Notes on deep active subspaces

W.N. Edeling

February 19, 2021

Contents

1 Active subspaces

The Active Subspace method, introduced in [2], is a class of methods for forward propagation of uncertainty in high-dimensional input spaces. It attempts to circumvent the curse of dimensionality by dimension reduction of the input space. Specifically, it projects the input vector $\mathbf{x} \in \mathbb{R}^D$ to a lower-dimensional subspace $\mathbf{y} \in \mathbb{R}^d$, via a tall-and-skinny matrix $W_1 \in \mathbb{R}^{D\times d}$ of orthogonal basis vectors, such that $W_1^T W_1 = I_d$, where $d < D$ and I_d is the d-dimensional identity matrix. The active subspace is thus given by

$$\mathbf{y} = W_1^T \mathbf{x} \in \mathbb{R}^d, \tag{1}$$

where the main idea is that the dimension reduction simplifies the task of obtaining an accurate surrogate model. If we denote this surrogate by g, we thus want to find a model that satisfies

$$f(\mathbf{x}) \approx g\left(\mathbf{y}\right) = g\left(W_1^T\ \mathbf{x}\right). \tag{2}$$

2 Classical means of finding W_1

Because $\mathbf{y}$ is a linear transformation of the inputs $\mathbf{x}$, it opens up the possibility of findings directions along which the model varies most. This is especially useful if a model varies significantly in a direction that is not aligned with the coordinate axes of $\mathbf{x}$. In 'classical' active subspaces [2], dimension reduction is achieved by rotating the coordinate system such that it is aligned with the directions of most variability, after which only the most dominant directions are retained. To find these directions, the following average gradient matrix is constructed:

$$C = \int \left(\nabla f\left(\mathbf{x}\right)\right)\left(\nabla f\left(\mathbf{x}\right)\right)^T p(\mathbf{x})\mathbf{dx} \tag{3}$$

Here, $p(\mathbf{x})$ is the chosen probability density function (pdf) of the inputs $\mathbf{x}$. Since C is a symmetric, positive semi-definite matrix, it has the following spectral decomposition

$$C = [W_1 W_2]\begin{bmatrix}\Lambda_1 & 0\\ 0 & \Lambda_2\end{bmatrix}[W_1 W_2]^T.$$

Hence the matrix W_1, used to project $\mathbf{x}$ to $\mathbf{y}$ in (1), are the first d eigenvectors of C, which in turn correspond to the d largest eigenvalues in Λ_1. These d eigenvectors form an orthonormal basis aligned with the directions of most variability. Note that C is averaged over $p(\mathbf{x})$, and that in practise, the integral in (3) is often approximated using a Monte Carlo approach.

The approach described here is intuitive, and has nice theoretical properties, such as computable error bounds [2]. The downside however, is that the gradient $\nabla f(\mathbf{x})$ must be available. This requires the availability of an adjoint solver, or one must approximate the gradients using for instance finite differences. This downside has prompted the development of other active subspace methods which do not require access to the gradient. Some of these methods involve Gaussian processes [5], whereas others use deep learning. We will focus on the latter.

3 Deep active subspaces

In [7], an approach is described in which artificial neural networks (ANNs) are used for g, and where W_1 is found using back propagation. Like the classical active subspace method, the construction (2) is retained. Moreover, the column vectors of W_1 still form an orthogonal basis. The difference is that the column vectors of W_1 are no longer the eigenvectors of the gradient matrix C, but instead are constructed using Gram-Schmidt orthogonalization. As such, W_1 is parametrized by a matrix Q of the same dimension ($\in \mathrm{R}^{D\times d}$), where the non-orthonormal column vectors $\mathbf{q}_i \in \mathbb{R}^D$ are made orthonormal via

$$\mathbf{w}_i = \mathbf{q}_i - \sum_{j=1}^{i-1}\left(\frac{\mathbf{w}_j^T\mathbf{q}_i}{\mathbf{w}_j^T\mathbf{w}_j}\right)\mathbf{w}_j, \quad i = 1, \cdots, d. \tag{4}$$

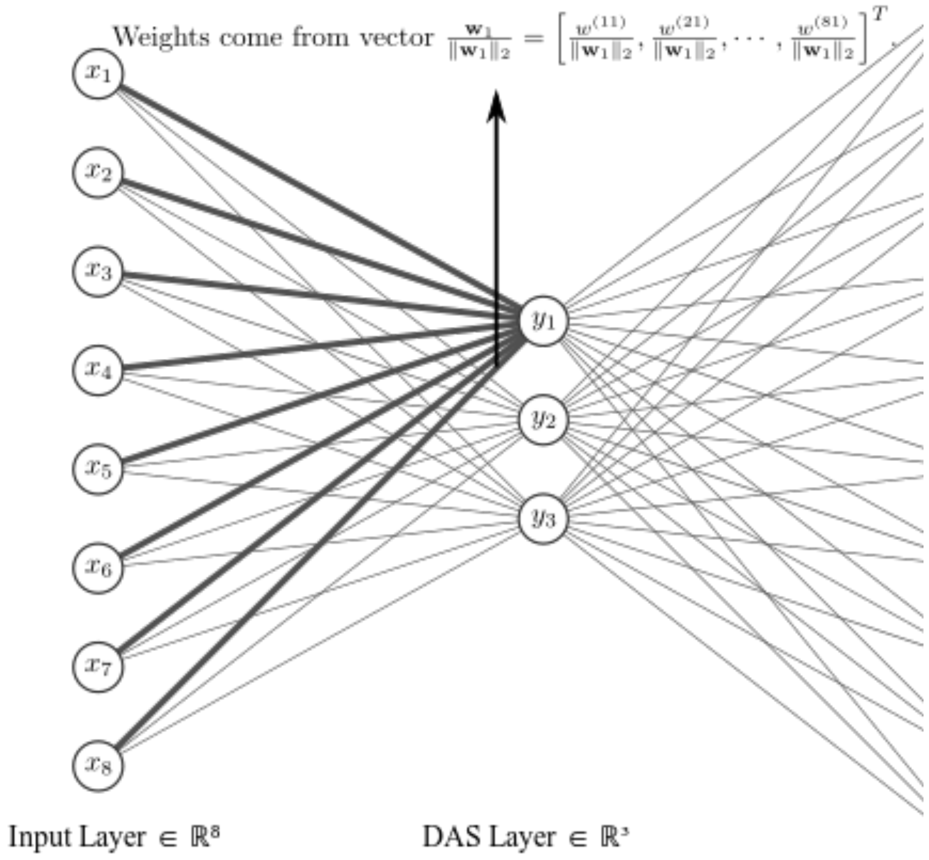
That is, we start with $\mathbf{w}_1 := \mathbf{q}_1$, and for all subsequent vectors $\mathbf{q}_i$ we subtract the projections of $\mathbf{q}_i$ onto each vector $\mathbf{w}_j$ which has previously been orthogonalized. This leaves us with a orthogonal basis $[\mathbf{w}_1(\mathbf{q}_1)\ \ \mathbf{w}_2(\mathbf{q}_1,\mathbf{q}_2)\ \ \cdots\ \ \mathbf{w}_d(\mathbf{q}_1,\mathbf{q}_2\cdots,\mathbf{q}_d)]$. Finally, to obtain an orthonormal basis, each column vector is divided by its length. such that our final weight matrix becomes

$$W_1(Q) = \left[\frac{\mathbf{w}_1(\mathbf{q}_1)}{\|\mathbf{w}_1(\mathbf{q}_1)\|_2}\ \ \frac{\mathbf{w}_2(\mathbf{q}_1,\mathbf{q}_2)}{\|\mathbf{w}_2(\mathbf{q}_1,\mathbf{q}_2)\|_2}\ \ \cdots\ \ \frac{\mathbf{w}_d(\mathbf{q}_1,\mathbf{q}_2\cdots,\mathbf{q}_d)}{\|\mathbf{w}_d(\mathbf{q}_1,\mathbf{q}_2\cdots,\mathbf{q}_d)\|_2}\right]. \tag{5}$$

Note that the projection $\mathbf{y} = \Phi\left(W_1^T\mathbf{x}\right) = W_1^T\mathbf{x}$ also occurs in a layer of a neural network if the activation function $\Phi(\cdot)$ is linear. Thus, we can interpret the matrix W_1 as a weight matrix of the first hidden layer (with d neurons and linear activation), connected to an input layer through which $\mathbf{x}$ is passed. Each column vector $\mathbf{w}_i$ contains the all weights connecting the input layer to the i-th neuron of the first hidden layer, see Figure 1. Since the first hidden layer has only d neurons, and its weight matrix is determined from a Gram-Schmidt procedure, we call the layer the Deep Active Subspace (DAS) layer.

The surrogate $g(\mathbf{y})$ is the ANN from the DAS layer onward, see Figure 2. Each hidden layer has a weight matrix $W_i \in \mathbb{R}^{p+1\times p}$, assuming that all hidden layers have p neurons plus 1 bias neuron. As per usual, these weight matrices are optimized through the back propagation algorithm [1], in which the gradient $\partial L/\partial W_i$ is computed, where L is the loss function.

The situation in the DAS layer is different. Since $W_1 = W_1(Q)$, we need to optimize Q instead of the weight matrix, and therefore back propagation requires $\partial L/\partial Q$ instead of $\partial L/\partial W_1$. The authors of [7]



Weights come from vector $\frac{\mathbf{w}_1}{\|\mathbf{w}_1\|_2} = \left[\frac{w^{(11)}}{\|\mathbf{w}_1\|_2}, \frac{w^{(21)}}{\|\mathbf{w}_1\|_2}, \cdots, \frac{w^{(81)}}{\|\mathbf{w}_1\|_2}\right]^T$
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
y_1
y_2
y_3
Input Layer $\in \mathbb{R}^8$
DAS Layer $\in \mathbb{R}^3$

Figure 1: Diagram showing a Deep Active Subspace (DAS) Layer with $D = 8$ and $d = 3$. The thick lines contain the weights coming from the first column vector of $W_1(Q)$. Likewise, the second column vector contains the weights of all lines ending at neuron y_2, etc.

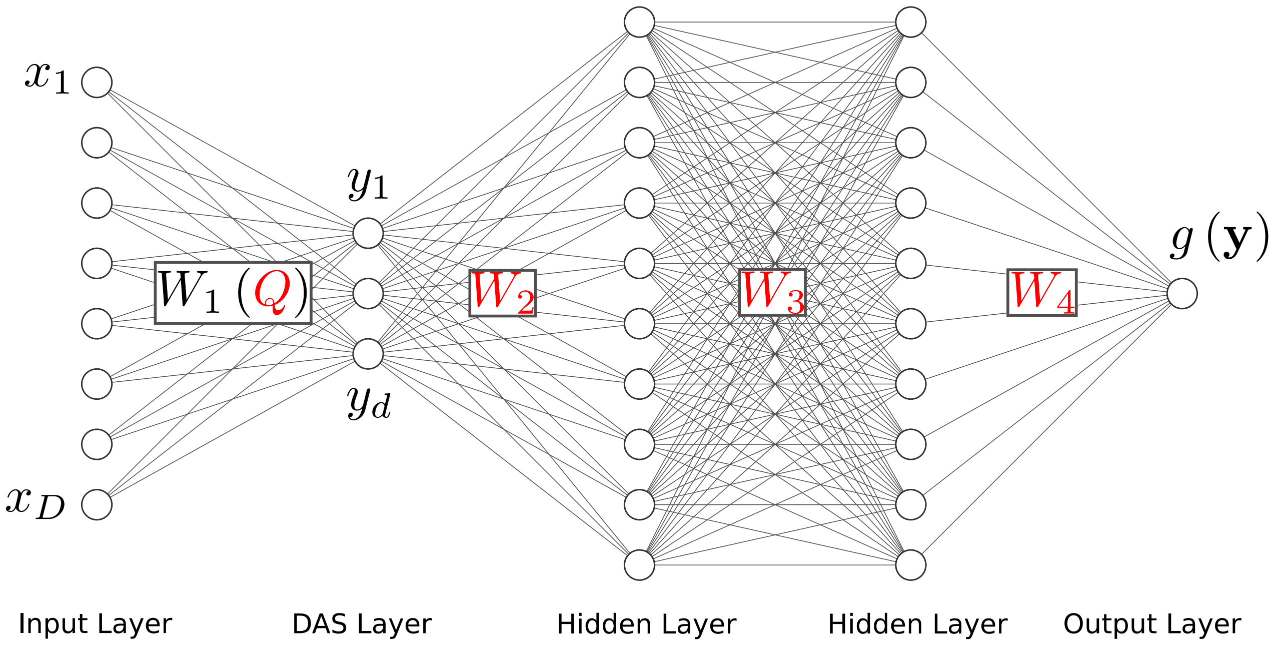


Figure 2: A full schematic of a DAS surrogate. The red matrices (Q, W_2, W_3 and W_4) are optimized using stochastic gradient descent and back propagation.

suggest to use automatic differentiation. This does make sense, since although $\mathbf{w}_k/\|\mathbf{w}_k\|_2$ is algebraic and differentiable, it quickly becomes a complicated expression involving a very large number of q_{ij} terms. Here, q_{ij} are the D entries, $i = 1, \cdots, D$, of $\mathbf{q}_j$. That said, we will show that we can use matrix calculus to find a simple expression for $\partial L/\partial Q$. We will do so in the next section, first to give us more insight, and perhaps to find ways of improving computational efficiency.

4 Differentiating Gram Schmidt

The gradient matrix with respect to Q is given by

$$\frac{\partial L}{\partial Q} = \begin{bmatrix} \frac{\partial L}{\partial q_{11}} & \cdots & \frac{\partial L}{\partial q_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial q_{D1}} & \cdots & \frac{\partial L}{\partial q D_{1d}} \end{bmatrix} \in \mathbb{R}^{D\times d}, \tag{6}$$

where each entry is computed via the chain rule as

$$\frac{\partial L}{\partial q_{ij}} = \frac{\partial L}{\partial w_{11}} \cdot \frac{\partial w_{11}}{\partial q_{ij}} + \frac{\partial L}{\partial w_{12}} \cdot \frac{\partial w_{12}}{\partial q_{ij}} + \cdots + \frac{\partial L}{\partial w_{Dd}} \cdot \frac{\partial w_{Dd}}{\partial q_{ij}}, \quad i = 1, \cdots, D, \quad j = 1, \cdots, d \tag{7}$$

This expansion contains Dd terms, although some $\partial w_{kl}/\partial q_{ij}$ terms can be zero. This can be seen by examining (5), in which the dependence of the $\mathbf{w}$ vectors on the $\mathbf{q}$ vectors is made explicit. In particular, we can see that $\partial w_{kl}/\partial q_{ij} = 0$ whenever $j > l$. The terms of the summation (7) are those resulting from the elementwise multiplication of the matrices $\partial L/\partial W_1$ and $\partial W_1/\partial q_{ij}$. Thus, we can rewrite (7) in shorthand as

$$\frac{\partial L}{\partial q_{ij}} = \left\langle \frac{\partial L}{\partial W_1}, \frac{\partial W_1}{\partial q_{ij}} \right\rangle_F, \tag{8}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product. Thus, the gradient we must compute to back propagate the loss through the DAS layer is given by

$$\frac{\partial L}{\partial Q} = \begin{bmatrix} \left\langle \frac{\partial L}{\partial W_1}, \frac{\partial W_1}{\partial q_{11}} \right\rangle_F & \cdots & \left\langle \frac{\partial L}{\partial W_1}, \frac{\partial W_1}{\partial q_{1d}} \right\rangle_F \\ \vdots & \ddots & \vdots \\ \left\langle \frac{\partial L}{\partial W_1}, \frac{\partial W_1}{\partial q_{D1}} \right\rangle_F & \cdots & \left\langle \frac{\partial L}{\partial W_1}, \frac{\partial W_1}{\partial q_{Dd}} \right\rangle_F \end{bmatrix}. \tag{9}$$

Note that $\partial L/\partial W_1$ will be available through standard back propagation. However, the Dd matrices $\partial W_1/\partial q_{ij}$ will require us to compute the derivatives of the Gram-Schmidt vectors $\mathbf{w}_i$ with respect to the original, non-orthogonal vectors $\mathbf{q}_k$.

4.1 Derivatives of unnormalized Gram-Schmidt vectors

We will first compute the derivative of $\mathbf{w}_i$, before it is normalized by its length $\|\mathbf{w}_i\|_2$:

$$\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_k} = \frac{\partial \mathbf{q}_i}{\partial \mathbf{q}_k} - \sum_{j=1}^{i-1} \frac{\partial}{\partial \mathbf{q}_k} \left[\left(\frac{\mathbf{w}_j^T \mathbf{q}_i}{\mathbf{w}_j^T \mathbf{w}_j} \right) \mathbf{w}_j \right], \quad i = 1, \cdots, d, \quad k = 1, \cdots, d. \tag{10}$$

A brute-force computation of this derivative using a computer algebra system will show that this quickly becomes a complicated expression with a very large number of terms. However, we can find a simple expression for this derivative by computing it in an iterative fashion:

$i = 1$:

$$\mathbf{w}_1 = \mathbf{q}_1 \Rightarrow \frac{\partial \mathbf{w}_1}{\partial \mathbf{q}_1} = I_D =: D_{11}, \quad \text{and} \quad \frac{\partial \mathbf{w}_1}{\partial \mathbf{q}_i} = 0 \quad \text{for } i > 1. \tag{11}$$

$i = 2$:

$$\mathbf{w}_2 = \mathbf{q}_2 - \left(\frac{\mathbf{w}_1^T\mathbf{q}_2}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1 \tag{12}$$

First we compute the 'shear' derivative $\partial\mathbf{w}_2/\partial\mathbf{q}_1$:

$$\begin{aligned}
\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_1} &= -\frac{\partial}{\partial\mathbf{q}_1}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_2}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right] \\
&= -\frac{\partial}{\partial\mathbf{w}_1}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_2}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right]\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_1} \\
&= \underbrace{-\left[\frac{1}{\mathbf{w}_1^T\mathbf{w}_1}\cdot\mathbf{w}_1\mathbf{q}_2^T - \frac{2\mathbf{w}_1^T\mathbf{q}_2}{(\mathbf{w}_1^T\mathbf{w}_1)^2}\cdot\mathbf{w}_1\mathbf{w}_1^T + \frac{\mathbf{w}_1^T\mathbf{q}_2}{\mathbf{w}_1^T\mathbf{w}_1}\cdot I_D\right]}_{=:D_{21}}\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}1} \\
&= D_{21}\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_1} = D_{21}D_{11}.
\end{aligned} \tag{13}$$

Here, the second equality is just the chain rule, and in the third we used the following matrix calculus identity:

$$D_{ij} := -\frac{\partial}{\partial\mathbf{w}_j}\left[\left(\frac{\mathbf{w}_j^T\mathbf{q}_i}{\mathbf{w}_j^T\mathbf{w}_j}\right)\mathbf{w}_j\right] = -\left[\frac{1}{\mathbf{w}_j^T\mathbf{w}_j}\cdot\mathbf{w}_j\mathbf{q}_i^T - \frac{2\mathbf{w}_j^T\mathbf{q}_i}{(\mathbf{w}_j^T\mathbf{w}_j)^2}\cdot\mathbf{w}_j\mathbf{w}_j^T + \frac{\mathbf{w}_j^T\mathbf{q}_i}{\mathbf{w}_j^T\mathbf{w}_j}\cdot I_D\right], \tag{14}$$

which is derived in Appendix A. Hence, we can compute D_{21}, and the matrix $D_{11} := \partial\mathbf{w}_1/\partial\mathbf{q}_1 = I_D$ was already computed in the previous iteration. The matrix-matrix product of D_{21} and D_{11} yields $\partial\mathbf{w}_2/\partial\mathbf{q}_1$.

We now compute the 'normal' derivative $\partial\mathbf{w}_2/\partial\mathbf{q}_2$:

$$\begin{aligned}
\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_2} &= I_D - \frac{\partial}{\partial\mathbf{q}_2}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_2}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right] \\
&= I_D - \frac{\mathbf{w}_1\mathbf{w}_1^T}{\mathbf{w}_1^T\mathbf{w}_1} \\
&= D_{11} - \frac{\mathbf{w}_1\mathbf{w}_1^T}{\mathbf{w}_1^T\mathbf{w}_1} =: D_{22}
\end{aligned} \tag{15}$$

In the second equality we made use of the identity:

$$\frac{\partial}{\partial\mathbf{q}_i}\left[\left(\frac{\mathbf{w}_j^T\mathbf{q}_i}{\mathbf{w}_j^T\mathbf{w}_j}\right)\mathbf{w}_j\right] = \frac{\mathbf{w}_j\mathbf{w}_j^T}{\mathbf{w}_j^T\mathbf{w}_j}, \tag{16}$$

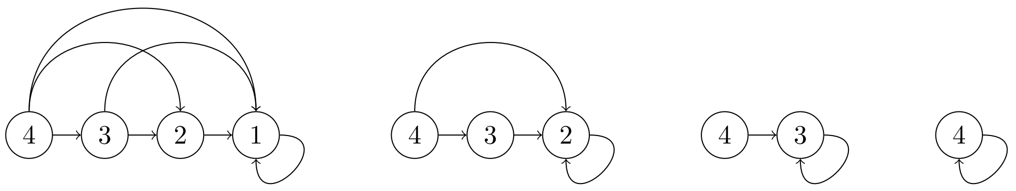
also derived in Appendix A. This holds if $\mathbf{q}_i$ does not depend upon $\mathbf{w}_j$, which is true in (15) since $\mathbf{w}_1 = \mathbf{w}_1(\mathbf{q}_1)$. Also, since $\mathbf{w}_2 = \mathbf{w}_2(\mathbf{q}_1, \mathbf{q}_2)$, we have $\partial\mathbf{w}_2/\partial\mathbf{q}_k = 0$ for $k > 2$. Further note that in (24), we can write $\partial\mathbf{w}_2/\partial\mathbf{q}_2$ as the difference between $D_{11} := \partial\mathbf{w}_1/\partial\mathbf{q}_1$ and $\mathbf{w}_1\mathbf{w}_1^T/\mathbf{w}_1^T\mathbf{w}_1$, and that we have defined this expression for $\partial\mathbf{w}_2/\partial\mathbf{q}_2$ as D_{22}.

$i = 3$:

$$\mathbf{w}_3 = \mathbf{q}_3 - \left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1 - \left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\mathbf{w}_2 \tag{17}$$

We again compute the 'shear' derivatives first:

$$\begin{aligned}
\frac{\partial\mathbf{w}_3}{\partial\mathbf{q}_1} &= -\frac{\partial}{\partial\mathbf{q}_1}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right] - \frac{\partial}{\partial\mathbf{q}_1}\left[\left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\mathbf{w}_2\right] \\
&= \underbrace{-\frac{\partial}{\partial\mathbf{w}_1}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right]}_{=:D_{31}}\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_1} \underbrace{-\frac{\partial}{\partial\mathbf{w}_2}\left[\left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\mathbf{w}_2\right]}_{=:D_{32}}\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_1} \\
&= D_{31}\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_1} + D_{32}\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_1} = D_{31}D_{11} + D_{32}D_{21}D_{11}.
\end{aligned} \tag{18}$$



(a) All pairwise paths from 4 to 1, always ending with $(1,1)$. Corresponds to indices of $\partial\mathbf{w}_4/\partial\mathbf{q}_1$.

(b) All pairwise paths from 4 to 2, always ending with $(2,2)$. Corresponds to indices of $\partial\mathbf{w}_4/\partial\mathbf{q}_2$.

(c) All pairwise paths from 4 to 3, always ending with $(3,3)$. Corresponds to indices of $\partial\mathbf{w}_4/\partial\mathbf{q}_3$.

(d) The only pairwise paths from 4 to 4. Corresponds to index of $\partial\mathbf{w}_4/\partial\mathbf{q}_4$.

Figure 3: The directed graphs that generate the indices of the D_{ij} matrices of (21).

And

$$\begin{aligned}
\frac{\partial\mathbf{w}_3}{\partial\mathbf{q}_2} &= -\frac{\partial}{\partial\mathbf{q}_2}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right] - \frac{\partial}{\partial\mathbf{q}_2}\left[\left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\mathbf{w}_2\right] \\
&= -\frac{\partial}{\partial\mathbf{w}_1}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\mathbf{w}_1\right]\underbrace{\frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_2}}_{0} \underbrace{- \frac{\partial}{\partial\mathbf{w}_2}\left[\left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\mathbf{w}_2\right]}_{:=D_{32}}\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_2} \\
&= D_{32}\frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_2} = D_{32}D_{22}.
\end{aligned} \tag{19}$$

The ‘normal’ derivative is given by:

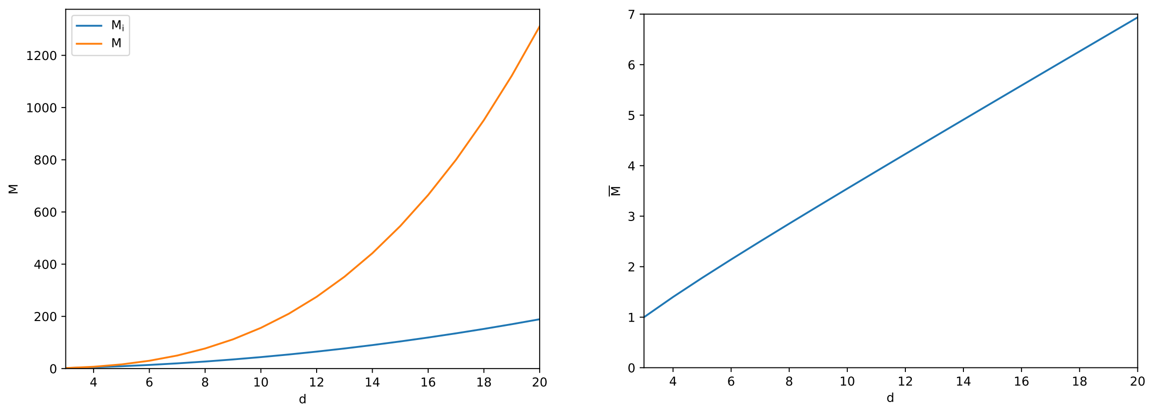
$$\begin{aligned}
\frac{\partial\mathbf{w}_3}{\partial\mathbf{q}_3} &= I_D - \frac{\partial}{\partial\mathbf{q}_3}\left[\left(\frac{\mathbf{w}_1^T\mathbf{q}_3}{\mathbf{w}_1^T\mathbf{w}_1}\right)\right] - \frac{\partial}{\partial\mathbf{q}_3}\left[\left(\frac{\mathbf{w}_2^T\mathbf{q}_3}{\mathbf{w}_2^T\mathbf{w}_2}\right)\right] \\
&= \underbrace{I_D - \frac{\mathbf{w}_1\mathbf{w}_1^T}{\mathbf{w}_1^T\mathbf{w}_1}}_{D_{22}} - \frac{\mathbf{w}_2\mathbf{w}_2^T}{\mathbf{w}_2^T\mathbf{w}_2} = D_{22} - \frac{\mathbf{w}_2\mathbf{w}_2^T}{\mathbf{w}_2^T\mathbf{w}_2} =: D_{33}
\end{aligned} \tag{20}$$

Finally, consider the case for $i = 4$ in shorthand notation only:

$$\begin{aligned}
\frac{\partial\mathbf{w}_4}{\partial\mathbf{q}_1} &= D_{41}D_{11} + D_{42}D_{21}D_{11} + D_{43}D_{31}D_{11} + D_{43}D_{32}D_{21}D_{11} \\
\frac{\partial\mathbf{w}_4}{\partial\mathbf{q}_2} &= D_{42}D_{22} + D_{43}D_{32}D_{22} \\
\frac{\partial\mathbf{w}_4}{\partial\mathbf{q}_3} &= D_{43}D_{33} \\
\frac{\partial\mathbf{w}_4}{\partial\mathbf{q}_3} &= D_{33} - \frac{\mathbf{w}_3\mathbf{w}_3^T}{\mathbf{w}_3^T\mathbf{w}_3} =: D_{44}
\end{aligned} \tag{21}$$

The structure is now apparent. The gradients $\partial\mathbf{w}_i/\partial\mathbf{q}_k$, when completely expanded as in (21), are determined by a series of matrix-matrix multiplications, the indices of which come from all pairwise paths of a directed graph from $i \to k$, ending with $k \to k$. To see this, consider the graphs of Figure 3, and compare this to the indices of the D_{ij} matrices appearing in the expressions of (21).

The graph gives some insight into the structure of each derivative, as it allows us to directly expand all terms that make up each gradient. However, we will not directly use it in practice to compute the gradients. Instead, we will start with the gradient of $\mathbf{w}_1$, then compute the gradients of $\mathbf{w}_2$ etc. This is because at any given $\mathbf{w}_i$, we can reuse the results from the previous iteration, thereby avoiding repeated



(a) The total number of matrix multiplication M_i at a given i, and the total cumulative cost M.

(b) The total number of matrix-matrix multiplications, divided by the number of gradients that are computed.

Figure 4: The number of matrix-matrix multiplications as a function of d.

matrix multiplications. This is clear when we write the 'shear' gradients as

$$\boxed{\frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_k} = \sum_{j=1}^{i-1} D_{ij}\frac{\partial \mathbf{w}_j}{\partial \mathbf{q}_k}, \quad i \neq k, \;\; i > 1, \;\; i < k} \tag{22}$$

At any given $i > 1$, all $\partial \mathbf{w}_j/\partial \mathbf{q}_k$ terms above have either already been computed at the previous iterations, or are zero when $k > j$. If we count the minimum number of matrix-matrix multiplications that are required to compute all shear gradients at a given i, we find that when $k = 1$, we get $i-1$ matrix-matrix multiplications. As an example, consider $\partial \mathbf{w}_4/\partial \mathbf{q}_1 = D_{41}\partial \mathbf{w}_1/\partial \mathbf{q}_1 + D_{42}\partial \mathbf{w}_2/\partial \mathbf{q}_1 + D_{43}\partial \mathbf{w}_3/\partial \mathbf{q}_1$, which requires $i-1 = 3$ matrix multiplications. Technically however, D_{41} is multiplied by $\partial \mathbf{w}_1/\partial \mathbf{q}_1 = I_D$, so the first product we never have to compute. When $k = 2$, we still have $i-1$ terms in (22). However, the first term will include $\partial \mathbf{w}_1/\partial \mathbf{q}_2 = 0$. Likewise, when $k = 3$ the first two terms will be zero. Thus the total number of required matrix multiplications M_i is $M_i = (i-1) + (i-2) + \cdots + 2 + 1 - 1$, where we subtracted 1 at the end because we do not count multiplication by I_D. Finally, the total number of matrix multiplication M to compute all non-zero gradients of $\mathbf{w}_i$, for all $i = 1, \cdots, d$, is therefore given by

$$M = \sum_{i=3}^{d} M_i = \sum_{i=3}^{d}\sum_{j=2}^{i-1} j. \tag{23}$$

We start counting at $i = 3$ because $i = 1$ and $i = 2$ require no matrix multiplications, see (11) and (13). In Figure 4(a) we plot M versus d. which shows that for $d = 20$ we would already need to perform more than 1200 matrix-matrix multiplications. That said, it should be noted that this is the cost of computing *all* gradients $\partial \mathbf{w}_i/\partial \mathbf{q}_k$ which require matrix-matrix multiplication. In the case of $d = 20$, the total number of such gradients is given by $2 + 3 + \cdots + 19 = 189$. Hence, it is not particularly surprising that a large number of multiplications are required. Figure 4(b) shows the total number of multiplications divided by the number of gradients which are computed, which suggests that the average number of matrix-matrix multiplications scales linearly with d. Still, for high d the number of matrix-matrix multiplications M can be significant, especially considering we will have to redo the gradient calculations every stochastic gradient descent step when training the neural network. That said, in practise this might not pose a real problem, as the whole point of creating a DAS surrogate model is keeping d low in the first place.

The preceding pertained to the shear gradients. From (13), (20) and (21), we can see that the normal gradients are computed as:

$$\boxed{D_{ii} := \frac{\partial \mathbf{w}_i}{\partial \mathbf{q}_i} = D_{i-1,i-1} - \frac{\mathbf{w}_{i-1}\mathbf{w}_{i-1}^T}{\mathbf{w}_{i-1}^T\mathbf{w}_{i-1}}}, \tag{24}$$

with $D_{11} := I_D$. Note that no matrix-matrix multiplication is involved in computing these gradients.

Finally, let us note that we verified the correctness of (22) and (24), by comparing our numerical result with the results from a computer algebra system, which used symbolic math to directly compute the derivatives from the definition of the $\mathbf{w}_i$ (Eq. (4)).

4.2 Derivatives of normalized Gram-Schmidt vectors

Equations (22) and (24) give simple iterative expressions for $\partial\mathbf{w}_i/\partial\mathbf{q}_k$. However, the weight vectors of the DAS layers are normalized (see Figure 1), and so we need to compute $\partial(\mathbf{w}_i/\|\mathbf{w}_i\|_2)/\partial\mathbf{q}_k$. As shown in Appendix A, we can just premultiply $\partial\mathbf{w}_i/\partial\mathbf{q}_k$ with a matrix which only depends upon $\mathbf{w}_i$, to obtain the gradient of the corresponding normed vector:

$$\boxed{\frac{\partial}{\partial\mathbf{q}_k}\left(\frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}\right) = \left[\frac{I_D}{\|\mathbf{w}_i\|_2} - \frac{\mathbf{w}_i\mathbf{w}_i^T}{\|\mathbf{w}_i\|_2^3}\right]\frac{\partial\mathbf{w}_i}{\partial\mathbf{q}_k}}. \tag{25}$$

The validity of this expression was also verified with the use of a computer algebra system.

4.3 Assembling the loss gradient

Computing all $\partial(\mathbf{w}_i/\|\mathbf{w}_i\|_2)/\partial\mathbf{q}_k$ gives us all the information we need to assemble the loss gradient (9), provided that standard back propagation has provided $\partial L/\partial W_1$. However, the information is not yet in the right place to compute the Frobenius inner products $\langle\partial L/\partial W_1, \partial W_1/\partial q_{ij}\rangle_F$, because these require $\partial W_1/\partial q_{ij}$ instead of $\partial\mathbf{w}_i/\partial\mathbf{q}_k$. The matrices $\partial\mathbf{w}_i/\partial\mathbf{q}_k$ and $\partial W_1/\partial q_{ij}$ are given by

$$\frac{\partial\mathbf{w}_i}{\partial\mathbf{q}_k} = \begin{bmatrix} \frac{\partial w_{1i}}{\partial q_{1k}} & \frac{\partial w_{1i}}{\partial q_{2k}} & \cdots, & \frac{\partial w_{1i}}{\partial q_{Dk}} \\ \frac{\partial w_{2i}}{\partial q_{1k}} & \frac{\partial w_{2i}}{\partial q_{2k}} & \cdots, & \frac{\partial w_{2i}}{\partial q_{Dk}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial w_{Di}}{\partial q_{1k}} & \frac{\partial w_{Di}}{\partial q_{2k}} & \cdots, & \frac{\partial w_{Di}}{\partial q_{Dk}} \end{bmatrix} \in \mathbb{R}^{D\times D}, \quad \frac{\partial W_1}{\partial q_{ij}} = \begin{bmatrix} \frac{\partial w_{11}}{\partial q_{ij}} & \frac{\partial w_{12}}{\partial q_{ij}} & \cdots, & \frac{\partial w_{1d}}{\partial q_{ij}} \\ \frac{\partial w_{21}}{\partial q_{ij}} & \frac{\partial w_{22}}{\partial q_{ij}} & \cdots, & \frac{\partial w_{2d}}{\partial q_{ij}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial w_{D1}}{\partial q_{ij}} & \frac{\partial w_{D2}}{\partial q_{ij}} & \cdots, & \frac{\partial w_{Dd}}{\partial q_{ij}} \end{bmatrix} \in \mathbb{R}^{D\times d}. \tag{26}$$

To compute the Frobenius inner products $\langle\partial L/\partial W_1, \partial W_1/\partial q_{ij}\rangle_F$, we need to assemble the Dd matrices $\partial W_1/\partial q_{ij}$. From (26), we see that this is done as

$$\frac{\partial W_1}{\partial q_{ij}} = \left[\mathrm{col}\left(i, \frac{\partial\mathbf{w}_1}{\partial\mathbf{q}_j}\right) \; \mathrm{col}\left(i, \frac{\partial\mathbf{w}_2}{\partial\mathbf{q}_j}\right) \; \cdots \; \mathrm{col}\left(i, \frac{\partial\mathbf{w}_d}{\partial\mathbf{q}_j}\right)\right] \tag{27}$$

where $\mathrm{col}(i, A)$ returns the i-th column of matrix A. Hence, we can easily assemble $\partial W_1/\partial q_{ij}$ by selecting the i-th column of the d derivatives $\partial\mathbf{w}_i/\partial\mathbf{q}_j$, $i = 1, \cdots, d$. In fact, if we store all $\partial\mathbf{w}_i/\partial\mathbf{q}_k$ derivatives in a 3D array of shape $[d^2, D, D]$, we can form the $\partial W_1/\partial q_{ij}$ matrices by just taking 2D slices from this array.

5 Initial results

Here we show some initial results we obtained with the DAS method. We start with a simple analytic test function.

5.1 Analytic test function

Here we consider the following analytic test function:

$$f(\mathbf{x}) = \prod_{i=1}^{D}\frac{a_i x_i^2 + 1}{2^D} \tag{28}$$

Here, x_i are random variables which are uniformly distributed between 0 and 1, which we sample by simple Monte Carlo. We can control the active dimension of the function via the coefficients a_i. For our

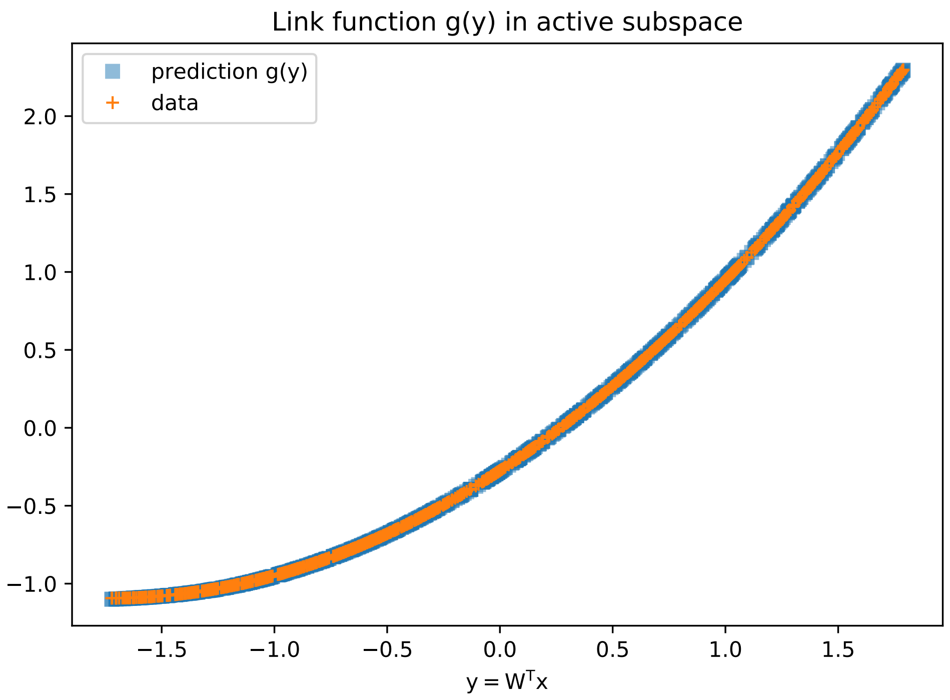


Figure 5: The DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$ versus $\mathbf{y} = W_1^T\mathbf{x}$. Here, we used the analytic test function (28) with $a_1 = 1$ and $a_i = 0$ for $i > 1$, and $d = 1$.

first test we set $D = 10$, $a_1 = 1$ and $a_i = 0$ for $i > 1$. This is the simplest possible test, where only a single input contributes, and where the active dimension is aligned with the coordinate axis of x_1.

We used a DAS surrogate with 2 hidden layers (not counting the DAS layer), with 50 neurons each, tanh activation and the squared loss function. Furthermore, we select $d = 1$. No hyper parameter tuning was performed to find a better surrogate. In Figure 5 we show the prediction $g(\mathbf{y})$ and the data $f(\mathbf{x})$ versus the 1D active subspace $\mathbf{y}$, which show a near perfect overlap.

Furthermore, in this simple example we can also examine the weight matrix W_1, which (after training), is given by:

```
array([[ 9.99999897e-01],
        [-6.75565115e-06],
        [ 2.22074156e-04],
        [ 7.73883499e-05],
        [-1.95599807e-04],
        [ 2.69798310e-04],
        [ 4.28601992e-05],
        [ 5.63268797e-05],
        [-1.78787309e-04],
        [-5.90488430e-05]]),
```

which is a good approximation to the true active subspace $W_1^T = [1, 0, \cdots, 0]$.

Next we select a different a_i, namely $a_i = 1/2^i$, while we keep $d = 1$. Figure 6 show the prediction in the active subspace. The noise in the data indicates that the $d = 1$ assumption is not exactly correct, although $g(\mathbf{y})$ still displays a good fit. The equivalent results for $d = 2$ are shown in Figure 7, which also shows a good fit between data and prediction. Finally, Figure 8 shows the probability density functions of $g(\mathbf{y})$ and $f(\mathbf{x})$, evaluated on a validation data set. This shows that we can sample the DAS surrogates outside the training set (although within the support of the input pdfs of course), and still get reasonable predictions.

After this initial sanity check we move on to a more computationally demanding example.

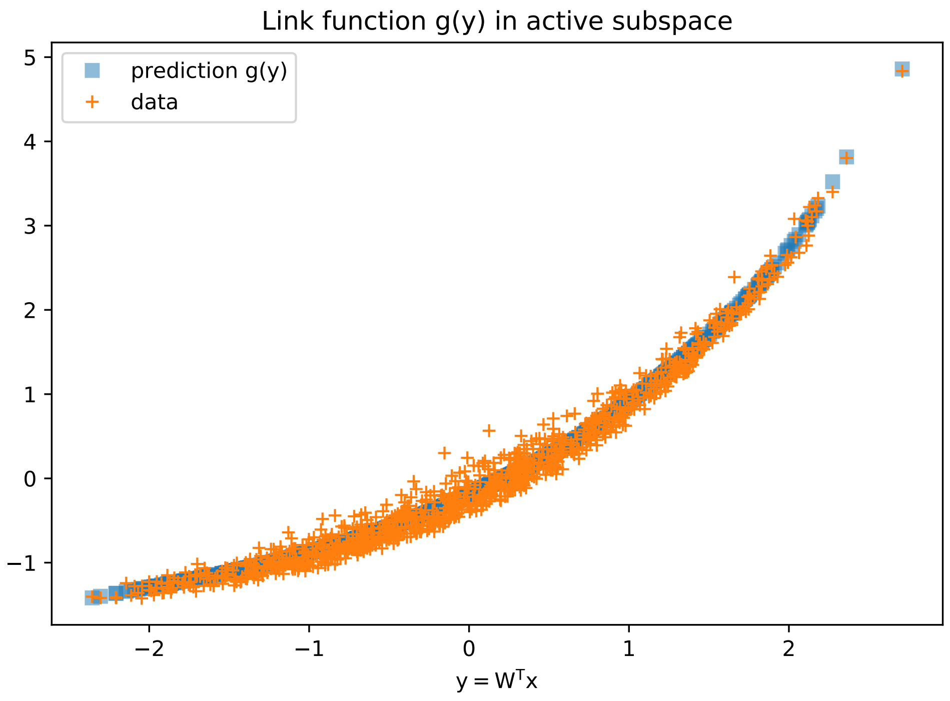
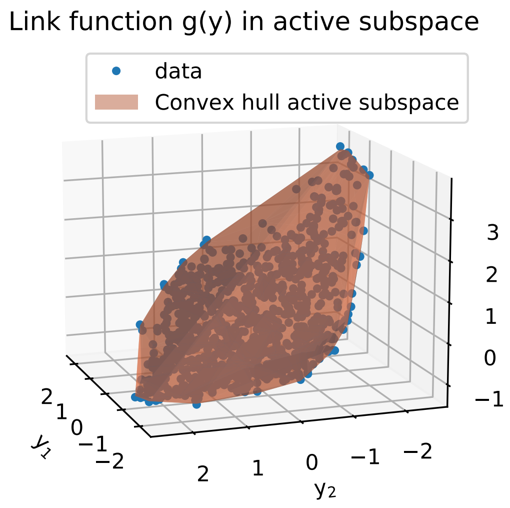


Figure 6: The DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$ versus $\mathbf{y} = W_1^T\mathbf{x}$. Here, we used the analytic test function (28) with $a_i = 1/2^i$ and $d = 1$.



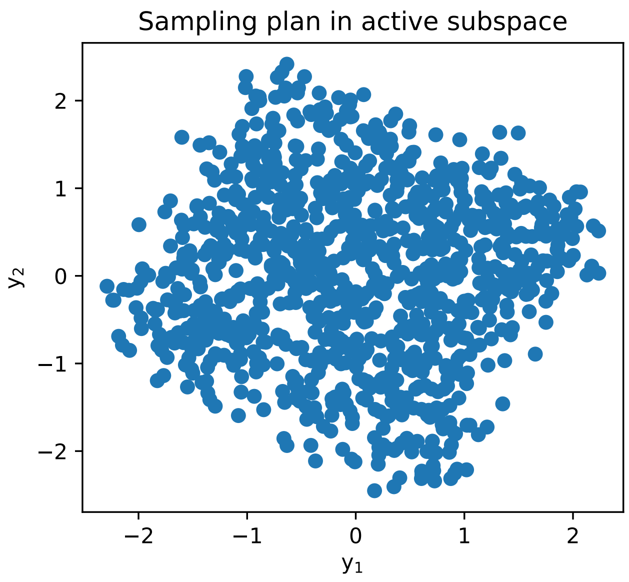


Figure 7: Left: the DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$ versus $\mathbf{y} = W_1^T\mathbf{x}$. Here, we used the analytic test function (28) with $a_i = 1/2^i$ and $d = 2$. Right: the sampling plan in the active subspace.

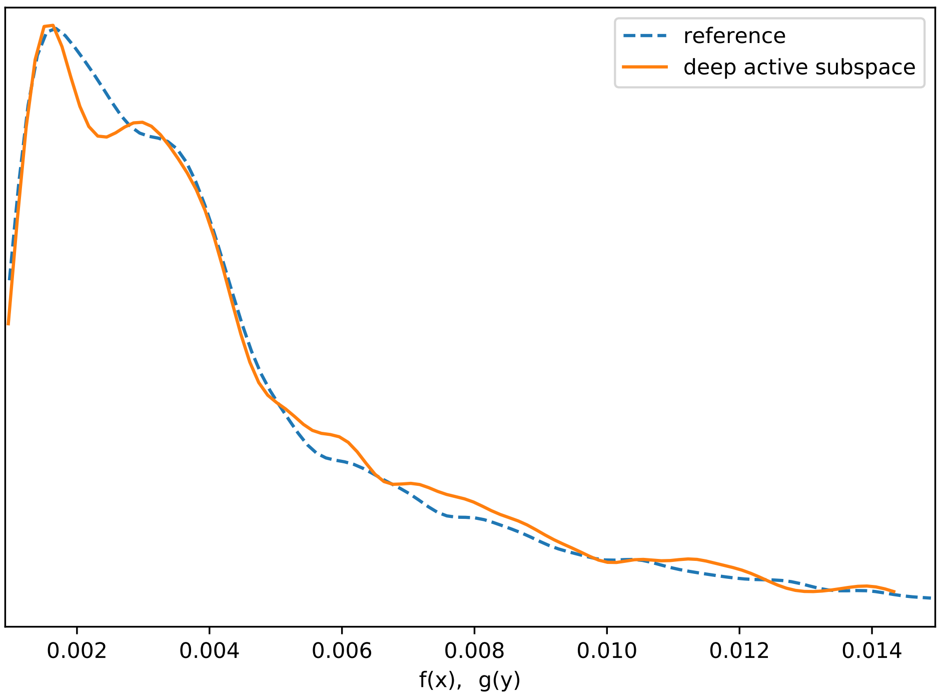


Figure 8: The pdf of the DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$. Here, we used the analytic test function (28) with $a_i = 1/2^i$ and $d = 2$. Evaluated outside the training data set.

5.2 Covidsim

Our next example concerns CovidSim, which is an individual-based simulation code developed by the MRC Centre for Global Infectious Disease Analysis at Imperial College London [4]. It has been used to explore various non-pharmaceutical interventions with the aim of reducing the transmission of the coronavirus.

CovidSim has a large number of input parameters, but we will use the same $D = 19$ input distributions from [3], and sample these 1500 times using Monte Carlo. A single sample of CovidSim takes about 10 minutes, when run in parallel on a single node of the PSNC Eagle supercomputer, which has 28 cores. A rough estimate of the total computational cost is therefore $1500/6 \cdot 28 = 7000$ core hours.

We start again with $d = 1$, and plot the prediction of the total COVID19 death count (after two years), versus the 1D active subspace, see Figure 9. Again, the prediction seems reasonable, although $d = 1$ is almost certainly not the best possible choice. However, an advantage of $d = 1$ is that W_1 has a simple interpretation in this case. Since W_1 is now just a single column vector, the (absolute) magnitude of each of its 19 entries can be considered as a measure for the importance of the associated input. We can compare this to the first-order Sobol indices computed in [3], reproduced in Figure 10. In Table 1, we show the names of the 8 most-important input parameters, rank ordered in descending order according to their first-order Sobol index S_i at the final time index of Figure 10. Furthermore, this table also displays the same rank ordering according to the absolute magnitude of W_1 weights. Note that the order matches exactly between the two methods. This is quite striking, considering that different sensitivity analysis methods, as well as different data sets, were used. More testing is required to see if this generally happens.

As a final show of results, consider Figures 11-12, which display the active subspace plot and pdf comparison, where in this case $d = 3$.

6 Open questions

The following list contains some random questions that I still have pertaining this method.

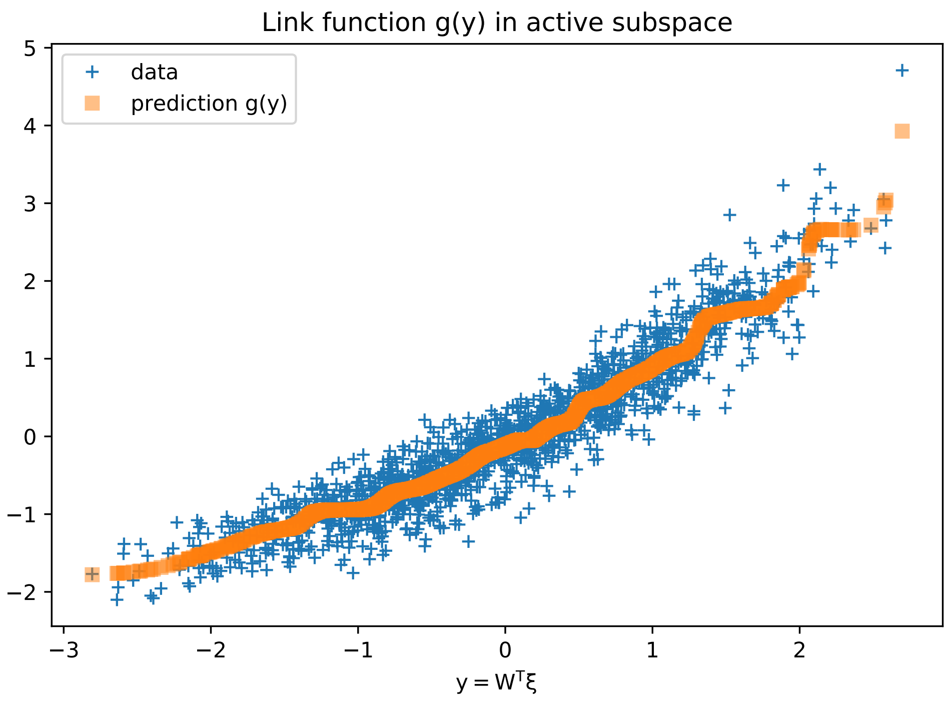


Figure 9: The final death count prediction of CovidSim versus the active subspace ($d = 1$).

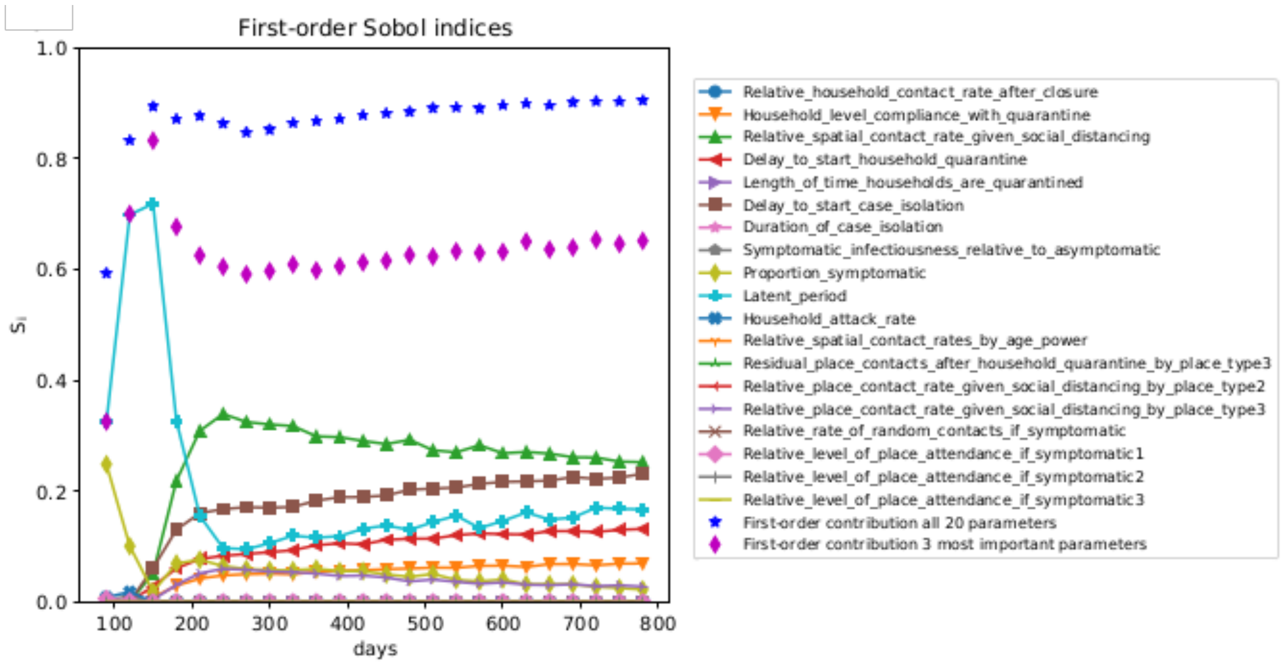


Figure 10: The time-varying first-order Sobol indices of CovidSim from [3].

Sobol index order	**W_1 weight magnitude order**
Relative spatial contact rate given social distancing	Relative spatial contact rate given social distancing
Delay to start case isolation	Delay to start case isolation
Latent period	Latent period
Delay to start household quarantine	Delay to start household quarantine
Household level compliance with quarantine	Household level compliance with quarantine
Relative place contact rate given SD by place type3	Relative place contact rate given SD by place type3
Proportion symptomatic	Proportion symptomatic

Table 1: The 8 most-important CovidSim input parameters, in descending order, computed with two independent sensitivity analysis methods. The data sets used were also independent.

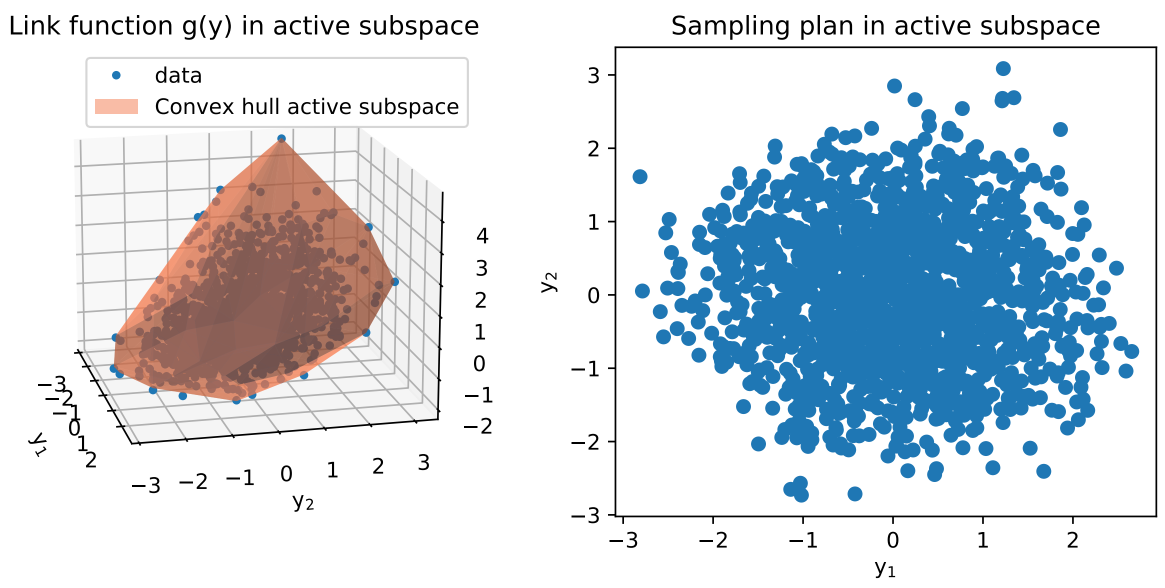


Figure 11: Left: the DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$ versus a two-dimensional slide of the three-dimensional active subspace $\mathbf{y} = W_1^T\mathbf{x}$. Here, we used the final predicted death count of CovidSim as our quantity of interest. Right: the sampling plan projected on the (y_1, y_2) plane of the active subspace.

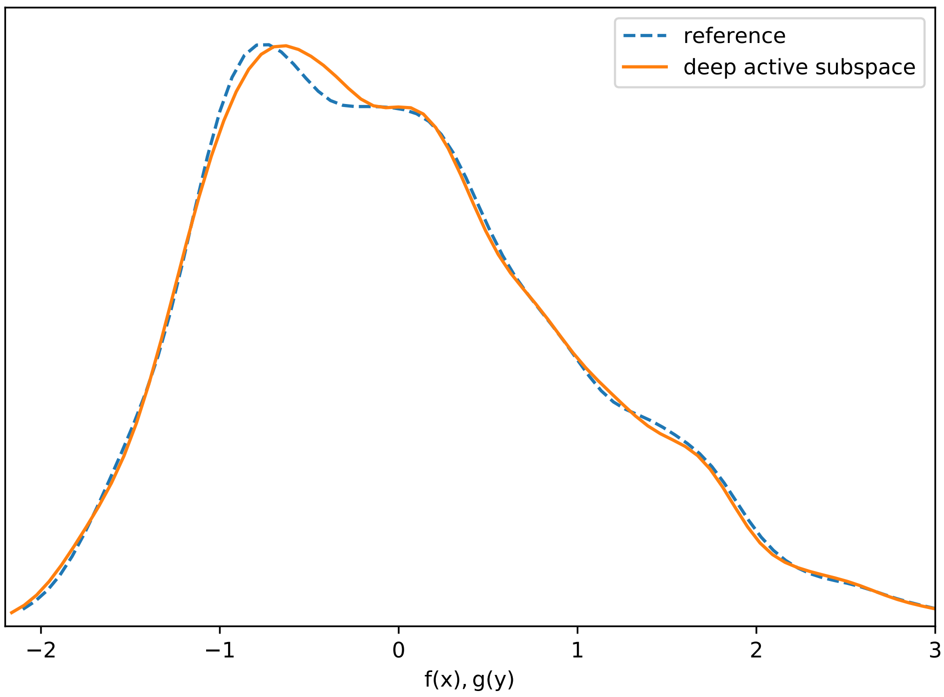


Figure 12: The pdf of the DAS prediction $g(\mathbf{y})$ and data $f(\mathbf{x})$ for the CovidSim model. The horizontal axis displays the cumulative death count, although standardized using the sample mean and standard deviation.

- How does the method perform when the output is not scalar? Classical active subspaces have trouble dealing with non-scalar outputs, because the gradient matrix C become harder to compute and interpret. Here, we could just increase the number of output neurons.
- Does a manual implementation of the gradient $\partial L/\partial Q$ offer the possibility of computational speedup compared to an implementation which uses automatic differentiation? For example, I can examine my expression, and make sure that I compute all $\mathbf{w}_i\mathbf{w}_i^T$ and $\mathbf{w}_i\mathbf{q}_j^T$ matrices only once, despite the fact that they occur multiple times.
- What is the best hyper parameter tuning method? In addition to the number of layers, neurons etc, d is also a hyper parameter which we must tune.
- How does a DAS surrogate, once validated, stack up to for instance a dimension-adaptive Stochastic Collocation surrogate?
- Likewise, what is the performance of a DAS surrogate compared to a vanilla feed forward network? In other words, is the addition of a DAS layer worth the effort?
- Can we can extract a new sensitivity metric from W_1, also for $d > 1$? Or should we just use the DAS surrogate in an existing Sobol MC procedure?

References

[1] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[2] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[3] Wouter Edeling, Arabnejad Hamid, Robert Sinclair, Diana Suleimenova, Krishnakumar Gopalakrishnan, Bartosz Bosak, Derek Groen, Imran Mahmood, Daan Crommelin, and Peter Coveney. Model uncertainty and decision making: Predicting the impact of covid-19 using the covidsim epidemiological code. 2020.

[4] Neil Ferguson, Daniel Laydon, Gemma Nedjati Gilani, Natsuko Imai, Kylie Ainslie, Marc Baguelin, Sangeeta Bhatia, Adhiratha Boonyasiri, ZULMA Cucunuba Perez, Gina Cuomo-Dannenburg, et al. Report 9: Impact of non-pharmaceutical interventions (npis) to reduce covid19 mortality and healthcare demand. 2020.

[5] Xiaoyu Liu and Serge Guillas. Dimension reduction for gaussian process emulation: An application to the influence of bathymetry on tsunami heights. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):787–812, 2017.

[6] KB Petersen and MS Pedersen. The matrix cookbook, version 20121115. *Technical Univ. Denmark, Kongens Lyngby, Denmark, Tech. Rep*, 3274, 2012.

[7] Rohit Tripathy and Ilias Bilionis. Deep active subspaces: A scalable method for high-dimensional uncertainty propagation. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

A Matrix calculus identities

We will derive a number of useful matrix calculus identities here. Let $\mathbf{w}$ and $\mathbf{q}$ be two vectors in $\mathbb{R}^{D\times 1}$. Then

$$\frac{\partial}{\partial \mathbf{w}}\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right) = \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{q}^T - \frac{2\,\mathbf{w}^T\mathbf{q}}{\left(\mathbf{w}^T\mathbf{w}\right)^2}\cdot\mathbf{w}^T \in \mathbb{R}^{1\times D} \tag{29}$$

Proof: This follows directly from the quotient rule:

$$\frac{\partial}{\partial \mathbf{w}}\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right) = \frac{\frac{\partial}{\partial \mathbf{w}}[\mathbf{w}^T\mathbf{q}]\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\mathbf{q}\frac{\partial}{\partial \mathbf{w}}[\mathbf{w}^T\mathbf{w}]}{\left(\mathbf{w}^T\mathbf{w}\right)^2} = \frac{\mathbf{q}^T\cdot\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\mathbf{q}\cdot 2\mathbf{w}^T}{\left(\mathbf{w}^T\mathbf{w}\right)^2}. \tag{30}$$

which yields (29). Going one step further, we have to following identity:

$$\boxed{\frac{\partial}{\partial \mathbf{w}}\left[\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\mathbf{w}\right] = \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{w}\mathbf{q}^T - \frac{2\mathbf{w}^T\mathbf{q}}{(\mathbf{w}^T\mathbf{w})^2}\cdot\mathbf{w}\mathbf{w}^T + \frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\cdot I_D} \in \mathbb{R}^{D\times D} \tag{31}$$

Proof: Apply the product rule:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}}\left[\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\mathbf{w}\right] &= \mathbf{w}\frac{\partial}{\partial \mathbf{w}}\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right) + \left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\cdot I_D \\
&= \mathbf{w}\left[\frac{1}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{q}^T - \frac{2\,\mathbf{w}^T\mathbf{q}}{\left(\mathbf{w}^T\mathbf{w}\right)^2}\cdot\mathbf{w}^T\right] + \left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\cdot I_D \\
&= \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{w}\mathbf{q}^T - \frac{2\mathbf{w}^T\mathbf{q}}{(\mathbf{w}^T\mathbf{w})^2}\cdot\mathbf{w}\mathbf{w}^T + \frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\cdot I_D
\end{aligned} \tag{32}$$

where in the second equality we inserted (29). The final identity we need is given by

$$\boxed{\frac{\partial}{\partial \mathbf{q}}\left[\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\mathbf{w}\right] = \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot\mathbf{w}\mathbf{w}^T} \in \mathbb{R}^{D\times D} \tag{33}$$

Proof: Apply the product rule:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{q}}\left[\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\mathbf{w}\right] &= \mathbf{w}\frac{\partial}{\partial \mathbf{q}}\left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right) + \left(\frac{\mathbf{w}^T\mathbf{q}}{\mathbf{w}^T\mathbf{w}}\right)\underbrace{\frac{\partial \mathbf{w}}{\partial \mathbf{q}}}_{0} \\ &= \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot \mathbf{w}\frac{\partial}{\partial \mathbf{q}}\left(\mathbf{w}^T\mathbf{q}\right) = \frac{1}{\mathbf{w}^T\mathbf{w}}\cdot \mathbf{w}\mathbf{w}^T \qquad (34)\end{aligned}$$

Note that we have taken $1/\mathbf{w}^T\mathbf{w}$ out of the differentiation operator, and that we therefore assume that $\mathbf{w}$ does not depend upon $\mathbf{q}$. This is perhaps confusing, as we do not make this assumption elsewhere. However, in the context of deriving the derivatives of the Gram-Schmidt vectors, this assumption always holds when we apply (33), and it will be made clear by the addition of subscripts to $\mathbf{w}$ and $\mathbf{q}$.

Finally, a standard formula (see e.g. [6]), for the gradient of a normed vector is:

$$\frac{\partial}{\partial \mathbf{w}}\left(\frac{\mathbf{w}}{\|\mathbf{w}\|_2}\right) = \frac{I_D}{\|\mathbf{w}\|_2} - \frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^3}. \qquad (35)$$

In our case, a useful related identity is:

$$\boxed{\frac{\partial}{\partial \mathbf{q}}\left(\frac{\mathbf{w}}{\|\mathbf{w}\|_2}\right) = \left[\frac{I_D}{\|\mathbf{w}\|_2} - \frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|_2^3}\right]\frac{\partial \mathbf{w}}{\partial \mathbf{q}}}, \qquad (36)$$

which follows directly from the chain rule.