Machine Learning

LIN Qingfeng
March 26, 2021

1 Introduction

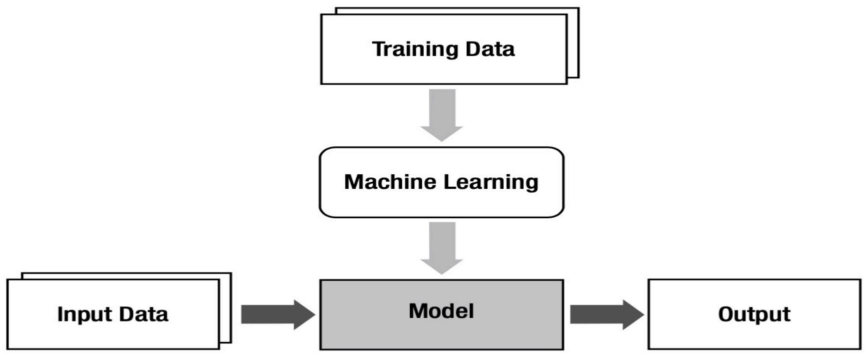

Figure 1: The concept of Machine Learning

- Definition:
  Machine Learning is the technique used to find (or learn) a model from the data. It is suitable for problems that involve intelligence, such as image recognition and speech recognition, where physical laws or mathematical equations fail to produce a model
- Challenges with Machine Learning:
  - Generalization: The distinctness of the training data and input data is the structural challenge that Machine Learning faces. And the process used to make the model performance consistent regardless of the training data or the input data is called generalization.

    The success of Machine Learning heavily relies on how well the generalization process is implemented. In order to prevent performance degradation due to the differences between the training data and actual input data, we need a sufficient amount of unbiased training data.
  - Overfitting: If you believe that every element of the training data is correct and fits the model precisely, you will get a model with lower generalizability. This is called overfitting.

    Regularization and validation are the typical approaches used to solve the overfitting problem. Regularization is a numerical method that yields the simplest model as possible. In contrast, validation tries to detect signs of overfitting during training and takes action to prevent it. A variation of validation is cross-validation.
- Types of Machine Learning:
  - Supervised Learning: In supervised learning, each training dataset should consist of input and correct output pairs. The two most common types of application of supervised learning are **classification** and **regression**.
  - Unsupervised Learning: In unsupervised learning, the training data of the unsupervised learning contains only inputs without correct outputs.The two most common types of application of supervised learning are **clustering** and **density estimation**.
  - Reinforcement Learning: The technique of reinforcement learning is concerned with the problem of finding suitable actions to take in a given situation in order to maximize a reward. Here the learning algorithm is not given examples of optimal outputs, in contrast to supervised learning, but must instead discover them by a process of trial and error.

2 The Framework of Machine Learning

- **Frequentist:** The parameters in **Frequentist** are assumed to be deterministic but unknown.

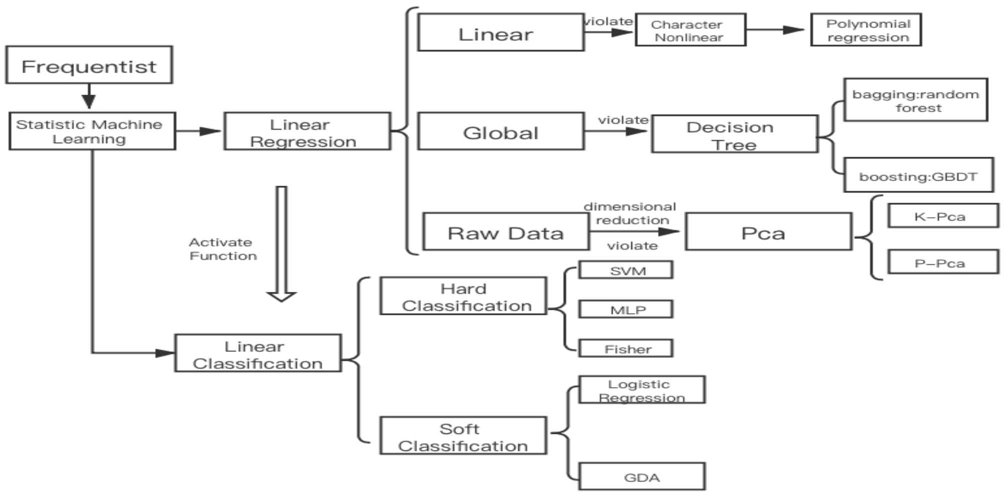

Figure 2: The roadmap of frequentist

- **Bayesian:** The parameters in **Bayesian** are assumed to be a random variable θ.

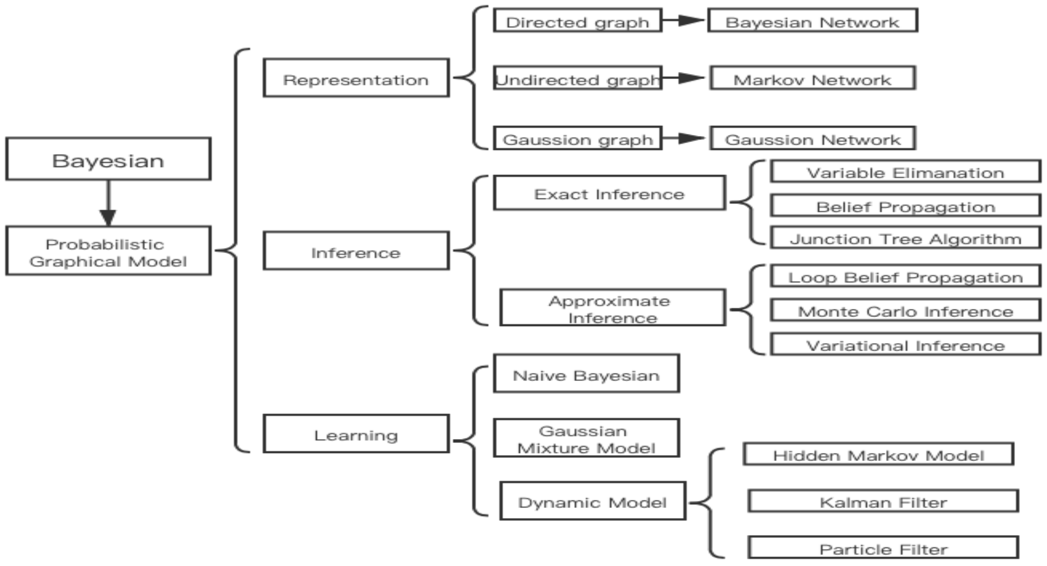

Figure 3: The roadmap of Bayesian

2.1 Frequentist versus Bayesian

Table 1: Frequentist vs. Bayesian

Frequentist	Bayesian	The Algorithm in Frequentist	The Algorithm in Bayesian
Linear regression	Bayesian linear regression	closed-form solution	closed-form solution
Logistic regression	Bayesian Logistic regression	Newton Iteration	Laplace approximation
Neural network	Bayesian Neural network	gradient decent	Laplace approximation
SVM	RVM	Quadratic Optimization	Laplace approximation
Gaussian mixture model	Bayesian Gaussian mixture model	EM	Variation inference
Probabilistic PCA	Bayesian probabilistic PCA	closed-form solution/EM	Laplace approximation

2.2 Optimization/Approximation

- Linear/Quadratic/Convex optimization; Lagrange multiplier; Gradient decent; Newton iteration;
- Laplace approximation; Expectation Maximation (EM); Variational inference;
- MCMC/Gibbs sampling.

2.3 Objective function/ Error function/Estimato

- Likelihood: The problem in MLE
- Marginal likelihood: The estimation in emprical Bayes/evidence approximation
- Sum-of-square error: The problem in Regression
- Posterior:The problem in MAP
- Negative log likelihood/cross-entropy:Logistic regression
- Exponential error: Adaboost
- Hinge error: SVM

3 Machine Learning Model

3.1 Linear Regression Model

- **Model:**

$$f(x) = w^T x^* \tag{1}$$

Where, $x^* = \left[x^T, 1\right]^T, x \in R^n, w \in R^{n+1}, w$ is learning parameter.

The loss function is,

$$L(w) = \sum_{i=1}^{m} (y_i - f(x_i))^2 = \sum_{i=1}^{m} \left(y_i - w^T x_i^*\right)^2 = (Y - X^* w)^T (Y - X^* w) \tag{2}$$

Thus, the optimal parameter w^* satisfies,

$$w^* = \arg\min_{w} L(w) \tag{3}$$

- **Algorithm:**
  - closed-form solution
  - gradient descent
- **Comments:**
  - Linear Regression is limited to linear relationships
    Solution: Polynomial Regression

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{4}$$

  - Overfitting problem
    Solutions: Regularization
    Lasso:

$$L(w) = \sum_{i=1}^{m} \left(y_i - f\left(x_i\right)\right)^2 + \lambda ||w||_1 \tag{5}$$

    Ridge:

$$L(w) = \sum_{i=1}^{m} \left(y_i - f\left(x_i\right)\right)^2 + \alpha ||w||_2 \tag{6}$$

  - The problem of convergence
    Solution: Using Cross-entropy as loss function

$$f(x) = \delta\left(w^T x^*\right) = \frac{1}{1 + e^{-(w^T x^*)}} \tag{7}$$

$$L(w) = -\sum_{j=1}^{n} y_j \log\left(f\left(x_j\right)\right) + \left(1 - y_j\right) \log\left(1 - f\left(x_j\right)\right) \tag{8}$$

3.2 Principal Component Analysis(PCA)

- **Motivation**: PCA is used abundantly because it is a simple, non-parametric method of extracting relevant information from confusing data sets. With minimal additional effort PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified structure that often underlie it.
- **A MORE GENERAL SOLUTION: SVD**

  Let $\mathbf{X}$ be an arbitrary $m \times n$ matrix and $\mathbf{X}^T\mathbf{X}$ be a rank r, square, symmetric $n \times n$ matrix.
- $\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_r\}$ is the set of orthonormal $n \times 1$ eigenvectors with associated eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ for the symmetric matrix $\mathbf{X}^T\mathbf{X}$

$$\left(\mathbf{X}^T\mathbf{X}\right) \hat{\mathbf{v}}_i = \lambda_i \hat{\mathbf{v}}_i \tag{9}$$

- $\sigma_i \equiv \sqrt{\lambda_i}$ are positive real and termed the singular values.
- $\{\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_r\}$ is the set of orthonormal $m \times 1$ vectors defined by $\hat{\mathbf{u}}_\mathbf{i} \equiv \frac{\mathbf{1}}{\sigma_i}\mathbf{X}\hat{\mathbf{v}}_\mathbf{i}$

- We summarize the equations above, we have,

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{10}$$

- Compared this method with eigenvector method, we define a new matrix Y,

$$\mathbf{Y} = \frac{1}{\sqrt{n-1}}\mathbf{X}^T \tag{11}$$

  where each column of $\mathbf{Y}$ has zero mean. The definition of $\mathbf{Y}$ becomes clear by analyzing $\mathbf{Y}^T\mathbf{Y}$.

$$\begin{aligned}
\mathbf{Y}^T\mathbf{Y} &= \left(\frac{1}{\sqrt{n-1}}\mathbf{X}^T\right)^T \left(\frac{1}{\sqrt{n-1}}\mathbf{X}^T\right) \\
&= \frac{1}{n-1}\mathbf{X}^{TT}\mathbf{X}^T \\
&= \frac{1}{n-1}\mathbf{X}\mathbf{X}^T \\
\mathbf{Y}^T\mathbf{Y} &= \mathbf{C}_{\mathbf{X}}
\end{aligned} \tag{12}$$

- By construction $\mathbf{Y}^T\mathbf{Y}$ equals the covariance matrix of $\mathbf{X}$. We know that the principal components of $\mathbf{X}$ are the eigenvectors of $\mathbf{C}_{\mathbf{X}}$. If we calculate the SVD of $\mathbf{Y}$, the columns of matrix $\mathbf{V}$ contain the eigenvectors of $\mathbf{Y}^T\mathbf{Y} = \mathbf{C}_{\mathbf{X}}$. Therefore, the columns of $\mathbf{V}$ are the principal components of $\mathbf{X}$.

3.3 Support Vector Machine

- **Motivation:**
  Basic idea of support vector machines: just like one layer or multi-layer neural networks. Optimal hyperplane for linearly separable patterns.
  Extend to patterns that are not linearly separable by transformations of original data to map into new space-the Kernel function.
- **Model:**
- The optimization problem is,

$$\begin{gathered}
\min_{w,b} \tfrac{1}{2} w^T w \\
\text{s.t. } y_i \left(w^T x_i + b\right) \geq 1, i = 1, 2, \ldots, N
\end{gathered} \tag{13}$$

- **Algorithm:**
  - Using Lagrange dual function, we can easily solve this QP problem.
  - Using SMO.
- **Comments:**
  - Traditional SVM can not solve non-linear separable problem.
    Solution: Kernal SVM

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \phi\left(x_i\right)^T \phi\left(x_j\right) - \sum_{i=1}^{N} \alpha_i \tag{14}$$

3.4 Gaussian Mixture Model

- The problem is,

$$\begin{aligned}\Theta_{MLE} &= \underset{\Theta}{\arg\max}\mathcal{L}(\Theta \mid X)\\ &= \underset{\Theta}{\arg\max}\left(\sum_{i=1}^{n} \log \sum_{l=1}^{k} \alpha_l \mathcal{N}\left(X \mid \mu_l, \Sigma_l\right)\right)\end{aligned} \tag{15}$$

For each iteration of the E-M algorithm, we perform:

$$\Theta^{(g+1)} = \underset{\theta}{\arg\max}\left(\int_z \log(p(X, Z \mid \theta)) p\left(Z \mid X, \Theta^{(g)}\right)\right) dz \tag{16}$$

How to define $p(X, Z \mid \Theta)$,

$$p(X, Z \mid \Theta) = \prod_{i=1}^{n} p\left(x_i, z_i \mid \Theta\right) = \prod_{i=1}^{n} \underbrace{p\left(x_i \mid z_i, \Theta\right)}_{\mathcal{N}\left(\mu_{z_i}, \Sigma_{z_i}\right)} \underbrace{p\left(z_i \mid \Theta\right)}_{\alpha_{z_i}} = \prod_{i=1}^{n} \alpha_{z_i} \mathcal{N}\left(\mu_{z_i}, \Sigma_{z_i}\right) \tag{17}$$

How to define $p(Z \mid X, \Theta)$

$$p(Z \mid X, \Theta) = \prod_{i=1}^{n} p\left(z_i \mid x_i, \Theta\right) = \prod_{i=1}^{n} \frac{\alpha_{z_i} \mathcal{N}\left(\mu_{z_i}, \Sigma_{z_i}\right)}{\sum_{l=1}^{k} \alpha_l \mathcal{N}\left(\mu_l, \Sigma_l\right)} \tag{18}$$

Now we compute $Q\left(\Theta, \Theta^{(g)}\right)$,

$$\begin{aligned}Q\left(\Theta, \Theta^{(g)}\right) &= \int_{z_1} \cdots \int_{z_n} \left(\sum_{i=1}^{n} \ln p\left(z_i, x_i \mid \Theta\right) \prod_{i=1}^{n} p\left(z_i \mid x_i, \Theta^{(g)}\right)\right) dz_1, \ldots dz_n\\ &= \sum_{i=1}^{n}\left(\int_{z_i} \ln p\left(z_i, x_i \mid \Theta\right) p\left(z_i \mid x_i, \Theta^{(g)}\right) dz_i\right) \qquad z_i \in \{1, \ldots, k\}\\ &= \sum_{z_i=1}^{k} \sum_{i=1}^{n} \ln p\left(z_i, x_i \mid \Theta\right) p\left(z_i \mid x_i, \Theta^{(g)}\right)\\ &= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln\left[\alpha_l \mathcal{N}\left(x_i \mid \mu_l, \Sigma_l\right)\right] p\left(l \mid x_i, \Theta^{(g)}\right)\end{aligned} \tag{19}$$

$$\begin{aligned}Q\left(\Theta, \Theta^{(g)}\right) &= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln\left[\alpha_l \mathcal{N}\left(x_i \mid \mu_l, \Sigma_l\right)\right] p\left(l \mid x_i, \Theta^{(g)}\right)\\ &= \sum_{l=1}^{k} \sum_{i=1}^{n} \ln\left(\alpha_l\right) p\left(l|x_i, \Theta^{(g)}\right) + \sum_{l=1}^{k} \sum_{i=1}^{n} \ln\left[\mathcal{N}\left(x_i \mid \mu_l, \Sigma_l\right)\right] p\left(l|x_i, \Theta^{(g)}\right)\end{aligned} \tag{20}$$

The first term contains only α and the second term contains only μ, Σ. So we can maximize both terms independantly.
Maximizing α means that:

$$\frac{\partial \sum_{l=1}^{k} \sum_{i=1}^{n} \ln\left(\alpha_l\right) p\left(l \mid x_i, \Theta^{(g)}\right)}{\partial \alpha_1, \ldots, \partial \alpha_k} = [0 \ldots 0] \quad \text{subject to} \sum_{l=1}^{k} \alpha_l = 1 \tag{21}$$

We can solve μ, Σ in the same way above.

3.5 Naive Bayes

- **Model:**
$$p\left(X_1, X_2, \ldots, X_n \mid Y\right)=p\left(X_1 \mid Y\right) \cdot p\left(X_2 \mid Y\right) \cdots p\left(X_n \mid Y\right) \tag{22}$$
Thus, the joint distribution is,
$$p\left(X_1, X_2, \ldots, X_n, Y\right)=p(Y) \prod_{i=1}^n p\left(X_i \mid Y\right) \tag{23}$$
The classifier based on Naive Bayes is,
$$f(x)=\arg \max _{c_k} P\left(Y=c_k\right) \prod_{i=1}^n P\left(X_i=x_i \mid Y=c_k\right) \tag{24}$$

3.6 Hidden Markov Model

- **Model:**
Discrete Transition Probability:
$$p\left(q_t \mid q_1, \ldots, q_{t-1}, y_1, \ldots, y_{t-1}\right)=p\left(q_t \mid q_{t-1}\right) \tag{25}$$
Continous/Discrete Measurement probability:
$$p\left(y_t \mid q_1, \ldots, q_{t-1}, q_t, y_1, \ldots, y_{t-1}\right)=p\left(y_t \mid q_t\right) \tag{26}$$
- The HMM Parameter λ (discrete measurement case) contains:
$$\lambda=\{A, B, \pi\} \tag{27}$$
where π is the probability of the initial state.

Three major operations of HMM:
$$\begin{gathered}\text{Evaluate } p(Y \mid \lambda) \\ \lambda_{MLE}=\arg \max_\lambda p(Y \mid \lambda) \\ \arg \max_Q p(Y \mid Q, \lambda)\end{gathered} \tag{28}$$
- In Evaluate Problem, we usually compute,
$$\begin{aligned} p(Y \mid \lambda) & =\sum_Q[p(Y, Q \mid \lambda)]=\sum_{q_1=1}^k \ldots, \sum_{q_T=1}^k\left[p\left(y_1, \ldots, y_T, q_1, \ldots q_T \mid \lambda\right)\right] \\ & =\sum_{q_1=1}^k \ldots, \sum_{q_T=1}^k\left[p\left(y_1, \ldots, y_T, q_0, q_1, \ldots q_T \mid \lambda\right)\right] \\ & =\sum_{q_1=1}^k \ldots, \sum_{q_T=1}^k p\left(q_1\right) p\left(y_1 \mid q_1\right) p\left(q_2 \mid q_1\right) \ldots p\left(q_t \mid q_{t-1}\right) p\left(y_t \mid q_t\right)\end{aligned} \tag{29}$$
we usually use forward/backward algorithm.
- In parameter learning, we usually use EM algorithm
- In decoding problem, we usually use viterbi algorithm.

3.7 Kalman Filter

- The data model is,

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \\ y_k &= H_k x_k + v_k \\ E\left(w_k w_j^T\right) &= Q_k \delta_{k-j} \\ E\left(v_k v_j^T\right) &= R_k \delta_{k-j} \\ E\left(w_k v_j^T\right) &= 0 \end{aligned} \tag{30}$$

  The error between the true state and the estimated state is denoted as,

$$\tilde{x}_k = x_k - \hat{x}_k \tag{31}$$

- Suppose we want to find the estimator that minimizes (at each time step) a weighted two-norm of the expected value of the estimation error $\tilde{x}_k$:

$$\min E\left[\tilde{x}_k^T S_k \tilde{x}_k\right] \tag{32}$$

  where S_k is diagonal positive definite user-defined weighting matrix.

- If $\{w_k\}$ and $\{v_k\}$ are Gaussian, zero-mean, uncorrelated, and white, then the Kalman filter is the solution to the above problem.

3.8 Particle Filter

- **Model**:

$$\begin{aligned} x_{k+1} &= f_k\left(x_k, w_k\right) \\ y_k &= h_k\left(x_k, v_k\right) \end{aligned} \tag{33}$$

  The functions $f_k(\cdot)$ and $h_k(\cdot)$ are time-varying nonlinear system and measurement equations. The noise sequences $\{w_k\}$ and $\{v_k\}$ are assumed to be independent and white with known pdf's.The goal of a Bayesian estimator is to approximate the conditional pdf of x_k based on measurements $y_1, y_2, \cdots, y_k$. This conditional pdf is denoted as

$$p\left(x_k \mid Y_k\right) = \text{ pdf of } x_k \text{ conditioned on measurements } y_1, y_2, \cdots, y_k \tag{34}$$

  Our goal is to find a recursive way to compute the conditional pdf $p\left(x_k \mid Y_k\right)$

$$\begin{aligned} p\left(x_k | Y_k\right) &= \frac{p\left(Y_k | x_k\right)}{p\left(Y_k\right)} p\left(x_k\right) \\ &= \frac{p\left[\left(y_k, Y_{k-1}\right) | x_k\right]}{p\left(y_k, Y_{k-1}\right)} \underbrace{\frac{p\left(x_k | Y_{k-1}\right) p\left(Y_{k-1}\right)}{p\left(Y_{k-1} | x_k\right)}}_{p(x_k)} \\ &= \frac{p\left(x_k, y_k, Y_{k-1}\right)}{p\left(x_k\right) p\left(y_k, Y_{k-1}\right)} \frac{p\left(x_k, Y_{k-1}\right) p\left(Y_{k-1}\right)}{p\left(Y_{k-1}\right) p\left(Y_{k-1} | x_k\right)} \\ &= \frac{p\left(y_k | x_k\right) p\left(x_k | Y_{k-1}\right)}{p\left(y_k | Y_{k-1}\right)} \end{aligned} \tag{35}$$

3.9 Variation Inference

- **Motivation:**

$$q(Z) \rightarrow p(Z \mid X) \tag{36}$$

Where X is observation data, Z is parameter or latent variables.

- **Main idea:**

$$p(X)=\frac{p(X, Z)}{p(Z \mid X)} \tag{37}$$

$$\begin{aligned} \ln p(X) &= \ln p(X, Z)-\ln p(Z \mid X) \\ &= \ln \tfrac{p(X, Z)}{q(Z)}-\ln \tfrac{p(Z \mid X)}{q(Z)} \end{aligned} \tag{38}$$

Next, calculate the expectation on distribution q(Z) for both sides,

$$\begin{aligned} \ln p(X) &= \int q(Z) \ln \left\{\tfrac{p(X, Z)}{q(Z)}\right\} d Z-\int q(Z) \ln \left\{\tfrac{p(Z \mid X)}{q(Z)}\right\} d Z \\ &= \int q(Z) \ln \left\{\tfrac{p(X, Z)}{q(Z)}\right\} d Z+\int q(Z) \ln \left\{\tfrac{p(q(Z)}{Z \mid X)}\right\} d Z \\ &= \mathcal{L}(q)+K L(q \mid p) \end{aligned} \tag{39}$$

Thus, the objective function is,

$$q^*(Z)=\arg \max_{q(Z)} \int q(Z) \ln \left\{\frac{p(X, Z)}{q(Z)}\right\} d Z \tag{40}$$

For simplicity, $q(Z)$ satisfies,

$$q(Z)=\prod_{i=1}^M q_i\left(Z_i\right) \tag{41}$$

Thus, we have,

$$\ln q_j^*\left(Z_j\right)=E_{i \neq j}[\ln p(X, Z)]+\text { const } \tag{42}$$

3.10 Sampling Method

- **Motivation**:
  In many machine learning problems, we are actually interested in the posterior distribution $p(\theta \mid \text{Data }) \propto p(\text{data} \mid \theta) p(\theta)$

  Sampling is used to calculate mean, variance, and function expectations for a complex probability distribution with a known definition.

  Algorithm:

  - Rejection Sampling
  - Importance Sampling
  - Monte Carlo Markov Chain
  - Gibbs Sampling

4 Deep Learning

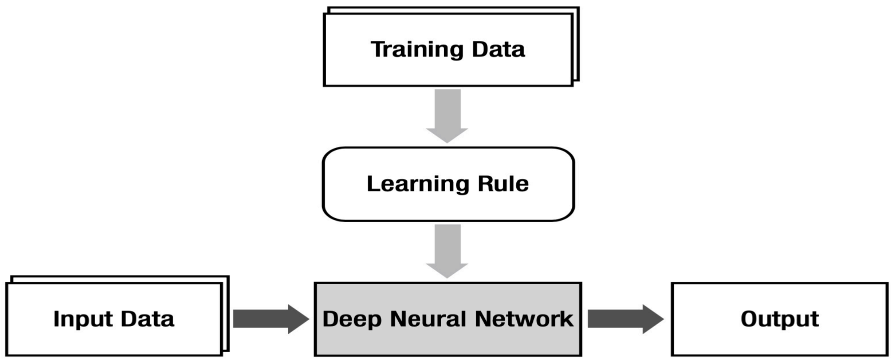

Figure 4: The concept of Deep Learning

- **Motivation:** Deep Learning is a Machine Learning technique that employs the deep neural network. There is a lot of misunderstanding about what depth means. People thought there was no need of deep neural networks. A shallow neural network with a single layer of hidden units is sufficient to represent any function with required degree of accuracy. This called the universal approximation property. But, it doesn't tell us how many units is required. With a deep neural network we can represent the same function as that of a shallow neural network but more cheaply ie., with less number of hidden units. The number of units needed can be exponentially larger for a shallow network compared to a network that is deep enough.
- **Main Idea:** Add the hidden layers.
- Back Propagation Algorithm : Solve the training problem of the multi-layer neural network. The significance of the BP algorithm was that it provided a systematic method to determine the error of the hidden nodes.

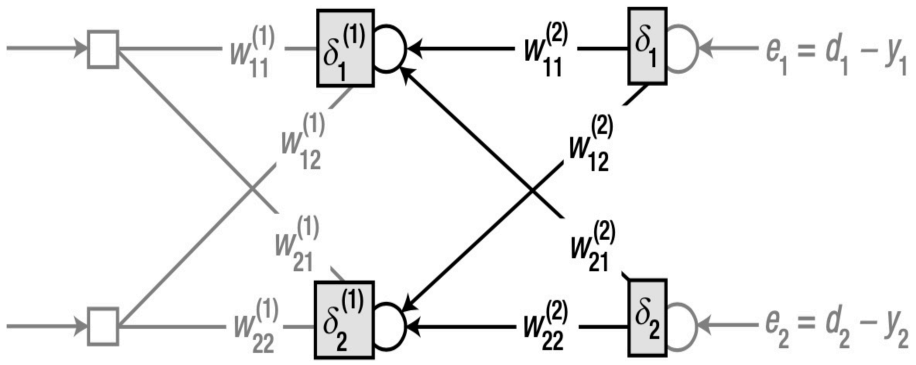

Figure 5: Train the neural network using the back-propagation algorithm

- **Application:**
  - DNN
  - CNN
  - RNN
  - GAN
  - GNN