Axes that matter: PCA with a difference

Brian Huge and Antoine Savine extend differential machine learning and introduce a new breed of supervised principal component analysis to reduce the dimensionality of derivatives problems, with applications including the specification and calibration of pricing models, identification of regression features in least-squares Monte Carlo and preprocessing simulated datasets for (differential) machine learning

Differential machine learning

Giles and Glasserman's 'smoking adjoints' paper introduced algorithmic adjoint differentiation (AAD) to the financial industry and applied it to the computation of pathwise differentials in the context of Monte Carlo (MC) simulations.

Fast risk reports are computed by averaging pathwise differentials across scenarios. We argued in Huge & Savine (2020) that pathwise differentials have a bigger story to tell. We demonstrated that supervised machine learning (ML) models such as artificial neural networks (ANNs) learn pricing and risk much more effectively from pathwise differentials. Differential ML more generally encompasses applications of AAD pathwise differentials in all kinds of ML. In this article, we address the problem of dimension reduction and develop a new breed of differential principal component analysis (PCA) to effectively encode the market state into low-dimensional representations. In contrast to the customary average of risk reports, differential PCA leverages the covariance of pathwise differentials.

Dimension reduction

Consider the market state $X \in \mathbb{R}^n$ on some exposure date T. This means that the n entries of X are T-measurable random variables, which fully describe the state of the market on date T. Dimension reduction consists of an encoder function g from $\mathbb{R}^n$ to $\mathbb{R}^p$ with $p \leqslant n$ such that the encoding $L = g(X)$ provides a lower-dimensional representation of the state X.

Dimension reduction underlies the design and calibration of Derivatives models, albeit often implicitly. Derivatives models are essentially a probabilistic description of the evolution of the market state X between the exposure date T and some maturity date T^*, generally written under a risk-neutral measure with parameters calibrated to implied volatility surfaces. We call universal models probabilistic descriptions of the entire state X. They generally require a heavy development effort. Their implementation is often slow, and their parameters are typically roughly calibrated to market data by best fit. Universal models are useful for complex exotics and hybrids, aggregated netting sets or trading books for regulatory computations such as value adjustments (XVA) or counterparty credit risk (CCR), or the simulation of training datasets for learning pricing and risk functions, as discussed in Huge & Savine (2020).

In many standard situations, more effective lightweight models are implemented by focusing on a small number of features relevant to a given application. For example, a basket option depends only on the underlying basket under a Gaussian assumption, irrespective of its constituents. Similarly, a European swaption depends only on its underlying swap (and its volatility), irrespective of the rest of the yield curve. Industry standard swaption models such as Black, Heston or stochastic alpha beta rho (SABR) effectively model the underlying swap rate with increasingly complex dynamics, in contrast to term-structure models of the entire yield curve, suitable for more exotic instruments.

Identifying the correct features is rather trivial with simple products and restrictive assumptions, but it is much harder in more complicated situations. And the stakes are considerable: modelling irrelevant features is merely wasteful, but ignoring the relevant features may lead to incorrect prices, ineffective risk management, and wrong conclusions. For example, it was customary in the late 1990s to calibrate Bermudan option models to only co-terminal European swaptions. A more careful analysis in Andersen & Andreasen (2001) demonstrated the strong dependency of Bermudan options on short rates, and showed that ignoring the behaviour of short rates implied by caplet volatility led to incorrect prices. Claims were made in the literature of a positive dependency of Bermudan option prices on interest-rate correlation, whereas the contrary is generally true with correctly calibrated models. Bermudan options are fundamentally (at least) two-factor products (ignoring stochastic volatility), so models must correctly represent the joint dynamics of co-terminal and short rates, and, in particular, simultaneously calibrate to co-terminal swaptions and caplets.

Feature selection, or (equivalently) dimension reduction, is generally performed manually, often implicitly, and prone to error, with considerable consequences for pricing and risk. The aim of this article is to show how to perform dimension reduction in an explicit, principled, automatic and reliable manner.

Least-squares Monte Carlo

Another corner in which dimension reduction is necessary is Longstaff and Schwartz's least-squares Monte Carlo (LSM), initially designed for Bermudan options in the context of high-dimensional Libor market models (LMMs) implemented with MC. Linear regression on basis functions of the high-dimensional state X is intractable. Many recent articles suggest replacing regression with ML models robust at high dimensions, such as ANNs. But practitioners are often reluctant to forfeit the lightweight development, analytic efficiency (at least in a low dimension) and reliability of linear regression. The alternative is dimension reduction.

Practical LSM implementations generally get around the curse of dimensionality by performing regressions on a low number of relevant features $L_i = g_i(X)$, $i \in [1, p]$, of the state $X \in \mathbb{R}^n$. The resulting performance depends directly on the correct selection of the regression variables L_i, usually performed manually and hardcoded in the implementation. For Bermudan options, many implementations pick the long-term swap to maturity and the short-term swap to the next call date. This is a reasonable choice for standard Bermudan options in the standard LMM, but it does not scale to more complicated models or products. With stochastic volatility, for example, the volatility state constitutes another necessary feature. And guessing regression variables for callable exotics such as constant maturity swap spread steepeners may be perilous. What we need is a general algorithm to reliably identify the relevant features given an arbitrary transaction and pricing model.

In addition to the continuation values of callable instruments, the LSM has been widely applied to the estimation of future values of trading books or netting sets in the context of regulations such as XVA, CCR, FRTB or Simm-MVA, where values are usually computed in high-dimensional universal models, and their estimation is greatly facilitated by prior dimension reduction. We have observed empirically that the dimension is often sufficiently reducible to enable lightweight linear regression. More robust approximators such as ANN also perform better at a lower dimension. Of course, handcrafted feature selection rules do not scale to trading books, so the need for a reliable algorithm is even more urgent in this context.

Dimension-reduction algorithms are encapsulated in the encoding function $L = g(X)$. We will mainly explore linear encodings of the form $L = GX$, where $G \in \mathbb{R}^{p\times n}$ is a deterministic matrix.

Implementation details are covered, along with Python code, in our GitHub repo.[1] The notebooks 'DifferentialRegression' and 'DifferentialPCA' (best read in this order) discuss the implementation of the main algorithms, along with code and basic application examples. The more advanced 'Bermudan5F' notebook presents a practical application to Bermudan options, and reproduces the numerical examples that follow. All the notebooks are self-contained and run on Google Colab.

Dimension reduction done wrong: classic PCA

PCA is perhaps the simplest, and certainly the most celebrated, dimension-reduction algorithm. It has been extensively applied in many scientific fields, as well as in finance (both in academia and on trading desks), although it is, in our opinion, an incorrect tool to use for Derivatives. To better understand why, let us briefly recall the mechanics of PCA.

Consider an axis in $\mathbb{R}^n$ specified by the unit vector u. Define the variation of X along u by the $\boldsymbol{L}^2$ magnitude of the coordinates of X on u:

$$\mathrm{var}_X(u) = E[(X \cdot u)^2]$$

A collection of n mutually orthogonal unit vectors $u_1, \ldots, u_n$ constitutes an orthonormal basis of $\mathbb{R}^n$, so that the coordinates of X can be rewritten in terms of this basis:

$$X = \sum_{i=1}^{n} (X \cdot u_i) u_i$$

PCA identifies a particular basis (u_i) where the coordinates of X are mutually orthogonal, measures the variation of X along the axes u_i and performs dimension reduction by truncating the coordinates with immaterial variation. It is easily shown that an appropriate basis is given by the normalised eigenvectors of the covariance matrix of X, $C_X = E[XX^{\mathrm{T}}]$, and that the corresponding eigenvalues measure the variation of the coordinates of X in the eigenvector basis. PCA therefore reduces to eigenvalue decomposition:

$$C_X = P_X D_X P_X^{\mathrm{T}}$$

where D_X is the diagonal matrix of eigenvalues and P_X is the matrix of normalised eigenvectors in columns. The only data input is the covariance matrix C_X, usually estimated from a dataset of m independent realisations of X. We do not discuss in this article the subtleties of covariance, eigenvalue and eigenvector estimation from finite datasets. This is covered in the statistics and ML literature (see López de Prado (2019) for an in-depth presentation along with robust estimation techniques). In what follows, we assume sufficiently large datasets for a correct estimation.[2]

With P_X and D_X conveniently sorted by decreasing eigenvalues, the first p columns of P_X span the subspace of $\mathbb{R}^n$ with maximum variation, and the remaining $q = n - p$ columns span the error space, where the variation of X is minimised to $\varepsilon = \sum_{i=p+1}^{n} (D_X)_{ii}$. Depending on the application, PCA is applied with fixed dimension p or fixed tolerance ε, the latter being the safer option. By truncating the q least significant axes of variation, PCA projects X onto the subspace spanned by the first p eigenvectors, resulting in the low-dimensional encoding:

$$L = GX \quad \text{with } G = \tilde{D}_X^{-1/2} \tilde{P}_X^{\mathrm{T}}$$

where $\tilde{P}_X$ is the $n \times p$ matrix of the first p columns of P_X, $\tilde{D}_X$ is the upper-left $p \times p$ corner of D_X, and normalisation by $\tilde{D}_X^{-1/2}$ is optional but often convenient. Finally, X is best reconstructed from L in the sense of $\boldsymbol{L}^2$ by application of the pseudo-inverse operator:

$$\tilde{X} = HL \quad \text{with } H = G^{\mathrm{T}}(GG^{\mathrm{T}})^{-1} = \tilde{P}_X \tilde{D}_X^{1/2}$$

which re-expresses L in terms of the standard basis. PCA guarantees that the magnitude $e_X = E[(X - \tilde{X})^2]$ of the reconstruction error is bounded by ε. In fact, PCA may equivalently be expressed as the error minimisation problem $\min_{G\in\mathbb{R}^{p\times n}} e_X$.

Our objection is that PCA solves the wrong objective: we do not care about reconstruction fidelity. A large reconstruction error is acceptable when aligned with a direction of insignificant risk. Equivalently, variation is not the correct metric: a small variation of X in a direction of major risk may significantly affect the transaction. We do not want to rank axes by variation; we want to rank them by relevance for a given transaction or trading book.

Consider, for instance, an option on the spread of two strongly correlated assets with similar volatilities. Here, the state $X = (X_1, X_2)$ is the pair of asset prices. Variation is concentrated along the diagonal $X_1 = X_2$, which accounts for a major fraction of the variance and constitutes the principal PCA axis. The orthogonal anti-diagonal $X_2 = -X_1$ has negligible variation and is typically truncated by PCA. But the value of the spread option depends precisely on $X_2 - X_1$, as revealed by gradients proportional to $(-1, 1)$, exactly under Gaussian assumptions and approximately otherwise. Despite little variation, the anti-diagonal state coordinate is all that matters for the spread option. By truncating it, PCA completely misses the value and risk. What is safe to truncate is the diagonal. Despite the strong variation, diagonal coordinates are irrelevant for the spread option, for which the value is essentially constant across all states on the diagonal axis. This very simple example illustrates that PCA is unsafe (ie, it misses the principal axes of value and risk along the anti-diagonal) and ineffective (ie, it fails to truncate the irrelevant diagonal).

PCA fails precisely because it is completely unsupervised. It works only with state variation, irrespective of cashflows. Similar to all unsupervised algorithms, PCA is 'one size fits all'. Truncation is the same for all Derivatives; hence, it cannot be safe or effective. We have seen with the spread example that PCA may truncate the principal axes of value and risk. Consider now two Derivatives books, one an option on a basket of stocks, the other a basket of

[1] *See https://github.com/differential-machine-learning/notebooks.*

[2] *The notation in this paper is theoretical. X denotes a random vector in dimension n, not an m-by-n design matrix, and $C_X = E[XX^{\mathrm{T}}]$ denotes the true covariance of X rather than an estimate.*

single-stock options. Under the Gaussian assumption, the first is a one-factor product, depending only on the initial basket price. The second one, however, is a multi-factor book whose risk cannot be summarised in one linear feature of the state. One-size-fits-all solutions cannot reduce dimensions correctly. We want to identify the axes that matter most for a given schedule of cashflows and truncate the orthogonal space with negligible relevance. We need supervision.

Dimension reduction done right: risk PCA

Let us then supervise PCA and introduce price and risk labels:

$$V = v(X) \in \mathbb{R} \quad \text{and} \quad \Delta = \frac{\partial v(X)}{\partial X} \in \mathbb{R}^n$$

for a given transaction or trading book. To build insight, consider again an option on a basket of stocks with fixed weights u. Its value is a (nonlinear) function of the initial price of the underlying basket, that is, $v(X) = f(u \cdot X)$, exactly under the Gaussian assumption, approximately otherwise. It follows that the weight vector u is proportional to risk sensitivities:

$$\Delta(X) = \frac{\partial v(X)}{\partial X} = f'(u \cdot X)u \quad \text{hence } u = \frac{\Delta(X)}{|\Delta(X)|}$$

so the only meaningful feature $u \cdot X$ of the state X is immediately identified by a single risk report in some arbitrary state. Notice in this case that the stochastic vector $\Delta(X)$ has deterministic direction u and random magnitude $f'(u \cdot X)$.

Of course, identifying risk factors with a single risk report will not work in general. Consider, for example, a delta-hedged option. The risk report in the current state cannot even see the underlying stock as a risk factor. It is clear that multiple risk reports in different states are generally necessary in order to reliably identify all the relevant risk factors. Simulating a large number of risk reports is often intractable. The cost of pricing a complex transaction or trading book in many different scenarios is prohibitive, even with risk sensitivities efficiently computed with AAD. We will resolve this matter next. For now, we continue to build insight to formalise a dimension-reduction algorithm.

To generalise the previous example, consider a book that depends on p linearly independent combinations $w_i \cdot X$ of the state. The dependency on w_i is easily expressed in terms of orthogonal unit vectors u_i such that:

$$v(X) = f(u_1 \cdot X, \ldots, u_p \cdot X) \quad \text{hence } \Delta(X) = \sum_{i=1}^{p} (\partial_i f) u_i$$

where $\partial_i f$ denotes the derivative of f with respect to the ith argument. We see that the axes u_i form the basis of the space of risk reports Δ, which thus spans the same p-dimensional subspace as the p normalised eigenvectors with non-zero eigenvalues of the covariance matrix of the risk reports, $C_\Delta = E[\Delta\Delta^{\mathrm{T}}]$, with dimension n and rank p. It follows that the number p and directions u_i of the relevant features are identified by the eigenvalue decomposition of C_Δ.

Conversely, consider a dataset of risk reports where the risk with respect to state variable X_i is consistently zero, that is, $\partial v(X)/\partial X_i = 0$. It is clear, then, that, irrespective of the variation of X_i, the ith coordinate of X is irrelevant for this particular transaction and safely truncated, so states are represented by the remaining $n - 1$ coordinates, and dimension is reduced by 1. More generally, when the magnitude of directional risk $E[(u \cdot \Delta)^2]$ along some axis u is immaterial with respect to some threshold ε, direction u is irrelevant and safely truncated, so that states are represented by their coordinates in the $(n - 1)$-dimensional subspace orthogonal to u without materially affecting the transaction. Those irrelevant axes live in the error space of the covariance matrix C_Δ of risk reports, spanned by its q least significant eigenvectors with cumulative eigenvalues bounded by ε.

Those considerations suggest performing dimension reduction by PCA on risk reports Δ in place of states X. More formally, we define the weak relevance of some axis u by the $\boldsymbol{L}^2$ magnitude of risk along this axis (strong relevance is defined in the next section):

$$\mathrm{rel}(u) = E[(u \cdot \Delta)^2]$$

In other words, relevance is the variation of risk. Orthogonal axes of decreasing relevance are given by the eigenvectors P_Δ of C_Δ, where relevance is measured by the corresponding eigenvalues D_Δ. Dimension reduction is performed by truncating P_Δ to $\tilde{P}_\Delta$ by removal of the last q columns: risk PCA is PCA on risks.

The lower-dimensional encoding of state X is given by $L = GX$ with encoder $G = \tilde{P}_\Delta^{\mathrm{T}}$ (normalisation is unnecessary here), decoder $H = \tilde{P}_\Delta$ and reconstruction $\tilde{X} = HL$. The state reconstruction error $e_X = E[|X - \tilde{X}|^2]$ is unbounded, because reconstruction fidelity is not the objective here. What is bounded by ε is the risk approximation error:

$$e_\Delta = E[|\Delta - \Pi\Delta|^2] = E[|\Sigma\Delta|^2]$$

where the projection operator $\Pi = HG$ and the error operator $\Sigma = I_n - \Pi$. The price approximation error is also bounded. Rewriting the price as a function of the lower-dimensional state encoding:

$$\tilde{V} = v(\tilde{X}) = v(HL) = v(\Pi X)$$

by first-order Taylor expansion, using the symmetry of Σ and the Cauchy–Schwarz inequality, results in:

$$\begin{aligned} E[(V - \tilde{V})^2] &\approx E[(\Delta \cdot (X - \tilde{X}))^2] = E[(\Delta \cdot \Sigma X)^2] \\ &= E[(\Sigma\Delta \cdot X)^2] \leqslant E[|\Sigma\Delta|^2]E[|X|^2] \leqslant \varepsilon E[|X|^2] \end{aligned}$$

Finally, risk sensitivities are re-expressed in terms of derivatives with respect to L, denoted by:

$$S = \frac{\partial \tilde{V}}{\partial L} = H^{\mathrm{T}}\Delta \in \mathbb{R}^p$$

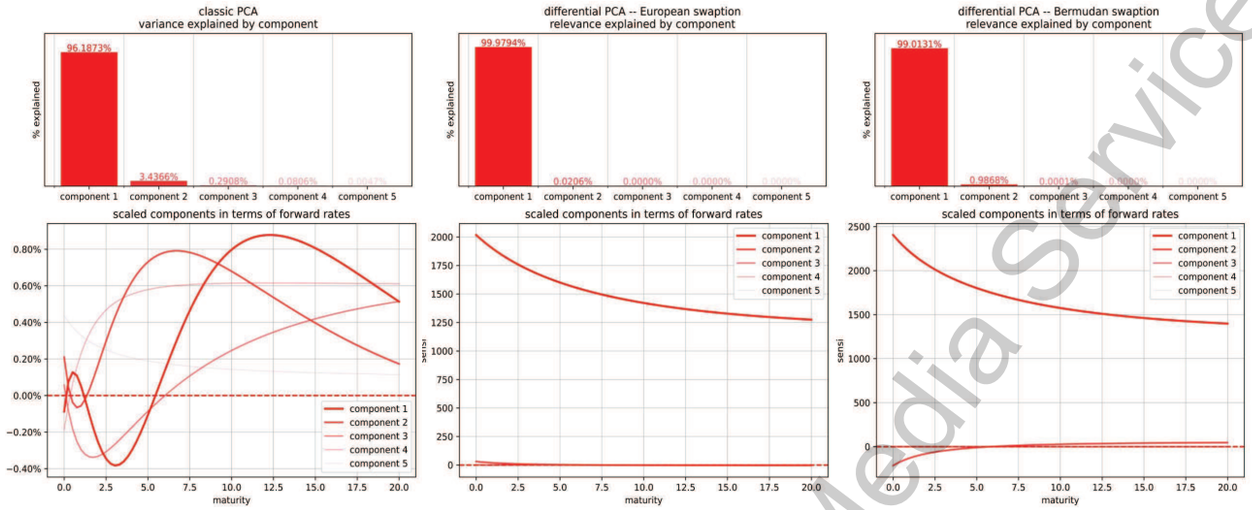
where it can easily be seen that risks with respect to features L are indeed orthogonal:

$$E[SS^{\mathrm{T}}] = H^{\mathrm{T}} C_\Delta H = \tilde{D}_\Delta$$

Risk PCA therefore provides a safe dimension reduction where the magnitude of truncated risk does not exceed a specified threshold ε. Price approximation error is also bounded, proportionally to ε. Because it implicitly minimises the risk approximation error e_Δ, risk PCA is optimal in the sense that further (linear) dimension reduction cannot be performed without increasing the magnitude of risk errors above ε. Similar to classic PCA, the implementation is remarkably lightweight and limited to the eigenvalue decomposition of C_Δ. However, because the estimation of C_Δ requires a simulated dataset of risk reports with prohibitive computation cost, risk PCA does not provide a practical algorithm: it is more a framework for reasoning about dimension reductions.

Before we turn to the practical matter of performing risk PCA without the expense of simulated risk reports, let us briefly discuss a very simple and

1 Numerical example: five-factor Gaussian interest-rate model



Although the variation of the Gaussian factors is almost fully explained by the first two components, the three remaining components, despite negligible variation, translate into significant deformations of the forward curve. In particular, the fifth component is a major driver of the short end. It follows that dimension cannot be safely reduced by truncation of the (apparently) insignificant components.

Differential PCA correctly identifies the European swaption as a one-factor instrument, depending only on the underlying forward swap (given deterministic volatility). Dimension is safely reduced to one by truncation of the four irrelevant factors.

By contrast, the Bermudan swaption depends on two independent factors: besides a shift accounting for 99% of the relevance, a second factor, in the shape of a steepening focused on the short end, accounts for a non-negligible 1%. Dimension is safely reduced to two by truncation of the three irrelevant factors, but further reduction to dimension one deletes critical information from data.

Eigenvectors are expressed in terms of forward rates and normalised by root eigenvalue. See 'Bermudan5F.ipynb' on GitHub for detailed discussion and code

useful way to discriminate between factors affecting the transaction in a linear or nonlinear manner. We introduced risk PCA by eigenvalue decomposition of the non-central covariance matrix $C_\Delta = E[\Delta\Delta^{\mathrm{T}}]$, decomposing $\mathbb{R}^n$ into directions of orthogonal risk, with risk magnitude measured by eigenvalues. An alternative representation is given by the eigenvalue decomposition of the usual, central covariance matrix $\mathrm{cov}(\Delta) = C_{\Delta - E\Delta}$. The subtraction of the average risk effectively sets the magnitude of constant risks to zero. Directions of constant risk, which are the axes of linear risk, are therefore truncated, and axes of nonlinear risk are picked up. Central risk PCA sorts and truncates axes by nonlinear relevance.

In general, we want to correctly identify all directions of risk with a non-central risk PCA. In specific applications, we may want to consider only nonlinear risk axes with central risk PCA. Notice that one effect of central risk PCA is to remove first-order hedges from the picture. A delta-hedged option, for example, is analysed and encoded in an identical way to the naked option. It is often useful in practice to perform both flavours of risk PCA.

Practical dimension reduction: differential PCA

In order to turn risk PCA into a practical algorithm, we must circumvent the prohibitive simulation of a dataset (X, V, Δ) of m independent value and risk reports. We solved the problem in Huge & Savine (2020), in the spirit of the original LSM paper, by learning the pricing function v from a dataset of cashflow samples Y in place of prices V and pathwise differentials $Z = \partial Y/\partial X$ in place of risk reports $\Delta = \partial V/\partial X$. Each training example X is labelled by one payoff Y, independently sampled by simulation of one MC path under the risk-neutral measure of the pricing model conditional on the market state X at T. Payoffs are sampled for the computation cost of one MC path, a fraction of the cost of pricing, and pathwise differentials Z are efficiently computed with AAD for a small additional cost. The entire dataset of m examples (X, Y, Z) is therefore generated for a computation cost similar to a single pricing by MC, which is orders of magnitude cheaper than a collection of risk reports.

Given that $V = E[Y \mid X]$ (ignoring discounting) and $\Delta = E[Z \mid X]$ (assuming appropriate smoothing of discontinuous cashflows), Y and Z are unbiased (if noisy) estimates of V and Δ, respectively. This observation allows training on the tractable dataset (X, Y, Z) in place of the prohibitively expensive (X, V, Δ): a result we have formally demonstrated by showing that pricing functions learned by universal approximators from the two datasets by minimisation of the mean-squared-error (MSE) both converge to the correct pricing function in the limit of infinitely large datasets and infinite capacity.

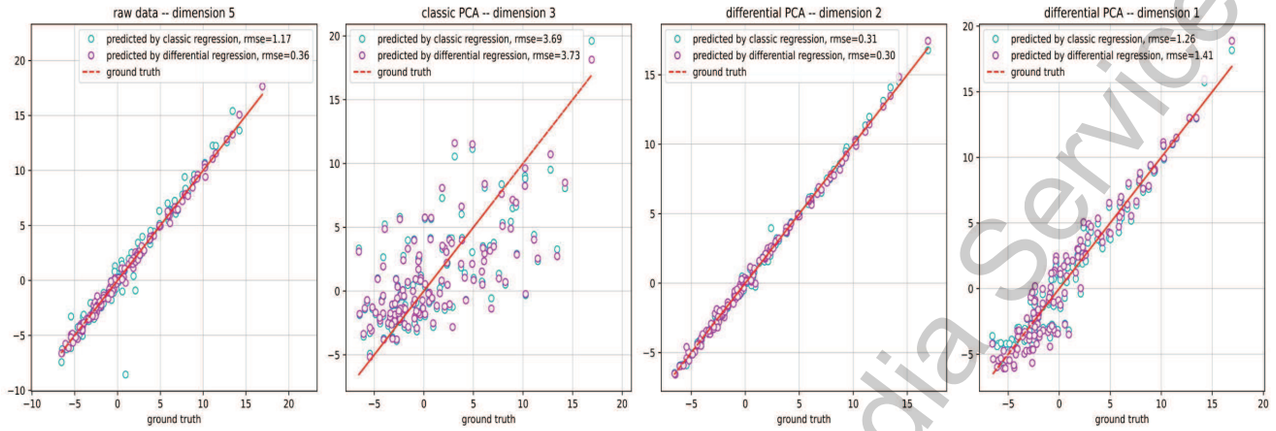
Can we similarly approximate risk PCA (also known as eigenvalue decomposition of $\mathrm{cov}(\Delta)$) by differential PCA, defined as the eigenvalue decomposition of $\mathrm{cov}(Z)$? At first sight, the answer will be no, because the covariance matrices do not in general coincide. By the law of total covariance:

$$\mathrm{cov}(Z) = \mathrm{cov}(\Delta) + E[\mathrm{cov}(Z \mid X)]$$

the additional covariance picked up in the simulation of the payoff does not vanish in the limit of infinitely large datasets and generally affects the eigenvalues and eigenvectors. It turns out, however, that differential PCA is a close approximation of risk PCA, and, perhaps more importantly, a conservative approximation, due to the following key property. Define $\Pi = \tilde{P}_Z \tilde{P}_Z^{\mathrm{T}}$ as the projection operator and $\Sigma = I_n - \Pi$ as the corresponding error operator; then, using Jensen's inequality, we have:

$$E[|\Sigma\Delta|^2] = E[|E(\Sigma Z \mid X)|^2] \leqslant E[E(|\Sigma Z|^2|X)] = E[|\Sigma Z|^2]$$

2 Continuation value of the Bermudan option in the five-factor model, learned from 8,192 examples



Although differential regression learns a decent continuation value from raw data in dimension five (classic regression needs more data), this cannot be generalised to arbitrary exotics or trading books, where dimension is often too high for regression without prior dimension reduction.

Classic PCA removed critical information from the data by truncation of the (apparently insignificant) fourth and fifth components, so the continuation value cannot be correctly learned from reduced data.

By contrast, differential PCA safely reduces dimension to the two identified relevant factors. The continuation value learned from reduced data is more accurate than with raw data (slightly with differential regression and substantially with classic regression), providing evidence that no meaningful information was lost by truncation of the three irrelevant components.

However, attempting to further reduce dimension to one by truncation of the differential PCA's second factor, which explains 1% of the total relevance, removes relevant information from data so the continuation value can no longer be learned accurately. This is consistent with Andersen and Andreasen's classic result, verified here from simulated data by a straightforward application of differential PCA.

Performance measured on 128 out-of-sample scenarios against ground truth. See 'Bermudan5F.ipynb' on GitHub for detailed discussion and code

The true risk magnitude truncated by differential PCA is lower than the magnitude of the truncated differentials. By construction of differential PCA, $e_Z = E[|\Sigma Z|^2] \leqslant \varepsilon$; hence, the approximation error of true risk is also bounded by ε. And it immediately follows that the price error e_V is bounded too, as seen previously. By the triangular inequality, the risk approximations from differential PCA and a hypothetical risk PCA are both distant from the true risk by less than ε, so they must be at a distance less than 2ε from each other. The same immediately also applies to price approximation. It follows that differential PCA is a conservative, close approximation of risk PCA.

Differential PCA minimises the differential reconstruction error $e_Z = E[|Z - \tilde{Z}|^2]$ and decomposes the state space according to a stronger measure of relevance:

$$\mathrm{Rel}(u) = E[(u \cdot Z)^2]$$

Strong irrelevance implies weak irrelevance (and weak relevance implies strong relevance), so strongly irrelevant axes truncated by differential PCA are also weakly irrelevant; hence, they would be truncated by a hypothetical risk PCA too. Differential PCA is safer than risk PCA. Intuitively, it correctly reconstructs differentials path by path, whereas risk PCA reconstructs them only on average. Conversely, differential PCA may consider relevant some features that risk PCA would otherwise truncate. We can always relax the threshold ε accordingly. The implementation of differential PCA is otherwise a carbon copy of risk PCA, and all the comments made previously for risk PCA apply.

The computational complexity of differential PCA is dominated by the simulation of the dataset, the computation of the covariance matrix C_Z and its eigenvalue decomposition. We have seen that the dataset is efficiently simulated with AAD at a cost similar to a single pricing by MC. This also holds for multiple exposure dates, because we can reuse the same simulated cashflows, aggregated in different ways. The estimation of $C_Z = E[ZZ^{\mathrm{T}}]$ has complexity $O(n^2 m)$ for m examples on each exposure date. A high-performance computing library such as Intel's Math Kernel Library (MKL) computes the covariance of 32,768 examples in dimension 1,024 in about 0.25 seconds on a midrange trader workstation, or 1 second in dimension 2,048. The eigenvalue decomposition of the n-by-n covariance matrix has cubic complexity. MKL computes the eigenvalues and eigenvectors of a real symmetric $1,024 \times 1,024$ matrix in about 0.05 seconds, or 0.4 seconds in dimension 2,048. The cumulative complexity of other computations is insignificant. Differential PCA takes only a fraction of a second to process one horizon date, a time negligible by risk computation standards, irrespective of the size of the trading book and in dimensions up to several thousands. When in doubt, processing is easily executed on a graphics processing unit, with a library such as TensorFlow giving another order of magnitude speedup.

Nonlinear features and differential auto-encoders

Similar to its classical counterpart, differential PCA identifies only linear features of a state of the form $L = GX$ (with reconstruction $\tilde{X} = HL$). The dimension may be further reduced by considering a wider class of nonlinear encodings of the form $L = g(X)$, with reconstruction $\tilde{X} = h(L)$. A now classic ML construct known as an auto-encoder (AE) effectively extends PCA to nonlinear encoding. The encoder $g\colon \mathbb{R}^n \to \mathbb{R}^p$ and decoder $h\colon \mathbb{R}^n \to \mathbb{R}^p$ are represented by two ANNs, with n and p units, respectively, in their input layer and p and n units, respectively, in their output layer, combined into one ANN with a characteristic bottleneck of size p on the junction. The connection weights of the AE are trained by explicit minimisation of the reconstruction error e_X, whose gradient with respect to the connection weights is efficiently computed by back-propagation on every iteration. (AEs have also been explored in the finance literature, notably in Kondratyev (2018), in the context of interest-rate curves.)

It is therefore tempting to extend differential PCA to differential AE, in an attempt to identify the most relevant nonlinear features of the state X for a given transaction or trading book. But we already know the answer: the one nonlinear feature of the state that correctly encapsulates all relevant information is the price $v(X)$. Learning the relevant nonlinear features is identical to learning the pricing function v. We have discussed the pricing problem in depth in Huge & Savine (2020), where we introduced differential ANNs and demonstrated their effectiveness for learning pricing functions from differential datasets (X, Y, Z).

More specifically, the output layer of a regression ANN is a linear combination of the nonlinear features encoded in the penultimate layer. The *raison d'être* of ANNs is to learn relevant nonlinear features in their hidden layers. This is what distinguishes them from linear regressions (where regression features are fixed), and qualifies them as artificial intelligence constructs. This also explains why ANNs are resilient at high dimensions, whereas linear regression is not.

Differential autoeconders are implicitly embedded in differential ANNs. Although differential PCA identifies linear features of the state, of which the price is a nonlinear function, differential ANNss identify nonlinear features of the state and combine them linearly into a price. The inspection of the features encoded in the penultimate layer provides insight into price formation and helps explain and interpret pricing by ML.

Differential linear regression

One benefit of differential PCA is to potentially reduce the dimension sufficiently to enable linear regression. There is no guarantee that the dimension is sufficiently reducible in a given context, but we have observed empirically that this is often the case. When linear regression is applicable by prior differential PCA or otherwise, its differential variant offers considerable benefits. Note that a training set for differential PCA already includes differential labels, and these differential labels may be further leveraged to improve regression.

Recall that linear regression approximates Y by a linear combination $\beta \cdot \phi(X)$ of k basis functions $\phi(X) \in \mathbb{R}^k$ of X, most commonly monomials or basis spline functions. Because the number k of basis functions coincides with the dimension of the weight vector β and grows exponentially with dimension n, linear regression quickly becomes intractable and vulnerable to overfitting when the dimension grows. The weight vector is found by the minimisation of $\text{MSE} = E[(Y - \beta \cdot \phi(X))^2]$, with a unique and analytic solution:

$$\beta^* = C_\phi^{-1} C_{\phi,Y}$$

where $C_\phi = E[\phi(X)\phi(X)^\text{T}]$ and $C_{\phi,Y} = E[\phi(X)Y]$ are estimated over a training set, and the inversion is often stabilised with singular-value decomposition and further Tikhonov regularised, that is, $\beta^* = (C_\phi + \lambda I_k)^{-1} C_{\phi Y}$ for some regularisation strength λ.

Linear regression does not use differential labels Z. We argued in Huge & Savine (2020) that when those labels are available, as they are in finance with AAD, a much stronger form of regularisation can be applied by minimising a combination of value and derivative errors. In the case of linear regression, we have a mixed optimisation objective:

$$\beta^* = \arg\min \left\{ E[(Y - \beta \cdot \phi(X))^2] + \sum_{i=1}^{n} \lambda_i E[(Z_i - \beta \cdot \partial_i \phi(X))^2] \right\}$$

where $\partial_i \phi(X) = \partial \phi(X) / \partial X_i \in \mathbb{R}^k$ and the relative weights λ_i are commonly set to $E[Y^2]/E[Z_i^2] \in \mathbb{R}$.

Unlike other forms of regularisation, differential regularisation does not introduce a bias and more generally massively improves the performance of ML models, as discussed in Huge & Savine (2020) in the context of ANNs. The conclusions carry over to linear regression as a particular and simple case. In addition, the solution remains analytic and can be given by a modified normal equation:

$$\beta^* = \left(C_\phi + \sum_{i=1}^{n} \lambda_i C_{\partial_i \phi} \right)^{-1} \left(C_{\phi Y} + \sum_{i=1}^{n} \lambda_i C_{\partial_i \phi, Z_i} \right)$$

where:

$$C_{\partial_i \phi} = E[\partial_i \phi(X) \partial_i \phi(X)^\text{T}] \in \mathbb{R}^{k \times k}$$
$$C_{\partial_i \phi, Z_i} = E[\partial_i \phi(X) Z_i] \in \mathbb{R}^k$$

are estimated over a training set.

Conclusion

We have extended the ideas of differential machine learning introduced in Huge & Savine (2020) to resolve the matter of dimension reduction, introducing a safe, effective and efficient differential PCA. Similar to differential ANNs or regression, differential PCA is based on AAD pathwise differentials. Unlike differential ANNs or regression, differential PCA is not just a much more effective variant of PCA but also a very different algorithm. PCA cannot safely reduce the dimension for a given collection of event-driven cashflows, even in the limit of an infinite dataset. Only differential PCA can.

The factors revealed by differential PCA help to explain and manage the specific risks of a given trading book and calibrate pricing models in a reliable manner. In the context of least-squares Monte Carlo, differential PCA safely identifies a minimum number of regression variables, tidying up a long-standing loose end with respect to Bermudan options and other callable exotics. Differential PCA provides a very effective preprocessing of datasets for learning pricing and risk functions, often reducing the dimension sufficiently to enable lightweight linear regression. Although more sophisticated ML models such as ANNs are robust at a high dimension, they remain faster and more stable at lower dimensions. Differential PCA is lightweight, fast, safe and essentially zero cost, and hence best implemented as a systematic preprocessing step. ■

Brian Huge is a senior specialist quant at Saxo Bank, while Antoine Savine is a chief quantitative analyst with Superfly Analytics at Danske Bank. They developed differential machine learning while working with Superfly Analytics at Danske Bank.
Email: brian.huge@saxobank.com, antoine.savine@danskebank.com.

REFERENCES

Andersen L and J Andreasen, 2001
Factor dependence of bermudan swaptions: fact or fiction?
Journal of Financial Economics
62(1), pages 3–37

Huge B and A Savine, 2020
Differential machine learning: the shape of things to come
Risk October, pages 76–81,
www.risk.net/7688441

Kondratyev A, 2018
Curve dynamics with artificial neural networks
Risk June, pages 74–79,
www.risk.net/5632496

López de Prado M, 2019
A robust estimator of the efficient frontier
Preprint, SSRN