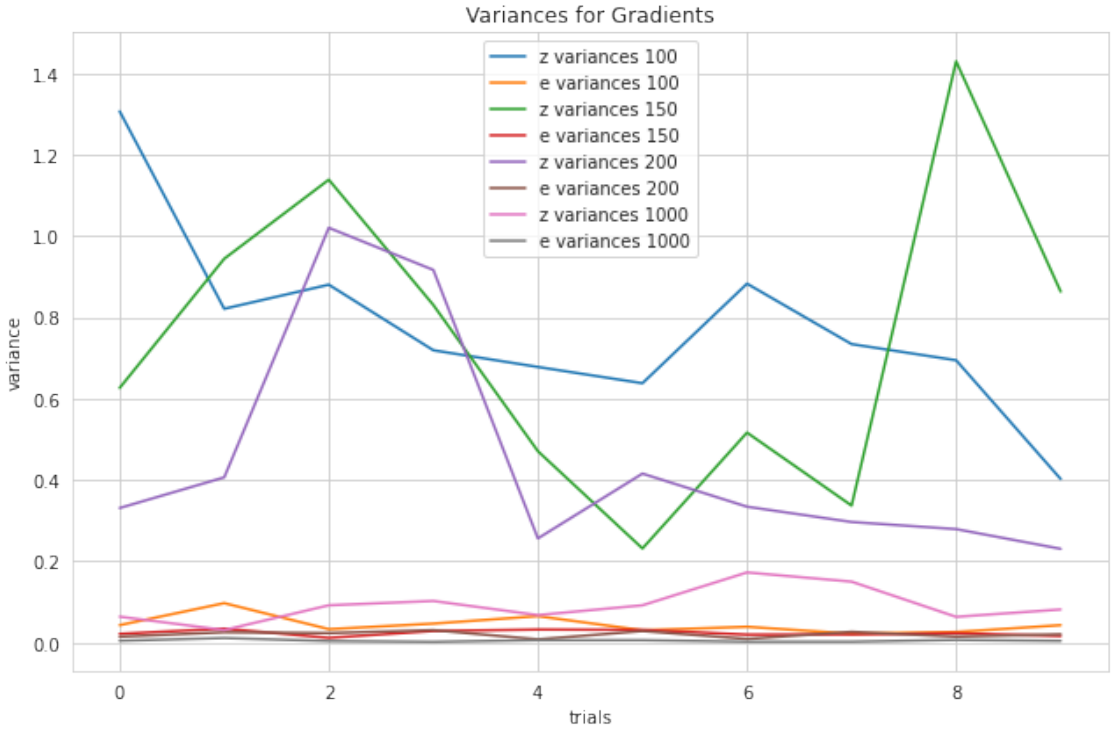
[11]:
```
tensor(0.8094)
```

[12]:
```
avg_e
```

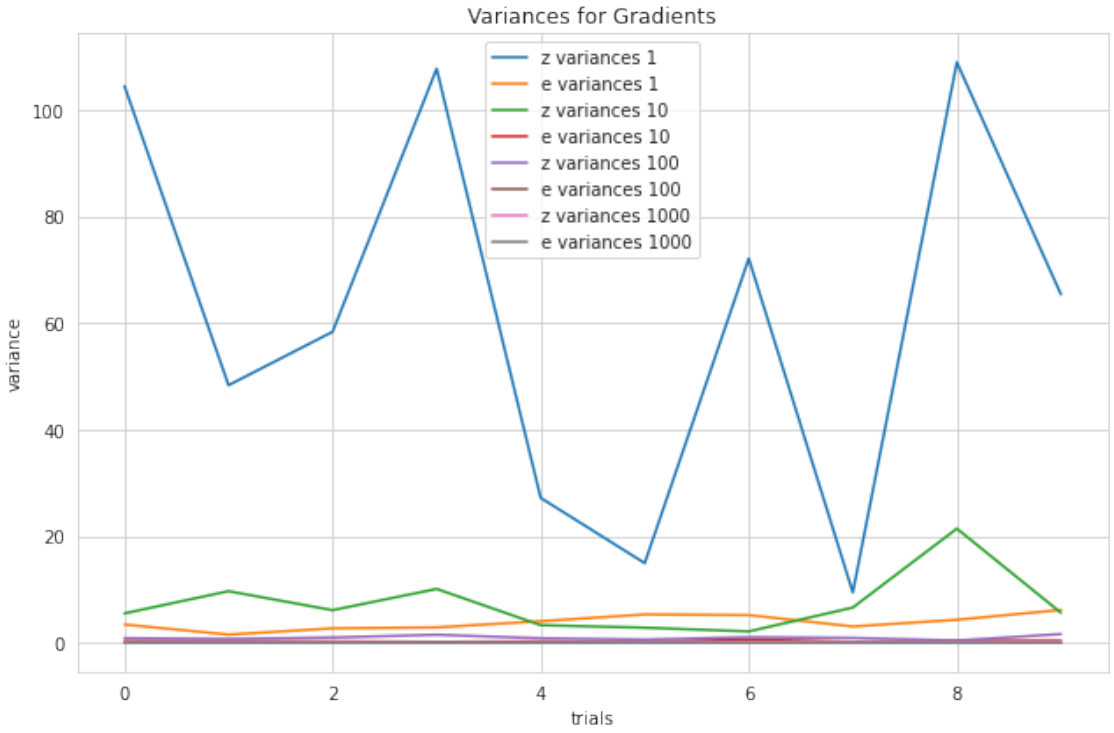
[12]:
```
tensor(0.0406)
```

We plot the variances

[13]:
```
plot_variances_aplha(10,[100,150,200,1000],2)
```



[14]:
```
plot_variances_aplha(10,[1,10,100,1000],2)
```


Variances for Gradients
z variances 1
e variances 1
z variances 10
e variances 10
z variances 100
e variances 100
z variances 1000
e variances 1000
variance
trials
0
20
40
60
80
100
0
2
4
6
8