Neural Networks and Deep Learning

Chapter · February 2019

DOI: 10.1017/9781108380690.007

2 authors, including:

Steven L. Brunton
University of Washington Seattle
421 PUBLICATIONS **33,189** CITATIONS

SEE PROFILE

Chapter 6: Neural Networks and Deep Learning

from

Data Driven Science & Engineering

Machine Learning, Dynamical Systems, and Control

Cambridge University Press, 2019

Steven L. Brunton
Mechanical Engineering
University of Washington

J. Nathan Kutz
Applied Mathematics
University of Washington

Book website: databookuw.com

Contents

Chapter 6

Neural Networks and Deep Learning

Neural networks (NNs) were inspired by the Nobel prize winning work of Hubel and Wiesel on the primary visual cortex of cats [259]. Their seminal experiments showed that neuronal networks were organized in hierarchical layers of cells for processing visual stimulus. The first mathematical model of the NN, termed the Neocognitron in 1980 [193], had many of the characteristic features of today's deep convolutional NNs (or DCNNs), including a multilayer structure, convolution, max pooling and nonlinear dynamical nodes. The recent success of DCNNs in computer vision has been enabled by two critical components: (i) the continued growth of computational power, and (ii) exceptionally large labeled data sets which take advantage of the power of a *deep* multi-layer architecture. Indeed, although the theoretical inception of NNs has an almost four-decade history, the analysis of the ImageNet data set in 2012 [310] provided a watershed moment for NNs and deep learning [324]. Prior to this data set, there were a number of data sets available with approximately tens of thousands of labeled images. ImageNet provided over 15 million labeled, high-resolution images with over 22,000 categories. DCNNs, which are only one potential category of NNs, have since transformed the field of computer vision by dominating the performance metrics in almost every meaningful computer vision task intended for classification and identification.

Although ImageNet has been critically enabling for the field, NNs were textbook material in the early 1990s with a focus typically on a small number of layers. Critical machine learning tasks such as principal component analysis (PCA) were shown to be intimately connected with networks which included back propagation. Importantly, there were a number of critical innovations which established multilayer feedforward networks as a class of universal approximators [255]. The past five years have seen tremendous advances in NN architectures, many designed and tailored for specific application areas. Inno-

vations have come from algorithmic modifications that have led to significant performance gains in a variety of fields. These innovations include pretraining, dropout, inception modules, data augmentation with virtual examples, batch normalization, and/or residual learning (See Ref. [216] for a detailed exposition of NNs). This is only a partial list of potential algorithmic innovations, thus highlighting the continuing and rapid pace of progress in the field. Remarkably, NNs were not even listed as one of the top 10 algorithms of data mining in 2008 [562]. But a decade later, its undeniable and growing list of successes on challenge data sets make it perhaps the most important data mining tool for our emerging generation of scientists and engineers.

As already shown in the last two chapters, all of machine learning revolves fundamentally around optimization. NNs specifically optimize over a compositional function

$$\underset{\mathbf{A}_j}{\operatorname{argmin}} \left(f_M(\mathbf{A}_M, \cdots, f_2(\mathbf{A}_2, f_1(\mathbf{A}_1, \mathbf{x})) \cdots) + \lambda g(\mathbf{A}_j) \right) \tag{6.1}$$

which is often solved using stochastic gradient descent and back propagation algorithms. Each matrix $\mathbf{A}_k$ denotes the weights connecting the neural network from the kth to $(k+1)$th layer. It is a massively underdetermined system which is regularized by $g(\mathbf{A}_j)$. Composition and regularization are critical for generating expressive representations of the data and preventing overfitting, respectively. This general optimization framework is at the center of deep learning algorithms, and its solution will be considered in this chapter. Importantly, NNs have significant potential for overfitting of data so that cross-validation must be carefully considered. Recall that *if you don't cross-validate, you is dumb.*

6.1 Neural networks: 1-Layer networks

The generic architecture of a multi-layer NN is shown in Fig. 6.1. For classification tasks, the goal of the NN is to map a set of input data to a classification. Specifically, we train the NN to accurately map the data $\mathbf{x}_j$ to their correct label $\mathbf{y}_j$. As shown in Fig. 6.1, the input space has the dimension of the raw data $\mathbf{x}_j \in \mathbb{R}^n$. The output layer has the dimension of the designed classification space. Constructing the output layer will be discussed further in the following.

Immediately, one can see that there are a great number of design questions regarding NNs. How many layers should be used? What should be the dimension of the layers? How should the output layer be designed? Should one use all-to-all or sparsified connections between layers? How should the mapping between layers be performed: a *linear mapping* or a *nonlinear mapping*? Much like the tuning options on SVM and classification trees, NNs have a significant number of design options that can be tuned to improve performance.

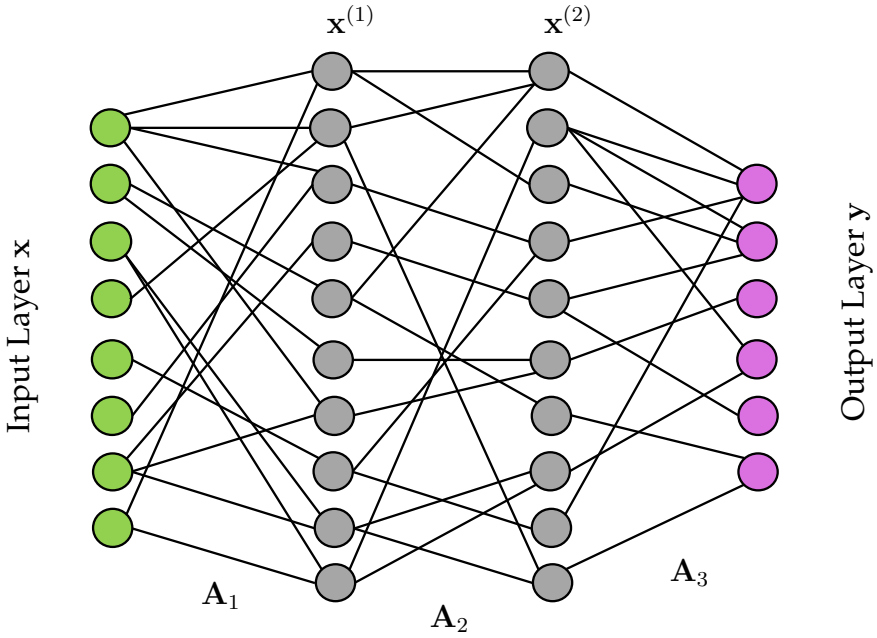

Figure 6.1: Illustration of a neural net architecture mapping an input layer $\mathbf{x}$ to an output layer $\mathbf{y}$. The middle (hidden) layers are denoted $\mathbf{x}^{(j)}$ where j determines their sequential ordering. The matrices $\mathbf{A}_j$ contain the coefficients that map each variable from one layer to the next. Although the dimensionality of the input layer $\mathbf{x} \in \mathbb{R}^n$ is known, there is great flexibility in choosing the dimension of the inner layers as well as how to structure the output layer. The number of layers and how to map between layers is also selected by the user. This flexible architecture gives great freedom in building a good classifier.

Initially, we consider the mapping between layers of Fig. 6.1. We denote the various layers between input and output as $\mathbf{x}^{(k)}$ where k is the layer number. For a linear mapping between layers, the following relations hold

$$\mathbf{x}^{(1)} = \mathbf{A}_1\mathbf{x} \tag{6.2a}$$
$$\mathbf{x}^{(2)} = \mathbf{A}_2\mathbf{x}^{(1)} \tag{6.2b}$$
$$\mathbf{y} = \mathbf{A}_3\mathbf{x}^{(2)}. \tag{6.2c}$$

This forms a compositional structure so that the mapping between input and output can be represented as

$$\mathbf{y} = \mathbf{A}_3\mathbf{A}_2\mathbf{A}_1\mathbf{x}. \tag{6.3}$$

This basic architecture can scale to M layers so that a general representation between input data and the output layer for a linear NN is given by

$$\mathbf{y} = \mathbf{A}_M\mathbf{A}_{M-1}\cdots\mathbf{A}_2\mathbf{A}_1\mathbf{x}. \tag{6.4}$$

This is generally a highly underdetermined system that requires some constraints on the solution in order to select a unique solution. One constraint is immediately obvious: The mapping must generate M distinct matrices that give the best mapping. It should be noted that linear mappings, even with a compositional structure, can only produce a limited range of functional responses due to the limitations of the linearity.

Nonlinear mappings are also possible, and generally used, in constructing the NN. Indeed, nonlinear activation functions allow for a richer set of functional responses than their linear counterparts. In this case, the connections between layers are given by

$$\begin{aligned} \mathbf{x}^{(1)} &= f_1(\mathbf{A}_1, \mathbf{x}) && \text{(6.5a)} \\ \mathbf{x}^{(2)} &= f_2(\mathbf{A}_2, \mathbf{x}^{(1)}) && \text{(6.5b)} \\ \mathbf{y} &= f_3(\mathbf{A}_3, \mathbf{x}^{(2)}). && \text{(6.5c)} \end{aligned}$$

Note that we have used different nonlinear functions $f_j(\cdot)$ between layers. Often a single function is used; however, there is no constraint that this is necessary. In terms of mapping the data between input and output over M layers, the following is derived

$$\mathbf{y} = f_M(\mathbf{A}_M, \cdots, f_2(\mathbf{A}_2, f_1(\mathbf{A}_1, \mathbf{x}))\cdots) \tag{6.6}$$

which can be compared with (6.1) for the general optimization which constructs the NN. As a highly underdetermined system, constraints should be imposed in order to extract a desired solution type, as in (6.1). For big data applications such as ImageNET and computer vision tasks, the optimization associated with this compositional framework is expensive given the number of variables that must be determined. However, for moderate sized networks, it can be performed on workstation and laptop computers. Modern stochastic gradient descent and back propagation algorithms enable this optimization, and both are covered in later sections.

A one-layer network

To gain insight into how an NN might be constructed, we will consider a single layer network that is optimized to build a classifier between dogs and cats. The dog and cat example was considered extensively in the previous chapter. Recall that we were given images of dogs and cats, or a wavelet version of dogs and cats. Figure 6.2 shows our construction. To make this as simple as possible, we consider the simple NN output

$$\mathbf{y} = \{\text{dog}, \text{cat}\} = \{+1, -1\} \tag{6.7}$$

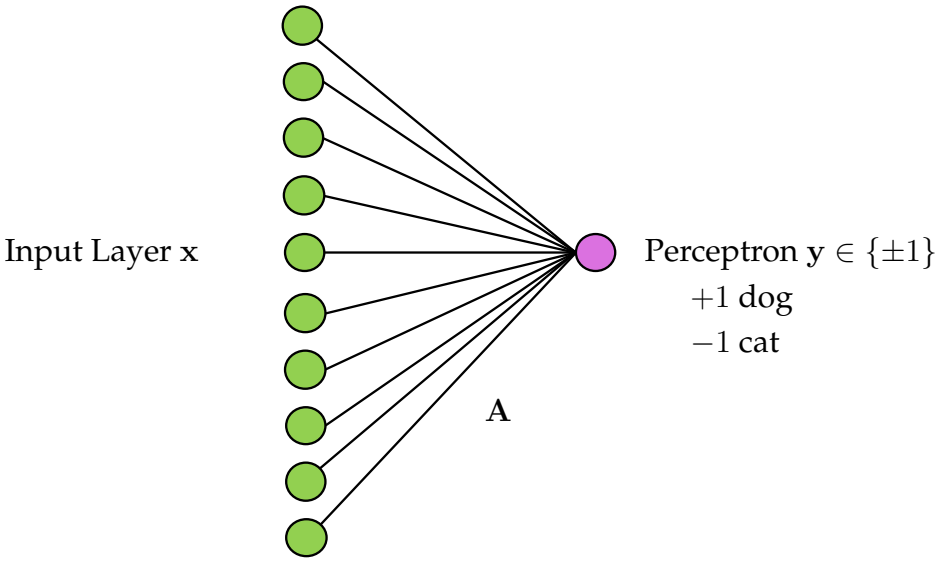

Figure 6.2: Single layer network for binary classification between dogs and cats. The output layer for this case is a perceptron with $\mathbf{y} \in \{\pm 1\}$. A linear mapping between the input image space and output output layer can be constructed for training data by solving $\mathbf{A} = \mathbf{Y}\mathbf{X}^\dagger$. This gives a least square regression for the matrix $\mathbf{A}$ mapping the images to label space.

which labels each data vector with an output $\mathbf{y} \in \{\pm 1\}$. In this case the output layer is a single node. As in previous supervised learning algorithms the goal is to determine a mapping so that each data vector $\mathbf{x}_j$ is labeled correctly by $\mathbf{y}_j$.

The easiest mapping is a linear mapping between the input images $\mathbf{x}_j \in \mathbb{R}^n$ and the output layer. This gives a linear system $\mathbf{AX} = \mathbf{Y}$ of the form

$$\mathbf{AX}=\mathbf{Y} \;\rightarrow\; [a_1\ a_2\ \cdots\ a_n] \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \\ | & | & & | \end{bmatrix} = [+1\ \ +1\ \cdots\ -1\ \ -1] \tag{6.8}$$

where each column of the matrix $\mathbf{X}$ is a dog or cat image and the columns of $\mathbf{Y}$ are its corresponding labels. Since the output layer is a single node, both $\mathbf{A}$ and $\mathbf{Y}$ reduce to vectors. In this case, our goal is to determine the matrix (vector) $\mathbf{A}$ with components a_j. The simplest solution is to take the pseudo-inverse of the data matrix $\mathbf{X}$

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^\dagger. \tag{6.9}$$

Thus a single output layer allows us to build a NN using least-square fitting. Of course, we could also solve this linear system in a variety of other ways, including with sparsity-promoting methods. The following code solves this problem through both least-square fitting (**pinv**) and the LASSO.

Code 6.1: 1-layer, linear neural network.

```
load catData_w.mat; load dogData_w.mat; CD=[dog_wave
```

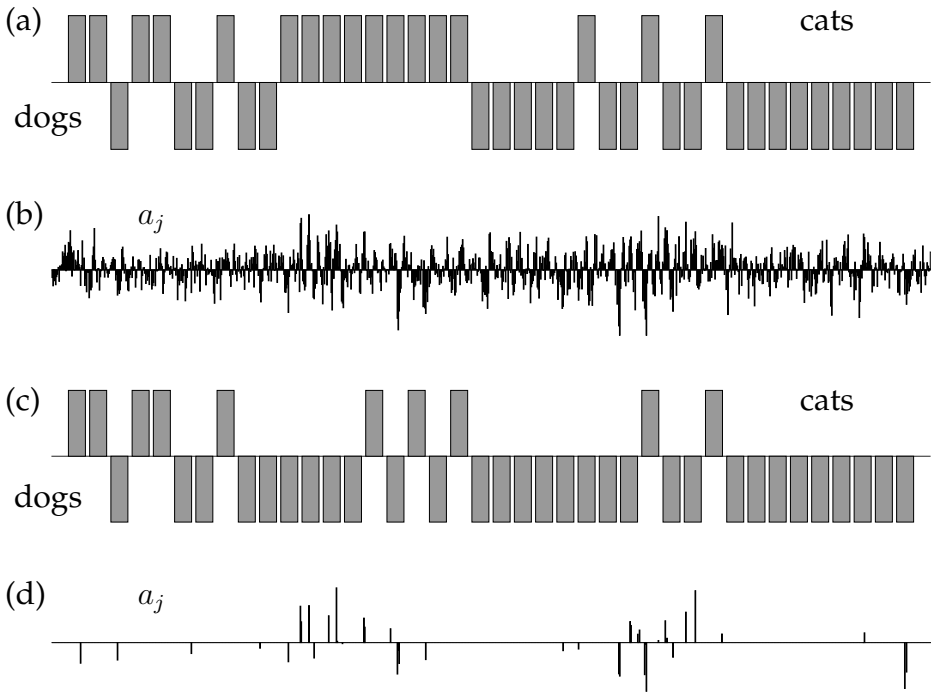

Figure 6.3: Classification of withheld data tested on a trained, single-layer network with linear mapping between inputs (pixel space) and a single output. (a) and (c) are the bar graph of the output layer score $\mathbf{y} \in \{\pm 1\}$ achieved for the withheld data using a pseudo-inverse for training and the LASSO for training respectively. The results show in both cases that dogs are more often misclassified than cats are misclassified. (b) and (d) show the coefficients of the matrix $\mathbf{A}$ for the pseudo-inverse and LASSO respectively. Note that the LASSO has only a small number of nonzero elements, thus suggesting the NN is highly sparse.

```
   cat_wave];
train=[dog_wave(:,1:60) cat_wave(:,1:60)];
test=[dog_wave(:,61:80) cat_wave(:,61:80)];
label=[ones(60,1); -1*ones(60,1)].';

A=label*pinv(train); test_labels=sign(A*test);
subplot(4,1,1), bar(test_labels)
subplot(4,1,2), bar(A)
figure(2), subplot(2,2,1)
A2=flipud(reshape(A,32,32)); pcolor(A2), colormap(gray)

figure(1), subplot(4,1,3)
A=lasso(train.',label.','Lambda',0.1).';
test_labels=sign(A*test);
bar(test_labels)
subplot(4,1,4)
```

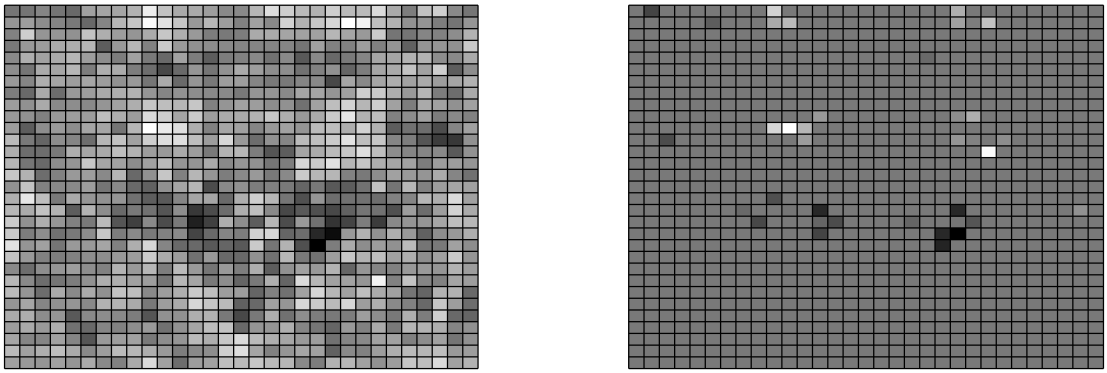

Figure 6.4: Weightings of the matrix $\mathbf{A}$ reshaped into 32×32 arrays. The left matrix shows the matrix $\mathbf{A}$ computed by least-square regression (the pseudo-inverse) while the right matrix shows the matrix $\mathbf{A}$ computed by LASSO. Both matrices provide similar classification scores on withheld data. They further provide interpretability in the sense that the results from the pseudo-inverse show many of the features of dogs and cats while the LASSO shows that measuring near the eyes and ears alone can give the features required for distinguishing between dogs and cats.

```
bar(A)
figure(2), subplot(2,2,2)
A2=flipud(reshape(A,32,32)); pcolor(A2), colormap(gray)
```

Figures 6.3 and 6.4 show the results of this linear single-layer NN with single node output layer. Specifically, the four rows of Fig. 6.3 show the output layer on the withheld test data for both the pseudo-inverse and LASSO methods along with a bar graph of the 32×32 (1024 pixels) weightings of the matrix $\mathbf{A}$. Note that all matrix elements are nonzero in the pseudo-inverse solution, while the LASSO highlights a small number of pixels that can classify the pictures as well as using all pixels. Figure 6.4 shows the matrix $\mathbf{A}$ for the two solution strategies reshaped into 32×32 images. Note that for the pseudo-inverse, the weightings of the matrix elements $\mathbf{A}$ show many features of the cat and dog face. For the LASSO method, only a few pixels are required that are clustered near the eyes and ears. Thus for this single layer network, interpretable results are achieved by looking at the weights generated in the matrix $\mathbf{A}$.

6.2 Multi-layer networks and activation functions

The previous section constructed what is perhaps the simplest NN possible. It was linear, had a single layer, and a single output layer neuron. The potential generalizations are endless, but we will focus on two simple extensions of the

NN in this section. The first extension concerns the assumption of linearity in which we assumed that there is a linear transform from the image space to the output layer: $\mathbf{Ax} = \mathbf{y}$ in (6.8). We highlight here common nonlinear transformations from input-to-output space represented by

$$\mathbf{y} = f(\mathbf{A}, \mathbf{x}) \tag{6.10}$$

where $f(\cdot)$ is a specified *activation function* (transfer function) for our mapping.

The linear mapping used previously, although simple, does not offer the flexibility and performance that other mappings offer. Some standard activation functions are given by

$$f(x) = x \quad - \text{ linear} \tag{6.11a}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases} \quad - \text{ binary step} \tag{6.11b}$$

$$f(x) = \frac{1}{1 + \exp(-x)} \quad - \text{ logistic (soft step)} \tag{6.11c}$$

$$f(x) = \tanh(x) \quad - \text{ TanH} \tag{6.11d}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \quad - \text{ rectified linear unit (ReLU).} \tag{6.11e}$$

There are other possibilities, but these are perhaps the most commonly considered in practice and they will serve for our purposes. Importantly, the chosen function $f(x)$ will be differentiated in order to be used in gradient descent algorithms for optimization. Each of the functions above is either differentiable or piecewise differentiable. Perhaps the most commonly used activation function is currently the ReLU, which we denote $f(x) = \text{ReLU}(x)$.

With a nonlinear activation function $f(x)$, or if there are more than one layer, then standard linear optimization routines such as the pseudo-inverse and LASSO can no longer be used. Although this may not seem immediately significant, recall that we are optimizing in a high-dimensional space where each entry of the matrix $\mathbf{A}$ needs to be found through optimization. Even moderate to small problems can be computationally expensive to solve without using specialty optimization methods. Fortunately, the two dominant optimization components for training NNs, stochastic gradient descent and backpropagation, are included with the neural network function calls in MATLAB. As these methods are critically enabling, both of them are considered in detail in the next two sections of this chapter.

Multiple layers can also be considered as shown in (6.4) and (6.5). In this case, the optimization must simultaneously identify multiple connectivity matrices $\mathbf{A}_1$, $\mathbf{A}_2, \cdots \mathbf{A}_M$, in contrast to the linear case where only a single matrix is determined $\bar{\mathbf{A}} = \mathbf{A}_M \cdots \mathbf{A}_2 \mathbf{A}_M$. The multiple layer structure significantly increases the size of the optimization problem as each matrix element of the M

matrices must be determined. Even for a one layer structure, an optimization routine such as **fminsearch** will be severely challenged when considering a nonlinear transfer function and one needs to move to a gradient descent-based algorithm.

MATLAB's neural network toolbox, much like TensorFlow in python, has a wide range of features which makes it exceptionally powerful and convenient for building NNs. In the following code, we will train a NN to classify between dogs and cats as in the previous example. However, in this case, we allow the single layer to have a nonlinear transfer function that maps the input to the output layer. The output layer for this example will be modified to the following

$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \{\text{dog}\} \text{ and } \mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \{\text{cat}\}. \tag{6.12}$$

Half of the data is extracted for training, while the other half is used for testing the results. The following code builds a network using the **train** command to classify between our images.

Code 6.2: Neural network with nonlinear transfer functions.

```
load catData_w.mat; load dogData_w.mat;
CD=[dog_wave cat_wave];

x=[dog_wave(:,1:40) cat_wave(:,1:40)];
x2=[dog_wave(:,41:80) cat_wave(:,41:80)];
label=[ones(40,1)  zeros(40,1);
       zeros(40,1) ones(40,1)].';

net = patternnet(2,'trainscg');
net.layers{1}.transferFcn = 'tansig';

net = train(net,x,label);
view(net)
y = net(x);
y2= net(x2);
perf = perform(net,label,y);
classes2 = vec2ind(y);
classes3 = vec2ind(y2);
```

In the code above, the **patternnet** command builds a classification network with two outputs (6.12). It also optimizes with the option **trainscg** which is a *scaled conjugate gradient backpropagation*. The **net.layers** also allows us to specify the transfer function, in this case hyperbolic tangent functions (6.11d). The **view**(**net**) command produces a diagnostic tool shown in Fig. 6.5 that summarizes the optimization and NN.

The results of the classification for a cross-validated training set as well as a

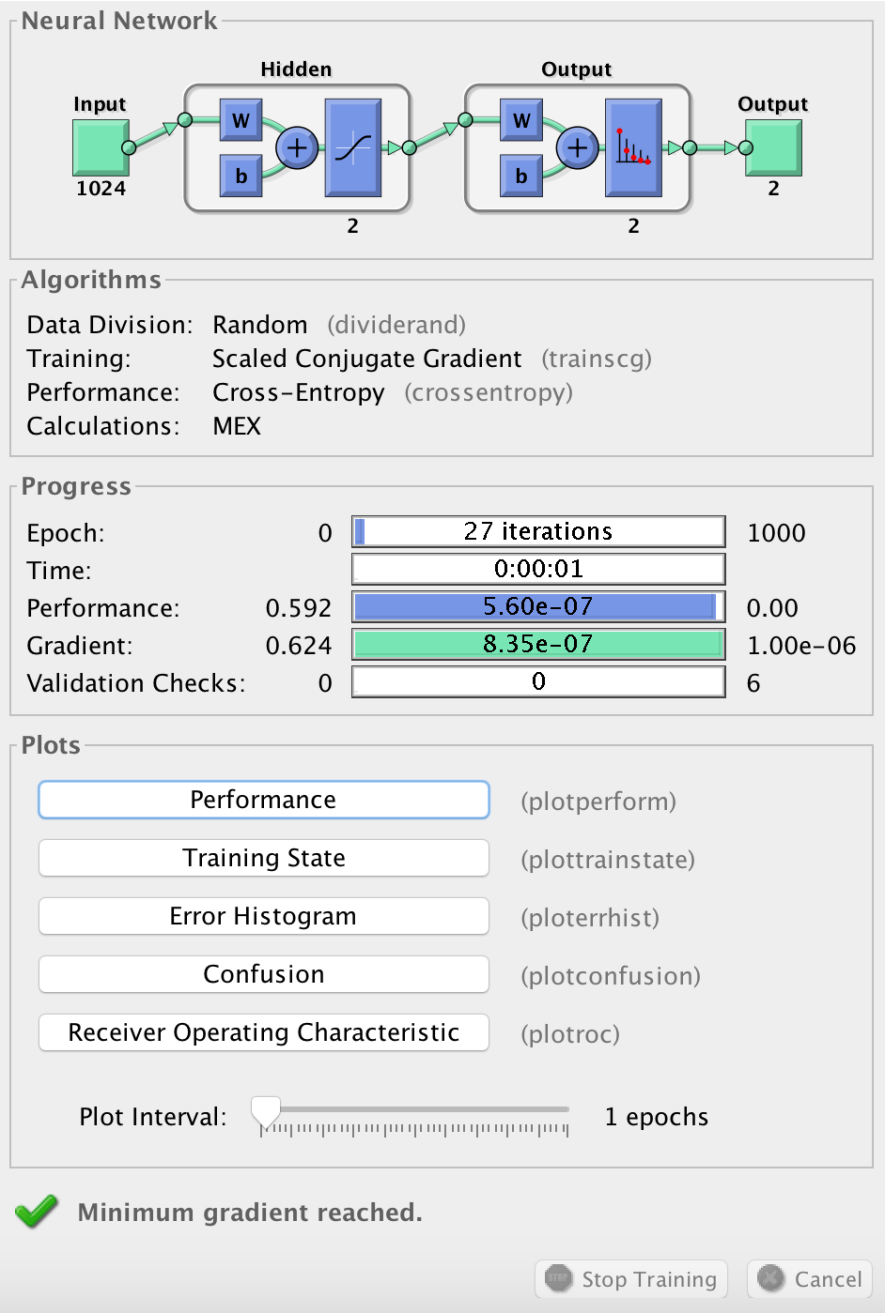

Figure 6.5: MATLAB neural network visualization tool. The number of iterations along with the performance can all be accessed from the interactive graphical tool. The performance, error histogram and confusion buttons produce Figs. 6.7-6.9 respectively.

withhold set are shown in Fig. 6.6. Specifically, the desired outputs are given by the vectors (6.12). For both the training and withhold sets, the two components of the vector are shown for the 80 training images (40 cats and 40 dogs) and the

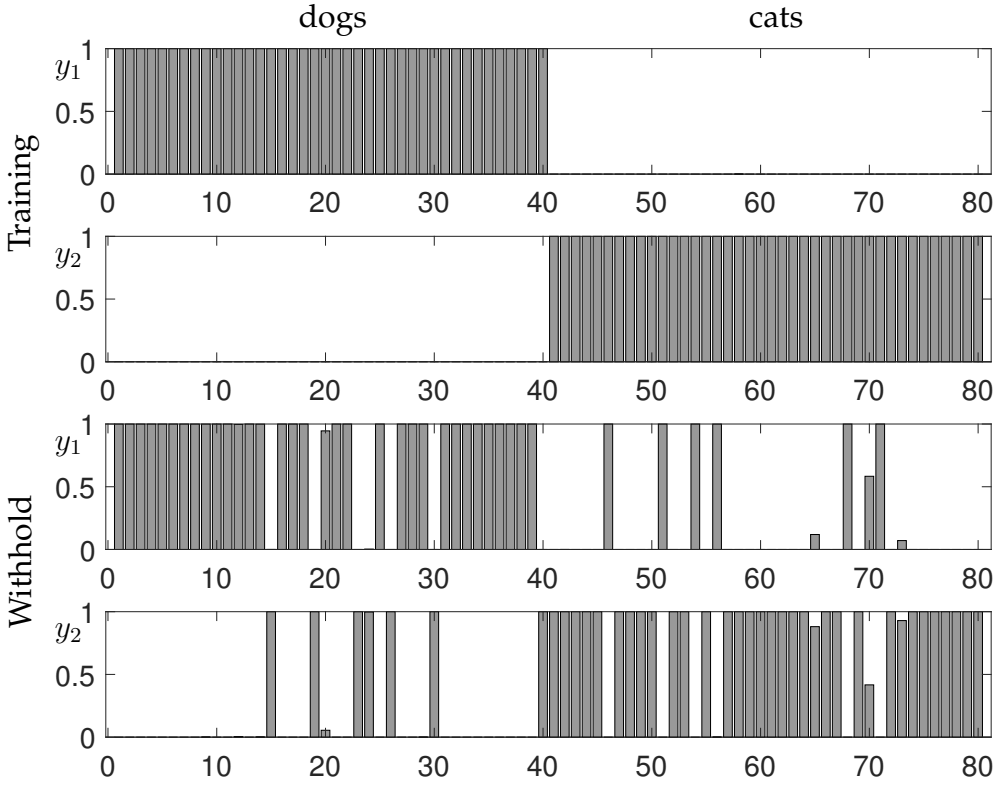

Figure 6.6: Comparison of the output vectors $\mathbf{y} = [y_1\ y_2]^T$ which are ideally (6.12) for the dogs and cats considered here. The NN training stage produces a cross-validated classifier that achieves 100% accuracy in classifying the training data (top two panels for 40 dogs and 40 cats). When applied to a withheld set, 85% accuracy is achieved (bottom two panels for 40 dogs and 40 cats).

80 withheld images (40 cats and 40 dogs). The training set produces a perfect classifier using a one layer network with a hyperbolic tangent transfer function (6.11d). On the withheld data, it incorrectly identifies 6 of 40 dogs and cats, yielding an accuracy of $\approx 85\%$ on new data.

The diagnostic tool shown in Fig. 6.5 allows access to a number of features critical for evaluating the NN. Figure 6.7 is a summary of the performance achieved by the NN training tool. In this figure, the training algorithm automatically breaks the data into a training, validation and test set. The back-propagation enabled, stochastic gradient descent optimization algorithm then iterates through a number of training epochs until the cross-validated error achieves a minimum. In this case, twenty-two epochs is sufficient to achieve a minimum. The error on the test set is significantly higher than what is achieved for cross-validation. For this case, only a limited amount of data is used for training (40 dogs and 40 cats), thus making it difficult to achieve great performance. Regardless, as already shown, once the algorithm has been trained it can be used to evaluate new data as shown in Fig. 6.6.

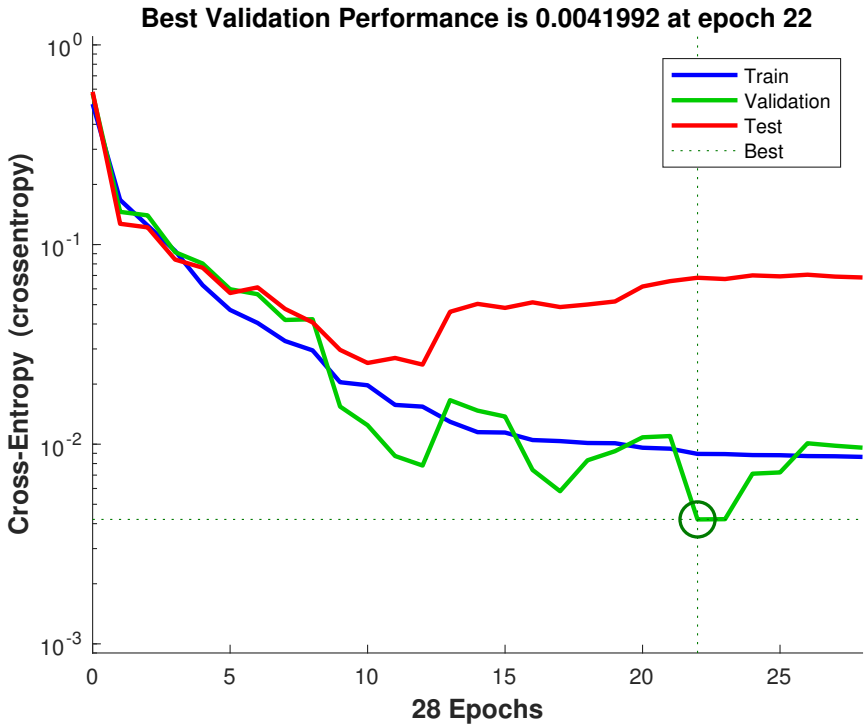

Figure 6.7: Summary of training of the NN over a number of epochs. The NN architecture automatically separates the data into training, validation and test sets. The training continues (with a maximum of 1000 epochs) until the validation error curve hits a minimum. The training then stops and the trained algorithm is then used on the test set to evaluate performance. The NN trained here has only a limited amount of data (40 dogs and 40 cats), thus limiting the performance. This figure is accessed with the **performance** button on the NN interactive tool of Fig. 6.6.

There are two other features easily available with the NN diagnostic tool of Fig. 6.5. Figure 6.8 shows an error histogram associated with the trained network. As with Fig. 6.7, the data is divided into training, validation, and test sets. This provides an overall assessment of the classification quality that can be achieved by the NN training algorithm. Another view of the performance can be seen in the confusion matrices for the training, validation, and test data. This is shown in Fig. 6.9. Overall, between Figs. 6.7 to 6.9, high-quality diagnostic tools are available to evaluate how well the NN is able to achieve its classification task. The performance limits are easily seen in these figures.

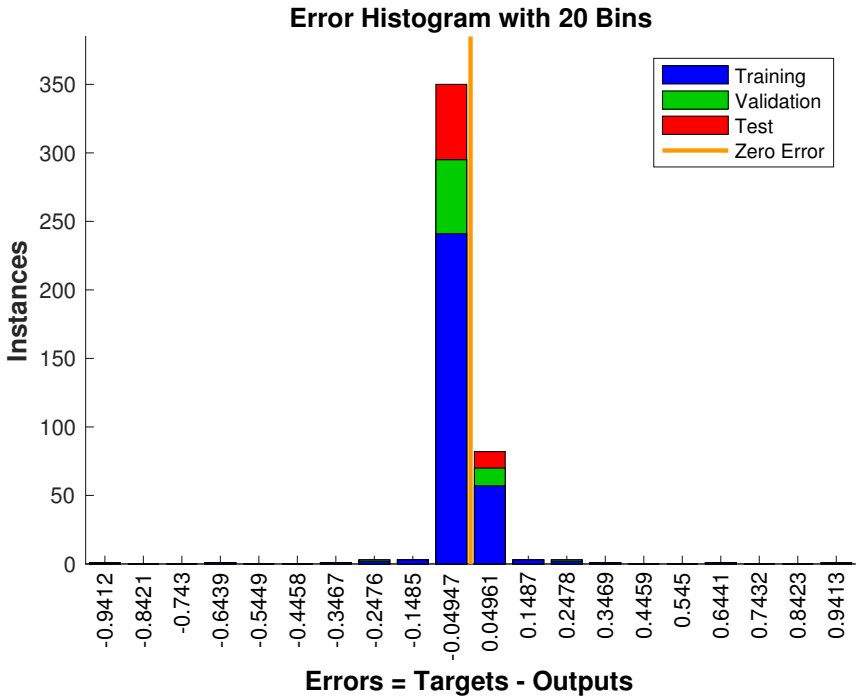

Figure 6.8: Summary of the error performance of the NN architecture for training, validation and test sets. This figure is accessed with the **errorhistogram** button on the NN interactive tool of Fig. 6.6.

6.3 The backpropagation algorithm

As was shown for the NNs of the last two sections, training data is required to determine the weights of the network. Specifically, the network weights are determined so as to best classify dog versus cat images. In the 1-layer network, this was done using both least-square regression and LASSO. This shows that at its core, an optimization routine and objective function is required to determine the weights. The objective function should minimize a measure of the misclassified images. The optimization, however, can be modified by imposing a regularizer or constraints, such as the ℓ_1 penalization in LASSO.

In practice, the objective function chosen for optimization is not the true objective function desired, but rather a proxy for it. Proxies are chosen largely due to the ability to differentiate the objective function in a computationally tractable manner. There are also many different objective functions for different tasks. Instead, one often considers a suitably chosen loss function so as to approximate the true objective. Ultimately, computational tractability is critical for training NNs.

The backpropagation algorithm (backprop) exploits the compositional nature of NNs in order to frame an optimization problem for determining the weights of the network. Specifically, it produces a formulation amenable to

Figure 6.9: Summary of the error performance through confusion matrices of the NN architecture for training, validation and test sets. This figure is accessed with the **confusion** button on the NN interactive tool of Fig. 6.6.

standard gradient descent optimization (See Sec. **??**). Backprop relies on a simple mathematical principle: the chain rule for differentiation. Moreover, it can be proven that the computational time required to evaluate the gradient is within a factor of five of the time required for computing the actual function itself [44]. This is known as the Baur-Strassen theorem. Figure 6.10 gives the simplest example of backprop and how the gradient descent is to be performed. The input-to-output relationship for this single node, one hidden layer network, is given by

$$y = g(z, b) = g(f(x, a), b). \tag{6.13}$$

Thus given a function $f(\cdot)$ and $g(\cdot)$ with weighting constants a and b, the output error produce by the network can be computed against the ground truth as

$$E = \frac{1}{2}(y_0 - y)^2 \tag{6.14}$$

where y_0 is the correct output and y is the NN approximation to the output.

The goal is to find a and b to minimize the error. The minimization requires

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz}\frac{dz}{da} = 0\,. \tag{6.15}$$

A critical observation is that the compositional nature of the network along with the chain rule forces the optimization to backpropagate error through the network. In particular, the terms $dy/dz\ dz/da$ show how this backprop occurs. Given functions $f(\cdot)$ and $g(\cdot)$, the chain rule can be explicitly computed.

Backprop results in an iterative, gradient descent update rule

$$a_{k+1} = a_k + \delta\frac{\partial E}{\partial a_k} \tag{6.16a}$$

$$b_{k+1} = b_k + \delta\frac{\partial E}{\partial b_k} \tag{6.16b}$$

where δ is the so-called learning rate and $\partial E/\partial a$ along with $\partial E/\partial b$ can be explicitly computed using (6.15). The iteration algorithm is executed to convergence. As with all iterative optimization, a good initial guess is critical to achieve a good solution in a reasonable amount of computational time.

Backprop proceeds as follows: (i) A NN is specified along with a labeled training set. (ii) The initial weights of the network are set to random values. Importantly, one must not initialize the weights to zero, similar to what may be done in other machine learning algorithms. If weights are initialized to zero, after each update, the outgoing weights of each neuron will be identical, because the gradients will be identical. Moreover, NNs often get stuck at local optima where the gradient is zero but that are not global minima, so random weight initialization allows one to have a chance of circumventing this by starting at many different random values. (iii) The training data is run through the network to produce an output $\mathbf{y}$, whose ideal ground-truth output is $\mathbf{y}_0$. The derivatives with respect to each network weight is then computed using backprop formulas (6.15). (iv) For a given learning rate δ, the network weights are updated as in (6.16). (v) We return to step (iii) and continue iterating until a maximum number of iterations is reached or convergence is achieved.

As a simple example, consider the linear activation function

$$f(\xi, \alpha) = g(\xi, \alpha) = \alpha\xi. \tag{6.17}$$

In this case we have in Fig. 6.10

$$z = ax \tag{6.18a}$$

$$y = bz. \tag{6.18b}$$

We can now explicitly compute the gradients such as (6.15). This gives

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz}\frac{dz}{da} = -(y_0 - y)\cdot b\cdot x \tag{6.19a}$$

$$\frac{\partial E}{\partial b} = -(y_0 - y)\frac{dy}{db} = -(y_0 - y)z = -(y_0 - y)\cdot a\cdot x. \tag{6.19b}$$

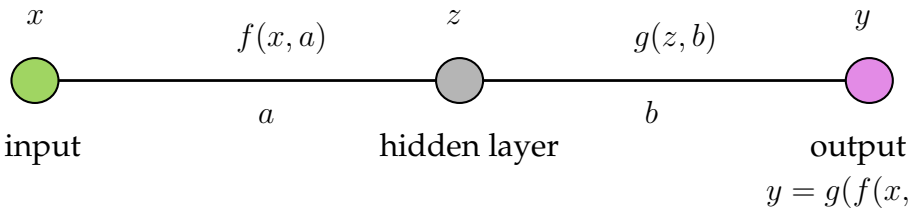

Figure 6.10: Illustration of the backpropagation algorithm on a one-node, one hidden layer network. The compositional nature of the network gives the input-output relationship $y = g(z, b) = g(f(x, a), b)$. By minimizing the error between the output y and its desired output y_0, the composition along with the chain rule produces an explicit formula (6.15) for updating the values of the weights. Note that the chain rule backpropagates the error all the way through the network. Thus by minimizing the output, the chain rule acts on the compositional function to produce a product of derivative terms that advance backward through the network.

Thus with the current values of a and b, along with the input-output pair x and y and target truth y_0 , each derivative can be evaluated. This provides the required information to perform the update (6.16).

The backprop for a deeper net follows in a similar fashion. Consider a network with M hidden layers labeled z_1 to z_m with the first connection weight a between x and z_1. The generalization of Fig. 6.10 and (6.15) is given by

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz_m}\frac{dz_m}{dz_{m-1}}\cdots\frac{dz_2}{dz_1}\frac{dz_1}{da}\,. \tag{6.20}$$

The cascade of derivates induced by the composition and chain rule highlights the backpropagation of errors that occurs when minimizing the classification error.

A full generalization of backprop involves multiple layers as well multiple nodes per layer. The general situation is illustrated in Fig. 6.1. The objective is to determine the matrix elements of each matrix $\mathbf{A}_j$. Thus a significant number of network parameters need to be updated in gradient descent. Indeed, training a network can often be computationally infeasible even though the update rules for individual weights is not difficult. NNs can thus suffer from the curse of dimensionality as each matrix from one layer to another requires updating n^2 coefficients for an n-dimensional input, assuming the two connected layers are both n-dimensional.

Denoting all the weights to be updated by the vector $\mathbf{w}$, where $\mathbf{w}$ contains all the elements of the matrices $\mathbf{A}_j$ illustrated in Fig. 6.1, then

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \delta\nabla E \tag{6.21}$$

where the gradient of the error ∇E, through the composition and chain rule,

produces the backpropagation algorithm for updating the weights and reducing the error. Expressed in a component-by-component way

$$w_{k+1}^j = w_k^j + \delta \frac{\partial E}{\partial w_k^j} \tag{6.22}$$

where this equation holds for the jth component of the vector $\mathbf{w}$. The term $\partial E/\partial w^j$ produces the backpropagation through the chain rule, i.e. it produces the sequential set of functions to evaluate as in (6.20). Methods for solving this optimization more quickly, or even simply enabling the computation to be tractable, remain of active research interest. Perhaps the most important method is stochastic gradient descent which is considered in the next section.

6.4 The stochastic gradient descent algorithm

Training neural networks is computationally expensive due to the size of the NNs being trained. Even NNs of modest size can become prohibitively expensive if the optimization routines used for training are not well informed. Two algorithms have been especially critical for enabling the training of NNs: *stochastic gradient descent* (SGD) and backprop. Backprop allows for an efficient computation of the objective function's gradient while SGD provides a more rapid evaluation of the optimal network weights. Although alternative optimization methods for training NNs continue to provide computational improvements, backprop and SGD are both considered here in detail so as to give the reader an idea of the core architecture for building NNs.

Gradient descent was considered in Sec. **??**. Recall that this algorithm was developed for nonlinear regression where the data fit takes the general form

$$f(x) = f(x, \boldsymbol{\beta}) \tag{6.23}$$

where $\boldsymbol{\beta}$ are fitting coefficients used to minimize the error. In NNs, the parameters $\boldsymbol{\beta}$ are the network weights, thus we can rewrite this in the form

$$f(\mathbf{x}) = f(\mathbf{x}, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) \tag{6.24}$$

where the $\mathbf{A}_j$ are the connectivity matrices from one layer to the next in the NN. Thus $\mathbf{A}_1$ connects the first and second layers, and there are M hidden layers.

The goal of training the NN is to minimize the error between the network and the data. The standard root-mean square error for this case is defined as

$$\underset{\mathbf{A}_j}{\operatorname{argmin}} E(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = \underset{\mathbf{A}_j}{\operatorname{argmin}} \sum_{k=1}^{n} (f(\mathbf{x}_k, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) - \mathbf{y}_k)^2 \tag{6.25}$$

which can be minimized by setting the partial derivative with respect to each matrix component to zero, i.e. we require $\partial E/\partial(a_{ij})_k = 0$ where $(a_{ij})_k$ is the ith row and jth column of the kth matrix ($k = 1, 2, \cdots M$). Recall that the zero derivate is a minimum since there is no maximum error. This gives the gradient $\nabla f(\mathbf{x})$ of the function with respect to the NN parameters. Note further that $f(\cdot)$ is the function evaluated at each of the n data points.

As was shown in Sec. **??**, this leads to a Newton-Raphson iteration scheme for finding the minima

$$\mathbf{x}_{j+1}(\delta) = \mathbf{x}_j - \delta \nabla f(\mathbf{x}_j) \tag{6.26}$$

where δ is a parameter determining how far a step should be taken along the gradient direction. In NNs, this parameter is called the *learning rate*. Unlike standard gradient descent, it can be computationally prohibitive to compute an optimal learning rate.

Although the optimization formulation is easily constructed, evaluating (6.25) is often computationally intractable for NNs. This due to two reasons: (i) the number of matrix weighting parameters for each $\mathbf{A}_j$ is quite large, and (ii) the number of data points n is generally also large.

To render the computation (6.25) potentially tractable, SGD does not estimate the gradient in (6.26) using all n data points. Rather, a single, randomly chosen data point, or a subset for *batch gradient descent*, is used to approximate the gradient at each step of the iteration. In this case, we can reformulate the least-square fitting of (6.25) so that

$$E(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = \sum_{k=1}^{n} E_k(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) \tag{6.27}$$

and

$$E_k(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = \left(f_k(\mathbf{x}_k, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) - \mathbf{y}_k\right)^2 \tag{6.28}$$

where $f_k(\cdot)$ is now the fitting function for each data point, and the entries of the matrices $\mathbf{A}_j$ are determined from the optimization process.

The gradient descent iteration algorithm (6.26) is now updated as follows

$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_k(\mathbf{w}_j) \tag{6.29}$$

where $\mathbf{w}_j$ is the vector of all the network weights from $\mathbf{A}_j$ ($j = 1, 2, \cdots, M$) at the jth iteration, and the gradient is computed using only the kth data point and $f_k(\cdot)$. Thus instead of computing the gradient with all n points, only a single data point is randomly selected and used. At the next iteration, another randomly selected point is used to compute the gradient and update the solution. The algorithm may require multiple passes through all the data to converge, but each step is now easy to evaluate versus the expensive computation of the

Jacobian which is required for the gradient. If instead of a single point, a subset of points is used, then we have the following batch gradient descent algorithm

$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_K(\mathbf{w}_j) \tag{6.30}$$

where $K \in [k_1, k_2, \cdots k_p]$ denotes the p randomly selected data points k_j used to approximate the gradient.

The following code is a modification of the code shown in Sec. **??** for gradient descent. The modification here involves taking a significant subsampling of the data to approximate the gradient. Specifically, a batch gradient descent is illustrated with a fixed learning rate of $\delta = 2$. Ten points are used to approximate the gradient of the function at each step.

Code 6.3: Stochastic gradient descent algorithm.

```
h=0.1; x=-6:h:6; y=-6:h:6; n=length(x);
[X,Y]=meshgrid(x,y); clear x, clear y

F1=1.5-1.6*exp(-0.05*(3*(X+3).^2+(Y+3).^2));
F=F1 + (0.5-exp(-0.1*(3*(X-3).^2+(Y-3).^2)));
[dFx,dFy]=gradient(F,h,h);

x0=[4  0 -5]; y0=[0 -5  2]; col=['ro','bo','mo'];
for jj=1:3
  q=randperm(n); i1=sort(q(1:10));
  q2=randperm(n); i2=sort(q2(1:10));
  x(1)=x0(jj); y(1)=y0(jj);
  f(1)=interp2(X(i1,i2),Y(i1,i2),F(i1,i2),x(1),y(1));
  dfx=interp2(X(i1,i2),Y(i1,i2),dFx(i1,i2),x(1),y(1));
  dfy=interp2(X(i1,i2),Y(i1,i2),dFy(i1,i2),x(1),y(1));

  tau=2;
  for j=1:50
    x(j+1)=x(j)-tau*dfx; % update x, y, and f
    y(j+1)=y(j)-tau*dfy;
    q=randperm(n); ind1=sort(q(1:10));
    q2=randperm(n); ind2=sort(q2(1:10));
    f(j+1)=interp2(X(i1,i2),Y(i1,i2),F(i1,i2),x(j+1),y(j+1))
    dfx=interp2(X(i1,i2),Y(i1,i2),dFx(i1,i2),x(j+1),y(j+1));
    dfy=interp2(X(i1,i2),Y(i1,i2),dFy(i1,i2),x(j+1),y(j+1));
    if abs(f(j+1)-f(j))<10^(-6) % check convergence
        break
    end
  end
  if jj==1; x1=x; y1=y; f1=f; end
  if jj==2; x2=x; y2=y; f2=f; end
```

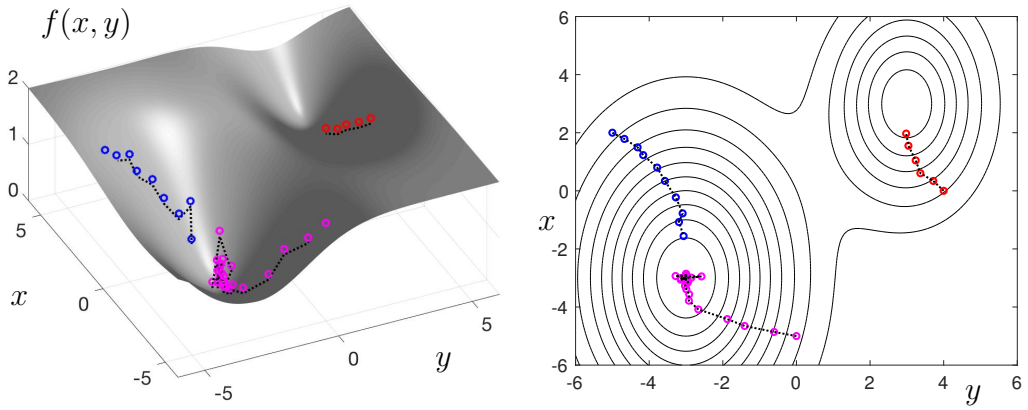

Figure 6.11: Stochastic gradient descent applied to the function featured in Fig. **??**(b). The convergence can be compared to a full gradient descent algorithm as shown in Fig. **??**. Each step of the stochastic (batch) gradient descent selects 100 data points for approximating the gradient, instead of the 10^4 data points of the data. Three initial conditions are shown: $(x_0, y_0) = \{(4,0),(0,-5),(-5,2)\}$. The first of these (red circles) gets stuck in a local minima while the other two initial conditions (blue and magenta) find the global minima. Interpolation of the gradient functions of Fig. **??** are used to update the solutions.

```
  if jj==3; x3=x; y3=y; f3=f; end
  clear x, clear y, clear f
end
```

Figure 6.11 shows the convergence of SGD for three initial conditions. As with gradient descent, the algorithm can get stuck in local minima. However, the SGD now approximates the gradient with only 100 points instead of the full 10^4 points, thus allowing for a computation which is three orders of magnitude smaller. Importantly, the SGD is a scalable algorithm, allowing for significant computational savings even as the data grows to be high-dimensional. For this reason, SGD has become a critically enabling part of NN training. Note that the learning rate, batch size, and data sampling play an important role in the convergence of the method.

6.5 Deep convolutional neural networks

With the basics of the NN architecture in hand, along with an understanding of how to formulate an optimization framework (backprop) and actually compute the gradient descent efficiently (SGD), we are ready to construct *deep convolution neural nets* (DCNN) which are the fundamental building blocks of *deep learning* methods. Indeed, today when practitioners generally talk about NNs for practical use, they are typically talking about DCNNs. But as much as we

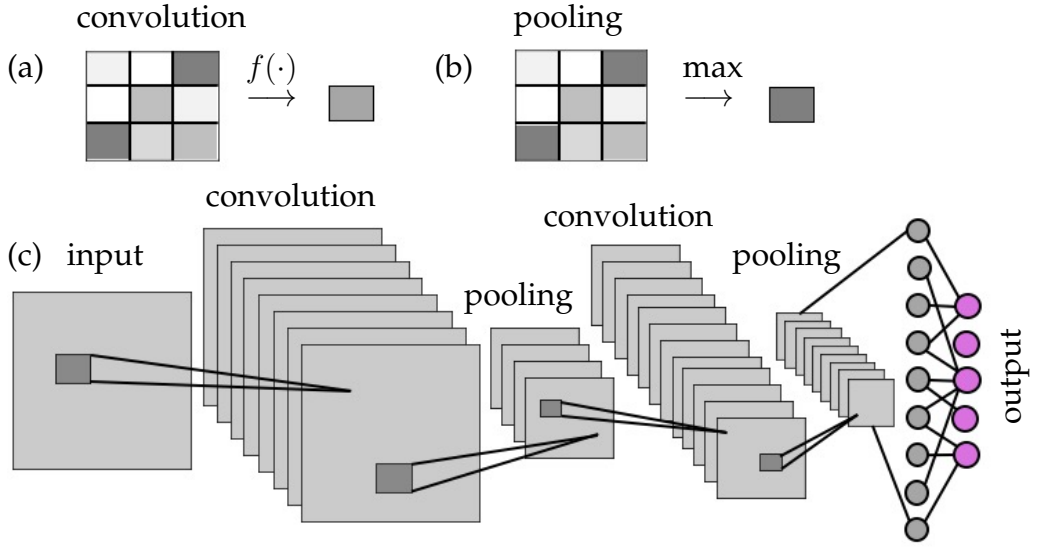

Figure 6.12: Prototypical DCNN architecture which includes commonly used convolutional and pooling layers. The dark gray boxes show the convolutional sampling from layer to layer. Note that for each layer, many functional transformations can be used to produce a variety of feature spaces. The network ultimately integrates all this information into the output layer.

would like to have a principled approach to building DCNNs, there remains a great deal of artistry and expert intuition for producing the highest performing networks. Moreover, DCNNs are especially prone to overtraining, thus requiring special care to cross-validate the results. The recent textbook on deep learning by Goodfellow et al. [216] provides a detailed an extensive account of the state-of-the-art in DCNNs. It is especially useful for highlighting many rules-of-thumb and tricks for training effective DCNNs.

Like SVM and random forest algorithms, the MATLAB package for building NNs has a tremendous number of features and tuning parameters. This flexibility is both advantageous and overwhelming at the same time. As was pointed out at the beginning of this chapter, it is immediately evident that there are a great number of design questions regarding NNs. How many layers should be used? What should be the dimension of the layers? How should the output layer be designed? Should one use all-to-all or sparsified connections between layers? How should the mapping between layers be performed: a *linear mapping* or a *nonlinear mapping*?

The prototypical structure of a DCNN is illustrated in Fig. 6.12. Included in the visualization is a number of commonly used convolutional and pooling layers. Also illustrated is the fact that each layer can be used to build multiple downstream layers, or *feature* spaces, that can be engineered by the choice of activation functions and/or network parametrizations. All of these layers are ultimately combined into the output layer. The number of connections that require updating through backprop and SGD can be extraordinarily high, thus

even modest networks and training data may require signifiant computational resources. A typical DCNN is constructed of a number of layers, with DCNNs typically having between 7-10 layers. More recent efforts have considered the advantages of a truly deep network with approximately 100 layers, but the merits of such architectures are still not fully known. The following paragraphs highlight some of the more prominent elements that comprise DCNNs, including convolutional layers, pooling layers, fully-connected layers and dropout.

Convolutional layers

Convolutional layers are similar to windowed (Gabor) Fourier transforms or wavelets from Chapter **??**, in that a small selection of the full high-dimensional input space is extracted and used for feature engineering. Figure 6.12 shows the convolutional windows (dark gray boxes) that are slid across the entire layer (light gray boxes). Each convolution window transforms the data into a new node through a given activation function, as shown in Fig. 6.12(a). The feature spaces are thus built from the smaller patches of the data. Convolutional layers are especially useful for images as they can extract important features such as edges. Wavelets are also known to efficiently extract such features and there are deep mathematical connections between wavelets and DCNNs as shown by Mallat and co-workers [358, 12]. Note that in Fig. 6.12, the input layer can be used to construct many layers by simply manipulating the activation function $f(\cdot)$ to the next layer as well the size of the convolutional window.

Pooling layers

It is common to periodically insert a Pooling layer between successive convolutional layers in a DCNN architecture. Its function is to progressively reduce the spatial size of the representation in order to reduce the number of parameters and computation in the network. This is an effective strategy to (i) help control overfitting and (ii) fit the computation in memory. Pooling layers operate independently on every depth slice of the input and resize them spatially. Using the max operation, i.e. the maximum value for all the nodes in its convolutional window, is called *max pooling*. In image processing, the most common form of max pooling is a pooling layer with filters of size 2×2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every max pooling operation would in this case be taking a max over 4 numbers (a 2x2 region in some depth slice). The depth dimension remains unchanged. An example max pooling operation is shown in Fig. 6.12(b), where a 3×3 convolutional cell is transformed to a single number which is the maximum of the 9 numbers.

Fully-connected layers

Occasionally, fully-connected layers are inserted into the DCNN so that different regions can be connected. The pooling and convolutional layers are *local* connections only, while the fully-connected layer restores *global* connectivity. This is another commonly used layer in the DCNN architecture, providing a potentially important feature space to improve performance.

Dropout

Overfitting is a serious problem in DCNNs. Indeed, overfitting is at the core of why DCNNs often fail to demonstrate good generalizability properties (See Chapter **??** on regression). Large DCNNs are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets for online implementation. Dropout is a technique which helps address this problem. The key idea is to randomly drop nodes in the network (along with their connections) from the DCNN during training, i.e. during SGD/backprop updates of the network weights. This prevents units from co-adapting too much. During training, dropout samples form an exponential number of different "thinned" networks. This idea is similar to the ensemble methods for building random forests. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and has shown to give major improvements over other regularization methods [499].

There are many other techniques that have been devised for training DCNNs, but the above methods highlight some of the most commonly used. The most successful applications of these techniques tend to be in computer vision tasks where DCNNs offer unparalleled performance in comparison to other machine learning methods. Importantly, the ImageNET data set is what allowed these DCNN layers to be maximally leveraged for human level recognition performance.

To illustrate how to train and execute a DCNN, we use data from MATLAB. Specifically, we use a data set that has a training and test set with the alphabet characters A, B, and C. The following code loads the data set and plots a representative sample of the characters in Fig. 6.13.

Code 6.4: Loading alphabet images.

```
load lettersTrainSet
perm = randperm(1500,20);
for j = 1:20
    subplot(4,5,j);
```

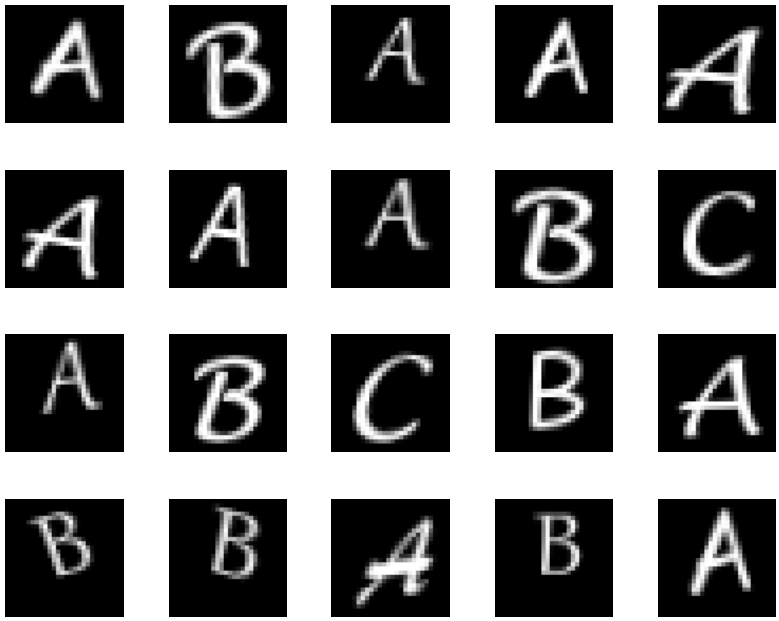

Figure 6.13: Representative images of the alphabet characters A, B, and C. There are a total of 1500 28×28 grayscale images (XTrain) of the letters that are labeled (TTrain).

```
    imshow(XTrain(:,:,:,perm(j)));
end
```

This code loads the training data, XTrain, that contains 1500 28×28 grayscale images of the letters A, B, and C in a 4-D array. There are equal numbers of each letter in the data set. The variable TTrain contains the categorical array of the letter labels, i.e. the truth labels. The following code constructs and trains a DCNN.

Code 6.5: Train a DCNN.

```
layers = [imageInputLayer([28 28 1]);
          convolution2dLayer(5,16);
          reluLayer();
          maxPooling2dLayer(2,'Stride',2);
          fullyConnectedLayer(3);
          softmaxLayer();
          classificationLayer()];
options = trainingOptions('sgdm');
rng('default') % For reproducibility
net = trainNetwork(XTrain,TTrain,layers,options);
```

Note the simplicity in how diverse network layers are easily put together. In addition, a ReLu activation layer is specified along with the training method

of stochastic gradient descent (sgdm). The **trainNetwork** command integrates the options and layer specifications to build the best classifier possible. The resulting trained network can now be used on a test data set.

Code 6.6: Test the DCNN performance.

```
load lettersTestSet;
YTest = classify(net,XTest);
accuracy = sum(YTest == TTest)/numel(TTest)
```

The resulting classification performance is approximately 93%. One can see by this code structure that modifying the network architecture and specifications is trivial. Indeed, one can probably easily engineer a network to outperform the illustrated DCNN. As already mentioned, artistry and expert intuition are critical for producing the highest performing networks.

6.6 Neural networks for dynamical systems

Neural networks offer an amazingly flexible architecture for performing a diverse set of mathematical tasks. To return to S. Mallat: *Supervised learning is a high-dimensional interpolation problem [358].* Thus if sufficiently rich data can be acquired, NNs offer the ability to interrogate the data for a variety of tasks centered on classification and prediction. To this point, the tasks demonstrated have primarily been concerned with computer vision. However, NNs can also be used for future state predictions of dynamical systems (See Chapter **??**).

To demonstrate the usefulness of NNs for applications in dynamical systems, we will consider the Lorenz system of differential equations [345]

$$\dot{x} = \sigma(y - x) \tag{6.31a}$$
$$\dot{y} = x(\rho - z) - y \tag{6.31b}$$
$$\dot{z} = xy - \beta z, \tag{6.31c}$$

where the state of the system is given by $\mathbf{x} = [x\ y\ z]^T$ with the parameters $\sigma = 10, \rho = 28$, and $\beta = 8/3$. This system will be considered in further detail in the next chapter. For the present, we will simulate this nonlinear system and use it as a demonstration of how NNs can be trained to characterize dynamical systems. Specifically, the goal of this section is to demonstrate that we can train a NN to learn an update rule which advances the state space from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$, where k denotes the state of the system at time t_k. Accurately advancing the solution in time requires a nonlinear transfer function since Lorenz itself is nonlinear.

The training data required for the NN is constructed from high-accuracy simulations of the Lorenz system. The following code generates a diverse set of initial conditions. One hundred initial conditions are considered in order to

generate one hundred trajectories. The sampling time is fixed at $\Delta t = 0.01$. Note that the sampling time is not the same as the time-steps taken by the 4th-order Runge-Kutta method [316]. The time-steps are adaptively chosen to meet the stringent tolerances of accuracy chosen for this example.

Code 6.7: Create training data of Lorenz trajectories.

```
% Simulate Lorenz system
dt=0.01; T=8; t=0:dt:T;
b=8/3; sig=10; r=28;

Lorenz = @(t,x)([ sig * (x(2) - x(1))        ; ...
                  r * x(1)-x(1) * x(3) - x(2) ; ...
                  x(1) * x(2) - b*x(3)         ]);
ode_options = odeset('RelTol',1e-10, 'AbsTol',1e-11);

input=[]; output=[];
for j=1:100  % training trajectories
    x0=30*(rand(3,1)-0.5);
    [t,y] = ode45(Lorenz,t,x0);
    input=[input; y(1:end-1,:)];
    output=[output; y(2:end,:)];
    plot3(y(:,1),y(:,2),y(:,3)), hold on
    plot3(x0(1),x0(2),x0(3),'ro')
end
```

The simulation of the Lorenz system produces to key matrices: **input** and **output**. The former is a matrix of the system at $\mathbf{x}_k$, while the latter is the corresponding state of the system $\mathbf{x}_{k+1}$ advanced $\Delta t = 0.01$.

The NN must learn the nonlinear mapping from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$. Figure 6.14 shows the various trajectories used to train the NN. Note the diversity of initial conditions and the underlying attractor of the Lorenz system.

We now build a NN trained on trajectories of Fig. 6.14 to advance the solution $\Delta t = 0.01$ into the future for an arbitrary initial condition. Here, a three-layer network is constructed with ten nodes in each layer and a different activation unit for each layer. The choice of activation types, nodes in the layer and number of layers are arbitrary. It is trivial to make the network deeper and wider and enforce different activation units. The performance of the NN for the arbitrary choices made is quite remarkable and does not require additional tuning. The NN is build with the following few lines of code.

Code 6.8: Build a neural network for Lorenz system.

```
net = feedforwardnet([10 10 10]);
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'radbas';
net.layers{3}.transferFcn = 'purelin';
```

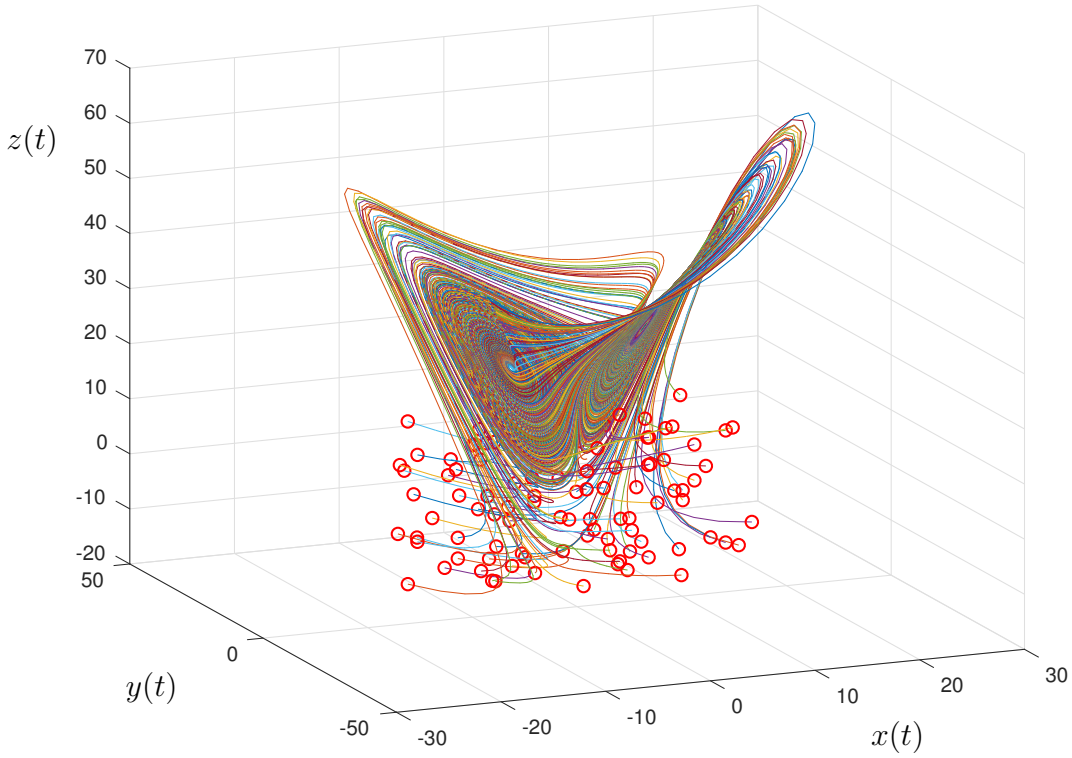

Figure 6.14: Evolution of the Lorenz dynamical equations for one hundred randomly chosen initial conditions (red circles). For the parameters $\sigma = 10, \rho = 28$, and $\beta = 8/3$, all trajectories collapse to an attractor. These trajectories, generated from a diverse set of initial data, are used to train a neural network to learn the nonlinear mapping from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$.

```
net = train(net,input.',output.');
```

The code produces a function **net** which can be used with a new set of data to produce predictions of the future. Specifically, the function **net** gives the nonlinear mapping from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$. Figure 6.15 shows the structure of the network along with the performance of the training over 1000 epochs of training. The results of the cross-validation are also demonstrated. The NN converges steadily to a network that produces accuracies on the order of 10^{-5}.

Once the NN is trained on the trajectory data, the nonlinear model mapping $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ can be used to predict the future state of the system from an initial condition. In the following code, the trained function **net** is used to take an initial condition and advance the solution Δt. The output can be re-inserted into the **net** function to estimate the solution $2\Delta t$ into the future. This iterative mapping can produce a prediction for the future state as far into the future as desired. In what follows, the mapping is used to predict the Lorenz solutions eight time units into the future from for a given initial condition. This can

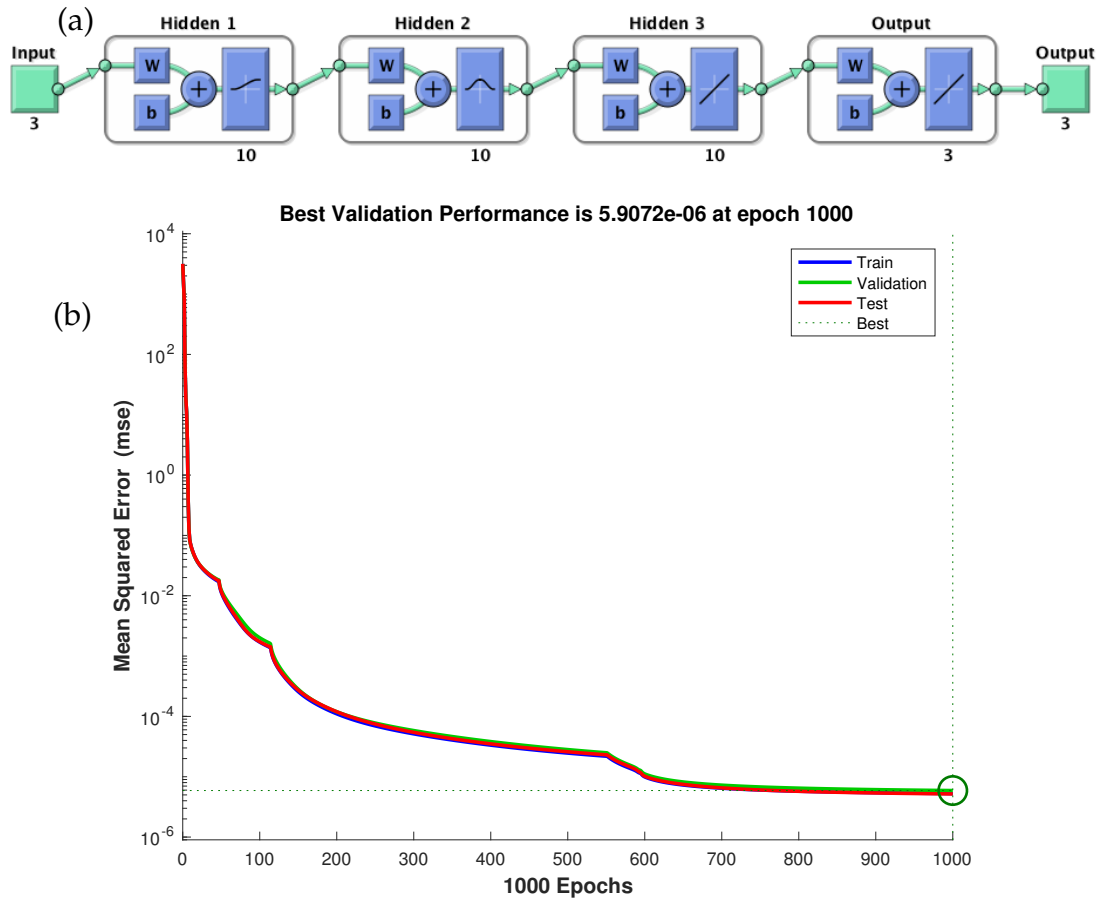

Figure 6.15: (a) Network architecture used to train the NN on the trajectory data of Fig. 6.14. A three-layer network is constructed with ten nodes in each layer and a different activation unit for each layer. (b) Performance summary of the NN optimization algorithm. Over 1000 epochs of training, accuracies on the order of 10^{-5} are produced. The NN is also cross-validated in the process.

then be compared against the ground truth simulation of the evolution using a 4th-order Runge-Kutta method. The following iteration scheme gives the NN approximation to the dynamics.

Code 6.9: Neural network for prediction.

```
ynn(1,:)=x0;
for jj=2:length(t)
    y0=net(x0);
    ynn(jj,:)=y0.'; x0=y0;
end
plot3(ynn(:,1),ynn(:,2),ynn(:,3),':','Linewidth',[2])
```

Figure 6.16 shows the evolution of two randomly drawn trajectories (solid lines) compared against the NN prediction of the trajectories (dotted lines). The

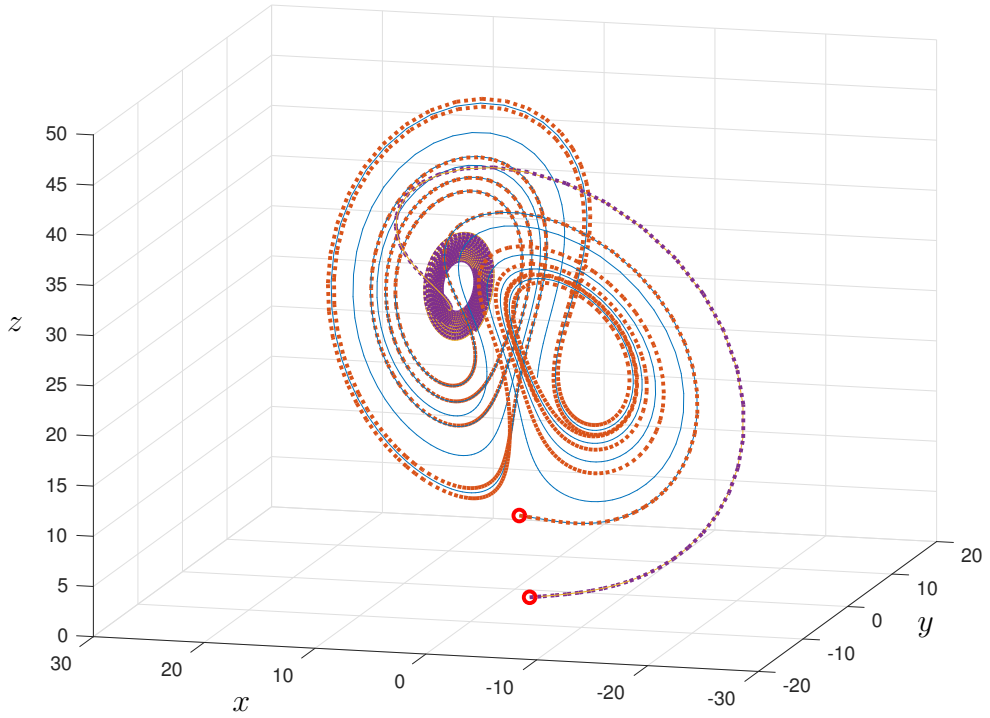

Figure 6.16: Comparison of the time evolution of the Lorenz system (solid line) with the NN prediction (dotted line) for two randomly chosen initial conditions (red dots). The NN prediction stays close to the dynamical trajectory of the Lorenz model. A more detailed comparison is given in Fig. 6.17.

NN prediction is remarkably accurate in producing an approximation to the high-accuracy simulations. This shows that the data used for training is capable of producing a high-quality nonlinear model mapping $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$. The quality of the approximation is more clearly seen in Fig. 6.17 where the time evolution of the individual components of $\mathbf{x}$ are shown against the NN predictions. See Sec. **??** for further details.

In conclusion, the NN can be trained to learn dynamics. More precisely, the NN seems to learn an algorithm which is approximately equivalent to a 4th-order Runge-Kutta scheme for advancing the solution a time-step Δt. Indeed, NNs have been used to model dynamical systems [215] and other physical processes [381] for decades. However, great strides have been made recently in using DNNs to learn Koopman embeddings, resulting in several excellent papers [550, 368, 513, 564, 412, 332]. For example, the VAMPnet architecture [550, 368] uses a time-lagged auto-encoder and a custom variational score to identify Koopman coordinates on an impressive protein folding example. In an alternative formulation, variational auto-encoders can build low-rank models that are efficient and compact representations of the Koopman operator from data [349]. By construction, the resulting network is both parsimonious and interpretable, retaining the flexibility of neural networks and

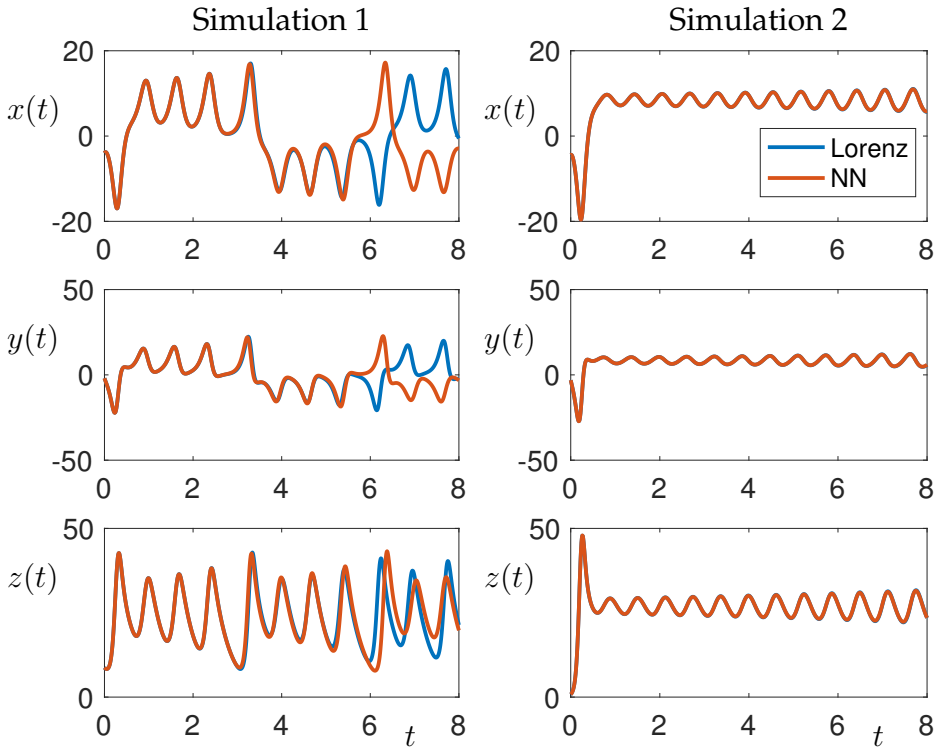

Figure 6.17: Comparison of the time evolution of the Lorenz system for two randomly chosen initial conditions (Also shown in Fig. 6.16). The left column shows that the evolution of the Lorenz differential equations and the NN mapping gives identical results until $t \approx 5.5$, at which point they diverge. In contrast, the NN prediction stays on the trajectory of the second initial condition for the entire time window.

the physical interpretation of Koopman theory. In all of these recent studies, DNN representations have been shown to be more flexible and exhibit higher accuracy than other leading methods on challenging problems.

6.7 The diversity of neural networks

There are a wide variety of NN architectures, with only a few of the most dominant architectures considered thus far. This chapter and book does not attempt to give a comprehensive assessment of the state-of-the-art in neural networks. Rather, our focus is on illustrating some of the key concepts and enabling mathematical architectures that have led NNs to a dominant position in modern data science. For a more in-depth review, please see [216]. However, to conclude this chapter, we would like to highlight some of the NN architectures that are used in practice for various data science tasks. This overview is inspired by the *neural network zoo* as highlighted by Fjodor Van Veen of the Asimov Institute

(http://www.asimovinstitute.org).

The neural network zoo highlights some of the different architectural structures around NNs. Some of the networks highlighted are commonly used across industry, while others serve niche roles for specific applications. Regardless, it demonstrates that tremendous variability and research effort focused on NNs as a core data science tool. Figure 6.18 highlights the prototype structures to be discussed in what follows. Note that the bottom panel has a key to the different type of nodes in the network, including input cells, output cells, and hidden cells. Additionally, the hidden layer NN cells can have memory effects, kernel structures and/or convolution/pooling. For each NN architecture, a brief description is given along with the original paper proposing the technique.

Perceptron

The first mathematical model of NNs by Fukushima was termed the Neocognitron in 1980 [193]. His model had a single layer with a single output cell called the perceptron, which made a categorial decision based on the sign of the output. Figure 6.2 shows this architecture to classify between dogs and cats. The perceptron is an algorithm for supervised learning of binary classifiers.

Feed forward (FF)

Feed forward networks connect the input layer to output layer by forming connections between the units so that they do not form a cycle. Figure 6.1 has already shown a version of this architecture where the information simply propagates from left to right in the network. It is often the workhorse of supervised learning where the weights are trained so as to best classify a given set of data. A feedforward network was used in Figs. 6.5 and 6.15 for training a classifier for dogs versus cats and predicting time-steps of the Lorenz attractor respectively. An important subclass of feed forward networks is *deep feed forward* (DFF) NNs. DFFs simply put together a larger number of hidden layers, typically 7-10 layers, to form the NN. A second important class of FF is the *radial basis network*, which uses radial basis functions as the activation units [88]. Like any FF network, radial basis function networks have many uses, including function approximation, time series prediction, classification, and control.

Recurrent neural network (RNN)

Illustrated in Fig. 6.18(a), RNNs are characterized by connections between units that form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence [172]. Unlike feedforward neural net-

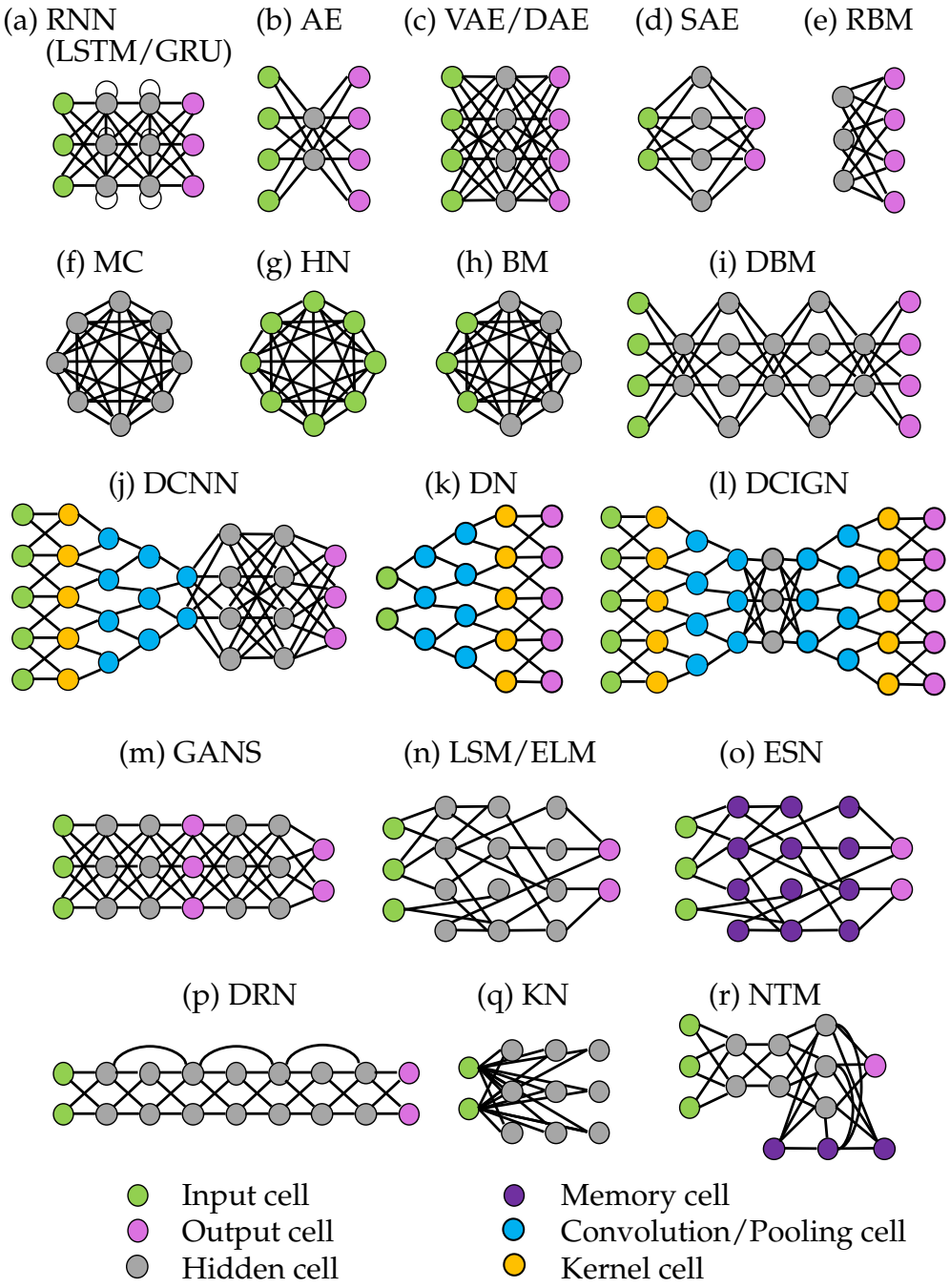

Figure 6.18: Neural network architectures commonly considered in the literature. The NNs are comprised of input nodes, output nodes, and hidden nodes. Additionally, the nodes can have memory, perform convolution and/or pooling, and perform a kernel transformation. Each network, and their acronym is explained in the text.

works, RNNs can use their internal state (memory) to process sequences of inputs. The prototypical architecture in Fig. 6.18(a) shows that each cell feeds back on itself. This self-interaction, which is not part of the FF architecture, allows for a variety of innovations. Specifically, it allows for time delays and/or feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of two key innovations: *long-short term memory* (LSTM) networks [248] and *gated recurrent units* (GRU) [132]. LSTM is of particular importance as it revolutionized speech recognition, setting a variety of performance records and outperforming traditional models in a variety of speech applications. GRUs are a variation of LSTMs which have been demonstrated to exhibit better performance on smaller datasets.

Auto encoder (AE)

The aim of an auto encoder, represented in Fig. 6.18(b), is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. For AEs, the input and output cells are matched so that the AE is essentially constructed to be a nonlinear transform into and out of a new representation, acting as an approximate identity map on the data. Thus AEs can be thought of as a generalization of linear dimensionality reduction techniques such as PCA. AEs can potentially produce nonlinear PCA representations of the data, or nonlinear manifolds on which the data should be embedded [71]. Since most data lives in nonlinear subspaces, AEs are an important class of NN for data science, with many innovations and modifications. Three important modifications of the standard AE are commonly used. The *variational auto encoder* (VAE) [290] (shown in Fig. 6.18(c)) is a popular approach to unsupervised learning of complicated distributions. By making strong assumptions concerning the distribution of latent variables, it can be trained using standard gradient descent algorithms to provide a good assessments of data in an unsupervised fashion. The *denoising auto encoder* (DAE) [541] (shown in Fig. 6.18(c)) takes a partially corrupted input during training to recover the original undistorted input. Thus noise is intentionally added to the input in order to learn the nonlinear embedding. Finally, the *sparse auto encoder* (SAE) [432] (shown in Fig. 6.18(d)) imposes sparsity on the hidden units during training, while having a larger number of hidden units than inputs, so that an autoencoder can learn useful structures in the input data. Sparsity is typically imposed by thresholding all but the few strongest hidden unit activations.

Markov chain (MC)

A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in

the previous event. So although not formally a NN, it shares many common features with RNNs. Markov chains are standard even in undergraduate probability and statistics courses. Figure 6.18(f) shows the basic architecture where each cell is connected to the other cells by a probability model for a transition.

Hopfield network (HN)

A Hopfield network is a form of a RNN which was popularized by John Hopfield in 1982 for understanding human memory [254]. Figure 6.18(g) shows the basic architecture of an all-to-all connected network where each node can act as an input cell. The network serves as a trainable content-addressable *associative* memory system with binary threshold nodes. Given an input, it is iterated on the network with a guarantee to converge to a local minimum. Sometimes it converge to a false pattern, or memory (wrong local minimum), rather than the stored pattern (expected local minimum).

Boltzmann machine (BM)

The Boltzmann machine, sometimes called a stochastic Hopfield network with hidden units, is a stochastic, generative counterpart of the Hopfield network. They were one of the first neural networks capable of learning internal representations, and are able to represent and (given sufficient time) solve difficult combinatoric problems [246]. Figure 6.18(h) shows the structure of the BM. Note that unlike Markov chains (which have no input units) or Hopfield networks (where all cells are inputs), the BM is a hybrid which has a mixture of input cells and hidden units. Boltzmann machines are intuitively appealing due to their resemblance to the dynamics of simple physical processes. They are named after the Boltzmann distribution in statistical mechanics, which is used in their sampling function.

Restricted Boltzmann machine (RBM)

Introduced under the name *Harmonium* by Paul Smolensky in 1986 [493], RBMs have been proposed for dimensionality reduction, classification, collaborative filtering, feature learning, and topic modeling. They can be trained for either supervised or unsupervised tasks. G. Hinton helped bring them to prominence by developing fast algorithms for evaluating them [397]. RBMs are a subset of BMs where restrictions are imposed on the NN such that nodes in the NN must form a bipartite graph (See Fig. 6.18(e)). Thus a pair of nodes from each of the two groups of units (commonly referred to as the "visible" and "hidden" units, respectively) may have a symmetric connection between them; there are no connections between nodes within a group. RBMs can be used in deep learn-

ing networks and deep belief networks by stacking RBMs and optionally fine-tuning the resulting deep network with gradient descent and backpropagation.

Deep belief network (DBN)

DBNs are a generative graphical model that are composed of multiple layers of latent hidden variables, with connections between the layers but not between units within each layer [52]. Figure 6.18(i) shows the architecture of the DBN. The training of the DBNs can be done stack by stack from AE or RBM layers. Thus each of these layers only has to learn to encode the previous network, which is effectively a greedy training algorithm for finding locally optimal solutions. Thus DBNs can be viewed as a composition of simple, unsupervised networks such as RBMs and AEs where each sub-network's hidden layer serves as the visible layer for the next.

Deep convolutional neural network (DCNN)

DCNNs are the workhorse of computer vision and have already been considered in this chapter. They are abstractly represented in Fig. 6.18(j), and in a more specific fashion in Fig. 6.12. Their impact and influence on computer vision cannot be overestimated. They were originally developed for document recognition [325].

Deconvolutional network (DN)

Deconvolutional Networks, shown in Fig. 6.18(k), are essentially a reverse of DCNNs [567]. The mathematical structure of DNs permit the unsupervised construction of hierarchical image representations. These representations can be used for both low-level tasks such as denoising, as well as providing features for object recognition. Each level of the hierarchy groups information from the level beneath to form more complex features that exist over a larger scale in the image. As with DCNNs, it is well suited for computer vision tasks.

Deep convolutional inverse graphics network (DCIGN)

The DCIGN is a form of a VAE that uses DCNNs for the encoding and decoding [313]. As with the AE/VAE/SAE structures, the output layer shown in Fig. 6.18(l) is constrained to match the input layer. DCIGN combine the power of DCNNs with VAEs, which provides a formative mathematical architecture for computer visions and image processing.

Generative adversarial network (GAN)

In an innovative modification of NNs, the GAN architecture of Fig. 6.18(m) trains two networks simultaneously [217]. The networks, often which are a combination of DCNNs and/or FFs, train by one of the networks generating content which the other attempts to judge. Specifically, one network generates candidates and the other evaluates them. Typically, the generative network learns to map from a latent space to a particular data distribution of interest, while the discriminative network discriminates between instances from the true data distribution and candidates produced by the generator. The generative network's training objective is to increase the error rate of the discriminative network (i.e., "fool" the discriminator network by producing novel synthesized instances that appear to have come from the true data distribution). The GAN architecture has produced interesting results in computer vision for producing synthetic data, such as images and movies.

Liquid state machine (LSM)

The LSM shown in Fig. 6.18(n) is a particular kind of spiking neural network [352]. An LSM consists of a large collection of nodes, each of which receives time varying input from external sources (the inputs) as well as from other nodes. Nodes are randomly connected to each other. The recurrent nature of the connections turns the time varying input into a spatio-temporal pattern of activations in the network nodes. The spatio-temporal patterns of activation are read out by linear discriminant units. This architecture is motivated by spiking neurons in the brain, thus helping understand how information processing and discrimination might happen using spiking neurons.

Extreme learning machine (ELM)

With the same underlying architecture of an LSM shown in Fig. 6.18(n), the ELM is a FF network for classification, regression, clustering, sparse approximation, compression and feature learning with a single layer or multiple layers of hidden nodes, where the parameters of hidden nodes (not just the weights connecting inputs to hidden nodes) need not be tuned. These hidden nodes can be randomly assigned and never updated, or can be inherited from their ancestors without being changed. In most cases, the output weights of hidden nodes are usually learned in a single step, which essentially amounts to learning a linear model [108].

Echo state network (ESN)

ESNs are RNNs with a sparsely connected hidden layer (with typically 1% connectivity). The connectivity and weights of hidden neurons have memory and are fixed and randomly assigned (See Fig. 6.18(o)). Thus like LSMs and ELMs they are not fixed into a well-ordered layered structure. The weights of output neurons can be learned so that the network can generate specific temporal patterns [263].

Deep residual network (DRN)

DRNs took the deep learning world by storm when Microsoft Research released Deep Residual Learning for Image Recognition [237]. These networks led to 1st-place winning entries in all five main tracks of the ImageNet and COCO 2015 competitions, which covered image classification, object detection, and semantic segmentation. The robustness of ResNets has since been proven by various visual recognition tasks and by non-visual tasks involving speech and language. DRNs are very deep FF networks where there are extra connections that pass from one layer to a layer two to five layers downstream. This then carries input from an earlier stage to a future stage. These networks can be 150 layers deep, which is only abstractly represented in Fig. 6.18(p).

Kohonen network (KN)

Kohonen networks are also known as self-organizing feature maps [298]. KNs use competitive learning to classify data without supervision. Input is presented to the KN as in Fig. 6.18(q), after which the network assesses which of the neurons closely match that input. These self-organizing maps differ from other NNs as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space. This makes KNs useful for low-dimensional visualization of high-dimensional data.

Neural Turing machine (NTM)

An NTM implements a NN controller coupled to an external memory resource (See Fig. 6.18(r)), which it interacts with through attentional mechanisms [219]. The memory interactions are differentiable end-to-end, making it possible to optimize them using gradient descent. An NTM with a LSTM controller can infer simple algorithms such as copying, sorting, and associative recall from input and output examples.

Suggested reading

Texts

(1) Deep learning, by I. Goodfellow, Y. Bengio and A. Courville, 2016 [216].

(2) Neural networks for pattern recognition, by C. M. Bishop, 1995 [63].

Papers and reviews

(1) Deep learning, by Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015 [324].

(2) Understanding deep convolutional networks, by S. Mallat, *Phil. Trans. R. Soc. A*, 2016 [358].

(3) Deep learning: mathematics and neuroscience, by T. Poggio, *Views & Reviews, McGovern Center for Brains, Minds and Machines*, 2016 [430].

(4) Imagenet classification with deep convolutional neural, by A. Krizhevsky, I. Sutskever and G. Hinton, *Advances in neural information processing systems*, 2012 [310].

Bibliography

[1] R. Abraham and J. E. Marsden. *Foundations of mechanics*, volume 36. Benjamin/Cummings Publishing Company Reading, Massachusetts, 1978.

[2] R. Abraham, J. E. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications*, volume 75 of *Applied Mathematical Sciences*. Springer-Verlag, 1988.

[3] M. Agrawal, S. Vidyashankar, and K. Huang. On-chip implementation of ECoG signal data decoding in brain-computer interface. In *Mixed-Signal Testing Workshop (IMSTW), 2016 IEEE 21st International*, pages 1–6. IEEE, 2016.

[4] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] H.-S. Ahn, Y. Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1099–1121, 2007.

[6] H. Akaike. Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, 21(1):243–247, 1969.

[7] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[8] W. Amrein and A.-M. Berthier. On support properties of Lp-functions and their Fourier transforms. *Journal of Functional Analysis*, 24(3):258–267, 1977.

[9] D. Amsallem, J. Cortial, and C. Farhat. On-demand cfd-based aeroelastic predictions using a database of reduced-order bases and models. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, page 800, 2009.

[10] D. Amsallem and C. Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM Journal on Scientific Computing*, 33(5):2169–2198, 2011.

[11] D. Amsallem, M. J. Zahr, and K. Washabaugh. Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction. *Advances in Computational Mathematics*, 41(5):1187–1230, 2015.

[12] J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[13] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney, et al. *LAPACK Users' guide*, volume 9. Siam, 1999.

[14] J. L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

[15] C. A. Andersson and R. Bro. The n-way toolbox for matlab. *Chemometrics and intelligent laboratory systems*, 52(1):1–4, 2000.

[16] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on image processing*, 1(2):205–220, 1992.

[17] A. C. Antoulas. *Approximation of large-scale dynamical systems*. SIAM, 2005.

[18] H. Arbabi and I. Mezić. Ergodic theory, dynamic mode decomposition and computation

of spectral properties of the Koopman operator. *SIAM J. Appl. Dyn. Syst.*, 16(4):2096–2126, 2017.

[19] K. B. Ariyur and M. Krstić. *Real-Time Optimization by Extremum-Seeking Control*. Wiley, Hoboken, New Jersey, 2003.

[20] T. Askham and J. N. Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM J. Appl. Dyn. Syst.*, 17(1):380–416, 2018.

[21] P. Astrid. Fast reduced order modeling technique for large scale LTV systems. In *American Control Conference, 2004. Proceedings of the 2004*, volume 1, pages 762–767. IEEE, 2004.

[22] K. J. Aström and R. M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[23] M. Azeez and A. Vakakis. Proper orthogonal decomposition (POD) of a class of vibroimpact oscillations. *Journal of Sound and vibration*, 240(5):859–889, 2001.

[24] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[25] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, Dec. 2007.

[26] S. Bagheri. Koopman-mode decomposition of the cylinder wake. *Journal of Fluid Mechanics*, 726:596–623, 2013.

[27] S. Bagheri, L. Brandt, and D. Henningson. Input-output analysis, model reduction and control of the flat-plate boundary layer. *J. Fluid Mechanics*, 620:263–298, 2009.

[28] S. Bagheri, J. Hoepffner, P. J. Schmid, and D. S. Henningson. Input-output analysis and control design applied to a linear model of spatially developing flows. *Appl. Mech. Rev.*, 62(2):020803–1..27, 2009.

[29] Z. Bai, S. L. Brunton, B. W. Brunton, J. N. Kutz, E. Kaiser, A. Spohn, and B. R. Noack. Data-driven methods in fluid dynamics: Sparse classification from experimental data. In *invited chapter for Whither Turbulence and Big Data in the 21st Century*, 2015.

[30] Z. Bai, E. Kaiser, J. L. Proctor, J. N. Kutz, and S. L. Brunton. Dynamic mode decomposition for compressive system identification. *arXiv preprint arXiv:1710.07737*, 2017.

[31] Z. Bai, T. Wimalajeewa, Z. Berger, G. Wang, M. Glauser, and P. K. Varshney. Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA Journal*, 53(4):920–933, 2014.

[32] M. J. Balajewicz, E. H. Dowell, and B. R. Noack. Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier–Stokes equation. *Journal of Fluid Mechanics*, 729:285–308, 2013.

[33] M. Balasubramanian, S. Zabic, C. Bowd, H. W. Thompson, P. Wolenski, S. S. Iyengar, B. B. Karki, and L. M. Zangwill. A framework for detecting glaucomatous progression in the optic nerve head of an eye using proper orthogonal decomposition. *IEEE Transactions on Information Technology in Biomedicine*, 13(5):781–793, 2009.

[34] B. Bamieh and L. Giarré. Identification of linear parameter varying models. *International Journal of Robust and Nonlinear Control*, 12:841–853, 2002.

[35] A. Banaszuk, K. B. Ariyur, M. Krstić, and C. A. Jacobson. An adaptive algorithm for control of combustion instability. *Automatica*, 40(11):1965–1972, 2004.

[36] A. Banaszuk, S. Narayanan, and Y. Zhang. Adaptive control of flow separation in a planar diffuser. *AIAA paper*, 617:2003, 2003.

[37] A. Banaszuk, Y. Zhang, and C. A. Jacobson. Adaptive control of combustion instability using extremum-seeking. In *American Control Conference, 2000. Proceedings of the 2000*, volume 1, pages 416–422. IEEE, 2000.

[38] S. Banks. Infinite-dimensional Carleman linearization, the Lie series and optimal control of non-linear partial differential equations. *International journal of systems science*, 23(5):663–675, 1992.

[39] R. G. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–120, 2007.

[40] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[41] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[42] J. Basley, L. R. Pastur, N. Delprat, and F. Lusseyran. Space-time aspects of a three-dimensional multi-modulated open cavity flow. *Physics of Fluids (1994-present)*, 25(6):064105, 2013.

[43] J. Basley, L. R. Pastur, F. Lusseyran, T. M. Faure, and N. Delprat. Experimental investigation of global structures in an incompressible cavity flow using time-resolved PIV. *Experiments in Fluids*, 50(4):905–918, 2011.

[44] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.

[45] J. F. Beaudoin, O. Cadot, J. L. Aider, and J. E. Wesfreid. Bluff-body drag reduction by extremum-seeking control. *Journal of Fluids and Structures*, 22:973–978, 2006.

[46] J.-F. Beaudoin, O. Cadot, J.-L. Aider, and J.-E. Wesfreid. Drag reduction of a bluff body using adaptive control methods. *Physics of Fluids*, 18(8):085107, 2006.

[47] R. Becker, R. King, R. Petz, and W. Nitsche. Adaptive closed-loop control on a high-lift configuration using extremum seeking. *AIAA Journal*, 45(6):1382–92, 2007.

[48] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, 1997.

[49] G. Bellani. Experimental studies of complex flows through image-based techniques. 2011.

[50] B. A. Belson, J. H. Tu, and C. W. Rowley. Algorithm 945: modred—a parallelized model reduction library. *ACM Transactions on Mathematical Software*, 40(4):30, 2014.

[51] M. Benedicks. On Fourier transforms of functions supported on sets of finite Lebesgue measure. *Journal of mathematical analysis and applications*, 106(1):180–183, 1985.

[52] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[53] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox. *Model Reduction and Approximation: Theory and Algorithms*, volume 15. SIAM, 2017.

[54] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015.

[55] P. Benner, J.-R. Li, and T. Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numerical Linear Algebra with Applications*, 15(9):755–777, 2008.

[56] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. B. Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Journal of Advanced Robotics*, 29(5):331–343, 2015.

[57] G. Berkooz, P. Holmes, and J. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Ann. Rev. Fluid Mech.*, 25:539–575, 1993.

[58] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):141–183, 1991.

[59] S. A. Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[60] P. Binetti, K. B. Ariyur, M. Krstić, and F. Bernelli. Formation flight optimization using extremum seeking feedback. *Journal of Guidance, Control, and Dynamics*, 26(1):132–142, 2003.

[61] G. D. Birkhoff. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.

[62] G. D. Birkhoff and B. O. Koopman. Recent contributions to the ergodic theory. *Proceedings of the National Academy of Sciences*, 18(3):279–282, 1932.
[63] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.
[64] C. M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.
[65] D. Bistrian and I. Navon. Randomized dynamic mode decomposition for non-intrusive reduced order modelling. *International Journal for Numerical Methods in Engineering*, 2016.
[66] D. A. Bistrian and I. M. Navon. An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs POD. *International Journal for Numerical Methods in Fluids*, 2015.
[67] P. Bondi, G. Casalino, and L. Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal on Robotics and Automation*, 4(1):14–22, 1988.
[68] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.
[69] J. L. Borges. The library of Babel. *Collected fictions*, 1998.
[70] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.
[71] H. Boulard and Y. Kamp. Autoassociative memory by multilayer perceptron and singular values decomposition. *Biol Cybern*, 59:291–294, 1989.
[72] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[73] S. Boyd, L. O. Chua, and C. A. Desoer. Analytical foundations of volterra series. *IMA Journal of Mathematical Control and Information*, 1(3):243–282, 1984.
[74] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2009.
[75] A. I. Bratcu, I. Munteanu, S. Bacha, and B. Raison. Maximum power point tracking of grid-connected photovoltaic arrays by using extremum seeking control. *CEAI*, 10(4):3–12, 2008.
[76] L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.
[77] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[78] L. Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.
[79] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.
[80] I. Bright, G. Lin, and J. N. Kutz. Compressive sensing and machine learning strategies for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(127102):1–15, 2013.
[81] I. Bright, G. Lin, and J. N. Kutz. Classification of spatio-temporal data via asynchronous sparse sampling: Application to flow around a cylinder. *SIAM Multiscale modeling and simulation*, 14(2):823–838, 2016.
[82] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.
[83] D. Bristow, M. Tharayil, A. G. Alleyne, et al. A survey of iterative learning control. *Control Systems, IEEE*, 26(3):96–114, 2006.
[84] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.
[85] A. Broad, T. Murphey, and B. Argall. Learning models for shared control of human-machine systems with unknown dynamics. *Robotics: Science and Systems Proceedings*, 2017.
[86] R. W. Brockett. Volterra series and geometric control theory. *Automatica*, 12(2):167–176, 1976.

[87] D. Broomhead and R. Jones. Time-series analysis. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 423, pages 103–121. The Royal Society, 1989.

[88] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[89] B. W. Brunton, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse sensor placement optimization for classification. *SIAM Journal on Applied Mathematics*, 76(5):2099–2122, 2016.

[90] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods*, 258:1–15, 2016.

[91] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(19):1–9, 2017.

[92] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLoS ONE*, 11(2):e0150171, 2016.

[93] S. L. Brunton, X. Fu, and J. N. Kutz. Extremum-seeking control of a mode-locked laser. *IEEE Journal of Quantum Electronics*, 49(10):852–861, 2013.

[94] S. L. Brunton, X. Fu, and J. N. Kutz. Self-tuning fiber lasers. *IEEE Journal of Selected Topics in Quantum Electronics*, 20(5), 2014.

[95] S. L. Brunton and B. R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67:050801–1–050801–48, 2015.

[96] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[97] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC NOLCOS*, 49(18):710–715, 2016.

[98] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz. Compressed sensing and dynamic mode decomposition. *Journal of Computational Dynamics*, 2(2):165–191, 2015.

[99] S. L. Brunton and C. W. Rowley. Maximum power point tracking for photovoltaic optimization using ripple-based extremum seeking control. *IEEE Transactions on Power Electronics*, 25(10):2531–2540, 2010.

[100] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1716–1732, 2014.

[101] D. Buche, P. Stoll, R. Dornberger, and P. Koumoutsakos. Multiobjective evolutionary algorithm for the optimization of noisy combustion processes. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4):460–473, 2002.

[102] M. Budišić and I. Mezić. An approximate parametrization of the ergodic partition using time averaged observables. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 3162–3168. IEEE, 2009.

[103] M. Budišić and I. Mezić. Geometry of the ergodic quotient reveals coherent structures in flows. *Physica D: Nonlinear Phenomena*, 241(15):1255–1269, 2012.

[104] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanism a). *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[105] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[106] P. A. Businger and G. H. Golub. Algorithm 358: Singular value decomposition of a complex matrix [f1, 4, 5]. *Communications of the ACM*, 12(10):564–565, 1969.

[107] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.
[108] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C. M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, et al. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28(6):30–59, 2013.
[109] E. J. Candès. Compressive sensing. *Proceedings of the International Congress of Mathematics*, 2006.
[110] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11–1–11–37, 2011.
[111] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.
[112] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications in Pure and Applied Mathematics*, 8(1207–1223), 59.
[113] E. J. Candes and T. Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.
[114] E. J. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.
[115] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, pages 21–30, 2008.
[116] Y. Cao, J. Zhu, Z. Luo, and I. Navon. Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal decomposition. *Computers & Mathematics with Applications*, 52(8):1373–1386, 2006.
[117] Y. Cao, J. Zhu, I. M. Navon, and Z. Luo. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 53(10):1571–1583, 2007.
[118] K. Carlberg, M. Barone, and H. Antil. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.
[119] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.
[120] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.
[121] T. Carleman. Application de la théorie des équations intégrales linéaires aux systémes d'équations différentielles non linéaires. *Acta Mathematica*, 59(1):63–87, 1932.
[122] T. Carleman. Sur la théorie de l'équation intégrodifférentielle de boltzmann. *Acta Mathematica*, 60(1):91–146, 1933.
[123] T. Carleman. Sur les systemes lineaires aux dérivées partielles du premier ordrea deux variables. *CR Acad. Sci. Paris*, 197:471–474, 1933.
[124] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35:283–319, 1970.
[125] R. Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011, 2011.
[126] A. Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, 78(7):808–817, 2000.
[127] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[128] K. K. Chen and C. W. Rowley. Normalized coprime robust stability and performance guarantees for reduced-order controllers. *IEEE Transactions on Automatic Control*, 58(4):1068–1073, 2013.

[129] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of dynamic mode decomposition: Boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012.

[130] Y. Chen, K. L. Moore, and H.-S. Ahn. Iterative learning control. In *Encyclopedia of the Sciences of Learning*, pages 1648–1652. Springer, 2012.

[131] S. Cherry. Singular value decomposition analysis and canonical correlation analysis. *Journal of Climate*, 9(9):2003–2009, 1996.

[132] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[133] J. Choi, M. Krstić, K. Ariyur, and J. Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on Automatic Control*, 47(2):318–323, FEB 2002.

[134] Y. Choi, D. Amsallem, and C. Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *arXiv preprint arXiv:1506.07849*, 2015.

[135] T. Colonius and K. Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197:2131–2146, 2008.

[136] J. W. Cooley, P. A. Lewis, and P. D. Welch. Historical notes on the fast Fourier transform. *Proceedings of the IEEE*, 55(10):1675–1677, 1967.

[137] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[138] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[139] M. C. Cross and P. C. Hohenberg. Pattern formation outside of equilibrium. *Reviews of modern physics*, 65(3):851, 1993.

[140] J. P. Crutchfield and B. S. McNamara. Equations of motion from a data series. *Complex systems*, 1:417–452, 1987.

[141] M. Dam, M. Brøns, J. Juul Rasmussen, V. Naulin, and J. S. Hesthaven. Sparse identification of a predator-prey system from simulation data of a convection model. *Physics of Plasmas*, 24(2):022310, 2017.

[142] B. C. Daniels and I. Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature communications*, 6, 2015.

[143] B. C. Daniels and I. Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015.

[144] S. Das and D. Giannakis. Delay-coordinate maps and the spectra of Koopman operators. *arXiv preprint arXiv:1706.08544*, 2017.

[145] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE transactions on information theory*, 36(5):961–1005, 1990.

[146] L. Davis et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

[147] S. T. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):1–19, 2016.

[148] 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 (methodological)*, pages 1–38, 1977.

[149] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *Automatic Control, IEEE Transactions on*, 41(7):930–942, 1996.

[150] D. Donoho. 50 years of data science. In *Based on a Presentation at the Tukey Centennial Workshop*. NJ Princeton, 2015.
[151] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
[152] D. L. Donoho and M. Gavish. Code supplement to "The optimal hard threshold for singular values is $4/\sqrt{3}$". http://purl.stanford.edu/vg705qn9070, 2014.
[153] D. L. Donoho, I. M. Johnstone, J. C. Hoch, and A. S. Stern. Maximum entropy and the nearly black object. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 41–81, 1992.
[154] D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.
[155] J. C. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978.
[156] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.
[157] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis. State-space solutions to standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847, 1989.
[158] P. Drineas and M. W. Mahoney. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear algebra and its applications*, 420(2-3):553–571, 2007.
[159] Z. Drmac and S. Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.
[160] Q. Du and M. Gunzburger. Model reduction by proper orthogonal decomposition coupled with centroidal voronoi tessellations (keynote). In *ASME 2002 Joint US-European Fluids Engineering Division Conference*, pages 1401–1406. American Society of Mechanical Engineers, 2002.
[161] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.
[162] J. A. Duersch and M. Gu. Randomized QR with column pivoting. *SIAM Journal on Scientific Computing*, 39(4):C263–C291, 2017.
[163] D. Duke, D. Honnery, and J. Soria. Experimental investigation of nonlinear instabilities in annular liquid sheets. *Journal of Fluid Mechanics*, 691:594–604, 2012.
[164] D. Duke, J. Soria, and D. Honnery. An error analysis of the dynamic mode decomposition. *Experiments in fluids*, 52(2):529–542, 2012.
[165] G. E. Dullerud and F. Paganini. *A course in robust control theory: A convex approach*. Texts in Applied Mathematics. Springer, Berlin, Heidelberg, 2000.
[166] R. Dunne and B. J. McKeon. Dynamic stall on a pitching and surging airfoil. *Experiments in Fluids*, 56(8):1–15, 2015.
[167] T. Duriez, S. L. Brunton, and B. R. Noack. *Machine Learning Control: Taming Nonlinear Dynamics and Turbulence*. Springer, 2016.
[168] T. Duriez, V. Parezanović, L. Cordier, B. R. Noack, J. Delville, J.-P. Bonnet, M. Segond, and M. Abel. Closed-loop turbulence control using machine learning. *arXiv preprint arXiv:1404.4589*, 2014.
[169] T. Duriez, V. Parezanovic, J.-C. Laurentie, C. Fourment, J. Delville, J.-P. Bonnet, L. Cordier, B. R. Noack, M. Segond, M. Abel, N. Gautier, J.-L. Aider, C. Raibaudo, C. Cuvier, M. Stanislas, and S. L. Brunton. Closed-loop control of experimental shear flows using machine learning. AIAA Paper 2014-2219, 7th Flow Control Conference, 2014.
[170] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
[171] J. L. Eftang, A. T. Patera, and E. M. Rønquist. An" hp" certified reduced basis method for

parametrized elliptic partial differential equations. *SIAM Journal on Scientific Computing*, 32(6):3170–3200, 2010.
[172] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
[173] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.
[174] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Compressed dynamic mode decomposition for real-time object detection. *Journal of Real-Time Image Processing*, 2016.
[175] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Randomized dynamic mode decomposition. *arXiv preprint arXiv:1702.02912*, 2017.
[176] N. B. Erichson, K. Manohar, S. L. Brunton, and J. N. Kutz. Randomized CP tensor decomposition. *arXiv preprint arXiv:1703.09074*.
[177] N. B. Erichson, S. Voronin, S. L. Brunton, and J. N. Kutz. Randomized matrix decompositions using R. *arXiv preprint arXiv:1608.02148*, 2016.
[178] T. Esram, J. W. Kimball, P. T Krein, P. L. Chapman, and P. Midya. Dynamic maximum power point tracking of photovoltaic arrays using ripple correlation control. *Ieee Transactions On Power Electronics*, 21(5):1282–1291, Sept. 2006.
[179] R. Everson and L. Sirovich. Karhunen–Loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664, 1995.
[180] N. Fabbiane, O. Semeraro, S. Bagheri, and D. S. Henningson. Adaptive and model-based control theory applied to convectively unstable flows. *Appl. Mech. Rev.*, 66(6):060801–1–20, 2014.
[181] B. Feeny. On proper orthogonal co-ordinates as indicators of modal activity. *Journal of Sound and Vibration*, 255(5):805–817, 2002.
[182] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368, 1922.
[183] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.
[184] P. J. Fleming and R. C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10:1223–1241, 2002.
[185] J. Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822.
[186] J. B. J. Fourier. *The analytical theory of heat*. The University Press, 1878.
[187] J. E. Fowler. Compressive-projection principal component analysis. *IEEE Transactions on Image Processing*, 18(10):2230–2242, 2009.
[188] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
[189] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
[190] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004.
[191] X. Fu, S. L. Brunton, and J. Nathan Kutz. Classification of birefringence in mode-locked fiber lasers using machine learning and sparse representation. *Optics express*, 22(7):8585–8597, 2014.
[192] K. Fukagata, S. Kern, P. Chatelain, P. Koumoutsakos, and N. Kasagi. Evolutionary optimization of an anisotropic compliant surface for turbulent friction drag reduction. *Journal of Turbulence*, 9(35):1–17, 2008.
[193] F. Fukushima. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetic*, 36:193–202, 1980.
[194] H. Gao, J. Lam, C. Wang, and Y. Wang. Delay-dependent output-feedback stabilisation of

discrete-time systems with time-varying state delay. *IEE Proceedings-Control Theory and Applications*, 151(6):691–698, 2004.
[195] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.
[196] J. L. Garriga and M. Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010.
[197] C. Gauss. Nachlass: Theoria interpolationis methodo nova tractata, volume werke. *Königliche Gesellschaft der Wissenschaften, Göttingen*, 1866.
[198] C.-F. Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*, volume 1. Henricus Dieterich, 1823.
[199] N. Gautier, J.-L. Aider, T. Duriez, B. Noack, M. Segond, and M. Abel. Closed-loop separation control using machine learning. *Journal of Fluid Mechanics*, 770:442–457, 2015.
[200] M. Gavish and D. L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.
[201] M. Gazzola, O. V. Vasilyev, and P. Koumoutsakos. Shape optimization for drag reduction in linked bodies using evolution strategies. *Computers & Structures*, 89(11):1224–1231, 2011.
[202] G. Gelbert, J. P. Moeck, C. O. Paschereit, and R. King. Advanced algorithms for gradient estimation in one-and two-parameter extremum seeking controllers. *Journal of Process Control*, 22(4):700–709, 2012.
[203] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):643–660, 2001.
[204] J. J. Gerbrands. On the relationships between SVD, KLT and PCA. *Pattern recognition*, 14(1):375–381, 1981.
[205] A. C. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.
[206] A. C. Gilbert, J. Y. Park, and M. B. Wakin. Sketched SVD: Recovering spectral features from compressive measurements. *ArXiv e-prints*, 2012.
[207] A. C. Gilbert, M. J. Strauss, and J. A. Tropp. A tutorial on fast Fourier sampling. *IEEE Signal Processing Magazine*, pages 57–66, 2008.
[208] B. Glaz, L. Liu, and P. P. Friedmann. Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework. *AIAA journal*, 48(10):2418–2429, 2010.
[209] P. J. Goddard and K. Glover. Controller approximation: approaches for preserving H_∞ performance. *IEEE Transactions on Automatic Control*, 43(7):858–871, 1998.
[210] D. E. Goldberg. *Genetic algorithms*. Pearson Education India, 2006.
[211] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.
[212] G. Golub, S. Nash, and C. Van Loan. A Hessenberg-Schur method for the problem $ax + xb = c$. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979.
[213] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerical Mathematics*, 14:403–420, 1970.
[214] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
[215] R. Gonzalez-Garcia, R. Rico-Martinez, and I. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Comp. & Chem. Eng.*, 22:S965–S968, 1998.
[216] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.
[217] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville,

and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[218] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[219] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[220] A. Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[221] M. S. Grewal. Kalman filtering. In *International Encyclopedia of Statistical Science*, pages 705–708. Springer, 2011.

[222] M. Grilli, P. J. Schmid, S. Hickel, and N. A. Adams. Analysis of unsteady behaviour in shockwave turbulent boundary layer interaction. *Journal of Fluid Mechanics*, 700:16–28, 2012.

[223] J. Grosek and J. N. Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.

[224] M. Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):1139–1173, 2015.

[225] F. Gueniat, L. Mathelin, and L. Pastur. A dynamic mode decomposition approach for large and arbitrarily sampled systems. *Physics of Fluids*, 27(2):025113, 2015.

[226] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002.

[227] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[228] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific Computing*, 33:2580–2594, 2011.

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

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

[231] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal of Numerical Analysis*, 2(3):303–323, 1982.

[232] S. Han and B. Feeny. Application of proper orthogonal decomposition to structural vibration analysis. *Mechanical Systems and Signal Processing*, 17(5):989–1001, 2003.

[233] N. Hansen, A. S. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.

[234] D. Harrison Jr and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[235] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA working papers in phonetics*, 16:1–84, 1970. Available at `http://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf`.

[236] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[237] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[238] M. Heath, A. Laub, C. Paige, and R. Ward. Computing the singular value decomposi-

tion of a product of two matrices. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1147–1159, 1986.

[239] M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, 1984.

[240] W. Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. In *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pages 478–504. Springer, 1985.

[241] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis. *Theoretical and Computational Fluid Dynamics*, 31(4):349–368, 2017.

[242] M. S. Hemati, M. O. Williams, and C. W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids (1994-present)*, 26(11):111701, 2014.

[243] K. K. Herrity, A. C. Gilbert, and J. A. Tropp. Sparse approximation via iterative thresholding. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, pages III–III. IEEE, 2006.

[244] J. S. Hesthaven, G. Rozza, and B. Stamm. Certified reduced basis methods for parametrized partial differential equations. *SpringerBriefs in Mathematics*, 2015.

[245] T. Hey, S. Tansley, K. M. Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[246] G. E. Hinton and T. J. Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[247] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. In *Proceedings of the 3rd Annual Allerton Conference on Circuit and System Theory*, pages 449–459, 1965.

[248] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[249] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[250] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[251] P. Holmes and J. Guckenheimer. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, Heidelberg, 1983.

[252] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge, 2nd paperback edition, 2012.

[253] E. Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. *Communications on Pure and Applied mathematics*, 3(3):201–230, 1950.

[254] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[255] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[256] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, Sept. 1933.

[257] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, Oct. 1933.

[258] C. Huang, W. E. Anderson, M. E. Harvazinski, and V. Sankaran. Analysis of self-excited combustion instabilities using decomposition techniques. In *51st AIAA Aerospace Sciences Meeting*, pages 1–18, 2013.

[259] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[260] P. J. Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.
[261] S. J. Illingworth, A. S. Morgans, and C. W. Rowley. Feedback control of flow resonances using balanced reduced-order models. *Journal of Sound and Vibration*, 330(8):1567–1581, 2010.
[262] E. Jacobsen and R. Lyons. The sliding DFT. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.
[263] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.
[264] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*. Springer, 2013.
[265] M. C. Johnson, S. L. Brunton, N. B. Kundtz, and J. N. Kutz. Extremum-seeking control of a beam pattern of a reconfigurable holographic metamaterial antenna. *Journal of the Optical Society of America A*, 33(1):59–68, 2016.
[266] R. A. Johnson and D. Wichern. *Multivariate analysis*. Wiley Online Library, 2002.
[267] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.
[268] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.
[269] S. Joshi and S. Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2009.
[270] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.
[271] J. N. Juang. *Applied System Identification*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1994.
[272] J. N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985.
[273] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman. Identification of observer/Kalman filter Markov parameters: Theory and experiments. Technical Memorandum 104069, NASA, 1991.
[274] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, volume 3, pages 182–193. Orlando, FL, 1997.
[275] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.
[276] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *arXiv preprint arXiv:1707.01146*, 2017.
[277] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society of London A*, 474(2219), 2018.
[278] E. Kaiser, B. R. Noack, L. Cordier, A. Spohn, M. Segond, M. Abel, G. Daviller, J. Östh, S. Krajnović, and R. K. Niven. Cluster-based reduced-order modelling of a mixing layer. *Journal of Fluid Mechanics*, 754:365–414, 2014.
[279] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.
[280] K. Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung, vol. 37. *Annales AcademiæScientiarum Fennicæ, Ser. A. I*, 1947.
[281] K. Kasper, L. Mathelin, and H. Abou-Kandil. A machine learning approach for constrained sensor placement. In *American Control Conference (ACC), 2015*, pages 4479–4484. IEEE, 2015.
[282] A. K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM*

Journal on Scientific Computing, 26(4):1214–1233, 2005.
[283] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.
[284] A. R. Kellems, S. Chaturantabut, D. C. Sorensen, and S. J. Cox. Morphologically accurate reduced order modeling of spiking neurons. *Journal of computational neuroscience*, 28(3):477–494, 2010.
[285] J. Kepler. *Tabulae Rudolphinae, quibus Astronomicae scientiae, temporum longinquitate collapsae Restauratio continetur*. Ulm: Jonas Saur, 1627.
[286] G. Kerschen and J.-C. Golinval. Physical interpretation of the proper orthogonal modes using the singular value decomposition. *Journal of Sound and Vibration*, 249(5):849–865, 2002.
[287] G. Kerschen, J.-c. Golinval, A. F. Vakakis, and L. A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics*, 41(1-3):147–169, 2005.
[288] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Communications in Mathematical Science*, 1(4):715–762, 2003.
[289] N. J. Killingsworth and M. Krstc. PID tuning using extremum seeking: online, model-free performance optimization. *IEEE Control Systems Magazine*, February:70–79, 2006.
[290] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[291] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(1):103–108, 1990.
[292] V. C. Klema and A. J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980.
[293] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, and F. Noé. Data-driven model reduction and transfer operator approximation. *Journal of Nonlinear Science*, pages 1–26, 2018.
[294] R. Koch. *The 80/20 Principle*. Nicholas Brealey Publishing, 1997.
[295] R. Koch. *Living the 80/20 way*. Audio-Tech Business Book Summaries, Incorporated, 2006.
[296] R. Koch. *The 80/20 principle: the secret to achieving more with less*. Crown Business, 2011.
[297] R. Koch. *The 80/20 principle and 92 other powerful laws of nature: the science of success*. Nicholas Brealey Publishing, 2013.
[298] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.
[299] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.
[300] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.
[301] B. O. Koopman and J.-v. Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences of the United States of America*, 18(3):255, 1932.
[302] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93(149–160), 2018.
[303] M. Korda and I. Mezić. On convergence of extended dynamic mode decomposition to the Koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.
[304] P. Koumoutsakos, J. Freund, and D. Parekh. Evolution strategies for automatic optimization of jet mixing. *AIAA journal*, 39(5):967–969, 2001.
[305] K. Kowalski, W.-H. Steeb, and K. Kowalksi. *Nonlinear dynamical systems and Carleman linearization*. World Scientific, 1991.
[306] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selec-*

tion, volume 1. MIT press, 1992.
[307] J. R. Koza, F. H. Bennett III, and O. Stiffelman. Genetic programming as a darwinian invention machine. In *Genetic Programming*, pages 93–108. Springer, 1999.
[308] B. Kramer, P. Grover, P. Boufounos, M. Benosman, and S. Nabi. Sparse sensing and dmd based identification of flow regimes and bifurcations in complex flows. *SIAM J. Appl. Dyn. Syst.*, 16(2):1164–1196, 2017.
[309] J. P. Krieger and M. Krstic. Extremum seeking based on atmospheric turbulence for aircraft endurance. *Journal of Guidance, Control, and Dynamics*, 34(6):1876–1885, 2011.
[310] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[311] M. Krstic, A. Krupadanam, and C. Jacobson. Self-tuning control of a nonlinear model of combustion instabilities. *IEEE Tr. Contr. Syst. Technol.*, 7(4):424–436, 1999.
[312] M. Krstić and H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36:595–601, 2000.
[313] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.
[314] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
[315] K. Kunisch and S. Volkwein. Optimal snapshot location for computing pod basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(3):509–529, 2010.
[316] J. N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press, 2013.
[317] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.
[318] J. N. Kutz, X. Fu, and S. L. Brunton. Multi-resolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.
[319] J. N. Kutz, S. Sargsyan, and S. L. Brunton. Leveraging sparsity and compressive sensing for reduced order modeling. In *Model Reduction of Parametrized Systems*, pages 301–315. Springer, 2017.
[320] S. Lall, J. E. Marsden, and S. Glavaški. Empirical model reduction of controlled nonlinear systems. In *IFAC World Congress*, volume F, pages 473–478. International Federation of Automatic Control, 1999.
[321] S. Lall, J. E. Marsden, and S. Glavaški. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control*, 12(6):519–535, 2002.
[322] Y. Lan and I. Mezić. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013.
[323] A. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on automatic control*, 24(6):913–921, 1979.
[324] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.
[325] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[326] J. H. Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424, 2011.
[327] K. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(5):684–698, 2005.
[328] A. M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[329] V. Lenaerts, G. Kerschen, and J.-C. Golinval. Proper orthogonal decomposition for model updating of non-linear mechanical systems. *Mechanical Systems and Signal Processing*, 15(1):31–43, 2001.
[330] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.
[331] R. Leyva, C. Alonso, I. Queinnec, A. Cid-Pastor, D. Lagrange, and L. Martinez-Salamero. MPPT of photovoltaic systems using extremum-seeking control. *Ieee Transactions On Aerospace and Electronic Systems*, 42(1):249–258, Jan. 2006.
[332] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.
[333] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, and C. Wu. Proper orthogonal decomposition and its applications- part i: Theory. *Journal of Sound and vibration*, 252(3):527–544, 2002.
[334] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–588. ACM, 2013.
[335] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104:20167–20172, 2007.
[336] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
[337] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.
[338] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999.
[339] S. Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.
[340] M. Loeve. *Probability Theory*. Van Nostrand, Princeton, NJ, 1955.
[341] J.-C. Loiseau and S. L. Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.
[342] J.-C. Loiseau, B. R. Noack, and S. L. Brunton. Sparse reduced-order modeling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.
[343] R. W. Longman. Iterative learning control and repetitive control for engineering practice. *International journal of control*, 73(10):930–954, 2000.
[344] E. N. Lorenz. Empirical orthogonal functions and statistical weather prediction. Technical report, Massachusetts Institute of Technology, Dec. 1956.
[345] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.
[346] D. M. Luchtenburg and C. W. Rowley. Model reduction using snapshot-based realizations. *Bulletin of the American Physical Society*, 56, 2011.
[347] J. Lumley. Toward a turbulent constitutive relation. *Journal of Fluid Mechanics*, 41(02):413–434, 1970.
[348] B. Lusch, E. C. Chi, and J. N. Kutz. Shape constrained tensor decompositions using sparse representations in over-complete libraries. *arXiv preprint arXiv:1608.04674*, 2016.
[349] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018.
[350] F. Lusseyran, F. Gueniat, J. Basley, C. L. Douay, L. R. Pastur, T. M. Faure, and P. J. Schmid. Flow coherent structures and frequency signature: application of the dynamic modes decomposition to open cavity flow. In *Journal of Physics: Conference Series*, volume 318, page 042036. IOP Publishing, 2011.

[351] Z. Ma, S. Ahuja, and C. W. Rowley. Reduced order models for control of fluids using the eigensystem realization algorithm. *Theor. Comput. Fluid Dyn.*, 25(1):233–247, 2011.
[352] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.
[353] A. Mackey, H. Schaeffer, and S. Osher. On the compressive spectral method. *Multiscale Modeling & Simulation*, 12(4):1800–1827, 2014.
[354] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3:123–224, 2011.
[355] A. J. Majda and J. Harlim. Physics constrained nonlinear regression models for time series. *Nonlinearity*, 26(1):201, 2012.
[356] A. J. Majda and Y. Lee. Conceptual dynamical models for turbulence. *Proceedings of the National Academy of Sciences*, 111(18):6548–6553, 2014.
[357] S. Mallat. *A wavelet tour of signal processing*. Academic press, 1999.
[358] S. Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.
[359] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.
[360] J. Mandel. Use of the singular value decomposition in regression analysis. *The American Statistician*, 36(1):15–24, 1982.
[361] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, 2(1):52–63, 2016.
[362] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A*, 473(2204):1–16, 2017.
[363] J. Mann and J. N. Kutz. Dynamic mode decomposition for financial trading strategies. *Quantitative Finance*, pages 1–13, 2016.
[364] K. Manohar, B. W. Brunton, J. N. Kutz, and S. L. Brunton. Data-driven sparse sensor placement. *Invited for IEEE Control Systems Magazine*, 2017.
[365] K. Manohar, S. L. Brunton, and J. N. Kutz. Environmental identification in flight using sparse approximation of wing strain. *Journal of Fluids and Structures*, 70:162–180, 2017.
[366] K. Manohar, E. Kaiser, S. L. Brunton, and J. N. Kutz. Optimized sampling for multiscale dynamics. *SIAM Multiscale modeling and simulation*, 17(1):117–136, 2019.
[367] K. Manohar, J. N. Kutz, and S. L. Brunton. Optimized sensor and actuator placement for balanced models. *arXiv preprint arXiv:1812.01574*, 2018.
[368] A. Mardt, L. Pasquali, H. Wu, and F. Noé. VAMPnets: Deep learning of molecular kinetics. *Nature Communications*, 9(5), 2018.
[369] J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*. Springer-Verlag, 2nd edition, 1999.
[370] P.-G. Martinsson. Randomized methods for matrix computations and analysis of high dimensional data. *arXiv preprint arXiv:1607.01649*, 2016.
[371] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30:47–68, 2011.
[372] J. L. Maryak, J. C. Spall, and B. D. Heydon. Use of the Kalman filter for inference in state-space models with unknown noise distributions. *IEEE Transactions on Automatic Control*, 49(1):87–90, 2004.
[373] L. Massa, R. Kumar, and P. Ravindran. Dynamic mode decomposition analysis of detonation waves. *Physics of Fluids (1994-present)*, 24(6):066101, 2012.
[374] L. Mathelin, K. Kasper, and H. Abou-Kandil. Observable dictionary learning for high-dimensional statistical inference. *Archives of Computational Methods in Engineering*,

25(1):103–120, 2018.
[375] R. Maury, M. Keonig, L. Cattafesta, P. Jordan, and J. Delville. Extremum-seeking control of jet noise. *Aeroacoustics*, 11(3&4):459–474, 2012.
[376] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.
[377] I. Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Ann. Rev. Fluid Mech.*, 45:357–378, 2013.
[378] I. Mezić. *Spectral operator methods in dynamical systems: Theory and applications*. Springer, 2017.
[379] I. Mezić and A. Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1):101–133, 2004.
[380] I. Mezić and S. Wiggins. A method for visualization of invariant sets of dynamical systems based on the ergodic partition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(1):213–218, 1999.
[381] M. Milano and P. Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.
[382] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
[383] Y. Mizuno, D. Duke, C. Atkinson, and J. Soria. Investigation of wall-bounded turbulent flow using dynamic mode decomposition. In *Journal of Physics: Conference Series*, volume 318, page 042040. IOP Publishing, 2011.
[384] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.
[385] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
[386] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
[387] J. P. Moeck, J.-F. Bourgouin, D. Durox, T. Schuller, and S. Candel. Tomographic reconstruction of heat release rate perturbations induced by helical modes in turbulent swirl flames. *Experiments in Fluids*, 54(4):1–17, 2013.
[388] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, AC-26(1):17–32, 1981.
[389] C. C. Moore. Ergodic theorem, ergodic theory, and statistical mechanics. *Proceedings of the National Academy of Sciences*, 112(7):1907–1911, 2015.
[390] K. L. Moore. *Iterative learning control for deterministic systems*. Springer Science & Business Media, 2012.
[391] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.
[392] T. W. Muld, G. Efraimsson, and D. S. Henningson. Flow structures around a high-speed train extracted using proper orthogonal decomposition and dynamic mode decomposition. *Computers & Fluids*, 57:87–97, 2012.
[393] T. W. Muld, G. Efraimsson, and D. S. Henningson. Mode decomposition on surface-mounted cube. *Flow, Turbulence and Combustion*, 88(3):279–310, 2012.
[394] S. Müller, M. Milano, and P. Koumoutsakos. Application of machine learning algorithms to flow modeling and optimization. *Annual Research Briefs*, pages 169–178, 1999.
[395] I. Munteanu, A. I. Bratcu, and E. Ceanga. Wind turbulence used as searching signal for MPPT in variable-speed wind energy conversion systems. *Renewable Energy*, 34(1):322–327, Jan. 2009.
[396] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[397] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[398] D. Needell and J. A. Tropp. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010.

[399] J. v. Neumann. Proof of the quasi-ergodic hypothesis. *Proceedings of the National Academy of Sciences*, 18(1):70–82, 1932.

[400] N. Nguyen, A. Patera, and J. Peraire. A best points interpolation method for efficient approximation of parametrized functions. *International Journal for Numerical Methods in Engineering*, 73(4):521–543, 2008.

[401] Y. Nievergelt and Y. Nievergelt. *Wavelets made easy*, volume 174. Springer, 1999.

[402] B. R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.

[403] B. R. Noack, T. Duriez, L. Cordier, M. Segond, M. Abel, S. L. Brunton, M. Morzyński, J.-C. Laurentie, V. Parezanovic, and J.-P. Bonnet. Closed-loop turbulence control with machine learning methods. *Bulletin Am. Phys. Soc.*, 58(18):M25.0009, p. 418, 2013.

[404] B. R. Noack, M. Morzynski, and G. Tadmor. *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media, 2011.

[405] F. Noé and F. Nuske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation*, 11(2):635–655, 2013.

[406] E. Noether. Invariante variationsprobleme nachr. d. könig. gesellsch. d. wiss. zu göttingen, math-phys. klasse 1918: 235-257. *English Reprint: physics/0503066, http://dx. doi. org/10.1080/00411457108231446*, page 57, 1918.

[407] F. Nüske, B. G. Keller, G. Pérez-Hernández, A. S. Mey, and F. Noé. Variational approach to molecular kinetics. *Journal of chemical theory and computation*, 10(4):1739–1752, 2014.

[408] F. Nüske, R. Schneider, F. Vitalini, and F. Noé. Variational tensor approach for approximating the rare-event kinetics of macromolecular systems. *J. Chem. Phys.*, 144(5):054105, 2016.

[409] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the A. I. E. E.*, pages 617–644, FEB 1928.

[410] G. Obinata and B. D. Anderson. *Model reduction for control system design*. Springer Science & Business Media, 2012.

[411] C. M. Ostoich, D. J. Bodony, and P. H. Geubelle. Interaction of a Mach 2.25 turbulent boundary layer with a fluttering panel using direct numerical simulation. *Physics of Fluids (1994-present)*, 25(11):110806, 2013.

[412] S. E. Otto and C. W. Rowley. Linearly-recurrent autoencoder networks for learning dynamics. *arXiv preprint arXiv:1712.01378*, 2017.

[413] Y. Ou, C. Xu, E. Schuster, T. C. Luce, J. R. Ferron, M. L. Walker, and D. A. Humphreys. Design and simulation of extremum-seeking open-loop optimal control of current profile in the DIII-D tokamak. *Plasma Physics and Controlled Fusion*, 50:115001–1–115001–24, 2008.

[414] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110(46):18368–18373, 2013.

[415] C. Pan, D. Yu, and J. Wang. Dynamical mode decomposition of Gurney flap wake flow. *Theoretical and Applied Mechanics Letters*, 1(1):012002, 2011.

[416] V. Parezanović, T. Duriez, L. Cordier, B. R. Noack, J. Delville, J.-P. Bonnet, M. Segond, M. Abel, and S. L. Brunton. Closed-loop control of an experimental mixing layer using machine learning control. *arXiv preprint arXiv:1408.3259*, 2014.

[417] V. Parezanovic, J.-C. Laurentie, T. Duriez, C. Fourment, J. Delville, J.-P. Bonnet, L. Cordier, B. R. Noack, M. Segond, M. Abel, T. Shaqarin, and S. L. Brunton. Mixing

layer manipulation experiment – from periodic forcing to machine learning closed-loop control. *Journal Flow Turbulence and Combustion*, 94(1):155–173, 2015.
[418] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(7–12):559–572, 1901.
[419] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz. Localized discrete empirical interpolation method. *SIAM Journal on Scientific Computing*, 36(1):A168–A192, 2014.
[420] B. Peherstorfer and K. Willcox. Detecting and adapting to parameter changes for reduced models of dynamic data-driven application systems. *Procedia Computer Science*, 51:2553–2562, 2015.
[421] B. Peherstorfer and K. Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.
[422] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM Journal on Scientific Computing*, 37(4):A2123–A2150, 2015.
[423] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. *arXiv preprint arXiv:1710.06759*, 2017.
[424] S. D. Pendergrass, J. N. Kutz, and S. L. Brunton. Streaming GPU singular value and dynamic mode decompositions. *arXiv preprint arXiv:1612.07875*, 2016.
[425] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge Univ Press, 1955.
[426] R. Penrose and J. A. Todd. On best approximate solutions of linear matrix equations. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 52, pages 17–19. Cambridge Univ Press, 1956.
[427] L. Perko. *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media, 2013.
[428] M. Phan, L. G. Horta, J. N. Juang, and R. W. Longman. Linear system identification via an asymptotically stable observer. *Journal of Optimization Theory and Applications*, 79:59–86, 1993.
[429] M. A. Pinsky. *Introduction to Fourier analysis and wavelets*, volume 102. American Mathematical Soc., 2002.
[430] T. Poggio. Deep learning: mathematics and neuroscience. *Views & Reviews, McGovern Center for Brains, Minds and Machines*, pages 1–7, 2016.
[431] P. Poncet, G.-H. Cottet, and P. Koumoutsakos. Control of three-dimensional wakes using evolution strategies. *Comptes Rendus Mecanique*, 333(1):65–77, 2005.
[432] C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.
[433] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz. Exploiting sparsity and equation-free architectures in complex systems (invited review). *The European Physical Journal Special Topics*, 223(13):2665–2684, 2014.
[434] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.
[435] J. L. Proctor and P. A. Eckhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International health*, 7(2):139–145, 2015.
[436] H. Qi and S. M. Hughes. Invariance of principal components under low-dimensional random projection of the data. IEEE International Conference on Image Processing, October 2012.
[437] S. Qian and D. Chen. Discrete Gabor transform. *IEEE transactions on signal processing*, 41(7):2429–2438, 1993.
[438] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, pages 232–256, 1997.
[439] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology.

Control engineering practice, 11(7):733–764, 2003.
[440] Q. Qu, J. Sun, and J. Wright. Finding a sparse vector in a subspace: Linear sparsity using alternating directions. In *Advances in Neural Information Processing Systems 27*, pages 3401—-3409, 2014.
[441] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*, volume 92. Springer, 2015.
[442] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A – Modeling, Simulation & Appplications*. Springer, 2013.
[443] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
[444] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
[445] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
[446] C. R. Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.
[447] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems*, 20(3):38–52, 2000.
[448] S. Raychaudhuri, J. M. Stuart, and R. B. Altman. Principal components analysis to summarize microarray experiments: application to sporulation time series. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 455. NIH Public Access, 2000.
[449] R. H. Reichle, D. B. McLaughlin, and D. Entekhabi. Hydrologic data assimilation with the ensemble Kalman filter. *Monthly Weather Review*, 130(1):103–114, 2002.
[450] B. Ren, P. Frihauf, R. J. Rafac, and M. Krstić. Laser pulse shaping via extremum seeking. *Control Engineering Practice*, 20:674–683, 2012.
[451] B. Ristic, S. Arulampalam, and N. J. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2004.
[452] A. J. Roberts. *Model emergent dynamics in complex systems*. SIAM, 2014.
[453] C. A. Rohde. Generalized inverses of partitioned matrices. *Journal of the Society for Industrial & Applied Mathematics*, 13(4):1033–1035, 1965.
[454] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31:1100–1124, 2009.
[455] C. Rowley. Model reduction for fluids using balanced proper orthogonal decomposition. *Int. J. Bifurcation and Chaos*, 15(3):997–1013, 2005.
[456] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D*, 189:115–129, 2004.
[457] C. W. Rowley and J. E. Marsden. Reconstruction equations and the Karhunen–Loève expansion for systems with symmetry. *Physica D: Nonlinear Phenomena*, 142(1):1–19, 2000.
[458] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 645:115–127, 2009.
[459] S. Roy, J.-C. Hua, W. Barnhill, G. H. Gunaratne, and J. R. Gord. Deconvolution of reacting-flow dynamics using proper orthogonal and dynamic mode decompositions. *Physical Review E*, 91(1):013001, 2015.
[460] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(e1602614), 2017.
[461] T. P. Sapsis and A. J. Majda. Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems. *Proceedings of the National Academy of Sciences*, 110(34):13705–13710, 2013.
[462] S. Sargsyan, S. L. Brunton, and J. N. Kutz. Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries. *Physical Review E*, 92(033304), 2015.
[463] S. Sarkar, S. Ganguly, A. Dalal, P. Saha, and S. Chakraborty. Mixed convective flow stabil-

ity of nanofluids past a square cylinder by dynamic mode decomposition. *International Journal of Heat and Fluid Flow*, 44:624–634, 2013.

[464] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science. 47th Annual IEEE Symposium on*, pages 143–152, 2006.

[465] T. Sayadi and P. J. Schmid. Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers. *Theoretical and Computational Fluid Dynamics*, pages 1–14, 2016.

[466] T. Sayadi, P. J. Schmid, J. W. Nichols, and P. Moin. Reduced-order representation of near-wall structures in the late transitional boundary layer. *Journal of Fluid Mechanics*, 748:278–301, 2014.

[467] H. Schaeffer. Learning partial differential equations via data discovery and sparse optimization. In *Proc. R. Soc. A*, volume 473, page 20160446. The Royal Society, 2017.

[468] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences USA*, 110(17):6634–6639, 2013.

[469] H. Schaeffer and S. G. McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.

[470] R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[471] M. Schlegel, B. R. Noack, and G. Tadmor. Low-dimensional Galerkin models and control of transitional channel flow. Technical Report 01/2004, Hermann-Föttinger-Institut für Strömungsmechanik, Technische Universität Berlin, Germany, 2004.

[472] P. J. Schmid. Dynamic mode decomposition for numerical and experimental data. *J. Fluid. Mech*, 656:5–28, 2010.

[473] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25(1-4):249–259, 2011.

[474] P. J. Schmid and J. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In *61st Annual Meeting of the APS Division of Fluid Dynamics*. American Physical Society, Nov. 2008.

[475] P. J. Schmid, D. Violato, and F. Scarano. Decomposition of time-resolved tomographic PIV. *Experiments in Fluids*, 52:1567–1579, 2012.

[476] E. Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. i teil. entwicklung willkürlichen funktionen nach system vorgeschriebener. *Math. Ann.*, 3:433–476, 1907.

[477] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[478] M. D. Schmidt, R. R. Vallabhajosyula, J. W. Jenkins, J. E. Hood, A. S. Soni, J. P. Wikswo, and H. Lipson. Automated refinement and inference of analytical models for metabolic networks. *Physical biology*, 8(5):055011, 2011.

[479] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[480] G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[481] A. Seena and H. J. Sung. Dynamic mode decomposition of turbulent cavity flows for self-sustained oscillations. *International Journal of Heat and Fluid Flow*, 32(6):1098–1110, 2011.

[482] E. Sejdić, I. Djurović, and J. Jiang. Time–frequency feature representation using energy concentration: An overview of recent advances. *Digital Signal Processing*, 19(1):153–183, 2009.

[483] O. Semeraro, G. Bellani, and F. Lundell. Analysis of time-resolved PIV measurements of a confined turbulent jet using POD and Koopman modes. *Experiments in Fluids*, 53(5):1203–1220, 2012.

[484] O. Semeraro, F. Lusseyran, L. Pastur, and P. Jordan. Qualitative dynamics of wavepackets in turbulent jets. *Physical Review Fluids*, 2(094605), 2017.
[485] G. Shabat, Y. Shmueli, Y. Aizenbud, and A. Averbuch. Randomized LU decomposition. *Applied and Computational Harmonic Analysis*, 2016.
[486] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
[487] A. S. Sharma, I. Mezić, and B. J. McKeon. Correspondence between Koopman mode decomposition, resolvent mode decomposition, and invariant solutions of the Navier-Stokes equations. *Physical Review Fluids*, 1(3):032402, 2016.
[488] E. Shlizerman, E. Ding, M. O. Williams, and J. N. Kutz. The proper orthogonal decomposition for dimensionality reduction in mode-locked lasers and optical systems. *International Journal of Optics*, 2012, 2011.
[489] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM Journal on Scientific Computing*, 29(3):1268–1288, 2007.
[490] L. Sirovich. Turbulence and the dynamics of coherent structures, parts I-III. *Q. Appl. Math.*, XLV(3):561–590, 1987.
[491] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.
[492] S. Skogestad and I. Postlethwaite. *Multivariable feedback control*. Wiley, Chichester, 1996.
[493] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.
[494] G. Solari, L. Carassale, and F. Tubino. Proper orthogonal decomposition in wind engineering. part 1: A state-of-the-art and some prospects. *Wind and Structures*, 10(2):153–176, 2007.
[495] G. Song, F. Alizard, J.-C. Robinet, and X. Gloerfelt. Global and Koopman modes analysis of sound generation in mixing layers. *Physics of Fluids (1994-present)*, 25(12):124101, 2013.
[496] D. C. Sorensen and Y. Zhou. Direct methods for matrix Sylvester and Lyapunov equations. *Journal of Applied Mathematics*, 2003(6):277–303, 2003.
[497] M. Sorokina, S. Sygletos, and S. Turitsyn. Sparse identification for nonlinear optical communication systems: SINO method. *Optics express*, 24(26):30433–30443, 2016.
[498] J. C. Spall. The Kantorovich inequality for error analysis of the Kalman filter with unknown noise distributions. *Automatica*, 31(10):1513–1517, 1995.
[499] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[500] W.-H. Steeb and F. Wilhelm. Non-linear autonomous systems of differential equations and Carleman linearization procedure. *Journal of Mathematical Analysis and Applications*, 77(2):601–611, 1980.
[501] R. F. Stengel. *Optimal control and estimation*. Courier Corporation, 2012.
[502] G. W. Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.
[503] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012.
[504] A. Surana. Koopman operator based observer synthesis for control-affine nonlinear systems. In *55th IEEE Conference on Decision and Control (CDC*, pages 6492–6499, 2016.
[505] A. Surana and A. Banaszuk. Linear observer synthesis for nonlinear systems using Koopman operator framework. *IFAC-PapersOnLine*, 49(18):716–723, 2016.
[506] Y. Susuki and I. Mezić. A prony approximation of Koopman mode decomposition. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 7022–7027. IEEE, 2015.

[507] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
[508] A. Svenkeson, B. Glaz, S. Stanton, and B. J. West. Spectral decomposition of nonlinear systems with memory. *Phys. Rev. E*, 93:022211, Feb 2016.
[509] S. Svoronos, D. Papageorgiou, and C. Tsiligiannis. Discretization of nonlinear control systems via the Carleman linearization. *Chemical engineering science*, 49(19):3263–3267, 1994.
[510] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(8):831–836, 1996.
[511] K. Taira and T. Colonius. The immersed boundary method: a projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007.
[512] N. Takeishi, Y. Kawahara, Y. Tabei, and T. Yairi. Bayesian dynamic mode decomposition. *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.
[513] N. Takeishi, Y. Kawahara, and T. Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.
[514] N. Takeishi, Y. Kawahara, and T. Yairi. Subspace dynamic mode decomposition for stochastic Koopman analysis. *Physical Review E*, 96(033310), 2017.
[515] F. Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics*, 898:366–381, 1981.
[516] Z. Q. Tang and N. Jiang. Dynamic mode decomposition of hairpin vortices generated by a hemisphere protuberance. *Science China Physics, Mechanics and Astronomy*, 55(1):118–124, 2012.
[517] R. Taylor, J. N. Kutz, K. Morgan, and B. Nelson. Dynamic mode decomposition for plasma diagnostics and validation. *Review of Scientific Instruments*, 89(053501), 2018.
[518] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
[519] Z. Ting and J. Hui. Eeg signal processing based on proper orthogonal decomposition. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 636–640. IEEE, 2012.
[520] S. Tirunagari, N. Poh, K. Wells, M. Bober, I. Gorden, and D. Windridge. Movement correction in DCE-MRI through windowed and reconstruction dynamic mode decomposition. *Machine Vision and Applications*, 28(3-4):393–407, 2017.
[521] C. Torrence and G. P. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological society*, 79(1):61–78, 1998.
[522] G. Tran and R. Ward. Exact recovery of chaotic systems from highly corrupted data. *SIAM Multiscale modeling and simulation*, 15(3):1108–1129, 2017.
[523] L. N. Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.
[524] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.
[525] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.
[526] J. A. Tropp. Recovery of short, complex linear combinations via l_1 minimization. *IEEE Transactions on Information Theory*, 51(4):1568–1570, 2005.
[527] J. A. Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing*, 86(3):589–602, 2006.
[528] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.
[529] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.
[530] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[531] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk. Beyond Nyquist: Efficient sampling of sparse bandlimited signals. *IEEE Transactions on Information Theory*, 56(1):520–544, 2010.
[532] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Randomized single-view algorithms for low-rank matrix approximation. *arXiv preprint arXiv:1609.00048*, 2016.
[533] J. H. Tu and C. W. Rowley. An improved algorithm for balanced POD through an analytic treatment of impulse response tails. *J. Comp. Phys.*, 231(16):5317–5333, 2012.
[534] J. H. Tu, C. W. Rowley, E. Aram, and R. Mittal. Koopman spectral analysis of separated flow over a finite-thickness flat plate with elliptical leading edge. *AIAA Paper 2011*, 2864, 2011.
[535] J. H. Tu, C. W. Rowley, J. N. Kutz, and J. K. Shang. Spectral analysis of fluid flows using sub-Nyquist-rate PIV data. *Experiments in Fluids*, 55(9):1–13, 2014.
[536] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *J. Comp. Dyn.*, 1(2):391–421, 2014.
[537] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
[538] R. Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. 2004.
[539] C. Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992.
[540] D. Venturi and G. E. Karniadakis. Gappy data and reconstruction procedures for flow past a cylinder. *Journal of Fluid Mechanics*, 519:315–336, 2004.
[541] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.
[542] S. Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf*, 1025, 2011.
[543] S. Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 4:4, 2013.
[544] S. Voronin and P.-G. Martinsson. RSVDPACK: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures. *arXiv preprint arXiv:1502.05366*, 2015.
[545] A. Wang et al. An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC, 2003.
[546] H. H. Wang, M. Krstić, and G. Bastin. Optimizing bioreactors by extremum seeking. *Adaptive Control and Signal Processing*, 13(8):651–669, 1999.
[547] H. H. Wang, S. Yeung, and M. Krstić. Experimental application of extremum seeking on an axial-flow compressor. *IEEE Transactions on Control Systems Technology*, 8(2):300–309, 2000.
[548] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Physical Review Letters*, 106:154101–1–154101–4, 2011.
[549] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.
[550] C. Wehmeyer and F. Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148(241703), 2018.
[551] G. Welch and G. Bishop. An introduction to the Kalman filter, 1995.
[552] P. Whitle. *Hypothesis testing in time series analysis*, volume 4. Almqvist & Wiksells, 1951.
[553] O. Wiederhold, R. King, B. R. Noack, L. Neuhaus, L. Neise, W. an Enghard, and M. Swoboda. Extensions of extremum-seeking control to improve the aerodynamic performance

of axial turbomachines. In *39th AIAA Fluid Dynamics Conference*, pages 1–19, San Antonio, TX, USA, 2009. AIAA-Paper 092407.

[554] K. Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & fluids*, 35(2):208–226, 2006.

[555] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[556] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the Koopman operator: extending dynamic mode decomposition. *Journal of Nonlinear Science*, 6:1307–1346, 2015.

[557] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. A kernel approach to data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.

[558] D. M. Witten and R. Tibshirani. Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):753–772, 2011.

[559] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Journal of Applied and Computational Harmonic Analysis*, 25:335–366, 2008.

[560] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(2):210–227, 2009.

[561] C. J. Wu. On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103, 1983.

[562] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[563] H. Ye, R. J. Beamish, S. M. Glaser, S. C. Grant, C.-h. Hsieh, L. J. Richards, J. T. Schnute, and G. Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences*, 112(13):E1569–E1576, 2015.

[564] E. Yeung, S. Kundu, and N. Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*, 2017.

[565] B. Yildirim, C. Chryssostomidis, and G. Karniadakis. Efficient sensor placement for ocean measurements using low-dimensional concepts. *Ocean Modelling*, 27(3):160–173, 2009.

[566] X. Yuan and J. Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12, 2009.

[567] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535, 2010.

[568] C. Zhang and R. O. nez. Numerical optimization-based extremum seeking control with application to ABS design. *IEEE Transactions on Automatic Control*, 52(3):454–467, 2007.

[569] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *arXiv preprint arXiv:1707.02876*, 2017.

[570] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *IEEE Robotics and Automation (ICRA)*, pages 528–535, 2016.

[571] W. Zhang, B. Wang, Z. Ye, and J. Quan. Efficient method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order models. *AIAA journal*, 50(5):1019–1028, 2012.

[572] S. Zlobec. An explicit form of the moore-penrose inverse of an arbitrary complex matrix. *SIAM Review*, 12(1):132–134, 1970.

[573] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of*

the Royal Statistical Society: Series B (Statistical Methodology), 67(2):301–320, 2005.