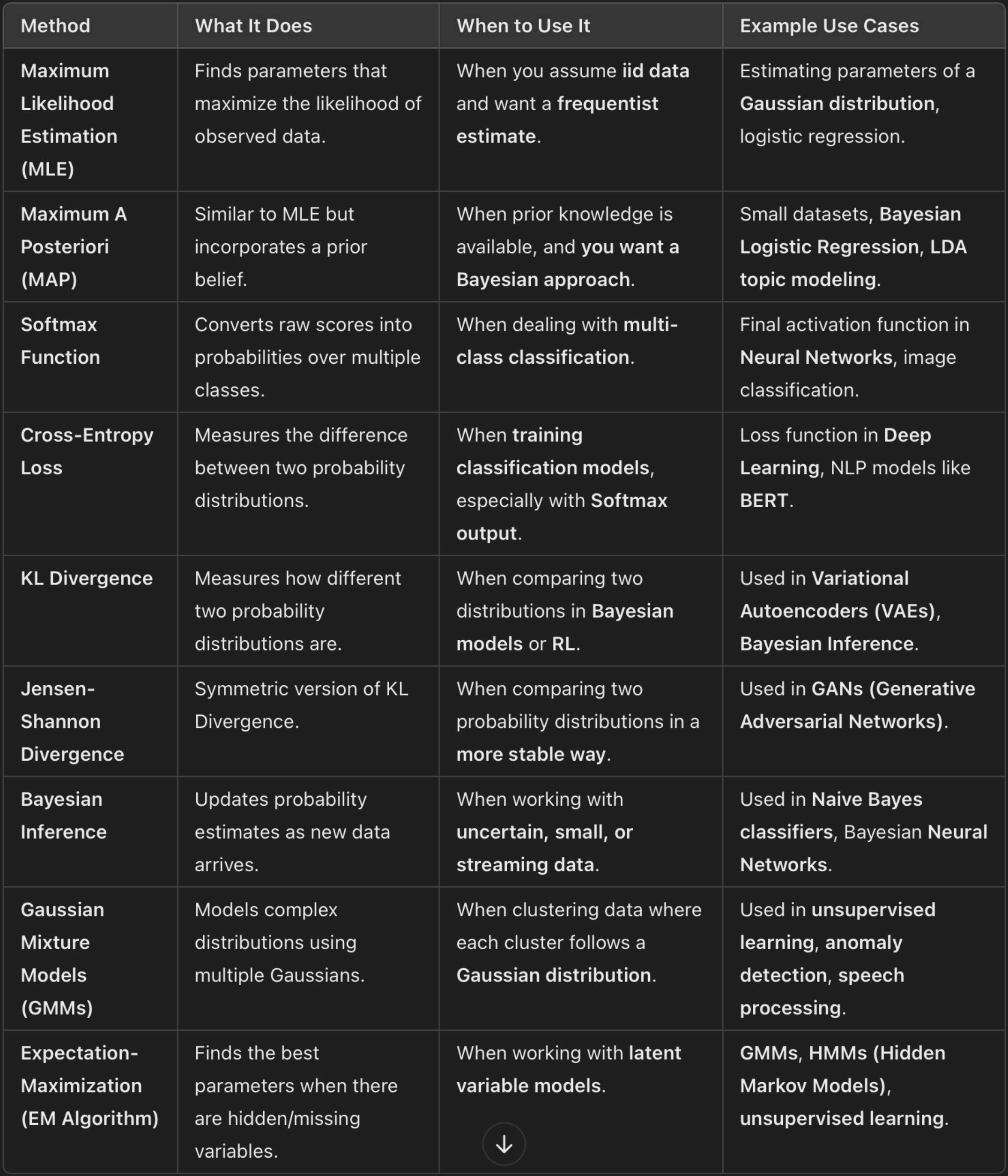
Statistical Methods in Machine Learning – A Deep Dive



Method	What It Does	When to Use It	Example Use Cases
Maximum Likelihood Estimation (MLE)	Finds parameters that maximize the likelihood of observed data.	When you assume **iid data** and want a **frequentist estimate**.	Estimating parameters of a **Gaussian distribution**, logistic regression.
Maximum A Posteriori (MAP)	Similar to MLE but incorporates a prior belief.	When prior knowledge is available, and **you want a Bayesian approach**.	Small datasets, **Bayesian Logistic Regression**, **LDA topic modeling**.
Softmax Function	Converts raw scores into probabilities over multiple classes.	When dealing with **multi-class classification**.	Final activation function in **Neural Networks**, image classification.
Cross-Entropy Loss	Measures the difference between two probability distributions.	When **training classification models**, especially with **Softmax output**.	Loss function in **Deep Learning**, NLP models like **BERT**.
KL Divergence	Measures how different two probability distributions are.	When comparing two distributions in **Bayesian models** or **RL**.	Used in **Variational Autoencoders (VAEs)**, **Bayesian Inference**.
Jensen-Shannon Divergence	Symmetric version of KL Divergence.	When comparing two probability distributions in a **more stable way**.	Used in **GANs (Generative Adversarial Networks)**.
Bayesian Inference	Updates probability estimates as new data arrives.	When working with **uncertain, small, or streaming data**.	Used in **Naive Bayes classifiers**, Bayesian **Neural Networks**.
Gaussian Mixture Models (GMMs)	Models complex distributions using multiple Gaussians.	When clustering data where each cluster follows a **Gaussian distribution**.	Used in **unsupervised learning**, **anomaly detection**, **speech processing**.
Expectation-Maximization (EM Algorithm)	Finds the best parameters when there are hidden/missing variables.	When working with **latent variable models**.	**GMMs**, **HMMs (Hidden Markov Models)**, **unsupervised learning**.

1. Maximum Likelihood Estimation (MLE)

Intuition: What Problem Does MLE Solve?

MLE is a method for **estimating parameters** of a probability distribution that **maximize the likelihood of observed data**.

Example: Coin Flip

Imagine flipping a biased coin n **times**, and we want to estimate the probability p of landing heads. Given outcomes $X = [H, T, H, H, T]$, the best estimate of p should be the one that **maximizes the likelihood** of observing these results.

Mathematical Formulation

Given data $X = \{x_1, x_2, ..., x_n\}$ and a probability model $P(X|\theta)$ with parameters θ, we estimate θ by:

$$\theta^* = \arg\max_{\theta} P(X|\theta)$$

For independent data points:

$$L(\theta) = \prod_{i=1}^{n} P(x_i|\theta)$$

Taking the **log-likelihood** (to make computation easier):

$$\log L(\theta) = \sum_{i=1}^{n} \log P(x_i|\theta)$$

To find the best θ, we **differentiate** and solve:

$$\frac{d}{d\theta} \log L(\theta) = 0$$

Pseudo Code for MLE

```plaintext
1. Define the probability distribution P(X | θ).
2. Compute the log-likelihood function.
3. Differentiate with respect to θ.
4. Solve for θ that maximizes the likelihood.
```

2. Maximum A Posteriori (MAP) Estimation

Intuition: Why Do We Need MAP?

MLE maximizes **only the likelihood**, ignoring any **prior knowledge**. MAP fixes this by incorporating **Bayes' Theorem**.

Example: Coin Flip Again

- Suppose we **already believe** the coin is biased towards heads.
- MLE **ignores** this prior belief.
- MAP **combines likelihood and prior**.

Mathematical Formulation

Using **Bayes' Rule**:

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

MAP estimation **maximizes the posterior**:

$$\theta^* = \arg\max_\theta P(\theta|X)$$

Using **logarithms**:

$$\theta^* = \arg\max_\theta [\log P(X|\theta) + \log P(\theta)]$$

Difference from MLE?

- MLE: **Only uses** likelihood $P(X|\theta)$.
- MAP: **Also considers** prior $P(\theta)$.

Pseudo Code for MAP

plaintext

```
1. Define the likelihood function P(X | θ).
2. Define the prior distribution P(θ).
3. Compute the posterior using Bayes' rule.
4. Maximize log posterior to find best θ.
```

3. Softmax Function

Intuition: Why Do We Need Softmax?

Softmax converts a **vector of arbitrary real numbers** into **probabilities** that sum to 1.

Example: Multi-Class Classification

Given class scores $[3.1, 2.5, 1.2]$, we want to **convert them into probabilities**.

Mathematical Formulation

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

where:

- z_i is the **raw score** for class i.
- $P(y_i)$ is the **probability** of class i.

Pseudo Code for Softmax

plaintext Copy Edit

```plaintext
1. Compute exponentials of all input scores.
2. Sum all exponentials.
3. Divide each exponentiated score by the sum.
```

4. Cross-Entropy Loss

Intuition: Why Do We Need Cross-Entropy?

Cross-entropy measures **how different two probability distributions are**. It is used as a **loss function** in classification.

Example: Classification

If the model predicts **P(cat) = 0.9**, and the correct label is **cat**, we want the loss to be **small**. If it predicts **P(dog) = 0.9**, we want the loss to be **high**.

Mathematical Formulation

For binary classification:

$$H(p, q) = -\sum p_i \log q_i$$

where:

- p is the **true probability** (ground truth).
- q is the **predicted probability**.

For multi-class:

$$L = -\sum_{i=1}^{n} y_i \log \hat{y}_i$$

Pseudo Code for Cross-Entropy

plaintext

```
1. Compute -log of predicted probability for the correct class.
2. Sum across all training examples.
3. Take the average.
```

1. Kullback-Leibler (KL) Divergence

Intuition: What Does KL Divergence Do?

KL Divergence measures **how different one probability distribution is from another.**

Example Intuition

- Suppose we have two distributions:
    - $P(x)$ (true distribution)
    - $Q(x)$ (our model's approximation)
- KL divergence tells us **how much information is lost** when we approximate $P(x)$ using $Q(x)$.

Mathematical Formulation

For discrete distributions:

$$D_{\mathrm{KL}}(P||Q) = \sum_i P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

For continuous distributions:

$$D_{\mathrm{KL}}(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx$$

- If $P(x) = Q(x)$, KL divergence is **zero.**
- **KL divergence is not symmetric:** $D_{\mathrm{KL}}(P||Q) \neq D_{\mathrm{KL}}(Q||P)$.

Pseudo Code for KL Divergence

```plaintext
1. Take two probability distributions P and Q.
2. Compute log(P / Q) for each value.
3. Multiply by P.
4. Sum over all values.
```

2. Jensen-Shannon Divergence (JSD)

Intuition: Why Use JSD Instead of KL Divergence?

- KL divergence is **not symmetric**.
- JSD **fixes this by averaging KL divergences** between two distributions and a midpoint distribution.

Mathematical Formulation

Define the midpoint distribution:

$$M(x) = \frac{1}{2}(P(x) + Q(x))$$

JSD is:

$$D_{\text{JS}}(P||Q) = \frac{1}{2}D_{\text{KL}}(P||M) + \frac{1}{2}D_{\text{KL}}(Q||M)$$

- Unlike KL divergence, **JSD is symmetric**.
- JSD is **bounded between 0 and 1**, making it more stable.

Pseudo Code for Jensen-Shannon Divergence

plaintext Copy Edit

```plaintext
1. Compute M = (P + Q) / 2.
2. Compute KL(P || M) and KL(Q || M).
3. Take the average of both KL divergences.
```

3. Bayesian Inference

Intuition: Why Use Bayesian Inference?

- Unlike **MLE**, Bayesian inference **updates beliefs** as more data is observed.
- Instead of a single **best** parameter value, it provides a **probability distribution over parameters**.

Mathematical Formulation (Bayes' Theorem)

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

where:

- $P(\theta|X)$ → **Posterior** (updated belief about θ after seeing data)
- $P(X|\theta)$ → **Likelihood** (how well θ explains data)
- $P(\theta)$ → **Prior** (belief about θ before seeing data)
- $P(X)$ → **Evidence** (normalizing constant)

Pseudo Code for Bayesian Inference

plaintext Copy Edit

```plaintext
1. Define prior P(θ).
2. Compute likelihood P(X | θ).
3. Compute posterior P(θ | X) using Bayes' rule.
```

4. Gaussian Mixture Models (GMMs)

Intuition: Why Use GMM?

- **K-Means assumes clusters are spherical** (equal variance).
- GMM allows **clusters of different shapes** by using **multiple Gaussian distributions**.

Mathematical Formulation

A GMM models data as a **weighted sum of multiple Gaussians**:

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

where:

- π_k → Mixing coefficient (probability of cluster k)
- $\mathcal{N}(x|\mu_k, \Sigma_k)$ → Gaussian distribution for cluster k
- K → Number of clusters

Pseudo Code for GMM

```plaintext
1. Initialize means, covariances, and mixing coefficients.
2. E-Step: Compute probabilities of each point belonging to each cluster.
3. M-Step: Update parameters using weighted averages.
4. Repeat until convergence.
```

5. Expectation-Maximization (EM Algorithm)

Intuition: Why Use EM?

- When data has **missing values or latent variables**, direct optimization is difficult.
- EM **iteratively estimates hidden variables and optimizes parameters**.

Mathematical Formulation

1. **E-Step**: Compute the **expected value** of hidden variables.
2. **M-Step**: Maximize the expected likelihood.

For parameter θ:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}\left[\log P(X, Z|\theta)|X, \theta^{(t)}\right]$$

Update:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

Expectation-Maximization (EM) Algorithm – Pseudo Code with Explicit Math

1. **Initialize parameters:**

$$\theta^{(0)} = \{\pi_k, \mu_k, \Sigma_k\} \quad \text{(randomly initialized)}$$

2.

Repeat until convergence:

Step 1: Expectation Step (E-Step)

Compute the **responsibilities** (posterior probability of cluster k for each data point i):

$$\gamma_{ik} = \frac{\pi_k^{(t)} \mathcal{N}(x_i | \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)} \mathcal{N}(x_i | \mu_j^{(t)}, \Sigma_j^{(t)})}$$

where $\mathcal{N}(x|\mu, \Sigma)$ is the Gaussian probability density function:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Step 2: Maximization Step (M-Step)

Update the **mixing coefficients**:

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}$$

Update the **means**:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

Update the **covariances**:

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{N} \gamma_{ik}}$$

Step 3: Compute Log-Likelihood

Compute the **log-likelihood** to track convergence:

$$L(\theta^{(t)}) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k^{(t)} \mathcal{N}(x_i | \mu_k^{(t)}, \Sigma_k^{(t)})$$

Step 4: Check Convergence

If the **change in log-likelihood** is below a threshold ϵ:

$$|L(\theta^{(t+1)}) - L(\theta^{(t)})| < \epsilon$$

then **stop the iterations**.

Key Takeaways

✅ The E-step computes expected assignments of data points to clusters (soft clustering).

✅ The M-step updates the model parameters using these expectations.

✅ Convergence is determined by log-likelihood improvement.

✅ Used in Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), and other latent variable models.