Part-1: Regression and Model Selection

21MAT311: Mathematics For Intelligent Systems - 6

Prepared by

Abhijith M S

School of Artificial Intelligence
Amrita Vishwa Vidyapeetham, Coimbatore

February 5, 2024

Contents

1. ML: AN Introduction
2. Regression
   - Linear Regression
   - Non-Linear Regression
   - Gradient Descent Method
3. Model Selection
4. Appendix

Current Section

1. ML: AN Introduction
2. Regression
   - Linear Regression
   - Non-Linear Regression
   - Gradient Descent Method
3. Model Selection
4. Appendix

Before we begin

- This course aims to discuss about the mathematical concepts behind machine learning (ML) algorithms.
- More time will be spend on concepts rather than on the implementation of any particular ML algorithm using Python/ MATLAB.
- So shall we begin with the most inevitable question of this course?

What is Machine Learning ?

Machine Learning

- Machine Learning is the science (and art) of programming computers so they can learn from data.
- Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed. - Arthur Samuel, 1959

Machine Learning

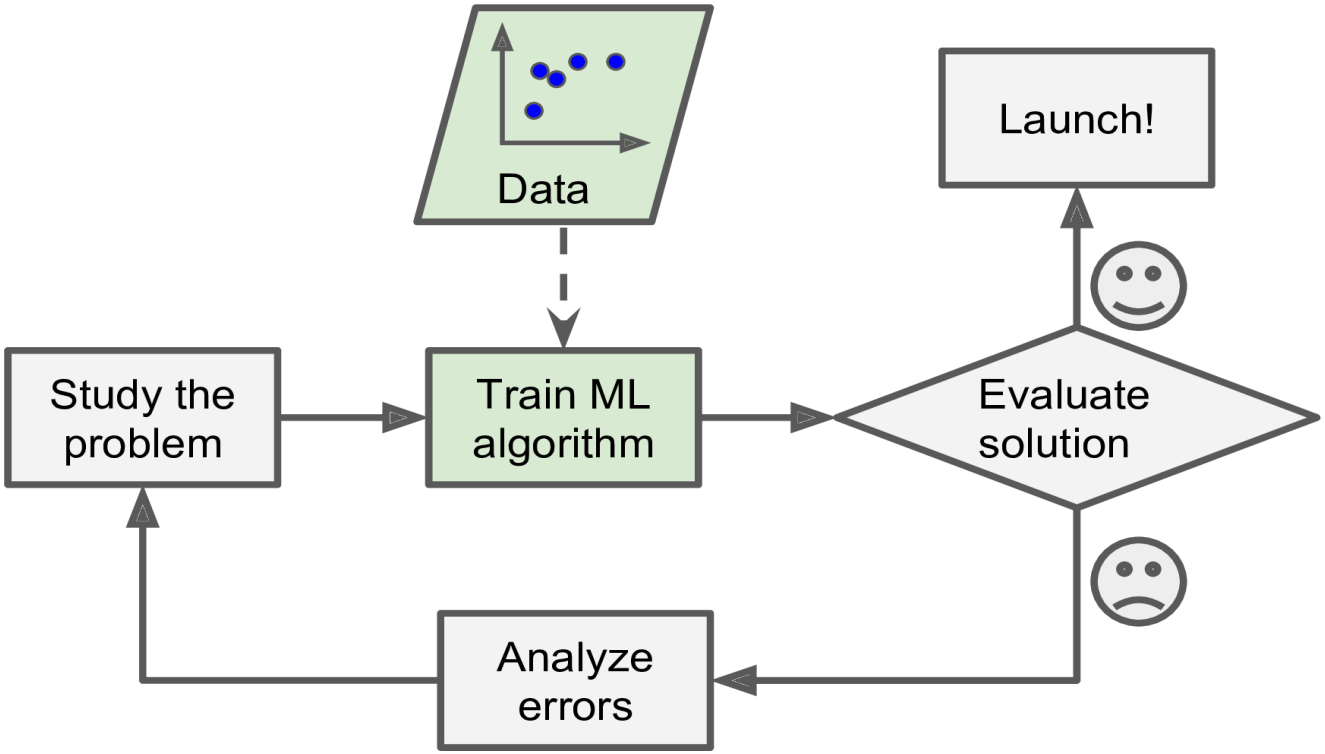

Figure 1: Machine Learning approach (*"Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow", Geron Aurelien, O'Reilly Media, Inc., (2022).*).

Machine Learning

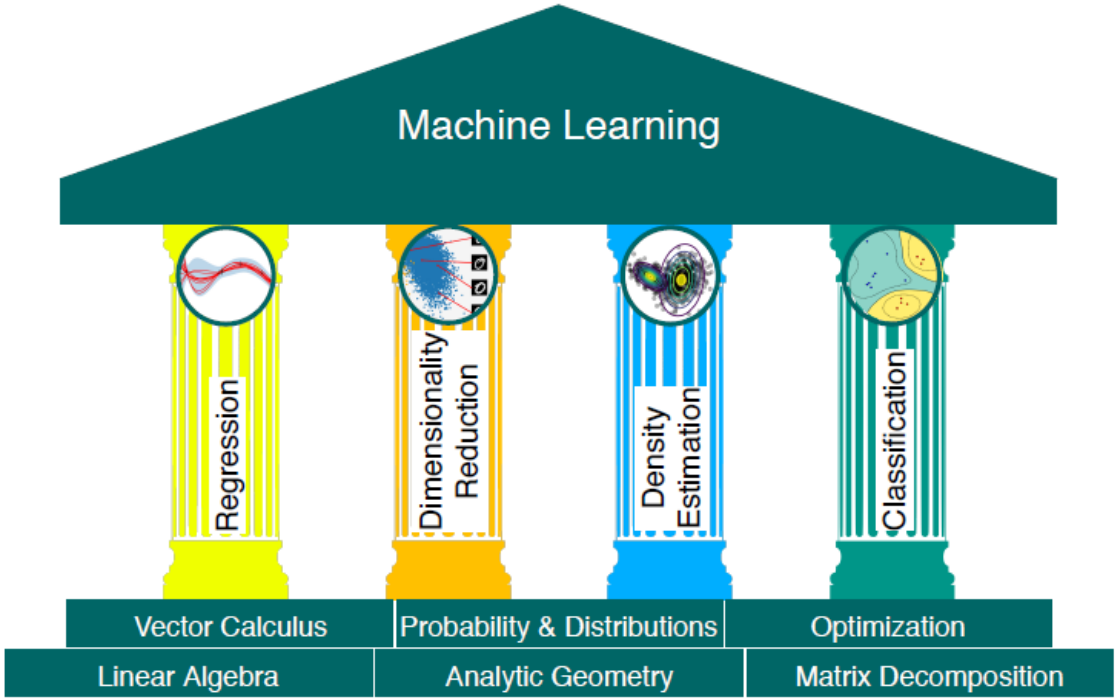

Figure 2: The foundations and four pillars of machine learning (*"Mathematics for Machine Learning", Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong, Cambridge University Press, (2020)*).

Machine Learning

- Machine learning can be broadly classified into supervised, unsupervised, semi-supervised and reinforcement learning.
- Here the focus is on: supervised and unsupervised learning.
- We start with Supervised learning: specifically with Regression.

Since a broad explanation on the basics of AI and ML is out of scope of this course, I leave the work to you. Please refer " Machine learning: a probabilistic perspective," by Kevin P Murphy, (MIT press, 2012) for more details.

Current Section

1 ML: AN Introduction

2 Regression
- Linear Regression
- Non-Linear Regression
- Gradient Descent Method

3 Model Selection

4 Appendix

Linear Regression

- Consider that the dataset consists of vectors **x** (input variable) and **y** (the output variable) with 'n' data instances.
- **x** and **y** $\in \mathcal{R}^n$.
- The intention here is to find a function of the form;

$$\mathbf{y} = f(\mathbf{x}, \beta) \tag{1}$$

  where β is vector with parameters and let $\beta \in \mathcal{R}^m$.

- We begin with linear regression, hence assuming a linear function f(**x**, β).

Linear Regression: Fit a straight line onto two datapoints

x	y
2	1
4	3

- Assume that our data set consists of only two data points, as given above.
- Here attempt is to fit a linear function of the form $\mathbf{y} = f(\mathbf{x}, \beta)$, and we go with (an equation for a straight line):

$$\mathbf{y} = \beta_1 \mathbf{x} + \beta_2 \tag{2}$$

where the unknowns β_1 is the slope and β_2 is the y-intercept of the straight line we fit.

Linear Regression: Fit a straight line onto two data points

- For two data points, we get a system of linear equations (two equations and two unknowns),as given below:

$$\beta_1 * (2) + \beta_2 = 1$$
$$\beta_1 * (4) + \beta_2 = 3$$

- The matrix form the above equation is:

$$\begin{bmatrix} 2 & 1 \\ 4 & 1 \end{bmatrix} \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 3 \end{Bmatrix} \tag{3}$$

- The above equation is of the form Ax = b. (What A, x and b represent in Eqn.(3))

Linear Regression: Fit a straight line onto two data points

- The Eqn.(3) (re-written below) is of the form Ax = b.

$$\begin{bmatrix} 2 & 1 \\ 4 & 1 \end{bmatrix} \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 3 \end{Bmatrix}$$

- Is Eqn.(3)) solvable?
- If solution exists, What is the nature of the solution of Eqn.(3))?
- What changes in Eqn.(3) will bring changes in the nature of the solution?

Linear Regression: Fit a straight line onto three datapoints

x	y
2	1
4	3
6	2

- Now consider that our data set consists of three data points, as given above.
- Here also we fit a straight line through the data (note that the number of unknowns remain unchanged compared to the previous case):

$$\mathbf{y} = \beta_1 \mathbf{x} + \beta_2 \tag{4}$$

where the unknowns β_1 is the slope and β_2 is the y-intercept of the straight line we fit.

Linear Regression: Fit a straight line onto two data points

- For three data points, we get an over-determined system of linear equations (with three equations and two unknowns), as given below:

$$\beta_1 * (2) + \beta_2 = 1$$
$$\beta_1 * (4) + \beta_2 = 3$$
$$\beta_1 * (6) + \beta_2 = 2$$

- The matrix form the above equation is:

$$\begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \end{bmatrix} \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 3 \\ 2 \end{Bmatrix} \tag{5}$$

- What is the nature of the solution of an over-determined system of linear equations?

Linear Regression: Fit a straight line onto two data points

Note:

- An over-determined system of equations can either have an exactly one solution or no solution at all.
- When do you think an over-determined system of equations has exactly one solution?

- The matrix equation is of the form: $\mathrm{A}\mathbf{x} = \mathbf{b}$

$$\begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \end{bmatrix} \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 3 \\ 2 \end{Bmatrix} \tag{6}$$

- As you might have figured out, in such cases we can't solve Ax = b, directly by taking an inverse of [A].

Linear Regression: Fit a straight line onto 'n' data points

- Extending the logic we have from the previous case, we can write an equation of the form:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}_{n\times 2} \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix}_{2\times 1} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{Bmatrix}_{n\times 1} \tag{7}$$

- Let $\beta = \begin{Bmatrix} \beta_1 \\ \beta_2 \end{Bmatrix}$ be the vector of unknown parameters.
- In most cases we expect an over-determined system (with more datapoints than the number of parameters).

Linear Regression: Fit a polynomial onto 'n' data points

- Any polynomial fit of any degree 'p', also reduces to a matrix equation of the form:

$$A\mathbf{x} = \mathbf{b}$$

$$\begin{bmatrix} x_1^p & x_1^{p-1} & \dots & x_1^2 & x_1 & 1 \\ x_2^p & x_2^{p-1} & \dots & x_2^2 & x_2 & 1 \\ \vdots & \vdots & & & & \\ x_n^p & x_n^{p-1} & \dots & x_n^2 & x_n & 1 \end{bmatrix}_{n\times(p+1)} \begin{Bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{(p+1)} \end{Bmatrix}_{(p+1)\times 1} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{Bmatrix}_{n\times 1} \tag{8}$$

- Here the unknown parameter vector $\beta = \begin{Bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{(p+1)} \end{Bmatrix}$ has (p+1) elements in it.
- Here also mostly we come across an over-determined system (with more datapoints than the number of parameters (i.e. n $>$ (p+1))).

Solve Ax = b: Over-determined systems

- As you have seen before, any type of linear regression ultimately reduces to an equation of the form: Ax = b. (Please don't confuse **x** with the input variables (also represented as **x**), here **x** represents the vector with unknown parameters or weights (β))
- We encounter an over-determined system with no exact solutions in most of the cases. (Reminder: an over-determined system can have an exact solution when **b** lies in the column space of A, but possibilities of that to occur in a real-life problem is very less.)
- So ending up in an equation of the form Ax=b, with no possible solutions for x, is something inevitable.
- So how we can find a suitable solution for **x**.

Key conclusions and a proper definition for 'Regression'

- As you might have figured it out, we usually find solution **x** (containing the model parameters (β)), exhibiting goodness-of-fit to the data.

- Before looking at how we determine the goodness-of fit let us give a proper definition for regression (which is applicable to both linear and non-linear types) as given below:

Regression assumes a general relationship between independent variables **X**, dependent variables **Y**, and some unknown parameters β as:

$$\mathbf{Y} = f(\mathbf{X}, \beta)$$

where the regression function "f(.)" is typically prescribed and the parameters β are found by optimizing the goodness-of-fit of this function to data.

Key Conclusions

- Importantly, regression discover relationships among variables by optimization.
- Broadly speaking, machine learning is framed around regression techniques, which are themselves framed around optimization based on data.
- Thus, at its absolute mathematical core, machine learning and data science revolve around positing an optimization problem.
- Of course, the success of optimization itself depends critically on defining an objective function to be optimized.

Goodness of Fit

- Different error metrics are employed to quantify the goodness of fit, such as;
  - Maximum Error (ℓ_∞ norm)

$$E_\infty(f) = \max_{1<i<n} |f(x_i) - y_i| \tag{9}$$

  - Mean Absolute Error (ℓ_1 norm)

$$E_1(f) = \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - y_i| \tag{10}$$

  - Least Square Error (ℓ_2 norm)

$$E_2(f) = \left(\frac{1}{n}\sum_{i=1}^{n} |f(x_i) - y_i|^2\right)^{\frac{1}{2}} \tag{11}$$

  - Additionally, one can more broadly consider the error based on the 'ℓ_p-norm. (What will be the equation for $\ell_p - norm$?)

Optimization as the Cornerstone of Regression

- In general, regression, so as any machine learning algorithm, can be considered as an optimization problem.
- Assuming the regression as an optimization, brings in a lot of flexibility.

- A linear-regression problem can be thought of as the following optimization problem:

$$\min \left((Ax - b)^T (Ax - b) + \quad \lambda_1 ||x||_1 + \quad \lambda_2 ||x||_2 \right) \qquad (12)$$

Different methods of Regression

$$\min \left((Ax - b)^T (Ax - b) + \quad \lambda_1 ||x||_1 + \quad \lambda_2 ||x||_2 \right) \tag{13}$$

Regression Type	λ_1	λ_2
Least-square (**pinv**)	0	0
LASSO	>0	$= 0$
Ridge	$= 0$	> 0
Elastic Net	> 0	> 0

Table 1: Regression Types

Write a MATLAB code to execute the following

- Generate an input variable (named 'x') of length 100×1 with random real number values (between 0 and 1).
- Generate an output variable, $y = x^3 + \mathcal{N}(\mu,\sigma)$. (use mean $\mu = 0$; and standard deviation, $\sigma = 0.1$).
- Fit a regression model (polynomial fit of degree 19) between x and y, using the following regression algorithms: (a). Least-Square Regression, (b). LASSO Regression, (c). Ridge Regression (d). Elastic net Regression. Compare the error values and also values of 20 parameters or weights obtained for each regression type.
- Also perform a least square polynomial fit from powers 1 to 19 upon the same data. Compare the errors obtained for each polynomial fit.
- How you may choose a good model out of different possibilities? What insights from this exercise you think will help you in choosing the model?

Non-linear Regression

- Earlier we defined regression as a method to find a general relationship between independent variables **X**, dependent variables **Y**, and some unknown parameters β as:

$$\mathbf{Y} = f(\mathbf{X}, \beta)$$

  where the regression function 'f(.)' is typically prescribed and the parameters β are found by optimizing the goodness-of-fit of this function to data.
- So far, we considered the regression function 'f(**X**, β)' to be linear (with respect β).
- What if the regression function is non-linear, say: Y = β_1 + sin(β_2X+β_3) + cos(β_4X+β_5) ?

Non-Linear Regression

- We have in a non-linear regression problem;

$$Y = f(X, \beta)$$

let X and Y $\in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$

- Here also we define the least square error as:

$$E_2(\beta) = \sum_{i=1}^{n} (f(x_i, \beta) - y_i)^2$$

- At the optimum (minimum) value of E_2;

$$\frac{\partial E_2}{\partial \beta_j} = 0 \text{ where j = 1, 2, . . ., m}$$

$$\frac{\partial E_2}{\partial \beta_j} = \sum_{i=1}^{n} (f(x_i, \beta) - y_i) \frac{\partial f}{\partial \beta_j} = 0$$

Non-linear Regression

- Hence in a non-linear regression problem, we end up in a system of linear equations.
- Unlike the system of non-linear equations, there are no general methods to solve the system of non-linear equations.
- Hence we use iterative methods and the convergence of which heavily depend on the initial guess we make.
- The iterative methods are primarily based on gradient descent.

Non-linear Regression

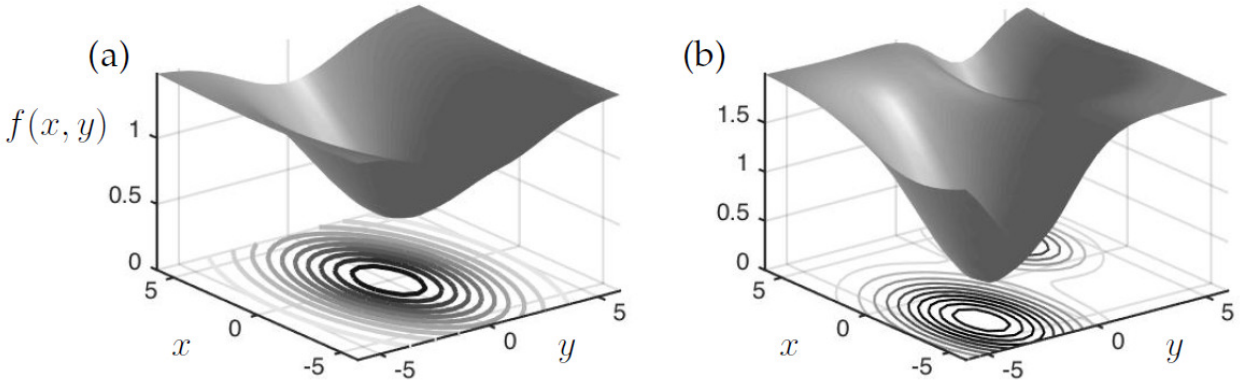

Figure 3: (a). Pure convex function and (b). Non-convex function

- The resulting objective function can be either convex or non-convex.
- Convex functions have many guarantees of convergence while using the gradient descent method.
- For non-convex functions, local minima and an inability to compute gradient directions (derivatives that are near zero) limit the success of gradient descent.

Gradient Descent Method

- For a high dimensional function f(x);

$$\nabla f(x) = 0$$

- The gradient (∇f(x)) of a function is always directed towards the direction in which f(x) increases.
- Hence the direction of steepest descent towards the minimum point of f(x), is given by - ∇f(x)
- The geometry of the steepest descent suggests the construction of an algorithm whereby the next point in the iteration is picked by following the steepest descent, so that,

$$x_{k+1}(\delta) = x_k - \delta \nabla f(x_k)$$

Gradient Descent

- The parameter δ determines how far to move along the gradient descent direction.
- The parameter is obtained by solving the equation below:

$$\frac{\partial F}{\partial \delta} = -\nabla f(x_{k+1}) \nabla f(x_k) = 0 \tag{14}$$

where,

$$F(\delta) = f(x_{k+1}(\delta))$$

- Eqn.14 indicates that the parameter δ is chosen in a way that the gradient value at the current iteration ($\nabla f(x_k)$) and that of the next iteration ($\nabla f(x_{k+1})$) are orthogonal.

Gradient Descent

- The convergence of the gradient descent method depends on the choice of the initial guess, and the parameter δ.
- Also, the gradient descent method requires the computation of the gradients from the data.
- A wide range of innovations have attempted to speed up this dominant nonlinear optimization procedure, including alternating descent methods.

Current Section

1. ML: AN Introduction
2. Regression
   - Linear Regression
   - Non-Linear Regression
   - Gradient Descent Method
3. Model Selection
4. Appendix

Model Selection: Over-fitting

- It is observed from one of the previous MATLAB coding problems that by increasing the model complexity blindly (by choosing a polynomial fit of higher degrees) the error never subdues but increases.
- We should avoid trying to model every minor variation in the input, since this is more likely to be noise than true signal.
- A high degree polynomial results in a curve that is very "wiggly". It is unlikely that the true function has such extreme oscillations. Thus using such a model might result in inaccurate predictions of future outputs.
- Overfitting affects generalizability of the model.
- In general overfitting can be tackled by employing cross-validation and computing information criteria.

k-fold Cross-Validation

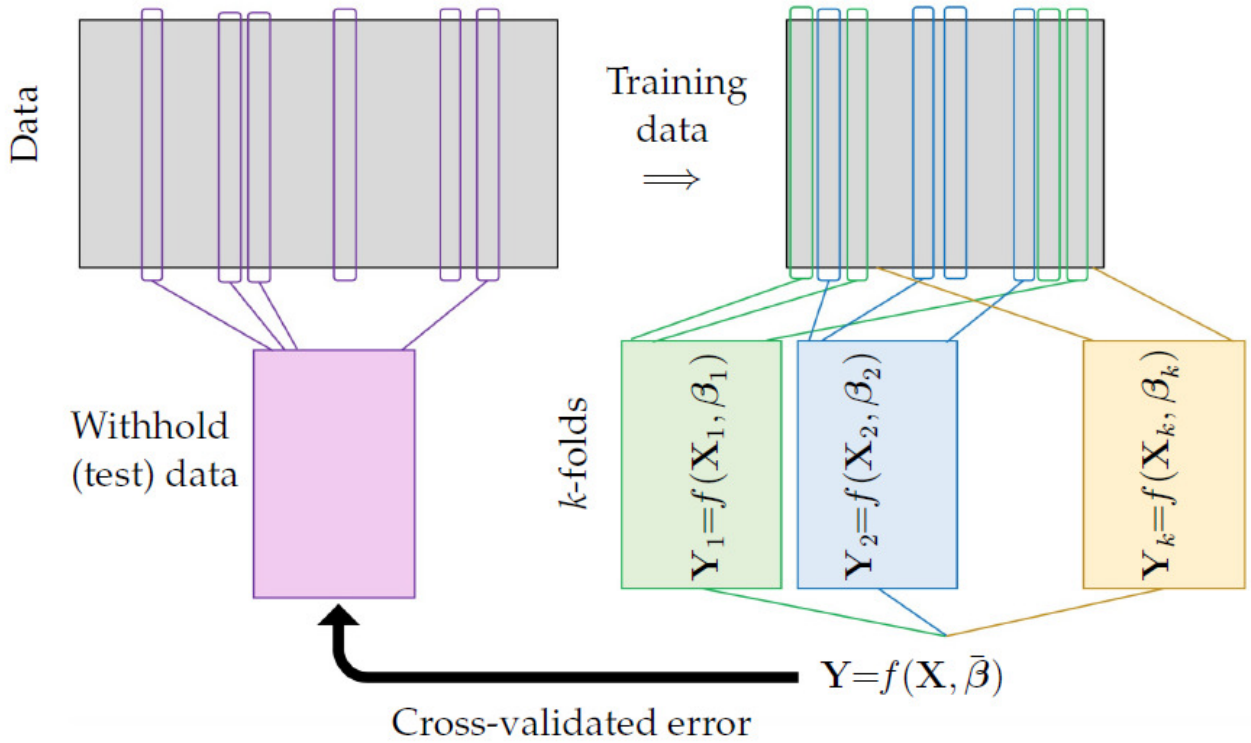

Figure 4: Procedure for k-fold cross-validation of models. (Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, by S. L. Brunton, and J. N Kutz)

Leave-p-Out Cross-Validation

- In this case, p samples of the training data are removed from the data and kept as the validation set.
- A model is built on the remaining training data, and the accuracy of the model is tested on the p withheld samples.
- This is repeated with a new selection of p samples until all the training data has been part of the validation data set.
- The accuracy of the model is then evaluated on the withheld data from averaging the accuracy of the models and the loadings produced from the various partitions of the data.

Information Criteria

- The model selection based on information criteria is based on the theory of Kullback - Leibler (KL) divergence.
- KL-divergence between two models f(X; β) and g(X; μ) is defined as;

$$\mathcal{I}(f,g) = \int f(X,\beta)\ ln\left(\frac{f(X,\beta)}{g(X,\mu)}\right)\ dX \tag{15}$$

  where β and μ are parameters of the models f(X; β) and g(X; μ), respectively.
- From an information theory perspective, the quantity $\mathcal{I}$(f; g) measures the information lost when g is used to represent f.
- If f = g, then the log term is zero (i.e., log(1) = 0) and I(f; g) = 0, so that there is no information lost.
- In practice, 'f' will represent the truth, or measurements of an experiment, while 'g' will be a model proposed to describe 'f'.

Information Criterion: AIC and BIC

- A Japanese statistician named Dr. Hirotugu Akaike combined maximum-likelihood estimation (MLE) with the KL divergence score to produce what is now called the Akaike information criterion (AIC).

$$AIC = 2K - 2\ ln|\mathcal{L}(\hat{\mu}|\mathbf{x})| \tag{16}$$

where 'K' is the number of parameters used, $\mathcal{L}(|\mathrm{x})$ expresses the maximum likelihood of the parameters ($\hat{\mu}$) given the data (x).

Information Criterion: AIC and BIC

- AIC was later modified by Gideon Schwarz to the so-called Bayesian information criterion (BIC).

$$BIC = ln(n)K - 2\, ln|\mathcal{L}(\hat{\mu}|\mathbf{x})| \tag{17}$$

where 'n' is the number of the data instances or sample size considered.

Current Section

1. ML: AN Introduction
2. Regression
   - Linear Regression
   - Non-Linear Regression
   - Gradient Descent Method
3. Model Selection
4. Appendix

Appendix 1: Solve Ax = b

- Least-squares fitting to linear models has critical advantages over other norms and error-metrics. Specifically, the optimization is inexpensive, since the error can be computed analytically.

- For any system of equations of the form:

$$[A]_{m\times n}\{\mathbf{x}\}_{n\times 1} = \{\mathbf{b}\}_{m\times 1}$$

- We can find a least square solution (also least norm solution) by using the pseudo inverse of $[A^{\dagger}]$; (However note that such a solution is restrictive and practical computations demand more flexibility on the choice of parameters)

$$\hat{\mathbf{x}} = A^{\dagger} b$$

Appendix 1: Solve Ax = b: Over-determined Systems

- For over determined systems, we have m $>$ n.
- The linear system of equations has the following form:

$$[A]_{m\times n}\{\mathbf{x}\}_{n\times 1} = \{\mathbf{b}\}_{m\times 1}$$

- Let us assume the system has full column rank, then, rank of [A] = r = n.
- In that case, $A^T A$ is invertible and hence we can directly find the least square equation as given below:

$$\hat{x} = (A^T A)^{-1} A^T b \tag{18}$$

The Eqn.(18) is valid only when A^TA is invertible.

Appendix 1: Solve Ax = b: Least Square Solution

The Least Square error solution when $A^T A$ is invertible

For obtaining minimum least square error, the optimization is given by;

$$\hat{x} = arg\min_{\mathbf{x}} (A\mathbf{x} - \mathbf{b})^2$$

At the local minimum, the gradient of the objective function is zero. Hence,

$$\frac{\partial}{\partial x} (A\mathbf{x} - \mathbf{b})^2 = 0$$

$$\frac{\partial}{\partial x} (A\mathbf{x} - \mathbf{b})^T (A\mathbf{x} - \mathbf{b}) = \frac{\partial}{\partial x} \left(\mathbf{x}^T A^T A\mathbf{x} - \mathbf{x}^T A^T \mathbf{b} - \mathbf{b}^T A\mathbf{x} + \mathbf{b}^T \mathbf{b} \right) = 0$$

$$2A^T A\mathbf{x} - A^T \mathbf{b} - \left(\mathbf{b}^T A \right)^T = 0$$

$$\mathbf{x} = (A^T A)^{-1} Ab \text{ (if } A^T A \text{ is invertible)}$$

Appendix 1: Solve Ax = b

- **Pseudo-inverse:**
  - For every matrix A = UΣV^T (using SVD), there exists a pseudo inverse defined as $A^\dagger = V\Sigma^\dagger U^T$.
  - Hence an approximate solution of the form $\hat{\mathbf{x}} = A^\dagger b$, is possible, in all scenarios.
  - However there are a few shortcomings for this approach.
- Suppose A is square and invertible:
  - **Elimination methods**.
- Suppose m $>$ n, with r = n; (over-determined system with mostly no solutions)
  - **Solve the normal equations**: $A^T Ax = A^T b$.
  - Solution to normal equation, gives the leas-squares solution to x (denoted here as $\hat{x}$).

Refer the book titled 'Linear Algebra and Learning from Data' by Gilbert Strang for more details

$\mathcal{END}$