A basis-invariant version of attribution

Jacob Hilton

March 14, 2024

Let X be a random vector taking values in $\mathbb{R}^n$, and let $f : \mathbb{R}^n \to \mathbb{R}$ be any differentiable function. For a central example, X could be the activations of a neural network at some layer, and f could be some scalar function of the network's output, such as a particular component of the output, or the loss, or something else.

We would like to determine which directions of variance in X are "important" to the function f. Intuitively, the importance of a direction depends on both how much variance there is in that direction, and on how much each unit of variance affects the output of f. This idea is reflected in wide variety of attribution methods, such as integrated gradients [Sundararajan et al., 2017]. However, such methods generally rely on a choice of basis, and so a further dimensionality reduction step such as non-negative matrix factorization is typically applied to the results of the attribution method to obtain the important directions [Olah et al., 2018, Hilton et al., 2020].

Here we present an alternative method of finding the directions of variance in X that are important to f that does not rely on a choice of basis. The method has several attractive theoretical properties: it optimizes an intuitive objective, it is invariant to linear reparameterizations, and it generalizes principal component analysis (PCA) in a natural way. Furthermore, it is straightforward to apply in practice: it has the same time complexity as PCA (up to a small constant factor), it is numerically stable as long as there are actually important directions to be found, and it can be implemented in about 10 lines of code.

1 Derivation of the method

The essential idea is to find an $n \times n$ projection matrix P of rank at most k, i.e. satisfying $P^2 = P$ with at most a k-dimensional image, to minimize

$$\mathbb{E}\left[\left(f\left(X\right) - f\left(\boldsymbol{\mu} + P\left(X - \boldsymbol{\mu}\right)\right)\right)^2\right],$$

where $\boldsymbol{\mu}$ is the mean of X. The intuition behind this objective is that after projecting onto the important directions of variance, our estimate of f should be as accurate as possible. Despite the apparent generality of this objective, it turns out that it can be approximately solved in a straightforward way using singular value decomposition.

To see this, take a first-order Taylor expansion of f about X to approximate the objective

as

$$\begin{aligned}
&\mathbb{E}\left[\left(\nabla f\left(X\right)^{\mathsf{T}}\left(I-P\right)\left(X-\boldsymbol{\mu}\right)\right)^{2}\right]\\
&=\mathbb{E}\left[\operatorname{tr}\left(\left(I-P\right)\left(X-\boldsymbol{\mu}\right)\left(X-\boldsymbol{\mu}\right)^{\mathsf{T}}\left(I-P^{\mathsf{T}}\right)\nabla f\left(X\right)\nabla f\left(X\right)^{\mathsf{T}}\right)\right]\\
&\approx\operatorname{tr}\left(\left(I-P\right)\boldsymbol{\Sigma}\left(I-P^{\mathsf{T}}\right)\boldsymbol{\Pi}\right),
\end{aligned}$$

where

$$\boldsymbol{\Sigma}:=\mathbb{E}\left[\left(X-\boldsymbol{\mu}\right)\left(X-\boldsymbol{\mu}\right)^{\mathsf{T}}\right]\qquad\text{and}\qquad\boldsymbol{\Pi}:=\mathbb{E}\left[\nabla f\left(X\right)\nabla f\left(X\right)^{\mathsf{T}}\right],$$

and we have approximated X and $\nabla f\left(X\right)$ as independent in the last step. Note that $\boldsymbol{\Sigma}$ is the covariance matrix of X and $\boldsymbol{\Pi}$ is the "uncentered covariance matrix" of $\nabla f\left(X\right)$.

Since $\boldsymbol{\Sigma}$ and $\boldsymbol{\Pi}$ are positive semi-definite, we may write them as $\boldsymbol{\Sigma}=AA^{\mathsf{T}}$ and $\boldsymbol{\Pi}=GG^{\mathsf{T}}$ (these letters are intended to be reminiscent of "activation" and "gradient"). The approximate objective can then be written as

$$\left\|G^{\mathsf{T}}A-G^{\mathsf{T}}PA\right\|_{F}^{2},$$

where $\left\|.\right\|_{F}$ denotes the Frobenius norm. In other words, we are being asked to approximate $G^{\mathsf{T}}A$ by the rank-k matrix $G^{\mathsf{T}}PA$. But by the Eckart–Young–Mirsky theorem, the best rank-k approximation to $G^{\mathsf{T}}A$ of any form is obtained by taking the singular value decomposition of $G^{\mathsf{T}}A$,

$$G^{\mathsf{T}}A=USV^{\mathsf{T}},$$

zeroing out all but the top k singular values to obtain S_k, and using the approximation US_kV^{T}. Hence we can take

$$P=AVS_{k}^{+}U^{\mathsf{T}}G^{\mathsf{T}},$$

where $^{+}$ denotes the Moore–Penrose pseudoinverse (i.e., we take the reciprocal of all remaining non-zero singular values), and it is straightforward to check that this satisfies $G^{\mathsf{T}}PA=US_kV^{\mathsf{T}}$ and that $P^2=P$, i.e., P is indeed a projection.

Our choice of P is not the unique solution to this optimization problem. However, if there are no ties for the rth largest singular value, then the matrix $\boldsymbol{\Pi}P\boldsymbol{\Sigma}$ is unique. Hence the image of P is unique if $\boldsymbol{\Pi}$ is invertible and the kernel of P is unique if $\boldsymbol{\Sigma}$ is invertible. Furthermore, if there are no ties for the rth largest singular value, then our choice of P is the unique solution that is continuous as a function of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Pi}$ in a local neighborhood.[1]

2 Practical algorithm

The above derivation suggests the following algorithm to find the $n\times n$ projection matrix P:

1. Collect p independent samples $X^{(1)},\ldots,X^{(p)}$ and compute $\nabla f\left(X^{(1)}\right),\ldots,\nabla f\left(X^{(p)}\right)$.

[1] To see this, note that if there are no ties for the rth largest singular value, then $\boldsymbol{\Pi}P\boldsymbol{\Sigma}$ is unique, and the selection of the largest r singular values is continuous. The eigenvalue and singular value decompositions can be made continuous in a local neighborhood, and the rest of the algorithm is continuous, and so the entire algorithm can be made continuous in a local neighborhood. Hence our choice of P can be obtained as the limit of the algorithm applied to invertible matrices for $\boldsymbol{\Sigma}$ and $\boldsymbol{\Pi}$, for which P is unique.

2. Compute empirical approximations to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\Pi}$:

$$\hat{\boldsymbol{\mu}} := \frac{1}{p}\sum_{i=1}^{p} X^{(i)}$$
$$\widehat{\boldsymbol{\Sigma}} := \frac{1}{p}\sum_{i=1}^{p} \left(X^{(i)} - \hat{\boldsymbol{\mu}}\right)\left(X^{(i)} - \hat{\boldsymbol{\mu}}\right)^{\mathsf{T}}$$
$$\widehat{\boldsymbol{\Pi}} := \frac{1}{p}\sum_{i=1}^{p} \nabla f\left(X^{(i)}\right)\nabla f\left(X^{(i)}\right)^{\mathsf{T}}$$

3. Use eigenvalue decomposition to write $\widehat{\boldsymbol{\Sigma}} = AA^{\mathsf{T}}$ and $\widehat{\boldsymbol{\Pi}} = GG^{\mathsf{T}}$.

4. Use singular value decomposition to write $G^{\mathsf{T}}A = USV^{\mathsf{T}}$ with U and V orthogonal and S diagonal.

5. Select the largest $r = \min(k, \operatorname{rank}(S))$ diagonal entries of S and take the corresponding columns of U and V to obtain U_r, S_r and V_r.

6. Return $P_A = AV_rS_r^{-\frac{1}{2}}$ and $P_G = GU_rS_r^{-\frac{1}{2}}$ representing the rank-r projection matrix $P = P_AP_G^{\mathsf{T}}$ in factored form.

The "important directions" we wanted are then given by the columns of P_A, which form a basis for the image of P, or alternatively by the columns of P_G, which form a basis for the orthogonal complement of the kernel of P. The two are related by $P_A \propto \widehat{\boldsymbol{\Sigma}}P_G$ and $P_G \propto \widehat{\boldsymbol{\Pi}}P_A$, where $\propto$ indicates that the columns are multiples of one another. Which of the two it makes sense use depends on whether we are interested in the domain or the range of the projection.

The time complexity of the algorithm can be improved slightly by combining steps 2 and 3. Instead of constructing $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\boldsymbol{\Pi}}$, we may apply the reduced singular value decomposition directly to the matrices of centered activations and gradients, which saves time if $p \leq n$. Alternatively, we could take A and G to be the centered activations and gradients directly when $p \leq n$. With either of these optimizations, the time complexity of the algorithm is $O(np\min(n,p))$, ignoring the cost of computing the activation and gradient samples.

Here is an illustrative PyTorch implementation that works for $k \leq \operatorname{rank}(S)$:

```
def get_projection(acts, grads, k):
    assert acts.shape == grads.shape
    acts = acts - acts.mean(-2, keepdim=True)
    A_vecs, A_vals, _ = torch.linalg.svd(acts.mT, full_matrices=False)
    G_vecs, G_vals, _ = torch.linalg.svd(grads.mT, full_matrices=False)
    A = A_vecs * A_vals.unsqueeze(-2)
    G = G_vecs * G_vals.unsqueeze(-2)
    U, S, Vh = torch.linalg.svd(G.mT @ A)
    U = U[..., :k]
    S = S[..., :k].unsqueeze(-2)
    V = Vh.mT[..., :k]
    return (A @ (V * S ** (-0.5)), G @ (U * S ** (-0.5)))
```

The only part of the algorithm that can be numerically unstable is taking the largest singular values to the power of $-\frac{1}{2}$. Typically this is not a problem because k is small compared to $\operatorname{rank}(S)$, but it is still a good idea to incorporate error handling for this case, perhaps using a cutoff relative to the largest singular value.

3 Theoretical properties

3.1 Invariance properties

In addition to not relying on a choice of basis, our method is invariant to linear reparameterizations. In other words, if T is any invertible $n \times n$ matrix and we define $X_T = TX$ and $f_T(\mathbf{x}) = f\left(T^{-1}\mathbf{x}\right)$, then applying the method to X_T and f_T gives the projection TPT^{-1}, which is the same projection in the new basis.

To get some more intuition about why these invariance properties are desirable, it is instructive to contrast our algorithm with the following more conventional approach: (a) collect activation and gradient samples; (b) take the pointwise product using a particular choice of basis to obtain attributions; (c) apply dimensionality reduction to the attributions. In this approach, suppose there are two dimensions with equally large activations, but equal and opposite gradients. Then the two dimensions will have large but opposite attributions, and so the dimensionality reduction method may think that moving in opposite directions along these dimensions is an important direction. However, in reality, moving in this combined direction actually has no effect at all on the output of f, so this is misleading. Moreover, we could have noticed this by taking a linear reparameterization that chose the combined direction as a basis vector, since the attribution for that basis vector would have been zero.

Essentially, our algorithm avoids this pitfall by taking dot products between activations and gradients, which are invariant to linear reparameterizations, instead of pointwise products, which are not. However, naively applying this to a list of samples just gives a list of dot products, from which it is unclear how to obtain the important directions. Our algorithm solves this by effectively taking the dot product between *every* activation and *every* gradient, and applying dimensionality reduction to the result. This sounds infeasible because of the quadratic dependence on the number of samples, but we can actually do this in linear time by factorizing the empirical covariance matrices of the activations and gradients.

3.2 Relationship to PCA

The objective of principal component analysis (PCA) is to find an $n \times n$ projection matrix P of rank at most k to minimize

$$\mathbb{E}\left[\|(X - \boldsymbol{\mu}) - P(X - \boldsymbol{\mu})\|^2\right] = \operatorname{tr}\left((I - P)\,\boldsymbol{\Sigma}\left(I - P^{\mathsf{T}}\right)\right),$$

which can be thought of either as the expected reconstruction error (left expression), or as the total variance unexplained (right expression). Thus PCA can be thought of as the special case of our algorithm in which $\boldsymbol{\Pi} = I$.

In PCA, the optimal projection P satisfies $P = P^{\mathsf{T}}$, i.e., P is an orthogonal projection, meaning that the image of P and the orthogonal complement of the kernel of P are equal. This is why PCA only produces one set of directions, whereas our algorithm produces two sets of directions, one for the image and one for the orthogonal complement of the kernel.

Although this may be counterintuitive, it is necessary in order to produce a projection that is invariant to linear reparameterizations. Another way to think about this is that requiring P to be orthogonal would effectively be privileging the standard inner product on $\mathbb{R}^n$. Instead, the P chosen by our method is orthogonal according to an inner product defined by $\mathbf{\Pi}$, in the sense that that $\mathbf{\Pi}^{\frac{1}{2}} P \mathbf{\Pi}^{-\frac{1}{2}}$ is an orthogonal projection if $\mathbf{\Pi}$ is invertible. Dually, $\mathbf{\Sigma}^{-\frac{1}{2}} P \mathbf{\Sigma}^{\frac{1}{2}}$ is an orthogonal projection if $\mathbf{\Sigma}$ is invertible.

This property is helpful for understanding how our method behaves when $\mathbf{\Sigma}$ or $\mathbf{\Pi}$ is not invertible, meaning that the optimization problem does not have a unique solution. In this case our method chooses a solution such that $P\mathbf{\Sigma}$ and $\mathbf{\Pi}P$ are symmetric matrices. For example, consider the linear case $f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$, which implies that $\mathbf{\Pi} = \mathbf{w}\mathbf{w}^\mathsf{T}$. In this case any rank-1 projection of the form $P = \frac{\mathbf{v}\mathbf{w}^\mathsf{T}}{\mathbf{w}^\mathsf{T}\mathbf{v}}$ satisfies $f(\boldsymbol{\mu} + P(X - \boldsymbol{\mu})) = f(X)$ and therefore optimizes our original objective perfectly. Our method chooses the projection $P = \frac{\mathbf{\Sigma}\mathbf{w}\mathbf{w}^\mathsf{T}}{\mathbf{w}^\mathsf{T}\mathbf{\Sigma}\mathbf{w}}$, which is the unique solution for which $P\mathbf{\Sigma}$ is a symmetric matrix, implying that $\mathbf{\Sigma}^{-\frac{1}{2}} P \mathbf{\Sigma}^{\frac{1}{2}}$ is an orthogonal projection if $\mathbf{\Sigma}$ is invertible. In other words, our method chooses an orthogonal projection when $\mathbf{\Sigma} = I$, and otherwise chooses a solution so as to remain invariant to linear reparameterizations.

There are also other ways of thinking about our method in terms of orthogonal projections. Firstly, our method can be thought of as applying an orthogonal projection on the "inside" of $\mathbf{\Sigma} = AA^\mathsf{T}$. Specifically, even though P is not necessarily orthogonal, $P\mathbf{\Sigma}P^\mathsf{T} = AQA^\mathsf{T}$ for the orthogonal projection $Q := V_r V_r^\mathsf{T}$. Secondly, $P\mathbf{\Sigma}P^\mathsf{T}$ can be thought of as applying an orthogonal projection to $\mathbf{\Sigma}$ in "precision space". Specifically, $P\mathbf{\Sigma}P^\mathsf{T} = \left(Q'\mathbf{\Sigma}^+Q'\right)^+$, where $^+$ denotes the Moore–Penrose pseudoinverse and $Q' := P_A \left(P_A^\mathsf{T} P_A\right)^{-1} P_A^\mathsf{T}$ is the orthogonal projection onto the image of P.

4 Extensions

4.1 Fast approximate version

The time complexity of the algorithm can be improved to $O(npk)$ by using an approximate algorithm for singular value decomposition that only computes the top k singular values and vectors, also known as *reduced SVD*. For this to be possible, we first need to avoid constructing and decomposing $\widehat{\mathbf{\Sigma}}$ and $\widehat{\mathbf{\Pi}}$, or even applying singular value decomposition directly to the matrices of centered activations and gradients. To achieve this, we can skip steps 2 and 3 of the algorithm and instead and take A and G to be the centered activations and gradients and directly. More precisely, we can compute $\hat{\boldsymbol{\mu}} := \frac{1}{p}\sum_{i=1}^{p} X^{(i)}$ and take the ith column of A to be $\frac{1}{\sqrt{p}}\left(X^{(i)} - \boldsymbol{\mu}\right)$ and the ith column of G to be $\frac{1}{\sqrt{p}}\nabla f\left(X^{(i)}\right)$ (in fact, the constants $\frac{1}{\sqrt{p}}$ have no effect and can be dropped).

Naively, this actually makes the time complexity of the algorithm worse if $p \geq n$, because $G^\mathsf{T}A$ is now a large matrix that takes $O\left(np^2\right)$ time to compute. However, it turns out that there are approximate algorithms for reduced SVD, such as subspace iteration [Halko et al., 2011, Algorithm 4.4], that only make use of the matrix by multiplying it or its transpose by an $n \times O(k)$ matrix. Hence we can apply such an algorithm to $G^\mathsf{T}A$ in factored form in $O(npk)$ time by first multiplying the matrix by A and then by G^T, or by G and then by A^T for the transpose, without ever actually needing to compute $G^\mathsf{T}A$.

4.2 Cross-covariance version

Our method can also be thought of as finding an approximation of rank at most k to the covariance matrix $\boldsymbol{\Sigma}$. Specifically, if we take $\boldsymbol{\Sigma}^{(k)} := P\boldsymbol{\Sigma}P^{\mathsf{T}}$, then $G^{\mathsf{T}}\boldsymbol{\Sigma}^{(k)}G$ is the best approximation of rank at most k to $G^{\mathsf{T}}\boldsymbol{\Sigma}G$.

This motivates a corresponding version for cross-covariance matrices. Given two random variables X and Y, let

$$\boldsymbol{\Sigma}_{XY} := \mathbb{E}\left[(X - \boldsymbol{\mu}_X)(Y - \boldsymbol{\mu}_Y)^{\mathsf{T}}\right] \qquad \text{and} \qquad \boldsymbol{\Pi}_{XY} := \mathbb{E}\left[\nabla f(X)\nabla f(Y)^{\mathsf{T}}\right],$$

where $\boldsymbol{\mu}_X$ is the mean of X and $\boldsymbol{\mu}_Y$ is the mean of Y. Then, writing $\boldsymbol{\Pi}_{XY} = G_X G_Y^{\mathsf{T}}$, we could ask for an approximation $\boldsymbol{\Sigma}_{XY}^{(k)}$ to $\boldsymbol{\Sigma}_{XY}$ such that $G_X^{\mathsf{T}}\boldsymbol{\Sigma}_{XY}^{(k)}G_Y$ is the best approximation of rank at most k to $G_X^{\mathsf{T}}\boldsymbol{\Sigma}_{XY}G_Y$. Such an approximation is given by

$$\boldsymbol{\Sigma}_{XY}^{(k)} := \boldsymbol{\Sigma}_{XY}G_Y U_Y \left(S_k^2\right)^+ U_X^{\mathsf{T}} G_X^{\mathsf{T}}\boldsymbol{\Sigma}_{XY},$$

where $U_X S^2 U_Y^{\mathsf{T}}$ is the singular value decomposition of $G_X^{\mathsf{T}}\boldsymbol{\Sigma}_{XY}^{(k)}G_Y$, and $\left(S_k^2\right)^+$ zeros out all but the top k singular values of S^2 and takes the reciprocal of all remaining non-zero singular values. Note, however, that unlike in the original setup, this approximation depends on the particular choice of G_X and G_Y. A natural choice is to use a factorization of $\boldsymbol{\Pi}_{XY}$ based on its singular value decomposition.

4.3 Hessian-based version

If $\nabla f(X)$ is small, perhaps because X is close to a local minimum, then we may wish to approximate $\nabla f(X) \approx 0$ and take a second-order Taylor expansion of f about X instead. This gives us the objective

$$\mathbb{E}\left[\left(\tfrac{1}{2}(X - \boldsymbol{\mu})^{\mathsf{T}}\left(I - P^{\mathsf{T}}\right)H_f(X)(I - P)(X - \boldsymbol{\mu})\right)^2\right],$$

where H_f is the Hessian (the matrix of second partial derivatives) of f. If we again approximate X and $H_f(X)$ as independent, and additionally approximate this mean square by a squared mean, then we obtain the approximate objective

$$\operatorname{tr}\left((I - P)\boldsymbol{\Sigma}\left(I - P^{\mathsf{T}}\right)\mathbb{E}\left[\tfrac{1}{2}H_f(X)\right]\right)^2.$$

If the expected Hessian is positive semi-definite, then we can ignore the square, and we get exactly the same objective as before, except that $\boldsymbol{\Pi}$ has been replaced by half the expected Hessian. If the expected Hessian is negative semi-definite, then we can negate it to reduce to the positive semi-definite case.

This gives us an alternative method if we are able to obtain a positive semi-definite estimate of the expected Hessian or its negation. One simple way to do this is to take the absolute value of its eigenvalues, which has the effect of changing the minimization objective from the squared difference of two squared Frobenius norms to the sum of the squared norms. Note that the two methods are closely related, since the expected outer product of the gradient can be thought of as an approximation to (a multiple of) the expected Hessian, as in the derivation of the Gauss–Newton algorithm or in properties of the Fisher information matrix.

4.4 Vector-valued version

It is easy to generalize our method to the case where $f : \mathbb{R}^n \to \mathbb{R}^m$ is vector-valued. We take the minimization objective to be

$$\mathbb{E}\left[\left\|f\left(X\right) - f\left(\boldsymbol{\mu} + P\left(X - \boldsymbol{\mu}\right)\right)\right\|^2\right],$$

which is just the sum of the minimization objectives for each scalar-valued component of f. The method proceeds in exactly the same way, except that now we have

$$\boldsymbol{\Pi} = \sum_{j=1}^{m} \mathbb{E}\left[\nabla f_j\left(X\right) \nabla f_j\left(X\right)^{\mathsf{T}}\right],$$

or, more succinctly, $\boldsymbol{\Pi} = \mathbb{E}\left[J_f\left(X\right)^{\mathsf{T}} J_f\left(X\right)\right]$, where J_f is the Jacobian of f.

References

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

J. Hilton, N. Cammarata, S. Carter, G. Goh, and C. Olah. Understanding RL vision. *Distill*, 2020. doi: 10.23915/distill.00029. `https://distill.pub/2020/understanding-rl-vision`.

C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. `https://distill.pub/2018/building-blocks`.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.