Reflective Reinforced Learning with Sensational Reward in Higher Dimensional Space

Than Lwin Aung

Mango AI

Strategy

A strategy is an action or a sequence of actions, associated with an outcome.

More formally, a **Strategy** is a set of order pairs of **Action** and **Outcome**, such that:

$$S = \{a, o\}$$

OR

$$S = \{a_1, \ldots, a_n, o\}$$

Thus, strategic thinking implies that we are able to think a sequence of actions and their outcome.

Although a strategy can have one or many actions, it can only have a single outcome.

Decision

A decision is finding the best strategy from a collection of strategies.

More formally, a **Decision** is an **Optimization** of a set of strategies with respect to their outcomes:

$$D = \arg max \{s_1, \dots, s_n\}$$

To put it more simply, a decision is finding an optimal strategy which has a maximum payoff or minimum cost or simply optimal outcome.

An agent, which is able to make a decision, is commonly known as **Decision Maker** (DM) or simply **Player**.

Decision Making is generally considered as: Player vs. Environment (PVE) and Player vs. Player (PVP).

Decision Matrix

When we define a table with strategies and outcomes, we get a Decision Matrix.

More formally, a **Decision Matrix** is an **Joint Set** of strategies with respect to their joint outcomes:

Player / Environment (Nature)

Player

	S_1	**S_2**	**S_3**
S_a	O_1	O_2	O_3
S_b	O_4	O_5	O_6
S_c	O_7	O_8	O_9

Dominant Strategy

Dominant Strategy is the Strategy with Optimal Outcome.

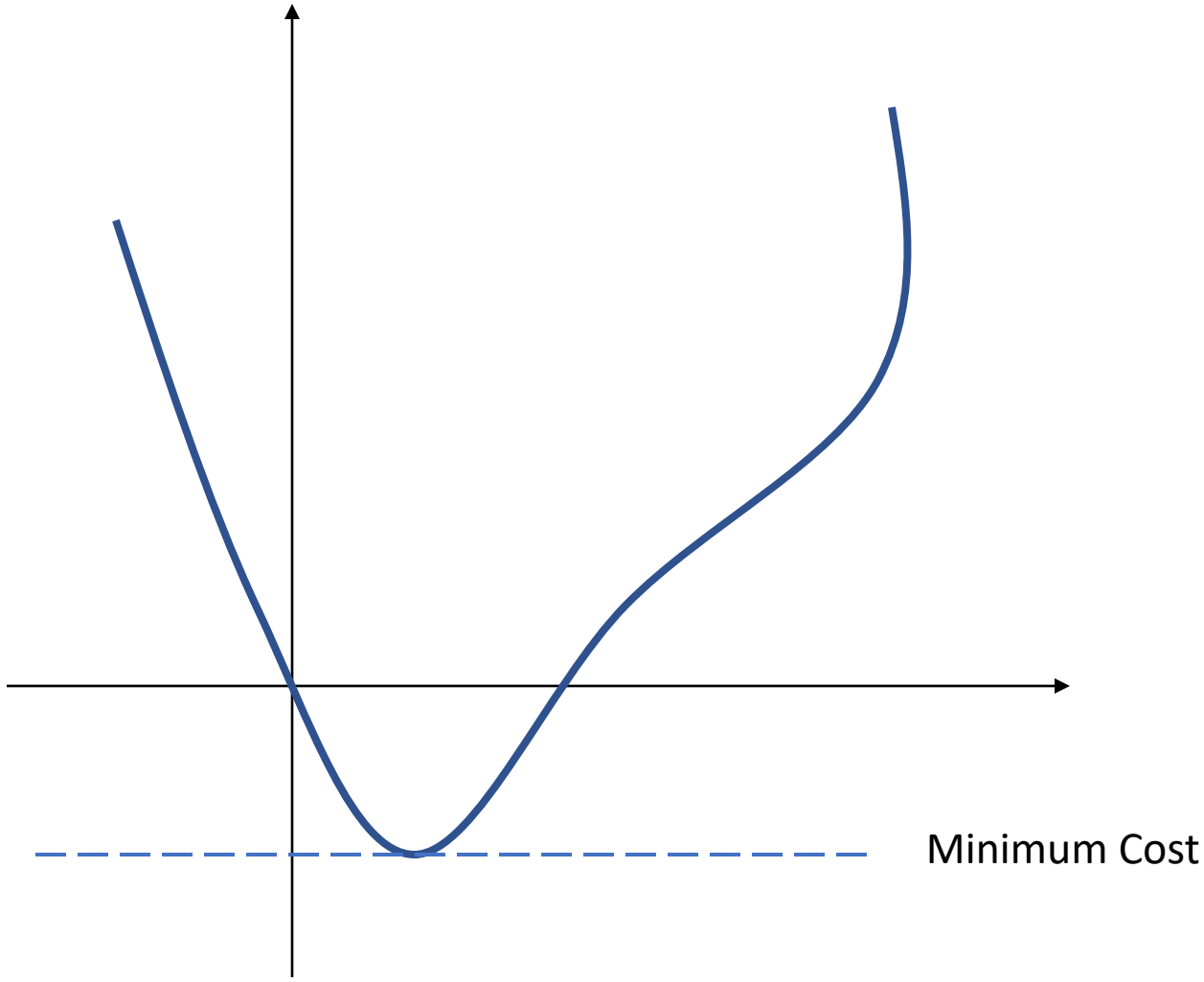
More formally, a Dominant Strategy exists if and only if it satisfies the following constraints:

$$S_m > S_1 > \cdots > S_n$$

As long as there exists a dominant strategy in a set of strategies, it is possible to make a “Rational Decision”, i.e., to choose the strategy with optimal outcome (S_m).

To make a “Rational Decision”, the outcomes of strategies cannot be cyclic.

Dominant Strategy



Rationality

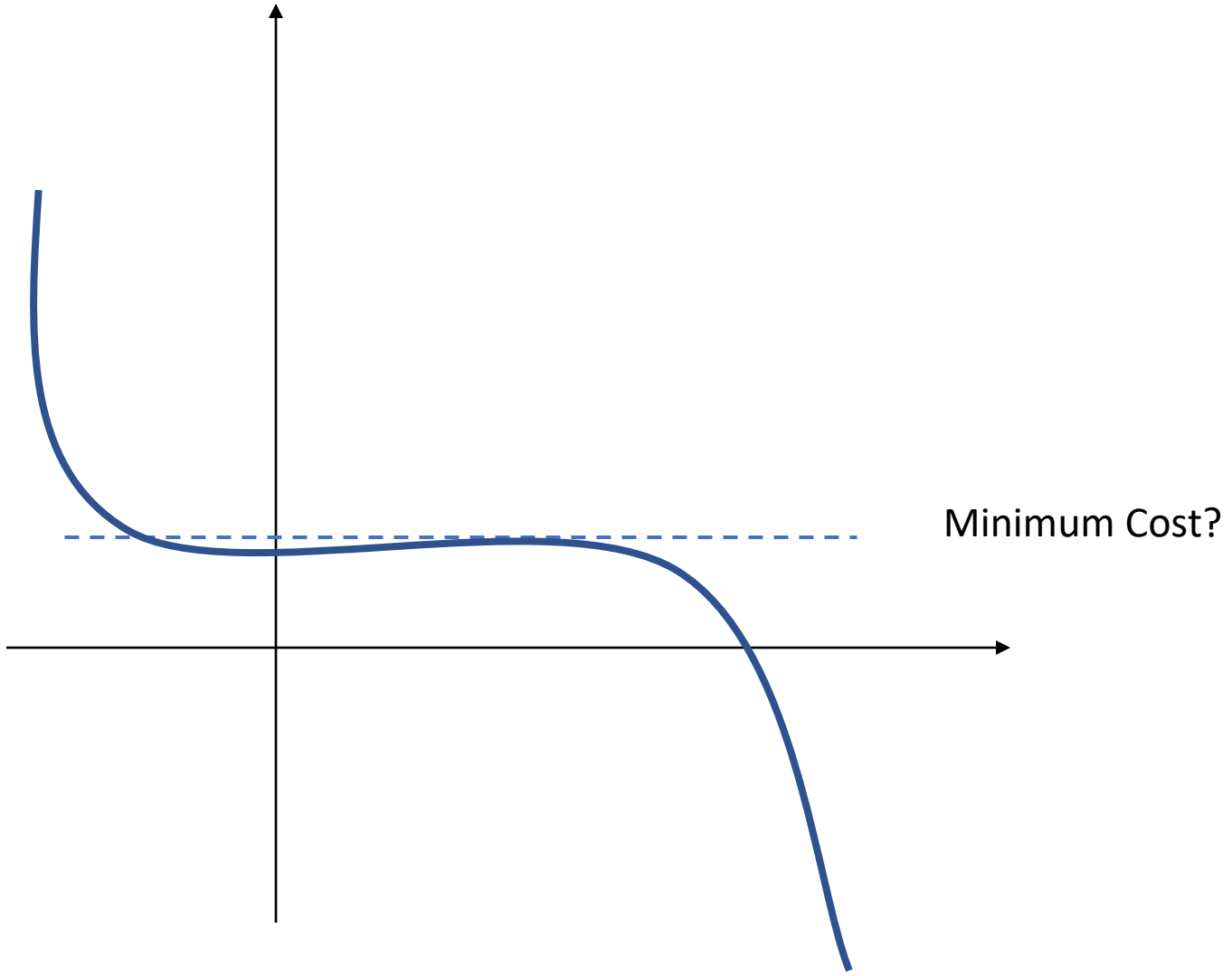
Rationality is the **precondition** in which the following constraint is satisfied to allow a Dominant Strategy, $\mathbf{S_m}$, such that:

$$s_m > s_1 > \cdots > s_n$$

Rationality is the also the **postcondition** in which Dominant Strategy, $\mathbf{S_m}$, is always selected, if it exists.

$$D = \arg max \{s_1, \ldots, s_n\}$$

Saddle Point



Nash Equilibrium

Sadly, a dominant strategy cannot exists all the time; instead of reaching a maximum or minimum, a set of strategies can be stuck at Saddle Point.

Formally, a **Nash Equilibrium** is defined as a Saddle Point, in which Strategies of Players (DM) reach an Equilibrium, without any dominant strategy :

To satisfy Nash Equilibrium, the following constraints mush exist:

$$S_m > S_1 > \cdots > S_n > S_m$$

A famous example is the Battle of Sexes, in which the wife wants to see a movie while the husband wants to see a soccer match.

Sequential Decisions

So far, a Decision is represented as a Set of Order Pairs of Strategies and Outcomes, as we consider a “Single Decision”.

However, decisions can be sequential decisions, in which one outcome of a decision could affect the following decisions.

Therefore, a sequential decision need to consider intermediate outcomes (**States**), in addition to **Actions** and final outcome (**Goal**).

Formally, a **Sequential Decision Process** is an **Optimization** of a sequence of decisions from **Initial State** to **Final State**, which defines the **Trajectory of Decision Sequence**.

$$SDP = \arg max \{\{d_{11}, \dots, d_{1n}\}, \dots, \{d_{m1}, \dots, d_{mn}\}\}$$

Plan

A plan is a sequence of decisions, which defines a **single trajectory** of sequence of decisions.

More formally, a **Plan** is a **Deterministic Trajectory** of Sequential Decisions.

$$Plan = \{d_1, \dots, d_n\}$$

Actually, a **Software Program** can be considered as a “Plan”, in which a well-defined sets of instructions are pre-written which cannot be changed at Run-Time or Production Environment.

Policy

A policy is a sequence of decisions, which defines **multiple possible trajectories** of sequence of decisions.

More formally, a **Policy** is a **Stochastic Trajectory** of Sequential Decisions:

$$Policy = \arg max \{\{d_{11}, \dots, d_{1n}\}, \dots, \{d_{m1}, \dots, d_{mn}\}\}$$

What makes a Policy different from a Plan is that a Policy is adaptable to the environment.

In addition, Policy can be optimized with Policy Learning, which is later known as “Reinforced Learning”.

Markov Decision Process

Markov Decision Process is a Policy Optimization Algorithm, with the assumption that current decision is only affected by previous outcome of the decision.

More formally, a **Markov Decision Process** is a **Stochastic Policy Optimization** which satisfies the following conditions:

$$Model = P(s', r \mid s, a)$$

$$Policy = \pi(s \mid a)$$

Stochastic Trajectory

Although, a **Markov Decision Process** is represented with two Perspectives such as Model and Policy, we can actually see it with another two Perspectives: Rewards (r) and Trajectories (τ).

$$P(\tau\,|\pi) = P(s_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

$$R(\tau) = \sum_{t=0}^{T} P(r_{t+1}|s_t, a_t)$$

From this we can see, Expected Future Value (Outcome) is

$$J(\tau\,|\pi) = \int P(\tau\,|\pi)\, R(\tau)$$

Hidden Markov Model

Actually, in view of Trajectory and Reward, a **Markov Decision Process** can provide us a deeper perspective with respect to Hidden Markov Model.

Stochastic Trajectory in **Markov Decision Process**

$$P(\tau\,|\pi) = P(s_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

Stochastic Distribution of Observable Variable given Latent (Hidden) Variable in **Hidden Markov Model**

$$P(x \mid h) = P(x_o)\prod_{t=0}^{T} P(x_{t+1} \mid h_t)P(h_t)$$

Language Model

Now, let us consider a Language Model, which is actually **Autoregressive Model** with Next Token Prediction.

A Language Model, especially a Pre-trained Model, can be considered as a **Variant of Hidden Markov Model**:

$$P(x) = P(x_0) \prod_{t=0}^{T} P(x_{t+1} \mid x_t) P(x_t)$$

From this, we can conclude that Language Model can generate a Stochastic Trajectory by predicting the next token, similar to the Stochastic Trajectory in Markov Decision Process.

$$P(\tau \mid \pi) = P(s_0) \prod_{t=0}^{T} P(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t)$$

Policy Learning by Language Model

Now, it raises a good question as to whether we can optimize the Stochastic Trajectory generated by Language Model as Policy Model, such that:

$$J(\tau\,|x) = \int P(\tau\,|x)\, R(\tau)$$

Actually, Large Language Model, with RLHF (Reinforced Learning with Human Feedback), already proved that it works perfectly well, as long as we can provide a good **Reward Model R(τ)**.

However, if we go a little bit further, ask a question as to whether it can directly learn the **Reward Model from the Environment**.

Feature Representation

A feature is a Latent Variable, defined in Latent (Feature) Space, which represents Differentiable Patterns, which can be measured and processed.

Transformation of Data (X) to Feature (F) is known as **Feature Map (ψ),** which can have different or same dimensions from Data.

$$F = \varphi(X)$$

The similarity of Features can be computed as **Dot Product** in **Inner Product Space**.

$$S = \varphi(X).\varphi(Y)$$

Representational Learning

Features can be extracted from a dataset, and learning Features of a Dataset is commonly known as Feature Learning or Representational Learning.

A set of Most Likely Features which represents the whole Dataset is:

$$F_{max} = \arg max \{\varphi_0 (X), \dots, \varphi_n (X)\}$$

However, a dataset is just a collection of samples, which defines **Sample Space (S)**, which is also a subset of Universal **Population Space (U)**.

So, even if we know the Features of Samples (Dataset), can we also know the Features of Population?

Random Sampling

If we are collecting random samples (dataset) from a population, such a set of random samples defines a Sample Space, commonly known as IID (Independently and Identically Distributed).

Interestingly, the distribution of any random samples is **Normal (Gaussian) Distribution**, regardless of the original distribution of the population from which the samples are taken. It is also known as **Central Limit Theorem (CLT)**.

Therefore, Central Limit Theorem gives us some glimpse of relationship between the randomly sampled Dataset and the whole Population.

$$\sqrt{n}\left(\bar{X}_n - \mu\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right)$$

Representational Learning From Samples

Central Limit Theorem suggests that it is possible to learn Most Likely Features of the Population from Random Samples, as long as we have enough samples.

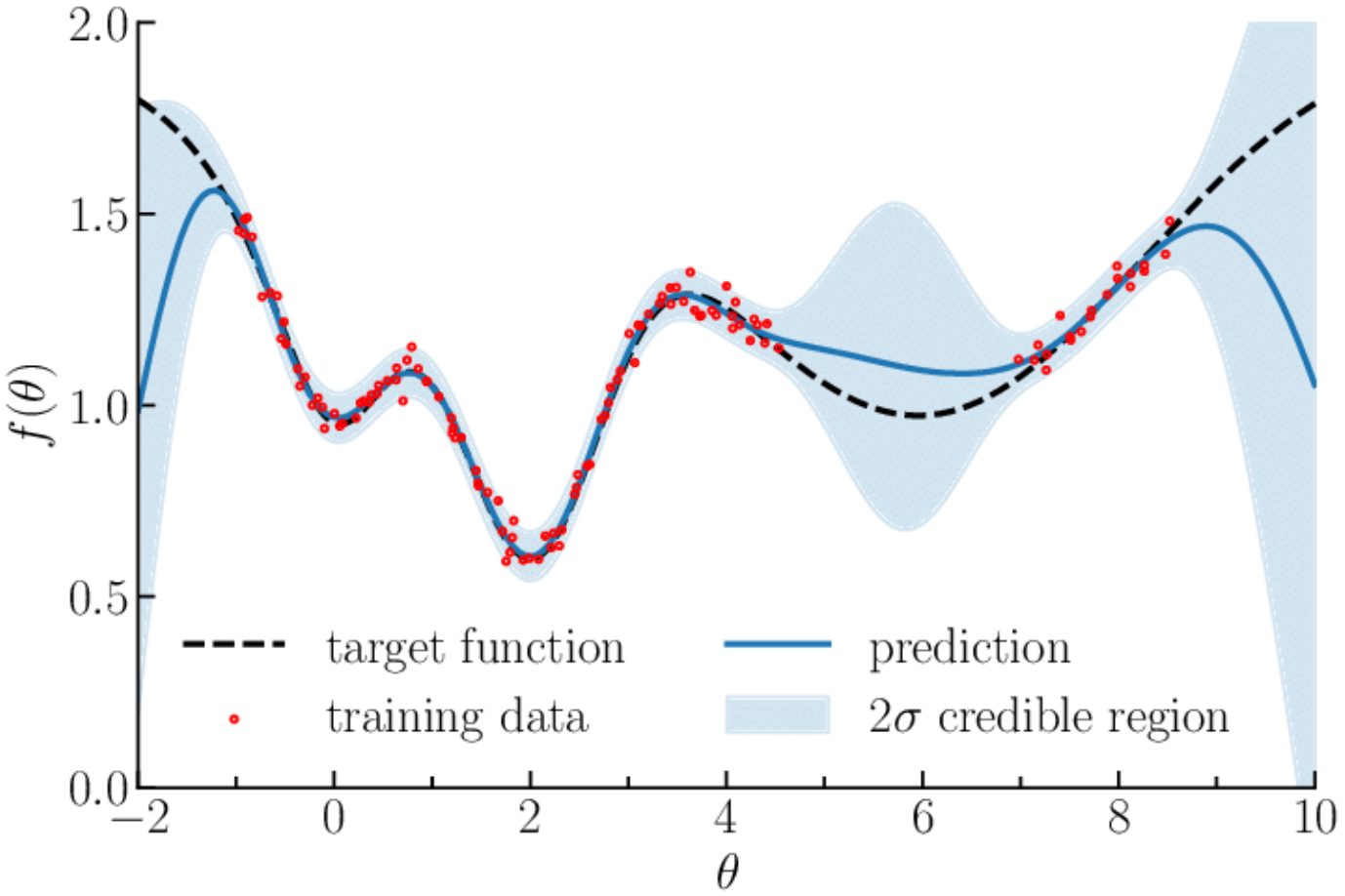
Generally, we optimize the **Maximum Likelihood** of Possible Features from **Random Samples**.

$$P(F \mid X) = \arg max \{\varphi_0 (X), \dots, \varphi_n (X)\}$$

Actually, the Stochastic Process of finding the maximum likelihood of features from random samples can be formally defined as **Gaussian Process**.

Gaussian Process

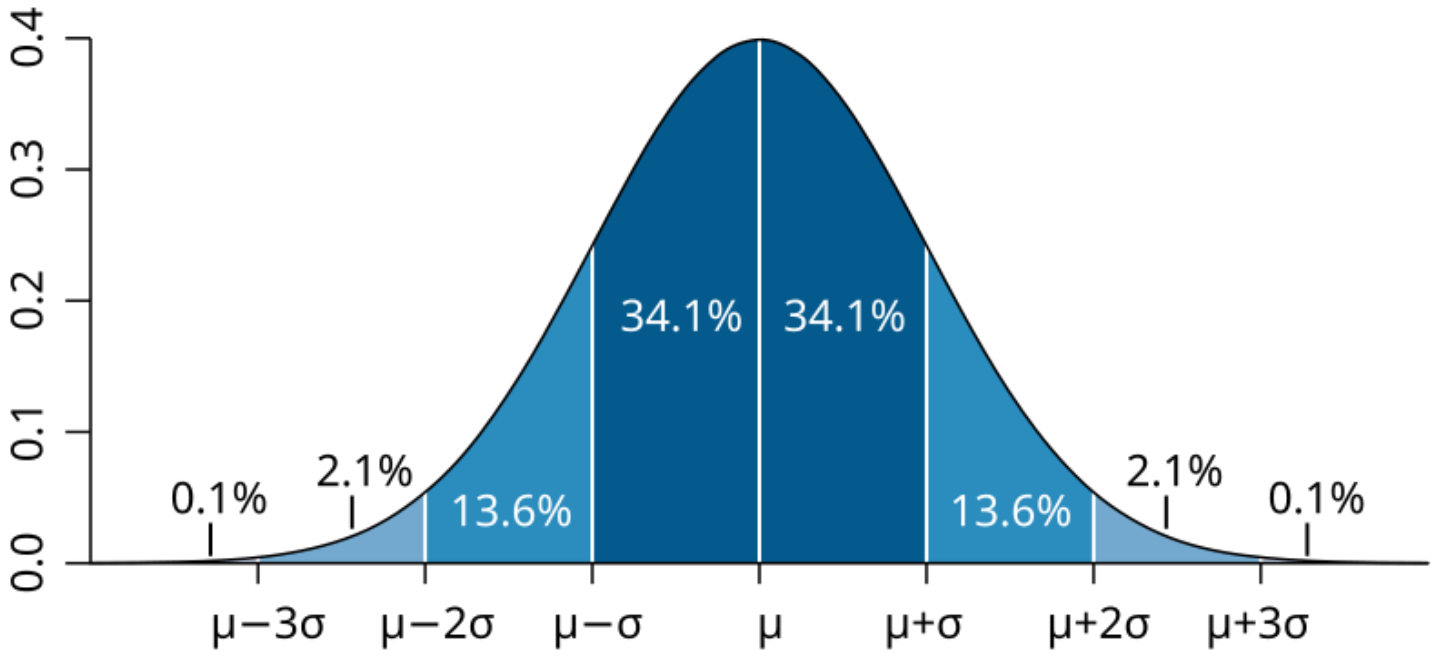
Gaussian Process is the stochastic process of finding the maximum likelihood of possible features (function) from a set of random samples, based on Gaussian Distribution.



Gaussian Distribution

Gaussian Distribution is one of **Exponential Family** of Probability Distribution, the same family as Boltzmann Distribution, which we will discuss later.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Variance and Covariance

In Gaussian Process, there are two important measures, namely Variance and Covariance.

Variance (σ^2) defines the Norm (Distance)of Individual Samples from the Mean (μ), and

$$Variance = |X - \mu|$$

Covariance defines the Norm (Distance) of individual samples from each other.

$$Covariance = |X - X'|$$

Therefore, Covariance is to measure how similar one sample is to another, while Variance measure how similar one sample is to average.

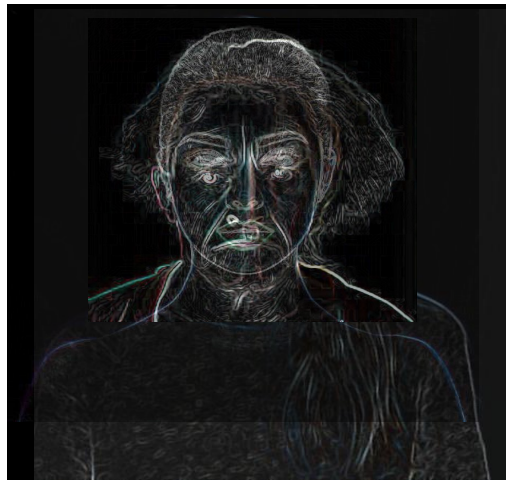
Gaussian Process

When we converge the Features of 3 random samples, we have:

Sample 1

Sample 2

Sample 3



Possible Features

Gaussian Process

After converging 4000 random sample photos of different Asian Ethnic Groups, we have:

Korean Burmese Cambodian Chinese Thai Samoan

** Ethnicity is to define how genetical features (genealogy) have common similarity within a certain group of people. Burmese, for example, are the mixture of Sino-Tibetan and Austro-Asiatic Genes, which are different from Cambodia and Thai.

Inner Product

Inner product (Dot Product) between two **Vectors** is defined as:

$$W.X = \sum_{i=0}^{N} w_i x_i$$

However if **W** represents a Vector of Probabilities, interestingly, it will give us an Expectation (μ):

$$P(x).X = \sum_{i=0}^{N} P(x_i) x_i$$

Softmax

Softmax is a **Normalized Exponential Function** which is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=0}^{N} e^{z_j}}$$

Interestingly, Softmax redefines **Z** onto a Probability Space (0, 1) by Normalization.

Projection Function

As we know, Inner product (Dot Product) between two **Vectors** is defined as:

$$W.X = \sum_{i=0}^{N} w_i x_i$$

However, depending on which **W** or **X** represents, we can have various interesting functions.

Thus, Inner Product defines a more generic projection function onto a Vector Space, known as **Inner Product Space**.

Attention Function

If **W** represents Softmax Scoring Function, with Query (q) and Key (k):

$$W = \alpha(q, k) = \frac{\alpha(q, k_i)}{\sum \alpha(q, k_j)}$$

Then, we will have Attention Projection (Pooling) function from the Inner Product, such that:

$$W.V = \alpha(q, k).v = \sum_{i=0}^{N} \alpha(q, k_i) v_i$$

Uncertainty

If **P(X)** represents a Probability, then what a **negative logarithm of probability** will represent?

Actually, a **Surprisal** is defined by the Inverse of Probability, such that:

$$Suprisal = \frac{1}{P(x)}$$

Thus, the Logarithm of Suprisal actually defines the **Amount of Randomness**:

$$Uncertainty = U(x) = -\log P(x) = log\left(\frac{1}{P(x)}\right)$$

Actually, it makes sense. When we say some **Probability**, it can also mean some **Uncertainty**.

Entropy

Then, again, the Expectation (Average) of Uncertainty is what defines an **Entropy**:

$$Entropy = H(x) = \sum_{i=0}^{N} P(x_i)[-\log(P(x_i))]$$

Actually, the Amount of Randomness is related to all Possible **States** (b). If we assume binary states, then:

$$Information = \sum_{i=0}^{N} P(x_i)\left[-\log_{\mathrm{b}}\left(P(x_i)\right)\right]$$

Then, Information could represent the Measure of Randomness in all possible binary states.

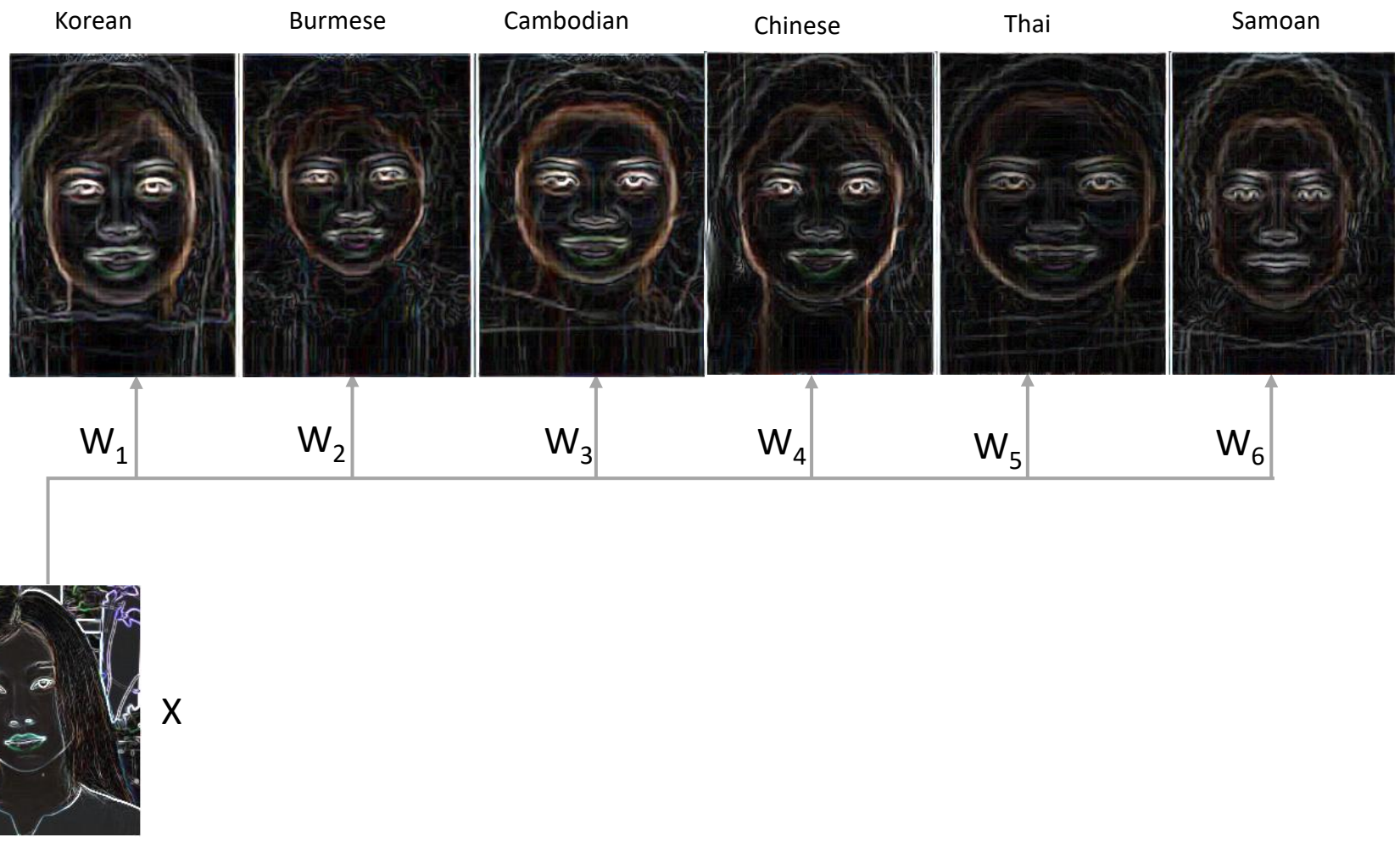
Variance of Uncertainty

If **Entropy** represents the Expectation of Uncertainty, what could possibly be the Variance of Uncertainty?

$$|U(x) - H(x)|$$

Will it tell us that Uncertainty will vary (deviate) differently across the Probability Distribution?

Prediction



What would be her ethnicity?

Prediction

The Dot Product between **X** and **W** defines the Similarity, such that

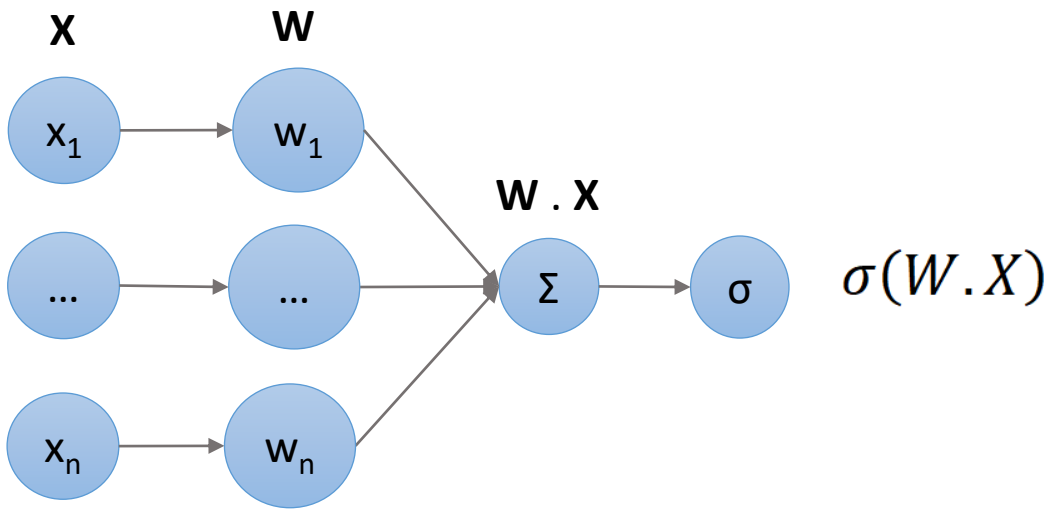
$$W.X = \sum_{i=0}^{N} w_i x_i$$

Then, **Softmax** of **W**.**X** represents the highest probable similarity between **X** and **W**:

$$\sigma(W.X)$$

Perceptron

Interestingly, a basic Unit of Neural Network, known as **Perceptron**, can be loosely defined as:



As we can see before, the **Probability** and the **Uncertainty** are inversely related: more probability means less uncertainty.

So, the question is what **W** really represents.

Kolmogorov Complexity

Kolmogorov Complexity defines a Complexity of a Computable String:

$$K(x) = min\{|\langle M, w\rangle|\}$$

A string is no longer compressible or computable if:

$$K(x) \geq |x|$$

What **Kolmogorov Complexity** really tells us is that **Computation** can be thought of as a **Compression** as it predicts the Uncertainty (Randomness) by Computation.

Because if we can predict the Randomness, it is no longer the Randomness; thus Uncertainty is reduced..

Therefore, Computation can be actually thought of as the **Minimization of Uncertainty**.

Neural Optimization

Traditionally, we assume that the Process of Neural Optimization (Deep Learning) as **Maximizing the Probability**.

However, what if we assume Neural Optimization actually is **Minimizing the Uncertainty**, such that:

$$Neural\ Optimization = argmin\{-logP(x)\}$$

Actually, **Minimizing the Uncertainty** makes more sense as it can define the **Lower Bound** on Minimum Uncertainty, as previously defined in **Kolmogorov Complexity**.

** Even though LLM can generate an Essay on Astronomy which includes thousands of characters, it is quite impossible for LLM to generate a sequence of winning lottery number which only involves 10 alphanumerical characters.

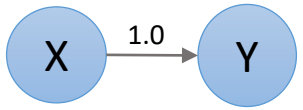
Minimizing Uncertainty

Minimizing Uncertainty also make sense if we look at from **Energy Perspective**.

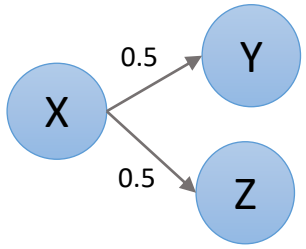
Energy is the ability to change the States of a System. Whenever energy is applied to a system, the states of the system change.

However, how these states change will be defined by the **Degree of Randomness**.

In other words, **Randomness** allows a **Variable Change** of Possible States.



X will definitely change to Y

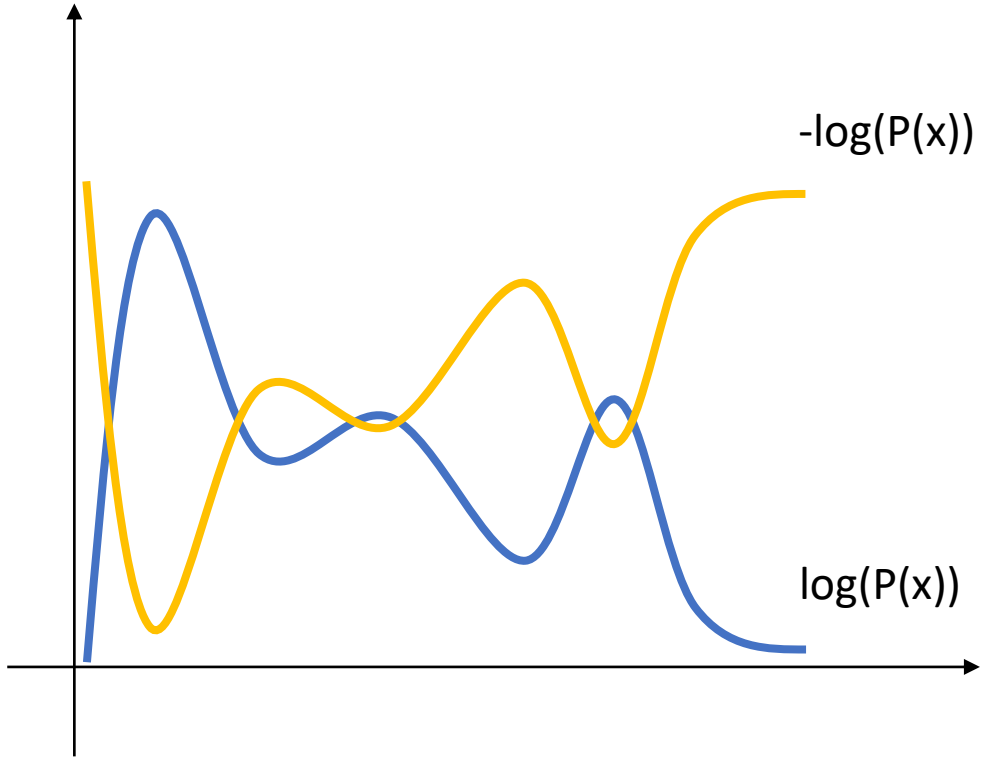


X will randomly change to Y or Z

** Randomness is highest when all possible states can vary equally likely, as prediction is impossible in that case.

Inverse Probability

As we already stated that –log(P(x)) represents the Inverse Probability, which defines Uncertainty.



Maximum and Minimum Entropy

Maximum Entropy defines the *Maximum* of **Average Uncertainty** or Highest **Average Randomness**.

$$max\ \{H(x)\} = argmax\ \left\{\sum_{i=0}^{N} P(x_i)[-\log\ (P(x_i))]\right\}$$

Inversely, Minimum Entropy defines the *Minimum* of **Average Uncertainty** or Lowest **Average Randomness**.

$$min\ \{H(x)\} = argmin\ \left\{\sum_{i=0}^{N} P(x_i)[-\log\ (P(x_i))]\right\}$$

Boltzmann Distribution

Boltzmann distribution (also known as **Gibbs distribution**) is a probability distribution in which the probability of the states of a system can be represented as a function of that **Energy** and the **Temperature** of the system.

$$P(x_i) = \frac{1}{Q}\, e^{\left(-\frac{\varepsilon_i}{kT}\right)}$$

Where **ε** = **Energy** and **T** = **Temperature**

$$Q = \sum_{i=0}^{M} e^{\left(-\frac{\varepsilon_j}{kT}\right)}$$

Energy Function

Energy Function $\mathbf{f_\theta}$ is based on a general form of Boltzmann Distribution, which is a subset of **Exponential Distribution**.

$$P_\theta(x) = \frac{1}{Z_\theta}\, e^{-f_\theta(x)}$$

If we take **Gradient** of $\log P_\theta(x)$, we get :

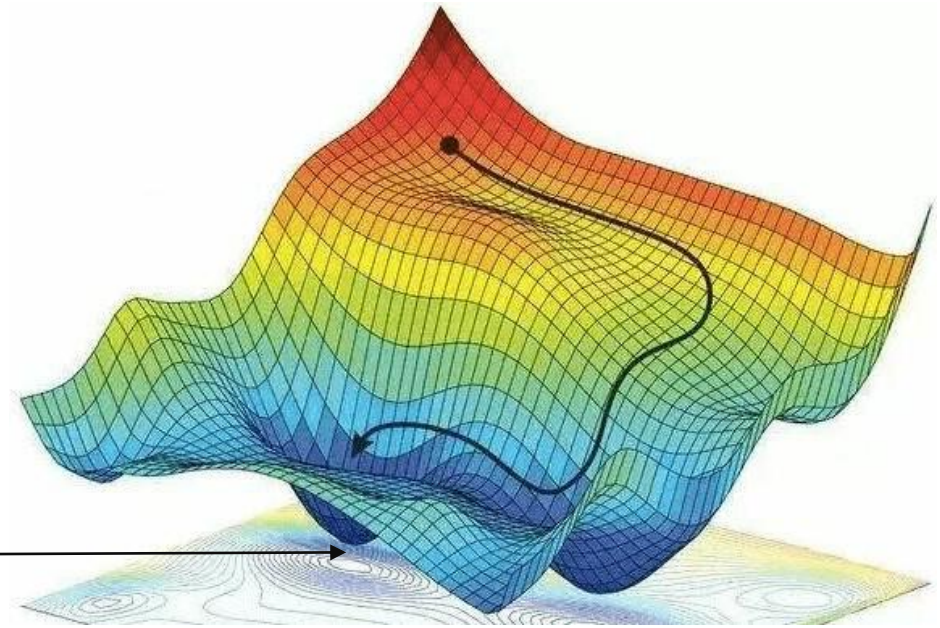
$$\begin{aligned}
\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}} \log\left(\frac{1}{Z_{\boldsymbol{\theta}}} e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})}\right) \\
&= \nabla_{\boldsymbol{x}} \log \frac{1}{Z_{\boldsymbol{\theta}}} + \nabla_{\boldsymbol{x}} \log e^{-f_{\boldsymbol{\theta}}(\boldsymbol{x})} \\
&= -\nabla_{\boldsymbol{x}} f_{\boldsymbol{\theta}}(\boldsymbol{x})
\end{aligned}$$

Energy Landscape

As we can see, the Gradient of Energy Function ($\mathbf{f_\theta}$) is:

$$\nabla_x \mathrm{f}_\theta(\mathrm{x}) = -\nabla_x log P_\theta(x)$$

Gradient of Energy Function defines the Gradient of Uncertainty (Negative Gradient of Probability) in Energy Landscape.



Neural Optimization By Minimization of Uncertainty

Therefore, we can conclude that Neural Optimization is equivalent to Minimizing Uncertainty in Energy Landscape.

$$\min\{f_\theta(x)\} = \operatorname{argmin}\{[-logP_\theta(x)]\}$$

However, what does Energy Function ($\mathbf{f_\theta}$) really represent? In other words, how do we define Energy Function $\mathbf{f_\theta}(x)$?

Kernel Function

A kernel function is a **similarity function** between two data points by taking **dot product** (inner product).

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle = \varphi(x).\varphi(x')$$

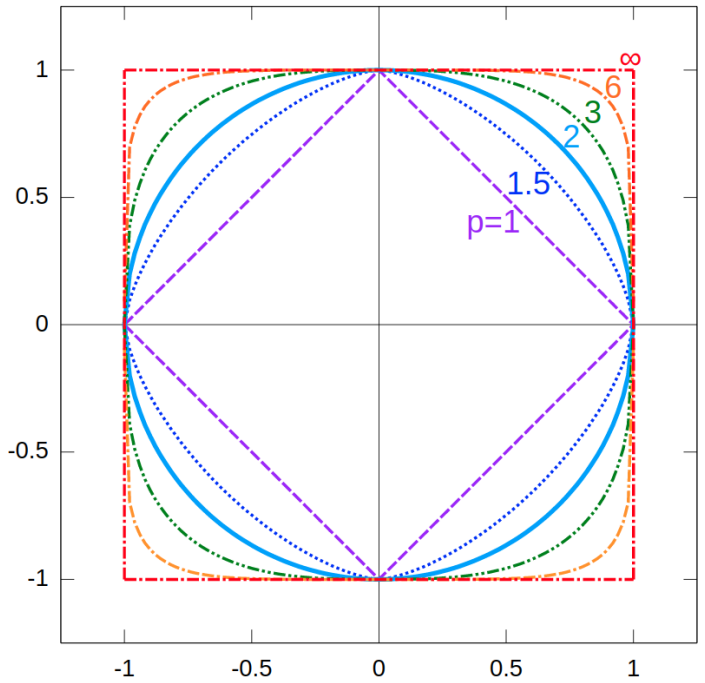
where $\varphi: X \to V$ is a feature map.

Interestingly, if **k**(x, x') is used in Gaussian Process, **k** represents Covariance Matrix **Σ**(x, x'), as Covariance measures **Norm Distance** between x and x', such that:

$$covariance = |x - x'|$$

Norm Distance

$$|x - x'|_p = \left(\sum_{i=0}^{N} |x_i - x'_i|^p \right)^{\frac{1}{p}}$$



Energy Function

Now, let us define the Energy Function **f**(x, x'; Θ) as a Kernel Function, such that:

$$\Phi(x, x'; \theta) = \nabla_\theta f(x; \theta). \nabla_\theta f(x'; \theta)$$

where $\nabla_\theta f(x; \theta) = \nabla_\theta[-log P_\theta(x)]$, which is the gradient of uncertainty.

Now, with this definition, the Energy Function **f**(x, x'; Θ) will define that the similarity between x and x' can be measured as uncertainty: the more similarity, the less uncertainty.

From the Energy Perspective in **Energy Based Model** (EBM), the Energy Function **f**(x, x'; Θ) is defined such that |x – x'| will have lower energy values if they are similar (closer).

Neural Tangent Kernel

Actually, such Kernel Function is formally known as Neural Tangent Kernel:

$$\Phi(x, x'; \theta) = \nabla_\theta f(x; \theta). \nabla_\theta f(x'; \theta)$$

Neural Tangent Kernel plays a vital role for general convergence in Neural Network, as proposed by Jacob et. al.

When we assume Neural Tangent Kernel as a Covariance Matrix **Σ**(x, x'), Neural Network becomes Gaussian Process, formerly known as *Neural Network Gaussian Process (NNGP).*

It has been proved that Neural Network with Infinite Width can converge to Global Minimum with enough data points.

However, as Neural Optimization is equivalent to Minimizing Uncertainty, **Kolmogorov Complexity** still defines the Lower Bound on Minimum Uncertainty.

Weng, Lilian. (Sep 2022). Some math behind neural tangent kernel. Lil'Log. https://lilianweng.github.io/posts/2022-09-08-ntk/.

Morphism

Now let us look into how **Knowledge Representation** can be interpreted by Kernel Function, such that:

$$\varphi(X).\varphi(Y)$$

Let us define a Tensor **X: V_1 x … x V_k** , where **V** is a N dimensional vector. Then,

$$\varphi(X) = \Phi$$

φ(X) defines a Homomorphism which maps from **X** to **ϕ**. Then, φ(X) is **Isomorphic** when it has an inverse, such that:

$$\varphi^{-1}\big(\varphi(X)\big) = X$$

Isomorphism

However, if φ(X) is an isomorphic map, then it can represent the relation of the Dimension of **X** to that of **ϕ**. In that case, there are three possibilities:

$$\varphi(X) \begin{cases} \dim\{\phi\} > \dim\{X\} \\ \dim\{\phi\} = \dim\{X\} \\ \dim\{\phi\} < \dim\{X\} \end{cases}$$

Again, if we define **φ(X)** with more familiar term such as "**Encoding**" of **X**, then $\boldsymbol{\varphi^{-1}(X)}$ can be defined as "**Decoding**" of **ϕ**.

However, if we interpret the isomorphic map with Entropy H(x), instead of Dimension of Space, what it can possibly represent?

Encoding

When we see φ(X) as an Encoding of **X**, then the Encoded Information of X will be represented as **ϕ**. Then, we can measure the Entropy of **X** and **ϕ**. Then, Entropy of **ϕ** can be:

$$\varphi(X) \begin{cases} \mathrm{H}(\phi) > \mathrm{H}(X) \\ \mathrm{H}(\phi) = \mathrm{H}(X) \\ \mathrm{H}(\phi) < \mathrm{H}(X) \end{cases}$$

However, what is significant that defines **H(ϕ) > H(X)**, meaning Entropy of **ϕ** is greater than Entropy of **X**?

$$\mathrm{H}(\phi) > \mathrm{H}(X)$$

In that case, it implies that additional information is added to **ϕ** by its Isomorphic Map **φ(X)**. If we put it in more familiar terms, **H(ϕ) > H(X)** implies "**Encryption**", in which more information is added to make it more difficult to decipher. Then, the inverse of φ(X), **φ^{-1}(X)** is the "**Decryption**".

As a rule of thumbs, Encryption will increase the Entropy (Information) of Original Tensor X.

Embedding

However, when we look at the special case in which Entropy of ϕ is less than Entropy of X:

$$H(\phi) < H(X)$$

In that case, it implies that additional information is removed in ϕ by its Isomorphic Map **φ(X)**, thereby reducing the Entropy of X.

Actually, reducing the Entropy of X, **H(X)** by its Isomorphic Map is really desirable, and in this special case, we can define the **ϕ** as the Latent (Feature) Space of X.

In addition, we can define a more general "**Abstraction**" by any **Isomorphic Map**.

Isomorphic Abstraction

If φ(X) is an Isomorphic Map, with its inverse $\varphi^{-1}(X)$, which maps from Tensor **X** to **ϕ**, such that:

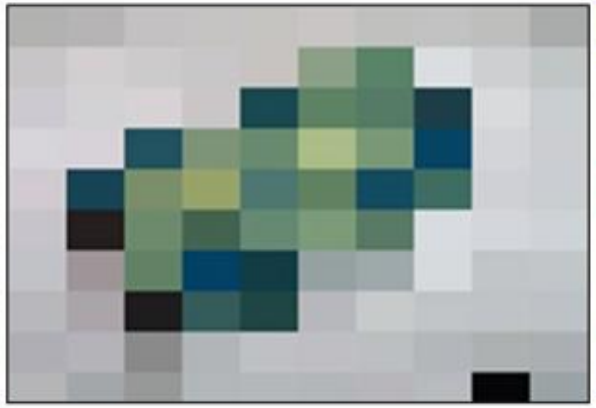
$$H(\phi) < H(X)$$

Then, we can define **ϕ** as the Isomorphic Abstraction of X by φ(X) in Latent Space, which implies the what is the minimum amount of information of X, to which φ(X) can be reduced.

32 bit Colors

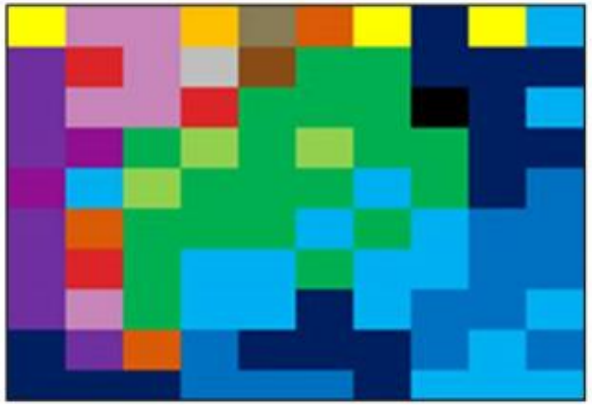
X

8 bit Colors



ϕ'

4 bit Colors

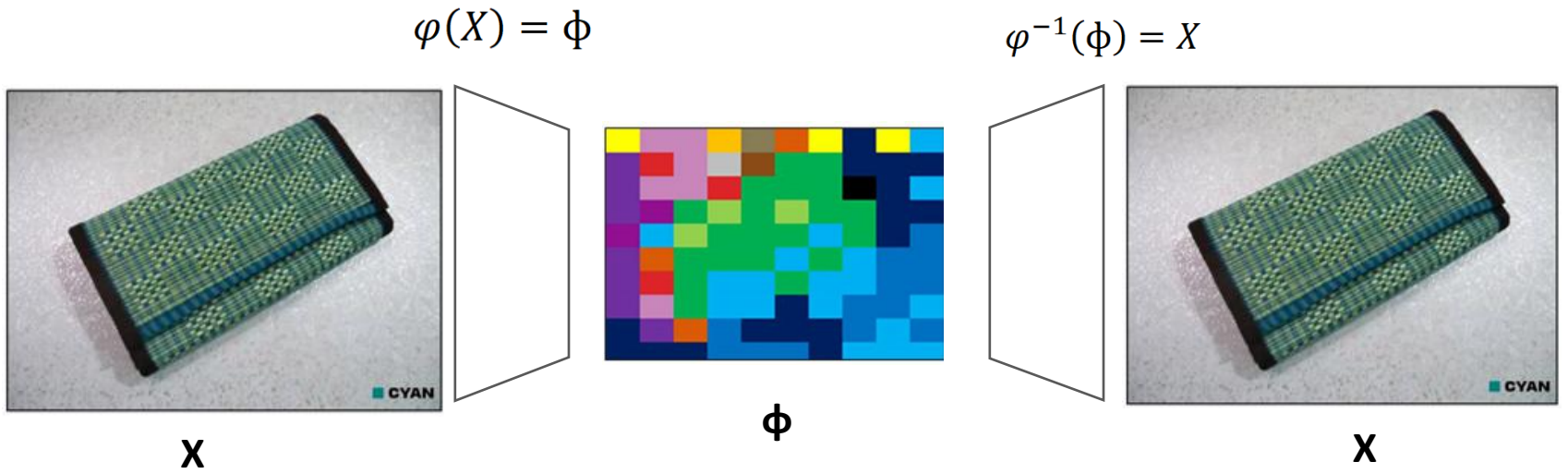


ϕ

If ϕ and ϕ' can transform back to X, then for all practical purpose X, ϕ, ϕ' are equivalent, although their latent (feature) spaces are different

Encoder-Decoder

Isomorphic Abstraction can be best understood with Encoder-Decoder.



** In a topology space, a Donut and a Tea Cup is said to be equivalent as they have equivalent **Homeomorphism**. Likewise, those two images have equivalent **Isomorphism**.

Joint Embedding

Actually, Isomorphic Abstraction still holds true even for multimodal Tensors $\{X_1, \ldots, X_m\}$, which can be jointly embedded with $\boldsymbol{\varphi(X_1, \ldots, X_m) = \phi}$ such that:

$$\mathrm{H}(\phi) < \mathrm{H}(X_1, \ldots, X_m)$$

From the perspective of **Kolmogorov Complexity**, **Isomorphic Abstraction** of $\boldsymbol{\varphi(X_1, \ldots, X_m)}$ represents the Reduction of Complexity (Entropy), thereby reducing its **Uncertainty**.

It makes sense because "**Abstraction**" could be interpreted as "**Summarization**".

Now, let us look into what the **Inner Product** between Isomorphic Abstraction really represent:

$$\varphi(X).\varphi(Y)$$

Inner Product Space

The inner product between Isomorphic Abstractions defines an Inner Product Space Ψ, such that:

$$\varphi(X).\varphi(Y) = \Psi$$

But if we project Ψ onto Norm Space defined by:

$$|\Psi|_p = \left(\varphi(X).\varphi(Y)\right)^{\frac{1}{p}}$$

Then, it defines the Distance between **X** and **Y** in Norm Space, and this distance measure the Similarity between X and Y.

However, what really is **Inner Product Space**?

Dot Product

If **V** is N Dimensional Vector and **U** is N Dimensional Vector, then Dot Product **u**.**v** can be defined as:

$$u.v = \sum_{i=1}^{N} u_i v_i$$

Therefore, the **result** of **Dot Product** represents the **Area of Intersection** between two vectors projected onto each other.

Then, Norm of Dot Product defines the Length of Area of Intersection, which is the Square Root of the Area:

$$|u.v|_2 = \sqrt{u.v}$$

Intersection of Space

Thus, if **X** is a Tensor **X : V_1 x … x V_k** with **N** Dimensional Vector **V** and **Y** is also a Tensor **Y : U_1 x … x U_k**, with **M** Dimensional Vector **U**, then Inner Product Space **Ψ** can be defined as **Tensor Contraction (Trace)**:

$$\Psi = \sum X_{i_1 \dots i_k} Y_{j_1 \dots j_k} v_{i_1} \otimes \cdots \otimes v_{i_k} \otimes u_{j_1} \otimes \cdots \otimes u_{j_k}$$

Then, the space defined by **Tensor Contraction** is the Inner Product Space, in which Latent Spaces of **X** with (N^k) Dimension and **Y** with (M^k) Dimension intersect. We can safely assume that the **result** of **Inner Product** represents the **Hyper Volume of Intersection**.

Then, it make senses that **Norm Space** represent a **P^{th} Root of that Volume** as a Distance (Length).

However, what does **Intersection of Latent Space** imply?

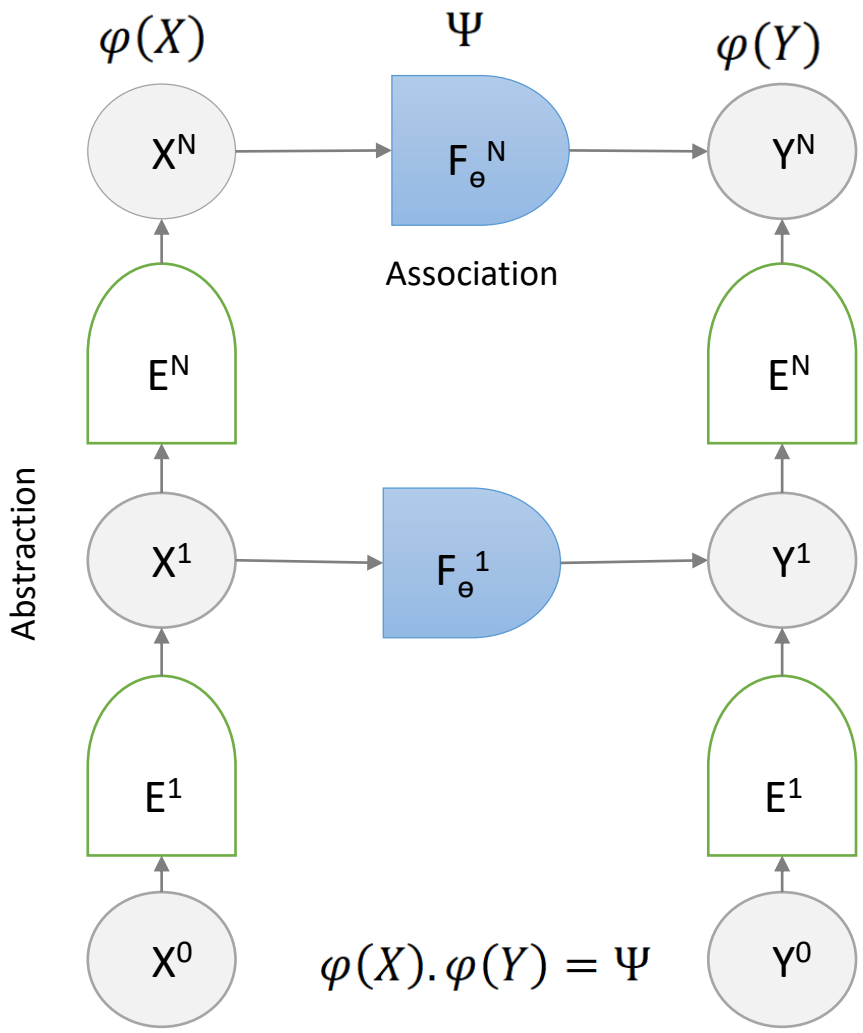
Knowledge Representation

In fact, Knowledge can be represented as **Association of Abstraction** which can be interpreted as the Intersection of Latent Spaces.

$$\varphi(X).\varphi(Y) = \Psi$$

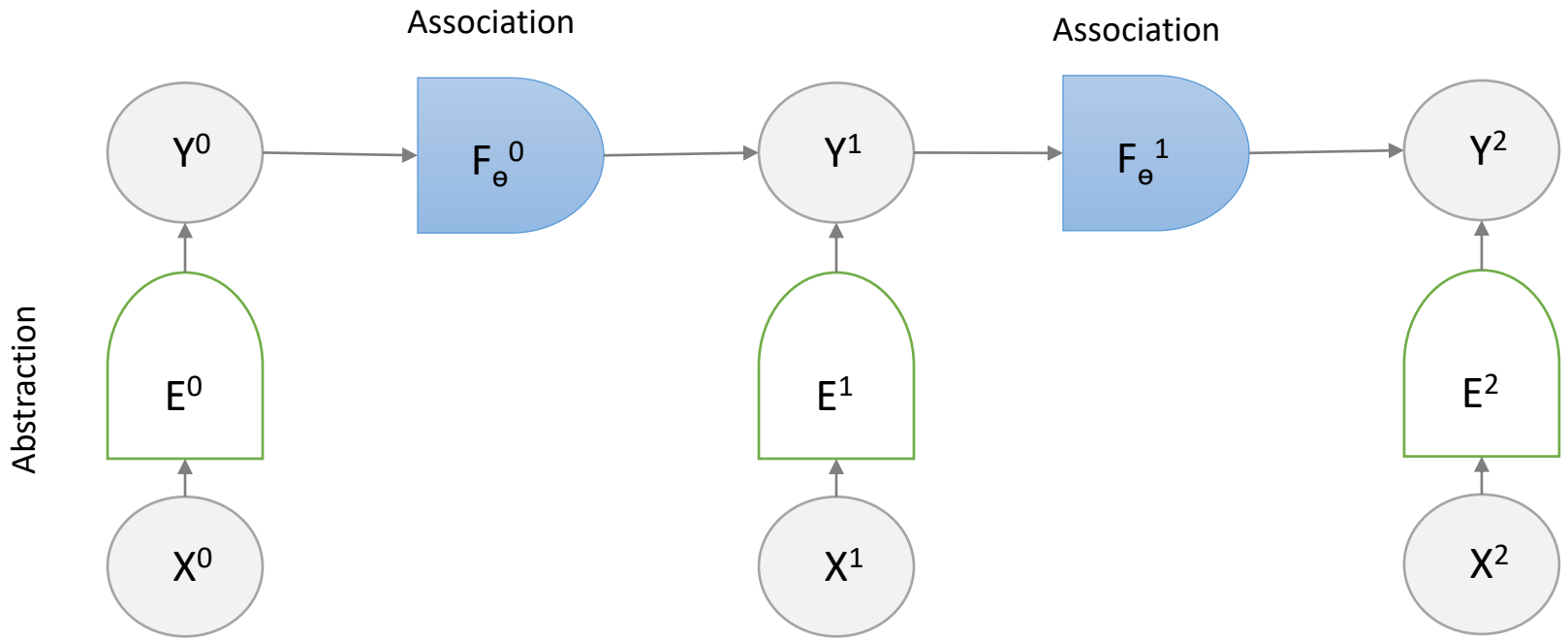
Hierarchical Associations of Abstractions can be used to represent **Knowledge Graph**, which represents how similar features coexist in Latent Space.

Knowledge Graph



Sequential Knowledge Graph

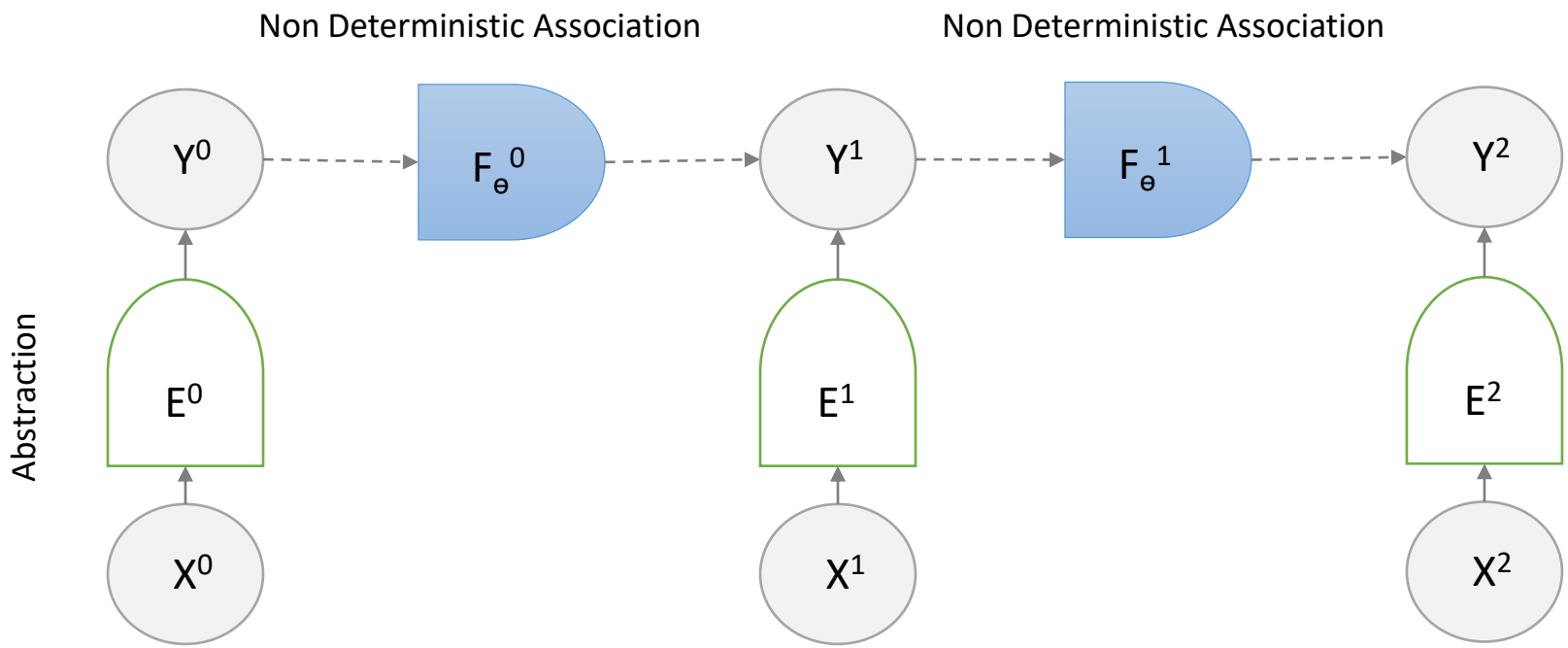
However, when a Knowledge Graph is dependent on Previous State, it can be defined a Sequential Knowledge Graph.



Generally speaking, Sequential Knowledge Graph can represent any Knowledge of Trajectory of Events, including Languages or Sequential Events.

Stochastic Sequential Knowledge Graph

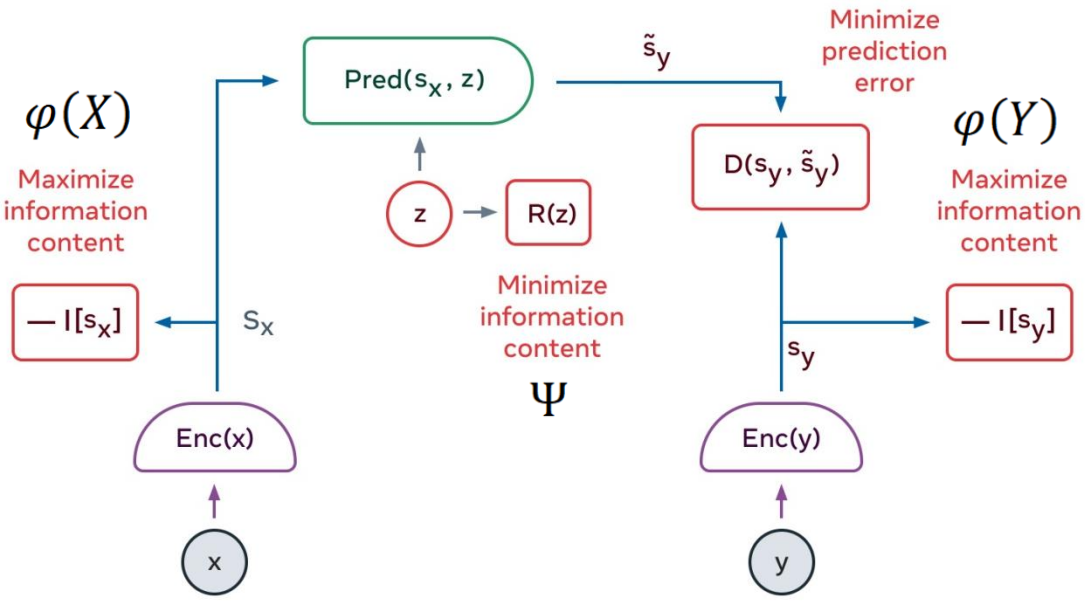
However, if Sequential Dependency of Sequential Knowledge Graph is stochastic or non-deterministic, then:



How can we understand Stochastic Sequential Knowledge Graph in Neural Network Architecture?

Joint Embedding Predictor

In early 2020s, **Dr. Yann Lecun** proposed Joint Embedding Predictive Architecture (JEPA) as a basic architecture for general intelligence.

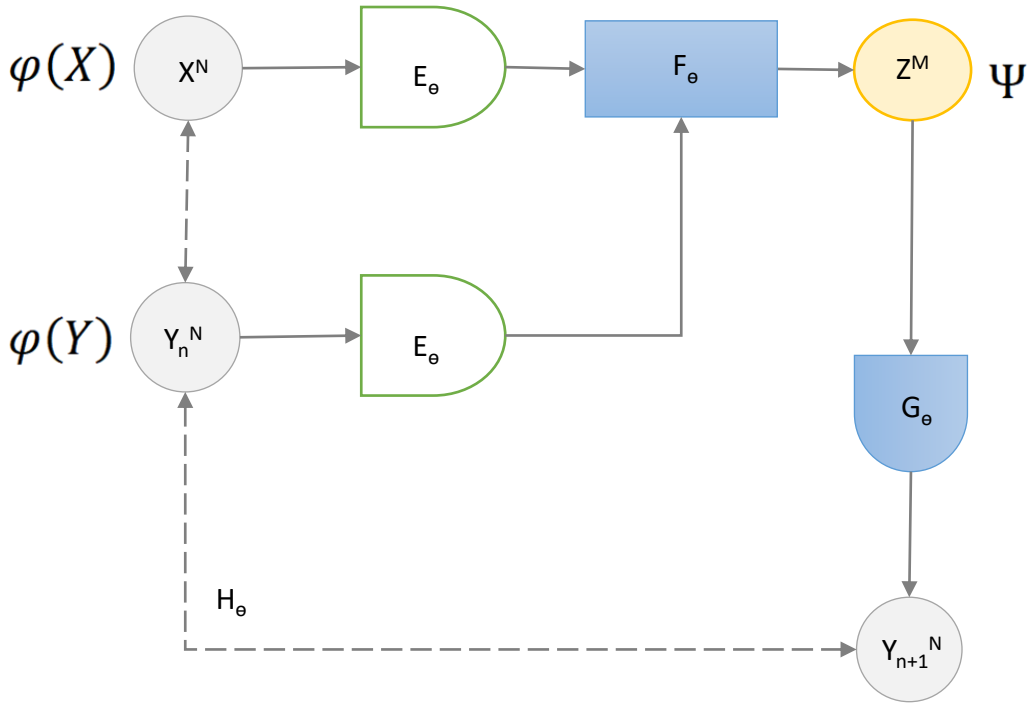


Courtesy of Dr. Yann Lecun: https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/

As we can see, Joint Embedding Predictor can generally represent Stochastic Sequential Knowledge Graph.

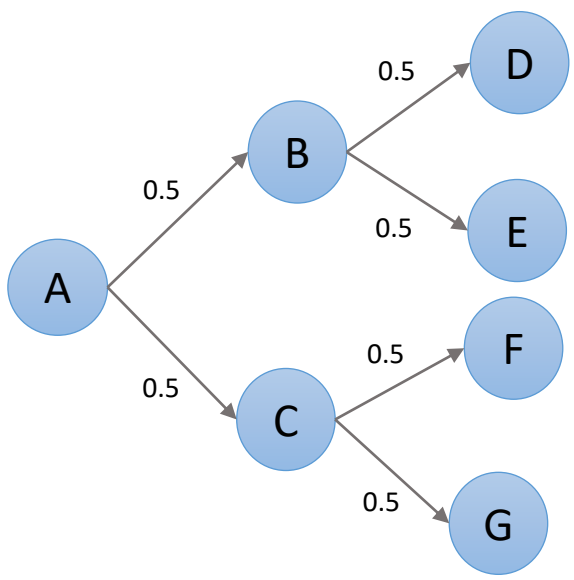
Transformer

Actually, Transformer Architecture can be assumed as a special case of JEPA, which can generally represent Sequential Knowledge Graph by learning Joint Embedded Association with **Cross Attention**.

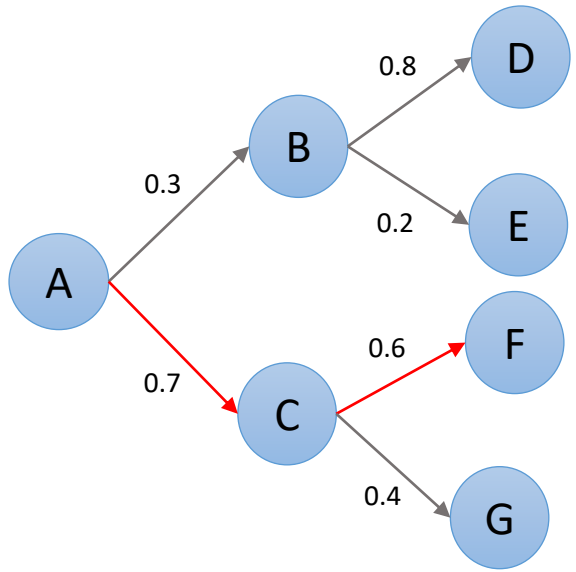


However, if we analyze Stochastic Sequential Knowledge Graph from the perspective of Kolmogrov Complexity, how do we understand it?

Lower Bound of Minimum Uncertainty



Uncompressible

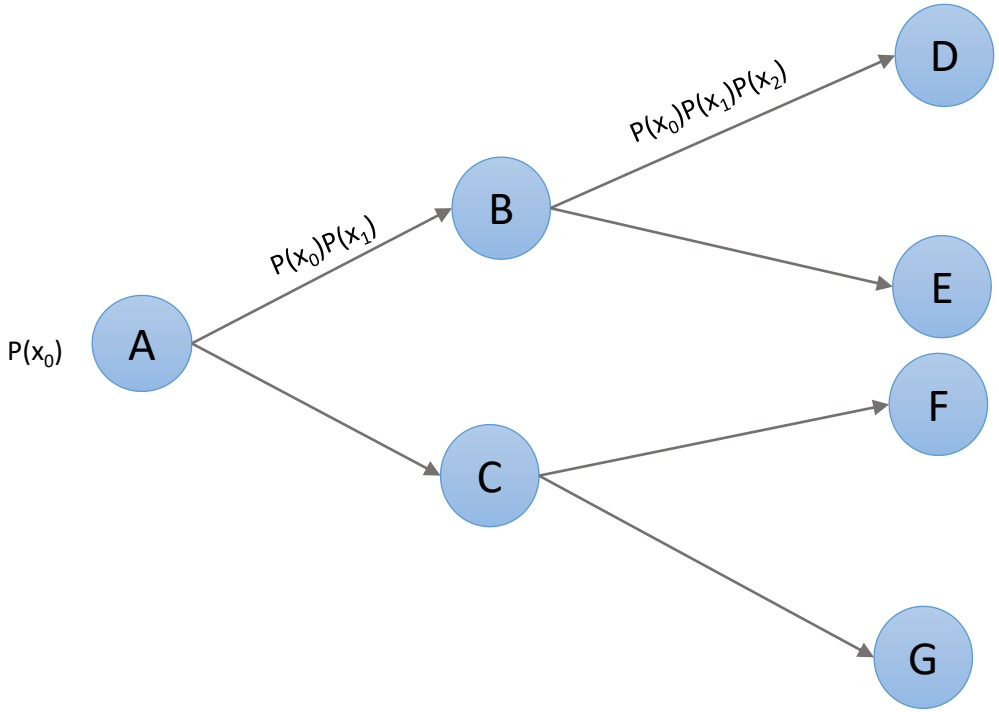


Compressible

As long as the chain of random events are not randomly uniformly distributed, it is possible to predict by minimizing the uncertainty. However, if the chain of random events are uniformly distributed, it is impossible to predict.

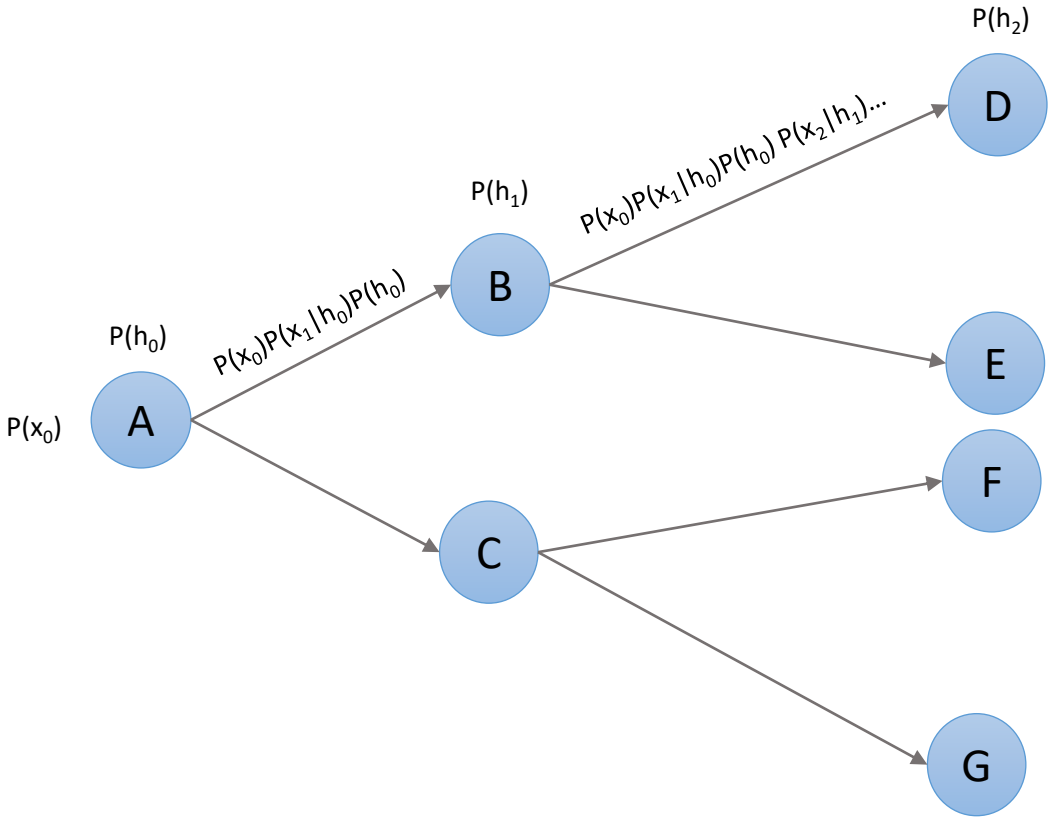
Actually, Neural Optimization is, in a sense, finding a **Trajectory of Minimum Uncertainty**.

Trajectory of Independent Events



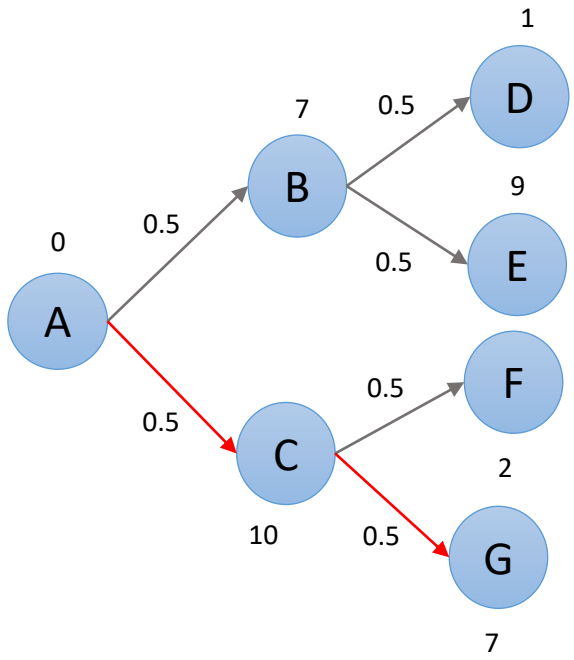
$$P(x) = \prod_{t=0}^{N} P(x_t)$$

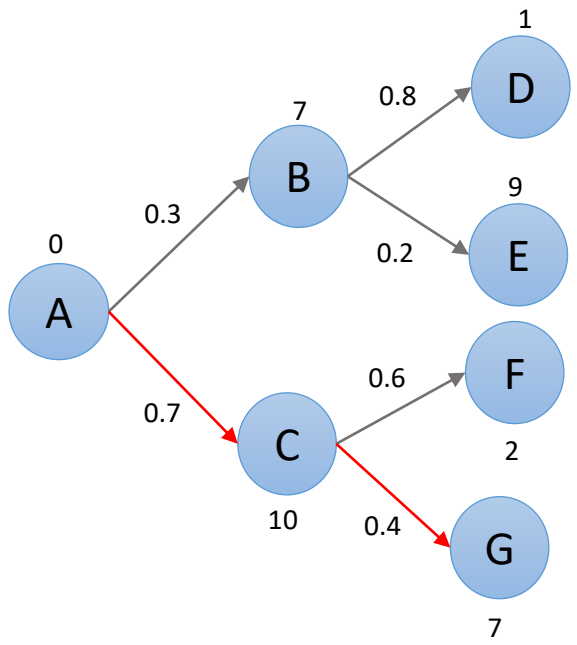
Trajectory of Latent Events



$$P(x|h) = P(x_0)\prod_{t=0}^{N} P(x_{t+1}|h_t)P(h_t)$$

Trajectory with Rewards

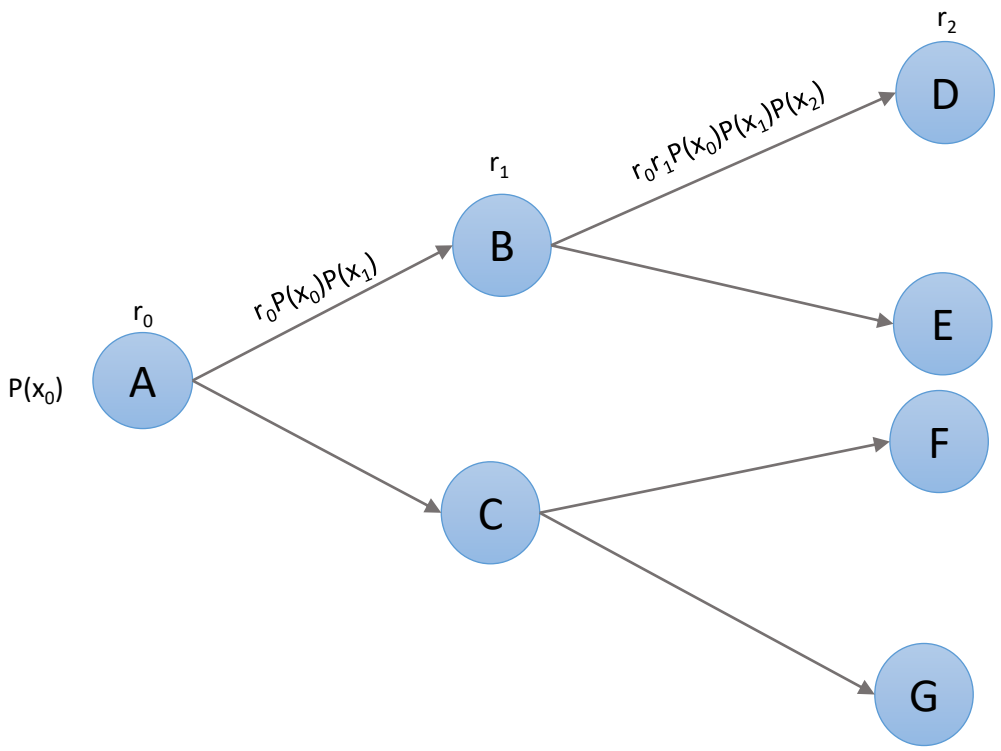




Interestingly, if we apply rewards to Trajectory of Minimum Uncertainty (1/P), even if the chain of random events are uniformly distributed, it is possible to predict based on maximum rewards, although predictions will be subjected to rewards.

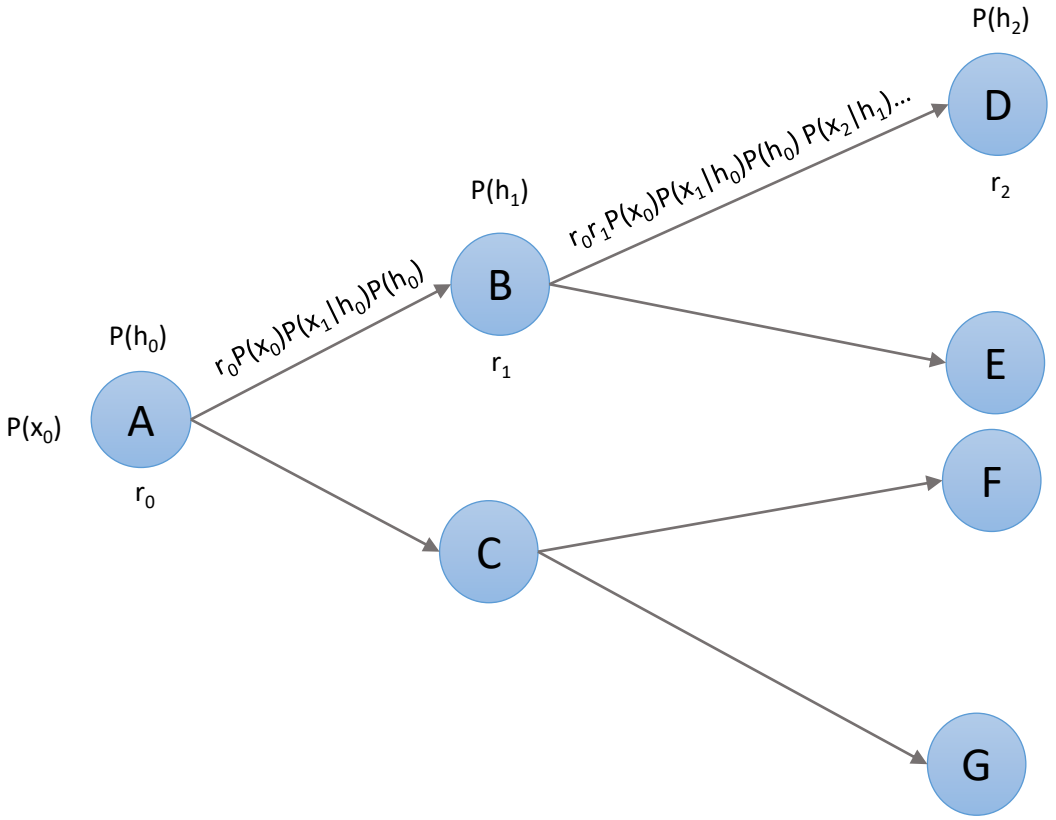
Therefore, we can conclude that rewards also play an vital role, in addition to uncertainty, to determine the Trajectory of Chain of Events.

Trajectory of Independent Events with Rewards



$$\tau(x) = \prod_{t=0}^{N} r_t P(x_t)$$

Trajectory of Latent Events with Rewards



$$\tau(x|h,r) = P(x_0)\prod_{t=0}^{N} r_t P(x_{t+1}|h_t)P(h_t)$$

Stochastic Trajectory with Rewards

As we can see, Stochastic Trajectory with Rewards can represented as :

$$\tau(x \mid h, r) = P(x_0) \prod_{t=0}^{T} r_t P(x_{t+1}|h_t) P(h_t)$$

We can also break it down into:

$$\tau(x \mid h) = P(x_0) \prod_{t=0}^{T} P(x_{t+1}|h_t) P(h_t)$$

$$R(\tau) = \sum_{t=0}^{T} P(r_{t+1}|x_t, h_t)$$

Latency Model and Policy Model

As we can see, when we assume P(h) as **Latent Variable**:

$$\tau(x \mid h, r) = P(x_0) \prod_{t=0}^{T} r_t P(x_{t+1} \mid h_t) P(h_t)$$

However, when we assume **P**(h) as **Policy Variable** such that **π**(a | h):

$$\tau(x \mid \pi, r) = P(x_0) \prod_{t=0}^{T} r_t P(x_{t+1} \mid a_t, h_t) \pi(a_t \mid h_t)$$

As we can see, when we assume P(h) as **Latent Variable**, it is Hidden Markov Model (HMM), and when we assume **P**(h) as **Policy Variable**, it is Markov Decision Process (MDP).

Policy Optimization

As **Policy Optimization** involves **Reward**, in addition to **Uncertainty**, Policy Optimization can be defined as:

$$\max\{J(\tau\,|\pi)\} = argmax\left\{\int P(\tau \mid \pi)R(\tau)\right\}$$

As we can see Policy Optimization is no longer a Minimization of Uncertainty; instead it is the Maximization of Reward, although Uncertainty still plays an important role.

We can define Policy Optimization as Maximization of Reward Under Uncertainty.

So, what really is **Reward**?

World Model

As we can see previously in **Markov Decision Process** (MDP), we can represent the **Dynamics of the Environment** (World) with the states of the Environment, along with their rewards.

Thus, we can conclude **World Model** can be represented by two aspects: **States** and **Rewards**.

States of the World can be represented by **State** (**Knowledge) Model**, while Rewards can be represented by **Reward Model**.

$$Model = P(\boldsymbol{s}', \boldsymbol{r} \mid s, a)$$

Thus, it begs for a question as to why we need Rewards along with the States of the Environment?

However, what really are rewards in the first place?

Reward

Generally speaking, a reward represents a **preferential score** (judgment) to measure the **usefulness** of the states of both the environment and the system.

A reward can be broadly categorized as **Intrinsic Reward** and **Extrinsic Reward**.

Intrinsic Reward implies that a reward can be defined internally within the systems, while **Extrinsic Reward** implies that it cannot be defined internally.

At any rate, the definition of reward is incomplete without looking into **Biological Reward Systems**.

Biological Reward

The **survival** of an living organism is highly dependent on the States of the **Environment**, and some states of the Environment are indeed **harmful** to its survival, though some states are **useful**.

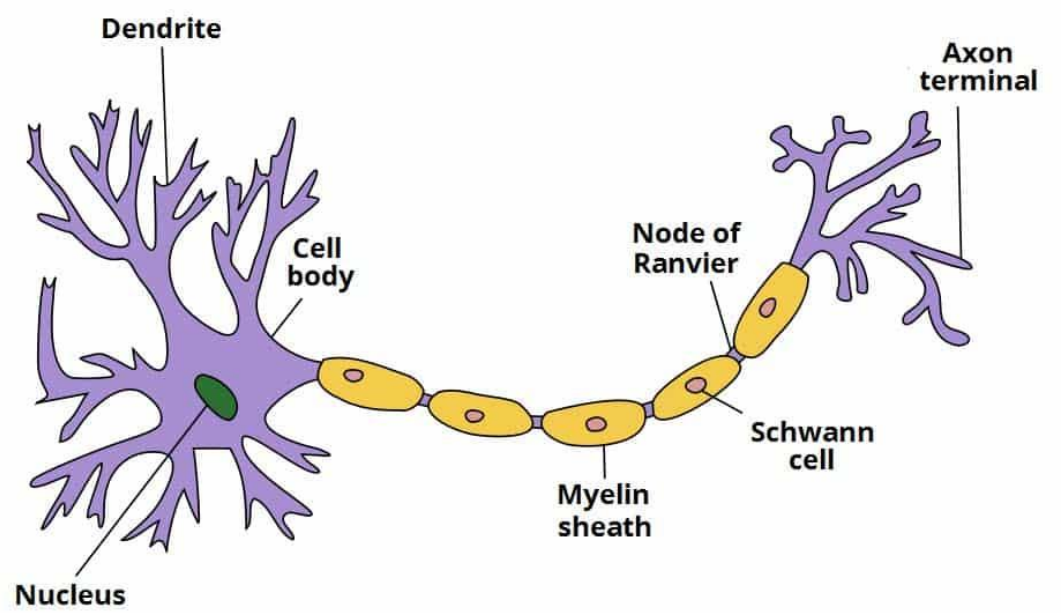
Thus, for its survival, an organism needs to define **preferential score** to measure the usefulness of the states of the environment in order to identify food, dangers, predators, mates and so on.

Therefore, most of biological rewards can be assumed as "**Intrinsic Rewards**" as it is defined internally by an living organism to measure the usefulness of the states for its survival.

Let us look into the basic building blocks of Biological Intelligence.

Neuron

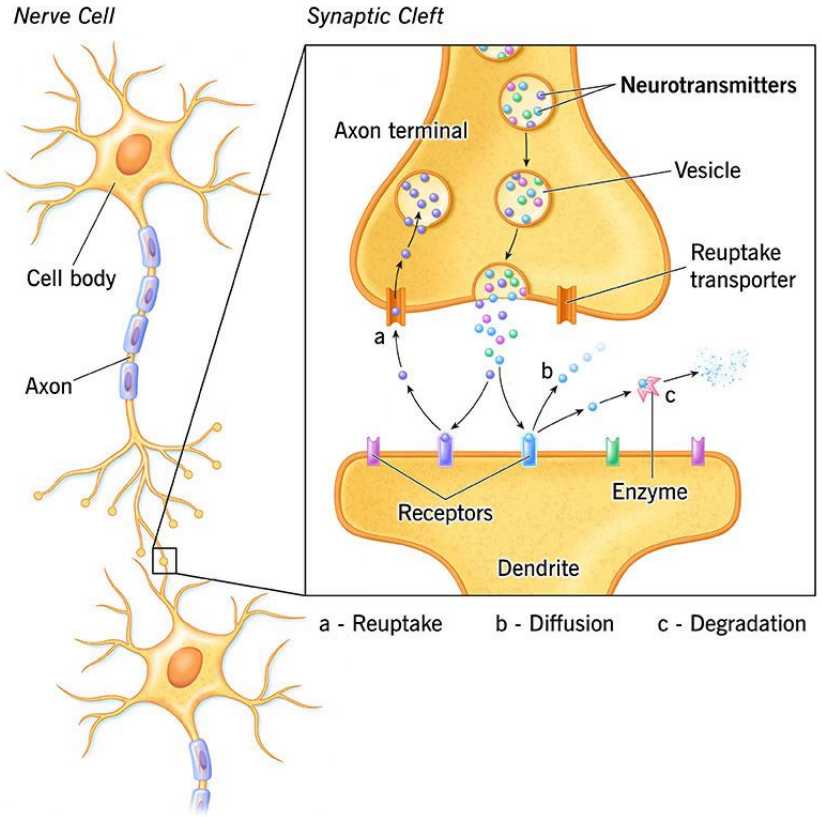
Let us look into a typical biological nerve cell, commonly known as Neuron, which is a building block of biological intelligence.



Actually, Artificial Neuron, known as **Perceptron**, which is the basic building block of Artificial Neural Network, is inspired by the process of biological neuron.

Neural Connections

Each neuron is connected by a special node called Synapse, which transmits, inhibits, or modulates chemical messages called Neural Transmitters. However, what is more interesting is neural transmitters also act as biological reward signals.



Courtesy of Cleveland Clinic

Neural Transmitters

Neurotransmitters are substances which neurons use to communicate with one another and with their target tissues in the process of **synaptic transmission** (neurotransmission). Their job is to carry chemical signals from one neuron to the next target cell. The next target cell can be another nerve cell, a muscle cell or a gland.

Neurotransmitters can transfer 3 types of actions:

Excitatory. Excitatory neurotransmitters **excite** the neuron, meaning the message continues to be passed along to the next cell. Examples of excitatory neurotransmitters include glutamate, epinephrine and norepinephrine.

Inhibitory. Inhibitory neurotransmitters **block** the chemical message from being passed along any farther. Gamma-aminobutyric acid (GABA), glycine and serotonin are examples of inhibitory neurotransmitters.

Modulatory. Modulatory neurotransmitters **influence** the effects of other chemical messengers. They adjust how cells communicate at the synapse. They also affect a larger number of neurons at the same time.

Neural Transmitters

Courtesy of Kenhub.com

Category	Neurotransmitter	Precursor Molecule(s)	Postsynaptic Effect	Location(s)	Type of Receptor(s)
Monoamines	Norepinephrine	Tyrosine	Excitatory	CNS: Brain and spinal cord PNS: ANS (sympathetic division)	Metabotropic
	Dopamine	Tyrosine	Excitatory and Inhibitory	CNS: Brain and spinal cord	Metabotropic
	Serotonin	Tryptophan	Excitatory and Inhibitory	CNS: Brain	Metabotropic
Amino Acids	**Glutamate**	Glutamine	Excitatory	CNS: Brain (major neurotransmitter of the brain)	Ionotropic, Metabotropic
	GABA	Glutamine	Inhibitory	CNS: Brain and spinal cord	Ionotropic, metabotropic
Neuropeptides	Substance P	Amino Acids	Excitatory and Inhibitory	CNS: Brain and spinal cord PNS: Enteric nervous system	Metabotropic
	Opioids	Amino Acids	Excitatory and Inhibitory	CNS: Brain and spinal cord (pain control)	Metabotropic
Other	Acetylcholine	Acetyl-CoA and choline	Excitatory	CNS: Brain and spinal cord PNS: Neuromuscular junction and ANS	Ionotropic, Metabotropic

CNS = Central Nervous System, ANS = Autonomic Nervous System, PNS = Peripheral Nervous System

Survival and Social Instincts

However, a from broader perspective, biological rewards can be grouped into survival rewards and social rewards.

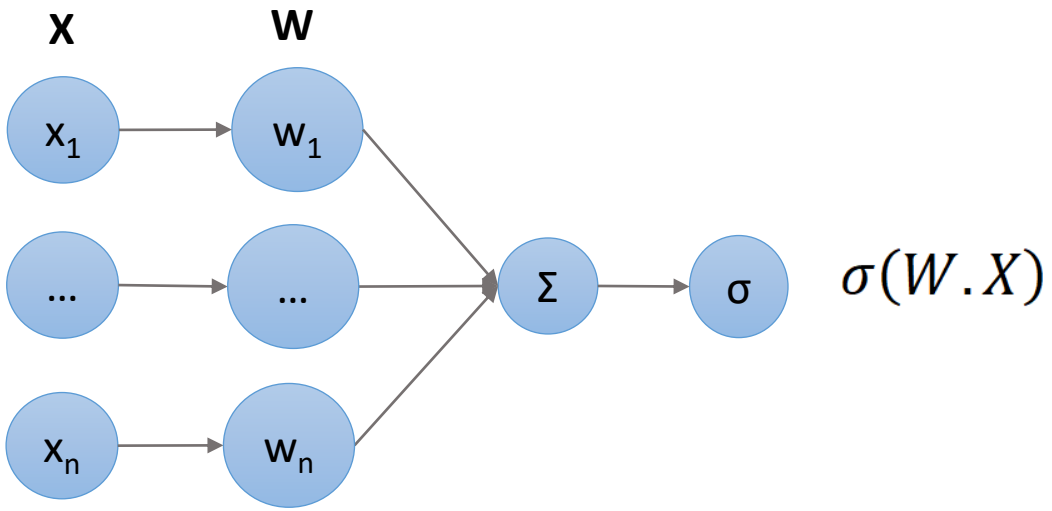
Survival Instincts measure the usefulness of the states of the environment in terms of survival rewards, concerning individual survival.

However, Social Instincts primarily measure the usefulness of the states the environment in terms of social rewards, concerning group survival.

Although further research is still necessary, it could not be wrong to assume that complex behaviors of intelligent beings, such as humans, are the results of integral behaviors, driven by Survival and Social Instincts.

Artificial Neuron

An artificial neuron, commonly known as Perceptron is defined mathematically as:



$$\sigma(W.X)$$

Unlike a biological neuron, an artificial neuron does not distribute its weights through Neural Transmitters. Instead, the distribution of weights in artificial neurons is defined by **Forward and Backward Propagation**.

Therefore, an artificial neuron does not have intrinsic reward systems. So, how do we define rewards in Artificial Neural Network (ANN)?

Artificial Reward

Artificial Rewards defined in Artificial Neural Network (ANN) are actually **Extrinsic** Rewards, unlike **Intrinsic** Biological Rewards.

$$J(\tau\,|x) = \int P(\tau\,|x)\, R(\tau)$$

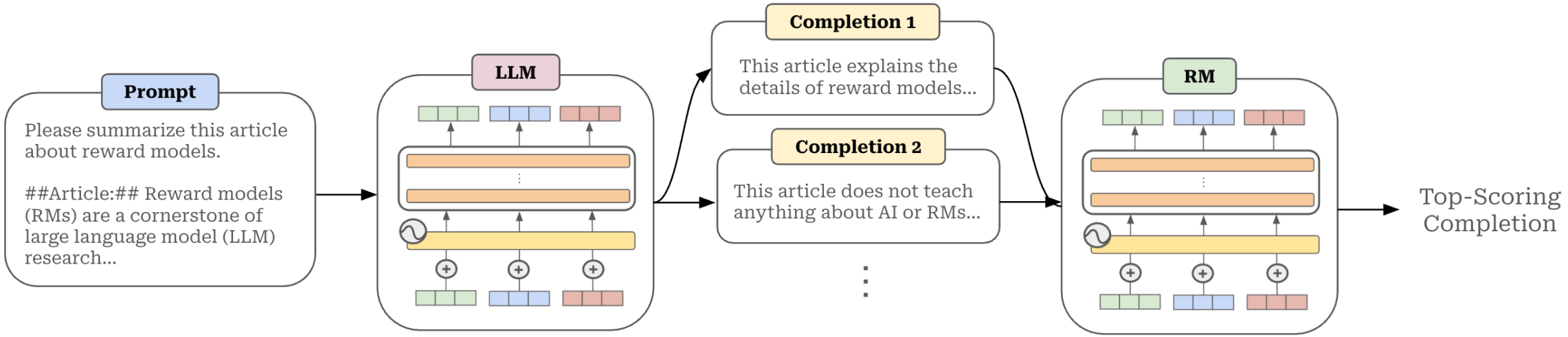
Therefore, Reward Model in the context Artificial Intelligence, is externally defined as a **Computational Model** to measure the usefulness of Internal and External States of the Systems.

For example, a reward model for a chess game will compute the reward for the winning states of the game.

Reward Model

In Artificial Intelligence, especially in Large Language Model (LLM) or other generative models, Reward Model (RM) is primarily used to evaluate (compute) the usefulness of the outputs of generative models.

For example, with Sentiment Analysis, the usefulness of the generated sentences or images can be evaluated as sentiment scores: positive, negative or neutral.



At any rate, without Reward Model, it would be difficult to determine if AI Generated contents, texts or images or others, are desirable or not.

Bradley–Terry Model

Bradley-Terry Model is a probability model for the outcome of **pairwise comparisons** between **items**. Given a pair of items i and *j,* drawn from some population, the probability of the pairwise comparison $i > j$ to be true is defined as:

$$\Pr(i > j) = \frac{p_i}{p_i + p_j}$$

We can also define the pairwise comparisons as Log Probability such that:

$$\log \frac{\Pr(i > j)}{1 - \Pr(i > j)} = \log \frac{\Pr(i > j)}{\Pr(j > i)} = \beta_i - \beta_j$$

Sensational Reward Model

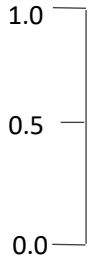
Generally speaking, Sensational Reward Model can interpreted as Bradley-Terry Model for a pairwise comparison, by measuring the pleasantness (preference) of the two images or two paragraphs or other two items.

A

Pleasantness: 0.13

B

Pleasantness: 0.81

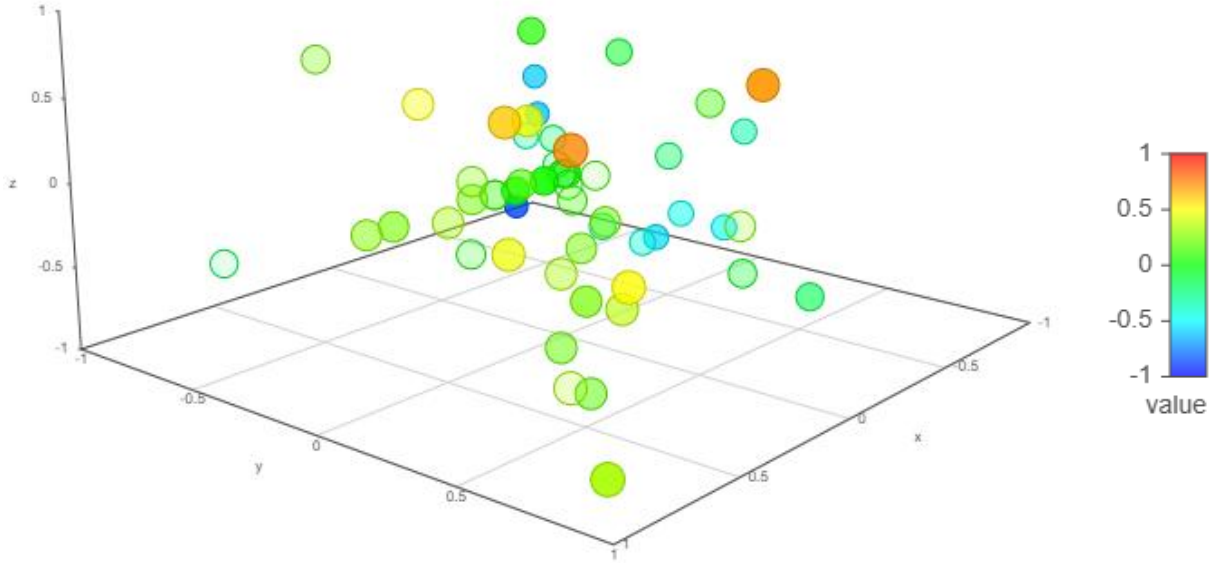


As we can see, the **vibe** of the images seem to align with our sentiments i.e., Image B is more preferable, but why?

Reward in Higher Dimensional Space

Actually, Artificial Rewards can be defined as **Scalars** in One Dimensional Space or as **Vectors** in Multi-Dimensional Space.

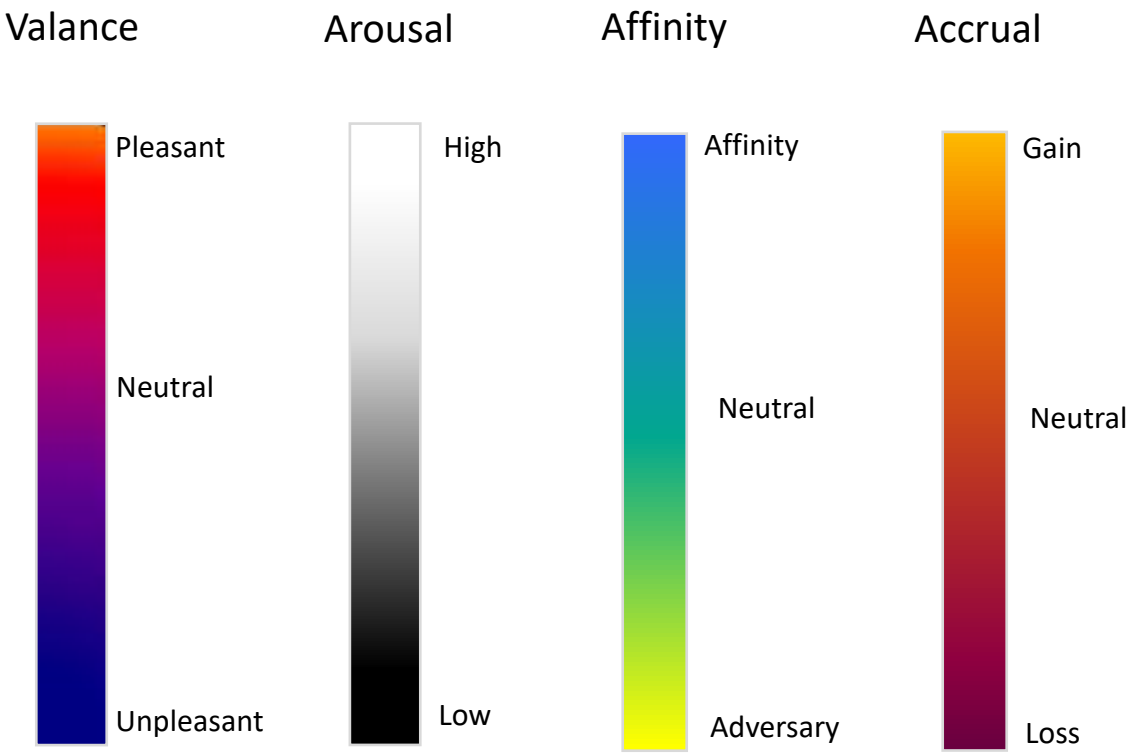
If we extend the idea of **Scalar Preferential Score** on pairwise comparison to **Vector Preferential Score**, we can define the Reward Model in Higher Dimensional Space.



Sentiment Space

To define Sensational Reward in Higher Dimensional Space, Additional Dimensions are necessary. Interestingly, according to Dimensional Emotion Theory, Human Emotions can be analyzed in terms of Dimensional Spectrum.

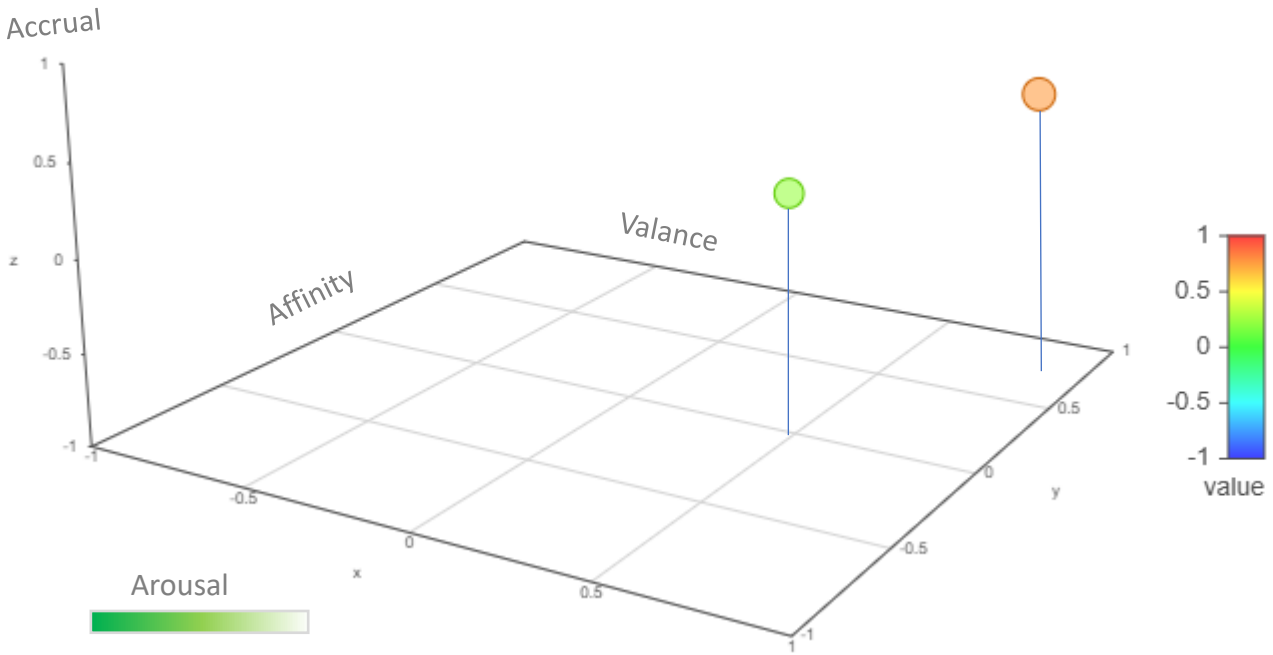
Therefore, I would like to propose **V3A** Space to compute Sensational Reward in 4 Dimensional Sentiment Space.



Sensational Reward in Higher Dimensional Space

The alignment of **Sensational Rewards** with **Human Sentiment** could possibly give us a mechanism to analyze and control the outputs of Generative AI, to be aligned with the values of humanity.

Given a contextual situation, AI will generate a sequence of actions or contents, which can be measured as sensational reward in higher dimensional space.



Understanding Reward in Preferential Latent Space

Mathematically, Reward can be formally defined as a Map **ϕ**: **T** → **v** in which **T** is a Tensor and **v** is a vector.

$$v = \phi(T)$$

If **v** is **Zero** dimensional vector, then Reward is represented by a Scalar; however, if **v** is **N** dimensional Vector, the Reward is represented by a Vector.

We can assume **ϕ** as a **Projection**, which reduces Higher Dimensional Tensor Space to Lower Dimensional Vector Space, called **Preferential Latent Space**, in which we can represent preferential score or reward.

For example, if we are to grade the essays written by students, with the score from 1 to 6, we actually reducing the higher dimensional space of Text Tokens (1500 words or 6000 Bytes) to the lower dimensional space of preferential score from 1 to 6 (4 Bytes).

Understanding Feature and Preferential Latent Space

Although we can both represent Feature Space and Preferential Space as a Map **M**: **X** → **Y** in which **X** is a tensor and **Y** is also tensor, there is a difference.

$$\varphi(X) = Y$$

Feature Map φ(X) is an **Isomorphic Map**, in which the inverse of φ, $\varphi^{-1}(Y)$ exists, such that:

$$\varphi^{-1}(\varphi(X)) = X$$

However, Preferential Map ϕ(X) is **not** necessarily an **Isomorphic Map**, i.e., the inverse of ϕ does not need to exist.

$$\phi(X) = Y$$

Difference between Feature and Preferential Latent Space

Feature Map $\varphi(X)$ represents "**Abstraction**", from which it is necessary to inverse back to the original, while Preferential Map $\phi(X)$ represents "**Preference**", from which it is not necessary to inverse.

It make sense.

"**Abstraction**" can be interpreted as "**Understanding**": when we understand something, we are able to explain it by "Summarizing" the original version, and we are also able to explain it by "Expanding" the summary back to the original version.

However, "**Preference**" can be interpreted as "**Favoring**": when we like something, we are not necessarily able to explain it.

"What is coffee?" is explainable. "Why do you like coffee?" is not explainable.

Global and Local Preferential Latent Space

Let us define a Tensor Space T : $\bigotimes^k U$, $U \in R^N$, and a Map $\boldsymbol{\phi} : \mathbf{T} \rightarrow \mathbf{v}$ where $\mathbf{v} \in R^M$:

$$v = \phi(T)$$

If $\boldsymbol{\phi}$ project the **entire space** of **T** to **v**, then **v** will represent Global Preferential Latent Space of T.

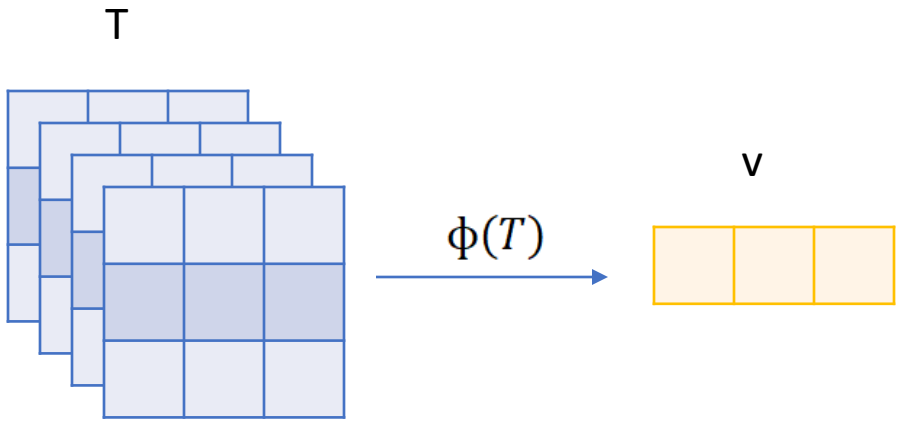
However, if $\boldsymbol{\phi}$ projects the **partial space** of **T**, defined by Tensor Contraction $\mathbf{C_{i,j}(T)}$, then **v** will represent the Local Preferential Space of T.

It makes sense.

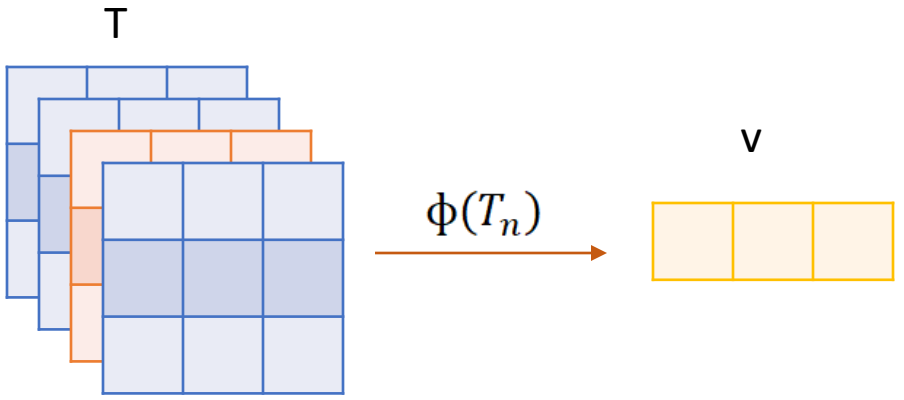
If we watch a movie, with less than 30% Rotten Tomatoes ratings, it implies the overall movie is bad. However, there could some scenes we like, which are particularly good, despite the entire movie being bad.

Global and Local Projection

Global Projection of Preferential Latent Space will map the Entire Space of T onto v:



Local Projection of Preferential Latent Space will map the Partial Space of T onto V:



Segmentation of Global to Local Projection

Semantic Segmentation, also know as Image Segmentation, can be easily understood with Segmentation of Global Projection. We can define $\boldsymbol{\phi(T_n)}$ as the **Partition** of $\boldsymbol{\phi(T)}$; different partitions can have their own local preferential spaces, i.e., we will feel differently across different regions of the same image.

ϕ(T) can project the entire Space of T, or it can project segmented Space of T.

Autonomous Intelligence

The Holy Grail of Artificial Intelligence, I presume, is the Autonomous Intelligence. To be more specific, Autonomous Intelligence can be broadly defined as:

"The Intelligence , from **Perception to Action,** to make **Independent** decisions under Uncertainty and Scarcity."

However, there are many obstacles to achieve such Intelligence artificially, even if we assume such Intelligence is achievable in the first place.

Nonetheless, it would serve us as an **Asymptotic Boundary** for Artificial Intelligence Development.

However, let us look into what it means by "Autonomy".

Autonomy and Agency

“Autonomy implies self-determination and the freedom to make choices and act according to one's own values and goals. It's about having control over one's destiny and being able to act in accordance with one's own desires and beliefs."

“Agency is the capacity to act and take initiative, even when acting under the guidance or within the goals of others. It's about the ability to perform actions and make choices, regardless of whether those choices are autonomous or influenced by external factors.”

From those, we can at least conclude that **Artificial Autonomy** is already a **Paradox** itself; as Autonomy cannot be Artificial.

That’s why, I presume, “Agentic Intelligence” seems more plausible.

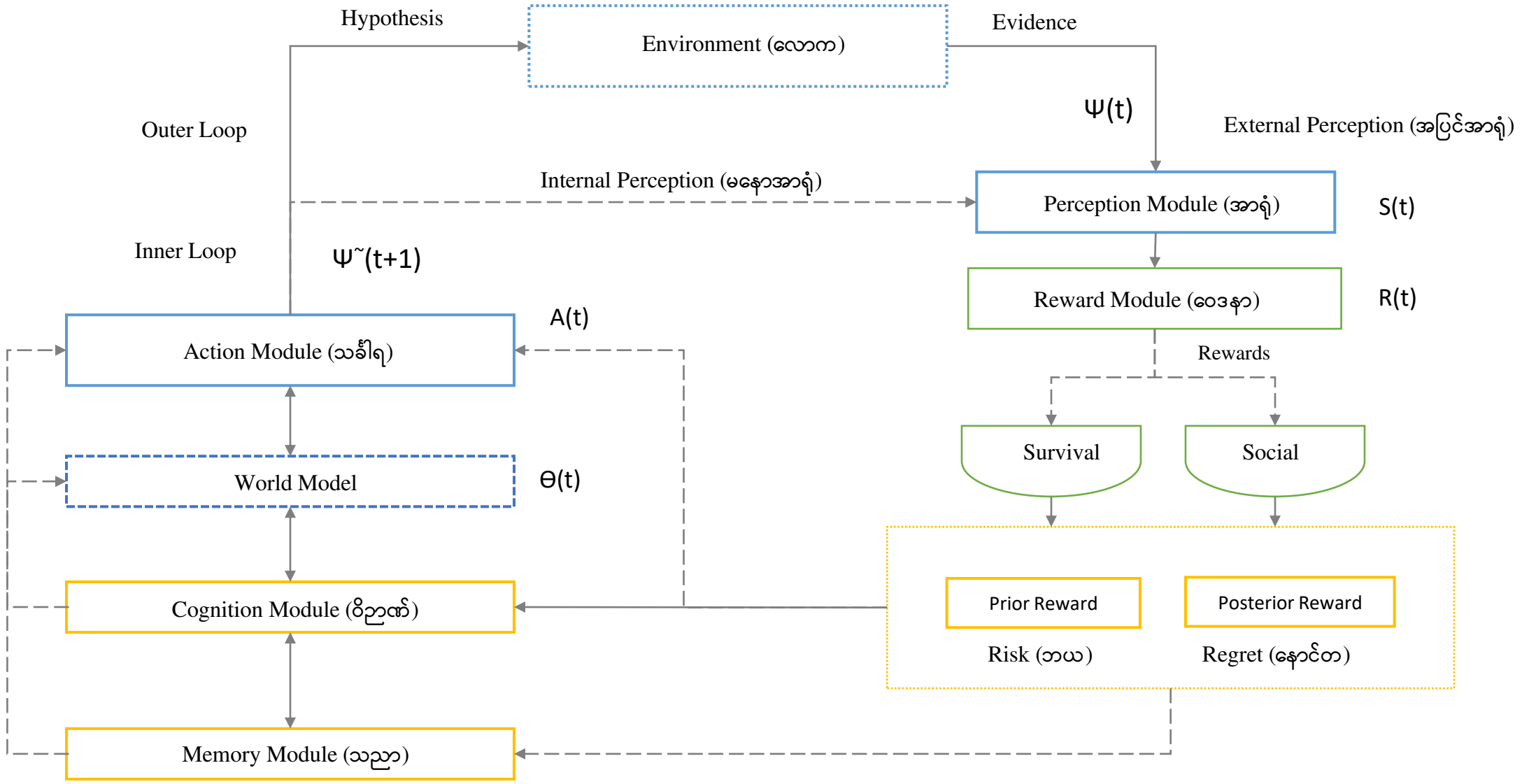
** Please do note that random errors of Artificial Intelligence is not because of its autonomous nature but because of its stochastic nature.

Agent

An Agent can be, therefore, defined as a **Decision Maker** (DM), which can take appropriate actions within the boundary of the guidance or the goal of others, under the Uncertainty and Scarcity of the states of the Environment.

Let us look into the Hypothetical Modules of Artificial Intelligence Agent, from Perception to Action, in order to achieve the "**Agentic Intelligence**", as we have already discussed most of the fundamental concepts.

Modules


Hypothesis
Environment (လောက)
Evidence
Outer Loop
Ψ(t)
External Perception (အပြင်အာရုံ)
Internal Perception (မနောအာရုံ)
Perception Module (အာရုံ)
S(t)
Inner Loop
Ψ˜(t+1)
Reward Module (ဝေဒနာ)
R(t)
A(t)
Action Module (သခ်ၤရ)
Rewards
Survival
Social
World Model
Θ(t)
Prior Reward
Posterior Reward
Cognition Module (ဝိညာဏ်)
Risk (ဘယ)
Regret (နောင်တ)
Memory Module (သညာ)