What Is Machine Learning?

A gentle introduction

Shounak Shirodkar

MINRO, IIIT-Bangalore

August 23, 2022

Preface

Note

These slides are specifically designed keeping in mind the needs of our project, and a conscious effort was made to avoid details where necessary, particularly in covering the whole arena of classical machine learning methods that are usually taught in a standard course. Wherever possible, such topics have been discussed in brief.
The ultimate focus has been on steering the discussion towards neural nets and deep learning.

Introduction

What is Machine Learning?

1. **ML** is a field of inquiry devoted to understanding and building methods that ‘learn’, ie. methods that leverage data to improve performance on some set of tasks.
2. It is seen as a part of artificial intelligence.
3. ML algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.

Contrast with Programming

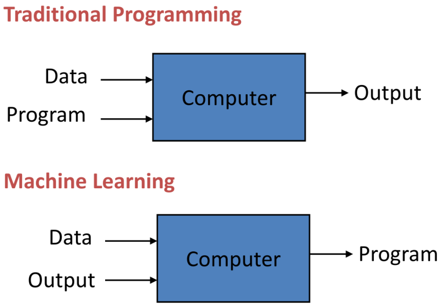


Figure: A fitting depiction of the contrast between the goals and process for programming and machine learning. Credits: Eric Eaton's slides

Basic Notations

- The equality sign $=$ is used in equations and for assignment. Sometimes $:=$ and $\leftarrow$ are also used for assignment.
- Σ is the **summation** operator (eg: $\Sigma_{i=1}^{4} i = 10$). Π is the **product** operator (eg: $\Pi_{i=1}^{4} i = 4! = 24$).
- Similarly, for sets, $\cup$ denotes **union** ($\{1,2,3\} \cup \{2,7\} \equiv \{1,2,3,7\}$), and $\cap$ denotes **intersection** ($\{1,2,3\} \cap \{2,7\} \equiv \{2\}$).
- Set **subtraction** is denoted by $\setminus$ or $-$ ($\{1,2,3\} - \{2,7\} \equiv \{1,3\}$).
- Note that curly brackets $\{\}$ usually denote unordered collections, while $[]$ denote sequences or ordered collections.
- $A \subset B$ denotes that A is a subset of B, similarly $A \supset B$ denotes A is a superset of B.
- For any set S, $|S|$ denotes its **cardinality**, or count of elements.

Vectors

- A **vector** basically refers to an ordered collection or sequence of quantities, and is used in various contexts (such as displacement and velocity in physics to tuples in linear algebra).
- A **vector space** is a *set* which is *closed* under addition and scaling (multiplication by scalar terms) ie. if $u, v \in S \rightarrow (\lambda_1 u + \lambda_2 v \in S)$, where λ_i are scalars.
- The **dimension** of a vector space is intuitively the number of independent directions in the space. In other words, it is the size of its **basis**.
- A subset of vectors $V \subseteq S$ is called a **basis** if every element of S may be written uniquely as a finite linear combination of elements of V. Every vector space has a basis, and every basis for a given vector space has the same cardinality.

Vector Operations

- For scalar multiplication of form $\lambda\vec{p}$, the resultant $\vec{q} = (\lambda x_1, \lambda y_1, \lambda z_1, \dots)$, provided $\vec{p} = (x_1, y_1, z_1, \dots)$.
- Two important binary operations on vectors are : the **dot product** and the **cross product**. However, cross product is not as widely used in machine learning, and is therefore omitted here.
- $\vec{p} = (x_1, y_1, z_1), \vec{q} = (x_2, y_2, z_2)$, then the dot product $\vec{p} \cdot \vec{q} = x_1x_2 + y_1y_2 + z_1z_2$, which is notably a scalar.
- For both addition and product, the vectors have to be **compatible** ie. of same dimension.
- The transpose of a vector $V \in \mathbb{R}^{1\times n}$ is $V^T \in \mathbb{R}^{n\times 1}$.
- For example,

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Matrices

- **Matrices** are two-dimensional ordered rectangular arrays of elements. A vector is a special type of matrix with 1 row or 1 column.
- An example of matrix dot product:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 1^2 + 2^2 + 3^2 & 1.4 + 2.5 + 3.6 \\ 4.1 + 5.2 + 6.3 & 4^2 + 5^2 + 6^2 \end{bmatrix}$$

- The transpose of a matrix $M \in \mathbb{R}^{m \times n}$ is $V^T \in \mathbb{R}^{n \times m}$.
- For example,

$$\begin{bmatrix} 1 & 2 & 3 \\ 14 & 15 & 16 \end{bmatrix}^T = \begin{bmatrix} 1 & 14 \\ 2 & 15 \\ 3 & 16 \end{bmatrix}$$

Matrices

- Another important unary operation for matrices is the **inverse**.
- The primary property of inverses (of square matrices) is that $A^{-1}A = AA^{-1} = I$. For non-square matrices, a left and a right inverse exist.
- A square matrix is invertible iff it is not singular (or degenerate) ie. its determinant is 0.
- A matrix has several left inverses if it has more rows than columns, and none if it has more columns than rows. The reverse applies for right inverses.
- An example usage is in solving a linear system of equations $Ax = b$. For the system to have a unique solution, we need at least as many equations as the number of variables. The solution vector x can be obtained simply by the operation $x = A^{-1}b$ (by left-multiplying both sides with the left-inverse of A, as $A^{-1}Ax = A^{-1}b$).

Matrix Operations

- For scalar multiplication of form λA, the result matrix $c_{ij} = \lambda a_{ij}$, where a_{ij} is a general element of A and λ is a scalar.
- The criteria for **compatibility** for matrix addition and multiplication (ie. dot product) differ.
- For matrix addition, both matrices need to have the same dimensions ie. of form $\mathbb{R}^{m\times n}$ and $\mathbb{R}^{m\times n}$. The result matrix $c_{ij} = a_{ij} + b_{ij}$.
- For dot product $A \cdot B$ to be valid (note that the order is important here), their dimensions should be of form $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times o}$.
- An example of matrix dot product:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 1^2 + 2^2 + 3^2 & 1.4 + 2.5 + 3.6 \\ 4.1 + 5.2 + 6.3 & 4^2 + 5^2 + 6^2 \end{bmatrix}$$

- For matrix calculus, please see this paper.

Tensor

- A **tensor** is an algebraic object that describes a *multilinear* relationship between sets of algebraic objects related to a vector space. The precise definition does not concern us.
- Due to its construction, a tensor can be viewed as a *wrapper class* over scalars and vectors.
- A **scalar** is a tensor of rank 0. A vector is a tensor of rank 1. A matrix $m \times n$ has rank $\leq min(m, n)$. Note that the *rank* of a matrix is the dimension of the vector space generated by its columns as basis.
- The machine learning library Torch implements the concept of Tensor. Its Pythonic version, PyTorch is more widely used.

Functions

- A function is a relation that associates each element $x \in X$, where X is the **domain**, to a single element y of another set Y, the **codomain** of the function. The **range** is the subset of the codomain which denotes the values that are actually taken in the mapping $X \to Y$.
- An **argument** of a function refers to a value provided to obtain the function's result. The **argmax** function is used to find an optimal argument to obtain the maximum possible output.
- Most real-world functions usually have **extremae**, points where the function takes extreme, either high (**maxima**) or low (**minima**) values, when compared to the **neighborhood** they lie in.
- A **local** extrema is a point of exteme value in a finite local interval, while **global** extrema is a point of extreme value globally.

Functions

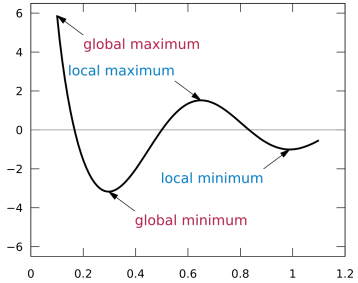


Figure: A plot of a finite curve/function explaining the notion of global and local

Derivatives and Gradients

- A **derivative** f' of function f describes the rate at which f grows. Derivatives are obtained through **differentiation**.
- Often when the function is complex, **chain rule** is used to simplify the process. For example, $F(x) = f(g(x))$, then $F'(x) = f'(g(x))g'(x)$.
- A **gradient** is a generalization of derivative (restricted to the scalar world) to vector spaces. So a gradient is a vector of **partial derivatives**.
- A **partial derivative** refers to a derivative with respect to one variable, while treating all other variables as constant.
- For example, the gradient ∇F of $F(x, y, z) = xy + 4z$, is

$$\frac{\partial(xy+4z)}{\partial(x,y,z)} = \left[\frac{\partial(xy+4z)}{\partial x} \; \frac{\partial(xy+4z)}{\partial y} \; \frac{\partial(xy+4z)}{\partial z}\right] = [y \;\; x \;\; 4]$$

Probability

- The **probability** of an event is the numerical possibility of it happening out of all possible scenarios.
- Since probabilities denote a fractional likelihood, the sum of all eventual probabilities for a specific scenario is 1. Also, complementary probabilities sum to 1.
- There are two prominent and competing schools of thought in probability: **frequentist** and **Bayesian** probability.
- *Frequentists* subscribe to the notion that the probability of a random event denotes the relative frequency "in the long run" ie. when the experiment is repeated indefinitely.
- *Bayesian subjectivists* subscribe to the notion that probability is essentially a **degree of belief**, factoring in expert knowledge and experimental data.

Bayesian Probability

- The conditional probability $Pr(X = x|Y = y)$ is the probability of an RV X having a value x given that another RV Y has a values y.
- Bayes' theorem states that

$$Pr(Y|X) = \frac{Pr(X|Y)Pr(Y)}{Pr(X)}$$

Here, typically X is the **evidence** (or observed data) and Y is the **hypothesis**.
$P(Y|X)$ is called **posterior**, $P(X|Y)$ is called **likelihood**, $P(Y)$ is called **prior** and $P(X)$ is called **marginal** (calculated as $\sum P(X|Y_i)P(Y_i)$).

Random variables

- **Random variables** (in short, RV), usually denoted by capital italics such as X, are the numerical outcomes of random phenomena. They can be **discrete** or **continuous**.
- The probability distribution of a **discrete** RV is called **PMF** (probability mass function). The sum of these discrete probabilities equals 1.
- That of a **continuous** RV is called **PDF** (probability density function). The area under the PDF curve equals 1.
- An RV has three primary **statistics**: mean (μ), variance and standard deviation (σ).

Statistics for RVs

- The **expectation** μ of an RV X, $\mathbb{E}(X)$ is

$$\mathbb{E}(X) := \sum_{x \in X} (x \cdot Pr[X = x])$$

  Expectation is alternatively called mean or average. Note that the summation is agnostic of continuity. This becomes a summation or an integral depending on whether or not the distribution is discrete.

- The **standard deviation** σ is

$$\sigma := \sqrt{\mathbb{E}(X - \mu)^2}$$

- The **variance** is merely σ^2.

Correlation

- In statistics, **correlation** or **dependence** is the statistical relationship between two RVs X and Y under consideration.
- In the broader sense, correlation can mean any kind of association, but is usually associated with a *linear* relationship.
- Identifying correlation between variables is useful in understanding data; for example, the correlation between cloudy weather and rainfall.
- Note that correlation does not imply **causation**.
- As an example, a basic measure of correlation, Pearson's correlation coefficient $\rho_{X,Y}$ for two RVs X, Y is

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

Unbiased Estimators

- Computers can only handle discrete data. Thus, even though we can not produce continuous distributions during computation, we can **sample** from them.
- Usually the function of interest is not known, and can only be sampled from. In such a case, we do not have access to the true statistics of the complete data, and have to use their **unbiased estimators** instead.
- $\hat{\theta}(S_X)$ is said to be an unbiased estimator of some statistic θ calculated using a sample S_X drawn from an unknown distribution if $\mathbb{E}(\hat{\theta}(S_X)) = \theta$.
- For example, assume a sample $S_X = \{x_i\}_{i=1}^{N}$ from some distribution f_X. The unbiased estimator of the unknown $\mathbb{E}(X) = \frac{1}{N}(\sum_{i=1}^{N} x_i)$.

Types

Types of Machine Learning

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the “signal” or “feedback” available to the learning system.

- **Supervised Learning**: (example inputs & outputs ie. labeled data) $\rightarrow$ learn a generalised mapping to “predict” outputs for unseen inputs.
- **Unsupervised Learning**: (Unlabeled data) $\rightarrow$ find structure of input.
- **Reinforcement Learning**: (Learner interacts with dynamic environment with feedback mechanism of rewards/penalties) $\rightarrow$ maximize reward.

Notion of **label**: a label is an informative feature, added to or already present in raw data, that provides context.

Types of Machine Learning

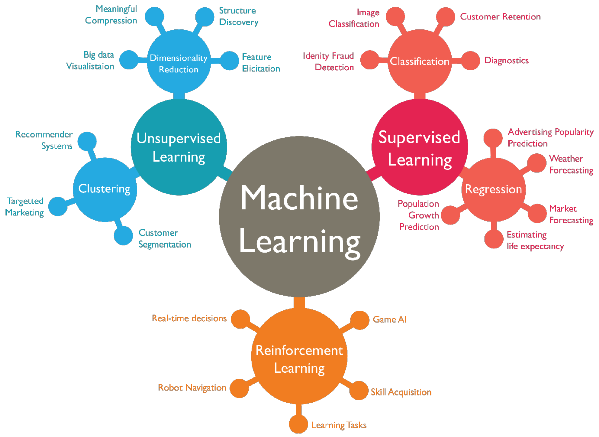


Figure: Different types of machine learning and the associated tasks.

Point of Focus

- Our task, time-series forecasting, is by nature a supervised regression task.
- Note that a technique misleadingly named **logistic regression** is used in classification, but it is not relevant to us.
- Due to our focus being on regression, other topics will be omitted from the discourse.
- Our ultimate aim is to steer the discussion towards cutting-edge deep learning.

Regression

Introduction

- Regression analysis predates computers. In fact, machine learning is sometimes called computational statistics, because the advent of computation allowed important advances to be made.
- The goal of regression is to estimate the relationship between a **dependent variable** (**outcome** or **response variable** or **label**) and one or more **independent variables** (**predictors** or **covariates** or **explanatory variables** or **features**).
- In a sense, regression may be viewed as a way of *curve-fitting*.
- We will start with a demonstration of **classical linear regression**.

Assumptions

Since we wish to demonstrate only a classical version of linear regression, we will assume the following:

- The observational data (or the training data) sample is representative of the data at large, statistically speaking.
- The independent variables are measured with minimal error.
- Deviations from the model, known as **residuals**, when conditioned on covariates, have an expected value of 0 ie. $\mathbb{E}(\epsilon_i | X_i) = 0$.
- The residuals are **homoscedastic** ie. they have the same variance across observations.
- The residuals are mutually uncorrelated.

Most of these do not apply to real-world data. There are techniques to handle heteroscedastic residuals which will not be discussed here.

Problem Statement

- Suppose a set of labeled data $\{(\vec{x_i}, y_i)\}_{i=1}^{N}$, such that $\vec{x_i}$ is a D-dimensional vector, and y_i is a real-valued target. Every feature, part of $\vec{x_i}$, is also real-valued.
- Our goal is build a model $f_{\vec{w},b}(x)$ such that $f_{w,b}(\vec{x}) = \vec{w}\vec{x} + b$. Here, $\vec{w}$ represents a vector of **coefficients** while b is called **bias**. Collectively, these are called **parameters**.
- Note that **parameters** are variables learnt by the model during the training process, while **hyperparameters** are set manually to control the training process.
- The goal is thus to find optimal $(\vec{w_*}, b_*)$ such that the model is as *good* as possible.

Performance Metric

- The curve here is called the **hyperplane**, because its dimension is always 1 less than that of its *ambient space*. The hyperplane is to be such that it is as close to all the training data points.
- Our procedure uses the **mean-squared error** or MSE: $\frac{1}{N}(f_{\vec{w},b}(x_i) - y_i)^2$, as a **loss function** (or **objective** or **cost function** or **empirical risk**) to find optimal $(\vec{w}_*, b_*)$.
- Typically, the most popular choice for optimization procedure is **stochastic gradient descent**.
- SGD is based on the simple principle of heading down the gradient of the loss function and carrying updates according to a set **learning rate**.

Closed-form Solution

- Note that for classical linear regression, a closed-form solution is available by simply minimizing MSE. This might not be, and is usually not, the case elsewhere.
- Thus, the optimal weight can be denoted as

$$\begin{aligned} W &= argmin_W MSE \\ &= argmin_W \frac{1}{2}(XW - Y)^T(XW - Y) \end{aligned}$$

- The gradient of the cost function which must be zero at the minima can be derived as

$$\nabla_W \frac{1}{2}(XW - Y)^T(XW - Y) = 0$$

Upon simplifying this, we obtain $W = (X^TX)^{-1}X^TY$

Polynomial Regression

- What if the relationship between the dependent and the independent variables is not linear? A linear hyperplane will never be able to generalise well to such a distribution.
- In such cases, **polynomial regression** might be deployed instead. Here, the relationship is modeled as an *n*th degree polynomial. For example

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 & \dots \beta_m \end{bmatrix} \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} + \vec{\epsilon}$$

Typical Problems

- **Suboptimal parameters**: when the small update size leads to the return value getting stuck in a local minima instead of a global minima.
- **Overfitting**: when the hyperplane fits the training data too well, but does not generalise to unseen data. **Regularization** is used to curb overfitting by making the curve simpler.
- **Underfitting**: when the hyperplane fits the data too loosely.

The latter two can be summed up in the "**bias-variance tradeoff**" (bias refers to underfitting, and variance refers to overfitting). Decreasing one tends to increase the other.

Neural Nets

Introduction

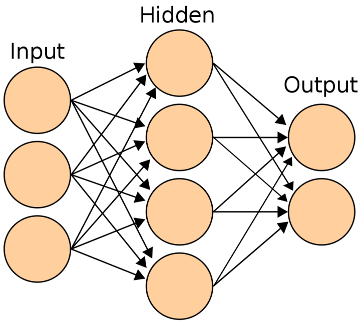


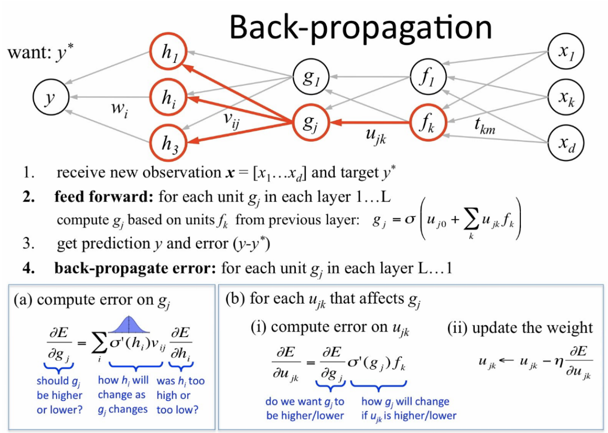
Figure: A representative diagram of an NN

- Neural nets or neural networks or NNs are computing systems inspired by biological neural networks.
- An NN is a collection of connected nodes analogous to neurons, and the connections themselves analogous to neural synapses.

Structure

- There is an input and an output layer. The input layer has the dimensions of the processed training data. The output layer has the dimensions of the expected output.
- There may be multiple hidden layers in between the terminal layers. NNs with a large number of hidden layers are usually called **deep** NNs.
- Every neuron has a set of weight parameters that need to be learnt.
- The output of every neuron is passed through an **activation function** before being sent ahead into the network. All activation functions need to be differentiable for backpropagation to work.
- Simple feedforward NNs usually do not have any cycles. Recurrent NNs have a cyclic structure in order to facilitate **memory**.

Backpropagation



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Figure: Backpropagation: the search/optimization procedure used in neural nets

Common Problems

- **Instability**: The idea of "stability" is how much does a model change due to small perturbations in training data. Neural nets have a tendency to overfit and tend to be unstable.
- **Vanishing gradient problem**: During backpropagation updates, the gradient might be very small, leading to the weight getting stuck without updates. If this happens at too many nodes, the model might stop training.
- **Exploding gradient problem**: When the gradient is nearly vertical, there are large weight updates, leading to drastic changes in the model.

Learning

Components of a Machine Learning Algorithm

- **Representation**: a construction of the target function that has to learn parameters for optimal performance.
- **Optimization**: a method used as a subroutine within the representation to search for the best parameters.
- **Evaluation**: a procedure to test the performance of our model (possibly against pre-existing **benchmarks** or a chosen **baseline**).

Define a Learning Task

- Choose the training experience.
- Choose what exactly is to be learnt ie. a **target function**.
- Choose a suitable representation for the target function.
- Choose a suitable learning algorithm given the type of training data and the task to infer the target function from training experience.

Define Improvement Criterion

"Improve on task T, with respect to performance metric P, based on experience E."

For example,
T = Categorize email messages as spam or legitimate;
P = Percentage of email messages correctly classified;
E = Database of emails, some with human-given labels

Various Function Representations

- Numerical functions
  - Linear Regression
  - Neural networks
  - Support Vector Machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic Graphical Models
  - Naïve Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMMs)
  - Probabilistic Context Free Grammars (PCFGs)
  - Markov networks

Various Search/Optimization Algorithms

- Gradient descent
  - Perceptron
  - Backpropagation
- Dynamic Programming
  - HMM learning
  - PCFG learning
- Divide and Conquer
  - Decision tree induction
  - Rule learning
- Evolutionary Computation
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)
  - Neuro-evolution

Various Performance Metrics

- Accuracy
- Precision and Recall
- Squared error
- Likelihood
- Posterior probability
- Cost/Utility
- Margin
- Entropy
- KL Divergence

Typical Workflow

The typical machine learning workflow is an iteration loop over the following:

1. Prior domain knowledge and specifying goals of task.
2. Data integration, selection, cleaning, pre-processing etc.
3. Learn models.
4. Interpret results.
5. Consolidate and deploy discovered knowledge.

Recommended Reading

Recommended Reading

- An up-to-date ML glossary covering the spectrum from basic definitions to advanced and cutting-edge terms.
- Andriy Burkov's **The Hundred Page Machine Learning Book**
- Christopher Bishop's **Pattern Recognition and Machine Learning**

ज्ञानमुत्तमम्