Information Content and Analysis of X-ray Absorption Spectroscopy and X-ray Emission Spectroscopy

Samantha Tetef

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Gerald T. Seidler, Chair

Joshua J. Kas

Miguel Morales

Program Authorized to Offer Degree:

Physics

University of Washington

**Abstract**

Information Content and Analysis of X-ray Absorption Spectroscopy and X-ray Emission Spectroscopy

Samantha Tetef

Chair of the Supervisory Committee:

Professor Gerald T. Seidler

Physics

Data science and machine learning (ML) methods are revolutionizing scientific analysis and data processing. As a case in point, ML applied to X-ray spectroscopies has recently exploded, showcasing its effectiveness in fields such as electrical energy storage and chemical catalysis. Here, I include comprehensive computational studies of machine learning techniques for extraction of chemically relevant information from X-ray spectra, including X-ray absorption near edge structure (XANES) and valence-to-core X-ray emission spectra (VtC-XES). We focused on applying unsupervised ML to extract import chemical motifs in ensembles of organic molecules, including sulforganics and phosphorganics. We used these motifs to compare the chemical information encoded in XANES versus VtC-XES spectra. Through these studies, we introduced new unsupervised machine learning methods to the community, such as a variational autoencoder, t-distributed stochastic neighbor embedding, and uniform manifold projection, in the effort to create a data-driven approach to increasing the chemical sensitivity of XANES and VtC-XES. Moreover, we developed a pipeline to choose important XANES energies and increased computational effectiveness of linear combination fitting algorithms using compressed sensing techniques. This pipeline reduces measurement time and increases reliability of experimental results. Throughout these studies, we developed open-source tools for future researchers to utilize, including an API that interacts with PubChem to efficiently download and store metadata. Furthermore, these new tools will help automate future classification, characterization, and data compression of various X-ray spectroscopy techniques.

Table of Contents

[Acknowledgements 0](#_Toc124936682)

[1 Chapter 1 – Introduction and Overview of X-ray Spectroscopy 1](#_Toc124936683)

[1.1 X-ray absorption spectroscopy 1](#_Toc124936684)

[1.1.1 Experimental modes 2](#_Toc124936685)

[1.1.2 M- K- and L- Edges 4](#_Toc124936686)

[1.1.3 Regions of XAS spectra: XANES 5](#_Toc124936687)

[1.1.3.1 XANES processing 6](#_Toc124936688)

[1.1.4 Regions of XAS spectra: EXAFS 8](#_Toc124936689)

[1.1.4.1 EXAFS processing 8](#_Toc124936690)

[1.1.5 Wavelets 11](#_Toc124936691)

[1.2 X-ray emission spectroscopy – 3d transition metals 14](#_Toc124936692)

[1.3 X-ray emission spectroscopy – phosphorus and sulfur 17](#_Toc124936693)

[1.4 Are XANES and VtC-XES complimentary? 17](#_Toc124936694)

[1.5 References 18](#_Toc124936695)

[2 Chapter 2 – Survey of theoretical methods 0](#_Toc124936696)

[2.1 Density Functional Theory (DFT) (3-4 pgs) 0](#_Toc124936697)

[2.1.1 Time-dependent density functional theory (TD-DFT) 0](#_Toc124936698)

[2.2 Theoretical method for VtC-XES and XANES calculations (3 pgs) 0](#_Toc124936699)

[2.3 NWChem: A closer look (3 pgs) 2](#_Toc124936700)

[2.4 NWChem: The reliability of the results (3-5 pgs) 2](#_Toc124936701)

[2.5 NWChem: Implementation (maybe) 2](#_Toc124936702)

[2.6 References 2](#_Toc124936703)

[3 Chapter 3 – The Bane of the “Inverse” Problem 0](#_Toc124936704)

[3.1 Overview of classical and quantum inverse problems 0](#_Toc124936705)

[3.1.1 Classical inverse problems 0](#_Toc124936706)

[3.1.2 Quantum mechanical inverse problems 1](#_Toc124936707)

[3.2 Workarounds to the “inverse” problem 2](#_Toc124936708)

[3.2.1 Repeat the forward problem 2](#_Toc124936709)

[3.2.2 Bayesian approach 3](#_Toc124936710)

[3.2.3 Linear Combination Fitting (LCF) to references 4](#_Toc124936711)

[3.2.4 Machine Learning 6](#_Toc124936712)

[3.3 Inverse problem for EXAFS 7](#_Toc124936713)

[3.3.1 Sources of information loss in EXAFS 7](#_Toc124936714)

[3.3.2 Steps taken in EXAFS analysis to combat inverse problem 8](#_Toc124936715)

[3.4 Inverse problem for XANES 8](#_Toc124936716)

[3.4.1 Sources of information loss in XANES 8](#_Toc124936717)

[3.4.2 Steps taken to combat the inverse problem in XANES 9](#_Toc124936718)

[3.5 References 9](#_Toc124936719)

[4 Chapter 4 – Introduction to Machine Learning 0](#_Toc124936720)

[4.1 Machine Learning Basics 0](#_Toc124936721)

[4.1.1 Bayes theorem and maximum likelihood estimation 0](#_Toc124936722)

[4.1.2 Bias-variance tradeoff and model complexity 2](#_Toc124936723)

[4.2 Supervised Machine Learning 5](#_Toc124936724)

[4.2.1 Regression 5](#_Toc124936725)

[4.2.1.1 Linear and multivariate regression 5](#_Toc124936726)

[4.2.1.2 Ridge and LASSO regression 6](#_Toc124936727)

[4.2.2 Classification 8](#_Toc124936728)

[4.2.2.1 Support Vector Machine (SVM) 8](#_Toc124936729)

[4.2.3 Models for both regression and classification 9](#_Toc124936730)

[4.2.3.1 Decision Tree 9](#_Toc124936731)

[4.2.3.2 Random Forest 10](#_Toc124936732)

[4.2.3.3 Neural Networks 11](#_Toc124936733)

[4.2.3.3.1 Perceptrons 12](#_Toc124936734)

[4.2.3.3.2 Multilayer perceptron 13](#_Toc124936735)

[4.2.3.3.3 Generative Adversarial Networks 14](#_Toc124936736)

[4.2.4 Interpretability versus effectiveness 15](#_Toc124936737)

[4.2.5 Metrics 16](#_Toc124936738)

[4.2.5.1 Regression metrics 16](#_Toc124936739)

[4.2.5.2 Classification metrics 16](#_Toc124936740)

[4.2.5.3 Cross validation 17](#_Toc124936741)

[4.2.6 Uncertainty estimation 17](#_Toc124936742)

[4.2.6.1 Aleatoric versus epistemic uncertainty 17](#_Toc124936743)

[4.2.6.2 Models that incorporate uncertainty 18](#_Toc124936744)

[4.2.6.2.1 Gaussian Process 18](#_Toc124936745)

[4.2.6.2.2 Monte Carlo Dropout 19](#_Toc124936746)

[4.2.6.2.3 Bayesian Neural Network 19](#_Toc124936747)

[4.2.6.2.4 Mixed Density Network 20](#_Toc124936748)

[4.3 Unsupervised Machine Learning 21](#_Toc124936749)

[4.3.1 Dimensionality Reduction 21](#_Toc124936750)

[4.3.1.1 The curse of dimensionality 21](#_Toc124936751)

[4.3.1.2 Linear transformations 22](#_Toc124936752)

[4.3.1.2.1 Principal Component Analysis (PCA) 22](#_Toc124936753)

[4.3.1.2.2 Singular Value Decomposition (SVD) 23](#_Toc124936754)

[4.3.1.2.3 Nonnegative matrix factorization (NMF) 23](#_Toc124936755)

[4.3.1.3 Nonlinear transformations 23](#_Toc124936756)

[4.3.1.3.1 Variational Autoencoder (VAE) 24](#_Toc124936757)

[4.3.1.3.2 t-distributed Stochastic Neighbor Embedding (t-SNE) 25](#_Toc124936758)

[4.3.1.3.3 Uniform Manifold Approximation and Projection (UMAP) 26](#_Toc124936759)

[4.3.2 Clustering 26](#_Toc124936760)

[4.3.2.1.1 K-means 27](#_Toc124936761)

[4.3.2.1.2 K-nearest neighbors 27](#_Toc124936762)

[4.3.2.1.3 Density-based clustering 28](#_Toc124936763)

[4.3.2.1.4 Agglomerative hierarchical clustering 28](#_Toc124936764)

[4.3.3 Feature Selection 29](#_Toc124936765)

[4.3.3.1 Metrics 29](#_Toc124936766)

[4.3.3.2 Recursive Feature Elimination (RFE) 30](#_Toc124936767)

[4.4 References 32](#_Toc124936768)

# Acknowledgements

Thank you, Jerry. Thank you Seidler Lab. Thank you collaborators. Thank you, friends and family.

# Chapter 1 – Introduction and Overview of X-ray Spectroscopy

The following chapter gives a brief overview of X-ray absorption and emission spectroscopies, including nomenclature and typical uses. For a more detailed discussion, see the many excellent review articles and key papers on XAFS, including but not limited to Rehr and Albers [Rehr, 2000 #95], Sayers, Stern, and Lytle [Sayers, 1971 #241], Bunker [Bunker, #92], Ashley and Doniach [Ashley, 1975 #242], Newville, [Newville, 2014 #246], and Glatzel and Bergmann [Glatzel, #58].

Specifically, I will focus on the uses and chemical and electrical sensitivities of absorption versus emission spectroscopies, namely X-ray absorption near edge fine structure (XANES) and Valence-to-Core X-ray Emission Spectroscopy (VtC-XES). I will compare their uses in the context of the strength of the information they encode in their respective spectra, framing the question of whether their information is complementary – a major contribution of this dissertation.

## X-ray absorption spectroscopy

X-ray absorption spectroscopy (XAS) has been used as an important tool in many fields of science, such as materials science, physics, biology, chemistry, geosciences, and electronics. XAS, which produces spectra known as X-ray absorption fine structure (XAFS), is a bulk probe of both electronic and geometric structure around a chosen atomic species and is sensitive to properties such as oxidation state, valency, coordination, and bond length. When X-rays are shone onto a sample, they can either kick the electron out of the system (for X-ray Photoelectron Spectroscopy, or XPS) or push the system into an excited stated by kicking the electron to an unoccupied energy level (for XAS). There is an intrinsic lifetime of this excited state unique to the atomic species and hole. Finally, an electron from an occupied state will fall back down to fill the hole (creating X-ray Emission Spectroscopy, or XES) or radiate via two-electron processes, such as through the Auger−Meitner effect, as demonstrated in Fig. 1.

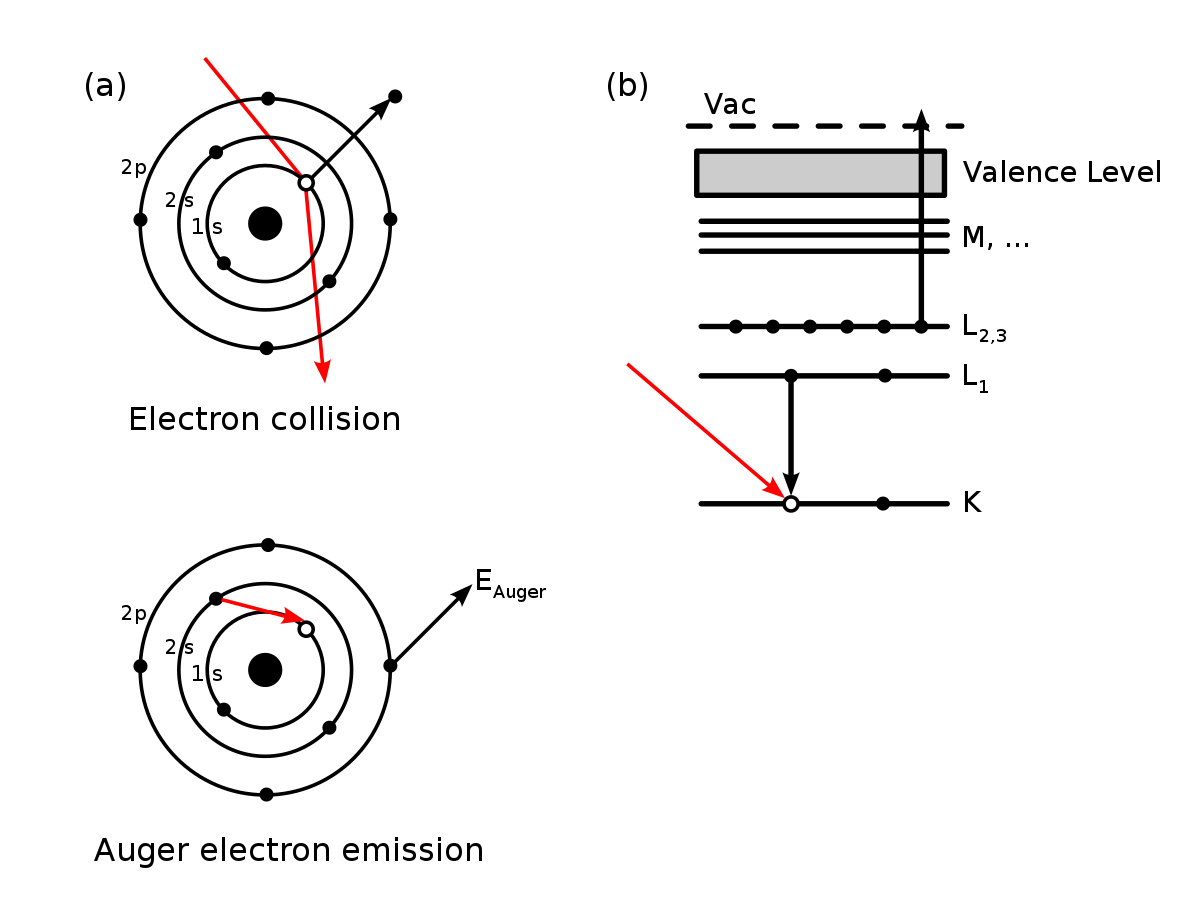


Fig. 1 The Auger-Meitner [Matsakis, 2019 #255;Meitner, 1922 #256] effect is a two-electron process, where an inner-shell electron falls to fill the core hole, thus emitting a photon, but the rest of the extra energy is dissipated by emitting a valence electron.

### Experimental modes

Experimentally, XAS can be performed either in transmission or florescence mode. Transmission mode is when the monochromatized X-rays must pass through the sample before entering the detector. Florescence mode occurs when the monochromatized X-rays hit the sample and the fluorescence off the sample enters the detector. In florescence mode, there is no background shape (besides stray scatter, if there is any). In this case, the absorption coefficient µ(E) can be solved for as If(E)/ I0(E).

Fluorescence mode, or X-ray Fluorescence (XRF), is different from X-ray emission spectroscopy (XES) due to the difference in resolution and thus their uses; in XES, the high resolution allows for distinct differentiation between the emission lines, whereas XRF is low-resolution over a much larger energy range and is thus used for qualitative identification of elemental composition.

When in transmission mode, the shape of the background is important to identify the XAS spectra, given by the absorption coefficient µ(E). Transmission-mode XAS is governed by Beer’s law

I(E) = I0(E)e-µ(E)x (1)

where x is the sample thickness, I(E) is the loss spectra (i.e., the photon counts after passing through the sample) and I0(E) is the X-ray spectrum without sample. A cross-section is shown in Fig. 2. [Rehr, 2000 #95] We could solve for x\*µ(E) as – ln (I(E) / I0(E)). We will normalize out the sample thickness (x) later.

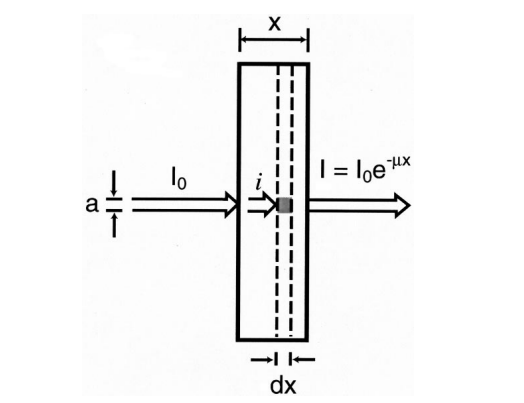


Fig. 2 Absorption coefficient for XAS. Taken from Rehr and Albers. [Rehr, 2000 #95]

### M- K- and L- Edges

µ(E) looks like a set of stairs, where each step, or edge, corresponds to an excitation energy of different inner-shell electrons. Each edge has a name and are commonly referred to using the IUPAC notation, as shown in Fig. 3. Exciting electrons from the 1s (n = 1) shell is called K-edge spectroscopy, while exciting electrons from the n = 2 shell are the L edges and the n = 3 the M edges. Each edge is then broken down further based the orbital angular momentum quantum number (l) and the magnetic quantum number (j). For example, exciting the 2s shell is the L1 edge, exciting from the 2p1/2 shell is the L2 edge, and from the 2p3/2 shell is the L3 edge. The same pattern continues for the M edges.

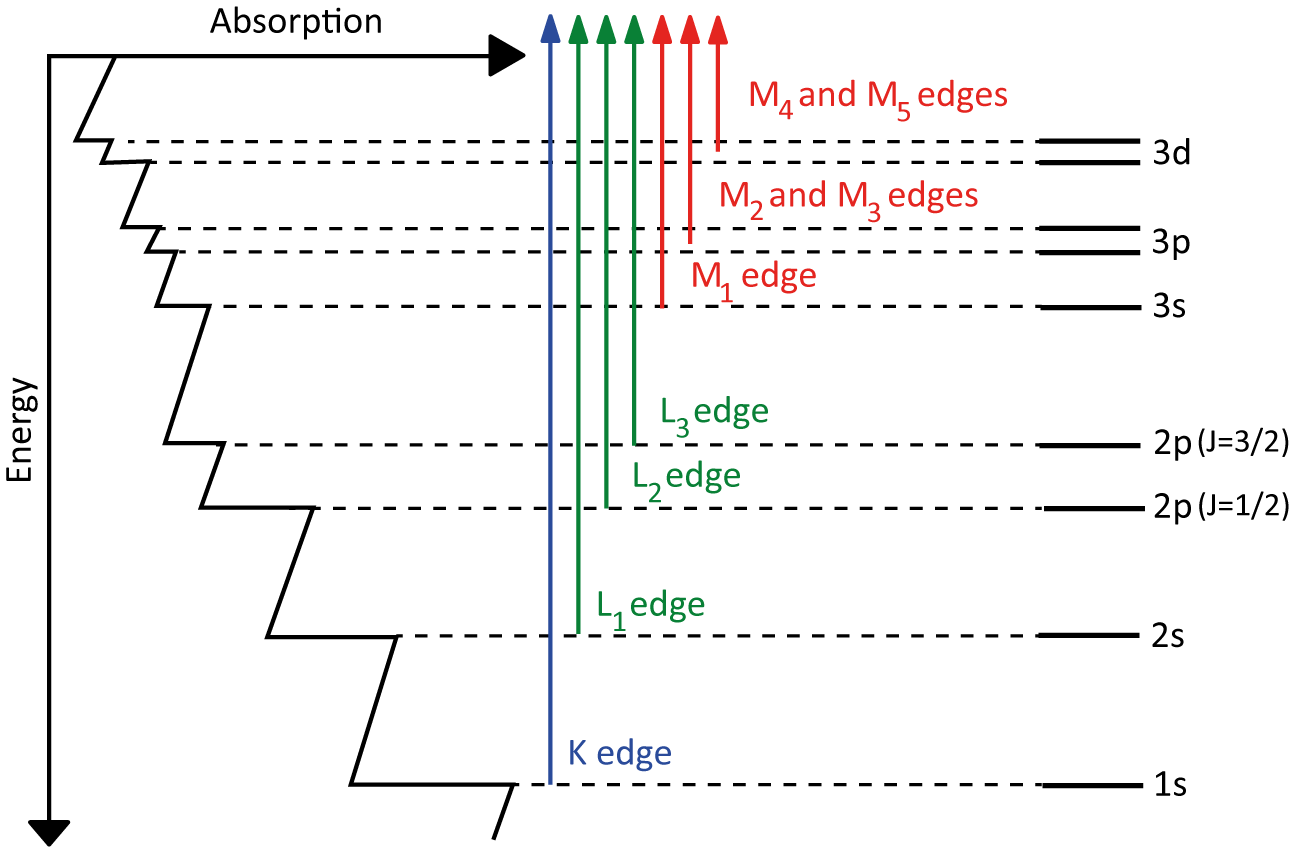


Fig. 3 Absorption edge of an element can be broken down into K, L, and M edges.

### Regions of XAS spectra: XANES

Zooming in onto these edges, you will see characteristic oscillations, as shown in Fig. 4. The XAS spectrum can be broken into two different regions – X-ray absorption near edge fine structure (XANES) and Extended X-ray Absorption Fine Structure (EXAFS). [Rehr, 2000 #95] XANES includes any pre-edge features, shoulders, and the region around the edge step, while the EXAFS includes the oscillations at higher energy.

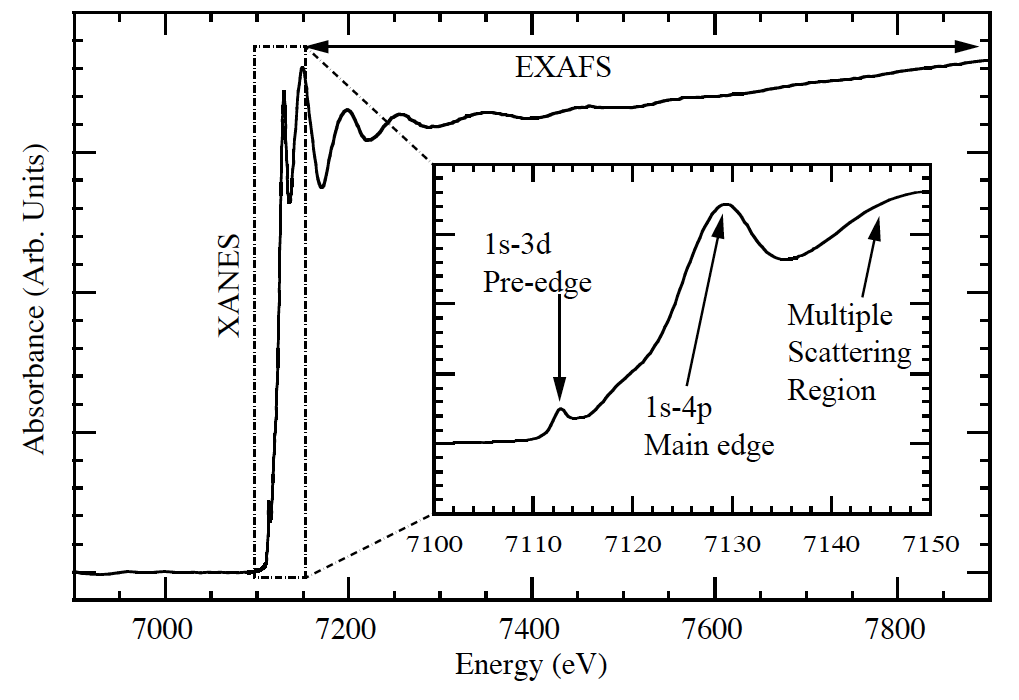


Fig. 4 The XANES and EXAFS regions of XAS spectra.

X-ray absorption near edge fine structure (XANES) is the region of XAS around the absorption edge, from any pre-edge features to about 50 eV past the edge, depending on the system. Because XANES probes electronic states near the Fermi level, it is sensitive to local electronic structure around the chosen atomic species, such as oxidation, spin, and valency. Calculating XANES spectra requires full multiple scattering theory and has broadening to transitions due to the lifetime of the core hole.

#### XANES processing

Because spectral features in the XANES spectra are often correlated, meaning two chemical properties can cause the same spectral trends, the most common analysis for XANES spectra is linear combination fitting onto reference spectra. Selecting reference spectra can be tricky because they must encapsulate the experimental domain, so their choice relies on prior knowledge of the scientific system. Moreover, differences in the second and third coordination shells between the reference structures and experimental sample can have negative influences on the reliability of these fits. [Jahrman, 2022 #206]

Furthermore, proper fitting requires proper “normalization,” as defined by the standard processing tools in Athena [Ravel, 2005 #97] and Larch [Newville, 2013 #193]. Normalizing XANES spectra is defined as fitting a pre-edge line and a post-edge polynomial function (either constant, linear, or quadratic), where the edge is determined as the maximum of the derivative. The pre-edge fit is then subtracted from the entire XANES spectrum, and the post-edge is rescaled such that the fitted polynomial curve falls along the line y = 1 after the edge. This process not only removes global scaling due to differences in sample thickness (i.e., number of atoms), but gives a consistent spectral shape. [Newville, 2014 #246] An example of this process is shown in Fig. 5.

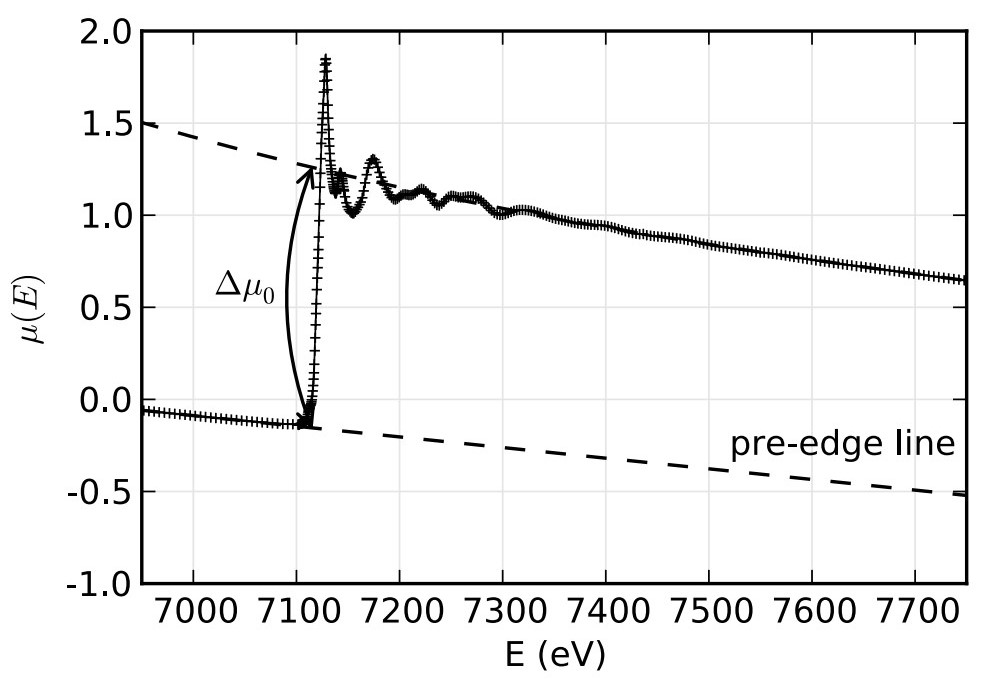


Fig. 5 An example of the normalization process for XAFS, where the pre-edge line is set to be along the y = 0 line, and the post edge line is set to be along the y = 1 line such that the edge step Δµ is one. Taken from Newville. [Newville, 2014 #246]

### Regions of XAS spectra: EXAFS

Extended X-ray Absorption Fine Structure (EXAFS) is weak multiple scattering and the higher energy oscillations past the absorption edge. EXAFS is sensitive to local geometric structure, such as coordination and bond length. Calculating EXAFS involves multiple scattering and the assumption of a muffin-tin potential, and much work has been done to utilize real-space Green’s functions to quickly calculate EXFAS via the FEFF program. [Rehr, 2010 #26]

#### EXAFS processing

An important aspect of EXAFS analysis derives from the parameters in the EXAFS equation. The EXAFS equation [Sayers, 1971 #241] is given as

(2)

where k = 2π/λ is the photoelectron vector, Nj is the number of equivalent scatterers (or coordination number), f(k) is the scattering factor, σ2j is the Debye-Waller factor, rj is the distance from the absorbing atom to the scatterer, and η(k) is the phase shift. [Sayers, 1971 #241] Because f(k) and η(k) can be calculated for a certain value of k, the only parameters left are Nj, rj, and σ2j. Because the Debye-Waller factor is just a term that represents disorder, it is simply used in fits but not an important scientific parameter. That leaves the two scientific important parameters – Nj and rj – left.

This analysis is done using the community standard tools: Athena [Ravel, 2005 #97] and its newer cousin Larch [Newville, 2013 #193]. Typically, you start with the normalized spectra following the same procedure in the XANES section of setting the edge to go from zero to one. Then, by removing the smooth post-edge background function you get the isolated XAFS χ(E), where χ(E) = (µ - µ0) / Δµ. Then, through a Fourier transform, the χ(E) is transformed to χ(k), where k is momentum, or

(3)

From χ(k), different fits and analysis can be performed to achieve the free parameters in the EXAFS equation. Often, because the high k features get washed out compared to the high amplitude but low k features, people will “k-weight” χ(k), i.e., multiplying χ(k) by either k2 or k3 to pronounce the higher k features. You can then Fourier transform the k-weighted χ(k) intro R space to achieve bond lengths. [Newville, 2014 #246]

Seminally, this spectroscopy and analysis technique was shown in Sayers, Stern, and Lytle [Sayers, 1971 #241] on crystalline and amorphous Ge, as shown in Fig. 6. Here, the crystalline Ge (Fig. 6a) clearly has more structure and detail than the amorphous Ge (Fig. 6b).

Chart

Description automatically generated

Fig. 6 Smoothed experimental EXAFS data for (a) crystalline Ge and (b) amorphous Ge. Only the oscillatory part χ of the absorption edge is shown. Figure taken from Sayers, Stern, and Lytle. [Sayers, 1971 #241]

Taking the Fourier Transform of the data using the EXAFS equation is shown in Fig. 7. Note that the absolute value of the amplitudes has no physical significance, but the relative values of the peaks can be interpreted using the equation

(4)

where the amplitude can indicate the relative number of atoms in that coordination shell, or with that bond distance. However, information about both the thermal and disorder in the system is needed to determine the appropriate broadening of the peaks. [Sayers, 1971 #241]

Diagram

Description automatically generated

Fig. 7 Fourier transform of the data in Fig. 6. φ(r), a radial structure function, compares amorphous and crystalline Ge. Numbers over the peaks indicate the measured distances inÅ. Figure taken from Sayers, Stern, and Lytle. [Sayers, 1971 #241]

### Wavelets

Another qualitative analysis of EXAFS that has seen recent traction is wavelet analysis. Wavelet transforms have shown to be a qualitative excellent way for filtering EXAFS contributions by Z, or atomic number, of the scattering species. This benefit capitalizes on the fact that larger Z atoms have smaller (spatially) electron orbitals [Muñoz, 2003 #247] and thus have a larger spread in momentum, which allows scattering in a larger k range than lighter Z atoms. A demonstration of the wavelet transform can be seen in Fig. 8, which shows the k-weighted χ(k) data visualized in three-dimensions, where one dimension is “distance” (from a k-weighted transform), the second dimension is momentum k, and the third dimension is the value of the modulus squared of the wavelet transform.

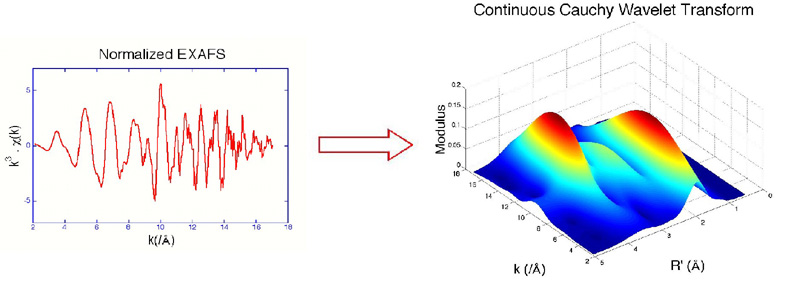


Fig. 8 Wavelet analysis can separate contributions based on the atomic number Z of the scatterer. Taken from Munoz, et al. [Muñoz, 2003 #247]

For a time-series dataset, where sampling occurs using the typical Shannon-Nyquist sampling rate, you can perform a Fourier transform to convert into frequency space. However, this analysis assumes signals are “stationary,” meaning they occur throughout the entire time of the signal, or there are no “events”.

One way to solve this issue is to evenly divide time and frequency resolution. However, this approach can be an issue if, for example, signals very in time duration. Wavelet transforms use the idea that low frequency signals occur over longer periods of time and thus need better frequency resolution. Conversely, higher frequency signals occur over a short time scale and thus need better time resolution. The difference in resolution can be seen in Fig. 9.



Fig. 9 Wavelet analysis is like Fourier transforming using different frequency (or conversely time) sampling rates to create a “multiresolution” transform.

This change in resolution is generated using a “mother” wavelet and adjusting the scale (and thus resolution). For example, this mother wavelet could look like

where a is the squishiness, or scale, of the wavelet and b is the time interval. Examples of specific wavelets ψ (which must be discrete) include the Haar, Mexican hat, Daubechies, and Coiflet wavelets, depending on the type of data. The final wavelet transform is obtained by taking the inner product of the signal with all the wavelets (which form an orthogonal basis), given as

This inner product gives a complex value, where the modulus squared can then be interpreted, as is the case for XAFS wavelet analysis.

## X-ray emission spectroscopy – 3d transition metals

X-ray emission spectroscopy (XES) occurs when an excited atom emits a photon as an electron falls to fill the vacancy. Emission lines are similarly named as XAS by the hole location, or shell in which the electron was originally excited from. For example, having electrons fall into a 1s core hole is K fluorescence. The initial shell of the falling electron then dictates whether the spectra are Kα (2p to 1s) or Kβ (3p to 1s). A breakdown of the K emission lines and their corresponding names can be seen in Fig. 10.

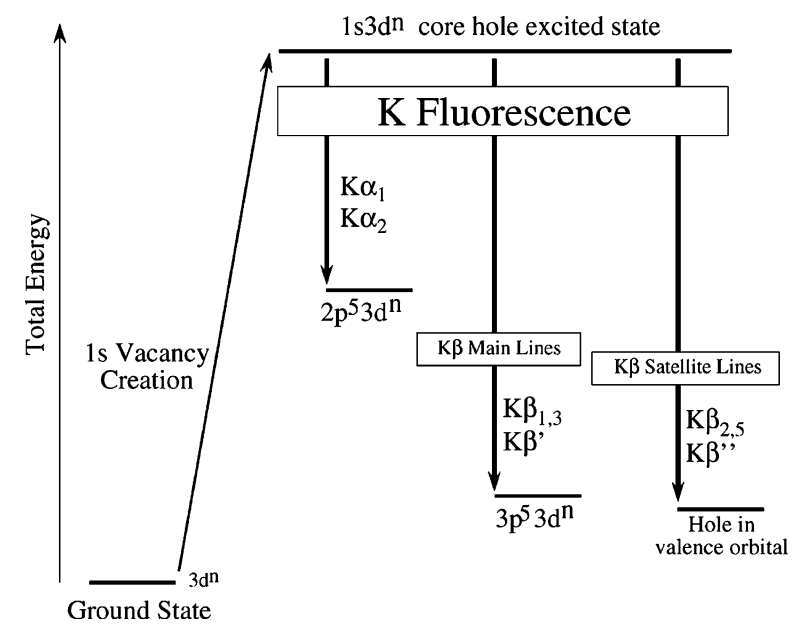


Fig. 10 Conventional naming of florescence lines.

The relative intensity of the different K emission lines is demonstrated in Fig. 11. Here, the Kα lines are much more intense (by a factor of about 3) than the Kβ lines. Furthermore, the Kβ satellite lines (or electrons falling from the valence shell, i.e., the 3d orbital for 3d transition metals) are smaller by a factor of about 500. This difference in photon counts is exactly due to dipole selection rules and is important to consider when deciding on which emission line to probe.

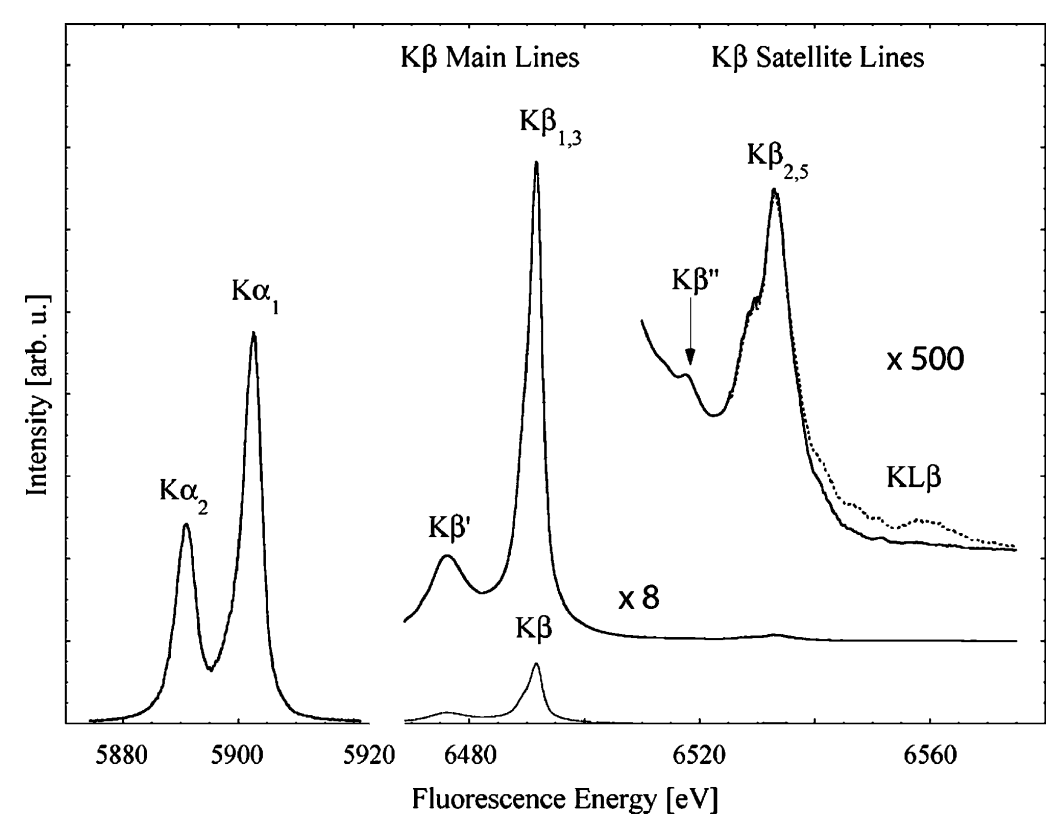


Fig. 11 The Mn K-fluorescence lines for MnO. [Glatzel, #58]

Core-to-core X-ray Emission Spectroscopy (CtC-XES) includes the Kα and Kβ lines (at least for 3d transitions metals). An in-depth discussion of CtC-XES is included in Glatzel and Bergmann. [Glatzel, #58] The utility of XES was notably demonstrated in Bergmann, et al. [Bergmann, 1998 #243], which studied the oxidation state of Mn in Photosystem II via the Kβ emission lines.

One benefit of Kα emission spectra is that it is not sensitive to second order environmental factors as the orbitals are too deep to be drastically hybridized. Thus, it is expected that the total oscillator strength of the Kα lines is proportional to the number of atoms. Thus, the integral of the Kα spectra can be theoretically used as a normalization, or scaling factor, for the other K emission lines.

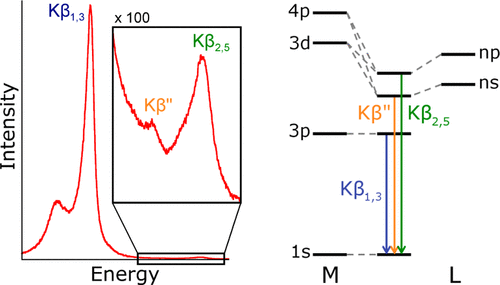


Fig. 12 Molecular orbital perspective of Kβ and its satellite lines.

For 3d transition metals, the Kβ satellite lines are also called Valence-to-Core XES (VtC-XES) andis highly sensitive to ligand identity due to the hybridization of orbitals, as shown in Fig. 12. Study of VtC-XES has been relatively new because only the recent improvements in synchrotron and lab-based spectrometers has made this experiment possible. Notably, Pollock and coworkers have developed a theoretical approach to calculating VtC-XES [Pollock, 2014 #244] and used it to study the iron oxidation in the iron-molybdenum cofactor (FeMoco) in nitrogenase. [Lancaster, 2011 #245] However, although VtC-XES is sensitive to ligand identity and other ligand properties, it is not a good quantitative method (rather, it qualitatively follows trends) to determining those properties. [Pollock, 2015 #68]

## X-ray emission spectroscopy – phosphorus and sulfur

For elements with their valence shells in the 3p orbital, such as phosphorus and sulfur, the Valence-to-Core XES (VtC-XES), which were the Kβ satellite lines for the 3d transition metals, now becomes just the Kβ satellite lines, meaning VtC-XES becomes dipole allowed and thus is way stronger in intensity compared to the non-dipole allowed transitions for the 3d shell. A large section of this dissertation is focused on the VtC-XES of both phosphorus and sulfur. Works have shown that VtC-XES for phosphorus and sulfur can help identify chemical classes. [Holden, 2020 #37;Yasuda, 1979 #35;Yasuda, 1984 #143;Mathe, 2021 #258]

## Are XANES and VtC-XES complimentary?

XANES and VtC-XES are often seen as “complimentary” [Mori, 2010 #73;MacMillan, 2015 #81;Qureshi, 2021 #82] as they are sensitive similar properties even though they probe different states, as demonstrated in Fig. 13. Much of this dissertation focused on analyzing whether VtC-XES and XANES are indeed complementary in terms of the strength of the chemical information they encode, or if this information is just highly coincidental.

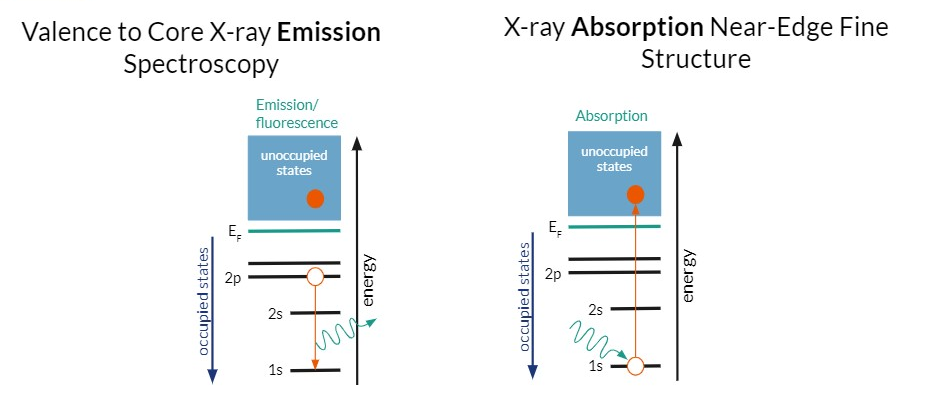


Fig. 13 VtC-XES probes occupied electronic states while XANES probes the unoccupied electronic states.

For example, Jahram et al. [Jahrman, 2020 #116] showed that the VtC-XES of V of various V oxides lacked distinct spectral features because of the high symmetry and simple bonding environment rather than the lack of sensitivity of VtC-XES. Additionally, it is expected that 3d transition metals without any 3d electrons (e.g., Cr6+) or with complete 3d shells (e.g., Zn2+) will have sensitivities in either XANES or VtC-XES, but not both. Or one technique, such as XANES, will be too sensitive and thus encode too much information, creating correlated features and thus muddling experimental conclusions. [Jahrman, 2022 #206] Thus, these examples bring up the questions of the strength of chemically relevant information encoded in each spectroscopy technique.

## References

2. Chapter 2 – Survey of theoretical methods

The following chapter will be an introduction of the theoretical methods used in a Chapters 5 and 6 of this dissertation, specifically the context of NWChem and its use time-dependent density functional theory (TD-DFT) and the reliability of the resulting calculations for the system discussed.

* 1. Density Functional Theory (DFT) (3-4 pgs)

DFT historical context. Self-consistent field.

* + 1. Time-dependent density functional theory (TD-DFT)

The problem with modeling excited states is the many-body problem, so a large factor in choosing a level of theory depends on whether it can do a good (enough) job approximating excited state. However, this concern is only an issue if you are indeed dependent on excited states, such as simulated a pump-probe system of XES spectra.

Somewhere talk about hybrid functionals and Gaussian basis sets?

* 1. Theoretical method for VtC-XES and XANES calculations (3 pgs)

DFT is the most popular way to calculate both XES and XAS spectra. The following section is an overview of available codes and their levels of theory, however it is neither a comprehensive list nor a complete discussion of the applications of each code. For a more complete discussion, see Nascimento, et al. [Nascimento, 2022 #264] and Rana, et al. [Rana, 2022 #266].

Starting with the “simplest” theory and going to the most sophisticated, we will start with atomic model codes, which include quanty and crispy. These codes use atomic cross-sections, multiplet theory with (mostly) fitted parameters, and model Hamiltonians. However, recent work has been done to remove these empirical parameters and replace them with *ab initio* calculated ones. (cite Charles’ paper, if published)

Next are the DFT level theories, which include WIEN2k, ABINIT, VASP, and CASTEP. These codes are accurate for ground state properties but are less reliable for excited states; they use a “final state rule” with a core hole. A more sophisticated code is FEFF, which uses quasi-particle Green’s function theory. FEFF is appropriate for excited states and is very efficient, but it is missing some many-body physics. These properties, along with its historical context and supported GUI, make it one of the most popular choices when calculation crystallographic structures.

Then there are the Bethe-Salpeter Equation (BSE) codes, which include Exc!ting and OCEAN. These are the most accurate but the most demanding. They are less established and less user friendly, although they are still missing some physics. Finally, there the most sophisticated quantum chemistry codes like MRCI, MRCC, CASPT2, and QMC. Although these codes are highly accurate, their complexity is basically intractable. For an overview of all codes and their theory levels, see Table 1.

|  |  |  |
| --- | --- | --- |
| **Code Name** | **Theory Level** |  |
| NWChem | Time-dependent DFT (TD-DFT) |  |
| Orca | Time-dependent DFT (TD-DFT) |  |
| VASP | DFT |  |
| FEFF | Real-space multiple-scattering theory |  |
| Ocean | Bethe-Salpeter Equation |  |
| FDMES | Finite Element |  |
| CTM4XAS | Mulitplet |  |
| Quanty | Multiplet |  |

Table 1 Overview of theoretical codes used for XAFS and XES calculations.

Cite successes and limitations

* 1. NWChem: A closer look (3 pgs)

Explain under the hood/ more in more depth explanation of the theory in NWChem

* 1. NWChem: The reliability of the results (3-5 pgs)

Cite examples

* 1. NWChem: Implementation (maybe)
  2. References

# Chapter 3 – The Bane of the “Inverse” Problem

In science, an “inverse” problem is using observations to try to calculate the factors that caused them. This idea is distinct from the forward problem, which starts from causes and then calculates the effects. Inverse problems are often termed “ill-posed”, which means they either (1) don’t have a solution, (2) the solution is not unique, or (3) the solution’s behavior does not change continuously with the initial conditions. In X-ray spectroscopy, the “inverse” problem – going from spectra to structure – often runs into problems with both (2) and (3). In this chapter, I will give some examples of inverse problems (both in classical and quantum mechanics), I will discuss some work arounds people have used to combat the “ill-posed” aspect of inverse problems, and then will discuss these applications and their limitations and uses for both XAFS (EXAFS and XANES) and XES spectra.

* 1. Overview of classical and quantum inverse problems
     1. Classical inverse problems

One example of a classical inverse problem is the question whether you can hear the shape of the drum.

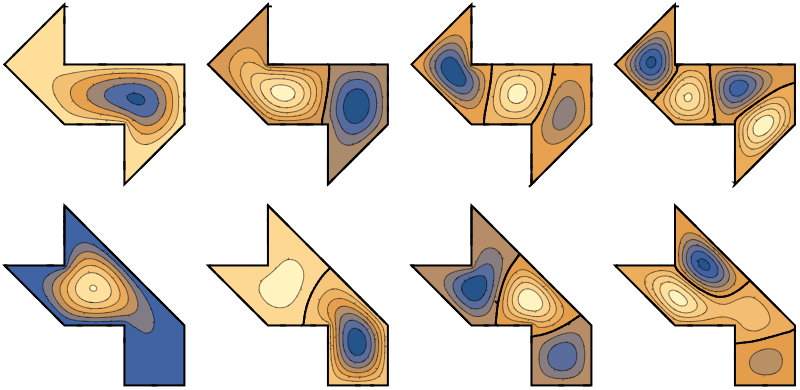


Fig. 1 Can you hear the shape of a drum? This question is one example of a classical inverse problem.

In addition to source reconstruction from acoustics, calculating the density of the Earth from measurements of its gravity field and reconstructing a three-dimensional object from its two-dimensional shadows are other examples of classical inverse problems.

* + 1. Quantum mechanical inverse problems

Some examples of quantum mechanical inverse problems are determining which arrangement of atoms yield the observed macroscopic properties or finding the potential that generates a particular spectrum. Unsurprisingly, X-ray spectroscopy is included in the list of quantum mechanical inverse problems.

* 1. Workarounds to the “inverse” problem

Because there is no proper solution to the “inverse” problem, as there is not a well-defined problem to fix, people have used various work arounds. All these work arounds rely on some amount of prior knowledge of the system. Thus, they turn the inverse problem into an informed inverse problem, making it possible to find a solution.

* + 1. Repeat the forward problem

One way to work around the forward problem is to repeatedly perform the forward problem, each time changing the input structure. One can then adjust the input structures depending on the output, which is often theoretically calculated spectra. This process is often how theorists change parameters in their calculations to improve their theoretically calculated spectra and thus match theory and experiment. However, repeating the forward problem is very time intensive, especially when calculation-intensive theories like density functional theory (DFT), or there is a large uncertainty in the possible input structure and thus a large parameter space must be explored.

Diagram

Description automatically generated

Fig. X An example of repeatedly solving the forward problem via theoretical calculations of Ru L3 edge XANES spectra of a series of Ru(II) and Ru(III) complexes. Taken from Nascimento and Govind. [Nascimento and Govind, 2022]

* + 1. Bayesian approach

Another work around is using formal Bayesian statistics to estimate the inverse problem. Although this process has been shown to work, it has not gained any traction in the community because it is very involved; not only is it very inaccessible, but it is computationally and experimentally intensive. Moreover, it also involves formalizing prior knowledge of the system by formalizing a prior in Bayes rule.

Chart, histogram

Description automatically generated

Fig. X Taken from Krappe and Rossner. [Krappe and Rossner, 2009]

[Krappe and Rossner, 2002; Krappe and Rossner, 2009; Rehr, 2005; Rossner, 2006]

* + 1. Linear Combination Fitting (LCF) to references

The most common method is linear combination fitting onto reference spectra. This approach assumes that because XAS is an average bulk probe, any components of different structures will contribute directly proportion to their concentration, or percentage of makeup. For example, an experimental sample with 2/3 the iron atoms in a 2+ oxidation state and the other 1/3 of the iron atoms in a 3+ oxidation state will result in a spectrum composed of two parts of an iron 2+ oxidation state reference and one part of an iron 3+ oxidation state reference. Obviously, things can get complicate quickly with the more properties and unknown parameters one must control for. Furthermore, choosing an appropriate library, or reference set, is critical in that it must find a balance between spanning a large enough domain to cover the experimental space but also not have redundant or correlated spectra. This issue becomes especially problematic with the highest uncertainty, or littlest prior knowledge, of the system.

Finally, this method propagates any errors, especially when fitting to theoretical spectra. It can also propagate any systematic errors in the experiment or normalization. It is especially unreliable if all your reference compounds have different second or third coordination shells, which is often the case for solution studies where reference compounds are usually crystalline.

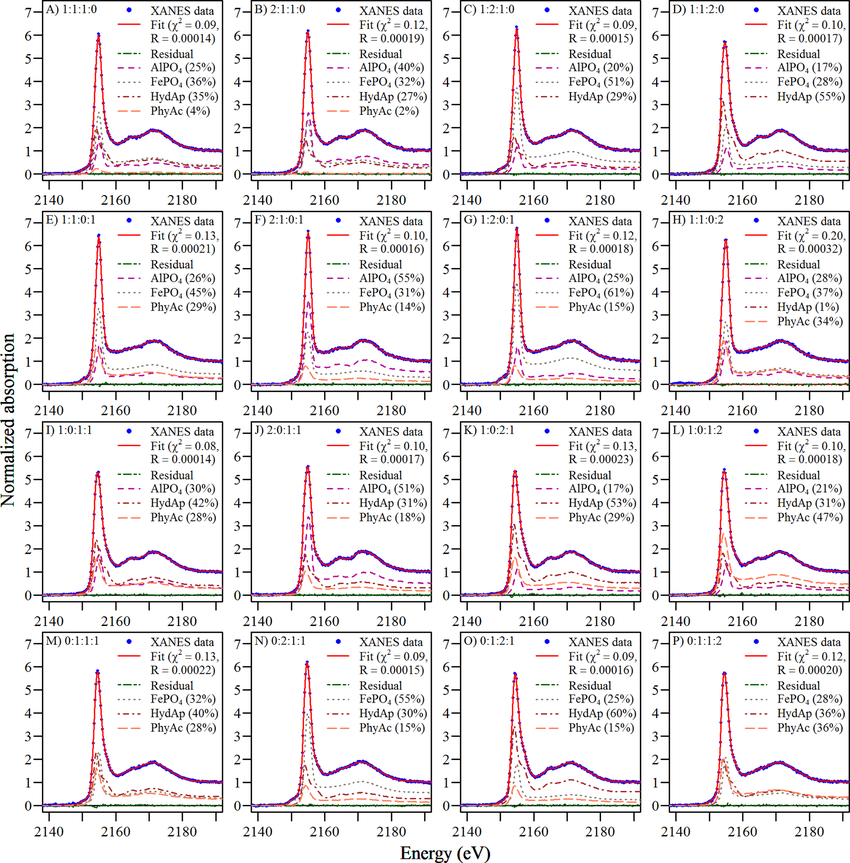


Fig. X Linear combination fitting to phosphorus XANES spectra. Taken form Werner and Prietzel. [Werner and Prietzel, 2015]

* + 1. Machine Learning

Cite Nascimento and Govind, 2022 on TD-DFT providing good enough training data for ML studies

* 1. Inverse problem for EXAFS
     1. Sources of information loss in EXAFS

Because EXAFS involves taking the Fourier transform, the loss of phase information is a particularly prominent issue in EXAFS analysis. Moreover, both thermal and structural disorder in structure can cause broadening in EXAFS data, both of which are encompassed in the Debye-Waller factor in the EXAFS equation.

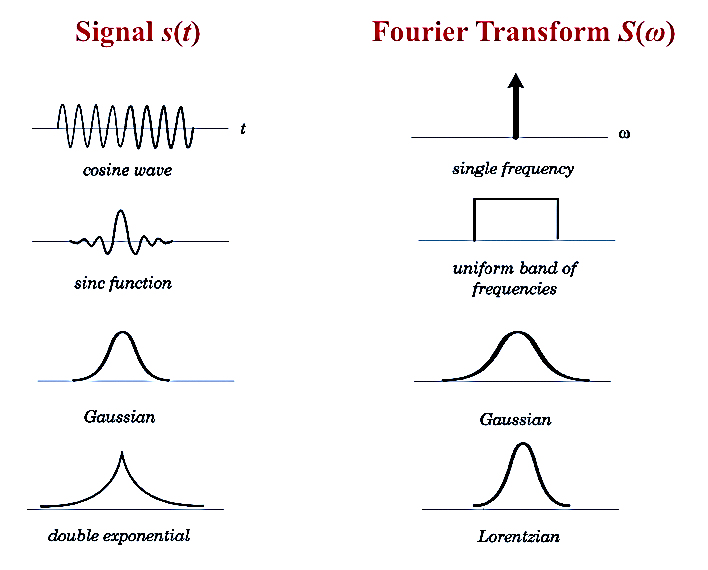


Fig. X Fourier transform pairs of certain signals.

* + 1. Steps taken in EXAFS analysis to combat inverse problem

EXAFS analysis is about incorporating more and more prior information into every step. First, you need to know the element you are probing and then the elemental make-up of the first coordination shell.

* 1. Inverse problem for XANES
     1. Sources of information loss in XANES

Information is lost in XANES spectra partly due to the Heisenberg uncertainty principle, which states that you cannot know exactly both momentum and position, or energy and time in the case of X-ray spectra. Because excited states have inherent lifetimes, electronic transitions are broadened in energy. Moreover, limits on experiment apparatuses, such as the resolution of your monochromator, have inherent resolution. Thus, any transition too close in energy will be smoothed out and indistinguishable from each other. Other types of spectral broadening can occur from more classical phenomena, such as plasmons and thermal vibrations.

Another reason why obtaining chemical or geometrical properties directly from spectra is difficult is because of correlated spectra, meaning the two different chemical properties can cause the same spectral feature. Or effects on spectra can be in overlapping energy regions, making it impossible to discern the cause of the feature. [Jahrman, 2022]

* + 1. Steps taken to combat the inverse problem in XANES

All four of the previously stated work arounds for the inverse problem (repeat the forward problem, Bayesian analysis, compare to a reference, and machine learning) have been applied to XANES spectra.

* 1. References

1. Chapter 4 – Introduction to Machine Learning

Machine learning (ML) has seen recent explosions in popularity and applications, catalyzed by advances in both models and accessibility of large data sets. In the most general sense, ML can be described as data-driven pattern recognition by performing linear decomposition onto nonlinear transformations; these transformations are often called *basis functions*. ML can be divided into three main categories: (1) supervised machine learning, (2) unsupervised machine learning, and (3) reinforcement learning.

Here, I will focus on supervised and unsupervised learning, which are often discussed together, as reinforcement learning is an exciting and complicated field unto itself. This chapter will give a brief overview of the most popular supervised and unsupervised methods as well as give context to the more specialized methods used in later chapters.

* 1. Machine Learning Basics

In this section, I will cover the basics of machine learning. However, it is helpful to start with a statistical approach to model generation, as machine learning models and algorithms often pull from both frequentist and Bayesian approaches.

* + 1. Bayes theorem and maximum likelihood estimation

It is helpful to first consider the case where we observe a random variable, or a sampling from some inherent probability distribution. The goal of our model is to then use these observations to predict new outcomes based on initial conditions. To accurately do so, we want our model, characterized by the *parameters* , to have the best given our *data*, which is comprised of target variables and input parameters .

It is helpful to approach this problem from a Bayesian point of view using Bayes theorem, which states . Rephrased in words, this equation states the posterierior equals the likelihood times the prior, divided by the evidence. Let’s say we have no prior knowledge, and we can normalize the distribution later, so we can ignore the prior and evidence, respectively. Thus, the likelihood represents the probability of observing the data given the model parameters. Maximizing this likelihood, called *Maximum Likelihood Estimation (MLE)*, is a frequentist approach but is easily motivated by Bayesian inference (if using naive or uniform priors). However, if you have prior knowledge about the form of the solution, then you would want to use the purely Bayesian approach to solving this problem, called *Maximum a Posteriori (MAP)*.

Thus, given a set of observations for , the likelihood we want to maximize is given by . However, the maximum of any monotonic function is the same as the maximum of the log of that function. Thus, we can take the log of the entire equation and use log rules to transform this *cost, or objective, function* into . Here, we can start to make assumptions about the form of the probability distribution. If we assume is normally distributed around y, then we get an ordinary least squares solution (OLS). This is also equivalent to chi-square minimization. However, again, this is only true for Gaussian noise.

* + 1. Bias-variance tradeoff and model complexity

Assuming your data is being pulled from some inherent distribution and is thus a random variable, the expected error in your fit, i.e., , can be decomposed into three terms – the irreducible error, the bias-squared terms, and the variance term. The irreducible error comes from the noise in the data – here, we assume your target data y can be represented by , where is a deterministic function that depends on the input parameters and model parameters (or weights) , added to random fluctuations via , which we can model as , or sampling from a Gaussian distribution. [Bishop, #195]

The bias-squared term is the error, or deviation, of the data points from the model’s predictions, while the variance term represents the deviation of predictions from the real answer with each new draw (or realization) of the random variable. Essentially, the bias-squared term represents error in a single sample, while the variance represents the error from multiple samples. As can be expected, the bias-squared and variance terms come into play when choosing the complexity of the model.

Diagram

Description automatically generated

Fig. 1 Model complexity and total error – the bias versus variance tradeoff.

The model complexity, which can easily be seen as the number of hyperparameters or the degree of the polynomial we choose to fit, can be determined by looking at a similar figure as the one shown in Fig. 1. Here, we have plotted the total error (the sum of the irreducible, bias-squared, and variance terms) versus model complexity. A simple model might predict every point as the mean of all points; thus, the errors draw to draw will be similar (low variance) but within a draw might be high (high variance). On the other hand, a complex model might perfectly predict every data point it has seen (low bias) but will fail in regions where the model has not seen data before (high variance error). Thus, the goldilocks complexity occurs in the minimum of the total error, which represents the best balance between the bias-squared error and the variance error term.

The reason why bias-variance tradeoff is important is because it indicates how generalizable your predictions are when predicting a new draw of your random variable. Fig. 2 shows how the complex models with a high variance and low bias will overfit, meaning that even though accuracy on the training data is great, the accuracy on new or test data is bad. Underfitting occurs when simple models can’t predict well (training and test accuracies are both the same, but low). Ideally, both the training and test accuracies are high and the same, which occurs when a good balance between both error terms is met.

Graphical user interface, application, Word

Description automatically generated

Fig. 2 The bias-variance tradeoff impacts the generalizability of the model.

To deal with this overfitting problem, you often have a validation set during model training. Throughout training, you check results on a dataset the model has not seen before (and is therefore not training data) but is not in the test set. The test set should only be used as a final metric and should not impact any decisions about model architecture; otherwise, predictions become biased, and the test set is no longer an accurate metric for the generalizability of your model. People often use a validation set to determine hyperparameters, model complexity, or any other training decision before a final evaluation.

* 1. Supervised Machine Learning
     1. Regression

Oftentimes, people want a quantitative prediction of a target variable, or output, given a set of inputs, or features. Predicting a quantitative target is called *regression* and is one of the two types of supervised machine learning.

* + - 1. Linear and multivariate regression

Again, let’s say we have a target variable , where is a deterministic function that depends on the input parameters and model parameters (or weights) . We incorporate random fluctuations via , which we can model as , or Gaussian noise. The analytic solution to finding the optimal weights, or , is , where is a matrix composing of all observed . This solution, a line or hyperplane that can be visualized in Fig. 3, is thus determined by all the input and output variables and is linear because it is purely matrix multiplication and inversion.

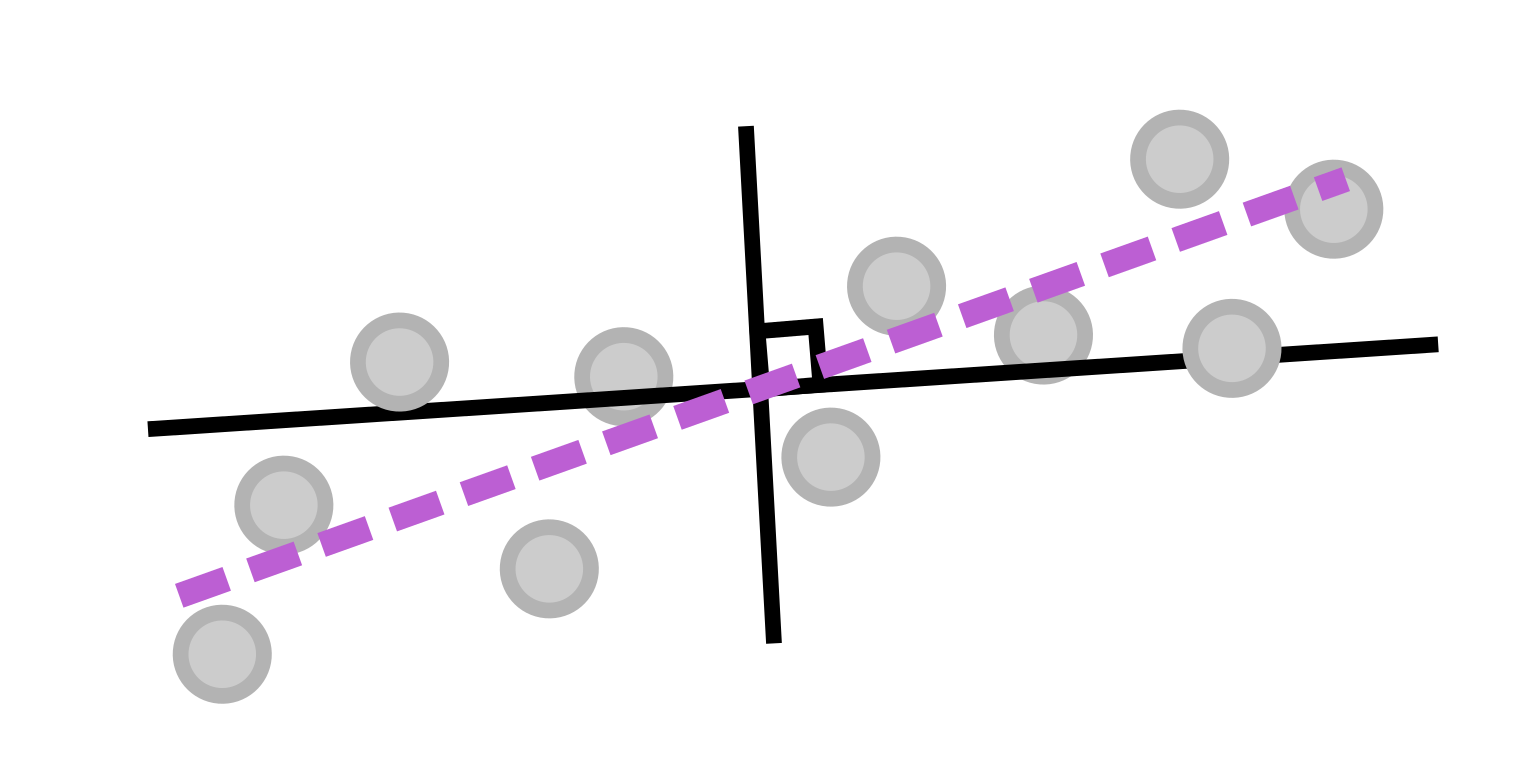


Fig. 3 Linear regression is finding a line (or hyperplane) that most follows the linear trends in the data, and as a bonus, it has an analytical solution.

Multivariate regression is the same as linear regression, except when the output variable () is multidimensional (and thus becomes a matrix Y). Thus, the epsilon term becomes a matrix (instead of a vector) and the weight vector becomes a matrix W as well. However, the solution looks the same – .

* + - 1. Ridge and LASSO regression

Including a regularization term in your fits, i.e., penalizing high weights by including a new term in your objective function (which is essentially a Lagrangian you are trying to minimize), has been shown to have better generalizability. First, a general norm looks like , where p identifies the type of norm. Thus, to add a regularization term to your objective function would results in something like , where the second term (the regularization term) is modified in scale by the Lagrange multiplier. Setting p to be two is called the L2 norm, which represents the standard Euclidean distance, and is called *ridge regression*. Setting p to be one is called the L1 norm, which is essentially taking the absolute value, and is called LASSO (Least Absolute Shrinkage and Selection Operator) regression.

Diagram, schematic

Description automatically generated

Fig. 4 The regularization term affects the sparsity of the solution.

The type of regularization is important because it impacts the sparsity of the solution. For example, let’s say you have two features, or input variables, and thus your weight vector is also two-dimensional. The equipotential surfaces (in other words, solutions where the overall loss (given by the norm) is the same) look like the green shapes in Fig. 4 – LASSO gives diamond shaped surfaces while Ridge gives circular surfaces. Solutions to the first term (i.e., the data-driven term) are represented by the concentric ellipses in red. All these solutions (i.e., any combination of the two weight components that fall along one of those red ellipses) may equally explain the data, but the individual elements of the weight vector will be different.

An overall solution that minimizes the regularization term can be found by slowly growing the green equipotential surface from the origin until it intersects with one of the red curves. By doing so, the L1 norm will more likely converge on a solution with sparse solutions (or solutions along one axis of the weight vector) because most of the volume is distributed along the axis, whereas the Ridge equipotential surfaces jut out in all directions.

* + 1. Classification

Instead of learning a (multidimensional) line that follows the trends in the data as in regression, classification finds a *decision boundary*, which is often a line, or in higher dimensions, a (hyper)plane, that best separates data points. Thus, classification predicts categorical targets.

* + - 1. Support Vector Machine (SVM)

Support vector machines (SVMs) generate \*unique\* decision boundaries by including some leeway, or margin, in the fit in which they try to maximize. This unique solution is called the maximum-margin hyperplane. A demonstration of a SVM is seen in Fig. 5. Note that, in order to maximize the margin, the width of which is given by 2 / ||w||, one must minimize w. Because the solution is completely determined by points on or within the margin (indicated by the points on the dashed line in Fig. 5), these points are called *support vectors*. Moreover, the SVM activation function can be modified to be soft, meaning it can allow for a few outliers to either get close or cross the decision boundary, a beneficial property for nonlinearly separable data. [Russell, 2010 #229]

A close-up of a speedometer

Description automatically generated with low confidence

Fig. 5 The SVM maximizes the distance from the decision boundary between all data points **and** it maximizes the margin around this decision boundary. The margin ensures that the solution is unique.

* + 1. Models for both regression and classification

Most machine learning models can be adapted for both types of supervised machine learning. The next section will give an overview of the most popular models.

* + - 1. Decision Tree

Decision trees are easy to interpret, which is one of their biggest strengths, but decision trees easily overfit, which is their biggest downfall, because they look for the strongest correlations between input and output features. However, correlation does not necessarily mean causation, so having a good representative set of features is critical when using a decision tree. [Russell, 2010 #229] A diagram of a decision tree can be seen in Fig. 6.

Diagram

Description automatically generated

Fig. 6 A decision tree is composed of decision nodes (composed of a question about the data), branches (which depend on the answer to the question, which are typically interpreted as a “yes” or “no” answers), and leaf nodes (the target variables).

Decision trees make their branches by recursively looking for the strongest correlation between input features and the target variables. The strongest correlations appear near the top (or root) node, while further categorizations occur on lower branches. Thus, decision trees are most naturally used for classification problems.

* + - 1. Random Forest

A random forest is a collection of decision trees and gains its strength from “ensemble” learning. Essentially, random forests try to avoid the overfitting issues with decision trees by combining many models (decision trees) together and taking a majority vote of each tree. The individual trees are then “weakly” trained, meaning they are trained on different subsets of the training dataset, or on predicting different classes. [Breiman, 2001 #233]

Diagram

Description automatically generated

Fig. 7 A random forest is a collection of decision trees, where the overall decisions are a majority voting or averaged result from each tree.

* + - 1. Neural Networks

Neural networks are the most versatile machine learning model, and the hardest to interpret, which is why they are notoriously labeled as “black boxes.” However, their strengths are derived from the nonlinear transformations interspersed with linear transformations throughout the network. Here, nonlinear means that the output does not correlate 1:1 with the input and therefore the transformation cannot be inverted. Errors in the final predictions are then propagated throughout the model (starting from the output to the input) via *backprop*, which relies on the derivatives of the weights. However, the modern python tools use automatic differentiation, which numerically solves for the derivative (instead of finding an analytic solution), thus turning this issue into a quick and painless behind-the-scenes aspect of model training.

* + - * 1. Perceptrons

The basic building block of neural networks is the perceptron. Perceptrons were originally invented in 1943 by McCulloch and Pitts [McCulloch, 1943 #230] but were horrible at generalizing predictions and thus were ignored for a long time, and instead people focused on kernel-based models. That is, until 1986, when David Rumelhart, Geoffrey Hinton, and Ronald Williams [Rumelhart, 1986 #231] had the idea of combining them in “layers” and machine learning algorithms and their applications exploded.

A picture containing text, clock

Description automatically generated

Fig. 8 The composition of a perceptron – linear combination of inputs via a weight vector which then get summed and passed to a nonlinear activation function.

Perceptrons are composed of four different steps: 1. Input, 2. weights, 3. Summation, and 4. activation function. The first three steps are the linear transformation, where inputs are weighted and summed together (the weight vector is what is “learned”). Then, that summation is passed to a nonlinear activation function. Activation functions that are commonly used are sigmoid, tanh, ReLu, step, and hinge. All these activation functions are zero for negative values and then (typically) move towards one around the origin (when the input is around zero). Activation functions like sigmoid, tanh, and the step function max out at one, making them good for output layers (where they could be interpreted as a probability, for example).

* + - * 1. Multilayer perceptron

As mentioned earlier, perceptrons strung together in layers make for a powerful model and were thus called multilayer perceptrons (MLPs). MLPs are just another name for your standard neural networks. [Russell, 2010 #229] A diagram of this model can be seen in Fig. 9.

Diagram

Description automatically generated

Fig. 9 Diagram of a fully connected MLP, or standard neural network, with three hidden layers.

In an MLP, each input node corresponds to an input feature. Then, the inputs are typically passed to any number of hidden layers, or layers that are neither input nor output layers, where each input goes to every node in the next layer, and every node in the next layer receives every input (at least for a fully connected model). Finally, the weights are passed to an output layer, where the dimension of the target variables (often) dictates how many output nodes there are. A deep neural network is simply an MLP with many hidden layers.

* + - * 1. Generative Adversarial Networks

Neural networks start to get fancy very quickly. There are recurrent neural networks, transformer neural networks, and convolutional neural networks. One interesting network of note is generative adversarial networks (GANs). [Ian J. Goodfellow, 2014 #234] GANs generate new data by having one model that creates data, seeded with random noise, called a generator. A parallel model, called the discriminator, compares real samples to the generated, or fake, samples and the entire model is penalized if the discriminator can distinguish between real and fake inputs. Obviously, this process requires an intricate balance between training the generator and discriminator, and many modifications have been made to encourage this, such as using the Wasserstein distance metric in a Wasserstein GAN (WGAN).

Shape

Description automatically generated with low confidence

Fig. 10 A generative adversarial network (GAN) can make new realistic looking samples by balancing a discriminator and generator during training.

* + 1. Interpretability versus effectiveness

In general, deciding what model to use depends on the problem, but also on the desired interpretability of the results. While linear regression is the most limited in its scope, only working well for linear data, it is the most interpretable as the importance of features can be directly interpreted using the learned weights. Decision trees and SVMs are likely easy to interpret. The second tier in interpretability includes ensemble models, such as a random forest, as the majority voting hides the interpretation behind another layer. Finally, neural networks are the hardest to interpret as the vast amount of weights and layers make it difficult to correlate features to specific outcomes.

Text

Description automatically generated with medium confidence

Fig. 11 Summary of the interpretability versus accuracy (or strength) of each machine learning model.

* + 1. Metrics
       1. Regression metrics

Regression metrics quantitatively compare predictions to true values. The most common are mean squared error (MSE), root mean squared error (RMSE), and mean average error (MAE). A good way to visualize regression predictions is by making a correlation curve, or plotting true (target) values versus predictions, where a perfect model would fall along the y = x line.

* + - 1. Classification metrics

Reporting an appropriate classifications metric is critical as there are often drastically different interpretations of results depending on the metric you use; for example, it is more important to identify the probability of false negatives for cancer detection that it is to purely report accuracy. Here are some common metrics to consider.

Accuracy is the sum of true positives and true negatives divided by the total number of predictions. Accuracy is a good metric where distinguishing between false positives (Type 1 errors) and false negatives (Type 2 errors) does not matter. A good way to visualize all predictions is by forming a confusion matrix. Other metrics include precision (true positive divided by true positive plus false positive) and recall (true positive divided by true positive plus false negative). An important metric that is often reported is the F1 score as it combines precision and recall. It is defined as precision times recall dived by the sum of precision and recall. This metric punishes extremes values more and places importance on false positives and false negatives.

* + - 1. Cross validation

As stated earlier, cross validation is super important for determining any model hyperparameters and gauging the generalizability of your model. The most basic type of cross validation is k-fold cross validation. Here, the training data is split into k different chunks, where the model will be trained on k – 1 of the sections, and the last section will be used as a validation set. Throughout training repetitions, often called epochs in the case of neural networks, the validation set will systematically move between every k section. Then, an average and standard deviation can be calculated using the k predictions on the different validation sets.

* + 1. Uncertainty estimation

There are a handful of ways to formally estimate uncertainty of predictions from machine learning models (which is essential for scientific interpretation) that are better than just rerunning the model and seeing how much its predictions change.

* + - 1. Aleatoric versus epistemic uncertainty

First, let’s understand the basics of uncertainty from a statistical perspective. Uncertainty can be divided into two different types – aleatoric and epistemic. Aleatoric uncertainty is from the internal randomness of phenomena while epistemic uncertainty is from the lack of knowledge of the system, i.e., hidden variables in your system that may be affecting the outcome of the event. [Hüllermeier, 2021 #235]

* + - 1. Models that incorporate uncertainty

The following are a few models used in machine learning that formally incorporate uncertainty into their predictions.

* + - * 1. Gaussian Process

The first model to discuss is a Gaussian process (GP). GPs are non-parametric kernel-based methods that incorporate Bayes rule into their predictions. [Rasmussen, 2006 #145] Thus, they can give an estimate of uncertainty, or conversely confidence, to every output. A GP performs well on smoothly varying data, such as a multidimensional gaussian, but poorly on discontinuous or sharply featured data, such as a step function.

Chart

Description automatically generated

Fig. 12 A Gaussian process gives estimates of uncertainty depending on the location of the training data.

* + - * 1. Monte Carlo Dropout

The second and most common way to estimate uncertainty of a prediction from a neural network is by using dropout during predictions. Dropout means randomly setting weights in the network to zero, thus randomly canceling out some correlations. [Yarin Gal, 2016 #236] This process has been shown to estimate Bayesian uncertainty, and it is the easiest to implement as it does not require any architecture modifications.

Diagram

Description automatically generated

Fig. 13 Monte Carlo dropout during test predictions is the easiest and most common way to estimate uncertainty because it does not require special model architecture.

* + - * 1. Bayesian Neural Network

The third way to incorporate uncertainties is to use a Bayesian neural network, which essentially learns distributions instead of weights. This difference means twice as many parameters to learn, assuming Gaussian distributions, which can be described by a mean and standard deviation. [Goan, 2020 #237]

Diagram

Description automatically generated

Fig. 14 A Bayesian neural network learns distributions of weights instead of just the weights themselves.

* + - * 1. Mixed Density Network

Finally, the last model that incorporates uncertainty into predictions is a mixed density network. Here, the final output layer is expanded to include the parameters of the desired distributions (such as mean and standard deviation, not just the mean) and the loss function is adjusted accordingly to reflect this interpretation. [MacKay, 1995 #238]

Diagram

Description automatically generated

Fig. 15 A mixed density network is another easy implementation of a neural network that can formally account for uncertainty.

* 1. Unsupervised Machine Learning

Most of my work focused on unsupervised machine learning. Unsupervised machine learning is looking for trends in data, but with no target variable. Instead, it is purely a data-driven pattern recognition technique. Often, unsupervised machine learning is used to disentangle correlated features (features selection) or look for the most important information in your data. This analysis can speed up computations or make predictions more interpretable.

* + 1. Dimensionality Reduction
       1. The curse of dimensionality

The curse of dimensionality refers to the phenomenon that points that are far away in some dimensional space look even farther away in higher dimensions, following an exponential explosion. This trend means that points that might have started close now look just as far away from every other point. This phenomenon is from the r3 law, or that your Jacobian in high dimensions depends on the radius r by increasing orders of magnitude. This curse is an issue when trying to identify similarities in high-dimensional data. Thus, dimensionality reduction tries to solve this issue by finding a lower-dimensional representation of your original dataset. [Russell, 2010 #229]

* + - 1. Linear transformations
         1. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an algorithm that finds the eigenvalue and eigenvector decomposition of a dataset such that the eigenvectors are specially ordered to be in order of importance, where importance is described as “explaining the most variance in the dataset” or, equivalently, minimizing the distance data points need to be projected onto that eigenvector. [Press, 2007 #211]

Chart, scatter chart

Description automatically generated

Fig. 16 PCA tried to maximize explained variance, or equivalently minimize the distances needed for the data points to be projected onto that eigenvector (or basis vector).

* + - * 1. Singular Value Decomposition (SVD)

Singular value decomposition (SVS) is the more complete form of PCA and thus takes more computational power (because of a matrix inversion). [Press, #211] SVD represents the original matrix A as A = UDVT, where U is composed of left singular vectors, D is a matrix with the singular values along its diagonal, and V is composed of right singular vectors. SVD is unique up to rearranging the order of the singular values.

* + - * 1. Nonnegative matrix factorization (NMF)

Nonnegative matrix factorization (NMF) is like PCA, except it forces the eigenvalues to be positive, which is beneficial for physical variables that likewise can’t be negative. [Lee, 1999 #136] NMF calculates two non-negative matrices whose product reproduces the original dataset, or A = WH, where the column vectors in W are linearly combined using the coefficients in the columns of H.

* + - 1. Nonlinear transformations

Nonlinear transformations have strengths over linear methods because they can capture nonlinear trends in data. Effectively, they aren’t constrained to basis vectors that are (hyper)planes but can have a weaving manifold to explain the data, as shown in Fig. 17a.

Diagram

Description automatically generated

Fig. 17 The benefits of nonlinear dimensionality reductions algorithms. [Tetef, 2021 #140]

* + - * 1. Variational Autoencoder (VAE)

An autoencoder, the architecture shown in Fig. 17b, is a special type of neural network that is composed of two sequential networks – an encoder and a decoder. The encoder takes data, in the original d dimensions, and reduces it to a lower dimensional representation called a *latent space*. The decoder then takes the latent space and expands the dimensionality back to the original d dimensions. By trying to reconstruct the original input in the output layer, the autoencoder iteratively learns how to maximize the information being squeezed through the bottleneck layer, or latent space.

An autoencoder can then be modified to be a variational autoencoder (VAE) by learning a distribution in the latent space instead of a deterministic embedding. This modification is done by learning two parameters for each latent space dimension (instead of one) and interpreting one as a mean and the other as a standard deviation. Then, the model will randomly vary the input going to the decoder by sampling from that learned mean and standard deviation. This property allows the latent space to be generative and thus create new data that it hasn’t seen before by interpolating through the latent space. [Hinton, 2006 #101]

* + - * 1. t-distributed Stochastic Neighbor Embedding (t-SNE)

t-distributed stochastic neighbor embedding (t-SNE) [van der Maaten, 2008 #114] is a tool that is often used for visualizing data in two or three dimensions (it doesn’t work in higher dimensions) but can be used as a dimensionality reduction technique as well. t-SNE creates a graph-based similarity representative of the original data by calculating the joint conditional probability distribution between every pair of points to effectively get a similarity matrix. It then projects the data points to a lower dimension (say two) and tries to match the distances between each data point in the reduced space such that the subsequently calculated similarity matrix in the lower dimensions matches the one originally generated in the higher dimension. Ideally, this process means that there is no data compression or loss of information since the similarity between points is retained. This process is demonstrated in Fig. 17c.

However, t-SNE generates a “non-parametric” embedding, which means that it needs the entire dataset every time it makes a lower dimensional representation. There are no weight vectors it can save to do a transformation later. “Non-parametric” does not mean it has no hyperparameters. In fact, the one hyperparameter it does have is called perplexity, which represents your suspected minimum cluster size.

Moreover, t-SNE is excellent for looking at local similarities, but at the cost of loss of global structure. That means you must interpret clustering the t-SNE reduced space carefully. Data points in the same cluster can be interpreted as similar, but distances between clusters cannot tell you exactly how similar or different the two clusters are, only that they are different.

* + - * 1. Uniform Manifold Approximation and Projection (UMAP)

Uniform Manifold Approximation and Projection (UMAP) [McInnes, 2020 #142] is very similar to t-SNE in that it constructs a graphic similarity representation of the original high-dimensional data and tries to match that similarity metric in the reduced dimensional embedding. However, because of a different choice in the cost function, it retains global similarity, unlike t-SNE. UMAP has two hyperparameters – the minimum distance between clusters and the average number of neighbors a data point in a cluster is expected to have. These hyperparameters let you tune how global versus local you want your similarity metric and thus how tightly your clusters appear in your reduced space.

* + 1. Clustering

Although a reduced representation of your data is beneficial in and of itself, it can be helpful to apply a clustering algorithm to that reduced space. Clustering algorithms work better on fewer dimensions exactly because of the curse of dimensionality. Moreover, clustering can help identify unbiased classes in your dataset and be a precursor to supervised machine learning.

* + - * 1. K-means

K-means clustering is a centroid-based clustering algorithm, i.e., it tries to balance the center of mass among all the clusters. K-means randomly place k centroids, where k is the number of clusters, and then it moves the centroid locations until they are the furthest apart from each other while being the center of mass for the nearby data points. Thus, k-means clustering works well if you know the expected number of clusters and the data has a Gaussian distribution (no Swiss rolls).

Diagram

Description automatically generated

Fig. 18 K-means clustering balances center of mass.

* + - * 1. K-nearest neighbors

K-nearest neighbors (KNN) [Fukunaga, 1975 #239] clustering is similar to K-means in that it is centroid-based in that it calculates the probability of a new point belonging to a class based on the nearby data points, where the hyperparameter k represents the expected number of members in each cluster. Thus, KNN is a **supervised** clustering approach, unlike most other unsupervised clustering algorithms. However, the algorithm allows clusters to be more globular or amorphous in shape.

* + - * 1. Density-based clustering

Dbscan [Hahsler, 2019 #147] is a density-based clustering algorithm, which means it looks at the density of nearby data to determine whether the points belong in the same cluster or not. It has one main hyperparameter, epsilon, which is effectively the expected radius of a cluster. The algorithm will glob data points together with the expectation that almost all points in one cluster will fall within a radius of epsilon from another other point in the same cluster.

Diagram

Description automatically generated

Fig. 19 dbscan uses the hyperparameter epsilon to determine the radius of an expected cluster.

* + - * 1. Agglomerative hierarchical clustering

Agglomerative hierarchical clustering [Murtagh, 1983 #240] is a recursive process that groups data points together one at a time until all data points belong to the same cluster. Using a divisive algorithm does the opposite – splitting the dissimilar data points from each other until every point belongs to its own cluster. Both result in a dendrogram, a tree-like graphical representation of your data, where “cuts” can be taken horizontally across them to determine clusters.

Chart

Description automatically generated with medium confidence

Fig. 20 Dendrograms represent similarity of points using the distances between them as a metric.

* + 1. Feature Selection
       1. Metrics

Feature selection is critical when input variables are correlated as it makes models more interpretable. The type of metric one might use to perform feature selection depends on the type of input and output variables, as shown in Fig. 21. Notice that all the metrics are supervised. For categorical inputs and outputs, as is often the case in scientific research, a chi-squared metric is traditionally used. Chi-squared can be separated into three main types: tests for goodness of fit, tests for independence, and tests for homogeneity. The “goodness of fit” chi-square test is its most common usage, which determines the probability that a variable comes from a specific distribution.

Diagram

Description automatically generated

Fig. 21 Supervised feature selection metrics for different types of input of output variables.

* + - 1. Recursive Feature Elimination (RFE)

Recursive feature elimination [Jeon, 2020 #185] is a wrapper-based feature selection method that, as the name suggests, recursively prunes the input, or feature space, such that the most important features remain. The algorithm decides the importance of features by training a base machine learning model, such as a linear regressor as shown in Fig. 22, and then calculating the prediction accuracy for that set of features. To remove a feature, it will repeat this process by enumerating through every combination of N- 1 features, where N is the current size of the feature set. In each step, it removes the feature that corresponded to the lowest accuracy. The algorithm continues until a desired number of features remains.

A screenshot of a computer

Description automatically generated with low confidence

Fig. 22 Recursive feature elimination picks the best features by correlating them with the best score (e.g., accuracy) of predicting the target variables. [Tetef, 2023 #232]

* 1. References