Linear Methods

Shyue Ping Ong

University of California, San Diego

NANO281

Overview

Preliminaries

- We will go very deep into linear models.
- Most of you probably have seen linear models in some form, but we will start from scratch to further illustrate key concepts such as bias and variance.
- We will then discuss techniques such as regularization and transformation of inputs in the context of linear methods.

Notation

- Capital letters, e.g., X denote variables.
- Lower-case letters e.g., x, denote observations.
- Dummy index j to denotes different variables, e.g., X_j
- Dummy index i to denotes different observations, e.g., x_i
- Bolded variables are vector/matrices, e.g., $\mathbf{y}$, $\mathbf{X}$

Linear Regression

Linear Regression

Simplest possible model between target and feature

$$Y = f(X_1, X_2, ..., X_p) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

X_j can be:

- Quantitative inputs
- Transformations of quantitative inputs, e.g., log, exp, powers, etc. Basis expansions, e.g., $X_2 = X_1^2$, $X_3 = X_1^3$
- Interactions between variables
- Encoding of levels of inputs

Supervised learning

- Given a set of paired observations $\{x_{ij}, y_i\}$, what are the model parameters (in this case, the coefficients β_j) that are "optimal"?
- "Optimal" is typically defined as minimization of some **loss function** (also known as **cost function**) that measures the error of the model.

Least squares regression

Consider the simple case of

$$Y = \beta_0 + \beta_1 X_1$$

In least squares regression, the loss function is defined as the sum squared error given the N observations:

$$
\begin{aligned}
L(Y, \hat{f}(X)) &= \sum_{i=1}^{N}(y_i - f(x_i))^2 \\
&= \sum_{i=1}^{N}(y_i - \beta_0 - \beta_1 x_{i1})^2
\end{aligned}
$$

What are the optimal parameters β_0 and β_1?

Derivation in class...

Reformulating the general multiple linear regression as a vector equation. . .

Considering N observations of

$$y_i = \beta_0 + \beta_1 x_{i1} + + \beta_2 x_{i2} + ... + \beta_p x_{ip}$$

Let

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ ... \\ \beta_p \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & ... & x_{1p} \\ 1 & x_{21} & x_{22} & ... & x_{2p} \\ \vdots & & & & \\ 1 & x_{N1} & x_{N2} & ... & x_{Np} \end{pmatrix},$$

So,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

Note that $\mathbf{y}$ is a $N \times 1$ vector, $\boldsymbol{\beta}$ is a $(p + 1) \times 1$ vector, and $\mathbf{X}$ is a $N \times (p + 1)$ matrix.

Reformulating the general multiple linear regression as a vector equation...

$$L = RSS = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Assuming (for the moment) that $\mathbf{X}$ has full column rank, and hence $\mathbf{X}^T\mathbf{X}$ is positive definite, It can be shown using the same principles that the following unique solution for $\boldsymbol{\beta}$ is:

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ \hat{\mathbf{y}} &= \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

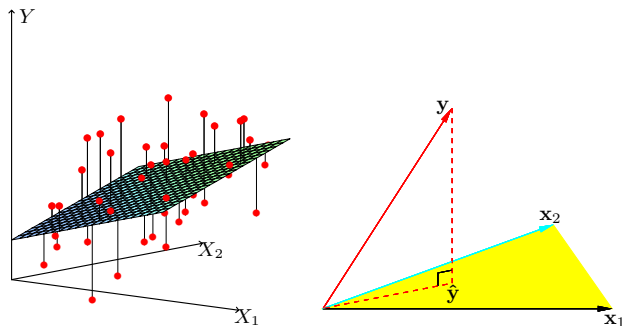# Graphic representation of MLR with two dependent variables

Figure: MLR minimizes sum square of residuals. The projection $\hat{\mathbf{y}}$ represents the vector of the least squares predictions onto the hyperplane spanned by the input vectors $\mathbf{x_1}$ and $\mathbf{x_2}$. [2].

Validity of least squares criterion

- Observations are independently drawn at random.
- Variance of **y** is constant given by σ^2.

$$\mathrm{var}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

- and σ is estimated using:

$$\sigma^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Hypothesis Testing for Coefficients

- To derive insights into a model, we often want to know which of the input parameters are the most relevant to the target.
- Under assumptions of the errors in y follow a Gaussian distribution $N(0, \sigma^2)$, the errors in $\hat{\boldsymbol{\beta}}$ also have a Gaussian distribution $N(\beta, (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2)$
- Hypothesis testing can be carried out for whether a particular β_j is 0 using the following test statistic:

$$t_j = \frac{\hat{\boldsymbol{\beta}}_{\boldsymbol{j}}}{\sigma\sqrt{v_j}}$$

where v_j is the jth diagonal element of $(\mathbf{X}^T\mathbf{X})^{-1}$. t_j has a t distribution with $N - p - 1$ degrees of freedom (dof).

Hypothesis Testing for Groups of Coefficients

- More often, we want to test groups of coefficient for significance. E.g., to the k levels of a categorical variable.
- We will use the following F statistic:

$$F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS}_1)/(p_1 - p_0)}{\mathrm{RSS}_1/(N - p_1 - 1)}$$

where RSS_0 is the RSS of the larger model with $p_0 + 1$ parameters and RSS_1 is the RSS of the smaller model with $p_1 + 1$ parameters with $p_0 - p_1$ parameters set to zero. The F statistic has a distribution of $F_{p_1 - p_0, N - p_1 - 1}$.

Gauss-Markov Theorem

- Consider the estimator $\hat{\theta}$ for a variable θ.

$$\begin{aligned} \text{MSE} &= E(\hat{\theta} - \theta)^2 \\ &= \text{var}(\hat{\theta}) + [E(\hat{\theta}) - \theta]^2 \end{aligned}$$

- The MSE can be broken down into the variance of the estimate itself and the square of the bias.

Gauss-Markov Theorem

The least squares estimator has the smallest variance among all linear *unbiased* estimators.

- However, there can be estimators that are biased with smaller MSE.

Example materials data

- Target: Bulk modulus of elements (from Materials Project)
- Candidate features:
  - Melting point (MP)
  - Boiling point (MP)
  - Atomic number (Z)
  - Electronegativity (χ)
  - Atomic radius (r)
- Question: Why these features?
- We will add some transformations of these inputs as well, i.e., the square and square root of the electronegativity and atomic radius.

Using pandas for easy data manipulation

```python
import pandas as pd
# Read in data and set first column as index.
data = pd.read_csv("element_data.csv", index_col=0)
# Generate transformations as additional columns.
data["X^2"] = data["X"] ** 2
data["sqrt(X)"] = data["X"] ** 0.5
data["r^2"] = data["r"] ** 2
data["sqrt(r)"] = data["r"] ** 0.5
# Define our features, which is all the columns
# excluding K, which is the target.
features = [c for c in data.columns if c != "K"]
x = data[features]
y = data["K"]
```

Recommendation: Go through the 10 minute guide to pandas.

MLR in scikit-learn

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(x, y)
print(ref.coef_)
print(reg.intercept_)
```

- Note that x should contain the features only - there is no need to add a 1 column for the intercept. By default, the parameter fit_intercept in sklearn.linear_model.LinearRegression is True. You can set it to False to do a MLR without intercept.
- Documentation: link.

Model selection

Model selection

Model performance

- We will take a brief digression into model assessment and selection before continuing on to other linear methods.
- Model performance is related to its performance on *independent test data*, i.e., one cannot simply report a model's performance on training data alone.
- Note that this section is deliberately limited to high level concepts that are needed to continue further in exploration of linear methods. A more detailed discussion will be performed in later lectures.

Typical measures of model performance

- Mean squared error (MSE):

$$L(Y, \hat{f}(X)) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Mean absolute error (MAE):

$$L(Y, \hat{f}(X)) = \frac{1}{N} \sum_{i=1}^{N} |y_i - f(x_i)|$$

- Test error: L over independent test set.
- Training error: L over training set.

Training and test errors with model complexity

- Model complexity increases as the number of parameters increases (e.g., number of independent variables in MLR).
- Training errors **always** decrease with increasing model complexity.
- However, test errors do not have a monotonic relationship with model complexity. Test errors are high when model complexity is too low (underfitting) or too high (overfitting).
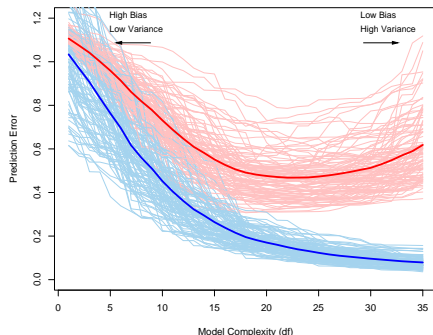
Training, validation and test data

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
- Ideal data-rich situation: Divide data into three parts:
  - Training set: For training the model.
  - Validation set: For estimating prediction error to select the model.
  - Test set: For assessing the generalization error of the final model.
- Typical training:validation:test split is 50:25:25 or 80:10:10, or in very data-poor situations, maybe even 90:5:5.
- Note that at no point in the model fitting process should the test set be "seen".

K-fold cross validation (CV)

- Simplest and most widely used approach for model validation.
- Data set is split into K buckets (usually by random).
- Typical values of K is 5 or 10. $K = N$ is known as "leave-one-out" CV.

Train	Train	Validate	Train	Train

- CV score is computed on the validate data set after training on the train data:

$$CV(\hat{f}^{-k(i)}, \alpha) = \frac{1}{N_{k(i)}} \sum_{i=1}^{N_{k(i)}} L(y_i, \hat{f}^{-k(i)}(x_i, \alpha))$$

- assuming the kth data bucket has $N_{k(i)}$ data points and $\hat{f}^{-k(i)}$ refers to the model fitted with the kth data left out ($N - N_{k(i)}$ data in fitting).

CV in scikit-learn

```
from sklearn.model_selection import cross_validate, KFold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
cv_results = cross_validate(ridge, z, y, cv=kfold)
```

- Note that we have customized the KFold object passed to the cross_validate method. The reason is that our element data is non-random by default. So we want to perform shuffling prior to doing the splits.
- Documentation: link.

Characteristics of the example materials dataset

- Before proceeding further, let us try to tease out some aspects of the dataset.
- Quite clearly, there are correlations between some sets of variables.
- In other words, the input features are **non-orthonormal** with each other.
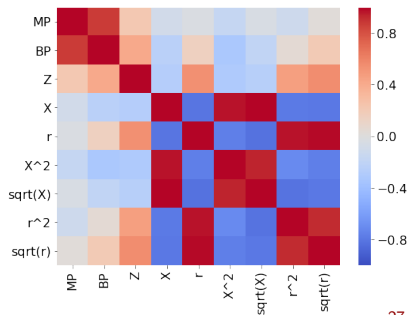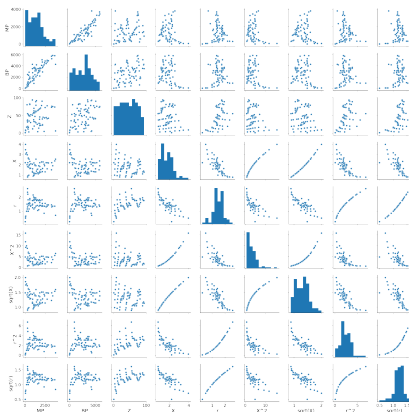
Beyond least squares

Beyond least squares

Model selection

- Often, we want to improve on the least squares model.
  - To improve prediction accuracy by sacrificing some bias for reduced variance.
  - To improve interpretability by reducing number of features or descriptors.
- Three main approaches:
  1. Subset selection
  2. Shrinkage methods
  3. Dimension reduction

Subset selection

Best subset selection

- Brute force approach.
- From p parameters, find the subset of k parameters that results in the smallest RSS.
- Combinatorially expensive for large p and large k.
- Note that the best subset for a larger k does not necessarily include the best subset for a smaller k.

Forward- or backward-stepwise selection

- Forward: Start with intercept, and iteratively add feature that most improves the fit.
- Backward: Start with full model, and sequentially deletes the feature with least impact on the fit.

Shrinkage methods

- Subset methods is discrete, i.e., retains/discards variables, and tends to exhibit high variance.
- Shrinkage methods are more continuous and do not suffer as much from high variability.
- Basic concept: instead of finding the parameters that minimizes the RSS only, we add a penalty term that penalizes more complex models, e.g., models with larger coefficients or larger number of coefficients. This "shrinks" the coefficients, in some cases, to 0.

Ridge regression (L_2 regularization)

$$\hat{\beta^{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- $\lambda \geq 0$ is the shrinkage parameter. The larger the λ, the greater the shrinkage.
- Also equivalent to:

$$\hat{\beta^{ridge}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j)^2$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t$$

Ridge regression - Key details

- Intercept (β_0) is not part of penalty term.
- Inputs should be scaled prior to performing ridge regression, typically by centering to the mean and scaling to unit variance:

$$z_j = \frac{x_j - \mu_{x_j}}{s_{x_j}}$$

LASSO (L_1 regularization)

$$\beta^{L\hat{A}SSO} = \underset{\beta}{\mathrm{argmin}} \left\{ \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- Least Absolute Shrinkage and Selection Operator
- $\lambda \geq 0$ is the shrinkage parameter. The larger the λ, the greater the shrinkage.
- Also equivalent to:

$$\beta^{L\hat{A}SSO} = \underset{\beta}{\mathrm{argmin}} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_j)^2$$

$$\text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t$$

LASSO regression - Key details

- Intercept (β_0) is not part of penalty term.
- Inputs should be scaled prior to performing lasso regression, just as in ridge regression.

Subset vs ridge vs LASSO

- Consider a set of orthonormal features.
  - Ridge: proportional shrinkage. No coefficients are set to zero.
  - LASSO: "soft" thresholding. Translates coefficients by a factor, truncating at zero.
  - Best-subset: "hard" thresholding. Drops all coefficients below a certain threshold.
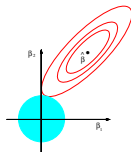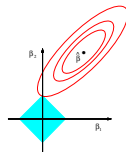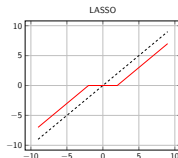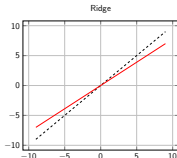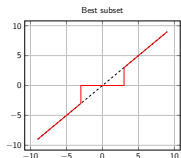
Other variants of shrinkage methods

- Elastic net penalty:

$$\lambda \left(\alpha \sum_{j=1}^{p} \beta_j^2 + (1 - \alpha) \sum_{j=1}^{p} |\beta_j| \right)$$

- Least angle regression

Derived input directions

- General concept: transforms input $\mathbf{X}$ into a smaller subset of $\mathbf{z_m}$ and regress on $\mathbf{z_m}$
- Principal component regression:
    - Transform non-orthonormal features into orthonormal directions using Principal Component Analysis (PCA).
    - Choose M directions that have the highest eigenvalues (explains the most variance) and discards the rest.
    - Will revisit at a later lecture.

Partial Least Squares (PLS)

- Algorithm:
  1. Compute $\phi_{1i} = <\mathbf{x_j}, \mathbf{y}>$ for each j.
  2. First transformed direction $\mathbf{z_1} = \sum_j \phi_{1i}\mathbf{x_j}$, i.e., each direction is weighted by strength of effect on $\mathbf{y}$.
  3. Regress $\mathbf{y}$ on $\mathbf{z_1}$ to obtain θ_1, orthogonalize $\mathbf{x_1}, ...\mathbf{x_p}$ wrt $\mathbf{z_1}$ via $x'_j = x_j - \frac{<\mathbf{z_1}, \mathbf{x_j}>}{<\mathbf{z_1}, \mathbf{z_1}>}\mathbf{z_1}$.
  4. Repeat until $M \leq p$ coefficients are obtained.

- Finds directions with high variance and high correlation with response.

Preliminaries

- It is highly unlikely that the true function $f(X)$ is linear in X.
- In some cases, linearity is a reasonable assumption, e.g., a first order Taylor series expansion:

$$f(x) = f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2!} + f'''(a)\frac{(x-a)^3}{3!} + ...$$

- Examples where this is used in materials science - linear elasticity (Hooke's law), etc.
- More frequently, we perform a transformation of inputs to create a linear basis expansion.

General concept

- Express:

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

  where h_m is the mth transformation of X.

- This is known as a linear basis expansion in X.
- The key lies in choice of the basis functions h_m.

Examples of basis expansions

- $h_m(X) = X_j^2, h_m(X) = X_i X_j$
  - Polynomial expansion to higher-order Taylor series terms.
  - No. of terms increases exponentially with degree of polynomial. For p variables, we have $O(p^2)$ square and cross-product terms in a quadratic model. For a degree d polynomial, we have $O(p^d)$.
- $h_m(X) = log(X_j), sqrt(X_j), exp(iX_j)$: non-linear transformations in X.
- $h_m(X) = I(L_m \leq X_k < U_m)$: Piece-wise division of regions of X.
  E.g., cubic splines.
- $h_m(X) = RBF(||X - X_m||)$: radial basis function, e.g., Gaussian.
- Typically, basis functions are used simply to allow a more flexible representation of the data. The basis functions can span a very large (sometimes infinite) set, from which a selection has to be made:
  - Restriction - Truncate the choice of basis functions using some criteria.
  - Selection - Choose basis functions that contribute significantly to the fit.
  - Regularization - Use the whole and/or very large subset and apply regularization techniques (e.g., ridge or LASSO) to restrict coefficients.
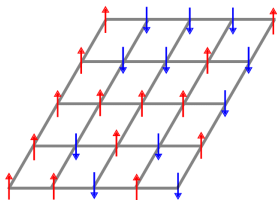
Linearization from physical laws

- Arrhenius law:

$$r = A\exp(-\frac{E_a}{RT}) \longrightarrow log(r) = log(A) - \frac{E_a}{RT}$$

- Ising model:

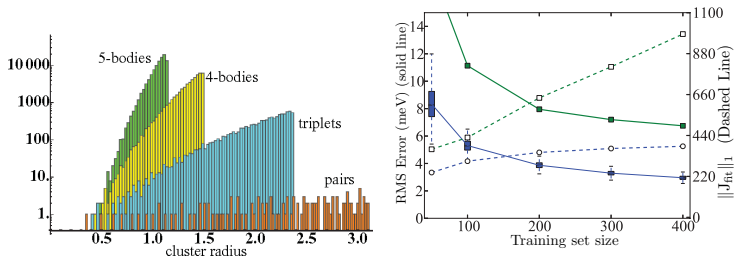$$H(\sigma) = -\sum_{<i,j>} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j$$

Compressive sensing for cluster expansions

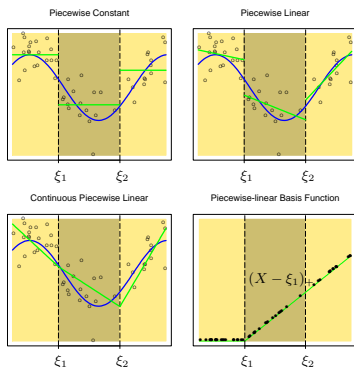- Cluster expansion of energy on lattice points:

$$H(\sigma) = E_0 + \sum_f J_f \prod_f (\sigma)$$

- σ is the vector representing occupation of lattice sites, $\prod_f$ are the cluster basis functions, J_f are effective cluster interactions (ECIs).
- Compressive sensing: essentially a LASSO to solve for ECIs.[3]

Piecewise polynomials

$$h_1(X) = I(X < \xi_1), h_2(X) = I(\xi_1 \leq X < \xi_2), h_3(X) = I(X \geq \xi_2)$$
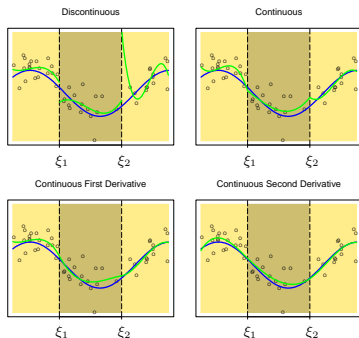

Parameters:

- No. of knots
- Order of polynomial
- Continuity at knots (value, first derivative, second derivative, etc.). For a polynomial of order N, we usually want all derivatives $< N$ to be continuous.

Cubic splines

Piecewise Cubic Polynomials

Discontinuous | Continuous

ξ_1 ξ_2 ξ_1 ξ_2

Continuous First Derivative | Continuous Second Derivative

ξ_1 ξ_2 ξ_1 ξ_2

- Probably the most commonly used.
- Continuous 1st and 2nd derivatives.
- Natural cubic spline: polynomial is linear beyond boundaries.
- Smoothing spline: Use regularization to control complexity:

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2$$
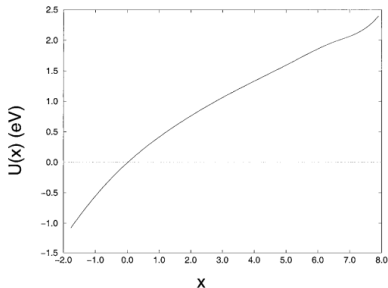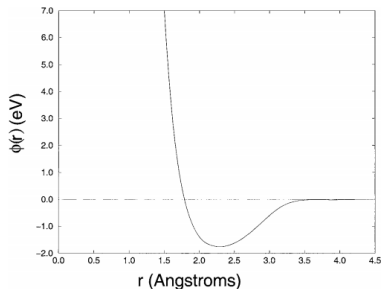$$+ \lambda \int \{f''(t)\}^2 dt$$

Examples of cubic spline fitting

- Spline-based Modified Embedded Atom Method (MEAM)

$$E = \sum_{i<j} \phi(r_{ij}) + \sum_i U(n_i),$$

$$n_i = \sum_j \rho(r_{ij}) + \sum_{i<k, j, k!=i} f(r_{ij}) f(r_{ik}) g[cos(\theta_{jik})]$$

where ϕ, U, ρ, f and g can be approximated by cubic splines.

Demo: Cubic spline fitting in scipy

```python
import numpy as np
## Import CubicSpline from scipy
from scipy.interpolate import CubicSpline

## x, y data for generating the spline fitting
x = np.arange(10)
y = np.sin(x)

## Fit the spline
cs = CubicSpline(x, y)

## Generate new x values
xs = np.arange(-0.5, 9.6, 0.1)

## Perform the interpolation on the new points
ys = cs(xs)
```

Gaussian basis functions

$$h_m(x) = \exp(-k(x - x_m)^2)$$

- Gaussian functions centered at x_m.
- Other similar types of functions include Lorentzian ($h_m(x) = \frac{1}{1+kx^2}$), Gaussian-Lorentzian, Voigtian, Pearson type IV, and beta profiles.
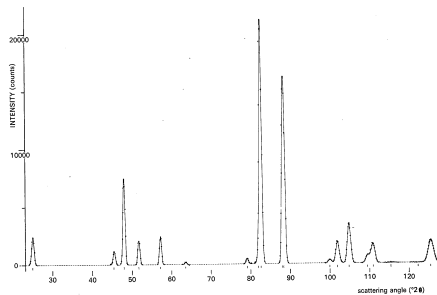
Example: Rietveld refinement

Figure: Neutron powder diffraction diagram of $CaUO_4$

- Least squares fitting of theoretical line profile to match a measured diffraction pattern (e.g., X-ray, neutron).[4]

# Example: Rietveld refinement, contd.

- Peak shape function:

$$PSF(\theta) = \Omega(\theta) \otimes \Lambda(\theta) \otimes \Psi(\theta) + b(\theta)$$

- Ω: Instrument broadening, Λ: Wavelength dispersion, Ψ: Specimen function.

- For single phase, minimize:

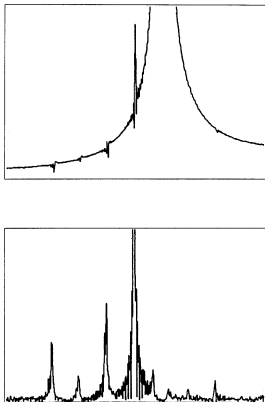$$\Phi = \sum_{i=1}^{N} w_i \left(Y_i^{obs} - \left(b_i + K \sum_{j=1}^{m} I_j y_j(x_j) \right) \right)^2$$

- where $y_j(x_j)$ is typically a pseudo-Voigt (mix of Gaussian and Lorentizan function) function.

- Note that the background (b_i) holds no useful structural information and should be minimized in experiments.

Wavelet smoothing
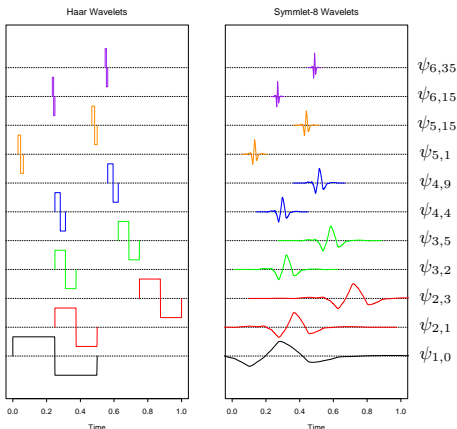
- Complete orthonormal basis
- Shrink and select toward **sparse** representation.
- Able to represent both time and frequency localization efficiently (Fourier basis can only do frequency localization).

Example: NMR Spectroscopy

Applications:

- Suppression of large unwanted spectral line (left).
- Rephasing spectrum perturbed by time-dependent magnetic field.
- Noise filtering
- Detecting phases in a mixture

Figure: Subtraction of a large spectral line: (top) the original spectrum of polyethylene, (bottom) reconstructed spectrum after removal of CH_2 peak.[1]

Example: Fourier transform for analysis of extended X-ray absorption fine structure (EXAFS)
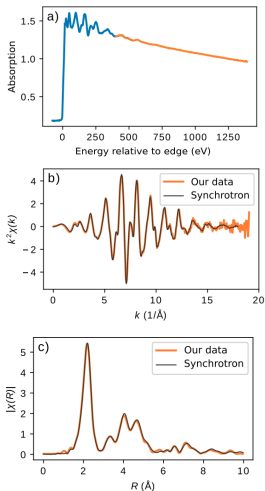
- (a) The extended edge (orange part) contains information of atom chemical environment.

- (b) Subtract the background, convert energy to k-space unit, and multiply the normalized intensity by k^2

- (c) Fourier transform k-space information to real space and obtain the first shell bond length.

Bibliography

📄 D. Barache, J-P. Antoine, and J-M. Dereppe.
The Continuous Wavelet Transform, an Analysis Tool for NMR
Spectroscopy.
Journal of Magnetic Resonance, 128(1):1–11, September 1997.

📄 Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
*The Elements of Statistical Learning: Data Mining, Inference, and
Prediction, Second Edition*.
Springer, New York, NY, 2nd edition edition, 2016.

📄 Lance J. Nelson, Gus L. W. Hart, Fei Zhou, and Vidvuds Ozoliņš.
Compressive sensing as a paradigm for building physics models.
Physical Review B, 87(3):035125, January 2013.

📄 H. M. Rietveld.
A profile refinement method for nuclear and magnetic structures.
Journal of Applied Crystallography, 2(2):65–71, June 1969.

The End