Notation Guide

The notation used in this chapter is as follows. Please note that many variables correspond to different *specific* instances throughout the chapter, however we have tried to remain consistent whenever possible.

General Symbols and Mathematical Operators				
$\mathbf{A}$	A general matrix		$\|\|A\|\|$	Frobenius (2) norm
$\mathbf{A}^{\mathbf{T}}$	Transpose of a matrix		$\underline{\boldsymbol{a}}, \boldsymbol{a}$	A vector or quantity in $\mathbb{R}^N$
Inputs and Outputs				
$\mathbf{x}$	feature vector		$\mathbf{y}$	target vector
$\mathbf{X}$	a matrix with feature vectors as the rows		$\mathbf{Y}$	a matrix with target vectors as the rows
N	Number of samples in a dataset			
Linear Algebra and Kernel Space				
U_A	Eigenvectors of a matrix		$\widehat{U}_A$	The top eigenvectors of a matrix
Λ_A	Eigenvalues of a matrix, arranged in a diagonal matrix		Σ	Singular values of a matrix, arranged in a rectangular matrix
$\mathbf{K}$	Kernel matrix, $\equiv \mathbf{X}\mathbf{X}^{\mathrm{T}}$ (Gram matrix, with linear kernel)		k	Kernel function
ϕ	Reproducing kernel Hilbert space (RKHS) feature vector		$\mathbf{\Phi}$	Matrix containing RKHS features as rows
ℓ	Loss function		ρ	similarity function
$\mathbf{P}$	projector from one space into another		$\mathbf{T}$	Latent space projection
$\mathbf{C}$	Covariance matrix, $\equiv \mathbf{X}^T\mathbf{X}$			
Miscellaneous				
$d(i,j)$	Distance between points i and j		$\boldsymbol{s}$	species of an atom
O	Big-O notation		α	PCovR mixing parameter
σ	Standard deviation or Gaussian width		p, q	Probability distribution

Methods

Descriptors for encoding chemical information

This section will cover an important precursor to many unsupervised tasks – numerical description of molecules and materials. We will briefly discuss the importance of descriptor selection, reviewing the current slate of popular descriptors and directing readers to relevant publications to select the appropriate representation. We will then cover how one may *compare* said representations using similarity kernels.

Unsupervised learning methods, by design, aim to detect similarities between samples within a dataset. We encode these similarities in **descriptors** or **fingerprints**, i.e., a set of features contained in numerical vectors or tensors that span the relevant information needed to distinguish samples.

What makes a good descriptor?

A good feature must, and at a minimum, distinguish between like and unlike data points. In constructing a good descriptor, we rely on the concepts of **completeness, smoothness,** and **symmetrization.**

Completeness means that for each descriptor, there is a one-to-one relationship between molecule/material and descriptor. For example, molecular weight is not a complete representation because multiple conformations correspond to the same value. Likewise, atomic formulae can describe many different molecules, and are therefore not complete.

Typically, we can satisfy this criterion by constructing a representation that transforms **smoothly** from one configuration to those nearby in chemical space. **Smoothness** simply means that the descriptor changes continuously as the atoms move. Later, you'll see two descriptors called "distance matrices" and "adjacency matrices". The former is smooth because the distances change smoothly with moving atoms. The latter (the adjacency matrix) is not, because when an atom moves outside of a cutoff radius, the descriptor changes discontinuously.

It is also important that descriptors exhibit the appropriate **symmetry** for the ML model. For example, if you analyze a dataset to assess the number of large molecules present, the descriptor must transform with the number of atoms in the molecule. However, if you are analyzing the same dataset for bonding motifs or point group symmetry, you would like molecules to appear similar regardless of the number of atoms.

As disappointing as it may be, there is no universally *good* descriptor but rather a wealth of descriptors that may be appropriate for one class of ML problems or another. Later, we will discuss how we might transform descriptors or assess their suitability for different ML tasks but let us look at a few examples first!

Popular Descriptors for Machine Learning

Many simple fingerprints exist in the form of easy-to-assess structural characteristics. As a first pass, scientists often use features in the form of a **one-hot encoding**, where each feature contains a discrete classification, such as the number of atoms or space group. Binary encodings such as these are typically best suited for problems where the importance of these classifiers is known a-priori.

There are also many **identifiers,** such as a chemical formula, database name, or SMILES string [1], that are used when comparing molecules or materials of sufficiently different chemical composition. SMILES strings specifically have been used to generate new ML descriptors for supervised models [2] and are popular in the fields of cheminformatics and bioinformatics, being increasingly used in generative models [3].

In quantum chemistry, we are often interested in the space of atomic configurations, typically encoded with the Cartesian coordinates (x, y, z) and element identities (s) in a set of vectors $\{\{x_1, y_1, z_1, s_1\}, \ldots, \{x_N, y_N, z_N, s_N\}\}$ (see for example, Cartesian coordinates of the cyclohexane molecule, Figure 1 (a-b)). However, these vectors are not typically appropriate to enumerate an atomic configuration, as a single configuration could adopt various values based upon the choice of origin, orientation, or ordering of the atoms, thereby not accounting for translational, rotational, and permutative symmetry, respectively. This **symmetrization**, or the transformation of a representation to account for molecular symmetry, is a key step to many models.

We can transform Cartesian coordinates in a variety of manners to encode these symmetries. The simplest way to achieve translational (and rotational) symmetry is to compute **internal coordinates** such as bond lengths or angles, to form a distance or adjacency matrix describing the configuration (as shown for cyclohexane in Figure 1 (c-d)).

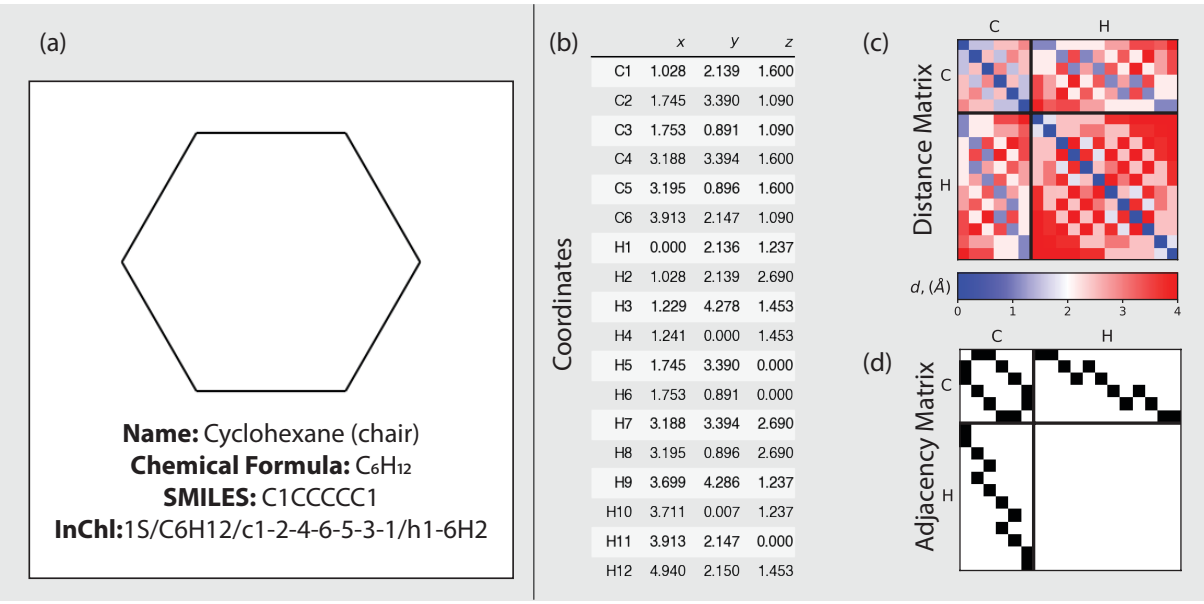

Figure 1. Various fingerprints and identifiers for the cyclohexane molecule (a). We can take the Cartesian coordinates of the molecule (b) and compute the interatomic distances to give a distance matrix (c), where each column and row correspond to the atoms in (b). Within this distance matrix, we can see the atoms which are neighbors of each other in light blue, which forms a Boolean adjacency matrix (d). A notebook constructing each of these fingerprints (and others) is stored at: https://github.com/rosecers/unsupervised-ml.

We can also represent an atomic environment by enumerating the relationship between a given atom and its sets of neighbors. This “atom-centered” approach has led to multiple classes of descriptors, including the Smooth Overlap of Atomic Potentials (SOAP) [4] and Behler–Parinello Symmetry Functions (BPSF) [5]. Mathematically, we can compute these atom-centered representations by superimposing a density field (e.g., a Gaussian or Delta function) onto each coordinate. This formulation mimics the near-sightedness of atomic interactions, i.e., the effects of a single atom will extend from their nucleus, decaying with distance. These types of descriptors can encode many-body correlations [6] and be extended to include long-range effects [7]. These descriptors are detailed further in Chapter 13: Kernel Method Potentials.

It is also popular to represent molecules and materials using a **Coulomb matrix** [8]**,** where each row corresponds to the 'atomic energies' and electrostatic repulsions between an nucleus and all other nuclei in the system (chapter 13). Although typically reserved for supervised models, Coulomb matrices are worth noting for their popularity.

Thus far, we have discussed descriptors that encode the rotational and translational symmetry of atomic systems. However, when comparing *collections* of atoms, we often need to achieve symmetry with respect to the permutations of atoms, which can be done by sorting, summing, or by taking a histogram of the atom-centered representations [9]. For example, say we have a molecule where each atom i is represented by its radial distribution function (a histogram showing where other atoms are as a function of distance, typically denoted g(r)). To represent the molecule, we take the average: $G(r) = \frac{1}{N}\sum_{i=1}^{N} g_i(r)$.

When you are working in a "big-data" regime, it is also possible to *engineer* features, i.e., use deep learning and neural networks to detect the discriminating features using raw inputs, often our Cartesian coordinates (x, y, z) and element identities (s). This topic could be a chapter on its own, so we point the interested reader to the works of [10], [11], and [12] for more insight into autoencoders and representation learning.

Kernels

A **kernel** is used to encode the similarity between one feature vector $\mathbf{x}_\mathrm{i}$ and another $\mathbf{x}_\mathrm{j}$. Chapter 9 will discuss, at length, the use of kernels in machine learning for supervised and unsupervised methods, where the kernel constitutes the similarity between any set of features and the features used to train the model. As listed in the notation guide, we will be using the variable $\mathbf{K}$ to denote a kernel matrix where each element K_{ij} denotes the similarity between feature vectors $\mathbf{x}_\mathrm{i}$ and $\mathbf{x}_\mathrm{j}$, where two identical features will typically result in a kernel value of 1. The **kernel function** k is used to assess the similarity between two feature vectors. Here we aim to introduce kernel feature spaces for the sake of introducing kernel PCA and choose to leave the more rigorous discussion of kernels to chapter 9.

A kernel matrix $\mathbf{K}$ is given by:

$$\mathrm{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \quad (1)$$

where $\mathbf{x}_\mathrm{i}$ and $\mathbf{x}_\mathrm{j}$ are the descriptors for configurations i and j. $k(a, b)$ is a kernel function, with popular kernels being the linear kernel ($k(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b}$, being a dot-product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$)), the radial basis function (RBF) kernel ($k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma||\boldsymbol{a} - \boldsymbol{b}||^2)$, where $\|\boldsymbol{a} - \boldsymbol{b}\|$ is the Euclidean distance between samples) and the polynomial kernels ($k(\boldsymbol{a}, \boldsymbol{b}) = (\gamma a^T b + c)^d$). A linear kernel matrix can also be called **the Gram matrix**. In quantum chemistry, there are various ways to combine the kernel similarity of multiple atoms to represent a molecule or crystal. For example, when building an averaged kernel, we construct our kernel such that

$$k(\mathcal{X}_A, \mathcal{X}_B) = \frac{1}{N_A N_B}\sum_{i\in A}\sum_{j\in B} k(\mathbf{x}_i, \mathbf{x}_j) \quad (2)$$

where the kernel between structures *A* and *B* (and their representations $\mathcal{X}_A$ and $\mathcal{X}_B$) is the average of the kernels of each of their atomic environments. It is important to note that with non-linear kernels, this is not equal to the kernel of the averaged representations, i.e.

$$k(\mathcal{X}_A, \mathcal{X}_B) \neq k(\frac{1}{N_A}\sum_{i\in A} \boldsymbol{x_i}, \frac{1}{N_B}\sum_{j\in B} \boldsymbol{x}_i) \quad (3)$$

When measuring similarities across a potential energy landscape, it can be useful to employ the *best-match kernel* [13], where the kernel only compares corresponding atoms.

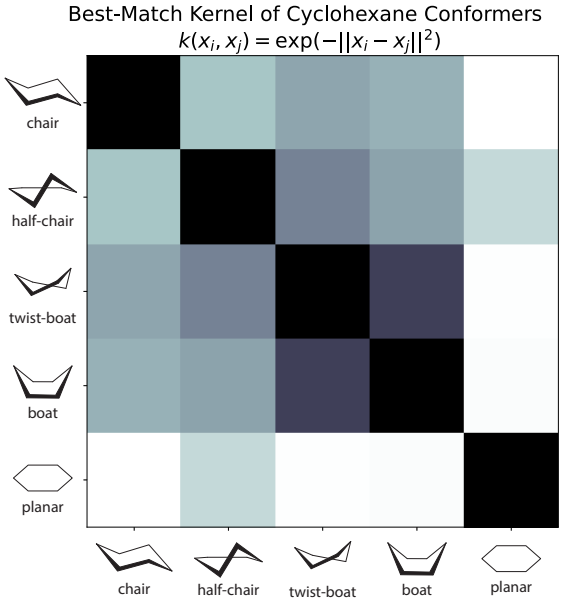

Figure 2. Best-match kernel of known cyclohexane conformers. Here, we compare the 3-body SOAP descriptors of each conformer using an RBF kernel, and average over the similarity of like-carbon atoms (i.e., in the comparison of the chair and half-chair conformations, we average over the similarities of C1-C1, C2-C2, and so on). Where the kernel is darkest ($K_{ij} = 1$), the molecules are most similar, as seen by dark squares along the diagonal. The planar conformation is the most different ($K_{ij} \to 0$), to all other conformations but shows some similarity to the half-chair conformation, which also has four co-planar, consecutive carbon atoms. The twist-boat and boat conformers are most similar to each other, consistent with what we know about cyclohexane strain.

When a kernel $\mathbf{K}$ is both symmetric and positive definite, Mercer's theorem states that there exists a reproducing kernel Hilbert space (RKHS) mapping $\mathbf{x} \to \phi$ where

$$\mathrm{K_{ij}} = \phi_\mathrm{i}^\mathrm{T}\phi_\mathrm{j} \quad (4)$$

Here ϕ is a *new, potentially infinite* feature vector for which the linear kernel is $\mathbf{K}$, even if $\mathbf{K}$ is not a linear kernel of $\mathbf{x}$. ϕ is not unique, and $\mathbf{K}$ may map onto many different RKHS. For readers that wish to investigate further the construction and mathematics of RKHS, we point you to the excellent review by Hofmann, et al [14].

Dimensionality Reduction / Mapping

This section will cover methods that take data of arbitrary dimensions and reduce it to a finite number of dimensions for further analysis, both quantitative and qualitative.

There are many different motivations for dimensionality reduction (DR), and each of these goals can motivate different approaches to the task. Traditionally, dimensionality reduction results in a reduced space that is a combination of the original feature space. However, we will also cover a class of unsupervised techniques that aim to preserve the original features, known as *feature selection* techniques.

Principal Components Analysis (PCA)

Typically, one is looking for a smaller feature space that contains as much information as possible about the initial data space — this leads to the archetypal DR technique: Principal Components Analysis.

Theory

Suppose we have our features in a matrix $\mathbf{X}$ whose rows contain our feature vectors $\mathbf{x}_\mathbf{i}$. As is necessary before doing a PCA, we standardize $\mathbf{X}$ so that the column means are all 0 and the variance is 1. We want a matrix $\mathbf{T}$ (typically called the **latent-space projection**) that is a linear combination of the features in $\mathbf{X}$, such that $\mathbf{T} = \mathbf{X}\,\mathbf{P}$, where $\mathbf{P}$ is a **projector**.

In PCA, we choose $\mathbf{P}$ to minimize the loss

$$\ell = ||\,\mathbf{X} - \mathbf{X}\,\mathbf{P}\mathbf{P}^T||^2 \qquad (5)$$

Where $\mathbf{P}^T$ denotes the transpose of $\mathbf{P}$. This loss is typically called the **reconstruction error**, as it encodes the loss of reconstructing the input space after projection into PCA-space.

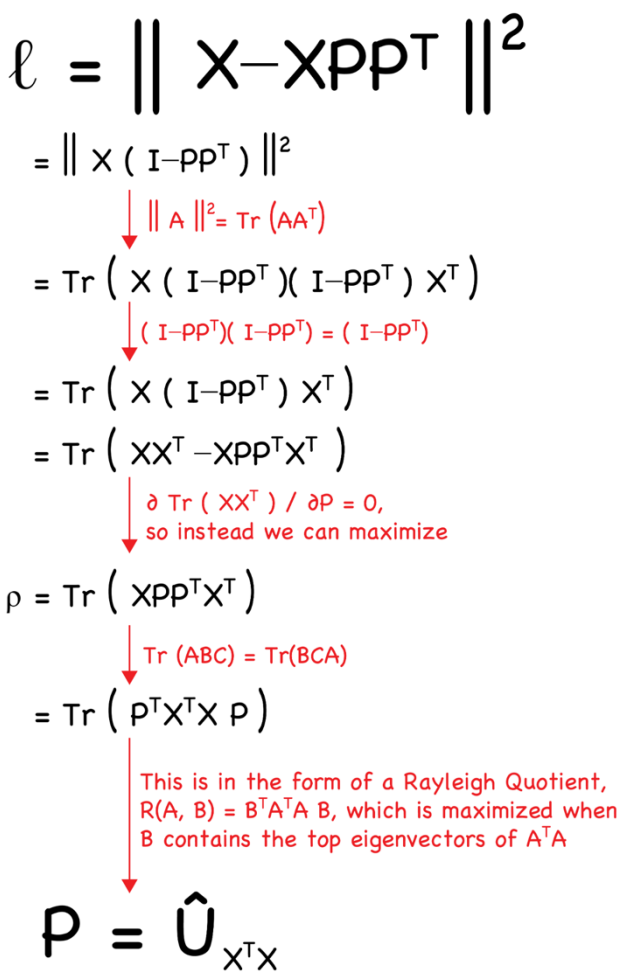

Figure 3. "Chalkboard" derivation of PCA projector from loss function ℓ given in Eq. (5). Here **X** is our inputs, **P** is the projector into PCA space, and **I** is the identity matrix.

As shown in Figure 3, we minimize the loss when $\mathbf{P}$ contains the top eigenvectors $\hat{\mathrm{U}}_{\mathrm{C}}$ of the covariance matrix $\mathbf{C} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$, also known as the **principal components**. Therefore,

$$\mathbf{T} = \mathbf{X}\,\hat{\mathrm{U}}_{\mathrm{C}} \qquad (6)$$

Where the ^ denotes the truncation to the desired number of eigenvectors. We can "whiten" or normalize the projection by including a factor of $\Lambda^{-1/2}$ (where Λ is a square matrix containing the eigenvalues of $\mathbf{C}$), giving $\mathbf{T} = \mathbf{X}\,\hat{\mathrm{U}}_{\mathrm{C}}\Lambda^{-1/2}$. We can compute the PCA of a matrix $\mathbf{X}$ either through an eigen-decomposition of C or a singular value decomposition (SVD) of $\mathbf{X}$, where

$$\boldsymbol{X} = U_K \Lambda U_C^T \qquad (7)$$

where again $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ and $\mathbf{K} = \mathbf{X}\mathbf{X}^T$, and selecting the desired number of eigenvectors. This relationship also suggests an alternative route to obtaining the principal components, as $\mathbf{T} = \mathbf{X}\,\hat{\mathrm{U}}_{\mathrm{C}}\Lambda^{-1/2} = \hat{\mathrm{U}}_K\Lambda^{1/2}$.

The resulting projection gives what we consider "the highest variance" view of our data. PCA is often used as a preliminary step in analysis, such as in the case studies later in the chapter, where we use PCA on our original features to reduce the dimensions while maintaining greater than 99% of the original variance. PCA can also be used to simplify even the easiest of visualization tasks. For example, if our input is the position of atoms in a molecule, the PCA will align the atoms, as seen in Figure 4.