YWTN Artificial Intelligence Equations

1. **Linear Regression:** $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$
    - **This is the standard linear regression equation, correctly representing the relationship between the dependent variable y and the independent variables $x_1, x_2, \ldots, x_n$.**
2. **Logistic Regression:** $p(y = 1 \mid x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$
    - **This is the correct logistic regression equation for binary classification, where $p(y = 1 \mid x)$ is the probability of the event occurring.**
3. **Support Vector Machines (SVM):** $f(x) = w^T x + b$
    - **This is the correct decision function for a linear SVM, where w is the weight vector, x is the input feature vector, and b is the bias term.**
4. **Neural Networks:** $a_i = f\left(\sum_{j=1}^{n} w_{ij}\, x_j + b_i\right)$
    - **This is the correct activation function for a neuron in a neural network, where a_i is the activation, w_{ij} are the weights, x_j are the inputs, and b_i is the bias.**
5. **Convolutional Neural Networks (CNNs):** $Y = f(W * X + b)$
    - **This is the correct equation for a convolutional layer in a CNN, where Y is the output feature map, W is the filter, X is the input image, and b is the bias.**
6. **Recurrent Neural Networks (RNNs):** $h_t = f(W_h h_{t-1} + W_x x_t + b_h)$
    - **This is the correct equation for the hidden state in an RNN, where h_t is the hidden state at time t, W_h and W_x are weight matrices, and b_h is the bias.**
7. **Reinforcement Learning (Q-learning):** $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_a Q(s', a) - Q(s, a)\right]$
    - **This is the correct Q-learning update rule, where $Q(s, a)$ is the expected reward for taking action a in state s, α is the learning rate, r is the immediate reward, and γ is the discount factor.**
8. **Bayes' Theorem:** $P(A \mid B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
    - **This is the correct form of Bayes' Theorem, used for updating probabilities based on new evidence.**

9. **Michaelis-Menten Equation:** $v = \frac{V_{\max}[S]}{K_m + [S]}$
    - **This is the correct Michaelis-Menten equation, describing the rate of enzymatic reactions.**
10. **Hardy-Weinberg Principle:** $p^2 + 2pq + q^2 = 1$
    - **This is the correct equation for the Hardy-Weinberg equilibrium in population genetics.**
11. **Poiseuille Equation:**
    - **The equation is:** $Q = \frac{\pi r^4 \Delta P}{8\eta l}$ **where** Q **is the volumetric flow rate,** r **is the radius of the tube,** ΔP **is the pressure difference,** η **is the dynamic viscosity, and** l **is the length of the tube.**
12. **Price Equation:**
    - **The equation is:** $\Delta z = \text{Cov}(w_i, z_i) + \mathbb{E}(w_i \Delta z_i)$ **where** Δz **is the change in the average trait value,** w_i **is the relative fitness, and** z_i **is the trait value.**
13. **Gradient Descent Update Rule:**
    - **The equation is:** $\theta = \theta - \alpha \cdot \nabla J(\theta)$ **where** θ **are the parameters,** α **is the learning rate, and** $J(\theta)$ **is the cost function.**
14. **Stochastic Gradient Descent (SGD):**
    - **The equation is:** $\theta = \theta - \alpha \cdot \nabla J\left(\theta; x^{(i)}, y^{(i)}\right)$ **where** $x^{(i)}$ **and** $y^{(i)}$ **are a single training example.**
15. **Naive Bayes Classifier:**
    - **The equation is:** $P(y \mid x_1, \ldots, x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$ **where** $P(y \mid x_1, \ldots, x_n)$ **is the posterior probability.**
16. **Gini Impurity:**
    - **The equation is:** $Gini(t) = 1 - \sum_{i=1}^{c} p_i^2$ **where** p_i **is the proportion of class** i **instances, and** c **is the number of classes.**
17. **Gaussian Mixture Models (GMM):**
    - **The equation is:** $p(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x \mid \mu_k, \Sigma_k)$ **where** π_k **is the mixture coefficient, and** $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ **is the Gaussian distribution.**

18. **Chi-Square Test:**
    - **The equation is: $\chi^2 = \sum \frac{(O-E)^2}{E}$ where O is the observed frequency, and E is the expected frequency.**
19. **ANOVA F-Statistic:**
    - **The equation is: $F = \frac{\text{Mean Square Between}}{\text{Mean Square Within}}$**
20. **Confidence Interval for Population Mean:**
    - **The equation is: $\pm z \cdot \frac{\sigma}{\sqrt{n}}$ where is the sample mean, z is the critical value, σ is the population standard deviation, and n is the sample size.**
21. **Cox Proportional Hazards Model:**
    - **The equation is: $h(t \mid X) = h_0(t) \cdot \exp(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)$ where $h(t \mid X)$ is the hazard rate, $h_0(t)$ is the baseline hazard, and X_i are the covariates**
22. **Euler-Lagrange Equation:**
    - **The document mentions the Euler-Lagrange equation but does not provide the complete equation. The correct form is: $\frac{\partial L}{\partial y} - \frac{d}{dx}\left(\frac{\partial L}{\partial y'}\right) = 0$ where L is the Lagrangian, $y(x)$ is the function to be determined, and $y'(x)$ is its derivative.**
23. **Black-Scholes Equation:**
    - **The document mentions the Black-Scholes equation but does not provide the complete equation. The correct form is: $\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$ where $V(S,t)$ is the option price, σ is the volatility, S is the stock price, and r is the risk-free interest rate.**

The Goldman–Hodgkin–Katz (GHK) flux equation and its derivative, the GHK voltage equation, describe the ionic flux and membrane potential in terms of ionic concentrations and membrane permeability. These equations are fundamental to understanding the electrochemical basis of membrane potentials in cells.

---

1. **GHK Flux Equation**

The GHK flux equation for a monovalent ion M^+ is expressed as:

$$J_M = P_M \frac{zF^2}{RT} \frac{[M^+]_i - [M^+]_o e^{-zFV/RT}}{1 - e^{-zFV/RT}}$$

where:

- J_M is the ionic flux (amount of ion crossing the membrane per unit area per unit time),
- P_M is the permeability of the membrane to the ion M^+,
- z is the valence of the ion ($+1$ for M^+),
- F is the Faraday constant ($96{,}485$ C/mol),
- R is the gas constant (8.314 J/mol\cdotpK),
- T is the absolute temperature (in kelvins),
- V is the membrane potential (in volts),
- $[M^+]_i$ and $[M^+]_o$ are the intracellular and extracellular concentrations of M^+, respectively.

This equation considers the driving forces of diffusion and electric fields.

---

2. **GHK Voltage Equation**

For a membrane permeable to multiple ions, the GHK voltage equation gives the membrane potential V at steady state. For a membrane permeable to monovalent positive (M^+) and negative (A^-) ions, the equation is:

$$V = \frac{RT}{F} \ln\left(\frac{P_{M^+}[M^+]_o + P_{A^-}[A^-]_i}{P_{M^+}[M^+]_i + P_{A^-}[A^-]_o}\right)$$

where:

- P_{M^+} and P_{A^-} are the permeabilities of the membrane to M^+ and A^-, respectively,
- $[M^+]_o$ and $[A^-]_o$ are the extracellular concentrations,
- $[M^+]_i$ and $[A^-]_i$ are the intracellular concentrations.

This equation reflects the contributions of all permeant ions to the membrane potential, weighted by their relative permeabilities.

---

3. **Special Case for a Single Ion**

For a single monovalent ion, the GHK voltage equation simplifies to the Nernst equation:

$$V = \frac{RT}{zF} \ln\left(\frac{[M^+]_o}{[M^+]_i}\right)$$

GHK Flux Equation:

- Models how ions move across a membrane driven by both concentration gradients and the membrane potential.
- The numerator includes the net driving force (difference in ionic concentrations adjusted for voltage).
- The denominator accounts for the exponential nature of voltage effects.

GHK Voltage Equation:

- Predicts the steady-state membrane potential based on ion concentrations and permeabilities.
- Ions with higher permeability and larger gradients contribute more to V.

These equations are foundational in physiology, particularly for excitable cells like neurons and muscle fibers.

1. **Linear Regression (Supervised Learning)**

Linear regression is often used to predict a continuous outcome (e.g., pain intensity scores) based on one or more features (e.g., age, medication type, genetic markers).

Equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where:

- y = the predicted output (e.g., pain score).
- β_0 = the intercept.
- $\beta_1, \ldots, \beta_n$ = coefficients that represent the relationship between each feature and the outcome.
- $x_1, x_2, \ldots, x_n$ = input features (e.g., patient age, surgery type, etc.).
- ϵ = error term (accounting for model inaccuracies).

In AI, regression models are used to predict outcomes based on known relationships between features and results.

---

2. **Logistic Regression (Classification)**

For binary outcomes (e.g., predicting whether a patient will experience chronic pain after surgery), **logistic regression** is used. It calculates the probability of an event occurring.

Equation:

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

Where:

- $p(y = 1 \mid x)$ = probability of the event happening (e.g., chronic pain).
- e = base of the natural logarithm.
- Other terms are the same as in linear regression.

This is useful for classification problems, where the model predicts the probability of different outcomes (e.g., chronic vs. acute pain).

---

3. **Support Vector Machines (SVM)**

SVM is a classification technique that finds the optimal hyperplane to separate different classes (e.g., predicting if a patient will respond well to a particular treatment).

Equation:

$$f(x) = w^T x + b$$

Where:

- $f(x)$ = decision function.
- w = weight vector.
- x = input features (e.g., patient data).
- b = bias term.
- T = transpose (used to align the dimensionalities).

The SVM maximizes the margin between different classes, where the margin is the distance between the closest points of each class and the decision boundary.

---

4. **Neural Networks (Deep Learning)**

Neural networks are composed of multiple layers of neurons (nodes), and each neuron is activated by a mathematical function (e.g., sigmoid, ReLU). The network learns by adjusting weights and biases during training.

Equation: The activation of each neuron can be represented as:

$$a_i = f\left(\sum_{j=1}^{n} w_{ij}\, x_j + b_i\right)$$

Where:

- a_i = activation of the i-th neuron.
- w_{ij} = weight from the j-th input to the i-th neuron.
- x_j = input to the neuron.
- b_i = bias term for the neuron.
- f = activation function (e.g., ReLU, Sigmoid).

The output from each layer is passed to the next until the final output is calculated. In pain prediction, this could involve layers that analyze various data like pain scores, imaging, and physiological signals.

---

5. **Convolutional Neural Networks (CNNs) for Image Data**

CNNs are used for processing image data, such as analyzing MRI or ultrasound images to identify pain-related biomarkers.

Equation:

$$Y = f(W * X + b)$$

Where:

- Y = output (feature map).
- W = filter (weights).

- X = input image data.
- b = bias term.
- $*$ = convolution operation.

CNNs use convolutional layers to extract local features from images (e.g., anatomical features from MRI scans), which are then processed further in the network.

---

6. **Recurrent Neural Networks (RNNs) for Sequential Data**

RNNs are used when the data has a temporal (sequential) component, such as predicting pain over time or modeling patient behavior.

Equation:

$$h_t = f(W_h h_{t-1} + W_x x_t + b_h)$$

Where:

- h_t = hidden state at time t.
- W_h = weight matrix for the previous state.
- h_{t-1} = previous hidden state.
- W_x = weight matrix for the current input.
- x_t = input at time t.
- b_h = bias term.
- f = activation function (e.g., tanh, ReLU).

This equation models the relationship between previous pain episodes and future pain, which is crucial in chronic pain prediction.

---

7. **Reinforcement Learning**

In reinforcement learning (RL), an agent learns through trial and error, with the goal of maximizing cumulative reward. In pain management, an RL system might optimize treatment strategies by testing different interventions and adjusting based on patient feedback.

Equation (Q-learning):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_a Q(s',a) - Q(s,a) \right]$$

Where:

- $Q(s,a)$ = expected reward for taking action a in state s.
- α = learning rate (controls how much new information overrides the old).
- r = immediate reward (e.g., pain relief from treatment).
- γ = discount factor (controls importance of future rewards).
- $\max_a Q(s',a)$ = maximum expected future reward from next state s'.

This equation helps the AI optimize pain treatment by learning which actions lead to the best outcomes over time.

Transformer Equations

Transformers are the foundation of many modern AI systems, including GPT-like models. They primarily use the **attention mechanism** and deep neural network concepts.

1. Scaled Dot-Product Attention

Attention is a function that maps a query Q, key K, and value V to an output. The scaled dot-product attention is:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- Q: Query matrix.
- K: Key matrix.
- V: Value matrix.
- d_k: Dimensionality of the key vectors (used to scale the dot product).
- softmax: A function that normalizes the output to probabilities.

2. Multi-Head Attention

Multi-head attention allows the model to attend to information from different representation subspaces simultaneously:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

Where each head is computed as:

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

- W_i^Q, W_i^K, W_i^V: Parameter matrices for the i-th head.
- W^O: Output weight matrix.

3. Positional Encoding

Since transformers lack inherent sequentiality, positional encoding is added to the input embeddings:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Where:

- pos: Position in the sequence.
- i: Dimension index.
- d_{model}: Embedding dimension.

Variational Autoencoders (VAEs)

VAEs are generative models that encode data into a latent space and generate new samples by sampling from this space.

1. Evidence Lower Bound (ELBO)

The objective of a VAE is to maximize the ELBO, which is a combination of reconstruction loss and latent regularization:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(z|x)}[\log p(x \mid z)] - D_{\text{KL}}\left(q(z \mid x) \parallel p(z)\right)$$

Where:

- $q(z \mid x)$: Approximate posterior distribution (learned by the encoder).
- $p(z)$: Prior distribution (commonly standard normal).
- $\log p(x \mid z)$: Log-likelihood of reconstructing x from z.
- D_{KL}: Kullback-Leibler divergence, penalizing divergence between $q(z \mid x)$ and $p(z)$.

2. Reparameterization Trick

To make sampling differentiable, VAEs use the reparameterization trick:

$$z = \mu(x) + \sigma(x) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$$

Where:

- $\mu(x)$: Mean of the latent space distribution.
- $\sigma(x)$: Standard deviation of the latent space distribution.

Generative Adversarial Networks (GANs)

GANs consist of two neural networks: a generator G and a discriminator D, trained adversarially.

1. GAN Objective

The goal is for G to generate realistic data while D distinguishes real from fake data:

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log\left(1 - D\big(G(z)\big)\right)]$$

Where:

- p_{data}: Distribution of real data.
- p_z: Latent space distribution (commonly standard normal or uniform).
- $G(z)$: Generated sample.

Diffusion Models

Diffusion models generate data by gradually transforming random noise into structured outputs.

1. Forward Process

Noise is incrementally added to the data x_0 over T steps:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t) I\big)$$

Where:

- x_t: Data at step t.
- α_t: Noise scheduling parameter.

2. Reverse Process

The reverse process learns to denoise x_t step-by-step to recover x_0:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big)$$

3. Training Objective

The model is trained to minimize the denoising error:

$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}[\| \epsilon - \epsilon_\theta(x_t, t) \|^2]$$

Where:

- ϵ: True noise added in the forward process.
- ϵ_θ: Predicted noise by the model.

Attention-Based Models in Unconventional Applications

1. Cross-Attention

Cross-attention allows one sequence (e.g., text) to attend to another (e.g., image embeddings):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q, K, V are derived from separate modalities.

Reinforcement Learning with Transformers (RLHF)

Transformers fine-tuned with Reinforcement Learning from Human Feedback (RLHF) update their policies based on rewards:

$$\pi_{\text{new}}(a \mid s) \propto \pi_{\text{old}}(a \mid s)\exp\left(\frac{R(s,a)}{\beta}\right)$$

Where:

- π: Policy.
- $R(s, a)$: Reward for state-action pair.
- β: Temperature parameter controlling exploration.

Connections Pain Prediction and AI Equations

1. **Prediction of Pain Trajectories**:
    - **Concept in Document**: The document describes the use of machine learning to predict acute pain, chronic pain development, and medication responses.
    - **Equation Connection**:
        - Logistic regression ($p(y \mid x) = \frac{1}{1+e^{-(\beta_0+\sum\beta_i x_i)}}$) could classify patients at risk of developing chronic pain.
        - Neural networks ($a_i = f\left(\sum w_{ij}x_j + b_i\right)$) could model more complex, non-linear relationships between patient features and pain outcomes.
2. **Real-Time Treatment Adaptation**:
    - **Concept in Document**: Wearable sensors combined with AI dynamically adjust pain management plans based on real-time data.
    - **Equation Connection**:
        - Recurrent Neural Networks (RNNs) ($h_t = f(W_h h_{t-1} + W_x x_t + b_h)$) can process sequential sensor data to detect pain patterns and suggest immediate adjustments.
        - Reinforcement Learning ($Q(s, a)$) could optimize treatment regimens by learning the best actions (e.g., medication adjustments) for different patient states.
3. **Advanced Ultrasound Guidance**:
    - **Concept in Document**: AI enhances ultrasound imaging by identifying anatomical features for more precise interventions.
    - **Equation Connection**:
        - Convolutional Neural Networks (CNNs) ($Y = f(W * X + b)$) can process ultrasound images to segment tissues, overlay critical structures, and guide clinicians in real-time.
4. **Self-Management Tools**:
    - **Concept in Document**: AI-powered apps for cognitive-behavioral therapy or tailored exercise recommendations.
    - **Equation Connection**:

- Transformers (Attention$(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$) could power chatbots for therapy sessions by understanding and responding to patient queries effectively.

1. Breeder's equation: where R is the response to selection, h^2 is the narrow-sense heritability, and S is the selection differential.

$$R = h^2 S$$

2. Hardy–Weinberg principle: where p and q are the allele frequencies for a gene with two alleles.

$$p^2 + 2pq + q^2 = 1$$

3 Hill equation: where θ is the fraction of ligand-binding sites occupied, $[L]$ is the ligand concentration, K_d is the dissociation constant, and n is the Hill coefficient.

$$\theta = \frac{[L]^n}{K_d^n + [L]^n}$$

4. Lotka–Volterra equations: where x is the prey population, y is the predator population, and $\alpha, \beta, \gamma, \delta$ are parameters.

$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$

5. Michaelis–Menten equation: where v is the reaction velocity, $V_{\max}$ is the maximum reaction velocity, $[S]$ is the substrate concentration, and K_m is the Michaelis constant.

$$v = \frac{V_{\max}[S]}{K_m + [S]}$$

6. Poiseuille equation: where Q is the volumetric flow rate, r is the radius of the tube, ΔP is the pressure difference, η is the dynamic viscosity, and l is the length of the tube.

$$Q = \frac{\pi r^4 \Delta P}{8\eta l}$$

7. Price equation: where Δz is the change in the average trait value, w_i is the relative fitness, z_i is the trait value, Cov is the covariance, and $\mathbb{E}$ is the expected value.

$$\Delta z = \text{Cov}(w_i, z_i) + \mathbb{E}(w_i \Delta z_i)$$

8. Bayesian Theorem for Probability Updating: where $P(A \mid B)$ is the posterior probability, $P(B \mid A)$ is the likelihood, $P(A)$ is the prior, and $P(B)$ is the marginal probability.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

9. K-Means Clustering Objective Function: where S_i are the clusters, x are the data points, μ_i are the centroids, and k is the number of clusters.

$$\sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

10. Principal Component Analysis (PCA) - Eigenvalue Equation: where A is the covariance matrix, v is the eigenvector, and λ is the eigenvalue.

$$A \cdot v = \lambda \cdot v$$

11. Gradient Descent Update: where θ are the parameters, α is the learning rate, and $J(\theta)$ is the cost function.

$$\theta = \theta - \alpha \cdot \nabla J(\theta)$$

12. Stochastic Gradient Descent (SGD) for Machine Learning: where $x^{(i)}$ and $y^{(i)}$ are a single training example, and other terms are as defined above.

$$\theta = \theta - \alpha \cdot \nabla J\left(\theta; x^{(i)}, y^{(i)}\right)$$

13. Naive Bayes Classifier: where $P(y \mid x_1, \ldots, x_n)$ is the posterior probability, and $\prod$ indicates the product over all features.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P\,(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

14. Decision Trees - Gini Impurity for Splitting: where p_i is the proportion of class i instances, and c is the number of classes.

$$Gini(t) = 1 - \sum_{i=1}^{c} p_i^2$$

15. Gaussian Mixture Models (GMM) - Probability Density: where π_k is the mixture coefficient, $\mathcal{N}(x \mid \mu_k, \Sigma_k)$ is the Gaussian distribution, and K is the number of components.

$$p(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

16. Chi-Square Test for Independence: where O is the observed frequency and E is the expected frequency.

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

17. ANOVA F-Statistic:

$$F = \frac{\text{Mean Square Between}}{\text{Mean Square Within}}$$

18. Confidence Interval for Population Mean: where X is the sample mean, z is the critical value, σ is the population standard deviation, and n is the sample size.

$$\bar{X} \pm z \cdot \frac{\sigma}{\sqrt{n}}$$

19. Cox Proportional Hazards Model: where $h(t \mid X)$ is the hazard rate, $h_0(t)$ is the baseline hazard, X_i are the covariates, and β_i are the coefficients.

$$h(t \mid X) = h_0(t) \cdot \exp\left(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p\right)$$

20. Bayesian Theorem for Probability Updating: Used for updating probabilities based on new evidence, essential in Bayesian networks and statistical inference.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

21. K-Means Clustering Objective Function: Where S is the set of clusters, μ_i is the centroid of cluster i, and x is a data point. This minimizes within-cluster variance.

$$\sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

22. Principal Component Analysis (PCA) - Eigenvalue Equation: Where A is the covariance matrix, v is an eigenvector (principal component), and λ is the corresponding eigenvalue.

$$A \cdot v = \lambda \cdot v$$

23. Gradient Descent Update Rule: Where θ are the parameters to be updated, α is the learning rate, and $\nabla J(\theta)$ is the gradient of the cost function with respect to θ.

$$\theta := \theta - \alpha \cdot \nabla J(\theta)$$

24. Stochastic Gradient Descent (SGD): Similar to gradient descent but uses one sample or mini-batch at a time.

$$\theta := \theta - \alpha \cdot \nabla J\left(\theta; x^{(i)}, y^{(i)}\right)$$

25. Naive Bayes Classifier: Based on the assumption of conditional independence between every pair of features given the value of the class variable.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P\left(x_i \mid y\right)}{P(x_1, \ldots, x_n)}$$

26. Decision Trees - Gini Impurity: Where p_i is the probability of an item with class i at node t, and c is the number of classes.

$$Gini(t) = 1 - \sum_{i=1}^{c} p_i^2$$

27. Gaussian Mixture Models (GMM): Where π_k are the mixture weights, μ_k and Σ_k are the mean and covariance of component k, and $\mathcal{N}$ denotes the normal distribution.

$$p(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

28. Chi-Square Test for Independence: Where O is observed frequency, and E is expected frequency under the null hypothesis.

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

29. ANOVA F-Statistic: Used to test if there's a significant difference between group means.

$$F = \frac{\text{Mean Square Between}}{\text{Mean Square Within}}$$

30. Confidence Interval for Population Mean: For large samples or known population standard deviation σ, where z is the z-score for the desired confidence level, $\bar{X}$ is the sample mean, and n is the sample size.

$$\bar{X} \pm z \cdot \frac{\sigma}{\sqrt{n}}$$

31. Cox Proportional Hazards Model: Used in survival analysis, where $h(t \mid X)$ is the hazard at time t given covariates X, $h_0(t)$ is the baseline hazard, and β_i are the coefficients.

$$h(t \mid X) = h_0(t) \cdot \exp\left(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p\right)$$

32. The Euler-Lagrange equation is fundamental in the calculus of variations. It's used to find the function $y(x)$ that extremizes a given functional $J[y]$:

$$J[y] = \int_a^b L\left(x, y(x), y'(x)\right) dx$$

where L is the Lagrangian, $y(x)$ is the function to be determined, and $y'(x)$ is the derivative of y with respect to x. The necessary condition for $y(x)$ to extremize J is:

$$\frac{\partial L}{\partial y} - \frac{d}{dx}\left(\frac{\partial L}{\partial y'}\right) = 0$$

For $J[y] = \int_0^1 (y'^2 - y^2)\ dx$, with $L(x, y, y') = y'^2 - y^2$:

$$\frac{\partial L}{\partial y} = -2y, \quad \frac{\partial L}{\partial y'} = 2y'$$

The Black-Scholes equation, used for pricing European-style options, can be thought of as having conceptual parallels with the Euler-Lagrange equation in the context of optimization under market conditions. The equation for a call option price $V(S, t)$ is:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

where:

- σ is the volatility of the stock's returns,
- r is the risk-free interest rate,
- S is the stock price,
- t is time.

Connections to Euler-Lagrange:

- Both equations deal with optimization under constraints; while Euler-Lagrange optimizes a functional, Black-Scholes optimizes option pricing under no-arbitrage conditions.
- The terms in the Black-Scholes equation can be seen as analogous to parts of the Lagrangian in the Euler-Lagrange context:
    - $\frac{\partial V}{\partial t}$ reflects time decay of option value.
    - $\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$ accounts for volatility effects on option pricing.

- $rS\frac{\partial V}{\partial S}$ and $-rV$ handle the cost of carry and discounting effects, respectively.

While not directly derived from variational calculus like the Euler-Lagrange equation, the Black-Scholes model uses similar principles of optimization within a stochastic framework to find the fair price of options.