

ASTR8150/PHYS8150

Signal Analysis

Fourier Transforms, Gaussian Processes

Fabien Baron

Georgia State University

fbaron@gsu.edu

Fall 2021

DFT: Discrete Fourier Transform

- The Discrete Fourier Transform (DFT) transforms a sequence of N complex numbers $\{\mathbf{x}_n\} := x_0, x_1, \dots, x_{N-1}$ into another sequence of complex numbers, $\{\mathbf{X}_k\} := X_0, X_1, \dots, X_{N-1}$, which is defined by

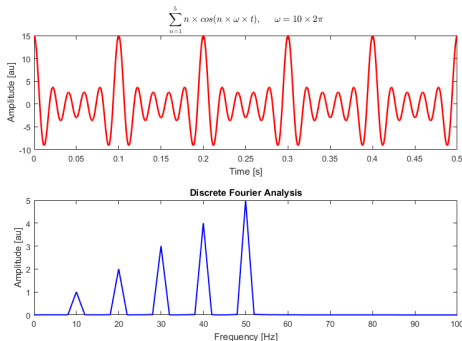
$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right) \right].$$

- The inverse transform is given by:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}$$

- The DFT is a linear transformation and thus can also be written in terms of a DFT matrix \mathbf{F}
- The transform is sometimes denoted by the symbol \mathcal{F} or \mathbf{F} , as in $\mathbf{X} = \mathcal{F}\{\mathbf{x}\}$ or $\mathcal{F}(\mathbf{x})$ or $\mathbf{F}\mathbf{x}$.
- When scaled appropriately it becomes a unitary matrix $\mathcal{F}\mathcal{F}^{-1} = I$ and the X_k thus be viewed as coefficients of x in an orthonormal basis.

FFT: Fast Fourier Transform



- Evaluating this definition directly requires $O(N^2)$ operations: there are N outputs X_k , and each output requires a sum of N terms. An FFT is any method to compute the same results in $O(N \log N)$ operations.
- The Fast Fourier Transform is computed on a uniformly sampled array and returns a uniformly sampled **spatial frequency decomposition**.

NFFT: Non-equispaced Fourier Transform

- The NFFT is a generalization of the FFT so that it works on a non-uniformly sampled signal and/or returns non-uniformly sampled spatial frequency decomposition.

Variance and covariance: a reminder

- Variance of a scalar-valued random variable X :

$$\sigma_X^2 = \text{var}(X) = E[(X - E[X])^2] = E[(X - E[X]) \cdot (X - E[X])]$$

- Covariance between two scalar-valued random variables X and Y :

$$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

- Covariance matrix of random vector \mathbf{X} :

$$\begin{aligned}\Sigma_{\mathbf{X}\mathbf{X}} &= \begin{bmatrix} E[(X_1 - E[X_1])(X_1 - E[X_1])] & E[(X_1 - E[X_1])(X_2 - E[X_2])] & \cdots & E[(X_1 - E[X_1])(X_n - E[X_n])] \\ E[(X_2 - E[X_2])(X_1 - E[X_1])] & E[(X_2 - E[X_2])(X_2 - E[X_2])] & \cdots & E[(X_2 - E[X_2])(X_n - E[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - E[X_n])(X_1 - E[X_1])] & E[(X_n - E[X_n])(X_2 - E[X_2])] & \cdots & E[(X_n - E[X_n])(X_n - E[X_n])] \end{bmatrix} \\ &= E[(\mathbf{X} - E[\mathbf{X}])^\top (\mathbf{X} - E[\mathbf{X}])]\end{aligned}$$

- Cross-covariance matrix of random vectors \mathbf{X} and \mathbf{Y} :

$$\Sigma_{\mathbf{X}\mathbf{Y}} = E[(\mathbf{X} - E[\mathbf{X}])^\top (\mathbf{Y} - E[\mathbf{Y}])]$$

Linear regression

- $y_i = f(x_i) + \epsilon_i = \theta_1 + \theta_2 x_i + \epsilon_i$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- In matrix notation

$$\begin{aligned}\mathbf{Y} &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \end{bmatrix}^\top \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \boldsymbol{\epsilon} \\ &= \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}^\top \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \boldsymbol{\epsilon} = \mathbf{X}^\top \boldsymbol{\theta} + \boldsymbol{\epsilon} \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})\end{aligned}$$

- Non-diagonal $\boldsymbol{\Sigma}$ will be used in the case the data points are covariant
- Likelihood of $\boldsymbol{\theta}$:

$$\mathcal{L}(\boldsymbol{\theta} | \mathbf{Y}) = \Pr(\mathbf{Y} | \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{Y} - \mathbf{X}^\top \boldsymbol{\theta})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \mathbf{X}^\top \boldsymbol{\theta})}$$

Linear regression: mean solution and predictor

- Likelihood of θ :

$$\mathcal{L}(\theta|\mathbf{Y}) = \Pr(\mathbf{Y}|\theta) = \frac{1}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{Y} - \mathbf{X}^\top \theta)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{Y} - \mathbf{X}^\top \theta)}$$

- Maximum likelihood (take the log, derive with respect to θ) results in the **normal equation** giving the most likely θ :

$$\hat{\theta} = (\mathbf{X}\boldsymbol{\Sigma}^{-1}\mathbf{X}^\top)^{-1}\mathbf{X}\boldsymbol{\Sigma}^{-1}\mathbf{Y}$$

- Given $\hat{\theta}$ and any new \mathbf{X}_* , we can now predict \mathbf{Y}_* , using the predictor **projection "hat" matrix \mathbf{H}** defined as:

$$\mathbf{Y}_* = (\mathbf{X}_*)^\top \hat{\theta} = \underbrace{(\mathbf{X}_*)^\top (\mathbf{X}\boldsymbol{\Sigma}^{-1}\mathbf{X}^\top)^{-1} \mathbf{X}\boldsymbol{\Sigma}^{-1}}_{\mathbf{H}} \mathbf{Y} = \mathbf{H}\mathbf{Y}$$

- All this is not fully Bayesian... What about $\Pr(\hat{\theta})$, or $\Pr(\mathbf{Y}_*)$?

Marginal and Conditional Gaussians

- Important theorem, used when both the prior and likelihood are normally distributed. If we have:

$$\begin{aligned}\Pr(\mathbf{x}) &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \\ \Pr(\mathbf{y}|\mathbf{x}) &\sim \mathcal{N}(\mathbf{Ax} + \mathbf{b}, \boldsymbol{\Sigma})\end{aligned}$$

then we will have

$$\begin{aligned}\Pr(\mathbf{y}) &\sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \boldsymbol{\Sigma} + \mathbf{A}\boldsymbol{\Lambda}\mathbf{A}^\top) \\ \Pr(\mathbf{x}|\mathbf{y}) &\sim \mathcal{N}((\boldsymbol{\Lambda}^{-1} + \mathbf{A}^\top \boldsymbol{\Sigma}^{-1} \mathbf{A})^{-1} (\mathbf{A}^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}^{-1} \boldsymbol{\mu}), (\boldsymbol{\Lambda}^{-1} + \mathbf{A}^\top \boldsymbol{\Sigma}^{-1} \mathbf{A})^{-1})\end{aligned}$$

- Reference: Bishop "Pattern Recognition and Machine Learning", eqs 2.113 to 2.117

Application to the Linear Regression case

- Somewhat confusing, but we have: $\mathbf{y} \rightarrow \mathbf{Y}$, $\mathbf{x} \rightarrow \boldsymbol{\theta}$, $\mathbf{A} \rightarrow \mathbf{X}^\top$, $\mathbf{b} \rightarrow \mathbf{0}$, and $\Sigma \rightarrow \Sigma$
- We need to set $\Pr(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ with possibly the edge case of $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Lambda}^{-1} \rightarrow \mathbf{0}$ to simulate a nearly uniform prior.
- We already have:

$$\Pr(\mathbf{Y}|\boldsymbol{\theta}, \mathbf{X}) \sim \mathcal{N}(\mathbf{X}^\top \boldsymbol{\theta}, \Sigma)$$

- The posterior distribution for $\boldsymbol{\theta}$, whose mean is the MAP:

$$\begin{aligned}\Pr(\boldsymbol{\theta}|\mathbf{Y}, \mathbf{X}) &\sim \mathcal{N}((\boldsymbol{\Lambda}^{-1} + \mathbf{X}\Sigma^{-1}\mathbf{X}^\top)^{-1} (\mathbf{X}\Sigma^{-1}\mathbf{Y} + \boldsymbol{\Lambda}^{-1}\boldsymbol{\mu}), (\boldsymbol{\Lambda}^{-1} + \mathbf{X}\Sigma^{-1}\mathbf{X}^\top)^{-1}) \\ &= \mathcal{N}(\boldsymbol{\Gamma}^{-1}\mathbf{X}\Sigma^{-1}\mathbf{Y}, \boldsymbol{\Gamma}^{-1}) \text{ for common case } \boldsymbol{\mu} = \mathbf{0}\end{aligned}$$

- And the predictive distribution is (applying the previous slide again):

$$\begin{aligned}\Pr(\mathbf{Y}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{Y}) &= \int \Pr(\mathbf{Y}_*|\boldsymbol{\theta}, \mathbf{X}_*, \mathbf{X}, \mathbf{Y}) \Pr(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\theta} \\ &\sim \mathcal{N}(\mathbf{X}_*^\top \boldsymbol{\Gamma}^{-1} \mathbf{X} \Sigma^{-1} \mathbf{Y}, \mathbf{X}_*^\top \boldsymbol{\Gamma}^{-1} \mathbf{X}_*)\end{aligned}$$

Beyond Linear Regression: basis functions

- Before we had $\mathbf{Y} = \mathbf{X}^\top \boldsymbol{\theta}$, but now we'd prefer a more flexible scheme $\mathbf{Y} = \boldsymbol{\phi}(\mathbf{X})^\top \boldsymbol{\theta}$ where $\boldsymbol{\phi}$ represent a function basis such as polynomials $\boldsymbol{\phi}(\mathbf{X}) = (1, x, x^2, x^3)^\top$. The matrix $\Phi(\mathbf{X})$ is the aggregation of columns $\boldsymbol{\phi}(\mathbf{X})$ into a matrix.
- Amazingly, the same analysis works, so that the predictive distribution becomes:

$$\Pr(\mathbf{Y}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{Y}) \sim \mathcal{N}(\Phi(\mathbf{X}_*)^\top \boldsymbol{\Gamma}^{-1} \Phi(\mathbf{X}) \boldsymbol{\Sigma}^{-1} \mathbf{Y}, \Phi(\mathbf{X}_*)^\top \boldsymbol{\Gamma}^{-1} \Phi(\mathbf{X}_*))$$

where $\boldsymbol{\Gamma} = \boldsymbol{\Lambda}^{-1} + \Phi(\mathbf{X})^\top \boldsymbol{\Sigma}^{-1} \Phi(\mathbf{X})$

- This means we can find predictive distribution for basis functions. But what if we don't want to specify any functional form for these functions ?

Gaussian process

- A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.
- A Gaussian process is a distribution over functions, rather than over variables
- We define the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ and we note $\mathbf{f} \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ so that:

$$m(\mathbf{x}) = E[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

The squared exponential kernel

- The squared exponential is a classic choice for a first exposition to Gaussian processes

$$\Sigma = \text{cov}(f(x), f(x')) = k(x, x') = \sigma_k^2 e^{-\frac{1}{2} \frac{(x-x')^2}{l^2}}$$

- It is parametrized by hyperparameter l , the characteristic length-scale of the process over which correlation is strong, and σ_k , the strength of correlation.
- Samples $\mathbf{Y} \sim \mathcal{N}(\mathbf{M}, \mathbf{K})$ can be generated by :
 - 1 computing Σ using the kernel expression over a given range for \mathbf{X} .
 - 2 computing the Cholesky decomposition \mathbf{L} of $\Sigma = \mathbf{L}\mathbf{L}^\top$
 - 3 computing $\mathbf{Y} = \mathbf{m} + \mathbf{L}\mathbf{u}$ where $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Demo:

<https://distill.pub/2019/visual-exploration-gaussian-processes/>

Predictions: posterior

- Typical example is data $\mathbf{Y} = f(\mathbf{X}) + \epsilon$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I})$.
- $\text{cov}(y_i, y_j) = k(x_i, x_j) + \sigma_n^2 \delta_{ij} \rightarrow \text{cov}(\mathbf{Y}) = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}$
- The joint distribution of the measurements and the predictions is:

$$\begin{bmatrix} \mathbf{Y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

- The predictive distribution is: $\Pr(\mathbf{f}_* | \mathbf{X}, \mathbf{Y}, \mathbf{X}_*) \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$, with

$$\begin{aligned} \bar{\mathbf{f}}_* &= \mathbf{K}(\mathbf{X}_*, \mathbf{X}) (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{Y} \\ \text{cov}(\mathbf{f}_*) &= \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}(\mathbf{X}_*, \mathbf{X}) \end{aligned}$$

Marginalization of kernel parameters

- The prior $\Pr(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ and likelihood $\Pr(\mathbf{Y}|\mathbf{f}) \sim \mathcal{N}(\mathbf{f}(\mathbf{X}), \sigma_n^2 \mathbf{I})$ give the marginal likelihood over the function values $\Pr(\mathbf{Y}|\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I})$
- For kernels with hyperparameters (e.g. σ_k and l for the squared exponential), one can write the expression for the log-marginal likelihood as a function of the hyperparameters, then optimize it.
- This allows us to find the best parameters supported by the data, and in turn to refine our future predictions.

Going beyond this introduction

- "Gaussian Processes for Machine Learning", Rasmussen & Williams, PDF downloadable here:
<http://www.gaussianprocess.org/gpml/>
- "Pattern Recognition and Machine Learning", Bishop, with examples by contributors in Matlab <http://prml.github.io/> or Python <https://github.com/ctgk/PRML>