Artificial Intelligence 2: Maximum Likelihood (I)

Shan He

School for Computer Science
University of Birmingham

Outline of Topics

Balls in urns

Example: Suppose we have an urn that contains only black and white balls, but we have no idea about the proportion of each. You pull a number of balls with replacement and get the sequence D: BBWBWWBWWWBW.

Question: what is the proportion of black balls that is most likely to draw that sequence?

Formulating the urn problem

Denote the proportion of black ball, or the probability of drawing a black ball as θ, we know the probability follows the Bernoulli distribution: $P(B) = \theta$ and $P(W) = 1 - \theta$ for a white ball.

Since each draw is I.I.D, we calculate the probability of getting that sequence D as the product of the probabilities of each of the individual draws, which is equivalent to the :

$$L(\theta|D) = \prod_{i=1}^{N} \begin{cases} \theta & \text{if } d_i = B \\ 1 - \theta & \text{if } d_i = W \end{cases}$$

This probability is called **likelihood** function L, which is a function of θ, given the samples (observation) D.

Likelihood: The probability of observing your data given a particular model, i.e., the Bernoulli distribution. The question is how to find θ.

Likelihood function: formal definition

Let $X_1, X_2, X_3, ..., X_n$, denoted compactly using a vector $\boldsymbol{X} = (X_1, X_2, X_3, ..., X_n)$ be a random sample from a distribution with a parameter θ. Suppose we have observed that $X_1 = x_1, X_2 = x_2, X_3 = x_3, ..., X_n = x_n$, denote a vector $\boldsymbol{x} = (x_1, x_2, x_3, ..., x_n)$, we can define the likelihood function as

1. If X_i's are discrete:

$$L(\theta|\boldsymbol{x}) = L(\theta|x_1, x_2, \ldots, x_n) \quad (1)$$
$$= P_{\boldsymbol{X}}(\boldsymbol{x}; \theta), \quad (2)$$

   where $P_{\boldsymbol{X}}(\boldsymbol{x}; \theta)$ is the PMF of $\boldsymbol{X}$ parametrised by θ

2. If X_i's are continuous:

$$L(\theta|x) = L(\theta|x_1, x_2, \ldots, x_n) \quad (3)$$
$$= f_{\boldsymbol{X}}(\theta|\mathbf{x}) \quad (4)$$

Note: In general, θ can be a vector, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$.

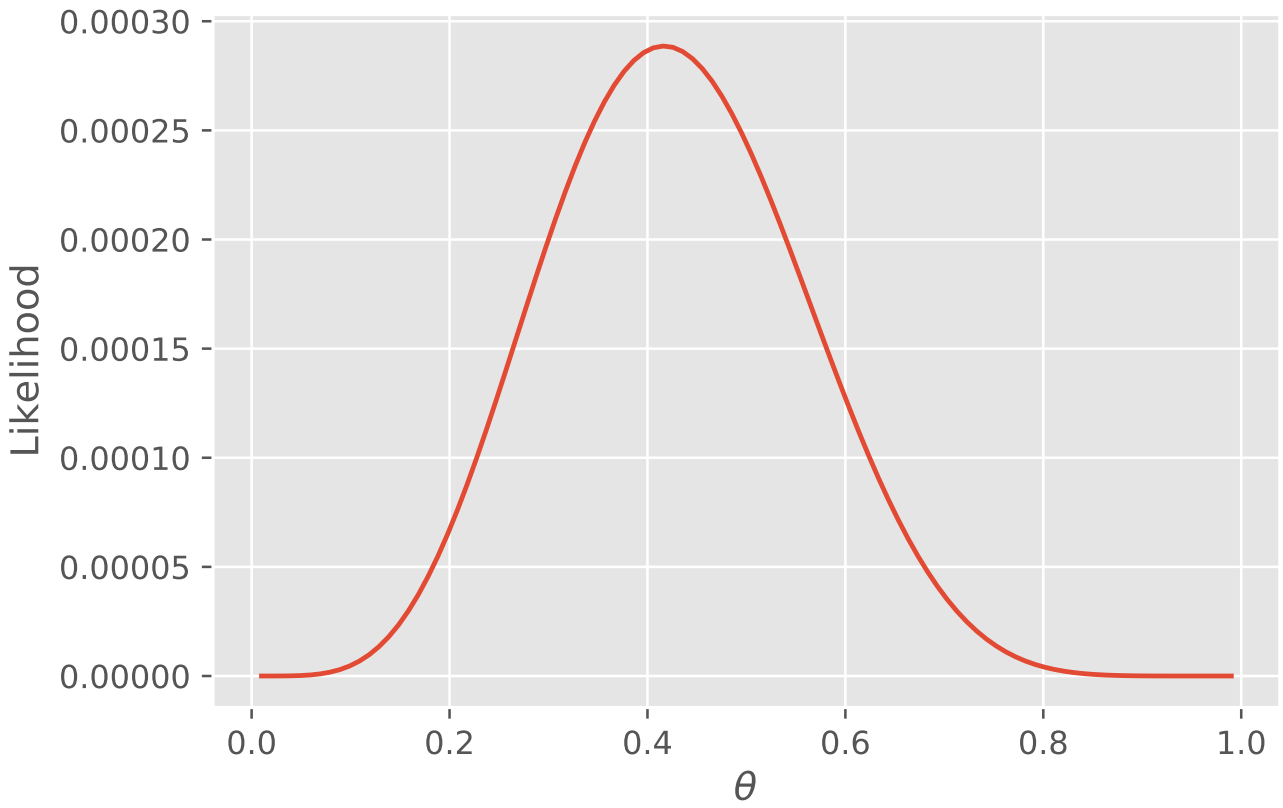

Figure: Likelihood function $L(\theta|D)$, $\theta \in [0, 1]$

Difference between the Probability and Likelihood

Probability: a number $p \in [0, 1]$ between 0 to 1 to describe how likely an event is to occur, or how likely it is that a proposition is true, assuming we know the distribution of the data.

Likelihood: a function that measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters. It is a function of the unknown parameters, e.g., θ.

Solving the urn problem: Maximum likelihood Estimator

Question: Given the above plot, what is this value θ for the urn?
Answer This is value of θ at the peak of the likelihood graph above, which is $\theta = 5/12$. Mathematically, this is defined as
Maximum likelihood estimation: Informally, based solely on the data, Maximum likelihood estimation searches the best parameters of a probability distribution that makes the data most likely.

The Maximum Likelihood Estimator (MLE)

Formally, Let $\boldsymbol{X} = (X_1, X_2, X_3, ..., X_n)$ be a random sample from a distribution with a parameter θ. Suppose we have observed the values of $\boldsymbol{X}$ as $\boldsymbol{x} = (x_1, x_2, x_3, ..., x_n)$, a maximum likelihood estimate of θ, denoted as $\hat{\theta}_{MLE}$ is a value of θ that maximises the likelihood function, i.e.,

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\, L(\theta|D) \tag{5}$$

Maximum Likelihood Estimator (MLE)

A maximum likelihood estimator (MLE) of the parameter θ, denoted as $\hat{\Theta}_{MLE}$ is a random variable $\hat{\Theta}_{MLE} = \hat{\Theta}_{MLE}(\boldsymbol{X})$ whose value when $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ is given by $\hat{\theta}_{MLE}$.

Solving the urn problem: Maximum likelihood

Question: Given the optimisation problem as formulated in equation 5, how to solve it?

Option 1: Exhaustive search – only works for low dimensional problems such as this.

- Grid search: usually used for tuning hyper-parameters of a machine learning model. See Hyperparameter optimization - Grid Search

Option 2: Optimization algorithms – a more general way to solve the problem.

Cost functions

Cost function: A function that maps a set of events into a number that represents the "cost" of that event occurring. Also known as the **loss function** or **objective function**.

Cost function for likelihood: a general one-to-one mapping with likelihood – the negative logarithm of the likelihood function:

$$J(\theta, D) = -\log(L(\theta|D))$$

Question: Why use the negative logarithm of the likelihood function as the cost function, i.e., $-\log(L(\theta|D))$

Why should we use the cost function?

Question: Why use the negative logarithm of the likelihood function as the cost function, i.e., $-\log(L(\theta|D))$

Answers:

- **Convention**: By convention, many optimisation problems are minimisation problems
- **Convenience**: Taking the logarithm changes multiplication to addition, i.e., $\log(AB) = \log(A) + \log(B)$, which is easier to differentiate
- **Numerically stable**: Product of θ, which is a probability will converge quickly to zero, which might cause problems for computers who are limited by machine precision.

Cost function for likelihood

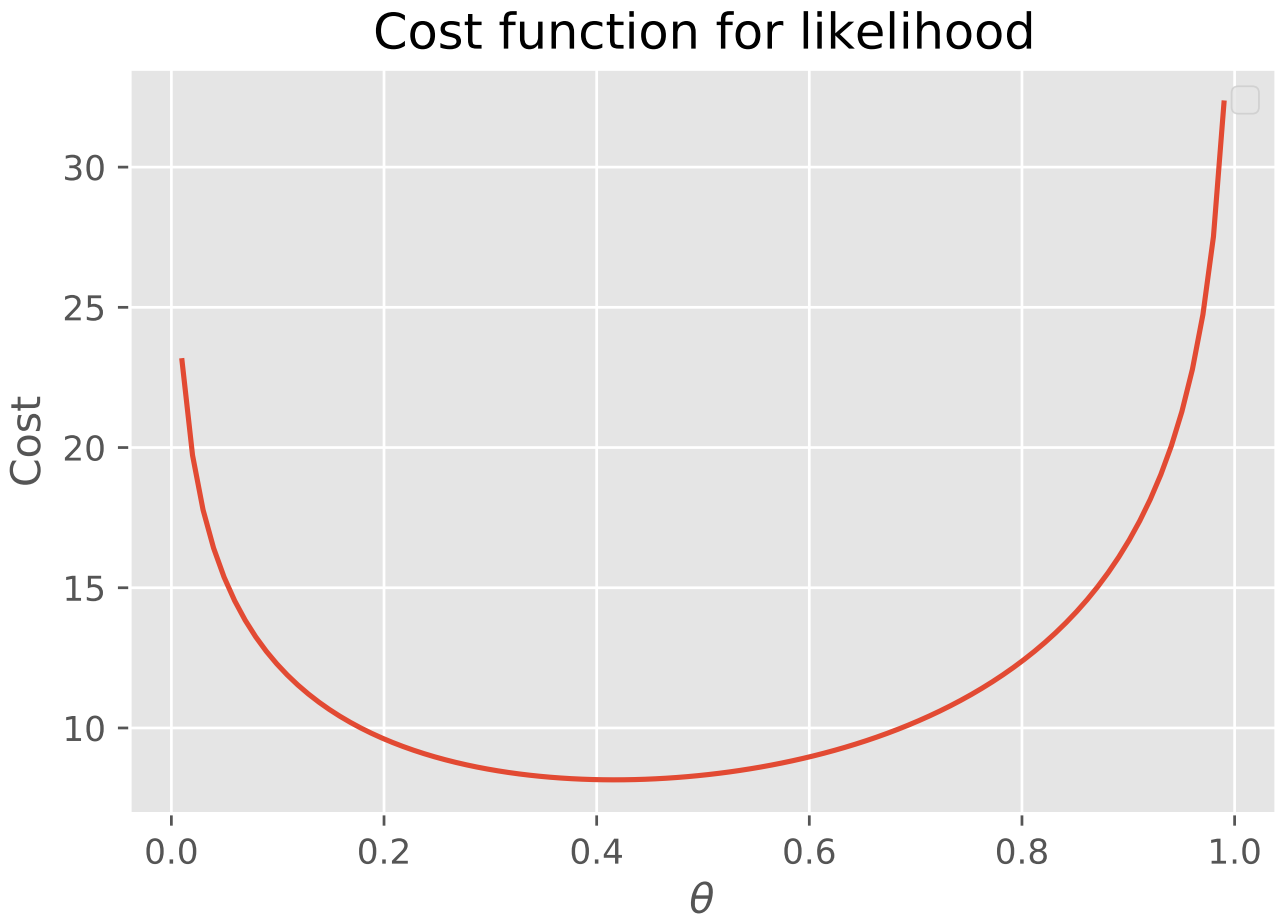

Figure: Cost function for likelihood

Introduction to optimisation

Optimization: finding the best solution from among the set of all feasible solutions.

Optimisation procedure:

1. Constructing a Model ✓
2. Determining the Problem Type
3. Selecting an optimisation algorithm ✓

Optimization Taxonomy

Introduction to optimisation

Machine learning $\Longrightarrow$ optimisation problems
Some examples:

- **Supervised learning**: Given some training data, we want to train a machine learning model to explain the data. The training process is essentially a process of finding a optimal set of parameters of this model and the optimality is defined by an objective function
- **Unsupervised learning**: Given some unlabelled samples, we aim to divide them into multiple clusters or groups, of which the samples in same group are as similar as possible but samples in different group are as different as possible.

The First-Order Optimality Condition

Optimisation: For a function $g(\boldsymbol{w})$ of N dimensional independent variables $\boldsymbol{w} \in \mathbb{R}^N$, the optimisation problem is

$$\underset{\boldsymbol{w}}{\operatorname{argmin}}\, g(\boldsymbol{w})$$

A $\boldsymbol{w}^*$ is the local minimum if it satisfies
First-order necessary condition for optimality:

$$\nabla_{\boldsymbol{w}} g(\boldsymbol{w}^*) = 0_{N\times 1} \tag{6}$$

The point which satisfy the condition is also called **stationary point**.
Note: A stationary point can be minimum maximum and saddle points

Saddle Point

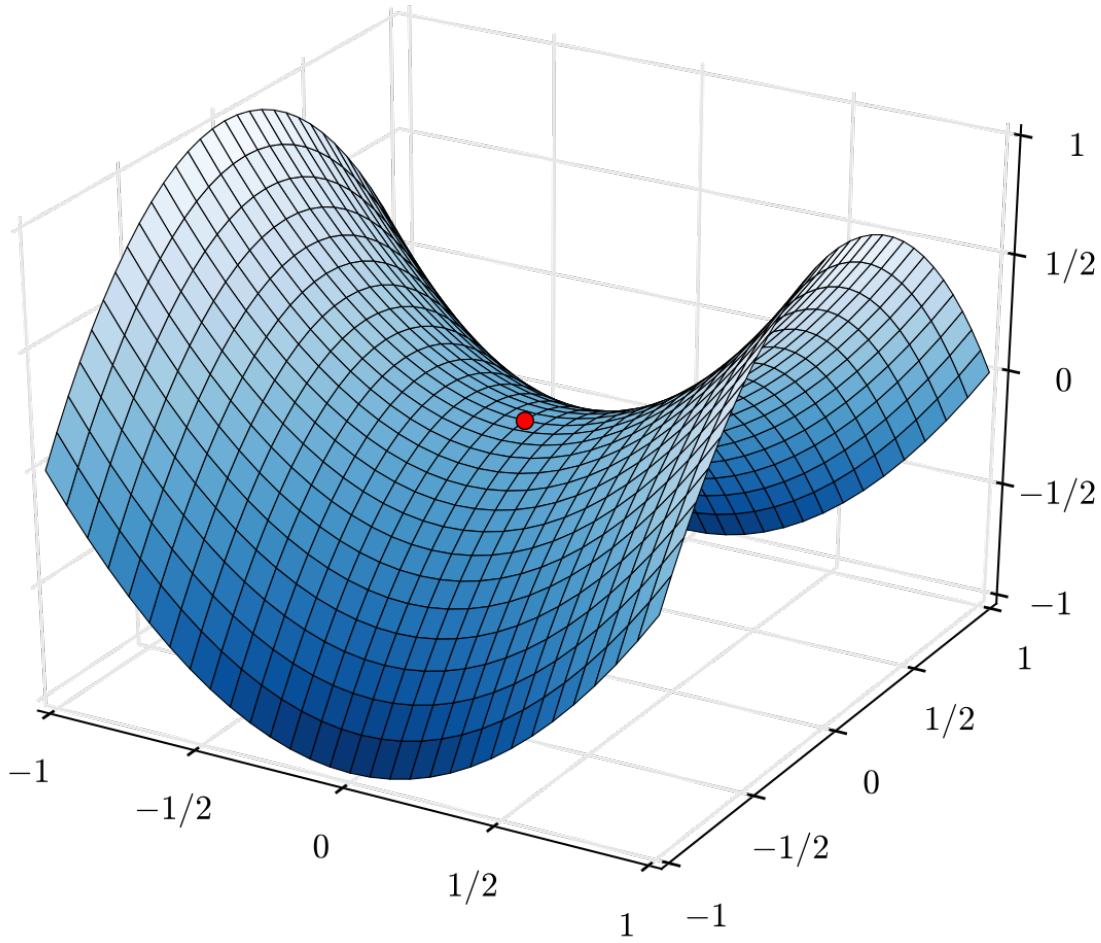

Figure: Saddle point By Nicoguaro - Own work, CC BY 3.0

The First-Order Optimality Condition

The equation of first-order necessary condition (equation 6) can be written as a system of N first order equations:

$$\frac{\partial}{\partial w_1} g(\boldsymbol{w}^*) = 0$$
$$\frac{\partial}{\partial w_2} g(\boldsymbol{w}^*) = 0$$
$$\vdots$$
$$\frac{\partial}{\partial w_N} g(\boldsymbol{w}^*) = 0$$

Question: can we solve this system of N first-order equations to identify the minimum points of the function g?

Gradient descent

Gradient descent: a first-order iterative optimization algorithm for finding a local minimum of a differentiable cost function.

Idea: to employ the **negative gradient** at each step to decreases the cost function.

Two ingredients: the negative gradient consists

- a direction – determined by the gradient at the point
- a magnitude, sometimes called step size

Intuition: start at any value of parameter θ, then change θ in the direction that decreases the cost function, and keep repeating until there is only the tiniest decrease in cost with each step.

Gradient descent

Formally, we define the negative gradient of a cost function $J(\theta)$ as

$$-\nabla_\theta J = -\frac{dJ(\theta)}{d\theta},$$

then we need to choose a magnitude or step size parameter η (also called the learning rate) so that the update equation becomes:

$$\theta(t+1) = \theta(t) - \eta\nabla_\theta J(\theta(t)) = \theta(t) - \eta\frac{dJ(\theta(t))}{d\theta},$$

Question: we know that $J(\theta) = -\log(\theta^5(1-\theta)^7)$, apply the rules we learned to calculate the derivative.

Gradient descent

Pseudo-code of gradient descent:

Initialisation: Start at any value of parameter θ

repeat

change the parameter θ in the direction that decreases the cost function $J(\theta)$

until the decrease in cost with each step is very small

end