Module 1: Numerical Methods

- Topic 1 (Python Basics)
  - Python vs. Matlab
  - Packages
  - Defining Functions
  - Plotting Data
- Topic 2 (Linear Algebra)
  - Matrix-vector multiplication
    - *Function that uses for loops to multiply matrix and vector*
    - *Vandermonde matrix*
  - Norms, inner products, and orthogonality
    - Inner product = dot product
    - *Create an orthonormal version of the Vandermonde matrix*
      - Both orthogonal and normalized
    - Orthogonal = inner product is zero (vectors are at right angles to each other AND have no projection onto each other)
      - We subtract off the projection of one vector onto another
  - Rank, Inverses, and Linear Systems
    - Rank = number of linearly independent columns/rows
      - Always less than or equal to the minimum of m and n (rows and columns)
      - Underconstrained vs overconstrained
      - Invertible
  - Eigen and Singular Value Decompositions
    - Calculate eigenvalues of a matrix with the eigvals function
    - Calculate both eigenvalues and eigenvectors with function eig
- Topic 3 (Linear Regression)
  - Simple linear regression
    - Use model of the form y = Xw + E if X is a first-order Vandermonde matrix
    - Epsilon is the “error” between the model and the actual data
    - We assume that the error follows a normal distribution
    - GOAL: use the data y_i to recover the “best fit” line
    - Derive linear regression by minimizing the sum of squared errors = find the line that gives the lowest squared errors
    - Set up a “cost function” that quantifies the squared errors
    - We want to take derivative of the loss function with respect to the parameters/weights w and set it equal to zero to get the weights that minimize our error

    - Polynomial regression
        - Use model of the form y = Xw + E if X is a higher-order Vandermonde matrix
        - y = X@w + epsilon
        - *Vandermonde function* (get nth order Vand. matrix, A= left hand side, b = right hand side, use linear algebra solver to get least squares regression weights, prediction (yhat) is dot product of Vand. matrix and new weights, sum of squared errors takes differences between y (X@w + epsilon) and yhat
        - *Compute the sum of squared errors as a function of the order of the polynomial used to fit the data*
    - General linear regression
        - Using the form y = Xw + E for many different types of linear regression
            - Different from generalized linear model (where the error term is assumed to follow a distribution other than normal)
        - Polynomials are not always a good fit, so we can use multiple Gaussian distributions (use the formula for new X values)
        - GOAL: find weights (parameters) that best match the data
        - If we add more Gaussian peaks, we can just use a set number of Gaussians with fixed standard deviation and space them evenly
        - *Gaussian features function*
        - The fit gets much better as more possible peaks are added
        - Issues with this approach
        - *Plot the Gaussian basis functions used in the previous code block*
    - Linear regression in scikit-learn
        - Solving our general linear regression models directly from linear algebra requires us to set up a linear system and solve it BUT *we can just use scikit-learn*, import LinearRegression, and fits & predicts the model
- Topic 4 (Numerical Optimization)
    - Nonlinear regression
        - Use pandas to read in excel file & Use matplotlib to plot
        - Sometimes we want to optimize models that are not linear → minimize the sum of squared errors by using our loss function that depends on 3 parameters (for Gaussian, weights, mu/mean, sigma/standard deviation)
        - We use a new vector lamda that contains all the parameters
        - Use multivariate calculus but getting derivative can be complicated
        - We use a *new Gaussian loss function* that takes a vector lamda
        - Test the loss function (result should be 0)

- Automatic Differentiation - NOT ON EXAM
- Gradient Descent
  - Newton's method (treat as root finding problem and use second derivative to iteratively optimize)
  - Gradient descent/ascent (increase or decrease the guess by "walking" along the gradient)
    - Use an initial guess and a fixed step size
  - Start with some initial guess and iteratively improve it
- Optimization with Scipy
  - We use scipy.minimize to do numerical optimization (without having to take a derivative)
  - BFGS method/algorithm (most commonly-used algorithm)
    - We have to use the function g that only takes a single argument (the variable we want to optimize with respect to)

Module 2: Regression

- Topic 1 (Non-parametric models)
  - Machine-Learning Perspective on Regression
    - Model inputs = features of a data point
    - Hperparameters = number of parameters to include (control the complexity of the final model)
    - Machine learning differs from regular regression because it seeks to optimize parameters (lamda) AND complexity (nu = hyperparameters) to obtain a model that generalizes to new input data
  - Non-parametric models
    - Basic idea: construct a loss function that quantifies how well your model fits the data → minimize the loss function with respect to the model parameters
    - Loss function could be sum of squared errors or some other measure of error
    - Parametric model = has parameters that do not explicitly depend on or include input points (ex. Polynomial regression → number of parameters is fixed with respect to the number of data points)(good for extrapolation)
    - Nonparametric model (good for interpolation)
    - *Piecewise linear*
  - Kernel regression
    - RBF = radial basis function (most commonly used kernel)

        - *RBF function*
- Topic 2: Model Validation
    - Accuracy Metrics
        - SST, SSE, R2 = *calc_stats*
        - MAE, RMSE, Parity plots, error histogram
    - Cross Validation
        - Hold out (train_test_split)
        - K fold (KFold(n_splits = #))
        - Leave p out
        - bootstrapping
    - Quantifying Error and Uncertainty
        - Standard deviation of error
        - Regression error
        - Bootstrapping (resampling)
        - Gaussian Process Regression
- Topic 3: Complexity Optimization
    - Information Criteria
        - BIC
        - AIC
    - Regularization
        - KRR = kernel ridge regression
    - LASSO Regularization
    - Hyperparameter Tuning
        - GridSearchCV
- Topic 4: High dimensional regression
    - Visualization of features
    - Scaling features and outputs
    - Multi-Linear Regression
    - Dimensionality Reduction
    - Principal Component Regression
        - Way to see measure r^2 values for each feature to see which one is most responsible for the variance in the data for high dimensional data