Principled machine learning

Yordan Raykov and David Saad

***Abstract*—We introduce the underlying concepts which give rise to some of the commonly used machine learning methods ~~ideas~~, excluding deep-learning machines and neural networks. We point to their advantages, limitations and potential use in various areas of photonics. The main methods covered include parametric and non-parametric regression and classification techniques, kernel-based methods and support vector machines, decision trees, probabilistic models, Bayesian graphs, mixture models, Gaussian processes, message passing methods and visual informatics.**

***Index Terms*—Statistical machine learning, kernel-based methods, probabilistic methods, deciion trees, message passing techniques, dimensionality reduction, visual informatics**

I. Introduction

Engineering successes of Deep Learning Machines (DLM) have both fascinated and bewildered the engineering and scientific communities in recent years, drawing attention to the potential presented by this new form of established neural network methodology. While there is clear evidence for the success of DLM, the substantial effort invested in trying to understand how they work and to establish a clear, principled and rigorous mathematical framework has had limited success. Nevertheless, neural networks in general and DLM in particular have become a ubiquitous and commonly-used tool in many application domains. So much so, that the term *machine learning* has almost become synonymous to neural networks-based tools.

Machine learning refers to a collection of *data-driven* methods, both principled and heuristic, aimed at carrying out a range of non-trivial tasks including regression, classification, optimization, forecasting, dimensionality reduction and visual informatics. Our view is that Artificial Intelligence encompasses all methods used for carrying out "intelligent" tasks, including rule-based methods and heuristics, while machine-learning techniques focus on data-driven methods, one of which are neural networks and DLM is a specific manifestation of it.

While neural networks and DLM have gained popularity in the last decade and are commonly used across a broad range of applications, there are several reasons to consider more principled machine learning techniques. Some of the methods offer interpretability and explainability of the results obtained, which are particularly important when critical decisions should be taken and for gaining insight into the rationale behind the decisions; kernel-based methods extend interpolation techniques, allowing one to reflect the nature of the expected functions through the choice of kernel; probabilistic methods offer confidence levels, as they estimate the uncertainty in the outcomes and accommodate noise and missing data in a principled, natural and controlled way; other probabilistic methods such as mixture model-based density estimation and message passing techniques deliver controlled approximations to hard modeling, inference and optimization tasks; visual analytics and dimensionality reduction facilitate the mapping of a high dimensionality data onto a low-dimensinonal space where they can be intuitively understood by users and decision makers. Many of the methods are inherently adaptable and do not require costly retraining when new data ~~is~~are observed.

The aim of this paper is not to provide a comprehensive review of machine learning methods, many such reviews exist already, but to introduce the main concepts behind some of the pivotal methods in machine learning research, excluding DLM and neural networks, and point to exemplar potential applications in photonics. We will provide a brief description of the principles behind the different methods and refer the reader to the corresponding literature for the specific details.

Section II reviews the use of established regression and classification methods and highlights recent advances in this area, while Sec. III introduces nonparametric regression and kernel methods followed by the suggestion of decision trees and clustering techniques. Probabilistic approaches and reviewed in Sec. IV including Bayesian graphs, mixture models, Gaussian processes and message-passing techniques. Finally, we introduce visual informatics methods in Sec. VI followed by a brief review of the different methods used in the broad area of photonics in Sec. VII. The conclusions in Sec. VIII point to material that has not been reviewed in this tutorial and to the potential use of principled machine learning in photonics applications.

II. Ordinary regression and beyond

The majority of prediction tasks can be efficiently structured as *regression* problems in which the aim is to infer the associative mapping $f : \mathcal{R}^P \rightarrow \mathcal{R}$ between P-dimensional data $\boldsymbol{X} = \{x_p^{(n)}\}_{n=1,p=1}^{N,P}$ and desired continuous targets $Y = \{y^{(n)}\}_{n=1}^{N}$; N is the number of observed samples. Once we have learned to link $\boldsymbol{X}$ and $\boldsymbol{Y}$, we can make predictions about $\boldsymbol{Y}$ given $\boldsymbol{X}$. One way to unify different *supervised* (when target data are provided) machine learning paradigms is to think of them as different ideas for constraining and learning the approximate mapping f.

The simplest non-trivial[1] assumption about f is that it can be represented as a linear map, i.e. $f(\boldsymbol{x}^{(n)}) = \beta_0 + \sum_{p=1}^{P} \beta_p x_p^{(n)}$. Commonly, we augment the input vector $\boldsymbol{x}^{(n)}$

Yordan Raykov is affiliated with the Horizon Digital Economy Research Institute and the Statistics and Probability Research Group, University of Nottingham, United Kingdom e-mail: yordan.raykov@nottingham.ac.uk

David Saad is with the Non-linearity and Complexity Research Group, Aston University, Birmingham B4 7ET, United Kingdom e-mail: D.Saad@aston.ac.uk

Manuscript received October, 2021.

[1] Excluding the constant map f.

in order to merge the intercept β_0 and the linear coefficients into simple vector form of the map: $f(\boldsymbol{x}^{(i)}) = \boldsymbol{\beta}\widehat{\boldsymbol{x}}^{(n)}$ with $\widehat{\boldsymbol{x}}^{(n)} = [1, \boldsymbol{x}^{(n)}]^T$. In *ordinary least squares regression* we assume that the mismatch between a linear model and the observed targets, can be adequately captured using the squared Euclidean distance between the two. Determining the regression parameters from data (commonly termed) as *training*) is then achieved by optimizing the objective:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \sum_{n=1}^{N} \left(y^{(n)} - \boldsymbol{\beta}\widehat{\boldsymbol{x}}^{(n)} \right)^2 \tag{1}$$

For most applications, the ordinary linear assumption is too rigid. However, it remains a go-to benchmark in part due to its simple resulting inference. We can find the parameters of the optimal least squares regression map f directly by differentiating Eq. (1) and solving for $\boldsymbol{\beta}$. The solution is available in a closed form:

$$\begin{aligned} \widehat{\boldsymbol{\beta}}_1 &= \left(\boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{Y} \\ \widehat{\boldsymbol{\beta}}_0 &= \boldsymbol{Y} - \boldsymbol{\beta}_1 \boldsymbol{X} \end{aligned} \tag{2}$$

Note that the more flexible regression formalisms we review, require iterative training procedures. We can rarely hope for ”perfect fit” of a linear model, so it is common to use a random variable (RV) capturing the residual error of the regression model, i.e. assuming $f(\boldsymbol{X}) = \boldsymbol{\beta}\boldsymbol{X} + \epsilon$. This facilitates statistical inference of quantities such as confidence intervals, an estimate expected error or significance of goodness-of-fit.

As long as linearity with respect to the regression parameters is kept, the optimization objective remains convex and solvable in closed form. This result is used to motivate a wider set of regression algorithms, trained by solving the objective function associated with *generalized linear models*:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \sum_{n=1}^{N} \left(y^{(n)} - g(\boldsymbol{\beta}\boldsymbol{X}) \right)^2 \tag{3}$$

where $g(\cdot)$ are *link functions* that take any convenient parametric form[2]. Common choices for $g(\cdot)$ include: log, exponential, logit and other functions [1]. Note that appropriate choice of a function (such as the logit - $\log \frac{x}{1-x}$) can be used to constrain the regression to a classification problem, in a probabilistic setting. In Fig. 1, we display a simple univariate regression example, approached with different generalized linear regressions (a)-(c), as well as a simple nonparametric regression (d), motivated in the next section. Single layer neural networks can be seen as special types of generalized linear regression.

1) Interaction effect: The same principle can be used to construct predictors which reflect pairwise and higher order interactions between the original variables, for example assuming $\widehat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$. It is easy to see that the space of potential models (and parameters to determine) grows as a power low with the order of interactions which is a particular problem in large dimensional data P. Even if capture only pairwise interactions, the computational complexity would be in the order of $O(P^2)$. This challenge can be addressed implicitly by adopting *sparse* regression model, shrinking the data dimensionality [2], [3], or explicitly using *kernel methods* to compute more efficiently second order interactions [4]. Recent work [5] proposes a generic, fully probabilistic framework for regression, which allows one to estimate all pairwise significant interactions with linear complexity in P.

2) Collinearity: The core problem of channel equalization in digital communication, where the channel affects the transmitted sequence with different distortions, can be seen as a regression problem in which one aims to estimate a map $f(\cdot)$ which predicts samples $y^{(n)}$ from the transmitted signals using multiple channel responses $\left\{ x_1^{(n)}, \ldots, x_m^{(n)} \right\}$ [6]. Assuming a linear model, the estimation task requires learning a set of weights $\{\beta_1, \ldots, \beta_m\}$ reflecting the individual contribution of each of the m channels. However, a lot of the channel responses often reflect the same information about the transmitted signal at a given time (particularly if one assumes linear-only distortion). In this scenario, if one uses a large number m of correlated channels, it leads to redundancy in the representation and an estimate of more complex regression models[3] than needed. In linear regression, this problem is known as *collinearity* which is not just computationally inconvenient, it also leads to less robust regression models which are overly confident in their prediction and perform poorly out-of-sample. Channel variables should have minimal overlap between the information they carry. This desired property has motivated multiple studies on sparse regression techniques [7]–[10], which describe different mechanisms for controlling the growth of regression models.

One of the most well known and still computationally simple sparse regression technique is the *least absolute shrinkage and selection operator* (LASSO) regression [8]. LASSO augments the defining objective of the least squares regression with an l_1 constraint on the weights $\boldsymbol{\beta}$, solving:

$$\begin{aligned} &\underset{\boldsymbol{\beta}}{\text{minimize}} \sum_{n=1}^{N} \left(y^{(n)} - g(\boldsymbol{\beta}\boldsymbol{X}) \right)^2 \\ &\text{subject to} \sum_{j=1}^{m} |\beta_j| \le \lambda \end{aligned} \tag{4}$$

where λ is a pre-specified free parameter that determines the degree of regularization.

It is worth highlighting that the sparse linearity assumption is also exploited in modern neural network architectures where ReLU (Rectified Linear Unit) dense layers in essence predict continuous outputs using multiple “nearly linear” regression with certain input observations suppressed to 0 (i.e. rather then whole variables).

3) Piecewise linearity: In both of the examples above, one derives flexible regression maps, keeping the associated training tractable in closed form. This simplicity largely stems from the global implicit assumption that the regression parameters $\boldsymbol{\beta}$ are shared across all data pairs $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$

[2] The effect of $f(\cdot)$ can be trivially consumed by the pre-processing of our data, so for brevity we will assume identity.

[3] In the context of linear models, complexity is measured with respect to the number of dimensions.

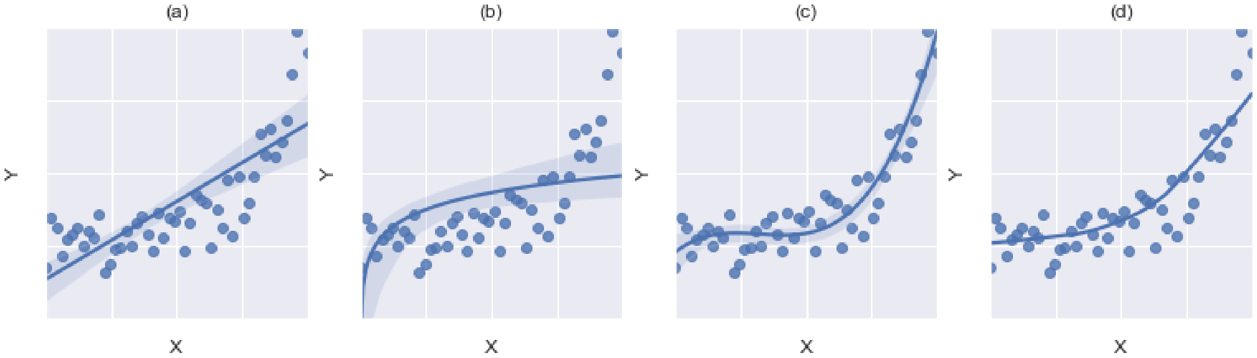

Fig. 1. Different regression curves fitted to example synthetic pairs $\{x^{(n)}, y^{(n)}\}_{n=1}^{N}$. (a) ordinary least squares fit; (b) log-linear regression; (c) order 3 polynomial regression; (d) locally polynomial regression (or smoother). Confidence intervals are indicated by the shaded areas, with the exception of (d) where confidence intervals are omitted due to the added complexity.

and $\boldsymbol{\beta}$ is obtained from a *sufficient statistics* of the full set of samples. This means that we cannot account for potential changes in the regression map, for instance if dependence on the data is heteroscedastic. Perhaps the simplest examples of this problem are inference problems where outliers are present in the data.

In channel equalization, if a small number of channel responses are large in value and far from the majority of responses (i.e. for example due to distortion), the estimates of $\boldsymbol{\beta}$ would be strongly influenced by an insignificant number of disproportionately large channel responses $\boldsymbol{x}^{(n)}$. Another practical scenario is that of *stratified regression*, where the expected map behaves differently in different regions of the input space. For example, if we know of a fixed number of different environments where one expects different distortion types in channel responses. The impact of some trivial statistical outliers can be mitigated by using a *robust regression assumption* [11] which uses transformed sufficient statistics that ignore the tail of the predictor values (i.e. training values with many standard deviations away from their sample mean). Given prior information about the partitioning of the input space, we can derive an explicit stratified regression, but as we move towards "local" or piece-wise regression models, the problem quickly becomes more conceptually and computationally challenging.

Consider the least squares regression problem of Eq. (3), a natural way to express the heterogeneity assumption of the underlying regression map f would be to assume that f is a weighted superposition of a fixed number of simpler maps:

$$\widehat{y}^{(n)} = \sum_{k=1}^{K} \omega_k g\left(\boldsymbol{\beta}\boldsymbol{X}\right) \tag{5}$$

Determining the parameters would require optimization of an appropriate loss function with respect to $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$; this can be done iteratively using more computationally intensive inference algorithms. Decomposing learnable functions into a discrete superposition of simpler components is often applied beyond the scope of simple linear models and is widely referred to as *mixture of experts* [12], [13]. In the special linear case, we refer to the regression approach above as *weighted least squares* regression, and depending on the assumption specified for $\boldsymbol{\omega}$, we can extend this to flexible probabilistic piece-wise linear regression for a number of underlying heteroscedasctic components [14]. Assuming $K = N$ one can derive nonparametric kernel regression variations as discussed below.

III. Nonparametric Regression and Kernel Methods

Nonparametric regression and classification algorithms allow for inference without making explicit assumptions about the parametric form of the underlying regression curve f or decision boundary. This is done by learning observation-specific contributions towards a target estimator, rather then inferring some fixed set of parameters describing the shape of f. Nonparametric regression methods are not parameter free, but the number of their parameters depends on the number of training data points, rather then the assumed parametric form of f. An intuitive example of nonparametric regression are running average algorithms which estimate new targets as a weighted superposition of historic values[4].

A. Kernel density estimation

Given an observed set of values $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$, the *kernel density estimator* (KDE) is a nonparametric method to estimate the probability density of $\boldsymbol{X}$. We write the KDE as:

$$\widehat{f}_{\sigma}(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} K_{\sigma}\left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right) \tag{6}$$

where $K(\cdot)$ denotes a *kernel function* which satisfies properties of a *distance function* and is commonly used for measuring different types of proximity in machine learning problems; σ denotes the lengthscale of the kernel. The choice of σ controls the trade-off between the bias of the estimator and its variance: KDE with smaller σ are less biased, but likely to overfit the sample set as demonstrated in Fig. 2. The

[4] Moving averages are a very simple type of kernel smoothers and they have been used since at least the late 19th century [15].

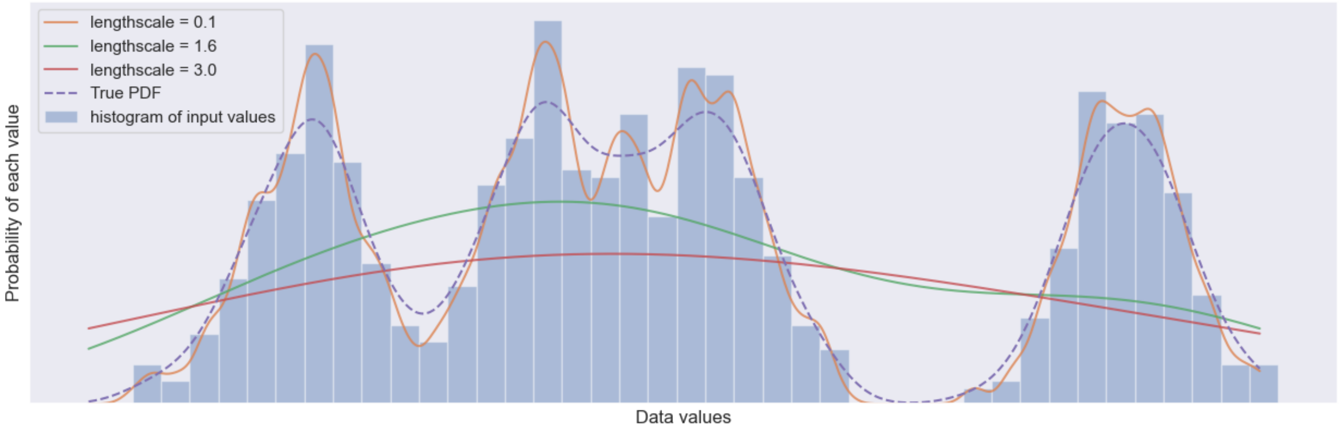

Fig. 2. Kernel density estimation of 4-component Gaussian mixture using varying kernel lengthscale and the Nadaraya–Watson kernel estimation. Kernel lengthscale which is too low leads to *overfitting* and too high lengthscale leads to *underfitting* the sample set.

choice of kernel $K_\sigma(\cdot)$ is often dictated by the measurements $\boldsymbol{X}$: polynomial kernel takes into account both individual input features and combination of features when determining their similarity; periodic kernel allows one to compare input features originating from repeating patterns and others [16]. Note that in the special case where $\sigma \to 0$ and $K(\cdot)$ becomes a Dirac delta function, KDE acts as a sample histogram.

1) Kernel regression: The KDE allows one to define a data driven density model for observed $\boldsymbol{X}$. Using this definition, it is possible to construct a *model agnostic* regression estimator from an observed set of pairs $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$ for unseen outputs such that $y^* = f(\boldsymbol{x}^*)$. The quantity of interest is the expectation $E[Y|\boldsymbol{X}=\boldsymbol{x}] = \int \frac{P(\boldsymbol{x},y)}{P(\boldsymbol{x})}dy$ from Bayes rule. Using the KDE for $P(\boldsymbol{x})$ from Eq. (6) and the product KDE for $P(\boldsymbol{x},y) = \frac{1}{N}\sum_{n=1}^{N} K_\sigma\left(\boldsymbol{x}-\boldsymbol{x}^{(n)}\right) K_\sigma\left(y-y^{(n)}\right)$, one obtains the Nadaraya–Watson kernel regression estimator [17], [18]:

$$f_\sigma(\boldsymbol{x}) = \frac{\sum_{n=1}^{N} K_\sigma(\boldsymbol{x}-\boldsymbol{x}^{(n)})y^{(n)}}{\sum_{n=1}^{N} K_\sigma(\boldsymbol{x}-\boldsymbol{x}^{(n)})} \tag{7}$$

The Nadaraya–Watson kernel estimates unseen targets as locally weighted averages of known targets, with weights corresponding to the kernel distance. There are few other popular ways of deriving nonparametric kernel regression estimators, where one of the key differences arises from the distinction between external and internal approaches for dealing with the unknown mapping f [19]. The Nadaraya–Watson kernel estimator is the most notable example of external approaches where data is first smoothed, before the support at observed targets is computed. In contrast, methods such as the Priestley–Chao estimator [20] and the Gasser-Muller estimators [21] are called internal approaches, where one first modifies the empirical function (i.e. focusing on unbiasedness) and kernel smoothing is carried out later.

2) K-nearest neighbour regression: A practical downside of Nadaraya–Watson kernel regression in Eq. (7) is that all targets $y^{(n)}$ are used to estimate a new target y^*, even though only the variables $\boldsymbol{x}^{(n)}$ closest to $\boldsymbol{x}^*$ contribute significantly. A popular and practical special case of kernel regression is the *K-nearest neighbour regression* (KNN), where the estimator for unseen targets y^* only uses the closest K-number of known responses $y^{(1,\dots,K)}$ measured in terms of distance between the corresponding samples $\boldsymbol{x}^{(1,\dots,K)}$ and $\boldsymbol{x}^*$. Formally, using Eucledian distance to rank neighbours, the KNN estimator for unseen y^* is:

$$\widehat{y}^* = \sum_{k=1}^{K} \frac{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\| y^{(k)}}{\sum_{k=1}^{K}\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|} \tag{8}$$

The Euclidean distance in Eq. (8) can be replaced by kernel distances, similar to Eq. (7). For the classification variant of KNN, typically the most common class among the top K neighbours is selected, rather then a weighted average, where K is subjectively selected using domain knowledge or cross validation.

3) Kernel ridge regression: A less explicit way to restrict the complexity of the kernel regression estimator, is to regularize for the complexity of $\widehat{f}$:

$$\widehat{f} = \underset{f\in\mathcal{H}}{\arg\min}\, \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - f\left(\boldsymbol{x}^{(n)}\right)\right)^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{9}$$

where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) with kernel $K(\cdot)$ and λ is a regularization parameter penalizing functions f with too large RKHS norm. Any solution $\widehat{f}$ takes the form: $\widehat{f}(\cdot) = \sum_{n=1}^{N}\alpha_n K(\cdot, \boldsymbol{x}^{(n)})$. Substituting this form in Eq. (9), one gets the training objective of the *kernel ridge regression*:

$$\widehat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in R^N}{\arg\min}\, \frac{1}{2}\|y - \boldsymbol{K\alpha}\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^T\boldsymbol{K\alpha} \tag{10}$$

where $\boldsymbol{K}$ denotes the kernel matrix with elements $K_{ij} = K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$ and $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]$. Training of kernel ridge regression proceeds with standard gradient optimization of Eq. (10) with respect to $\boldsymbol{\alpha}$.

B. Support vector machines

In the early days of machine learning, the scientific community was fascinated by one of the earliest machines

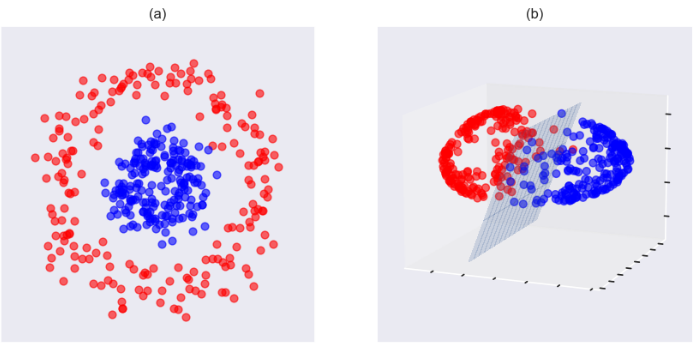

Fig. 3. (a) Synthetic data of two non-linearly-separable classes; (b) Kernel expansion of the synthetic data in 3-D using RBF kernel (detailed later), where the data becomes separable by a plane.

of this type called the *perceptron* [22], only to discover that it is highly limited to linearly-separable problems which represent a diminishing fraction of all classification problems as the dimension of the system grows (number of free variables in the input vector to be classified). For a data set comprising real or binary input vectors of dimensionality D, $\left\{\boldsymbol{s}^{(1)}, \boldsymbol{s}^{(2)}, \ldots, \boldsymbol{s}^{(N)}\right\}$ and the corresponding binary outputs $\left\{t^{(1)}, t^{(2)}, \ldots, t^{(N)}\right\}$, the perceptron classifies any new input $y = \text{sign}\left(\boldsymbol{w} \cdot \boldsymbol{s} + b\right)$ by determining the weight vector $\boldsymbol{w}$ and bias b on the basis of the data given.

The concept of Support Vector Machines (SVM) [23] originated from the idea that mapping any problem to a high-enough dimensional space facilitates linear separability in high dimension. Once the separating hyper-plane has been obtained from data, any new input could be classified at the high dimensional space. Figure 3 illustrates ~~an example of~~ how two-class non-linearly-separable observations can be mapped onto three dimensional space where they are linearly separable. Moreover, separating the two classes in the high-dimensional space, one can identify key input vectors that are sufficient for determining the hyper-plane; they are termed *support vectors* as demonstrated in Fig.4 and lie of the most distanced hyper-planes separating the two classes in the high-dimensional space. Once these have been established, any new point can be classified *without mapping it to the high dimensional space*, using the corresponding kernel.

The input vectors $\boldsymbol{s}$ are mapped to the high-dimensional space vectors $\boldsymbol{x}$ and are separated by a hyperplane. The distance between the two hyperplanes on which the support vectors lie is $2/\|\boldsymbol{w}\|$. Minimizing $\|\boldsymbol{w}\|^2$

$$L_p = \frac{\|w\|^2}{2} - \sum_{n=1}^{N} \alpha^{(n)} t^{(n)} (\boldsymbol{x}^{(n)} \cdot \boldsymbol{w} + b) + \sum_{n=1}^{N} \alpha^{(n)}$$

w.r.t the weight vector in the high-dimensional space $\boldsymbol{w}$ and the bias b, given the classification constraints manifested through Lagrange multipliers $\alpha^{(n)} \geq 0$, which maximize the separation and can be obtained via various numerical methods. This will help identify the support vectors (SV), the set of vectors closest to the decision boundary in the high-dimensional space.

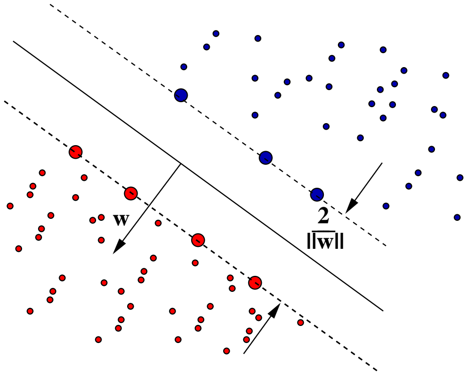

Fig. 4. Separation of classes in the high dimensional space, maximizing the distance between the two groups. Points on the closest hyperplanes, marked by larger symbols, are termed support vectors and are sufficient for establishing the two hyperplanes.

The mapping of the input vectors $\boldsymbol{s}$ to the feature space $\boldsymbol{x}$ can take different forms, for instance polynomial $x_{(ij)} = s_i s_j$ mapping, which we denote as $\Phi(\boldsymbol{s}) = \boldsymbol{x}$. Classification of a new vector takes the form

$$f(\boldsymbol{s}) = \sum_{n=1}^{SV} \alpha^{(n)} \; t^{(n)} \; \Phi(\boldsymbol{s}^{(n)}) \cdot \Phi(\boldsymbol{s}) + b.$$

Exploiting the properties of the mapping $\Phi(\cdot)$ and corresponding kernel $K(\cdot)$ to substitute $\Phi(\boldsymbol{s}^{(n)}) \cdot \Phi(\boldsymbol{s}) = K(\boldsymbol{s}^{(n)} \cdot \boldsymbol{s})$ simplifies the procedure, so that classification can take place directly without mapping the problem to the high-dimensional space; for polynomial maps of degree q - $K(\boldsymbol{s}, \boldsymbol{s}') = (\boldsymbol{s} \cdot \boldsymbol{s}' + 1)^q$. Other mappings can also be considered resulting in simple kernels, e.g., $\exp\left\{-\frac{\|\boldsymbol{s}-\boldsymbol{s}'\|^2}{2\sigma^2}\right\}$ for Radial Basis Function mappings and $\tanh(\kappa \boldsymbol{s} \cdot \boldsymbol{s}' - \delta)$ for perceptron-like mappings, where σ^2, κ and δ are some coefficients. Such mappings exist if Mercer's condition is obeyed [23] (amounting to positive definiteness of the kernel).

Originally, SVM was offered as a parameterless and straightforward approach to classification. However, the presence of noise and the representation of real-valued functions, necessitate the introduction of additional parameters, where noise estimation and the nature of the objective functions for representing real-valued functions require more information and insight [24]. Arguably, the most appropriate problems to be studied by SVM are classification tasks, although they are also being employed as kernel functions for the representation of real-valued functions.

C. Decision trees

Decision trees is~~are~~ another class of methods for supervised learning tasks. Their popularity is due to the fact they can often be used *off-the-shelf*. Decision trees rely on small amount of hyperparameters and are invariant under scaling and various other transformations. This reduces the importance of understanding data properties in training decision trees [25].

Single decision trees are among a small family of machine learning models which are easily interpretable, although tree ensembles are widely considered as black-box methods.

Decision trees allow one to construct complex regression curves f by successively partitioning the input feature space into many local regions using inductive logic. At the root of a decision tree are the input variables. Then, rules based on the variables' values are selected, to best differentiate observations based on one or more dependent variable. Decision tree algorithms vary in their splitting strategies, architecture and learning mechanisms but some notable examples include:

1) Classification And Regression Tree (CART): Given a pair of inputs $\boldsymbol{X}$ and responses $\boldsymbol{Y}$, the leaves in a trained CART [26] specify a non-overlapping partitioning of $\{\boldsymbol{X}, \boldsymbol{Y}\}$ with each pair of values belonging to a single leaf. The data pairs in different leaves are considered similar in terms of the prediction task, and hence one trains simple local models (i.e. such as linear or constant) for each subset of data sharing a leaf. The prediction of the CART for new data $\boldsymbol{x}_{T+1}$ depends on the leaf to which it would be assigned. To construct a CART, one needs to specify a measure for ranking how appropriate successive splits are. The most common measures for this are *gini impurity* and *entropy*.

Assuming the responses are categorical and the fraction of items in class k is denoted by π_k, the gini impurity for each branch is $1 - \sum_{k=1}^{K} \pi_k^2$; the weighted sum of the gini impurities for both branches resulting from a candidate split are computed. Then, the same procedure is repeated for all candidate splits and the one with lowest sum of gini impurities is selected. Using the entropy to measure the quality of splits results in replacing $1 - \sum_{k=1}^{K} \pi_k^2$ with $-\sum_{k=1}^{2} \pi_k \log_2(\pi_k)$. One starts with all the data at the root and at each iteration, a number of single feature-splits are considered, partitioning the data into two[5] further branches. For example, a branch with all the data associated with feature value $x_{p\cdot} > h$ for an arbitrary index p, and another branch with all data for which $x_{p\cdot} \leq h$. For each resulting branch, the same procedure is repeated recursively, until input constraints on the tree depth, leaf size or accuracy improvement are satisfied. Note that a considerable restriction of CARTs is the limited amount of joint effects between variables captured as the process is done in a greedy fashion.

2) Multivariate and adaptive regression spline (MARS): MARS [27] is a non-parametric technique for fitting piecewise linear regression based on a form of recursive partitioning. MARS assumes that one can predict responses $\boldsymbol{Y}$ with a model of the form:

$$f(\boldsymbol{X}) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{X}) \tag{11}$$

where $h_m(\boldsymbol{X})$ are basis functions from the set $\{(\boldsymbol{X}_j - t)_+, (t - \boldsymbol{X}_j)_+\}$ with $t \in \{x_j^{(1)}, \ldots, x_j^{(N)}\}$ for any feature $j = 1, \ldots, P$. Note that the CART model can be represented in this framework using the basis functions $I(\boldsymbol{X}_j > h)$ and $I(\boldsymbol{X}_j \leq h)$. The first stage of the MARS algorithm performs a *forward pass*, which starts with a constant model $f^{(1)} = \beta_0$ and iteratively builds up predictive power by adding the basis $\beta_1(\boldsymbol{X}_j - t) + \beta_2(t - \boldsymbol{X}_j)$, which decreases most the training error. The procedure is repeated until some preset number of bases has been reached. After the forward pass, the inferred model is large and likely to overfit the data. Therefore, a *backward pass* which prunes the number of basis functions used by penalizing the model complexity while keeping the highest *generalized cross-validation score* (GCV) [28], defined as the: GSV = RSS / (N(1-(effective number of parameters)/N^2)). The effective number of parameters is the $(\text{number of MARS terms}) + \lambda \times \frac{(\text{number of MARS terms}-1)}{2}$.

3) Ensembles of trees: Despite the computational convenience and interpretable structure of decision trees, single trees often overfit the data and lead to poor empirical performance in realistic conditions. This challenge has motivated ensemble frameworks for training multitude of decision trees and combining their predictions into a single model outcome [29], [30]. Whereas complex decision trees minimize the bias, they lead to large variance when evaluated on test data. In contrast, tree ensembles tend to reduce the variance, at the expense of loss of interpretability and potentially a small increase in the bias.

4) Random forests: Most random forest implementations train an ensemble of CART trees [31]. Assume one wishes to train a random forest of B trees. Given pairs of $\{\boldsymbol{X}, \boldsymbol{Y}\}$, you can use *bootstrapping*[6] [32] to generate B training sets $\{\boldsymbol{X}^{(1)}, \boldsymbol{Y}^{(1)}\}, \ldots, \{\boldsymbol{X}^{(B)}, \boldsymbol{Y}^{(B)}\}$ with replacement where each set $\{\boldsymbol{X}^{(b)}, \boldsymbol{Y}^{(b)}\}$ has N samples. The first *bagging* (bootstrap aggregating) procedure proceeds with training B of the trees independently on the generated data. However, this strategy induces correlation between the different trees in the ensemble since important predictors will be selected in many of the B trees. To address this issue, random forest involve an additional step which is to select only a random subset of the features in $\boldsymbol{X}$ - a process known as *feature bagging* [33]. Once the B trees have been trained, the predicted outcome is most commonly obtained either by averaging the predictions $\widehat{\boldsymbol{Y}}^{(1)}, \ldots, \widehat{\boldsymbol{Y}}^{(B)}$ for regression tasks or by selecting the most common class prediction for classification problems.

5) Gradient boosted trees and XGBoost: Gradient boosted forests (common variation being the *Extreme Gradient Boosting* (XGBoost)) are known to typically outperform classical random forests [25]. Both approaches use the same model of ensembles of CARTs, but vary in the way the trees are trained. Whereas in random forests trees are trained in parallel, after feature bagging, in gradient boosted forest,~~s~~ trees are trained one at a time and subsequently a new tree is added which best optimizes our objective; for manageable computation the scheme is greedy, so the parameters of the tree learned at step t are kept fixed for the remaining steps. The objective which gradient boosted forests optimize takes the general form:

$$\text{obj}_{(t)} = \sum_{n=1}^{N} \mathcal{L}\left(\boldsymbol{y}^{(n)}, \widehat{\boldsymbol{y}}_{(t-1)}^{(n)} + f_{(t)}(\boldsymbol{x}^{(n)})\right) + \Omega\left(f_{(t)}\left(\boldsymbol{x}^{(n)}\right)\right) \tag{12}$$

[5] Most commonly used CARTs offer binary splits, allowing to condition on the same feature at multiple levels of the tree.

[6] A process whereby new datasets are generated by randomly sampling subsets from the data.

where $\mathcal{L}(\cdot)$ is a loss function of choice (such as the mean square error), $f_{(t)}\left(\boldsymbol{x}^{(n)}\right)$ is the function represented by the CART trees for $t = 1, \ldots, B$ and $\Omega\left(f_{(t)}\left(x^{(n)}\right)\right)$ is a regularizer balancing the complexity of individual trees. The second order Taylor expansion for the loss in Eq. (12), provides the following objective for new trees:

$$\begin{aligned}\widehat{f}_{(t)}\left(\boldsymbol{x}^{(n)}\right) = \underset{f_{(t)}}{\arg\min} \sum_{n=1}^{N} \Big[g^{(n)} f_{(t)}\left(\boldsymbol{x}^{(n)}\right) + \\ + h^{(n)} f_{(t)}^{2}\left(\boldsymbol{x}^{(n)}\right)\Big] + \Omega\left(f_{(t)}\left(\boldsymbol{x}^{(n)}\right)\right)\end{aligned} \tag{13}$$

where $g^{(n)}$ and $h^{(n)}$ are the first and second derivative of the loss $\mathcal{L}\left(\boldsymbol{y}^{(n)}, \widehat{\boldsymbol{y}}_{(t-1)}^{(n)}\right)$ with respect to $\widehat{\boldsymbol{y}}_{(t-1)}^{(n)}$. Having optimized one tree at a time, one can represent the function as a weighted aggregated contribution of the trees, where the weights correspond to their scores (e.g., gini impurity).

6) Practical limitations: Variants of random forests algorithms have shown state-of-the-art performance across a wide range of supervised learning tasks. However, well known weakness are their *lack of interpretability*, *global structure* and *expensive batch training*:

- In many applications, the goal is not simply to maximize empirical performance on the available sample of data, but also to infer key input features and their structural relationship to the output. This is particularly important to ensure the algorithm does not leverage sample-specific futile causal effects or when one wishes to incorporate parametric assumptions about changes in the test data distribution [34].
- Unlike parametric models and most kernel methods, regression trees hardly allow us to incorporate global assumptions about the distribution of the different features. As such, they can end up learning an overly complex representation of the data, making the algorithms slow and sensitive.
- The most popular random forest variants operate on data batches, making them inappropriate for many photonics application which require online methods. There are online random forest methods [35], which grow trees incrementally, but suffer from inefficient memory cost and require substantially more training data then their batch counterpart. Alternatively, instead of constructing an ensemble of heuristic CARTs, one can use *Mondrian processes* [36] (self-consistent hard-partitioning stochastic process) to specify distributions over tree structures. Sampling a collection of independent random trees from a Mondrian process, one can obtain a variant of random forest known as the Mondrian forests [37], having the convenient property that their online distribution is the same as that of batch Mondrian forests potentially alleviating the batch training challenge.

IV. Probabilistic modelling

Despite the heuristic origin of many of the popular pattern recognition algorithms, often the estimation problem can be described as finding the joint distribution over all unknown quantities, from which conditional and marginal probabilities can be estimated. In classical regression tasks, the RVs to be considered would be the regression (or kernel) parameters, the noise and other factors. Specifying a probabilistic model forces one to make explicit modelling assumptions; it also allows one to test the statistical significance of the different assumptions, querying the probability of all unknown quantities in different scenarios. The probabilistic view of machine learning is partially motivated by the thesis that human intelligence is not deterministic, but relies on decision-making under uncertainty [38].

As a practical example of probabilistic reasoning, consider the problem of modulation format indentificaiton for square M-quadrature amplitude modulation ~~automatic modulation formats identification~~, in the presence of optical channel impairments. The input for automatic modulation might be amplitude histograms (i.e. after analog-to-digital conversion and chromatic dispersion compensation at the receiver) which can be written as a vector of D bins $\boldsymbol{x} = (x_1, \ldots, x_D)^T \in \mathcal{R}^D$. The output is a discrete decision variable, $y \in \{C_1, \ldots, C_M\}$ with C_m, for $m = 1, \ldots, M$, indicating the quadrature phase-shift keying formats ~~different modulation formats~~ (i.e. 16-QAM or 64-QAM ~~10 Gbps NRZ-OOK, 40 Gbps NRZ-DQPSK etc.~~). The RVs in the problem are $(\boldsymbol{x}, y)$ and the joint distribution is:

$$p(\boldsymbol{x}, y) = p(x_1, \ldots, x_D, y) \tag{14}$$

In this example, one considers the primary probabilistic modelling objective to be the estimation of the joint probability from Eq. (14), or at least finding maximum likelihood estimate of the joint probability and model parameters, leading to the most likely estimates. Having estimated $p(\boldsymbol{x}, y)$, one can directly address standard machine learning objective such as making predictions about y from $\boldsymbol{x}$, evaluating the uncertainty associated with such a prediction, and the conditions in which the prediction is reliable. Without making assumptions about the form of Eq. (14), the estimation is rarely tractable so typically we make some assumptions about the conditional relations between $\boldsymbol{x}$, y and other unobserved *latent variables*, which help in simplifying the problem. In modulation format identification, one may assume latent *multimodality* in the map between $\boldsymbol{x}$ and y depending on the different channel impairments, independence between the different bins $x_1, \ldots, x_D$, or place higher or lower importance on certain bins as predictors (i.e. the role of the distribution matcher in probabilistic constellation). The following section will provide a basic introduction into probabilistic graphical models and how they are used to tackle practical machine learning challenges.

A. Probabilistic graphical models

Estimation of the joint likelihood for all unknown variables is infeasible for most practical machine learning problems due to the *curse of dimensionality* where the number of data needed for the estimation grows exponentially with the system's dimension. This motivates us to specify explicitly how RVs are conditioned on each other, in the form of a *probabilistic graphical model* (PGM). A PGM describes graphically the

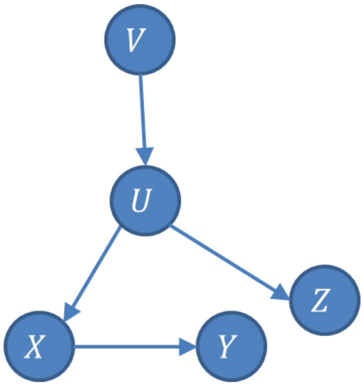

Fig. 5. Illustration of a probabilistic graphical model which specifies the relationship between five RVs: U, V, X, Y, Z.

conditioning relationships between the RVs in a problem as edges in a graph where each RV is represented by a node and arrows (i.e edges for directed graphical models) depicting the direction of the conditioning.

Consider the model of Fig. 5 which specifies five RVs U, V, X, Y, Z with the joint distribution $p\left(u, v, x, y, z\right)$. Using the *chain rule*, the joint probability can be written as:

$$p\left(u, v, x, y, z\right) = p\left(y|u, v, x, z\right) \times$$
$$\times p\left(z|u, v, x\right) p\left(x|u, v\right) p\left(u|v\right) p\left(v\right)$$

However, using the conditional relationships represented in the PGM of Fig. 5, the joint probability simplifies to:

$$p\left(u, v, x, y, z\right) = p\left(y|x\right) p\left(z|u\right) p\left(x|u\right) p\left(u|v\right) p\left(v\right)$$

Now, one only needs to estimate a few simpler conditional probabilities, which is more tractable than estimating the full joint distribution. Therefore, PGMs allow one to encode complex dependencies between RVs while mitigating the curse of dimensionality.

Assume that all the observed data are captured by the RVs $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)}$ and the unknown parameters are modelled using a set of parameters $\boldsymbol{\Theta}$ which themselves are RVs. The process of *training* can be seen as finding the best parameters $\boldsymbol{\Theta}$ which fit the realizations $\mathcal{D} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\}$ of the RVs $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(N)}$, which are the observed data samples, given some functional relationship between $\boldsymbol{\Theta}$ and $\boldsymbol{X}$ specified by a PGM. The *likelihood function* is a measure of goodness of fit for a set of parameters $\boldsymbol{\Theta}$ and is defined as the conditional probability: $\mathcal{L}(\boldsymbol{\Theta} = \boldsymbol{\theta}) = p\left(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}|\boldsymbol{\theta}\right)$ for $\boldsymbol{\theta}$ and $\boldsymbol{x}$ being realizations of the corresponding RVs. Maximization and inference of $\mathcal{L}(\boldsymbol{\theta})$ is typically difficult and a very common simplification is the assumption that the individual examples are independent and identically distributed (i.i.d.) given the model parameters. The joint likelihood can then be factorized to take the form:

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{n=1}^{N} p\left(\boldsymbol{x}^{(n)}|\theta\right) \tag{15}$$

Finding the best parameters $\boldsymbol{\theta}$ is often done using *maximum likelihood* (ML) or *maximum-a-posteriori* (MAP) principles. The ML estimator for parameters $\boldsymbol{\theta}$ is the solution of:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}\right) = p\left(D|\boldsymbol{\theta}\right) \ , \tag{16}$$

while the MAP estimator for $\boldsymbol{\theta}$ also takes into account the prior belief about the parameters, solving:

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}} p\left(\boldsymbol{\theta}|\mathcal{D}\right) \ . \tag{17}$$

Differentiating the negative log of Eq. (15), which is typically easier to deal with since products over data become summations, leads to closed form solutions only for a limited family of simple distributional models $p(\boldsymbol{x}^{(n)}|\boldsymbol{\theta})$, such as a single Gaussian or Poisson distribution. Even after exploiting conditional independence, for most practical PGMs the joint probability of interest $p\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ cannot be easily estimated or maximized. It is possible to draw asymptotically unbiased samples of the complete joint probability $p\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ using advanced numerical methods such as variants of Markov Chain Monte Carlo (MCMC) algorithms. An introduction to MCMC techniques can be found in another tutorial [39].

B. Mixture models

Many complex densities can be approximated using a superposition of simple building blocks such as exponential family distributions. The combined probability of M components takes the form:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{j=1}^{M} p(j) p(\boldsymbol{x}^{(n)}|\boldsymbol{\theta}_j) \ , \tag{18}$$

where $p(j)$ denotes mixing probability associated with component j with $\sum_j p(j) = 1$ and $\boldsymbol{\theta}_j$ specifies the component parameters. The difficulty is in determining optimal set of parameters $\boldsymbol{\theta}_j$, jointly for all j components, which maximize the likelihood. This cannot be trivially done by maximizing Eq. (18) and requires a numerical iterative procedure. A good approximate and locally optimal solution is provided by the Expectation-Maximization algorithm [40], which first calculates the *responsibility* of component j for data $\boldsymbol{x}^{(n)}$ - $p(j|\boldsymbol{x}^{(n)})$ (the Expectation step):

$$p(j|x^{(n)}) = \frac{p(j) \ p(\boldsymbol{x}^{(n)}|\boldsymbol{\theta}_j)}{\sum_k p(k) \ p\left(\boldsymbol{x}^{(n)}|\boldsymbol{\theta}_k\right)} \ , \tag{19}$$

where in the Gaussian mixture case $p(\boldsymbol{x}^{(n)}|\boldsymbol{\theta}_j)$ is a single Gaussian probability and $\boldsymbol{\theta}_j$ are the means and variances associated with a component. The expectation step is followed by the maximization of the parameters $\boldsymbol{\theta}_j$ and $p(j)$ for all $j = 1, \ldots, M$ (Maximization step). The Expectation step facilitates the calculation of the Maximization step and vice versa, and the iterative process has been proved to converge to a likelihood minimum value (not necessarily the global one). In the extreme case where only one component can be associated to an example, the Expectation-Maximization algorithm can be reduces to the K-means algorithm [41]. Mixture models have been used as building blocks for complex

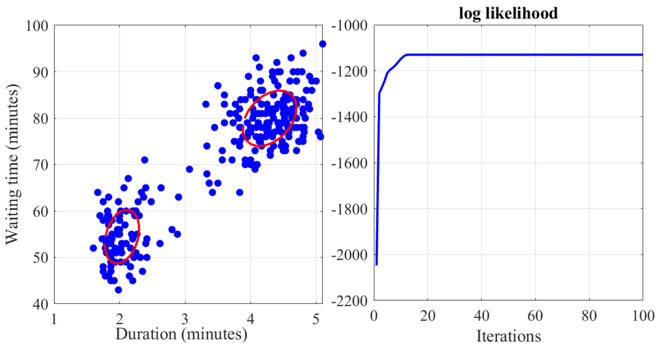

Fig. 6. Modeling the Old Faithful eruption data (duration and waiting time). Left - final position and variance of the two Gaussians. Right - the log-likelihood value increasing with each iteration of the Expectation-Maximization algorithm.

predictive algorithms, via mixture of experts [12]. They have also been extended to variants allowing for flexible inference of the *number* of components [42], [43] and deep generative models where the shape of the components is described by neural networks [44].

An example for the way the Expectation-Maximization algorithm works is presented in Fig. 6, modeling waiting time between eruptions (y axis) and duration of the eruption (x axis) of the Old Faithful geyser in Yellowstone National Park[7]. A Gaussian mixture model with two components has been used to model the data. Starting from random location and parameters for the two components, one can see how the log-likelihood value is increasing with each iteration of the Expectation-Maximization algorithm (right), ending with a good separation of the two components and a reasonable representation of the data.

C. Other flexible distributions

The presented probabilistic models show how complex distributions can be represented as a combination of simple patterns, described in some parametric form, such as Gaussian density. However, it is worth pointing out that a large family of complex densities can be constructed using simple recursive transformations of some parametric density. The transformations can be guided by some algebraic knowledge one has about the random variables summarizing the data, or using some generic recursive principles. The latter is the motivation behind the growing field of *normalizing flows* [45] that are often used to construct arbitrary densities which serve as building blocks in deep generative models, but in theory can also be used to construct more flexible interpretable probabilistic models. For an introduction to the concept underpinning normalizing flows, we refer the reader to [46].

D. Gaussian processes

One of the difficulties in the Bayesian approach is choosing the right prior, hyperprior, and their most appropriate parameters. When applied to parameterized models, one ends up with a posterior distribution of parameter values which can be used to infer variable states or values (e.g., regression or classification). The choice of prior has a fundamental impact on the posterior and the functions generated by the combination of data (through the likelihood) and prior. Perhaps a more direct approach would be to select the properties of the *functions* we expect to obtain. Additionally, we saw that Generalized Linear Models with the Gaussian noise-model assumption and Gaussian prior for the parameters, result in a Gaussian posterior parameter distribution where computing with the posterior is easy but may require a huge number of basis functions to cover a high dimensional space. Gaussian processes offer an alternative approach for tractable function interpolation based on the data and the expected function properties, e.g., being smooth or rugged.

Given N i.i.d. data points $\boldsymbol{x}$ and their corresponding outputs observations t, $\mathcal{D} = \left\{(\boldsymbol{x}^{(1)}, t^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, t^{(N)})\right\}$, and a noise model $p(t|y)$ for the underlying uncorrupted output y, one would like to infer the y^* value for a given input $\boldsymbol{x}^*$. Writing $\mathbf{y} = (y^{(1)}, \ldots, y^{(N)})$, $\mathbf{x} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)})$ and $\mathbf{t} = (t^{(1)}, \ldots, t^{(N)})$, the corresponding probability becomes:

$$\begin{aligned} p(y^*|\boldsymbol{x}^*, D) &= \int p(y^*, \mathbf{y}|\boldsymbol{x}^*, \mathbf{x}, \mathbf{t}) d\mathbf{y} \\ &= \int p(\mathbf{t}|y^*, \mathbf{y}, x^*, \mathbf{x}) p(y^*, \mathbf{y}|\boldsymbol{x}^*, \mathbf{x}) d\boldsymbol{y} \\ &\propto \int \underbrace{p(\mathbf{t}|\mathbf{y})}_{likelihood} \underbrace{p(y^*, \mathbf{y}|\boldsymbol{x}^*, \mathbf{x})}_{prior} d\mathbf{y}\ . \end{aligned} \tag{20}$$

The Gaussian noise assumption gives the likelihood for the targets $\mathbf{t}$ of the form $\mathcal{N}\left(\mathbf{y}, \sigma^2\mathbf{I}\right)$. The prior is imposed directly on the function space, determining the smoothness of the function. *Defining* the prior $p(\mathbf{y}, y^*|\mathbf{x}, \boldsymbol{x}^*)$ to be Gaussian (zero mean for convenience):

$$p(\mathbf{y}, y^*|\boldsymbol{x}, x^*) = \mathcal{N}\left(0, K(\mathbf{x}, x^*)\right) \tag{21}$$

one obtains a *Gaussian* joint posterior over $\mathbf{y}, y^*$ so that calculating the marginal prediction for y^* is straightforward. The properties of the obtained function rely on the properties of the matrix $\boldsymbol{K}$; the elements of $K_{ij} = c(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ represent the covariance of the y values given the corresponding $\boldsymbol{x}$ values. Two commonly used covariance matrices are the Ornstein-Uhlenbeck covariance $c(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \alpha \exp\left(-\lambda|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}|\right)$ for rugged functions, where λ relates to the length scale in x space and α a range variable, and a Gaussian covariance function $c(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \alpha \exp\left(-\lambda\left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right)^2\right)$ for smooth functions, where the parameters play similar roles. Other types of covariance matrices have also been considered for dedicated tasks [47]. The relation between Gaussian processes, neural networks, deep learning machines [48], [49], radial basis functions and generalized linear models are well understood [47], [50].

Having chosen a covariance function one calculates the prior given data

$$p(\mathbf{y}_+) \propto \exp\left[-\frac{1}{2}\mathbf{y}_+^{\mathrm{T}}\boldsymbol{K}_+^{-1}\mathbf{y}_+\right] \tag{22}$$

[7] Figures were plotted using the BRML Matlab toolbox obtained from http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.Software

where $\mathbf{y}_+^{\mathrm{T}} = \left[y(\boldsymbol{x}^1), y(\boldsymbol{x}^{(2)}), \ldots y(\boldsymbol{x}^{(N)}), y(\boldsymbol{x}^*)\right]$. Separating the new data point $\boldsymbol{x}^*$ from the data, one can write

$$\boldsymbol{K}_+ = \left[\begin{array}{cc} \boldsymbol{K}_{xx} & \boldsymbol{K}_{xx^*} \\ \boldsymbol{K}_{xx^*}^{\mathrm{T}} & \boldsymbol{K}_{x^*x^*} \end{array}\right]$$

where

$$\begin{aligned} (\boldsymbol{K}_{xx})_{ij} &= c(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) && (23) \\ \boldsymbol{K}_{xx^*}^{\mathrm{T}} &= \left[c(\boldsymbol{x}^*, \boldsymbol{x}^{(1)}), c(\boldsymbol{x}^*, \boldsymbol{x}^{(2)}), \ldots c(\boldsymbol{x}^*, \boldsymbol{x}^{(N)})\right] \\ \boldsymbol{K}_{x^*x^*} &= c(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{aligned}$$

The posterior becomes

$$\begin{aligned} p(\mathbf{y}_+|\mathcal{D}) &\propto p(\mathbf{y}_+)p(\mathcal{D}|\mathbf{y}_+) && (24) \\ &\propto \exp\left[-\frac{1}{2}\mathbf{y}_+^{\mathrm{T}}K_+^{-1}\mathbf{y}_+ - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(t_i - y_i)^2\right] \end{aligned}$$

and the calculation of the corresponding output probability follows straightforwardly $p(y^*|\boldsymbol{x}^*, D) = \mathcal{N}(\boldsymbol{m}, \Sigma)$ where the mean and covariance are given by

$$\begin{aligned} &\boldsymbol{m} = K_{xx^*}\left(K_{xx} + \sigma^2\right)^{-1}\boldsymbol{t} && (25) \\ &\Sigma = K_{x^*x^*} - K_{xx^*}^{T}\left(K_{xx} + \sigma^2\mathbf{I}\right)^{-1}K_{xx^*}\ . \end{aligned}$$

Gaussian processes provide a simple, principled and intuitive approach for regression, where the smoothness and other properties of the resulting functions are determined through the covariance function elements which constitute the Gaussian prior matrix. The covariance matrix can incorporate periodicity, variability depending on input value (heteroscedastic Gaussian processes) and physical model insights [47]. While Gaussian processes have been applied for classification tasks [51], they are naturally more suitable for regression problems.

To demonstrate the way Gaussian processes operate we have used 40 data points generated by a noisy sin function using the Gaussian-like and Ornstein-Uhlenbeck covariances, where the assumed noise model is of zero mean and variance $\sigma^2 = 0.04$. The results are shown in Fig. 7 for both Gaussian-like (top) and Ornstein-Uhlenbeck (bottom) covariances, where data are marked by crosses, mean value and standard error bars are represented by red and blue solid lines, respectively. Clearly the Gaussian-like covariance elements lead to a smoother function. Note that the error bars are larger where no data is present; Gaussian processes can be perceived as a generalized interpolation method.

The main drawback in the application of Gaussian processes is the need to invert the large matrix that includes all examples, as in Eq. (25), which scales cubically with the number of examples, becoming computationally demanding for large data sets. Reducing the algorithmic complexity using the sparsity assumption for approximate inference [52], via methods such as pseudo-inputs [53] and variational inference [54] reduces the representational power of Gaussian processes, especially when the number of training data is large. Sparsifying the covariance matrix, through setting an effective window or by diluting the matrix may also reduce the computational cost as well as an online sample-by-sample inversion [50].

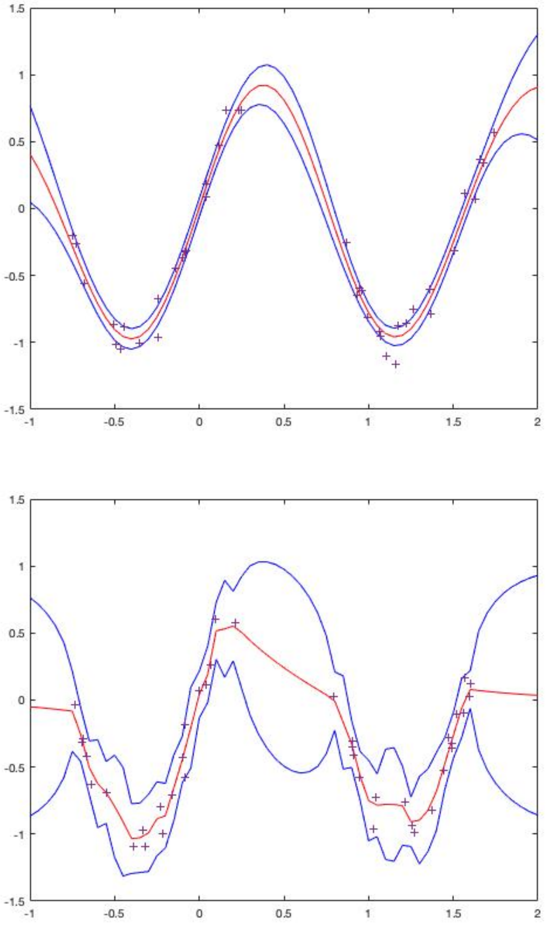

Fig. 7. Data generated by a noisy sin function (40 points, random noise) modeled by a Gaussian process using Gaussian-like (top) and Ornstein-Uhlenbeck (bottom) covariances. Data are marked by crosses, mean value and standard error bars are represented by red and blue solid lines, respectively.

E. Mean field and message passing algorithms

While probabilistic inference has been highly successful in addressing many inference and optimization problems, it is generally intractable due to the computational complexity required for obtaining exact solutions and principled approximation methods can be helpful for obtaining solutions in polynomial time. Mean field (MF) methods, originated in statistical physics and provide a range of approximation tools. They assume that the inference problem can be described in the form of minimizing a cost function (Hamiltonian in the statistical physics terminology [55]) $\mathcal{H}(\boldsymbol{x}|\mathcal{D})$ with dynamical variables $\boldsymbol{x}$, to be inferred, and predetermined (fixed) RVs or observations.

The approach is applicable to a variety of systems including both binary and continuous variables. However, for simplicity we will restrict the presentation to binary variables $x_1, \ldots, x_P \in [-1, +1]$. The probability of finding the system in any state $\boldsymbol{x}$ takes the form

$$P(\boldsymbol{x}|\mathcal{D}, \beta) = \frac{e^{-\beta\mathcal{H}(\boldsymbol{x}|\mathcal{D})}}{\mathcal{Z}(\mathcal{D}, \beta)} \qquad (26)$$

with β being a control parameter (inverse temperature in physics, the higher β is the more deterministic the state is) and the normalization $\mathcal{Z}(\mathcal{D},\beta) = \mathrm{Tr}_{\boldsymbol{x}} e^{-\beta\mathcal{H}(\boldsymbol{x}|\mathcal{D})}$ is termed the partition function, where the trace represents a summation over all $\boldsymbol{x}$ values. The exact inference of variable states would be the MAP estimator $\widehat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}|\mathcal{D},\beta)$, which is intractable as the search increases exponentially with the number of variables ~~$\mathcal{N}$~~$\mathcal{P}$. An equally difficult alternative is the Marginal Posterior Maximizer

$$\widehat{x}_i = \arg\max_{x_i} P(x_i|\mathcal{D},\beta), \tag{27}$$

which converts the maximization to a local operation but requires a global marginalization. The crux behind mean field methods is to address the global problem by introducing localized, distributive and scaleable operations.

The family of MF approximations represents one of the most promising approaches for scalable distributive inference and optimization. The spirit of the MF is simple; to approximate a true intractable distribution with a tractable one, factorized with respect to dynamical variables. Since the factorized model can usually be calculated quite easily the required computation becomes significantly less demanding than that of sampling techniques. MF approaches were developed within the physics community and include a large number of variations, depending on the objectives of the calculation and properties of the system examined.
The simplest approximation is to replace the interaction of each variable x_i with all others by a single effective field h_i, such that

$$P(\boldsymbol{x}|\mathcal{D},\beta) \propto \prod_{i=1}^{P} e^{-\beta x_i h_i} \tag{28}$$

which gives rise to a set of localized equations that can be solved iteratively. This approximation can also be viewed as a variational approach where one aims to minimize the distance between exact and approximate distributions [50].

The shortcoming of naive MF methods is that they ignore the reciprocal interaction between variables. In densely connected systems, where each variable interacts weakly with most of the other variables in the system, simple correction terms have been added to compensate for the reciprocal interaction [56]. However, many of the inference and optimization problems to be solved involve sparsely connected systems where individual variables interact with a small number of variables, with respect to the system size. These include many network based inference tasks, combinatorial optimization and information theoretical problems.

Message-passing or belief propagation [57] is a commonly used method of reduced computational complexity for~~aimed at reducing the computational complexity of~~ inferring variable values on sparsely interacting variables (e.g., on graphs). It has been developed independently in physics [58], computer science [59] and information theory [60]. Being directly linked to established statistical physics methodology [61], the approach also facilitates the analysis of the system's behavior at a macroscopic level, identifying the emergence of collective behavior. At the heart of the approach is the *assumption* that variable values are mostly affected by those of their immediate neighbours due to the sparsity of the interactions and of the negligible role played by long-range correlations. The method is therefore based on passing messages - conditional probabilities - iteratively between interacting variables until they stabilize and then inferring the variable values from the converged messages.

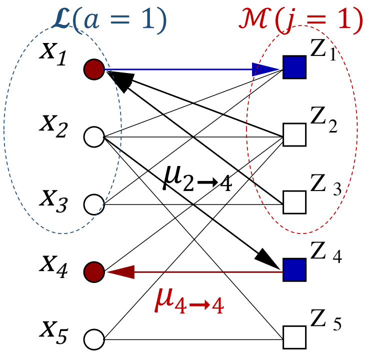

Fig. 8. An example bipartite graph representation, where variable nodes $\boldsymbol{x}$ appear on the left and factor nodes $\boldsymbol{Z}$, representing the interaction between them, on the right. Lines represent interaction between variables at the factor nodes; arrows are example messages from variable to factor ($\mu_{x_j \to Z_a}$) and vice versa ($\mu_{Z_a \to x_j}$). The neighboring nodes to factor a are marked by $\mathcal{L}(a)$ (in the figure for $a = 1$), while the neighboring factors to node j are marked by $\mathcal{M}(j)$ (in the figure for $j = 1$).

There are different ways to explain message passing methods [55], [57], through approximate minimization of the Bethe free energy using a variational approach, modeling the joint probability of the system using factorized interactions between variable groups and others [57]. The explanation provided here relies on mapping the problem onto a bipartite graph, where variable nodes appear on the left and factor nodes, representing the interaction between them, on the right, as shown in Fig. 8. Factors can represent validating a rule (e.g., two neighboring nodes having different colors in the graph coloring problem), observations that depend on the interacting variable values or some probability of the interacting variables to be in a given state. The factors variable j interacts with are denoted by $\mathcal{M}(j)$ and the set of variables interacting through factor a as $\mathcal{L}(a)$. The messages represent conditional probabilities of variable S_j assuming a certain state given all the factors in $\mathcal{M}(j)$ except factor a - $\mu_{x_j \to Z_a}$ and the probability of factor Z_a given that variable x_j is in some state

- $\mu_{Z_a \to x_j}$. This gives rise to a closed set of equations

$$\begin{aligned}
\mu_{x_j \to Z_a} &= P(x_j \mid \{Z_b : b \in \mathcal{M}(j)/a\}) \\
&= \alpha_{aj}\, p(x_j) \prod_{b \in \mathcal{M}(j) \setminus a} \mu_{Z_b \to x_j} \\
\mu_{Z_a \to x_j} &= P(Z_a | x_j) \\
&= \sum_{\{x_i : i \in \mathcal{L}(a) \setminus j\}} P(Z_a \mid x_j, \{x_i : i \in \mathcal{L}(a) \setminus j\}) \\
&\times \prod_{i \in \mathcal{L}(a) \setminus j} \mu_{x_i \to Z_a}
\end{aligned}$$

that can be iterated until the messages stop changing. The coefficient α_{aj} is a normalization factor obtained by summing over all possible x_j values. The derivation is based on simple identities and hinges on the assumption the due to the sparse nature of the interactions, the joint probabilities $P(\{Z_b : b \in \mathcal{M}(j)/a\} \mid x_j)$ and $P(\{x_i : i \in \mathcal{L}(a) \setminus j\} \mid \{Z_b : b \in \mathcal{M}(j)/a\}$ can be factorised. *This is exact on trees and works well on locally tree-like graphs.*

Once the messages have converged, one can calculate the marginal values for the variables

$$P(x_j | \{Z_b : b \in \mathcal{M}(j)\}) = \alpha_j\, p(x_j) \prod_{b \in \mathcal{M}(j)} \mu_{Z_b \to x_j} \ ,$$

α_j being a normalization coefficient, to infer variable values.

Many variants of message passing methods exist, that deal with densely connected graphs [62], that provide an even simpler approximation (Approximate Message Passing) [63] and that can account for short loops [64], [65] and the emergence of long range correlations [66].

Message-passing methods have been useful in solving hard combinatorial problems [55], decoding in low-density error correcting codes [67], [68], compressed sensing [69], resource allocation [70] and routing [71]–[73]. Of particular interest to optical communication networks is addressing edge-disjoint routing [74] and resource allocation. They work effectively and scale well, typically linearly or quadratically, with respect to the number of free variables, but break down in the hard regime where long-range correlations between variables are formed.

V. Computational complexity

When navigating through the modelling choices presented in the sections above, one is often driven by the limited computational resources and scale of the problem, rather then seeking the most appropriate formalism. Whereas there are some notable cases in which a-priori computational complexity figures are very difficult to estimate, we list few examples from each family of algorithms. As a benchmark, note that, in principle, the task of training neural networks is NP-hard [75] and requires exponential time with respect to the number of free variables. Nevertheless, for obtaining an acceptable approximation, one typically considers the computational time complexity of neural networks trained with standard back-propagation as the product of the complexity per epoch and the undetermined number of epochs: the time complexity for two layer multilayer perceptron is therefore:

$$\begin{aligned}
\mathcal{O}(&\text{number of epochs} \cdot N \cdot \text{number of nodes in layer 1} \cdot \\
&\cdot \text{number of nodes in layer 2})
\end{aligned} \tag{29}$$

Unlike neural networks, parametric methods and kernel methods are commonly trained using variety of inference paradigms which would largely determine their computational complexity. Ordinary linear regression can be trained in linear complexity in terms of the number of data points N with complexity $\mathcal{O}\left(N \cdot P^2 + P^3\right)$. The complexity of linear models can grows exponentially although if we consider pairwise interactions linear models the complexity becomes: $\mathcal{O}\left(N^2 \cdot P^2 + P^3\right)$. Using the kernel trick [76] showed how smarter training could limit the time complexity to $\mathcal{O}\left(N \cdot P^2 + P^3\right)$, even for pairwise interaction models. The training of piecewise linear models introduces a non-convex optimization problem which whose complexity is difficult to estimate, as it typically involves approximate inference strategies such as the expectation maximization algorithms introduced in Sec. IV. Training kernel regression methods as well as probabilistic extensions such as Gaussian processes involves prohibitive cubic complexity $\mathcal{O}\left(N^3\right)$ when approached naively. This has historically slowed down the adoption of many methods from these families. However, inducing variables [77] and sparse Gaussian process approximations [78], [79] can be used to reduce this complexity to $\mathcal{O}\left(N \cdot M^2\right)$ with M denoting the number of the inducing variables. For more rigid but scalable supervised methods, we have K-nearest neighbour regression and classifiers with complexity constrained by the number of neighbours with $\mathcal{O}\left(N \cdot K\right)$. Maximum margin methods such as the support vector machines exhibit minimum time complexity of $\mathcal{O}\left(N^2\right)$. Decision trees introduced in Section III-C are trained with efficient gradient descent algorithms making their complexity proportionate to $\mathcal{O}\left(N \cdot \log(N)\right)$ and the number of trees and their maximum depth: CART trees require $\mathcal{O}\left(\text{depth of tree} \cdot N \cdot \log(N)\right)$, whereas random forests and XGboost can be trained at $\mathcal{O}\left(\text{depth of tree} \cdot \log(N) \cdot \text{number of trees}\right)$. However, since training individual trees in tree ensembles can be distributed efficiently across multiple cores, inference in different random forest algorithms is considered quite scalable. Finally, being mostly applied to sparse graphs, message passing techniques presented in Sec. IV-E, scales linearly with the number of variables $\mathcal{O}\left(P\right)$ although the number of iterations required for convergence is unspecified, but is typically much smaller than P.

VI. Visual Informatics and dimensionality reduction

The ability to appropriately represent information from high dimensional structures is central to many of the success stories in statistical machine learning. Structured data objects such as spectograms, images or voice recordings can be efficiently represented in lower dimensional spaces, with minimal loss of information provided that we utilize some defining properties of these input types. For instance, that pixel intensities are predictive of their immediate neighbours and periodic signals are encoded in the first few harmonics.

Dimensionality reduction unifies a wide set of algorithms for utilizing specific properties of the data to map it onto a lower dimensional and simpler representation. They can then form a part of the: (1) exploratory analysis stage by facilitating complex data visualization; (2) simplify the classification or clustering stages by reducing the dimensionality of the problem, or (3) be utilized for noise removal (i.e. including removal of outliers) or for removing data redundancy. Despite the plethora of algorithms and heuristics which are in this family, many statistical machine learning tools can be stratified into one of the following categories: (1) interpretable and generative dimensionality reduction; (2) manifold embedding methods and (3) neural network methods.

A. Generative models in visual informatics

In this broad class of methods, explicit assumptions about the underlying data structure are made, as well as, the functional form of the mapping $f : \mathcal{R}^D \to \mathcal{R}^P$ that relates the lower dimensional latent (i.e. P-dimensional) space with the original D-dimensional data space. The detailed modelling assumptions will allow us to formally test different aspects of model selection and infer missing data.

1) Linear dimensionality reduction: Assume a linear map f which transforms input data into scalar latent variables $z = \boldsymbol{y}^T \boldsymbol{w}$ where $\boldsymbol{w} \in \mathcal{R}^D$ is a projection vector. If the input $\boldsymbol{y}$ has variance Σ, one can write the variance of the latent z as:

$$\begin{aligned} \text{var}[z] &= E\left[z^2\right] - E[z]^2 \\ &= \boldsymbol{w}^T E\left[\boldsymbol{y}\boldsymbol{y}^T\right]\boldsymbol{w} - \boldsymbol{w}^T E[\boldsymbol{y}]\, E\left[\boldsymbol{y}^T\right]\boldsymbol{w} = \boldsymbol{w}^T \Sigma \boldsymbol{w} \end{aligned} \tag{30}$$

A common goal is then to find $\boldsymbol{w}$ that maximizes var$[z]$; to make this a well-posed problem we may assume unit length. This can be written formally as the optimization problem:

$$\begin{aligned} &\text{maximize} && \boldsymbol{w}^T \Sigma \boldsymbol{w} \\ &\text{subject to} && \|\boldsymbol{w}\| = 1 \ , \end{aligned} \tag{31}$$

or as maximizing the Lagrangian $L(\boldsymbol{w}, \lambda) = \boldsymbol{w}^T \Sigma \boldsymbol{w} + \lambda\left(\boldsymbol{w}^T \boldsymbol{w} - 1\right)$. The gradient is $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, \lambda) = 2\boldsymbol{\Sigma}\boldsymbol{w} - 2\lambda\boldsymbol{w}$ such that the solution to the optimization problem is: $\boldsymbol{\Sigma}\boldsymbol{w} = \lambda\boldsymbol{w}$. This is just the *eigenvector equation* for the covariance of $\boldsymbol{y}$, so it follows that var$[z] = \lambda \boldsymbol{w}^T \boldsymbol{w} = \lambda$. The maximum-variance solution is obtained by finding the largest eigenvalue and corresponding eigenvector of $\boldsymbol{\Sigma}$. The vector $\boldsymbol{w}$ is known as the first *principal component* (PC) $\boldsymbol{w}_1$ of $\boldsymbol{y}$ and λ_1 the largest eigenvalue. Inductively, a high-dimensional $\boldsymbol{z}$ can be obtained with $\boldsymbol{z} \in \mathcal{R}^D$, if $P = D$ we use projection matrix $\boldsymbol{W} \in \mathcal{R}^{D \times D}$ which in fact does not reduce dimensionality, but only whitens the data (see Fig. 9). In *principal component analysis* (PCA) [80], the columns of $\boldsymbol{W}$ are assumed orthogonal, which is implied when taking the eigenvectors of the covariance matrix as PCs. Each component is a different eigenvector of the sample covariance and components are ranked by their corresponding eigenvalues. The relative size of the eigenvalue reflects the ratio of explained variance by the matching PC. The orthogonality of the PCs means that PCA projections of the input de-couple the features and reduce redundancy in multi-channel signals. An exemplar application of PCA for reducing redundancy in microwave photonics systems is presented in [81]. Common uses of PCA beyond dimensionality reduction are (1) feature whitening and (2) anomaly detection, simple examples of which are displayed in Fig. 9. Given an input data with correlated features $\{\boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)}\}_{n=1}^N$, PCA estimates PCs $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ which project the data onto a new uncorrelated space (i.e. Fig. 9 (a)-(b)). The fact that PCs preserve the direction of the largest variance makes lower dimensional projections naturally suitable for separating statistical outliers (i.e. Fig. 9 (c)-(d)).

So far PCA was derived based on the requirement of covariance structure preservation, omitting any formal probabilistic assumptions. Putting PCA into a more principled probabilistic framework allows for informed model assumptions. Assume data can be efficiently represented using a linear Gaussian latent variable model:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{32}$$

where the observed data is $\boldsymbol{y} \in \mathcal{R}^D$, $\boldsymbol{W} \in \mathcal{R}^{D \times K}$ is a transformation matrix; $\boldsymbol{x} \in \mathcal{R}^K$ are unknown multivariate Gaussian latent variables; $\boldsymbol{\mu} \in \mathcal{R}^D$ is a mean (offset) vector and $\boldsymbol{\epsilon}$ describes the model noise, typically Gaussian. Depending on the assumptions made about $\boldsymbol{x}$, $\boldsymbol{W}$ and $\boldsymbol{\epsilon}$, some widely used dimensionality reduction algorithms can be derived:

- PCA can be derived from Eq. (32) and additional assumptions that $\boldsymbol{\mu} = 0$, the vectors of $\boldsymbol{W}$ are orthogonal and the variance of the isotropic noise is 0, i.e. assume $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \mathbf{I}_D\right)$ and $\sigma \to 0$.
- Without the small variance asymptotics assumption, but having $\boldsymbol{W}$ with orthogonal columns and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \mathbf{I}_D\right)$, one recovers *probabilistic PCA* (PPCA) [82].
- In the case where the orthogonality assumption on $\boldsymbol{W}$ is omitted and a more flexible elliptical noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \text{diag}(\boldsymbol{\sigma}))$ is assumed with $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_D)$, one obtains the classic *factor analysis* (FA) [83].
- Variants of *independent component analysis* [84] can be obtained by assuming flexible elliptical noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \text{diag}(\boldsymbol{\sigma}))$ with $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_D)$, but also assuming a non-Gaussian distribution model for the latent variables $\boldsymbol{x} \in \mathcal{R}^K$, for example the multivariate Laplace distribution [85].

2) Gaussian process latent variable models: Similar to the way Gaussian processes were used to specify nonlinear regression approaches, one can also define interpretable nonlinear dimensionality reduction. To do this take the likelihood of a linear Gaussian latent variable model:

$$P\left(\boldsymbol{y}^{(n)} | \boldsymbol{x}^{(n)}, \boldsymbol{W}, \sigma\mathbf{I}\right) = \mathcal{N}\left(\boldsymbol{y}^{(n)} | \boldsymbol{W}\boldsymbol{x}^{(n)}, \sigma\mathbf{I}\right) \tag{33}$$

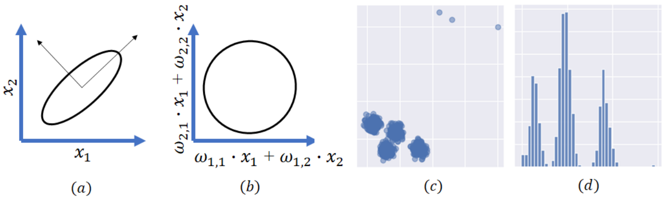

Fig. 9. Illustration of principal component analysis: (a) Given some input space described by correlated features x_1 and x_2, PCA estimates (b) new orthogonal axis which uncorrelate the feature space. (c) shows an example of clustered data with three present outliers; (d) displays a histogram of the 1-dimensional projection of this data onto the direction of its largest variance where outliers appear on the right edge of the figure.

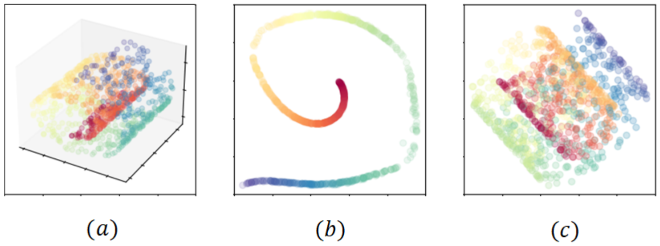

Fig. 10. Synthetic data generated on a swiss roll with random noise (a) shows the generated 1000 samples on a 3D scatter plot; (b) displays the top two latent dimensions of a GPLVM model fitted to the data; (c) displays a 2D projection of the swiss roll data using the eigenvectors with the largest two corresponding eigenvalues. Clearly, it is difficult to separate the different groups on the basis of (c).

and integrate out the projection matrix $\boldsymbol{W}$ so that the model can be expressed only in terms of the latent variables, giving the following likelihood across each dimension d [86]:

$$P\left(Y_d|\boldsymbol{X},\sigma\right) = \mathcal{N}\left(Y_d|\mathbf{0},\boldsymbol{X}\boldsymbol{X}^T\sigma\mathbf{I}\right) \tag{34}$$

where note that $P\left(\boldsymbol{Y}|\boldsymbol{X},\sigma\right) = \prod_{d=1}^{D} P\left(Y_d|\boldsymbol{X},\sigma\right)$ and $P\left(\boldsymbol{Y}|\boldsymbol{W},\sigma\right) = \prod_{i=1}^{N} P\left(\boldsymbol{y}^{(i)}|\boldsymbol{W},\sigma\right)$. The likelihoods from Eqs. (33) and (34) specify identical assumptions but also allow to interpret the linear Gaussian model as a special case of the more general latent variable model which assumes:

$$P\left(Y_d|\boldsymbol{X},\sigma\right) = \mathcal{N}\left(Y_d|\mathbf{0},\boldsymbol{K}\right) \tag{35}$$

with $\boldsymbol{K}$ denoting the covariance matrix of the underlying GP, which depends on the choice of kernel for the GP governing our model: for linear kernel $\kappa\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right) = \left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{x}^{(j)} + \beta^{-1}\delta\left(i,j\right)$ resulting in Eq. (34). Using alternative nonlinear kernels like the RBF kernel, a rich set of assumptions can be embedded about the lower dimensional space. Fig. 10 shows an example toy data generated to lie on a swiss roll surface in 3-D (Fig. 10(a)). PCA manages to rotate the data and infer the direction of the maximum variance (i.e. in Fig. 10(c)), but if the goal is capturing the geodesic distance structure and project points relative to their distance on the 3-D surface, the linear assumption falls short. In contrast, Fig. 10(b) shows the latent space inferred by a Bayesian GP-LVM (using 30 inducing points for training).

Similar to the Gaussian process regression models, GP-LVMs benefit from recent advances in scaling up inference in Gaussian processes. GP-LVMs have been augmented to fully Bayesian specification which allows to infer the kernel lengthscales [87], [88]; another extension [89] has also been proposed which allows for the inclusion of external covariates when learning the lower dimensional representation. Covariate GP-LVMs allow a user to infer a breakdown of feature-level variability, while inferring complex reduction of the high-dimensional input space.

3) Generative topographic mapping: Generative topographic mapping (GTM) [90] defines a non-linear, parametric mapping from a p-dimensional latent space ($p = 2, 3$ for visualization purposes) to the D-dimensional data space. Denote the continuous latent variables with $\boldsymbol{x} \in \mathcal{R}^p$ and assume the function $f(\boldsymbol{x},\boldsymbol{W})$ to be continuous and differentiable, GTM assumes that the likelihood of $\boldsymbol{y}$ can be written as:

$$P(\boldsymbol{y}|\boldsymbol{W},\beta) = \frac{1}{K}\sum_{k=1}^{K} P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{W},\beta) \tag{36}$$

where a grid of K densities describing the spread of data in the lower dimensional space is used. The underlying assumption is that each grid density is Gaussian with shared variance and mean determined by the associated embeddings:

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{W},\beta) = \mathcal{N}\left(\boldsymbol{y}|f(\boldsymbol{x},\boldsymbol{W}),\beta\right) \tag{37}$$

The ordering of the latent points reflects the ordering of the Gaussian centers in the data space (i.e. for continuous $f\left(\boldsymbol{x},\boldsymbol{W}\right)$), which ensures preservation of the local structure in the input. One can view GTMs as a constrained mixture model where all components depend on the mapping $f(\boldsymbol{x},\boldsymbol{W})$, the mixing coefficients are equal and fixed to $1/K$, and the component variance is shared. The motivation behind this setup can be found in early data visualization technique called *self-organizing maps* [91] where the goal is data visualization which preserves some of the high dimensional heterogeneity, rather then explicit clustering.

So far the form that f takes, has not been set: f can be expressed as a linear combination of fixed basis functions, $f_d(\boldsymbol{x},\boldsymbol{W}) = \sum_{m=1}^{M}\phi_m(\boldsymbol{x})w_{md}$. Common basis choices are:

- non-normalized Gaussian basis functions - $\phi_m(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_m\|^2}{2\sigma^2}\right)$ where $\boldsymbol{\mu}_m$ denote the centres of the Gaussian basis functions and σ is their common width;
- linear basis functions - $\phi_m(\boldsymbol{x}) = x_p$ where x_p denotes the p-th feature of $\boldsymbol{x}$;
- fixed basis functions (i.e. single bias term) that allows for the corresponding weights to act as biases: $\phi_m(\boldsymbol{x}) = 1$.

Since GP-LVMs utilize Gaussian processes to specify a full distribution over the non-linearities, they tend to provide a more flexible framework for modelling the relationship between the high-dimensional and low-dimensional spaces. However, embedding discrete variables allows for efficient and intepretable representations when it comes to cluster preservation. The constraints manifested by the K-point grid makes GTMs less flexible, than a GP-LVM representation, but also allow for faster inference.

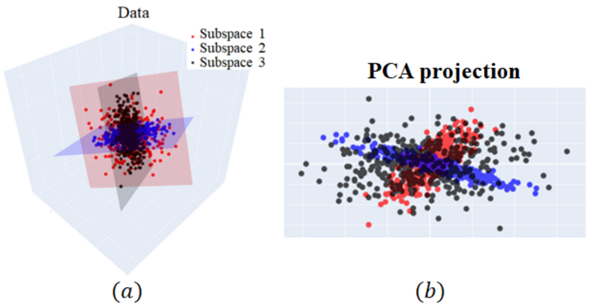

Fig. 11. Synthetically generated 3-dimensional linear Gaussian data which inherently lies on one of three 2-dimensional linear subspaces. The colored planes denote the subspace planes. (a) shows the raw data and the identified latent subspaces; (b) shows what a PCA transoformation of this data provides.

4) Piecewise linear dimensionality reduction: An alternative approach for constructing nonlinear dimensionality reduction is using a collection of local linear sub-models. We can augment the linear Gaussian model of Eq. (32) with set of discrete indicators $\boldsymbol{Z} = \left[\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}\right]^T$ that reflect the assumed clustering topology:

$$\boldsymbol{Y} = \boldsymbol{W}(\boldsymbol{X} \odot \boldsymbol{Z}) + \boldsymbol{E} \tag{38}$$

where $\odot$ denotes the *Hadamard product*, also known as the element-wise or *Schur product*.

If we assume that data lies on non-overlapping subspaces, we can adopt a mixture model setup with $\boldsymbol{Z} = \left[\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}\right]^T$ being one-hot encoding vectors. The basic setup would be assuming a *mixture of probabilistic PCA* models which would reduce the original D dimensional data into a set of different $\mathcal{R}^p$ lower dimensional representations. In each subspace, a set of points are modelled with a local probabilistic PCA where data is also inherently clustered into regions which can be modelled well with the same PCA. An illustration is shown in Fig. 11, where 3-dimensional data lie approximately on three 2-dimensional subspaces (i.e. each pair of subspaces sharing a single principal axis). The mixture of probabilistic PCA approach has been extended in a number of ways: proposing ways to learn the number of PCA components; learning the size of latent dimensionality p; and probabilistic priors which capture the uncertainty associated with transformer matrices $\boldsymbol{W}$.

Altogether different piecewise linear assumption would be that data shares different combinations of linear subspaces. Assume $\boldsymbol{Z} = \left[\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(N)}\right]^T$ in Eq. (38) is a binary indicator matrix selecting which of K hidden sources are active. The resulting models are known as *latent feature models* [85], [92], [93] and they provide more flexible means for structure decomposition by leveraging specified relationships in the observed data. The clustering topology in latent feature models can be more complex, with observations belonging to multiple components. In the context of visualization, depending on the probabilistic assumptions embedded in the indicator matrix $\boldsymbol{Z}$, models of the form in Eq. (38) capture a range of 1-dimensional linear subspaces where each data point is represented by a random combination of them. For example, the *adaptive* probabilistic PCA [93], allows for specifying the total number K of unique components which one thinks summarize the reduced space and one can also specify L - the smaller number of components (out of the total K) that each observation is associated with. In Fig. 11, with $K = 3$ and $L = 2$, an adaptive probabilistic PCA identifies three latent principal components which span the latent space, whereas a mixture of probabilistic PCA identifies three subspaces of two components.

B. Manifold embedding methods

Many algorithms focus on preserving the relative distance between points in the original data space while minimizing distortion in the low dimensional space. Unlike generative models, manifold embedding algorithms typically do not require specifying a full generative process for the data, but leverage generic assumptions about key information to be preserved in the dimensionality reduction process. A unification of many manifold embedding algorithms has been proposed in [76].

1) Multidimensional scaling: One of the simplest examples of manifold embedding algorithms is the *Multi-dimensional scaling* (MDS) [94]. MDS seeks a lower dimensional projection which preserves pairwise similarities of the input data. In the *classical* MDS, similarity is measured by Euclidean distances and the methods receive an input of the dissimilarity matrix $\widehat{d}$ with $\widehat{d_{ij}} = \|\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}\|$ for each i and j. MDS then seeks to find lower dimensional embedding of the data, $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$, minimizing the loss:

$$\mathcal{L}(\widehat{d_{ij}}) = \left(\frac{\sum_{i<j} \left(\widehat{d_{ij}} - f(d_{ij}) \right)^2}{\sum d_{ij}^2} \right)^{\frac{1}{2}} \tag{39}$$

where we also have $d_{ij} = \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|$; for the metric version of MDS f is a monotone function; non-metric MDS aims to preserve the relative ordering of the items, rather then pairwise Euclidean distances. If $f(\cdot)$ is an identity function, the loss can be optimized in a single step and this is known as the *classical* MDS, which infers identical projection to that of PCA.

It is well know that for high-dimensional data, Euclidean distance is often not a good measure of similarity between points, which has motivated multiple versions of MDS, varying in the way similarity matrices are computed. The most notable extension is the *Isomap* [95] which starts by creating a neighborhood graph. The shortest path is found between all pairs of inputs (using algorithm of choice such as Dijkstra's algorithm) and the length of the shortest paths is used as a measure of proximity between pairs of points, i.e. *geodesic distance*. Once the Euclidean distance is replaced with geodesic distance, Isomap proceeds identically to MDS.

2) t-distributed stochastic neighbourhood embedding: Somewhat in contrast to MDS and other related schemes, t-distributed stochastic neighbourhood embedding (t-SNE) focuses on minimizing the Kullback–Leibler (KL) divergence measure between the distribution of distances in the high dimensional inputs space and the distribution of pairwise distances in the embedding space. t-SNE assumes Gaussian densities to capture the joint probability of any two points in the input space; the conditional probability $p_{j|i}$ of point j given point i takes the form:

$$p_{j|i} = \frac{\exp(-\|\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}\|^2)/2\sigma^2}{\sum_{k \neq i} \exp(-\|\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(k)}\|^2/2\sigma^2)} \, . \tag{40}$$

The goal of t-SNE is to learn lower dimensional projections which preserve the structure in $p_{ji} = \frac{p_{j|i}+p_{i|j}}{2N}$ as close as possible by assuming Student-t distribution capturing the similarity between pairs of latent variables with:

$$q_{ij} = \frac{\left(1 + \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(l)}\|^2\right)^{-1}} \tag{41}$$

The training of t-SNE then minimizes the KL divergence between p_{ij} and q_{ij} using any standard gradient optimizer. This cost to be minimized takes the form:

$$C_{\text{t-SNE}} = C - \sum_{i \neq j} p_{ij} \log\left(q_{ij}\right) \tag{42}$$

where C a constant function of the data.

A key advantage of t-SNE is that the distribution matching allows one to reveal the structure at many scales on a single map, and in principle display in 2-D (or 3-D) that data lies in multiple different manifolds, instead of *crowding*[8] all samples for large D. However, the joint Gaussian probabilities cannot capture relationship between uncorrelated or far-off points. This is why t-SNE is known to preserve poorly global structure in the original data space. Other known downsides of t-SNE include poor scaling and tendency of producing clustered low space embedding even for uniformly distributed input.

C. Other manifold embedding methods

Another popular manifold embedding algorithm, the Uniform Manifold Approximation and Projection (UMAP), was not covered in this tutorial due to the need to introduce fuzzy simplicial set representations. UMAP has similar motivation to t-SNE, but similarity in the high-dimensional space is measured using local fuzzy simplicial set memberships based on smooth nearest neighbour distances. MDS also serves as a basis for flexible dimensionality reduction algorithms which use splines to preserve nonlinear structure in the pairwise similarity of the original data for instance in Neuroscale [96]. Nonlinear PCA was the motivating objective behind many autoencoder architectures, a big family of algorithms for visual informatics.

VII. Applications to Photonics

Many of the methods mentioned in this paper have been used already in various photonics applications [97] but there is still significant potential for using them more broadly. Exemplar applications are the use of Gaussian processes for image interpolation and for denoising, for division of focal plane sensors [98], for the optimization of complex optical structures [99], for modelling optical fiber communication behavior [100] and for predicting the bit error rate in Quadrature Phase Shift Keying (QPSK) wavelength-division multiplexing (WDM) communication system [101].

Bayesian probabilistic techniques, including mixture models, have been used for amplitude and phase noise characterization [102], for parameter estimation in a nonlinear state-space framework [103], for phase estimation [104] and for failure detection [105]. Linear regression has been used to estimate the BER of each new service requests [106] and for sparse identification for nonlinear optical communication systems using lasso regularization in a perturbative (linear) model [107].

Kernel based methods have been employed to solve a broad range of problems in the general area of photonics including, fibre optic nonlinearity mitigation, using K-means [108] and support vector machines [109]; support vector machines were also used for nonlinear phase noise mitigation for coherent optical systems machines [110], in quality of transmission prediction of unestablished lightpaths [111], in failure detection and identification [112] and in fiber nonlinearity equalization [113].

Message passing techniques have been used for stereo matching in image postfiltering [114], in pattern division multiple access in MMW-RoF Systems [115] and in decoding low-density parity-check decoded signals. Routing and network design could potentially benefit from message passing techniques [74] to provide scalable and principled routing techniques under variable objectives and constraints.

Of particular relevance to performance monitoring is the area of dimensionality reduction and visual informatics. While basic methods such as PCA have been used in many applications, from real-time feature extraction and dimensionality reduction in hyperspectral imaging [116] and optical performance monitoring, and modulation format identification [117] to modal characterization [118], more advanced techniques have not been used very often.

This is clearly not a comprehensive list of applications but merely points to a small representative selection. We would also like to point the reader to other reviews on the use of machine learning in optical communication that are somewhat related to the current tutorial [97], [103], [119], [120].

We are fully aware of the significant use of neural network methods in the area of photonics and their many successes in addressing some of the most difficult problems in this area. However, the aim of this review is primarily to explain the use of principled machine learning techniques and not a comparison against the performance offered by other methods. This comparison is problem dependent and hinges on what exactly constitutes success.

VIII. Conclusions

The review aims at introducing readers to principled and established machine learning methods, focusing on

[8] The volume of a sphere scales in higher dimensions D proportionally to r^D with r denoting the radius of the hypersphere. In 2-D there is much less area available for the same radius r, hence one may place items at a different relative distance from a reference point. Due to lack of room nearby one may place high dimensional items in a 2-D space far off than where they should (intuitively) be. In contrast, some items end up crowded in the center to stay close to all of the far-off points.

probabilistic, kernel based and decision tree techniques. It is not intended to be comprehensive and exhaustive but merely to explain the underlying fundamental ideas underpinning what we perceive to be the main directions, motivating their use and explaining their advantages and shortcomings. This tutorial should serve as a stepping stone to the use of principled machine learning algorithms in photonics applications.

Naturally, many methods have not been covered in this tutorial. Statistical machine learning is an extremely dynamic field, fueled by innovation. Recent branches of machine learning which have been omitted include:

- *Likelihood free methods* - which assume one can easily generate data samples but cannot construct a reliable generative model for the data [121]. These methods are not Bayesian and rely on statistical approaches that do not require a likelihood.
- *Causal inference methods* - most of this tutorial focuses on flexible paradigms for leveraging statistical associations in the data to do predictions or exploratory data analysis. The field of causal inference provides tools for going beyond correlating patterns and studying the cause-effect estimation tasks in different scenarios [122]. It encompasses assumptions and estimation strategies to allow one to draw causal relations from data.
- *Energy-based models* - energy-based learning provides a unified framework for non-probabilistic training of both probabilistic and heuristic approaches [123]. Stemming from statistical physics, they are trained to model an underlying data distribution from a sample dataset.

We hope the tutorial provides insight into the methods used in machine learning as well as basic understanding of their foundations, and that it will stimulate readers to use them for a variety of applications.

Acknowledgment

DS acknowledges support from the EPSRC Programme Grant TRANSNET (EP/R035342/1) and the Leverhulme trust (RPG-2018-092). YR acknowledges support by the EPSRC Horizon Digital Economy Research grant ‘Trusted Data Driven Products: EP/T022493/1 and grant ‘From Human Data to Personal Experience’: EP/M02315X/1. We would also like to thank Stylianos Sygletos for helpful comments on the manuscript.

References

[1] J. A. Nelder and R. W. Wedderburn, “Generalized linear models,” *Journal of the Royal Statistical Society: Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972.

[2] J. Piironen and A. Vehtari, “Sparsity information and regularization in the horseshoe and other shrinkage priors,” *Electronic Journal of Statistics*, vol. 11, no. 2, pp. 5018–5051, 2017.

[3] P. Zhao and B. Yu, “On model selection consistency of lasso,” *The Journal of Machine Learning Research*, vol. 7, pp. 2541–2563, 2006.

[4] G. Morota and D. Gianola, “Kernel-based whole-genome prediction of complex traits: a review,” *Frontiers in genetics*, vol. 5, p. 363, 2014.

[5] R. Agrawal, B. Trippe, J. Huggins, and T. Broderick, “The kernel interaction trick: Fast bayesian discovery of pairwise interactions in high dimensions,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 141–150.

[6] J. G. Proakis and M. Salehi, *Digital communications*. McGraw-Hill., 2008.

[7] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[8] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[9] T. Park and G. Casella, “The Bayesian LASSO,” *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 681–686, 2008.

[10] T. Sun and C.-H. Zhang, “Scaled sparse linear regression,” *Biometrika*, vol. 99, no. 4, pp. 879–898, 2012.

[11] D. F. Andrews, “A robust method for multiple linear regression,” *Technometrics*, vol. 16, no. 4, pp. 523–531, 1974.

[12] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[13] M. I. Jordan and R. A. Jacobs, “Hierarchical mixtures of experts and the EM algorithm,” *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[14] S. Wade, D. B. Dunson, S. Petrone, and L. Trippa, “Improving prediction from Dirichlet process mixtures via enrichment,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1041–1071, 2014.

[15] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[16] D. Duvenaud, “Automatic model construction with Gaussian processes,” Ph.D. dissertation, Computational and Biological Learning Laboratory, University of Cambridge, 2014.

[17] E. A. Nadaraya, “On estimating regression,” *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[18] G. S. Watson, “Smooth regression analysis,” *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

[19] M. Jones, S. Davies, and B. Park, “Versions of kernel-type regression estimators,” *Journal of the American Statistical Association*, vol. 89, no. 427, pp. 825–832, 1994.

[20] M. B. Priestley and M. Chao, “Non-parametric function fitting,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 3, pp. 385–392, 1972.

[21] T. Gasser, H.-G. Muller, and V. Mammitzsch, “Kernels for nonparametric curve estimation,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 238–252, 1985.

[22] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, pp. 65–386, 1958.

[23] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[24] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[25] T. Hastie, R. Tibshirani, and J. Friedman, “Random forests,” in *The elements of statistical learning*. Springer, 2009, pp. 587–604.

[26] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.

[27] J. H. Friedman, “Multivariate adaptive regression splines,” *The annals of statistics*, pp. 1–67, 1991.

[28] G. H. Golub, M. Heath, and G. Wahba, “Generalized cross-validation as a method for choosing a good ridge parameter,” *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[29] E. Kleinberg, “Stochastic discrimination,” *Annals of Mathematics and Artificial intelligence*, vol. 1, no. 1, pp. 207–239, 1990.

[30] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[31] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[33] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[34] M. A. Little and R. Badawy, “Causal bootstrapping,” *arXiv preprint arXiv:1910.09648*, 2019.

[35] G. Biau and E. Scornet, “A random forest guided tour,” *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[36] D. M. Roy and Y. Teh, “The mondrian process,” in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Curran Associates, Inc., 2009.

[37] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian forests: Efficient online random forests," *Advances in neural information processing systems*, vol. 27, pp. 3140–3148, 2014.

[38] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[39] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, vol. 50, no. 1, pp. 5–43, 2003.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[41] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information science and statistics, M. Jordan, J. Kleinberg, and B. Schölkopf, Eds. Springer, 2006, vol. 4, no. 4.

[42] Y. P. Raykov, A. Boukouvalas, F. Baig, and M. A. Little, "What to do when K-means clustering fails: a simple yet principled alternative algorithm," *PloS one*, vol. 11, no. 9, p. e0162259, 2016.

[43] J. W. Miller and M. T. Harrison, "Mixture models with a prior on the number of components," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 340–356, 2018.

[44] M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta, "Composing graphical models with neural networks for structured representations and fast inference," *Advances in neural information processing systems*, vol. 29, pp. 2946–2954, 2016.

[45] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1530–1538.

[46] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[47] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.*, ser. Adaptive computation and machine learning. MIT Press, 2006.

[48] J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," *ArXiv*, vol. abs/1711.00165, 2018.

[49] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Bayesian deep convolutional networks with many channels are Gaussian processes," in *ICLR*, 2019.

[50] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[51] D. Barber and C. Williams, "Gaussian processes for Bayesian classification via hybrid Monte Carlo," in *Advances in Neural Information Processing Systems 9*. MIT Press, 1997, pp. 340–346.

[52] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. 65, pp. 1939–1959, 2005.

[53] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2006.

[54] A. Damianou and N. D. Lawrence, "Deep Gaussian processes," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, C. M. Carvalho and P. Ravikumar, Eds., vol. 31. PMLR, 29 Apr–01 May 2013, pp. 207–215.

[55] M. Mézard and A. Montanari, *Information, physics, and computation*. Oxford University Press, 2009.

[56] D. J. Thouless, P. W. Anderson, and R. G. Palmer, "Solution of 'solvable model of a spin glass'," *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, vol. 35, no. 3, pp. 593–601, 1977.

[57] M. Opper and D. Saad, Eds., *Advanced mean field methods: theory and practice*, ser. Neural Information Processing. MIT, Feb. 2001.

[58] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*. World Scientific Publishing Company, 1987, vol. 9.

[59] P. Judea, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the Second National Conference on Artificial Intelligence (AAAI-82)*. CA: AAAI Press, 1982, pp. 133–136.

[60] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[61] Y. Kabashima and D. Saad, "Belief propagation vs. TAP for decoding corrupted messages," *Europhysics Letters (EPL)*, vol. 44, no. 5, pp. 668–674, dec 1998.

[62] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, pp. 764–785, 2010.

[63] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," *2011 IEEE International Symposium on Information Theory Proceedings*, pp. 2168–2172, 2011.

[64] G. T. Cantwell and M. E. J. Newman, "Message passing on networks with loops," *Proceedings of the National Academy of Sciences*, vol. 116, no. 47, pp. 23 398–23 403, 2019.

[65] A. Montanari and T. Rizzo, "How to compute loop corrections to the bethe approximation," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 10, p. P10011–P10011, Oct 2005. [Online]. Available: http://dx.doi.org/10.1088/1742-5468/2005/10/P10011

[66] M. Mézard, G. Parisi, and R. Zecchina, "Analytic and algorithmic solution of random satisfiability problems," *Science*, vol. 297, no. 5582, pp. 812–815, 2002.

[67] Y. Kabashima, T. Murayama, and D. Saad, "Typical performance of gallager-type error-correcting codes," *Phys. Rev. Lett.*, vol. 84, pp. 1355–1358, Feb 2000.

[68] Y. Kabashima, N. Sazuka, K. Nakamura, and D. Saad, "Tighter decoding reliability bound for gallager's error-correcting code," *Phys. Rev. E*, vol. 64, p. 046113, Sep 2001.

[69] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová, "Statistical-physics-based reconstruction in compressed sensing," *Phys. Rev. X*, vol. 2, p. 021005, May 2012.

[70] K. M. Wong and D. Saad, "Equilibration through local information exchange in networks," *Phys. Rev. E*, vol. 74, no. 1, p. 010104, 2006.

[71] C. H. Yeung and D. Saad, "Competition for shortest paths on sparse graphs," *Phys. Rev. Lett.*, vol. 108, no. 20, p. 208701, 2012.

[72] C. H. Yeung, D. Saad, and K. M. Wong, "From the physics of interacting polymers to optimizing routes on the London Underground," *Proceedings of the National Academy of Sciences*, vol. 110, no. 34, pp. 13 717–13 722, 2013.

[73] C. De Bacco, S. Franz, D. Saad, and C. H. Yeung, "Shortest node-disjoint paths on random graphs," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2014, no. 7, p. P07009, 2014.

[74] Y.-Z. Xu, H. F. Po, C. H. Yeung, and D. Saad, "Scalable node-disjoint and edge-disjoint multi-wavelength routing," *ArXiv*, vol. phys/2107.00609, 2021.

[75] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005800103

[76] A. Agrawal, A. Ali, and S. Boyd, "Minimum-distortion embedding," *arXiv preprint arXiv:2103.02559*, 2021.

[77] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.

[78] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," *Advances in neural information processing systems*, vol. 15, 2002.

[79] E. Snelson and Z. Ghahramani, "Local and global sparse gaussian process approximations," in *Artificial Intelligence and Statistics*. PMLR, 2007, pp. 524–531.

[80] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[81] T. F. De Lima, A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Scalable wideband principal component analysis via microwave photonics," *IEEE Photonics Journal*, vol. 8, no. 2, pp. 1–9, 2016.

[82] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[83] H. H. Harman, *Modern factor analysis.* Univ. of Chicago Press, 1960.

[84] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[85] D. Knowles and Z. Ghahramani, "Infinite sparse factor analysis and infinite independent components analysis," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 381–388.

[86] N. Lawrence and A. Hyvärinen, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models." *Journal of machine learning research*, vol. 6, no. 11, 2005.

[87] M. Titsias and N. D. Lawrence, "Bayesian Gaussian process latent variable model," in *Proceedings of the Thirteenth International*

Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, 2010, pp. 844–851.

[88] S. Ahmed, M. Rattray, and A. Boukouvalas, "GrandPrix: scaling up the Bayesian GPLVM for single-cell data," *Bioinformatics*, vol. 35, no. 1, pp. 47–54, 2019.

[89] K. Märtens, K. Campbell, and C. Yau, "Decomposing feature-level variation with covariate Gaussian process latent variable models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4372–4381.

[90] C. M. Bishop, M. Svensén, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.

[91] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[92] T. L. Griffiths and Z. Ghahramani, "Infinite latent feature models and the Indian buffet process," in *NIPS*, vol. 18, 2005, pp. 475–482.

[93] A. Farooq, Y. P. Raykov, P. Raykov, and M. A. Little, "Controlling for sparsity in sparse factor analysis models: adaptive latent feature sharing for piecewise linear dimensionality reduction," *arXiv preprint arXiv:2006.12369*, 2020.

[94] J. B. Kruskal, *Multidimensional scaling*. Sage, 1978, no. 11.

[95] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[96] D. Lowe and M. E. Tipping, "Neuroscale: novel topographic feature extraction using RBF networks," *Advances in neural information processing systems*, pp. 543–549, 1997.

[97] J. Mata, I. de Miguel, R. J. Durán, N. Merayo, S. K. Singh, A. Jukan, and M. Chamania, "Artificial intelligence (AI) methods in optical networks: A comprehensive survey," *Optical Switching and Networking*, vol. 28, pp. 43–57, 2018.

[98] E. Gilboa, J. P. Cunningham, A. Nehorai, and V. Gruev, "Image interpolation and denoising for division of focal plane sensors using Gaussian processes," *Opt. Express*, vol. 22, no. 12, pp. 15 277–15 291, Jun 2014.

[99] P.-I. Schneider, X. G. Santiago, C. Rockstuhl, and S. Burger, "Global optimization of complex optical structures using Bayesian optimization based on Gaussian processes," in *Digital Optical Technologies 2017*, B. C. Kress and P. Schelkens, Eds., vol. 10335, International Society for Optics and Photonics. SPIE, 2017, pp. 141 – 149.

[100] J. W. Nevin, F. J. Vaquero-Caballero, D. J. Ives, and S. J. Savory, "Physics-informed Gaussian process regression for optical fiber communication systems," *Journal of Lightwave Technology*, 2021.

[101] J. Wass, J. Thrane, M. Piels, R. Jones, and D. Zibar, "Gaussian process regression for WDM system performance prediction," in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE, 2017, pp. 1–3.

[102] D. Zibar, L. H. H. de Carvalho, M. Piels, A. Doberstein, J. Diniz, B. Nebendahl, C. Franciscangelis, J. Estaran, H. Haisch, N. G. Gonzalez, J. C. R. F. de Oliveira, and I. T. Monroy, "Application of machine learning techniques for amplitude and phase noise characterization," *J. Lightwave Technol.*, vol. 33, no. 7, pp. 1333–1343, Apr 2015.

[103] D. Zibar, M. Piels, R. Jones, and C. G. Schäeffer, "Machine learning techniques in optical communication," *J. Lightwave Technol.*, vol. 34, no. 6, pp. 1442–1452, Mar 2016.

[104] M. G. Taylor, "Phase estimation methods for optical coherent detection using digital signal processing," *J. Lightwave Technol.*, vol. 27, no. 7, pp. 901–914, Apr 2009.

[105] F. Musumeci, C. Rottondi, G. Corani, S. Shahkarami, F. Cugini, and M. Tornatore, "A tutorial on machine learning for failure management in optical networks," *J. Lightwave Technol.*, vol. 37, no. 16, pp. 4125–4139, Aug 2019.

[106] S. Oda, M. Miyabe, S. Yoshida, T. Katagiri, Y. Aoki, T. Hoshida, J. C. Rasmussen, M. Birk, and K. Tse, "A learning living network with open roadms," *J. Lightwave Technol.*, vol. 35, no. 8, pp. 1350–1356, Apr 2017.

[107] M. Sorokina, S. Sygletos, and S. Turitsyn, "Sparse identification for nonlinear optical communication systems: Sino method," *Opt. Express*, vol. 24, no. 26, pp. 30 433–30 443, Dec 2016. [Online]. Available: http://opg.optica.org/oe/abstract.cfm?URI=oe-24-26-30433

[108] D. Wang, M. Zhang, M. Fu, Z. Cai, Z. Li, H. Han, Y. Cui, and B. Luo, "Nonlinearity mitigation using a machine learning detector based on k-nearest neighbors," *IEEE Photonics Technology Letters*, vol. 28, no. 19, pp. 2102–2105, 2016.

[109] E. Giacoumidis, Y. Lin, J. Wei, I. Aldaya, A. Tsokanos, and L. P. Barry, "Harnessing machine learning for fiber-induced nonlinearity mitigation in long-haul coherent optical OFDM," *Future Internet*, vol. 11, no. 1, 2019.

[110] M. Li, S. Yu, J. Yang, Z. Chen, Y. Han, and W. Gu, "Nonparameter nonlinear phase noise mitigation by using m-ary support vector machine for coherent optical systems," *IEEE Photonics Journal*, vol. 5, pp. 7 800 312–7 800 312, 2013.

[111] C. Rottondi, L. Barletta, A. Giusti, and M. Tornatore, "Machine-learning method for quality of transmission prediction of unestablished lightpaths," *J. Opt. Commun. Netw.*, vol. 10, no. 2, pp. A286–A297, Feb 2018. [Online]. Available: http://www.osapublishing.org/jocn/abstract.cfm?URI=jocn-10-2-A286

[112] B. Shariati, M. Ruiz, J. Comellas, and L. Velasco, "Learning from the optical spectrum: Failure detection and identification," *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 433–440, 2019.

[113] T. Nguyen, S. Mhatli, E. Giacoumidis, L. Van Compernolle, M. Wuilpart, and P. Mégret, "Fiber nonlinearity equalizer based on support vector classification for coherent optical OFDM," *IEEE Photonics Journal*, vol. 8, no. 2, pp. 1–9, 2016.

[114] J. Kogler, F. Eibensteiner, M. Humenberger, C. Sulzbachner, M. Gelautz, and J. Scharinger, "Enhancement of sparse silicon retina-based stereo matching using belief propagation and two-stage postfiltering," *Journal of Electronic Imaging*, vol. 23, no. 4, p. 043011, 2014.

[115] S. Shen, Y.-W. Chen, Q. Zhou, and G.-K. Chang, "Demonstration of pattern division multiple access with message passing algorithm in MMW-RoF systems," in *Optical Fiber Communication Conference (OFC) 2020*. Optical Society of America, 2020, p. M2F.3.

[116] J. Zabalza, J. Ren, J. Ren, Z. Liu, and S. Marshall, "Structured covariance principal component analysis for real-time onsite feature extraction and dimensionality reduction in hyperspectral imaging," *Appl. Opt.*, vol. 53, no. 20, pp. 4440–4449, Jul 2014.

[117] M. C. Tan, F. N. Khan, W. H. Al-Arashi, Y. Zhou, and A. P. T. Lau, "Simultaneous optical performance monitoring and modulation format/bit-rate identification using principal component analysis," *J. Opt. Commun. Netw.*, vol. 6, no. 5, pp. 441–448, May 2014.

[118] A. Mourka, M. Mazilu, E. Wright, and K. Dholakia, "Modal characterization using principal component analysis: application to Bessel, higher-order Gaussian beams and their superposition," *Scientific Reports*, vol. 3, Mar. 2013.

[119] J. Thrane, J. Wass, M. Piels, J. C. M. Diniz, R. Jones, and D. Zibar, "Machine learning techniques for optical performance monitoring from directly detected pdm-qam signals," *Journal of Lightwave Technology*, vol. 35, no. 4, pp. 868–875, 2017.

[120] F. N. Khan, Q. Fan, C. Lu, and A. P. T. Lau, "An optical communication's perspective on machine learning and its applications," *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 493–516, 2019.

[121] S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of approximate Bayesian computation*. CRC Press, 2018.

[122] P. Spirtes, "Introduction to causal inference." *Journal of Machine Learning Research*, vol. 11, no. 5, 2010.

[123] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, *A tutorial on energy-based learning*. MIT Press, 2006.

Yordan P. Raykov is an Assistant Professor at the Horizon Institute at University of Nottingham. He received a BSc in mathematics from University of Leicester and PhD in machine learning from Aston University in 2013 and 2016 respectively. Following his PhD research, he worked with ARM on building a novel occupancy estimation device and with UCB Pharma, Radboudumc and Aston University on building algorithms for health monitoring of Parkinson's disease in free living using wearables. He joined Aston Unversity as a Lecturer at 2018 and University of Nottingham at 2021. His research lies at the interception of computer science and statistics, focusing on scalable latent structure discovery in sensor monitoring applications.

David Saad holds the 50th Anniversary Chair of Complexity Physics at Aston University, UK. He received a BA in physics and a BSc in electrical engineering from Technion, an MSc in physics and a Ph.D. in electrical engineering from Tel-Aviv University. He joined Edinburgh University in 1992 and Aston in 1995. His research focuses on the application of statistical physics methods to several fields, including neural networks, error-correcting codes, multi-node communication, network optimization, routing, noisy computation, epidemic spreading and advanced inference methods.